www.asix.com.pl

Dokumentacja użytkownika dla Asix.Evo

Dok. Nr PLP9E019
Wersja: 2018-10-15

ASKOM® i **asix®** to zastrzeżone znaki firmy **ASKOM Sp. z o. o., Gliwice**. Inne występujące w tekście znaki firmowe bądź towarowe są zastrzeżonymi znakami ich właścicieli.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną lub inną powoduje naruszenie praw autorskich niniejszej publikacji.

ASKOM Sp. z o. o. nie bierze żadnej odpowiedzialności za jakiekolwiek szkody wynikłe z wykorzystywania zawartych w publikacji treści.

Copyright © 2018, ASKOM Sp. z o. o., Gliwice

ASKOM Sp. z o. o., ul. Józefa Sowińskiego 13, 44-121 Gliwice,
tel. +48 32 3018100, fax +48 32 3018101,

http://www.askom.pl, e-mail: biuro@askom.pl

Spis treści

Dokumentacja użytkownika dla Asix.Evo 1

Asix.Evo - Pierwsze kroki 44

1 Informacje wstępne 45

1.1 Wymagania systemowe 45

1.2 Instalacja 45

2 Pierwsza aplikacja 46

2.1 Utworzenie nowej aplikacji 46

2.2 Edycja diagramu zegara 47

2.3 Edycja okna startowego 51

2.4 Ustawienie parametrów startowych 53

2.5 Ustawienia zabezpieczeń 55

2.6 Utworzenie skrótu 56

2.7 Uruchomienie aplikacji 57

2.8 Uruchomienie aplikacji w trybie przeglądarki 58

3 Aplikacja z obsługą rzeczywistych danych 60

3.1 Utworzenie aplikacji przy pomocy kreatora 61

3.1.1 Lokalizacja aplikacji i wybór języków 61

3.1.2 Szablon aplikacji 62

3.1.3 Konfiguracja stanowisk 62

3.1.4 Określenie sposobu pozyskiwania danych procesowych 63

3.1.5 Deklaracja baz zmiennych procesowych 69

3.1.6 Konfiguracja źródeł danych 70

3.1.7 Parametryzacja systemu alarmów 71

3.1.8 Inne ustawienia 72

3.2 Przegląd konfiguracji 73

3.3 Test komunikacji - panel podglądu zmiennych 75

3.4 Budowa okien i diagramów aplikacji 76

3.4.1 Utworzenie okna 76

3.4.2 Utworzenie diagramów 77

3.4.3 Sygnalizacja przekroczeń 82

3.4.4 Dodanie parametrów diagramu 85

3.4.5 Wprowadzenie wzorców 87

3.5 Dodanie stanowiska terminalowego 89

4 Aplikacja z obsługą alarmów 94
4.1 Parametryzacja systemu alarmów 94
Asix.Evo - Elementy wizualizacji 99
1 Hierarchia elementów wizualizacyjnych 100
2 Właściwości i zdarzenia elementów wizualizacyjnych 101
2.1 Prefiksy definicji właściwości 104
3 Okna i panele 108
3.1 Właściwości okien 108
3.2 Właściwości paneli 112
4 Diagramy 113
4.1 Właściwości diagramów 115
4.2 Rejestry 116
4.3 Zdarzenia diagramów 116
4.4 Właściwości grup 116
4.5 Zdarzenia grup 117
4.6 Właściwości osadzonych wzorców 117
4.7 Zdarzenia osadzonych wzorców 118
5 Menu 119
5.1 Parametry i kontrola elementów menu 120
5.2 Właściwości menu 121
5.3 Właściwości elementów menu 122
6 Obiekty 122
6.1 Grupy właściwości 124
6.1.1 Właściwości podstawowe 124
6.1.2 Grupy wielokrotne 125
6.1.3 Właściwości stanowe 126
6.2 Standardowe właściwości obiektów 129
6.3 Standardowe zdarzenia obiektów 133
7 Wzorce 136
8 Właściwości globalne 138
8.1 Zmiana wartości właściwości w trybie wykonywania aplikacji 139
Asix.Evo - Techniki budowy diagramów 140
1 Metody tworzenia uniwersalnych obiektów 141
1.1 Sufiksowa notacja nazw zmiennych 141

1.2 Użycie atrybutów zmiennych ... 142
1.3 Właściwości globalne ... 142
1.4 Wzorce i parametryzowane diagramy ... 143
2 Wyświetlenie wartości zmiennej procesowej w postaci tekstowej ... 143
2.1 Wyświetlenie sformatowanej wartości ... 144
2.2 Uzależnienie sposobu wyświetlenia tekstu od wartości ... 146
2.2.1 Użycie wyrażeń warunkowych ... 146
2.2.2 Użycie metody wielostanowej ... 147
2.3 Zastosowanie limitów ostrzegawczych wartości zmiennej ... 148
2.3.1 Zastosowanie notacji sufiksowej ... 148
2.3.2 Zastosowanie atrybutów zmiennej ... 149
2.4 Obsługa statusu monitorowanej zmiennej ... 152
3 Formatowanie wartości liczbowych ... 153
3.1 Konwersja wartości numerycznych ... 154
3.2. Konwersja wartości typu DateTime ... 155
3.3. Konwersja wartości typu TimeSpan ... 156
4 Animacja wyglądu obiektów ... 158
4.1 Animowane obrazki ... 158
4.2 Animacja ruchu rurociągów, taśmociągów i linii ... 159
4.2.1 Rurociąg ... 159
4.2.2 Taśmociąg ... 160
4.2.3 Linia ... 161
4.3 Migotanie poprzez zmianę właściwości obiektów ... 161
5 Zasady otwierania i zamykania okien synoptycznych ... 163
5.1 Otwieranie okien synoptycznych ... 163
5.1.1 Otwarcie okna w trakcie startu aplikacji ... 163
5.1.2 Otwarcie okna i diagramu akcją OpenWindow ... 164
5.1.3 Otwarcie diagramu bez użycia predefiniowanego okna synoptycznego ... 165
5.2 Kontrola położenia i rozmiaru okien ... 166
5.3 Zamykanie okien i diagramów ... 167
5.4 Kontrola wzajemnego przesłaniania okien synoptycznych ... 168
6 Organizacja operacji sterujących ... 168
6.1 Sterowania natychmiastowe ... 168
6.2 Sterowania opóźnione (z potwierdzeniem) ... 170

6.3 Sterowanie przy pomocy akcji operatorskiej SetVariable 171
6.4 Kontrola ważności sterowań 172
6.5 Kontrola uprawnień 172
6.5.1 Podwójne potwierdzanie operacji sterujących 173
7 Parametryzacja funkcji interaktywnych w obiektach pasywnych 175
7.1 Nawigacja poprzez tekstowe łącza 175
7.2 Podłączenie menu kontekstowego do obiektu 176
7.3 Strefy aktywności diagramu 176
7.4 Operacje samopowtarzalne 178
7.5 Obsługa klawiatury 179
8 Wykorzystanie efektu przezroczystości 180
8.1 Przezroczystość okien 180
8.2 Przezroczystość obiektów 180
8.3 Ukrywanie obiektów 181
8.4 Wykorzystanie koloru Transparent 182
9 Sterowanie zachowaniem obiektów 182
9.1 Użycie zmiennych wirtualnych 182
9.2 Modyfikacja właściwości obiektów 184
10 Zastosowania obiektów klasy Przycisk 185
10.1 Przycisk jednopozycyjny 185
10.1.1 Przycisk jednopozycyjny wykonujący akcję operatorską 186
10.1.2 Przycisk jednopozycyjny wykonujący operację sterującą 187
10.2 Przycisk jednopozycyjny z powtarzaniem 188
10.3 Przycisk jednopozycyjny z blokadą 189
10.4 Przycisk dwupozycyjny 190
10.5 Przycisk dwupozycyjny z opóźnionym sterowaniem 191
10.6 Wyłącznik 192
10.7 Sterowania bitowe 193
10.8 Grupowanie przycisków 195
11 Operacje sterujące w obiektach klasy Tekst 197
11.1 Sterowanie wartością liczbową wprowadzaną przez użytkownika 197
11.2 Sterowania z listy wyboru 198
12 Użycie suwaków w obiektach klasy Słupek 201
12.1 Użycie suwaka do sterowania wartościami nastaw 201

12.2 Użycie suwaka do sterowania wartością zadaną z podglądem nowej nastawy ... 202
13 Animacja ruchu i zmiana rozmiaru obiektów ... 204
13.1 Zmiana pozycji ... 205
13.2 Zmiana pozycji wewnątrz obszaru określonego innym obiektem ... 206
13.3 Pozycjonowanie grup i wzorców ... 206
14 Sygnalizacja i obsługa stanów alarmowych ... 207
14.1 Kontrola stanu i obsługa pojedynczych alarmów ... 207
14.2 Sygnalizacja stanu grupy alarmów ... 209
15 Kontrola pracy obiektów klasy Wykres ... 211
15.1 Sterowanie zakresem czasu wykresu ... 211
15.1.1 Modyfikacja właściwości obiektu Wykres ... 211
15.1.2 Sterowanie poprzez zmienną wirtualną ... 213
15.2 Kontrola wyświetlania przebiegów wzorcowych ... 214
16 Użycie wzorców ... 216
Asix.Evo - Parametryzacja aplikacji ... 219
1 Konfiguracja stanowisk aplikacji ... 220
1.1 Panel roboczy Stanowiska ... 220
1.1.1 Założenie nowego stanowiska ... 221
1.2 Ustawianie parametrów zależnych od stanowiska ... 223
1.3 Konfiguracja parametrów komunikacyjnych stanowisk ... 224
1.3.1 Aliasy serwerów ... 225
1.3.2 Stanowiska obce ... 225
1.4 Uruchomienie aplikacji w kontekście wybranego stanowiska ... 225
1.5 Diagnostyka połączeń ... 226
2 Konfigurowanie dostępu do danych bieżących i archiwalnych ... 226
2.1Konfiguracja serwera danych ... 226
2.2 Bazy definicji zmiennych procesowych ... 228
2.2.1 Utworzenie nowej bazy ... 228
2.2.2 Importowanie utworzonej wcześniej bazy ... 229
2.2.3 Edycja i przeglądanie bazy definicji zmiennych ... 230
2.2.4 Atrybuty zmiennych ... 233
2.2.5 Ustawianie statusu aktywności bazy ... 234
2.3 Kanały komunikacyjne ... 235
2.3.1 Drajwery komunikacyjne ... 236

2.3.1.1 Drajwer NONE 237
2.3.1.2 Drajwer Network 237
2.3.1.3 Drajwer Asix6 238
2.3.1.4 Drajwer Random 238
2.3.2 Rodzaje kanałów wirtualnych 238
2.4 Deklaracja archiwów wartości zmiennych procesowych 239
2.4.1 Rodzaje archiwów 240
2.4.1.1 Archiwum Pamięciowe 240
2.4.1.2 Archiwum Sieciowe 240
2.4.1.3 Archiwum Asix6 241
2.4.1.4 Archiwum Losowe Dane 241
3 Konfiguracja Terminarza 241
3.1Inne metody automatycznego wykonywania akcji operatorskich 245
4 Parametryzacja aplikacji wielojęzycznych 245
4.1Deklaracja języków aplikacji 246
4.2 Zmiana języka pracy 246
4.3Teksty programu 247
4.4 Teksty aplikacji 248
4.4.1 Napisy ogólnego przeznaczenia 249
5 System uprawnień 250
5.1Definiowanie użytkowników 250
5.2 Definiowanie ról 251
5.2.1 Uprawnienie „Prawo zarządzania elementami aplikacji w trybie pracy” 252
5.2.2 Uprawnienie „Prawo wysyłania sterowań” 252
5.2.3 Uprawnienie „Prawo potwierdzania alarmów” 253
5.2.4 Uprawnienie „Brak blokowania pulpitu” 253
5.3 Uprawnienia 253
5.3.1 Prawo zarządzania użytkownikami 253
5.3.2 Prawo zarządzania elementami aplikacji w trybie pracy 254
5.3.3 Prawo wysyłania sterowań 254
5.3.4 Prawo potwierdzania alarmów 254
5.3.5 Brak blokowania pulpitu 254
5.3.6 Brak automatycznego wylogowania 254
5.4 Ochrona użycia plików 255

5.5 Ochrona zapisu sterowań do zmiennych procesowych ... 256
5.6 Tryby pracy systemu zabezpieczeń ... 257
5.6.1 Tryb standardowy ... 257
5.6.2 Tryb centralny ... 257
5.7 Ustawienia specyficzne dla stanowiska ... 258
5.8 Integracja z domeną Windows ... 260
5.9 Użycie systemu zabezpieczeń w aplikacji ... 261
5.9.1 Logowanie użytkownika ... 261
5.9.2 Weryfikacja uprawnień ... 262
5.9.3 Zmiany w parametrach uprawnień w trybie wykonania aplikacji ... 262
5.9.4 Przeglądanie logu zdarzeń logowania ... 263
6 Kontrola pracy aplikacji (AsAudit) ... 264
6.1Rejestracja operacji sterujących ... 264
6.2 Rejestracja działań operatora ... 265
6.3 Kontrola integralności aplikacji ... 266
7 Notatki operatora ... 268
8 Współpraca z programem AsTrend ... 269
8.1Praca w przeglądarkowej wersji aplikacji ... 270
9 Współpraca z programem AsBase ... 270
9.1 Akcje operatorskie ... 271
9.2 Obiekt Tabela Asbase ... 273
9.3 Tryby pracy sprzęgu z programem AsBase ... 273
9.4 Filtr połączenia ... 275
9.4.1 Wstawki ... 276
9.5 AsBase w aplikacjach przeglądarkowych ... 277
9.6 Uprawnienia użytkowników ... 278
10 Współpraca z programem AsService ... 278
10.1 Wizualizacja stanu liczników ... 278
11 Synchronizacja plików aplikacji ... 279
11.1 Synchronizacja katalogu definicyjnego aplikacji ... 280
11.2 Synchronizacja katalogu roboczego ... 281
12 Użycie przebiegów wzorcowych ... 281
12.1 Edycja przebiegów ... 281
12.2 Wyświetlanie krzywych wzorcowych ... 283

12.3 Wybór krzywej wzorcowej ... 283
12.4 Transfer przebiegów ... 283
13 Klawiatura ekranowa ... 283
13.1 Obiekt Klawiatura ... 284
13.2 Klawiatura okienkowa ... 284
14 Strażnik Mocy ... 285
14.1 Konfiguracja bazy danych SQL ... 285
14.2 Tworzenie mierników prostych ... 285
14.3 Zarządzanie miernikami i limitami ... 286
14.4 Obiekt wizualizacyjny - wykres miernika strażnika mocy ... 288
14.5 Funkcje wyrażeń strażnika mocy ... 288
14.6 Integracja z systemem alarmów ... 289
15 Aplikacja Asix jako usługa systemu Windows ... 289
Asix.Evo - Obiekty ... 294
Obiekt Dziennik zdarzeń ... 295
Obiekt Klapa ... 297
Obiekt Klawiatura ... 301
Obiekt Komunikaty ... 304
Obiekt Kontener wzorców ... 307
Obiekt Kontroler Obrazu Wektorowego ... 312
Obiekt Kontroler Wykresu ... 315
Obiekt Kształt ... 317
Obiekt Linia ... 321
Obiekt Lista rozwijana ... 324
Obiekt Miernik ... 329
Obiekt Obraz ... 336
Obiekt Pole autoryzacji ... 341
Obiekt Przeglądarka ... 343
Obiekt Przycisk ... 345
Obiekt Punkt pracy ... 353
Obiekt Rurociąg ... 357
Obiekt Słupek ... 361
Obiekt Tabela alarmów aktywnych ... 369
Obiekt Tabela alarmów historycznych ... 377

Obiekt Tabela alarmów SQL 385
Obiekt Tabela Asbase 393
Obiekt Tabela Danych 401
Obiekt Tabela Zmiennych 408
Obiekt Taśmociąg 414
Obiekt Tekst 419
Obiekt Wielokąt 426
Obiekt Wykres 429
Obiekt Wykres Danych 442
Obiekt Wykres strażnika mocy 452
Obiekt Zbiornik 455
Asix.Evo - Obiekty zewnętrzne 459
1 Dostarczanie własnych obiektów do aplikacji Asix.Evo 460
2 Tworzenie kontrolek obiektów 460
2.1 Środowisko programistyczne 460
2.2 Klasa bazowa kontrolki 461
2.3 Tworzenie klasy kontrolki 464
2.4 Definiowanie własnych właściwości obiektu 464
2.5 Dostęp do standardowych właściwości obiektu 465
2.6 Definiowanie własnych zdarzeń obiektu 466
2.7 Używanie przeźroczystości w obiektach kontrolkowych 466
2.8 Pobieranie stanu i wartości zmiennych 467
2.9 Pobieranie wartości atrybutów zmiennych 467
2.10 Sterowanie zmiennych 468
2.11 Rejestrowanie i wyrejestrowywanie zdarzeń zmiany stanu zmiennych 469
2.12 Dostęp do wartości archiwalnych zmiennych 470
2.13 Dostęp do tekstów wielojęzycznych aplikacji 471
2.14 Dostęp do obrazków aplikacji 471
2.15 Wykonywanie akcji operatorskich 472
2.16 Dostęp do właściwości globalnych 472
2.17 Obsługa zdarzeń w trybie wektorowym 472
2.18 Przykład 1 – Definiowanie właściwości i zdarzeń 473
2.19 Przykład 2 – Praca ze zmiennymi 481
2.0 Przykład 3 – Obiekt wektorowy 484

Asix.Evo – Diagramy mapowe GIS 489
1 Wstęp 490
1.1 Diagramy GIS 490
1.2 Podkłady mapowe 490
1.2.1 Warstwy 491
1.3 Elementy wizualizacyjne 492
1.4 Współpraca z geograficznymi bazami MS SQL lub plikami Shapefile 492
2 Praca z diagramami mapowymi GIS 493
2.1 Konfiguracja źródła mapy 493
2.2 Tworzenie diagramu 493
2.3 Definiowanie właściwości podstawowych diagramu 495
2.4 Tworzenie i parametryzacja warstwy 496
2.5 Tworzenie i parametryzacja przycisków 497
2.6 Zarządzanie widocznością warstw / pasek narzędziowy 499
2.7 Tworzenie i parametryzacja obiektu Linia 500
2.8 Tworzenie i parametryzacja obiektu Region 502
2.9 Tworzenie i parametryzacja obiektu Znacznik 504
2.10 Tworzenie i parametryzacja obiektu Tekst 506
2.11 Tworzenie i parametryzacja obiektu Obraz 507
2.12 Tworzenie wzorców / Znaczników GIS 509
2.13 Przygotowanie źródła danych GIS 512
2.14 Deklarowanie map offline 515
Asix.Evo - Akcje operatorskie 517
1 Użycie akcji operatorskich 518
2 Akcje złożone 519
2.1 Parametry akcji złożonych 520
3 Lista akcji operatorskich 521
4 Opisy akcji operatorskich 526
AcceptAlarm 526
AcceptAlarmsOnTable 528
Actions 528
ActionSet 529
AddNote 530
AsbaseAdd 530

AsbaseClose 531
AsbaseDelete 532
AsbaseLoad 532
AsbaseLoadByValue 533
AsbaseNavigate 535
AsbaseOpen 536
AsbasePrint 538
AsbaseSave 539
AsbaseShow 540
AsbaseUpdate 541
AsbaseWait 542
Ask 542
AsreportDisplay 543
AstrendAsix6 544
AstrendDisplay 546
AstrendPrint 548
AstrendPrintToPdf 549
Break 551
CancelControls 551
ChangePassword 552
ClearSelection 553
CloseWindow 553
ConfirmRole 555
CopyWindow 556
EditPatterns 557
EditTimetables 557
EndAlarm 558
ExcludeAlarm 559
Execute 559
GetVarNames 560
GisGoTo 561
GisHideLayer 562
GisShowLayer 562
GisSwitchLayer 563

GisRefreshLayer 563
GisZoom 564
GisZoomIn 564
GisZoomOut 565
HidePanel 566
IncludeAlarm 566
Login 567
Logout 567
Message 568
MinimizeWindow 569
NextDiagram 570
Nothing 570
ObjectAction 571
OpenDiagram 574
OpenWindow 577
Perform 581
PlaySound 582
PlaySoundLooping 583
PowerGuardManager 583
PowerGuardMeter 584
PreviousDiagram 584
PrintScreen 585
RefreshVariable 586
ReportGenerate 587
ReportsManager 588
ResetGlobalPropery 588
Restart 589
Run 589
Script 590
ScriptClose 591
ScriptRestart 591
SecurityBrowser 592
SecurityManager 593
SelectPattern 594

SendControls 594
SetAppData 595
SetBounds 596
SetGlobalProperty 597
SetParameter 598
SetParameters 598
SetPosition 599
SetProperty 600
SetSelection 601
SetSize 602
SetTableVariable 603
SetVariable 603
SetWindowSize 605
ShowControlPanel 606
ShowKeyboard 606
ShowMenu 607
ShowNumericKeyboard 607
Speak 608
ShowPanel 608
StartAlarm 609
StopSound 609
SwitchLanguage 610
SwitchPanel 611
SwitchWorkMode 611
SydelShow 612
Terminate 613
ToggleBits 613
Transfer 614
VariableInfo 615
Zoom 616
ZoomIn 617
ZoomOut 617
Asix.Evo - Wyrażenia i funkcje 619
1 Użycie wyrażeń i funkcji 620

2 Składnia wyrażeń 620
2.1 Elementy wyrażeń 620
2.1.1 Liczby 620
2.1.2 Teksty 620
2.1.3 Stałe 621
2.1.4 Identyfikatory 621
2.1.5 Wywołania funkcji 621
2.1.6 Komentarze 622
2.2 Operatory 622
~ ! - (unarny) 622
* / % 622
+ - 623
<< >> 623
< <= > >= 623
== != 623
& 623
^ 624
| 624
&& 624
|| 624
?: 625
3 Wyliczanie wyrażeń 625
3.1 Typy i konwersja wartości 625
3.2 Kontekst użycia 626
3.3 Diagnostyka wyrażeń 627
4 Spis i opis funkcji 627
Abs 635
Acos 635
AbsToRelX 636
AbsToRelY 636
AlarmsControler 637
AlarmsCount 637
AlarmsGroupState 639
AlarmsMode 640

Add 641
And 641
Angle 642
AppData 642
ApplicationChanged 643
ArchiveData 639
AsbaseText 645
Asin 646
Asix6Attribute 646
Asix6Limit 647
AsserviceRead 648
Atan 651
Atan2 652
Attribute 652
AttributeOrDefault 653
Ceiling 654
ChannelState 654
CheckOPCStatus 655
CheckStateMask 656
CheckStateTemplate 657
Color 658
Concat 659
ContainsString 660
Cos 661
CurrentGisZoom 661
CursorX 662
CursorY 662
DeadBand 662
DelayedState 663
Div 664
E 665
Evaluate 665
EventContext 666
ExistsAttribute 667

ExistsVariable 667
Floor 668
Format 669
FromAsix6Date 670
FromAsix6Time 670
GetDictionary 671
GetDictionaryValue 671
GetMessage 672
GetStateDescription 673
GetStateText 674
HasPrivilege 676
HasRole 676
HasWaitingControl 677
HorizonAggregate 679
IsActive 680
IsActiveNote 680
IsAlarm 681
IsAlarmExcluded 681
IsAlarmUnaccepted 682
IsAltPressed 683
IsBlinkOff 683
IsBrowser 684
IsLiteBrowserTerminal 685
IsControlPressed 685
IsDiagramActive 685
IsDiagramActivePar 686
IsDiagramOpened 687
IsGisLayerVisible 688
IsMouseOver 689
IsMousePressed 690
IsRunning 690
IsSelected 691
IsShiftPressed 691
IsScriptWorking 691

IsWindowOpened 692
IsWorking 693
LastKeyPressed 693
Log 694
LocalProperty 694
LocalPropertyStatus 695
Log10 696
Max 697
Min 698
NewValue 698
OPCTime 699
Or 700
Parameter 700
Parameters 701
PI 702
Pow 702
PowerGuardRead 703
Property 703
RangeAggregate 704
RateOfChange 705
RelToAbsY 706
RawValue 707
Reg 708
RegStatus 708
RelToAbsX 709
Replace 710
RoleDescription 710
Round 711
Sequence 712
Sin 713
Slide 713
SplitString 714
Sqrt 715
StationDiagInfo 715

StationInfo 717
Substring 717
TableElement 718
TableLength 719
Tan 720
Text 721
TimetableRead 721
ToAsix6Date 722
ToAsix6Time 722
ToBool 723
ToDateTime 724
ToDouble 725
ToFullPath 726
ToInt8 726
ToInt16 727
ToInt32 727
ToInt64 728
ToLocalTime 729
ToLower 729
ToSingle 730
ToString 730
ToTable 731
ToTimeSpan 731
ToUInt8 732
ToUInt16 732
ToUInt64 733
ToUInt32 734
ToUpper 734
ToUTCTime 735
Truncate 736
ValueOrDefault 736
VarChanged 737
Variable 738
VariableOrDefault 739

VarIsGood 739
VarIsNotGood 740
VarName
..... 741
VarNames 741
VarStatus 742
VarStringValue 743
VarTime 743
VarTimeUtc 744
VarValueChangeTime 745
XOnScreen 745
XorColor 746
YOnScreen 747
Asix.Evo - System alarmów 749
1 Podstawowe pojęcia 750
2 Parametryzacja systemu alarmów 751
3 Role stanowisk 752
4 Baza definicji alarmów 754
4.1 Atrybuty 754
4.2 Teksty opisu alarmów 757
4.3 Grupy alarmów 758
I.4 Import z arkuszy Excel 759
5 Strategie wykrywania alarmów 761
5.1 Strategia bitowa 762
5.2 Strategia warunkowa 763
5.3 Strategia OPC 764
5.4 Strategia buforowa 765
5.5 Strategia Asix 766
5.6 Strategia AsService 767
5.7 Strategia Strażnik Mocy 768
5.8 Strategia Wago 768
5.9 Strategia zewnętrzna 769
5.10 Globalna strategia zewnętrzna 769

6 Obsługa alarmów zgłaszanych przez drajwery komunikacyjne typu Asix6 ... 770
7 Log alarmów aktywnych ... 770
8 Log alarmów historycznych ... 772
8.1 Archiwum alarmów SQL ... 773
9 Wizualizacja stanu alarmów ... 774
9.1 Tabele alarmów aktywnych ... 775
9.2 Tabele alarmów historycznych ... 775
9.3 Tabele alarmów SQL ... 775
9.4 Inne obiekty ... 776
10 Obsługa alarmów ... 777
10.1 Potwierdzanie alarmów ... 777
10.2 Wykluczenia alarmów ... 777
10.3 Wykluczenia jednorazowe ... 778
10.4 Filtrowanie alarmów ... 779
10.5 Sygnalizacja dźwiękowa ... 780
10.6 Drukowanie ... 781
11 Współpraca z AsAlert ... 782
Asix.Evo - Harmonogramy ... 784
1 Cechy modułu harmonogramów ... 785
1.1 Podstawowe pojęcia ... 785
1.2 Praca w trybie sieciowym ... 786
2 Parametryzacja harmonogramów ... 786
2.1 Tryb pracy modułu ... 786
2.2 Definiowanie harmonogramu ... 787
2.3 Predefiniowane kategorie dni ... 788
2.4 Uruchomienie okna edycji cykli harmonogramów ... 788
3 Obsługa harmonogramów ... 789
3.1 Edycja cykli harmonogramów ... 789
3.2 Edycja kategorii dni ... 794
Asix.Evo - Raporty skryptowe ... 798
1 System raportów skryptowych ... 799
2 Konfiguracja raportów skryptowych ... 799
3 Obsługa raportów skryptowych ... 801
4 Konstrukcja skryptów raportowych ... 802

4.1 Rodzaje raportów ... 802
4.2 Szkielet kodu generacji raportu ... 803
4.3 Tabela pionowa dla danych archiwalnych ... 807
4.4 Tabela pozioma dla danych archiwalnych ... 810
4.5 Kontrola zawartości komórek ... 813
4.6 Kolumny i wiersze wyliczane ... 817
4.7 Raport na podstawie danych przekazanych w kolekcji ICollection ... 820
4.8 Raport na podstawie danych przekazywanych w obiekcie typu DataTable ... 824
4.9 Użycie wykresu w treści raportu ... 826
4.10 Metody pozycjonowania tekstów ... 830
5 Model obiektowy skryptów raportowych ... 832
5.1 Interfejs IApplication ... 832
5.1.1 Metody ... 832
CreateReport ... 832
5.2 Klasa Report ... 832
5.2.1 Właściwości ... 832
Footer ... 832
Header ... 833
MigraDocument ... 833
MigraSection ... 833
5.2.2 Metody ... 833
AddChart ... 833
AddHorizontalTable ... 835
AddImage ... 836
AddPageBreak ... 836
AddParagraph ... 837
AddVerticalTable ... 837
Render ... 838
SetPageSetup ... 838
SetStyle ... 839
5.3 Klasa ReportHeader ... 840
5.3.1 Właściwości ... 840
MigraHeader ... 840
5.4 Klasa ReportFooter ... 840

5.4.1 Właściwości 841
MigraFooter 841
5.5 Klasa ReportHeaderFooter 841
5.5.1 Metody 841
AddImage 841
5.6 Klasa ReportParagraph 842
5.6.1 Właściwości 842
MigraParagraph 842
5.6.2 Metody 843
AddNumPagesField 843
AddPageField 843
AddText 844
5.7 Klasa ReportImage 844
5.7.1 Właściwości 844
MigraImage 844
5.8 Klasa ReportStyleStandardNames 845
5.8.1 Właściwości 845
Footer 845
Header 845
Normal 845
TableAltRow 846
TableErrorCell 846
TableHeader 846
TableRow 847
5.9 Klasa ReportStyle 847
5.9.1 Właściwości 847
Alignment 847
FontBold 848
FontColor 848
FontItalic 848
FontName 848
FontSize 849
MigraStyle 849
Name 849

ShaderColor 849
5.9.2 Metody 850
AddTabStops 850
AddTabStop 850
5.10 Klasa ReportTable 851
5.10.1 Właściwości 851
EndTimeColumn 851
FirstStyle 851
MigraTable 852
NextStyle 852
StartTimeColumn 852
5.10.2 Metody 853
AddCalculatedColumn 853
AddCalculatedRow 854
AddColumn 855
AddRow 857
GenerateContent 858
LeftIndent 859
SetEdgeStyle 859
SetQueryData 859
5.11 Klasa ReportTableColumn 860
5.11.1 Właściwości 860
AggrName 860
Callback 860
ColumnAltStyle 861
ColumnStyle 861
ErrorMode 861
ErrorStyle 862
Format 862
HeaderStyle 862
HeaderText 862
ItemName 863
MigraColumn 863
ReportTable 863

Width ... 863
5.12 Klasa ReportTableRow ... 864
5.12.1 Właściwości ... 864
AggName ... 864
Callback ... 864
ErrorMode ... 864
ErrorStyle ... 865
FirstStyle ... 865
Format ... 865
HeaderText ... 865
ItemName ... 866
MigraRow ... 866
NextStyle ... 866
ReportTable ... 866
5.13 Klasa ReportTableCalcColumn ... 867
5.13.1 Właściwości ... 867
CalcType ... 867
Callback ... 867
ColumnStyle ... 868
ColumnAltStyle ... 868
ErrorMode ... 868
ErrorStyle ... 868
Format ... 869
HeaderText ... 869
HeaderStyle ... 869
MigraColumn ... 869
ReportTable ... 870
Width ... 870
5.14 Klasa ReportTableCalcRow ... 870
5.14.1 Właściwości ... 870
CalcType ... 870
Callback ... 871
ErrorMode ... 871
ErrorStyle ... 871

FirstStyle 872
Format 872
HeaderText 872
MigraRow 872
NextStyle 873
ReportTable 873
5.15 Klasa ReportTableCell 873
5.15.1 Właściwości 873
CalcColumn 873
CalcRow 874
Column 874
Content 874
Index 875
MigraCell 875
Row 875
RowIdx 875
RawValue 876
SourceValues 876
Style 876
ValueStatus 876
5.15.2 Metody 877
AvgValue 877
MaxValue 877
MinValue 877
SumValue 877
5.16 Klasa CellValue 878
5.16.1 Właściwości 878
Content 878
Status 878
TimeStamp 878
Value 879
5.17 Klasa ReportChart 879
5.17.1 Właściwości 879
MigraChart 879

MigraLegend ... 879
5.17.2 Metody ... 880
AddLegend ... 880
AddSeries ... 880
AddXSeries ... 881
CustomizeXAxis ... 882
CustomizeYAxis ... 882
GenerateContent ... 883
SetQueryData ... 883
SetXAxisTitle ... 883
SetYAxisTitle ... 884
5.18 Klasa ReportChartSeries ... 884
5.18.1 Właściwości ... 884
AggrName ... 884
ItemName ... 885
MigraSeries ... 885
5.18.2 Metody ... 885
SetDataLabels ... 885
SetMarkersStyle ... 886
Asix.Evo - Skrypty ... 887
1 Podstawy użycia skryptów ... 888
1.1 Utworzenie skryptu ... 888
1.2 Edycja skryptów ... 889
1.3 Metody uruchamiania skryptów ... 890
1.3.1 Skrypty interakcyjne ... 890
1.3.2 Skrypty autonomiczne ... 891
1.4 Zarządzanie aktywnymi skryptami ... 892
1.5 Struktura kodu skryptu ... 893
2 Przykłady typowych skryptów ... 894
2.1 Skrypty jednorazowe ... 894
2.2 Skrypty rezydentne ... 896
2.2.1 Pętle w metodzie Initialize ... 896
2.2.2 Wykorzystanie funkcji RegisterTimeEvent ... 898
2.2.3 Tworzenie wątków ... 899

2.2.4 Monitorowanie zmian zmiennej procesowej 900
2.2.5 Monitorowanie stanu alarmów 901
2.3 Odczyt parametrów uruchomienia skryptu 901
2.4 Odczyt wartości bieżących zmiennych procesowych 903
2.4.1 Cykliczny odczyt wartości bieżących zmiennych procesowych 908
2.4.2 Odczyt wartości bieżących sterowany odświeżaniem zmiennej 910
2.5 Ustawianie wartości zmiennych procesowych 910
2.5.1 Wyliczanie wartości pochodnych zmiennych 910
2.5.2 Pozyskiwanie wartości ze źródeł zewnętrznych 912
2.6 Odczyt wartości archiwalnych zmiennych procesowych 914
2.7 Użycie zmiennych tablicowych 918
2.7.1 Odczyt zmiennych tablicowych 918
2.7.2 Zapis zmiennych tablicowych 919
2.8 Transfer przebiegów wzorcowych 921
2.9 Kontrola stanu alarmów 923
2.9.1 Odczyt stanu alarmów aktywnych 923
2.9.2 Aktywne monitorowanie stanu alarmów 925
2.9.3 Zgłaszanie alarmów 925
2.10 Użycie kontekstu uruchomienia skryptu 928
2.11 Skrypty wielojęzyczne 930
2.12 Tworzenie interfejsu okienkowego 932
2.12.1 Wyświetlanie komunikatów 932
2.12.2 Okna budowane w treści skryptu 933
2.12.3 Użycie dołączanych bibliotek 937
3 Model obiektowy skryptów 939
3.1 Interfejs IApplication 939
3.1.1 Właściwości 939
ApplicationAddonsDirectory 939
ApplicationDirectory 939
ApplicationName 940
ApplicationWorkingDirectory 940
Asalert 940
Asbase 941
Aslogger 941

Asservice ... 941
AsixDataServerStarted ... 942
ComputerName ... 942
ComputerAddress ... 942
CurrentApplicationLanguage ... 943
CurrentStationName ... 943
CurrentUserId ... 943
CurrentUserName ... 943
CurrentUserRoles ... 944
ExecutionContext ... 944
IsAsixUser ... 944
IsDomainUser ... 945
ProgramVersion ... 945
RDSComputerName ... 945
RDSComputerAddress ... 946
ScriptName ... 946
ScriptSettings ... 946
SystemUser ... 946
3.1.2 Metody ... 947
ApplicationText ... 947
CreateThread ... 947
ExecuteAction ... 948
GetAlarm ... 949
GetAlarmDomainController ... 949
GetAlarmDomains ... 950
GetAlarmDomainStationMode ... 950
GetAlarmDomainStatistics ... 950
GetActiveAlarms ... 951
GetAppData ... 951
GetChannelState ... 951
GetDefinedAlarms ... 952
GetPowerGuardMeter ... 952
GetTrendPatternPoints ... 953
GetVariable ... 953

GetVariableArcData 954
GetVariableAttribute 955
GetVariableNames 956
GetVariableValue 956
GlobalProperty 957
HasRole 957
LoginUser 957
OPCTime 958
ReadVariables 959
RegisterAppDataEvent 959
RegisterAlarmDomainEvents 960
RegisterTimeEvent 962
RemoveAppData 963
ReportMessage 963
RestartScript 964
SetAppData 964
SetVariableBits 965
SetVariables 966
SetVariablesSync 967
SetVariableValue 969
Sleep 969
TerminateScript 970
ToggleBits 970
UnregisterEvent 971
3.2 Interfejs IContext 972
3.2.1 Właściwości 972
Diagram 972
DiagramObject 972
IsWebBrowser 972
MainVariable 973
ScriptStartTime 973
SourceFilePath 973
WebServerAddress 974
3.2.2 Metody 974

IsDiagramActive ... 974
IsDiagramOpened ... 974
IsRuntime ... 975
IsScriptWorking ... 975
IsWindowOpened ... 975
RestartApplication ... 976
3.3 Interfejs IScriptSettings ... 976
3.3.1 Właściwości ... 976
AllowMultipleInstances ... 976
RestartIfFailed ... 976
3.4 Interfejs IVariable ... 977
3.4.1 Właściwości ... 977
Name ... 977
IsRefreshed ... 977
VariableState ... 977
3.4.2 Metody ... 978
OrderArcData ... 978
Read ... 979
RegisterStateEvent ... 980
SetVariableBits ... 981
SetVariableBitsSync ... 982
SetVariableValue ... 983
SetVariableValueSync ... 984
ToggleBits ... 985
ToggleBitsSync ... 986
UnregisterStateEvent ... 986
VariableAttribute ... 987
3.5 Klasa VariableState ... 987
3.5.1 Właściwości ... 987
AsDouble ... 987
AsInteger ... 987
FormattedStatus ... 988
FormattedValue ... 988
IsBad ... 988

IsGood ... 989
IsUncertain ... 989
RawValue ... 989
Status ... 990
TimeStamp ... 990
TimeStampUtc ... 990
Value ... 991
3.6 Klasa ArcQuery ... 991
3.6.1 Konstruktor ... 991
3.6.2 Właściwości ... 992
Aggregate ... 992
AggregateInterval ... 992
EndTime ... 992
ProcessedDataQuery ... 993
StartTime ... 993
3.7 Interfejs IArcQueryResult ... 993
3.7.1 Właściwości ... 993
ErrorDescription ... 993
Query ... 994
ResultStatus ... 994
Values ... 994
VariableName ... 995
3.7.2 Metody ... 995
CancelQuery ... 995
RegisterQueryResultEvent ... 995
UnregisterQueryResultEvent ... 997
3.8 Klasa ArcVariableState ... 998
3.8.1 Właściwości ... 998
AsDouble ... 998
AsInteger ... 999
EndTimeStamp ... 999
EndTimeStampUTC ... 999
FormattedStatus ... 999
IsBad ... 1000

IsUncertain 1000
Status 1001
TimeStamp 1001
TimeStampUTC 1002
Value 1002
3.9 Interfejs IVariableControlState 1002
3.9.1 Właściwości 1003
ErrorMsg 1003
ControlStatus 1003
VariableName 1003
3.10 Interfejs IVariableControlResult 1004
3.10.1 Właściwości 1004
ErrorMsg 1004
ControlStatus 1004
3.10.2 Metody 1005
Wait 1005
3.11 Interfejs IAlarm 1005
3.11 Interfejs IAlarm 1006
3.11.1 Właściwości 1006
AlarmState 1006
Definition 1006
Domain 1006
Id 1007
3.11.2 Metody 1007
Accept 1007
Cancel 1008
Raise 1008
RegisterStateEvent 1009
UnregisterStateEvent 1010
3.12 Klasa AlarmState 1011
3.12.1 Właściwości 1011
AcceptanceOperatorId 1011
AcceptanceStationName 1011
AcceptanceTimeStamp 1012

BeginDetectionTimeStamp 1012
BeginExprValue 1012
BeginTimeStamp 1013
EndDetectionTimeStamp 1013
EndExprValue 1013
EndTimeStamp 1013
FormattedBeginText 1014
FormattedEndText 1014
IsAccepted 1014
IsActive 1015
IsExcluded 1015
IsFinished 1015
WasFiltered 1016
3.13 Klasa AlarmProperties 1016
3.13.1 Właściwości 1016
Action 1016
DetectionParameters 1016
BeginText 1017
EndText 1017
Id 1017
IsStartOnly 1018
Priority 1018
Strategy 1018
3.13.2 Metody 1019
GetAlarmAttribute 1019
3.14 Klasa AlarmDomainStatistics 1019
3.14.1 Właściwości 1019
ActiveAlarmsCount 1019
DefinedAlarmsCount 1020
ExcludedAlarmsCount 1020
FilteredAlarmsCount 1020
HistoricalAlarmsCount 1020
HistoricalLogFileSize 1021
UnacceptedAlarmsCount 1021

3.15 Klasa TrendPatternPoint ... 1022
3.15.1 Właściwości ... 1022
TimeStamp ... 1022
Value ... 1022
3.16 Klasa AsalertAccess ... 1022
3.16.1 Metody ... 1023
Send ... 1023
SetAccessData ... 1023
3.17 Klasa AsbaseAccess ... 1024
3.17.1 Metody ... 1024
ExecuteQuery ... 1024
ReadArchive ... 1024
ReadLoads ... 1025
ReadRecipes ... 1026
3.18 Klasa AsloggerAccess ... 1027
3.18.1 Metody ... 1028
CreateSeries ... 1028
DeleteSeries ... 1028
ExecuteQuery ... 1029
ReadPlans ... 1029
ReadPoints ... 1030
ReadSeries ... 1030
ReadSeriesAggregates ... 1031
ReadSeriesData ... 1032
SetAccessData ... 1033
3.19 Klasa AsloggerSeries ... 1033
3.19.1 Właściwości ... 1033
SeriesLength ... 1033
WriteCount ... 1034
3.19.2 Metody ... 1034
AddPointsRecord ... 1034
AddPointValues ... 1034
Save ... 1035
3.20 Klasa AsserviceAccess ... 1035

3.20.1 Metody 1036
ReadArcCounters 1036
ReadCounters 1037
3.21 Klasa AsserviceAccess.Names 1038
3.22 Interfejs IPowerGuardMeter 1041
3.22.1 Właściwości 1041
Name 1041
MeterState 1041
3.22.2 Metody 1042
RegisterStateEvent 1042
UnregisterStateEvent 1042
3.23 Klasa PowerGuardMeterState 1043
3.23.1 Właściwości 1043
CurrentCycleStartTime 1043
CurrentCycleEndTime 1043
CurrentCycleLimit 1043
CurrentCycleAverageValue 1043
CurrentCycleSafeRemainingValue 1043
UnitText 1044
IsPredictionSafe 1044
Załącznik 1. Rodzaje agregatów danych archiwalnych 1044
Asix.Evo - Uruchomienie aplikacji WWW 1046
1. Wymagania 1047
2. Pierwsze uruchomienie 1047
3 Sprawdzanie używanych i dostępnych licencji WWW 1050
Asix.Evo - Jak to zrobić 1051
1 Uruchamianie programu i zmiana trybów pracy 1052
1.1 Wystartowanie programu w trybie architekta 1052
1.2 Uruchomienie aplikacji w trybie wykonawczym 1052
1.3 Przejście z trybu wykonawczego do trybu edycyjnego 1052
1.4 Edycja dużych okien i diagramów 1053
1.5 Testowanie działania diagramów w trakcie edycji 1053
1.6 Testowanie działania aplikacji w trakcie edycji 1053
2 Edycja właściwości 1054
2.1 Edycja interaktywna 1054

2.2 Edycja ręczna ... 1055
2.3 Użycie wartości wcześniej wprowadzonych ... 1055
2.4 Zmiana właściwości typu wyliczeniowego ... 1055
2.5 Wprowadzenie odwołania do zmiennej procesowej ... 1055
2.6 Wprowadzenie odwołania do atrybutu zmiennej ... 1055
2.7 Wprowadzenie odwołania do parametru diagramu lub wzorca ... 1055
2.8 Wprowadzenie odwołania do właściwości globalnej ... 1056
2.9 Wybór z okna wyboru elementu ... 1056
3 Edycja okien i diagramów ... 1056
3.1 Utworzenie okna lub diagramu ... 1056
3.2 Utworzenie grupy okien lub diagramów ... 1057
3.3 Zmiana lokalizacji okna, diagramu lub grupy okien lub diagramów ... 1057
3.4 Zmiana nazwy okna, diagramu lub grupy ... 1057
3.5 Utworzenie kopii okna lub diagramu ... 1057
3.6 Otwarcie okna lub diagramu do edycji ... 1057
3.7 Zmiana wyglądu okna ... 1057
3.8 Dodanie panelu do okna ... 1058
3.9 Zmiana położenia panelu ... 1058
3.10 Przyklejenie panelu do wybranej krawędzi okna ... 1058
3.11 Zmiana rozmiaru panelu ... 1058
3.12 Dołączenie menu do okna ... 1058
3.13 Utworzenie nowego obiektu ... 1058
3.14 Utworzenie nowego obiektu wierzchołkowego ... 1059
3.15 Wybranie elementu do edycji ... 1059
3.16 Wybranie wielu obiektów ... 1059
3.17 Wybranie wszystkich obiektów tego samego typu ... 1060
3.18 Wybranie wszystkich obiektów ... 1060
3.19 Utworzenie grupy obiektów ... 1060
3.20 Kopiowanie obiektów ... 1060
3.21 Zmiana pozycji obiektów ... 1061
3.22 Zmiana rozmiaru pojedynczego obiektu ... 1061
3.23 Zmiana rozmiaru grupy obiektów ... 1061
3.24 Edycja kształtu obiektów wierzchołkowych ... 1062
3.25 Transformacja kształtu obiektów wierzchołkowych ... 1062

3.26 Dopasowanie rozmiaru obiektu do rozmiaru obrazka 1062
3.27 Dopasowanie rozmiaru obiektu do rozmiaru diagramu 1062
3.28 Wycofanie zmian położenia i rozmiaru obiektów oraz odtworzenie obiektów usuniętych 1063
3.29 Tymczasowe ukrywanie obiektów w trybie edycji 1063
3.30 Blokowanie zmian rozmiaru i położenia obiektów 1063
3.31 Ustalenie kolejności przesłaniania się obiektów 1063
3.32 Wyrównywanie położenia obiektów 1064
3.33 Dosuwanie obiektów 1064
3.34 Rozsuwanie obiektów 1064
3.35 Nakładanie obiektów 1065
3.36 Utworzenie wzorca z obiektów diagramu 1065
3.37 Osadzenie wzorca na diagramie 1065
3.38 Odłączenie osadzonego wzorca od źródła 1065
3.39 Grupowa zamiana używanych zmiennych 1066
3.40 Podgląd wartości i symulacja zmiennych 1066
3.41 Dodanie parametrów do diagramu lub wzorca 1067
4 Inne operacje edycyjne 1068
4.1 Dodanie obrazka do aplikacji 1068
4.2 Edycja obrazka 1068
Znane problemy 1070
Rodzaje agregatów danych archiwalnych 1073
Serwer Danych 1076
1. Obsługa danych procesowych Serwera Danych 1077
1.1. Serwer Danych 1077
1.2. Asmen - menedżer komunikacyjny 1077
1.2.1. Deklarowanie kanałów transmisji 1079
1.2.2. Wyłączanie kanałów z obsługi 1081
1.2.3. Deklarowanie zmiennych procesowych 1082
1.2.4. Uruchamianie modułu Asmen 1084
1.2.5. Parametry konfiguracyjne modułu Asmen 1084
1.2.6. Funkcje przeliczające 1084
1.2.7. Diagnostyka poprawnego formatu liczb odczytywanych ze sterowników 1116
1.2.8. Drajwery komunikacyjne 1116

1.2.9. Wymiana danych poprzez lokalną sieć komputerową 1122
1.2.9.1. Deklaracja kanału sieciowego 1122
1.2.9.2 Deklaracja zmiennych procesowych w kanałach sieciowych 1124
1.2.9.3. Praca stacji roboczej w trybie POMOST 1124
1.2.9.4. Redundancja 1126
1.3. Aspad - program archiwizacji danych 1128
1.3.1. Ogólna charakterystyka programu Aspad 1128
1.3.1.1. Podstawowe cechy programu Aspad 1128
1.3.1.2. Parametry konfiguracyjne modułu Aspad 1129
1.3.1.3. Podstawowe rodzaje i typy archiwum Aspada 1129
1.3.1.3.1. Archiwum replikowane 1131
1.3.1.4. Cechy archiwizacji prowadzonej w plikach o dedykowanym formacie 1136
1.3.1.5. Cechy archiwizacji prowadzonej w relacyjnej bazie danych SQL 1137
1.3.1.6. Agregacja danych w systemie Asix 1137
1.3.2. Moduły związane z Aspadem 1137
1.3.3. Wymagania systemowe 1137
1.3.3.1. Pamięć operacyjna 1138
1.3.3.2. Pamięć dyskowa dla archiwum D, M, Y i H 1139
1.3.3.3. Pamięć dyskowa dla archiwum B 1139
1.3.4. Pliki tworzone przez programy modułu Aspad 1140
1.3.4.1. Pliki archiwalne D, M, Y i H 1140
1.3.4.2. Pliki archiwum B i P 1141
1.3.5. Konfigurowanie systemu przy instalacji modułu Aspad 1142
1.3.6. Konfigurowanie i uruchomienie modułu Aspad 1142
1.3.6.1. Specyfikacja odcinków czasu w plikach konfiguracyjnych 1142
1.3.6.2. Zadawanie podstawowych parametrów archiwizacji 1143
1.3.6.3. Automatyczne tworzenie kopii zapasowych 1143
1.3.6.4. Automatyczne kasowanie najstarszych plików 1143
1.3.6.5. Sieciowy dostęp do danych Aspada 1144
1.3.6.6. Parametry archiwizacji zmiennych 1144
1.3.6.7 Archiwizacja podstawowa i dodatkowa 1147
1.3.6.8. Atrybuty archiwizacji 1148
1.3.6.9. Deklaracja archiwizacji zmiennych - edytor parametrów 1149
1.3.7. Archiwizacja warunkowa 1150

1.3.7.1. Deklaracje warunków archiwizacji 1151
1.3.7.2. Operatory warunków archiwizacji 1153
1.3.7.3. Przykłady warunków 1155
1.3.7.4. Deklaracja in-line warunku archiwizacji 1156
1.3.8. Archiwizacja typu B 1157
1.3.8.1. Specyficzne cechy archiwizacji typu B 1157
1.3.8.2. Dodatkowe wymagania archiwum typu B 1157
1.3.8.3. Uzupełnianie danych w archiwum B ze sterownika 1158
1.3.8.4. Synchronizacja archiwum z bliźniaczą bazą danych 1159
1.3.8.5. Deklaracja bazy danych 1160
1.3.8.6. Deklaracja zamiennych do archiwizacji w bazie danych 1160
1.3.8.7. Dodatkowe informacje o archiwizowanych punktach 1161
1.3.8.8. Archiwizacja w wielu bazach danych 1162
1.3.8.9. Archiwizacja niesekwencyjna 1162
1.3.8.10. Przeglądanie zawartości bazy danych archiwum B 1162
1.3.9. Przebieg wzorcowy typu P 1163
1.3.9.1. Deklaracja przebiegu wzorcowego 1163
1.3.9.2. Generacja przebiegów wzorcowych 1164
1.3.10. Agregacja danych archiwalnych 1164
1.3.10.1. Wymagania agregacji 1164
1.3.10.2 Parametryzacja 1164
1.3.10.3 Zestaw agregatów 1165
1.3.11 Archiwum X - obsługa danych ze znacznikami czasu o dokładności poniżej sekundy ... 1167
1.4. Aslink - moduł sieciowy 1169
1.4.1. Ogólna charakterystyka modułu Aslink 1170
1.4.1.1. Usługi wiarygodnego przesyłu danych 1170
1.4.1.2. Usługa synchronizacji czasu 1170
1.4.1.3. Usługi zdalnego zapisu plików 1170
1.4.1.4. Konfiguracje wielokartowe 1171
1.4.2. Typy obsługiwanych rodzajów komunikacji 1171
1.4.3. Parametryzacja modułu ASLINK 1172
1.4.3.1. Identyfikacja stacji 1172
1.4.3.2. Dostęp do sieci przy komunikacji poprzez NetBIOS 1172
1.4.3.3. Synchronizacja czasu 1174

1.4.3.4. Zabezpieczenie hasłem 1175
1.4.3.5. Połączenia ze starszymi wersjami modułu sieciowego 1176
1.4.3.6. Diagnostyka 1176
1.5. Obsługa alarmów dla modułów AsTrend, AsAlarm i AsRaport 1176
2. Parametryzacja Serwera Danych 1177
2.1. Praca z programem Architekt 1177
2.1.1. Wstęp 1177
2.1.2. Środowisko pracy modułu Architekt 1177
2.1.2.1. Uruchomienie modułu Architekt 1177
2.1.2.2. Zamykanie modułu Architekt 1177
2.1.2.3. Okno główne modułu Architekt 1177
2.1.2.3.1 Drzewo modułów 1180
2.1.2.3.2. Blok parametrów konfiguracyjnych 1183
2.1.2.4. Indywidualne opcje pracy programu Architekt 1184
2.1.2.5. Menu główne, dostępne polecenia 1187
2.1.3. Wskazówki dotyczące obsługi opcji domyślnych w modułach konfiguracyjnych 1196
2.1.4. Zarządzanie opcjami za pomocą okna z zestawieniem wszystkich zadeklarowanych opcji 1196
2.1.5. Importowanie wybranych opcji z zewnętrznego pliku *.xml 1197
2.2. Obsługa baz definicji zmiennych aplikacji systemu Asix 1198
2.2.1 Wstęp 1198
2.2.1.1. Tryb tworzenia bazy danych 1200
2.2.1.2. Format bazy danych 1200
2.2.1.2.1. Łączone bazy definicji zmiennych 1201
2.2.1.3. Atrybuty zmiennych 1202
2.2.1.4. Nazewnictwo atrybutów i tabel 1203
2.2.2. Tworzenie bazy w trybie Edytora 1204
2.2.2.1. Edycja bazy definicji zmiennych – Edytor definicji 1207
2.2.2.1.1. Dodawanie, usuwanie i modyfikowanie zmiennych 1207
2.2.2.1.2. Dodawanie, usuwanie i natychmiastowa edycja zmiennych przy użyciu poleceń menu lokalnego / dodawanie zmiennych systemowych / wstawianie definicji zmiennej z kanału OPC 1208
2.2.2.1.3. Dodawanie i usuwanie atrybutów zmiennych 1210
2.2.2.1.4. Edycja wartości atrybutu dla kilku zmiennych jednocześnie 1211
2.2.2.1.5. Wybór widocznych atrybutów zmiennych 1213

2.2.2.1.6. Edytory atrybutów zmiennych 1215
2.2.2.2. Tworzenie schematu bazy definicji zmiennych – Edytor schematu 1224
2.2.2.2.1. Grupowanie zmiennych 1226
2.2.2.2.2. Zestawy atrybutów 1229
2.2.2.2.3. Parametry funkcji przeliczających 1230
2.2.2.2.4. Tłumaczenia nazw atrybutów zmiennych 1231
2.2.2.2.5. Atrybuty zastępcze 1232
2.2.2.2.6. Deklarowanie języków atrybutów 1234
2.2.2.3. Import modyfikacji z arkusza Excel 1234
2.2.2.4. Deklarowanie długości nazwy zmiennej 1236
2.2.3. Tworzenie bazy w trybie Generatora 1237
2.2.3.1. Opcje generowania bazy definicji zmiennych z wielu źródeł danych 1239
2.2.3.2. Generowanie bazy z plików ZEZ/DB 1240
2.2.3.3. Generowanie bazy definicji zmiennych z arkusza Excel 1240
2.2.3.4. Generowanie bazy definicji zmiennych na podstawie bazy OLE DB 1240
2.2.3.5. Generowanie bazy definicji zmiennych na podstawie plików XML 1241
2.2.3.6. Import definicji zmiennych z serwera OPC / OPC UA 1242
2.2.3.7. Zbiorcza baza definicji zmiennych 1243
2.2.3.8. Uaktualnianie źródeł danych 1243
2.2.3.9. Transformacja nazw i wartości atrybutów 1243
2.2.3.10. Nadpisywanie atrybutów przez wybraną tabelę źródła danych 1244
2.2.3.11. Deklarowanie maksymalnej długości nazwy zmiennej 1246
2.2.4. Przeglądanie zmiennych 1247
2.2.4.1. Wyszukiwanie wartości atrybutu i filtrowanie rekordów po wartości atrybutu 1249
2.2.5 Praca ze skoroszytem Excel jako źródłem danych o atrybutach zmiennych bez generowania bazy definicji zmiennych 1250
2.2.6. Przygotowanie zewnętrznego źródła danych – arkusz Excel 1250
2.2.6.1. Atrybuty zmiennych 1251
2.2.6.2. Grupowanie 1252
2.2.6.3. Zestawy atrybutów 1252
2.2.6.4. Parametry funkcji przeliczających 1253
2.2.6.5. Nagłówki 1254
2.2.6.6. Języki atrybutów 1255
2.2.6.7. Atrybuty zastępcze 1256
2.2.6.8. Parametry transformacji nazw atrybutów i wartości atrybutów 1257

2.2.6.9. Zmienne używane ... 1259
2.2.6.10. Szablon bazy definicji zmiennych ... 1260
2.2.6.11. Szablony wartości atrybutów ... 1260
2.2.6.12. Asix VarDef Excel Addin - edycja atrybutów bazy definicji zmiennych ... 1260
2.2.7. Przechowywanie opisów stanów zmiennych w bazie definicji zmiennych ... 1262
2.2.8. Przechowywanie opisów stanów bitów w bazie definicji zmiennych ... 1264
2.2.9. Konwertowanie bazy definicji zmiennych z formatu Paradox na Jet lub MS SQL ... 1266
2.2.10. Konwertowanie bazy definicji zmiennych z plików tekstowych ... 1272
2.2.10.1. Import definicji zmiennych z plików tekstowych - atrybuty grupujące ... 1273
2.2.11. Eksport bazy do arkusza Excel ... 1273
2.2.12. Dostęp z poziomu innych programów ... 1275
2.2.13. Uaktualnianie bazy definicji zmiennych on-line ... 1275
2.2.14. Zmienne systemowe ... 1276
2.2.15. Mirroring ... 1276
2.3. Dodanie definicji alarmów SQL do bazy definicji zmiennych ... 1277
2.4. Utworzenie pliku konfiguracyjnego Serwera Danych ... 1278
2.4.1. Kreator w trybie importu bazy z serwera OPC / OPC UA ... 1285
2.5. Zadeklarowanie pliku konfiguracyjnego Serwera Danych ... 1289
2.6. Import plików konfiguracyjnych *.INI ... 1290
2.7. Konfiguracja parametrów startowych ... 1292
2.7.1. Deklaracja programów startowanych jednocześnie z aplikacją ... 1292
2.7.2. Podstawowe parametry Panelu Kontrolnego i Konstruktora ... 1294
2.7.3. Definiowanie zależności pomiędzy komputerami i obszarami ... 1296
2.7.4. Deklaracja aktywnych zmiennych ... 1297
2.7.5. Deklaracja zewnętrznej bazy definicji zmiennych ... 1298
2.8. Konfiguracja parametrów danych bieżących ... 1300
2.8.1. Parametry standardowe ... 1300
2.8.2. Parametry zaawansowane ... 1302
2.8.3. Parametry diagnostyczne ... 1305
2.8.4. Definicja kanałów komunikacyjnych ... 1309
2.8.5. Zarządzanie kanałami ... 1315
2.9. Konfiguracja parametrów danych bieżących dotyczących serwera sieci ... 1316
2.9.1. Dane standardowe ... 1316
2.9.2. Dane zaawansowane ... 1318

2.9.3. Parametry diagnostyczne 1321
2.10. Konfiguracja danych archiwalnych 1322
2.10.1. Dane standardowe 1322
2.10.2. Deklaracja czasu oczekiwania na zgłoszenie serwerów 1324
2.10.3. Parametryzacja okresu uzupełniania danych 1326
2.10.4. Deklaracja wersji plików archiwum standardowego 1327
2.10.5. Parametry diagnostyczne 1328
2.10.6. Śledzenie próbkowanych wartości 1330
2.10.7. Debugowanie zapytań 1332
2.10.8. Parametryzacja cyklicznego pobierania danych historycznych dla wybranych drajwerów komunikacyjnych 1333
2.10.9. Parametryzacja archiwum 1334
2.10.9.1. Parametry standardowe 1338
2.10.9.2. Parametry archiwizacji agregatów 1348
2.10.9.3. Zarządzanie uzupełnianiem danych 1349
2.10.9.4. Zarządzanie odczytem danych 1351
2.10.9.5. Parametry określające sposób tworzenia kopii zapasowej archiwum 1352
2.10.9.6. Deklaracja synchronizacji zawartości archiwum typu B z danymi w bazie MS SQL 1354
2.10.9.7. Deklaracja wartości o określonym statusie 1358
2.10.10. Zarządzanie archiwami 1359
2.11. Konfiguracja modułu sieci 1360
2.11.1. Nazwa komputera 1362
2.11.2. Komunikacja poprzez TCP/IP 1363
2.11.3. Komunikacja poprzez NetBIOS 1364
2.11.4. Parametry zaawansowane 1375
2.11.5. Parametry diagnostyczne 1377
2.12. Grupa Pozostałe 1379
2.12.1. Deklaracja parametrów wprowadzanych bezpośrednio 1379
Załącznik 1 1381

Asix.Evo - Pierwsze kroki

1 Informacje wstępne

Dodatek Asix.Evo jest modułem rozszerzającym możliwości pakietu Asix. Nowa funkcjonalność wprowadzona w Asix.Evo to przede wszystkim:

a) Całkowicie nowy system wizualizacji stanu procesu wraz z nowym środowiskiem budowy aplikacji.

b) Nowa technologia aplikacji internetowych, która pozwala na uzyskanie identycznej jak aplikacje okienkowe funkcjonalności. Nie jest wymaga konwersja aplikacji na format HTML.

c) Zmodernizowany system obsługi alarmów.

Aplikacje wykonane w technologii Asix.Evo wykorzystują elementy standardowego pakietu Asix w zakresie odczytu danych bieżących, zapisu i odczytu archiwów danych procesowych oraz komunikacji ze stanowiskami wykonującymi standardowe aplikacje Asix. Aplikacje Asix.Evo mogą również współpracować z programami Astrend i Asbase.

W dalszej części dokumentacji określeniem **Asix** będą oznaczane wszystkie informacje odnoszące się bezpośrednio do elementów wchodzących w skład podstawowego pakietu Asix.

1.1 Wymagania systemowe

Asix.Evo wymaga do działania systemów operacyjnych Windows XP SP2 lub nowszych, lub systemów serwerowych Windows Server 2003 lub nowszych. Dodatkowo wymagana jest instalacji platformy .NET 3.51.

W przypadku aplikacji internetowych wymagana jest instalacja modułu IIS w wersji 5 lub nowszej dla stanowiska serwera aplikacji. Stanowiska klienckie muszą wykorzystywać przeglądarkę Internet Explorer 5 lub nowszą.

1.2 Instalacja

Dodatek Asix.Evo instalowany jest w ramach standardowego instalatora pakietu Asix. Z reguły należy instalować pełny pakiet - łącznie ze wszystkimi elementami Asixa klasycznego.

Można zainstalować tylko elementy Asix.Evo wtedy, gdy:

a) stanowisko pełni wyłącznie rolę serwera internetowego;

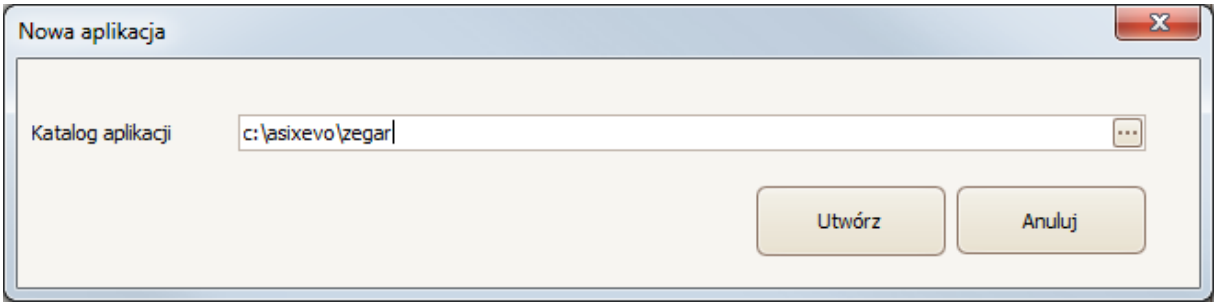
b) stanowisko jest typu terminalowego pobierającego dane wyłącznie sieciowo ze stanowisk serwerowych Asix.Evo oraz w aplikacji nie są używane programy AsTrend i AsBase.

2 Pierwsza aplikacja

W rozdziale tym zostanie przedstawiony kompletny proces utworzenia i uruchomienia bardzo prostej aplikacji Asix.Evo. Zadaniem tej aplikacji będzie wyświetlenie pojedynczego okna, w którym będzie wyświetlana aktualna godzina.

2.1 Utworzenie nowej aplikacji

Po uruchomieniu programu AsixEvo wykonujemy komendę *Nowa aplikacja…* z menu *Plik*. Zostanie wyświetlone poniższe okno dialogowe, w którym należy podać nazwę katalogu, w którym będą przechowywane pliki konfiguracyjne.



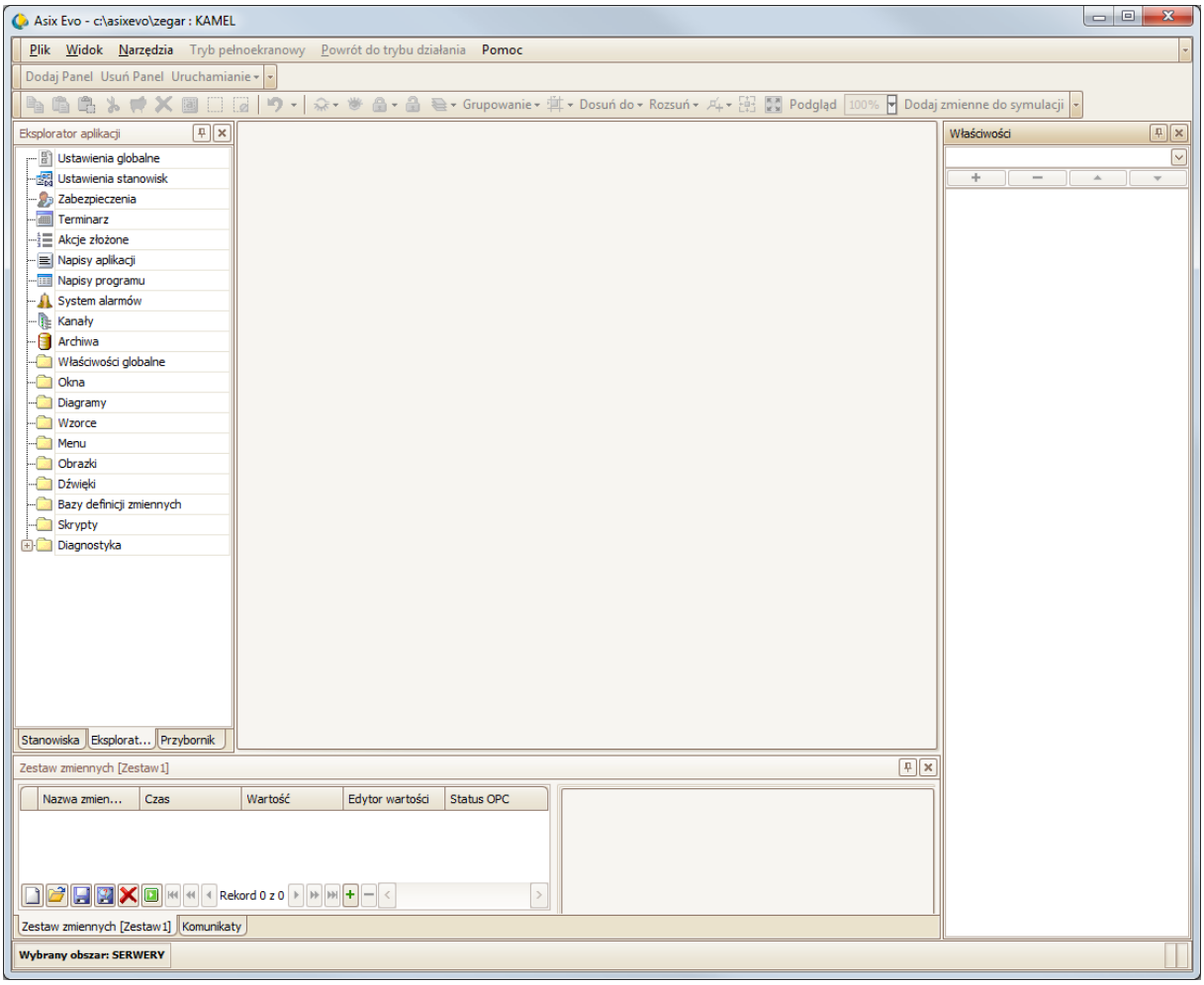
Rys. Okno deklaracji katalogu aplikacji.

W przypadku aplikacji Asix.Evo praktycznie wszystkie używane pliki znajdują się w dwóch katalogach. Pliki konfiguracyjne znajdują się w katalogu (i jego podkatalogach) podanym w powyższym oknie. Pliki robocze (np. archiwa, pliki logów) są zapisywane w katalogu roboczym, którego nazwa jest tworzona automatycznie. W tym przypadku byłaby to nazwa c:\asixevo\zegar_work. Lokalizację katalogu roboczego można zmienić w *Ustawieniach stanowisk.*

Po utworzeniu aplikacji w panelu *Eksplorator aplikacji* (menu *Widok* > *Eksplorator aplikacji*) pokazane jest drzewo elementów składowych aplikacji. Poprzez ten panel można uzyskać dostęp do paneli edycyjnych elementów, zazwyczaj poprzez podwójne kliknięcie na elemencie drzewa eksploratora.

UWAGA:

Wygląd okna roboczego może się różnić od pokazanego na przykładowych ilustracjach. Wygląd okna, położenie i rodzaj paneli, paski narzędziowe mogą być parametryzowane przy pomocy komend z menu *Widok* oraz metodami przeciągnij i upuść.

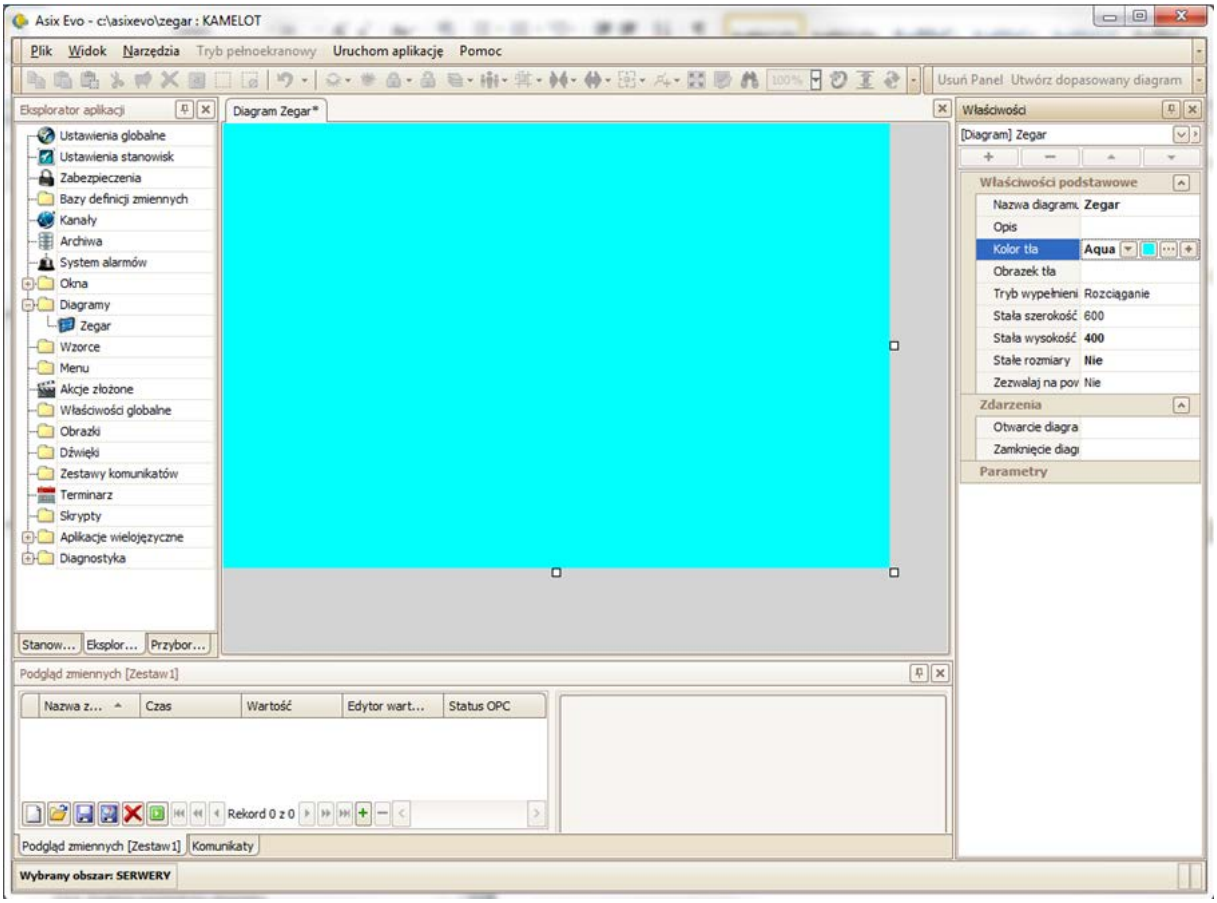


Rys. Okno główne programu Asix.Evo.exe.

2.2 Edycja diagramu zegara

Pierwszym krokiem w budowie naszej aplikacji będzie utworzenie diagramu, w którym następnie umieścimy obiekt pokazujący aktualny czas. W tym celu po kliknięciu prawym przyciskiem na elemencie *Diagram* eksploratora aplikacji z menu kontekstowego wybieramy komendę *Utwórz diagram*. Poniższa ilustracja pokazuje okno programu AsixEvo, w którym w centralnym panelu wyświetlany jest diagram w trybie edycji. W panelu właściwości pokazywane są wartości wszystkich właściwości opisujących parametry diagramu. Panel *Właściwości* jest podstawowym elementem używanym w trakcie edycji diagramów i okien. W przedstawionym przypadku została już zmieniona nazwa diagramu na *Zegar*, kolor tła diagramu, oraz właściwość *Stałe rozmiary*. Dostępne są dwa układy panelu właściwości: klasyczny i kompaktowy. Tryb pracy można zmienić korzystając z przycisku  w górnym prawym narożniku panelu. W pierwszym przykładzie będziemy korzystać z układu klasycznego, w którym wszystkie właściwości pokazywane są na jednej liście.

Na poniższej ilustracji została już zmieniona nazwa diagramu na Zegar, kolor tła diagramu, oraz właściwość *Stałe rozmiary*.



Rys. Okno edycji diagramu.

Kolor tła został zmieniony przy pomocy specjalizowanego okna wyboru koloru, otwieranego przy pomocy wskazanego poniżej przycisku w polu edycyjnym właściwości *Kolor tła*.



Rys. Opcja wyboru koloru tła diagramu.

Zestaw dostępnych przycisków może się różnić w zależności od typu właściwości.

Ustawienie właściwości *Stałe rozmiary* na *Nie* powoduje, że w trakcie wykonywania aplikacji rozmiar diagramu będzie dopasowywał się automatycznie do rozmiaru okna, w którym będzie wyświetlany. Wartość właściwości typu Tak/Nie zmienia się najszybciej poprzez podwójne kliknięcie w obszarze wartości.

Kolejnym krokiem będzie umieszczenie na diagramie obiektu typu *Tekst*, który będzie wyświetlał aktualną datę i godzinę. W tym celu należy:

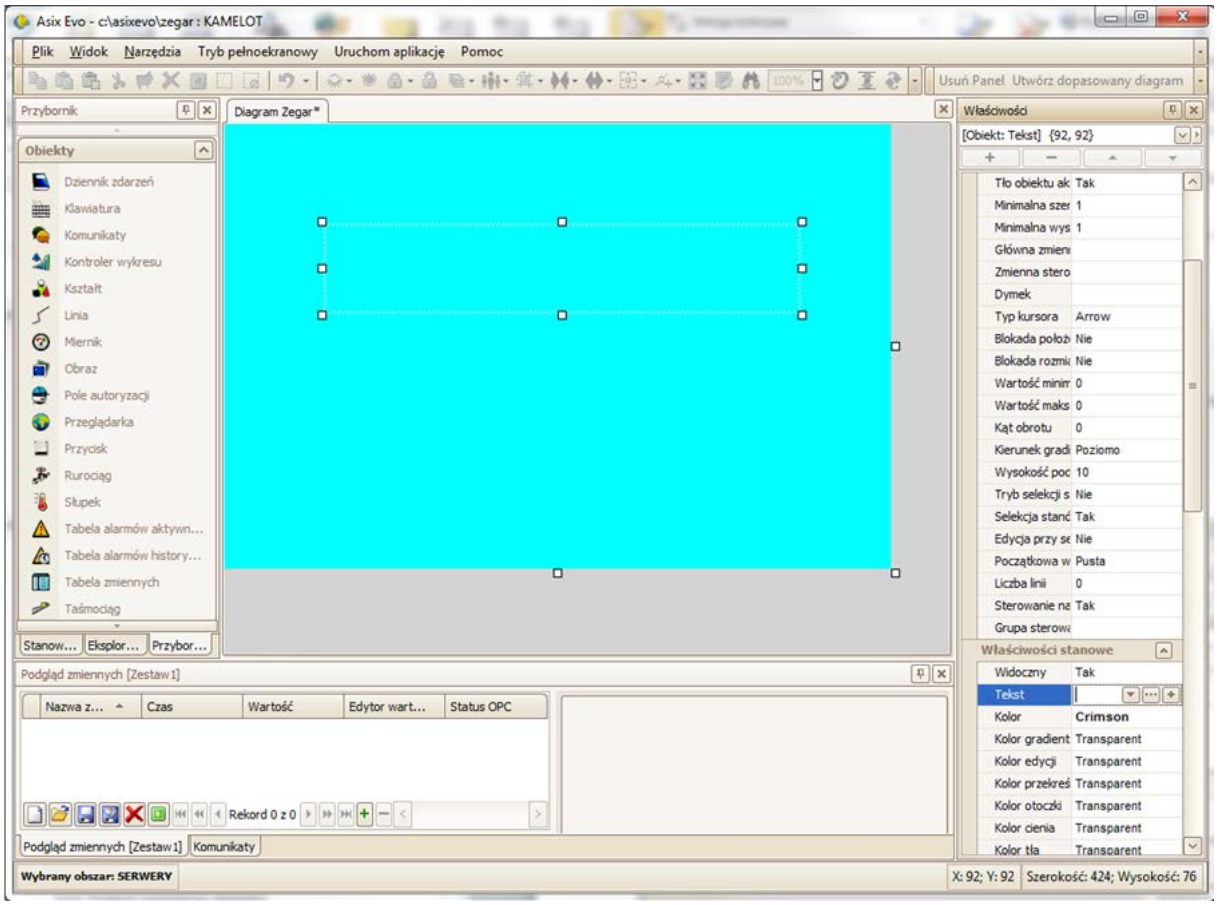
a. **Uaktywnić panel *Przybornik*. (menu *Widok* > *Przybornik*)**

b. **Rozwinąć sekcję *Obiekty* przybornika.**

c. **Wybrać przez kliknięcie obiekt typu *Tekst.***

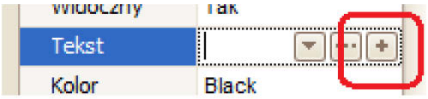
d. **Przy pomocy myszki zaznaczyć w polu diagramu obszar, w którym aktualny czas będzie wyświetlany.**

Następna ilustracja pokazuje okno programu po wykonaniu tych czynności.



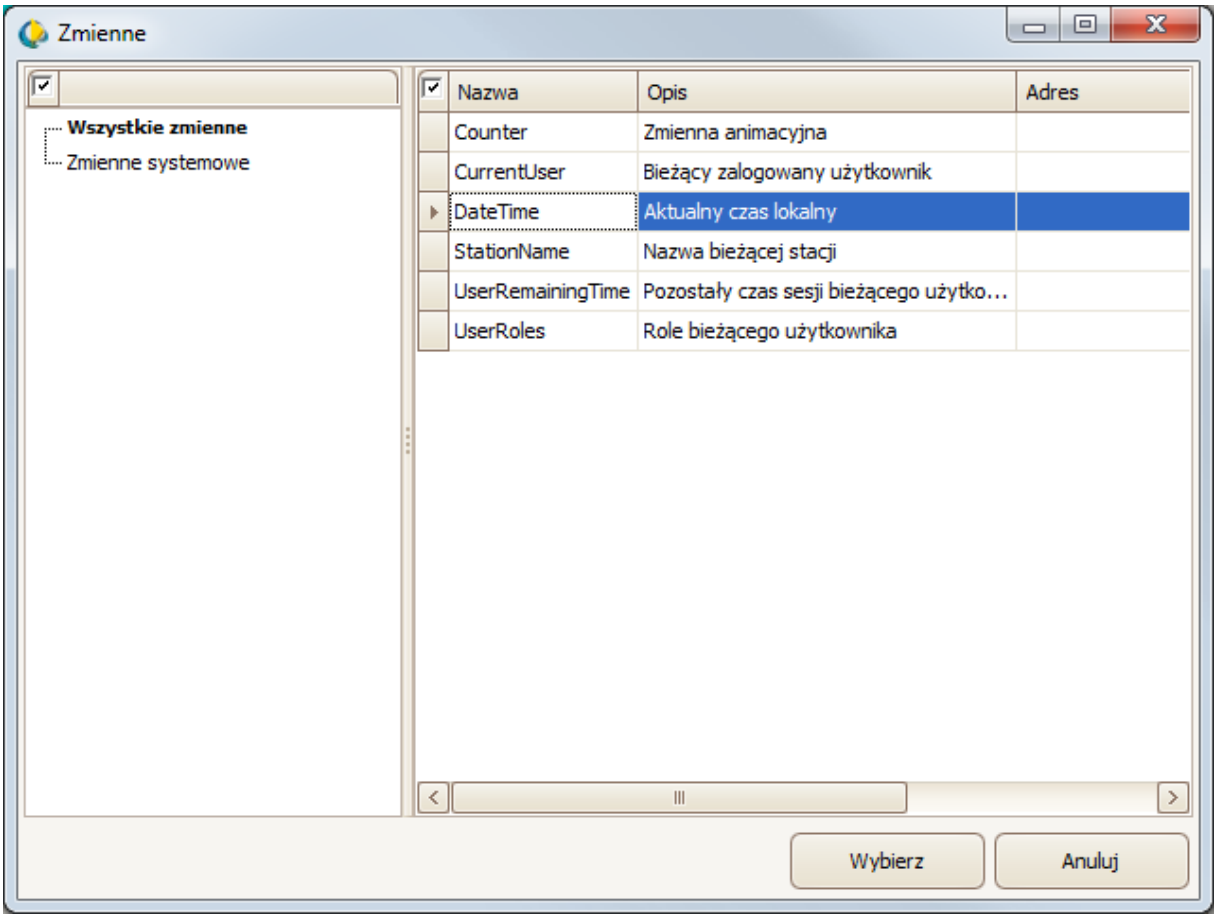
Rys. Okno edycji diagramu - po wstawieniu obiektu Tekst.

Obiekt *Tekst* jeszcze nie wyświetla niczego. Żeby to zmienić należy w panelu *Właściwości* zlokalizować właściwość *Tekst* obiektu.



Rys. Obiekt Tekst - określanie treści.

Przy pomocy ostatniego z przycisków otwieramy okno wyboru zmiennej procesowej.

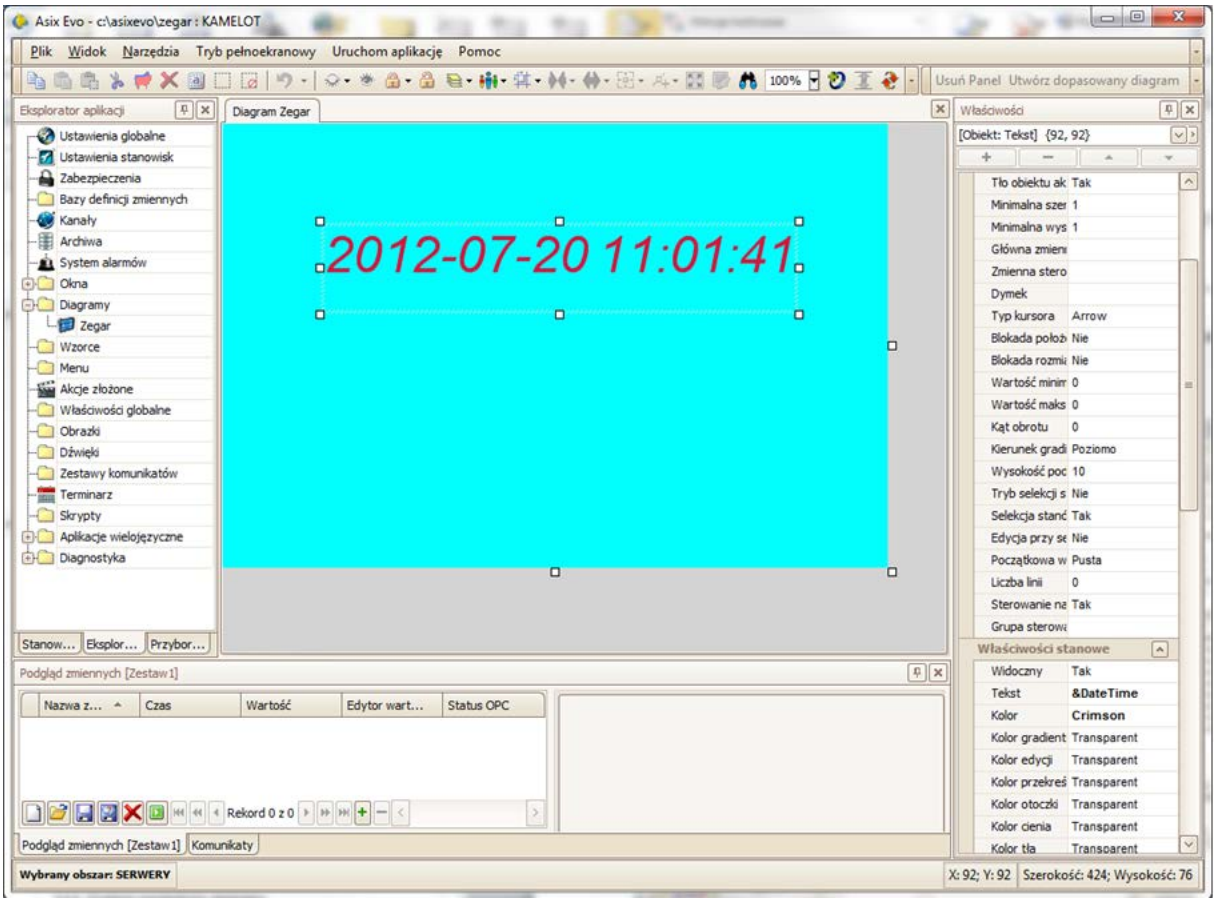


Rys. Okno wyboru zmiennej.

Na tym etapie dostępny jest tylko predefiniowany zestaw zmiennych systemowych opisujących wewnętrzny stan aplikacji. Zmiennych tych nie trzeba definiować. Poza tym zmienne te są używane tak samo jak zmienne procesowe pobierane ze sterowników PLC i mierników. Należy wybrać zmienną *DateTime*, której wartość jest zawsze równa aktualnemu czasowi.

Efekt tych operacji pokazany jest na następnej ilustracji. Zmienione zostały też właściwości *Kolor* i *Styl czcionki*. Wartości zmienionych właściwości wyświetlane są pogrubioną czcionką.

W polu właściwości *Tekst* znajduje się wartość *&DateTime.* Znak & jest skrótową notacją oznaczającą odwołanie do wartości zmiennej procesowej podanej po znaku & (inne metody odwołania do wartości zmiennych zostaną podane w następnych rozdziałach dokumentacji). Ponieważ wartość zmiennej jest typu *DateTime* (nie mylić z nazwą zmiennej), to obiekt automatycznie formatuje wyświetlany tekst zgodnie z ustawieniami systemowymi daty i czasu (metody jawnego formatowania zostaną podane w następnych rozdziałach).



Rys. Okno edycji diagramu - po parametryzacji obiektu Tekst.

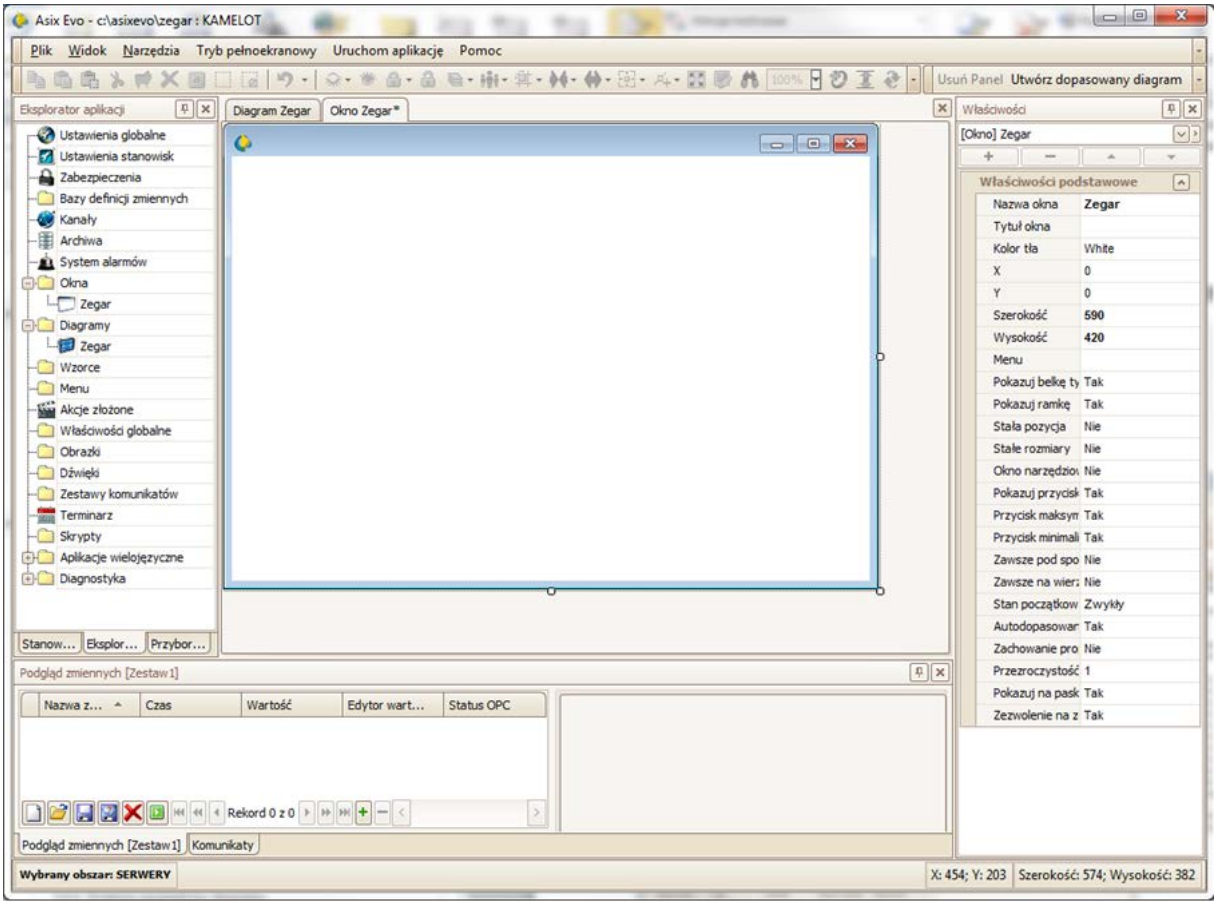
2.3 Edycja okna startowego

Kolejnym krokiem jest utworzenia okna, w którym będzie wyświetlony diagram *Zegar*. Okno to będzie uruchamiane automatycznie na starcie programu w trybie aplikacyjnym.

W systemie Asix.Evo **diagramy obrazujące stan procesu wyświetlane są zawsze w ramach okna**. Okna są obiektami systemu Windows i posiadają typowe właściwości okien, np. linia nagłówka, menu, ramka. **Wewnątrz okna w tzw. panelach wyświetlane są diagramy**. W każdej aplikacji jest zdefiniowane przynajmniej jedno okno.

Tworzymy nowe okno poprzez użycie komendy *Utwórz okno* z menu kontekstowego elementu *Okna* drzewa eksploratora aplikacji. Następnie zmieniamy nazwę okna we właściwości *Nazwa okna,* oraz ustalamy rozmiar okna. **Ustawienie rozmiaru okna jest bardzo istotne, ponieważ jest to rzeczywisty rozmiar okna w momencie uruchomienia aplikacji.**

Efekt jest widoczny na następnej ilustracji.



Rys. Panel edycji okna aplikacji.

Kolejny krok to dodanie do okna panelu. W naszej aplikacji wystarczający będzie pojedynczy panel wypełniający cały obszar okna. W tym celu należy wykonać:

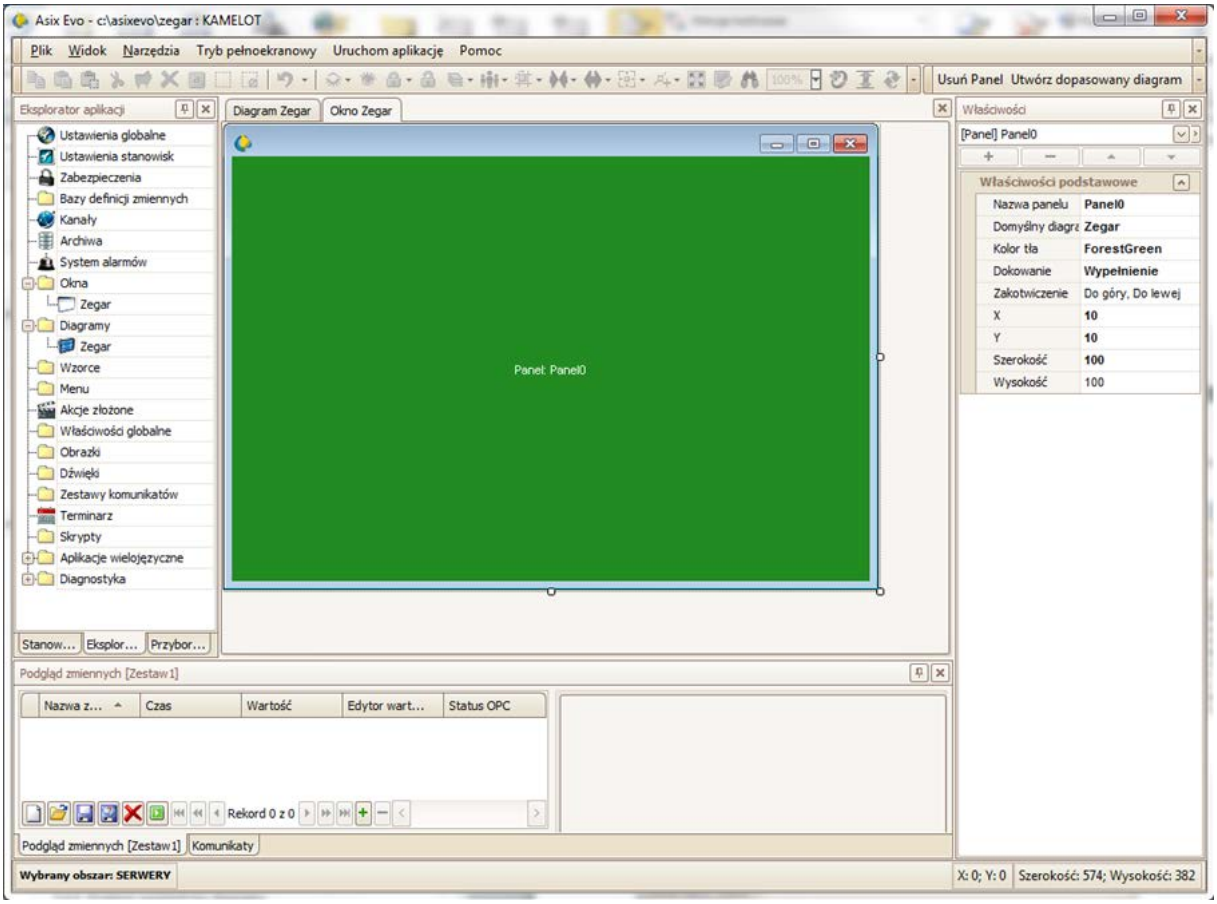
a. **Z menu kontekstowego okna wykonać komendę *Dodaj Panel*.**

b. **Tryb dokowania panelu ustawić na *Wypełnienie*. (Podwójne kliknięcia w polu właściwości *Dokowanie* przełączają kolejno dopuszczalne wartości.)**

c. **Ustawić właściwość *Domyślny diagram* na *Zegar*. Właściwość ta określa jaki diagram zostanie wyświetlony w panelu w momencie**

otwarcia okna. W trakcie dalszego działania aplikacji można załadować do panelu dowolny inny diagram.

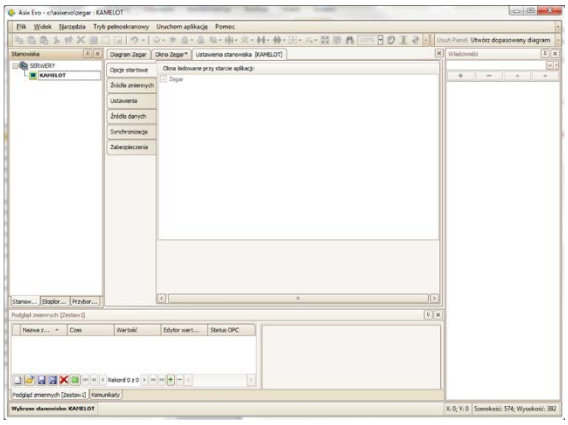
Efekt tych działań pokazany jest poniżej.



Rys. Panel edycji okna aplikacji - widok z wstawionym panelem.

2.4 Ustawienie parametrów startowych

Kolejnym krokiem jest ustawienie okna startowego aplikacji. Najpierw podwójnym kliknięciem na elemencie *Ustawienia stanowisk* otwieramy panel edycji ustawień aplikacji zależnych od aktywnego stanowiska. Dodatkowo wybieramy jeszcze wyświetlenie panelu *Stanowiska (*menu *Widok > Stanowiska).*



Rys. Ustawienia stanowiska.

W momencie utworzenia aplikacji zostało także utworzone pojedyncze stanowisko pracy o nazwie zgodnej z nazwą komputera umieszczone w obszarze o nazwie *Serwery*. Można oczywiście tworzyć więcej stanowisk, ale w naszym przypadku nie jest to konieczne. Obowiązuje hierarchiczne dziedziczenie ustawień. Ustawienia z obszaru nadrzędnego obowiązują na stanowisku, o ile nie zostały jawnie przedefiniowane. Poprzez klikanie na węzłach drzewa stanowisk można przełączyć się na edycję ustawień dla właściwego stanowiska lub obszaru.

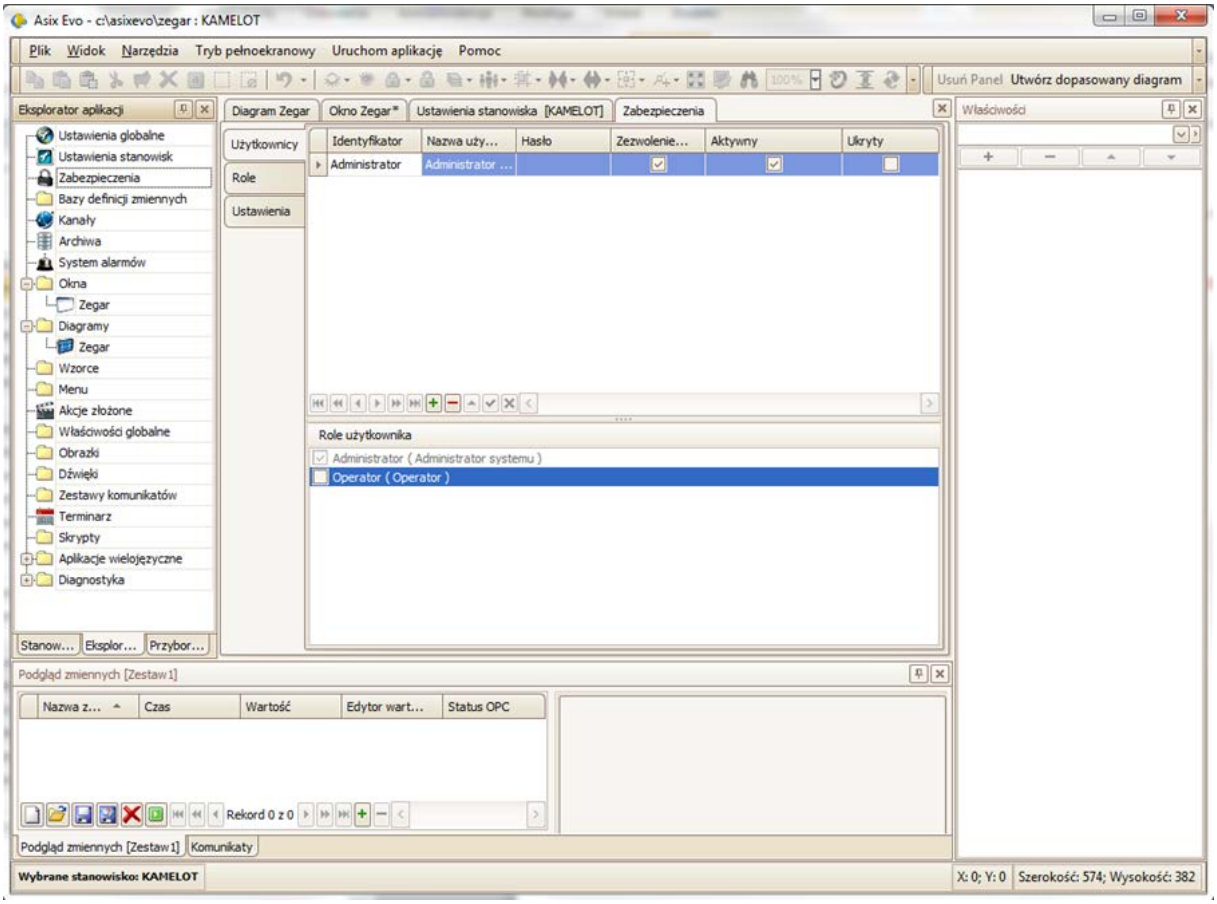
Na dolnym pasku statusowym wyświetlana jest nazwa wybranego stanowiska/obszaru, którego parametry są aktualnie edytowane. Natomiast w linii tytułu wyświetlana jest nazwa stanowiska w kontekście, którego program aktualnie pracuje. Nie ma ona nic wspólnego z edycją parametrów. Jeżeli nie zostało to inaczej ustawione, to domyślnie jest to stanowisko o nazwie zgodnej z systemową nazwą komputera.

Należy pamiętać o tym, że wariantowym ustawieniom podlegają jedynie parametry z panelów *Ustawienia stanowisk, Kanały* i *Archiwa*. Większość pozostałych elementów aplikacji jest wspólna dla wszystkich stanowisk (np. maski i okna). Niektóre elementy mają własny system wariantowych ustawień.

Jedynym ustawieniem, które należy zmienić w przypadku naszej aplikacji jest zaznaczenie okna *Zegar* jako okna otwieranego na starcie aplikacji.

2.5 Ustawienia zabezpieczeń

Ostatnią rzeczą jest dokończenie parametryzacji zabezpieczeń. Po podwójnym kliknięciu na węźle *Zabezpieczenia* w drzewie elementów *Eksploratora Aplikacji* wyświetlony zostanie panel edycji zabezpieczeń.



Rys. Panel edycji zabezpieczeń.

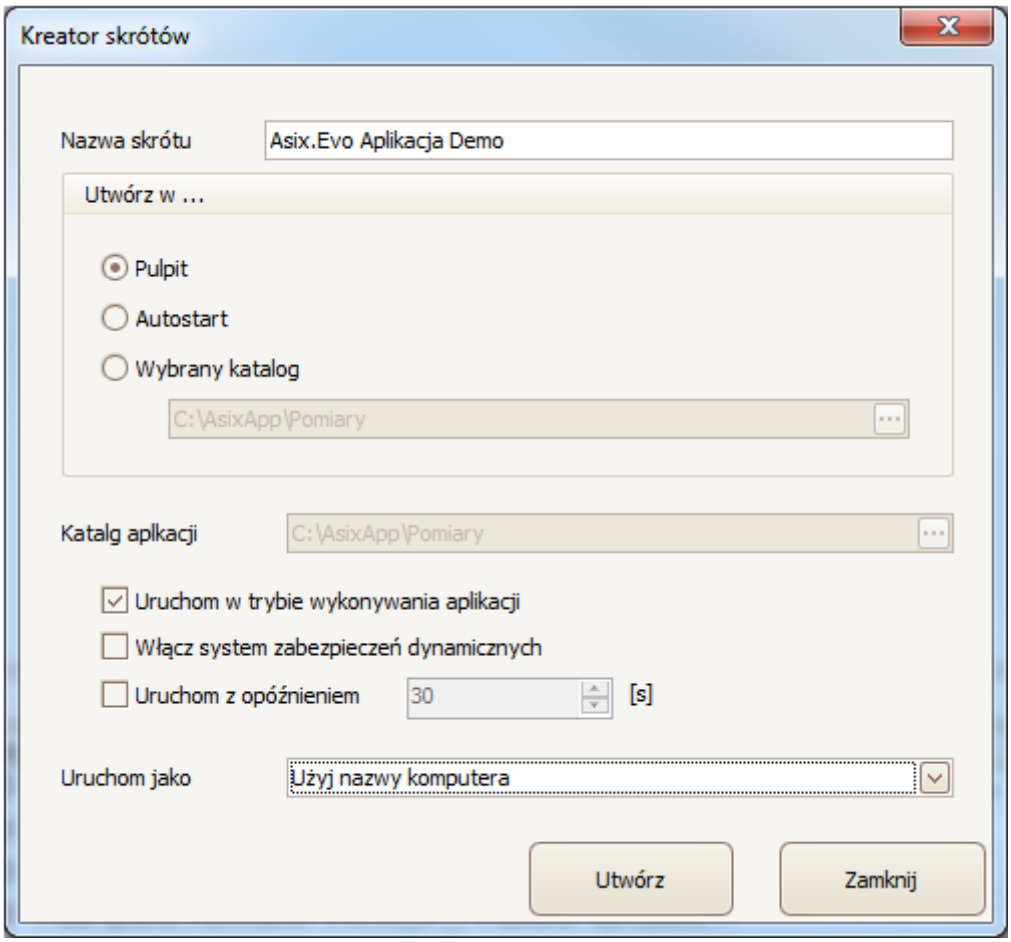
W momencie utworzenia aplikacji został stworzony użytkownik *Administrator* o pełnych uprawnieniach. Warto teraz ustawić hasło logowania tego użytkownika – po utworzeniu użytkownika hasło jest puste.

W rzeczywistych aplikacjach z reguły konieczne będzie utworzenie użytkowników o mniejszych uprawnieniach.

Aplikacja jest teraz gotowa. Należy jeszcze korzystając z komendy *Zapisz…* z menu *Plik* zapisywać wszystkie wykonane parametryzacje.

2.6 Utworzenie skrótu

Kolejnym krokiem jest utworzenie skrótu służącego do uruchomienia aplikacji. Najłatwiej to wykonać korzystając z okna kreatora otwieranego komendą *Kreator skrótów…* z menu *Narzędzia*.



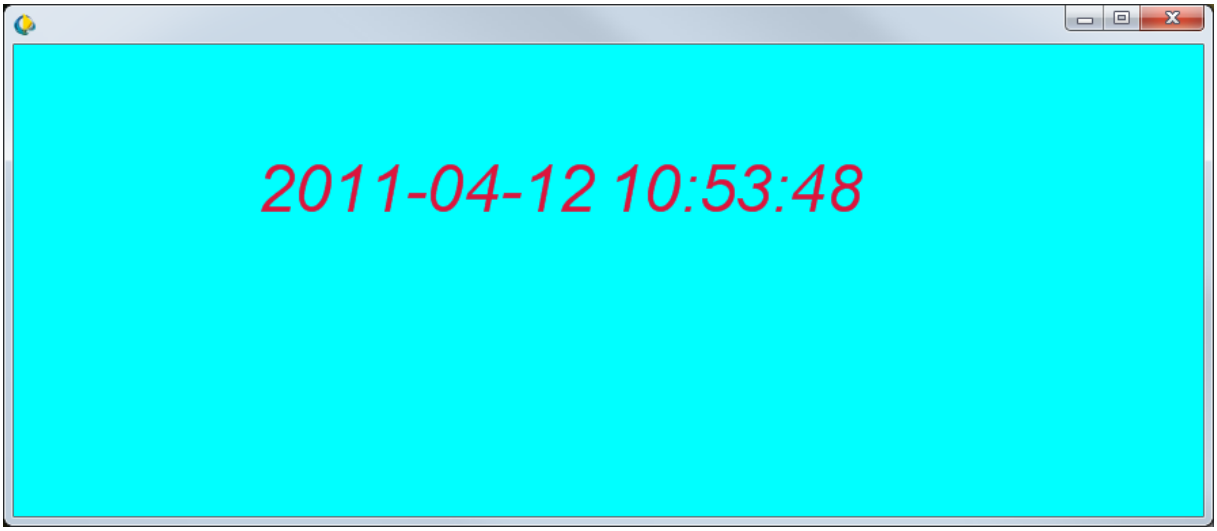
Rys. Okno kreatora skrótów.

Przy pokazanych ustawieniach po naciśnięciu przycisku *Utwórz* na pulpicie zostanie utworzony skrót startujący naszą aplikację.

Można teraz zakończyć działanie programu AsixEvo.

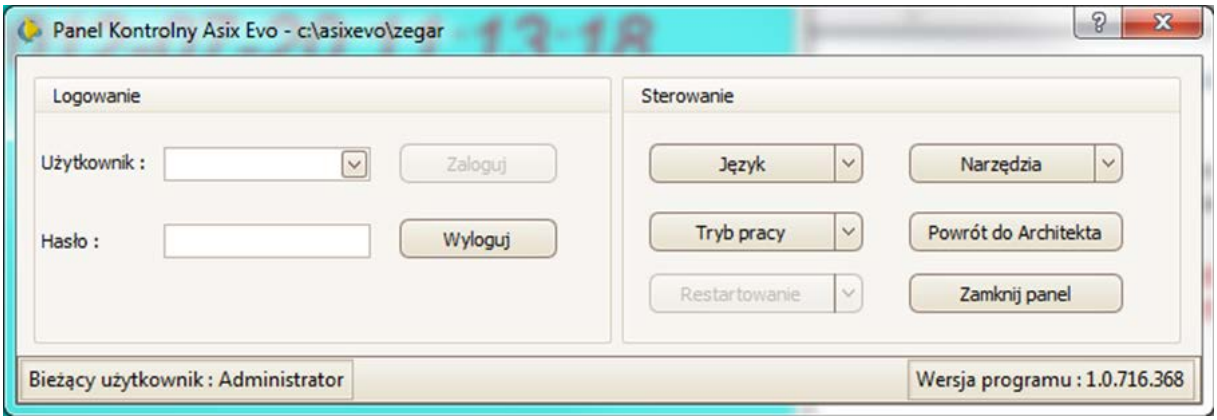
2.7 Uruchomienie aplikacji

Uruchomienie aplikacji poprzez utworzony skrót powoduje otwarcie zaprojektowanego okna.



Rys. Okno uruchomionej aplikacji.

Okno może być swobodnie przemieszczane i skalowane. Obiekt diagramu dopasowuje się do zmian rozmiaru okna. Nie można natomiast zamknąć okna. Wynika to z faktu, że nie jest zalogowany żaden użytkownik. W aplikacjach AsixEvo użytkownik otrzymuje interfejs stworzony całkowicie przez projektanta. Nie są wyświetlane żadne dodatkowe okna. Jedynym elementem interfejsu, który jest automatycznie dodawany do każdej aplikacji jest obsługa przycisku ***Ctrl-Shift-F1***. **Użycie tej kombinacji klawiszy powoduje wyświetlenie okienka *Panelu Kontrolnego***.

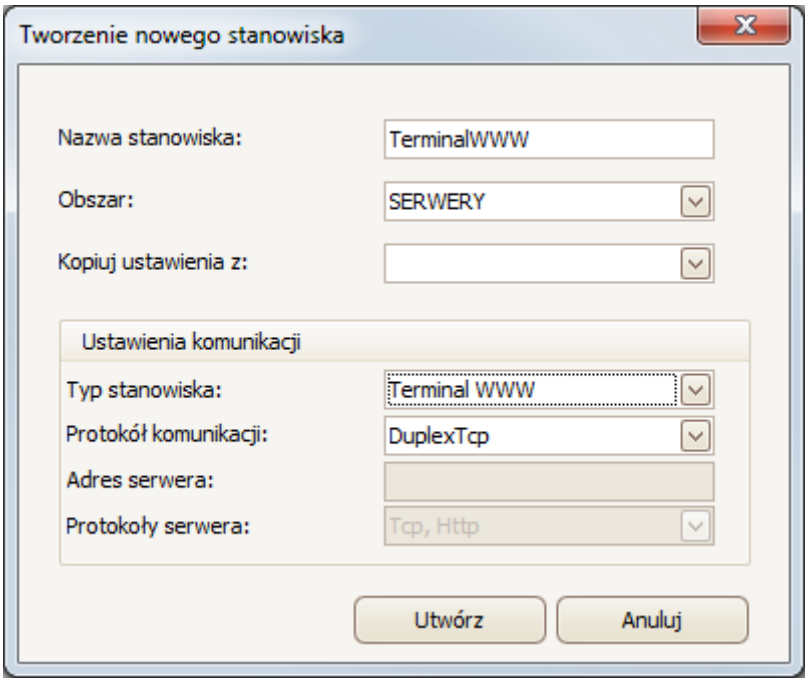


Rys. Okno Panelu Kontrolnego.

Okno to pozwala na wykonanie podstawowych operacji administracyjnych. Po zalogowaniu na konto *Administrator* będzie możliwe zakończenie pracy aplikacji.

2.8 Uruchomienie aplikacji w trybie przeglądarki

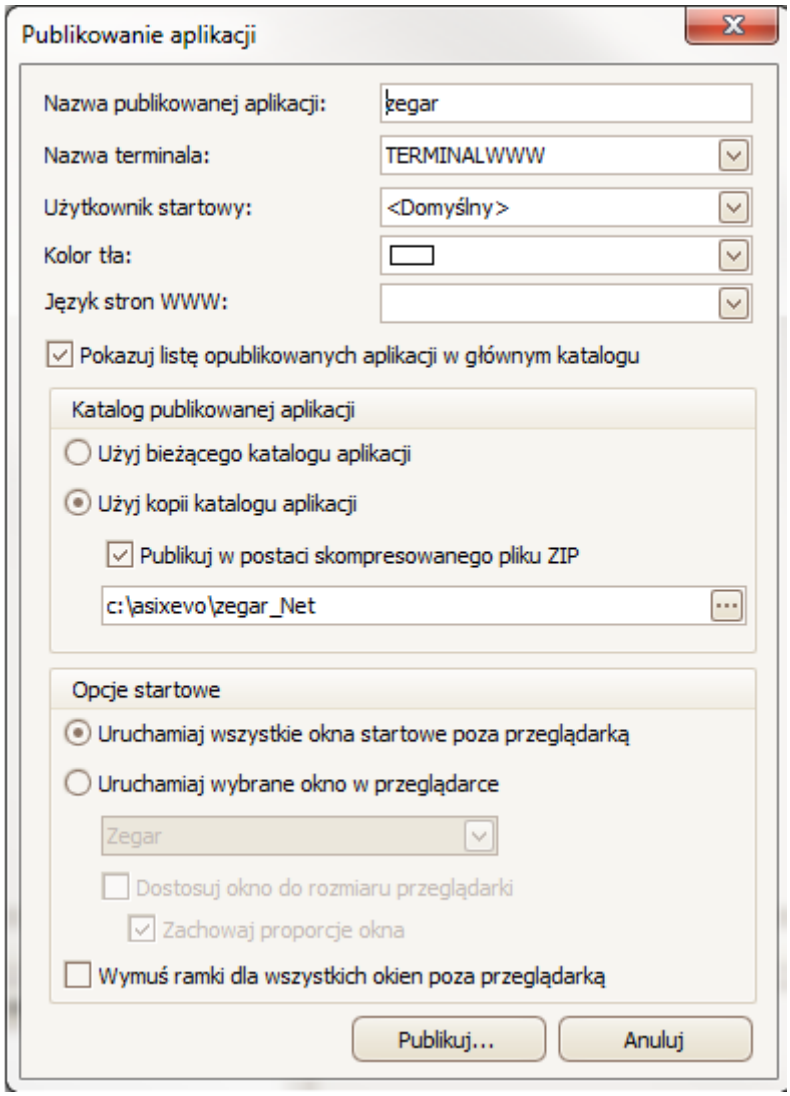
Pierwszym krokiem w procesie uruchomienia aplikacji w wersji przeglądarkowej jest stworzenie nowego stanowiska, w kontekście którego będą uruchamiane wszystkie instancje aplikacji na komputerach klienckich. W tym celu w panelu roboczym *Stanowiska* należy korzystając z menu kontekstowego węzła *Serwery* użyć komendę *Nowe stanowisko…*. Wyświetlone zostanie poniższe okno dialogowe.



Rys. Okno tworzenia nowego stanowiska.

Należy teraz podać nazwę stanowiska oraz w polu *Typ stanowiska* wybrać typ *Terminal WWW*. Zostanie utworzone nowe stanowisko w obszarze *SERWERY* . Będzie ono dziedziczyło ustawienia z tego obszaru. W naszym przykładzie nie trzeba nic więcej parametryzować, ale w przypadkach bardziej złożonych wymagana byłaby np. odpowiednia parametryzacja kanałów komunikacyjnych i archiwów.

Kolejnym krokiem jest opublikowanie aplikacji Asix.Evo na serwerze internetowym IIS. Warunkiem wstępnym jest zainstalowanie i odpowiednie skonfigurowanie serwera IIS – najlepiej użyć do tego programu instalacyjnego pakietu Asix.Evo. Do publikacji aplikacji używana jest komenda *Publikuj…* z menu *Plik*. Na następnej ilustracji pokazano przykładową parametryzację procesu publikacji.



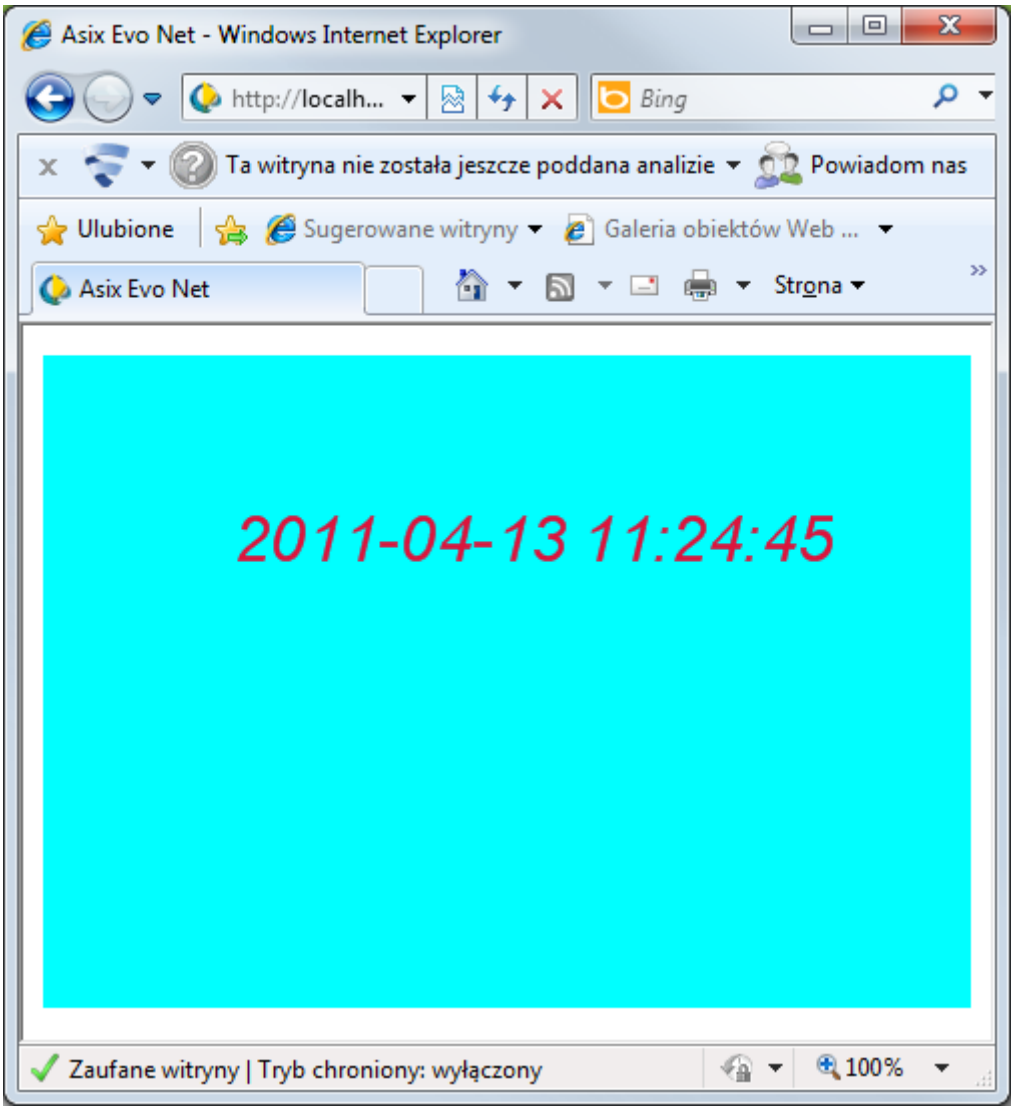
Rys. Okno publikacji aplikacji.

Najważniejsze opcje to *Nazwa publikowanej aplikacji* (używanej następnie jako część adresu www) i *Nazwa terminala*, która musi być nazwą utworzonego wcześniej stanowiska typu *Terminal WWW*.

Po wykonaniu operacji publikacji aplikacja jest już dostępna. Po uruchomieniu przeglądarki internetowej należy podać adres aplikacji Asix.Evo jako *localhost**evonet**zegar*. W miejsce *localhost* można podać nazwę lub adres ip komputera.

UWAGA: W przypadku korzystania z wersji demonstracyjnej lub braku licencji Asix4Internet dostęp do aplikacji będzie możliwy tylko na komputerze, na którym uruchomiony jest serwer IIS.

Poniższa ilustracja pokazuje diagram naszej aplikacji w przeglądarce Internet Explorer.



Rys. Okno aplikacji w przeglądarce IE.

3 Aplikacja z obsługą rzeczywistych danych

Zadaniem kolejnej aplikacji będzie prezentacja wartości bieżących i archiwalnych dla dwóch rzeczywistych zmiennych pomiarowych. Stworzony zostanie pełny szkielet aplikacji z przełączanymi diagramami synoptycznymi. Pokazany zostanie pełny cykl budowy aplikacji, począwszy od utworzenia nowej aplikacji, poprzez utworzenie bazy zmiennych pomiarowych, zdefiniowanie kanałów komunikacyjnych i archiwów, aż po zbudowanie części wizualnej. W dalszej części rozdziału zostanie też pokazane jak przekształcić aplikację jednostanowiskową na aplikację sieciową, złożoną z serwera i stanowiska terminalowego.

Aplikacja będzie pokazywała wartości dwóch pomiarów, każdy na osobnym diagramie. Wartość pomiaru będzie wyświetlana na wirtualnym mierniku i w postaci tekstowej. Oprócz tego będzie wyświetlany wykres pokazujący historię zmian pomiaru. Aplikacja będzie wyposażona w proste menu pozwalające na przełączanie pokazywanych diagramów.

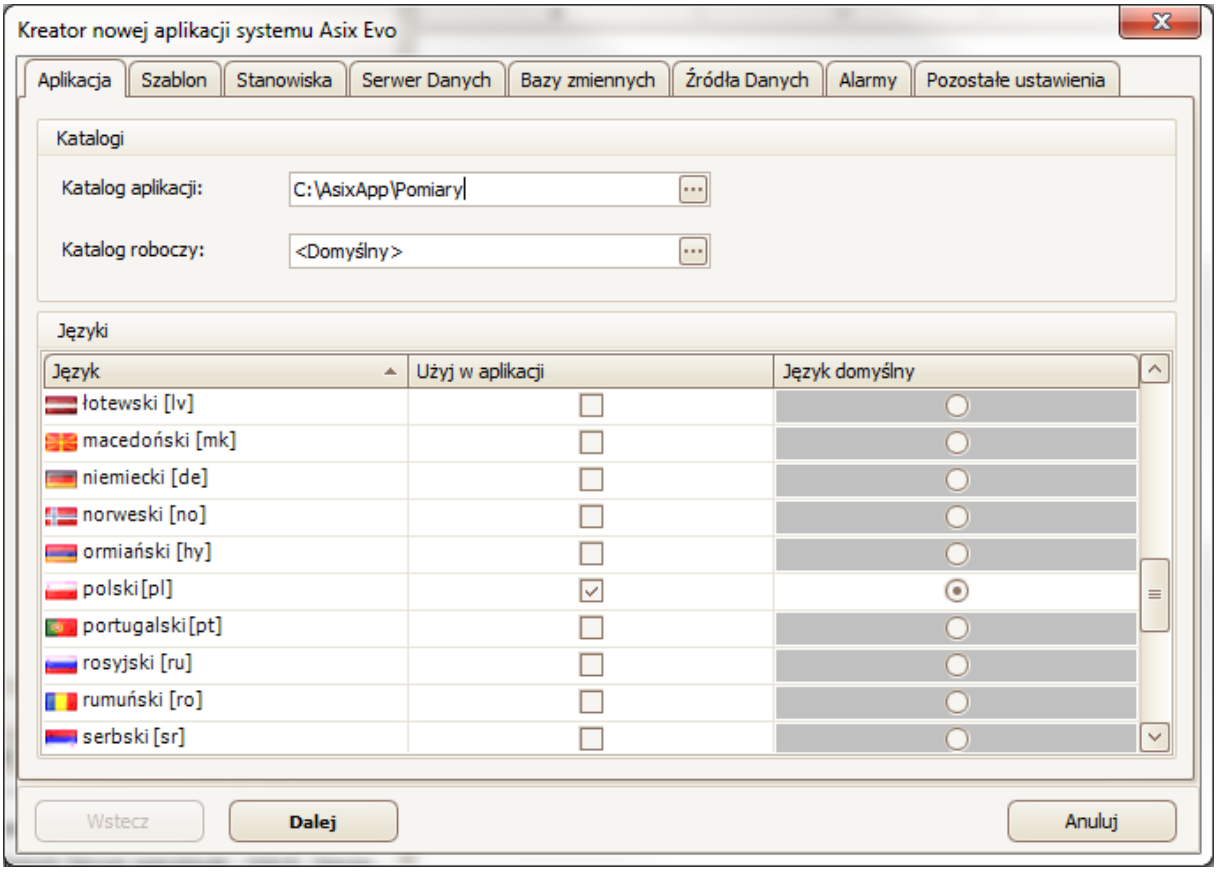
3.1 Utworzenie aplikacji przy pomocy kreatora

Tym razem utworzymy nową aplikację przy pomocy kreatora. Jest to zalecana metoda postępowania. Kreatora uruchamiamy przy pomocy komendy *Nowa aplikacja – kreator* z menu *Plik*. Kreator pozwala w momencie utworzenia aplikacji, wygenerować jej szkielet oraz ustawić wstępnie wiele parametrów niezbędnych do pracy.

3.1.1 Lokalizacja aplikacji i wybór języków

Pierwsza zakładka kreatora pozwala na podanie katalogu, w którym pliki definicyjne aplikacji będą zapisywane. W razie konieczności można także podać katalog, w którym będą przechowywane pliki robocze: archiwa, logi itp..

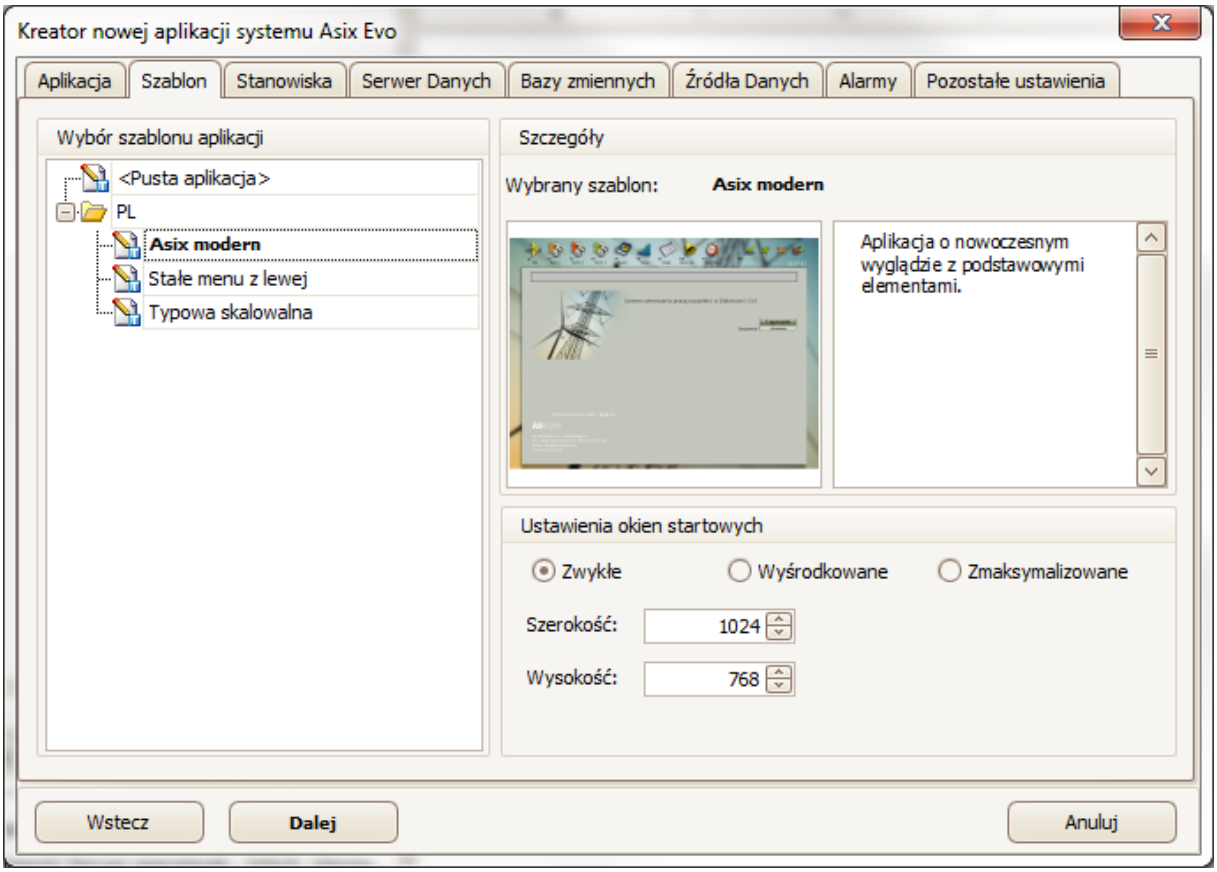
Można też określić w jakich językach aplikacja będzie używana. Standardowe ustawienie to pojedynczy język wybrany zgodnie z ustawieniami systemu operacyjnego.



Rys. Kreator aplikacji - zakładka 'Aplikacja'.

3.1.2 Szablon aplikacji

Ta zakładka pozwala na wygenerowanie szkieletu aplikacji – początkowego zestawu okien, diagramów i symboli graficznych.

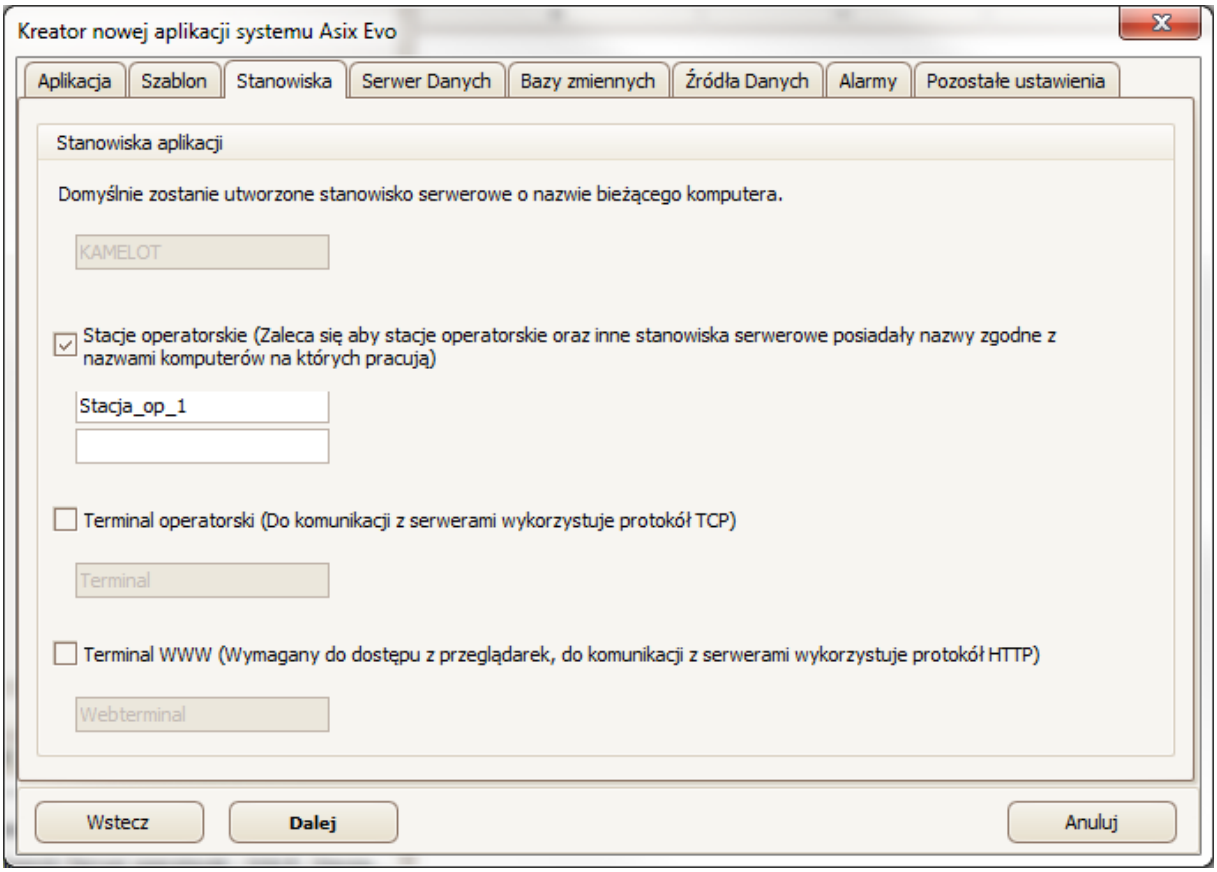


Rys. Kreator aplikacji - zakładka 'Szablon'.

Można wybrać któryś z proponowanych wariantów, ale w przypadku naszej aplikacji wybieramy wariant pustej aplikacji. Będziemy budować aplikację od podstaw.

3.1.3 Konfiguracja stanowisk

Zakładka *Stanowiska* pozwala na utworzenie wybranego zestawu stanowisk dostosowanego do wymagań aplikacji.



Rys. Kreator aplikacji - zakładka 'Stanowiska'.

Kreator standardowo zakłada jedno stanowisko o nazwie zgodnej z nazwą komputera, na którym aplikacja jest tworzona. Oprócz tego można dodać jeszcze kilka stanowisk o różnym charakterze. Stanowiska te z założenia, będą służyły do wykonywania aplikacji. W razie konieczności utworzenia większej liczby stanowisk, należy to wykonać po zakończeniu kreatora w panelu roboczym.

W naszym przypadku, oprócz stanowiska edycyjnego tworzymy tylko jedno dodatkowe stanowisko stacji operatorskiej. W dalszej części przykładu pokażemy metodę ręcznego dodania stanowiska typu terminalowego.

3.1.4 Określenie sposobu pozyskiwania danych procesowych

W aplikacjach Asix.Evo dostęp do danych procesowych bieżących i archiwalnych odbywa się z wykorzystaniem jednej z trzech strategii:

a. Dostęp poprzez serwer danych.

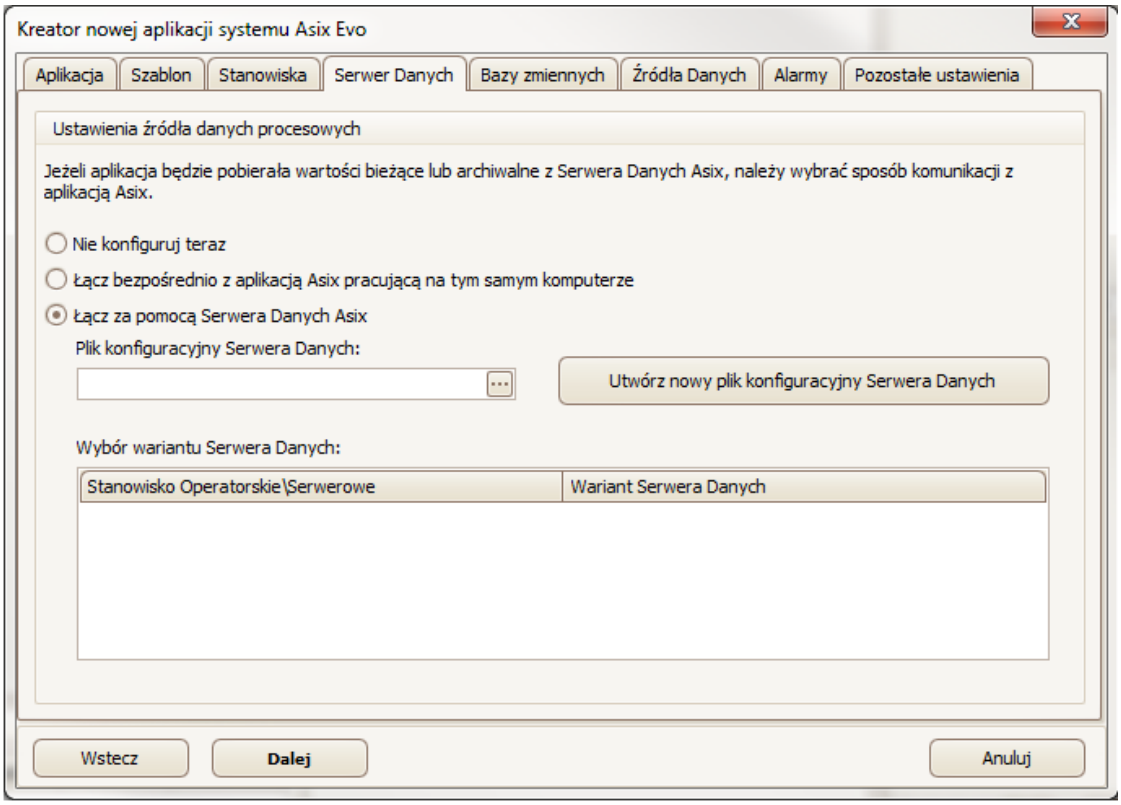
Podstawowy tryb odczytu danych. Serwer danych jest modułem programowym, który wewnętrznie uruchamia standardowe moduły Asmen, Aspad i Aslink znane z wcześniejszych wersji pakietu. Pozwala to wykorzystać wszystkie dotychczasowe drajwery komunikacyjne pakietu Asix do odczytu danych z urządzeń fizycznych. Możliwa jest także dwustronna wymiana danych z niezależnymi stanowiskami wykonującymi aplikacje Asix starego typu poprzez połączenia sieciowe Aslink'a.

b. Dostęp z bezpośrednim połączeniem z lokalną aplikacją Asix.

Rzadko używany tryb pozwalający na dostęp do danych poprzez aplikację Asix starego typu uruchomioną na tym samym komputerze. Typowym przypadkiem jest uruchomienie serwera Asix.Evo jako źródła danych dla aplikacji przeglądarkowej.

c. Dostęp sieciowy mechanizmami Asix.Evo.

Tryb używany na stanowiskach terminalowych okienkowych i WWW – stanowisko pobiera dane z serwera Asix.Evo, żadne elementy serwera danych nie są używane.



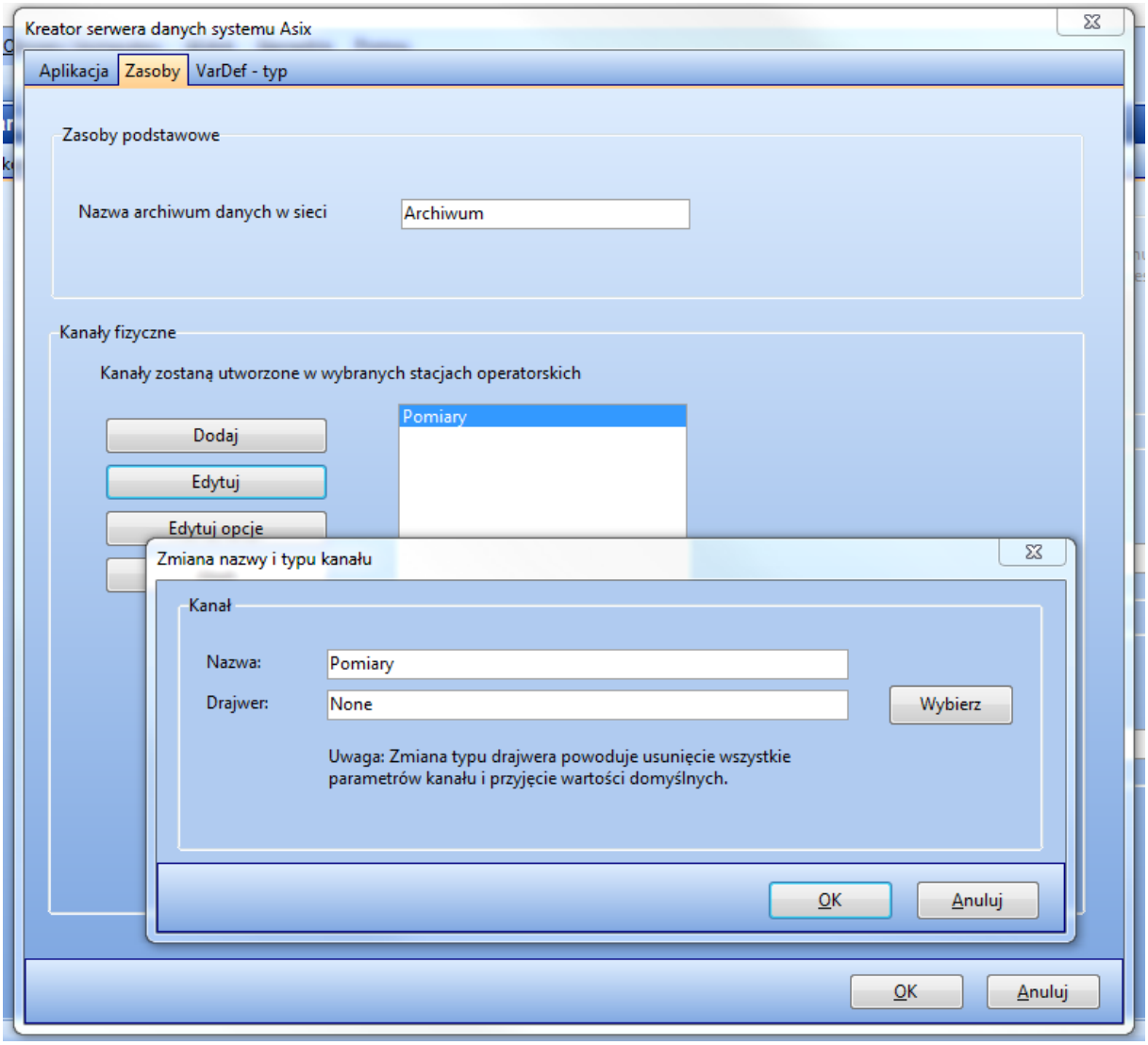
Rys. Kreator aplikacji - zakładka 'Serwer Danych'.

W naszej aplikacji będziemy stosować dostęp do danych poprzez Serwer Danych Asix. Żeby skonfigurować Serwer należy użyć przycisk *Utwórz nowy plik konfiguracyjny Serwera Danych*. Spowoduje to uruchomienie programu *Architekt* w trybie kreatora aplikacji Asix.Evo.

Program Architekta służy także do konfiguracji aplikacji starego typu. Szczegółowy opis jego użycia znajduje się osobnym podręczniku.

Patrz: ***podręcznik Architekta - plik: Architekt.PDF/CHM***.

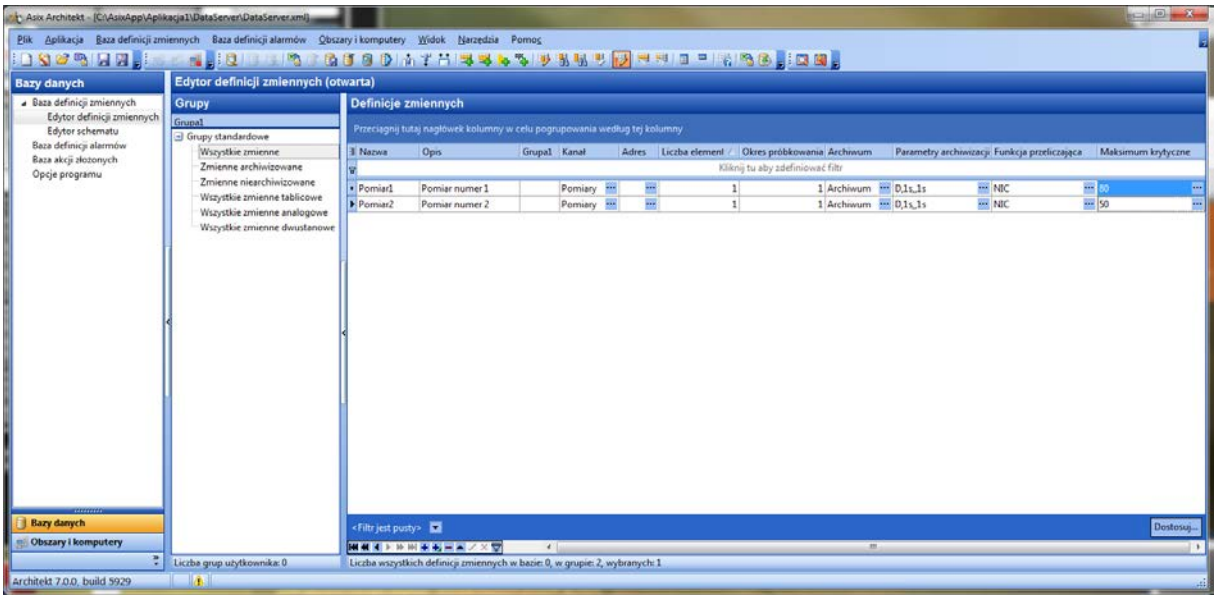
W trybie kreatora aplikacji Asix.Evo wyświetlane jest poniższe okno.



Rys. Kreator serwera danych systemu Asix.

Dla nas najważniejsza jest zakładka *Zasoby.* Służy ona do założenia archiwum danych procesowych oraz konfiguracji kanałów komunikacyjnych. W polu *Nazwa archiwum danych w sieci* wprowadzamy nazwę *Archiwum*. Następnie dodajemy kanał komunikacyjny o nazwie *Pomiary*. Jeżeli posiadamy dostęp do sterownika, to należy wybrać właściwy drajwer komunikacyjny. Jeżeli nie mamy sterownika, to wybieramy drajwer typu *None*. Pozwoli to nam na symulację danych. W przypadku rzeczywistych drajwerów warto też użyć przycisk *Edytuj opcje* w celu skonfigurowania parametrów pracy drajwera.

Po zamknięciu okna kreatora, program Architekt otworzy panel edycyjny bazy definicji zmiennych procesowych.



Rys. Edytor definicji zmiennych procesowych.

Każda zmienna procesowa opisana jest zestawem atrybutów. Początkowo wyświetlany jest tylko zestaw podstawowy. Używając przycisk  znajdujący się z lewej strony linii nagłówków kolumn, można włączyć wyświetlanie dodatkowych atrybutów. Ponieważ w aplikacji będziemy potrzebować atrybut *Maksimum krytyczne,* to musimy go teraz włączyć.

W aplikacji będziemy używać dwóch zmiennych o nazwach *Pomiar1* i *Pomiar2*. Przy pomocy przycisku *+* paska narzędziowego dodajemy kolejno dwie zmienne. Ich definicja powinna być zgodna z poniższą tabelką.

Tabela: Przykładowa deklaracja zmiennych procesowych.

	Pomiar1	**Pomiar2**
Nazwa	Pomiar1	Pomiar2
Opis	Pomiar numer 1	Pomiar numer 2
Kanał	Pomiary	Pomiary
Liczba elementów	1	1
Okres próbkowania	1	1
Archiwum	Archiwum	Archiwum
Parametry archiwizacji	D,1s,,1s	D,1s,,1s
Funkcja przeliczająca	NIC	NIC
Maksimum krytyczne	80	50

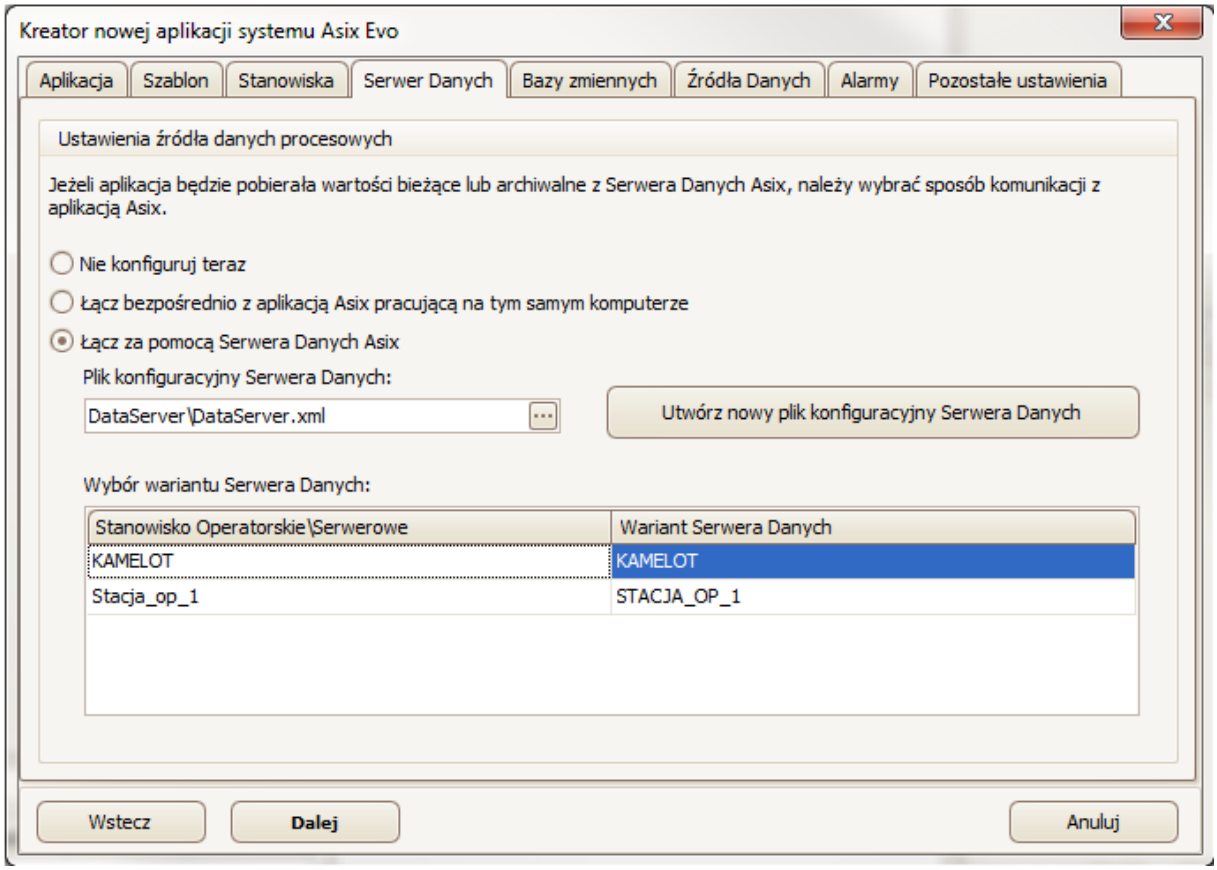
Uwaga:

W przypadku tworzenia zmiennych o zbliżonej definicji warto skorzystać z komendy *Wstaw kopie definicji zmiennej* z menu kontekstowego edytora bazy definicji zmiennych.

Jeżeli w trakcie zakładania kanału *Pomiary* wybraliśmy inny drajwer niż *None*, to konieczne będzie też podanie adresu zmiennej w sterowniku w atrybucie *Adres*. Sposób adresowania zależy od wybranego drajwera i jest opisany w jego dokumentacji.

Po wprowadzeniu definicji zmiennych kończymy pracę programu *Architekt* i wracamy do kreatora aplikacji Asix.Evo.

Poniżej pokazane jest okno kreatora po zakończeniu pracy programu Architekt.



Rys. Kreator aplikacji - zakładka 'Serwer Danych' - po zakończeniu pracy programu Architekt.

Okno zostało uzupełnione o nazwę pliku konfiguracyjnego serwera danych oraz zostały ustawione powiązania pomiędzy stanowiskami aplikacji Evo a wariantami pracy serwera danych. Ze względu na to, że tworzyliśmy nowy plik konfiguracyjny serwera danych, to warianty serwera zostały automatycznie utworzone zgodnie z konfiguracją stanowisk aplikacji Evo. Jeżeli użylibyśmy istniejącego pliku serwera danych (np. z aplikacji utworzonej wcześniejszą wersją programu), to w przypadku braku zgodności nazw stanowisk i wariantów powiązanie należałoby wykonać ręcznie.

3.1.5 Deklaracja baz zmiennych procesowych

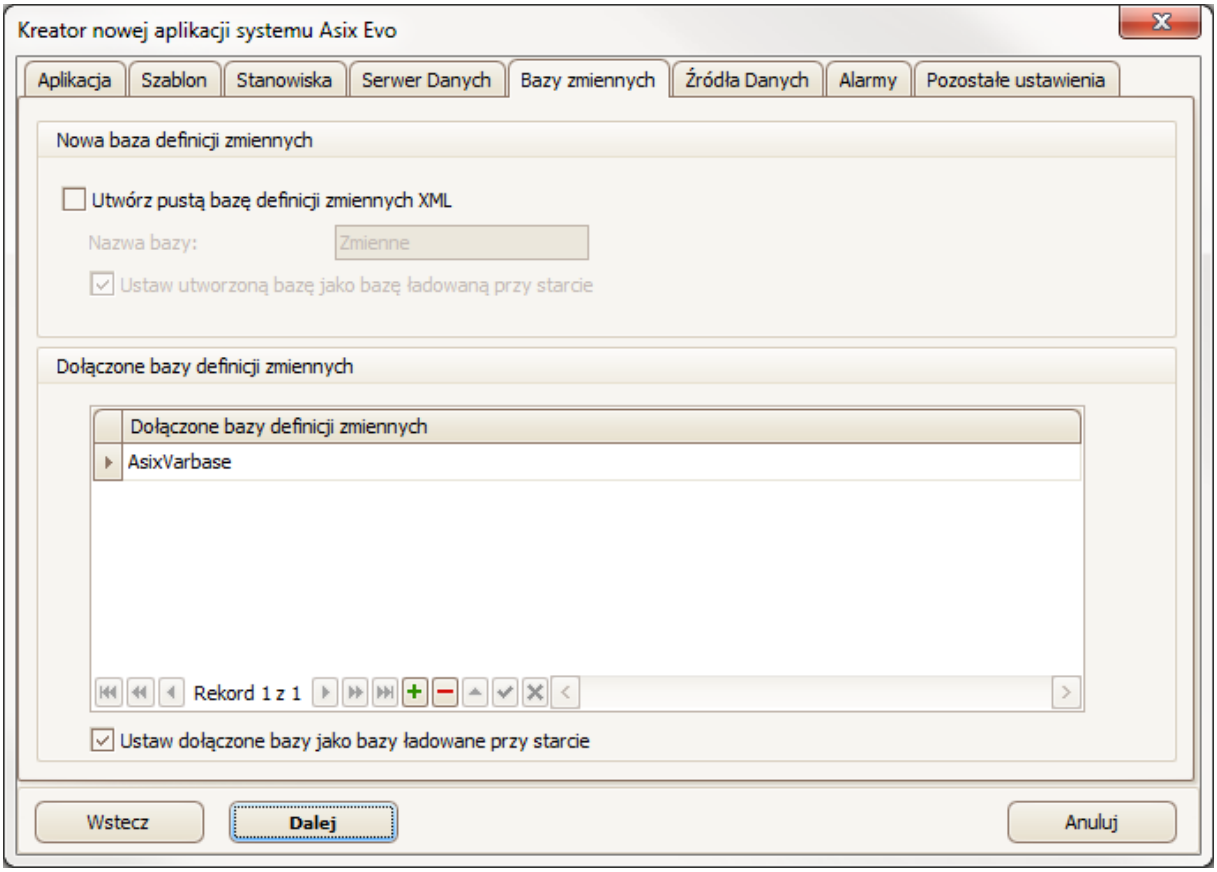
Kolejna zakładka kreatora dotyczy konfiguracji baz zmiennych. Bazy definicji zmiennych są kluczowym elementem każdej aplikacji. Aplikacja powinna posiadać przynajmniej jedną bazę zmiennych. Asix.Evo może używać bazy zmiennych w trzech formatach:

a. Bazy zmiennych w formacie XML

b. Bazy zmiennych w formacie MDB, używane także we wcześniejszych wersjach systemu Asix

c. Bazy zmiennych przechowywane w bazach danych typu SQL, również używane we wcześniejszych wersjach pakietu

W praktyce, bazy XML stosowane są wyłącznie do definicji tzw. zmiennych wirtualnych używanych w aplikacji, których wartości wyliczane są przy pomocy wewnętrznych obliczeń, np. skryptów.



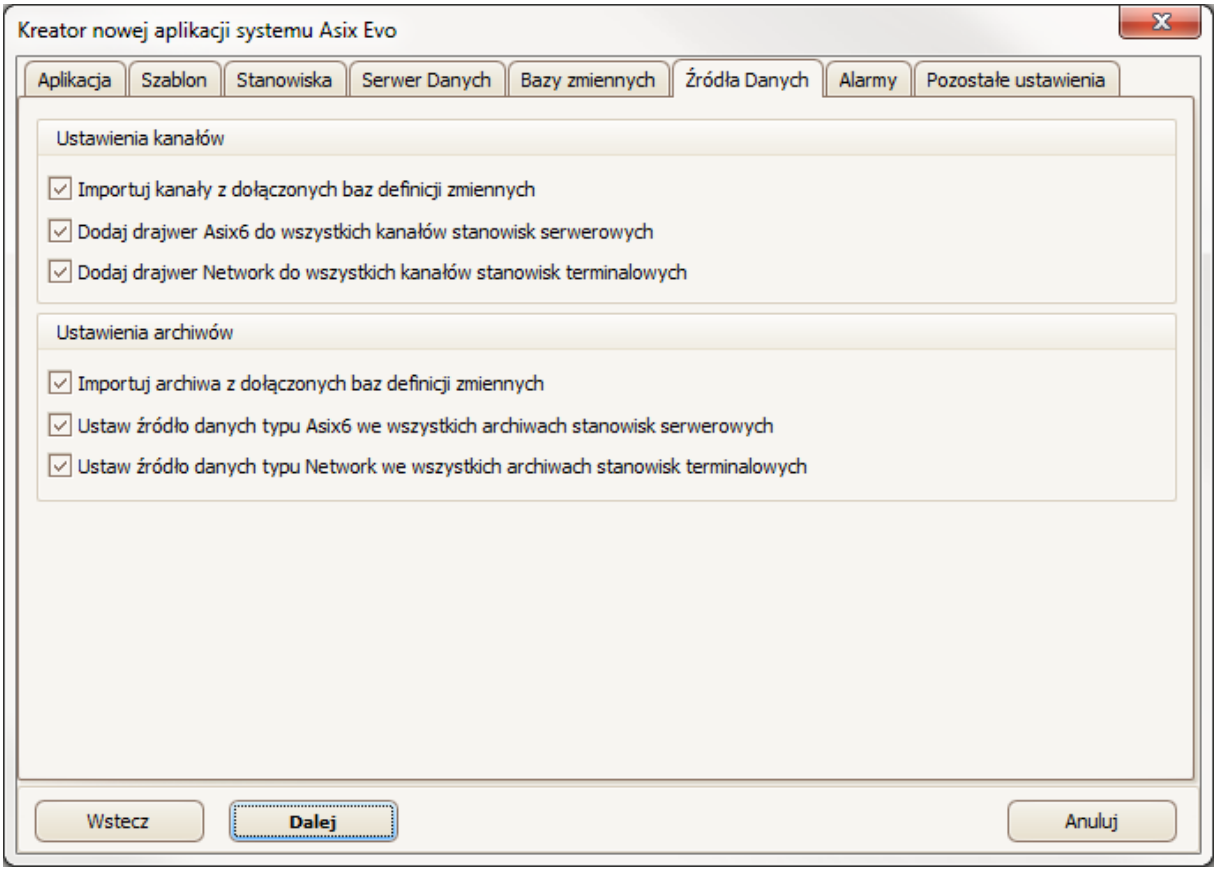
Rys. Kreator aplikacji - zakładka 'Bazy zmiennych'.

W wyniku wcześniejszych kroków kreatora mamy już wybraną utworzoną programem *Architekt* bazę zmiennych.

Na tym etapie moglibyśmy utworzyć nową bazę typu XML lub dołączyć inne utworzone wcześniej bazy MDB. W naszej przykładowej aplikacji nie jest to jednak konieczne.

3.1.6 Konfiguracja źródeł danych

Zakładka *Źródła danych* pozwala wstępnie skonfigurować kanały komunikacyjne i archiwa.

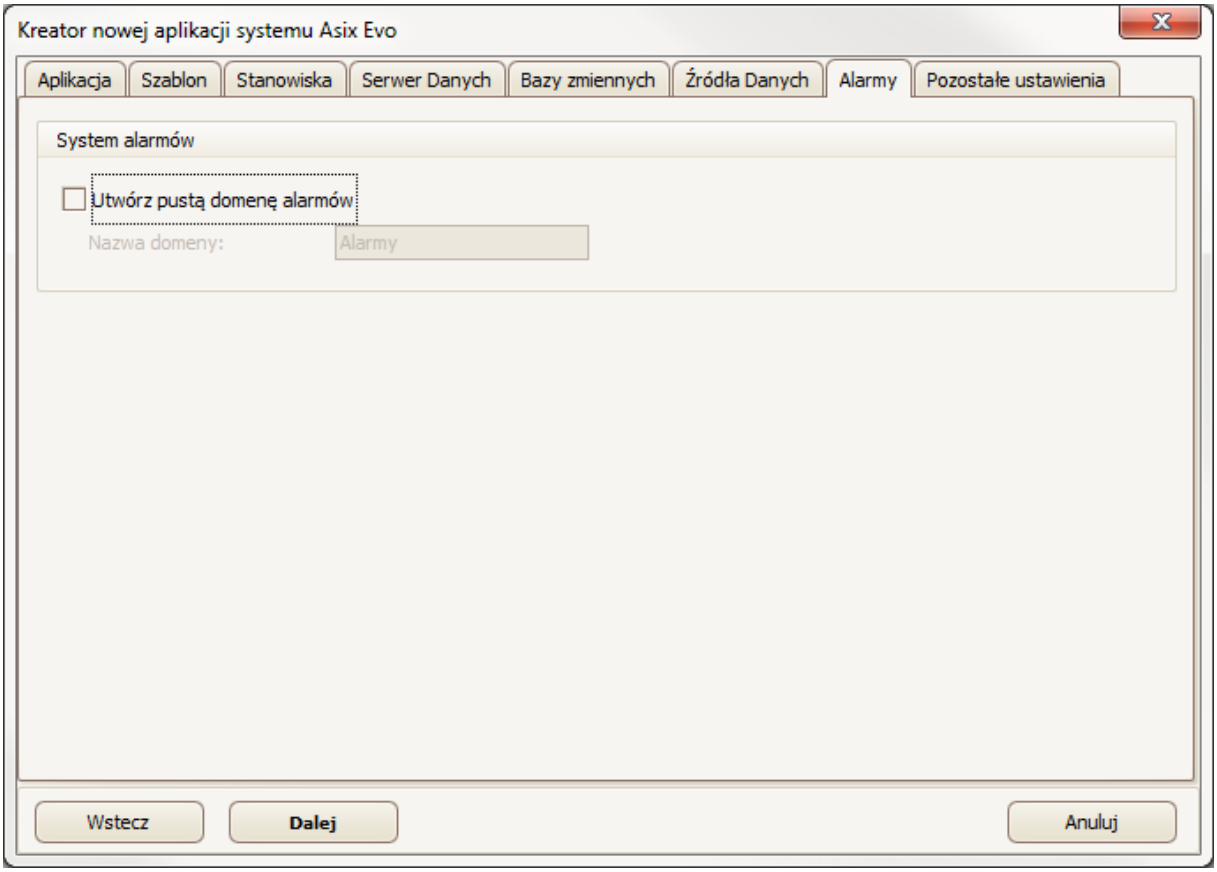


Rys. Kreator aplikacji - zakładka 'Źródła Danych'.

Pokazane ustawienia z reguły są optymalne. Kreator odczyta zawartość wybranych wcześniej baz definicji zmiennych i na jej podstawie utworzy wszystkie wymagane kanały i archiwa aplikacji. Na stanowiskach serwerowych (z fizycznym dostępem do sterowników i lokalną archiwizacją) włączone zostaną drajwery i archiwa typy Asix. Stanowiska terminalowe będą miały konfigurację ustawioną na pozyskiwanie danych bieżących i archiwalnych z serwerów.

3.1.7 Parametryzacja systemu alarmów

W zakładce *Alarmy* można ustawić nazwę domeny alarmowej. Domena alarmowa jednoznacznie identyfikuje zbiór alarmów, czyli archiwum alarmów, definicje alarmów i strategie wykrywania. Jest to odpowiednik nazwy zasobu alarmowego w starszych aplikacjach Asix.

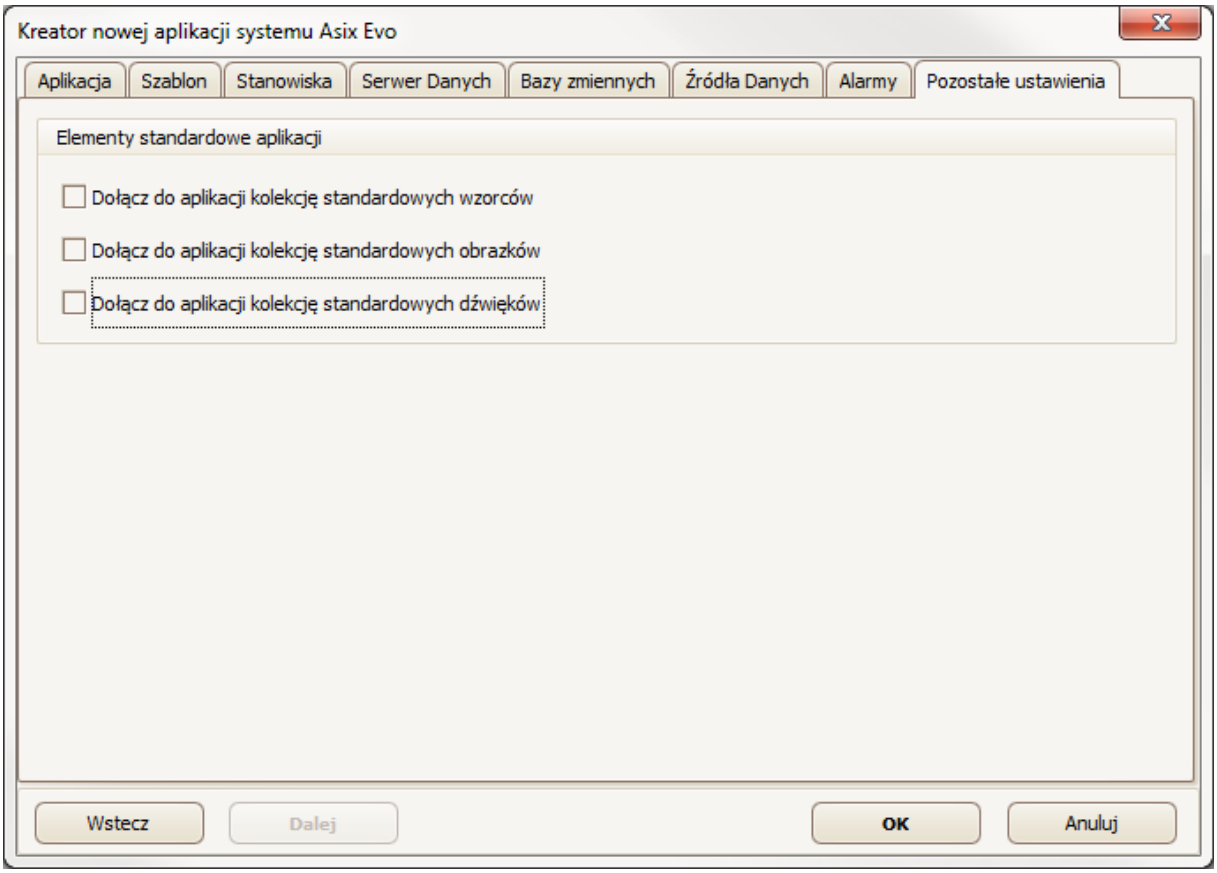


Rys. Kreator aplikacji - zakładka 'Alarmy'.

W odróżnieniu do starszych aplikacji systemu Asix, w aplikacjach typu Asix.Evo stanowisko może pracować jednocześnie w kilku domenach alarmowych. Dodanie kolejnych domen oraz ustawienie pozostałych parametrów systemu alarmów można wykonać później, w panelu roboczym parametryzacji systemu alarmów.

W naszym przykładzie nie będziemy jednak teraz tworzyć domeny alarmowej. Pokażemy później, jak taką domenę skonfigurować ręcznie.

3.1.8 Inne ustawienia



Rys. Kreator aplikacji - zakładka 'Pozostałe ustawienia'.

Ostatnia zakładka pozwala na skopiowanie do katalogu aplikacji predefiniowanych wzorców obiektów, obrazków i dźwięków. Można to również wykonać później przy pomocy funkcji selektywnego importu z odpowiednich paneli roboczych.

Ponieważ nie będziemy stosować standardowych elementów, to wyłączamy wszystkie funkcje importu.

3.2 Przegląd konfiguracji

Aplikacja została wstępnie skonfigurowana. Przyjrzymy się teraz ustawieniom wykonanym przez kreator.

Wybieramy panel *Stanowiska* (jeżeli nie był otwarty to używamy komendy *Stanowiska* z menu *Widok*). W panelu pokazana jest aktualna hierarchia stanowisk: w obszarze *Serwery* umieszczone jest nasze stanowisko robocze oraz dodatkowa stacja operatorska *Stacja_OP_1*. Ustawienia parametrów aplikacji są hierarchiczne. Znaczy to, że ustawienia wykonane w obszarze *Serwery* obowiązują także na podrzędnych stanowiskach, o ile nie zostaną tam zmienione.

Wybieramy węzeł *Serwery* i przechodzimy do panelu *Eksplorator aplikacji* (jeżeli nie był otwarty to używamy komendy *Eksplorator aplikacji* z menu *Widok*).

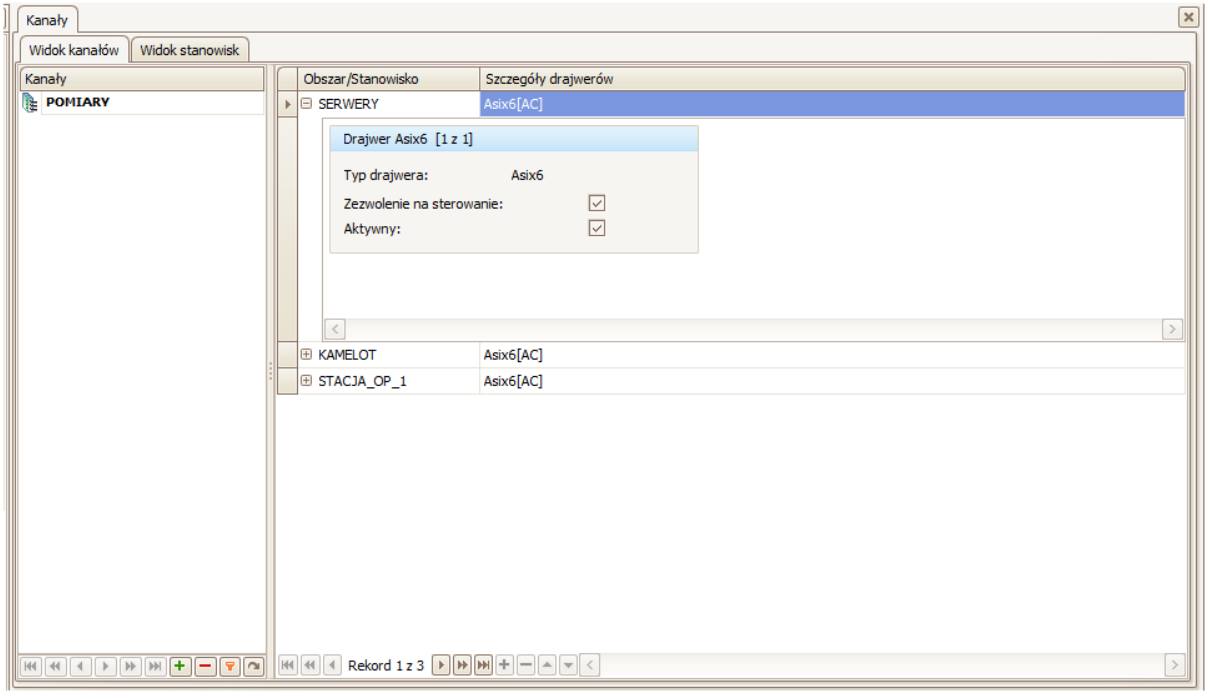
Klikamy podwójnie na węźle *Ustawienia globalne* i otwartym panelu roboczym wybieramy zakładkę *Komunikacja*. W wyświetlanej tabelce znajdują się dane opisujące role jaką pełnią w aplikacji poszczególne stanowiska. Szczególnie istotny jest adres stanowiska. Dla stanowisk serwerowych powinien być podany właściwy adres IP lub systemowa nazwa komputera. Jeżeli nazwa stanowiska i nazwa komputera są identyczne, to adres może pozostać pusty.

Wybieramy węzeł *Ustawienia stanowisk* i otwieramy zakładkę *Źródła danych*. Możemy zauważyć, że nasza baza zmiennych *AsixVarBase* została wybrana jako baza aktywna, ładowana przy starcie aplikacji. Przechodzimy do zakładki *Źródła danych*. W ramce *Ustawienia źródła danych procesowych* wybrany jest tryb pracy z wykorzystaniem serwera danych – typowy tryb pracy w przypadku odczytu danych bezpośrednio ze sterowników. Przy pomocy przycisku  można ponownie uruchomić program *Architekt* w celu zmiany konfiguracji kanałów komunikacyjnych i archiwów lub modyfikacji zawartości bazy definicji zmiennych.

Od momentu zakończenia pracy kreatora serwer danych powinien już pracować. Ikonę serwera można znaleźć w systemowym pasku zadań.

Rozwijamy węzeł *Bazy definicji zmiennych* w panelu eksploratora aplikacji. Otwieramy panel przeglądu zmiennych klikając na węźle bazy *AsixVarbase*. W otwartym panelu powinniśmy zobaczyć definicje zmiennych wprowadzonych wcześniej do bazy w programie *Architekt*. W przypadku baz typu XML możliwa byłaby bezpośrednia edycja zmiennych. Dla baz MDB można tylko przeglądać bazę. Bazy MDB są tworzone wyłącznie za pośrednictwem programu *Architekt*.

Korzystając z węzła *Kanały* panelu eksploratora otwieramy panel roboczy konfiguracji kanałów.

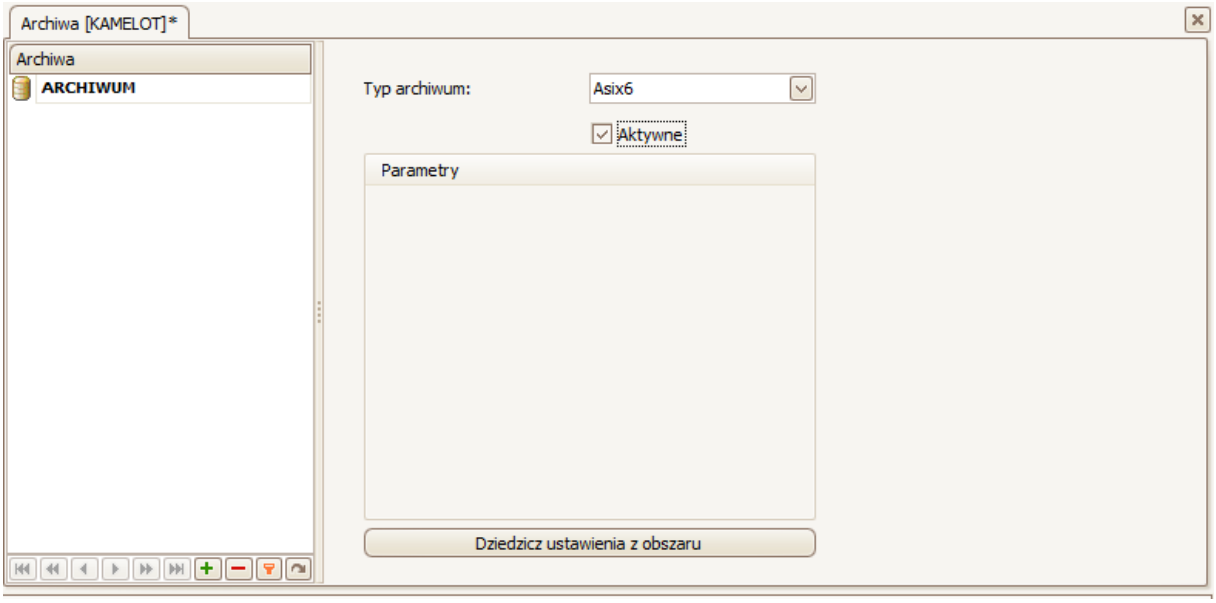


Rys. Panel roboczy konfiguracji kanałów.

Na liście kanałów znajduje się kanał *Pomiary* użyty wcześniej przy definicji zmiennych w programie Architekt. Po prawej stronie panelu pokazany jest sposób zdefiniowania wybranego kanału w poszczególnych obszarach i stanowiskach aplikacji.

Z każdym kanałem związany jest drajwer odpowiedzialny za pozyskiwanie wartości zmiennych. W przypadku korzystania z kanałów serwera danych, zawsze należy korzystać z kanału typu *Asix6*. Rzeczywisty typ drajwera komunikacyjnego wynika z definicji zapisanej w konfiguracji serwera danych.

Korzystając z węzła Archiwa panelu eksploratora otwieramy panel roboczy konfiguracji archiwów.



Rys. Panel roboczy konfiguracji archiwów.

Podobnie jak w przypadku kanałów, dla archiwów tworzonych przez moduł *Aspad* wbudowany w serwer danych, jako typ archiwum zawsze należy wybierać *Asix*.

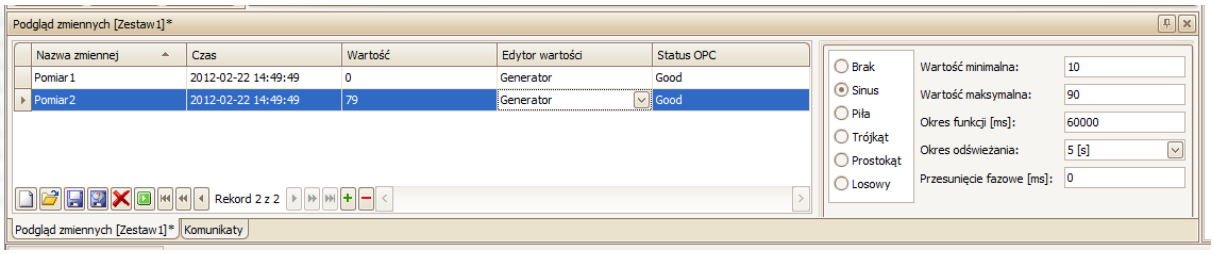
3.3 Test komunikacji - panel podglądu zmiennych

Zanim przejdziemy do budowy części wizualnej aplikacji przyjrzymy się jeszcze panelowi podglądu zmiennych. Panel podglądu powinien być już otwarty. W razie konieczności można go otworzyć komendą *Podgląd zmiennych* z menu *Widok.*

Panel poglądu zmiennych to narzędzie wspierające proces tworzenia aplikacji. Jego dwie zasadnicze funkcje to:

- Test komunikacji ze sterownikami oraz kontrola poprawności adresacji zmiennych w sytuacji, gdy część wizualna aplikacji nie jest jeszcze wykonana.
- Symulacja danych w przypadku, gdy nie ma możliwości podłączenia do sterownika, lub oprogramowanie sterownika nie jest jeszcze gotowe.

W naszej aplikacji panel podglądu będzie służył do symulacji danych.



Rys. Panel podglądu zmiennych.

Dodajemy do panelu wyświetlanie obu naszych zmiennych procesowych. Natychmiast po dodaniu aktualna wartość zmiennych zacznie być monitorowana. W przypadku użycia drajwera *None* wartości będą zerowe. W przypadku korzystania z innego drajwera powinny to być wartości odczytane ze sterownika.

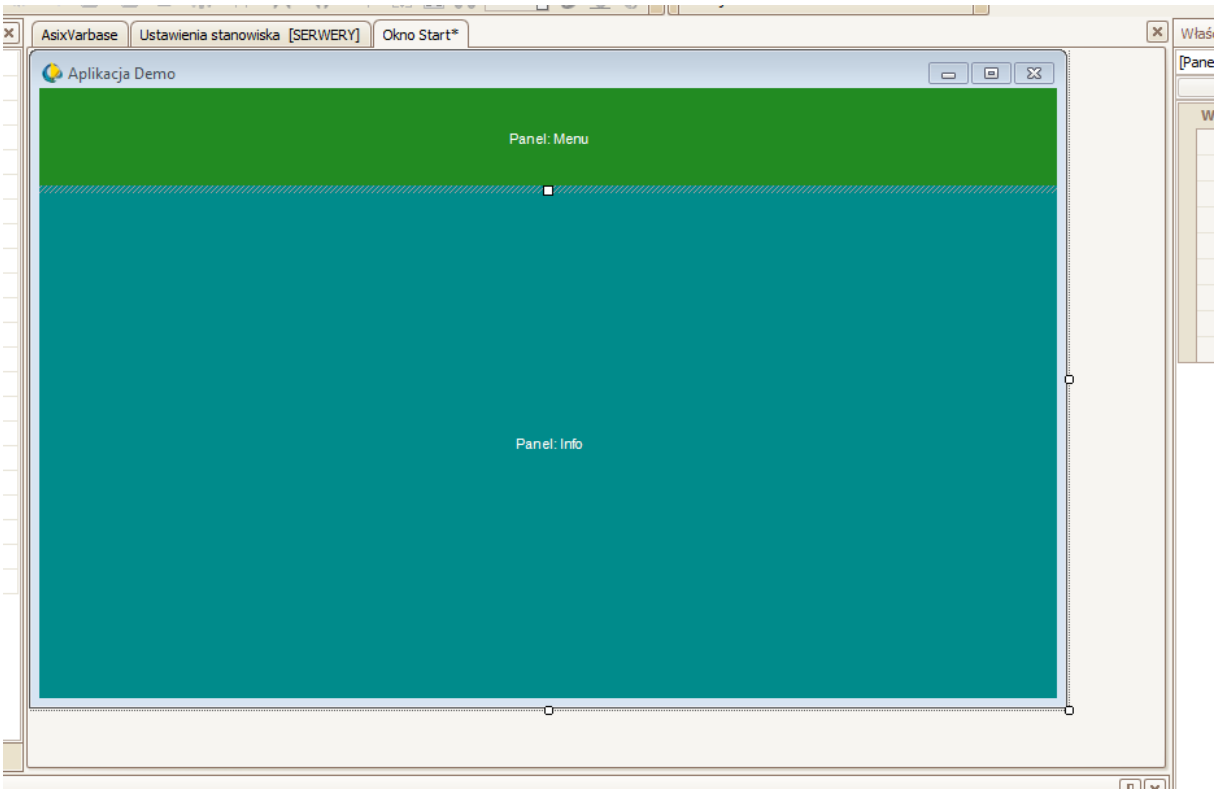
W kolumnie *Edytor wartości* można wybrać tryb wpisywania nowych wartości,. Ponieważ chcemy uzyskać automatyczną symulacje danych, to wybieramy tryb *Generator*. Następnie w prawej części panelu ustawiamy parametry symulacji w taki sposób, aby uzyskać przebieg zmienny w zakresie mniej więcej od wartości 0 do 100. Po zdefiniowaniu parametrów, proces symulacji należy jeszcze zainicjować przyciskiem .

3.4 Budowa okien i diagramów aplikacji

Po utworzeniu aplikacji, zdefiniowaniu zmiennych procesowych, kanałów komunikacyjnych i archiwów można przejść do fazy budowy części wizualnej aplikacji.

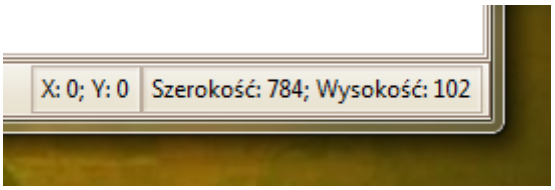
3.4.1 Utworzenie okna

Poprzez podwójne kliknięcie na elemencie *Okna* panelu eksploratora tworzymy okno główne aplikacji. W panelu właściwości zmieniamy nazwę okna na *Start*, wpisujemy tytuł okna, a przy pomocy myszki zmieniamy rozmiar okna. Nowy rozmiar można też określić we właściwościach *Szerokość* i *Wysokość*. Następnie, używając menu kontekstowego okna dodajemy dwa panele, w których będą wyświetlane diagramy synoptyczne. Pierwszy panel nazywamy *Menu*, a tryb dokowania ustawiamy na *Do góry*. Drugi panel nazywamy *Info*, a tryb dokowania ustawiamy na *Wypełnienie*. Efekt powyższych działań prezentuje ilustracja.



Rys. Edycja okna aplikacji - deklaracji paneli.

Należy zwrócić uwagę na sposób dopasowywania się paneli do okna w przypadku zmian jego rozmiaru.



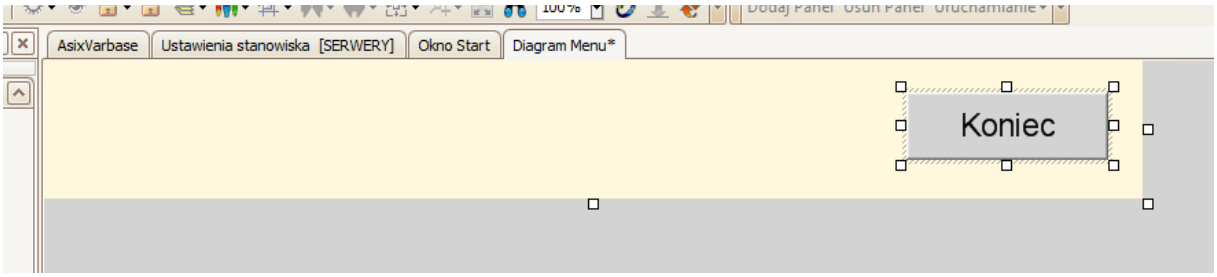
Rys. Parametry rozmiaru okna aplikacji.

W dolnym narożniku okna programu można zobaczyć jaki jest rzeczywisty rozmiar panelu. Informacja ta może przydać się w trakcie tworzenia diagramów.

3.4.2 Utworzenie diagramów

Kolejny krok to utworzenie dwóch diagramów. Można by to zrobić klikając podwójnie na elemencie *Diagramy* panelu eksploratora aplikacji. Chcemy jednak żeby rozmiary diagramów były zgodne z rozmiarami paneli. Ponieważ nie chcemy ustawiać rozmiarów ręcznie, to zastosujemy komendę *Utwórz dopasowany diagram* z menu kontekstowego panelu okna. Wykonujemy komendę dwa razy dla obu paneli i tworzymy diagramy: *Menu i Pomiar*. Zgodne rozmiary paneli i diagramów pozwolą na edycję diagramów w takim rozmiarze w jakim będą one wyświetlane. Jeżeli edycyjne rozmiary diagramów będą niezgodne z rozmiarami paneli, to sposób wyświetlenia zależy od właściwości *Stałe rozmiary* diagramu. Wybranie wartości *Nie* (np. przez podwójne kliknięcie w polu wartości) spowoduje, że diagramy zostaną odpowiednio przeskalowane. Można też wybrać kolor tła diagramu, ewentualnie obrazek (tapetę) wyświetlaną w tle.

Rozpoczynamy od edycji diagramu *Menu*. W panelu *Przybornik* wybieramy obiekt *Przycisk*, a następnie na diagramie zaznaczamy miejsce, w którym chcemy umieścić przycisk. Celem jest uzyskanie diagramu pokazanego na ilustracji.

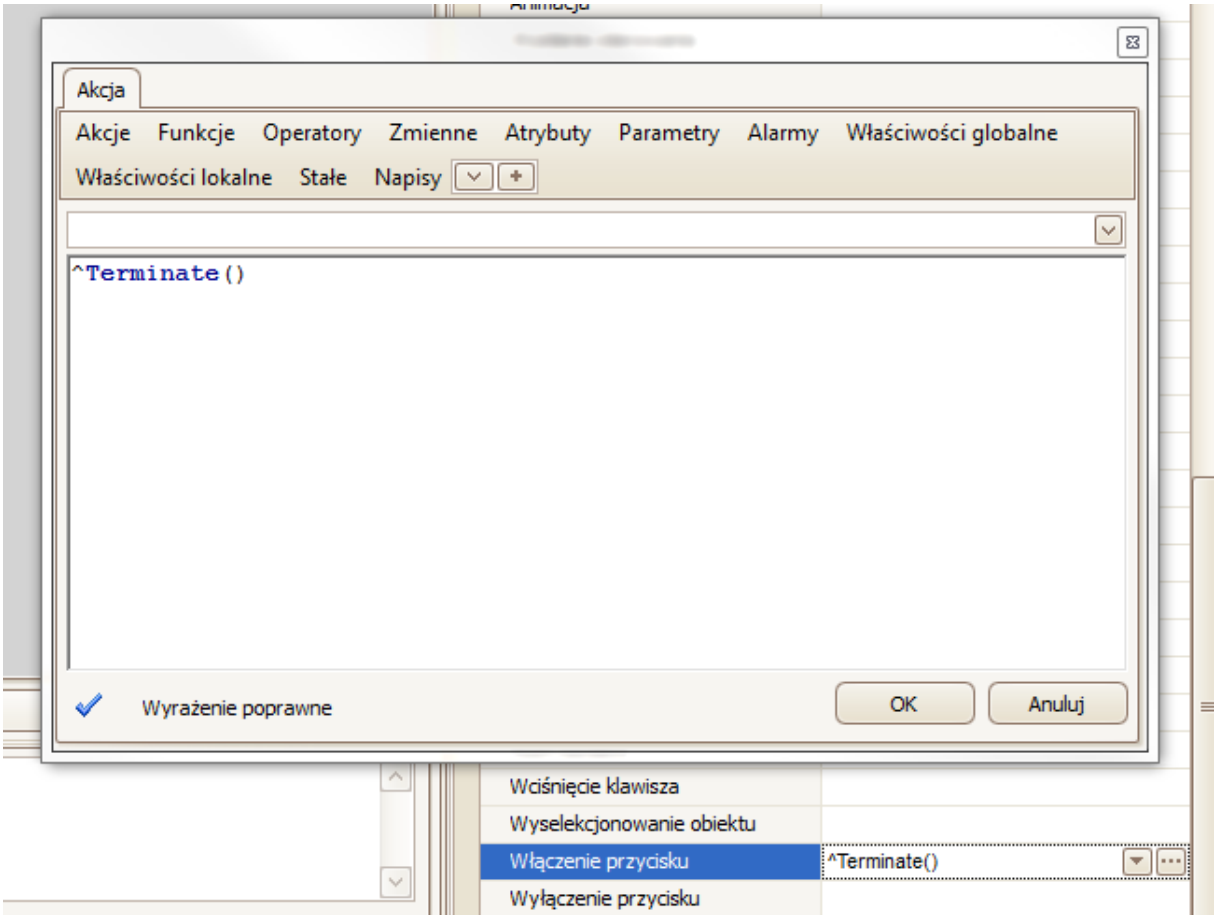


Rys. Edycja diagramu - dodanie obiektu Przycisk.

Można dostosować wygląd obiektu do własnych wymagań zmieniając właściwości odpowiedzialne za typ, kształt i kolorystykę. Obiekt z ilustracji ma wybrany typ *Standardowy przycisk Windows* oraz margines tekstu ustawiony na 20%. Dodatkowo warto zmienić właściwość *Typ kursora* na *Hand*.

Spowoduje to, że w trakcie wykonania aplikacji kursor myszki będzie zmieniał kształt po najechaniu na przycisk. Tekst *Koniec* należy wpisać we właściwości *Tekst wyłączenia* w grupie właściwości stanowych obiektu.

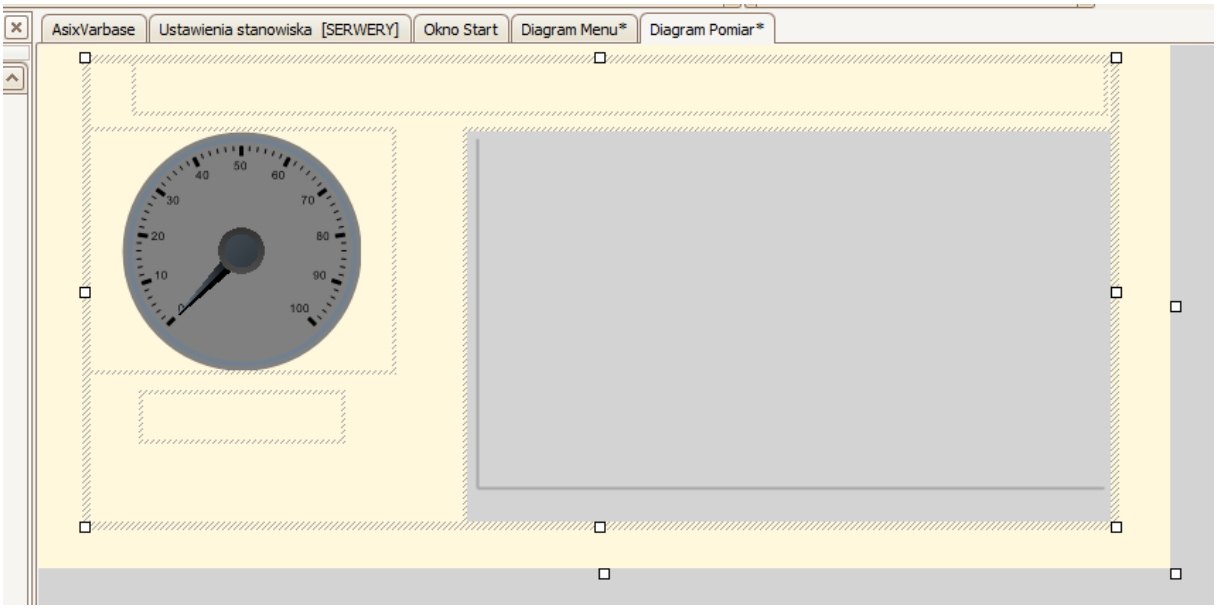
Zadaniem obiektu będzie zakończenie pracy aplikacji. W tym celu należy w obsłudze zdarzenia *Włączenie przycisku* wpisać wywołanie akcji *Terminate.*



Rys. Okno edycji właściwości przycisku - deklaracja akcji.

Po wejściu do okna edycji właściwości z menu *Akcje/Sterowanie aplikacją* wybieramy akcję *Terminate.* Spowoduje to wpisanie do pola edycyjnego treści akcji operatorskiej. Znak ^ jest przedrostkiem oznaczającym, że definicja właściwości jest treścią akcji operatorskiej.

Można teraz przejść do edycji diagramu *Pomiar*. Rozpoczynamy od umieszczenia na diagramie dwóch obiektów typu *Tekst*, jednego obiektu *Miernik* oraz jednego obiektu *Wykres*.



Rys. Edycja diagramu - dodanie obiektów: Miernik i Wykres.

Diagram powinien wyglądać w sposób zbliżony do powyższego. Obiekty *Tekst* na razie nie wyświetlają żadnych informacji.

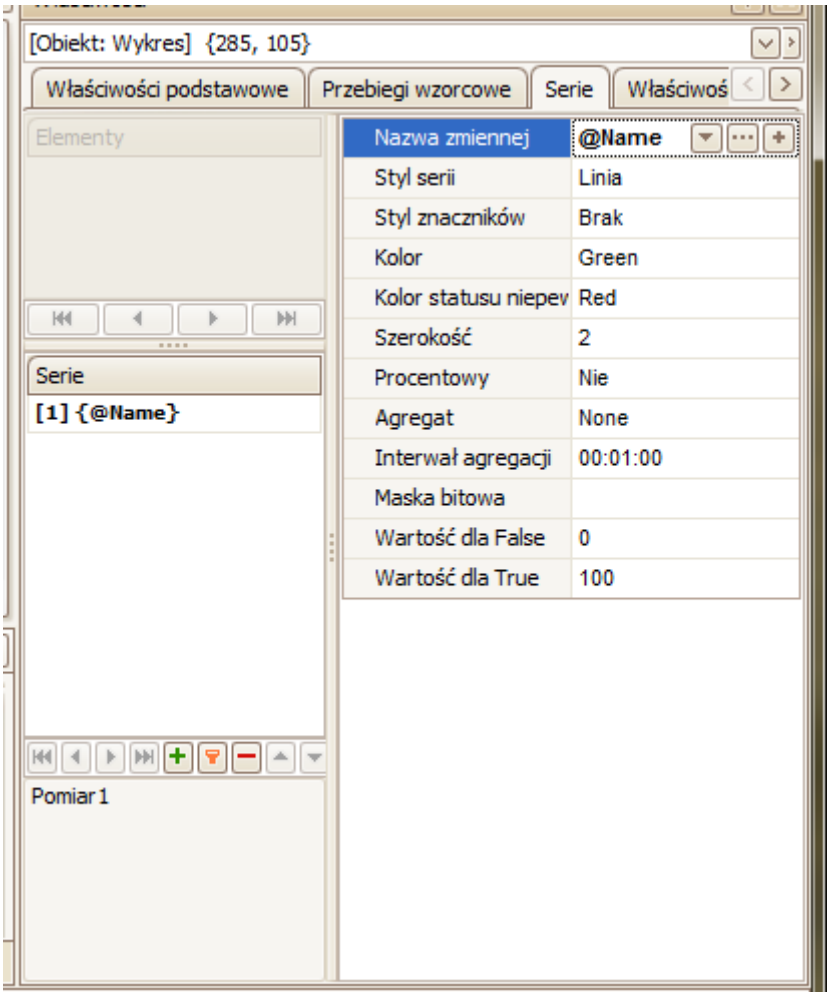
Obiekt *Tekst* umieszczony w górnej części diagramu będzie wyświetlał opis zmiennej pomiarowej pobierany z bazy zmiennych procesowych. W polu właściwości *Główna zmienna* używamy przycisku **+.** W oknie selekcji zmiennych wybieramy zmienną *Pomiar1*. Następnie w polu właściwości *Text* rozpoczynamy edycję wartości wprowadzając znak @. Znak @ oznacza odwołanie do atrybutu zmiennej procesowej. Wyświetlona zostanie lista wszystkich atrybutów, należy wybrać atrybut *Description*. W efekcie właściwość *Text* powinna przyjąć wartość *@Description*, co oznacza wyświetlenie zawartości atrybutu *Description (Opis zmiennej*) dla zmiennej głównej obiektu. Po wykonanych zmianach na diagramie powinien pojawić się opis zmiennej.

W obiekcie *Miernik* ustawiamy właściwość *Główna zmienna* na wartość *Pomiar1*, a w polu właściwości *Wartość* wprowadzamy znak *#*. Oznacza on odwołanie do wartości zmiennej głównej obiektu. Wskazówka miernika powinna się odchylić zgodnie z aktualną wartością zmiennej *Pomiar1,* odczytywanej ze sterownika lub wymuszanej poprzez panel *Podgląd zmiennej*.

W drugim obiekcie *Tekst* ustawiamy właściwość *Główna zmienna* na wartość *Pomiar1*, a w polu właściwości *Tekst #* oraz zmieniamy *Wyrównywanie w poziomie* na *Wyśrodkowanie*. Po zmianach obiekt pokaże aktualną wartość kontrolowanej zmiennej.

W obiekcie *Wykres* ustawiamy właściwość *Główna zmienna* na wartość *Pomiar1.* Obiekt ma wstępnie założoną jedną serię pomiarową.

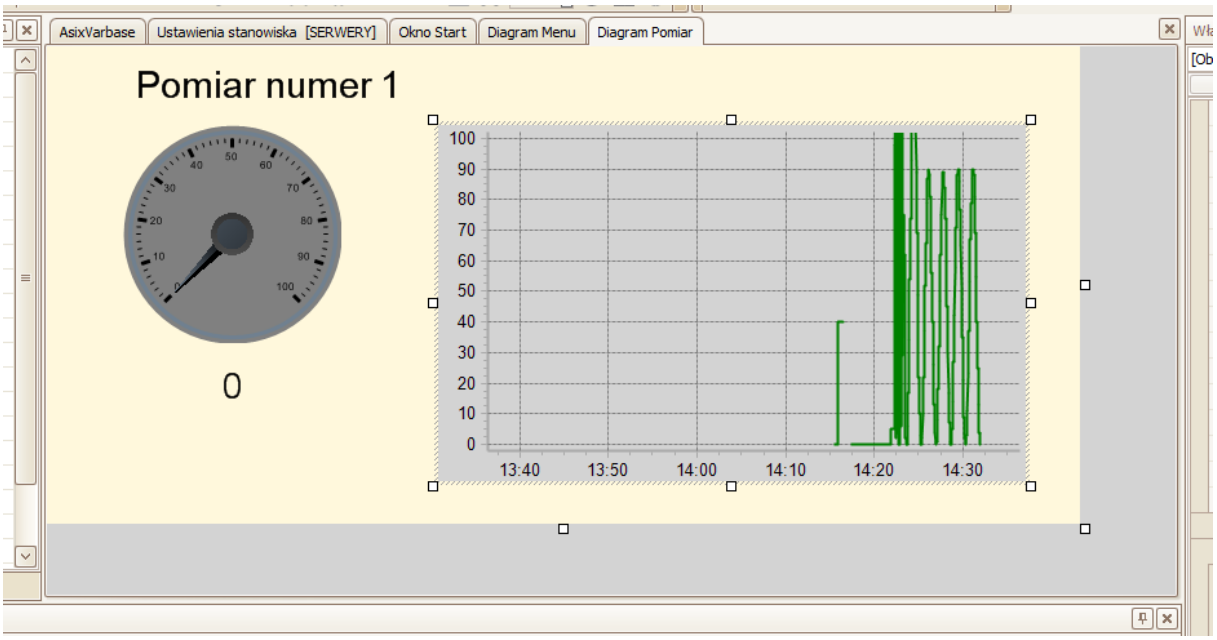
Poniższa ilustracja pokazuje panel właściwości, tym razem w trybie kompaktowym. Należy wybrać zakładkę *Serie*. Na liście serii będzie dostępna pojedyncza seria. Dodatkowe serie można byłoby dodać stosując przyciski paska narzędziowego położonego poniżej listy serii. W prawej części panelu we właściwości *Nazwa zmiennej* wprowadzamy wartość @Name.



Rys. Parametryzacja obiektu Wykres - definicja serii pomiarowej.

Oznacza to odwołanie do atrybutu *Name* zmiennej głównej, czyli w praktyce do nazwy zmiennej głównej. Alternatywny sposób parametryzacji to podanie nazwy zmiennej bezpośrednio we właściwościach serii, bez wprowadzania nazwy zmiennej głównej.

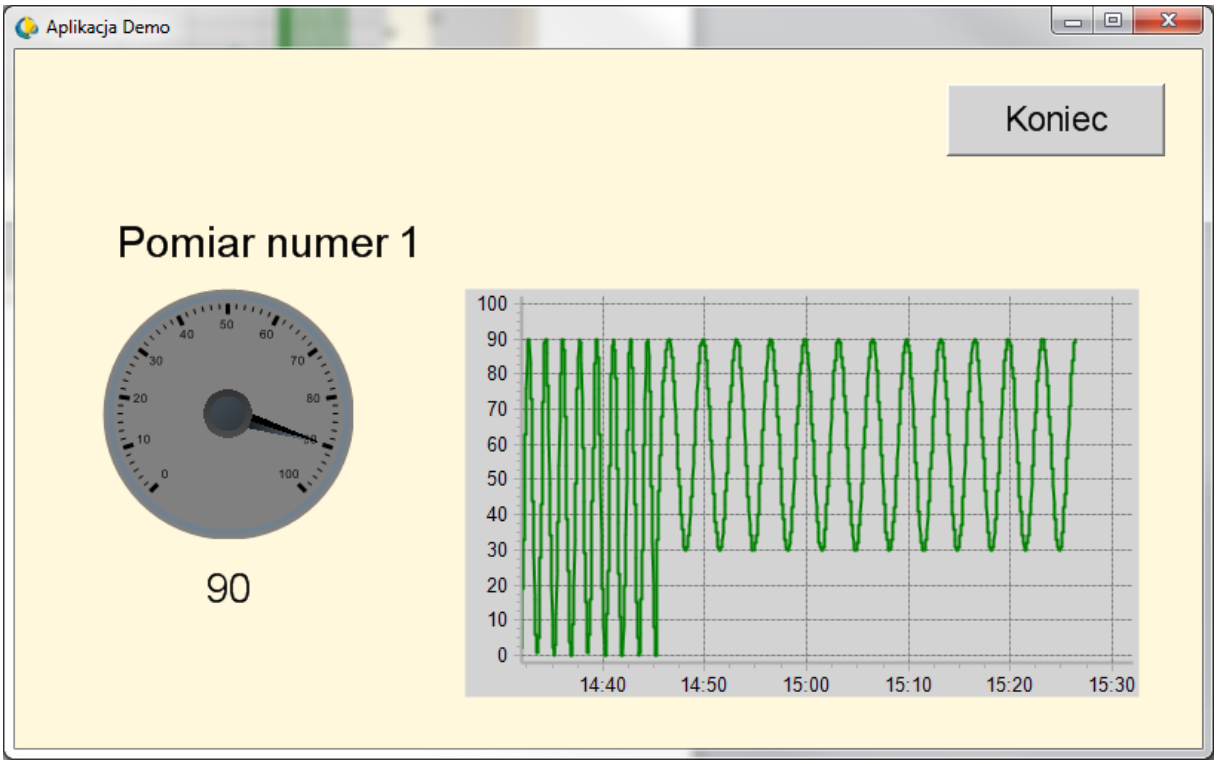
Po wykonaniu powyższych czynności uzyskujemy końcową postać diagramu pokazaną poniżej.



Rys. Końcowa postać edycji diagramu.

Pozostały jeszcze do wykonania ostatnie czynności konfiguracyjne. Wracamy do parametryzacji okna *Start*. W ustawieniach paneli *Menu* i *Info* należy ustawić właściwość *Domyślny diagram* odpowiednio na *Menu* i *Pomiar*. W panelu eksploratora wybieramy *Ustawienia stanowisk* i w zakładce *Opcje startowe* wybieramy okno *Start* jako okno początkowe aplikacji.

Możemy teraz wykonać test pracy aplikacji. Pierwszy wariant to utworzenie przy pomocy kreatora skrótu uruchamiającego aplikację. Prostszy sposób (ale nadający się wyłącznie do uproszczonego testu) to wybranie panelu edycyjnego okna *Start* i z paska narzędziowego edycji okien wybrać komendę *Uruchamianie/Uruchom normalnie*. W efekcie uruchomienia okna powinniśmy uzyskać efekt zbliżony do pokazanego na poniższej ilustracji.

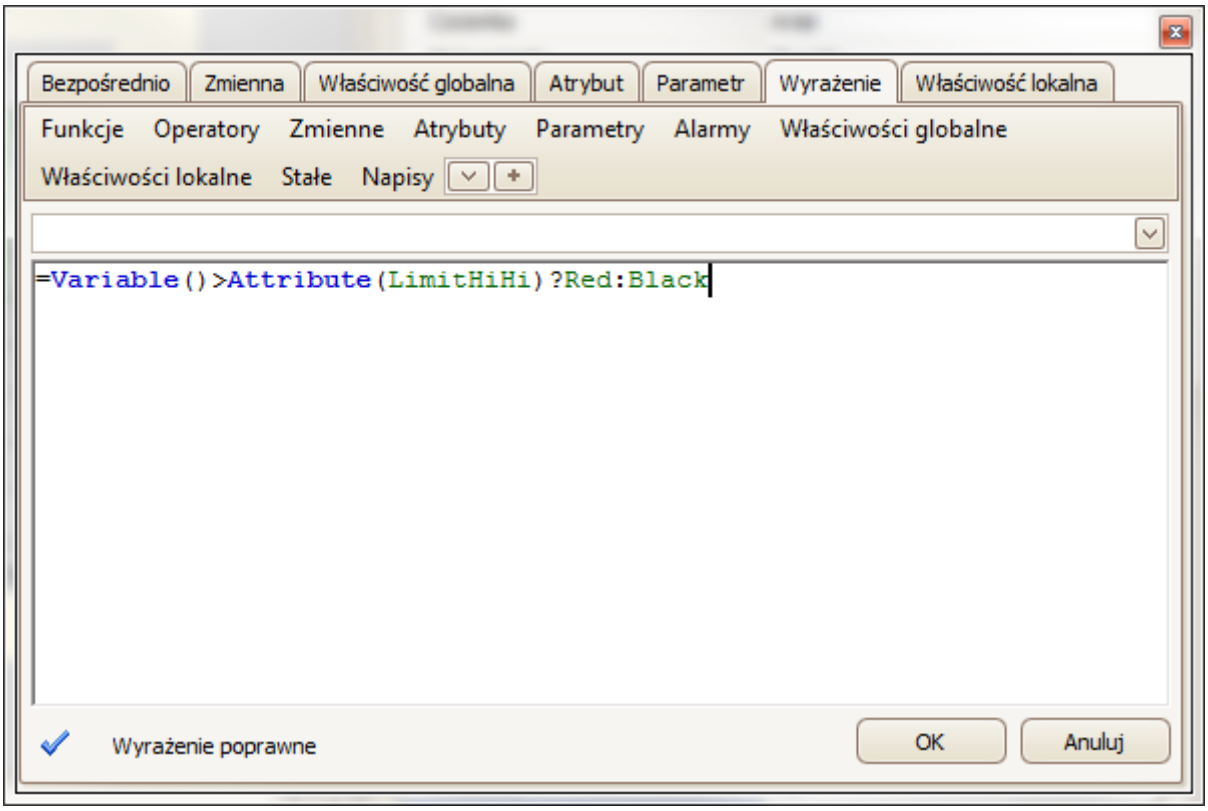


Rys. Podgląd diagramu.

3.4.3 Sygnalizacja przekroczeń

Kolejnym krokiem w rozwoju naszej aplikacji będzie dodanie sygnalizacji o przekroczeniu przez pomiar wartości górnego ograniczenia. Wartość ograniczenia wprowadziliśmy wcześniej w bazie zmiennych w atrybucie *Maksimum krytyczne*.

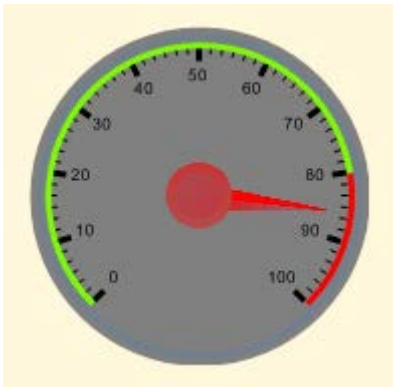
Obiekt *Miernik* posiada wbudowany mechanizm oznaczania na skali zakresów poprawności pomiarów. W pierwszym kroku włączamy sygnalizację ustawiając właściwość *Pokaż limity* na *Tak*. Pozostaje ustawić zakresy. Ponieważ chcemy tyko pokazywać przekroczenie maksimum krytycznego, to właściwości *Wartość LL* i *Wartość L* ustawiamy na 0, a *Wartość H* i *Wartość HH* na *@LimitHiHi.* W efekcie, zakres od 0 do wartości atrybutu *LimitHiHi* (maksimum krytyczne) będzie traktowany jako zakres wartości normalnych – na skali będzie rysowane oznaczenie w kolorze właściwości *Kolor wartości normalnej*. Zakres powyżej maksimum krytycznego kontrolowanej zmiennej będzie oznaczony kolorem podanym we właściwości *Kolor HH*. Oprócz oznaczenia zakresów na skali miernika, będziemy jeszcze zmieniać kolor wskazówki w zależności od aktualnej wartości pomiaru. W tym celu we właściwości *Kolor wskazówki* należy wprowadzić wyrażenie pokazane na poniższej ilustracji.



Rys. Deklaracja wyrażenia zmieniającego kolor wskazówki Miernika w zależności od wartości zmiennej procesowej.

Wyrażenie ma postać wyrażenia arytmetycznego ***warunek?wartość_dla_true:wartość_dla_false***. Interpretacja pokazanego wyrażenie jest następująca: jeżeli wartość zmiennej głównej *Variable()* jest większa od wartości limitu krytycznego zmiennej głównej (*Attribute(LimitHiHi)*), to użyj kolor *Red*, w przeciwnym wypadku użyj kolor *Black*.

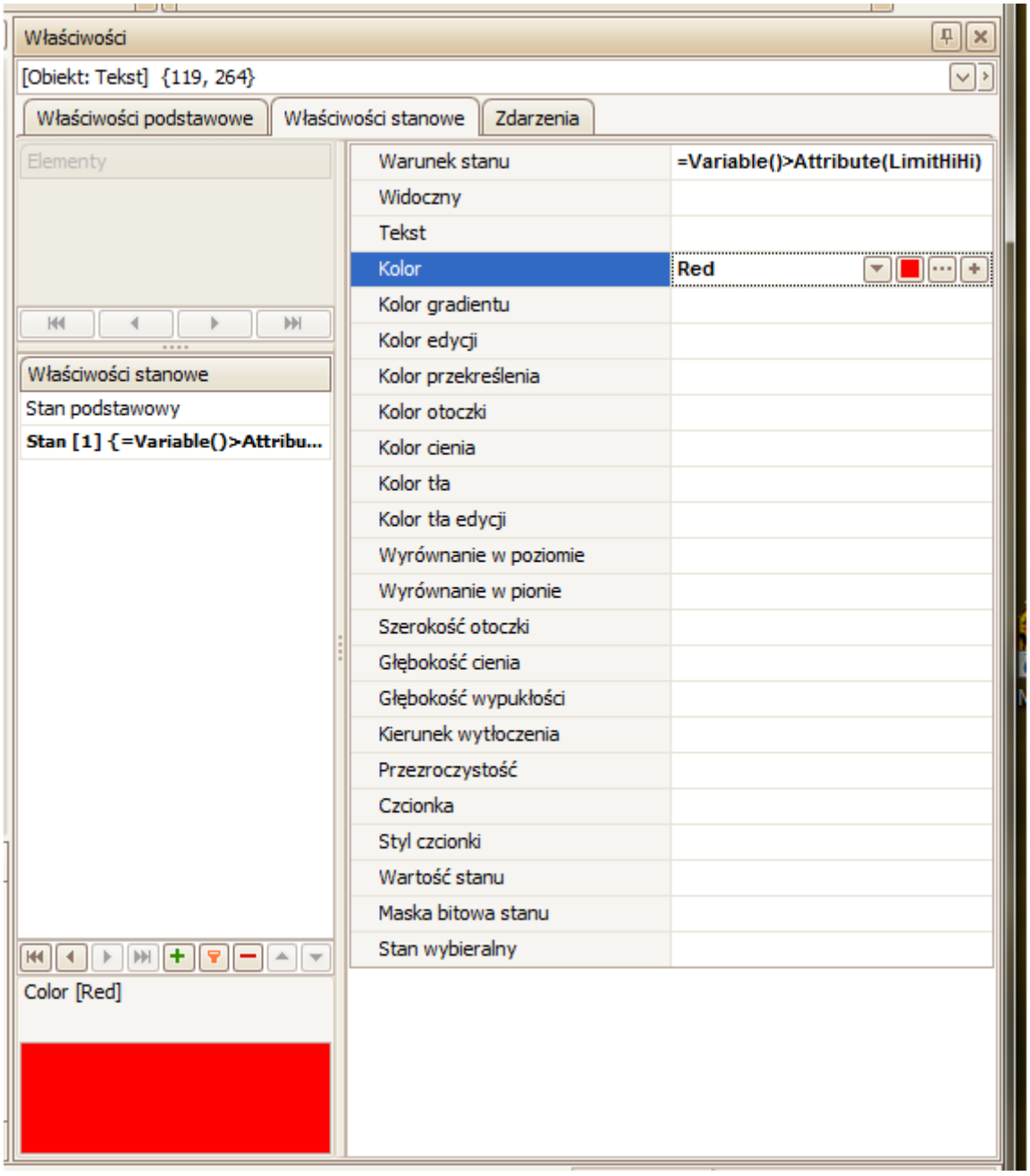
W końcowym efekcie powinniśmy uzyskać postać miernika pokazaną poniżej.



Rys. Obiekt Miernik z zadeklarowaną zmianą kolorow w zależności od wartości zmiennej procesowej.

Przejdziemy teraz do obiektu *Liczba* pokazującego wartość zmiennej. Oprócz sygnalizacji przekroczenia, będziemy również chcieli w specjalny sposób obsłużyć błędy odczytu zmiennej. Za kolor wyświetlanego teksu odpowiada właściwość *Kolor* znajdująca się w grupie właściwości stanowych. Moglibyśmy użyć wyrażenia podobnego do tego zastosowanego w obiekcie miernika, ale w tym przypadku użyjemy mechanizmu właściwości stanowych.

W panelu właściwości wybieramy zakładkę *Właściwości stanowe*, a następnie używamy przycisku  paska narzędziowego znajdującego się poniżej listy stanów. Początkowo na liście znajduje się tylko zawsze istniejący *Stan podstawowy*. Użycie przycisku spowoduje dodanie nowego stanu, na liście pojawi się dodatkowy stan jako *Stan [1]{}*. W grupie stanowej znajduje się identyczny zestaw właściwości jak w tzw. grupie podstawowej oraz jedna dodatkowa właściwość *Warunek stanu*.



Rys. Deklaracja zmiany koloru obiektu Liczba w zależności od wartości zmiennej procesowej.

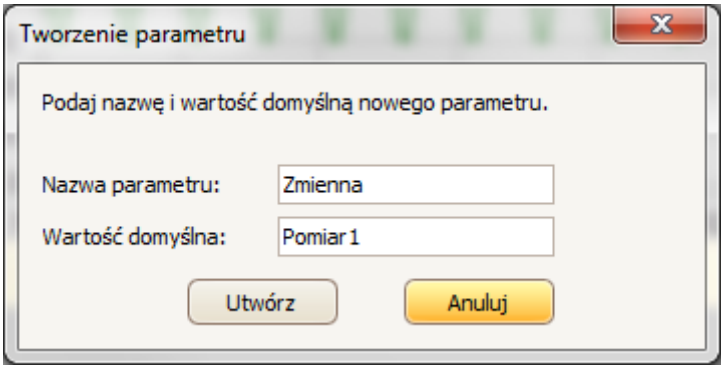
Jako warunek stanu wprowadzamy wyrażenie *=Variable()>Attribute(LimitHiHi)*. Właściwość *Kolor* ustawiamy na *Red*. Warunek stanu przyjmuje wartość *true* wtedy, gdy wartość zmiennej głównej przekroczy maksimum krytyczne. W tym momencie wszystkie ustawienia zdefiniowane jawnie w grupie (w naszym przypadku właściwość *Kolor*) będą miały pierwszeństwo przed wszystkimi ustawieniami we wcześniejszych grupach – po przekroczenia limitu wartość zmiennej będzie wyświetlana w kolorze czerwonym.

Dodajemy do obiektu kolejny stan – warunek stanu ustawiamy na *=VarIsNotGood()*, właściwość *Tekst* na *Brak danej*, a właściwość *Kolor* na *Yellow*. W efekcie, gdy status głównej zmiennej będzie niepoprawny, to zamiast wartości zmiennej będzie wyświetlony tekst *Brak danej* w kolorze żółtym. Należy zwrócić uwagę, że w momencie, gdy oba dodatkowe stany będą aktywne, to pierwszeństwo będzie miała definicja właściwości *kolor* w ostatniej grupie stanowej.

3.4.4 Dodanie parametrów diagramu

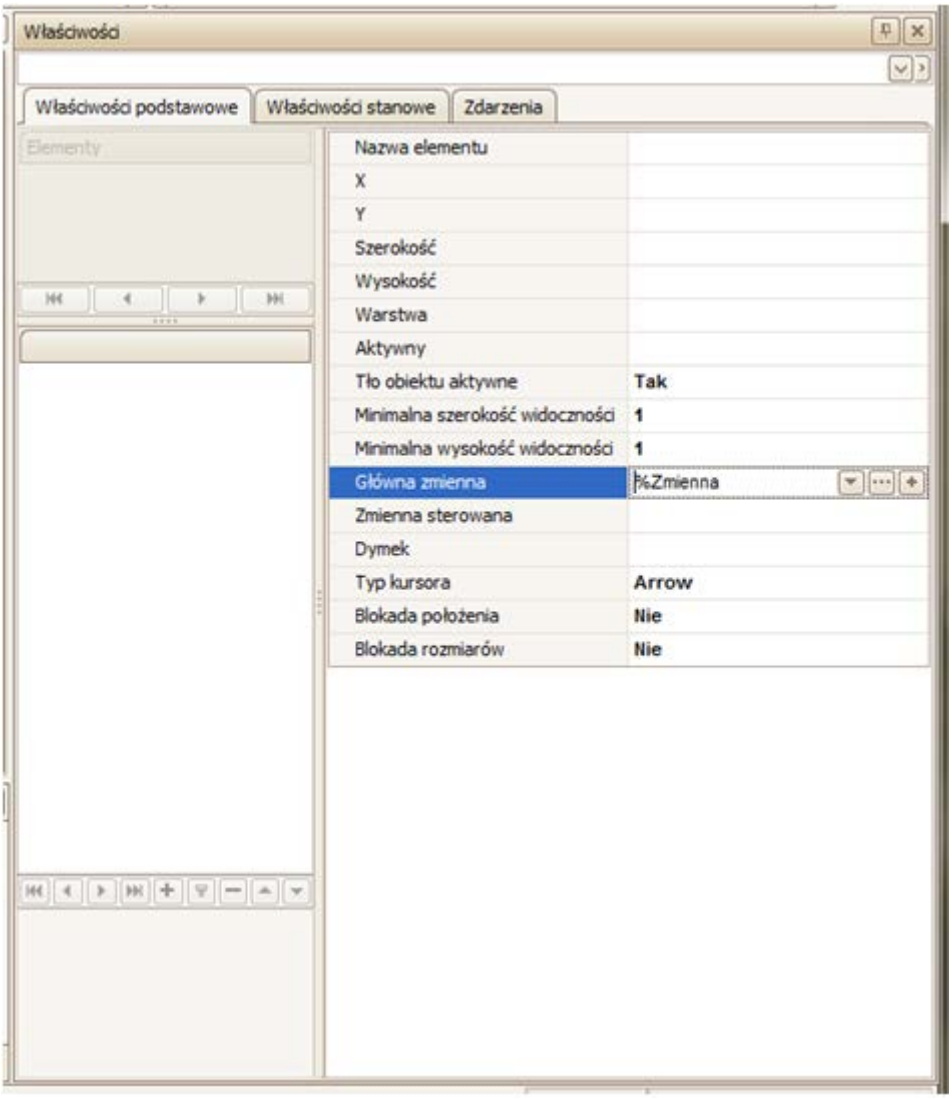
Uzyskaliśmy teraz końcową postać wizualną diagramu *Pomiar*. Naszym celem jest jednak pokazanie dwóch pomiarów. Moglibyśmy oczywiście zrobić kopię diagramu *Pomiar* i zamienić wszystkie odwołania do zmiennej *Pomiar1* na zmienną *Pomiar2*. Ponieważ oba diagramy mają wyglądać tak samo, lepszym rozwiązaniem jest użycie mechanizmu diagramów parametryzowanych. Będziemy posiadać tylko jeden diagram służący do pokazywania dowolnej liczby zmiennych. W przypadku późniejszych ewentualnych zmian będziemy musieli modyfikować tylko ten jeden diagram.

Rozpoczynamy od dodania parametru. Klikamy w obszarze diagramu poza granicami obiektów. W panelu właściwości zostaną pokazane parametry diagramu. Wybieramy linię *Parametry*, a następnie przyciskiem **+** dodajemy parametr i definiujemy go zgodnie z poniższą ilustracją.



Rys. Deklaracja parametru dla diagramu.

W kolejnym kroku zaznaczamy wszystkie obiekty diagramu. W panelu właściwości zostaną pokazane tylko właściwości wspólne dla wszystkich czterech obiektów diagramu.



Rys. Właściwości wspólne obiektów.

Zmieniamy właściwość *Główna zmienna* na wartość *%Zmienna*. Prefiks % oznacza odwołanie do parametru diagramu. Ponieważ parametryzowaliśmy nasze obiekty w taki sposób, żeby nazwa zmiennej pojawiała się tylko we właściwości *Główna zmienna,* to po zmianie definicji nasze obiekty są całkowicie przełączone na pokazywanie zmiennej podanej w parametrze diagramu. Nie powinniśmy zauważyć żadnej zmiany w wyglądzie i działaniu diagramu, wartość domyślna parametru jest równa *Pomiar1*, czyli w dalszym ciągu pokazujemy te same dane.

Wracamy do edycji diagramu *Menu*. Dodamy dwa przyciski służące do przełączania diagramów. Ponieważ chcemy zachować jednorodną stylistykę, to zaznaczamy przycisk *Koniec* i dwukrotnie przyciskamy klawisz ctrl-d tworząc dwie kopie obiektu. Modyfikujemy właściwość *Tekst włączenia* obu nowych obiektów. Zmieniamy również akcje podane w obsłudze zdarzenia *Włączenie przycisku*. Akcja dla pierwszego przycisku powinna wyglądać tak:

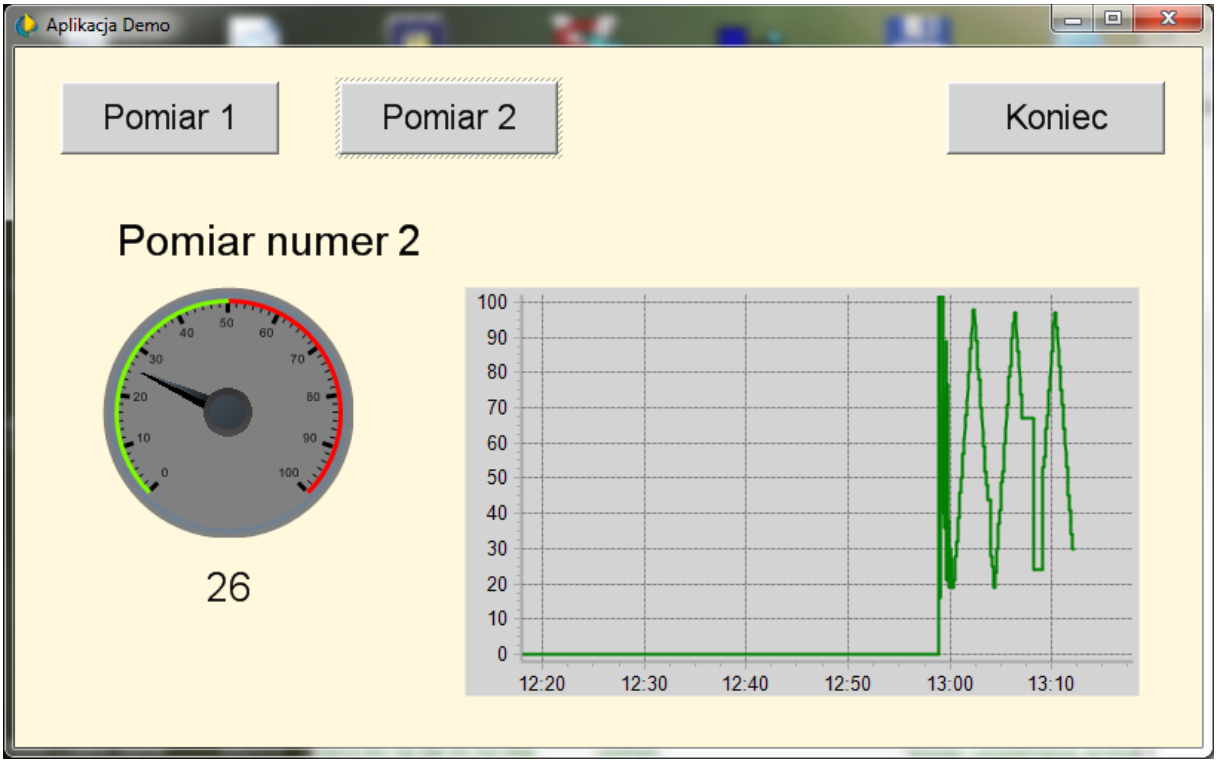
```
^OpenWindow(null,Info,Pomiar,"Zmienna=Pomiar1")
```

Interpretacja akcji jest następująca: w oknie bieżącym (*null)* w panelu *Info* otwórz diagram *Pomiar* ustawiając wartość parametru *Zmienna* na *Pomiar1*.

W podobny sposób analogiczny należy zmodyfikować akcję dla przycisku drugiego.

Aplikacja jest gotowa, można ją uruchomić w trybie pełnym lub testowym. Efekt powinien wyglądać w sposób zbliżony do poniższego.

Należy zwrócić uwagę na to, jak obiekty dostosowują się do parametrów pobieranych z bazy definicji zmiennych (opis i zakres zmiennej).

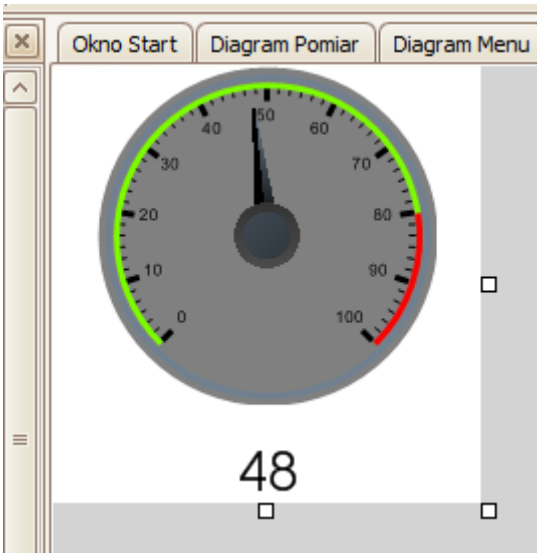


Rys. Widok diagramu z możliwością przełączania się pomiędzy wariantami dla dwóch zmiennych.

3.4.5 Wprowadzenie wzorców

Wykonamy jeszcze jedną zmianę w aplikacji. Chcemy pokazać jeszcze diagram zbiorczy ze wszystkimi pomiarami. Na tym diagramie będziemy wyświetlać tylko miernik i liczbową wartość pomiaru. Ponieważ każdy pomiar ma wyglądać identycznie, a nie chcemy wielokrotnie kopiować obiektów, to posłużymy się mechanizmem wzorców.

Przechodzimy do edycji diagramu *Pomiar*, zaznaczamy obiekty miernika i wartości pomiaru, a następnie przyciskamy klawisz *ctrl-p*. Spowoduje to utworzenia wzorca złożonego z wybranych obiektów.



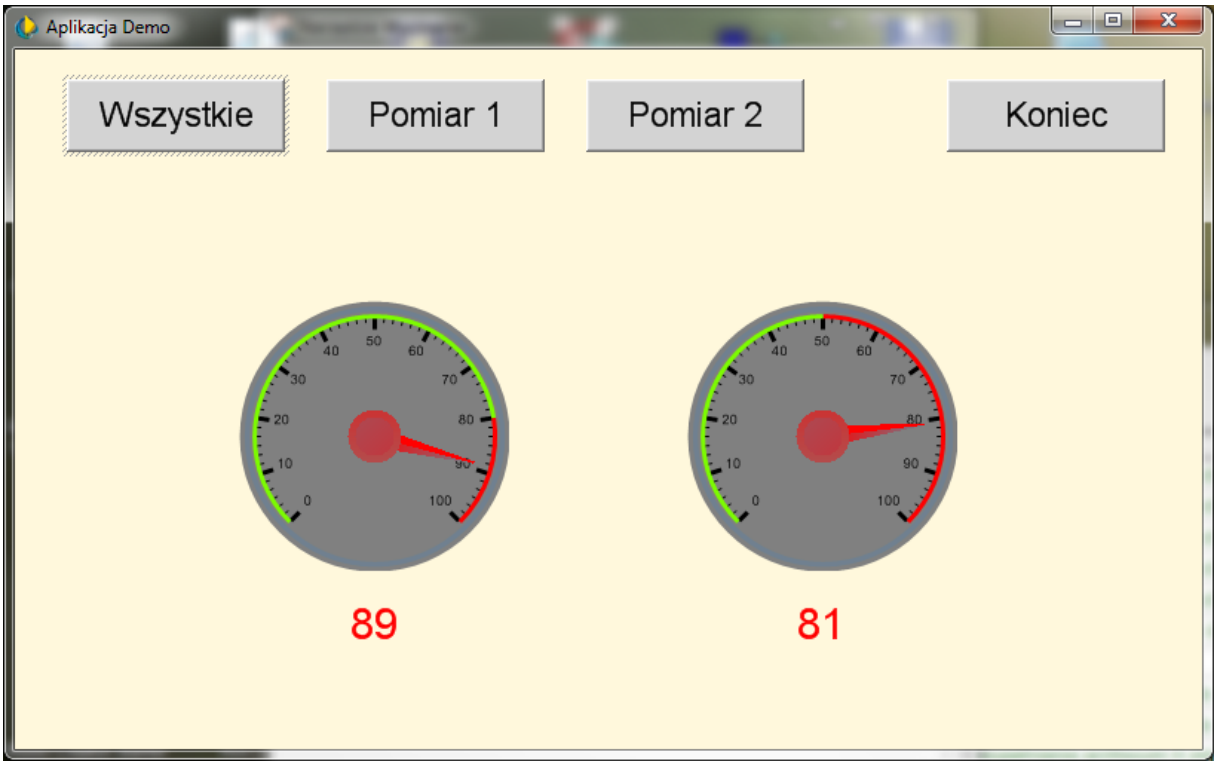
Rys. Wzorzec kompozycji obiektów: Miernik i Liczba.

W panelu właściwości zmieniamy nazwę wzorca na *WzorPomiar*. Można zauważyć, że wzorzec ma już automatycznie dodany parametr. Wynika to z faktu, że diagram z którego pochodziły obiekty taki parametr też posiadał. Chociaż nie jest to konieczne, usuniemy ten parametr (przyciskiem -), a następnie założymy nowy o innej nazwie. Zaznaczamy wszystkie obiekty wzorca klawiszem *ctrl-a*. W panelu właściwości zmieniamy nazwę głównej zmiennej na *%WZmienna*. Ponieważ taki parametr jeszcze nie istnieje, program spyta czy ma on zostać założony – ustawiamy wartość domyślną na *Pomiar1* i akceptujemy utworzenie parametru. Pozostało jedynie zapisać wzorzec w pliku (np. komendą *Zapisz* z menu kontekstowego zakładki *Wzorzec WzorPomiar*).

Wracamy do edycji diagramu *Pomiar*. Usuwamy obiekty miernika i wartości pomiaru, w ich miejsce wstawiamy wzorzec. W panelu *Przybornik* z zakładki *Wzorce* wybieramy wzorzec *WzorPomiar*, a w obszarze diagramu pokazujemy miejsce umieszczenia wzorca. Po kliknięciu na dowolnym fragmencie osadzonego wzorca w panelu właściwości zobaczymy, że mamy możliwość zmiany tylko właściwości osadzenia – nie ma dostępu do obiektów umieszczonych wewnątrz wzorca. Pozostaje nam tylko ustawić wartość parametru wzorca *WZmienna*. W polu edycyjnym wprowadzamy *%Zmienna*. Efektem tego będzie przekazanie od obiektów wzorca (poprzez parametr *WZmienna)* nazwy zmiennej pomiarowej z parametru *Zmienna* diagramu. Po zmianie diagram powinien działać identycznie z poprzednią wersją.

Utworzymy teraz nowy diagram o nazwie *Wszystkie.* Wstawiamy do diagramu dwie instancje wzorca *WzorPomiar*. Jako wartość parametru *WPomiar* podajemy odpowiednio nazwy zmiennych *Pomiar1* i *Pomiar2*. We właściwościach panelu *Info* okna *Start*, ustawiamy diagram *Wszystkie* jako diagram domyślny. Na koniec do diagramu *Menu* dodamy jeszcze jeden przycisk, który będzie wykonywał akcję *^OpenWindow(null,Info,Wszystkie,null)*.

W trybie wykonania aplikacja powinna wyglądać w sposób zbliżony do poniższego.



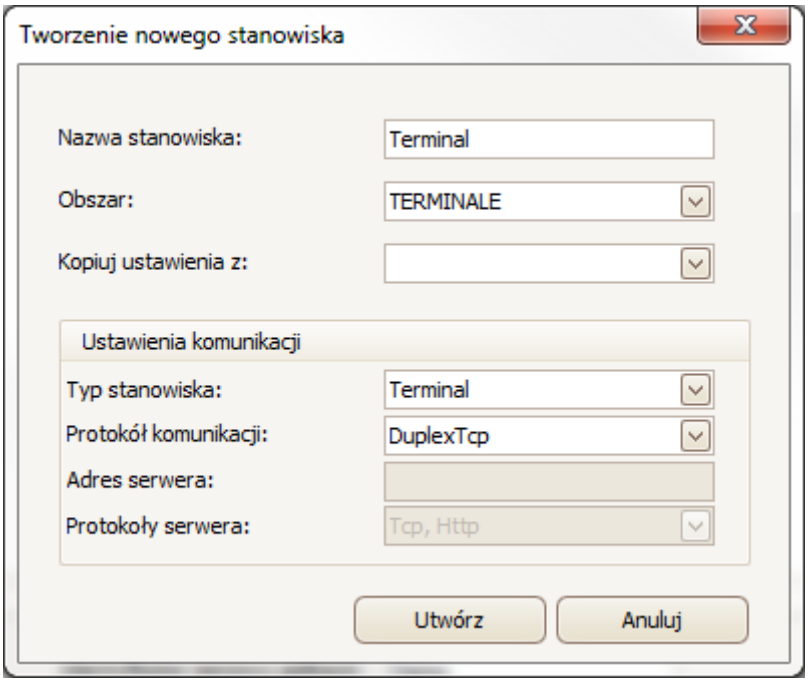
Rys. Edycja diagramu - efekt powielenia obiektów przy użyciu wzorca.

Spróbujmy jeszcze sprawdzić jak zachowa się aplikacja po zmianie wzorca. Przesuńmy we wzorcu wartość tekstową pomiaru w wolne miejsce w obszarze miernika. Po zapisaniu wzorca, wszystkie diagramy automatycznie dostosują się do jego nowej postaci.

3.5 Dodanie stanowiska terminalowego

W trakcie zakładania naszej aplikacji utworzyliśmy tylko pojedyncze stanowisko. Jest to wystarczające w przypadku prostych aplikacji działających na pojedynczym komputerze. Teraz jednak rozszerzymy projekt o pracę w konfiguracji sieciowej, w której pojedynczy komputer pełni rolę serwera danych dla grupy stanowisk terminalowych.

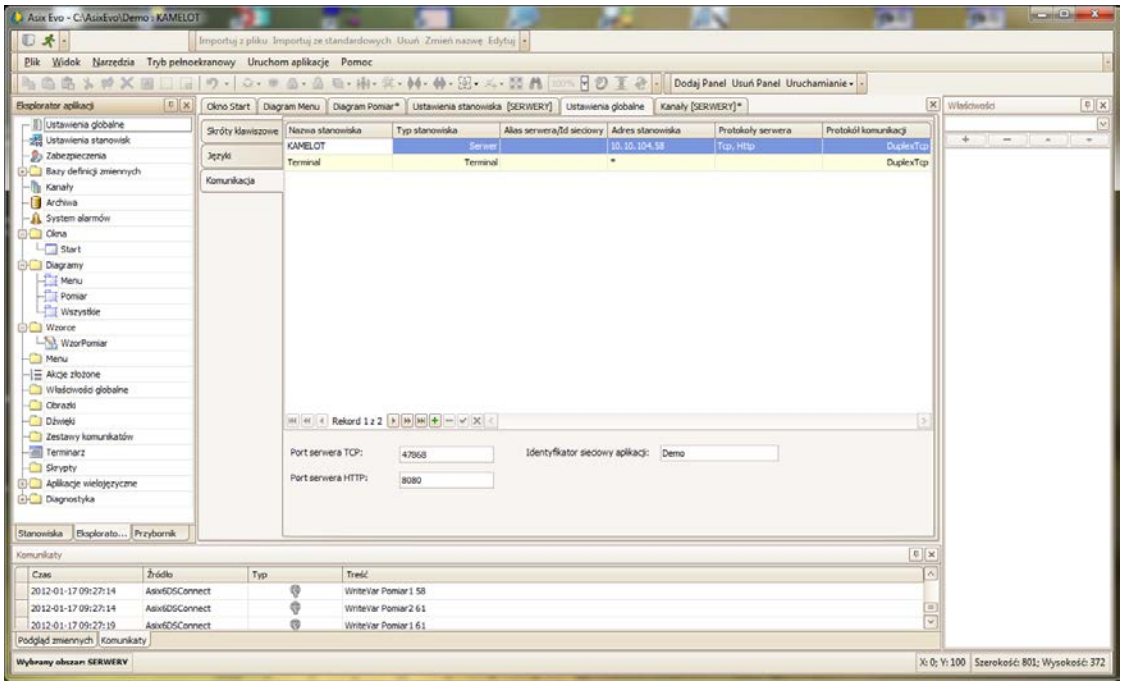
Rolę serwera będzie pełniło stanowisko, którego używaliśmy do tej pory. Musimy tylko założyć dodatkowe stanowisko terminalowe. Otwieramy panel *Stanowiska* i przy pomocy menu kontekstowego, komendą *Nowe stanowisko*, tworzymy nowe stanowisko w obszarze *TERMINALE*.



Rys. Tworzenie nowego stanowiska terminalowego.

Istotne jest aby w ramce *Ustawienia komunikacji* ustawić typ stanowiska na *Terminal*.

Stanowisko zostało utworzone. Pozostaje jeszcze skonfigurować kilka ustawień w szczególności sposób obsługi kanałów komunikacyjnych i archiwów. Rozpoczynamy od zakładki *Komunikacja* w panelu *Ustawienia globalne*.



Rys. Ustawienia globalne dla stanowiska.

Wszystkie parametry powinny być już poprawnie ustawione. Ale należy zweryfikować poniższe ustawienia:

a. Czy typ stanowiska jest ustawiony prawidłowo? Dla stanowiska, które będzie udostępniało dane koniecznie musi być wybrany typ *Serwer*.

b. Czy w kolumnie *Adres stanowiska* znajdują się poprawne wartości? Adres ten służy do lokalizacji stanowiska w sieci. W przypadku serwera musi być to systemowa nazwa komputera lub jego adres IP. Parametr nie jest istotny dla stanowisk terminalowych. Terminale nigdy nie są bezpośrednio adresowane. Poza tym w kontekście stanowiska terminalowego można uruchomić dowolną liczbę komputerów.

W panelu pokazany jest numer portu TCP, który będzie używany do komunikacji pomiędzy stanowiskami. Ważne jest, aby ten port był odblokowany w systemowej zaporze ogniowej.

Przechodzimy teraz do panelu *Stanowiska* i wybieramy w nim obszar *TERMINALE*. Wszystkie dalsze ustawienia będziemy podawać dla tego obszaru. Stanowisko *Terminal* będzie dziedziczyło wszystkie ustawienia z powodu przynależności do obszaru.

Wybieramy panel *Ustawienia stanowisk* i wykonujemy następujące zmiany w konfiguracji:

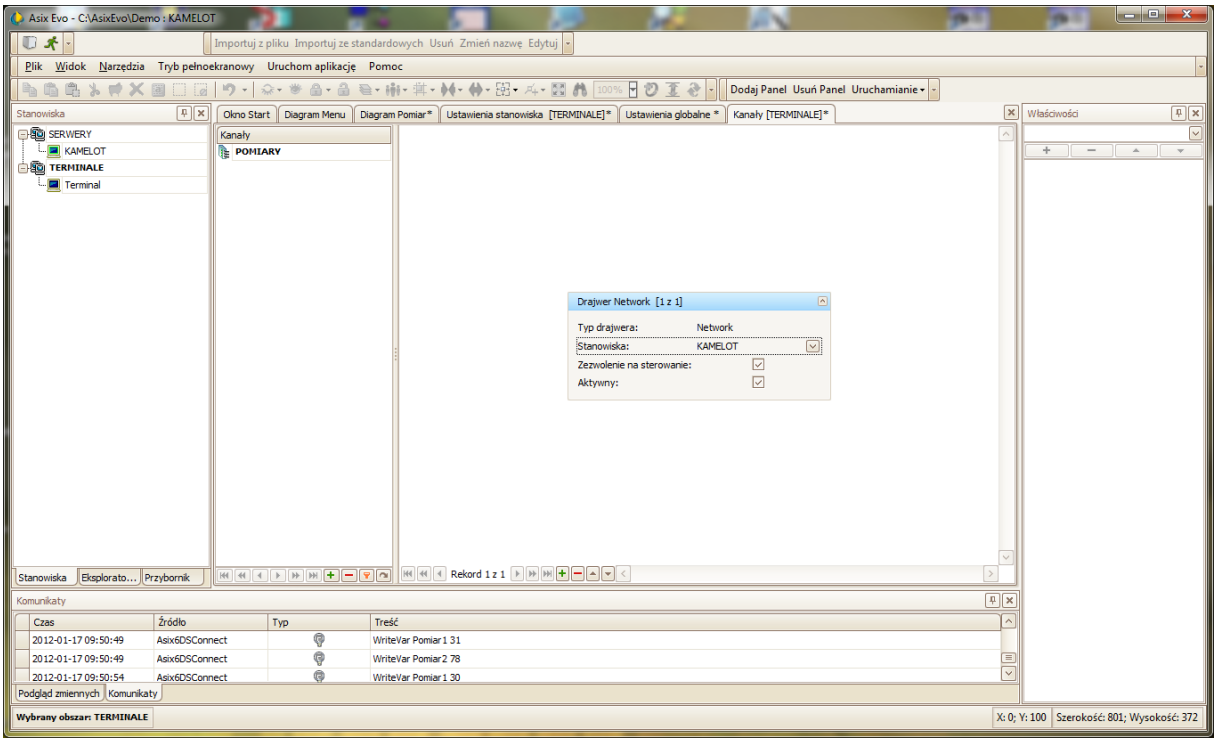
a. W zakładce *Opcje startowe* zaznaczamy okno *Start* jako okno startowe aplikacji.

b. W zakładce *Źródła zmiennych* zaznaczamy bazę *AsixVarbase* jako bazę aktywną.

c. W zakładce *Źródła danych* w ramce *Ustawienia źródła danych Asix* wybieramy *Nie używaj.*

d. W zakładce *Zabezpieczenia* jako użytkownika domyślnego wybieramy użytkownika *Administrator*. To ustawienie ułatwi obsługę naszej aplikacji testowej. Normalna użytkowa aplikacja powinna posiadać w pełni skonfigurowany system zabezpieczeń z dodanym interfejsem do logowania użytkownika.

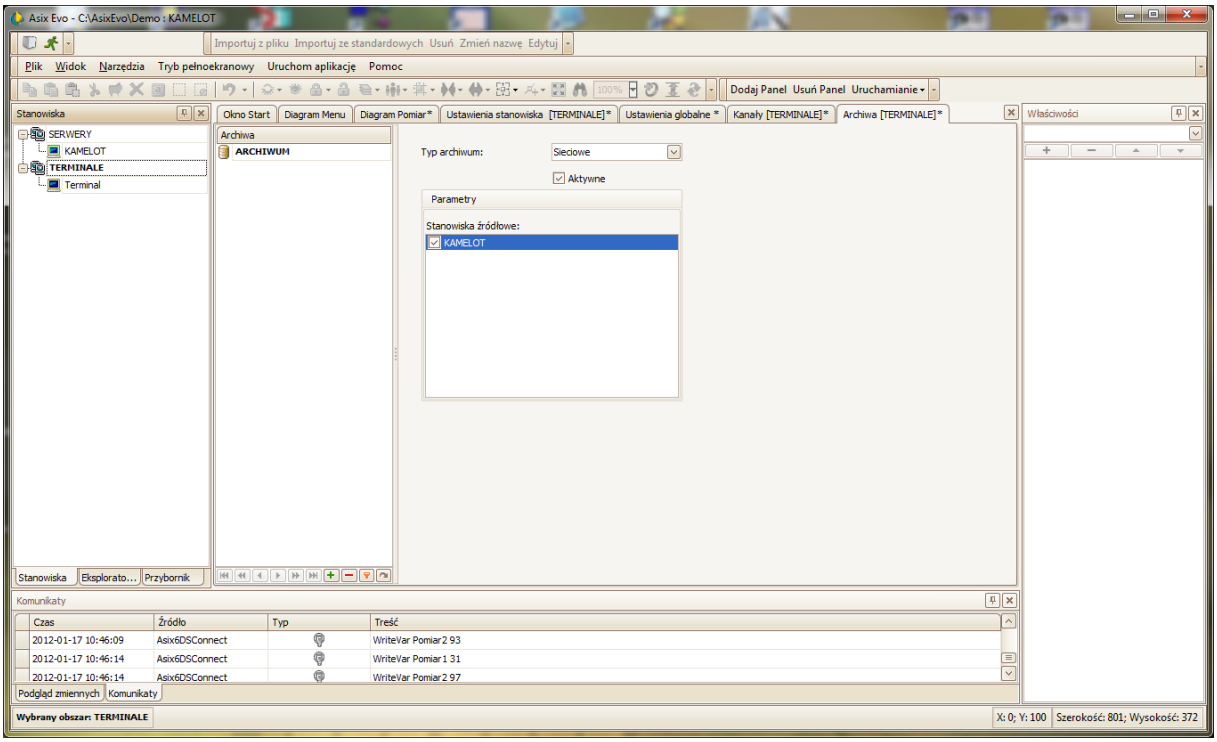
Kolejnym krokiem jest właściwa parametryzacja kanałów komunikacyjnych. W panelu eksploratora aplikacji wybieramy węzeł *Kanały*.



Rys. Parametryzacja kanałów komunikacyjnych dla stanowiska terminalowego.

W lewej części panelu *Kanały* znajduje się lista wszystkich kanałów komunikacyjnych aplikacji. W prawej części pokazane są drajwery komunikacyjne powiązane z wybranym kanałem. Powiązanie dotyczy stanowiska/obszaru wybranego w panelu *Stanowiska*. Aktualnie wybrane stanowisko/obszar jest też wyświetlone w opisie zakładki panelu *Kanały* oraz na pasku statusowym głównego okna programu. Naszym zadaniem jest dodanie, w kontekście obszaru *TERMINALE,* do kanału *POMIARY* drajwera komunikacyjnego typu *Network* (o ile taki drajwer nie został już dodany). Wykonujemy to przy pomocy przycisku **+** paska nawigacyjnego umieszczonego w dolnej części sekcji kanałów. W parametrach drajwera bardzo istotne jest ustawienie właściwości *Stanowiska*. Jest to lista stanowisk typu serwerowego, z których stanowisko terminalowe będzie mogło pobierać dane . Należy wybrać nazwę naszego jedynego stanowiska serwerowego aplikacji.

W analogiczny sposób musimy jeszcze ustawić sposób odczytu danych archiwalnych w paneli *Archiwa.*

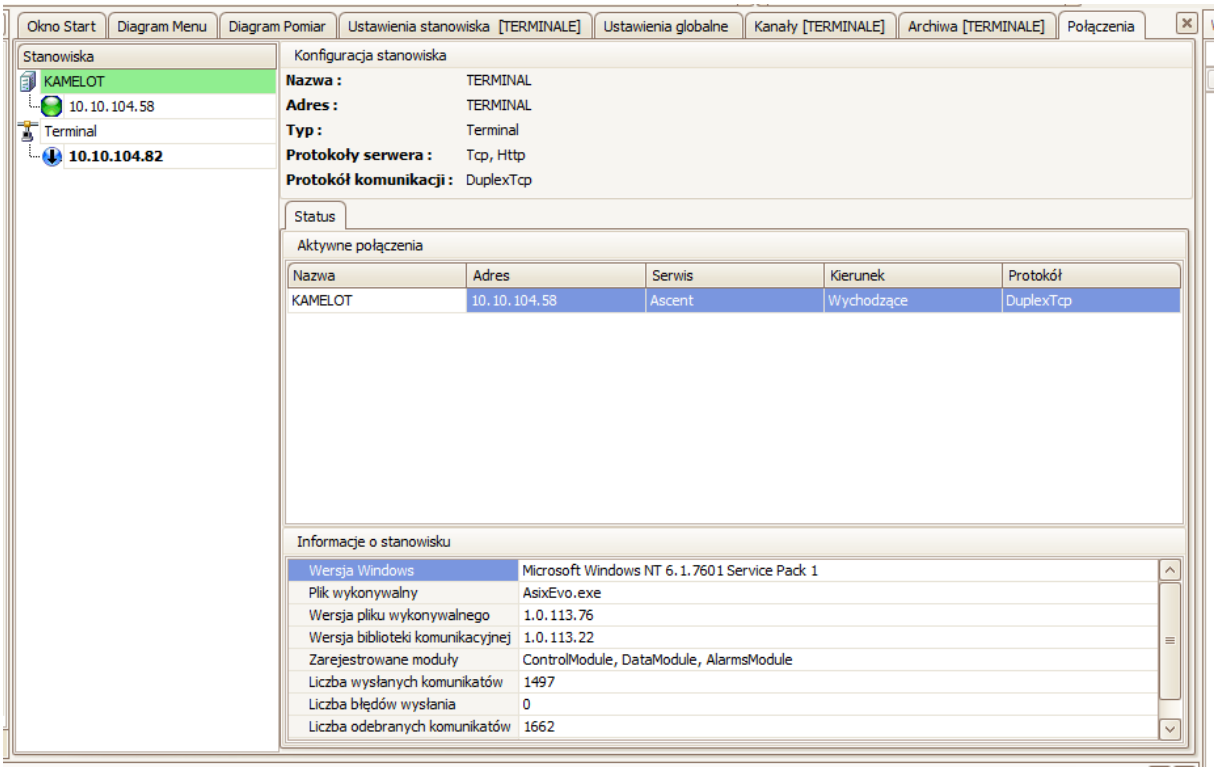


Rys. Parametryzacja archiwów dla stanowiska terminalowego.

Należy wybrać sieciowy typ archiwum oraz zaznaczyć nasze stanowisko serwerowe jako źródło odczytu danych.

Konfiguracja stanowiska terminalowego została zakończona. Można przejść do testu działania. W tym celu należy uruchomić stanowisko serwerowe oraz jedno lub kilka stanowisk terminalowych. Należy pamiętać, aby w trakcie tworzenia skrótów startowych dla stanowisk terminalowych, ustawić w kreatorze opcje *Uruchom jako* na wersję *Terminal*.

W trakcie działanie aplikacji w konfiguracji sieciowej można kontrolować stan systemu poprzez panel *Połączenia* dostępny z panelu eksploratora aplikacji – węzeł *Diagnostyka/Połączenia*.



Rys. Panel podglądu aktywnych połączeń dla stanowisk.

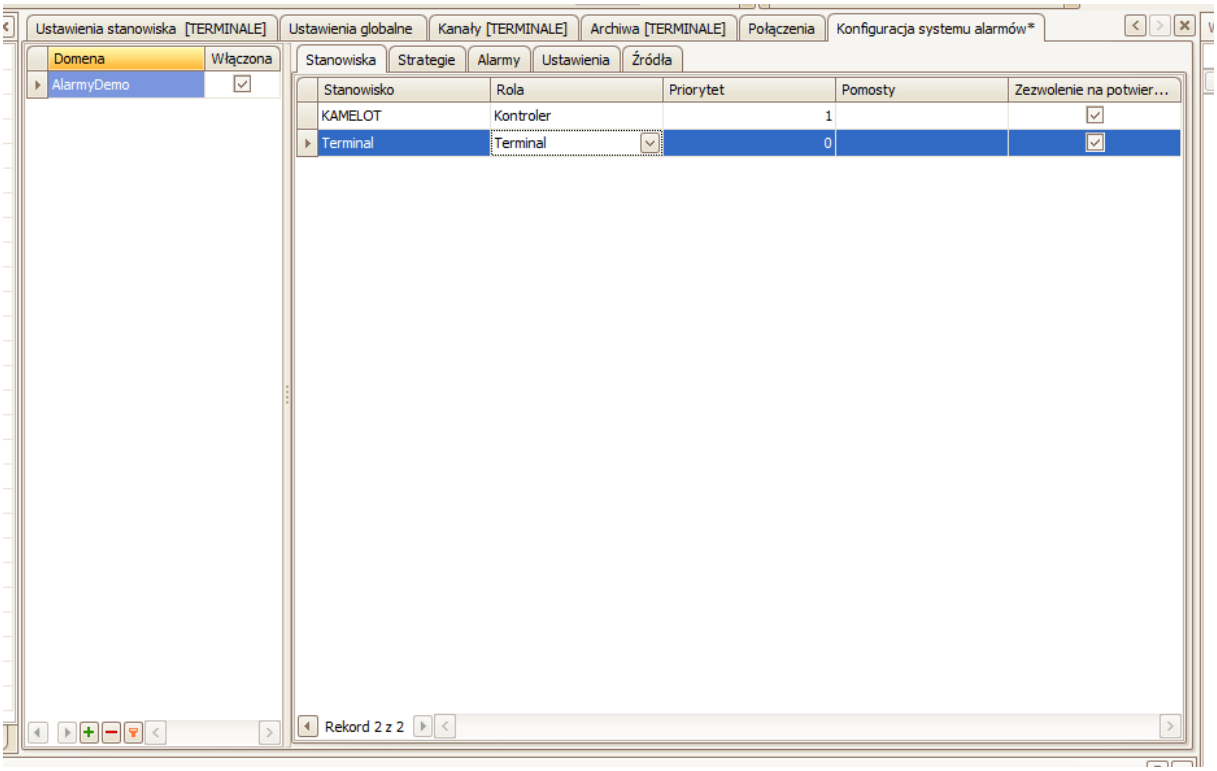
W panelu można zobaczyć wszystkie aktywne połączenia oraz podstawowe informacje o poszczególnych stanowiskach.

4 Aplikacja z obsługą alarmów

W tym rozdziale, nasza poprzednia aplikacja zostanie rozszerzona o obsługę zdarzeń alarmowych. Rozdział obejmie parametryzację systemu alarmów (definicje alarmów i strategie alarmowe) oraz dodanie do aplikacji elementów wizualnych związanych z prezentacją stanu alarmów.

4.1 Parametryzacja systemu alarmów

W panelu eksploratora wybieramy węzeł *System alarmów* i otwieramy panel dedykowany do konfiguracji systemu alarmów.



Rys. Panel dedykowany do konfiguracji systemu alarmów.

Panel składa się z dwóch części. W lewej znajduje się lista tzw. domen alarmów. Strona prawa służy do parametryzacji systemu dla wybranej domeny.

Rozpoczynamy od dodania (używając przycisku **+**) domeny o nazwie *AlarmyDemo*. Każda domena alarmowa stanowi jakby osobny system alarmów. Domeny są od siebie całkowicie niezależne. W typowej aplikacji wystarczające jest założenie pojedynczej domeny. Dodanie dodatkowych domen ma sens tylko wtedy, gdy chcemy łączyć w jednej aplikacji alarmy z kilku całkowicie niezależnych instalacji.

Po założeniu domeny należy określić jaką rolę w systemie alarmów będą pełnić poszczególne stanowiska. W każdej domenie potrzebny jest przynajmniej jeden *Kontroler*. Jest to stanowisko, które koordynuje pracę wszystkich innych stanowisk. Kontroler powinien pracować 24 godziny na dobę i w sensie komunikacyjnym musi być stanowiskiem typu *Serwer*. Możliwe jest posiadanie kilku kontrolerów. Stosuje się to z reguły, w przypadku posiadania redundantnej pary stanowisk operatorskich. Stanowiska pełniące rolę *Terminal* posiadają wszystkie funkcje związane z operatorską obsługą i przeglądaniem alarmów. Nie mogą jednak pracować samodzielnie. Informacje o wykrytych alarmach otrzymują od stanowisk kontrolerów. W naszej aplikacji demonstracyjnej kontrolerem będzie stanowisko serwerowe, a stanowisko *Terminal* będzie pełniło rolę terminala.

Kolejnym krokiem jest określenie jakie strategie wykrywania alarmów będą stosowane w aplikacji. Strategia jest tworem analogicznym do kanałów komunikacyjnych, ale dotyczy procesu wykrywania alarmów. Jej zadaniem jest kontrola stanu alarmów i dostarczanie informacji o zdarzeniach początku i

końca alarmu. Dostępne jest kilka typów strategii, które wykrywają alarmy różnymi metodami. W każdej domenie można skonfigurować dowolną ilość strategii alarmowych, także tego samego typu.

Wybieramy zakładkę *Strategie* i przy pomocy przycisku **+** zakładamy nową strategię o nazwie *Warunki*.

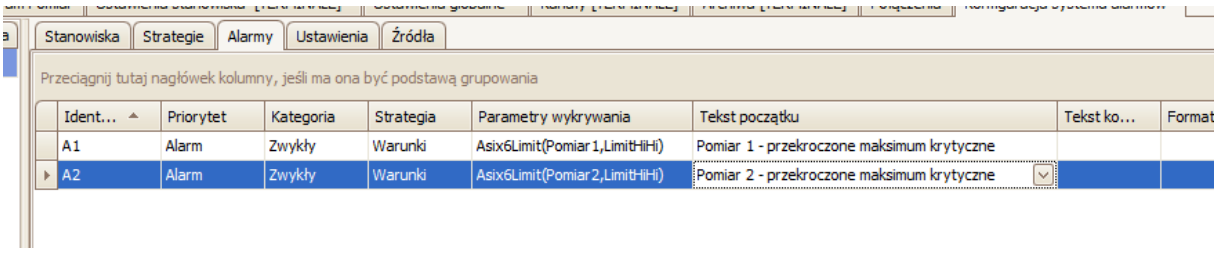


Rys. Deklaracja strategii alarmów.

Jako typ strategii wybieramy *Warunkowa*. Strategia warunkowa wykrywa alarmy na podstawie zajścia dowolnie zdefiniowanych warunków logicznych. Warunki zostaną określone w momencie definiowania bazy alarmów.

We właściwości *Kontrolery* podajemy listę wszystkich kontrolerów domeny, na których strategia może zostać uruchomiona. Typowo podaje się listę wszystkich kontrolerów. Lista może być ograniczona, jeżeli ze specyfiki działania strategii (głównie uwarunkowań komunikacyjnych) wynika, że niektóre stanowiska kontrolerów nie powinny uruchamiać danej strategii. W trakcie działania aplikacji, w danym momencie, strategia wykonywana jest tylko na jednym stanowisku.

Można teraz przejść do zakładki *Alarmy* i zdefiniować dwa alarmy, które będą używane w naszej aplikacji. Pokazuje to poniższe ilustracja.

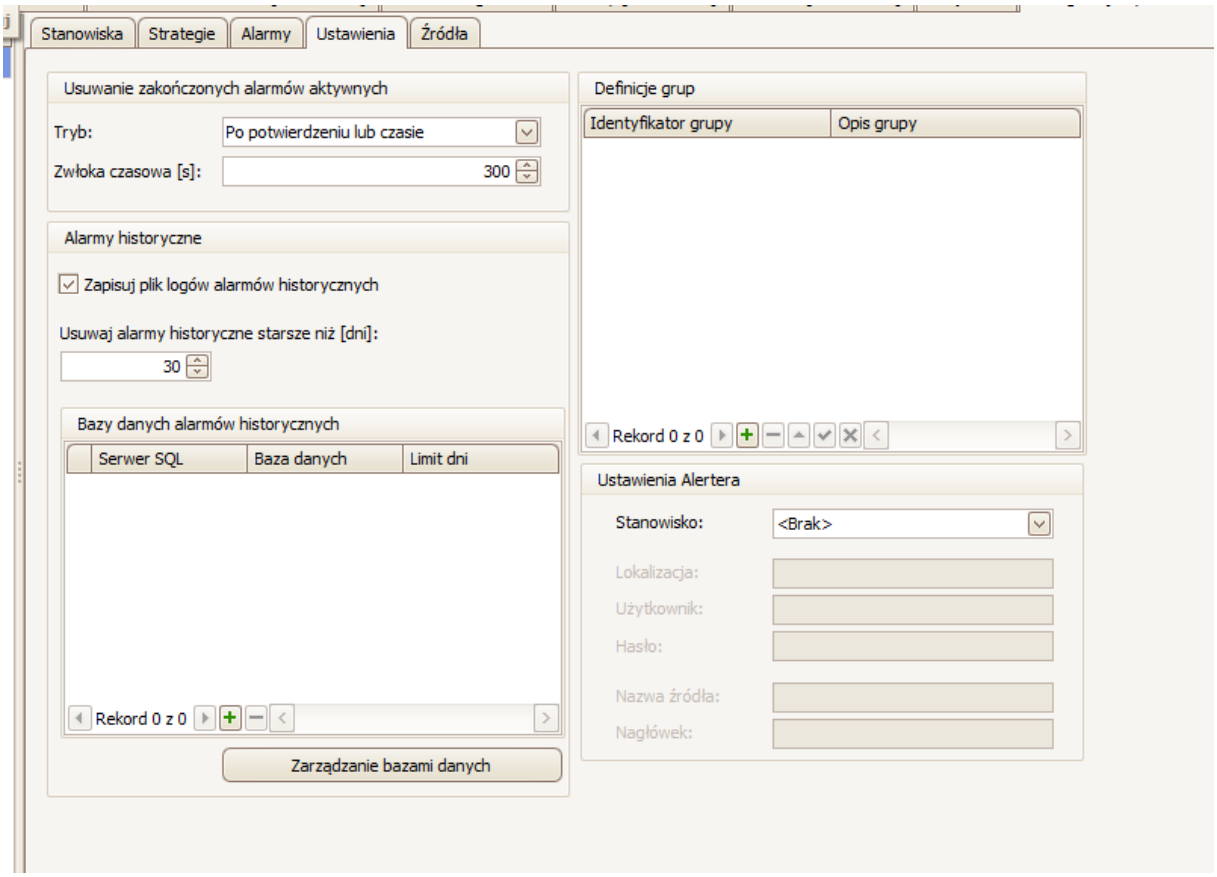


Rys. Deklaracja alarmów.

Każdy alarm posiada unikalny (w ramach domeny) identyfikator. Podstawowe właściwości alarmu to *Strategia* i *Parametry wykrywania.* We właściwości *Strategia* należy wskazać jedną z wcześniej zdefiniowanych strategii wykrywania alarmów. Format właściwości *Parametry wykrywania* zależy od typu wybranej strategii. Dla strategii warunkowej jest to wyrażenie arytmetyczne typu logicznego. Zmiana wartości wyrażenia z *false* na *true* oznacza zdarzenia początku alarmu. Użyta w przykładzie funkcja *Asix6Limit* porównuje wartość wybranej zmiennej z wartością podanego atrybutu. Funkcja intepretuje atrybut limitu na kilka sposobów: może to być limit liczbowy lub nazwa zmiennej, której wartość określa limit. W przypadku, gdy zachodzi potrzeba zdefiniowania dużej liczby alarmów,

zamiast ręcznej edycji można stosować import z plików programu Excel. Funkcje importu dostępne są w zakładce *Źródła*.

Przechodzimy do zakładki *Ustawienia* służącej do określenia pozostałych parametrów pracy systemu alarmów.



Rys. Ustawienia parametrów pracy systemu alarmów.

W ramce *Usuwanie zakończonych alarmów aktywnych* określamy sposób działania logu alarmów aktywnych w momencie wykrycia zdarzenia końca alarmu. W naszej konfiguracji alarm zakończony będzie przechowywany w logu jeszcze przez 5 minut lub do momentu potwierdzenia. Można też ustawić usuwanie natychmiastowe, ale w przypadku alarmów krótkotrwałych mogą być one niezauważone przez operatora.

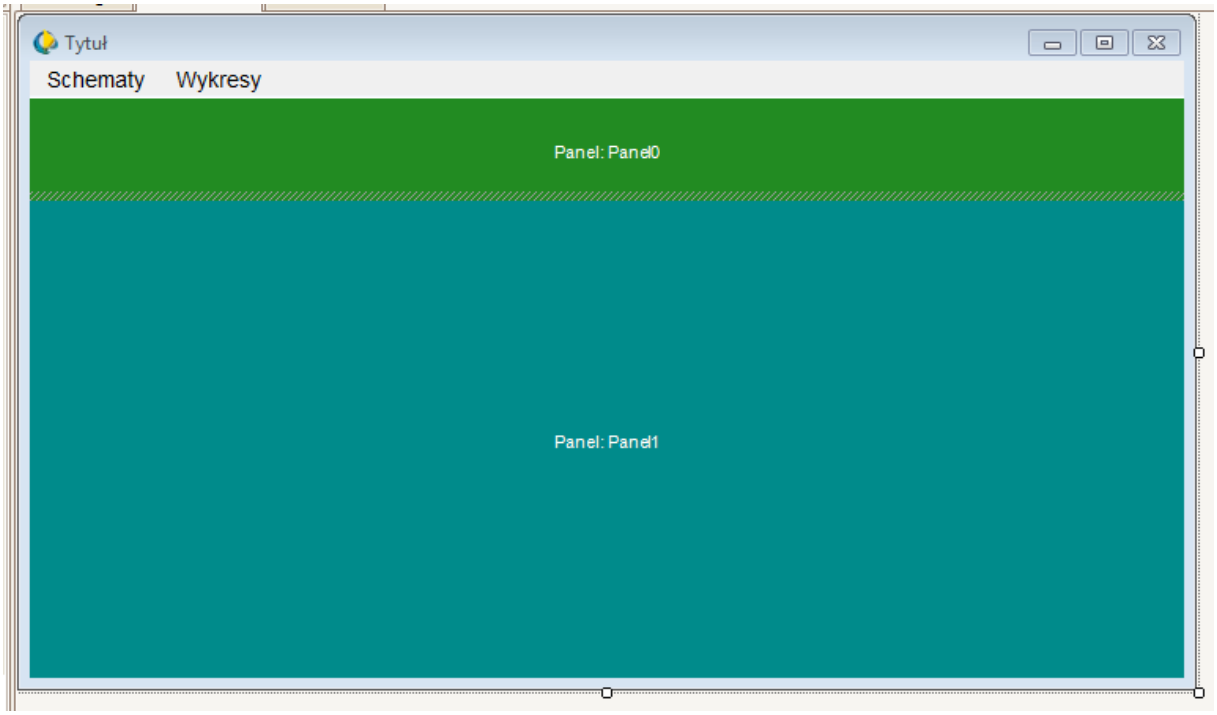
W ramce *Alarmy historyczne* określamy sposób archiwizacji zdarzeń alarmowych. W aplikacji mogą być stosowane dwa logi alarmów historycznych. Pierwszy, kontrolowany opcją *Zapisuj plik logów alarmów historycznych*, jest podstawowym archiwum stosowanym do obsługi alarmów. Archiwa typu SQL są opcjonalne. Są tworzone tylko na potrzeby programu AsAlarm oraz systemu raportowania.

Zapisujemy wszystkie wykonane zmiany. Od tego momentu system alarmów powinien zacząć pracę.

Asix.Evo - Elementy wizualizacji

1 Hierarchia elementów wizualizacyjnych

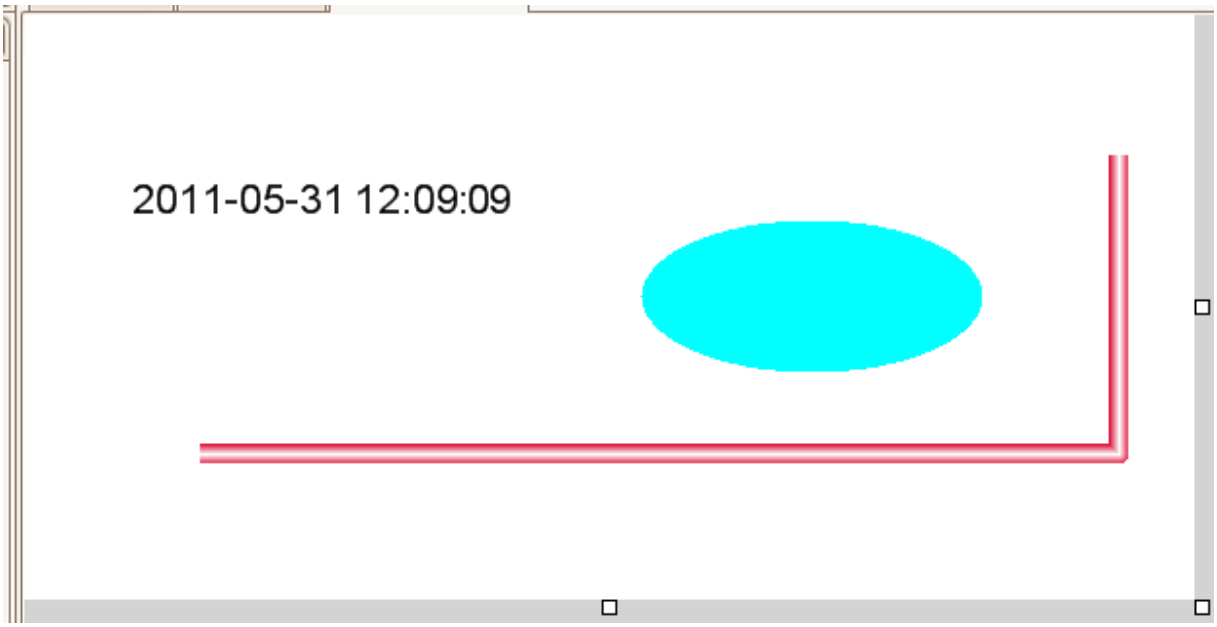
Każda aplikacja Asix.Evo jest wyświetlana w tzw. oknach wizualizacyjnych. Liczba używanych okien jest dowolna, ale zawsze musi być użyte przynajmniej jedno. Jest to okno startowe aplikacji. Okna wizualizacyjne są obiektami systemu Windows i posiadają typowe właściwości okien, np. linia nagłówka, przyciski systemowe, menu systemowe, ramka. Oprócz tych elementów typowych dla systemu operacyjnego, okno wizualizacyjne składa się z opcjonalnego menu oraz jednego lub kilku paneli. Poniższa ilustracja pokazuje okno z menu i dwoma panelami w trakcie edycji.



Rys. Okno aplikacji w trybie edycji.

Panele okien są to kontenery przeznaczone do wyświetlania diagramów synoptycznych. W trakcie wyświetlania okna, diagramy w panelach mogą być wymieniane przy pomocy akcji operatorskich.

Diagram synoptyczny to główny element wizualizacyjny aplikacji służący do prezentacji stanu i obsługi kontrolowanego procesu. Diagram składa się z osadzonych w nim obiektów i wzorców.



Rys. Diagram z osadzonymi obiektami.

Obiekt to pojedynczy element diagramu. Sposób prezentacji i zachowanie obiektu zależy od jego typu. Dostępne są proste obiekty graficzne oraz bardzo złożone obiekty służące do prezentacji dużej ilości informacji, np. obiekty tabel alarmowych.

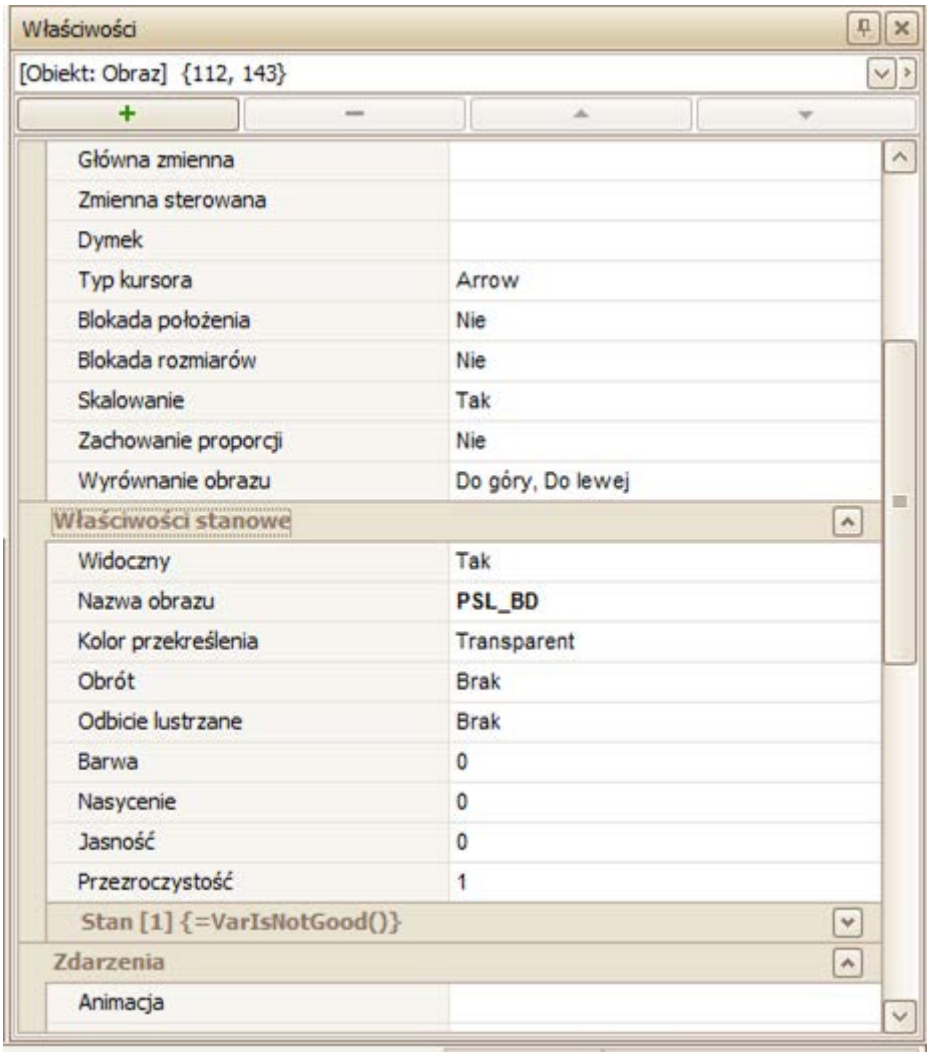
Wzorce to wstępnie sparametryzowane grupy obiektów. Przyspieszają proces tworzenia aplikacji poprzez wielokrotne osadzanie na diagramach przygotowanych wcześniej fragmentów wizualizacji.

Wszystkie elementy wizualizacyjne parametryzowane są poprzez zestawy właściwości i zdarzeń. Zestawy dostępnych właściwości i zdarzeń oraz możliwe metody ich parametryzacji są zależne od typu elementu. Dodatkowym elementem ułatwiającym parametryzację są tzw. właściwości globalne. Są to centralnie definiowane właściwości nie powiązane z żadnym konkretnym elementem aplikacji. Projektant aplikacji może odwołać się wielokrotnie do właściwości globalnej w definicjach różnych elementów. Stwarza to możliwość łatwej modyfikacji wyglądu lub zachowania aplikacji bez konieczności zmiany definicji właściwości wielu elementów składowych.

2 Właściwości i zdarzenia elementów wizualizacyjnych

Właściwości i zdarzenia elementów są definiowane w panelu *Właściwości*. Po wybraniu elementu w panelu jest wyświetlany odpowiedni zestaw właściwości i zdarzeń.

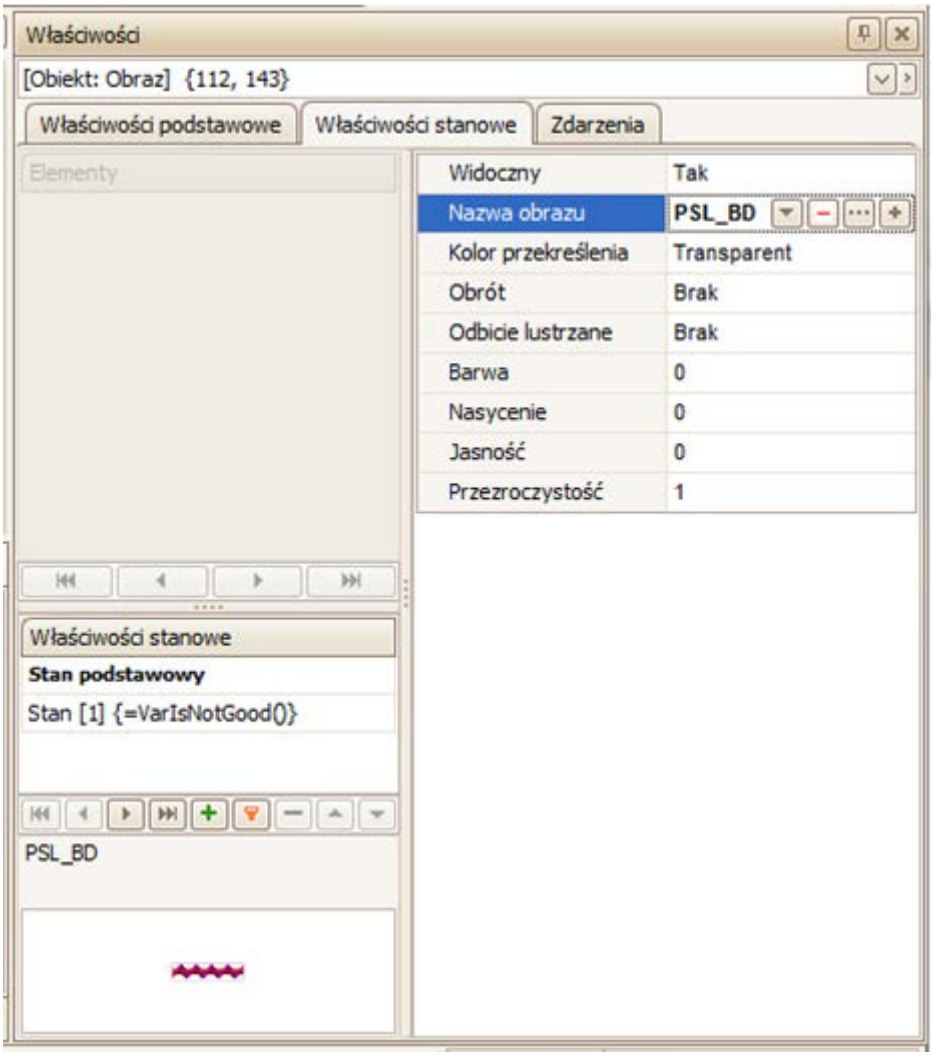
Dostępne są dwie wersje panelu właściwości – można zmienić tryb korzystając z przycisku w górnym prawym narożniku panelu.



Rys. Klasyczny panel właściwości elementów wizualizacyjnych.

Tzw. panel klasyczny pokazuje wszystkie właściwości, zdarzenia, parametry w jednej długiej liście z możliwością zwijania poszczególnych fragmentów listy. Panel jest węższy i pozwala na jedoczesne oglądanie elementów definicji tego samego typu.

Bardziej rozbudowany, panel kompaktowy pokazuje tylko elementy definicji wybrane przy pomocy zakładek i list wyboru. Panel jest szerszy, ale pozwala łatwiej wyszukiwać żądane elementy, szczególnie w przypadku rozbudowanych obiektów wielostanowych. Dodatkowo w lewym dolnym narożniku panelu, znajduje się pole podglądu wartości wynikowej dla wybranej właściwości.



Rys. Kompaktowy panel właściwości elementów wizualizacyjnych.

Właściwości elementu wizualizacyjnego są odpowiedzialne za jego wygląd i sposób działania. Typowym przykładem są właściwości służące do określenia koloru użytego do wyświetlenia elementu. Zdarzenia służą natomiast do wykonania akcji definiowanych przez projektanta w reakcji na zajście pewnych zdarzeń w trakcie wykonania aplikacji. Typowy przykład to zdarzenie kliknięcia klawiszem myszki w obszarze obiektu.

Zarówno właściwości jaki i zdarzenia mogą byś sparametryzowane w taki sposób, że ich wartość zmienia się w trakcie wykonywania aplikacji. Zmiany te mogą zależeć od zmian w stanie kontrolowanego procesu, od sposobu otwarcia elementu lub od uprawnień zalogowanego użytkownika i jego działań.

Definicja właściwości lub zdarzenia jest tekstem, który jest interpretowany w zależności od pierwszego znaku definicji i od kontekstu użycia. Definicje mogą być wprowadzane ręcznie lub przy pomocy okienek udostępnianych przez panel *Właściwości*.

2.1 Prefiksy definicji właściwości

Poniższa tabelka opisuje sposób interpretacji definicji właściwości w zależności od użytego znaku prefiksu.

Prefiks	*Znaczenie*	*Opis*
~ lub brak	Wartość bezpośrednia	Wartość właściwości jest podana bezpośrednio i nie ulega zmianie w trakcie używania elementu. Sposób podania wartości zależy od typu właściwości. Przykłady: Red Właściwości określające kolor akceptują nazwę koloru. Tak Właściwości typu logicznego akceptują teksty *Tak* i *Nie.* Płaski Specjalizowane właściwości mogą mieć własny zestaw dopuszczalnych wartości. 10 Duża część właściwości akceptuje wartości liczbowe.
#	Zmienna główna	W zależności od kontekstu użycia, notacja z prefiksem # pozwala odwołać się do nazwy lub wartości zmiennej głównej. Nazwa zmiennej głównej jest definiowana we właściwości Główna zmienna obiektu. Odwołania do zmiennej głównej można stosować tylko w kontekście obiektu. W przypadku właściwości, które reprezentują nazwę zmiennej (np. Zmienna sterująca) składnia notacji jest następująca: #[*sufiks*] Użycie tylko znaku # oznacza wprost nazwę zmiennej głównej. Dodanie sufiksu pozwala utworzyć nazwę utworzoną z połączenia nazwy i zmiennej głównej i sufiksu. Przykład: #_limit Jeżeli zmienna główna posiada nazwę *Z1*, to zostanie utworzona nazwa *Z1_limit*. Dla właściwości, które nie mają charakteru nazwy zmiennej, prefiks # oznacza odwołanie do wartości zmiennej. W tym przypadku składnia notacji jest następująca:

		#[*sufiks*][.*numer_bitu*[r]] Jeżeli sufiks nie jest podany, to pobierana jest bieżąca wartości zmiennej głównej obiektu. Jeżeli sufiks jest podany, to pobierana jest wartość zmiennej procesowej o nazwie złożonej z nazwy zmiennej głównej i podanego sufiksu. Opcjonalne użycie klauzuli numeru bitu pozwala zamiast wartości zmiennej, zwrócić wartość wybranego bitu jako wartość typy logicznego *true/false*. Znacznik *r* pozwala odwrócić logikę: wartość *true* jest zwracana, gdy wartość bitu jest równa 0. Wartość zwracana przez wyrażenie z prefiksem # musi być kompatybilna z typem właściwości, w kontekście której zostało użyte. Przykłady: # Zwraca wartość zmiennej głównej. Użycie we właściwości *Tekst* obiektu *Tekst* spowodowałoby wyświetlenie wartości zmiennej skonwertowanej na postać tekstową. #_limit Jeżeli zmienna główna posiada nazwę *Z1*, to pobrana zostanie wartość zmiennej *Z1_limit*. #.0 Zwraca wartość logiczną wyliczaną na podstawie stanu najmłodszego bitu wartości zmiennej głównej.
&	Zmienna procesowa	Pobranie wartości dowolnej zmiennej procesowej. Nazwę zmiennej należy podać po znaku prefiksu. Pełna składnia notacji jest następująca: *&nazwa_zmiennej*[.*numer_bitu*[r]] Opcjonalne użycie klauzuli numeru bitu pozwala zamiast wartości zmiennej, zwrócić wartość wybranego bitu jako wartość typy logicznego *true/false*. Znacznik *r* pozwala odwrócić logikę: wartość *true* jest zwracana, gdy wartość bitu jest równa 0. Wartość zwracana przez wyrażenie z prefiksem # musi być kompatybilna z typem właściwości, w kontekście której zostało użyte. Przykład: &z1 Pobranie wartości zmiennej *z1*.

@	Atrybut zmiennej	Pobranie wartości atrybutu zmiennej procesowej z bazy definicji zmiennych. Nazwę zmiennej procesowej i atrybutu należy podać po znaku prefiksu. Jeżeli nazwa zmiennej nie jest podana, to używana jest zmienna główna obiektu. Jeżeli nazwa atrybutu nie jest podana, to pobierany jest atrybut domyślny właściwości. Wartość atrybutu musi być kompatybilna z typem właściwości, w kontekście której została użyta. Przykład: @ Pobiera atrybut domyślny zmiennej głównej. Można użyć tej formy tylko wtedy, gdy właściwość ma zdefiniowany atrybut domyślny. @Description Pobiera opis zmiennej głównej obiektu. @Z1.Description Pobiera opis zmiennej *Z1*.
%	Parametr	Pobranie wartości parametru wzorca lub diagramu. Nazwę parametru należy podać po znaku prefiksu. Parametry są mechanizmem pozwalających na tworzenie wzorców i diagramów, których działanie zależy od parametrów dostarczanych dopiero w momencie użycia elementu. Wartość parametru musi być kompatybilna z typem właściwości, w kontekście której została użyta. Przykład: %zmienna Pobranie wartości parametru diagramu lub wzorca o nazwie *zmienna*.
$	Napis wielojęzyczny	Odwołanie do tekstu wielojęzycznego zdefiniowanego w panelu roboczym *Napisy aplikacji* (otwieranym za pośrednictwem panelu eksploratora aplikacji). Identyfikator napisu należy podać po znaku prefiksu. Prefiks $ można stosować tylko we właściwościach o znaczeniu tekstowym. W trakcie działania aplikacji, w miejscu zastosowania zostanie wyświetlony tekst w aktualnym języku pracy aplikacji.
!	Właściwość globalna	Pobranie wartości właściwości globalnej. Nazwę właściwości należy podać po znaku prefiksu. Wartość właściwości globalnej musi być kompatybilna z typem właściwości, w kontekście której została użyta. Przykład: !KolorLowLow

		Pobranie wartości właściwości globalnej o nazwie *KolorLowLow*.
=	Wyrażenie	Wyliczenie wartości wyrażenia arytmetycznego. Treść wyrażenia należy podać po znaku prefiksu. Wynik wyrażenia może być zależny od wielu czynników (np. wartości zmiennych procesowych). Jest wyliczany na bieżąco w trakcie działania aplikacji. Zobacz: *Wyrażenia*. Wynik obliczenia wyrażenia musi być kompatybilny z typem właściwości, w kontekście której wyrażenie zostało użyte. Przykład: =”Temperatura = ” + Variable(t1) Wynikiem wyrażenia jest tekst łączący napis *„Temperatura = ”* z wartością zmiennej *t1*.
^	Akcja operatorska	Definiuje treść akcji operatorskiej, która zostanie wykonana w momencie zajścia zdarzenia. Treść akcji należy podać po znaku prefiksu. W parametrach akcji mogą być wewnętrznie użyte wyrażenia. Stwarza to możliwość wykonywania akcji, których działanie zależy od czynników typu wartość zmiennej procesowej itp.. Zobacz: *Akcje operatorskie*. Przykład: ^SetVariable(”#”,Variable()+1) Akcja spowoduje zwiększenie bieżącej wartości zmiennej głównej obiektu o 1.

UWAGA:

Nie każdy prefiks jest dopuszczalny w każdym kontekście użycia. W definicji zdarzeń można używać wyłącznie prefiks akcji operatorskich. Niektóre właściwości mogą być wprowadzane tylko w sposób bezpośredni – dotyczy to przede wszystkim właściwości okien i paneli.

3 Okna i panele

Okno jest podstawowym elementem aplikacji, w ramach którego wyświetlane są diagramy służące do prezentacji stanu kontrolowanego procesu. Większość właściwości okna służy do określenia jego wyglądu i dostępnych funkcji systemowych.

Okno może posiadać własne menu, którego nazwa jest podawane we właściwości *Menu*. W zależności od sposobu zdefiniowania menu, może się ono wyświetlić wzdłuż dowolnej krawędzi okna.

Każde okno składa się z paneli, w których w trakcie pracy umieszczane są diagramy. Określenie, które diagramy mają zostać wyświetlone w poszczególnych panelach, odbywa się:

- na etapie definiowania panelu, we właściwości *Domyślny diagram.*
- w trakcie pracy aplikacji w treści akcji operatorskiej *OpenWindow*

W trakcie definiowania paneli istotne jest określenie ich rozmiarów, szczególnie w przypadku, gdy projektowane okno ma zezwolenie na zmianę rozmiaru. Panele mogą dostosowywać swój rozmiar do zmienianego rozmiaru okna. Służą do tego właściwości paneli *Dokowanie* i *Zakotwiczenie*. Sposób działania tych właściwości najlepiej sprawdzić eksperymentalnie. W przypadku paneli dokowanych istotne znaczenie ma kolejność definiowania paneli.

Okno może zostać otwarte poprzez:

- zaznaczenie okna jako okna startowego w ustawieniach stanowiska – w tym przypadku okno jest otwierane w trakcie startu aplikacji.
- użycie akcji *OpenWindow*
- użycie akcji *OpenDiagram* – w tym przypadku tworzone jest tymczasowe okno złożone z pojedynczego panelu wypełniającego całe okno, o rozmiarze dostosowanym do rozmiaru diagramu

3.1 Właściwości okien

Nazwa	*Typ wartości*	*Opis*
Nazwa okna	Nazwa	Nazwa identyfikująca okno. Pierwszy znak musi być literą, a następne literą, cyfrą lub znakiem _.
Tytuł okna	Tekst	Opis zawartości okna wyświetlany na belce tytułowej
Kolor tła	Kolor	Kolor używany do wyświetlenia tła okna. Ma znaczenie w trybie edycji lub wtedy, gdy obszar okna

		nie jest całkowicie przykryty panelami.
X	Liczba	Współrzędna X otwarcia okna w trybie wykonawczym wyrażona w pikselach. Współrzędna otwarcia może być zmieniona w akcji operatorskiej otwierającej okno lub wynikać z trybu otwarcia podanego we właściwości *Stan początkowy okna*.
Y	Liczba	Współrzędna Y otwarcia okna w trybie wykonawczym wyrażona w pikselach. Współrzędna otwarcia może być zmieniona w akcji operatorskiej otwierającej okno lub wynikać z trybu otwarcia podanego we właściwości *Stan początkowy okna*.
Szerokość	Liczba	Szerokość okna wyrażona w pikselach.
Wysokość	Liczba	Wysokość okna wyrażona w pikselach.
Menu	Nazwa menu	Opcjonalna nazwa menu używanego w oknie.
Pokazuj belkę tytułu	Logiczna	Określa, czy belka tytułowa okna ma być wyświetlana.
Pokazuj ramkę	Logiczna	Określa, czy ramka wokół okna ma być wyświetlana. Niektóre inne ustawienia mogą również wymusić wyświetlenie ramki.
Stała pozycja	Logiczna	Określa, czy użytkownik może zmienić pozycję okna.
Stałe rozmiary	Logiczna	Określa, czy użytkownik może zmienić rozmiar okna. Nawet jeżeli rozmiar jest zablokowany, to można go zmienić przy pomocy akcji operatorskiej *SetWindowSize*.

Okno narzędziowe	Logiczna	Zmienia wygląd belki tytułowej i jednocześnie blokuje wyświetlanie przycisków systemowych.
Pokazuj przyciski	Logiczna	Blokuje wyświetlanie przycisków systemowych na belce tytułowej.
Przycisk maksymalizacji	Logiczna	Określa, czy na belce tytułowej ma być wyświetlany przycisk maksymalizacji.
Przycisk minimalizacji	Logiczna	Określa, czy na belce tytułowej ma być wyświetlany przycisk minimalizacji.
Zawsze pod spodem	Logiczna	Określa, czy okno jest wyświetlane zawsze pod spodem innych okien (aplikacji Asix.Evo i innych). Zazwyczaj używane dla dużych okien startowych aplikacji. Pozwala ograniczyć używanie trybu *zawsze na wierzchu*.
Zawsze na wierzchu	Logiczna	Określa, czy okno jest wyświetlane zawsze nad innymi oknami. Powoduje to, że okno nieaktywne nie będzie przesłaniane. Tryb ten nie należy stosować dla dużych okien bo mogą spowodować niepożądane całkowite zasłonięcie innych okien. Alternatywną metodą jest stosowanie trybu *zawsze pod spodem* dla okien startowych.
Stan początkowy okna	Typ wyliczeniowy: Zwykły, Zminimalizowane, Zmaksymalizowane, Środek ekranu, Środek okna wywołujacego	Określa tryb otwarcia okna. Tryb *zwykły* powoduje otwarcie zgodnie z właściwościami *X, Y, Szerokość* i *Wysokość*. Tryb *zminimalizowany* minimalizuje okno, a *zmaksymalizowany* powiększa(lub pomniejsza) okno do rozmiaru ekranu. Opcja ta jest szczególnie przydatna, gdy aplikacja powinna się automatycznie dopasać do różnych rozdzielczości ekranu.

		Opcja *Środek ekranu* wyświetla okno na środku ekranu z zachowaniem oryginalnych rozmiarów a opcja Środek okna wywołującego wyświetla okno centralnie względem okna nadrzędnego.
Autodopasowanie do ekranu	Logiczna	Właściwość wykorzystywana jest w przypadku uruchamiania aplikacji na różnych komputerach z monitorami o różnych rozmiarach. Okno z ustawionym autodopasowaniem, automatycznie zmniejszy swoje rozmiary, jeżeli zostanie otwarte na monitorze , na którym nie zmieści się w całości.
Zachowanie proporcji	Logiczna	Wartość *Tak* oznacza, że w trakcie zmieniania wielkości okna przez użytkownika, zachowane zostaną oryginalne proporcje rozmiarów
Przezroczystość	Liczba <0,1>	Określa stopień przezroczystości okna w zakresie od 0 do 1. Wartość 1 oznacza okno nieprzezroczyste, wartość 0 oznacza okno całkowicie przezroczyste.
Pokazuj na pasku zadań	Logiczna	Określa, czy informacja o otwartym oknie ma być wyświetlana na pasku zadań.
Zezwolenie na zamykanie	Logiczna	Blokuje użytkownikowi możliwość zamknięcia okna. Okna zablokowane można zamknąć tylko przy pomocy akcji *CloseWindow*.

3.2 Właściwości paneli

Nazwa	*Typ wartości*	*Opis*
Nazwa panelu	Nazwa	Nazwa identyfikująca panel. Pierwszy znak musi być literą, a następne literą, cyfrą lub znakiem _.
Domyślny diagram	Nazwa diagramu	Nazwa diagramu, który należy automatycznie otworzyć w panelu. Używana, gdy okno jest oknem startowym aplikacji, lub gdy w akcji *OpenWindow* nie podano nazwy diagramu.
Domyślne parametry diagramu	Tekst	Parametry domyślnie otwieranego diagramu. Zestaw dozwolonych parametrów zależy od definicji diagramu. Parametry są podawane jako sekwencja par *nazwa_parametru=wartość_parametru* rozdzielonych znakiem średnika, np. Kolor=Red;Zmienna=v1
Kolor tła	Kolor	Kolor używany do wyświetlenia tła panelu. Ma znaczenie głównie w trybie edycji. W trybie wykonawczym używany tylko wtedy, gdy do panelu nie został podłączony diagram.
Dokowanie	Typ wyliczeniowy: Brak, Wypełnienie, Do góry, Do dołu, Do lewej, Do Prawej	Określa tryb dokowania panelu. Tryb *wypełnienie* oznacza zajęcie całego wolnego miejsca. Tryb dokowania do krawędzi oznaczają zajęcia miejsca przy wybranej krawędzi – wielkość panelu określona jest wtedy przez właściwość *Szerokość* lub *Wysokość*.
Zakotwiczenie	Typ wyliczeniowy wielokrotny: Brak, Do góry, Do dołu, Do lewej, Do prawej	Określa sposób wyliczania rozmiaru panelu, gdy zmieniany jest rozmiar okna. Zakotwiczenie do wybranej krawędzi oznacza, że utrzymywana jest stała odległość okna od tej krawędzi. Można zakotwiczyć panel do kilku krawędzi jednocześnie.
X	Liczba	Współrzędna X pozycji panelu wyrażona w pikselach, liczona względem lewego górnego narożnika okna. Nie ma znaczenia w przypadku paneli dokowanych.
Y	Liczba	Współrzędna Y pozycji panelu wyrażona w pikselach, liczona względem lewego górnego narożnika okna. Nie ma znaczenia w przypadku paneli dokowanych.

Szerokość	Liczba	Początkowa szerokość panelu wyrażona w pikselach. Może ulec zmianie w wyniku zmiany rozmiaru okna, gdy stosowane są tryby dokowania lub zakotwiczenia.
Wysokość	Liczba	Początkowa wysokość panelu wyrażona w pikselach. Może ulec zmianie w wyniku zmiany rozmiaru okna, gdy stosowane są tryby dokowania lub zakotwiczenia.
Widoczny przy starcie	Logiczny	Określa, czy panel jest domyślnie widoczny po uruchomieniu okna.

4 Diagramy

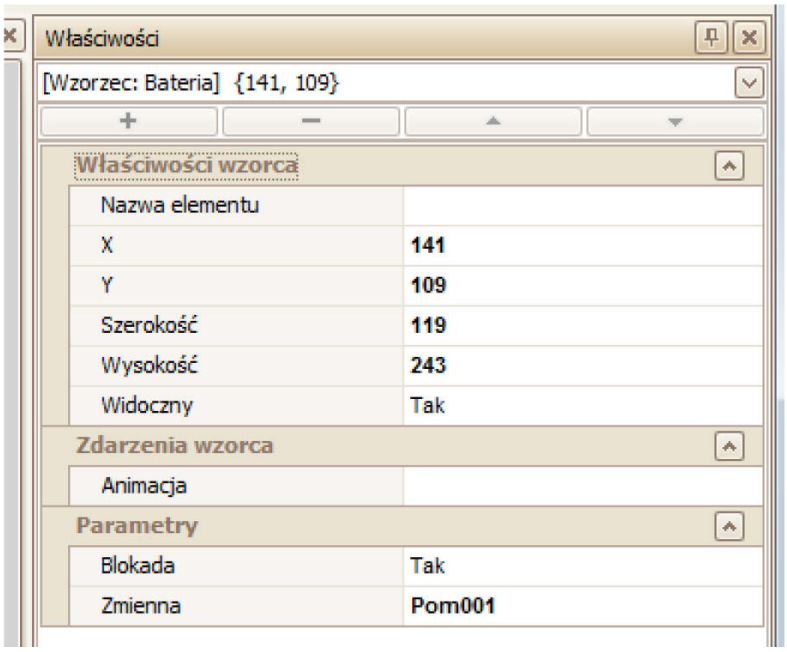
Diagram to główny element wizualizacyjny aplikacji służący do prezentacji stanu i obsługi kontrolowanego procesu. Diagramy są zawsze wyświetlane wewnątrz panelu okna. Okno może być otwarte poprzez:

- zadeklarowanie diagramu jako diagram domyślny panelu - w tym przypadku w momencie otwarcia okna (jeżeli nie będzie innych dyspozycji), diagram zostanie automatycznie wyświetlony.
- użycie nazwy diagramu w treści akcji *OpenWindow* - diagram zostanie otwarty we wskazanym panelu okna, akcja może dotyczyć nowo otwieranego okna lub okna otwartego wcześniej (wymiana diagramu).
- użycie nazwy diagramu w treści akcji *OpenDiagram* - diagram zostanie otwarty w automatycznie utworzonym oknie tymczasowym.

Istotnym aspektem definicji diagramu jest określnie, jak rozmiar diagramu dopasowywany jest do rozmiaru panelu. Służy do tego właściwość *Stałe rozmiary*. Jeżeli jest ona ustawiona na *Nie*, to rozmiar diagramu jest zawsze identyczny z rozmiarem panelu - w przypadku zmiany rozmiaru panelu położenie i wymiary wszystkich elementów diagramu jest przeskalowywane. Jeżeli tryb stałych rozmiarów został wybrany, to diagram nie jest skalowany, wyświetlany jest w oryginalnych rozmiarach, a w razie konieczności do panelu dodawane są suwaki pozwalające na wybór części widzialnej diagramu. Tryb stałych rozmiarów nie wyklucza jednak ręcznego skalowania diagramu. Ustawienie właściwości *Zezwalaj na powiększanie* na wartość *Tak* powoduje, że użytkownik może przeskalować wielkość diagramu przy pomocy rolki myszki (przy wciśniętym przycisku *Control*), ale w dalszym ciągu zmiana rozmiaru panelu nie ma wpływu na sposób wyświetlenia okna.

Diagram składa się z osadzonych w nim obiektów, grup obiektów i osadzonych wzorców. Grupy obiektów pozwalają na połączenie razem kilku obiektów znajdujących się na diagramie. Ułatwia to edycję obiektów (w szczególności selekcję, przesuwanie i kopiowanie obiektów grupy). Dodatkowo ułatwiona jest animacja ruchu grupy obiektów poprzez obsługę zdarzenia *Animate* grupy. Poza tym

obiekty wchodzące w skład grupy zachowują się identycznie jako obiekty niezgrupowane. Inna sytuacja jest w przypadku osadzonego wzorca. Wzorzec podobnie jak grupa składa się z zestawu obiektów. W tym przypadku definicje tych obiektów nie znajdują się jednak wewnątrz definicji diagramu. Wzorce stanowią osobno definiowany element aplikacji. W diagramie przechowywana jest tylko informacja o sposobie osadzenia wzorca. Informacja ta składa się z właściwości określających rozmiar i położenie wzorca oraz wartości parametrów, które zostały użyte we wzorcu.



Rys. Panel właściwości wzorca.

Podobnie jak grupa, wzorzec posiada zdarzenie *Animacja*, które pozwala na animację ruchu wzorca.

Diagram może być diagramem parametryzowanym. Pozwala to używać ten sam diagram do wyświetlania informacji z różnych źródeł lub modyfikację wyglądu bez zmiany definicji diagramu. Poprawne użycie parametrów składa się z dwóch kroków:

- Na etapie definicji diagramu definiowane są parametry diagramu. Definicja składa się z nazwy parametru oraz jego wartości domyślnej. Wartość domyślna używana jest w trakcie edycji diagramu oraz wtedy, gdy w akcji otwarcia diagramu nie zostały podane inne wartości parametru. Po zdefiniowaniu parametru można się do niego odwoływać we właściwościach obiektów poprzez notację sufiksową *%nazwa* lub wywołanie funkcji *Parameter.*
- W trakcie wykonywania aplikacji w akcjach otwarcia diagramu *OpenWindow* i *OpenDiagram* należy podać wartości parametrów. Parametry są podawane jako sekwencja par *nazwa_parametru=wartość_parametru* rozdzielonych znakiem średnika, np. *Kolor=Red;Zmienna=v1*.

Inne możliwości ustawiania parametrów to podanie wartości parametrów dla diagramu domyślnego w ustawieniach panelu lub użycie akcji *SetParameter* i *SetParameters*.

Diagram posiada zdarzenia, które pozwalają na wykonanie dowolnych akcji operatorskich w momencie otwierania i zamykania diagramu.

4.1 Właściwości diagramów

Nazwa	*Typ wartości*	*Opis*
Nazwa diagramu	Nazwa	Nazwa identyfikująca diagram. Pierwszy znak musi być literą, a następne literą, cyfrą lub znakiem _.
Opis	Tekst	Opis diagramu o znaczeniu jedynie informacyjnym.
Kolor tła	Kolor	Kolor używany do wypełnienia tła diagramu
Obrazek tła	Nazwa obrazka	Nazwa obrazka, który zostanie użyty jako tło diagramu.
Tryb wypełniania obrazu tła	Typ wyliczeniowy: Oryginalny, Kafelki, Wyśrodkowany, Rozciąganie	Sposób wyświetlenia obrazka tła. Tryb oryginalny oznacza wyświetlenie obrazka w oryginalnych rozmiarach w lewym, górnym narożniku diagramu. W trybie kafelek obrazek jest wyświetlany wielokrotnie, tak aby wypełnić cały obszar diagramu. W trybie wyśrodkowania obrazek jest wyświetlany w oryginalnych rozmiarach na środku diagramu. W trybie rozciągania rozmiar obrazka jest dopasowywany do rozmiaru diagramu.
Stała szerokość	Liczba	Początkowa szerokość diagramu wyrażona w pikselach w trybie *stałych rozmiarów.*
Stała wysokość	Liczba	Początkowa wysokość diagramu wyrażona w pikselach w trybie *stałych rozmiarów.*
Stałe rozmiary	Logiczna	Określa, czy rozmiary diagramu są niezależne od rozmiaru panelu, w którym został wyświetlony. Wartość *Nie* oznacza, że diagram jest zawsze rozmiaru panelu – zmiana rozmiaru panelu powoduje przeskalowanie diagramu. Wartość *Tak* powoduje, że diagram jest wyświetlany zgodnie z rozmiarami podanymi we właściwościach *Stała wysokość* i *Stała* szerokość. Jeżeli rozmiar panelu

		jest mniejszy od rozmiaru diagramu, to zostaną użyte suwaki.
Zezwalaj na powiększanie	Logiczna	Określa, czy w trybie *stałych rozmiarów* użytkownik może przeskalować wielkość diagramu przy pomocy rolki myszki (przy wciśniętym przycisku *Control*).

4.2 Rejestry

Każdy diagram posiada także osobną grupę właściwości ogólnego przeznaczenia zwaną *rejestrami*. Rejestry nie mają z góry zdefiniowanego znaczenia. Przeznaczone są do centralnego wyliczania wyrażeń, których wartości są następnie wykorzystywane w innych elementach diagramu. Dostęp do wartości rejestrów wykonywany jest za pośrednictwem funkcji *Reg* i *RegStatus*.

Użycie rejestrów ma sens wtedy, gdy takie same wyrażenia są wykorzystywane w dużej liczbie miejsc. Upraszcza to konstrukcję diagramu oraz jest wydajność i skraca czas ładowania diagramów.

Standardowo dostępnych jest 10 rejestrów. Ich liczbę można jednak zwiększyć w panelu *Ustawienia globalne* w zakładce *Ustawienia*.

4.3 Zdarzenia diagramów

Nazwa	*Opis*
Otwarcie diagramu	Zdarzenie uruchamiane w momencie otwarcia diagramu. Typowy przykład użycia to utworzenie połączenia do AsBase'a przy pomocy akcji *AsbaseOpen*.
Zamknięcie diagramu	Zdarzenie uruchamiane w momencie zamknięcia diagramu. Typowy przykład użycia to zamknięcie połączenia do AsBase'a przy pomocy akcji *AsbaseClose*.

4.4 Właściwości grup

Nazwa	*Typ wartości*	*Opis*
Nazwa elementu	Nazwa	Nazwa identyfikująca grupę obiektów.
X	Liczba	Współrzędna X pozycji grupy obiektów wyrażona w pikselach, liczona względem lewego górnego narożnika diagramu.
Y	Liczba	Współrzędna Y pozycji grupy obiektów wyrażona w pikselach, liczona względem lewego górnego narożnika diagramu.
Widoczny	Logiczna	Określa, czy grupa obiektów jest widoczna.

4.5 Zdarzenia grup

Nazwa	*Opis*
Animacja	Zdarzenie uruchamiane okresowe. Przeznaczone dla celów animacji ruchu grupy obiektów.

4.6 Właściwości osadzonych wzorców

Nazwa	*Typ wartości*	*Opis*
Nazwa elementu	Nazwa	Nazwa identyfikująca osadzony wzorzec.
X	Liczba	Współrzędna X pozycji wzorca obiektów wyrażona w pikselach, liczona względem lewego górnego narożnika diagramu.
Y	Liczba	Współrzędna Y pozycji wzorca wyrażona w pikselach, liczona względem lewego górnego narożnika diagramu.
Szerokość	Liczba	Szerokość obszaru osadzenia wzorca
Wysokość	Liczba	Wysokość obszaru osadzenia wzorca

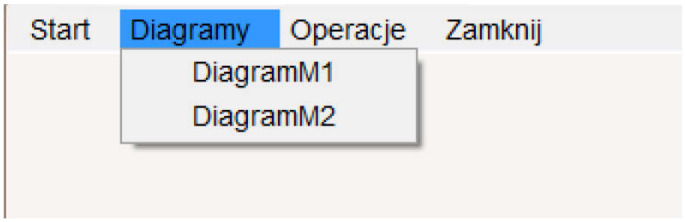
Warstwa	Liczba	Numer warstwy bazowej, względem której wyliczany jest numer warstwy obiektów składowych wzorca - numer warstwy wzorca jest dodawany do numeru warstwy obiektu. Obiekty o większym numerze warstwy przesłaniają obiekty o numerze mniejszym. Sposób wyświetlania obiektów o tym samym numerze warstwy wynika z kolejności ich zdefiniowania.
Dana pomocnicza	Logiczna	Właściwość nie ma określonego znaczenia. Może być użyta przez projektanta diagramu do dowolnych celów. Do wartości danej pomocniczej można się odwołać w wyrażeniach parametrów osadzenia wzorca za pomocą konstrukcji *LocalProperty(CustomData)*.
Widoczny	Logiczna	Określa czy grupa obiektów jest widoczna.

4.7 Zdarzenia osadzonych wzorców

Nazwa	*Opis*
Animacja	Zdarzenie uruchamiane okresowe. Przeznaczone dla celów animacji ruchu wzorca.

5 Menu

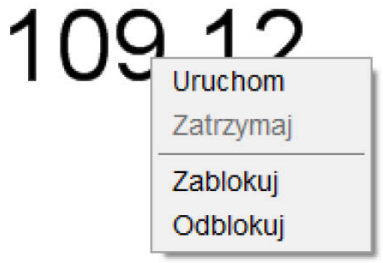
W aplikacjach Asix.Evo można stosować dwa rodzaje menu. Pierwszy rodzaj to menu powiązane z oknem. Można swobodnie definiować strukturę i wygląd menu. Menu podłączane jest do okna poprzez właściwość *Menu* okna. Właściwość *Dokowanie* określa sposób podłączenia menu do okna.





Rys. Przykład Menu powiązanego z oknem aplikacji.

Drugi rodzaj menu, to menu kontekstowe, z reguły podłączone do obiektów. Menu kontekstowe otwierane jest wyniku wykonania akcji operatorskiej *ShowMenu*.



Rys. Przykład Menu kontekstowego powiązanego z obiektem.

5.1 Parametry i kontrola elementów menu

Menu kontekstowe może być menu parametryzowanym. Pozwala to używać tego samego menu w wielu miejscach i uzależniać jego działanie od parametrów przekazanych do menu. Ogólne zasady użycia parametrów są identyczne jak dla parametrów diagramów.

W przypadku menu kontekstowego, parametry są do niego przekazywane w momencie wykonania akcji otwarcia menu *ShowMenu*.

W przypadku menu powiązanego z oknem, przekazanie parametrów odbywa się w momencie każdorazowego wykonania akcji otwarcia diagramu w oknie *OpenWindow*. Parametry podane w akcji są w pierwszym kroku przekazywane do diagramu, ale jeżeli zachodzi zgodność nazwy dowolnego parametru z nazwą parametru menu powiązanego z oknem, to jest on również przekazywany do menu.

Wygląd menu może być uzależniany od stanu aplikacji. W szczególności właściwość Widoczny elementu menu pozwala na ukrywanie zbędnych elementów, a właściwość *Aktywny* kontroluje możliwość wykonania akcji powiązanej z elementem menu. Działanie menu można uzależnić od uprawnień użytkownika:

```
HasRole(superoperator)
```

lub od stanu procesu:

```
Variable(v1)==0
```

5.2 Właściwości menu

Nazwa	*Typ wartości*	*Opis*
Nazwa menu	Nazwa	Nazwa identyfikująca menu. Pierwszy znak musi być literą, a następne literą, cyfrą lub znakiem _.
Kolor czcionki	Kolor	Kolor używany do wyświetlania tekstów w elementach menu.
Kolor tła	Kolor	Kolor tła elementów menu.
Kolor krawędzi	Kolor	Kolor obwódki wokół elementów menu.
Kolor tła podświetlenia	Kolor	Kolor tła elementów menu w momencie ich wybrania.
Kolor krawędzi podświetlenia	Kolor	Kolor obwódki wokół elementów menu w momencie ich wybrania.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania elementów menu.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Rozmiar czcionki	Liczba	Określa rozmiar używanej czcionki.
Dokowanie	Typ wyliczeniowy: Brak, Do góry, Do dołu, Do lewej, Do prawej	Określa przy której krawędzi okna należy umieścić menu. W przypadku menu kontekstowego obiektów należy stosować tryb *Brak*.

5.3 Właściwości elementów menu

Nazwa	*Typ wartości*	*Opis*
Tekst	Tekst	Tekst wyświetlany w pozycji menu
Nazwa obrazu	Nazwa obrazka	Nazwa obrazka wyświetlanego obok tekstu w pozycji menu.
Umieść po prawej	Logiczna	Określa, czy pozycja menu ma być wyświetlona po prawej stronie menu (u dołu dla menu dokowanych do pionowej krawędzi okna). Właściwość ma znaczenie tylko dla elementów pierwszego poziomu w menu powiązanym z oknem.
Widoczny	Logiczna	Określa, czy element menu ma być widoczny. Z reguły opcja jest wyliczana wyrażeniem uwzględniającym uprawnienia zalogowanego użytkownika lub wartości zmiennych procesowych.
Aktywny	Logiczna	Określa, czy element menu ma być aktywny. Elementy nieaktywne są wyświetlane wyszarzonym kolorem i nie wykonują akcji operatorskich. Z reguły opcja jest wyliczana wyrażeniem uwzględniającym uprawnienia zalogowanego użytkownika lub wartości zmiennych procesowych.
Separator	Logiczna	Określa, czy pozycja menu jest separatorem. Jeżeli tak, to w miejscu pozycji wyświetlana jest tylko linia separująca pozostałe pozycje w menu.
Akcja	Akcja operatorska	Akcja operatorska wykonywana w reakcji na wybranie elementu menu przez użytkownika.

6 Obiekty

Obiekty są podstawowym elementem wizualizacyjnym. Są umieszczane na diagramach i we wzorcach. Wygląd i funkcje obiektów zależą od klasy obiektów. Dostępne są poniższe klasy obiektów:

- Dziennik zdarzeń - wyświetla tabela komunikatów systemowych
- Klapa - wyświetlenie symbolu klapy z sygnalizacją stopnia otwarcia

- Klawiatura - klawiatura ekranowa
- Komunikaty - wyświetlenie tekstu pobranego z pliku definicji komunikatów
- Kontroler obrazu wektorowego - kontrola wyglądu wyświetlanych obrazów w formacie wektorowym SVG
- Kontroler wykresu - obiekt służący do sterowania pracą obiektów klasy *Wykres*
- Kształt - wyświetlenie kształtów na bazie prostokąta i elipsy
- Linia - wyświetlenie linii, również złożonych z wielu odcinków
- Miernik - wyświetlenie miernika wskazówkowego
- Obraz - wyświetlenie obrazka graficznego
- Pole autoryzacji - kontrolka do logowania użytkownika
- Przeglądarka - wyświetlenie stron www i zawartości dokumentów
- Przycisk - wyświetlenie przycisków w różnych stylach
- Punkt pracy - pokazuje położenie punktu w funkcji dwóch zmiennych określających współrzędne X i Y
- Rurociąg - obiekt w kształcie rurociągu
- Słupek - wyświetla dowolną wartość w postaci słupka o opcjonalnym suwakiem sterującym
- Tabela alarmów aktywnych - wyświetla tabelę pokazującego aktywne alarmy
- Tabela alarmów historycznych - wyświetla tabelę do przeglądania alarmów historycznych
- Tabela Asbase - wyświetla tabelę do przeglądania danych pochodzących z bazy danych modułu Asbase
- Tabela zmiennych - wyświetla wartości zmiennych procesowych w postaci tabelarycznej
- Taśmociąg - obiekt w kształcie taśmociągu
- Tekst - wyświetla informacje w postaci tekstowej
- Wielokąt - wyświetla kształt dowolnego wielokąta
- Wykres - wyświetla przebiegi zarchiwizowanych zmiennych lub krzywych wzorcowych
- Zbiornik - obiekt w kształcie zbiornika

Istnieje możliwość dodania do aplikacji własnych klas obiektów poprzez dołączenie odpowiednio napisanych modułów programowych.

Podstawowa funkcjonalność obiektu wynika z jego klasy. Projektant aplikacji ma jednak dużą swobodę dotycząca szczegółów działania obiektów. Zdecydowana większość właściwości obiektów może być dynamicznie zmienna w trakcie pracy aplikacji. Pozwala to na uzyskiwanie praktycznie dowolnych efektów animacyjnych.

Niektóre klasy obiektów posiadają wbudowane funkcje interaktywne (np. obiekty *Tekst*, *Słupek*, *Przycisk*, tabele alarmów). W przypadku obiektów nieinteraktywnych (np. Linia, Kształt, Rurociąg) projektant może dodać takie funkcje poprzez obsługę zdarzeń. Na przykład, w każdym obiekcie można obsłużyć zdarzenie kliknięcia myszką i wykonać w takim momencie dowolną akcję operatorską.

Klasy obiektów dzielą się na dwa typy:

- Graficzne - typ podstawowy
- Kontrolkowe - obiekty oparte na wykorzystaniu standardowych kontrolek systemowych. Obiektami kontrolkowymi są *Dziennik Zdarzeń*, *Kontroler wykresu*, *Pole autoryzacji*, *Przeglądarka, Tabela alarmów aktywnych*, *Tabela alarmów historycznych, Tabela zmiennych*, *Wykres.*

Podstawowym ograniczeniem obiektów kontrolkowych jest brak możliwości nakładania się obiektów na siebie. Obiekty graficzne mogą być umieszczane w tym samym obszarze z kontrolą wzajemnego przesłaniania.

Obiekty są parametryzowane poprzez zestawy właściwości i zdarzeń. Istnieją zbiory tzw. standardowych właściwości i zdarzeń, które są obecne praktycznie we wszystkich klasach. Każda klasa może wprowadzać dodatkowe właściwości i zdarzenia dla niej specyficzne.

6.1 Grupy właściwości

Właściwości obiektów są zawsze umieszczone w jakiejś grupie. Istnieją trzy typy grup właściwości opisane w poniższych rozdziałach.

6.1.1 Właściwości podstawowe

Grupa właściwości występujących jednokrotnie. Najważniejsza właściwość w tej grupie to *Główna zmienna*. Ustala ona kontekst obiektu. Przy poprawnej definicji obiektu (użyciu funkcji w wersjach kontekstowych, np. *Variable(),* lub notacji sufiksowej nazw zmiennych) właściwość pozwala na

tworzenie obiektów, które po skopiowaniu można przełączyć na pokazywanie innych danych poprzez zmianę tylko nazwy zmiennej głównej.

Wartości właściwości mogą ulegać dynamicznej zmianie poprzez użycie w ich definicjach wyrażeń i odwołań do elementów, których wartość ulega zmianie.

Przykład:

Właściwość: *Aktywny*

Definicja: = HasRole(DIR)

Status aktywności obiektu zmienia się w zależności od zalogowanego użytkownika.

Właściwość: *Dymek*

Definicja: @Description

Tekst wyświetlany w dymku będzie opisem zmiennej głównej obiektu

Właściwość: *Wartość* obiektu *Miernik*

Definicja: #

Położenie wskazówki obiektu *Miernik* będzie zależało od wartości zmiennej głównej obiektu.

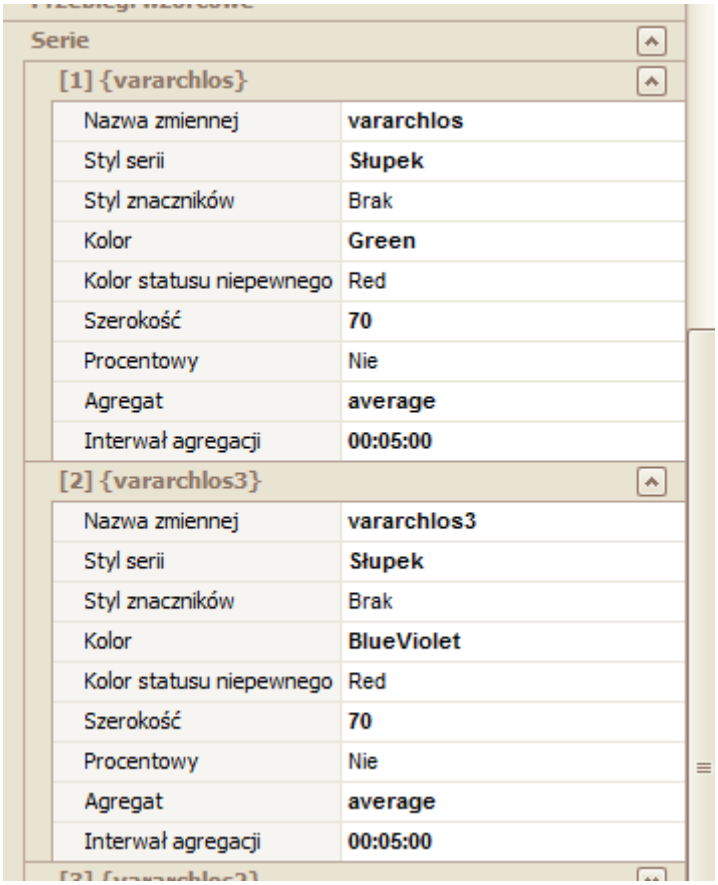
Właściwość: *Warstwa*

Definicja: =Variable(r1)>0?10:0

Jeżeli wartość zmiennej *r1* jest większa od 0, to numer warstwy ustawiany jest na 10. W przeciwnym wypadku jest równy 0. Powyższe wyrażenie pozwala dynamicznie kontrolować przesłanianie się obiektów.

6.1.2 Grupy wielokrotne

Grupy wielokrotne właściwości występują tylko w niektórych klasach obiektów, np. *Wykres*, *Tabela zmiennych.* Używane są wszędzie tam, gdzie w obiekcie tworzone są pewne powtarzalne elementy. Klasycznym przykładem są definicje wyświetlanych krzywych w obiekcie *Wykres*.



Rys. Grupy wielokrotne właściwości obiektu.

Projektant może utworzyć tylko tyle grup właściwości, ile jest potrzebnych. Każda grupa jest niezależna od pozostałych. Zasady wyliczania wartości są identyczne jak dla właściwości z grupy podstawowej.

6.1.3 Właściwości stanowe

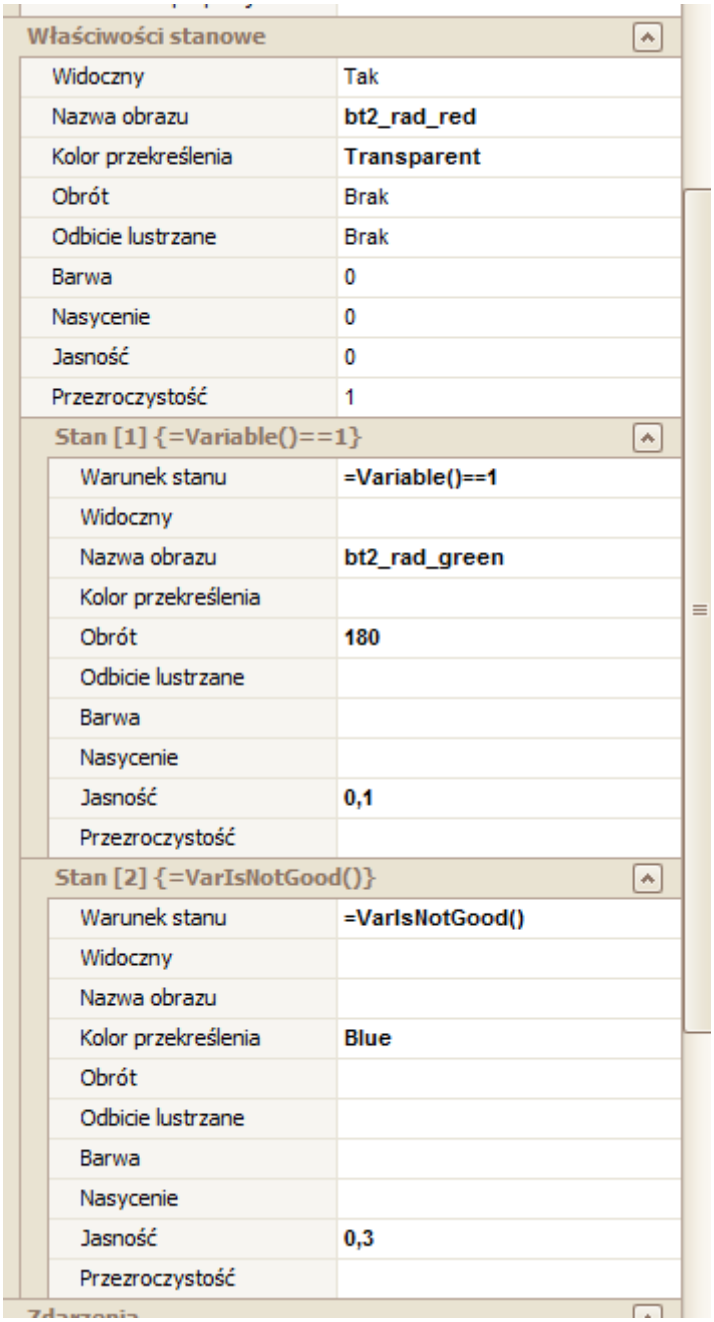
Właściwości stanowe to specjalny rodzaj właściwości pozwalający na łatwe tworzenie obiektów zmieniających swój wygląd w zależności od stanu aplikacji i kontrolowanego procesu. Możliwe jest uzyskanie podobnych efektów bez wykorzystania grup stanowych, ale wymaga to użycia znacznie bardziej skomplikowanych wyrażeń.

Idea działania właściwości stanowych polega na tworzeniu wielu grup identycznych właściwości. W każdej z tych grup, właściwości mogą być definiowane w różny sposób. Oprócz tego grupa posiada dodatkową właściwość *Warunek stanu* typu logicznego, która określa, czy grupa brana jest pod uwagę przy wyliczaniu efektywnej wartości właściwości użytej do wyświetlenia obiektu.

Algorytm wyliczania wartości właściwości jest następujący:

- Wyszukiwana jest grupa, której warunek stanu ma wartość *Tak,* a definicja właściwości nie jest pusta.
- Jeżeli jest tylko jedna taka grupa, to wartość właściwości wynika z definicji użytej w tej grupie.
- Jeżeli jest więcej takich grup, to wartość właściwości pobierana jest z definicji ostatniej takiej grupy.
- Jeżeli żadna grupa nie jest aktywna, lub w żadnej z aktywnych grup właściwość nie jest zdefiniowana, to wartość określana jest na podstawie definicji w grupie podstawowej (pierwszej).

Przykład:



Właściwości stanowe	
Widoczny	Tak
Nazwa obrazu	**bt2_rad_red**
Kolor przekreślenia	**Transparent**
Obrót	Brak
Odbicie lustrzane	Brak
Barwa	0
Nasycenie	0
Jasność	0
Przezroczystość	1
Stan [1] {=Variable()==1}	
Warunek stanu	**=Variable()==1**
Widoczny	
Nazwa obrazu	**bt2_rad_green**
Kolor przekreślenia	
Obrót	**180**
Odbicie lustrzane	
Barwa	
Nasycenie	
Jasność	**0,1**
Przezroczystość	
Stan [2] {=VarIsNotGood()}	
Warunek stanu	**=VarIsNotGood()**
Widoczny	
Nazwa obrazu	
Kolor przekreślenia	**Blue**
Obrót	
Odbicie lustrzane	
Barwa	
Nasycenie	
Jasność	**0,3**
Przezroczystość	

Rys. Przykład właściwości stanowych.

W pokazanym fragmencie definicji obiektu klasy *Obraz* pokazane jest, w jaki sposób wygląd obiektu uzależniony jest od wartości zmiennej oraz jej statusu. Grupa stanowa numer 1 aktywna jest wtedy, gdy wartość zmiennej głównej obiektu jest równa 0. Grupa stanowa numer 2 aktywna jest wtedy, gdy status zmiennej głównej jest niepoprawny. Sposób określenia wartości właściwości jest następujący:

Wartość zmiennej głównej jest różna od 1, a status zmiennej jest poprawny	Wartości wszystkich właściwości są pobierane z grupy podstawowej.
Wartość zmiennej głównej jest równa 1, status zmiennej jest poprawny	Wyświetlany obraz to bt2_rad_green, obraz jest obrócony o 180 stopni i ma zmienioną jasność ze współczynnikiem 0,1. Pozostałe ustawienia są pobierane z grupy podstawowej.
Wartość zmiennej głównej jest różna od 1, status zmiennej jest niepoprawny	Obrazek jest przekreślony i ma zmienioną jasność ze współczynnikiem 0,3. Pozostałe ustawienia są pobierane z grupy podstawowej, w szczególności wyświetlony obraz to bt2_rad_red.
Wartość zmiennej głównej jest równa 1, status zmiennej jest niepoprawny	Wyświetlany obraz to bt2_rad_green, obraz jest obrócony o 180 stopni, przekreślony i ma zmienioną jasność ze współczynnikiem 0,3. Współczynnik jasności pochodzi z grupy kontrolującej status zmiennej ponieważ jest ona zdefiniowana później.

6.2 Standardowe właściwości obiektów

Nazwa	*Typ wartości*	*Opis*
Nazwa elementu	Nazwa	Nazwa osadzonego obiektu. Pozwala zidentyfikować obiekt i zmieniać jego właściwości przy pomocy akcji *SetProperty*.
X	Liczba	Współrzędna X pozycji obiektu wyrażona w pikselach, liczona względem lewego górnego narożnika diagramu.
Y	Liczba	Współrzędna Y pozycji obiektu wyrażona w pikselach, liczona względem lewego górnego narożnika diagramu.
Szerokość	Liczba	Szerokość obszaru osadzenia obiektu wyrażona w pikselach.
Wysokość	Liczba	Wysokość obszaru osadzenia obiektu wyrażona w pikselach.
Warstwa	Liczba	Numer warstwy, na której wyświetlany jest obiekt. Obiekty o większym numerze

		warstwy przesłaniają obiekty o numerze mniejszym. Sposób wyświetlania obiektów o tym samym numerze warstwy wynika z kolejności ich zdefiniowania.
Główna zmienna	Nazwa zmiennej	Nazwa zmiennej głównej obiektu. Właściwość pozwalająca na centralne podanie w obiekcie nazwy zmiennej, od wartości i definicji której uzależnione jest działanie obiektu. W innych właściwościach obiektu można odwoływać się do zmiennej głównej przez kontekstowe funkcje typu *Variable* i *Attribute*, skrótowe odwołania do nazwy i wartości zmiennej głównej w postaci *#*, lub konstruować pochodne nazwy zmiennych stosując notację sufiksową *#sufiks*.
Zmienna sterowana	Nazwa zmiennej	Nazwa zmiennej, która jest zapisywana w efekcie użycia funkcji sterujących obiektu (dotyczy obiektów z wbudowanymi funkcjami sterującymi, np. *Tekst*). Przykład: # Zmienna sterowana jest identyczna ze zmienną główną obiektu. Jest to jedyny przypadek, gdy użycie notacji *#* oznacza nazwę zmiennej głównej, a nie jej wartość. #_s Nazwa zmiennej sterowanej powstaje przez połączenie nazwy zmiennej głównej i sufiksu *_s*.

		@ControlVariable Nazwa zmiennej sterowanej pobierana z atrybutu zmiennej głównej.
Dana pomocnicza	Logiczna	Właściwość nie ma określonego znaczenia. Może być użyta przez projektanta diagramu do dowolnych celów. Można się odwołać do wartości danej pomocniczej poprzez wyrażenie *LocalProperty(CustomData)*. Przykładowym zastosowaniem jest jednokrotne wyliczanie wyrażenia, które należałoby użyć w wielu innych miejscach.
Aktywny	Logiczna	Określa, czy obiekt jest aktywny. W przypadku obiektów z wbudowanymi funkcjami sterującymi wartość *False* blokuje tą funkcjonalność. Dla obiektów bez standardowych funkcji interaktywnych (np. *Kształt*) wartość właściwości nie ma bezpośredniego znaczenia, może być jednak kontrolowana w akcjach i funkcjach użytych we właściwościach i zdarzeniach obiektu – stan aktywności można pobrać funkcją *IsActive*.
Tło obiektu aktywne	Logiczna	Określa w jaki sposób obiekt reaguje na kliknięcia myszką w trybie działania. Jeżeli wartość jest równa Nie, to kliknięcia we fragmentach pustych (wyświetlanych kolorem *Transparent*) są ignorowane. Pozwala to na przykład, na użycie częściowo przezroczystego obrazka, który reaguje na kliknięcia tylko w obszarze nieprzezroczystym. Właściwość nie ma znaczenia dla obiektów typu kontrolkowego, np. *Przeglądarka*.
Minimalna szerokość widoczności	Liczba	Minimalna szerokość obiektu w pikselach, przy której obiekt jest wyświetlany. Jeżeli w efekcie przeskalowania diagramu szerokość obiektu spadnie poniżej

		wartości granicznej, to obiekt jest automatycznie ukrywany.
Minimalna wysokość widoczności	Liczba	Minimalna wysokość obiektu w pikselach, przy której obiekt jest wyświetlany. Jeżeli w efekcie przeskalowania diagramu wysokość obiektu spadnie poniżej wartości granicznej, to obiekt jest automatycznie ukrywany.
Dymek	Tekst	Definicja tekstu wyświetlanego w dymku podpowiedzi. Treść dymku może zależeć od wartość atrybutów z bazy zmiennych lub wartości zmiennych procesowych. Przykład: @Description Wyświetla opis zmiennej głównej obiektu =Format(”{0} = {1}”, Attribute(Name),Variable()) Wyświetla nazwę i wartość zmiennej głównej obiektu.
Typ kursora	Nazwa kursora	Systemowa nazwa typu kursora używanego w momencie najechania kursorem myszki na obiekt. Można w ten sposób sygnalizować gotowość obiektu na wykonanie operacji interaktywnych, np. wykonanie akcji po kliknięciu klawiszem myszki.
Blokada położenia	Logiczna	Określa, czy możliwa jest zmiana pozycji obiektu. Dotyczy tylko trybu edycji diagramu.
Blokada rozmiaru	Logiczna	Określa, czy możliwa jest zmiana rozmiaru obiektu. Dotyczy tylko trybu edycji diagramu.

Warunek stanu	Logiczna	Właściwość w grupie właściwości stanowych decydująca o tym, czy ustawienia grupy należy uwzględniać przy wyliczaniu efektywnych wartości właściwości stanowych.
Opis stanu	Tekst	Określa tekstowy opis grupy właściwości stanowych.
Widoczny	Logiczna	Określa, czy obiekt ma być wyświetlany. W trybie edycyjnym wartość właściwości jest ignorowana – obiekt jest zawsze widoczny.
Przezroczystość	Liczba <0,1>	Określa stopień przezroczystości obiektu w zakresie od 0 do 1. Wartość 1 oznacza obiekt nieprzezroczysty, wartość 0 oznacza obiekt całkowicie przezroczysty.

6.3 Standardowe zdarzenia obiektów

Nazwa	*Opis*
Animacja	Zdarzenie uruchamiane okresowe. Przeznaczone dla celów animacji ruchu obiektów.
Wysłanie sterowania	Zdarzenie uruchomiane w momencie wykonania akcji operatorskiej *SendControls*. Dotyczy tylko obiektów aktywnych.
Anulowanie sterowania	Zdarzenie uruchomiane w momencie wykonania akcji operatorskiej *CancelControls*. Dotyczy tylko obiektów aktywnych.
Wciśnięcie lewego przycisku	Zdarzenie uruchomiane w momencie wciśnięcie lewego klawisza myszki, gdy kursor znajduje się w obszarze obiektu.
Wciśnięcie prawego przycisku	Zdarzenie uruchomiane w momencie wciśnięcie prawego klawisza myszki, gdy kursor znajduje się w obszarze obiektu.
Wciśnięcie środkowego przycisku	Zdarzenie uruchomiane w momencie wciśnięcie środkowego klawisza myszki, gdy kursor znajduje się w obszarze obiektu.
Zwolnienie lewego przycisku	Zdarzenie uruchomiane w momencie zwolnienia lewego klawisza myszki, gdy kursor znajduje się w obszarze obiektu.

Zwolnienie prawego przycisku	Zdarzenie uruchomiane w momencie zwolnienia prawego klawisza myszki, gdy kursor znajduje się w obszarze obiektu.
Zwolnienie środkowego przycisku	Zdarzenie uruchomiane w momencie zwolnienia środkowego klawisza myszki, gdy kursor znajduje się w obszarze obiektu.
Kliknięcie lewym przyciskiem	Zdarzenie uruchomiane w momencie kliknięcia (przyciśnięcie i zwolnienie) lewym klawiszem myszki, gdy kursor znajduje się w obszarze obiektu.
Kliknięcie prawym przyciskiem	Zdarzenie uruchomiane w momencie kliknięcia prawym klawiszem myszki, gdy kursor znajduje się w obszarze obiektu.
Kliknięcie środkowym przyciskiem	Zdarzenie uruchomiane w momencie kliknięcia środkowym klawiszem myszki, gdy kursor znajduje się w obszarze obiektu.
Podwójne kliknięcie lewym przyciskiem	Zdarzenie uruchomiane w momencie podwójnego kliknięcia lewym klawiszem myszki, gdy kursor znajduje się w obszarze obiektu.
Podwójne kliknięcie prawym przyciskiem	Zdarzenie uruchomiane w momencie podwójnego kliknięcia prawym klawiszem myszki, gdy kursor znajduje się w obszarze obiektu.
Podwójne kliknięcie środkowym przyciskiem	Zdarzenie uruchomiane w momencie podwójnego kliknięcia środkowym klawiszem myszki, gdy kursor znajduje się w obszarze obiektu.
Trzymanie lewego przycisku	Zdarzenie uruchamiane okresowo, gdy kursor myszki znajduje się w obszarze obiektu i przyciśnięty jest lewy klawisz myszki.
Trzymanie prawego przycisku	Zdarzenie uruchamiane okresowo, gdy kursor myszki znajduje się w obszarze obiektu i przyciśnięty jest prawy klawisz myszki.
Trzymanie środkowego przycisku	Zdarzenie uruchamiane okresowo, gdy kursor myszki znajduje się w obszarze obiektu i przyciśnięty jest środkowy klawisz myszki.
Najechanie kursorem	Zdarzenie uruchamiane, gdy kursor myszki został przesunięty w obszar obiektu.
Wyjechanie kursorem	Zdarzenie uruchamiane, gdy kursor myszki został przesunięty poza obszar obiektu.
Ruch kursora	Zdarzenie uruchamiane, gdy kursor myszki przemieszcza się w ramach obszaru obiektu.
Wciśnięcie klawisza klawiatury	Zdarzenie uruchamiane, gdy zostanie przyciśnięty przycisk na klawiaturze i obiekt jest wyselekcjonowany. Jaki klawisz został przyciśnięty można sprawdzić przy pomocy funkcji *LastKeyPressed*.

	Przykład: Perform (LastKeyPressed()==q, SetVariable (v1, 1), Nothing()) Zmienna v1 jest ustawiana na 1, jeżeli został przyciśnięty klawisz *q*.
Wyselekcjonowanie obiektu	Zdarzenie uruchamiane w momencie wyselekcjonowania obiektu. Dotyczy tylko obiektów aktywnych.

7 Wzorce

Wzorce to wstępnie sparametryzowane grupy obiektów. Wzorce identyfikowane są przez nazwę. Po utworzeniu wzorzec może być wielokrotnie osadzany na diagramach. Jeżeli definicja dowolnego elementu wzorca ulegnie zmianie, to automatycznie wpływa to na wszystkie diagramy używające wzorca.

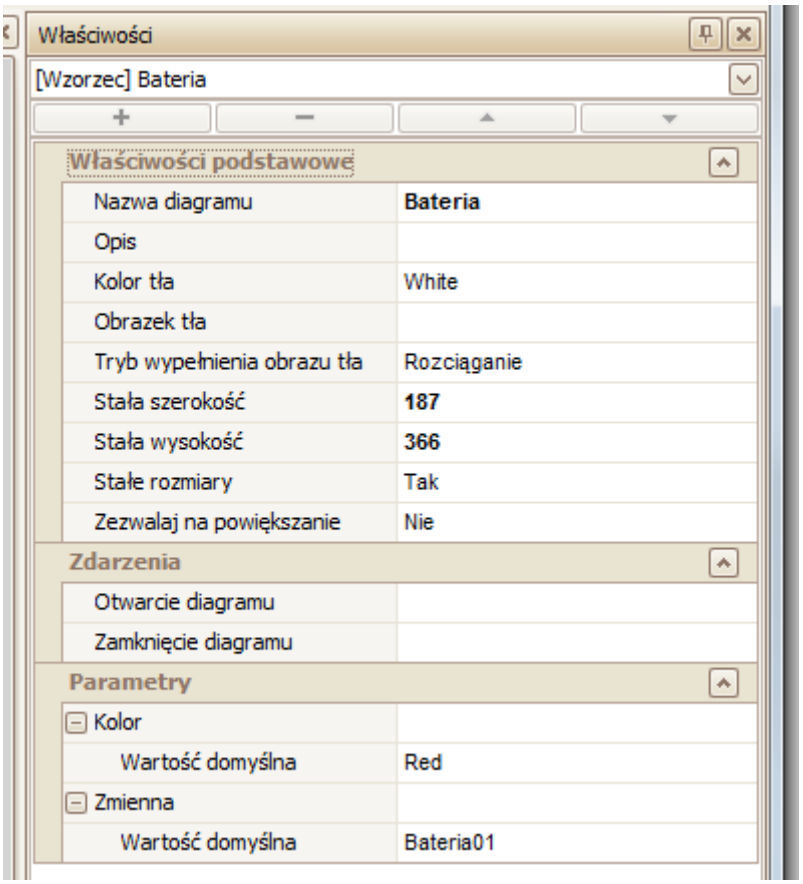
Wzorzec może składać się także z pojedynczego obiektu. Pozwala to przygotować obiekty, które realizują pewien schemat działania. Zmiana tego schematu nie wymaga zmiany diagramów.

Osadzony wzorzec może zostać przekształcony na zwykłe obiekty. Powoduje to utworzenie na diagramie kopii wszystkich obiektów wzorca. Od tego momentu obiekty pochodzące z wzorca traktowane są jak każdy inny obiekt diagramu.

Wzorce mają identyczny zestaw właściwości i zdarzeń jak diagramy.

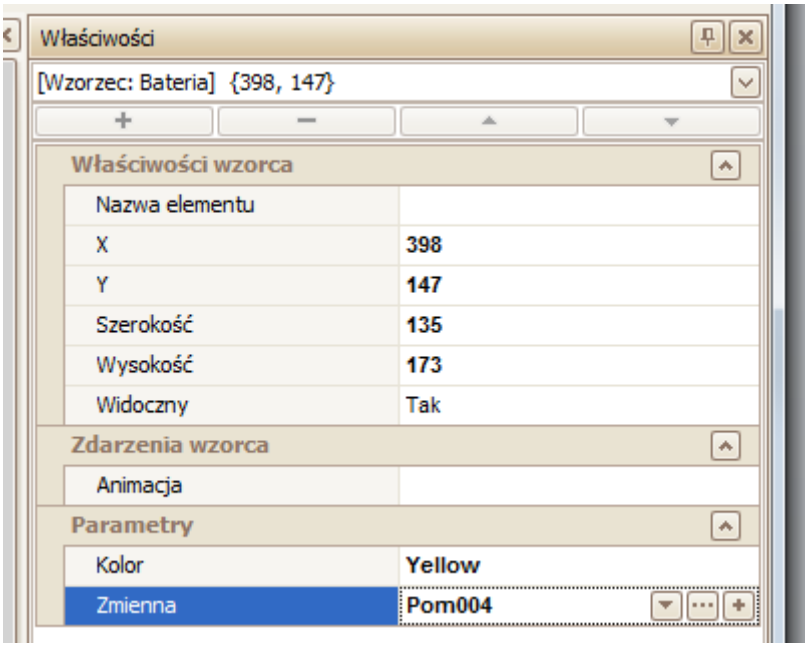
Kluczowym elementem definicji wzorca są parametry. Pozwala to wielokrotnie osadzać wzorzec z podaniem danych, na których wzorzec będzie pracować, w szczególności nazwy zmiennych procesowych.

Na etapie definicji wzorca definiowane są parametry wzorca. Definicja składa się z nazwy parametru oraz jego wartości domyślnej. Wartość domyślna używana jest w trakcie edycji wzorca oraz wtedy, gdy w miejscu osadzenia wzorca nie zostaną podane inne wartości parametru. Po zdefiniowaniu parametru można się do niego odwoływać we właściwościach obiektów składowych wzorca poprzez notację sufiksową *%nazwa* lub wywołanie funkcji *Parameter.*



Rys. Podstawowe właściwości wzorca.

W miejscu osadzenia wzorca należy podać właściwe wartości parametrów.

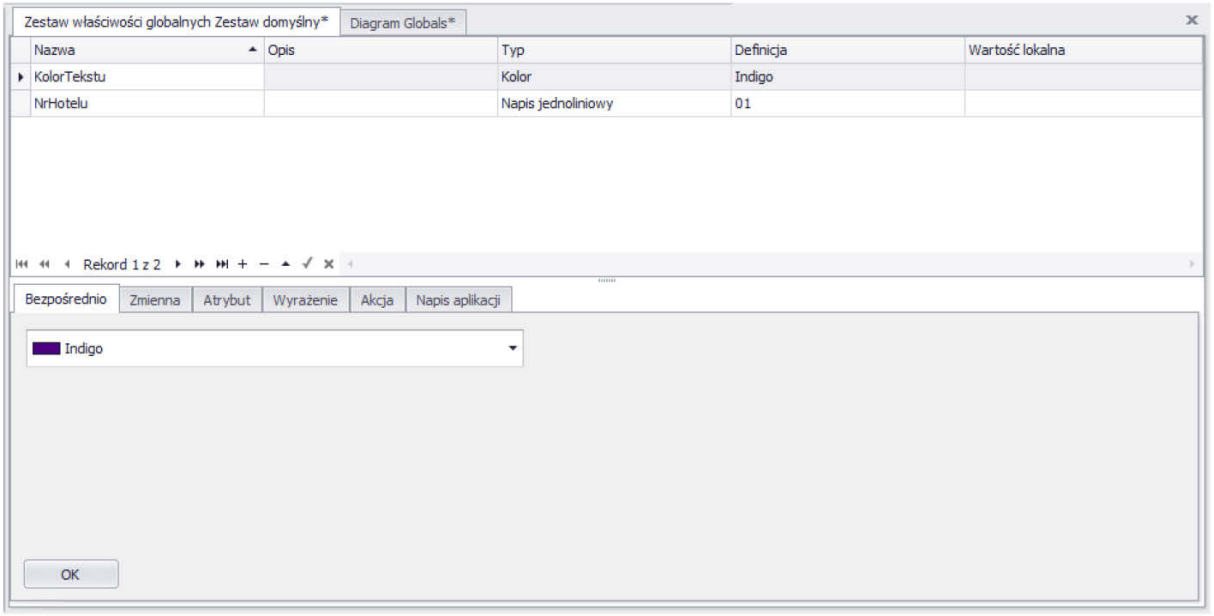


Rys. Właściwości wzorca.

8 Właściwości globalne

Właściwości globalne to centralnie definiowane właściwości niepowiązane z żadnym konkretnym elementem aplikacji. Projektant aplikacji może odwołać się wielokrotnie do właściwości globalnej w definicjach różnych elementów, np. we właściwościach obiektów. Stwarza to możliwość łatwej modyfikacji wyglądu lub zachowania aplikacji bez konieczności zmiany definicji właściwości wielu elementów składowych. Wartości właściwości globalnych mogą być modyfikowane w trybie wykonywania aplikacji - pozwala to na stworzenie w aplikacji mechanizmów pozwalających użytkownikom na zmianę sposobu jej działania.

Właściwości globalne definiowane są w panelu właściwości.



Rys. Panel właściwości globalnych.

Właściwość globalna jest definiowana poprzez jej nazwę, opcjonalny opis, typ właściwości i definicję wartości. Typ ma znaczenie tylko pomocnicze i służy do wyboru odpowiedniego edytora trybu bezpośredniego. Rzeczywista interpretacja definicji wartości właściwości zależy od miejsca jej użycia. Jeżeli przykładowa właściwość *KolorTekstu* zostanie użyta w obiekcie *Tekst* we właściwości *Kolor*, to spowoduje zmianę koloru wyświetlanego napisu. Jeżeli zostanie użyta we właściwości *Tekst*, to spowoduje wyświetlenie tekstu *Indigo*.

Kolumna *Wartość lokalna* jest kolumną tylko do odczytu. Pokazuje ona aktualną wartość właściwości wtedy, gdy została ona zmieniona w trybie wykonywania aplikacji przy pomocy akcji *SetGlobalProperty*.

Odwołania do właściwości globalnej wykonywane są przy pomocy funkcji *Property* lub notacji prefiksowej *!nazwa_właściwości*.

Definicja wartości właściwości najczęściej podawana jest w sposób bezpośredni. Można użyć innych metod, np. odwołań do atrybutów zmiennych czy wyrażeń. W tych przypadkach wyliczenie wartości właściwości odbywa się w miejscu i kontekście jej użycia.

Przykład:

Właściwość *Limit* jest zdefiniowana jako:

=Attribute(LimitHi)*1.1

Jeżeli właściwość *Limit* zostanie użyta w obiekcie *Tekst* we właściwości *Tekst*, poprzez odwołanie !Li*mit*, to zostanie wyświetlona wartość limitu górnego zmiennej głównej obiektu pomnożona przez 1,1.

Należy zwrócić uwagę, że w tym przypadku użycie wyrażenia *=Property(Limit)* nie jest równoważne. Zostałaby wyświetlona wartość właściwości *Limit* w postaci tekstowej *Attribute(LimitHi)*1.1* . Poprawne użycie to wyrażenie postaci *=Evaluate(Property(Limit))*.

8.1 Zmiana wartości właściwości w trybie wykonywania aplikacji

Dostępne są dwa mechanizmy zmiany właściwości globalnej w trybie wykonywania aplikacji. Pierwszy z nich polega na zmianie definicji właściwości w komendzie startowej programu. Mechanizm ten pozwala np. uzależnić działanie aplikacji od miejsca jej uruchomienia.

Zmiana właściwości w komendzie startowej polega na umieszczeniu w jej treści deklaracji w postaci:

&nazwa_właściwości=definicja

Przykład:

AsixEvo.exe &Department=KJ NrHotelu=04 -as=Terminal -run "c:\asixapp\app1"

W powyższym przykładzie uruchamiana jest aplikacja na stanowisku *Terminal*. Dodatkowo właściwość globalna *Department* jest ustawiana na wartość *KJ*.

Drugim mechanizmem zmiany wartości właściwości jest wykorzystanie akcji operatorskiej *SetGlobalProperty*. Wszystkie zmiany wartości wykonane przy jej pomocy są zapamiętywane. Po restarcie aplikacji, wartość właściwości zostanie automatycznie ustawiona na podstawie ostatniego wykonania akcji *SetGlobalProperty*. Powrót do oryginalnej wartości właściwości zdefiniowanej w aplikacji wymaga wykonania akcji *ResetGlobalProperty*. Zmiany wykonane akcją *SetGlobalProperty* są ograniczone tylko do stanowiska, na którym akcja została wykonana.

Asix.Evo - Techniki budowy diagramów

1 Metody tworzenia uniwersalnych obiektów

W trakcie projektowania diagramów należy dążyć do tego, żeby możliwe było łatwe kopiowanie utworzonych obiektów, oraz żeby łatwo i szybko wprowadzać zmiany na poziomie całej aplikacji. Ten rozdział opisuje dostępne mechanizmy i techniki postępowania służące do osiągnięcia wspomnianego celu. W kolejnych rozdziałach techniki te będą konsekwentnie stosowane w opisywanych przykładach.

1.1 Sufiksowa notacja nazw zmiennych

Dobrze skonstruowany obiekt monitorujący stan zmiennej procesowej pozwala na wykonanie swojej kopii i przełączenie go na monitowanie innej zmiennej, poprzez zmianę tylko pojedynczej właściwości. Pierwszym z mechanizmów ułatwiających taką konstrukcję jest właściwość *Zmienna główna* i sufiksowa notacja nazw zmiennych.

Właściwości stanowe, grupa podstawowa		
	Tekst	=VarStringValue(V1)
	Kolor	=Variable(V1)>50?Red:Black

Właściwości podstawowe		
	Główna zmienna	V1
Właściwości stanowe, grupa podstawowa		
	Tekst	#
	Kolor	=Variable()>50?Red:Black

Powyższe fragmenty parametryzacji obiektu *Tekst* działają identycznie. W pierwszym przykładzie nazwa zmiennej jest użyta dwa razy. Zrobienie kopii i zmiana zmiennej wymagałaby ręcznej edycji dwóch właściwości. W drugim wariancie nazwa zmiennej użyta jest tylko raz. Użycie zmiennej głównej pozwala na zastosowanie bezparametrowej funkcji *Variable* oraz notacji skrótowej *#,* oznaczając odwołanie do sformatowanej wartości zmiennej głównej. Wykonanie kopii obiektu jest bardzo proste, co więcej będzie można też użyć funkcji edytora *Grupowa zamiana zmiennych*.

Właściwości podstawowe		
	Główna zmienna	V1
	Zmienna sterowana	#Control
Właściwości stanowe, grupa podstawowa		
	Tekst	=VarStringValue(”#Value”)

Kolejny przykład pokazuje zastosowanie notacji sufiksowej bazującej na nazwie zmiennej głównej. Znak # użyty w kontekście nazwy zmiennej oznacza użycie wprost nazwy zmiennej głównej obiektu. W połączeniu z dodatkowym tekstem powoduje utworzenie nazwy zmiennej będącej połączeniem

nazwy zmiennej głównej i podanego tekstu. W tym przykładzie nazwa zmiennej sterowanej to *V1Control*, a wyświetlany tekst, to wartość zmiennej *V1Value*. Ponieważ nigdzie nie ma bezpośredniego odwołania do zmiennej głównej, to *V1* nie musi być nawet istniejącą zmienną.

1.2 Użycie atrybutów zmiennych

Kolejny mechanizm pozwalający na tworzenie uniwersalnych obiektów to atrybuty zmiennych procesowych.

Właściwości podstawowe		
	Główna zmienna	V1
Właściwości stanowe, grupa podstawowa		
	Tekst	#
	Kolor	***=Variable()>Attribute(LimitHi)?Red:Black***

Z reguły każda zmienna będzie miała własny poziom limitu krytycznego. Podanie jego wartości wprost wymagałoby każdorazowej modyfikacji wyrażenia dla właściwości *Kolor*. Funkcja *Attribute* pobiera atrybut *LimitHi* zmiennej głównej obiektu. Po zmianie nazwy zmiennej głównej nowa wartość limitu zostanie pobrana automatycznie. Dodatkową korzyścią z użycia atrybutów zmiennej jest to, że późniejsza zmiana wartości limitów w bazie definicji zmiennych nie będzie wymagała modyfikacji diagramów.

W bazie definicji zmiennych można umieścić dowolne atrybuty z predefiniowanego zestawu. Można też dodawać własne atrybuty o dowolnym znaczeniu.

1.3 Właściwości globalne

Właściwości globalne definiowane są w osobnym panelu roboczym otwieranym za pośrednictwem panelu eksploratora. Właściwości globalne pozwalają na stworzenie zestawu parametrów, które definiują zachowanie i wygląd obiektów na wszystkich diagramach. Zmiana wartości właściwości globalnej jest natychmiast widoczna we wszystkich obiektach, które odwołują się do tej właściwości.

Właściwości stanowe, grupa podstawowa		
	Tekst	#
	Kolor	!NormalColor

W powyższym przykładzie kolor tekstu wyświetlany jest w kolorze podanym we właściwości globalnej *NormalColor*. Późniejsza zmiana kolorystyki nie będzie wymagała zmiany w obiekcie.

Do wartości globalnej można odwołać się również przy pomocy funkcji *Property*.

Właściwości globalne zazwyczaj definiowane są w sposób bezpośredni. Możliwe są również inne sposoby, np. użycie wyrażenia.

=Color(Variable(),0,0)

Powyższe wyrażenie użyte we właściwości globalnej typu kolor, wylicza wartość koloru na podstawie wartości zmiennej głównej obiektu, który do właściwości się odwołuje. W dowolnym momencie można zmienić algorytm przeliczania, bez konieczności zmian w obiektach.

Jeżeli właściwość globalna nie jest zdefiniowana bezpośrednio, to należy się do niej odwoływać wyłącznie przez notację !.

1.4 Wzorce i parametryzowane diagramy

Wzorce i parametryzowane diagramy pozawalają tworzyć całe, predefiniowane zestawy obiektów do wielokrotnego użycia. Wzorce mogą być osadzane na diagramach, natomiast diagramy stanowią niezależną całość. Poprzez właściwą akcję otwarcia, diagramy parametryzowane mogą być użyte do pokazywania różnych danych (okienka typu „stacyjka").

W obu tych mechanizmach projektant definiuje pewien zestaw parametrów, które sterują pracą obiektów w osadzonym wzorcu lub otwartym diagramie. Większość właściwości jest jednak zdefiniowana bezpośrednio we wzorcu. Późniejsza modyfikacja wzorca(diagramu) jest natychmiast uwzględniana w miejscu ich użycia.

2 Wyświetlenie wartości zmiennej procesowej w postaci tekstowej

Rozdział pokazuje w jaki sposób można wyświetlić wartość zmiennej procesowej w postaci tekstowej. Pokazane zostaną sposoby formatowania wyświetlanego tekstu oraz metody zmiany atrybutów wyświetlania w zależności od różnych warunków. W szczególności opisany zostanie sposób tworzenia obiektów wielostanowych.

W aplikacjach Asix.Evo do wyświetlania wszelkiego rodzaju tekstów, zarówno statycznych jak i dynamicznych, służy obiekt klasy *Tekst*.

Uwaga:

W przypadku potrzeby wyświetlenia wartości wielu zmiennych w postaci tabelarycznej można posłużyć się także obiektem klasy *Tabela zmiennych*.

2.1 Wyświetlenie sformatowanej wartości

Wariant A

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, grupa podstawowa		
	Tekst	#

Wariant B

Właściwości stanowe, grupa podstawowa		
	Tekst	&VA001

Oba powyższe warianty parametryzacji powodują wyświetlenie wartości zmiennej *VA001*. Wartość zmiennej zostanie skonwertowana na tekst na podstawie atrybutu *Format* zapisanego w bazie definicji zmiennych. Na przykład, jeżeli format będzie postaci *f2*, to wartość zmiennej będzie przedstawiona z dwoma cyframi po przecinku. Jeżeli format w bazie zmiennych nie będzie zdefiniowany, to konwersja na tekst zostanie wykonana w sposób standardowy, zależny od typu wartości.

W wariancie A nazwa zmiennej podana jest we właściwości *Główna zmienna*, a tekst został określony poprzez notację skrótową *#,* oznaczającą pobranie wartości zmiennej głównej obiektu w postaci tekstowej.

W wariancie B użyta została tylko notacja skrótowa *&nazwa_zmiennej. O*znacza ona pobranie wartości podanej zmiennej w postaci tekstowej. Z powodów, które zostaną podane później, zalecane jest stosowanie wzorca postępowania z wariantu A. Wariant B należy stosować tylko w najprostszych sposobach parametryzacji.

Wariant C

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	=VarStringValue()

Wariant jest identyczny w działaniu z wariantem A. Zamiast notacji skrótowej użyta została funkcja *VarStringValue,* która zwraca wartość zmiennej głównej obiektu w postaci tekstowej.

Wariant D

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	=Variable()

Wariant D różni się od poprzedniego użyciem funkcji *Variable*. Funkcja ta zwraca wartość zmiennej głównej w postaci liczbowej. Konwersja na tekst zostanie wykonana w sposób automatyczny, atrybut *Format* nie będzie miał znaczenia.

Wariant E

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	=”Temperatura = ” + VarStringValue()

Wyświetlony tekst będzie składał się z połączenia tekstu *Temperatura =* oraz sformatowanej wartości zmiennej *VA001*.

Wariant F

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	=Format("Temperatura = {0} \u00b0C", VarStringValue())

Sformatowana wartość zmiennej *VA001* zostanie wstawiona do środka tekstu *Temperatura = °C,* w miejscu znacznika *{0}*. Funkcja *Format* pozwala w dowolny sposób tworzyć napisy na podstawie tekstu formatującego. W treści tekstu formatującego został pokazany sposób wstawiania do napisu znaku podanego w kodzie Unicode (\u00b0).

Wariant G

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*=Format("Temperatura = {0:f3} {1}", Variable(), Attribute(Unit))*

Ten wariant działa podobnie do poprzedniego z dwoma różnicami. Liczbowa wartość zmiennej głównej jest przekazywana do funkcji *Format*, a sposób konwersji określa znacznik *{0:f3}*. Pozwala to na formatowanie inne niż podane w bazie zmiennych. Dodatkowo, opis jednostki pobierany jest z bazy zmiennych z atrybutu *Unit(Jednostka)* poprzez wywołanie funkcji *Attribute*. Przykład ten pokazuje jedną z korzyści ze stosowania właściwości *Główna zmienna*. Bez tego, analogiczne wyrażenie wyglądałoby tak:

=Format("Temperatura = {0:f2} {1}", Variable(VA001),Attribute(VA001,Unit))

Nazwa zmiennej byłaby użyta dwukrotnie. Znacznie trudniejsze jest przełączenie obiektu na monitorowanie wartości innej zmiennej.

2.2 Uzależnienie sposobu wyświetlenia tekstu od wartości

W wielu przypadkach, oprócz pokazania wartości zmiennej, konieczna jest kontrola atrybutów wyświetlania. Typowym przypadkiem jest kontrola stanów przekroczeń limitów, błędów pomiarowych itp.

2.2.1 Użycie wyrażeń warunkowych

Wariant A

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		

	Tekst	*#*
	Kolor	*=Variable()>50?Red:Black*

We właściwości *Kolor* zostało użyte wyrażenie warunkowe. Jeżeli wartość zmiennej głównej przekroczy 50, to wartość zmiennej zostanie wyświetlona w kolorze czerwonym.

Wariant B

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*
	Kolor	*=Variable()>50?Red:Black*
	Styl czcionki	*=Variable()>50?Bold:””*

W wariancie B po przekroczeniu wartości 50 zmieniany jest kolor napisu oraz styl czcionki. Pomimo użycia mechanizmu zmiennej głównej, w dalszym ciągu rozwiązanie jest nieefektywne ze względu na dwukrotne użycie identycznego warunku logicznego. W przypadku, gdyby ten warunek musiał być zmieniony, nakład pracy i możliwość popełnienia błędu byłyby stosunkowo duże. Problem ten pozwala rozwiązać mechanizm stanów obiektu.

2.2.2 Użycie metody wielostanowej

Wariant C

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*

	Kolor	*Black*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()>50*
	Kolor	*Red*
	Styl czcionki	*Pogrubienie*

Każdy dodatkowo utworzony stan, oprócz pełnego zestawu właściwości stanu podstawowego, posiada dodatkową właściwość *Warunek stanu.* Jej wartość określa czy stan jest brany pod uwagę przy określaniu wartości właściwości. W przykładzie, gdy wartość zmiennej głównej przekroczy 50, stanowe definicje właściwości *Kolor i Styl czcionki*, będą miały pierwszeństwo przed definicjami dla stanu podstawowego. W efekcie uzyskany zostanie efekt jak w wariancie B, ale bez niepotrzebnego powtarzania warunku i bez wyrażeń warunkowych.

2.3 Zastosowanie limitów ostrzegawczych wartości zmiennej

Innym częstym problemem jest konieczność sygnalizacji przekroczeń wartości alarmowych. Wartości limitów alarmowych pomiaru niekiedy są przechowywane w innych zmiennych procesowych. Mogą być też zapisane w bazie zmiennych.

2.3.1 Zastosowanie notacji sufiksowej

W tym przykładzie zakładamy, że limity przechowywany są w zmiennych. Przy czy zachowana jest następująca konwencja nazewnicza: nazwy zmiennych limitów tworzone są przez dodanie sufiksów *_lolo, _lo, _hi, _hihi* do nazwy zmiennej pomiarowej.

Wariant A

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*
	Kolor	*Black*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()<Variable("#_lo")*

	Kolor	*Aqua*
Właściwości stanowe, stan numer 2		
	Warunek stanu	*=Variable()<Variable("#_lolo")*
	Kolor	*Blue*
Właściwości stanowe, stan numer 3		
	Warunek stanu	*=Variable()>Variable("#_hi")*
	Kolor	*Red*
Właściwości stanowe, stan numer 4		
	Warunek stanu	*=Variable()<Variable("#_hihi")*
	Kolor	*Yellow*

Utworzone zostały 4 stany kontrolujące przekroczenie 4 limitów. W wyrażeniach stanu porównywana jest wartość zmiennej głównej z wartością zmiennej limitu. Nazwy zmiennej typu *#_lolo,* to notacja sufiksowa, w której *#* reprezentuje nazwę zmiennej głównej.

Uzależnienie obiektu od nazw zmiennych jest minimalne, przełączenie obiektu na monitorowanie innej zmiennej wymaga jedynie zmiany nazwy zmiennej głównej.

Każdy ze stanów posiada własną definicje właściwości *Kolor*. Użyta do wyświetlenia zostanie wartość zdefiniowana dla stanu aktywnego. Jeżeli będzie więcej niż jeden aktywny stan, to wzięta zostanie wartość ze stanu o wyższym numerze. Dlatego, istotna jest kolejność definiowania stanów. Stan *_lolo* musi być definiowany po stanie *_lo,* a stan *_hihi* po stanie *_hi*.

2.3.2 Zastosowanie atrybutów zmiennej

Kolejny przykład zakłada, że wartości limitów przechowywane są w atrybutach w bazie zmiennych.

Wariant B

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*

	Kolor	*Black*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()<Attribute(LimitLo)*
	Kolor	*Aqua*
Właściwości stanowe, stan numer 2		
	Warunek stanu	*=Variable()<Attribute(LimitLoLo)*
	Kolor	*Blue*
Właściwości stanowe, stan numer 3		
	Warunek stanu	*=Variable()>Attribute(LimitHi)*
	Kolor	*Red*
Właściwości stanowe, stan numer 4		
	Warunek stanu	*=Variable()<Attribute(LimitHiHi)*
	Kolor	*Yellow*

Warunki stanu porównują wartość zmiennej z wartością odpowiedniego limitu pobranego z bazy zmiennych. Wartości limitów muszą być zapisane w bazie w postaci liczbowej.

Wariant C

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*
	Kolor	*Black*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()<Variable(Attribute(LimitLo))*
	Kolor	*Aqua*
Właściwości stanowe, stan numer 2		
	Warunek stanu	*=Variable()<Variable(Attribute(LimitLoLo))*

	Kolor	*Blue*
Właściwości stanowe, stan numer 3		
	Warunek stanu	*=Variable()>Variable(Attribute(LimitHi))*
	Kolor	*Red*
Właściwości stanowe, stan numer 4		
	Warunek stanu	*=Variable()<Variable(Attribute(LimitHiHi))*
	Kolor	*Yellow*

Warunki stanu porównują wartość zmiennej z wartością odpowiedniego limitu pobranego z bazy zmiennych. W tym przypadku jednak, wartości limitów zapisane w bazie są nazwami zmiennych (wartość atrybutu limitu jest przekazana do funkcji *Variable*).

Wariant D

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*
	Kolor	*Black*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()<Asix6Attribute(LimitLo)*
	Kolor	*Aqua*
Właściwości stanowe, stan numer 2		
	Warunek stanu	*=Variable()<Asix6Attribute(LimitLoLo)*
	Kolor	*Blue*
Właściwości stanowe, stan numer 3		
	Warunek stanu	*=Variable()>Asix6Attribute(LimitHi)*
	Kolor	*Red*
Właściwości stanowe, stan numer 4		

	Warunek stanu	*=Variable()<Asix6Attribute(LimitHiHi)*
	Kolor	*Yellow*

Ten przykład stanowi połączenie wariantów B i C. Funkcja *Asix6Atrribute* akceptuje zarówno limity liczbowe (zwracane bezpośrednio), jak i limity w postaci nazw zmiennych (zwracana jest wartość zmiennej). W bardziej skomplikowanych przypadkach można też użyć funkcję *Asix6Limit*.

2.4 Obsługa statusu monitorowanej zmiennej

Oprócz sygnalizacji przekroczeń, często wymagana jest również sygnalizacja problemów z odczytem zmiennych procesowych.

Wariant A

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	*#*
	Kolor	*Black*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()<Asix6Attribute(LimitLo)*
	Kolor	*Aqua*
Właściwości stanowe, stan numer 2		
	Warunek stanu	*=Variable()<Asix6Attribute(LimitLoLo)*
	Kolor	*Blue*
Właściwości stanowe, stan numer 3		
	Warunek stanu	*=Variable()>Asix6Attribute(LimitHi)*
	Kolor	*Red*

Właściwości stanowe, stan numer 4		
	Warunek stanu	*=Variable()<Asix6Attribute(LimitHiHi)*
	Kolor	*Yellow*
Właściwości stanowe, stan numer 5		
	Warunek stanu	*=VarIsNotGood()*
	Tekst	*=VarStringValue() + " ?"*
	Kolor przekreślenia	*Violet*

Stan numer 5 kontroluje status zmiennej głównej. Gdy jest on niepoprawny, do wartości dodawany jest znak zapytania, a wyświetlony tekst jest przekreślany. Niezależnie od statusu zmiennej, kontrola przekroczeń jest wykonywana w zwykły sposób.

3 Formatowanie wartości liczbowych

W rozdziale *2.1 Wyświetlenie sformatowanej wartości* zostały omówione podstawowe zasady konwersji wartości liczbowych na teksty. W tym rozdziale zostanie dokładniej opisana postać łańcuchów formatujących sterujących sposobem konwersji.

Łańcuchy formatujące używane są w dwóch miejscach. Pierwsze z nich, to atrybut *Format* bazy definicji zmiennych. Stosowany jest w przypadku domyślnej konwersji na tekst (użycie odwołania #, &nazwa lub funkcji *VarStringValue*). Drugie miejsce to parametr formatujący funkcji *Format*.

Postać łańcuchów formatujących jest zgodna z formatem używanym w bibliotekach platformy .NET. Pełny opis formatu można znaleźć w dokumentacji *.NET Framework* dostępnej w Internecie.

Łańcuch formatujący używany w wywołaniach funkcji *Format* ma poniższą strukturę:

{indeks,długość_pola:format_pola}

W atrybucie Format bazy zmiennych należy stosować wyłącznie element *format_pola*.

Element *indeks* określa numer parametru wywołania funkcji *Format*, którego dotyczy dany łańcuch formatujący. Pierwszy parametr ma indeks 0.

Element *długość_pola* określa minimalną długość tworzonego tekstu. Jeżeli długość sformatowanego tekstu będzie krótsza od zadeklarowanej, to zostanie on uzupełniony odpowiednią liczbą spacji. Spacje są dodawane z lewej strony gdy pole ma wartość dodatnią, lub z lewej w przeciwnym przypadku.

Element *format_pola* określa sposób konwersji liczby na tekst. Sposób jego użycia pokażemy na przykładach.

3.1 Konwersja wartości numerycznych

W przypadku konwersji wartości numerycznych format pola składa się z jednoliterowego specyfikatora typu oraz liczbowego pola precyzji. Interpretacji pola precyzji zależy od użytego typu.

Specyfikator typu	**Opis**	**Przykład**
D , d	Formatowanie liczb całkowitych. Można stosować dla wartości zmiennoprzecinkowych, ale w tym przypadku zostanie wykonane zaokrąglenie do liczby całkowitej. Pole precyzji określa minimalną liczbę cyfr, w razie konieczności zostaną dodane zera nieznaczące.	”D” 1234 -> 1234 ”D” 1234,56 -> 1235 ”D6” 1234 -> 001234
E , e	Formatowanie liczb zmiennoprzecinkowych w formie ekspotencjalnej. Pole precyzji określa liczbę cyfr ułamkowych.	”E” 1234,56 -> 1,234560E+003 ”e2” 1234,56 -> 1,23e+003
F , f	Formatowanie liczb zmiennoprzecinkowych w postaci części całkowitej i ułamkowej rozdzielonych systemowym separatorem. Pole precyzji określa liczbę cyfr ułamkowych.	”F” 1234,567 ->1234,57 ”F4” 1234,567 ->1234,5670 ”F1” 1234,567 ->1234,6
N , n	Formatowanie liczb zmiennoprzecinkowych w postaci części całkowitej i ułamkowej rozdzielonych systemowym separatorem. Oprócz tego grupy cyfr także zostaną rozdzielone separatorem. Pole precyzji określa liczbę cyfr ułamkowych.	”N” 1234,567 ->1 234,57 ”N0” 1234,567 ->1 235
P , p	Formatowanie liczb zmiennoprzecinkowych lub całkowitych w postaci procentowej z dodanym znakiem %. Wartość przed sformatowaniem zostanie pomnożona przez 100.Pole precyzji określa liczbę cyfr ułamkowych.	”P” 0,567 ->56,70% ”P1” 0,567 ->56,7%
X , x	Formatowanie liczb całkowitych w postaci szesnastkowej. Można stosować dla wartości	”X” 1234 ->4D3

	zmiennoprzecinkowych, ale w tym przypadku zostanie wykonane zaokrąglenie do liczby całkowitej. Pole precyzji określa minimalną liczbę cyfr, w razie konieczności zostaną dodane zera nieznaczące.	”x4” 1234 ->04d3

Oprócz formatowania opisanego powyżej można też użyć tzw. *Custom Numeric Format Strings*. Pełny opis znajduje się w dokumentacji platformy *.NET Framework.*

3.2. Konwersja wartości typu DateTime

W przypadku konwersji wartości typu *DateTime* można stosować standardowe formaty predefiniowane lub swobodnie tworzone formaty własne.

Poniższa tabelka pokazuje przykład formatowania standardowego.

Specyfikator typu	**Opis**	**Przykład**
d	Krótka data	2012-02-16
D	Długa data	16 lutego 2012
f	Długa data + krótki czas	16 lutego 2012 16:33
F	Długa data + krótki czas	16 lutego 2012 16:33:48
g	Pełna data + krótki czas	2012-02-06 16:33
G	Pełna data + długi czas	2012-02-06 16:33:48
M, m	Miesiąc + dzień	16 lutego
t	Krótki czas	16:33
T	Długi czas	16:33:48
Y, y	Rok + miesiąc	luty 2012

Jeżeli powyższe sposoby formatowania są niewystarczające , to można stosować formaty własne. Taki format zdefiniowany jest sekwencją specyfikatorów opisujących poszczególne pola tworzonego tekstu. Na przykład format *„dddd dd MMMM yyyy HH:mm:ss.fff”* powoduje utworzenie tekstu *„czwartek 16 lutego 2012 16:33:48.240”*.

Poniższa tabelka opisuje podstawowe specyfikatory składowe formatów własnych.

Specyfikator typu	**Opis**

d	Dzień miesiąca jedno lub dwucyfrowy
dd	Dzień miesiąca dwucyfrowy
ddd	Skrócona nazwa dnia tygodnia
dddd	Pełna nazwa dnia tygodnia
f	Ułamkowa część sekundy, w zależności od ilości specyfikatorów *f* można uzyskać różną dokładność czasu.
H	Godzina, jedno lub dwucyfrowa
HH	Godzina, dwucyfrowa (duże H - format 24h; małe h - format 12h)
m	Minuta , jedno lub dwucyfrowa
mm	Minuta , dwucyfrowa
M	Numer miesiąca, jedno lub dwucyfrowy
MM	Numer miesiąca, dwucyfrowy
MMM	Skrócona nazwa miesiąca
MMMM	Pełna nazwa miesiąca
s	Sekunda , jedno lub dwucyfrowa
ss	Sekunda, dwucyfrowa
yy	Dwie ostatnie cyfry numeru roku
yyyy	Numer roku
znaki neutralne	Kopiowane wprost do teksu wynikowego
znaki wewnątrz apostrofów ‘ ’	Kopiowane wprost do teksu wynikowego

3.3. Konwersja wartości typu TimeSpan

W przypadku konwersji wartości typu *TimeSpan* można stosować standardowe formaty predefiniowane lub swobodnie tworzone formaty własne.

Poniższa tabelka pokazuje przykład formatowania standardowego.

Specyfikator typu	**Opis**	**Przykład**
c	Krótka data ([-][d'.']hh':'mm':'ss['.'fffffff])	3.07.25:30
g	Krótki format ogólny ([-][d':']h':'mm':'ss[.FFFFFFF])	3.7.25:30
G	Długa format ogólny ([-]d':'hh':'mm':'ss.fffffff)	3.07.25.:30.0000000

Jeżeli powyższe sposoby formatowania są niewystarczające, to można stosować formaty własne. Taki format zdefiniowany jest sekwencją specyfikatorów opisujących poszczególne pola tworzonego tekstu. Poniższa tabelka opisuje specyfikatory składowe formatów własnych.

Specyfikator typu	**Opis**
d	Liczba pełnych dni w okresie czasu
dd - ddddddd	Liczba pełnych dni w okresie czasu z dodaną odpowiednią liczbą zer początkowych
h	Liczba pełnych godzin w okresie czasu nieuwzględnionych w liczbie dni
hh	Zawsze dwucyfrowa liczba pełnych godzin w okresie czasu nieuwzględnionych w liczbie dni
m	Liczba minut w okresie czasu nieuwzględniona w liczbie dni i godzin.
mm	Zawsze dwucyfrowa liczba minut w okresie czasu nieuwzględniona w liczbie dni i godzin
S	Liczba sekund w okresie czasu nieuwzględniona w liczbie dni, godzin i minut
ss	Zawsze dwucyfrowa liczba sekund w okresie czasu nieuwzględniona w liczbie dni, godzin i minut
f - fffffff	Część niepełnej sekundy z odpowiednią dokładnością
F - FFFFFFF	Część niepełnej sekundy z odpowiednią dokładnością, końcowe zera nie są wyświetlane
\	Znak następujący po \ jest kopiowany do tekstu wynikowego

znaki wewnątrz apostrofów ‘ ’	Kopiowane wprost do tekstu wynikowego

4 Animacja wyglądu obiektów

W wielu aplikacjach występuje potrzeba przedstawienia na diagramach wrażenia ruchu wybranych elementów lub dynamicznej sygnalizacji niektórych stanów. Rozdział ten opisuje w jaki sposób zrealizować wspomnianą funkcjonalność.

4.1 Animowane obrazki

Najprostszą metodą animacji jest użycie animowanych obrazków w formacie GIF. Jedyne co należy wykonać, to użyć nazwę animowanego obrazka we właściwości *Nazwa obrazu* obiektu klasy *Obraz*.

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Nazwa obrazu	*NoFire*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()&1*
	Nazwa obrazu	*Fire*

Jeżeli najmłodszy bit zmiennej głównej jest ustawiony, to aktywny staje się stan numer 1, w efekcie czego wyświetlony zostanie animowany obrazek *Fire*.

4.2 Animacja ruchu rurociągów, taśmociągów i linii

Niektóre klasy obiektów przygotowane są wstępnie do animacji ich ruchu. Parametryzacja ruchu polega na odpowiednim ustawieniu właściwości odpowiadających za wygląd obiektu oraz dynamicznej zmianie właściwości modyfikującej przesunięcie wyświetlanego kształtu. Z reguły do animacji wykorzystywana jest zmienna wewnętrzna *Counter*, która jest automatycznie inkrementowana ze stałym cyklem.

4.2.1 Rurociąg

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Kolor pasków	*Blue*
	Offset wypełnienia	*0*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()&1*
	Offset wypełnienia	*=Variable(Counter)*
Właściwości stanowe, stan numer 2		
	Warunek stanu	*=Variable()&2*
	Offset wypełnienia	*=-Variable(Counter)*2*
Właściwości stanowe, stan numer 3		
	Warunek stanu	*=Variable()&4*
	Offset wypełnienia	*=-Variable(Counter)*
Właściwości stanowe, stan numer 4		
	Warunek stanu	*=Variable()&8*

	Offset wypełnienia	*=-Variable(Counter)*2*

Ustawienie właściwości *Kolor pasków* powoduje, że rurociąg wyświetlany jest w postaci paskowanej. Zmiana właściwości *Offset wypełnienia* powoduje przesuwanie pasków, co pozwala na uzyskanie efektu przepływu w rurociągu. W zależności od ustawionego bitu zmiennej głównej, właściwość *Offset wypełnienia* jest zmieniana w różny sposób: mnożenie (lub ewentualne dzielenie) wartości zmiennej *Counter* steruje szybkością ruchu, zmiana znaku wyrażenia steruje kierunkiem ruchu.

4.2.2 Taśmociąg

Właściwości podstawowe		
	Główna zmienna	VA001
	Styl rolki początkowej	Napęd
	Styl rolki końcowej	Napęd
	Taśmociąg zgrzebłowy	Tak
Właściwości stanowe, stan podstawowy		
	Przesunięcie taśmy	*0*
	Obrót napędu początkowego	*0*
	Obrót napędu końcowego	*0*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()&1*
	Przesunięcie taśmy	*=Variable(Counter)*
	Obrót napędu początkowego	*=Variable(Counter)*
	Obrót napędu końcowego	*=Variable(Counter)*

Ustawienia właściwości *Styl rolki …* i *Taśmociąg zgrzebłowy* powodują wyświetlenie taśmociągu w sposób pozwalający na animację obrotu rolek końcowych oraz ruchu taśmy. Jeżeli najmłodszy bit zmiennej głównej jest ustawiony, to aktywny staje się stan numer 1, w którym wykonywana jest animacja ruchu. Zmiana właściwości *Przesunięcie taśmy* powoduje przesuwanie taśmociągu. Zmiana właściwości *Obrót napędu …* pozwala na animację obrotu odpowiedniej rolki końcowej.

4.2.3 Linia

Właściwości podstawowe		
	Główna zmienna	Speed
	Styl pasków	Kreska
Właściwości stanowe, stan podstawowy		
	Kolor pasków	*Blue*
	Offset wypełnienia	*=Variable()*

Ustawienie właściwości *Styl pasków* i *Kolor pasków* powoduje wyświetlenie linii w postaci paskowanej. *Offset wypełnienia* pobierany jest z wartości zmiennej *Speed*. Szybkość przesuwu pasków jest bezpośrednio zależna od szybkości zmian zmiennej – program sterownika może płynnie sterować szybkością ruchu. Brak zmian zmiennej *Speed* powoduje zatrzymanie ruchu.

4.3 Migotanie poprzez zmianę właściwości obiektów

Najbardziej uniwersalną metodę animacji jest cykliczna zmiana wybranej właściwości obiektu. Metodę tę można stosować dla obiektu dowolnej klasy oraz wykorzystywać dowolną właściwość odpowiedzialną za wygląd obiektu. Najczęściej zmiana właściwości jest synchronizowana wartością funkcji *IsBlinkOff*. Funkcja ta zwraca cyklicznie wartości *true* i *false.* Standardowy okres zmian to 500ms, ale można go zmienić w ustawieniach aplikacji. Wykorzystanie funkcji *IsBlinkOff* zapewnia identyczne tempo migotania wszystkich obiektów.

Poniżej pokazano kilka wariantów realizacji funkcji migotania.

Wariant A

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Widoczny	*TAK*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=Variable()==3*

	Widoczny	*=IsBlinkOff()*

Powyższy schemat można zastosować dla obiektu dowolnej klasy. Jeżeli zmienna główna ma wartość 3, to aktywny stan numer 1 powoduje, poprzez zmianę właściwości *Widoczny*, cykliczne pokazywanie i ukrywanie całego obiektu.

Wariant B

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Kolor	*Blue*
	Tekst	*#*
Właściwości stanowe, stan numer 1		
	Warunek stanu	=IsAlarm("A_"+Attribute(Name))
	Kolor	*=IsBlinkOff()?Red:Green*

W przykładzie B modyfikowany jest kolor wyświetlenia tekstu w obiekcie *Tekst*. W stanie normalnym wyświetlana jest wartość zmiennej głównej w kolorze niebieskim. Jeżeli aktywny jest alarm o identyfikatorze *A_VA001 (* utworzonym na podstawie nazwy zmiennej głównej), to wartość zmiennej wyświetlana jest naprzemiennie w kolorach czerwonym i zielonym. Wyborem koloru steruje wyrażenie warunkowe wykorzystujące funkcję *IsBlinkOff*.

Wariant C

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Nazwa obrazu	*wizard*
	Jasność	*0*
Właściwości stanowe, stan numer 1		

	Warunek stanu	=Variable()==3 && IsBlinkOff()
	Jasność	*-0,5*

W przykładzie C migotanie jest realizowane poprzez zmianę jasności wyświetlanego obrazka w obiekcie klasy *Obraz*. W tym przykładzie funkcja *IsBlinkOff* jest umieszona w wyrażeniu warunku stanu razem ze sprawdzeniem wartości zmiennej głównej. Jeżeli wartość zmiennej głównej jest równa 3, to w zależności od aktualnej wartości funkcji *IsBlinkOff*, jasność obrazka pobierana jest ze stanu podstawowego lub stanu numer 1.

5 Zasady otwierania i zamykania okien synoptycznych

Jednym z kluczowych elementów każdej aplikacji jest konstrukcja systemu nawigacji pomiędzy poszczególnymi diagramami aplikacji. Do otwierania i zamykania okien i diagramów służą akcje operatorskie *OpenWindow*, *OpenDiagram* i *CloseWindow*. Wybrane okna mogą być także otwierane automatycznie w trakcie uruchomienia aplikacji.

Akcje operatorskie mogą być podpięte do zdarzeń związanych z obsługą myszki, np. zdarzenia *Włączenie przycisku* obiektu klasy *Przycisk* lub zdarzeń *Kliknięcie lewym przyciskiem* itp. dla innych klas. Akcje przełączania okien i diagramów mogą być też użyte w definicjach menu aplikacji i klawiszy globalnych.

5.1 Otwieranie okien synoptycznych

5.1.1 Otwarcie okna w trakcie startu aplikacji

Wybrane okna mogą zostać uruchomione w trakcie startu aplikacji. W tym celu należy:

a. W panelu edycyjnym *Ustawienia stanowisk* w zakładce *Okna startowe* zaznaczyć okna, które mają zostać uruchomione w trakcie startu aplikacji. Ustawienie okien startowych jest specyficzne dla stanowiska. Wybór okien należy wykonać osobno dla każdego stanowiska lub ustawić jena poziomie obszaru.

b. W ustawieniach wszystkich paneli wchodzących w skład okien startowych odpowiednio ustawić właściwość *Domyślny diagram*.

c. Jeżeli domyślne diagramy posiadają parametry, to ich początkowe wartości należy podać jako wartości domyślne. Nie ma możliwości jawnego podania parametrów w przypadku okna uruchamianego na starcie.

W typowym przypadku na starcie aplikacji otwierane jest jedno okno startowe na każdym dostępnym monitorze.

5.1.2 Otwarcie okna i diagramu akcją OpenWindow

Akcja *OpenWindow* używana jest do otwarcia nowego predefiniowanego okna synoptycznego lub podmiany diagramu w wybranym panelu okna już otwartego. Może być też w celu uaktywnienia okna wcześniej otwartego bez zmiany wyświetlanych diagramów. Dokładny algorytm działania akcji znajduje się w opisie akcji.

Przykład 1.

OpenWindow(okno1,null,null,null)

Akcja otwiera okno *okno1* z domyślnym diagramami w panelach. Jeżeli *okno1* jest już otwarte, to zostanie ono uaktywnione bez zmiany wyświetlanych diagramów. Działanie tej akcji jest zbliżone do uruchamiania okien startowych.

Przykład 2.

OpenWindow(okno1,info,machine,”mid=8”)

Zadaniem akcji jest otwarcie okna *okno1* z diagramem *machine* w panelu *Info* i parametrem *mid* równym 8 lub podmiana diagramu w oknie *okno1* otwartym wcześniej. Jeżeli jest kilka otwartych okien *okno1*, to najpierw jest sprawdzane czy jest otwarte okno z diagramem *machine* w panelu *Info* i zgodnymi parametrami. Jeżeli tak, to akcja ograniczy się tylko do uaktywnienia już otwartego okna.

Akcja używana do podmiany diagramów w oknach głównych aplikacji lub do otwarcia okien tzw. stacyjek wtedy, gdy chcemy mieć otwartą tylko jedną stacyjkę.

Przykład 3.

OpenWindow(okno1,info,machine,”mid=8”,true)

Działanie akcji jest podobne do poprzedniego przykładu. Różnica polega na tym, że użyte (uaktywnione) zostanie okno wcześniej otwarte tylko przy całkowitej zgodności nazwy okna, diagramu i parametrów.

Akcja używana do otwierania okien tzw. stacyjek wtedy, gdy chcemy mieć otwartych dowolną liczbę stacyjek (ale każdą z innym diagramem lub parametrami).

Przykład 4.

OpenWindow(null,info,machine,”mid=8”)

Zadaniem akcji jest wymiana diagramu i parametrów w panelu *info* w oknie kontekstowym akcji operatorskiej. Okno kontekstowe wynika z miejsca zdefiniowania wykonywanej akcji, np. położenia obiektu *Przycisk*, którego przyciśnięcie powoduje wykonanie akcji.

5.1.3 Otwarcie diagramu bez użycia predefiniowanego okna synoptycznego

Możliwe jest także bezpośrednie otwarcie diagramu synoptycznego bez wcześniejszego definiowania okna. Pozwala na to akcja *OpenDiagram*. W wyniku jej działania tworzone jest wewnętrznie okno tymczasowe, w którym wyświetlany jest żądany diagram.

Przykład 1.

> *OpenDiagram(machine,"mid=8",stacyjka,$Normal, &CursorLocation, &CursorLocation,"FixedSize", $None)*
>
> Akcja otwiera diagram *machine* z parametrem *mid* równym 8 w oknie utworzonym automatycznie. Rozmiar okna jest zgodny z rozmiarem diagramu, a jego położenie wynika z pozycji kursora. Tytuł okna oraz jego wygląd i zachowanie są określone bezpośrednio w treści akcji. W tym przypadku tytułem okna jest tekst *stacyjka*, okno ma stały rozmiar, a pozostałe parametry wyglądu są ustawione domyślnie.Podobnie jak w przypadku akcji *OpenWindow*, przed otwarciem nowego okna wykonywane jest sprawdzenia czy diagram z podanymi parametrami nie jest już otwarty.
>
> Działanie akcji jest zbliżone do akcji *OpenWindow(okno1,info,machine,"mid=8",true)* i pozwala na otwieraniu wielu okien stacyjek jednocześnie.

Podstawowa różnica pomiędzy akcjami *OpenWindow* i *OpenDiagram* polega na miejscu zdefiniowania wyglądu i zestawu dostępnych operacji systemowych okna. W pierwszym przypadku jest to określone w parametrach predefiniowanego okna, w drugim w treści akcji. Definicja w treści akcji pozwala na dynamiczne dostosowanie wyglądu okna, a w szczególności tytułu okna.

Przykład 2.

> *OpenDiagram(machine,"varname="+Attribute(Name),"Zmienna "+Attribute(Name),$Normal, &CursorLocation, &CursorLocation,"FixedSize", $None)*
>
> W tym przypadku nazwa zmiennej głównej obiektu (w kontekście którego akcja jest wykonywana) jest przekazywana do parametrów diagramu (parametr *"varname="+Attribute(Name)*). Jednocześnie nazwa zmiennej dodawana jest do tytułu stacyjki (parametr *"Zmienna "+Attribute(Name)*).

Dodatkowa zaletą akcji *OpenDiagram* jest możliwość określenia trybu pracy. W pokazanym dotychczas trybie *$Normal* okno zachowuje się jak każde inne predefiniowane okno aplikacji. W trybie *$Dialog* tworzona jest stacyjka dialogowa – nie ma możliwości przejścia do innego okna do momentu zamknięcia stacyjki. W trybie *$Temp* okno jest automatycznie zamykane w chwili wyboru innego okna aplikacji.

5.2 Kontrola położenia i rozmiaru okien

Miejsce otwarcia nowych okien jest deklarowane bezpośrednio w akcjach *OpenWindow* i *OpenDiagram*. Pozycja może być deklarowana na cztery sposoby:

a. Współrzędne podane bezpośrednio

b. Współrzędne wynikające z definicji okna – stała *$Default*

c. Współrzędne wynikające z aktualnej pozycji myszki – stała *$CursorLocation*

d. Współrzędne wynikające z pozycji okna, z którego wykonano akcję otwarcia – stała *$ActiveLocation*

We wszystkich wariantach wywołania akcji otwarcia, jeżeli dojdzie do wykrycia sytuacji, że właściwe okno jest już otwarte, to nastąpi korekta pozycji istniejącego okna zgodnie z podanymi parametrami współrzędnych.

Przykład 1

```
OpenDiagram(machine,”mid=8”,stacyjka,$Normal, $ActiveLocation,
$ActiveLocation,”FixedSize”, $Closable)
```

Akcja otwiera diagram *machine* w miejscu okna, z którego akcja została zainicjowana. Dodatkowo parametr *$Closable* spowoduje, że to okno zostanie zamknięte. Takie użycie akcji pozwala na uzyskanie efektu samowymieniających się okien – nawet po zmianie przez użytkownika pozycji pierwszego okna, drugie zostanie otwarte w jego miejscu. Podobny efekt można uzyskać stosując wymianę diagramu w panelu okna stale otwartego.

Tryby *$CursorLocation i $ActiveLocation* w sposób naturalny otwierają okno na ekranie aktualnie używanym. Tryb *$Default* używa monitora z góry określonego. W trybie współrzędnych podawanych bezpośrednio można dynamicznie wyliczać właściwą pozycję.

Przykład 2.

```
OpenWindow(okno1,info,machine,”mid=8”,false,XOnScreen(0,0),YOnScreen(100,0),$None)
```

Użycie funkcji *XOnScreen* i *YOnScreen* pozwala wyliczyć pozycję w zależności od wybranego monitora. Drugi parametr obu funkcji określa numer monitora (0 oznacza monitor kursora myszki). Pokazana akcja otworzy okno na monitorze kursora myszki, przy lewej krawędzi ekranu, od linii numer 100.

Rozmiar otwieranych okien wynika z rozmiaru predefiniowanego okna (*OpenWindow)* lub rozmiary diagramu (*OpenDiagram)*. Można jednak zmienić rozmiar po operacji otwarcia okna przy pomocy akcji *SetWindowSize*.

5.3 Zamykanie okien i diagramów

Okna mogą być zamykane na kilka sposobów. Są to:

a. Zamknięcie okna przez użytkownika standardowymi metodami systemowymi, typu przycisk zamknięcia na ramce okna.

b. Automatyczne zamknięcie ze względu na otwarcie diagramu akcją *OpenDiagram* w trybie *$Temp.*

c. Użycie trybu wymiany innego niż *$None* w akcjach *OpenDiagram* i *OpenWindow.*

d. Użycie akcji *CloseWindow.*

Przykład 1.

OpenWindow(okno1,info,machine,”mid=8”,false,$CursorLocation,$CursorLocation,$Closable)

Akcja wykonuje otwarcie okna *okno1*, a następnie zamyka wszystkie okna bez blokady zamknięcia, które znajdują się na ekranie, na którym okno zostało otwarte (w tym przypadku ekran kursora myszki). Można użyć też stałej *$ExceptCurrent.* Działanie jest podobne, ale nie zostanie zamknięte okno, z którego akcję wykonano.

Okna zablokowne to te, które mają ustawioną właściwość *Zezwolenie na zamykanie* na *Nie*.

Przykład 2.

CloseWindow(null,false)

Akcja zamyka okno w kontekście którego została wykonana (okno bieżące).

Przykład 3.

CloseWindow(”ST*”,true,-1)

Akcja zamyka wszystkie okna o nazwie zaczynającej się od *ST* na wszystkich monitorach. Można też zamykać okna tylko na wskazanym monitorze lub monitorze kursora myszki (ostatni parametr równy 0).

5.4 Kontrola wzajemnego przesłaniania okien synoptycznych

Kolejnym istotnym elementem przy projektowaniu systemu okien jest kontrola ich wzajemnego przesłaniania. W typowym przypadku okno aktywne jest wyświetlane powyżej wszystkich innych okien aplikacji. W niektórych przypadkach może to być niekorzystne. Na przykład w przypadku konieczności wyświetlenia kilku okien typu stacyjka na tle dużego okna startowego standardowe zasady działania są niewystarczające.

Można jawnie kontrolować wzajemne przesłanianie przy pomocy właściwości okna *Zawsze pod spodem* i *Zawsze na wierzchu.* Można też stosować w akcji *OpenDiagram* atrybut *TopMost*. Okna „na wierzchu" wyświetlane są zawsze ponad normalnymi oknami, nawet wtedy, gdy nie są aktywne. Okna „pod spodem" zawsze są poniżej wszystkich innych okien.

Powyższe uwagi dotyczą także wszystkich innych okien uruchomionych na komputerze, niekoniecznie będących oknami aplikacji Asix.Evo. Z tego względu należy zachować dużą ostrożność w użyciu masek „na wierzchu". Mogą one przesłonić inne ważne okna. Nie należy otwierać w tym trybie dużych okien. Bezpieczniejszym rozwiązaniem jest otwarcie okien startowych aplikacji w trybie „pod spodem" i nie używanie okien „na wierzchu".

6 Organizacja operacji sterujących

Istotnym elementem prawie każdej aplikacji jest wprowadzenie mechanizmów służących wykonywaniu operacji sterujących pracą kontrolowanego obiektu. Niektóre obiekty posiadają wbudowane mechanizmy sterujące, np. *Tekst, Przycisk, Słupek*. W przypadku innych obiektów, można je rozszerzyć o funkcje sterowania korzystając z obsługi zdarzeń i funkcji *SetVariable*. Przykłady takich parametryzacji są opisane w innych rozdziałach dokumentacji.

W tym rozdziale zajmiemy się ogólnymi zasadami organizacji operacji sterujących w aplikacji.

6.1 Sterowania natychmiastowe

Najprostszym trybem wykonania operacji sterującej jest tzw. tryb natychmiastowy. Zaraz po wybraniu lub wprowadzeniu nowej nastawy, bez żadnego dodatkowego potwierdzenia, wysyłana jest do zmiennej sterowanej odpowiednio wartość.

Parametryzacja trybu natychmiastowego ogranicza się do określenia nazwy zmiennej sterowanej oraz wyboru trybu.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level
	Zmienna sterowana	LevelSet
	Sterowanie natychmiastowe	Tak

Właściwość *Aktywny* ustawiana na *Tak* włącza funkcje interaktywne obiektów, w szczególności funkcje sterujące. Właściwość *Sterowanie natychmiastowe* równa *Nie,* forsuje pracę w trybie natychmiastowym bez potwierdzania.

W pokazanym przykładzie nazwa zmiennej sterowanej podana jest bezpośrednio we właściwości *Zmienna sterowana*. Jest to parametryzacja poprawna, ale ze względu na ogólne zalecenie, żeby parametryzacja obiektu była sterowana wyłącznie przez nazwę zmiennej głównej i jej parametry, to zalecane są raczej poniższe konstrukcje.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level
	Zmienna sterowana	#Set
	Sterowanie natychmiastowe	Tak

Nazwa zmiennej sterowanej powstaje ze złożenia nazwy zmiennej głównej i sufiksu *Set*. W szczególności, użycie tylko znaku *#*, oznaczałoby, że nazwa zmiennej sterowanej jest taka sama jak nazwa zmiennej głównej.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level
	Zmienna sterowana	@ControlVariable
	Sterowanie natychmiastowe	Tak

Nazwa zmiennej sterowanej pobierane jest z atrybutu *ControlSetVariable* bazy zmiennych z wiersza opisującego definicję zmiennej *Level*. Atrybut *ControlSetVariable* nie należy do zestawu atrybutów standardowych – musi być dodany do bazy zmiennych na etapie jej tworzenia.

6.2 Sterowania opóźnione (z potwierdzeniem)

Tryb sterowania opóźnionego włączane jest poprzez ustawienie właściwości *Sterowanie natychmiastowe* na wartość *Nie*. Tryb stosowane jest wtedy, gdy chcemy uniknąć przypadkowego wysterowania lub wtedy, gdy chcemy wykonać w tym samym momencie całą grupę sterowań.

Parametryzacja opóźnionych sterowań składa się z dwóch elementów: sygnalizacji oczekiwania na wykonanie sterowania oraz zainicjowanie operacji wysłania danych.

Sygnalizacja stanu oczekiwania bazuje na użyciu funkcji *HasWaitingControl*. Obiekt oczekujący na sygnał wysłania powinien zmodyfikować swoje właściwości związane z wyglądem.

Właściwości stanowe, stan podstawowy		
	Tekst	Stan nierozpoznany
	Kolor	=HasWaitingControl()?Red:Black

W tym przykładzie obiekt *Tekst* zmienia kolor w trybie oczekiwania. Wykorzystane jest tutaj wyrażenie warunkowe. Jest to najprostsza forma, ale ma zastosowanie wtedy, gdy obiekt nie posiada innych stanów zmieniających właściwość koloru. Jest też niewygodna gdyby chcieliśmy zmienić więcej niż jedną właściwość.

Właściwości stanowe, stan x		
	Warunek stanu	=HasWaitingControl()
	Kolor	Cyan

Łatwiej jest użyć dodatkowego stanu z warunkiem *HasWaitngControl*. Stan ten należy ustawić jako ostatni stan obiektu (a przynajmniej po wszystkich stanach modyfikujących te same właściwości).

Wykonanie oczekujących operacji sterujących inicjowane jest przy pomocy akcji *SendControls*. W zależności od użytego parametru akcji, przycisk (lub inny obiekt) wykonujący akcję może powodować wysłanie sterowań ze wszystkich diagramów (*$All*), z bieżącego okna (*$Window*), z bieżącego diagramu (*$Diagram*) lub z obiektów o określonej nazwie.

Właściwości podstawowe		
	Aktywny	Tak
	Typ kursora	Hand

	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Kolor wyłączenia	=HasWaitingControl(”*”)?LightCoral:LightGray
	Tekst wyłączenia	Wyślij
Zdarzenia		
	Wyłączenie przycisku	*^SendControls($Diagram)*

Pokazany fragment parametryzacji obiektu *Przycisk* inicjuje wysłanie sterowań z diagramu, na którym został umieszczony. Dodatkowo sygnalizowany jest stan, czy są oczekujące sterowania. Wywołanie funkcji *HasWaitingControl* z parametrem wzorca nazwy obiektu pozwala na sprawdzenie czy istnieją obiektu o zgodnej nazwie oczekujące na wykonanie sterowania. Parametr * oznacza wszystkie obiekty niezależnie od nazwy. Mechanizm ten działa jednak tylko, dla obiektów znajdujących się na tym samym diagramie. Nie może być używany w przypadku globalnej inicjacji sterowań z innych diagramów.

Dostępny jest również mechanizm anulowania oczekujących sterowań. Do tego celu służy akcja *CancelControls* używana w podobny sposób jak akcja *SendControls*.

6.3 Sterowanie przy pomocy akcji operatorskiej SetVariable

W przypadku stosowania metod wysyłania nastaw nieopartych na wbudowanych mechanizmach obiektów konieczne jest użycie akcji operatorskiej *SetVariable*. Funkcja występuje w dwóch wariantach.

```
SetVariable(Alfa,1)
```

Ten wariant ustawia zmienną *Alfa* bezpośrednio na wartość 1.

```
SetVariable(Alfa,1,0xf)
```

Ten wariant ustawia wykonuje modyfikacje bitów zmiennej *Alfa*. Cztery najmłodsze bity są ustawiane na wartość 0001. Należy pamiętać o tym, że sterowania bitowe są mocno nieefektywne pod względem czasu wykonania i obciążenia łącza.

Chociaż bezpośrednie podanie nazwy zmiennej jest poprawne, to należy się starać o ile jest to możliwe stosować konwencję kontekstową.

```
SetVariable(””,1)
```

SetVariable(”#”,1)

SetVariable(”#Set”,1)

W kolejnych przykładach użyto: nazwy zmiennej sterowanej obiektu, nazwy zmiennej głównej obiektu, nazwy złożonej z nazwy zmiennej głównej i sufiksu *Set*. Warto zdefiniować zmienną główną także dla obiektów, które służą tylko do sterowania i użyć jej w akcji *SetVariable*.

6.4 Kontrola ważności sterowań

W przypadku sterowań z opóźnieniem istotne jest określenie jak długo wprowadzona nastawa jest ważna. Można dodać do aplikacji przycisk wykonujący akcję *CancelControls*. Pozwoli to operatorowi wycofać sterowanie bez zamykania diagramu. Istnieje też metoda automatyczna. W panelu *Ustawienia stanowiska* w zakładce *Ustawienia* można podać czas ważności sterowania w polu *Limit czasu sterowania.* W przypadku jego ustawienia wartości nastaw tracą ważność po upłynięciu podanego czasu.

Problem może też stanowić przypadek jest nastawa wybrana na diagramie typu stacyjka pozostaje ważna, ale sama stacyjka została przesłonięta innym oknem (czyli użytkownik nie widzi sygnalizacji sterowania). Rozwiązaniem jest użycie akcji *SendControls* z parametrem *$AllVisible*. Powoduje to, że sygnał wysłania dotyczy tylko obiektów na oknach, które nie są żaden sposób przesłonięte przez inne okno.

6.5 Kontrola uprawnień

W wielu aplikacjach bardzo istotna jest kontrola uprawnień aktualnie zalogowanego użytkownika do wykonywania operacji sterujących. W systemie zabezpieczeń istnieje uprawnienie *Prawo wysyłania sterowań*. Użytkownik musi pełnić rolę, w której wspomniane uprawnienie jest aktywne. Jeżeli tak nie będzie, to każde sterowanie przez niego wykonane zakończy się błędnie.

Z punktu widzenia bezpieczeństwa wystarczy odpowiednio sparametryzować system zabezpieczeń. Lepiej jest jednak zatroszczyć się kontrolę uprawnień zanim jeszcze dojdzie do próby wykonania sterowania. Obiekty posiadają dwie właściwości, które mogą być wykorzystane do tego celu: *Aktywny* i *Widoczny*. Obie właściwości są typu logicznego. Ustawiając właściwość *Aktywny* na *false* (*NIE*) blokuje wbudowane funkcje sterujące obiektów, a także reakcje na przyciśnięcie klawiszy myszki. Przy pomocy właściwości *Widoczny* można całkowicie ukryć niepożądane obiekty.

Właściwa procedura postepowania to zdefiniowanie roli, w której uprawnienie *Prawo wysyłania sterowań* jest aktywne, a następnie przyłączenie do niej wszystkich użytkowników, którzy mają mieć możliwość wykonywania sterowań. Potem wystarczy użyć wyrażenia wykorzystującego funkcję *HasRole* we właściwościach *Aktywny* lub *Widoczny* obiektów związanych z operacjami sterującymi.

Właściwości podstawowe		
	Aktywny	=HasRole(SuperOperator)
	Główna zmienna	AlarmLimit
	Zmienna sterowana	#
Właściwości stanowe, stan podstawowy		
	Tekst	#

Powyższy przykład pokazuje fragment parametryzacji obiekt *Tekst.* Jeżeli zalogowany użytkownik nie będzie pełnił roli *SuperOperator*, to obiekt będzie wyświetlał wartość nastawy, ale nie pozwoli na jej zmianę.

Ten sam mechanizm można także zastosować w trochę innym schemacie pracy. Załóżmy, że standardowy operator ma możliwość sterowania. Chcemy jednak, aby niektóre sterowania były dostępne tylko niektórym operatorom. W tym celu tworzymy dodatkową rolę (nie musi ona nawet mieć żadnych aktywnych uprawnień), przydzielamy ją właściwym użytkownikom i przy pomocy funkcji *HasRole* blokujemy newralgiczne obiekty.

6.5.1 Podwójne potwierdzanie operacji sterujących

Czasem potrzebny jest trochę inny tryb pracy, w którym operator prowadzi proces, ale wykonanie niektórych operacji wymaga dodatkowego potwierdzenia innej osoby. Nie chcemy jednak zmieniać zalogowanego użytkownika. W tym celu użyjemy akcję operatorską *ConfirmRole*.

Właściwości podstawowe		
	Aktywny	Tak
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Tekst wyłączenia	Wyślij
Zdarzenia		

	Wyłączenie przycisku	*^Actions(ConfirmRole(Kierownik,$Always), SendControls($All))*

Powyższy przykład to modyfikacja przycisku wysyłającego sygnał wykonania opóźnionych sterowań. Wykonywana akcja jest w tym przypadku złożeniem dwóch akcji *ConfirmRole* i *SendControls*. Użycie akcji *ConfirmRole* spowoduje otwarcia okna logowania. Po autoryzowaniu się użytkownika nastąpi sprawdzenie, czy pełni on rolę podaną w wywołaniu. Jeżeli tak, to kolejne akcje złożenia są wykonywane w kontekście nowego użytkownika. Po zakończeniu wszystkich akcji kontekst użytkownika zostanie odtworzony do stanu przed operacją. Błędna autoryzacja lub nie pełnienie wymaganej roli powoduje przerwanie wykonywania złożenia akcji. Zamiast akcji *Actions* można też stosować mechanizm akcji złożonych inicjowany wywołaniem *ActionSet*.

Taki sposób użycia akcji *SendControls* pozwala chronić obiekty z wbudowanymi mechanizmami sterującymi, np. *Tekst,* czy *Słupek*. Obiekty wykonujące sterowania (lub inne czynności) przy pomocy bezpośredniego wykonania akcji chroni się w podobny sposób.

Właściwości podstawowe		
	Aktywny	Tak
	Tryb przełącznika	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Tekst wyłączenia	Włącz
	Tekst włączenia	
Zdarzenia		
	Wyłączenie przycisku	*^Actions(ConfirmRole(Kierownik,$Always), SendControls($Diagram))*

W tym przypadku obiekt przycisk w obsłudze zdarzenia *Wyłączenie przycisku* żąda weryfikacji roli użytkownika.

Akcja *ConfirmRole* ma też inne zastosowania. Możemy jej użyć jeżeli chcemy, żeby przez wykonaniem operacji zalogowany operator ponownie się autoryzował. Pozwoli to zabezpieczyć się przed sytuacją, że przy chwilowo nieobecnym operatorze, do komputera podeszła przypadkowa osoba.

7 Parametryzacja funkcji interaktywnych w obiektach pasywnych

Duża część z dostępnych klas obiektów nie posiada żadnych wbudowanych mechanizmów interakcji z użytkownikiem. Ich standardowa funkcjonalność ogranicza się do wyświetlania informacji. Jednak istnieje możliwość dodania własnych funkcji interaktywnych do obiektów każdego typu. Można też rozszerzyć funkcje interaktywne dla obiektów, które już takie funkcje posiadają (np. *Przycisk)*.

Dodanie funkcji interaktywnych do obiektu opiera się głównie na obsłudze zdarzeń związanych z operacjami myszki (np. *Kliknięcie prawym przyciskiem*) i przyciśnięciami klawiszy. Uzupełnieniem jest zbiór funkcji zwracających stan myszki i klawiatury (np. *IsMouseOver* i *IsShiftPressed*), które użyte w definicjach właściwości obiektów, pozwalają uzależnić wygląd obiektów od działań operatora.

7.1 Nawigacja poprzez tekstowe łącza

Efekt, który chcemy uzyskać w tym przykładzie, to użycie na diagramach tekstowych łączy podobnych do łączy używanych w przeglądarce. Jako obiekt bazowy zostanie użyty obiekt klasy *Tekst* wyświetlający statyczny napis. Kliknięcie na napisie ma spowodować wykonanie akcji. Jednocześnie chcemy, żeby aktywność łącza była pokazywana wizualnie.

Właściwości podstawowe		
	Aktywny	Tak
	Typ kursora	Hand
Właściwości stanowe, stan podstawowy		
	Tekst	*Schemat wentylatora*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=IsMouseOver()*
	Kolor	*Blue*
	Styl czcionki	*Podkreślenie*
Zdarzenia		
	Kliknięcie lewym przyciskiem	*^OpenWindow(null,Info,Wentylator,null)*

Właściwość *Aktywny* ustawiona na *Tak* włącza pełną obsługę funkcji interaktywnych. Bez tego działałaby tylko obsługa zdarzenia kliknięcia przyciskiem myszki.

Właściwość *Typ kursora* określa postać kursora myszki w momencie najechania kursorem w obszar zajmowany przez obiekt.

Grupa stanowa kontrolowana wyrażeniem warunku *=IsMouseOver()* uaktywnia się w momencie najechania kursorem myszki w obszar obiektu. W efekcie tego kolor tekstu zmieniany jest na niebieski, a tekst wyświetlany jest z podkreśleniem.

Akcja *OpenWindow* w obsłudze zdarzenia *Kliknięcie lewym przyciskiem* powoduje otwarcie diagramu *Wentylator* w panelu *Info* bieżącego okna.

7.2 Podłączenie menu kontekstowego do obiektu

Chcemy uzyskać powszechnie używany mechanizm tak zwanego menu kontekstowego. Przyciśnięcie prawego przycisku myszki w momencie, gdy jej kursor znajduje się obszarze obiektu, ma spowodować wyświetlenie menu kontekstowego.

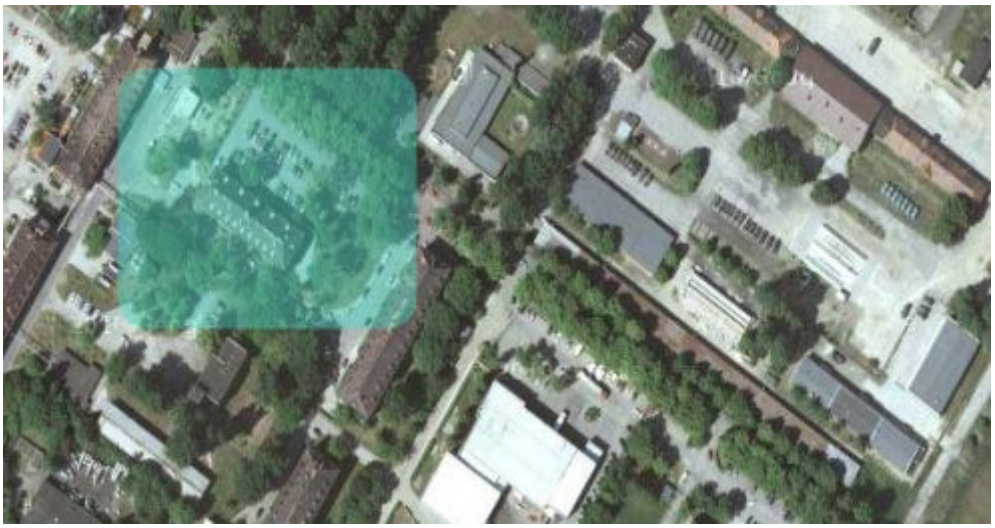
Jako obiekt bazowy może zostać użyty obiekt dowolnej klasy (za wyjątkiem obiektów kontrolkowych, np. *Przeglądarka*, które przejmują obsługę myszki). Rozpoczynamy od utworzenia i zdefiniowania menu kontekstowego. Użycie menu w obiekcie wymaga jedynie użycia akcji *ShowMenu* w obsłudze zdarzenia *Wciśnięcie prawego przycisku.*

Właściwości podstawowe		
	Zmienna główna	P1000
Zdarzenia		
	Wciśnięcie prawego przycisku	*^ShowMenu(MenuCtx,"v="+Attribute(Name))*

Akcja *ShowMenu* otwiera menu o nazwie *MenuCtx*, przekazując mu jako parametr nazwę zmiennej głównej obiektu, w kontekście którego menu zostało otwarte.

7.3 Strefy aktywności diagramu

W kolejnym przykładzie chcemy stworzyć mechanizm polegający na tym, że kliknięcie w wybranym obszarze diagramu spowoduje wykonanie właściwej akcji operatorskiej. W aktywnym obszarze diagramu może znajdować się dowolny zestaw obiektów. Obszar aktywny ma zostać uwidoczniony dopiero w momencie najechania na niego kursorem myszki. Poniższa ilustracja pokazuje obszar aktywności na tle dużego obrazka.



Jako obiekt, na bazie którego utworzymy strefę aktywności użyjemy obiekt klasy *Kształt.*

Właściwości podstawowe		
	Aktywny	Tak
	Typ kursora	Hand
	Typ kształtu	Prostokąt
	Promień zaokrąglenia	15
Właściwości stanowe, stan podstawowy		
	Kolor	*Transparent*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=IsMouseOver()*
	Kolor	*Turquoise*
	Przezroczystość	*0,4*
Zdarzenia		
	Kliknięcie lewym przyciskiem	^OpenWindow(null,Info,Askom,null)

Właściwość *Aktywny* ustawiona na *Tak* włącza pełną obsługę funkcji interaktywnych. Bez tego działałaby tylko obsługa zdarzenia kliknięcia przyciskiem myszki.

Właściwość *Typ kursora* określa postać kursora myszki w momencie najechania kursorem w obszar zajmowany przez obiekt.

W grupie podstawowej właściwość *Kolor* ustawiona jest na *Transparent*. Powoduje to, że w normalnym stanie obiekt pozostaje niewidoczny. Grupa stanowa kontrolowana wyrażeniem warunku *=IsMouseOver()* uaktywnia się w momencie najechania kursorem myszki w obszar obiektu. W efekcie tego zmienia się kolor obiektu i kształt staje się widoczny. Jednocześnie ustawienie stopnia przezroczystości powoduje, że obiekty położone pod kształtem pozostają widoczne.

Akcja *OpenWindow* w obsłudze zdarzenia *Kliknięcie lewym przyciskiem* powoduje otwarcie diagramu *Askom* w panelu *Info* bieżącego okna w momencie kliknięcia przyciskiem myszki w obszarze obiektu.

7.4 Operacje samopowtarzalne

Większość zdarzeń związanych z obsługą myszki ma charakter jednorazowy. W zestawie zdarzeń jest jednak para zdarzeń *Trzymanie lewego przycisku* i *Trzymanie prawego przycisku*, które są wywoływane wielokrotnie, dopóki wciśnięty jest odpowiedni przycisk i kursor myszki jest w obszarze obiektu.

Wykorzystamy zdarzenie *Trzymanie lewego przycisku* do inkrementacji wartości zmiennej wykonywanej w trakcie wciśnięcia przycisku. Użyty zostanie obiekt klasy *Obraz*.

Właściwości podstawowe		
	Aktywny	Tak
	Typ kursora	UpArrow
	Zmienna główna	P1000
Właściwości stanowe, stan podstawowy		
	Obrazek	*UpHand*
Zdarzenia		
	Trzymanie lewego przycisku	*^SetVariable("#",Variable()+1)*

Właściwość *Aktywny* ustawiona na *Tak* włącza pełną obsługę funkcji interaktywnych. Właściwość *Typ kursora* określa postać kursora myszki w momencie najechania kursorem w obszar zajmowany przez obiekt.

Akcja w obsłudze zdarzenia zmienia wartość zmiennej głównej (użyto skróconej nazwy *#*), poprzez zwiększenie jej poprzedniej wartości, odczytywanej funkcją *Variable()*,o jeden.

Okres zdarzenia przytrzymania deklarowany jest w ustawieniach stanowiska.

Alternatywą do pokazanej tutaj metody powtarzania akcji jest użycie obiektu klasy *Przycisk*, który ma mechanizm powtarzania wbudowany bezpośrednio w swoją logikę działania.

7.5 Obsługa klawiatury

Oprócz obsługi zdarzeń pochodzących od myszki można też reagować na zdarzenia przyciśnięcia klawiszy klawiatury. Przykład pokazuje obiekt klasy *Obraz*, który w reakcji na przyciśnięcie klawisza 'q' inkrementuje wartość zmiennej, a po przyciśnięciu klawisza 'w' ją dekrementuje.

Właściwości podstawowe		
	Aktywny	Tak
	Zmienna główna	P1000
Właściwości stanowe, stan podstawowy		
	Obrazek	*Arrows*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=IsSelected()*
	Barwa	*0,5*
Zdarzenia		
	Wciśnięcie klawisza	*^Actions(* *Perform(LastKeyPressed()==Q,SetVariable("#",Variable()+1),Nothing()),* *Perform(LastKeyPressed()==W,SetVariable("#",Variable()-1),Nothing()))*

Właściwość *Aktywny* ustawiona na *Tak* włącza pełną obsługę funkcji interaktywnych. Pozwala na wyselekcjonowanie obiektu przez użytkownika.

Grupa stanowa kontrolowana wyrażeniem warunku *=IsSelected()* uaktywnia się, gdy obiekt zostanie wybrany przez użytkownika. W efekcie tego zmieniany jest wygląd obrazka przez modyfikację właściwości *Barwa*.

Akcja *Actions* w obsłudze zdarzenia jest złożeniem dwóch warunkowych akcji *Perform*. Z kolei każda z akcji *Perform* sprawdza, czy przyciśnięty został właściwy klawisz i jeżeli test wypadł pomyślnie, to inkrementuje lub dekrementuje wartość zmiennej głównej.

Alternatywnymi metodami obsługi klawiatury są tzw. klawisza globalne deklarowane dla całej aplikacji oraz skróty klawiszowe obiektów klasy *Przycisk*. Podstawową różnicą w działaniu skrótów obiektu *Przycisk* jest to, że te skróty działają także wtedy, gdy obiekt nie jest wyselekcjonowany.

8 Wykorzystanie efektu przezroczystości

Jednym z mechanizmów, które można wykorzystać przy tworzeniu aplikacji jest przezroczystość obiektów i okien. Pozwala ona uzyskać ciekawe efekty graficzne oraz efektywnie wykorzystać powierzchnię ekranu poprzez wyświetlanie kilku informacji jednej nad drugą.

8.1 Przezroczystość okien

Stopień przezroczystości okna jest deklarowany we właściwości *Przezroczystość.* Przyjmuje ona wartości od 0 do 1, gdzie wartość 0 oznacza okno całkowicie przezroczyste, a wartość 1 okno bez efektu przezroczystości. Ustawienie wartości mniejszej od 1 powoduje, że całe okno (ramka i wyświetlany diagram z obiektami) są przezroczyste. Nie ma znaczenia sposób zdefiniowania obiektów diagramu.

8.2 Przezroczystość obiektów

Każdy obiekt może mieć indywidualnie kontrolowany stopień przezroczystości. Jest on ustawiony we właściwości *Przezroczystość,* znajdującej się w grupie właściwości stanowych. Przyjmuje ona wartości od 0 do 1, gdzie wartość 0 oznacza obiekt całkowicie przezroczysty, a wartość 1 obiekt nieprzezroczysty. Efekt przezroczystości dotyczy wszystkich graficznych elementów obiektu.

	Właściwości podstawowe	
	Zmienna główna	TranspValue
	Właściwości stanowe, stan podstawowy	
	Przezroczystość	#

Stopień przezroczystość jest sterowany wartością zmiennej *TranspValue*. Połączone z mechanizmem ustawiania zmiennej (np. obiekt *Słupek* z włączonym suwakiem) pozwala na płynne sterowanie stopniem przezroczystości obiektu przez użytkownika.

8.3 Ukrywanie obiektów

Czasami zachodzi potrzeba całkowitego ukrycia obiektu. Można to wykonać na dwa sposoby.

Właściwości podstawowe		
	Zmienna główna	P1000
Właściwości stanowe, stan podstawowy		
	Widoczny	*Nie*
	Tekst	*Błąd pomiaru*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=VarIsNotGood()*
	Widoczny	*Tak*

W tym przykładzie obiekt *Tekst* pozostaje zazwyczaj niewidoczny. Dopiero w momencie, gdy kontrolowana zmienna ma niepoprawny status, zmieniana jest wartość atrybutu *Widoczny* i obiekt wyświetla ostrzeżenie o błędzie odczytu zmiennej.

Właściwości podstawowe		
	Zmienna główna	P1000
Właściwości stanowe, stan podstawowy		
	Tekst	*Błąd pomiaru*
	Przezroczystość	*0*
Właściwości stanowe, stan numer 1		
	Warunek stanu	*=VarIsNotGood()*
	Przezroczystość	*1*

Ten przykład wizualnie działa w sposób identyczny do poprzedniego. Do ukrywania obiektu wykorzystywana jest przezroczystość. **Różnica polega na tym, że obiekt przezroczysty (nawet całkowicie) pozostaje na diagramie i może reagować na przyciśnięcie klawiszy myszki. Obiekt z ustawioną właściwością *Widoczny* takiej możliwości nie ma.**

8.4 Wykorzystanie koloru Transparent

Kolejnym elementem wpływającym na widoczność obiektu jest użycie koloru *Transparent.* Wybór tego koloru we właściwości związanej z pewnym elementem składowym obiektu, z reguły powoduje, że ten element nie jest wyświetlany.

Właściwości stanowe, stan podstawowy		
	Widoczny	*Tak*
	Kolor wskazówki	*Red*
	Kolor tła	*Transparent*
	Kolor otoczki	*Transparent*
	Kolor podziałki	*Transparent*
	Kolor czcionki	*Transparent*
	Przezroczystość	*1*

Pokazana powyżej parametryzacja obiektu *Miernik* powoduje, że wyświetlana jest tylko wskazówka miernika. Nakładając taki obiekt na inny, w pełni widoczny miernik, można uzyskać efekt miernika z dwoma wskazówkami.

9 Sterowanie zachowaniem obiektów

W tym rozdziale opiszemy metody zmiany wyglądu lub zachowania się obiektów w reakcji na interaktywne działania operatora. Metody tu opisane pozwalają na dostosowanie wyglądu diagramu w trakcie jego wyświetlania w trybie wykonania aplikacji.

9.1 Użycie zmiennych wirtualnych

Typowym mechanizmem, który możemy użyć to sterowanie wyglądem lub zachowaniem obiektów poprzez dodatkowe zmienne zdefiniowane w kanale wirtualnym (typu *None*). Obiekty kontrolowane uzależniają swój stan od takich zmiennych, a wartości zmiennych są sterowane za pośrednictwem innych obiektów. Cechą tej metody jest to, że wybrane przez użytkownika ustawienia są trwałe, po kolejnym zamknięciu i otwarciu diagramu pozostają niezmienione.

W metodzie zmiennych wirtualnych istotne jest też poprawne ustawienie wartości początkowych. Można tu zastosować akcję *SetVariable* wykonywaną w ramach zadania terminarza wykonywanego na starcie aplikacji lub w obsłudze zdarzenia *Otwarcie diagramu*.

Właściwości podstawowe		
	Główna zmienna	*A1000*
Właściwości stanowe, stan podstawowy		
	Widoczny	*=Variable(ShowNames)*
	Tekst	*@Name*

Pokazy wyżej fragment obiektu tekst uzależnia swój stan widoczności od wartości zmiennej wirtualnej *ShowNames*. Jeżeli jest ona niezerowa, to wyświetla na diagramie nazwę zmiennej.

Właściwości podstawowe		
	Aktywny	*Tak*
	Główna zmienna	*ShowNames*
	Zmienna sterowana	*#*
	Typ kursora	*Hand*
	Typ przycisku	*Standardowy przycisk Windows*
	Tryb przełącznika	*Nie*
	Sterowanie natychmiastowe	*Tak*
Właściwości stanowe, stan podstawowy		
	Włączony	*Nie*
	Wartość wyłączenia	*1*
	Tekst wyłączenia	*Pokaż nazwy*
Właściwości stanowe, stan 1		
	Warunek stanu	*=Variable()==0*
	Wartość wyłączenia	*0*
	Tekst wyłączenia	*Ukryj nazwy*

Powyższy obiekt klasy *Przycisk* służy do przełączania wartości zmiennej *ShowNames* z 0 na 1 lub na odwrót, w efekcie włączając lub wyłączając pokazywanie na diagramie obiektów wyświetlających nazwy zmiennych.

9.2 Modyfikacja właściwości obiektów

Kolejną metoda sterowania obiektami w trakcie pracy jest zastosowanie akcji operatorskiej *SetProperty*. Akcja ta pozwala zmienić wartość dowolnej właściwości w dowolnym nazwanym obiekcie. Zmiana dotyczy jednak tylko wartości właściwości. Po kolejnym zamknięciu i otwarciu diagramu, wartości właściwości zostaną ponownie wyliczone na podstawie ich definicji zapisanych w pliku diagramu.

Właściwości podstawowe		
	Nazwa elementu	***Chart***
	Wyświetlaj legendę	***Nie***

Powyższy fragment definicji obiektu klasy *Wykres* pokazuje obiekt o nazwie *Chart*, który w normalnym stanie nie wyświetla legendy.

Właściwości podstawowe		
	Aktywny	***Tak***
	Typ kursora	***Hand***
	Typ przycisku	***Standardowy przycisk Windows***
	Tryb przełącznika	***Nie***
	Sterowanie natychmiastowe	***Tak***
Właściwości stanowe, stan podstawowy		
	Włączony	***Nie***
	Tekst wyłączenia	***Pokaż legendę***
Zdarzenia		
	Wyłączenie przycisku	^SetProperty(Chart,ShowLegend,true)

Właściwości podstawowe

	Aktywny	***Tak***
	Typ kursora	***Hand***
	Typ przycisku	***Standardowy przycisk Windows***
	Tryb przełącznika	***Nie***
	Sterowanie natychmiastowe	***Tak***
Właściwości stanowe, stan podstawowy		
	Włączony	***Nie***
	Tekst wyłączenia	***Ukryj legendę***
Zdarzenia		
	Wyłączenie przycisku	^SetProperty(Chart,ShowLegend,false)

Powyższa para obiektów *Przycisk* zmienia wartość właściwości *ShowLegend* wszystkich obiektów, których nazwa jest równa *Chart*. W efekcie przyciski sterują pokazywaniem legendy przez obiekty wykresów umieszczonych na diagramie.

10 Zastosowania obiektów klasy Przycisk

Jednym z częściej stosowanych obiektów w aplikacjach jest obiekt klasy *Przycisk*. Zakres jego użycia jest bardzo szeroki, od organizacji schematu przełączania okien aplikacji po różnorodne sterowania zmiennymi procesowymi.

10.1 Przycisk jednopozycyjny

Najprostszym i prawdopodobnie najczęstszym przypadkiem jest użycie w trybie jednopozycyjnym. Działa wtedy w sposób podobny do standardowych przycisków systemu Windows. Przycisk pozostaje w trybie wyciśniętym. W momencie wciśnięcia przycisku myszki zmienia stan na wciśnięty, a w momencie puszczenia powraca automatycznie do stanu wyciśniętego. W trakcie tego procesu wykonywana jest akcja realizująca funkcję przycisku.

Ten tryb pracy wykorzystywany jest do prostych jednorazowych operacji, np. przełączania okien lub bezwarunkowych operacji sterujących.

10.1.1 Przycisk jednopozycyjny wykonujący akcję operatorską

Właściwości podstawowe		
	Aktywny	Tak
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Płaski	=!IsMouseOver()
	Tryb przełącznika	Nie
	Skrót klawiszowy	Alt+Q
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Kolor włączenia	LightGreen
	Tekst wyłączenia	Schemat
	Tekst włączenia	
Zdarzenia		
	Wyłączenie przycisku	*^OpenWindow(null,Info,Schemat,null)*

Właściwość *Typ kursora* powoduje, że kursor myszki zmienia się po najechaniu na obszar przycisku informując o gotowości do wykonania operacji. Właściwość *Typ przycisku* określa kształt przycisku. Jednocześnie wyrażenie użyte we właściwości *Płaski* powoduje, że dopóki kursor myszki nie najedzie na obiekt, to przycisk wyświetlany jest w postaci płaskiej. Dopiero po najechaniu obiekt wyświetli się w pełnej formie trójwymiarowej. Oczywiście można tego efektu nie stosować i ustawić wartość właściwości *Płaski* na *Nie.*

Ustawienie właściwości *Tryb przełącznika* na *Nie* powoduje, że przycisk pracuje w trybie jednopozycyjnym. Podanie skrótu klawiszowego powoduje, że oprócz użycia myszki, użytkownik może wykonać operację przy pomocy klawiatury.

Właściwość *Włączony* określa w jaki sposób wyświetlić przycisk. W przypadku trybu jednopozycyjnego powinna to być wartość *Nie*.

Większość właściwości stanowych określa atrybuty wyświetlania obiektu. Z reguły występują parami, dla stanu wciśnięcia i wyciśnięcia przycisku. W pokazanym przykładzie *Tekst wyłączenia* określa opis przycisku. Brak podanego tekstu dla właściwości *Tekst włączenia* powoduje, że niezależnie od stanu przycisku będzie wyświetlany zawsze tekst wyłączenia.

Użycie właściwości *Kolor włączenia* powoduje, że w momencie chwilowego wciśnięcia przycisku, kolor jego tła zmieni się sygnalizując dodatkowo wykonanie operacji. Ustawienie koloru i jakichkolwiek innych atrybutów włączenia nie jest oczywiście konieczne.

Akcja operatorska podana w obsłudze zdarzenia *Wyłączenie przycisku* wymienia diagram w panelu *Info* bieżącego okna. W przypadku obiektów *Przycisk* należy korzystać ze zdarzeń dedykowanych dla tej klasy obiektów, czyli *Włączenie przycisku i Wyłączeniu przycisku.* Chociaż w niektórych przypadkach podobne działanie można uzyskać stosując uniwersalne zdarzenia obsługi myszki (*Kliknięcie lewym przyciskiem* itp.*)*, to lepiej stosować zdarzenia dedykowane, które uwzględniają stan przycisku.

10.1.2 Przycisk jednopozycyjny wykonujący operację sterującą

W przypadku konieczności wykonania operacji związanych z wysłaniem wartości sterujących do zmiennych procesowych można stosować parametryzację opartą o wykonanie akcji operatorskiej *SetVariable*. Z reguły łatwiej jest jednak wykorzystać funkcjonalność wbudowaną w obiekt przycisk, przeznaczoną do wykonywania bezpośrednich sterowań.

Właściwości podstawowe		
	Aktywny	Tak
	Zmienna sterowana	Command
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Wartość wyłączenia	0x8
	Tekst wyłączenia	Włącz
	Tekst włączenia	

Zdefiniowanie właściwości *Wartość wyłączenia* powoduje, że w trakcie cyklu włączenie/wyłączenie przycisku, do zmiennej sterowanej *Command* zostanie wysłana podana szesnastkowo wartość 0x8 (16).

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level
	Zmienna sterowana	#
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Wartość wyłączenia	=Variable()+1
	Tekst wyłączenia	Podwyższ
	Tekst włączenia	

W kolejnym przykładzie wartość wyłączenia jest tworzona dynamicznie. Jest ona równa aktualnej wartości zmiennej głównej powiększonej o 1. Wciśnięcie i puszczenie przycisku spowoduje zwiększenie wartości zmiennej głównej o 1. Nazwa zmiennej sterowanej zdefiniowana jest przez # - oznacza to użycie nazwy zmiennej głównej.

10.2 Przycisk jednopozycyjny z powtarzaniem

Obiekt *Przycisk* pozwala też na pracę w trybie automatycznego powtarzania operacji sterującej lub akcji operatorskiej.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level
	Zmienna sterowana	#
	Typ kursora	Hand

	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
	Z powtarzaniem	Tak
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Wartość włączenia	=Variable()+1
	Wartość końca powtarzania	0
	Tekst wyłączenia	Podwyższ
	Tekst włączenia	

Tryb powtarzania został włączony poprzez ustawienie właściwości *Z powtarzaniem*. W trakcie przytrzymania wciśniętego przycisku wartość zdefiniowana w *Wartość włączenia* zostanie wielokrotnie wysłana do zmiennej sterowanej (tożsamej ze zmienną główną). W efekcie wartość zmiennej będzie inkrementowana w sposób ciągły. W momencie puszczenia przycisku zostanie wysłana wartość zdefiniowana we właściwości *Wartość końca powtarzania.* Podobny schemat dotyczy działań opartych na obsłudze zdarzeń. Cyklicznie wykonywane jest zdarzenie *Włączenie przycisku*, a sekwencja kończona jest zdarzeniem *Koniec powtarzania*.

Okres powtarzania sterowania lub akcji jest zdefiniowany w panelu parametrów stanowisk w zakładce *Ustawienia*.

10.3 Przycisk jednopozycyjny z blokadą

W przypadku ważnych sterowań istotne może być zapobieżenie przypadkowemu przyciśnięciu przycisku. Jeżeli nie chcemy korzystać z opisanego dalej mechanizmu opóźnionych sterowań, to można połączyć przyciśnięcie przycisku z wymaganiem jednoczesnego przyciśnięcia klawisza klawiatury.

Właściwości podstawowe		
	Aktywny	=IsAltPressed()
	Główna zmienna	Level
	Zmienna sterowana	#
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d

	Tryb przełącznika	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	Nie
	Wartość wyłączenia	=Variable()+1
	Kolor wyłączenia	=IsAltPressed()?LightGreen:LightGray
	Tekst wyłączenia	Podwyższ
	Tekst włączenia	

Jest to modyfikacja wcześniejszego przykładu zwykłego przycisku jednopozycyjnego inkrementującego wartość zmiennej. Właściwość *Aktywny* jest równa *true (TAK)* tylko wtedy, gdy użytkownik przyciska klawisz *Alt*. I tylko wtedy obiekt pozwala na wciśnięcie przycisku. Dodatkowo *Kolor wyłączenia* jest także modyfikowany w zależności do stanu klawisza *Alt* w celu sygnalizacji aktywności przycisku.

10.4 Przycisk dwupozycyjny

W kolejnym przykładzie zbudujemy przycisk dwupozycyjny, w którym o stanie wciśnięcia będzie decydowała wartość zmiennej procesowej. W momencie kliknięcia przycisk zmieni stan na przeciwny, jednocześnie wykonując odpowiednią operację sterującą.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Flag1
	Zmienna sterowana	#
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Tak
	Sterowanie natychmiastowe	Tak
Właściwości stanowe, stan podstawowy		
	Włączony	=Variable()

	Wartość wyłączenia	0
	Wartość włączenia	1
	Tekst wyłączenia	Off
	Tekst włączenia	On

Cała funkcjonalność zdefiniowana jest w pojedynczym stanie. Ustawienie właściwości *Tryb przełącznika* na *Tak* powoduje, że przycisk wykonuje ruch tylko w jedną stronę i wykonuje pojedyncze sterowanie.

Jeżeli zmienna *Flag1* jest równa 0, to przycisk jest wyciśnięty i wyświetlany jest opis *Off*. Po wciśnięciu klawisza myszki, przycisk zostaje wciśnięty, a opis zmieniony na *On*. W chwili puszczenia klawisza myszki *Wartość włączenia* jest wysyłana do zmiennej *Flag1*. W efekcie przycisk pozostanie w stanie wciśnięcia (włączenia), ponieważ wartość zmiennej głównej Flag1 zmieni się na niezerową. Po wykonaniu operacji może nastąpić chwilowy powrót przycisku do stanu pierwotnego *Off*, do momentu kiedy nowa wartość zmiennej głównej zostanie zwrotnie odczytana.

10.5 Przycisk dwupozycyjny z opóźnionym sterowaniem

Częstym przypadkiem jest konieczność użycia przycisku dwupozycyjnego, w którym operacja sterująca nie jest wykonywana od razu, lecz wymaga osobnego potwierdzenia. Schemat działania jest bardzo podobny do zwykłego przycisku dwupozycyjnego.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Flag1
	Zmienna sterowana	#Control
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Tak
	Sterowanie natychmiastowe	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	=Variable()

	Wartość wyłączenia	1
	Wartość włączenia	2
	Kolor włączenia	=HasWaitingControl()?LightGreen:LightGray
	Kolor wyłączenia	=HasWaitingControl()?LightCoral:LightGray
	Tekst wyłączenia	Off
	Tekst włączenia	On

Sterowanie natychmiastowe ustawione na *Nie* powoduje, że kliknięcie na przycisku powoduje tylko zmianę jego stanu na przeciwny. Przycisk jest w tym stanie do momentu zatwierdzenia sterowania akcją *SendControls* lub jego anulowania akcją *CancelControls*. Istotne jest, aby wizualnie pokazać, że obiekt oczekuje na zatwierdzenie operacji. W przykładzie zmieniany jest kolor przycisku. Wyrażenie warunkowe wykorzystuje funkcję *HasWatingControl*, która przyjmuje wartość *true*, gdy obiekt jest w stanie oczekiwania.

W przykładzie pokazano też, że sterowanie (komenda) jest wysyłane do innej zmiennej niż zmienna główna decydująca o stanie przycisku. Nazwa zmiennej sterowanej tworzona jest notacją sufiksową i w tym przykładzie jest to *Flag1Control*.

Podobnie jak w poprzednim przykładzie, po wykonaniu operacji może dojść do chwilowego powrotu przycisku do stanu początkowego, na czas potrzebny do wykonania sterowania, wykonania komendy przez sterownik i zwrotnego odczytu zmiennej głównej.

10.6 Wyłącznik

W tym przykładzie pokazana zostanie konfiguracja obiektu *Przycisk* w sposób, w którym jego działanie jest zgodne z działaniem obiektu *Wyłącznik* systemu Asix w wersji klasycznej. Przycisk pozostaje w stanie wyciśniętym, a jego opis określa aktualny tryb pracy urządzenia (zatrzymane/uruchomione). W momencie wykonania sterowania przycisk pozostaje wciśnięty do momentu zatwierdzenia operacji. Jednocześnie opis przycisku zmienia się i pokazuje jaka operacja zostanie wykonana (uruchom/zatrzymaj).

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	MStatus
	Zmienna sterowana	#Control

	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
	Sterowanie natychmiastowe	Nie
Właściwości stanowe, stan podstawowy		
	Włączony	=HasWaitingControl()
	Wartość wyłączenia	1
	Kolor włączenia	LightGreen
	Tekst wyłączenia	Uruchomiony
	Tekst włączenia	Zatrzymaj
Właściwości stanowe, stan 1		
	Warunek stanu	=Variable()&1
	Wartość wyłączenia	2
	Tekst wyłączenia	Zatrzymany
	Tekst włączenia	Uruchom

Obiekt używa dwóch zmiennych. Zmienna główna służy do określenia stanu urządzenia. W zależności od jej wartości zmieniają się opisy przycisku i wartość komendy sterującej. Wykorzystany został mechanizm stanów obiektu. Zmienna sterująca o nazwie powstałej z połączenia nazwy zmiennej głównej i sufiksu *Control* służy do wysyłania komend sterujących (2 oznacza komendę uruchomienia, a 1 zatrzymania urządzenia).

W powyższym schemacie działania istotne jest użycie wyrażenia *HasWaitingControl* we właściwości *Włączony*. W momencie wciśnięcia przycisku obiekt wchodzi w stan posiadania ważnej wartości sterującej. Pozostanie w tym stanie aż do momentu wykonania sterowania (akcją *SendControls)* lub jego anulowania. W tym czasie wartość wyrażenia jest równa *true i p*owoduje utrzymanie przycisku w stanie wciśniętym. Stan ważnej wartości jest dodatkowo sygnalizowany przez kolor podany we właściwości *Kolor włączenia*.

10.7 Sterowania bitowe

Czasami wykonywane sterowania mogą zmienić tylko pojedyncze bity kontrolowanej zmiennej. W tego typu przypadkach należy użyć obiektów przycisk w trybie sterowania bitowego. Poniższy przykład użycia pokazuje zmodyfikowany przypadek przycisku dwupozycyjnego.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Status
	Sterowanie bitowe	Tak
	Zmienna sterowana	#
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Tak
	Sterowanie natychmiastowe	Tak
Właściwości stanowe, stan podstawowy		
	Włączony	=Variable()&1
	Wartość wyłączenia	1
	Wartość włączenia	1
	Tekst wyłączenia	Off
	Tekst włączenia	On

Ustawienie właściwości *Sterowanie bitowe* na wartość *Tak* powoduje zmianę sposobu wykonywania operacji wysłania wartości do zmiennej sterowanej. Operacja wykonywana jest w dwóch krokach. Najpierw aktywnie odczytywana jest bieżąca wartość zmiennej sterowanej, następnie wyliczana i wysyłana jest nowa wartość zmiennej. Należy pamiętać o tym, że taki sposób pracy jest zdecydowanie mniej efektywny (pod względem komunikacji ze sterownikiem) niż zwykłe sterownia.

Stan przycisku jest kontrolowany wartością pierwszego bitu zmiennej głównej. Wyrażenie *Variable()&1* wykonuje operację iloczynu logicznego wartości zmiennej i liczby 1. W efekcie, przycisk jest wciśnięty gdy pierwszy bit jest ustawiony. Przy przejściu ze stanu wyłączenia(wyciśnięty) do włączonego, ustawiane są w zmiennej sterowane te bity, których wartość we właściwości *Wartość włączenia* jest równa 1 (w tym przypadku tylko najmłodszy bit). . Przy przejściu ze stanu włączenia(wciśnięty) do wyłączonego, zerowane są w zmiennej sterowane te bity, których wartość we właściwości *Wartość wyłączenia* jest równa 1 (w tym przypadku najmłodszy bit zmiennej zostanie ustawiony na 0).

W bardziej skomplikowanych przypadkach sterowań bitowych należy użyć akcji operatorskiej *SetVariable* w obsłudze zdarzeń *Włączenie przycisku* i *Wyłączenie przycisku*. Na przykład akcja

SetVariable("", 3, 0xf) spowoduje zmianę czterech najmłodszych bitów zmiennej sterowanej. Dwa najmłodsze bity zostaną ustawione na 1, a pozostałe na 0.

10.8 Grupowanie przycisków

Czasami konieczne jest użycie grupy przycisków służących do wykonywania wzajemnie sprzecznych operacji. Na poniższej ilustracji pokazane są przyciski, które włączają ruch urządzenia do przodu lub do tyłu.



Przyciski te powinny działać z potwierdzeniem. W wyniku tego użytkownik może wprowadzić oba przyciski w stan ważnej wartości sterującej. Wykonanie akcji *SendControls* spowodowałoby wykonanie dwóch operacji sterujących – końcowy efekt zależałby od kolejności definiowania obiektów.

Można jednak spowodować, żeby była możliwość wciśnięcia tylko jednego przycisku jednocześnie. Służą do tego grupy sterowań.

Przycisk 1

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Move
	Zmienna sterowana	#
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
	Sterowanie natychmiastowe	Nie
	Grupa sterowania	G1
Właściwości stanowe, stan podstawowy		

	Włączony	=HasWaitingControl()
	Wartość włączenia	1
	Kolor włączenia	LightGreen
	Tekst wyłączenia	<<<

Przycisk 2

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Move
	Zmienna sterowana	#
	Typ kursora	Hand
	Typ przycisku	Przycisk prostokątny 3d
	Tryb przełącznika	Nie
	Sterowanie natychmiastowe	Nie
	Grupa sterowania	G1
Właściwości stanowe, stan podstawowy		
	Włączony	=HasWaitingControl()
	Wartość włączenia	2
	Kolor włączenia	LightGreen
	Tekst wyłączenia	=”>>>”

W pokazanej parametryzacji istotna jest definicja właściwości *Grupa sterowań*. Nie mają znaczenia wartości właściwości, istotne jest tylko to, żeby były one identyczne dla obu przycisków. W danej chwili w stanie wciśnięcia (czyli ważnej wartości sterującej sprawdzanej funkcją *HasWaitingControl*) pozostanie tylko jeden z dwóch przycisków.

11 Operacje sterujące w obiektach klasy Tekst

Obiekty klasy *Tekst* oferują dwie metody wykonywania nastaw. W pierwszym trybie, stosowanym głównie w obiektach pokazujących wartości liczbowe zmiennych, użytkownik wprowadza bezpośrednio nową wartość zmiennej. W drugim trybie, stosowanym w obiektach opisujących tekstowo stany bitowe, sterowanie polega na wybraniu nowej nastawy z listy proponowanych wartości.

11.1 Sterowanie wartością liczbową wprowadzaną przez użytkownika

W tym trybie działania istotne jest, aby w wyraźny sposób rozróżnić poszczególne stany obiektu: stan pokazywania aktualnej wartości, stan wprowadzania nowej wartości i stan oczekiwania na wykonanie sterowania.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level
	Zmienna sterowana	#
	Wartość minimalna sterowania	@SteeringRangeFrom
	Wartość maksymalna sterowania	@SteeringRangeTo
	Tryb selekcji stanów	Nie
	Edycja przy selekcji	Nie
	Początkowa wartość edycji	Pusta
	Sterowanie natychmiastowe	Nie
Właściwości stanowe, stan podstawowy		
	Tekst	#
	Kolor	Black
	Kolor edycji	Blue
Właściwości stanowe, stan 1		
	Warunek stanu	=VarIsNotGood()
	Tekst	=Variable() + ”?”

	Kolor	Crimson
Właściwości stanowe, stan 2		
	Warunek stanu	=HasWaitingControl()
	Kolor	Cyan
	Styl czcionki	Kursywa

Właściwość *Aktywny* ustawiona na *Tak* włącza funkcje sterujące obiektu. Nazwa zmiennej sterowanej i głównej są identyczne. Określone są też limity wartości sterującej – limity pobierane są z odpowiednich atrybutów zmiennej głównej obiektu. Właściwość *Tryb selekcji* równa *Nie* oznacza, że nowa wartość będzie wprowadzana bezpośrednio przez użytkownika. Właściwość *Edycja przy selekcji* określa sposób rozpoczęcia edycji wartości nastawy. W przypadku *Nie*, po wybraniu obiektu trzeba przycisnąć klawisz *Enter* lub podwójnie kliknąć myszką. Dla wartości *Tak* edycja rozpocznie się natychmiast po wybraniu obiektu. Właściwość *Początkowa wartość edycji* określa tekst wyświetlany w obiekcie po rozpoczęciu edycji. Może to być tekst pusty, tekst ostatnio wprowadzony lub bieżący tekst wyświetlany.

Właściwość *Sterowanie natychmiastowe* ustawione na *Nie* oznacza, że po zakończeniu edycji (przyciskiem *Enter),* nowa wartość nastawy zostanie wysłana dopiero w momencie wykonania akcji sterującej *SendControls. Wartość Tak* oznaczałaby wysłanie natychmiast po zakończeniu edycji.

Właściwości stanu podstawowego określają kolor używany do wyświetlania wartości zmiennej w stanie normalnym oraz kolor używany w trakcie edycji nowej nastawy przez użytkownika.

Grupa właściwości numer 1 opisuje sposób wyświetlania tekstu, gdy status wartości zmiennej głównej jest niepoprawny. Zmieniony jest kolor tekstu oraz dodawany znak '?' po wartości zmiennej. Stan ten nie jest tutaj niezbędny, służy tylko do ilustracji parametryzacji sterowania w obiekcie wielostanowym.

Stan numer 2 określa sposób wyświetlenia nowej nastawy po tym jak użytkownik zakończył edycję, a zlecenie wykonania operacji (akcja *SendControls*) nie zostało jeszcze wykonane. Zmieniany jest kolor i styl czcionki. Stan kontrolujący oczekiwanie na wysłanie wartości z reguły umieszczany jest na końcu listy stanów, w ten sposób jego ustawienia mają pierwszeństwo. Odwrócenie kolejności w naszym przykładzie powodowałoby, że pomimo oczekiwania na sterowanie mógłby być użyty kolor tekstu zdefiniowany w grupie kontrolującej status zmiennej.

11.2 Sterowania z listy wyboru

Alternatywną metodą wykonywania sterowań poprzez obiekt *Tekst* jest użycie trybu selekcji stanów. Tryb ten stosowany jest z reguły w obiektach, które wyświetlają tekstowe opisy stanów. Sterowanie polega na wyborze właściwej pozycji z wyświetlanej listy możliwych operacji.



Zawartość listy sterowania wynika z definicji stanów obiektu.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	LineState
	Zmienna sterowana	#Control
	Typ kursora	Hand
	Tryb selekcji stanów	Tak
	Selekcja stanów przez menu	Tak
	Sterowanie natychmiastowe	Tak
Właściwości stanowe, stan podstawowy		
	Tekst	Stan nierozpoznany
	Kolor	Black
	Stan wybieralny	Nie
Właściwości stanowe, stan 1		
	Warunek stanu	#.0
	Tekst	Tryb ręczny
	Wartość stanu	8
	Maska bitowa stanu	8
	Stan wybieralny	#.0r
Właściwości stanowe, stan 2		
	Warunek stanu	#.1
	Tekst	Tryb awaryjny

	Wartość stanu	4
	Maska bitowa stanu	
	Stan wybieralny	#.1r
Właściwości stanowe, stan 3		
	Warunek stanu	#.3
	Tekst	Tryb automatyczny
	Wartość stanu	0
	Maska bitowa stanu	8
	Stan wybieralny	#.3r

Właściwość *Aktywny* ustawiona na *Tak* włącza funkcje sterujące obiektu. Nazwa zmiennej sterowanej jest tworzona na podstawie nazwy zmiennej głównej przez dodanie sufiksu *Control*. Właściwość *Tryb selekcji stanów* wybiera pracę w trybie selekcji, a *Selekcja stanów przez menu* określa, że wybór nowego stanu odbędzie się z automatycznie utworzonego menu kontekstowego. Wartość *Nie* oznaczałaby sekwencyjny wybór kolejnych stanów po każdym kliknięciu na obiekcie.

Właściwość *Sterowanie natychmiastowe* ustawione na *Tak* powoduje, że wykonanie sterowania nastąpi natychmiast po wyborze stanu. Możliwe jest oczywiście sterowanie opóźnione. W takim przypadku należałoby dodać sygnalizację stanu oczekiwania, np. przez użycie wyrażenia *HasWaitingControl()?Red:Black* w definicji właściwości *Kolor* lub dodanie kolejnego stanu.

Stan podstawowy obiektu definiuje wygląd obiektu w przypadku nieprawidłowej wartości zmiennej monitorowanej. Kolejne trzy stany służą do dekodowania wartości zmiennej na opis tekstowy, warunkiem stanu jest test ustawienia wybranego bitu. Właściwość *Stan wybieralny* określa, czy opis stanu ma pojawić się na liście wyboru operacji sterującej. Użyte wyrażenia powodują, że opis pojawi się na liście wtedy, gdy nie jest stanem aktualnym (wyrażenia są negacją warunków stanu).

W przykładzie użyto notacji prefiksowej # z podaniem numeru. Notacja ta pozwala odwołać się do stanu wybranego bitu zmiennej głównej. Dodanie *r* po numerze bitu oznacza odwrotną logikę – wartość zerowa bitu daje wynik *true*. Zapis #.0 jest równoważny wyrażeniu *Variable()&1*, a zapis *#.0r* to odpowiednik *!(Variable()&1)*.

Możliwe jest też definiowanie stanów obiektu służących wyłącznie do sterowania. W tym celu warunek stanu należy ustawić na *Nie*, a *Stan wybieralny* na *Tak*.

Właściwości *Wartość stanu* i *Maska bitowa stanu* określają sposób wykonania sterowania. Niezdefiniowanie maski oznacza sterowanie bezpośrednią wartością stanu. Zdefiniowana maska oznacza sterowanie bitowe. W pokazanym przykładzie stan numer 1 to zmiana czwartego bitu zmiennej sterowanej na 1, stan numer 2 to wysłanie wartości 4, a stan numer 3 to zmiana czwartego bitu na wartość 0.

Wygląd listy stanów (kolory i czcionka) jest parametryzowany w panelu *Ustawienia stanowisk* w zakładce *Ustawienia* w ramce *Ustawienia menu stanów*.

Podobnie wyglądającą funkcjonalność można uzyskać poprzez podłączenie do obiektu menu kontekstowego.

Zobacz też: *7.2. Podłączenie menu kontekstowego do obiektu*

12 Użycie suwaków w obiektach klasy Słupek

Podstawowym zadaniem obiektów klasy *Słupek* jest prezentacja wartości zmiennych procesowych w postaci słupków (poziomych lub pionowych). Można je też stosować jako obiekty statyczne do wyświetlenia na diagramie skali numerycznej.

W tym rozdziale zajmiemy się zasadami użycia suwaków wbudowanych w obiekty klasy *Słupek.*

12.1 Użycie suwaka do sterowania wartościami nastaw

W tym zastosowaniu słupek służy do pokazania aktualnej wartości nastawy, a jednocześnie pozwala na jej zmianę przy pomocy suwaka.

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	SliderSet
	Zmienna sterowana	#
	Wartość minimalna sterowania	@SteeringRangeFrom
	Wartość maksymalna sterowania	@SteeringRangeTo
	Styl wskaźnika	=IsMouseOver()?Holder:None
	Użyj limitów	Nie
	Wartość słupka	#

	Wartość wskaźnika	#
	Wartość minimalna	@SteeringRangeFrom
	Wartość maksymalna	@SteeringRangeTo
	Sterowanie natychmiastowe	Tak
Właściwości stanowe, stan podstawowy		
	Kolor wskaźnika	Transparent

Ponieważ wielkość słupka i położenia suwaka (wskaźnika) zależą od tej samej zmiennej, to zmienna sterowana została określona notacją skrótową #. Zakres zmienności suwaka zależy od zakresu podanego w parze właściwości *Wartość minimalna sterowania* i *Wartość maksymalna sterowania.* Zakres słupka zależy od pary właściwości *Wartość minimalna* i *Wartość maksymalna.* Ponieważ w naszym zastosowaniu słupek i suwak pokazują identyczną wartość, to oba zakresy zostały identycznie ustawione i bazują na ustawieniach zakresu sterowania zmiennej *SliderSet,* podanego w bazie definicji zmiennych. Bieżąca wartość słupka jest określona przez właściwość *Wartość słupka* – użycie notacji # oznacza pobranie wartości zmiennej głównej. W analogiczny sposób jest określone położenie suwaka (wskaźnika) we właściwości *Wartość wskaźnika*.

Użycie we właściwości *Styl wskaźnika* wyrażenia warunkowego *IsMouseOver()?Holder:None* powoduje, że w stanie normalnym suwak nie jest wyświetlany, natomiast po najechaniu kursorem myszki na obiekt, suwak się pokazuje i pozwala na zmianę nastawy.

Użycie koloru *Transparent* we właściwości *Kolor wskaźnika* nie powoduje ukrycia suwaka, a jedynie wyświetlenie go w formie półprzezroczystej.

12.2 Użycie suwaka do sterowania wartością zadaną z podglądem nowej nastawy

Kolejny przykład pokazuje, jak połączyć na jednym obiekcie klasy *Słupek* prezentację wartości pomiaru oraz jego wartości zadanej wraz z możliwością zmiany nastawy. Jednocześnie w trakcie zmieniania wartości zadanej (przeciągania suwaka) będzie pokazywana jej wartość w postaci tekstowej na osobnym obiekcie klasy *Tekst.*

Właściwości podstawowe		
	Aktywny	Tak
	Główna zmienna	Level

	Zmienna sterowana	#Set
	Zmienna podglądowa	#View
	Wartość minimalna sterowania	=Attribute("#Set",SteeringRangeFrom)
	Wartość maksymalna sterowania	=Attribute("#Set",SteeringRangeTo)
	Styl wskaźnika	=IsMouseOver()?Holder:Triangles
	Użyj limitów	Tak
	Wartość słupka	#
	Wartość wskaźnika	#Set
	Wartość minimalna	@DisplayRangeFrom
	Wartość maksymalna	@DisplayRangeTo
	Wartość LL	@LimitLoLo
	Wartość L	@LimitLo
	Wartość H	@LimitHi
	Wartość HH	@LimitHiHi
	Sterowanie natychmiastowe	Tak

Wysokość słupka będzie zależała od wartości zmiennej *Level*, a położenie suwaka(wskaźnika) od zmiennej *LevelSet* podanej w notacji sufiksowej. Do określenia położenia suwaka potrzebne jest podanie zakresu zmienności jego wartości we właściwościach *Wartość minimalna sterowania* i *Wartość maksymalna sterowania.* Użyte wyrażenia odwołują się do atrybutów zmiennej *LevelSet* zapisanych w bazie definicji zmiennych. Wysokość słupka zależy od zakresu podanego we właściwościach *Wartość minimalna* i *Wartość maksymalna.* W naszym przykładzie zakres ten określony jest przez atrybuty zakresu wyświetlania zmiennej głównej. Zakres zmienności słupka i suwaka nie muszą być zgodne, np. wielkość słupka może zależeć od wartości bezwzględnej, a wartość zadana (położenie suwaka) może być podawana procentowo.

Ustawienia właściwości *Użyj limitów, Wartość LL, Wartość L, Wartość H* i *Wartość HH* powoduje, że słupek będzie zmieniał kolor w zależności od przekroczenia poszczególnych limitów ostrzeżeń. Wartości limitów pobierana są z atrybutów zmiennej głównej *Level*.

Użycie we właściwości *Styl wskaźnika* wyrażenia warunkowego *IsMouseOver()?Holder:Triangles* powoduje, że w stanie normalnym położenie suwaka wyświetlane jest w postaci dwóch trójkątów na

tle paska słupka, natomiast po najechaniu kursorem myszki na obiekt, suwak zmienia kształt sygnalizując gotowość do wykonania nowej nastawy.

Definicja właściwości *Zmienna podglądowa #View* powoduje, że w trakcie przeciągania suwaka, wartość nastawy odpowiadającej jego chwilowej pozycji będzie ustawiana w zmiennej *LevelView*. Zmienna taka powinna być zdefiniowana w kanale wirtualnym, a jej wartość może być wyświetlana np. obiektem klasy *Tekst*.

Właściwości podstawowe		
	Główna zmienna	SliderView
Właściwości stanowe, stan podstawowy		
	Widoczny	=Variable()>-1e200
	Tekst	#

Jeżeli suwak nie jest w trakcie przeciągania, to zmienna podglądowa suwaka jest ustawiana na minimalną wartość typu Double. Wykorzystane jest to w definicji właściwości *Widoczny* obiektu *Tekst*. Obiekt jest widoczny na diagramie tylko wtedy, gdy suwak jest trakcie operacji wyboru nowej pozycji.

13 Animacja ruchu i zmiana rozmiaru obiektów

Rozdział pokazuje, w jaki sposób można dynamicznie zmieniać położenie i rozmiar obiektu na diagramie. Specyfika tego problemu polega na tym, że nie można bezpośrednio zmieniać właściwości obiektu odpowiedzialnych za położenie i rozmiar obiektu. Właściwości *X, Y, Szerokość, Wysokość* są zawsze podawane bezpośrednio.

W celu zmiany pozycji lub rozmiaru obiektu w trakcie działania aplikacji należy wykorzystać zdarzenie *Animacja* oraz akcje operatorskie *SetPosition, SetSize* i *SetBounds*. W zależności od potrzeb, w obsłudze zdarzenia *Animacja* należy zastosować jedną z wymienionych akcji z odpowiednio wyliczanymi parametrami położenia i/lub rozmiaru obiektu.

13.1 Zmiana pozycji

W poniższym przykładzie zmieniona zostanie pozycja obiektu typu *Obraz*. Położenie w osi X będzie sterowane zmienną procesową *CarPosX*, której wartość z zakresu od 0 do 100 określa przesunięcie obrazu w pikselach względem pozycji początkowej.

Właściwości podstawowe		
	Główna zmienna	CarPosX
Właściwości stanowe, stan podstawowy		
	Nazwa obrazu	*Car*
Zdarzenia		
	Animacja	*^SetPosition(RelToAbsX(100+Variable(),null)*

Pozycja początkowa określona jest na 100. Do pozycji początkowej dodawana jest wartość zmiennej *CarPosX.* Tak wyliczona pozycja w pikselach musi być następnie przeliczona na wartość bezwzględną przy pomocy funkcji *RelToAbsX*. Wszystkie parametry funkcji zmiany położenia i rozmiaru są przekazywane w wartościach bezwzględnych niezależnych od rozmiaru diagramu. Wartość bezwzględna 1 000 000 zawsze odpowiada prawej krawędzi diagramu. Wartość *null* w drugim parametrze akcji *SetPosition* oznacza, że współrzędna Y obiektu nie ulega zmianie.

W podobny sposób można zmieniać rozmiar obiektu, ale w tym przypadku należałoby użyć akcję *SetSize* lub *SetBounds.*

Akcja pokazana w powyższym przykładzie ma jednak jedną podstawową wadę – działa dobrze tylko wtedy, gdy diagram ma stałe rozmiary. W przypadku diagramu skalowanego przeliczenia pozycji muszą bazować na współrzędnych bezwzględnych.

Właściwości podstawowe		
	Główna zmienna	CarPosX
Właściwości stanowe, stan podstawowy		
	Nazwa obrazu	*Car*
Zdarzenia		
	Animacja	*^SetPosition(500000+100000*(Variable())/100.0), null)*

W powyższym przykładzie bazowe położenie obrazka, to środek diagramu (wartość 500 000 bezwzględna). Skrajne prawe położenie, przy wartości zmiennej *CarPosX* równej 100, to 60% szerokości diagramu.

13.2 Zmiana pozycji wewnątrz obszaru określonego innym obiektem

Powyższy problem przeliczania współrzędnych można ominąć stosując metodę względnego pozycjonowania obiektów. Zakres ruchu jest określony przez specjalny obiekt (może być niewidoczny), wewnątrz którego pozycjonowany jest obiekt właściwy.

Właściwości stanowe, stan podstawowy		
	Nazwa obrazu	*Car*
Zdarzenia		
	Animacja	*^SetPosition(* *LocalProperty(area,X) + (LocalProperty(area,Width) * (Variable(CarPosX)/100.0)),* *LocalProperty(area,Y) + (LocalProperty(area,Height) * (Variable(CarPosY)/100.0)))*

W przykładzie założono, że na diagramie znajduje się obiekt o nazwie *area*. Może to być np. prostokątny obiekt klasy *Kształt*. Do przeliczenia pozycji obrazka wykorzystywane są koordynaty obiektu *area,* odczytywane funkcją *LocalProperty*. Zmienne sterujące pozycją w obu osiach *CarPosX* i *CarPosY* mają zakres zmienności od 0 do 100.

Wygodą tego rozwiązania jest to, że zmiana zakresu ruchu wymaga jedynie zmiany położenia obiektu *area* – nie jest potrzebna zmiana wyrażenia pozycjonującego w obiekcie obrazka.

13.3 Pozycjonowanie grup i wzorców

Metody postępowania w przypadku grup obiektów i osadzonych wzorców jest identyczna jak w przypadku pojedynczych obiektów. Zdarzenie *Animacja* dla grup i wzorców pozwala kontrolować ich położenie i rozmiar.

14 Sygnalizacja i obsługa stanów alarmowych

Podstawowym mechanizmem przeglądania i obsługi alarmów są obiekty *Tabela alarmów aktywnych* i *Tabela alarmów historycznych.* Pokazują one stan alarmów w postaci tabelarycznej oraz posiadają zintegrowany pasek narzędziowy.

Zobacz też:

- podręcznik Pierwsze Kroki > 4. Aplikacja z obsługą alarmów;
- *podręcznik Obiekty > obiekt Tabela alarmów aktywnych, obiekt Tabela alarmów historycznych;*
- *podręcznik System alarmów*.

W rozdziale tym zajmiemy się jednak sposobami sygnalizacji i obsługi alarmów bezpośrednio na diagramach synoptycznych, z użyciem uniwersalnych obiektów.

14.1 Kontrola stanu i obsługa pojedynczych alarmów

W przykładzie pokazany zostanie typowy mechanizm sygnalizacji alarmu przy pomocy obiektu klasy *Obraz*. Stan obiektu będzie kontrolowany przy pomocy funkcji *IsAlarm*, *IsAlarmUnaccepted* i *IsAlarmExcluded*. Dodatkowo obiekt będzie służył do potwierdzania alarmu przy pomocy akcji operatorskiej *AcceptAlarm*. Inne akcje operatorskie związane z obsługą alarmów, to: *ExcludeAlarm* i *IncludeAlarm*.

Właściwości stanowe, stan podstawowy		
	Widoczny	Nie
	Nazwa obrazu	
Właściwości stanowe, stan 1		
	Warunek stanu	=IsAlarm(A1001)
	Widoczny	=!IsAlarmExcluded(A1001)
	Nazwa obrazu	AlrRed
Właściwości stanowe, stan 2		
	Warunek stanu	=IsAlarmUnaccepted(A1001)
	Widoczny	=!IsAlarmExcluded(A1001)

	Nazwa obrazu	=IsBlinkOff()?AlrYellow:null
Zdarzenia		
	Kliknięcie prawym przyciskiem	^AcceptAlarm(null,A1001)

W stanie podstawowym obrazek pozostaje niewidoczny. Stan numer 1 włącza się, gdy alarm o identyfikatorze *A1001* zmieni stan na aktywny (rozpoczęty). Stan numer 2 włącza się, gdy alarm jest niepotwierdzony. Ponieważ stan o wyższym numerze ma pierwszeństwo, to w okresie, w którym alarm pozostaje niepotwierdzony (niezależnie od tego czy jest zakończony), wyświetlany jest obraz *AlrYellow*. Dodatkowo, wykorzystanie wyrażenia warunkowego z funkcją *IsBlinkOff* powoduje migotanie obrazka. Jeżeli alarm jest aktywny, ale potwierdzony, to wyświetlany jest obrazek *AlrRed.* Należy pamiętać o tym, że kontrola stanu alarmów zakończonych zależy od użytego trybu pracy logu alarmów aktywnych. Jeżeli alarm zakończony zostanie usunięty z logu (po upłynięciu czasu przechowania alarmów zakończonych), to funkcja *IsAlarmUnaccepted* zwróci wartość *false* nawet wtedy, gdy alarm nie jest potwierdzony.

Ponieważ funkcje *IsAlarm* i *IsAlarmUnaccepted* zwracają rzeczywisty stan alarmu niezależnie od tego, czy został on przez operatora wykluczony z obsługi, to konieczne jest użycie funkcji *IsAlarmExcluded* we właściwości *Widoczny*. Jeżeli w aplikacji wykluczenia nie byłyby stosowane, to wystarczałoby użycie w we właściwości *Widoczny* obu stanów wartości *Tak*.

Akcja *AcceptAlarm* w obsłudze zdarzenia *Kliknięcie prawym przyciskiem* powoduje potwierdzenie alarmu *A1001*.

W przykładzie założono, że w aplikacji jest zdefiniowana tylko jedna domena alarmowa. Gdyby tak nie było, to należałoby użyć wariantów funkcji *IsAlarm* itp. z jawnym podaniem nazwy domeny.

Powyższy przykład, chociaż poprawny, ma jedną istotną wadę. Próba użycia obiektu, do kontroli innego alarmu wymaga modyfikacji pięciu właściwości. Zalecaną procedurą jest przekształcenie obiektu we wzorzec z parametrem określającym identyfikator alarmu i wywołanie funkcji *Parameter* wszędzie tam, gdzie identyfikator został użyty. Można też postąpić zgodnie z poniższym przykładem.

Właściwości podstawowe		
	Główna zmienna	*A1001*
Właściwości stanowe, stan podstawowy		
	Widoczny	*Nie*
	Nazwa obrazu	
Właściwości stanowe, stan 1		

	Warunek stanu	*=IsAlarm(LocalProperty(MainVar))*
	Widoczny	*=!IsAlarmExcluded(LocalProperty(MainVar))*
	Nazwa obrazu	*AlrRed*
Właściwości stanowe, stan 2		
	Warunek stanu	*=IsAlarmUnaccepted(LocalProperty(MainVar))*
	Widoczny	*=!IsAlarmExcluded(LocalProperty(MainVar))*
	Nazwa obrazu	*=IsBlinkOff()?AlrYellow:null*
Zdarzenia		
	Kliknięcie prawym przyciskiem	*^AcceptAlarm(null, LocalProperty(MainVar))*

Ten wariant działa dokładnie tak samo jak poprzedni. Identyfikator alarmu został jednak podany wyłącznie we właściwości *Główna zmienna*. Fakt, że taka zmienna nie istnieje, nie ma znaczenia. Decydujący jest kontekst użycia właściwości. W tym przypadku wartość właściwości jest pobierana przy pomocy funkcji *LocalProperty* i używana jako identyfikator alarmu.

14.2 Sygnalizacja stanu grupy alarmów

Korzystając z funkcji *AlarmsGroupState* można kontrolować zbiorczy stan alarmów w wybranej grupie lub nawet całej domenie. W przykładzie użyjemy półprzezroczysty obiekt klasy *Kształt* w celu sygnalizacji alarmów we fragmencie kontrolowanej instalacji.

Właściwości podstawowe		
	Warstwa	*10*
Właściwości stanowe, stan podstawowy		
	Widoczny	*Nie*
	Przezroczystość	*0,1*
Właściwości stanowe, stan 1		
	Warunek stanu	*=AlarmsGroupState(Blok1, "Strefa=S1")&1*

	Widoczny	*Tak*
	Kolor	*Cyan*
	Kolor otoczki	*Blue*
	Przezroczystość	*0,5*
Właściwości stanowe, stan 2		
	Warunek stanu	*=AlarmsGroupState(Blok1, "Strefa=S1")&4*
	Widoczny	*Tak*
	Kolor	*Coral*
	Kolor otoczki	*Red*
	Przezroczystość	*0,5*

Zakładamy obiekt klasy *Kształt* w takim miejscu, żeby przesłonił właściwy fragment diagramu. W tym celu może być konieczna zmiana właściwości *Warstwa*. W stanie podstawowym obiekt ma pozostać niewidoczny. Ponieważ wartość właściwości *Widoczny* jest ignorowana w trakcie edycji diagramu, to zmieniamy też właściwość *Przezroczystość* – ułatwi to edycję diagramu, obiekty pod obiektem *Kształt* będą widoczne. Następnie dodajemy dwa stany, służące do sygnalizacji alarmów. Każdy z nich ma indywidualnie ustawione atrybuty kolorów, właściwość *Widoczny* ustawioną na *Tak*, oraz odpowiedni poziom przezroczystości. W warunkach stanów użyto funkcję *AlarmsGroupState*. Funkcja kontroluje stan alarmów z domeny *Blok1*, dla których atrybut grupujący *Strefa* jest równy *S1.* Wartość zwracana przez funkcję może być interpretowana na różne sposoby. W naszym przypadku bitowy iloczyn logiczny z wartością 1 (test najmłodszego bitu) sprawdza, czy w grupie znajduje się alarm niepotwierdzony. Test z wartością 4 (test bitu numer 2) sprawdza, czy w grupie znajduje się przynajmniej jeden niepotwierdzony alarm aktywny. Ze względu na kolejność definicji stanów, jeżeli w grupie znajdzie się alarm aktywny niepotwierdzony, to wyświetlona zostanie „szybka” w kolorze *Coral*. Jeżeli takiego alarmu nie będzie, ale będzie przynajmniej jeden alarm zakończony i niepotwierdzony, to „szybka” będzie w kolorze *Cyan*.

Do sygnalizacji stanu grupy alarmów można też wykorzystać funkcję *AlarmsCount*, która zwraca liczbę alarmów w wybranej grupie z uwzględnieniem stanów aktywności i potwierdzenia.

15 Kontrola pracy obiektów klasy Wykres

Obiekty klasy *Wykres* posiadają własne, wbudowane interfejsy współpracy z użytkownikiem. Alternatywą jest też obiekt *Kontroler wykresu*, który pozwala sterować pracą wykresu. W niektórych scenariuszach pracy może być jednak konieczne dodanie własnych mechanizmów sterujących.

15.1 Sterowanie zakresem czasu wykresu

W standardowym zastosowaniu obiekty wykresów, po otwarciu diagramu pokazują zdefiniowany w obiekcie okres czasu wyliczany względem chwili bieżącej. Użytkownik, o ile nie zostanie to zablokowane, może później wybrać dowolny inny zakres czasu. Można jednak dodać dodatkowe mechanizmy wyboru zakresu czasu przez użytkownika, ewentualnie forsować zakres czasu na podstawie danych pobieranych z różnych źródeł.

15.1.1 Modyfikacja właściwości obiektu Wykres

Pokazywany na obiekcie zakres czasu zależy od trzech właściwości. Długość okresu czasu jest podawana we właściwości *Zakres czasu [min]*. Koniec zakresu (prawą stronę) określa właściwość *Ręczny czas bazowy*. Ma ona jednak znaczenie tylko wtedy, gdy dodatkowo właściwość *Tryb odświeżania* jest ustawiona na *Ręczny* – w przeciwnym wypadku koniec wykresu będzie określony przez chwilę bieżącą.

Wykorzystując akcję operatorską *SetProperty* można zmieniać zakres czasu wykresów.

Właściwości podstawowe		
	Nazwa elementu	*Chart*
	Ręczny czas bazowy	
	Zakres czasu [min]	*10*
	Tryb odświeżania	*Ręczny*

Powyższy fragment definicji obiektu klasy *Wykres* pokazuje obiekt o nazwie *Chart*, który po otwarciu diagramu wyświetli wykres dla ostatnich 10 minut. Istotne jest, żeby obiekt posiadał nazwę oraz wybrany był ręczny tryb odświeżania.

Poniższe obiekty klasy *Przycisk* służą do przełączania zakresu czasu.

Właściwości podstawowe		
	Aktywny	*Tak*

	Typ przycisku	*Standardowy przycisk Windows*
	Tryb przełącznika	*Nie*
	Sterowanie natychmiastowe	*Tak*
Właściwości stanowe, stan podstawowy		
	Włączony	*Nie*
	Tekst wyłączenia	*Ostatnia godzina*
Zdarzenia		
	Wyłączenie przycisku	*^Actions(SetProperty(Chart,ManualBaseTime,OPCTime("HOUR+1H")), SetProperty(Chart,TimeRange,60))*

Właściwości podstawowe		
	Aktywny	*Tak*
	Typ przycisku	*Standardowy przycisk Windows*
	Tryb przełącznika	*Nie*
	Sterowanie natychmiastowe	*Tak*
Właściwości stanowe, stan podstawowy		
	Włączony	*Nie*
	Tekst wyłączenia	*Ostatni kwadrans*
Zdarzenia		
	Wyłączenie przycisku	*^Actions(SetProperty(Chart,ManualBaseTime,OPCTime("Minute15+15M")), SetProperty(Chart,TimeRange,15))*

Oba obiekty wykorzystują akcję wielokrotną *Actions* z akcjami składowymi *SetProperty* do modyfikacji wartości właściwości *Ręczny czas bazowy* (nazwa wewnętrzna *ManualBaseTime*) i *Zakres czasu [min]* (nazwa wewnętrzna *TimeRange).* Zmiana właściwości *TimeRange* jest oczywista. W przypadku właściwości *ManualBaseTime* nowa wartość musi być konwertowalna na format *DateTime*. Może być podawana w postaci tekstowej, pobierana ze zmiennej typu *DateTime*, wyliczana z wykorzystaniem funkcji From*Asix6Date*. W przykładach użyto funkcję *OPCTime*, która w łatwy sposób pozwala określić okres czasu wyliczany na podstawie chwili bieżącej. Wyrażenie *OPCTime("HOUR+1H")* powoduje wyliczenie końca bieżącej godziny (np. dla chwili bieżącej 13:20, zwraca wartość 14:00), czyli wykres pokaże okres od początku bieżącej godziny do jej końca). W podobny sposób działa wyrażenia *OPCTime("Minute15+15M"),* które powoduje wyświetlenie bieżącego pełnego okresu 15-minutowego.

15.1.2 Sterowanie poprzez zmienną wirtualną

Alternatywnie do bezpośredniego ustawiania wartości właściwości wykresu można też sterować zakresem czasu za pośrednictwem zmiennych typu *DateTime*. Pozwala to na pełną automatyzację pracy.

Właściwości podstawowe		
	Ręczny czas bazowy	*=Variable(TimeMax)*
	Zakres czasu [min]	*10*
	Tryb odświeżania	*Ręczny*

W powyższym przypadku czas bazowy (koniec zakresu) wynika z wartości zmiennej *TimeMax.* Zmienna powinna mieć wartość typy *DateTime*. Ewentualnie należy wykonać odpowiednią konwersję, np. używając funkcję *ToDateTime.* Każdorazowa zmiana wartości zmiennej *TimeMax* automatycznie powoduje odpowiednią zmianę zakresu wykresu.

Nową wartość zmiennej *TimeMax* można ustawiać skryptem, pobierać bezpośrednio ze sterownika (przy odpowiednich konwersjach) lub wprowadzać przy pomocy obiektu *Tekst.*

Właściwości podstawowe		
	Aktywny	*Tak*
	Główna zmienna	*TimeMax*
	Zmienna sterowana	*#*
	Początkowa wartość edycji	*Bieżąca*

	Sterowanie natychmiastowe	*Tak*
Właściwości stanowe, stan podstawowy		
	Tekst	*#*

Jeżeli typ zmiennej *TimeMax* został określony jako *DateTime*, to nową wartość należy podać w sposób zgodny z jakimkolwiek systemowym formatem tekstowym daty i czasu. Po zweryfikowaniu poprawności formy nowej wartości, zostanie ona przeliczona na typ *DateTime* i zapisana w zmiennej *TimeMax*, wpływając w ten sposób na zakres czasu wyświetlanego na wykresie.

15.2 Kontrola wyświetlania przebiegów wzorcowych

Obiekty *Wykres* posiadają też możliwość wyświetlania tzw. przebiegów wzorcowych. Kluczowe jest wtedy określenie właściwego punktu zaczepienia przebiegu wzorcowego.

Właściwości podstawowe		
	Modyfikowalne przebiegi wzorcowe	*Nie*
Przebiegi wzorcowe, seria #1		
	Nazwa przebiegu	*Trend1*
	Zaczep przebiegu	*&PatternTime*

Właściwość *Modyfikowalne przebiegi wzorcowe* określa, czy użytkownik może zmieniać krzywą wzorcową oraz jej punkt zaczepienia (przy pomocy komend z menu kontekstowego obiektu). W przypadku, gdy chcemy sterować pracą obiektu własnymi mechanizmami, warto wyłączyć obsługę użytkownika.

Za wybór krzywej wzorcowej odpowiada właściwość *Nazwa przebiegu*, a za jej punkt zaczepienia właściwość *Zaczep przebiegu.* W naszym przykładzie punkt zaczepu ustawiany jest na podstawie wartości zmiennej *PatternTime*, która powinna być typu *DateTime.*

Właściwości podstawowe		
	Aktywny	*Tak*
	Typ przycisku	*Standardowy przycisk Windows*

	Tryb przełącznika	*Nie*
	Sterowanie natychmiastowe	*Tak*
Właściwości stanowe, stan podstawowy		
	Włączony	*Nie*
	Tekst wyłączenia	*Zaczep wzorzec*
Zdarzenia		
	Wyłączenie przycisku	*^SetVariable(PatternTime,OPCTime("NOW-5M"))*

Właściwości podstawowe		
	Aktywny	*Tak*
	Typ przycisku	*Standardowy przycisk Windows*
	Tryb przełącznika	*Nie*
	Sterowanie natychmiastowe	*Tak*
Właściwości stanowe, stan podstawowy		
	Włączony	*Nie*
	Tekst wyłączenia	*Ukryj wzorzec*
Zdarzenia		
	Wyłączenie przycisku	*^SetVariable(PatternTime,"2000-1-1")*

Pierwszy z obiektów *Przycisk*, ustawia zmienną *PatternTime* na wartość 5 minut przed chwilą bieżącą. Spowoduje to wyświetlenie krzywej wzorcowej od tego momentu.

Drugi z przycisków ukrywa przebieg wzorcowy poprzez przesunięcie go poza zakres czasu widoczny na obiekcie *Wykres*. Alternatywna metoda usuwania krzywej wzorcowej to zmiana nazwy przebiegu na nazwę pustą (nazwa przebiegu też może być podawana za pośrednictwem zmiennej wirtualnej).

16 Użycie wzorców

Wzorce obiektów w znacznym stopniu ułatwiają konstrukcję aplikacji. Pozwalają one na tworzenie „klocków", z których następnie aplikacja jest składana. Wzorzec może łączyć w całość grupę obiektów lub w prostszych przypadkach zawierać pojedynczy obiekt, ale z wstępnie ustawionymi właściwościami. Wzorce mogą posiadać parametry, które służą do określenia sposobu działania wzorca w miejscu jego osadzenia na diagramie.

W przykładzie zajmiemy się konstrukcją jednoobiektowego wzorca opartego na obiekcie klasy *Miernik*. Wzorzec będzie sygnalizował przekroczenie dwóch limitów alarmowych, a wartości limitów i zakresy wartości będą pobierane z bazy definicji zmiennych.

Tabela: Parametry wzorca.

Parametry wzorca		
EndAngle		
	Wartość domyślna	405
StartAngle		
	Wartość domyślna	135
Variable		
	Wartość domyślna	

Wzorzec będzie posiadał 3 parametry. *StartAngle* i *EndAngle* określają krańcowe kąty skali miernika. Maja one określone wartości domyślne – w miejscu osadzenia wzorca trzeba będzie podać wartość tych parametrów tylko wtedy, gdy będą one inne od domyślnych. Parametr *Variable* służy do przekazania nazwy zmiennej procesowej, której wartość będzie pokazywana na mierniku.

W trakcie przygotowania (edycji) wzorca, można podać testowe wartości parametrów. Pozwala to na sprawdzenie, czy wzorzec zachowuje się zgodnie z oczekiwaniami.

Częściową parametryzację obiektu *Miernik* stanowiącego zawartość wzorca pokazuje poniższa tabela.

Tabela: Wybrane parametry obiektu Miernik.

Właściwości podstawowe		
	Główna zmienna	%Variable

	Obrazek tła	!GaugeBkImage
	Pokaż limity	Tak
	Wartość	#
	Wartość minimalna	@DisplayRangeFrom
	Wartość maksymalna	@DisplayRangeTo
	Wartość LL	@DisplayRangeFrom
	Wartość L	@LimitLo
	Wartość H	@LimitHi
	Wartość HH	@DisplayRangeTo
	Kąt początkowy	%StartAngle
	Kąt końcowy	%EndAngle
	Kolor wartości poprawnej	!GaugeProperColor
	Kolor L	Yellow
	Kolor H	Yellow
Właściwości stanowe, stan podstawowy		
	Kolor wskazówki	!GaugePointerColor
	Kolor tła	!GaugeBkColor
	Kolor otoczki	=Property(GaugeBkImage) == "" ? Property(GaugeOutlineColor) : Transparent
	Kolor podziałki	!GaugeScaleColor
	Kolor czcionki	!GaugeFontColor

Zmienna główna obiektu jest zdefiniowana jako *%Variable*. Oznacza to odwołanie do parametru wzorca o nazwie *Variable*. Nazwa zmiennej zostanie przekazana do wzorca w miejscu jego osadzenia na diagramie. W podobny sposób zdefiniowane są właściwości *Kąt początkowy* i *Kąt końcowy*.

Część właściwości, m.in. *Obrazek tła*, zdefiniowane są przez odwołanie do właściwości globalnej. Odwołania tego typu określone są przez prefiks !. W efekcie, wygląd wzorca może być zmieniany

przez zmianę wartości właściwości globalnej. Ten sposób parametryzacji ma szczególnie sens wtedy, gdy właściwości globalne są współużywane przez wiele wzorców i obiektów – pozwala to na łatwą zmianę wyglądu całej aplikacji.

Wyrażenie *=Property(GaugeBkImage) == "" ? Property(GaugeOutlineColor) : Transparent* użyte we właściwości *Kolor otoczki* pokazuje warunkowe ustawienie koloru w zależności od wartości właściwości globalnej (pobieranej funkcją *Property*). Jeżeli właściwość globalna *GaugeBkImage* nie jest zdefiniowana (jest pusta), to kolor otoczki określa wartość właściwości globalnej *GaugeOutlineColor*. W przeciwnym wypadku otoczka jest ukrywana poprzez wybranie koloru przezroczystego.

Właściwość *Wartość* określona jest w sposób typowy – notacja # oznacza wartość zmiennej głównej. Ponieważ zmienna główna jest określona parametrem wzorca, to położenie wskazówki miernika będzie wynikało z wartości zmiennej o nazwie przekazanej w parametrze wzorca. Definicje właściwości zakresów i limitów alarmowych odwołują się do atrybutów zmiennej (prefiks *@)*. Tutaj też *n*astępuje pośrednie odwołanie do atrybutu zmiennej określonej parametrem wzorca.

Właściwości *Kolor L* i *Kolor H* zdefiniowane są bezpośrednio. Gdyby zaszła konieczność użycia innych kolorów, to możliwe byłyby dwie strategie. Można stworzyć bliźniaczy wzorzec o inaczej ustawionych kolorach przekroczeń lub dodać dwa dodatkowe parametry kolorów i zrobić jeden wzorzec uniwersalny.

Asix.Evo - Parametryzacja aplikacji

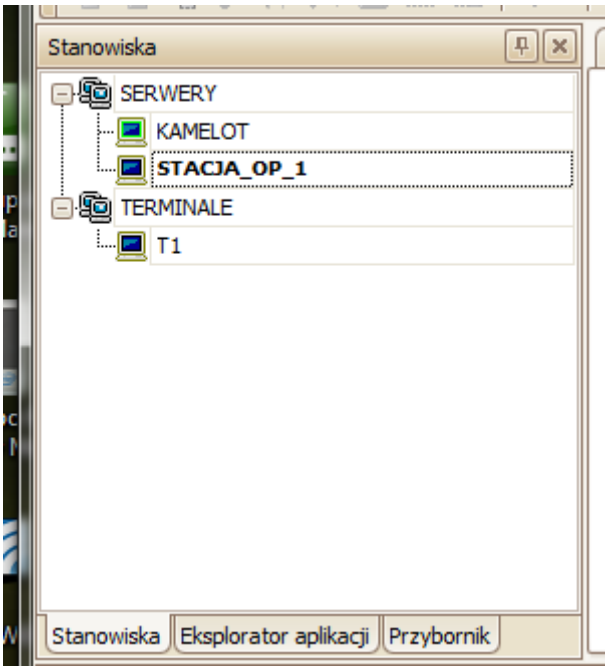
1 Konfiguracja stanowisk aplikacji

Konfigurując aplikację typu Evo, w ramach pojedynczego projektu należy zdefiniować parametry wszystkich stanowisk, które będą uczestniczyć w pracy aplikacji. Generalna zasada jest taka, że każde stanowisko typu serwerowego lub wolnostojące jest jawnie deklarowane, natomiast stanowiska terminalowe są parametryzowane wspólnie.

W trakcie tworzenia aplikacji przy pomocy kreatora można od razu utworzyć podstawowy zestaw stanowisk. Zawsze tworzone jest też stanowisko o nazwie zgodnej z nazwą komputera, na którym aplikacja została utworzona. Podstawowym celem tego stanowiska jest edycja aplikacji (aczkolwiek może być też użyte jako stanowisko produkcyjne).

1.1 Panel roboczy Stanowiska

Podstawowym panelem służącym do zarządzania stanowiskami aplikacji jest panel *Stanowiska*.



Rys. Panel roboczy 'Stanowiska'.

Na ilustracji pokazana jest konfiguracja stanowisk dla typowej małej aplikacji. W obszarze *Serwery* zdefiniowane jest jedno stanowisko inżynierskie do projektowania i testowania aplikacji oraz pojedyncza produkcyjna stacja operatorska. Dodatkowo zdefiniowane jest pojedyncze stanowisko

terminalowe. W trakcie działania aplikacji w kontekście tego stanowiska będzie można uruchomić dowolną ilość terminali.

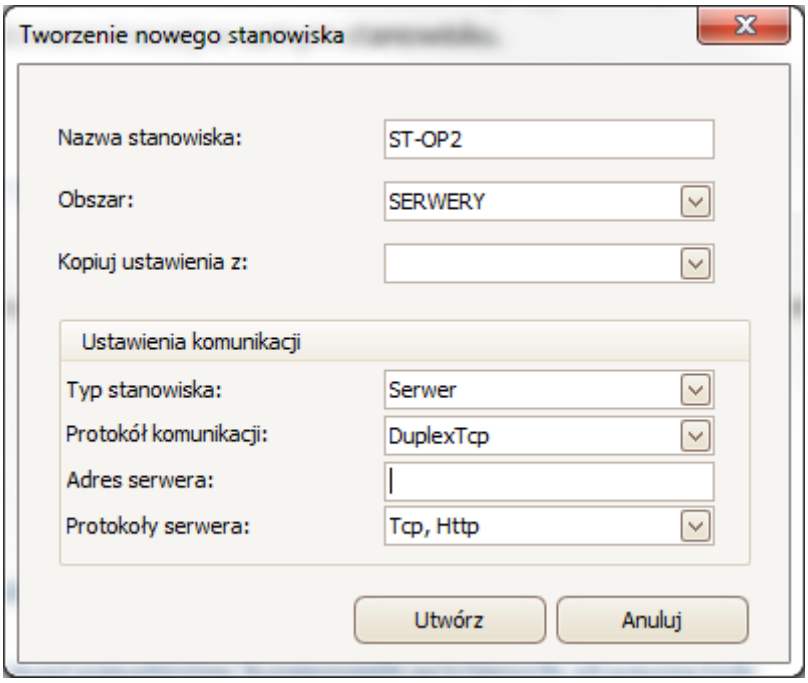
Panel *Stanowiska* służy dwóm celom. Po pierwsze, pozwala przełączać się pomiędzy stanowiskami w trakcie edycji parametrów zależnych od stanowiska. Po drugie panel służy do zakładania nowych stanowisk oraz zarządzania nimi. Komendy zarządzania stanowiskami dostępne są poprzez menu kontekstowe panelu. Dodatkowo można przesuwać stanowisko pomiędzy obszarami przy pomocy metody 'przeciągnij i upuść'.

Umieszczanie kilku stanowisk w pojedynczym obszarze pozwala na hierarchiczną parametryzację. Parametry zdefiniowane w kontekście obszaru obowiązują na wszystkich stanowiskach obszaru, o ile nie zostały zdefiniowane inaczej w samym stanowisku.

1.1.1 Założenie nowego stanowiska

Aby założyć nowe stanowisko, należy wykonać komendę *Nowe stanowisko* z menu kontekstowego panelu.

Znaczenie parametrów stanowiska opisuje poniższa tabela.



Rys. Okno tworzenia nowego stanowiska.

Tab.: Parametry stanowiska.

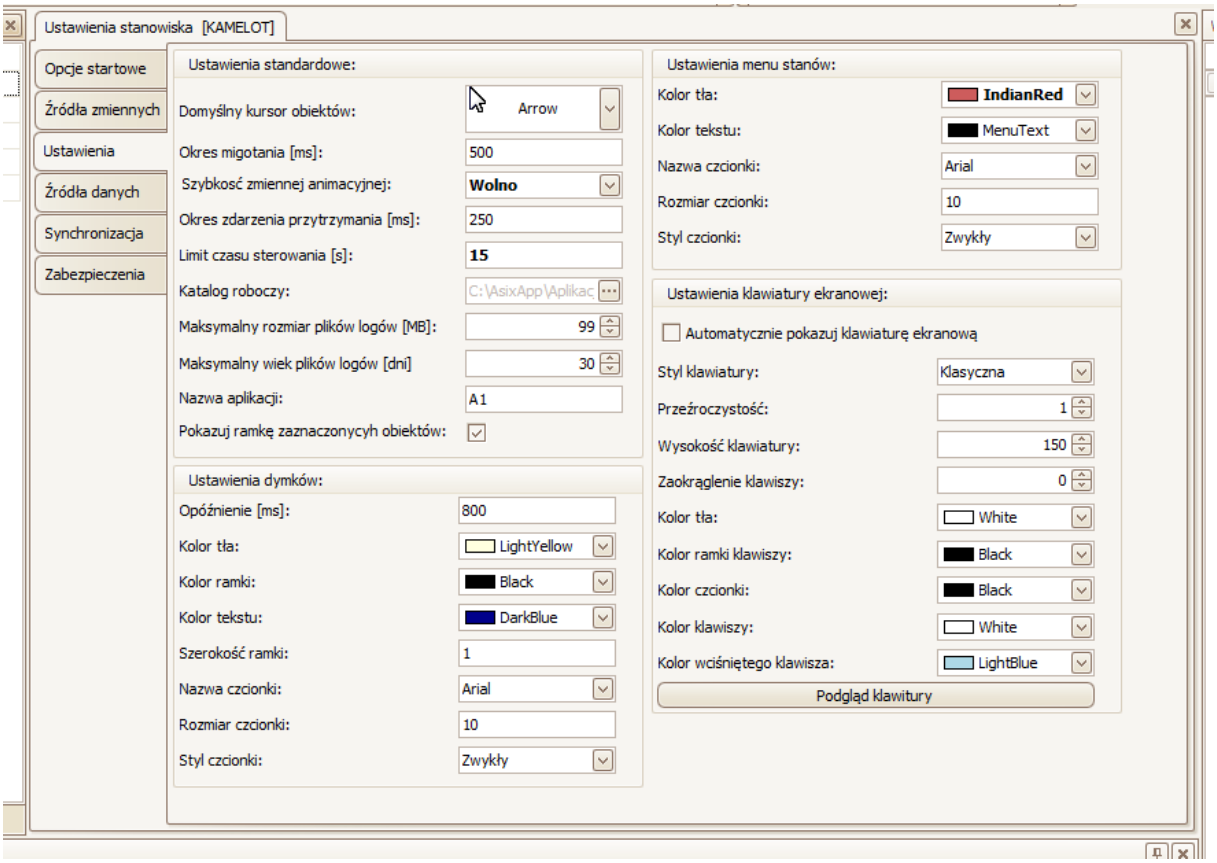
Parametr	Znaczenie

Nazwa stanowiska	Nazwa identyfikująca stanowisko. W przypadku stanowisk typu serwerowego korzystne jest użycie nazwy zgodnej z systemową nazwą komputera, na którym stanowisko będzie uruchomione.
Obszar	Nazwa obszaru, w którym należy umieścić stanowisko. Będzie ono dziedziczyło ustawienia obszaru. Można też założyć stanowisko poza jakimkolwiek obszarem.
Kopiuj ustawienia z	Nazwa innego stanowiska lub obszaru, z którego należy skopiować ustawienia. Używane w przypadku tworzenia bliźniaczych stanowisk.
Typ stanowiska	Kluczowy parametr określający przeznaczenie stanowiska. W zależności od wybranego typu zmienia się zakres możliwych funkcji stanowiska. Dostępne są poniższe typy: • Serwer – stanowisko może udostępniać swoje dane innym stanowiskom, w systemie alarmów może pełnić rolę kontrolera sieciowego. • Terminal – terminal typu okienkowego, który w systemie sieciowym pełni wyłącznie rolę klienta pobierającego dane. • Terminal WWW – terminal typu WWW, wszystkie dane pozyskiwane są mechanizmami sieciowymi ze stanowisk typu serwer. • Wolnostojąca – stanowisko używane w konfiguracjach bez komunikacji sieciowej.
Protokół komunikacyjny	Protokół używane do komunikacji typu klient danych. Możliwe rodzaje protokołów to: • DuplexTcp – podstawowy tryb komunikacji w sieciach lokalnych. Można używać również w sieciach rozległych, jeżeli transmisja nie jest blokowana przez zaporę. • *Tcp – protokół zbliżony do DuplexTCP, ale trochę mniej wydajny.* • Http – protokół przeznaczony głównie dla terminali WWW. Terminal WWW może jednak stosować protokół DuplexTcp, jeżeli komunikacja nie jest blokowana przez zaporę.
Adres serwera	Adres serwera, który jednoznacznie identyfikuje stanowisko w sieci. Może to być systemowa nazwa komputera lub adres IP. Pole może pozostać puste, jeżeli nazwa stanowiska jest zgodna z nazwą komputera.
Protokoły serwera	Protokoły, których stanowisko serwerowe używa do komunikacji z klientami. Zestaw wybranych protokołów musi odpowiadać protokołom używanym przez stanowiska, które będą łączyły się z serwerem.

1.2 Ustawianie parametrów zależnych od stanowiska

W skład zestawu danych konfiguracyjnych aplikacji wchodzą elementy wspólne dla wszystkich stanowisk (okna, diagramy, menu, obrazki itp.) oraz parametry definiowane osobno dla każdego stanowiska. Parametry stanowisk to konfiguracja kanałów komunikacyjnych i archiwów danych, terminarz akcji oraz tzw. ustawienia stanowisk. Panele robocze dla wszystkich tych grup ustawień otwierane są za pośrednictwem węzłów panelu eksploratora.

Cechą charakterystyczną paneli roboczych dotyczących konkretnego stanowiska jest to, że w nazwie zakładki wyświetlana jest nazwa stanowiska (obszaru), którego dotyczą pokazywane ustawienia. Przełączyć panel roboczy na inne stanowisko można poprzez kliknięcie na właściwym węźle panelu stanowisk.



Rys. Panel edycyjny ustawień dla stanowiska.

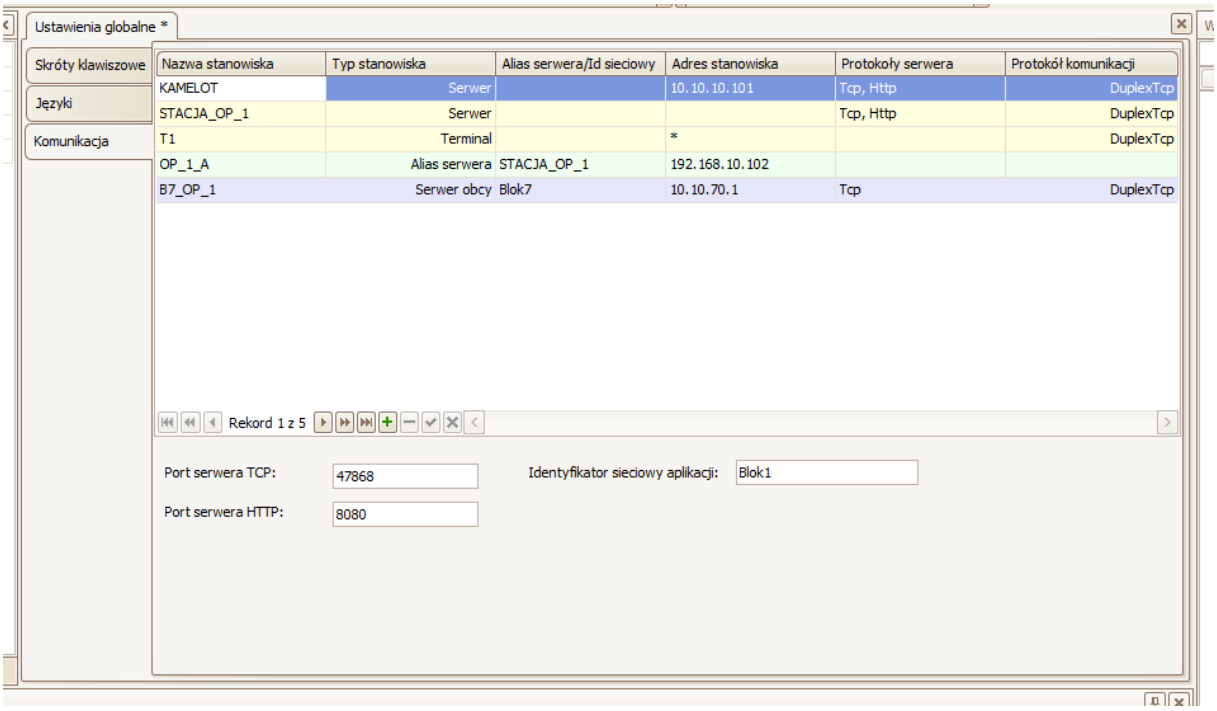
Panele edycyjne parametrów stanowisk zawsze pokazują ustawienia obowiązujące dla stanowiska. Wartości tych parametrów wynikają z ustawień domyślnych, ustawień wykonanych w obszarze nadrzędnym lub ustawień własnych. Jeżeli na danym stanowisku ustawienie zostało jawnie zmodyfikowane, to jest ono w panelu roboczym specjalnie oznaczane. Na przykład, na

powyższej ilustracji niektóre wartości (zmienione) wyświetlone są pogrubioną czcionką. Może się też zdarzyć, że niektórych parametrów nie można zmienić. Dzieje się tak wtedy, gdy parametr został zdefiniowany w obszarze nadrzędnym. Zmiana takich parametrów możliwa jest tylko w kontekście obszaru, w którym zostały zdefiniowane.

Ustawienia dotyczące stanowisk mogą się znajdować w innych panelach roboczych wspólnych dla wszystkich stanowisk. W tym przypadku wyświetlane są jednak tabele, które pokazują parametry dla wszystkich stanowisk. Jest to na przykład tabela określająca role wszystkich stanowisk w systemie obsługi alarmów. Inny przykład to tabela ustawień parametrów komunikacyjnych stanowisk.

1.3 Konfiguracja parametrów komunikacyjnych stanowisk

Panel roboczy edycji parametrów komunikacyjnych stanowisk otwierany jest poprzez wybór węzła *Ustawienia globalne* w panelu eksploratora aplikacji, a następnie wybranie zakładki *Komunikacja*.



Rys. Ustawienia globalne parametrów komunikacyjnych stanowisk.

Panel pokazuje ustawienia wszystkich stanowisk aplikacji oraz kilka parametrów globalnych. Parametry globalne określają numery portów używane w trakcie komunikacji pomiędzy stanowiskami oraz tekstowy identyfikator służący do rozróżnienia stanowisk z różnych aplikacji pracujących we wspólnej sieci.

Stanowiska wyświetlane w tabeli w większości odpowiadają stanowiskom z panelu roboczego *Stanowiska*. Parametry pochodzą z ustawień podanych w trakcie tworzenia stanowisk.

Do listy stanowisk przy pomocy przycisku + można dodać stanowiska typu specjalnego. Stanowiska te nie są pokazywane w panelu *Stanowiska* i nie można w ich kontekście uruchamiać aplikacji.

1.3.1 Aliasy serwerów

Aliasy serwerów stosowane są wtedy, gdy standardowe stanowisko typu serwerowego może być identyfikowane w sieci kilkoma adresami, na przykład posiada dwie karty sieciowe. W zależności od lokalizacji stanowisk terminalowych w sieci, taki serwer może być dostępny poprzez różne adresy.

Dodając alias należy określić nazwę stanowiska serwera oraz jego alternatywny adres.

1.3.2 Stanowiska obce

Stanowiska obce stosowane są wtedy, gdy zachodzi konieczność podłączenia się do serwera z innej aplikacji Asix.Evo. Standardowo możliwa jest komunikacja tylko między stanowiskami zdefiniowanymi w ramach tej samej aplikacji. Mechanizm stanowisk obcych pozwala na wymianę danych pomiędzy różnymi aplikacjami Asix.Evo.

Definiując stanowisko obce należy podać identyfikator sieciowy aplikacji, do której należy obcy serwer oraz adres tego serwera.

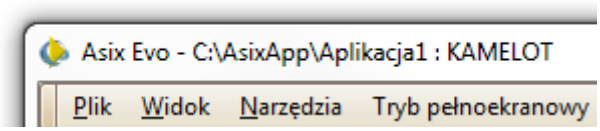
1.4 Uruchomienie aplikacji w kontekście wybranego stanowiska

W momencie uruchomienia programu Asix.Evo należy określić nazwę stanowiska, w kontekście którego program ma być wykonywany. Domyślnie, jeżeli nazwa nie zostanie podana, nastąpi próba wykonania programu z użyciem nazwy komputera. Jeżeli stanowisko o nazwie zgodnej z nazwą komputera nie jest zdefiniowane, to użytkownik będzie musiał wybrać stanowisko z listy wszystkich stanowisk. Można jednak sforsować nazwę stanowiska w linii komendy stosując opcję *as*.

```
AsixEvo -as=ST-OP2 "C:\AsixApp\Aplikacja1"
```

W przykładzie użyto stanowiska *ST-OP2*. Skróty uruchamiające aplikację można tworzyć przy pomocy kreatora skrótów, uruchamianego komendą z menu *Narzędzia*.

Nazwa stanowiska, w kontekście którego wykonywany jest program, wyświetlona jest na belce tytułowej okna programu.



Rys. Wyświetlenie nazwy stanowiska na diagramie.

W trybie wykonawczym, można wyświetlić nazwę stanowiska na diagramie, korzystając z wbudowanej zmiennej wirtualnej o nazwie *StationName*.

1.5 Diagnostyka połączeń

W trakcie pracy programu można kontrolować stan systemu komunikacyjnego w specjalnym panelu diagnostycznym. Możliwy jest wgląd w stan każdego z pracujących stanowisk aplikacji. Pokazywane są podstawowe informacje systemowe o stanowiskach oraz lista aktywnych połączeń.

W trybie pracy architekta, panel otwierany jest w panelu eksploratora aplikacji, korzystając z węzła *Diagnostyka/Połączenia*. W trybie wykonawczym panel diagnostyczny komunikacji dostępny jest poprzez okno panelu kontrolnego.

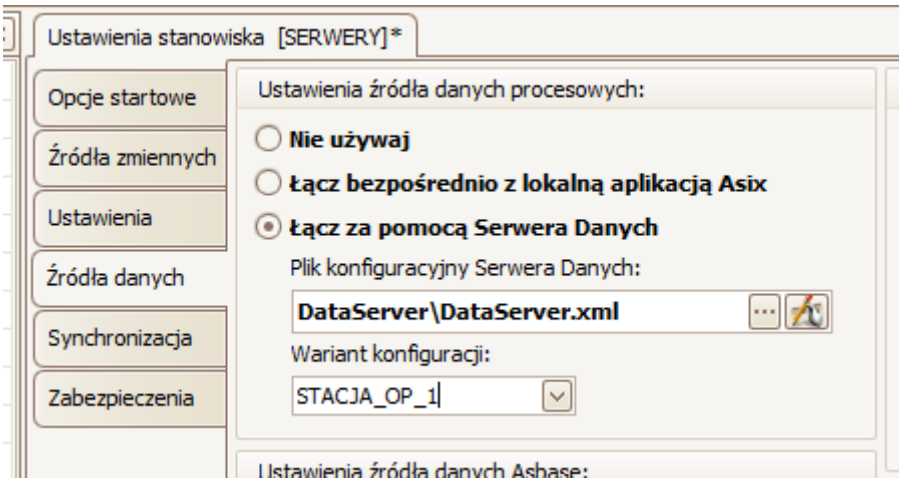
2 Konfigurowanie dostępu do danych bieżących i archiwalnych

Kluczowym elementem każdej aplikacji jest zdefiniowanie zestawu zmiennych procesowych oraz sposobu ich pozyskiwania i archiwizowania. Parametryzacja dostępu do danych procesowych wykonywana jest niezależnie dla każdego stanowiska aplikacji. Jedynym wyjątkiem jest baza zmiennych, która jest identyczna dla wszystkich stanowisk, aczkolwiek stanowiska mogą używać różnych baz definicji zmiennych.

2.1Konfiguracja serwera danych

Podstawową decyzję jaką należy podjąć to sposób pozyskiwania danych. Aplikacje typu Evo do odczytu wartości zmiennych ze sterowników oraz ich archiwizacji wykorzystuje moduły Asmen i Aspad, używane również w aplikacjach starego typu. Moduły te uruchamiane są w ramach tzw.**serwera danych**.

W panelu eksploratora aplikacji należy wybrać węzeł *Ustawienia stanowisk* i wybrać zakładkę *Źródła danych.*



Rys. Ustawienia źródła danych procesowych dla stanowiska.

W zakładce tytułowej pokazana jest nazwa stanowiska, które jest aktualnie parametryzowane. Zmienić stanowisko można w panelu *Stanowiska*.

Możliwa jest praca z wykorzystaniem jednej z trzech strategii:

a. *Nie używaj*

W tym trybie serwer danych nie jest używany, a co za tym idzie nie ma możliwości bezpośredniego odczytu danych ze sterowników oraz ich archiwizacji. Zasadniczo, tryb używany jest tylko na stanowiskach terminalowych okienkowych i WWW – stanowisko pobiera wszystkie dane z innego stanowiska aplikacji pracującego w trybie serwera.

b. *Łącz bezpośrednio z lokalną aplikacją Asix*

Rzadko używany tryb pozwalający na dostęp do danych poprzez aplikację Asix starego typu, uruchomioną na tym samym komputerze. Typowo stosowany w przypadku tworzenia przeglądarkowej wersji aplikacji starego typu. Aplikacja taka jest konwertowana do wersji Evo, a stanowisko serwerowe Asix.Evo uruchamiane jest na tym samym komputerze co oryginalny serwer i służy jako źródło danych dla stanowisk przeglądarkowych.

W tym wariancie serwer danych Asix.Evo uruchamiany jest w ramach procesu klasycznego Asixa. W tym celu należy włączyć użycie serwera w programie Architekt: zakładka *Programy/Asix.evo* w panelu *Parametry startowe*.

c. *Łącz za pomocą Serwera Danych*

Jest to podstawowy tryb pracy. Praktycznie w każdej aplikacji, przynajmniej jedno stanowisko pracuje w tym trybie. Serwer danych odczytuje dane ze sterowników kontrolujących proces oraz może je archiwizować. Serwer danych może być też wykorzystany do dwukierunkowej wymiany danych pomiędzy aplikacją Asix.Evo a aplikacją starego typu. Parametryzacja serwera danych wymaga podania nazwy pliku konfiguracyjnego oraz wariantu pracy. Jest to taki sam plik, jaki używany jest w aplikacjach starego typu. Do jego edycji służy program Architekt – obsługa programu opisana jest w jego własnej dokumentacji. Plik konfiguracyjny serwera danych może zostać

utworzony w trakcie budowy nowej aplikacji przy pomocy kreatora. Można też użyć plik stosowany wcześniej w aplikacji starego typu. Przy pomocy przycisku  można uruchomić program Architekta z załadowaniem właściwego pliku konfiguracyjnego. W przypadku aplikacji Asix.Evo rola Architekta ogranicza się do konfiguracji kanałów komunikacyjnych i archiwów oraz do tworzenia bazy definicji zmiennych.

2.2 Bazy definicji zmiennych procesowych

W bazie definicji zmiennych przechowywane są informacje o wszystkich zmiennych procesowych używanych w aplikacji. Dotyczy to zarówno zmiennych odczytywanych ze sterowników jak i zmiennych wirtualnych. Ogólne zasady użycia bazy definicji zmiennych nie uległy zmianie w stosunku do wcześniejszych wersji programu.

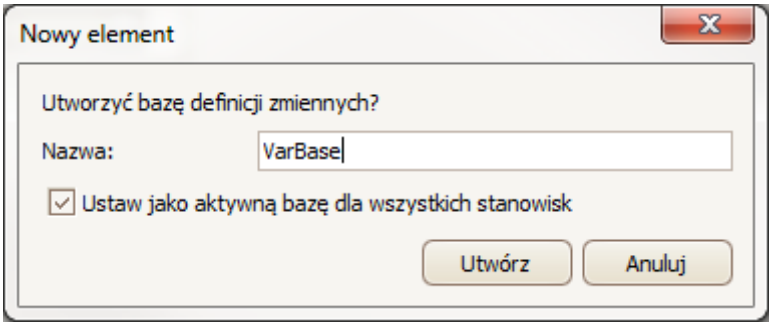
W aplikacjach Asix.Evo można stosować bazy definicji w formatach MDB, SQL lub XML. Bazy typu MDB i SQL pochodzą z wcześniejszych wersji programu. Do ich edycji lub generacji służy program Architekt. Szczegółowy opis zasad użycia baz MDB i SQL znajduje się w podręcznikach opisujących konstrukcje aplikacji klasycznych.

Bazy XML stosowane są tylko w aplikacjach Asix.Evo. Mają one znaczenie pomocnicze. Służą zasadniczo do definiowania zmiennych wirtualnych.

Bazy zmiennych przechowywane w bazach danych SQL stosowane są z reguły w dużych wielostanowiskowych aplikacjach, w których istotna jest centralna lokalizacja informacji o zmiennych.

2.2.1 Utworzenie nowej bazy

W panelu *Eksplorator aplikacji,* w menu kontekstowym węzła *Bazy definicji zmiennych* wykonaj komendę *Utwórz bazę definicji zmiennych*.



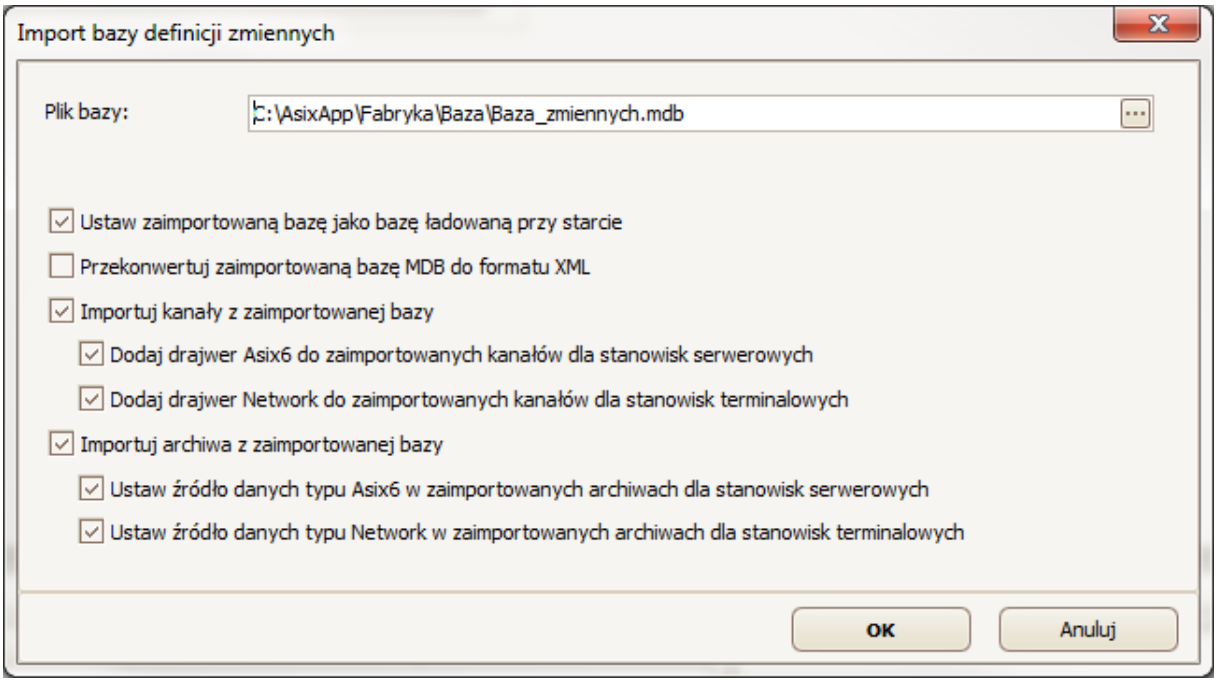
Rys. Okno utworzenia nowej bazy definicji zmiennych.

Wprowadź nazwę bazy i użyj przycisk *Utwórz*. Zostanie utworzona nowa baza typu **XML** ze standardowym zestawem atrybutów.

Bazy typu MDB i SQL tworzone są przy pomocy programu Architekt. Jeżeli aplikację utworzono przy pomocy kreatora, to baza typu MDB została założona i dołączona w trakcie pracy kreatora.

2.2.2 Importowanie utworzonej wcześniej bazy

Aby dodać do aplikacji istniejącą bazę zmiennych w formacie XML lub MDB, w panelu *Eksplorator aplikacji* w menu kontekstowym węzła *Bazy definicji zmiennych* wykonaj komendę *Importuj plikową bazę definicji zmiennych*.



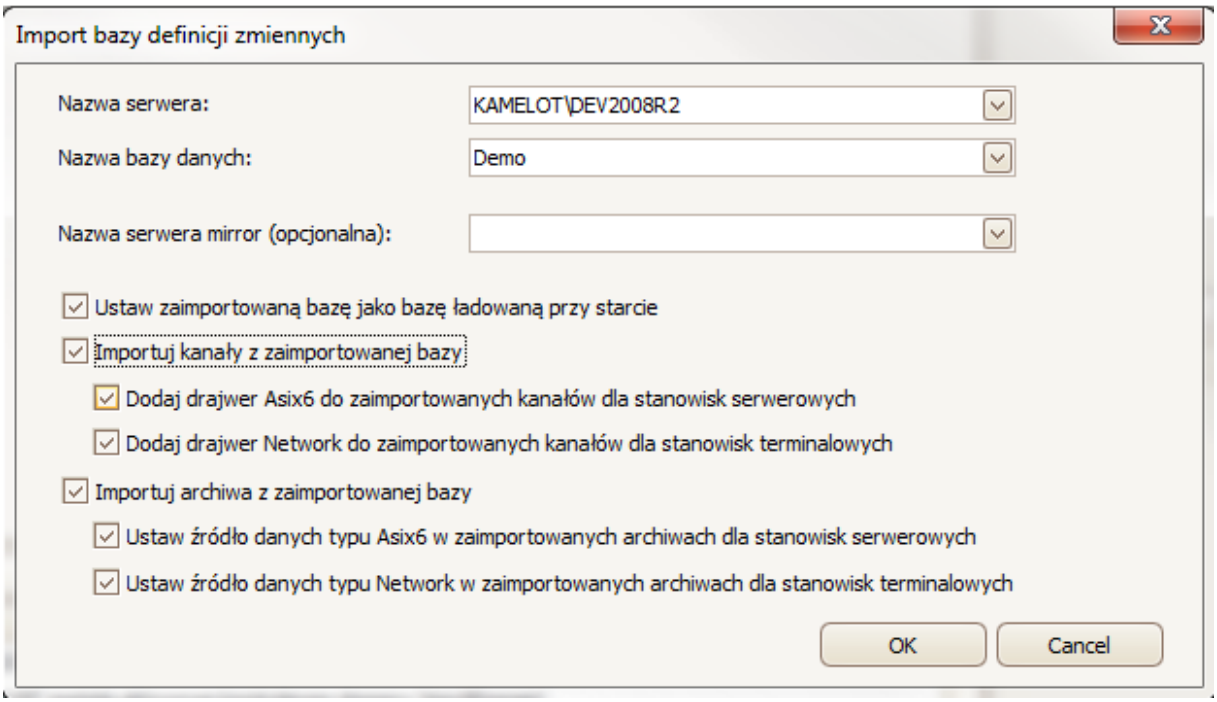
Rys. Okno importu bazy definicji zmiennych.

Wprowadź nazwę istniejącej bazy i użyj przycisk OK. W momencie importu plik bazy definicji zmiennych jest kopiowany do podkatalogu *VariableSources* katalogu definicyjnego aplikacji.

W okienku importu dostępnych jest szereg opcji sterujących procesem importu. Dotyczą one głównie sposobu automatycznego wykrywania i konfigurowania kanałów i archiwów użytych w definicjach zmiennych. Domyślne ustawienia są z reguły optymalne. Pozwalają one uniknąć ręcznego definiowania kanałów i archiwów. Typy kanałów i archiwów opisane są w kolejnych rozdziałach.

W przypadku dołączania bazy zmiennych typu SQL należy wykonać komendę *Dołącz bazę definicji zmiennych SQL*. W oknie importu należy podać nazwę serwera SQL oraz nazwę bazy zmiennych. Opcjonalnie można podać nazwę serwera lustrzanego, który używany jest wtedy, gdy nie uda się

odczytać bazy zmiennych ze standardowej lokalizacji. Pozostałe opcje importu są identyczne jak dla baz XML i MDB.

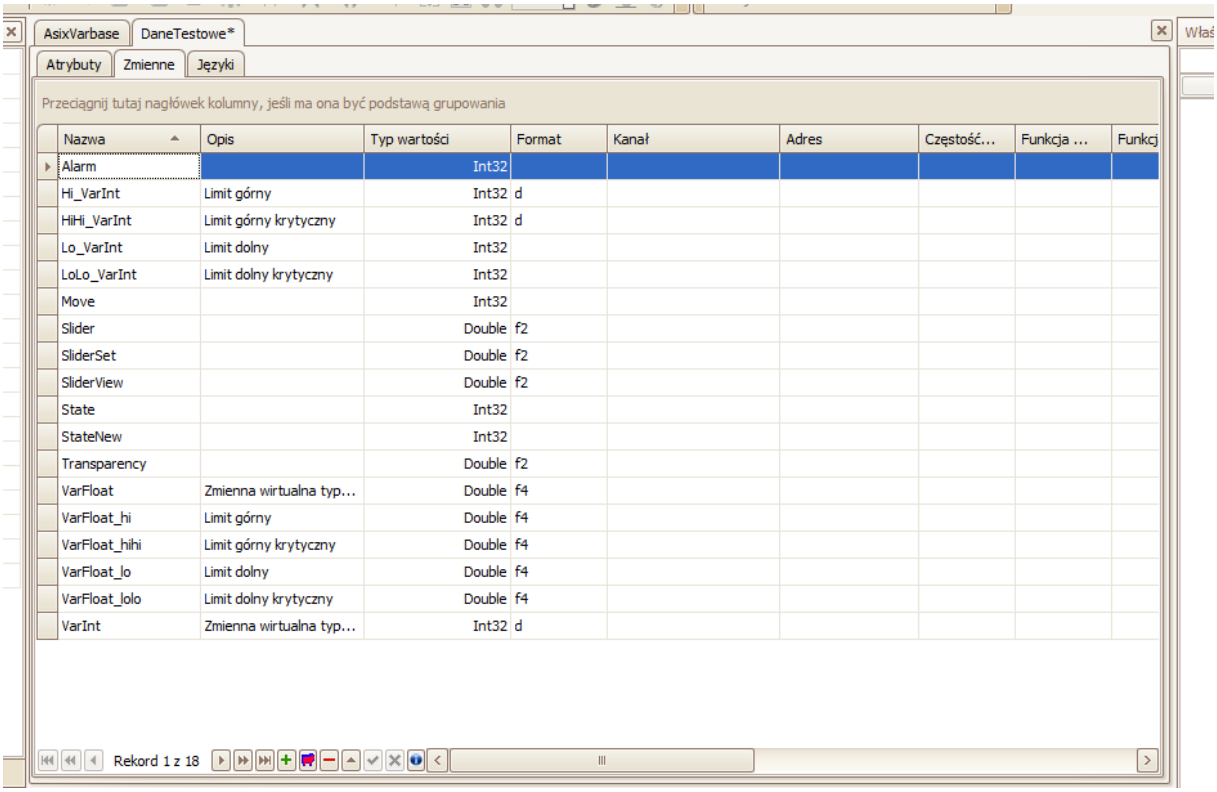


Rys. Okno importu bazy definicji zmiennych.

Cechą specyficzną baz zmiennych SQL jest to, że definicje zmiennych nie są kopiowane do katalogu definicyjnego – są zawsze czytane bezpośrednio z serwera SQL. Dotyczy to także terminali przeglądarkowych – muszą one mieć dostęp do serwera. Dlatego w aplikacjach opartych na pracy w trybie przeglądarkowym, jeżeli nie są one ograniczone do sieci lokalnej, zalecane jest użycie plikowych baz definicji zmiennych.

2.2.3 Edycja i przeglądanie bazy definicji zmiennych

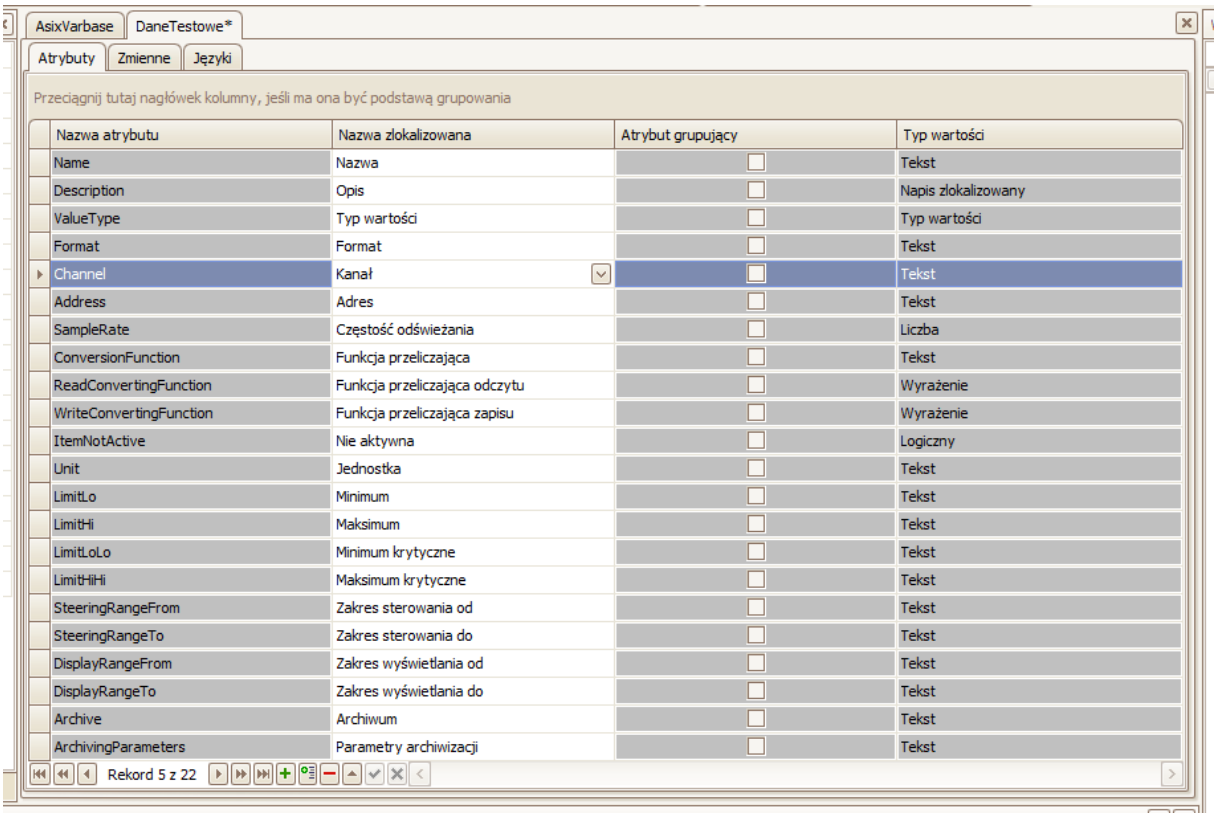
W panelu *Eksplorator aplikacji* kliknij podwójnie na węźle bazy definicji zmiennych. Otwarty zostanie poniższy panel roboczy bazy definicji zmiennych. **W przypadku baz typu MDB i SQL możliwe jest wyłącznie przeglądanie zawartości bazy. Dla baz XML dostępne są funkcje edycji zmiennych**.



Rys. Panel zestawu zmiennych bazy definicji zmiennych.

W zakładce *Zmienne* wykonywana jest edycja atrybutów zmiennych. Założenie nowej zmiennej lub jej usunięcie wykonywane jest przy pomocy przycisków zintegrowanego paska narzędziowego lub komend menu kontekstowego.

Bazy XML zakładane są ze standardowym zestawem atrybutów zmiennych. Zmienić ten zestaw można w zakładce *Atrybuty*.



Rys. Panel atrybutów zmiennych.

W panelu pokazane są opisy wszystkich atrybutów aktualnie używanych w bazie definicji zmiennych. Każdy atrybut zdefiniowany jest przez:

a. *Nazwa atrybutu* – niezależna od języka nazwa atrybutu, używana w trakcie parametryzacji diagramów w funkcji *Attribute* i w notacji skrótowej @.

b. *Nazwa zlokalizowana* – wielojęzyczna nazwa opisowa używana w oknie wyboru zmiennej i panelu definicji zmiennych.

c. *Atrybut grupujący* – opcja oznaczająca atrybut jako atrybut grupujący, czyli służący do budowy drzewa hierarchii w oknie wyboru zmiennej.

d. *Typ wartości* – określenie typu atrybutu, pozwala na dobranie odpowiedniego edytora pola.

Do istniejącej bazy można dodać własny atrybut o dowolnym znaczeniu, stosując przycisk  paska narzędziowego. Typowo stosowane jest to do dodania atrybutów grupujących. Można też dodać atrybuty predefiniowane, stosując przycisk .

Zakładka *Języki* pozwala na zadeklarowanie w jakich językach podawane są atrybuty opisowe zmiennych i atrybutów.

Edycja lub generacja baz MDB wykonywana jest przy pomocy programu Architekt. Jeżeli baza MDB zostanie zmodyfikowana w trakcie pracy nad aplikacją Asix.Evo, to można spowodować jej

przeładowanie komendą *Przeładuj aktywne bazy* z menu kontekstowego w panelu eksploratora aplikacji.

2.2.4 Atrybuty zmiennych

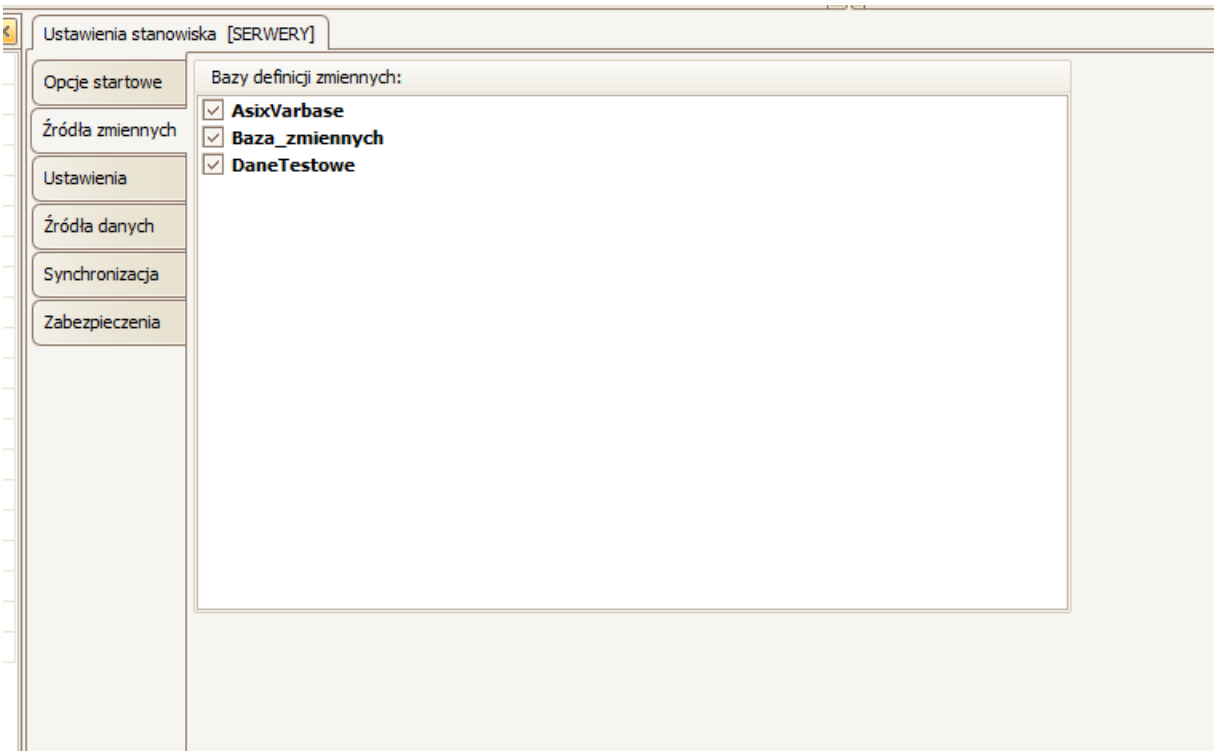
Większość atrybutów stosowanych w bazach definicji zmiennych aplikacji Asix.Evo jest identyczna jak w przypadku aplikacji klasycznych. Poniższa lista opisuje jedynie różnice w interpretacji atrybutów. Pełna lista atrybutów zmiennych znajduje się w dokumentacji Architekt.pdf/chm, w załączniku 1.

Nazwa	**Opis**
Typ wartości	Określa typ wartości zmiennej w konwencji .Net. W przypadku użycia bazy MDB nie posiadającej tego atrybutu, jest on tworzony wirtualnie na podstawie użytej w definicji zmiennej funkcji przeliczającej i formatu.
Format	Sposób formatowania tekstowego wartości w konwencji .Net. Skrócony opis formatów .Net można znaleźć w podręczniku *AsixEvo_Techniki_budowy_diagramów.pdf/chm.* W przypadku baz MDB atrybut tworzony jest wirtualnie według poniższych zasad: a. Jeżeli w bazie jest atrybut *FormatEvo,* to jest on traktowany wprost jako definicja atrybutu *Format* Asix.Evo. b. Jeżeli w bazie nie ma atrybutu *FormatEvo*, to atrybut *Format* bazy MDB (podawany w konwencji formatów aplikacji klasycznych) jest w trakcie ładowania tłumaczony na odpowiadający mu format .Net.
Funkcja przeliczająca odczytu	Atrybut ma postać wyrażenia, które wstępnie przelicza pobraną z drajwera wartość surową zmiennej. W treści wyrażenia odwołanie do wartości surowej wykonywane jest przy pomocy funkcji *RawValue*. W przypadku korzystania z serwera danych (kanał Asix6), niezależnie od definicji atrybutu funkcji przeliczającej odczytu, wartość zmiennej podlega standardowemu przeliczeniu przez Asmen'a na podstawie atrybutu *Funkcja przeliczająca*. W przypadku użycia bazy MDB nie posiadającej atrybutu *Funkcja przeliczająca odczytu*, jest on tworzony wirtualnie na podstawie użytej w definicji zmiennej formatu.

Funkcja przeliczająca zapisu	Atrybut ma postać wyrażenia, które przelicza wartość sterującą przed wysłaniem jej do drajwera. Pozostałe zasady użycia są identyczne jak dla atrybutu *Funkcja przeliczająca odczytu*.
Wartość początkowa	Określa wartość początkową zmiennej, nadawaną zmiennej w momencie uruchomienia aplikacji. Pozostałe zasady użycia są identyczne jak dla atrybutu *Funkcja przeliczająca odczytu*.

2.2.5 Ustawianie statusu aktywności bazy

Nie wszystkie bazy definicji zmiennych zdefiniowane w aplikacji muszą być używane na każdym stanowisku. W panelu eksploratora aplikacji kliknij podwójnie na węźle *Ustawienia stanowisk,* a następnie wybierz zakładkę *Źródła zmiennych.* W panelu *Stanowiska* wybierz obszar lub stanowisko.

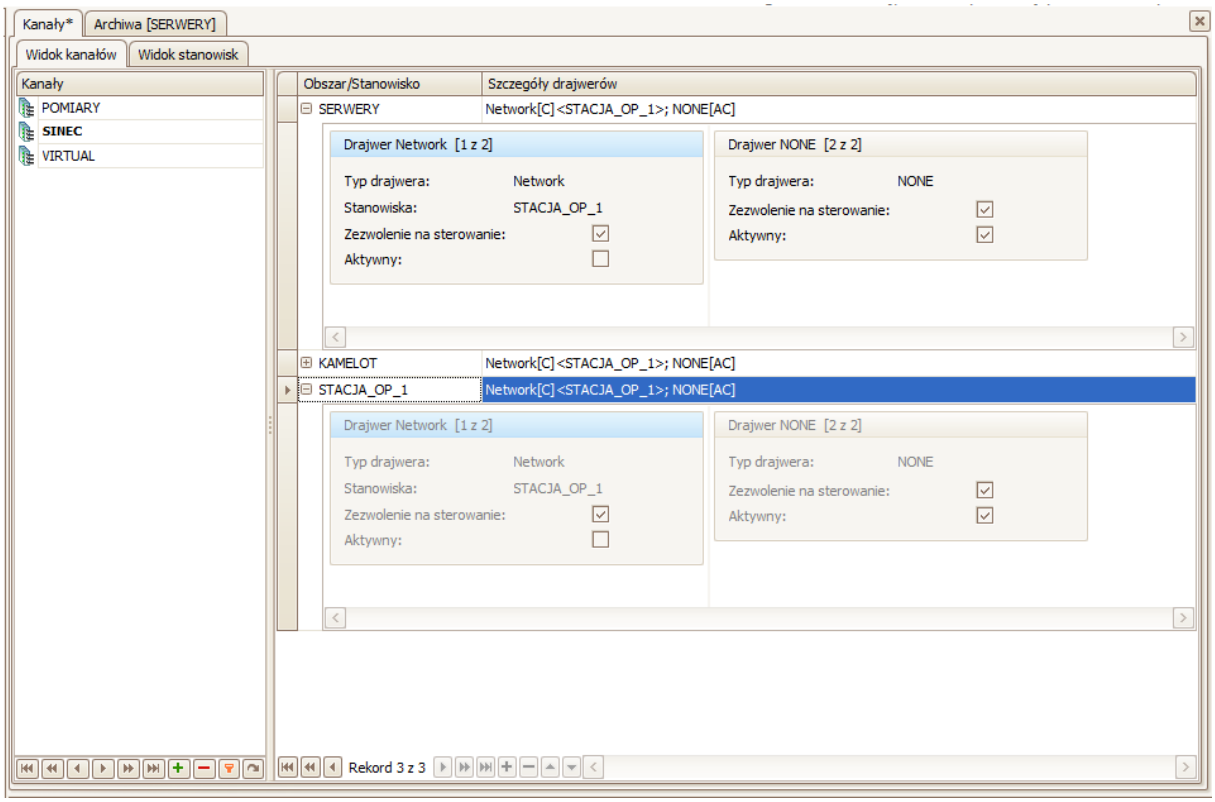


Rys. Ustawienie statusu aktywności bazy w ustawieniach stanowiska.

W otwartym panelu roboczym można wybrać bazy definicji zmiennych, które mają być używane na wybranym stanowisku (ładowane przy starcie).

2.3 Kanały komunikacyjne

Każdy kanał komunikacyjny użyty w definicjach zmiennych procesowych powinien być sparametryzowany pod kątem drajwerów komunikacyjnych używanych przez kanał. Panel roboczy parametryzacji kanałów otwierany jest przez podwójne kliknięcie na węźle *Kanały* panelu eksploratora aplikacji.



Rys. Panel parametryzacji kanałów komunikacyjnych.

Panel kanałów składa się z dwóch części. W tzw. *widoku kanałów*, w lewej pokazana jest lista wszystkich kanałów aplikacji. Lista ta może być utworzona podczas tworzenia aplikacji przy pomocy kreatora lub przy importowaniu bazy definicji zmiennych, na podstawie skanowania zawartości bazy. Można też, w dowolnym momencie, wykonać funkcję *Importuj kanały i archiwa* z menu kontekstowego węzła bazy definicji zmiennych w panelu eksploratora, w celu uaktualnienia listy kanałów. Niezależnie od metod automatycznych, przy pomocy przycisków paska narzędziowego i komend menu kontekstowego, można ręcznie dodawać, usuwać i kopiować kanały.

Prawa część panelu pokazuje sposób zdefiniowania kanałów na poszczególnych stanowiskach aplikacji. W wersji zwiniętej, wyświetlane są tylko nazwy kanałów ze skrótową definicją użytych drajwerów. W wersji rozwiniętej możliwa jest zmiana konfiguracji. Dla każdego kanału można dodać

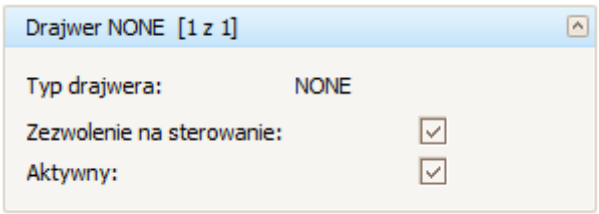
po 1 drajwerze z każdego typu. W trakcie działania aplikacji użyty zostanie pierwszy drajwer z listy, który ma ustawioną opcję aktywności i udało się go poprawnie uruchomić. Jeżeli pracujący drajwer nie jest drajwerem o najwyższym priorytecie i w trakcie jego działania system wykryje, że któryś z drajwerów o wyższym priorytecie może pracować, to nastąpi powrotne przełączenie na ten drajwer. Dodawać i usuwać drajwery oraz zmieniać ich kolejność można przy pomocy przycisków paska narzędziowego lub komend menu kontekstowego.

Litery A i C wyświetlane w skrótowym opisie drajwera oznaczają odpowiednio: stan aktywności drajwera i zezwolenie na sterowanie. Jeżeli tabliczki drajwera są wyszarzone, to znaczy to, że parametryzacja została odziedziczona z obszaru nadrzędnego.

W alternatywnej wersji panelu, tzw. *widoku stanowisk*, konfiguracja kanałów pokazywana jest z punku widzenia zdefiniowanych stanowisk. Ogólne zasady postępowania są identyczne jak dla widoku kanałów.

2.3.1 Drajwery komunikacyjne

Każdy z drajwerów posiada parametry standardowe oraz opcjonalny zestaw parametrów dla niego specyficznych.



Rys. Okno parametrów standardowych drajwera komunikacyjnego.

Parametry standardowe opisuje poniższa tabela.

Tab.: Opis parametrów standardowych drajwerów komunikacyjnych.

Nazwa	**Opis**
Typ drajwera	Określa typ użytego drajwera komunikacyjnego.
Zezwolenie na sterowania	Określa, czy na wybranym stanowisku (w obszarze) możliwe jest wykonywanie sterowań poprzez definiowany drajwer.

Aktywny	Określa, czy drajwer ma być używany parametr pozwala na wyłączenie drajwera bez usuwania jego definicji.

2.3.1.1 Drajwer NONE

Drajwer NONE to drajwer wirtualny, który nie wykonuje żadnych operacji komunikacyjnych. Służy wyłącznie do przechowywania zmiennych, których wartości ustawiane są przy pomocy skryptów lub akcji operatorskich. Jeżeli w kanale nie zdefiniowano żadnego aktywnego drajwera, to oznacza to użycie niejawnego drajwera NONE.

Tab.: Parametry niestandardowe drajwera None.

Nazwa	**Opis**
Inicjalizacja zmiennych	Ustawienie opcji powoduje, że wszystkie zmienne w kanale mają na starcie aplikacji ustawianą zerową wartość początkową i status Good. W przeciwnym wypadku zmienne mają nadawaną wartość Null i status Bad.

2.3.1.2 Drajwer Network

Drajwer Network służy do wymiany sieciowej danych pomiędzy stanowiskami aplikacji Asix.Evo. Stosowany na stanowiskach typu terminalowego w celu odczytu danych ze stanowisk typu serwerowego. (Serwery również mogą pobierać dane z innych serwerów).

Tab.: Parametry niestandardowe drajwera Network.

Nazwa	**Opis**
Stanowiska	Lista stanowisk aplikacji Asix.Evo typu serwerowego, do których drajwer może się podłączyć w celu pozyskania danych.

2.3.1.3 Drajwer Asix6

Drajwer Asix6 należy wybrać dla wszystkich kanałów, których zmienne są obsługiwane za pośrednictwem serwera danych, czyli w praktyce wszystkich kanałów fizycznych. Rzeczywisty typ kanału określony jest w pliku konfiguracyjnym serwera danych. Plik ten jest parametryzowany w programie Architekt. Zasady jego parametryzacji są identyczne jak dla aplikacji klasycznych i opisane są w podręczniku programu Architekt.

Zobacz też: 2.1. *Konfiguracja serwera danych*

Drajwer nie posiada żadnych parametrów niestandardowych.

2.3.1.4 Drajwer Random

Drajwer typu *Random* przeznaczony jest do użycia w fazie uruchamiania aplikacji bez dostępu do rzeczywistych źródeł danych. Efektem działania drajwera jest symulacja losowych wartości zmiennych w zakresie wynikającym z atrybutów minimalnej i maksymalnej wartości wyświetlania oraz z częstotliwością określoną w atrybucie częstotliwości odświeżania.

Drajwer nie posiada żadnych parametrów niestandardowych.

2.3.2 Rodzaje kanałów wirtualnych

Dodatkowego wyjaśnienia wymaga sposób deklarowania kanałów wirtualnych typu None. Można to wykonać na trzy sposoby. Poniższa tabela opisuje cechy specyficzne poszczególnych sposobów deklaracji.

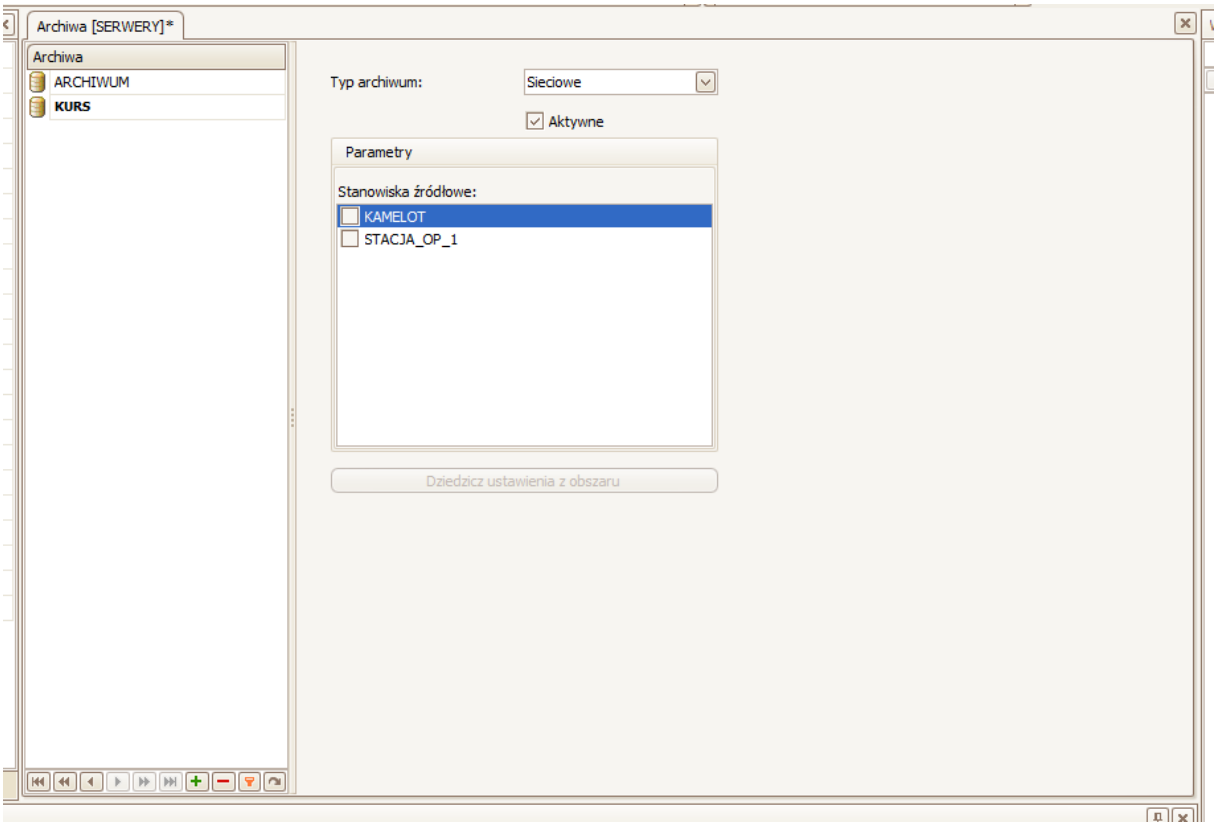
Tab.: Sposoby deklarowania kanałów wirtualnych typu None.

Sposób deklaracji	**Opis**
Zmienna bez podania nazwy kanału komunikacyjnego	Zmienne bez nazwy kanału przydzielane są automatycznie do domyślnego kanału wirtualnego. Typ zmiennej może być dowolnym typem .Net. Zmienne te nie mogą być jednak zapisywane w archiwum długookresowym Aspada. Zakres

	widzialności zmiennych ograniczony jest do lokalnego stanowiska – nie mogą być udostępniane mechanizmami sieciowymi.
Zmienne w kanałach z drajwerem typu None lub bez jawnie zdefiniowanego drajwera.	Typ zmiennej może być dowolnym typem .Net. Zmienne te nie mogą być zapisywane w archiwum długookresowym Aspada. Istnieje możliwość udostępniania wartości zmiennych na inne stanowiska wykonujące aplikację Asix.Evo.
Zmienne w kanałach z drajwerem Asix6 i kanałem typu None Asmena	Typ zmiennej może być dowolnym typem Asmena. Wartości zmiennych mogą być zapisywane w archiwum długookresowym Aspada. Istnieje możliwość udostępniania wartości zmiennych na inne stanowiska wykonujące aplikację typu klasycznego lub Asix.Evo.

2.4 Deklaracja archiwów wartości zmiennych procesowych

Każde archiwum użyte w definicjach zmiennych procesowych powinno być sparametryzowany pod kątem rodzaju archiwizacji. Panel roboczy parametryzacji archiwów otwierany jest przez podwójne kliknięcie na węźle *Archiwa* panelu eksploratora aplikacji.



Rys. Panel deklaracji archiwów wartości zmiennych procesowych.

Panel archiwów pokazuje parametryzację w kontekście konkretnego stanowiska lub obszaru, którego nazwa pokazana jest w zakładce tytułowej panelu. Stanowisko można zmienić za pośrednictwem panelu *Stanowiska*.

Panel archiwów składa się z dwóch części. W lewej pokazana jest lista wszystkich archiwów używanych w aplikacji. Lista ta może być utworzona podczas tworzenia aplikacji przy pomocy kreatora lub importowaniu bazy definicji zmiennych, na podstawie skanowania zawartości bazy. Można też, w dowolnym momencie, wykonać funkcję *Importuj kanały i archiwa* z menu kontekstowego węzła bazy definicji zmiennych w panelu eksploratora, w celu uaktualnienia listy archiwów. Niezależnie od metod automatycznych, przy pomocy przycisków paska narzędziowego i komend menu kontekstowego, można ręcznie dodawać, usuwać i kopiować archiwa.

> Prawa część panelu służy do wyboru rodzaju archiwum oraz zdefiniowania parametrów jego pracy. Należy pamiętać o tym, że oprócz parametrów archiwum, każda zmienna posiada indywidualne parametry archiwizacji podane w atrybucie *Parametry archiwizacji*.
>
> Przycisk *Dziedzicz ustawienia z obszaru* służy do ustawienia parametrów archiwum zgodnie z ustawieniami obszaru nadrzędnego. Jeżeli przycisk jest wyszarzony, to aktualne ustawienia są zgodne z ustawieniami obszaru nadrzędnego.

2.4.1 Rodzaje archiwów

2.4.1.1 Archiwum Pamięciowe

Pomocnicze archiwum, w którym wartości zmiennych przechowywane są wyłącznie w pamięci operacyjnej bez zapisywania w pliku dyskowym. Archiwum posiada dane tylko od momentu uruchomienia aplikacji.

Archiwum nie posiada żadnych parametrów konfiguracyjnych.

W atrybucie *Parametry archiwizacji* zmiennej należy podać horyzont archiwizacji wyrażony jako liczba minut.

2.4.1.2 Archiwum Sieciowe

Archiwum sieciowe służy do odczytu danych archiwalnych z innych stanowisk aplikacji Asix.Evo. Stosowany na stanowiskach typu terminalowego w celu odczytu danych ze stanowisk typu serwerowego.

W parametrach archiwum należy określić stanowiska serwerowe, z których należy pobierać dane archiwalne.

Zawartość atrybutu *Parametry archiwizacji* zmiennej nie ma dla archiwum sieciowego żadnego znaczenia.

2.4.1.3 Archiwum Asix6

Archiwum Asix6 przeznaczone jest do długookresowej archiwizacji wartości zmiennych w archiwach plikowych lub bazodanowych. Archiwa Asix6 są obsługiwane za pośrednictwem serwera danych. Szczegółowe parametry trybu pracy archiwum określone są w pliku konfiguracyjnym serwera danych. Plik ten jest parametryzowany w programie Architekt. Zasady jego parametryzacji są identyczne jak dla aplikacji klasycznych i opisane są w podręczniku programu Architekt.

Zobacz też: *2.1. Konfiguracja serwera danych*

Archiwum nie posiada żadnych parametrów konfiguracyjnych.

Atrybut *Parametry archiwizacji* zmiennych jest interpretowany identycznie jak w przypadku aplikacji klasycznych – należy go definiować za pośrednictwem programu Architekt.

2.4.1.4 Archiwum Losowe Dane

Pomocnicze archiwum, które tworzy symulowane przebiegi wartości zmiennych. Przeznaczone do użycia w celach demonstracyjnych lub do testowania pracy aplikacji bez tworzenia rzeczywistego archiwum wartości.

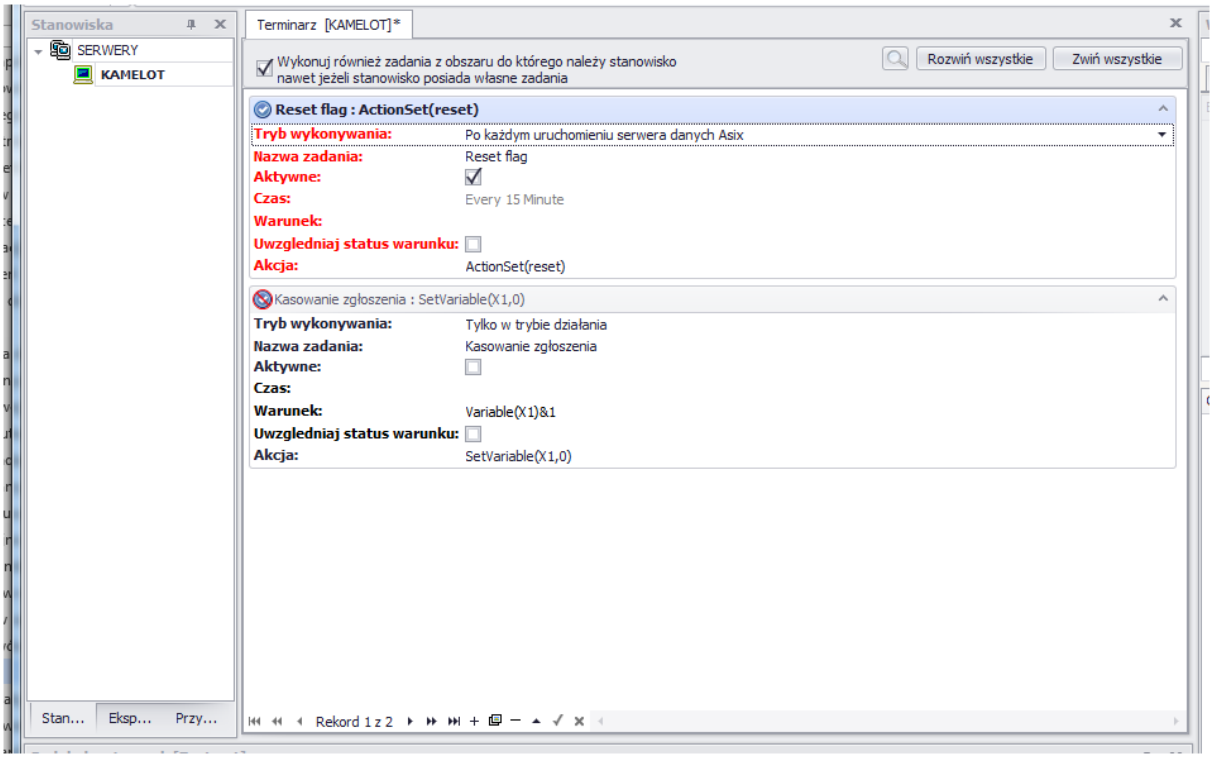
Archiwum nie posiada żadnych parametrów konfiguracyjnych.

Zawartość atrybutu *Parametry archiwizacji* zmiennej stanowią trzy liczby całkowite oddzielone spacjami. Pierwsza określa odstępy czasu pomiędzy zwracanymi próbkami [s], druga dolną granicę wartości, a trzecia górną granicę wartości.

3 Konfiguracja Terminarza

Mechanizm Terminarza pozwala na automatyczne wykonywanie akcji operatorskich. Akcje mogą być wykonywane w zadanym cyklu czasowym, w reakcji na zajście swobodnie definiowanych warunków lub w momentach zajścia niektórych zdarzeń systemowych.

Zadania terminarza definiowane są w panelu roboczym otwieranym za pośrednictwem węzła *Terminarz* w panelu eksploratora aplikacji.



Rys. Panel Terminarza.

Zadania terminarza należą do ustawień zależnych od stanowiska. Każde stanowisko może mieć własny zestaw zadań. Stanowisko może też dziedziczyć zadania z obszaru nadrzędnego. Dziedziczenie jest aktywne, gdy na stanowisku żadne zadanie własne nie zostało zdefiniowane lub została zaznaczona opcja *Wykonaj również zadania z obszaru do którego należy stanowisko...*

Nowe zadanie do terminarza dodawane jest przy pomocy przycisku + paska narzędziowego lub komendą z menu kontekstowego.

Właściwość*Nazwa zadania* ma znaczenie tylko informacyjne. Opcja *Aktywne* pozwala tymczasowo wyłączyć wykonywanie zadania bez jego usunięcia.

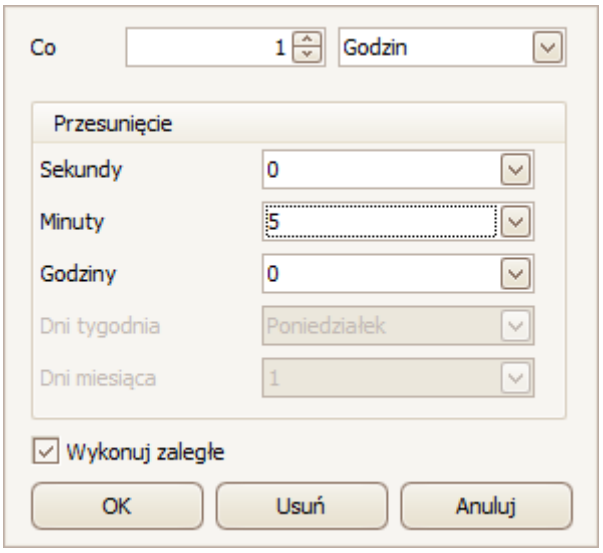
Za wybór momentu wykonania akcji operatorskiej zdefiniowanej w zadaniu odpowiedzialne są właściwości *Tryb wykonania, Czas, Warunek* i*Uwzględniaj status warunku*.

Właściwość *Tryb wykonania* może przyjmować wartości opisane w poniższej tabeli.

Tab. Wartości trybu wykonania akcji operatorskiej zdefiniowanej w zadaniu.

Wartość	Znaczenie
Zawsze	*Nie są nakładane żadne ograniczenia, o momencie wykonania decydują ustawienia właściwości Czas i Warunek.*
Tylko w trybie działania	Akcja wykonywana jest tylko wtedy, gdy aplikacja jest w trybie pełnego działania, o momencie wykonania decydują ustawienia właściwości *Czas* i *Warunek.*
Zawsze z wyjątkiem trybu architekta	Akcja wykonywana jest tylko wtedy, gdy aplikacja jest w trybie pełnego działania lub w trybie uproszczonej edycji, o momencie wykonania decydują ustawienia właściwości *Czas* i *Warunek.*
Jednokrotnie przy starcie aplikacji	Akcja wykonywana jednokrotnie w momencie załadowania aplikacji, niezależnie od trybu uruchomienia. Ten tryb jest często używany do uruchomienia tzw. skryptów rezydentnych aplikacji.
Jednokrotnie przy starcie aplikacji w trybie pracy	Akcja wykonywana jednokrotnie w momencie uruchomienia aplikacji w trybie działania (z parametrem *run*). Ten tryb jest często używany do uruchomienia tzw. skryptów rezydentnych aplikacji.
Jednokrotnie po pierwszym uruchomieniu serwera danych Asix	Akcja wykonywana jednokrotnie w momencie załadowania aplikacji po uruchomieniu serwera danych (w momencie, gdy komunikacja została już uruchomiona). Ten tryb jest często używany do uruchomienia tzw. skryptów rezydentnych aplikacji.
Po każdym uruchomieniu serwera danych Asix	Akcja wykonywana po każdorazowym uruchomieniu serwera danych (w momencie, gdy komunikacja została już uruchomiona). W przeciwieństwie do poprzedniego trybu, akcja jest wykonywana także wtedy, gdy serwer jest restartowany przy ciągle działającej aplikacji. Ten tryb jest często używany do inicjowania wartości zmiennych.
Przy zmianie bieżącego użytkownika	Akcja jest wykonywana przy każdej zmianie zalogowanego użytkownika.

Właściwość *Czas* służy do określenia cyklu wykonywania akcji operatorskiej.



Rys. Deklaracja cyklu wykonania akcji operatorskiej.

Oprócz podstawowego cyklu wykonania wyrażonego w dowolnej jednostce czasu, deklarowane jest też przesunięcie momentu początkowego. W pokazanym przykładzie akcja będzie wykonywana co jedną godzinę, 5 minut po każdej pełnej godzinie (0:05, 1:05 itd.). Wartość właściwości zostanie zapisana jako: *Every 1 Hour Offset: M(5)*.

Przykłady użycia:

Every 15 Minute - co 15 minut (w 0, 15, 30 i 45 minucie godziny)

Every 1 Hour - co 1 godzinę, dokładnie o pełnej godzinie

Every 8 Hour Offset: H(6) - co 8 godzin, o 6:00 , 14:00 i 22:00

Every 1 Week Offset: H(8)D(Monday) - co tydzień, w poniedziałek o godzinie 8:00

W okienku definicji czasu, można też wybrać opcję *Wykonuj* zaległe. Powoduje to, że w momencie startu aplikacji sprawdzane jest, czy poprzednie planowane wykonanie zostało zrealizowane. Jeżeli nie, to zaległe pojedyncze wykonanie jest realizowane poza normalnym kolejkowaniem.

Właściwość *Warunek* pozwala na definiowanie zadania, którego wykonanie zależy od spełnienia określonego warunku. Właściwość ma postać standardowego wyrażenia Asix.Evo i powinno zwracać wartość typu logicznego *true/false* (ewentualnie *różne od zera/równe zero*). Wzajemną zależność właściwości *Czas* i *Warunek* wyjaśnia poniższa tabela.

Tab. Zależność właściwości Czas i Warunek.

Sposób definicji	Znaczenie
Zdefiniowana tylko właściwość *Czas*	Akcja wykonywana zgodnie z zadanym cyklem czasowym
Zdefiniowana tylko właściwość *Warunek*	Akcja wykonywana w momencie zmiany wartości warunku z *false(równe zero)* na *true(różne od zera)*
Zdefiniowane właściwości Czas i Warunek	Akcja wykonywana zgodnie z zadanym cyklem czasowym pod warunkiem, że w chwili planowanego wykonania wyrażenie warunku ma wartość *true(różne od zera)*

Jeżeli opcja *Uwzględniaj status warunku* jest wybrana, to akcja zależna od warunku będzie wykonana tylko wtedy, gdy jest gwarancja poprawnego wyliczenia warunku. W przypadku, gdy używany jest tylko *Warunek*, status musi być poprawny dla poprzedniej i aktualnej wartości warunku.

3.1Inne metody automatycznego wykonywania akcji operatorskich

Oprócz mechanizmu terminarza, automatyczne wykonywanie akcji operatorskich możliwe jest również poprzez:

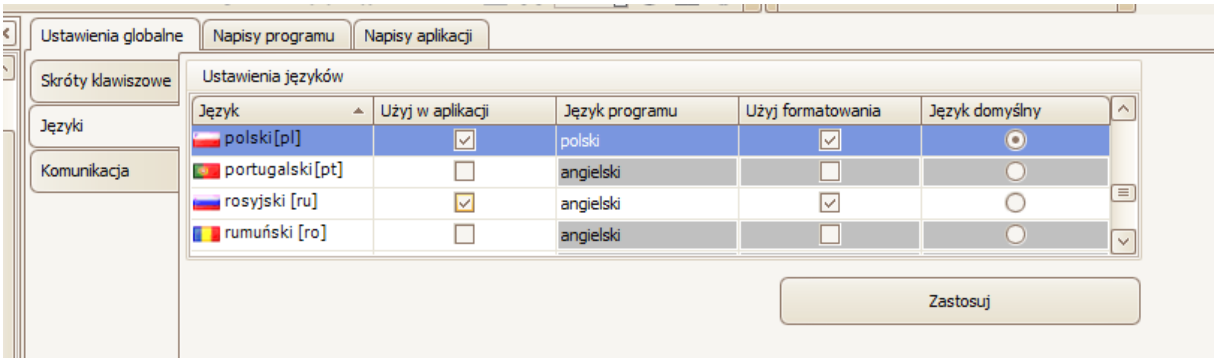
- Wykonywanie akcji operatorskich metodą *ExecuteAction* w kodzie skryptów aplikacyjnych.
- Akcje operatorskie zdarzeń *Otwarcie diagramu* i *Zamknięcie diagramu* deklarowane we właściwościach diagramów – akcje wykonywane w momencie otwarcia/zamknięcia diagramu.

4 Parametryzacja aplikacji wielojęzycznych

Asix.Evo pozwala na łatwe tworzenie aplikacji wielojęzycznych, czyli takich, które w trakcie pracy, na żądanie użytkownika, zmieniają język wszystkich wyświetlanych informacji tekstowych. Ponieważ teksty zapisane są w standardzie Unicode, możliwe jest użycie wszystkich języków, które ten standard obsługuje.

4.1Deklaracja języków aplikacji

Pierwszą operacją, którą należy wykonać tworząc aplikację wielojęzyczną, jest określenie języków, które będą używane. Można to wykonać już w trakcie tworzenia aplikacji przy pomocy kreatora. Alternatywnie, można użyć panel roboczy *Ustawienia języków*, otwierany za pośrednictwem węzła *Ustawienia globalne* panelu eksploratora aplikacji.



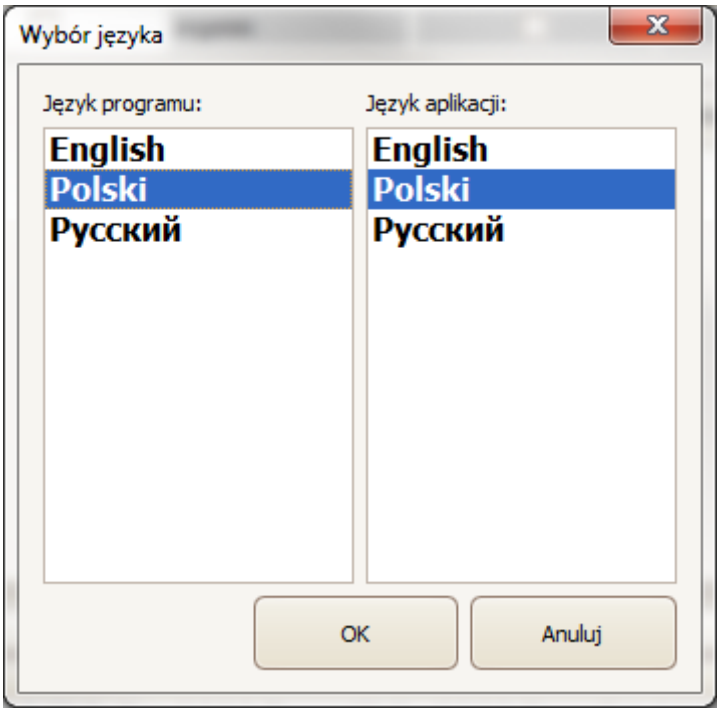
Rys. Panel deklaracji języków aplikacji.

W panelu ustawień języków należy zaznaczyć pole wyboru w kolumnie *Użyj w aplikacji* - dla każdego języka, który będzie używany w aplikacji. Oprócz tego dla używanych języków należy określić język programu. Język programu określa teksty stałe wbudowane w kod programu, np. opisy pól w oknach dialogowych, teksty komunikatów i komend w menu. Standardowo, można stosować dwa predefiniowane języki programu: polski i angielski. Projektant ma jednak możliwość zdefiniowania tekstów programu w innych językach.

Kolumna *Użyj formatowania* określa, czy przy przełączeniu aplikacji na dany język należy stosować formatowanie danych zgodne z ustawieniami narodowymi. Dotyczy to głównie wyświetlania dat. W kolumnie *Język domyślny* należy zaznaczyć - który z języków aplikacji należy użyć w momencie jej uruchomienia.

4.2 Zmiana języka pracy

W trakcie edycji aplikacji aktualny język może być przełączony przy użyciu komendy *Wybór języka* z menu *Narzędzia*.



Rys. Okno przełączania języka aplikacji i programu.

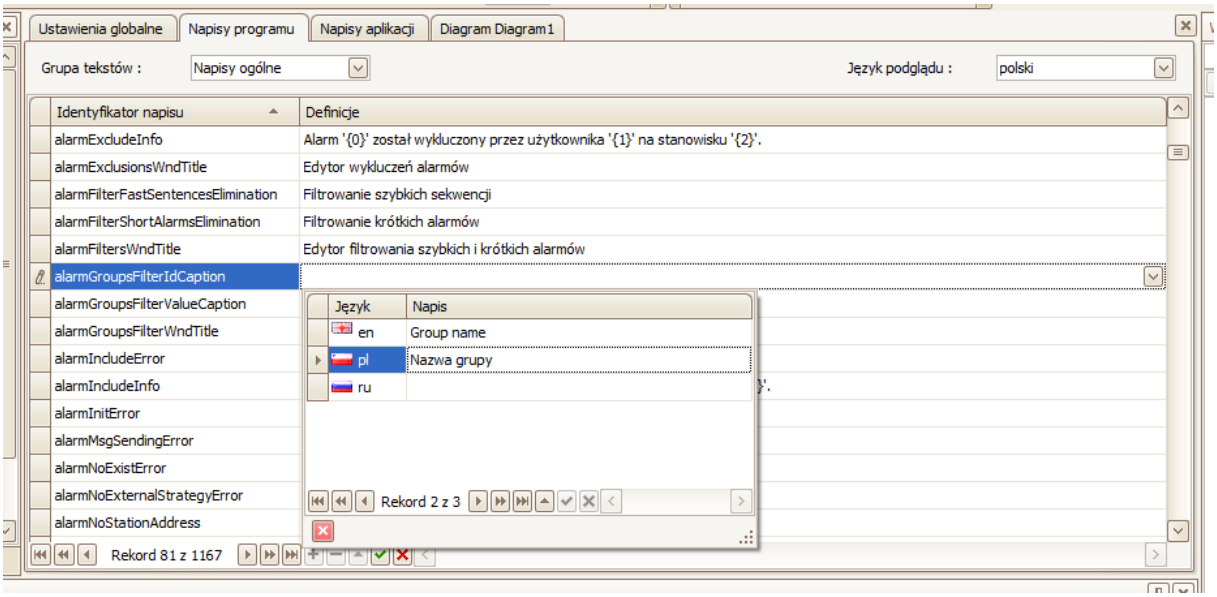
Korzystając z wyświetlonego okna, można dowolnie ustawić zarówno język aplikacji jak i język programu.

W trybie wykonywania aplikacji język można zmienić przy użyciu okienka panelu kontrolnego. Można też zaprojektować własny system przełączania języka wbudowany w diagramy aplikacji. W tym celu należy posłużyć się akcją operatorską *SwitchLanguage*. Akcja wymaga podania parametru w postaci dwuliterowego kodu języka. Kody języków wyświetlone są w panelu ustawień języków w kolumnie *Język*.

Kod aktualnie wybranego języka pracy może być odczytany za pośrednictwem zmiennej systemowej *ApplicationLanguageId*.

4.3Teksty programu

Jak już zostało wspomniane, Asix.Evo ma wbudowane dwa języki pracy: polski i angielski. Można jednak stosować inne języki, ale wymaga to jawnego określenia tekstów programu w tych językach. Do tego celu służy panel roboczy *Napisy programu,* otwierany za pośrednictwem węzła *Aplikacje wielojęzyczne/napisy aplikacji* panelu eksploratora aplikacji.



Rys. Panel roboczy deklaracji napisów programu.

W panelu wyświetlone są wszystkie teksty programu. W kolumnie *Identyfikator napisu* podany jest niezmienialny identyfikator tekstu, a w kolumnie *Definicje* właściwa treść napisu w języku wybranym w polu *Język podglądu*. Po wejściu w tryb edycji tekstu wyświetlane jest okienko służące do zdefiniowania tekstu we wszystkich używanych językach programu.

Nie ma obowiązku podawania tłumaczenia dla wszystkich napisów programu. Jeżeli napis nie zostanie przetłumaczony, to w razie konieczności jego użycia, zostanie zastosowana forma domyślna: polska lub angielska (w zależności od ustawień systemowych lub ustawień wybranych w programie *Wybór języka* pakietu Asix).

4.4 Teksty aplikacji

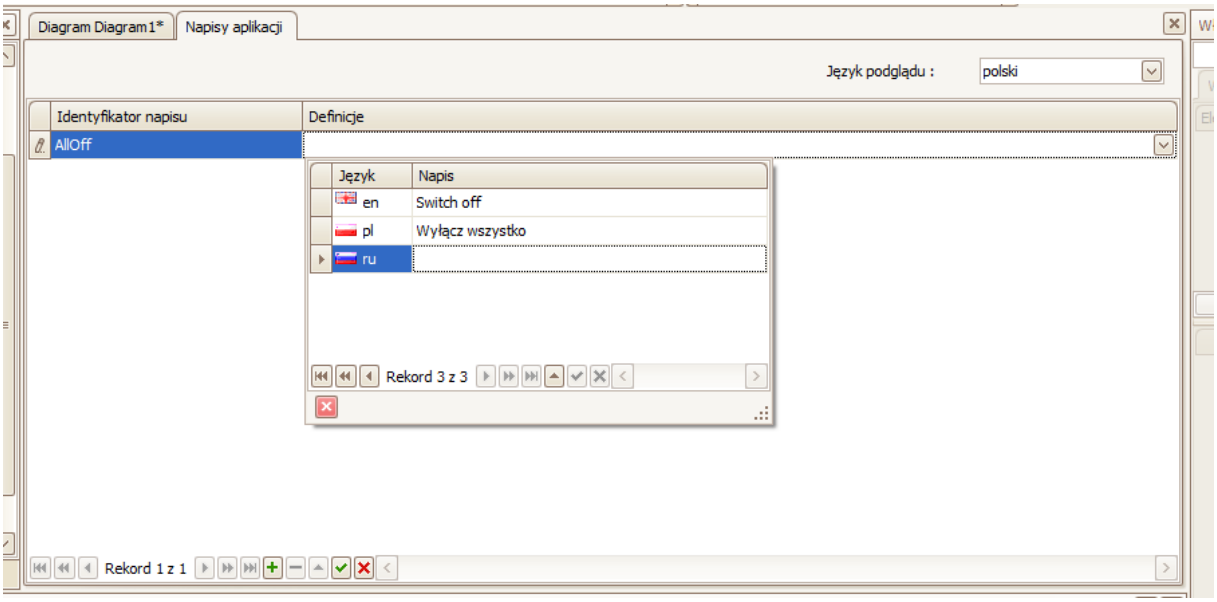
Teksty aplikacji to wszelkiego rodzaju napisy, stosowane w elementach projektowanej aplikacji, typu: teksty wyświetlane na diagramach, definicje alarmów, atrybuty zmiennych procesowych, komendy menu aplikacyjnych. Każdy z tych elementów definiowany jest w specyficzny dla siebie sposób, ale przeważnie teksty wprowadzane są w okienku napisu wielojęzycznego, pokazanym już w przykładzie definiowania tekstu programu. W przypadku tworzenia baz definicji alarmów i zmiennych na podstawie arkuszy kalkulacyjnych Excel poszczególne teksty podawane są w odpowiednio nazwanych osobnych kolumnach arkuszy. Szczegóły obsługi arkuszy znajdują się w podręczniku dotyczącym systemu alarmów oraz podręczniku programu Architekt.

W każdym przypadku w trakcie działania aplikacji całkowicie automatycznie wybierana jest wersja napisu zgodna z aktualnym językiem pracy aplikacji.

4.4.1 Napisy ogólnego przeznaczenia

W przypadku konieczności wyświetlenia na diagramie wielojęzycznego tekstu lub utworzenia wielojęzycznych menu aplikacyjnych należy użyć mechanizmu napisów aplikacji. Jest on w działaniu podobny do mechanizmu opisanych wcześniej napisów programów.

Panel roboczy *Napisy aplikacji* otwierany jest za pośrednictwem węzła *Aplikacje wielojęzyczne/Napisy aplikacji* panelu eksploratora aplikacji.



Rys. Panel roboczy deklaracji napisów aplikacji.

W przypadku napisu aplikacji w kolumnie *Identyfikator* należy podać dowolny unikalny, niezależny od języka identyfikator napisu. W kolumnie *Definicje* należy przy pomocy standardowego okienka napisu wielojęzycznego wprowadzić wszystkie wersje językowe napisu.

W celu użycia napisu aplikacji w miejscu jego zastosowania (np. we właściwości *Tekst* elementu menu lub właściwości *Tekst* wyłączenia obiektu *Przycisk*) należy odwołać się do napisu, stosując notację prefiksową typu $, np. *$AllOff*. Alternatywą jest użycie wyrażeniae z funkcją *Text*, np. =*Text(AllOff*). W obu przypadkach *AllOff* Parametrem funkcji Tekst jest identyfikatorem utworzonego napisu aplikacji.

Napis wielojęzyczny aplikacji może być także tworzony w trakcie edycji właściwości diagramów i obiektów. Użycie w definicji właściwości prefiksu $ z nieznanym dotychczas identyfikatorem, spowoduje otwarcie okienka pozwalającego na dodanie nowego napisu.

W skrypcie można odwołać się do napisu aplikacji, stosując metodę *ApplicationText* interfejsu *IApplication*.

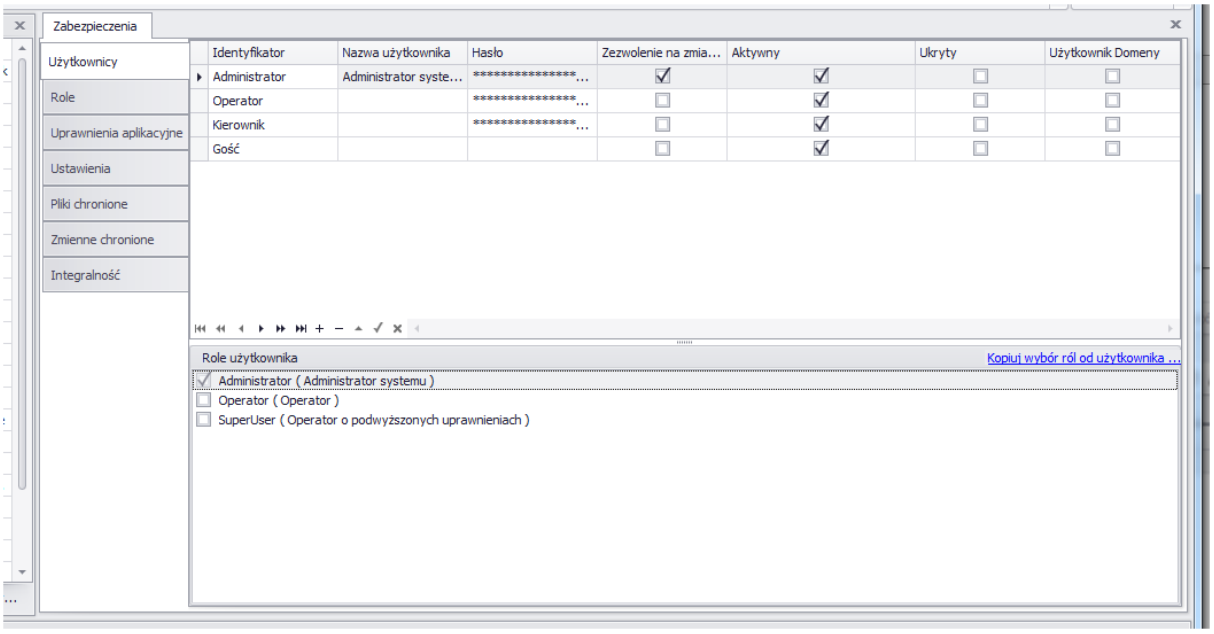
5 System uprawnień

System uprawnień aplikacji Asix.Evo oparty jest na systemie ról użytkowników. Każdy użytkownik aplikacji pełni w nim jedną lub kilka ról. Z faktu przynależności do konkretnej roli wynikają szczegółowe uprawnienia użytkownika. Inaczej mówiąc, użytkownik może wykonać chronioną operację, jeżeli przynajmniej w jednej z jego ról operacja ta jest dozwolona. Możliwy jest też taki schemat działania, w którym sam fakt przynależności do roli powoduje możliwość wykonania niektórych operacji.

Większość parametrów pracy systemu uprawnień jest ustawiana w panelu roboczym otwieranym poprzez węzeł *Zabezpieczenia* panelu eksploratora aplikacji. Część parametrów, związana przede wszystkim z opcjami automatycznego logowania i wylogowania, ustawiana jest panelu ustawień stanowisk.

5.1Definiowanie użytkowników

Lista aktualnie zdefiniowanych użytkowników wyświetlana jest na zakładce *Użytkownicy* panelu zabezpieczeń.



Rys. Panel ustawień systemu zabezpieczeń - definiowanie użytkowników aplikacji.

W momencie utworzenia aplikacji tworzony jest standardowo użytkownik o identyfikatorze *Administrator* należący do roli *Administrator.* Użytkownik ten posiada pełne uprawniania i nie może być usunięty. Przy pomocy przycisku *+* paska narzędziowego można dodać kolejnych użytkowników.

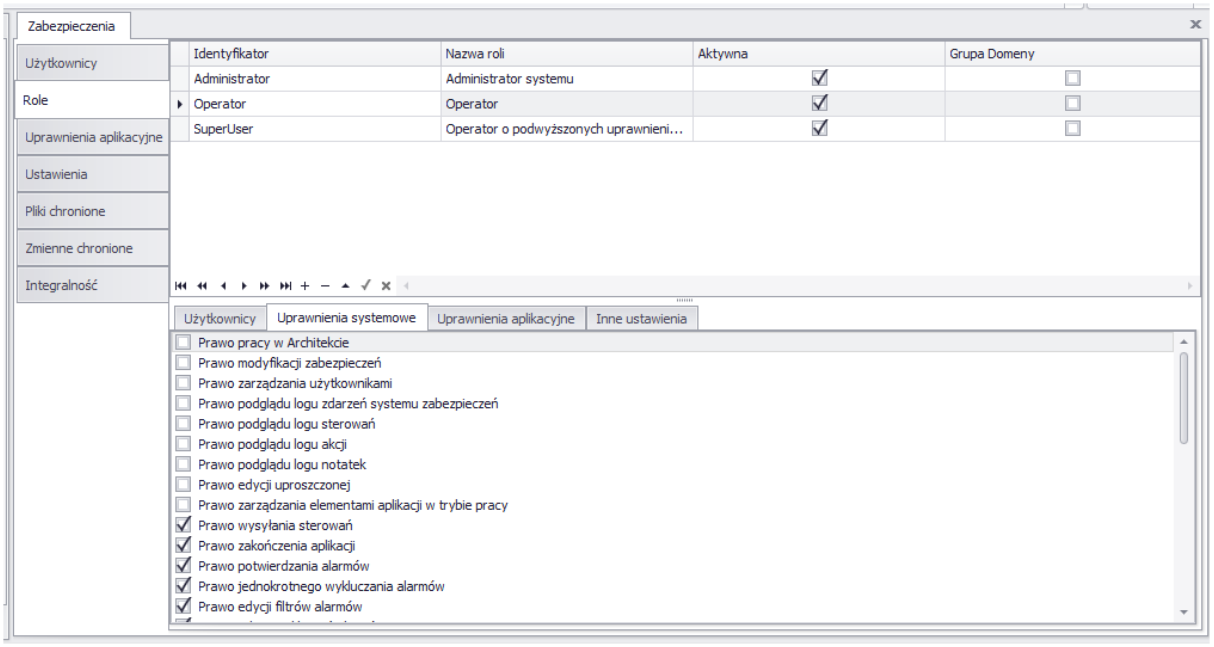
Użytkownik posiada tekstowy identyfikator, który używany jest w procesie logowania, oraz nazwę o znaczeniu wyłącznie informacyjnym. Należy również zdefiniować hasło logowania oraz zaznaczyć czy użytkownik będzie mógł samodzielnie zmienić hasło. Użytkownik *Administrator* ma początkowo hasło puste. W kolumnie *Aktywny* można wyłączyć możliwość logowania się użytkownika bez jego całkowitego usuwania. Należy zaznaczyć pole wyboru w kolumnie *Ukryty,* jeżeli identyfikator użytkownika nie ma się pokazywać na liście podpowiedzi w okienku logowania - zwiększa to poziom bezpieczeństwa. Oprócz powyższych parametrów należy wskazać role, jakie użytkownik pełni w aplikacji.

Kolumna *Użytkownik domeny* służy do określenia, czy użytkownik powinien być autoryzowany w domenie Windows. Zasady użycia domen zostały opisane w rozdziale *Integracja z domeną Windows*. W dolnej części panelu należy zaznaczyć, które role pełni wybrany użytkownik (a co za tym idzie, jakie będą jego możliwości). Przycisk *Kopiuj wybór ról od użytkownika* pozwala na ustawienie ról wybranego użytkownika w identyczny sposób, jak dla innego zdefiniowanego wcześniej użytkownika,

Lista użytkowników może być eksportowana i importowana do pliku arkusza Excela. Funkcje te dostępne są poprzez komendy menu kontekstowego. Funkcjonalność jest ograniczona wyłącznie do podstawowej definicji użytkowników - nie ma możliwości importu haseł i przynależności do grup.

5.2 Definiowanie ról

Zarządzanie rolami wykonywane jest w zakładce *Role*.



Rys. Panel ustawień systemu zabezpieczeń - zarządzanie rolami użytkowników aplikacji.

Standardowo tworzona jest rola *Administrator* posiadająca wszystkie szczegółowe uprawnienia. Roli tej nie można zmieniać. Przy pomocy przycisku + paska narzędziowego można dodać kolejne role.

Każda rola posiada tekstowy identyfikator, który używany jest np. w funkcji *HasRole* i akcji *ConfirmRole* oraz nazwę o znaczeniu wyłącznie informacyjnym. W kolumnie *Aktywny* można wyłączyć używanie roli bez jej całkowitego usunięcia.

Kolumna *Grupa domeny* służy do określenia, czy rola jest kojarzona z przynależnością użytkownika do identycznie nazwanej grupy w domenie Windows. Zasady użycia domen zostały opisane w rozdziale *Integracja z domeną Windows*.

W dolnej części panelu znajdują się zakładki służące do parametryzacji roli. Zakładka *Użytkownicy* pozwala wybrać użytkowników przydzielonych do roli. W zakładce *Uprawnienia systemowe* znajduje się lista wszystkich szczegółowych uprawnień standardowych - należy zaznaczyć te uprawnienia, które dana rola powinna posiadać. W zakładce *Uprawnienia aplikacyjne* znajduje się lista wszystkich uprawnień aplikacyjnych zdefiniowanych przez projektanta aplikacji - należy zaznaczyć te uprawnienia, które dana rola powinna posiadać.

Zakładka*Inne ustawienia* pozwala ustawić inne parametry pracy aplikacji zależne od roli, ale niezwiązane bezpośrednio z kontrolą uprawnień w aplikacji.

Oprócz określenia uprawnień w zakładce *Role*, zakres możliwości roli zależy też od parametryzacji praw dostępu do plików i praw sterowania wybranymi zmiennymi. Prawa te są ustawiane w osobnych zakładkach.

5.2.1 Uprawnienie „Prawo zarządzania elementami aplikacji w trybie pracy”

Uprawnienie określa, czy możliwe jest otwarcie okien diagnostycznych komunikacji i skryptów oraz okna podglądu zmiennych w trakcie pracy aplikacji (za pośrednictwem okna panelu kontrolnego).

5.2.2 Uprawnienie „Prawo wysyłania sterowań”

Uprawnienie określa, czy użytkownik może wykonywać operacje sterujące (zapis wartości do zmiennych procesowych). W przypadku sterowań wykonywanych poprzez kanał komunikacyjny typu *Network* możliwa jest dodatkowa blokada sterowań zakładana na poziomie drajwera – pozwala ona zablokować operacje sterujące na wybranych stanowiskach. Blokada sterowań w kanale sieciowym jest ważniejsza niż uprawnienia użytkowników.

5.2.3 Uprawnienie „Prawo potwierdzania alarmów”

Uprawnienie określa, czy możliwe jest potwierdzanie alarmów. Niezależnie od tego uprawnienia w konfiguracji systemu alarmów można zablokować możliwość wykonywania potwierdzeń na wybranych stanowiskach – ta blokada ma pierwszeństwo.

5.2.4 Uprawnienie „Brak blokowania pulpitu”

Uprawnienie określa, czy użytkownik pełniący rolę ma mieć pełny dostęp do wszystkich funkcji systemowych. Brak uprawnienia powoduje zablokowanie takich funkcji jak pasek zadań Windows, przyciski Windows, *Ctrl-Esc*, *Ctr-Alt-Del* itp. Blokady są jednak stosowane tylko wtedy, gdy aplikacja została uruchomiona z podaniem w linii komendy startowej programu opcji używania zabezpieczeń dynamicznych.

Uzupełnieniem zabezpieczeń dynamicznie kontrolowanych przez uprawnienie są dodatkowe zabezpieczenia ustawiane programem *Zabezpieczenia statyczne*, wchodzącym w skład pakietu.

5.3 Uprawnienia

W systemie uprawnień aplikacji Asix.evo występują dwa typy uprawnień. Uprawnienia systemowe są predefiniowane i służą do kontroli wybranych funkcji. Program automatycznie blokuje funkcjonalności zależne od tych uprawnień. Nie są wymagane żadne dodatkowe działania projektanta aplikacji. Uprawnienia aplikacyjne są swobodnie definiowane przez projektanta w zakładce *Uprawnienia aplikacyjne*. Definicja takiego uprawnienia składa się z identyfikatora oraz opisu. Projektant musi także zaimplementować użycie uprawnień aplikacyjnych. W tym celu w miejscach zależnych od wybranego uprawnienia może użyć funkcję *HasPrivilege* do skontrolowania czy użytkownik posiada uprawnienie (pełni rolę, która ma przydzielone to uprawnienie).

Nazwy uprawnień systemowych z reguły dobrze wyjaśniają ich znaczenie. W następnych podrozdziałach podano dodatkowe informacje dotyczące niektórych uprawnień.

5.3.1 Prawo zarządzania użytkownikami

Uprawnienie określa, czy zalogowany użytkownik może zarządzać listą zdefiniowanych użytkowników w trybie wykonywania aplikacji. Okno zarządzające jest otwierane akcją *SecurityManager*.

5.3.2 Prawo zarządzania elementami aplikacji w trybie pracy

Uprawnienie określa, czy możliwe jest otwarcie okien diagnostycznych komunikacji i skryptów oraz okna podglądu zmiennych w trakcie pracy aplikacji (za pośrednictwem okna panelu kontrolnego).

5.3.3 Prawo wysyłania sterowań

Uprawnienie określa, czy użytkownik może wykonywać operacje sterujące (zapis wartości do zmiennych procesowych). W przypadku sterowań wykonywanych poprzez kanał komunikacyjny typu *Network* możliwa jest dodatkowa blokada sterowań zakładana na poziomie drajwera - pozwala ona zablokować operacje sterujące na wybranych stanowiskach. Blokada sterowań w kanale sieciowym jest ważniejsza niż uprawnienia użytkowników.

5.3.4 Prawo potwierdzania alarmów

Uprawnienie określa, czy możliwe jest potwierdzanie alarmów. Niezależnie od tego uprawnienia w konfiguracji systemu alarmów można zablokować możliwość wykonywania potwierdzeń na wybranych stanowiskach - ta blokada ma pierwszeństwo.

5.3.5 Brak blokowania pulpitu

Uprawnienie określa, czy użytkownik pełniący rolę ma mieć pełny dostęp do wszystkich funkcji systemowych. Brak uprawnienia powoduje zablokowanie takich funkcji jak pasek zadań Windows, przyciski Windows, *Ctrl-Esc*, *Ctr-Alt-Del* itp. Blokady są jednak stosowane tylko wtedy, gdy aplikacja została uruchomiona z podaniem w linii komendy startowej programu opcji używania zabezpieczeń dynamicznych.

Uzupełnieniem zabezpieczeń dynamicznie kontrolowanych przez uprawnienie są dodatkowe zabezpieczenia ustawiane programem *Zabezpieczenia statyczne*, wchodzącym w skład pakietu.

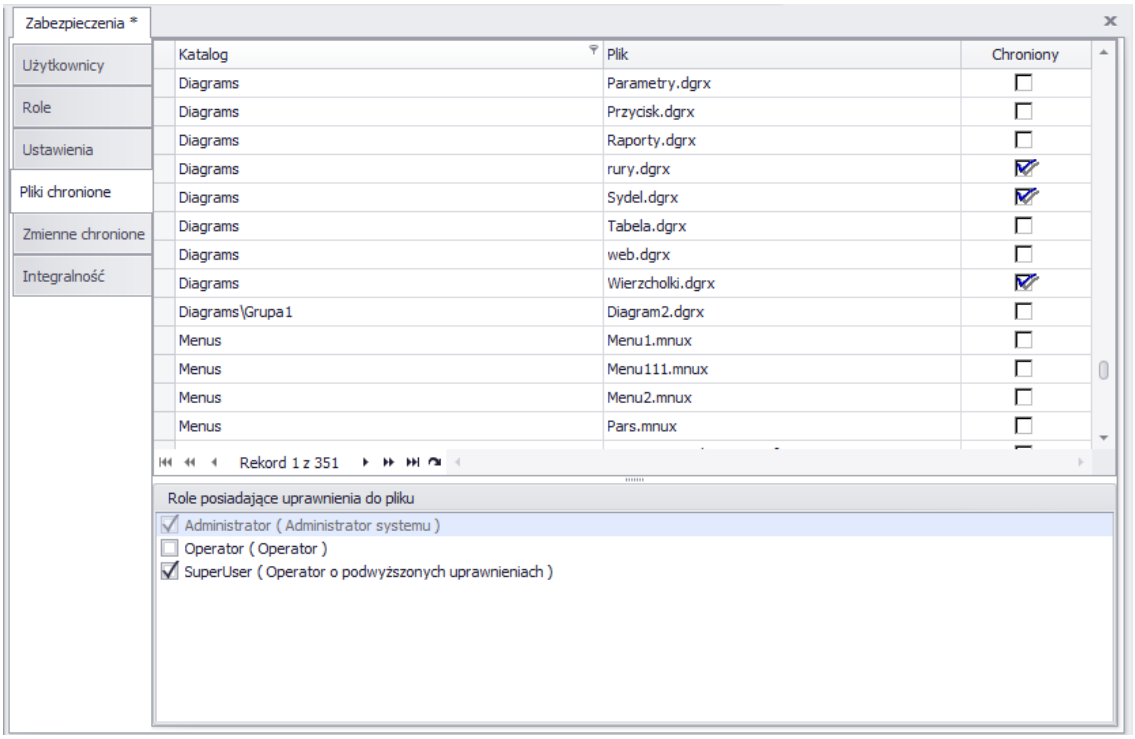
5.3.6 Brak automatycznego wylogowania

Posiadanie tego uprawnienia powoduje, że nawet wtedy, gdy włączone są mechanizmy automatycznego wylogowania, taki użytkownik nie zostanie wylogowany. Jest to jedyne uprawnienie, którego nie posiada wbudowany użytkownik Administrator.

5.4 Ochrona użycia plików

Oprócz uprawnień związanych z kontrolą wykonania konkretnych czynności możliwa jest także ochrona na poziomie dostępu do plików wchodzących w skład definicji aplikacji. Skutek takich zabezpieczeń zależy od rodzaju plików. W przypadku plików definicji diagramów można kontrolować próby wyświetlenia konkretnych diagramów. W przypadku plików definicji menu można kontrolować wyświetlenie menu kontekstowego. Ochrona pliku nie zawsze ma sens. Na przykład nie można chronić plików baz definicji zmiennych - zostaną one załadowane bez względu na uprawnienia.

Ochronę plików parametryzuje się w zakładce *Pliki chronione*.



Rys. Panel ustawień systemu zabezpieczeń - zarządzanie plikami chronionymi aplikacji.

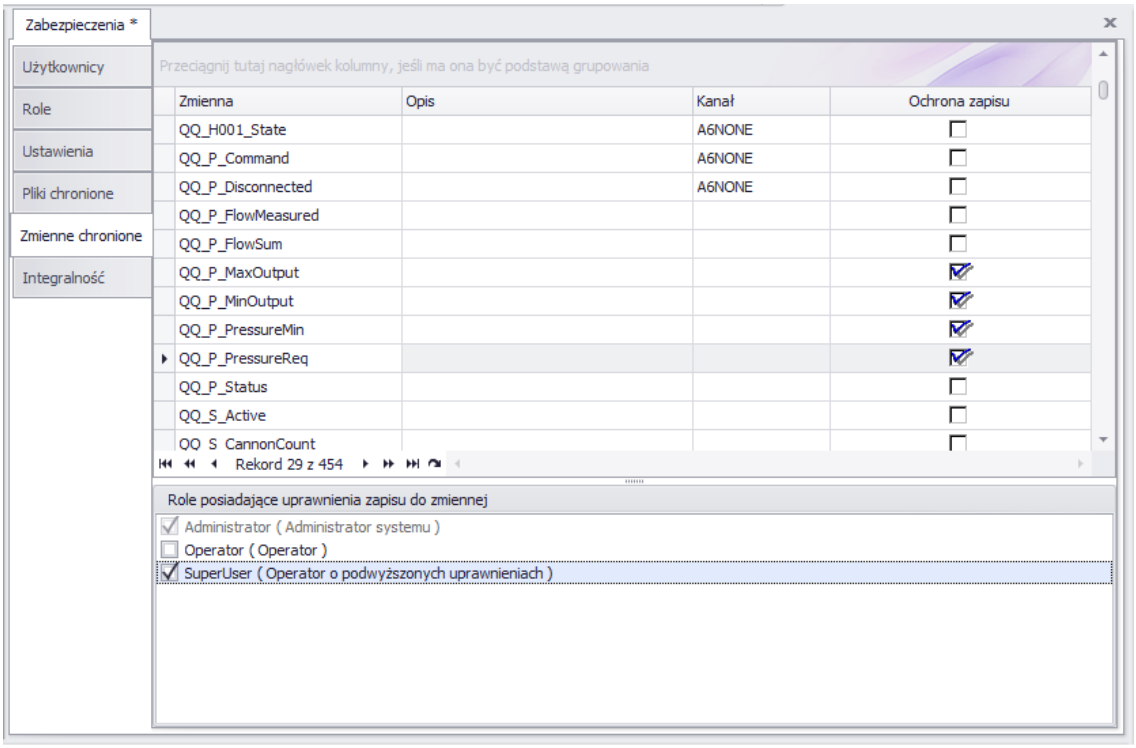
W górnej części panelu znajduje się lista wszystkich plików w katalogach definicyjnych aplikacji. Przy pomocy komend z menu kontekstowego linii nagłówka tabeli można ograniczyć liczbę wyświetlanych plików lub zmienić ich kolejność.

W kolumnie *Chroniona* zaznaczana jest opcja definiująca, czy dany plik podlega aktualnie ochronie.

W dolnej części panelu znajduje się lista zdefiniowanych ról. Należy zaznaczyć, które role mają prawo użycia wskazanego pliku. Plik jest chroniony, jeżeli przynajmniej jedna rola (oprócz roli *Administrator*) ma zaznaczone prawo użycia pliku. W przeciwnym przypadku plik nie podlega ochronie i jest dostępny dla wszystkich użytkowników.

5.5 Ochrona zapisu sterowań do zmiennych procesowych

Kolejnym elementem, który może być kontrolowany indywidualnie, jest prawo zapisu sterowań dla poszczególnych zmiennych procesowych. Typowo, prawo zapisu jest kontrolowane wspólnie dla wszystkich zmiennych poprzez uprawnienie *Prawo wysyłania sterowań*. Niezależnie od tego, w zakładce *Zmienne chronione* można ustawić uprawnienia indywidualne.



Rys. Panel ustawień systemu zabezpieczeń - zarządzanie zmiennymi chronionymi aplikacji.

W górnej części panelu znajduje się lista wszystkich zmiennych używanych w aplikacji. Przy pomocy komend z menu kontekstowego linii nagłówka tabeli można ograniczyć liczbę wyświetlanych zmiennych lub zmienić ich kolejność.

W kolumnie *Ochrona zapisu* zaznaczana jest opcja definiująca, czy zapis do danej zmiennej podlega aktualnie indywidualnej ochronie.

W dolnej części panelu znajduje się lista zdefiniowanych ról. Należy zaznaczyć, które role mają prawo sterowania wskazaną zmienną. Zmienna jest chroniona, jeżeli przynajmniej jedna rola (oprócz roli *Administrator*) ma zaznaczone prawo zapisu. W przeciwnym przypadku zmienna nie podlega indywidualnej ochronie i jest dozwolony zapis dla wszystkich użytkowników posiadających uprawnienie *Prawo wysyłania sterowań*.

5.6 Tryby pracy systemu zabezpieczeń

W zakładce *Ustawienia* panelu roboczego można wybrać tryb pracy systemu uprawnień.

5.6.1 Tryb standardowy

Tryb standardowy jest prostym systemem, w którym cała parametryzacja systemu przechowywana jest w pliku *security.xml*, zlokalizowanym w katalogu definicyjnym aplikacji. W związku z tym propagacja zmian wykonanych na jednym stanowisku wymaga skopiowania pliku na pozostałe stanowiska aplikacji. Można posłużyć się też mechanizmem synchronizacji definicji aplikacji.

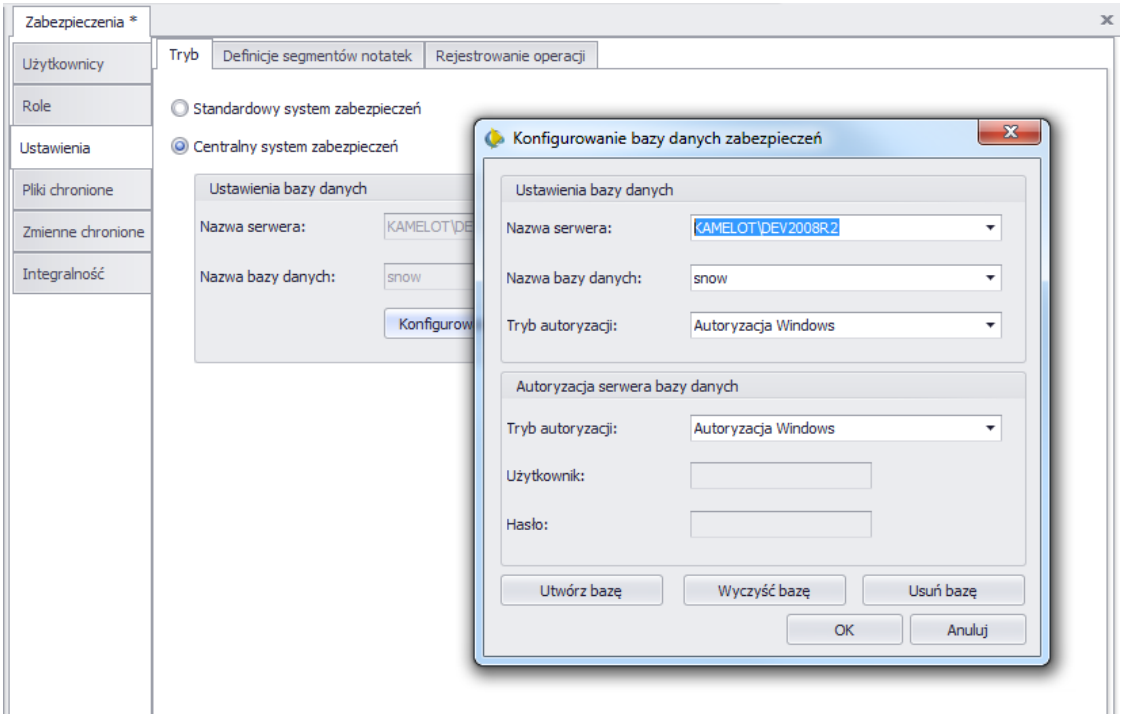
Tryb standardowy należy stosować w aplikacjach jednostanowiskowych lub wtedy, gdy parametry uprawnień są ustawione w fazie budowy aplikacji, a późniejsze zmiany występują okazjonalnie.

Pomimo tego, że parametry uprawnień są przechowywane w pliku typu xml, nie jest możliwa zmiana jego zawartości poza kontrolą programu Asix.Evo. W pliku zapisana jest suma kontrolna - każda nieautoryzowana zmiana zawartości zostanie wykryta.

5.6.2 Tryb centralny

W trybie centralnym wszystkie informacje dotyczące uprawnień przechowywane są w bazie danych serwera Microsoft SQL Server. Wszystkie zmiany uprawnień i użytkowników są natychmiast widziane na wszystkich stanowiskach. Dodatkowo tworzony jest centralny rejestr wszystkich zdarzeń związanych z pracą systemu uprawnień.

Zasada działania trybu centralnego jest zbliżona do działania modułu AsAudit w aplikacjach Asixa w starszych wersjach.

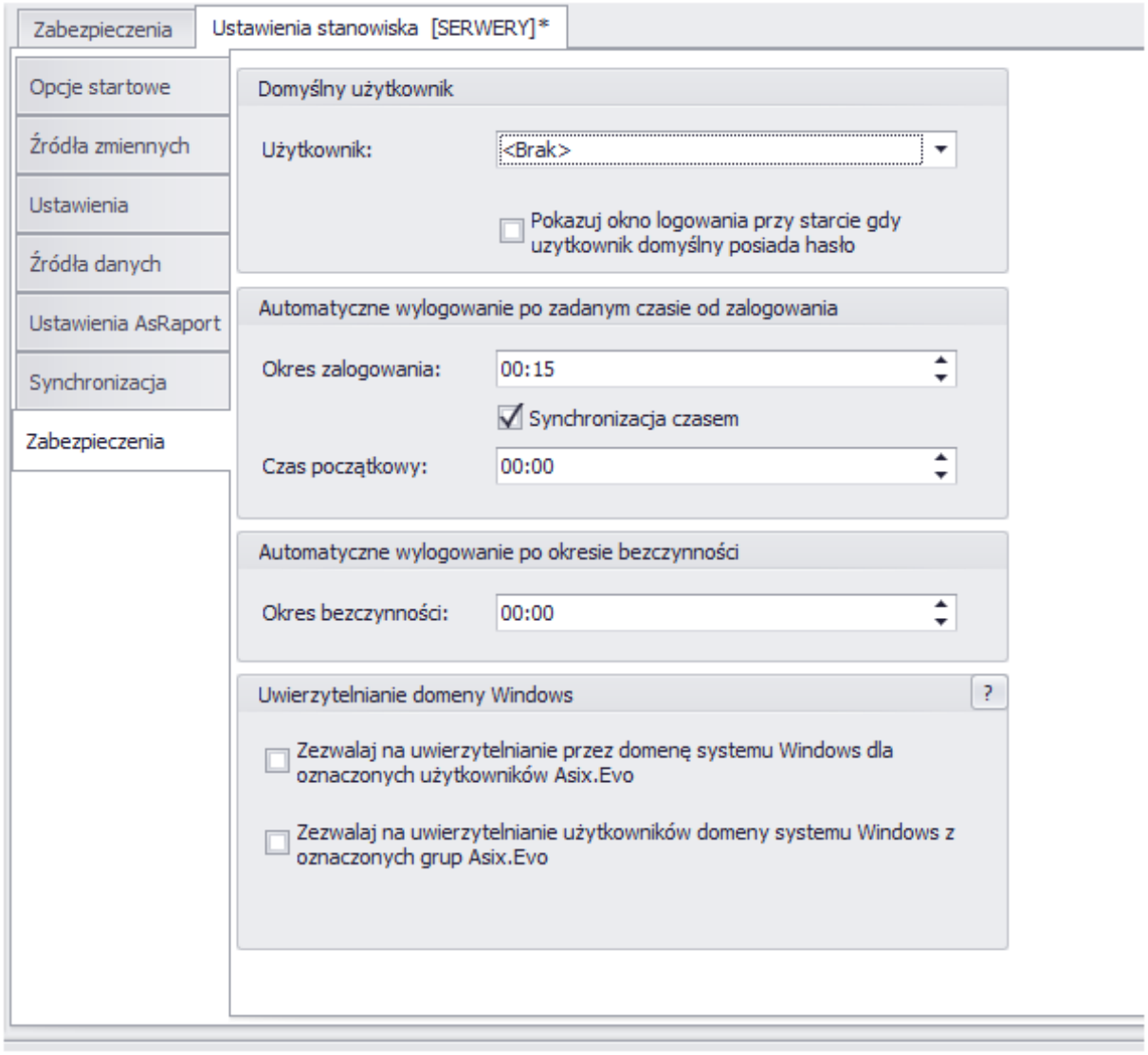


Rys. Panel parametryzacji trybu centralnego systemu zabezpieczeń.

Konfigurując tryb centralny należy określić nazwę serwera SQL, nazwę bazy uprawnień i używany tryb autoryzacji. Ponieważ zakładanie nowej bazy wymaga uprawnień administracyjnych serwera SQL, w ramce Autoryzacja serwera bazy danych należy jednorazowo określić sposób logowania do serwera SQL z uprawnieniami administracyjnymi.

5.7 Ustawienia specyficzne dla stanowiska

Większość parametrów systemu uprawnień jest definiowana globalnie dla całej aplikacji. Jest jednak kilka ustawień specyficznych dla poszczególnych stanowisk. Aby je zmienić, należy w panelu Stanowiska wybrać właściwe stanowisko lub obszar, za pośrednictwem panelu eksploratora otworzyć panel roboczy Ustawienia stanowisk i wybrać zakładkę Zabezpieczenia.



Rys. Panel ustawień systemu zabezpieczeń - panel definiowania indywidualnych ustawień zabezpieczeń dla indywidualnego stanowiska.

Aplikacja Asix.Evo może pracować bez zalogowanego użytkownika - dostępne są wszystkie niechronione funkcje, natomiast funkcje kontrolowane przez system uprawnień są zablokowane. Jeżeli chcemy, żeby poziom uprawnień dostępny dla wszystkich był wyższy, to należy określić tzw. użytkownika domyślnego. Należy zdefiniować rolę o podstawowym zestawie uprawnień, a następnie użytkownika pełniącego tą rolę. Takiego użytkownika podajemy jako użytkownika domyślnego. W efekcie, po starcie aplikacji zostanie on automatycznie zalogowany. Ponadto, w przypadku wykonania operacji wylogowania innego użytkownika, nastąpi powrót do użytkownika domyślnego.

Żeby użytkownik był automatycznie zalogowany, nie może on posiadać zdefiniowanego hasła. Jeżeli hasło posiada i dodatkowo wybrana jest opcja *Pokazuj okno logowania przy starcie, gdy użytkownik domyślny posiada hasło*, to przy starcie aplikacji zostanie wyświetlone okno logowania, pozwalające na wprowadzenie przez użytkownika hasła.

Inną możliwością jest wykorzystanie funkcji współpracy z domeną Windows i zadeklarowanie użytkownika domyślnego, jako <*Bieżący użytkownik domeny Windows*>. Stosowane to jest z reguły na stanowiskach terminalowych i przeglądarkowych. W takim przypadku program sprawdzi na starcie, czy aktualnie zalogowany użytkownik Windows jest zadeklarowany w ustawieniach aplikacji, jako użytkownik domenowy. Jeżeli tak, to zostanie on zalogowany automatycznie w aplikacji.

Można tez określić czas ważności operacji zalogowania - po upłynięciu podanego czasu od momentu zalogowania się użytkownika, zostanie on automatycznie wylogowany. Okres 00:00 oznacza logowanie bezterminowe.

Można też określić czas początkowy logowania, który pozwala na zdefiniowanie schematu logowania wymuszanego w konkretnych momentach czasu. Na przykład, okres zalogowania 08:00 i czas początkowy 6:00 oznacza, że wylogowanie użytkownika nastąpi o godzinie 6:00, 14:00 i 22:00.

Inną opcją automatycznego wylogowania użytkownika jest określenie maksymalnego dopuszczalnego czasu bezczynności. Jako bezczynność uważa się brak poruszania myszką i użycia klawiatury.

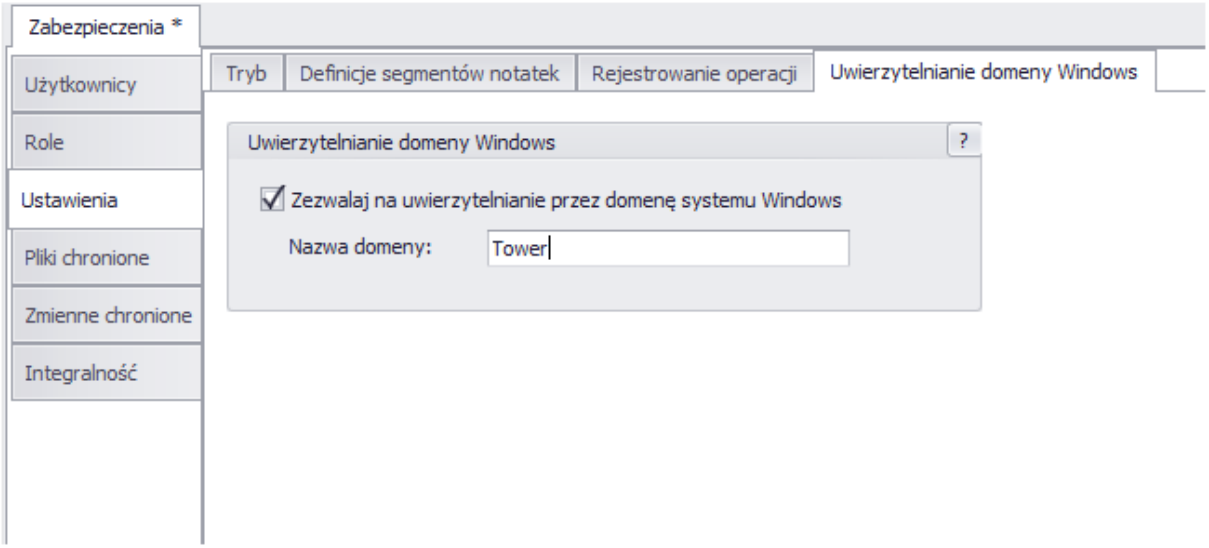
Pola wyboru w ramce *Uwierzytelnianie domeny Windows* dotyczą sposobu współpracy z domeną. Dla każdego stanowiska można zadeklarować, czy współpraca jest włączona.

5.8 Integracja z domeną Windows

System zabezpieczeń aplikacji Asix.Evo może wykorzystywać ustawienia zapisane w domenie Windows. Współpraca odbywa się na dwóch poziomach:

a) Autoryzacji logowania użytkowników.
b) Mapowania przynależności do grup na role systemu zabezpieczeń.

Pierwszym krokiem w konfiguracji współpracy z domeną jest podanie nazwy domeny w panelu konfiguracji zabezpieczeń w zakładce *Ustawienia/Uwierzytelnianie domeny Windows*.



Rys. Panel ustawień systemu zabezpieczeń - panel uwierzytelniania domeny Windows.

Kolejny krok to oznaczenie użytkowników i ról aplikacji, które mają być powiązane z domeną.

W przypadku użytkowników należy w jego definicji zaznaczyć pole wyboru w kolumnie Użytkownik Domeny. Dla tak oznaczonego użytkownika, w momencie jego próby zalogowania, prawidłowość autoryzacji jest weryfikowana z użyciem domeny Windows - nie ma znaczenia hasło użytkownika zdefiniowane w aplikacji Asix.Evo.

W przypadku ról, w kolumnie Grupa Domeny należy zaznaczyć, czy rola ma być kojarzona z przynależnością do grupy Windows. Takie połączenie spowoduje, że system dopuści zalogowanie każdego użytkownika domeny Windows (nawet wtedy, gdy taki użytkownik nie jest jawnie wymieniony na liście użytkowników aplikacji), jeżeli użytkownik został poprawnie zautoryzowany w domenie i należy do grupy o identycznej nazwie jak skojarzona rola. Takiemu użytkownikowi zostaną przedzielone uprawnienia wszystkich ról, dla których występuje zgodność nazw ról i grup.

Ostatnim krokiem jest zaznaczenie w ustawieniach zabezpieczeń poszczególnych stanowisk, które stanowiska będą współpracować z domeną i w jakim zakresie.

5.9 Użycie systemu zabezpieczeń w aplikacji

5.9.1 Logowanie użytkownika

Standardowo, logowanie użytkowników dostępne jest za pośrednictwem okna panelu kontrolnego. Projektant aplikacji może jednak dodać mechanizmy logowania bezpośrednio do diagramów synoptycznych. Pierwszy z tych mechanizmów to obiekt klasy *Pole autoryzacji*, zapewniający pełną obsługę logowania bezpośrednio na diagramie. Alternatywą jest użycie akcji operatorskiej *Login*, która wyświetla okienko logowania użytkownika. Uzupełnieniem jest akcja *Logout,* służąca do wylogowania użytkownika oraz akcja *ChangePassword* otwierająca okienko zmiany hasła.

Na diagramie można wyświetlić również informacje dotyczące aktualnego stanu systemu uprawnień. Pozwalają na to predefiniowane zmienne procesowe:

- *CurrentUser* - identyfikator aktualnie zalogowanego użytkownika;
- *CurrentUserName* - nazwa aktualnie zalogowanego użytkownika*;*
- *UserRoles* - identyfikatory wszystkich ról pełnionych przez aktualnie zalogowanego użytkownika;
- *UserRemainingTime* - czas pozostały do automatycznego wylogowania użytkownika;
- *UserInactivityTime* - czas, który upłynął od ostatniej aktywności użytkownika.
- *SecurityVersion* - numer wersji parametryzacji systemu zabezpieczeń. Numer systemu zabezpieczeń jest inkrementowany przy każdym zapisie zmodyfikowanych ustawień - nie dotyczy to tylko zapisów ograniczonych do zmiany haseł.
- *IntegrityVersion* - numer wersji aplikacji. Związany jest z kontrolą integralności aplikacji i pozwala oznaczyć wersję „zatwierdzoną". Numer aplikacji jest zapisywany w logu komunikatów przy każdym starcie. Numer jest inkrementowany przy każdym wywołaniu funkcji przeliczania sum kontrolnych w panelu Integralność.

5.9.2 Weryfikacja uprawnień

W zakresie kontroli standardowych uprawnień działanie programu jest całkowicie automatyczne. Możliwe jest jednak rozszerzenie możliwości aplikacji o własne funkcje. Pozwalają na to akcja operatorska *ConfirmRole* i funkcja *HasRole*.

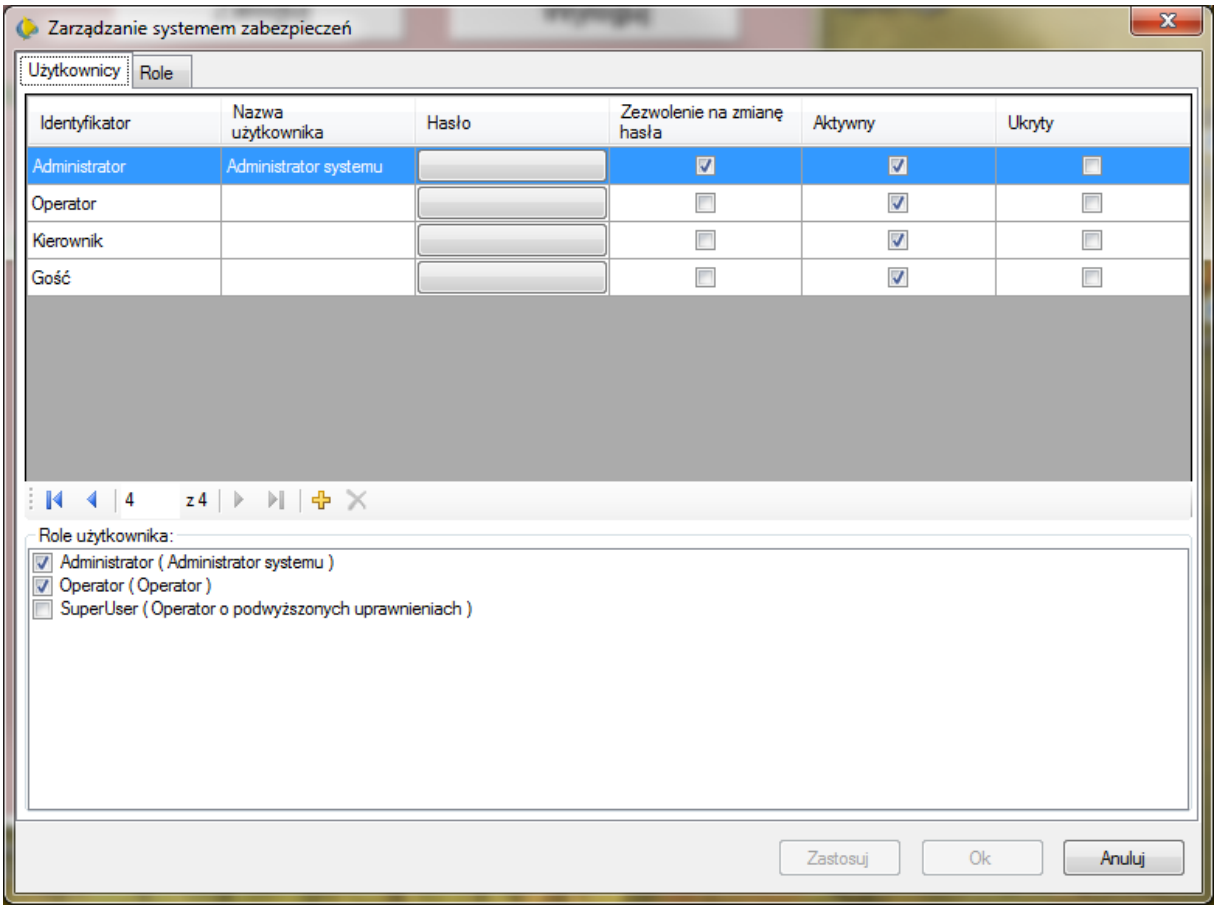
Akcja *ConfirmRole* pozwala na dodatkową autoryzację użytkownika poza systemem standardowego logowania. Może być używana w schemacie działania, w którym wykonanie jakiejś operacji wymaga autoryzacji w chwili jej wykonania, lub w przypadku, gdy potrzebne jest dodatkowe potwierdzenie przez użytkownika innego niż zalogowany.

Funkcja *HasRole* pozwala na uzależnienie wyglądu lub działania obiektów na diagramie w zależności od tego, czy użytkownik pełni podaną w wywołaniu rolę. Możliwe jest też sprawdzanie posiadania konkretnych uprawnień przy pomocy funkcji *HasPrivilege*. Dotyczy to zarówno uprawnień systemowych i aplikacyjnych, ale z reguły stosowane jest dla aplikacyjnych.

Przykłady zastosowań znajdują się w podręczniku *Techniki budowy diagramów* (Asix.Evo_Techniki_budowy_diagramów.PDF/CHM), w rozdziale *Kontrola uprawnień.*

5.9.3 Zmiany w parametrach uprawnień w trybie wykonania aplikacji

Typowo system uprawnień jest konfigurowany w trakcie projektowania aplikacji, podczas pracy w trybie edycyjnym. Możliwe jest jednak pełne zarządzanie uprawnieniami (definiowanie użytkowników i ról, zmiana uprawnień i haseł) podczas pracy w trybie wykonania. Pozwala na to akcja *SecurityManager*, która otwiera poniższe okno.



Rys. Okno parametryzowania uprawnień w trybie wykonania aplikacji.

Należy pamiętać o tym, że zmiany wykonane na jednym stanowisku są natychmiast widoczne na innych stanowiskach tylko wtedy, gdy system uprawnień pracuje w trybie centralnym. W trybie standardowym zmiany będą widoczne dopiero po skopiowaniu lub zsynchronizowaniu pliku uprawnień.

5.9.4 Przeglądanie logu zdarzeń logowania

W trybie standardowym zdarzenia logowania użytkowników rejestrowane są wyłącznie w zwykłym logu komunikatów. Dotyczą one tylko aktualnego stanowiska. W trybie centralnym tworzony jest osobny log zdarzeń dotyczących systemu uprawnień. Rejestrowane są w nim zdarzenia ze wszystkich stanowisk.

Okno przeglądania logu zdarzeń systemu uprawnień można otworzyć przy pomocy akcji operatorskiej *SecurityBrowser($Messages)*.

6 Kontrola pracy aplikacji (AsAudit)

Niektóre aplikacje (np. pracujące w dziedzinie farmaceutycznej) wymagają podwyższonego poziomu kontrolowania jej działania. System Asix.Evo oferuje w tym zakresie poniższe mechanizmy:

a) Rejestracja operacji sterujących wykonywanych przez operatorów systemu.

b) Rejestracja działań operatorów.

c) Kontrola integralności definicji aplikacji.

Powyższe mechanizmy w aplikacjach Asix typu klasycznego wspierane były przez opcjonalny moduł AsAudit. W przypadku aplikacji Asix.Evo mechanizmy kontroli są częścią systemu zabezpieczeń.

Mechanizmy kontroli wymagają spełnienia dwóch warunków wstępnych:

• System zabezpieczeń aplikacji musi pracować w trybie centralnym - zbierane dane zapisywane są w bazie danych używanej także w systemie zabezpieczeń.

• Wymagane jest posiadanie licencji systemu Asix z rozszerzeniem AsAudit.

6.1Rejestracja operacji sterujących

W ramach aplikacji Asix.Evo możliwa jest rejestracja operacji sterujących (zapisu zmiennych procesowych).

Wybór zmiennych, dla których wszystkie wykonane sterowania mają zostać zarejestrowane odbywa się w bazie definicji zmiennych. Baza musi posiadać atrybut *Rejestracja sterowania.* Wszystkie zmienne, dla których wartość tego atrybutu jest różna od zera, podlegają rejestracji. Nie są wymagana żadne inne ustawienia.

Dla każdego zarejestrowanego sterowania zapisane są informacje: kto, kiedy i na którym stanowisku wykonał operację. Zapisana jest też wartość zmiennej przed operacją sterującą oraz nowa wartość zmiennej.

Okno przeglądania logu operacji sterujących można otworzyć przy pomocy akcji operatorskiej *SecurityBrowser($Controls)*.

W przypadku operacji sterujących o charakterze bitowym możliwe jest wyświetlenie interpretacji tekstowej wykonanego sterowania. Porównywana jest aktualna i nowa wartość zmiennej, a dla zmienionych bitów tworzony jest opis tekstowy operacji. Opisy tekstowe tworzone są na podstawie zawartości bazy definicji zmiennych. Wykorzystywane są do tego celu atrybuty *Nazwy stanów*, *Zestaw stanów* i *Wartość stanu.* Algorytm wyliczania opisu sterowania dla pojedynczego bitu zmiennej jest następujący:

- Pobierana jest wartość atrybutu *Nazwy stanów* dla zmiennej sterowanej.
- W bazie definicji zmiennych wyszukiwane są wszystkie zmienne, których atrybut *Zestaw stanów* jest identyczny z wartością atrybutu *Nazwy stanów* pobranego w punkcie pierwszym. Typowo są to zmienne nieaktywne służące tylko do definiowania nazw stanów.
- Wśród zmiennych wybranych w poprzednim punkcie wyszukiwana jest ta, której wartość atrybutu *Wartość_stanu* jest identyczna z numerem bitu.
- Atrybut *Nazwy stanów* zmiennej wybranej w poprzednim punkcie powinien stanowić ciąg postaci *0=stan0;1=stan1.* W zależności od nowej wartości bitu zmiennej sterowanej wybierany jest odpowiedni opis.

Przykład

Nazwa	*Zmienna nieaktywna*	*Nazwy stanów*	*Zestaw stanów*	*Wartość stanu*
ENGINE_1	0	ENGINE_CONTROL		
EC_BIT0	1	0=OFF;1=ON	ENGINE_CONTROL	0
EC_BIT1	1	0=FWD;1=BWD	ENGINE_CONTROL	1
EC_BIT1	1	0=SLOW;1=FAST	ENGINE_CONTROL	2

Jeżeli zmienna *Engine1* zostanie zmieniona z wartości 2 na 3 (zmiana bitu numer 0 na wartość 1), to w opisie sterowania zostanie użyty napis *ON.*

6.2 Rejestracja działań operatora

Mechanizm rejestracji działań operatora pozwala na zarejestrowanie w bazie systemu zabezpieczeń części operacji wykonywanych przez operatora aplikacji. Możliwe jest rejestrowanie:

a) operacji otwarcia i zamknięcia diagramów,

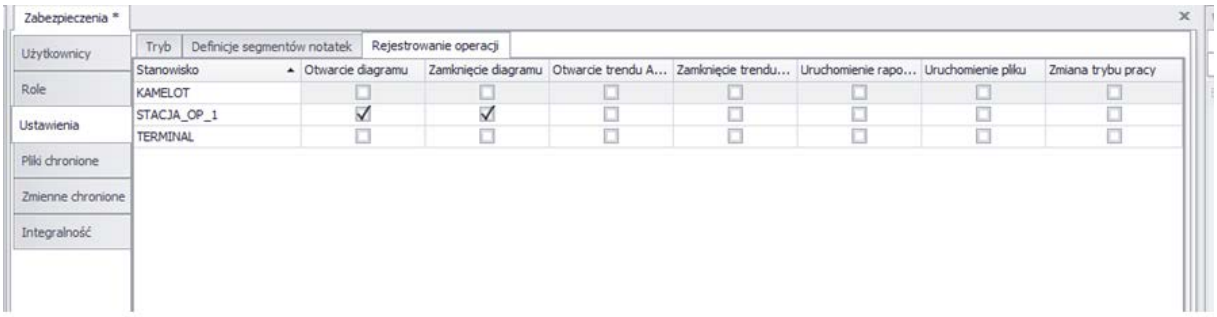
b) operacji otwarcia i zamknięcia trendów w programie AsTrend,

c) wyświetlenia raportów AsRaport,

d) uruchomienia zewnętrznych programów (przy pomocy akcji operatorskiej *Run*)

e) zmiany trybu pracy aplikacji (przejście w tryb edycji aplikacji)

Parametry rejestracji działań operatora ustawiane są w panelu ustawień systemu zabezpieczeń - w podzakładce *Rejestrowanie operacji* zakładki *Ustawienia*.



Rys. Panel ustawień systemu zabezpieczeń - parametryzowanie rejestracji działań operatora.

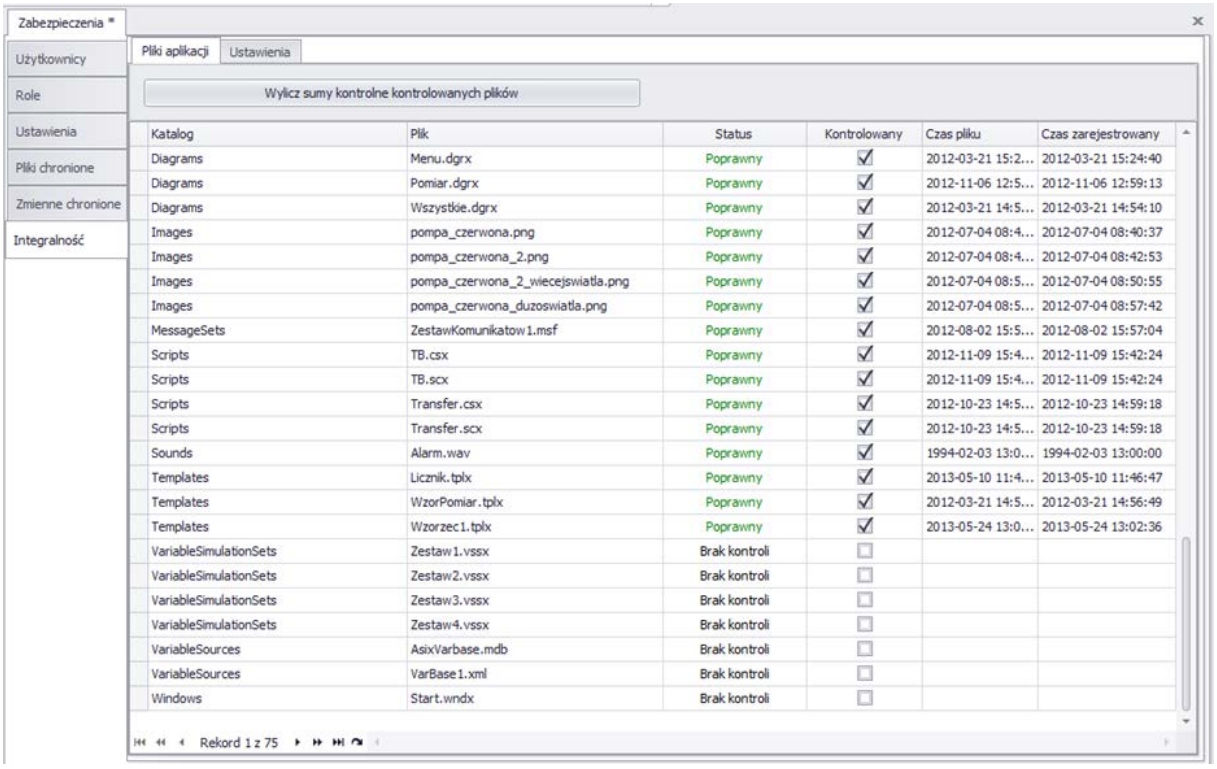
W zakładce wyświetlona jest lista wszystkich stanowisk aplikacji. Dla każdego stanowiska ustawia się rodzaje operacji, które na tych stanowiskach mają być rejestrowane.

Okno przeglądania logu działań operatora w trybie wykonywania aplikacji można otworzyć przy pomocy akcji operatorskiej *SecurityBrowser($Actions)*.

6.3 Kontrola integralności aplikacji

Kontrola integralności aplikacji polega na rejestrowaniu wersji wybranych plików definicyjnych aplikacji i sprawdzaniu, czy ich zawartość nie uległa zmianie w nieautoryzowany sposób. Nieautoryzowane pliki nie są używane w aplikacji. W szczególności mogą zablokować jej uruchomienie. Mechanizm ma zapewnić, że wykonana i oddana do użytku aplikacja pozostaje niezmienna.

Ochronę plików parametryzuje się w zakładce *Integralność* panelu systemu zabezpieczeń.



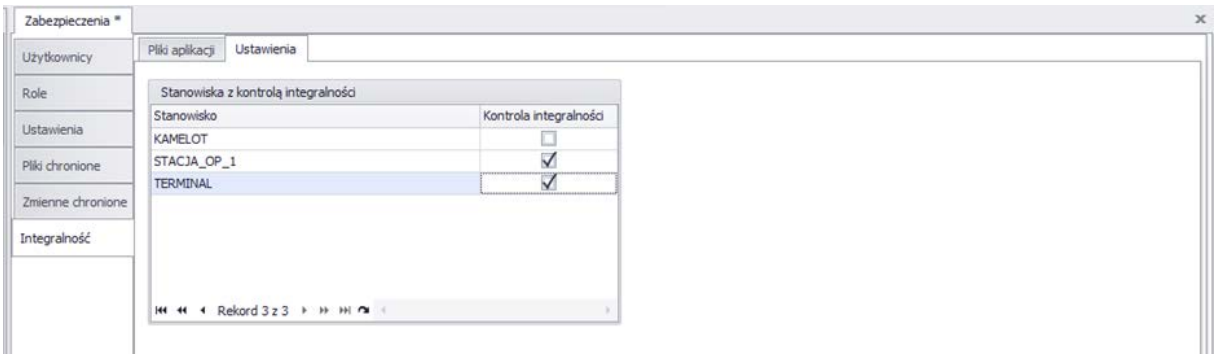
Rys. Panel ustawień systemu zabezpieczeń - parametryzowanie integralności - pliki aplikacji.

Główną część zakładki *Pliki aplikacji* zajmuje lista wszystkich plików znajdujących się w katalogach definicyjnych aplikacji. Przy pomocy komend z menu kontekstowego linii nagłówka tabeli można ograniczyć liczbę wyświetlanych plików lub zmienić ich kolejność.

W kolumnie *Status* podany jest aktualny stan pliku: czy podlega kontroli i czy jego zawartość jest poprawna. Kolumna Kontrolowany służy do włączania kontroli pliku.

Przycisk Wylicz sumy kontrolne kontrolowanych plików służy do przeliczenia sum kontrolnych, które zostaną zarejestrowane w bazie danych i będą później służyły do weryfikacji plików. Przycisk należy użyć wtedy, gdy aplikacji została przygotowana i sprawdzona i w tej właśnie wersji ma być oddana do użytkowania.

Dodatkowe parametry kontroli integralności są ustawiane w zakładce *Ustawienia*.



Rys. Panel ustawień systemu zabezpieczeń - parametryzowanie integralności - ustawienia.

Należy tutaj określić, na których stanowiskach kontrola integralności aplikacji ma być wykonywana. Typowo, jedno ze stanowisk pozostawia się bez kontroli - w celu przygotowywania na nim nowych wersji aplikacji (dostęp do zmienionych plików nie będzie blokowany).

7 Notatki operatora

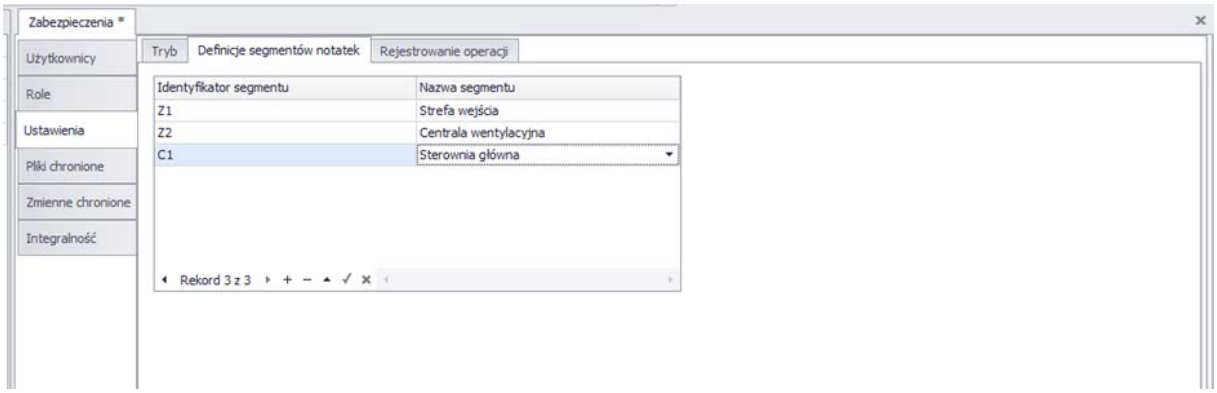
System notatek operatora służy do rejestrowania swobodnie tworzonych notatek wpisywanych przez użytkowników aplikacji. Każda notatka składa się z:

a) Tekstu - dowolna treść wpisana przez użytkownika.

b) Lokalizacji - każda notatka jest łączona z konkretną części aplikacji (instalacji) tzw. segmentem.

c) Flagi aktywności - służy do oznaczania nowych notatek. W szczególności istnieje mechanizm powiadamiania, że w danych segmencie aplikacji istnieje aktywna notatka.

Wstępna faza parametryzacji systemu notatek polega na zdefiniowaniu segmentów. Wykonywane jest to w panelu ustawień systemu zabezpieczeń - w podzakładce *Definicje segmentów notatek* zakładki *Ustawienia*.



Rys. Panel ustawień systemu zabezpieczeń - definicje segmentów notatek.

Dla każdego segmentu należy podać jego identyfikator (używany w akcjach operatorskich i funkcjach) oraz nazwę opisową.

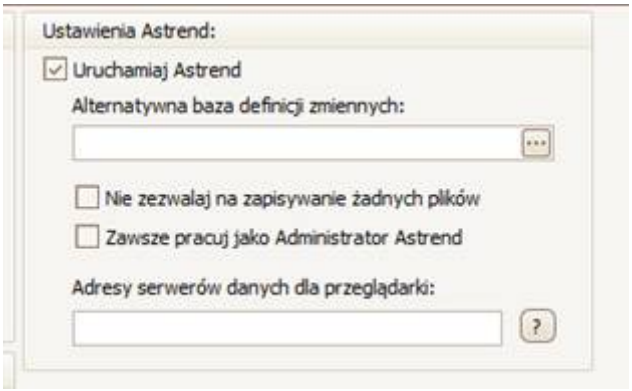
Użycie notatek w aplikacji wymaga zastosowania poniższych akcji operatorskich i funkcji:
a) Akcja *SecurityBrowser($Notes)*
Akcja służy do wyświetlenia zarejestrowanych notatek (wszystkich lub z wybranych segmentów). Dodatkowo okno przeglądarki pozwala na zmianę stanu aktywności notatek.

b) Akcja *AddNote*
Akcja służy do otwarcia okienka służącego do wprowadzenia nowej notatki.
c) Funkcja *IsActiveNote*

Funkcja zwraca informacje, czy we wskazanym segmencie istnieją aktywne notatki.

8 Współpraca z programem AsTrend

Aplikacje Asix.Evo mogą wykorzystywać program AsTrend do wyświetlania przebiegów danych archiwalnych zmiennych procesowych wraz z ich zaawansowaną analizą oraz do obsługi raportów programu AsTrend. Współpraca pomiędzy aplikacją Asix.Evo a programem AsTrend opiera się na wykorzystaniu akcji operatorskich AstrendDisplay, AstrendPrint i AstrendPrintToPdf. Wymagana jest jednak wcześniejsza konfiguracja użycia programu AsTrend. Konfiguracja wykonywana jest niezależnie dla każdego stanowiska aplikacji. Za pośrednictwem węzła Ustawienia stanowisk panelu eksploratora aplikacji należy otworzyć panel roboczy ustawień stanowiska i wybrać zakładkę Źródła danych. Za konfigurację programu AsTrend odpowiedzialny jest pokazany poniżej fragment panelu.



Rys. Parametry konfiguracyjne programu AsTrend.

Jeżeli akcje rodziny *Astrend*... będą używane, to należy zaznaczyć opcję Uruchamiaj Astrend. W przypadku aplikacji okienkowej spowoduje to dodatkowo uruchomienie programu AsTrend w momencie startu aplikacji - pozwoli to później na szybsze wykonanie akcji operatorskich.

Standardowo, do programu AsTrend przekazywana jest lista wszystkich baz definicji zmiennych typu MDB, zdefiniowanych w aplikacji. Jeżeli jest to działanie niepożądane, to można podać inną nazwę bazy definicji zmiennych w polu *Alternatywna baza definicji zmiennych*.

Zaznaczenie opcji *Nie zezwalaj na zapisywanie żadnych plików* spowoduje, że użytkownik aplikacji nie będzie mógł zmienić plików konfiguracyjnych trendów (plików trnx). Zaznaczenie opcji *Zawsze pracuj jako Administrator Astrend* powoduje, że użytkownik automatycznie uzyska wszystkie uprawnienia do funkcji programu AsTrend. Obie opcje mają znaczenie tylko w aplikacji pracującej w trybie okienkowym.

8.1Praca w przeglądarkowej wersji aplikacji

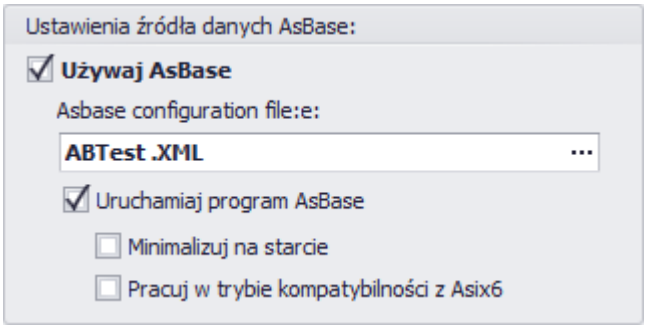
W przypadku pracy aplikacji Asix.Evo w trybie przeglądarkowym współpraca z AsTrend oparta jest również na przeglądarkowej wersji tego programu. Oznacza to, że w trakcie instalacji pakietu Asix należy zainstalować go łącznie ze składnikami internetowymi. Spowoduje to założenie w ramach serwera IIS katalogu programu AsTrend. Instalacja musi być wykonana na tym samym komputerze, na którym opublikowana jest aplikacja Asix.Evo.

Program AsTrend zazwyczaj pobiera dane historyczne bezpośrednio z serwera danych procesowych uruchomionego w ramach aplikacji Asix.Evo. Może też korzystać z danych udostępnianych przez serwery aplikacji klasycznych Asixa. Jeżeli parametr *Adresy serwera danych dla przeglądarki* nie zostanie jawnie zdefiniowany, to AsTrend będzie pobierał dane z komputera, na którym aplikacja Asix.Evo została opublikowana. Można jednak podać inne źródła danych historycznych - w parametrze *Adresy serwera danych dla przeglądarki* należy wprowadzić (rozdzielone przecinkiem) systemowe nazwy (lub adresy IP) wszystkich komputerów, które udostępniają dane historyczne.

9 Współpraca z programem AsBase

W przypadku aplikacji Asix.Evo współpraca z programem AsBase odbywa się w sposób bardzo podobny do tego, jaki był używany w aplikacjach klasycznych. Pierwszym krokiem w przygotowaniu aplikacji jest standardowa parametryzacja aplikacji AsBase - należy zdefiniować zestawy rejestracji i/lub grupy receptur oraz zestawy zmiennych służące do wymiany danych pomiędzy AsBase'em, sterownikami PLC oraz aplikacją Asix.Evo.

Jeżeli aplikacja Asix.Evo będzie współpracowała z programem AsBase, to wymagane jest włączenie odpowiedniego sprzęgu. Konfiguracja wykonywana jest niezależnie dla każdego stanowiska aplikacji. Za pośrednictwem węzła Ustawienia stanowisk panelu eksploratora aplikacji należy otworzyć panel roboczy ustawień stanowiska i wybrać zakładkę Źródła danych. Za konfigurację sprzęgu z programem AsBase odpowiedzialny jest pokazy poniżej fragment panelu.



Rys. Ustawienie połączenia z programem AsBase.

Włączenie opcji *Używaj Asbase* umożliwia współpracę z programem AsBase. W polu *Plik konfiguracyjny Asbase* należy podać nazwę pliku startowego xml aplikacji AsBase. Plik ten jest tworzony w trakcie konfigurowania aplikacji przy pomocy programu AsBase. Zawiera on przede wszystkim informacje o lokalizacji bazy danych AsBase. Opcja *Uruchamiaj program Asbase* powoduje, uruchomienie programu AsBase w momencie startu aplikacji. W przypadku aplikacji w technologii Evo współpraca z programem AsBase nie wymaga jego lokalnego uruchomienia. Opcja *pracuj w trybie kompatybilności z Asix6* określa sposób działania sprzęgu. Różnice pomiędzy poszczególnymi trybami pracy zostaną opisane w dalszej części rozdziału.

Dalsza współpraca pomiędzy Asix.Evo i AsBase odbywa się przy pomocy dwóch mechanizmów:

a) za pośrednictwem zmiennych użytych w zestawach zmiennych AsBase'a oraz rodziną akcji *Asbase*....

b) poprzez wyświetlenie danych w obiekcie *Tabela Asbase*.

9.1 Akcje operatorskie

Akcje operatorskie służące do współpracy z programem AsBase podzielone są na dwie grupy: akcje połączeniowe oraz akcje bezpołączeniowe. Akcje bezpołączeniowe pełnią rolę pomocniczą. Są to akcje AsbaseLoad, AsbaseLoadByValue, AsbaseSave, AsbaseShow i AsbasePrint. Mogą być one wykonane w dowolnym momencie.

Akcje trybu połączeniowego pozwalają na przeglądanie danych pochodzących z bazy danych AsBase'a przy pomocy obiektów diagramów synoptycznych. Schemat działania jest następujący:

- **Otwarcie połączenia**

Otwarcie połączenia wykonywane jest przy pomocy akcji *AsbaseOpen*. W akcji tej należy określić podzbiór danych, które będą pokazywane w ramach połączenia oraz zestaw zmiennych, za pomocą których dane będą przekazywane do aplikacji Asix.Evo. W akcji należy podać też identyfikator połączenia, którym będą się posługiwały pozostałe akcje trybu połączeniowego.

Akcję *AsbaseOpen* z reguły stosuje się w obsłudze zdarzenia *Otwarcie diagramu* diagramu, na którym dane AsBase'a będą pokazywane. W aplikacjach klasycznych za otwarcie połączenia odpowiedzialny był obiekt klasy *AsBase*.

- **Wizualizacja danych na diagramie**

Prezentacja danych wykonywana jest przy pomocy standardowych obiektów diagramowych.

Wykorzystywane są do tego zmienne procesowe wchodzące w skład zestawu zmiennych określonego w akcji *AsbaseOpen*. Zmienne te przyjmują wartości zgodne z wartościami pól zestawu danych AsBase'a dla aktywnego rekordu.

- **Nawigacja w ramach danych połączenia**

Akcje *AsbaseAdd, AsbaseDelete, AsbaseUpdate, AsbaseNavigate* pozwalają na wykonywanie operacji na danych połączenia. Każda z tych funkcji posiada parametr określający identyfikator połączenia. Operacje wykonywane przez te akcje z reguły odnoszą się do aktywnego rekordu połączenia. Szczególnie istotna jest akcja *AsbaseNavigate*, która zasadniczo służy do zmiany aktywnego rekordu, ale może też zmienić filtrowanie danych(określone początkowo w momencie tworzenia połączenia).

- **Zamknięcie połączenia**

Połączenie zamykane jest przy pomocy akcji *AsbaseClose.* Akcja z reguły używana jest w obsłudze zdarzenia *Zamknięcie diagramu* diagramu, na którym dane Asbase'a były pokazywane.

9.2 Obiekt Tabela Asbase

Przy pomocy obiektów klasy Tabela Asbase możliwe jest pokazanie danych pochodzących z bazy AsBase w postaci typowej tabeli. Istnieje możliwość wyboru pokazywanych kolumn oraz określenia kryterium selekcji rekordów. Obiekt Tabela Asbase jest alternatywą dla tabel wyświetlanych w programie AsBase.

Możliwa jest praca w trybie mieszanym, w którym wybrane kolumny (pola) zestawów rejestracji lub receptur są wyświetlane w tabeli, a inne dane wyświetlane są za pośrednictwem zmiennych z zestawów zmiennych (podobnie jak dla akcji operatorskich). Wartości zmiennych są synchronizowane ze zmianami aktywnego wiersza tabeli.

9.3 Tryby pracy sprzęgu z programem AsBase

Sprzęg z programem AsBase może pracować w dwóch trybach: w trybie własnym oraz w trybie kompatybilności Asix6. W trybie własnym (wewnętrznym) akcje operatorskie rodziny *Asbase...* wykonywane są samodzielnie, bez komunikacji z programem AsBase. W trybie kompatybilności, realizacja akcji przekazywana jest do uruchomionego lokalnie programu AsBase - działanie jest wtedy identyczne, jak w przypadku aplikacji wykonanych w technologii klasycznej.

Pomiędzy oboma trybami występują drobne różnice, które należy uwzględnić w trakcie projektowania aplikacji.

Uruchamianie lokalnej instancji programu AsBase

W trybie kompatybilności lokalne uruchomienie programu AsBase jest obowiązkowe. W przypadku trybu własnego jest to czynność opcjonalna, ale brak uruchomienia programu pociąga za sobą poniższe skutki:

a) Akcje *AsbasePrint* i *AsbaseShow* nie są wykonywane.

b) Nie są obsługiwane wewnętrzne funkcje programu AsBase, takie jak: warunki rejestracji i automatyczne receptury. Żeby te funkcje były obsługiwane, przynajmniej na jednym ze stanowisk aplikacji powinien zostać uruchomiony program AsBase.

Definiowanie zmiennych używanych w zestawach zmiennych

Zmienne w zestawach zmiennych używanych do przeglądania danych pobieranych z bazy danych AsBase'a powinny być deklarowane w lokalnych kanałach wirtualnych (typu *None*). W przypadku trybu kompatybilności możliwy jest tylko jeden wariant. Zmienne procesowe używane do wymiany danych powinny być zdefiniowane w kanale typu *None* modułu Asmen. Wynika to z faktu, że AsBase komunikuje się bezpośrednio z serwerem danych procesowych, uruchomionym w ramach aplikacji Asix.Evo. Podsumowując, kanał zmiennych wymiany danych jest zdefiniowany jako *None* w pliku konfiguracyjnym serwera danych (używając programu Architekt), a w aplikacji Asix.Evo jest deklarowany jako typ *Asix6*.

W przypadku trybu własnego, powyższy sposób definicji zmiennych jest dopuszczalny, ale lepszym rozwiązaniem jest użycia kanału wirtualnego zdefiniowanego tylko w aplikacji Asix.Evo. W tym celu należy zmienne wymiany umieścić w kanale komunikacyjnym, dla którego nie zadeklarowano jawnie żadnego drajwera lub użyto drajwer typu *None*. Można też w definicji zmiennych nie podać w ogóle nazwy kanału, co spowoduje umieszczenie zmiennych w domyślnym kanale wirtualnym tworzonym automatycznie.

W przypadku trybu własnego korzystne może być zdefiniowanie zmiennych wymiany w osobnej bazie definicji zmiennych (typu XML lub MDB), której nazwa nie jest przekazywana do serwera danych procesowych w ramach jego pliku konfiguracyjnego.

Konwersje wartości zmiennych dla pól typu *Data* i *Czas* Trwania

Pola zestawów rejestracji i grup receptur typu *Data* i *Czas Trwania* przechowywane są w bazie danych jako kolumny typu *datetime* i *int* (ilość milisekund). W trybie kompatybilności wartości tych pól wpisywane są do zmiennych wymiany w postaci liczb całkowitych (odpowiednio ilość sekund od 1.1.1970 i ilość milisekund) lub w postaci tekstowej, jeżeli został określonych parametr format definicji pól. Projektant musi zapewnić zgodność typu zmiennych z typem pól. Jeżeli wartości przekazywane są w postaci liczbowej, to można stosować funkcje *FromAsix6Date* i *FromAsix6Time* w celu konwersji wartości pól na typ *DateTime* i *TimeSpan*.

W przypadku trybu własnego wykonywana jest próba automatycznej konwersji wartości odczytywanej z bazy danych zgodnie z typem zmiennych wymiany. Przy czym konwersja dotyczy zarówno odczytu danych jak i funkcji powodujących zapis pól w bazie danych. Z reguły najkorzystniejsze jest użycie zmiennych typu *DateTime* i *TimeSpan*, odpowiednio dla pól typu *Data* i *Czas Trwania*. Dla zmiennych typu tekstowego konwersja wykorzystuje parametr *format* definicji pól lub wykonywane jest formatowanie standardowe. Dla innych typów zmiennych data przekazywana jest jako ilość sekund od 1.1.1970, a czas trwania jako ilość milisekund.

Różnice w wykonaniu akcji operatorskich

W przypadku trybu własnego akcja *AsbaseOpen* może być ponownie wykonana dla tego samego połączenia (bez wcześniejszego zamknięcia) w celu:

a) Odświeżenia danych pobranych z bazy danych w ramach połączenia w momencie jego pierwotnego otwarcia.

b) Zmiany wszystkich parametrów połączenia łącznie ze źródłami danych.

c) Zmiany wyrażenia filtrującego pobierane dane (w tym celu można też użyć akcję *AsbaseNavigate* w wariancie *$Filter*).

W trybie kompatybilności można stosować tylko akcję *AsbaseNavigate*, lub stosować metodę zamknięcia i ponownego otwarcia połączenia.

Akcja *AsbaseLoadByValue* jest dostępna tylko w przypadku pracy w trybie własnym.

9.4 Filtr połączenia

Jednym z parametrów akcji *AsbaseOpen* i *AsbseNavigate* (w wariancie *$Filter)* jest filtr pozwalający wybrać dane, które mają być użyte w połączeniu. Filtry można też stosować w obiekcie Tabela Asbase.

Składnia filtru jest zgodna z klauzulą WHERE języka SQL z dodatkowymi, opcjonalnymi rozszerzeniami pozwalającymi na wstawianie do tej klauzuli wartości zmiennych procesowych oraz ograniczeń czasowych. Słowa WHERE nie podaje się w definicji filtru. Istnieje też możliwość dodania do składni filtru klauzuli ORDER BY w celu uporządkowania danych.

Z reguły w definicji filtru używa się odwołań do pól zdefiniowanych w zestawie archiwizacji lub grupie receptur. Aby wartość pola zestawu rejestracji, receptury lub historii ładowań mogła być użyta jako element filtru, to identyfikator tego pola należy poprzedzić przedrostkiem "V_". W przypadku archiwum zestawu rejestracji, z każdą wartością pola może być związany status wartości oraz stempel czasowy tej wartości. Aby skorzystać z tych elementów, należy identyfikator pola zestawu rejestracji poprzedzić przedrostkiem "S_" lub "T_". W przypadku receptur przeliczanych można się odwołać do pola wartości procentowej w rekordzie załadowania stosując przedrostek „VP_”.

AsbaseOpen(conn1, $Archive, Farby, FarbyDgr, null, null, ”V_Akt=1”)

W powyższym przykładzie zastosowany filtr powoduje pobranie tylko tych danych z zestawu rejestracji *Farby,* dla których wartość pola o identyfikatorze *Akt* jest równa 1.

W przypadku połączeń typu relacyjnego, w których używane jest również podrzędne źródło danych, odwołania do pól źródeł danych należy dodatkowo uzupełnić o prefiks m dla pól źródła głównego (*master*) lub o prefiks d dla źródła podrzędnego (*detail*), np. *„m.V_Akt=1”*.

Oprócz odwołań do wartość pól definiowanych przez projektanta aplikacji, można również używać nazw stałych pól tabel AsBase, tj. takich pól, które zawsze są obecne w tabeli. Pola stałe opisuje poniższa tabela.

Tab. Pola stałe tabel AsBase'a.

Nazwa	Znaczenie
LOCALTIME	Lokalny czas zapisu rekordu lub załadowania receptury
UTCTIME	Czas UTC zapisu rekordu lub załadowania receptury
STATUS	Status operacji załadowania receptury
SOURCE	Źródło pochodzenia rekordu rejestracji
NAME	Nazwa receptury
LOADBY	Określa kto załadował recepturę

TARGET	Miejsce załadowania receptury
CREATEDBY	Autor receptury
BATCH	Wartość wsadu dla załadowania receptury przeliczanej.

9.4.1 Wstawki

W treści filtra można też użyć wstawek pozwalających uwzględnić aktualne wartości zmiennych procesowych oraz dynamicznie ograniczyć zakres czasu.

Wstawkę do filtru ujmuje się w nawiasy klamrowe.

Wstawka rozszerzająca pozwalająca zastosować wartość zmiennej procesowej ma postać:

{V:nazwa_zmiennej}

Wartość zmiennej wstawiana jest w miejsce wstawki.

Wstawka definiująca elementy filtru czasowego ma postać:

{T:kod[:identyfikator_pola]}

Kod określa zakres czasu. Możliwe kody to:

t - dzisiaj

y - wczoraj

sy - od wczoraj

tw - ten tydzień

lw - poprzedni tydzień

slw - od poprzedniego tygodnia

tm - ten miesiąc

lm - poprzedni miesiąc

slm - od poprzedniego miesiąca

ty - ten rok

ly - poprzedni rok

sly - od poprzedniego roku

Opcjonalny *identyfikator_pola* określa pole tabeli, do którego ma być zastosowany filtr czasowy. Jeśli ten element jest pominięty, to przyjmuje się, że filtr dotyczy czasu zapisu rekordu w tabeli. Jeśli identyfikator pola jest podany, to pole powinno być typu Data. Użycie tej wstawki powoduje, że cały tekst wstawki zostanie zastąpiony wyrażeniem (zgodnym ze składnią języka SQL) ograniczającym zbiór zapisów tabeli do takich, których wartość(czas) jest zawarty w przedziale czasowym określonym przez parametr kod.

Filtr zawierający wstawki nie jest modyfikowany w czasie trwania połączenia. Na przykład, jeśli podano filtr czasowy typu "dzisiaj" i połączenie jest otwarte drugą dobę, to pokazywane będą rekordy z poprzedniego dnia. Dopiero ponowne otwarcie połączenia lub ponowne zastosowanie tego samego filtru w akcji operatorskiej spowoduje aktualizację danych.

Wstawki nie mogą występować wewnątrz elementów filtra zawartych w nawiasach prostokątnych [i] oraz ujętych w cudzysłowy ' i ". Aby treść wstawki mogła być zawarta pomiędzy tymi ogranicznikami, to te ograniczniki muszą być zawarte w nawiasach klamrowych.

{'}{V:nazwa_zmiennej}{'}

Powyższy fragment filtru zostanie zastąpiony wartością zmiennej ujętą w cudzysłowy pojedyncze.

Mechanizm wstawek dotyczy tylko treści akcji operatorskich przy pracy w trybie kompatybilności Asix6. Jeżeli tryb kompatybilności nie jest używany, to filtry dynamicznie zmienne należy tworzyć przy pomocy wyrażeń. Metoda wyrażeń jest bardziej uniwersalna - można stosować ją w każdym trybie. W miejscu wstawek {V:} należy używać funkcję *Variable*, a zamiast wstawek czasowych funkcję *OPCTime*.

”V_Akt={V:Z1}”

”V_Akt=”+Variable(Z1)

Przykład pokazuje identycznie działający filtr w wersji ze wstawkami oraz budowany przy pomocy wyrażenia. Wybrane będą te rekordy, dla których wartość pola *Akt* będzie równa wartości zmiennej *Z1* (w chwili wykonania akcji).

9.5 AsBase w aplikacjach przeglądarkowych

W odróżnieniu od aplikacji klasycznych, aplikacji wykonane w technologii Evo mogą współpracować z programem AsBase podczas pracy w trybie przeglądarkowym. Wymagane jest, aby stanowiska typu *Terminal WWW* miały **wyłączony** tryb kompatybilności Asix6. Nie można też, na tych stanowiskach uruchamiać lokalnie programu AsBase.

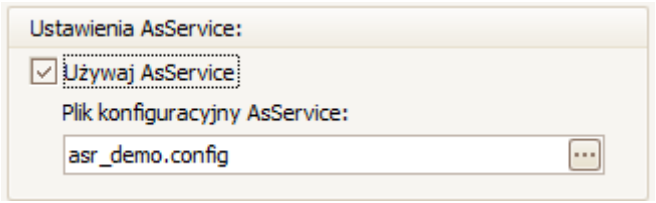
9.6 Uprawnienia użytkowników

Jeżeli program AsBase został uruchomiony w ramach połączonej aplikacji Asix.Evo, to używane są ustawienia zabezpieczeń takie, jakie zostały zdefiniowane w aplikacji Asix.Evo. Nie mają wtedy znaczenia ustawienia wybrane bezpośrednio w aplikacji AsBase. Niemniej jednak, jeżeli taka aplikacja AsBase zostanie uruchomiona samodzielnie, to ustawienia będą miały znaczenie. W typowej konfiguracji w aplikacji AsBase jest zdefiniowany pojedynczy operator administracyjny - chroni to bazę AsBase przed niepowołanym dostępem. Pozostali użytkownicy mają prawa określone w konfiguracji systemu zabezpieczeń Asix.Evo.

10 Współpraca z programem AsService

Aplikacje Asix.Evo mogą współpracować z programem AsService służącym do kontroli czasu pracy urządzeń oraz ilości ich załączeń. Możliwe jest wyświetlanie stanu liczników programu AsService oraz sprzęgniecie zdarzeń przekroczenia limitów wartości liczników z systemem alarmów programu Asix.Evo.

Współpraca z programem *AsService* wymaga przede wszystkim określenia lokalizacji bazy danych AsService. Wykonywane jest to w panelu roboczym *Ustawienia stanowiska,* w zakładce *Źródła danych*.



Rys. Deklaracja współpracy z programem AsService.

W polu *Plik konfiguracyjny AsService* należy podać nazwę pliku konfiguracyjnego utworzonego programem *AsService*, w którym zapisana jest lokalizacja i sposób dostępu do bazy danych.

Program Asix.Evo wykorzystuje bezpośredni odczyt zawartości bazy danych - nie jest konieczne lokalne uruchomienie programu *AsService.*

10.1 Wizualizacja stanu liczników

Wizualizacja stanu liczników wykonywana jest przy użyciu standardowych obiektów wizualizacyjnych. Wartości liczników pobierane są przy pomocy funkcji *AsserviceRead*. Funkcja posiada dwa parametry.

Pierwszy określa identyfikator licznika, którego dane będą odczytane, a drugi nazwę pola danych, które należy odczytać. Istnieje możliwość odczytu danych dotyczących bieżącego stanu licznika, wartości związanych z ostatnim resetem licznika oraz danych definicyjnych. Pełna lista pól znajduje się w opisie funkcji *AsserviceRead*.

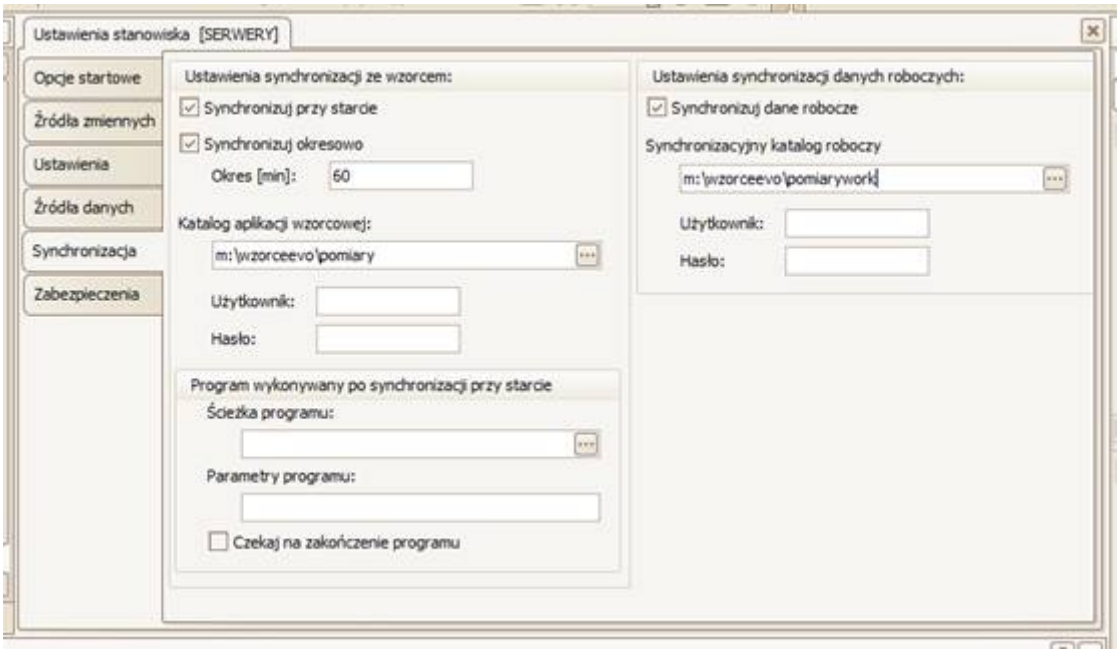
W przypadku korzystania z edytora wyrażeń można skorzystać z komend menu, pozwalających na interaktywny wybór identyfikatora licznika oraz nazwy pola.

11 Synchronizacja plików aplikacji

W przypadku instalacji wielostanowiskowych istotnym problemem staje się utrzymywanie identycznych wersji plików aplikacji na wszystkich stanowiskach. Dotyczy to głównie fazy, w której aplikacji jest jeszcze rozwijana i często dochodzi do zmian w plikach definicyjnych. Problem jest mniejszy w przypadku aplikacji w trybie przeglądarkowym - zmieniona wersja aplikacji jest jednokrotnie publikowana na serwerze, a stamtąd, automatycznie jest propagowana na wszystkie terminale WWW. W przypadku aplikacji okienkowych z pomocą przychodzi system synchronizacji aplikacji wbudowany w Asix.Evo.

Koncepcja działania systemu synchronizacji opiera się na zdefiniowaniu katalogu wzorcowego położonego w lokalizacji sieciowej dostępnej przez wszystkie stanowiska aplikacji. Zmienione pliki są wgrywane do katalogu wzorcowego, a stamtąd automatycznie transferowane na wszystkie stanowiska aplikacji.

Parametry modułu synchronizacji ustawiane są w panelu roboczym ustawień stanowiska w zakładce Synchronizacja.



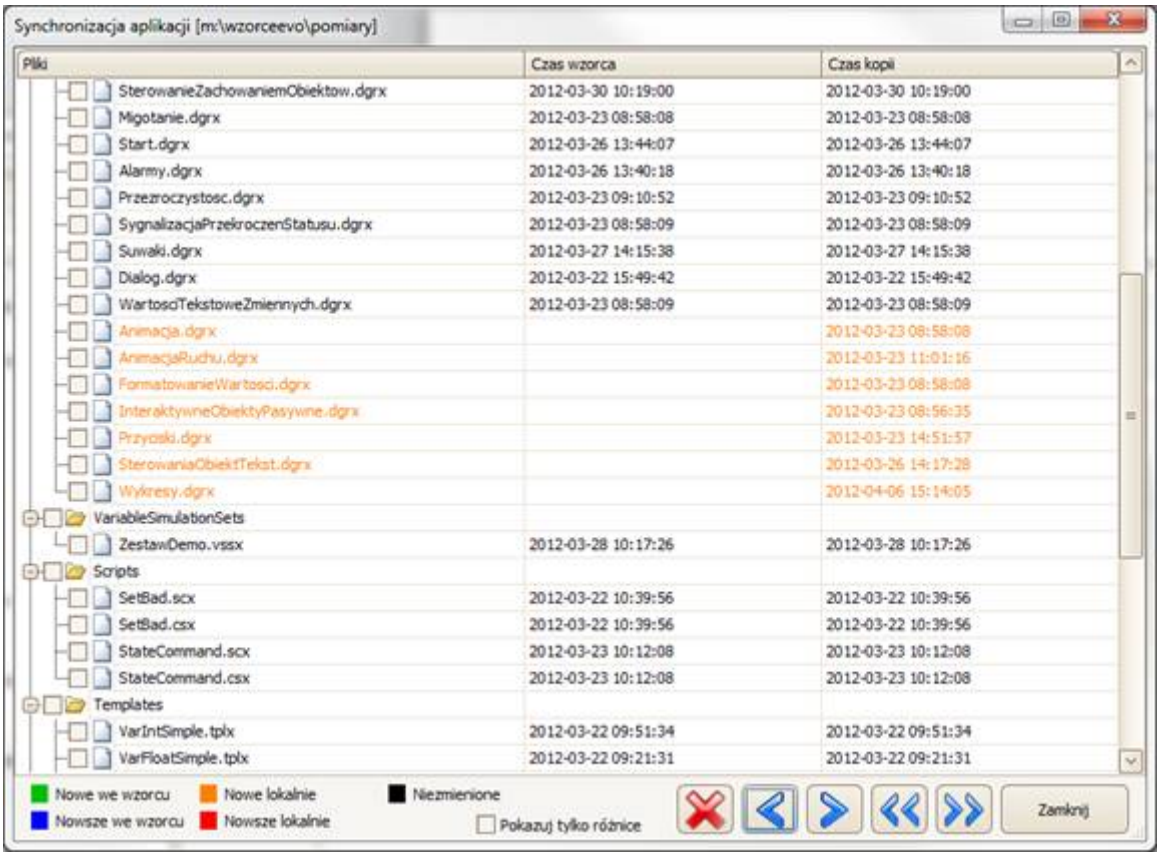
Rys. Parametry modułu synchronizacji.

11.1 Synchronizacja katalogu definicyjnego aplikacji

Katalog definicyjny aplikacji może być synchronizowany jednorazowo w momencie startu aplikacji lub/i okresowo z zadanym okresem czasu. W polu *Katalog aplikacji wzorcowej* należy podać lokalizację wzorca. Można odwołać się do katalogu na dysku lokalnie zamapowanym lub użyć nazwy UNC. Jeżeli do dostępu do katalogu wzorca wymagana jest specjalna autoryzacja systemowa, to w polach *Użytkownik* i *Hasło* należy podać identyfikator i hasło użytkownika, który posiada odpowiednie uprawnienia.

W przypadku synchronizacji wykonywanej na starcie aplikacji, w ramce *Program wykonywany po synchronizacji przy starcie* można zadeklarować program, który zostanie wykonany po zakończeniu synchronizacji. Program ten może wykonywać dowolne dodatkowe czynności wymagane przez aplikację do poprawnego zakończenia synchronizacji.

Automatyczna synchronizacja polega na transferze nowszych plików katalogu wzorcowego do lokalnego katalogu definicyjnego. Transferowane są też pliki, których nie ma w katalogu lokalnym. Kopiowanie w drugim kierunku -do katalogu wzorcowego odbywa się ręcznie. Można użyć narzędzi systemowych lub okna synchronizacji Asix.Evo (otwieranego komendą *Synchronizacja* z menu *Narzędzia).*



Rys. Okno 'Synchronizacja plikacji'.

W oknie pokazane są wszystkie pliki katalogu definicyjnego. Kolorami zaznaczone są różnice pomiędzy wzorcem, a katalogiem lokalnym. Przy pomocy przycisków okna można kopiować zaznaczone lub nowsze pliki w dowolnym kierunku oraz kasować wybrane pliki wzorca. W trakcie operacji synchronizacji można zapisać notatkę opisującą wykonane zmiany. Notatki zapisywane są w pliki *SynchronizationNotes.xml* w katalogu wzorca.

11.2 Synchronizacja katalogu roboczego

Parametryzacja synchronizacji katalogu roboczego ogranicza się do podania katalogu wzorca oraz ewentualnych danych użytkownika z uprawnieniami dostępu do katalogu.

Synchronizacja katalogu roboczego ogranicza się do synchronizowania lokalnych przebiegów wzorcowych przechowywanych w podkatalogu *TrendPatterns*. Działanie synchronizatora jest w pełni automatyczne. W przypadku zmiany lub dodania lokalnego przebiegu wzorcowego, plik opisujący przebieg jest natychmiast transferowany do katalogu wzorcowego. Funkcje służące do odczytu przebiegów wzorcowych zawsze sprawdzają, czy w katalogu wzorcowych nie ma uaktualnionych definicji i w razie konieczności kopiują je na dysk lokalny.

12 Użycie przebiegów wzorcowych

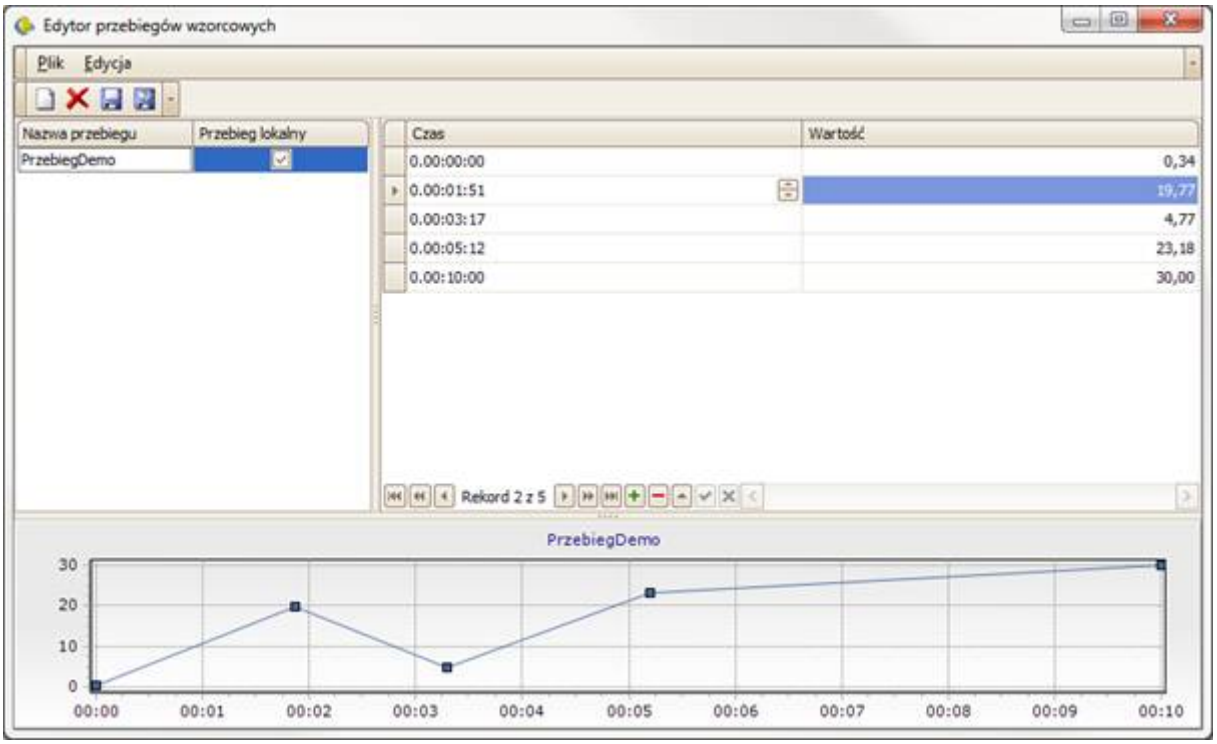
Przebiegi wzorcowe posiadają dwa typowe zastosowania:

- Wizualne porównanie rzeczywistych przebiegów wartości zmiennych procesowych względem idealnej krzywej wzorcowej.
- Transfer do sterownika danych określających sposób prowadzenia procesu.

W aplikacjach klasycznych przebiegi wzorcowe były przechowywane w specjalnym typie archiwum modułu Aspad. Dodatkowo, wybór i edycja przebiegów była wspierana przez programy *PEdit* i *PSelect*. Cała ta funkcjonalność jest teraz wbudowana bezpośrednio w program Asix.Evo.

12.1 Edycja przebiegów

Edycja i zarządzanie krzywymi wzorcowymi odbywa się w edytorze przebiegów, który można otworzyć komendą *Edytor przebiegów wzorcowych* w menu *Narzędzia*. W trakcie pracy w trybie wykonania aplikacji okno może być otwarte za pośrednictwem akcji operatorskiej *EditPatterns*.



Rys. Edytor przebiegów wzorcowych.

Przebiegi wzorcowe dzielą się na przebiegi wspólne oraz przebiegi lokalne użytkownika. Rodzaj przebiegu należy określić już w momencie jego tworzenia. Przebiegi wspólne muszą być tworzone w trybie edycji aplikacji. Plik opisujący taki przebieg jest zapisywany w podkatalogu *TrendPatterns* katalogu definicyjnego aplikacji. Przebieg wspólny nie może być zmieniony przez użytkownika w trakcie pracy aplikacji. Przebiegi lokalne są zapisywane w podkatalogu *TrendPatterns* katalogu roboczego aplikacji. Przebiegi lokalne mogą być zakładane i edytowane w trakcie pracy aplikacji. Z założenia są one używane na konkretnym stanowisku aplikacji. Jeżeli jednak jest włączona synchronizacja zawartości katalogu roboczego, to przebiegi lokalne mogą być propagowane na inne stanowiska aplikacji.

Punkty przebiegu wzorcowego mogą być edytowane w tabeli punktów. Można też przesuwać położenie punktów bezpośrednio na wykresie podglądowym. Lewy klawisz myszki służy do przesuwania punktów, prawy do przesuwania wykresu, a podwójne kliknięcie dopasowuje skale wykresu do definicji punktów.

W przypadku konieczności konwersji przebiegów wzorcowych z aplikacji klasycznej, można posłużyć się funkcją importu plików w formacie CSV.

12.2 Wyświetlanie krzywych wzorcowych

Przebiegi wzorcowe mogą być wyświetlone jako krzywa w obiekcie klasy *Wykres*. Obiekt *Wykres* może mieć wstępnie sparametryzowane pokazywane krzywe wzorcowe. Posiada też własne interfejsy pozwalające na wybór krzywej wzorcowej i jej punktu zaczepienia w trakcie pracy aplikacji.

Można też automatycznie sterować parametrami krzywych wzorcowych w obiekcie *Wykres*. Przykład takiego działania znajduję się w rozdziale *Kontrola wyświetlania przebiegów wzorcowych* w podręczniku *Asix.Evo - Techniki budowy diagramów*.

12.3 Wybór krzywej wzorcowej

W trakcie działania aplikacji można interaktywnie wybierać nazwę krzywej procesowej za pośrednictwem akcji SelectPattern. W wyniku wykonania tej akcji nazwa wybranej krzywej wpisywana jest do zmiennej procesowej. Za pośrednictwem tej zmiennej można np. sterować wyświetlaną krzywą w obiekcie Wykres.

12.4 Transfer przebiegów

Przebiegi wzorcowe mogą być również podstawą sterowania kontrolowanym procesem przez sterownik PLC. W tym celu konieczne jest przesłanie definicji krzywej do sterownika. Ze względu na brak standardów, Asix.Evo nie posiada wbudowanych funkcji transferu krzywych. Operacje tego typu należy wykonywać za pośrednictwem skryptów. W rozdziale *Transfer przebiegów wzorcowych* podręcznika *Asix.Evo - Skrypty* podany jest przykład przesłania definicji krzywej wzorcowej do sterownika.

13 Klawiatura ekranowa

W przypadku uruchomienia aplikacji na komputerze pozbawionym klawiatury może być konieczne użycie klawiatury ekranowej do wprowadzania wartości nastaw. Można wykorzystać klawiaturę wbudowaną w system operacyjnym. Asix.Evo dostarcza jednak dwa własne mechanizmy omówione w kolejnych rozdziałach.

13.1 Obiekt Klawiatura

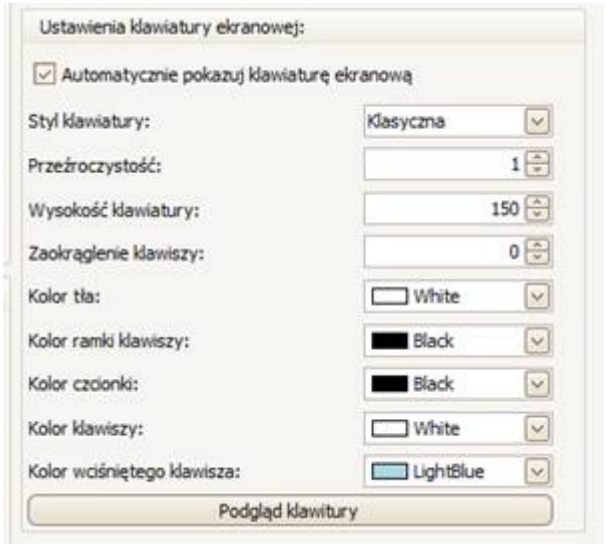
Obiekt klasy *Klawiatura* należy umieścić na diagramie, na którym znajdują się obiekty służące do wprowadzania nastaw. Klawiatura może być wyświetlana przez cały czas lub pracować w trybie tzw. autoukrywania (wyświetla się wtedy na tle diagramu dopiero po wejściu w tryb edycji).

Po wejściu w tryb edycji nastawy (np. w obiekcie klasy *Tekst*) można wprowadzać kolejne znaki, używając pól klawiatury.

13.2 Klawiatura okienkowa

Bardziej uniwersalnym mechanizmem jest klawiatura wyświetlana we własnym okienku. Pozwala ona nie tylko na wprowadzania nastaw w obiektach diagramu, ale także na edycję pól w okienkach dialogowych.

Wygląd i tryb działania klawiatury ekranowej określa się w panelu roboczym ustawień stanowiska w zakładce *Ustawienia*.



Rys. Ustawienia klawiatury ekranowej.

Zaznaczenie pola wyboru *Automatycznie pokazuj klawiaturę ekranową* powoduje, że okienko klawiatury wyświetla się automatycznie w momencie wykrycia wejścia w tryb edycji.

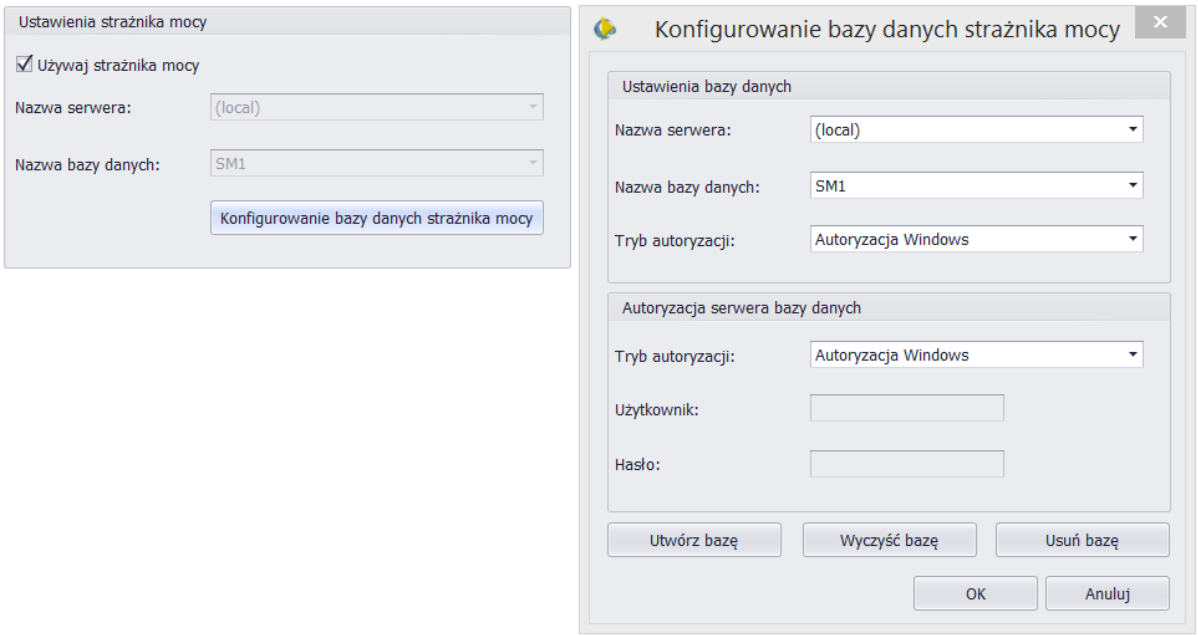
Jeżeli tryb automatyczny nie jest włączony lub wejście w edycję nastąpiło w elemencie aplikacji nie kontrolowanym bezpośrednio, to okienko klawiatury można włączyć przy pomocy akcji operatorskich *ShowKeyboard* i *ShowNumericKeyboard.*

14 Strażnik Mocy

Moduł Strażnika Mocy pozwala monitorować bieżące zużycie wybranych mediów (np. energii elektrycznej, gazu) i na podstawie prognozy alarmować z wyprzedzeniem zagrożenie przekroczenia mocy zamówionej, które pociąga za sobą dotkliwe opłaty dodatkowe na rzecz dostawcy energii/gazu. W innym scenariuszu, Moduł Strażnika może nadzorować prawidłową sytuację w procesie produkcyjnym, kontrolując zużycie mocy w stosunku do mocy zadanej wynikającej z prawidłowego jej poziomu dla określonego planu produkcyjnego.

14.1 Konfiguracja bazy danych SQL

W panelu *Eksplorator aplikacji - Strażnik mocy* w zakładce *Ustawienia* należy wybrać opcję *Konfigurowanie bazy danych strażnika mocy* i w okienku dialogowym po wskazaniu serwera SQL (np. (local)) nazwę bazy danych i tryb autoryzacji, kliknąć *Utwórz bazę* i zatwierdzić *OK*. Po zamknięciu okienka należy zaznaczyć opcję *Używaj strażnika mocy*.

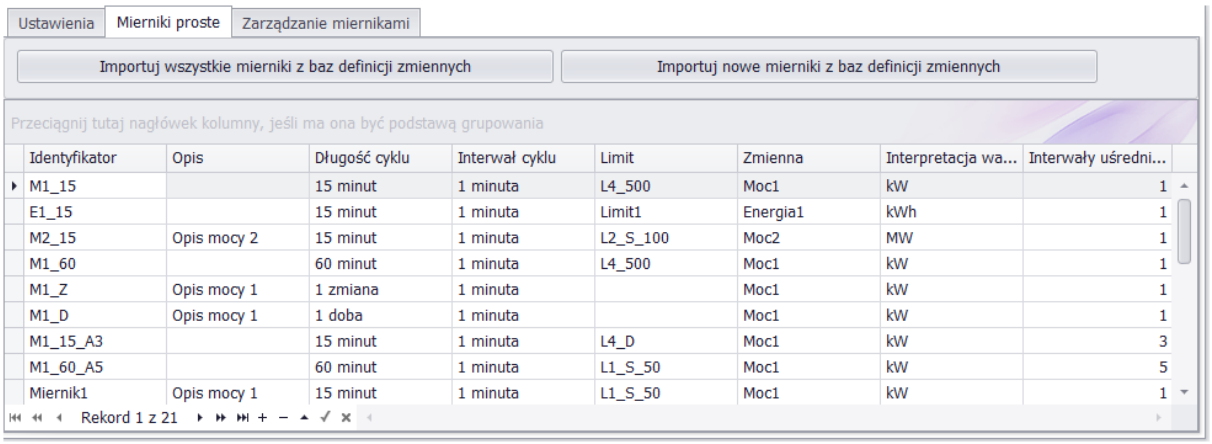


Rys. Konfiguracja bazy danych Strażnika Mocy.

14.2 Tworzenie mierników prostych

W zakładce *Mierniki proste* można dodawać mierniki powiązane ze zmiennymi indywidualnie lub importować je z bazy definicji zmiennych - o ile w bazie zostały zdefiniowane odpowiednie atrybuty zmiennych (`MeterId, MeterCycle, MeterInterval, MeterUnit, MeterLimit, MeterPrediction`). Każdy miernik powinien mieć skonfigurowany identyfikator, opis, jednostkę pracy (W, kW, MW,

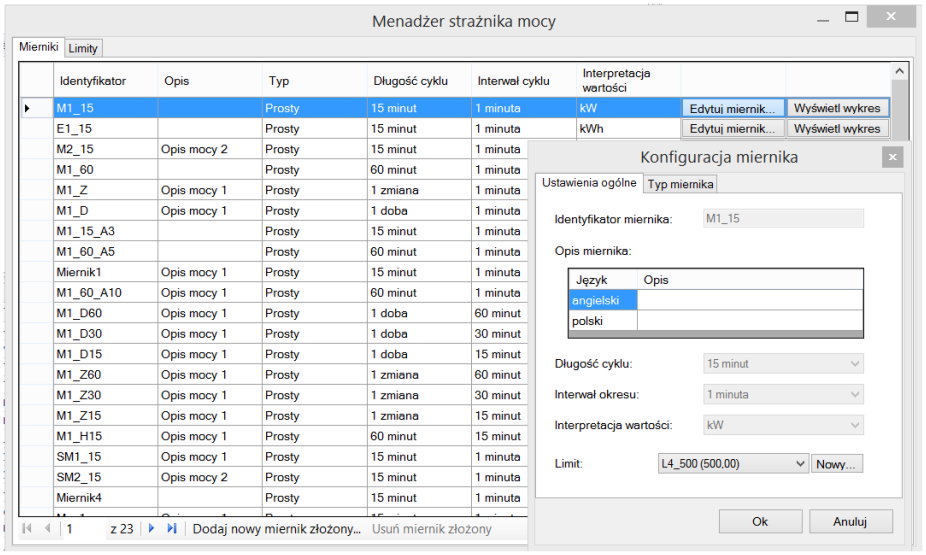
kWh, MWh, m³/h, m³, tgϕ), cykl pracy (`15m, 30m, 60m, Zmiana, Doba`) i interwał agregacji (`1m, 15m, 30m, 60m`) oraz parametryzowany limit (moc zmówioną). Mierniki współpracują zarówno z pomiarami wartości chwilowych (np. moc w kW) jak i z pomiarami zwracanymi przez liczniki (np. energii w kWH) - decyduje o tym wybrana jednostka (kolumna *Interpretacja wartości*). Mierniki proste muszą być skojarzone z odpowiednimi zmiennymi procesowymi będącymi dla nich źródłem danych. W kolumnie *Interwały uśredniania* można sparametryzować algorytm predykcji, poprzez określenie ile poprzedzających interwałów historycznych będzie uwzględnianych do predykcji wartości interwałów przyszłych w cyklu.



Rys. Tworzenie mierników prostych.

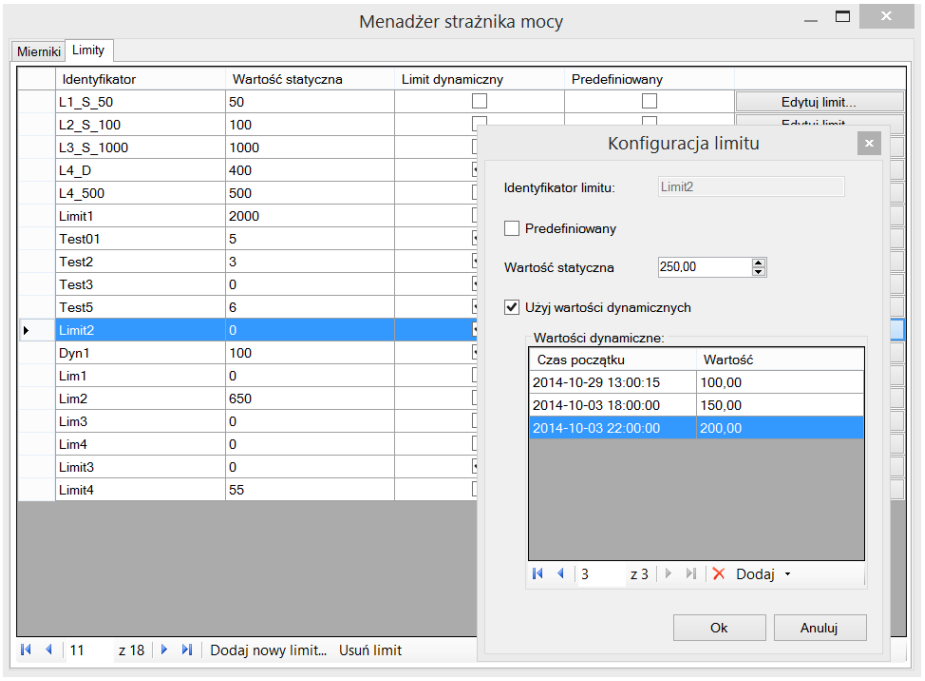
14.3 Zarządzanie miernikami i limitami

W kolejnej zakładce *Zarządzanie miernikami* można wywołać okno *Menadżera strażnika mocy,* które służy do zarządzania limitami i miernikami złożonymi. Umożliwia ono również przypisywanie limitów zarówno miernikom prostym jak i złożonym oraz wywoływanie podglądu pracy mierników w postaci wykresu. Okno to może być dostępne dla operatorów aplikacji, jeżeli zostanie wywołane akcją operatorską *PowerGuardManager()*.



Rys. Zarządzanie miernikami.

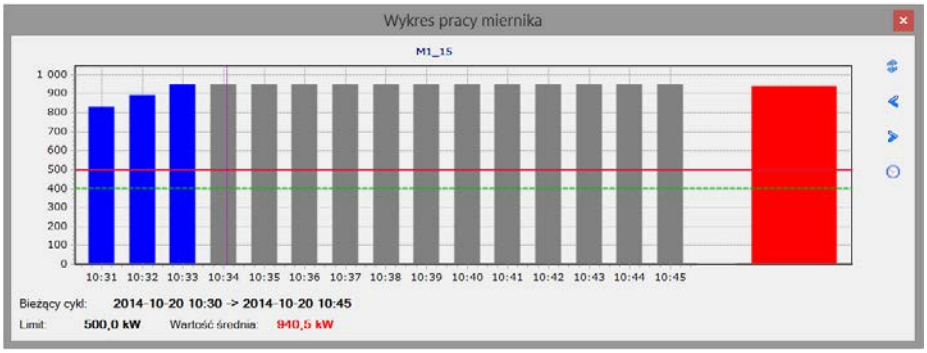
W zakładce *Limity* możemy dodawać i usuwać limity mierników. W przypadku mierników mocy limit jest rozumiany jako moc zamówiona. Limit może być statyczny - wtedy posiada zawsze stałą wartość, ale może też być dynamiczny - jeżeli jego wartość ma ulegać zmianom w czasie i być różna w kolejnych cyklach pracy. W przypadku wartości dynamicznych wszystkie wartości historyczne limitów są archiwizowane w bazie, aby możliwe było przeglądanie cyklów historycznych mierników z limitami, jakie faktycznie w danym czasie dla nich obowiązywały. Dodawać i zmieniać można tylko wartości dla cyklów przyszłych.



Rys. Zarządzanie limitami.

Jeżeli mamy stworzone jakieś limity, to w zakładce Mierniki możemy je przypisywać do poszczególnych mierników prostych lub złożonych, klikając w przycisk *Edytuj miernik...*

Po kliknięciu w przycisk *Wyświetl wykres* dla danego miernika wywołamy okno z wykresem, pozwalające obserwować interwały oraz średnią wartość prognozowaną w bieżącym cyklu. Ponadto, wykres wyświetla pionową linię czasu bieżącego, poziome linie limitu oraz poziomą linię, pokazującą jaka wartość jest bezpieczna dla pozostałych interwałów - tak aby całkowita średnia za cykl nie przekroczyła limitu. Wykres umożliwia także przegląd cyklów historycznych za pomocą przycisków nawigacyjnych z prawej strony. Okienko wykresu dla konkretnego miernika może być też wywoływane z wizualizacji przez operatora za pomocą akcji operatorskiej *PowerGuardMeter(meterId)*.



Rys. Wykres pracy miernika.

14.4 Obiekt wizualizacyjny - wykres miernika strażnika mocy

Obiekt ten przeznaczony do umieszczania na diagramach synoptycznych posiada jednakową funkcjonalność jak opisany powyżej wykres w postaci niezależnego okna. Ponadto możliwe jest dowolne sparametryzowanie jego wyglądu (kolorów) oraz dostępnych funkcji i wyświetlanych elementów składowych.

14.5 Funkcje wyrażeń strażnika mocy

Na diagramach wizualizacyjnych można bezpośrednio wyświetlać wartości związane z bieżącym stanem wskazanego miernika. Służyły do tego funkcja *PowerGuardRead(meterId, data).*

Poprzez przekazanie odpowiedniej stałej w drugim argumencie funkcji możliwe jest uzyskanie następujących danych:

$AverageValue - Przewidywana średnia wartość w bieżącym cyklu

$Limit - Bieżący limit

$CycleStartTime - Czas początku cyklu

$CycleEndTime - Czas końca cyklu

$IsPredictionSafe - Informacja czy przewidywana wartość średnia przekracza limit

$SafeRemainingValue - Średnia wartość bezpieczna dla pozostałych interwałów

14.6 Integracja z systemem alarmów

Generowanie zdarzeń alarmów związanych ze stanem mierników strażnika mocy jest możliwe dzięki nowej strategii alarmów "Strażnik mocy". Jeżeli mamy w danej domenie alarmów dodaną strategię "Strażnik mocy", to dla wybranych alarmów skojarzonych z tą strategią w parametrach wykrywania wystarczy podać identyfikator miernika. Alarm będzie wszczynany za każdym razem, gdy w bieżącym cyklu miernik wykryje zagrożenie, że średnia wartość śledzonego medium na koniec cyklu przekroczy zakładany limit (np. moc zamówioną) i będzie tak długo utrzymywany, aż zagrożenie nie minie w cyklu bieżącym lub kolejnych cyklach.

15 Aplikacja Asix jako usługa systemu Windows

System Asix od wersji Asix 8.0.4 posiada możliwość uruchomienia swojej aplikacji jako usługi systemu Windows.

Z definicji, usługą Windows jest proces wykonywany wewnątrz środowiska systemu operacyjnego Windows, niewymagający interakcji z użytkownikiem komputera. Proces taki nazywany jest również serwisem Windows. Usługa zwykle jest uruchamiana podczas startowania systemu operacyjnego i działa tak długo, jak długo system jest czynny.

Aplikacja Asix.Evo pracująca jako usługa Windows posiada pełną funkcjonalność dostępną dla tradycyjnych aplikacji Asix.Evo, jednak ze względu na specyfikę działania usług, praca w tym trybie całkowicie pozbawiona jest interfejsu użytkownika (użytkownik nie widzi działającej aplikacji). Takie rozwiązanie doskonale nadaje się do roli serwera aplikacji www lub Historiana - serwera zajmującego się gromadzeniem, przetwarzaniem, archiwizowaniem i udostępnianiem danych klientom systemu Asix.

Podsumowując, aplikacja Asix.Evo pracująca jako usługa systemowa:
- działa na koncie systemowym - nie wymaga logowania się użytkownika do systemu Windows;

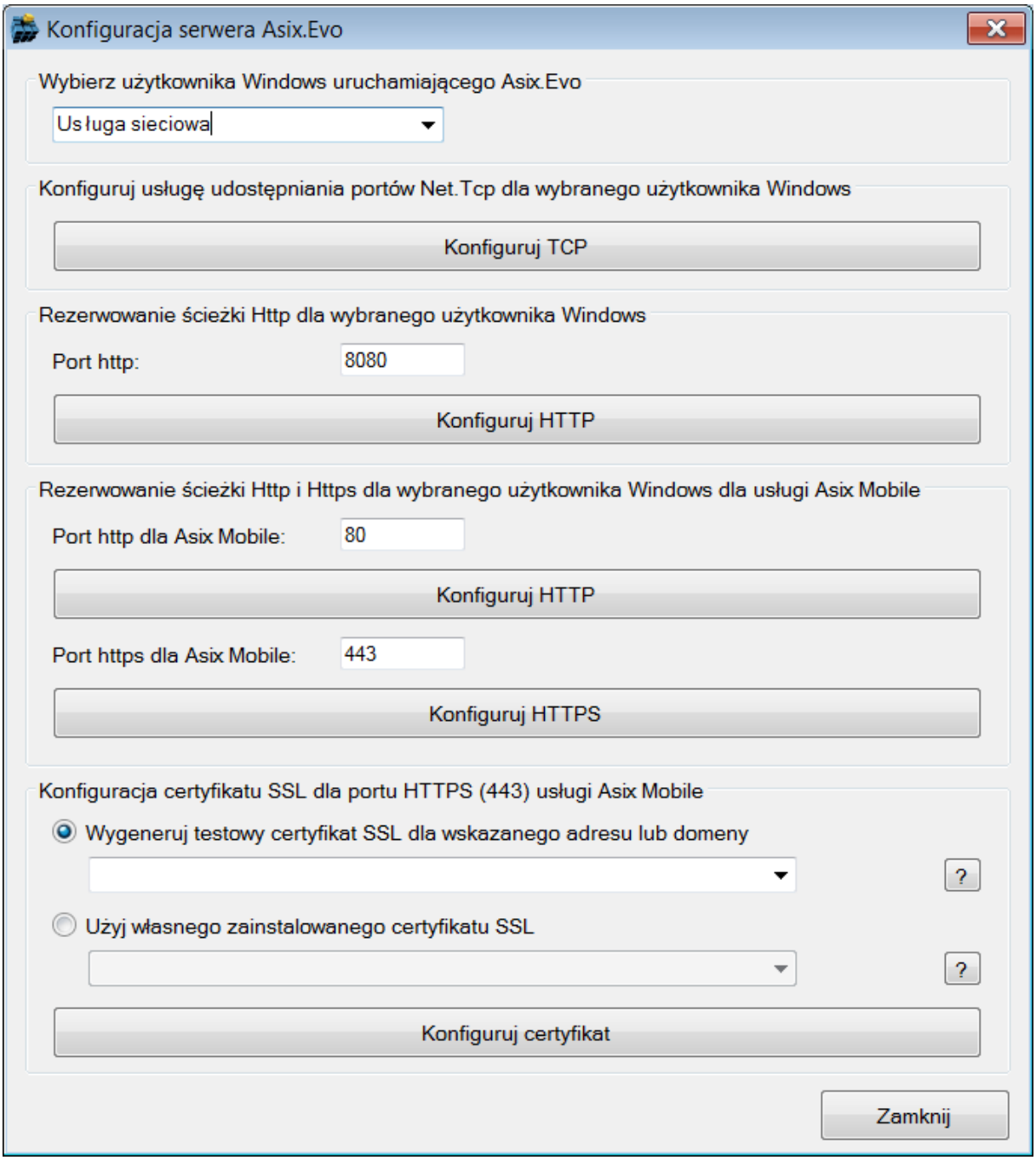
- uruchamia się automatycznie bezpośrednio po uruchomieniu systemu operacyjnego;
- pracuje bezobsługowo, bez interfejsu użytkownika.

Przygotowanie aplikacji

Skonfigurowanie aplikacji systemu Asix do pracy jako usługi Windows odbywa się w prosty sposób, przy użyciu specjalnie do tego celu przygotowanego konfiguratora.

W pierwszej kolejności należy przygotować aplikację do pracy w trybie usługi systemowej. Obejmuje to:

- przygotowanie aplikacji pozbawionej interfejsu użytkownika (należy pamiętać, że aplikacja uruchamiana jako serwis nie będzie posiadała elementów wizualnych: okien, diagramów, komunikatów itd.);
- skonfigurowanie komunikacji serwera dla użytkownika *Usługa sieciowa* (konstruktor AsixEvo.exe > *Ustawienia globalne* > *Komunikacja* > *Konfigurator komunikacji Windows*).

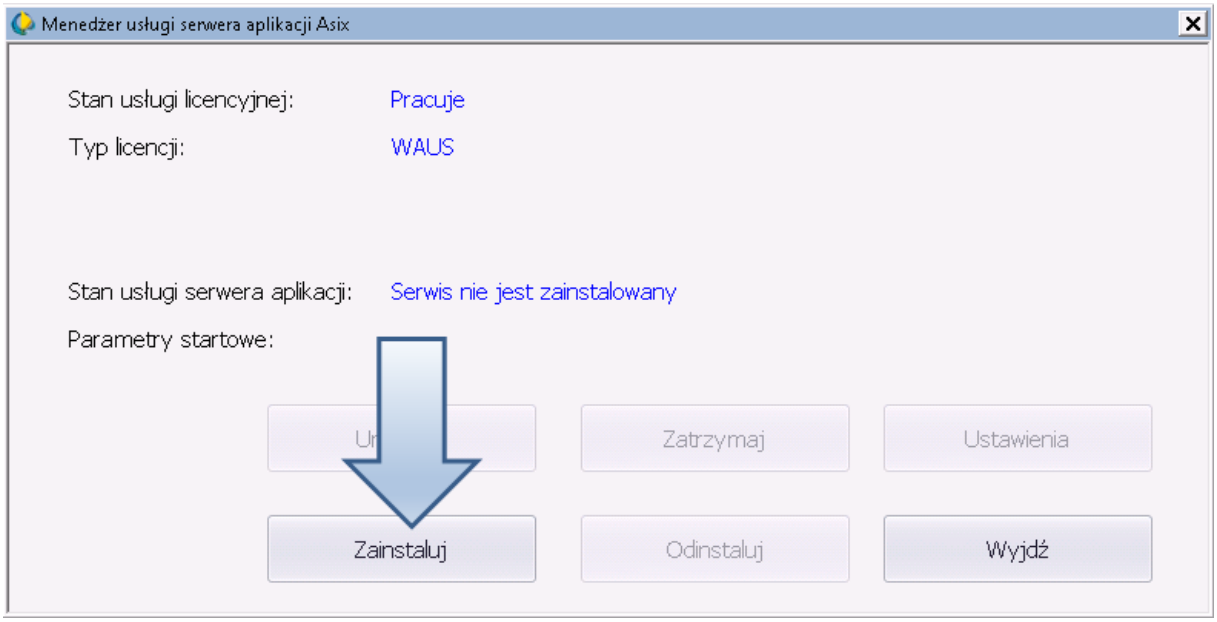


Rys. Konfigurowanie komunikacji serwera dla użytkownika 'Usługa sieciowa'.

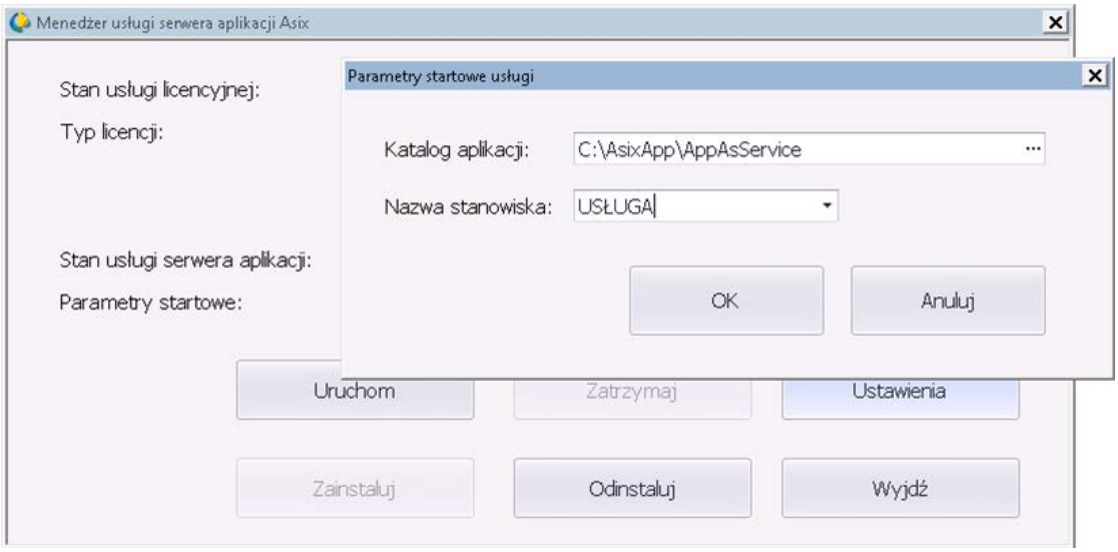
Instalacja usługi

W drugiej kolejności należy zainstalować usługę przy użyciu programu 'Menedżer usługi serwera aplikacji Asix'. Domyślna lokalizacja programu:

C:\Program Files (x86)\Askom\Asix\Asix.Evo\AsixServiceManager.exe



Rys. Uruchomienie instalacji usługi serwera aplikacji Asix.



Rys. Ustawienie parametrów startowych usługi i uruchomienie usługi.

Monitorowanie usługi

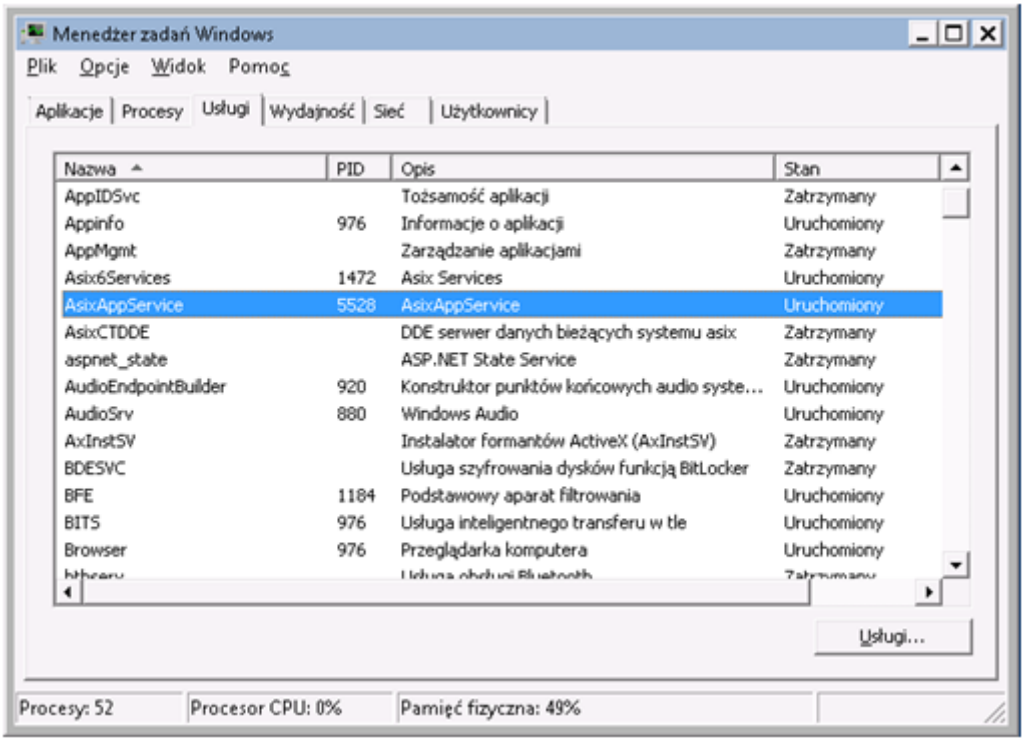
Monitorowanie poprawności działania usługi poprzez:

- wynik działania aplikacji,
- monitorowanie usług systemowych,

- logi systemowe.

Monitorowanie procesów:

- AsixAppService – aplikacja,
- AsixDataServiceHost – serwer danych.



Rys. Monitorowanie usług systemowych.

Zatrzymanie i odinstalowanie usługi

Zatrzymanie i odinstalowanie usługi możliwe jest przy użyciu programu 'Menedżer usługi serwera aplikacji Asix'. Domyślna lokalizacja programu:
C:\Program Files (x86)\Askom\Asix\Asix.Evo\AsixServiceManager.exe

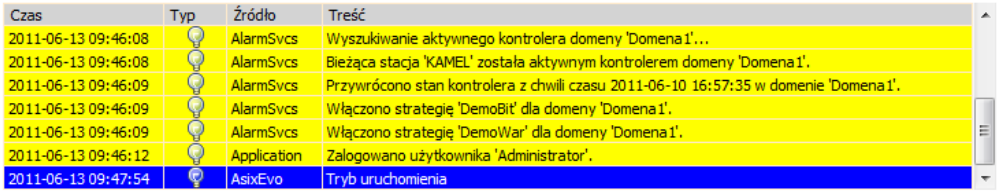
Asix.Evo - Obiekty

Obiekt Dziennik zdarzeń

Funkcje obiektu

Obiekt *Dziennik zdarzeń* pozwala na wyświetlenie na diagramie tabeli pokazującej ostatnie komunikaty systemowe. Są to te same komunikaty, które w trybie pracy architekta wyświetlane są w panelu *Komunikaty*. Zalecane jest użycie w aplikacji przynajmniej jednego obiektu *Dziennik zdarzeń* – umożliwia to łatwą diagnostykę pracy aplikacji.

Konfigurując dziennik zdarzeń możemy ustawić predefiniowane filtry typu komunikatów oraz filtry źródeł komunikatów. Dodatkowo można udostępnić lub zablokować operatorom funkcje sortowania i filtrowania w czasie pracy.



Rys. Przykład obiektu Dziennik zdarzeń.

Alternatywną metodą dostępu do logu komunikatów w trybie wykonawczym jest okno komunikatów, które można otworzyć za pośrednictwem okna panelu kontrolnego.

Obiekt *Dziennik zdarzeń* jest obiektem typu kontrolkowego.

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Zezwolenie na sortowanie	Logiczna	Określa, czy użytkownik może zmieniać kolejność wyświetlanych komunikatów. Zmiana kolejności jest inicjowana kliknięciem na nagłówku kolumny.

Zezwolenie na filtrowanie	Logiczna	Określa, czy użytkownik może ustawiać filtry wyboru komunikatów. Filtr może dotyczyć typu lub źródła pochodzenia komunikatu. Zmiana filtru odbywa się poprzez kliknięcie prawym klawiszem myszki na kolumnie *Typ* lub *Źródło*. Można wstępnie określić warunki filtrowania przy pomocy właściwości *Filtr typu komunikatu* i *Filtr źródła komunikatów.*
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki komunikatów	Liczba	Określa rozmiar czcionki używanej do wyświetlenia komunikatów.
Filtr typu komunikatów	Typ wyliczeniowy wielokrotny: Brak, Informacja, Błąd, Ostrzeżenie, Wyjątek	Określa typy komunikatów, które mają być wyświetlane.
Filtr źródła komunikatów	Tekst	Określa nazwę źródła komunikatów, z którego komunikaty mają być pokazywane.

Właściwości stanowe

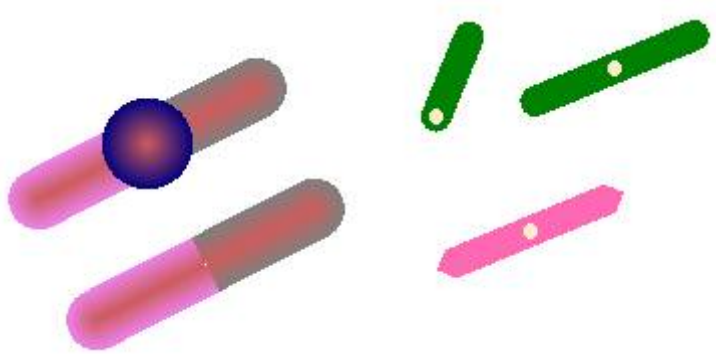
Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła obiektu.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze obiektu.
Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.
Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.

Kolor czcionki komunikatów	Kolor	Określa kolor tekstów wyświetlanych w wierszach komunikatów.
Kolor tła komunikatów	Kolor	Określa kolor tła wierszy komunikatów.
Kolor czcionki wybranego komunikatu	Kolor	Określa kolor tekstów wyświetlanych w wybranym wierszu komunikatów.
Kolor tła wybranego komunikatu	Kolor	Określa kolor tła wybranego wiersza komunikatów.

Obiekt Klapa

Funkcje obiektu

Obiekt *Klapa* służy do prezentacji graficznej nachylenia elementów obrotowych typu klapy, żaluzje, przesłony itp. Dla każdego obiektu można zdefiniować kąty początku i końca wychylenia elementu, określić typ elementu (jedno- lub dwuramienne) oraz kształt końcówek.



Rys. Przykłady obiektu Klapa.

Obiekt *Klapa* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

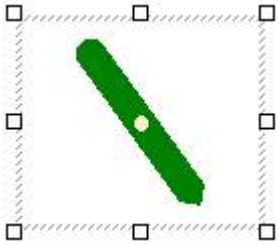
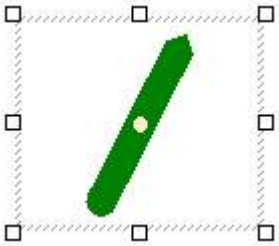
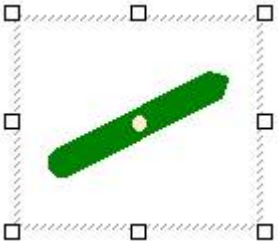
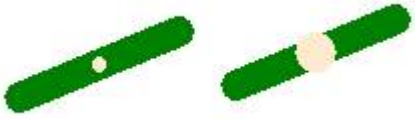
Zobacz: *Techniki_budowy_diagramów* > ***Parametryzacja funkcji interaktywnych w obiektach pasywnych***

Właściwości

Zobacz: *Elementy_wizualizacji* > ***Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wartość minimalna	Liczba	Dolny limit położenia klapy, odpowiadający kątowi początkowemu.
Wartość maksymalna	Liczba	Górny limit położenia klapy, odpowiadający kątowi końcowemu.
Kąt początkowy	Liczba	Określa nachylenie klapy dla wartości minimalnej, wyrażone jako kąt liczony względem poziomego (w prawo) położenia wskazówki. Skrajne maksymalne wychylenie określone jest właściwością *Kąt końcowy*. Wzajemna zależność obu właściwości kątów skali decyduje o kierunku obrotu wskazówki. W razie konieczności można stosować kąty większe od 360 stopni. W poniższe przykładach wartości minimalne i maksymalne wynoszą odpowiednio 0 i 100, a bieżąca wartość jest równa 30. Kąt początku 0, Kąt końca 90:  Kąt początku 0, Kąt końca 180::

		 Kąt początku 270, Kąt końca 360:  Kąt początku 360, Kąt końca 270: 
Kąt końcowy	Liczba	Określa nachylenie klapy dla wartości maksymalnej.
Szerokość klapki	Liczba	Określa szerokość klapy.
Dwa ramiona	Logiczna	Określa czy klapa posiada dwa ramiona. 
Rozmiar przykrywki osi	Liczba	Określa rozmiar przykrywki osi obrotu klapy. 

Styl pierwszej końcówki	Typ wyliczeniowy: Płaski, Okrągły Trójkątny	Określa sposób zakończenia pierwszego ramienia klapy. 
Styl drugiej końcówki	Typ wyliczeniowy: Płaski, Okrągły Trójkątny	Określa sposób zakończenia drugiego ramienia klapy.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Wartość	Liczba	Określa aktualne położenie wskazówki miernika. Wartość właściwości powinna być liczbą z zakresu określonego przez właściwości *Wartość minimalna* i *Wartość maksymalna*.
Kolor pierwszego ramienia	Kolor	Określa kolor pierwszego ramienia klapy.
Kolor gradientu pierwszego ramienia	Kolor	Określa kolor stosowany do wyświetlania pierwszego ramienia. Kolor będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu.
Kolor drugiego ramienia	Kolor	Określa kolor drugiego ramienia klapy.
Kolor gradientu drugiego ramienia	Kolor	Określa kolor stosowany do wyświetlania drugiego ramienia. Kolor będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu.
Kolor przykrywki osi	Kolor	Określa kolor przykrywki osi obrotu klapy. Kolor *Transparent* pozwala wyłączyć wyświetlanie przykrywki.

Kolor gradientu przykrywki osi	Kolor	Określa kolor stosowany do wyświetlania przykrywki osi. Kolor będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu.

Zdarzenia

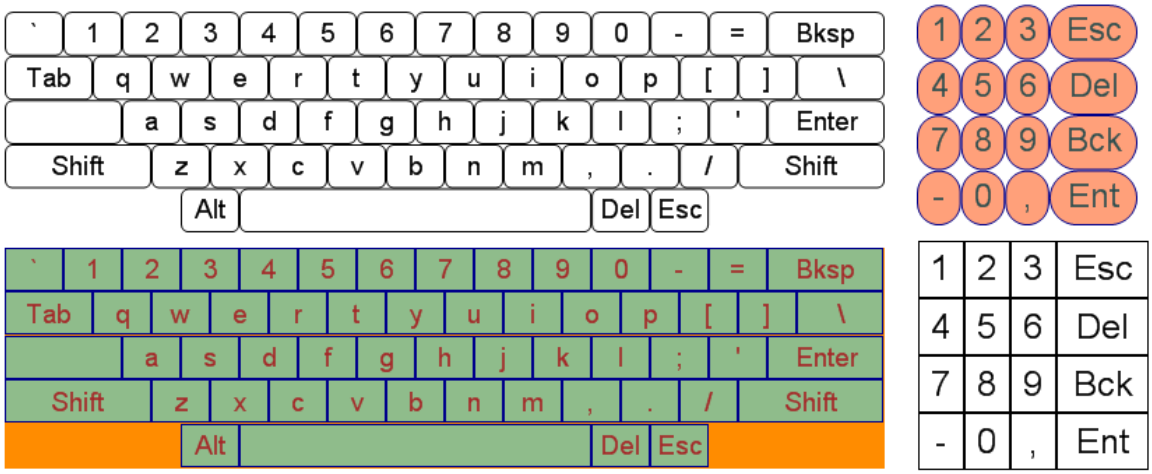
Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Klawiatura

Funkcje obiektu

Obiekt *Klawiatura* jest obiektem umożliwiającym wprowadzanie tekstu i wartości liczbowych za pomocą myszki lub ekranów dotykowych. Może pracować w trybie pełnej klawiatury alfanumerycznej lub jako prosta klawiatura numeryczna. Dla każdego obiektu klawiatury możemy ustawić kolory napisów, obwódki, tła, klawiszy i wciśniętych klawiszy oraz promień zaokrąglenia klawiszy.

Użytkownik po wybraniu pola służącego do wprowadzenia tekstu może używać klawiszy obiektu *Klawiatura* do wpisywania kolejnych znaków. Klawiatura może pracować w trybie automatycznego pokazywania - pojawia się w momencie rozpoczęcia wprowadzania tekstu. Może być też automatycznie pozycjonowana - ustawia się wtedy w pobliżu edytowanego tekstu.



Rys. Przykłady obiektu Klawiatura.



Rys. Przykłady zastosowania obiektu Klawiatura w połączeniu z obiektem Obraz.

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Rodzaj klawiatury	Typ wyliczeniowy: Klasyczna, Numeryczna	Określa, czy wyświetlona klawiatura posiada pełny zestaw klawiszy czy tylko klawisze numeryczne.
Promień zaokrąglenia klawiszy	Liczba	Określa promień zaokrąglenia klawiszy.
Automatyczne pokazywanie	Logiczna	Wartość *Tak* oznacza, że klawiatura jest normalnie ukryta i pojawia się tylko w momencie wprowadzania tekstu.
Automatyczne pozycjonowanie	Logiczna	Określa, czy pozycja klawiatury ma być automatycznie dostosowana do miejsca edycji tekstu.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor czcionki	Kolor	Określa kolor opisu klawiszy.
Kolor tła	Kolor	Określa kolor klawiatury poza obszarem klawiszy.
Kolor klawiszy	Kolor	Określa kolor tła klawiszy.
Kolor otoczki klawiszy	Kolor	Określa kolor używany do rysowania obwódki klawiszy.
Kolor zaznaczonych klawiszy	Kolor	Określa kolor służący do oznaczenia przyciskanych klawiszy.

Zdarzenia

Zobacz: *Elementy_wizualizacji > Standardowe zdarzenia obiektów*

Obiekt Komunikaty

Funkcje obiektu

Obiekt *Komunikaty* jest obiektem, który wyświetla tekstowe opisy stanów zdefiniowanych w pliku zestawu komunikatów. Działanie obiektu jest zbliżone do sposobu działania obiektu klasy KOMUNIKATY systemu Asix w wersji klasycznej. W plikach zestawu komunikatów definiowane są stany obiektu. Opis każdego stanu składa się z wartości stanu, napisu opisującego stan oraz atrybutów wyświetlania. W trakcie pracy aplikacji, na podstawie aktualnej wartości stanu wyszukiwany jest odpowiedni opis tekstowy w pliku zestawu komunikatów. Wartość stanu może przyjmować postać: numeryczną, zakres wartości, postać tekstową lub *. Znak '*' oznacza stan domyślny, wyświetlany gdy wartość zmiennej głównej obiektu Komunikaty nie pasuje do zdefiniowanych wartości stanów.



Rys. Przykłady obiektu Komunikaty.

Możliwości parametryzacji wyglądu wyświetlanych tekstów są ograniczone do wyboru koloru i czcionki. Obiekt nie posiada wbudowanych żadnych funkcji sterujących. Jeżeli wymagane jest rozszerzone formatowanie tekstów lub funkcje sterujące, to należy używać obiektów klasy *Tekst*.

Zestaw komunikatów tworzony jest przy użyciu komendy *Utwórz zestaw komunikatów*, dostępnej po kliknięciu prawym przyciskiem myszy na węźle *Zestaw komunikatów* w panelu *Eksplorator aplikacji* (menu *Widok* > *Eksplorator aplikacji*). Komenda tworzy zestaw komunikatów o domyślnej nazwie 'MessageSet1' (nazwę można zmienić klikając prawym przyciskiem myszy na wybranym zestawie komunikatów i wybierając komendę *Zmień nazwę*). W trakcie tworzenia zestawu komunikatów, w

podkatalogu 'MessageSets' głównego katalogu aplikacji tworzony jest plik zestawu komunikatów (o rozszerzeniu .msf). Patrz punkt: *Definiowanie plików zestawu komunikatów*.

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	**Typ wartości**	**Opis**
Liczba linii	*Liczba*	Steruje sposobem dopasowania wysokości czcionki do długości tekstu. Wartość 0 oznacza, że należy tak zmniejszyć czcionkę, aby cały tekst zmieścił się w obszarze obiektu. Skutkiem tego, w zależności od długości tekstu, może zostać użyta inna czcionka. Wartość większa od 0 oznacza, że o wysokości czcionki decyduje wyłącznie rozmiar obiektu. W razie konieczności tekst może zostać obcięty lub podzielony na kilka linii.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Wartość	Liczba lub tekst	Określa aktualną wartość stanu. Na podstawie wartości tej właściwości wyszukiwany jest w pliku zestawu komunikatów aktualny opis stanu.
Zestaw komunikatów	Nazwa zestawu komunikatów	Nazwa pliku zestawu komunikatów, z którego odczytywane są opisy stanów.
Kolor	Kolor	Określa domyślny kolor wyświetlanych tekstów. Używany wtedy, gdy w definicji stanu kolor napisu nie został określony.
Kolor przekreślenia	Kolor	Określa kolor klawiatury poza obszarem klawiszy.

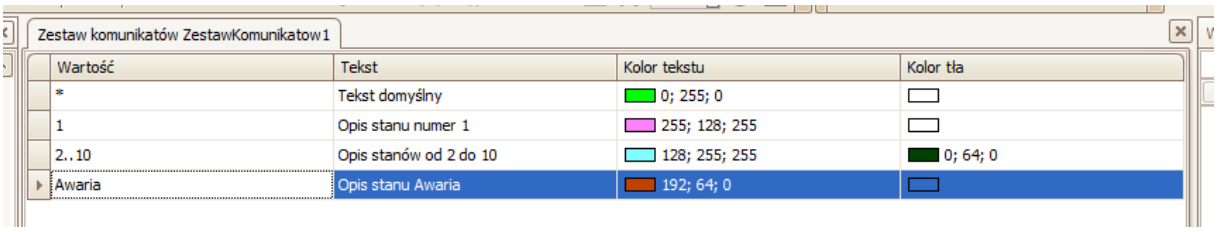
Kolor tła	Kolor	Określa domyślny kolor tła wyświetlanych tekstów. Używany wtedy, gdy w definicji stanu kolor tła napisu nie został określony.
Wyrównanie w poziomie	Typ wyliczeniowy: Do lewej, Wyśrodkowanie, Do prawej	Określa sposób poziomego pozycjonowania napisów wewnątrz obszaru obiektu.
Wyrównanie w pionie	Typ wyliczeniowy: Do góry, Wyśrodkowanie, Do dołu	Określa sposób pionowego pozycjonowania napisów wewnątrz obszaru obiektu.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania napisu.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.

Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Definiowanie plików zestawu komunikatów

Edycja pliku zestawu komunikatów odbywa się w panelu edycyjnym *Zestaw komunikatów*, uruchamianym po kliknięciu na wybranym zestawie komunikatów w węźle *Zestaw komunikatów*.



Rys. Zestaw komunikatów.

W kolumnie *Wartość* dopuszczalne są następujące typy danych:

- Znak *

Stan domyślny, używany wtedy gdy nie uda się dopasować do aktualnej wartości stanu żadnego innego komunikatu.

- *Numer stanu*

Numer stanu w postaci liczby całkowitej. Stan zostanie wybrany wtedy, gdy aktualna wartość stanu jest równa podanemu numerowi.

- Zakres numerów *numer_od..numer_do*

Zakres numerów stanu w postaci dwóch liczb całkowitych rozdzielonych znakami .. . Stan zostanie wybrany wtedy, gdy aktualna wartość stanu mieści się w podanym zakresie.

- *Nazwa stanu*

Tekstowa nazwa stanu. Stan zostanie wybrane wtedy, gdy aktualna wartość stanu w postaci tekstowej jest identyczna z podaną nazwą.

Możliwa jest także edycja plików zestawów komunikatów dowolnym edytorem zewnętrznym, pozwalającym na edycję plików zapisanych w kodzie Unicode. Plik zestawów znajduje się w podkatalogu *MessageSets* katalogu definicyjnego aplikacji.

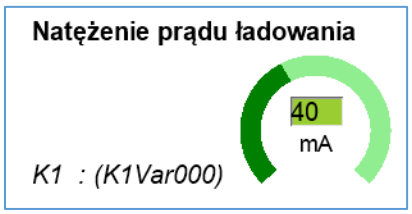
Obiekt Kontener wzorców

Funkcje obiektu

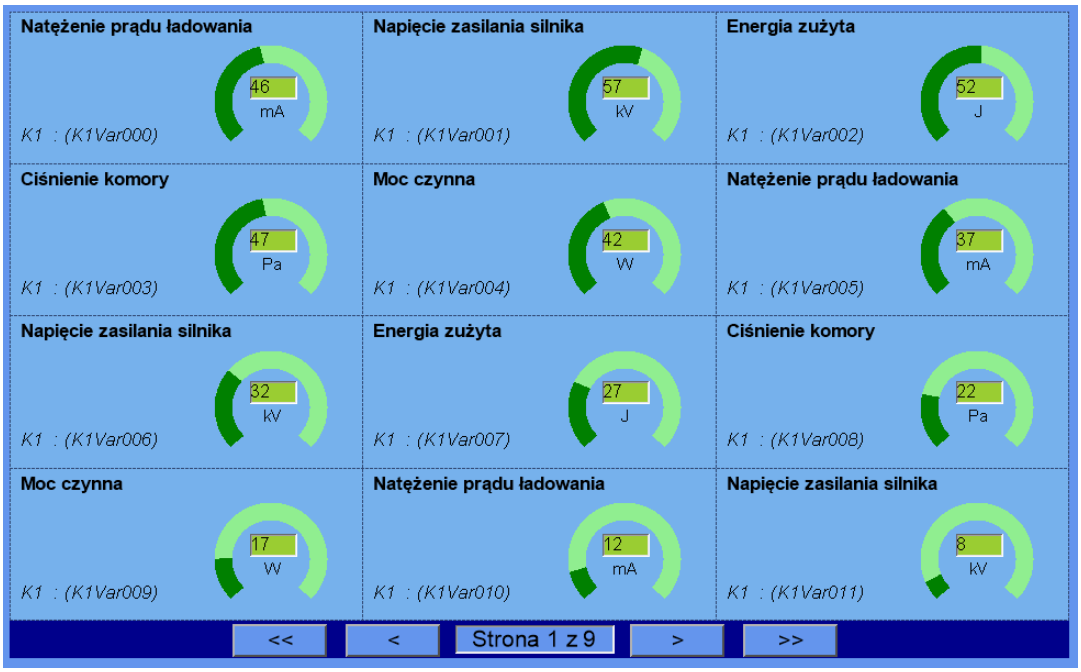
Obiekt Kontener wzorców ma za zadanie automatyczne powielanie wskazanego wzorca lub wzorców w oparciu o dostarczoną listę nazw wzorców lub dostarczone listy wartości parametrów wzorca. Pozwala na zdefiniowanie tabeli o zadanej liczbie wierszy i kolumn. W kolejnych komórkach tabeli są wyświetlane wzorce z następującymi po sobie wartościami parametrów. Jeżeli elementów do wyświetlania jest więcej niż komórek tabeli, to możliwe jest nawigowanie pomiędzy kolejnymi stronami za pomocą odpowiednich akcji operatorskich (akcja ObjectAction). Obiekt umożliwia także odczyt aktualne liczby elementów, całkowitej liczby stron oraz numeru bieżącej strony (funkcja ObjecData).

Przykład

Poniższy wzorzec pomiaru, którego parametrem jest nazwa zmiennej procesowej



może być powielony automatycznie za pomocą obiektu Kontener wzorców w następujący sposób:



Rys. Przykład obiektu Kontener wzorców.

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Źródło danych	DataTable	Pozwala ustanowić ogólne źródło danych dla obiektu w postaci typu wartości DataTable dostarczonego za pomocą funkcji AppData(..) lub jako wartość zmiennej. Dana aplikacyjna (AppData) lub zmienna może być ustawiana za pomocą skryptu jako wartość typu DataTable zawierająca zdefiniowane kolumny i wiersze. Nazwy wzorców oraz wartości parametrów będą pobierane ze wskazanej kolumny.
Liczba kolumn	Liczba	Decyduje o liczbie widocznych kolumn. Wartość minimalna to 1 a maksymalna to 50.
Liczba wierszy	Liczba	Decyduje o liczbie widocznych wierszy. Wartość minimalna to 1 a maksymalna to 50.
Nazwa wzorca/wzorców	Tekst, tablica lub kolekcja	Gdy zawiera nazwę jednego wzorca, to wzorzec ten jest powielany tyle razy, ile jest przekazanych wartości parametrów (najliczniejszy). Gdy zawiera listę nazw wzorców (oddzielone średnikami lub przekazane jako tablica, kolekcja lub kolumna tabeli, wynik funkcji VarNames(..)), to liczba wyświetlanych elementów wynika z liczebności tej listy. Jeżeli w obiekcie podane jest źródło danych (DataTable za pośrednictwem funkcji AppData(..)), to zamiast nazwy wzorca należy podać nazwę kolumny zawierającą nazwy wzorców.
Lewy margines *Prawy margines* *Górny margines* *Dolny margines*	Liczba	Właściwości te pozwalają określić procentową szerokość marginesu w każdym z kierunków.
Szerokość linii	Liczba	Pozwala zdefiniować grubość linii rozdzielających wiersze i kolumny.
Styl pasków	*Typ wyliczeniowy: Brak, Kreska, Kropka, Kreska-*	Definiuje styl malowania linii rozdzielających wiersze i kolumny.

	Kropka, Kreska-Kropka-Kropka	

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor	Kolor	Określa kolor tła tabeli.
Kolor linii	Kolor	Określa kolor linii rozdzielających wiersze i kolumny

Parametry wzorców

Nazwa	Typ wartości	Opis
Nazwa parametru	Tekst	Określa nazwę parametru wzorców
Wartości parametru	Tekst, tablica lub kolekcja	Właściwość ta może być zdefiniowana jako: • Lista wartości oddzielonych średnikami • Tablica - zmienna tablicowa lub zwracana przez funkcję AppData(..) • Kolekcja zwracana przez funkcję AppData(..) • Jeżeli w obiekcie podane jest źródło danych (DataTable za pośrednictwem funkcji AppData(..)), to należy podać w tym miejscu nazwę kolumny z wartościami parametru. • Wartości tablicowe zwracane przez funkcje: ToTable(..), Sequence(..), VarNames(..)

Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Akcje nawigowania

Dla obiektu Kontener wzorców w funkcji ObjectAction(..) dostępne są operacje:

- $First – Skacze do pierwszej strony
- $Previous – Przechodzi do poprzedniej strony
- $Next – Przechodzi do następnej strony
- $Last – Skacze do ostatniej strony
- $Reload – Przeładowuje wzorce
- $GoToPage – Skacze do wskazanej strony

Przykłady:
^ObjectAction("kw1", $Next)
^ObjectAction("kw1", $GoToPage, 5)

Wyświetlanie stanu obiektu

Dla obiektu Kontener funkcja ObjectData(..) obsługuje następujące dane:

- CurrentPage – Numer aktualnej strony
- PageCount – Całkowita liczba stron
- ElementCount – Całkowita liczba elementów

Przykład:
=ObjectData("kw1", "CurrentPage")

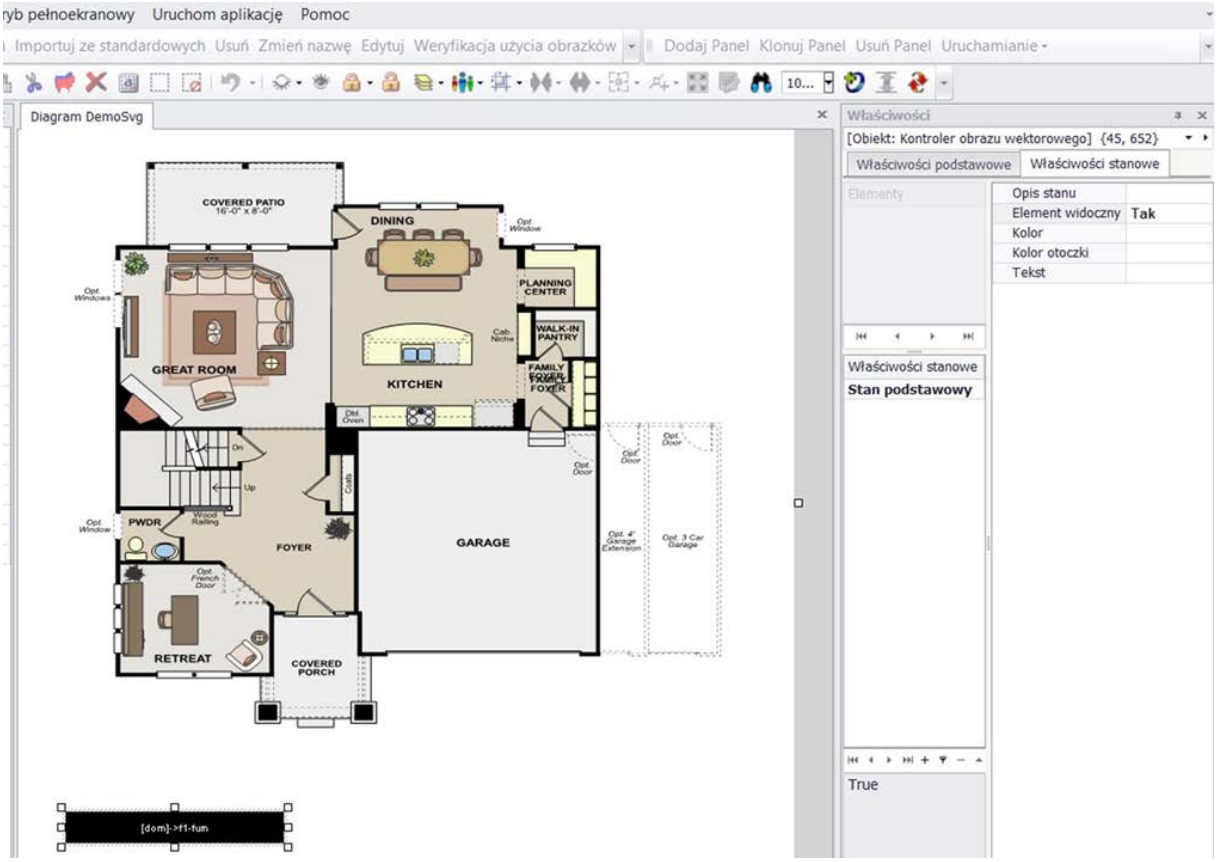
Obiekt Kontroler Obrazu Wektorowego

Funkcje obiektu

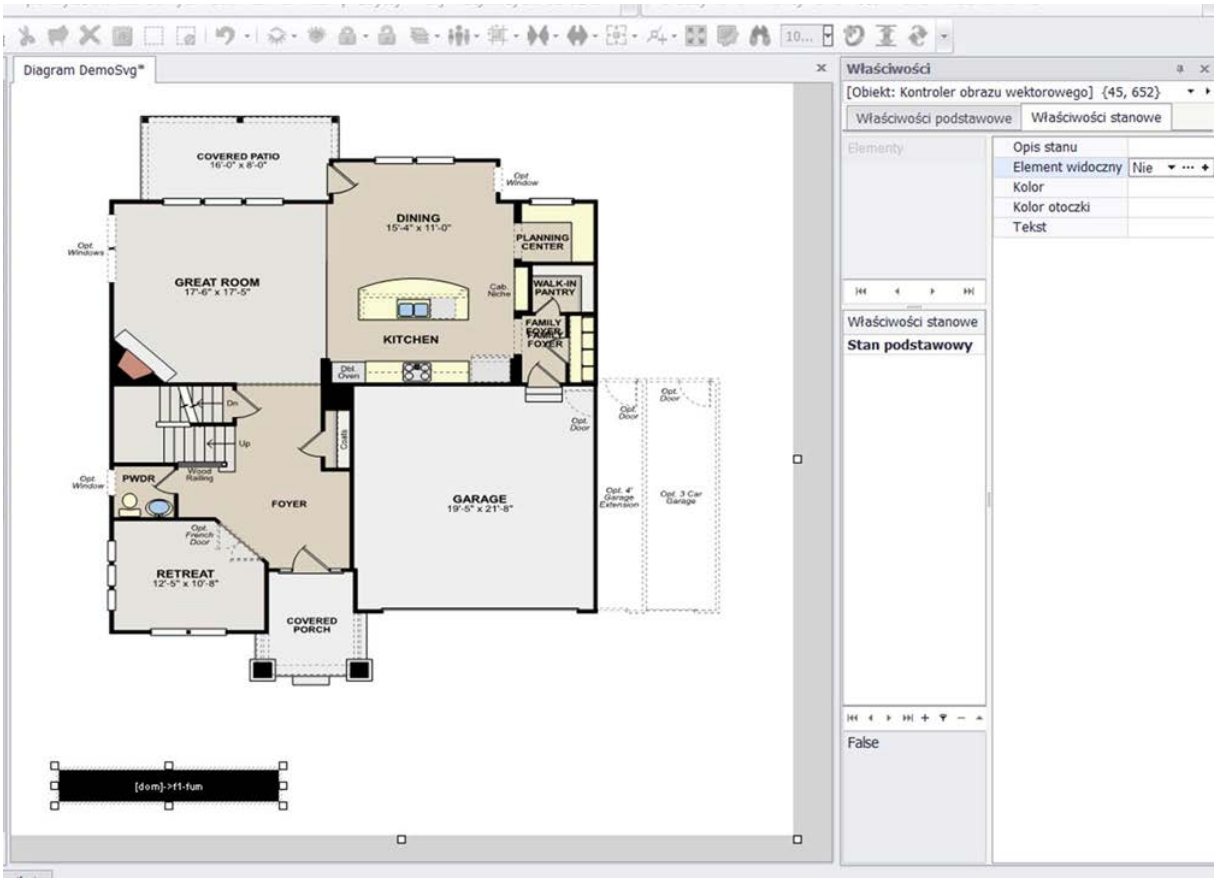
Obiekt*Kontroler obrazu wektorowego* pozwala na sterowanie wyglądem obrazu umieszczonego na diagramie. Może to być obraz zdefiniowany jako tło diagramu lub obraz wyświetlany przez obiekt typu *Obrazek*. Warunkiem działania obiektu jest to, żeby kontrolowany obraz był typu wektorowego SVG.

Połączenie pomiędzy obiektem kontrolera i obrazem wykonywane jest poprzez wskazanie nazwy obiektu *Obrazek* lub nazwy diagramu oraz określenie nazwy elementu umieszczonego wewnątrz kontrolowanego obrazu.

Kontrolować można takie atrybuty elementu obrazu jak: stan widoczności, kolor, tekst (dla elementów tekstowych).



Rys. Przykład użycia obiektu Kontroler Obrazu Wektorowego - sterowanie widocznością elementu wewnątrz obrazka.Cz.1z2



Rys. Przykład użycia obiektu Kontroler Obrazu Wektorowego - sterowanie widocznością elementu wewnątrz obrazka.Cz 2 z2

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Nazwa obiektu obrazu	Tekst	Określa jaki obraz będzie kontrolowany przez obiekt. Może to być nazwa dowolnego nazwanego obiektu typu Obrazek, umieszczonego na tym samym diagramie lub nazwa diagramu. Nazwa diagramu (lub brak nazwy) oznacza kontrolowanie obrazu umieszczonego w tle diagramu.

Identyfikator elementu obrazu	Tekst	Właściwość określa, który element obrazu wskazanego przez właściwość Nazwa obiektu obrazu będzie modyfikowany przez obiekt kontrolera. Może to być identyfikator pojedynczego elementu obrazu lub nazwa grupy elementów. W drugim przypadku pojedynczy obiekt kontrolera może modyfikować w identyczny sposób wiele elementów obrazu.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Element widoczny	Logiczna	Określa, czy kontrolowany element obrazu ma być widoczny.
Kolor	Kolor	Określa kolor kontrolowanego elementu obrazu.
Kolor otoczki	Kolor	Określa kolor otoczki kontrolowanego elementu obrazu.
Tekst	Tekst	Ustawia tekst w kontrolowanym elemencie. Element musi być typu tekstowego.

Obiekt Kontroler Wykresu

Funkcje obiektu

Obiekt *Kontroler wykresu* służy do sterowania pracą obiektów *Wykres.* Pozwala on na stworzenie obsługiwanego myszką interfejsu *o*bsługi wykresów. Obiekt ma postać paska narzędziowego – zestaw dostępnych przycisków oraz ich układ jest swobodnie parametryzowany.

Połącznie pomiędzy obiektem kontrolera i wykresu obywa się na zasadzie zgodności definicji właściwości *Grupa wykresów*. Możliwe jest jednoczesne podłączenie kilku obiektów *Wykres* – w tym przypadku wszystkie podłączone obiekty są sterowane w identyczny sposób.

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Grupa wykresów	Tekst	Określa nazwę grupy sterowanych obiektów Wykres. Obiekty sterowane muszą mieć identycznie ustawioną swoją nazwę grupy.
Przyciski	Typ wyliczeniowy wielokrotny: Przesuń w lewo, Przesuń w prawo, Przesuń w dół, Przesuń w górę, Powiększ w osi X, Pomniejsz w osi X, Powiększ w osi Y, Pomniejsz w osi Y, Cały zakres w lewo,	Właściwość określa, które funkcje sterujące (przyciski) mają być dostępne. Zestaw dostępnych funkcji w większości dotyczy zmiany pokazywanego zakresu czasu, zmian skali czasowej i wartościowej oraz pozycjonowania kursora. Przycisk *Czas aktualny* powoduje przestawienie wykresów na pokazywanie ostatnich danych. Przycisk *Domyślny zakres wartości* przełącza skalę wartości obiektów zgodnie z ustawieniami zapisanymi w definicji obiektów, natomiast przycisk *Stan początkowy* powoduje przestawienie wykresów w stan, w którym obiekty były bezpośrednio po otwarciu diagramu. Przycisk *Ustaw horyzont czasu* pozwala użytkownikowi wybrać dowolny zakres pokazywanego czasu. Tryb działania przycisków *Kursor w …* zależy od właściwości *Przyciągaj kursor* obiektu Wykres. Standardowo dotyczą one

	Cały zakres w prawo, Cały zakres w dół, Cały zakres w górę, Kursor w lewo/Poprzedni punkt, Kursor w prawo/Następny punkt, Kursor w dół/Poprzednia seria, Kursor w górę/Następna seria, Wyśrodkuj kursor, Czas aktualny, Domyślny zakres wartości, Stan początkowy, Ustaw horyzont czasu, Wyświetl legendę, Wyświetl pomoc, Włącz/Wyłącz kursor	swobodnego ruchu kursora. W trybie przyciągania służą do poruszania się po punktach serii lub wyboru serii.
Układ poziomy	Logiczna	Określa, czy przyciski są wyświetlane w układzie poziomym czy pionowym.
Dwie linie	Logiczna	Określa, czy przyciski są wyświetlane w układzie dwóch linii.

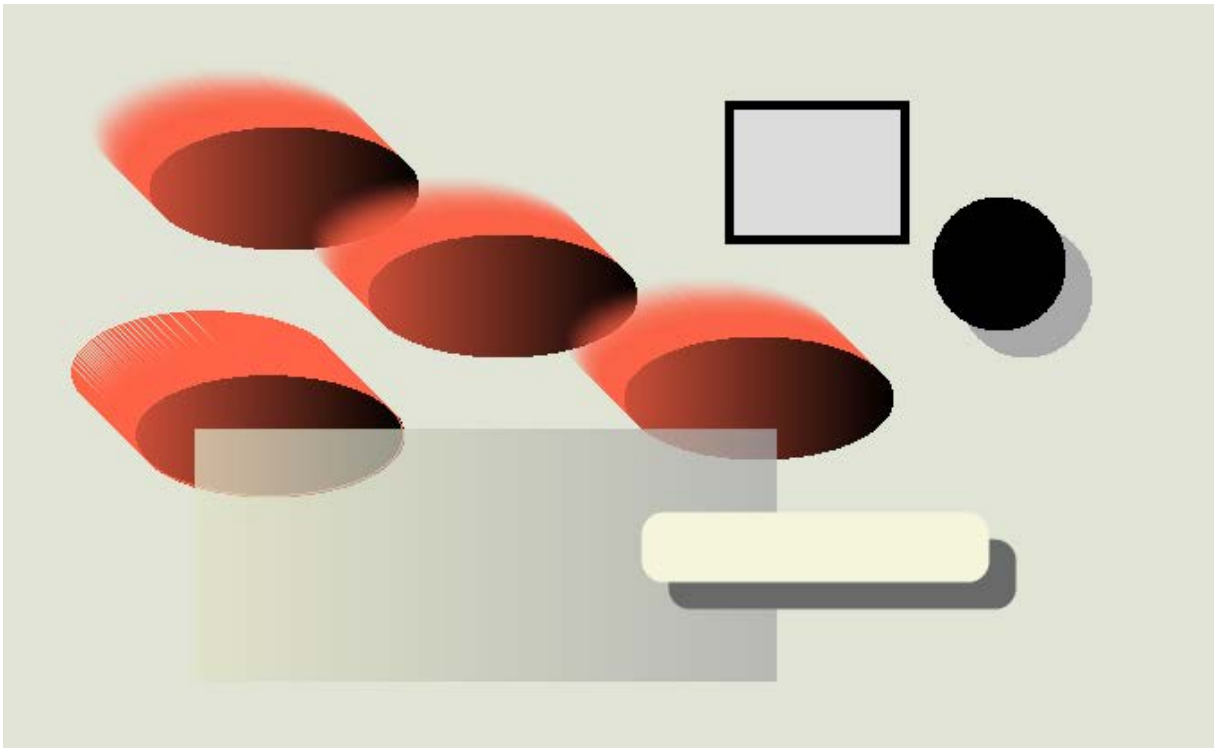
Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor tła	Kolor	Określa kolor tła przycisków.

Obiekt Kształt

Funkcje obiektu

Obiekt *Kształt* umożliwia wyświetlenie czterech podstawowych figur: prostokąt, kwadrat, koło i elipsa. Dla każdej figury można określić kolor wnętrza, grubość i kolor obwódki, głębokość i kolor cienia, głębokość i kolor wypukłości (smugi), kierunek i kolor gradientu oraz dla prostokąta i kwadratu - promień zaokrąglenia narożników.



Rys. Przykłady obiektu Kształt.

Obiekt *Kształt* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

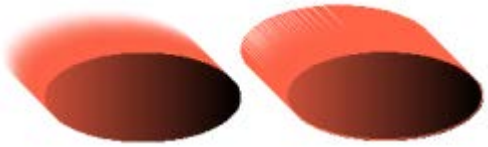
Zobacz: **Techniki_budowy_diagramów** > **Parametryzacja funkcji interaktywnych w obiektach pasywnych**

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

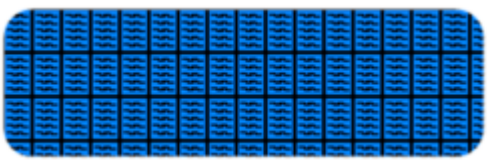
Właściwości podstawowe

Nazwa	Typ wartości	Opis
Typ kształtu	Typ wyliczeniowy: Prostokąt, Kwadrat, Elipsa, Koło	Określa typ rysowanej figury.
Kierunek gradientu	Typ wyliczeniowy: Poziomo, Pionowo, Ukośnie w górę, Ukośnie w dół	Określa sposób gradientowego wypełnienia tła rysowanego kształtu. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor gradientu we właściwości *Kolor gradientu*. 
Głębokość cienia	Liczba	Wartość większa od 0 powoduje wyświetlenie cienia kształtu. Cień rysowany jest z przesunięciem w dół i w prawo o podaną ilość pikseli. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor cienia we właściwości *Kolor cienia*.  Cień nie jest rysowany, jeżeli została ustawiona właściwość *Głębokość wypukłości*.
Głębokość wypukłości	Liczba	Wartość większa od 0 powoduje rysowanie kształtu z efektem wypukłości (rozmycia). Kolor wypukłości jest zmodyfikowanym kolorem podstawowym wypełnienia lub zmodyfikowanym kolorem

		otoczki (jeżeli otoczka jest rysowana). Wypukłość jest rozmyta - chyba że zostanie dodatkowo zastosowana otoczka. 
Kierunek wytłoczenia	Typ wyliczeniowy: Lewa góra, Prawa góra, Lewy dół, Prawy dół	Określa kierunek rysowania efektu wypukłości. 
Szerokość otoczki	Liczba	Określa szerokość (w pikselach) otoczki rysowanej wzdłuż krawędzi kształtu. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor otoczki we właściwości *Kolor otoczki*. 
Promień zaokrąglenia	Liczba	W przypadku kształtów prostokąt i kwadrat właściwość określa wyrażony w pikselach promień zaokrąglenia narożników. 
Kąt początkowy	Liczba	W przypadku kształtów koło i elipsa właściwość określa początek rysowania wycinka koła (elipsy). Ma znaczenie tylko wtedy, gdy właściwość Kąt rozwarcia ma wartość mniejszą od 360. Poniższe przykłady pokazują wycinki o kącie rozwarcia 145 stopni i kątach początkowych 0 i 90 stopni. 

Kąt rozwarcia	Liczba	W przypadku kształtów koło i elipsa właściwość określa kąt rysowanego wycinka koła (elipsy). Wartość 360 oznacza rysowanie pełnego koła (elipsy). Poniższe przykłady pokazują wycinki o kącie początkowym 0 i kątach rozwarcia 45 i 315 stopni. 
Sama otoczka	Logiczna	Określa czy rysowana ma być tylko zewnętrzna otoczka obiektu. W szczególności, właściwość pozwala na rysowanie łuków. 

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor	Kolor	Określa podstawowy kolor wypełnienia kształtu. Wybranie koloru *Transparent* i ustawienie niezerowej szerokości otoczki pozwala na rysowanie jedynie konturu kształtu.
Kolor gradientu	Kolor	Określa kolor stosowany do gradientowego wypełnienia tła kształtu. Kolor tła będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu. Sposób wypełniania określa właściwość *Kierunek gradientu.*
Obrazek tła	Nazwa obrazka	Nazwa obrazka wyświetlanego jako wypełnienie tła kształtu. Obrazek zawsze wyświetlany jest w trybie kafelków. 
Kolor otoczki	Kolor	Określa kolor otoczki.

Styl otoczki	Typ wyliczeniowy: Brak, Kreska, Kropka, Kreska-Kropka, Kreska-Kropka-Kropka	Określa sposób rysowania otoczki. Wartość *Brak* oznacza rysowanie otoczki pełną linią.
Kolor cienia	Kolor	Określa kolor cienia kształtu.

Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Linia

Funkcje obiektu

Obiekt Linia pozwala na tworzenie pojedynczych odcinków linii lub dowolnych płaskich figur geometrycznych złożonych z sekwencji odcinków. Kształt obiektu jest definiowany serią koordynat poszczególnych wierzchołków linii. Można definiować kształt końcówek linii oraz stosować linie przerywane. Możliwe jest też tworzenie płynnie wygiętych odcinków w trybie krzywych Beziera.



Obiekt *Linia* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

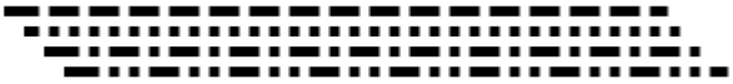
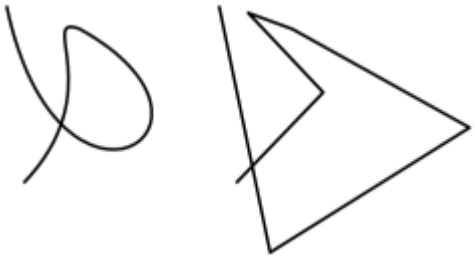
Zobacz: ***Techniki_budowy_diagramów > Parametryzacja funkcji interaktywnych w obiektach pasywnych***

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wierzchołki	Lista współrzędnych	Lista par X,Y współrzędnych kolejnych wierzchołków linii. Współrzędne X i Y pozycji wierzchołka wyrażone są w pikselach i liczone względem lewego górnego narożnika diagramu. Kolejne pary wierzchołków są rozdzielone znakiem średnika.
Grubość	Liczba	Określa grubość linii w pikselach.

Zaokrąglone złączenia	Logiczna	Określa, czy należy zaokrąglać połączenia pomiędzy poszczególnymi odcinkami linii. 
Styl początku	Typ wyliczeniowy: Brak, Płaski, Okrągły, Trójkątny, Strzałka	Określa styl rysowania pierwszego wierzchołka linii. 
Styl końca	Typ wyliczeniowy: Brak, Płaski, Okrągły, Trójkątny, Strzałka	Określa styl rysowania ostatniego wierzchołka linii.
Styl pasków	Typ wyliczeniowy: Brak, Kreska, Kropka, Kreska-Kropka, Kreska-Kropka-Kropka	Pozwala włączyć tryb rysowania linii przerywanych i określa sposób tworzenia przerw. 
Typ linii	Typ wyliczeniowy: Łamana, Krzywa Beziera	Określa czy linia składa się z prostych odcinków czy z krzywych Beziera. W trybie Beziera każdy odcinek jest definiowany przez cztery wierzchołki. Pierwszy i czwarty określają punkty końcowe. Drugi i trzeci służą do kontroli krzywizny. Poniższy obrazek pokazuje linię o tych samych wierzchołkach, ale różnym typie. 

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor	Kolor	Określa kolor linii.
Kolor pasków	Kolor	Określa kolor przerw w trybie linii przerywanej. Kolor *Transparent* pozwala na tworzenie przerw przezroczystych. 
Offset wypełnienia	Liczba	W trybie linii przerywanej pozwala na przesunięcie wzorca rysowania, co powoduje efekt animacji ruchu. Jednostką przesunięcia jest szerokość przerwy pomiędzy fragmentami linii. Przykład: =Variable(Counter)/10.0 Wyrażenie animuje ruch linii na bazie zmian zmiennej systemowej *Counter*. Dzielenie (ewentualnie mnożenie) wartości zmiennej służy do sterowania szybkością i płynnością animacji.

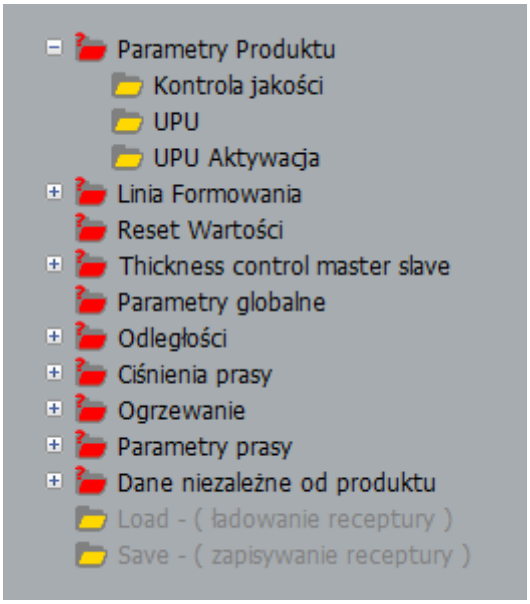
Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Lista rozwijana

Funkcje obiektu

Obiekt *Lista rozwijana* służy do wizualizacji danych w postaci hierarchicznej listy rozwijanej. W podstawowym zastosowaniu używany jest do tworzenia nawigacji sterującej wyborem wyświetlanego diagramu.



Z każdym węzłem listy można być skojarzona akcja operatorska, która wykonywana jest w momencie kliknięcia w obszarze węzła.

Struktura listy może być określona bezpośrednio w definicji obiektu. Dostępny jest też tryb pobierania struktury listy z zewnętrznego pliku. W tym przypadku plik może być tworzony dynamicznie przez zewnętrzne oprogramowanie lub skrypt. W takim zastosowaniu obiekt może być użyty, na przykład, do prezentacji hierarchicznych informacji pobieranych z bazy danych, opisujących wykonane produkty.

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Bezpośrednia struktura elementów	Tekst	Opis struktury listy rozwijanej w postaci tekstowej. Struktura opisana jest tekstem XML, który zawiera definicje właściwości wszystkich węzłów listy oraz ich wzajemną hierarchię. Struktura może być edytowana bezpośrednio, ale zalecanym trybem jest uruchomienie interakcyjnego edytora struktury, który można

		otworzyć przyciskiem znajdującym się w panelu edycji definicji właściwości.
Plik struktury elementów	Tekst	Nazwa pliku zawierającego opis struktury. Użycie tej właściwości stanowi alternatywę dla bezpośredniej definicji. Zawartość pliku ma identyczną postać jak opis we właściwości *Bezpośrednia struktura elementów*. Zawartość pliku może być edytowana przy pomocy niezależnych narzędzi, tworzona w sposób programowy lub zapisana w oknie edytora struktury.
Rozmiar czcionki	Liczba	Określa rozmiar czcionki używanej do wyświetlenia opisów węzłów.
Czcionka	*Nazwa czcionki*	*Systemowa nazwa czcionki używanej do wyświetlania opisów węzłów.*
Styl czcionki	*Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie*	*Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.*
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych węzłów wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, gdy rozmiar obiektu ulega zmianie w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba węzłów - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.
Margines tekstu [%]	Liczba	Podawana w procentach wielkość marginesu pozostawionego wokół tekstu. Właściwość pozwala sterować odstępami pomiędzy opisami poszczególnych węzłów.
Pokazuj plus/minus	Logiczna	Określa czy ikonka plis/minus używana do rozwijania i zwijania węzłów ma być pokazywana.
Pokazuj ikony	Logiczna	Określa czy wyświetlać ikony skojarzone z węzłami listy
Pełne tło elementów	Logiczna	Określa sposób kolorowania tła opisów węzłów. Pełne tło oznacza wypełnienie całej linii wiersza. W przeciwnym wypadku tło jest wypełniane tylko na długości tekstu.

Akcja przy podwójnym kliknięciu	Logiczna	Określa rodzaj kliknięcia na opisie węzła, które wyzwala wykonanie skojarzonej z węzłem akcji.
Zwiń/rozwiń przy podwójnym kliknięciu	Logiczna	Określa czy rozwinięcie węzła wymaga pojedynczego czy podwójnego kliknięcia na opisie węzła. Nie ma znaczenia dla rozwijania poprzez ikonkę plus/minus, dla której zawsze stosowane jest pojedyncze kliknięcie.

Właściwości stanowe

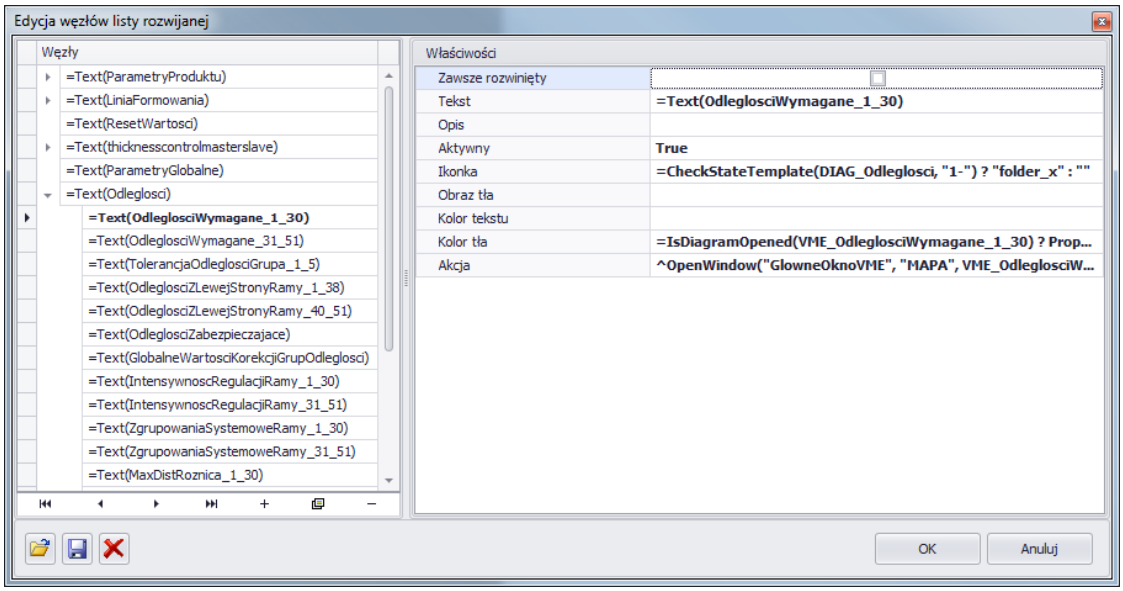
Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła obszaru listy.
Kolor tekstu elementu	Kolor	Określa domyślny kolor tekstów opisujących węzły.
Kolor tła elementu	Kolor	Określa domyślny kolor tła tekstów opisujących węzły. Kolorowanie tła można wyłączyć poprzez wybór koloru *Transparent*.
Kolor tekstu wybranego elementu	Kolor	Określa kolor tekstu opisującego aktualnie wybrany węzeł.
Kolor tła wybranego elementu	Kolor	Określa kolor tła tekstu opisującego aktualnie wybrany węzeł. Kolorowanie tła można wyłączyć poprzez wybór koloru *Transparent*.
Kolor tekstu nieaktywnego elementu	Kolor	Określa kolor tekstów opisujących nieaktywne węzły.
Kolor tła nieaktywnego elementu	Kolor	Określa kolor tła tekstów opisujących nieaktywne węzły. Kolorowanie tła można wyłączyć poprzez wybór koloru *Transparent*.
Ikona elementów	Tekst	Określa domyślną ikonę skojarzoną z węzłami listy.
Obrazek tła elementów	Tekst	Określa domyślny obraz używany do wypełnienia tła węzłów.

Zdarzenia

Obiekt nie obsługuje standardowych zdarzeń. Wykonywane akcje operatorskie są definiowane w ramach właściwości poszczególnych węzłów listy.

Edytor struktury listy

Edytor struktury listy otwierany jest za pośrednictwem panelu edycyjnego właściwości *Bezpośrednia struktura elementów.*



Lewa strona okna wyświetla listę zdefiniowanych węzłów. Zintegrowany pasek narzędziowy pozwala na tworzenie węzłów i zmianę ich kolejności. Do zmiany hierarchii węzłów należy stosować technikę pociągania węzłów. Po prawej stronie okna znajduje się panel służący do definiowania wartości właściwości wybranego na liście węzła.

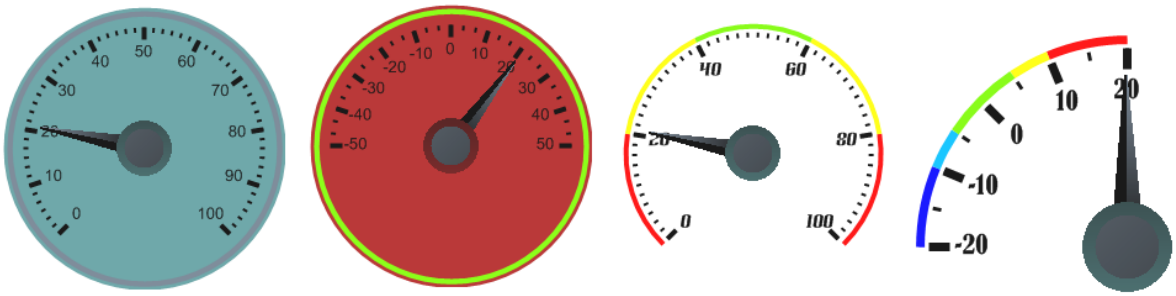
Nazwa	Typ wartości	Opis

Zawsze rozwinięty	Logiczna	Określa czy węzeł jest zawsze rozwinięty.
Tekst	Tekst	Określa tekst wyświetlany w wierszu węzła.
Opis	Tekst	Określa dodatkowy opis węzła. Jest wyświetlany w dymku otwieranym po najechaniu kursorem w obszar węzła.
Aktywny	Logiczna	Określa czy węzeł jest aktywny.
Ikonka	Tekst	Określa ikonkę skojarzoną z węzłem.
Obraz tła	Tekst	Określa obraz używany do wypełnienia tła węzła.
Kolor tekstu	Kolor	Określa indywidualny kolor tekstu opisującego węzeł.
Kolor tła	Kolor	Określa indywidualny kolor tła tekstu opisującego węzeł.
Akcja	Tekst	Określa akcję operatorską wykonywaną po kliknięciu na węźle.

Obiekt Miernik

Funkcje obiektu

Obiekt *Miernik* służy do prezentacji liczbowych wartości pomiarów w postaci wskazań zegara wskazówkowego. Dla każdego miernika można zdefiniować: zakres wartości, liczbę podziałek głównych i podpodziałek, kąty początku i końca wyświetlania podziałki. Możliwe jest wyświetlanie limitów (LL, L, H, HH) z indywidualnie ustawianym kolorem każdego limitu.



Rys. Przykłady obiektu Miernik.



Rys. Przykłady zastosowania obiektu Miernik w połączeniu z obiektem Obraz.

Obiekt *Miernik* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

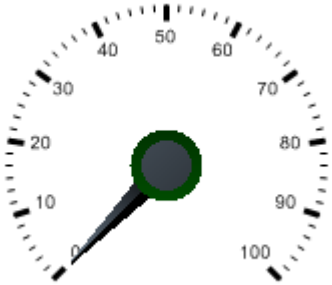
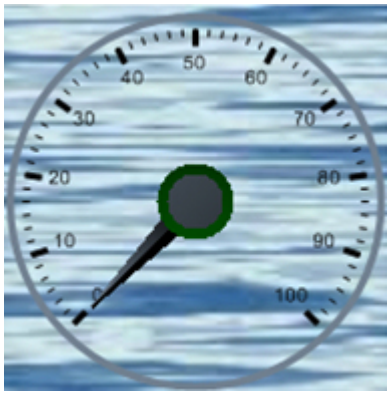
Zobacz: ***Parametryzacja funkcji interaktywnych w obiektach pasywnych (Asix.Evo_Techniki_budowy_diagramów.PDF/CHM)***

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

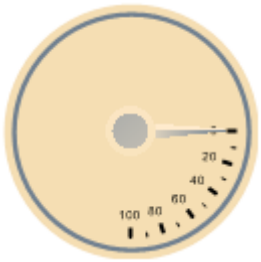
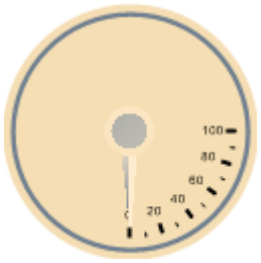
Właściwości podstawowe

Nazwa	Typ wartości	Opis

Bez tła	Logiczna	Właściwość pozwala na wyświetlenie miernika w uproszczonej postaci, w której wyświetlana jest tylko skala i wskazówka miernika. Tryb ten przeznaczony jest do tworzenia mierników z dwoma wskazówkami poprzez nałożenie na siebie dwóch mierników, z których jeden jest wyświetlany w trybie *Bez tła*.  Skalę można usunąć poprzez ustawienie koloru *Transparent* we właściwościach *Kolor podziałki* i *Kolor czcionki*.
Obrazek tła	Nazwa obrazka	Określa nazwę obrazka, który zostanie wyświetlony jako tło miernika. 
Pokaż limity	Logiczna	Określa, czy na skali miernika należy zaznaczyć osobnymi kolorami przedziały limitów.

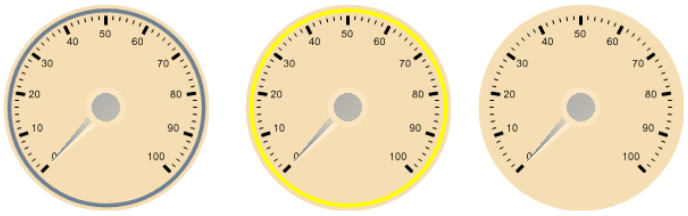
		
Pokaż limity w tle	Logiczna	Określa, czy przedziały limitów mają być zaznaczane na całym tle miernika. 
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania wartości podziałek skali miernika.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Format liczb podziałki	Tekst	Określa sposób formatowania liczb wyświetlanych na podziałce, np. *f1* oznacza wyświetlanie z jednym miejscem po przecinku.
Liczba podziałek	Liczba	Określa liczbę podziałek głównych na skali miernika.

Liczba podpodziałek	Liczba	Określą liczbę podpodziałek pomiędzy poszczególnymi podziałkami głównymi.
Wartość	Liczba	Określa aktualne położenie wskazówki miernika. Wartość właściwości powinna być liczbą z zakresu określonego przez właściwości *Wartość minimalna* i *Wartość maksymalna*.
Wartość minimalna	Liczba	Dolny limit zakresu skali miernika.
Wartość maksymalna	Liczba	Górny limit zakresu skali miernika.
Wartość LL	Liczba	Wartość limitu LL (minimum krytyczne) skali miernika.
Wartość L	Liczba	Wartość limitu L (minimum) skali miernika.
Wartość H	Liczba	Wartość limitu H (maksimum) skali miernika.
Wartość HH	Liczba	Wartość limitu HH (maksimum krytyczne) skali miernika.

Kąt początkowy	Liczba	Określa położenie początku skali miernika wyrażone jako kąt liczony względem poziomego (w prawo) położenia wskazówki. Koniec skali określany jest właściwością *Kąt końcowy*. Wzajemna zależność obu właściwości kątów skali decyduje o kierunku obrotu wskazówki. W razie konieczności można stosować kąty większe od 360 stopni. Kąt początku 0, Kąt końca 90:  Kąt początku 90, Kąt końca 0:  Kąt początku 90, Kąt końca 360:  Kąt początku 135, Kąt końca 405: 

Kąt końcowy	Liczba	Określa położenie końca skali miernika.
Kolor wartości poprawnej	Kolor	Określa kolor oznaczenia przedziału wartości normalnych.
Kolor LL	Kolor	Określa kolor oznaczenia przedziału wartości LL (minimum krytycznego).
Kolor L	Kolor	Określa kolor oznaczenia przedziału wartości L (minimum).
Kolor H	Kolor	Określa kolor oznaczenia przedziału wartości H (maksimum).
Kolor HH	Kolor	Określa kolor oznaczenia przedziału wartości HH (maksimum krytycznego).

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor wskazówki	Kolor	Określa kolor wskazówki.
Kolor tła	Kolor	Określa kolor tła miernika.
Kolor otoczki	Kolor	Określa kolor otoczki wokół miernika. Kolor *Transparent* pozwala wyłączyć wyświetlanie otoczki. 
Kolor podziałki	Kolor	Określa kolor podziałki na skali miernika. Kolor *Transparent* pozwala wyłączyć wyświetlanie podziałki.
Kolor czcionki	Kolor	Określa kolor opisów wartości na skali miernika. Kolor *Transparent* pozwala wyłączyć wyświetlanie wartości podziałek.

Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Obraz

Funkcje obiektu

Obiekt *Obraz* służy do wyświetlania jednego z obrazków znajdujących się w bibliotece obrazków aplikacji. Obsługiwane są obrazki typu BMP, JPG, GIF, PNG, TIFF, WMF, EMF, ICO, SVG. Można również używać obrazków w postaci animowanych plików GIF.



Rys. Przykłady obiektu Obraz (obrotowy wentylator jest animowanym gifem).

Wyświetlany obrazek może być skalowany z zachowaniem proporcji lub bez zachowania proporcji. Na obrazku można dokonywać również szereg transformacji: obroty (o 90, 180, 270 stopni), odbicia lustrzane, zmiana barwy, jasności i nasycenia.

Obiekt *Obraz* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

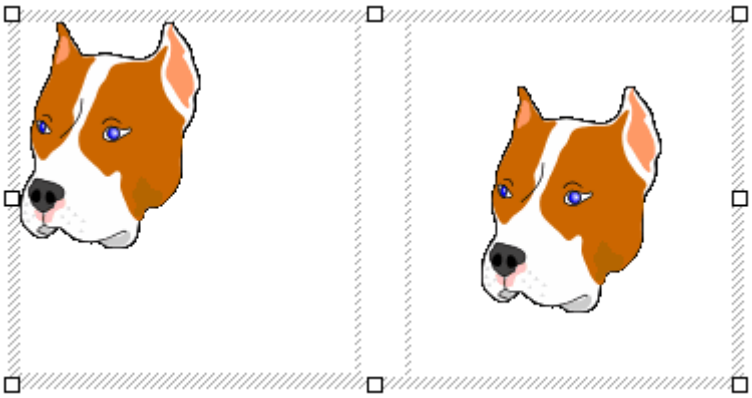
Zobacz: ***Techniki_budowy_diagramów > Parametryzacja funkcji interaktywnych w obiektach pasywnych***

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Skalowanie	Logiczna	Określa, czy obrazek będzie dopasowywany do zmian rozmiaru obiektu. Dotyczy to także przeskalowania obiektu w wyniku zmiany rozmiaru diagramu.  Wyłączenie skalowania powoduje, że obrazek wyświetlany jest zawsze w oryginalnym rozmiarze i może zostać obcięty z lewej lub dolnej strony.
Zachowanie proporcji	Logiczna	Określa, czy proces skalowania obrazka do rozmiaru obiektu zachowuje oryginalne proporcje obrazka.

		
Wyrównanie obrazu	Typ wyliczeniowy: Brak, Do góry, Do dołu, Do lewej, Do prawej	Określa w jaki sposób obrazek ma być umieszczony w ramach obszaru obiektu.  Właściwość ma znaczenie tylko wtedy, gdy nie jest włączony tryb skalowania obrazków bez zachowania proporcji. Brak wyrównania do krawędzi oznacza wyśrodkowanie obrazka.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Nazwa obrazu	Nazwa obrazka	Nazwa wyświetlanego obrazka. Obrazek musi znajdować się w bibliotece obrazków aplikacji.
Kolor przekreślenia	Kolor	Zdefiniowanie koloru innego niż *Transparent* powoduje narysowanie kreski przekreślającej obrazek.

		
Obrót	Typ wyliczeniowy: Brak, 90, 180, 270	Określa wartość obrotu obrazka.  W przypadku wyświetlenia obrazka w formacie SVG możliwe jest precyzyjne określenie kąta obrotu we właściwości Obrót SVG.
Odbicie lustrzane	Typ wyliczeniowy: Brak, X, Y, XY	Określa, czy obrazek będzie przekształcony poprzez odbicie lustrzane względem wybranej osi. 
Obrót SVG	Liczba	Określa obrót obrazka w prawo o kąt podany w stopniach. Właściwość ma zastosowanie tylko dla obrazków w formacie SVG.
Barwa	Liczba <-1,1>	Określa stopień modyfikacji barwy obrazka. Wartość 0 oznacza obrazek w wersji oryginalnej. 

Nasycenie	Liczba <-1,1>	Określa stopień modyfikacji nasycenia kolorów obrazka. Wartość 0 oznacza obrazek w wersji oryginalnej. 
Jasność	Liczba <-1,1>	Określa stopień modyfikacji jasności obrazka. Wartość 0 oznacza obrazek w wersji oryginalnej. 

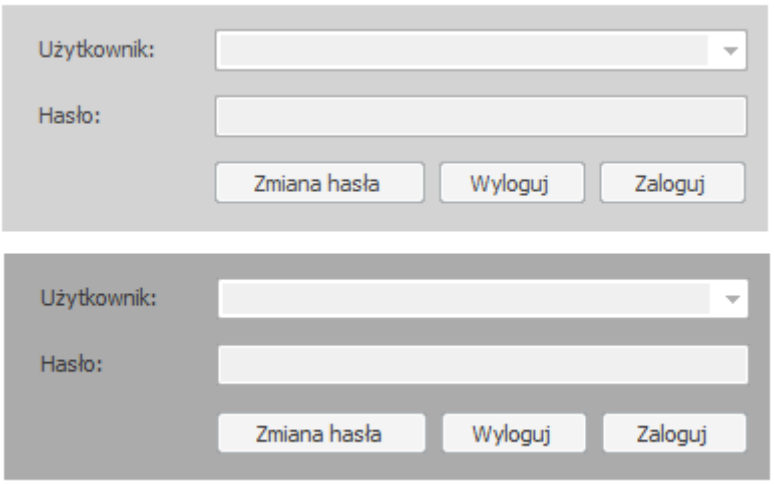
Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Pole autoryzacji

Funkcje obiektu

Obiekt *Pole autoryzacji* jest obiektem służącym do wykonywania operacji logowania, wylogowywania oraz zmiany haseł użytkowników. Konfigurując taki obiekt można określić kolor jego tła oraz zdecydować, czy widoczne mają być przyciski wylogowania i zmiany hasła.



Rys. Przykłady obiektu Pole autoryzacji.

Obiekt pozwala na wstawienie funkcji związanych z zarządzaniem użytkownikami bezpośrednio na diagram. Alternatywą jest używanie okienka *Panelu Kontrolnego,* ewentualnie stworzenie własnego interfejsu logowania z użyciem akcji operatorskich *Login* i *Logout*.

Obiekt *Pole autoryzacji* jest obiektem typu kontrolkowego.

Zobacz też

- ***Akcje_operatorskie*** **>** **akcja** ***Login***, ***Logout***

Właściwości

Zobacz: ***Elementy_wizualizacji*** **>** ***Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Zamknij bieżące okno po zalogowaniu	Logiczna	Wybranie wartości Tak spowoduje, że po poprawnym zalogowaniu zostanie zamknięte okno, w ramach którego obiekt został wyświetlony. Funkcję automatycznego zamykania należy stosować tylko wtedy, gdy obiekt został umieszczony na dedykowanym oknie przeznaczonym do logowania użytkownika (np. na diagramie otwieranym akcją *OpenDiagram*).

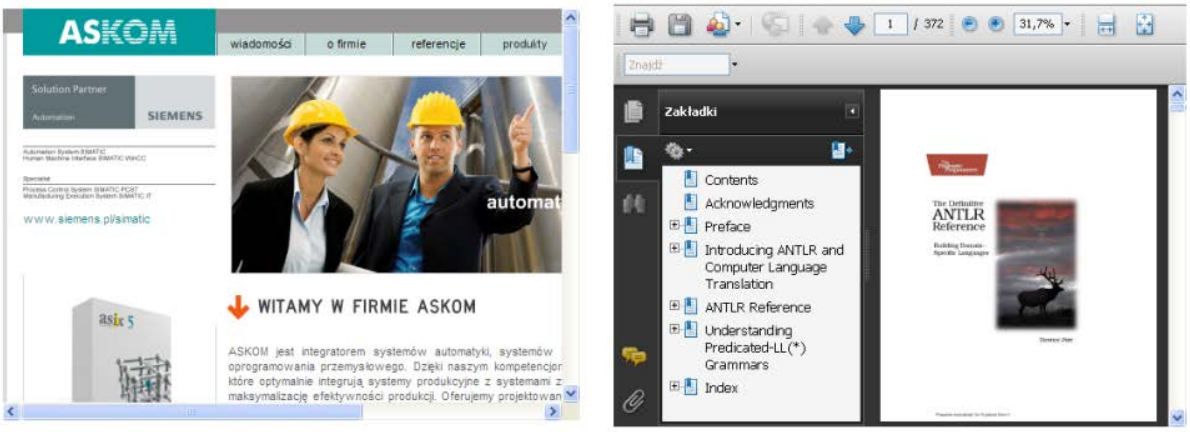
Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła obiektu.
Pokazuj przycisk zmiany hasła	Logiczna	Określa, czy przycisk Zmiana hasła ma być wyświetlany.
Pokazuj przycisk wylogowania	Logiczna	Określa, czy przycisk Wyloguj ma być wyświetlany.

Obiekt Przeglądarka

Funkcje obiektu

Obiekt *Przeglądarka* jest uniwersalnym obiektem służącym do wyświetlania na diagramie zawartości wskazanej strony WWW lub dowolnych dokumentów obsługiwanych przez przeglądarkę IE (np. PDF, niektóre dokumenty pakietu Office, filmy, itp.). Jednym z zastosowań tego obiektu może być wyświetlanie obrazu z kamer internetowych lub przemysłowych, jeżeli źródło takiego obrazu jest udostępnione w postaci serwisu lub strony HTML. Obiekt pozwala też na wyświetlenie zawartości podanej bezpośrednio w definicji obiektu.



Rys. Przykłady obiektu Przeglądarka.

Obiekt *Przeglądarka* jest obiektem typu kontrolkowego.

> ***UWAGA:***
> W przypadku, gdy aplikacja jest uruchamiana w trybie przeglądarkowym, to użyty w aplikacji obiekt typu *Przeglądarka* będzie działał tylko wtedy, gdy zostanie użyty na diagramie, który zostanie otwarty poza oknem głównym przeglądarki w niezależnym oknie aplikacji Asix.Evo (otwartym nie w przeglądarce).

Właściwości

Zobacz: *Elementy_wizualizacji* > *Standardowe właściwości obiektów*

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Zezwolenie na nawigację	Logiczna	Określa, czy użytkownik może zmieniać stronę poprzez użycie łączy obecnych na stronie wyświetlanej.
Pokazuj paski przewijania	Logiczna	Określa, czy w obiekcie przeglądarki będą pokazywane suwaki.
Zezwolenie na okienka dialogowe	Logiczna	Określa, czy w obiekt przeglądarki może wyświetlać okienka dialogowe. Zezwolenie to może być konieczne do poprawnej pracy niektórych witryn. W szczególności dotyczy to wyświetlania raportów tworzonych w usłudze Microsoft Reporting Services.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Url	Tekst	Adres Url strony html lub ścieżka do lokalnego pliku w formacie obsługiwanym przez przeglądarkę IE. Przykład: http://www.asix.com.pl - adres strony html c:\manuals\m1.pdf - wskazanie pliku na dysku lokalnym w formacie pdf
Zawartość	Tekst	Pozwala na zdefiniowanie wyświetlanej zawartości bezpośrednio w definicji obiektu. Wartość właściwości może być wyliczana wyrażeniem lub być ustawiana przez skrypt przy pomocy akcji operatorskiej *SetProperty*. Właściwość Zawartość ma pierwszeństwo przed właściwością *Url*. Przykład: Trochę tekstu - wyświetlenie tekstu wytłuszczoną czcionką

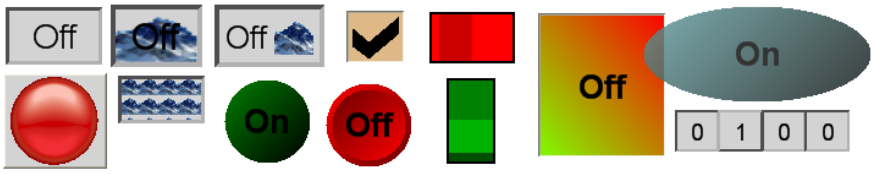
Obiekt Przycisk

Funkcje obiektu

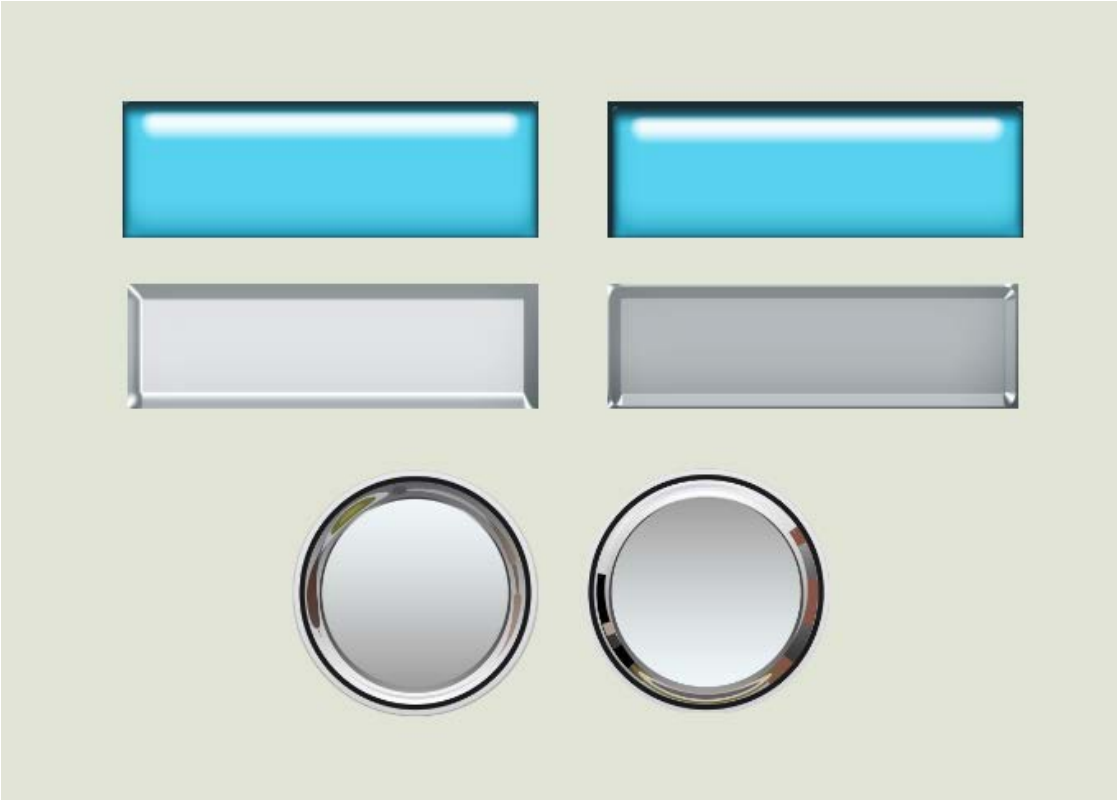
Obiekt *Przycisk* służy do wizualizacji przycisków pozwalających na realizację sterowania dwustanowego. Dostępne są różne typy i style przycisków -kolorystyka i efekty wizualne są swobodnie definiowane. Przeznaczenie przycisku może być opisane napisem lub/i obrazkiem.

Efektem działania przycisku jest ustawienie zmiennej sterowanej lub wykonanie dowolnej akcji operatorskiej. Dostępny jest tryb pracy "z powtarzaniem”, w którym sterowanie (akcja) jest wykonywane cyklicznie przez okres przytrzymywania przycisku w stanie wciśniętym.

Istnieje możliwość zdefiniowania skrótu klawiszowego, którym można aktywować funkcje przycisku. Dodatkowo można grupować przyciski w ramach tzw. grupy sterowania, dzięki czemu można zrealizować synchroniczne sterowanie poszczególnych bitów zmiennej.



Rys. Przykłady obiektu Przycisk.



Rys. Przykłady zastosowania obiektu Przycisk w połączeniu z obrazkiem.

Właściwości

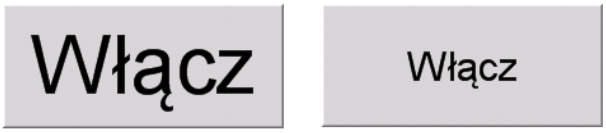
Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

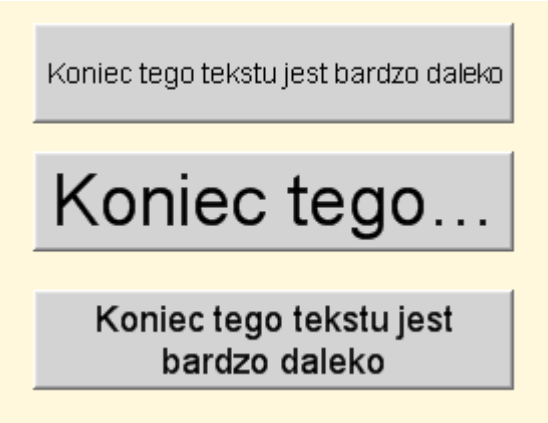
Właściwości podstawowe

Nazwa	Typ wartości	Opis
Sterowanie bitowe	Logiczna	Określa sposób wykonywania operacji sterujących wykonywanych przez obiekt. W przypadku wartości *Nie*, wartości sterujące podane we właściwościach *Wartość włączenia, Wartość wyłączenia* i *Wartość końca powtarzania* są wysyłane do zmiennej sterowanej bez żadnej modyfikacji. Jeżeli sterowanie bitowe jest włączone, to w przypadku wartości włączenia i końca powtarzania operacja sterująca polega na ustawieniu

		niezerowych bitów maski, a w przypadku wartości wyłączenia nastąpi wyzerowanie wszystkich bitów zmiennej sterowanej zgodnie z podaną maską bitową wyłączenia.
Typ przycisku	Typ wyliczeniowy: Przycisk prostokątny 3D, Przycisk eliptyczny 3D, Przełącznik poziomy, Przełącznik pionowy, Standardowy przycisk Windows, Pole wyboru	Określa ogólny wygląd i zachowanie przycisku.
Płaski	Logiczna	Określa, czy przycisk ma mieć postać płaską czy trójwymiarową. Przyciski trójwymiarowe obsługują animację efektów wciskania i puszczania przycisków.
Tryb przełącznika	Logiczna	Określa tryb pracy przycisku. W trybie przełącznika, obiekt w momencie kliknięcia myszką zmienia stan na przeciwny (wykonuje odpowiednią operację sterującą i zdarzenie). W trybie zwykłym przycisk z założenia działa jako przycisk jednopozycyjny - w momencie kliknięcia obiekt wykonuje operacje związane z przejściem do stanu przeciwnego i z powrotem do stanu początkowego.
Limit czasu odpowiedzi [ms]	Liczba	Właściwość używana tylko wtedy, gdy *Tryb przełącznika* jest równy *true*. Jeżeli obiekt nie jest w trakcie wyboru nowej wartości, to standardowo za jego stan odpowiada wartość właściwości *Włączony*. Jeżeli właściwość *Limit czasu odpowiedzi* jest ustawiona, to bezpośrednio po wykonaniu operacji sterującej, przez podany czas podtrzymywany jest stan przycisku, wynikający z wykonanej operacji - niezależnie od wartości właściwości *Włączony*. Ma to celu uniknięcie ruchów przycisku w sytuacji, kiedy czas wykonania operacji sterującej i reakcji obiektu jest zauważalnie długi.

Kierunek gradientu	Typ wyliczeniowy: Poziomo, Pionowo, Ukośnie w górę, Ukośnie w dół	Określa sposób gradientowego wypełnienia tła przycisku. Właściwość ma znaczenie tylko dla niektórych typów przycisków oraz wymaga ustawienia *właściwości Kolor gradientu wyłączenia* i *Kolor gradientu włączenia*.
Głębokość cienia	Liczba	Wartość większa od 0 powoduje wyświetlenie cienia przycisku (dotyczy tylko niektórych typów przycisków). Cień rysowany jest z przesunięciem w dół i w lewo o podaną ilość pikseli. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor cienia we właściwości *Kolor cienia*.
Szerokość otoczki	Liczba	Określa szerokość (w pikselach) otoczki rysowanej wokół przycisku. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor otoczki we właściwościach Kolor otoczki włączenia i *Kolor otoczki wyłączenia*.
Tryb obrazka	Typ wyliczeniowy: Kafelki, Rozciąganie, Wyśrodkowany, Skalowany przed tekstem, Skalowany za tekstem, Przylegający z góry, Przylegający z dołu, Przylegający z lewej, Przylegający z prawej, Skalowany w tle	Określa sposób, w jaki pozycjonowany jest obrazek wyświetlany na przycisku.
Obrazek widoczny	Logiczna	Określa, czy należy wyświetlać obrazek na tle przycisku.
Promień zaokrąglenia	Liczba	Określa promień zaokrąglenia narożników przycisków. Dotyczy tylko płaskich prostokątnych przycisków.
Wyrównanie tekstu w poziomie	Typ wyliczeniowy: Do lewej, Wyśrodkowanie, Do prawej	Określa sposób wyrównywania w poziomie tekstów opisujących przycisk.

Wyrównanie tekstu w pionie	Typ wyliczeniowy: Do góry, Wyśrodkowanie, Do dołu	Określa sposób wyrównywania w pionie tekstów opisujących przycisk.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania tekstów na przycisku.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Rozmiar czcionki	Liczba	Określa rozmiar czcionki używanej do wyświetlenia tekstów opisujących przycisk. Wartość 0 oznacza automatyczny dobór rozmiaru.
Margines tekstu [%]	Liczba	Określa wielkość marginesu uwzględnianego przy wyświetlaniu opisu przycisku. Właściwość ma znaczenie tylko wtedy, gdy włączony jest automatyczny dobór rozmiaru czcionki. 
Liczba linii	Liczba	Steruje sposobem dopasowania wysokości czcionki do długości tekstu. Wartość 0 oznacza, że należy tak zmniejszyć czcionkę, aby cały tekst się zmieścił w obszarze obiektu. Skutkiem tego, w zależności od długości tekstu, może zostać użyta inna czcionka. Wartość większa od 0 oznacza, że o wysokości czcionki decyduje wyłącznie rozmiar obiektu. W razie konieczności tekst może zostać obcięty lub podzielony na kilka linii.

		
Sterowanie natychmiastowe	Logiczna	Określa, czy w momencie użycia przycisku, wartość sterująca jest automatycznie wysyłana do zmiennej sterującej. Jeżeli nie, to do realizacji sterowania wymagane jest użycie akcji operatorskiej *SendControls*.
Z powtarzaniem	Logiczna	Określa, czy przycisk pracuje w trybie automatycznego powtarzania operacji sterującej. W trybie z powtarzaniem przez cały czas, w którym użytkownik trzyma wciśnięty klawisz myszki, wykonywana jest okresowo operacja ustawienia zmiennej sterowanej na wartość podaną we właściwości Wartość włączenia lub *Wartość wyłączenia*, oraz powtarzane jest zdarzenie *Włączenie przycisku* lub *Wyłączenie przycisku*. Po puszczeniu przycisku myszki wykonywane jest jednokrotnie zdarzenie *Koniec powtarzania*, oraz sterowanie zgodne z wartością podaną we właściwości *Wartość końca powtarzania*.
Okres powtarzania [ms]	Liczba	Określa, co ile milisekund wysyłana jest wartość sterująca w przypadku pracy w trybie z automatycznym powtarzaniem.
Skrót klawiszowy	Tekst	Określa nazwę kombinacji przycisków klawiatury, których przyciśnięcie przez użytkownika jest równoznaczne z kliknięciem myszką w obszarze obiektu. Właściwe nazwy skrótów najlepiej uzyskać przy pomocy okienka edycji właściwości. Przykłady nazw: Alt+Q, Shift+F10, Ctrl+Space

Grupa sterowania	Tekst	Określa nazwę grupy obiektów sterujących. Jeżeli obiekt wchodzi w stan oczekiwania na wysłanie sterowania (oczekiwanie na akcję *SendControls*), to wszystkie pozostałe obiekty grupy rezygnują z oczekiwania. Mechanizm pozwala organizować przyciski w grupy, które sterują pojedynczymi bitami tej samej zmiennej i gwarantują ustawienie tylko jednego, ostatnio wybranego bitu.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Włączony	Logiczna	Określa, czy przycisk jest aktualnie w stanie włączonym (wciśniętym) czy wyłączonym (wyciśniętym). Na podstawie stanu wybierane są właściwe opisy i kolory przycisku. Wartość właściwości może być ignorowana, jeżeli ustawiona jest właściwość *Limit czasu odpowiedzi*, a obiekt jest bezpośrednio po wykonaniu operacji sterującej.
Wartość wyłączenia	Tekst	Określa wartość sterującą używaną w przypadku włączenia (wciśnięcia) przycisku. W zależności od ustawienia właściwości *Sterowanie bitowe* jest to wartość wysyłana bezpośrednio lub maska bitowa ustawianych bitów.
Wartość włączenia	Tekst	Określa wartość sterującą używaną w przypadku wyłączenia (wyciśnięcia) przycisku. W zależności od ustawienia właściwości *Sterowanie bitowe* jest to wartość wysyłana bezpośrednio lub maska bitowa zerowanych bitów.
Wartość końca powtarzania	Tekst	Określa wartość używaną do zakończenia sekwencji sterowania w trybie z powtarzaniem. W zależności od ustawienia właściwości Sterowanie bitowe jest to wartość wysyłana bezpośrednio lub maska bitowa ustawianych bitów.
Kolor wyłączenia	Kolor	Określa podstawowy kolor tła przycisku w stanie wyłączenia.
Kolor włączenia	Kolor	Określa podstawowy kolor tła przycisku w stanie włączenia.

Kolor gradientu wyłączenia	Kolor	Określa kolor stosowany do gradientowego wypełnienia tła przycisku w stanie wyłączenia. Kolor tła będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu. Sposób wypełniania określa właściwość *Kierunek gradientu.*
Kolor gradientu włączenia	Kolor	Określa kolor stosowany do gradientowego wypełnienia tła przycisku w stanie włączenia. Kolor tła będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu. Sposób wypełniania określa właściwość *Kierunek gradientu.*
Kolor otoczki wyłączenia	Kolor	Określa kolor otoczki przycisku w stanie wyłączenia.
Kolor otoczki włączenia	Kolor	Określa kolor otoczki przycisku w stanie włączenia.
Kolor cienia	Kolor	Określa kolor cienia przycisku.
Obrazek wyłączenia	Nazwa obrazka	Określa nazwę obrazka wyświetlanego na przycisku stanie wyłączenia.
Obrazek włączenia	Nazwa obrazka	Określa nazwę obrazka wyświetlanego na przycisku stanie włączenia.
Tekst wyłączenia	Tekst	Określa napis wyświetlany na przycisku stanie wyłączenia.
Tekst włączenia	Tekst	Określa napis wyświetlany na przycisku stanie włączenia. Jeżeli jest pusty, to w każdym stanie przycisku stosowany jest tekst podany we właściwości Tekst wyłączenia.
Kolor tekstu wyłączenia	Kolor	Określa kolor napis wyświetlanego na przycisku stanie wyłączenia.
Kolor tekstu włączenia	Kolor	Określa kolor napis wyświetlanego na przycisku stanie włączenia.

Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

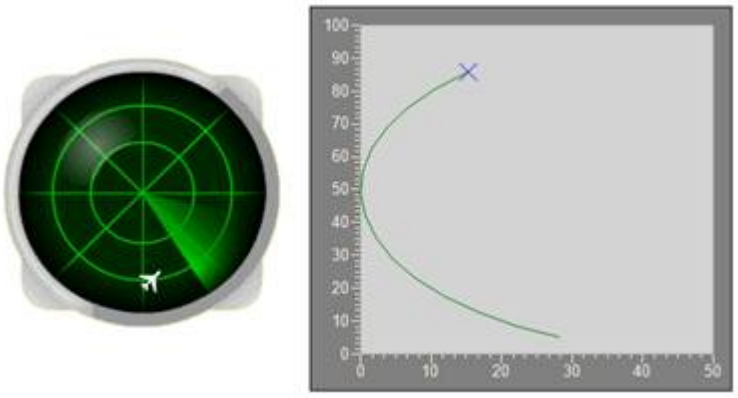
Nazwa	Opis
Włączenie przycisku	Zdarzenie wykonywane w przypadku włączenia (wciśnięcia) przycisku. Jeżeli sterowanie nie jest natychmiastowe, to zdarzenie wystąpi dopiero w momencie wykonania akcji *SendControls*.
Wyłączenie przycisku	Zdarzenie wykonywane w przypadku wyłączenia (wyciśnięcia) przycisku. Jeżeli sterowanie nie jest natychmiastowe, to zdarzenie wystąpi dopiero w momencie wykonania akcji *SendControls*.
Koniec powtarzania	Zdarzenie wykonywane jednokrotnie na zakończenie zakończenia sekwencji sterowania w trybie z powtarzaniem.

Obiekt Punkt pracy

Funkcje obiektu

Obiekt *Punkt pracy* służy do wizualizacji położenia punku pracy, wyliczanego jako funkcja dwóch wartości odpowiedzialnych za współrzędne X i Y. Położenie punktu pracy można pokazać na tle obrazka będącego częścią obiektu lub na tle innych obiektów. W ramach obiektu można wyświetlić podziałkę wartości.

Położenie punktu pracy można wyświetlić przy pomocy jednego z predefiniowanych kursorów lub zastosować dowolny obrazek. Możliwe jest rysowanie śladu ostatnich zmian położenia punktu pracy.



Rys. Przykłady obiektu Punkt Pracy

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wartość X	Liczba	Wartość współrzędnej X punktu pracy. Położenie na ekranie wyliczane jest z uwzględnieniem właściwości *Wartość minimalna X* i *Wartość maksymalna X.* Punkt pracy przemieszczany jest w całym obszarze obiektu, jeżeli nie są wyświetlane podziałki. W przeciwnym przypadku obszar zajęty przez podziałki jest pomijany.
Wartość minimalna X	Liczba	Wartość odpowiadająca skrajnemu lewemu położeniu punktu pracy.
Wartość maksymalna X	Liczba	Wartość odpowiadająca skrajnemu prawemu położeniu punktu pracy.
Wartość Y	Liczba	Wartość współrzędnej Y punktu pracy. Położenie na ekranie wyliczane jest z uwzględnieniem właściwości *Wartość minimalna Y* i *Wartość maksymalna Y.*
Wartość minimalna Y	Liczba	Wartość odpowiadająca skrajnemu dolnemu położeniu punktu pracy.
Wartość maksymalna Y	Liczba	Wartość odpowiadająca skrajnemu górnemu położeniu punktu pracy.
Styl wskaźnika	Typ wyliczeniowy: Brak, Krzyżyk, X, Okrągły, Obrazek	Określa sposób rysowania kursora pokazującego bieżącą pozycję punktu pracy. Wartość *Obrazek* oznacza użycie dowolnego obrazka specyfikowanego we właściwości stanowej *Obrazek wskaźnika*.
Rozmiar wskaźnika [%]	Liczba	Wielkość wskaźnika wyrażona w procentach obszaru roboczego obiektu. Wartość 0 oznacza użycie wskaźnika o stałej wielkości – niezmieniającego się przy zmianie rozmiaru obiektu.

Liczba punktów śladu	Liczba	Określa liczbę wyświetlanych punktów śladu zmiany położenia punktu pracy.
Szerokość linii	Liczba	Określa szerokość linii rysowania śladu zmian położenia punktu pracy.
Kasowanie punktów	Logiczna	Określa, czy w trakcie pracy obiektu, kliknięcie w obszarze obiektu będzie usuwało ślad zmian położenia punktu pracy.
Pokaż podziałkę X	Logiczna	Określa, czy rysowana jest podziałka pozioma obszaru roboczego.
Pokaż podziałkę Y	Logiczna	Określa, czy rysowana jest podziałka pionowa obszaru roboczego.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania wartości podziałek skali słupka.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Liczba podziałek X	Liczba	Określa liczbę podziałek głównych na poziomej skali obszaru roboczego
Liczba podpodziałek X	Liczba	Określą liczbę podpodziałek pomiędzy poszczególnymi podziałkami głównymi skali poziomej.
Liczba podziałek Y	Liczba	Określa liczbę podziałek głównych na pionowej skali obszaru roboczego
Liczba podpodziałek Y	Liczba	Określą liczbę podpodziałek pomiędzy poszczególnymi podziałkami głównymi skali pionowej.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor wskaźnika	Kolor	Określa kolor wskaźnika aktualnego położenia punktu pracy.
Obrazek wskaźnika	Nazwa obrazka	Określa nazwę obrazka używanego jako wskaźnik aktualnego położenia punktu pracy. Obrazek musi znajdować się w bibliotece obrazków aplikacji.
Kolor śladu	Kolor	Określa kolor linii pokazującej ślad zmian położenia punktu pracy.
Kolor tła	Kolor	Określa kolor tła obiektu (obszaru zajętego przez podziałki).
Obrazek tła	Nazwa obrazka	Nazwa obrazka służącego jako tekstura wypełnienia tła obiektu. Obrazek musi znajdować się w bibliotece obrazków aplikacji.
Kolor kanwy	Kolor	Określa kolor obszaru roboczego obiektu. Jeżeli wskaźnik ma być wyświetlany na tle innych obiektów, to należy wybrać kolor Transparent.
Kolor podziałki	Kolor	Określa kolor używany do wyświetlenia podziałki.

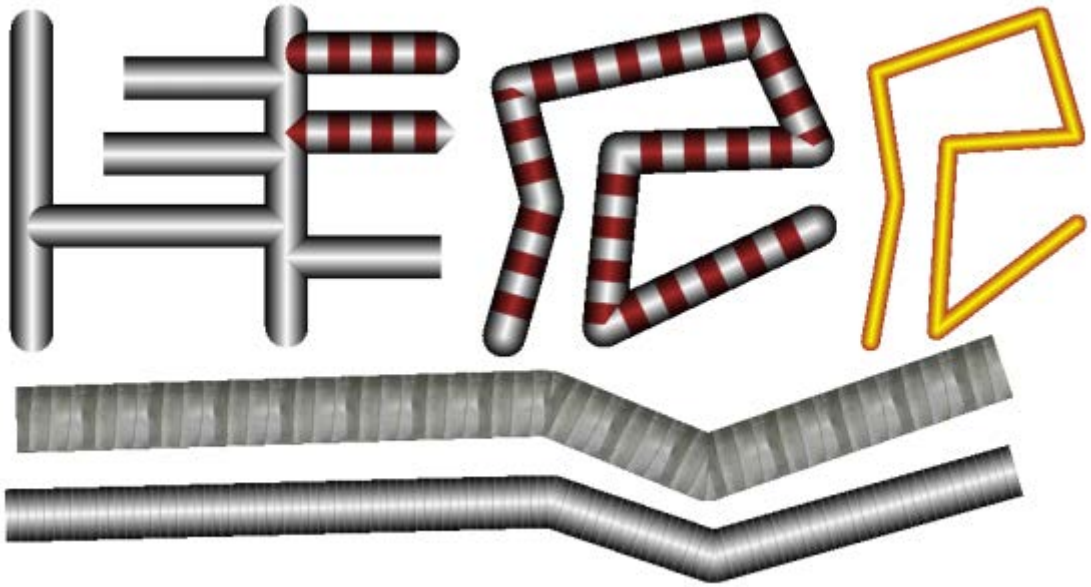
Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Rurociąg

Funkcje obiektu

Obiekt *Rurociąg* służy do prezentacji stanu rurociągów. Kształt obiektu jest definiowany serią koordynat poszczególnych wierzchołków rurociągu. Poprzez odpowiedni dobór kształtu końcówek rurociągu możliwe jest łączenie kilku obiektów w pojedynczy schemat przepływu. Obiekt pozwala na symulację efektu przepływu poprzez manipulację właściwościami definiującymi sposób wypełnienia obiektu.



Rys. Przykłady obiektu Rurociąg.



Rys. Przykład zastosowania obiektu Rurociąg (z opcją animacji ruchu: ciecz "płynie" w rurze) w połączeniu z obiektem Obraz.

Obiekt *Rurociąg* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

Zobacz: ***Techniki_budowy_diagramów > Parametryzacja funkcji interaktywnych w obiektach pasywnych***

Właściwości

Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

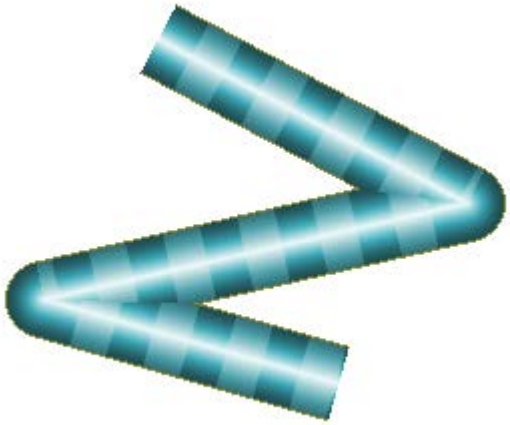
Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wierzchołki	Lista współrzędnych	Lista par X,Y współrzędnych kolejnych wierzchołków rurociągu. Współrzędne X i Y pozycji wierzchołka wyrażone są pikselach i

		liczone względem lewego górnego narożnika diagramu. Kolejne pary wierzchołków są rozdzielone znakiem średnika.
Grubość	Liczba	Określa grubość rurociągu wyrażoną w ilości pikseli.
Styl początku	Typ wyliczeniowy: Płaski, Okrągły, Okrągły koniec, Trójkątny, Ścięty z lewej, Ścięty z prawej	Określa sposób zakończenia początku rurociągu.  Poprzez odpowiedni dobór końcówek można łączyć wizualnie różne obiekty Rurociąg. 
Styl końca	Typ wyliczeniowy: Płaski, Okrągły, Okrągły koniec, Trójkątny, Ścięty z lewej, Ścięty z prawej	Określa sposób zakończenia końcówki rurociągu.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor	Kolor	Określa podstawowy kolor rurociągu. W przypadku korzystania z gradientu kolor podstawowy używany jest na zewnętrznych elementach rurociągu.
Kolor gradientu	Kolor	Określa kolor stosowany do gradientowego wypełnienia wnętrza obiektu. Kolor będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu.
Kolor pasków	Kolor	Określa podstawowy kolor pasków wyświetlanych w poprzek rurociągu. Paski, w połączeniu z

		właściwością *Offset wypełnienia*, pozwalają na uzyskanie efektu przepływu. 
Kolor gradientu pasków	Kolor	Określa kolor stosowany do gradientowego wypełnienia pasków wyświetlanych w poprzek rurociągu.
Kolor krawędzi	Kolor	Określa kolor używany od zaznaczenia konturu rurociągu.
Nazwa obrazu	Nazwa obrazka	Nazwa obrazka służącego jako tekstura wypełnienia tła rurociągu. Obrazek musi znajdować się w bibliotece obrazków aplikacji.
Offset wypełnienia	Liczba	W trybie wyświetlania rurociągu z poprzecznymi paskami (lub z wypełnieniem teksturą) właściwość pozwala na przesunięcie wzorca rysowania, co powoduje efekt animacji ruchu. Jednostką przesunięcia jest piksel. Przykład: =-Variable(Counter) Wyrażenie animuje ruch rurociągu na bazie zmian zmiennej systemowej Counter. Znak minusa powoduje, że symulowany ruch przebiega w kierunku początku rurociągu. Ewentualnie przemnożenie(podzielenie) wartości zmiennej pozwala na sterowanie szybkością i płynnością animacji.

Zdarzenia

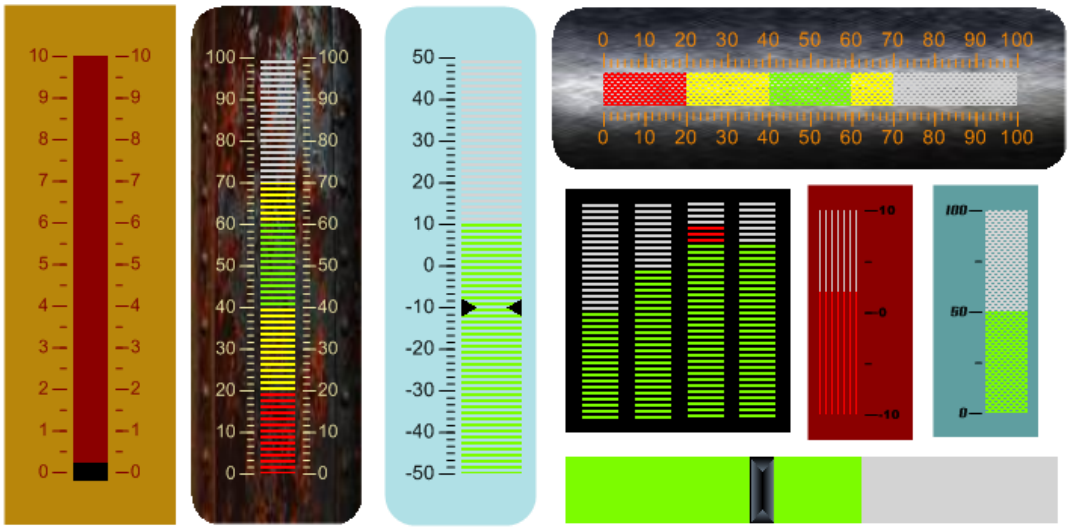
Zobacz: ***Elementy_wizualizacji*** > ***Standardowe zdarzenia obiektów***

Obiekt Słupek

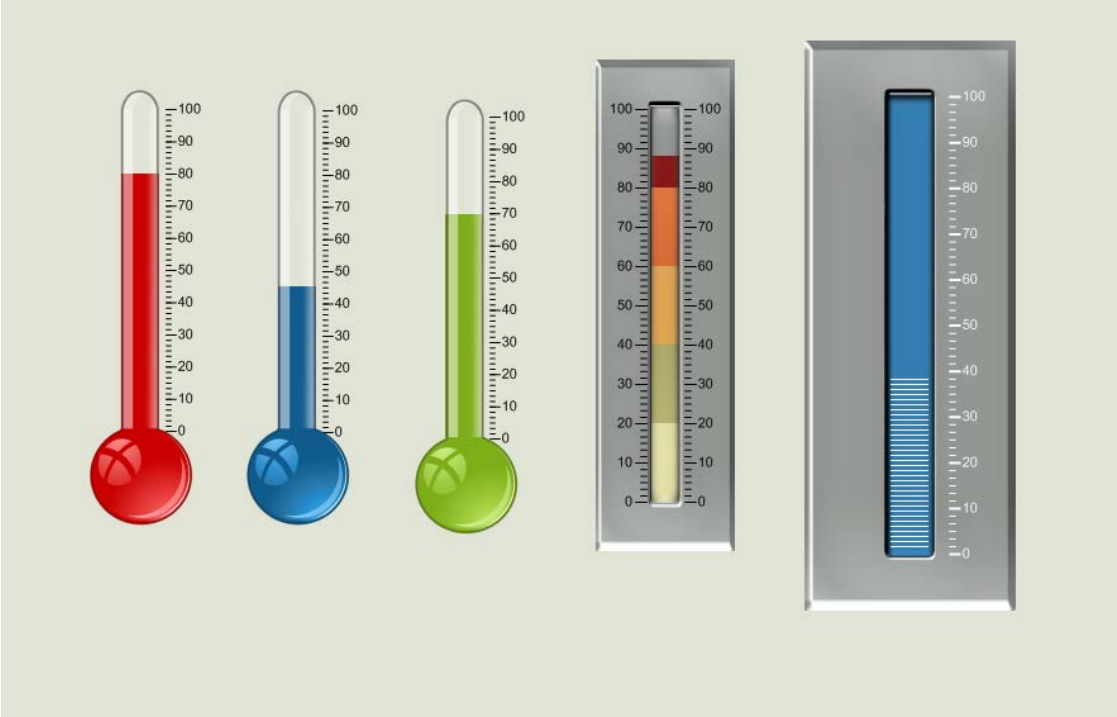
Funkcje obiektu

Obiekt *Słupek* jest wielofunkcyjnym obiektem służącym zarówno do wyświetlania wartości pomiarów w postaci słupka z podziałką jak i sterowania wartością podłączonej zmiennej za pomocą suwaka. Dla każdego słupka można zdefiniować: zakres wartości, liczbę podziałek głównych i podpodziałek, miejsce wyświetlania podziałki, styl kreskowania słupka. Możliwe jest wyświetlanie limitów (LL, L, H, HH) z indywidualnie ustawianym kolorem każdego limitu.

Obiekt ma wbudowane funkcje interaktywne służące do zmiany nastaw przy pomocy zintegrowanego suwaka. W procesie sterowania obiekt może współpracować z innymi obiektami poprzez tzw. zmienną podglądową - mechanizm ten pozwala pokazać wartość nastawy jeszcze przed wykonaniem rzeczywistego sterowania.



Rys. Przykłady obiektu Słupek.



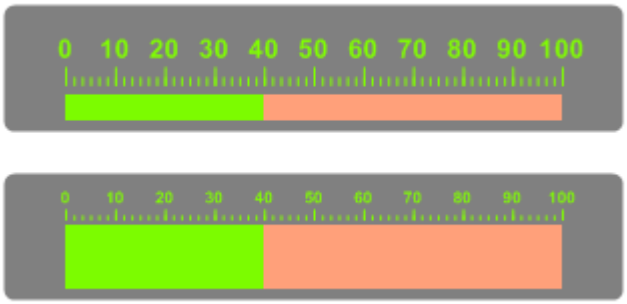
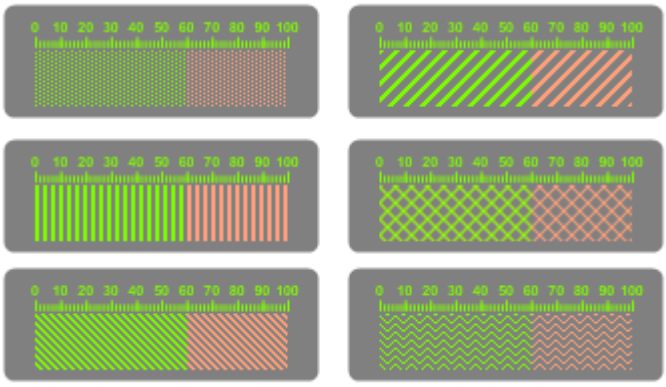
Rys. Przykłady zastosowania obiektu Słupek w połączeniu z obiektem Obraz.

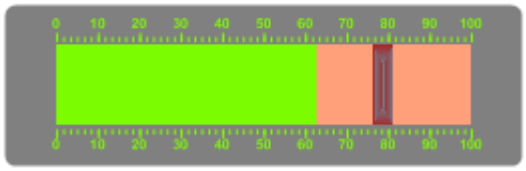
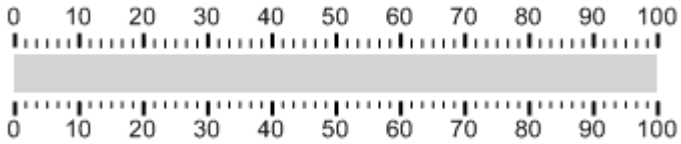
Właściwości

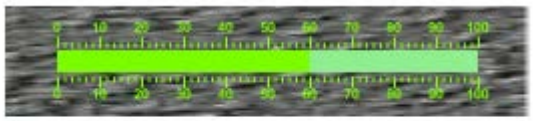
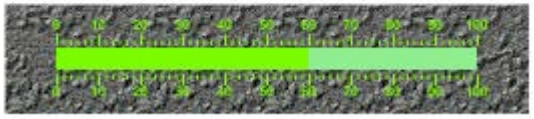
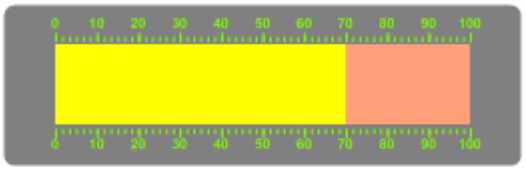
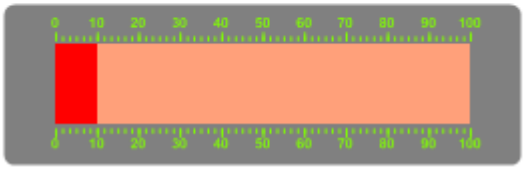
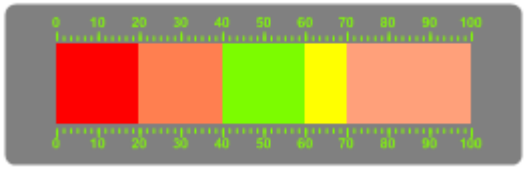
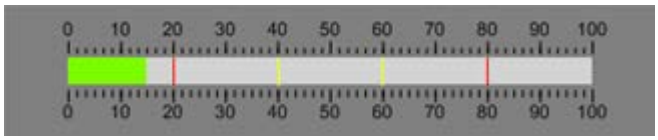
Zobacz: ***Elementy_wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

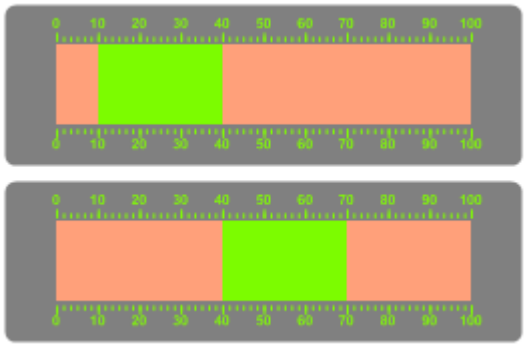
Nazwa	Typ wartości	Opis
Zmienna podglądowa	Nazwa zmiennej	Określa nazwę zmiennej podglądowej. Jeżeli obiekt pełni role sterujące, to w momencie przesuwania przez użytkownika suwaka zmienna podglądowa jest ustawiana na wartość, która byłaby wysłana do zmiennej sterowanej, gdyby w tym momencie operator puścił suwak. Pozwala to pokazać numeryczną wartość potencjalnej nastawy przy pomocy obiektu typu *Tekst*. Zmienna podglądowa powinna być typu *Double*. W momencie puszczenia suwaka wartość zmiennej

		podglądowej jest ustawiana na minimalną wartość typu *Double* - pozwala to wykryć stan, kiedy suwak nie jest używany i np. wygasić pokazywanie wartości podglądu.
Układ poziomy	Logiczna	Określa, czy obiekt jest rysowany w układzie pionowym lub poziomym.
Pokaż słupek	Logiczna	Właściwość pozwala na wyłączenie pokazywania słupka wartości. W tym wariancie obiekt może służyć do wyświetlania skali wartości. 
Szerokość słupka [%]	Procent	Określa jaki procent szerokości (wysokości) obiektu zajmuje słupek wartości. 
Styl kreskowania słupka	Typ wyliczeniowy: Horizontal, &ldots;	Określa sposób rysowania słupka wartości. 
Styl wskaźnika	Typ wyliczeniowy: Brak, Linia,	Określa sposób wyświetlania wskaźnika. Wskaźnik służy do prezentacji wartości nastawy związanej ze słupkiem wartości. Pozycja wskaźnika jest określona przez właściwość *Wartość*

	Trójkąty, Uchwyt	*wskaźnika* z uwzględnieniem limitów *Wartość minimalna sterowania* i *Wartość maksymalna sterowania*.  Jeżeli podana została nazwa zmiennej sterowanej (właściwość *Zmienna sterowana*) oraz obiekt jest aktywny (właściwość *Aktywny*), to wskaźnik pełni rolę suwaka, przy pomocy którego użytkownik może zmieniać wartość nastawy.
Pokaż podziałkę z lewej	Logiczna	Określa, czy rysowana jest podziałka po lewej (górnej) stronie słupka.
Pokaż podziałkę z prawej	Logiczna	Określa, czy rysowana jest podziałka po prawej (dolnej) stronie słupka.
Bez tła	Logiczna	Określa, czy rysowane jest tło obiektu. Jeżeli nie, to wyświetlany jest tylko słupek wartości oraz, jeżeli są włączone, prawa i lewa podziałka. 
Obrazek tła	Nazwa obrazka	Nazwa obrazka służącego jako tekstura wypełnienia tła obiektu. Obrazek musi znajdować się w bibliotece obrazków aplikacji.
Tryb wypełnienia obrazu	Typ wyliczeniowy: Kafelki, Rozciąganie	Określa, czy rozmiar obrazka tła jest dopasowywany do rozmiaru obiektu, czy też wyświetlany jest on wielokrotnie w oryginalnym rozmiarze, tak aby wypełnił całe tło.

		 
Użyj limitów	Logiczna	Określa, czy kolor słupka wartości ma być uzależniony od przekroczeń limitów określonych wartościami właściwości *Limit HH* itp.  
Dziel słupek według limitów	Logiczna	Określa, czy słupek wartości ma być rysowany segmentami o wielkościach i kolorach określonych limitami (właściwości *Limit HH* itp.) i kolorami (właściwości *Kolor HH* itp.) przekroczeń.  Dodatkowo, właściwość *Użyj limitów* musi być ustawiona na *Tak.*
Pokaż limity	Logiczna	Określa, czy wartości limitów (określonych wartościami właściwości Limit HH itp.) mają być pokazywane na tle słupka. 
Obrazek słupka	Nazwa obrazka	Nazwa obrazka służącego jako tekstura wypełnienia obszaru aktywnego (ruchomego) słupka. Obrazek musi znajdować się w bibliotece obrazków aplikacji.

Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania wartości podziałek skali słupka.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Format liczb podziałki	Tekst	Określa sposób formatowania liczb wyświetlanych na podziałce, np. *f1* oznacza wyświetlanie z jednym miejscem po przecinku.
Liczba podziałek	Liczba	Określa liczbę podziałek głównych na skali słupka.
Liczba podpodziałek	Liczba	Określą liczbę podpodziałek pomiędzy poszczególnymi podziałkami głównymi.
Wartość słupka	Liczba	Określa aktualną wysokość słupka. Wartość właściwości powinna być liczbą z zakresu określonego przez właściwości *Wartość minimalna* i *Wartość maksymalna*.
Wartość wskaźnika	Liczba	Określa pozycję wskaźnika. Wartość właściwości powinna być liczbą z zakresu określonego przez właściwości *Wartość minimalna sterowania* i *Wartość maksymalna sterowania*.
Wartość minimalna	Liczba	Dolny limit zakresu skali słupka.
Wartość maksymalna	Liczba	Górny limit zakresu skali słupka.
Wartość minimalna sterowania	Liczba	Wartość sterująca odpowiadająca skrajnemu lewemu (dolnemu) położeniu suwaka.
Wartość maksymalna sterowania	Liczba	Wartość sterująca odpowiadająca skrajnemu prawemu (górnemu) położeniu suwaka.
Krok sterowania	Liczba	Określa sposób wyliczania wartości sterujących przy pracy w trybie suwaka. Wartość kroku równa 0 oznacza, że wartość sterująca odpowiada miejscu kliknięcia w obszarze słupka.

		Wartość kroku większa od 0 oznacza, że istotne jest tylko, po której stronie suwaka nastąpiło kliknięcie. Wartość sterująca wyliczana jest poprzez dodanie lub odjęcie kroku sterowania od aktualnej wartości wskaźnika.
Limit czasu odpowiedzi [ms]	*Liczba*	Ustawienie limitu pozwala na zamaskowanie opóźnienia pomiędzy wysłaniem sterowania zmieniającego położenie suwaka, a faktycznym wykonaniem tego sterowania. W praktyce oznacza to, że po wykonaniu sterowania obiekt pokazuje suwak w miejscu wartości zadanej niezależnie od stanu faktycznego, a dopiero po upływie limitu czasu odpowiedzi zaczyna pokazywać wartość rzeczywistą. Limit powinien być na tyle długi, żeby w jego trakcie zdążyło się wykonać wysłanie wartości sterującej oraz odświeżenie wartości właściwości Wartość wskaźnika. Takie ustawienie pozwala ukryć efekt migotania stanu obiektu (pozycji wskaźnika) w trakcie wykonywania operacji sterowania.
Baza słupka	Liczba	Określa wartość bazową określająca podstawę słupka. Jeżeli słupek ma się zaczynać od początku skali, to wartość bazowa powinna być identyczna z wartością minimalną. 
Wartość LL	Liczba	Określa limit LL (minimum krytyczne) sygnalizacji przekroczeń. Można indywidualnie wyłączyć ten limit poprzez ustawienie go na wartość minimalną.
Wartość L	Liczba	Określa limit L (minimum) sygnalizacji przekroczeń. Można indywidualnie wyłączyć ten limit poprzez ustawienie go na wartość minimalną.
Wartość H	Liczba	Określa limit H (maksimum) sygnalizacji przekroczeń. Można indywidualnie wyłączyć ten limit poprzez ustawienie go na wartość maksymalną.

Wartość HH	Liczba	Określa limit HH (maksimum krytyczne) sygnalizacji przekroczeń. Można indywidualnie wyłączyć ten limit poprzez ustawienie go na wartość maksymalną.
Promień zaokrąglenia	Liczba	Określa wyrażony w pikselach promień zaokrąglenia narożników.
Sterownie natychmiastowe	Logiczna	Określa, czy w przypadku ustawienia nowej pozycji suwaka, odpowiadająca jej nastawa jest automatycznie wysyłana do zmiennej sterującej. Jeżeli nie, to do realizacji sterowania wymagane jest użycie akcji operatorskiej *SendControls*.
Grupa sterowania	Tekst	Określa nazwę grupy obiektów sterujących. Jeżeli obiekt wchodzi w stan oczekiwania na wysłanie sterowania (oczekiwanie na akcję *SendControls*), to wszystkie pozostałe obiekty grupy rezygnują z oczekiwania. Mechanizm pozwala zagwarantować, że jeżeli operator dysponował alternatywnymi metodami ustawienia tej samej wartości (np. obiekty *Słupek* i *Tekst*), to wykonane zostanie tylko ostatnie sterowanie.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor wskazówki	Kolor	Określa kolor wskazówki (suwaka).
Kolor tła	Kolor	Określa kolor tła obiektu.
Kolor tła słupka	Kolor	Określa kolor tła słupka wartości.
Kolor podziałki	Kolor	Określa kolor podziałki na skali słupka. Kolor *Transparent* pozwala wyłączyć wyświetlanie skali z zachowaniem układu słupka i tła.
Kolor wartości poprawnej	Kolor	Określa kolor słupka dla wartości z przedziału wartości normalnych.
Kolor LL	Kolor	Określa kolor słupka dla wartości z przedziału wartości LL (minimum krytycznego).
Kolor L	Kolor	Określa kolor słupka dla wartości z przedziału wartości L (minimum).

Kolor H	Kolor	Określa kolor słupka dla wartości z przedziału wartości H (maksimum).
Kolor HH	Kolor	Określa kolor słupka dla wartości z przedziału wartości HH (maksimum krytycznego).

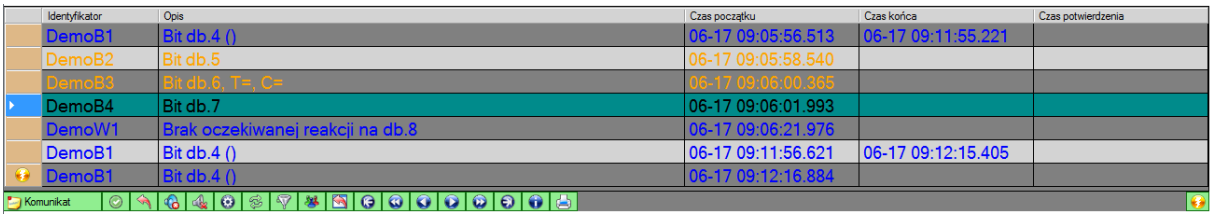
Zdarzenia

Zobacz: ***Elementy_wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Tabela alarmów aktywnych

Funkcje obiektu

Obiekt *Tabela alarmów aktywnych* jest obiektem umożliwiającym wyświetlanie oraz obsługę aktywnych alarmów. Tabela pokazuje alarmy z wybranej domeny alarmowej. Zestaw danych pokazywanych w tabeli jest swobodnie definiowany. Możliwe jest użycie zintegrowanego paska narzędziowego - zestaw dostępnych funkcji jest parametryzowany.



Rys. Przykłady obiektu Tabela alarmów aktywnych.

Obiekt *Tabela alarmów aktywnych* jest obiektem typu kontrolkowego.

Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Domena	Nazwa domeny	Określa nazwę domeny alarmowej, której alarmy są pokazywane w tabeli.
Grupy	Tekst	Określa początkowy filtr selekcji grup pokazywanych alarmów. Użytkownik może zmienić w trakcie pracy ten filtr przy pomocy okienka Filtrowanie grup alarmów. Definicja właściwość ma postać tekstu złożonego z par *<identyfikator_atrybutu_grupującego>=<wartość_atrybutu>* rozdzielonych znakami średnika. Przykład: Strefa="B1";Strefa="B2”;Typ=”Elektryczna"; Pokazywane będą alarmy, których atrybut *Strefa* jest równy B1 lub B2 i atrybut *Typ* jest równy Elektryczna
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki komunikatów	Liczba	Określa rozmiar czcionki używanej do wyświetlania informacji w wierszach alarmów.
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, jeżeli rozmiar obiektu zmienia się w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba wierszy - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.
Elementy paska narzędziowego	Typ wyliczeniowy wielokrotny: Brak paska, Wskaźnik aktywności,	*Określa zawartość paska narzędziowego tabeli. Dostępne są poniższe typy przycisków:* • *Wskaźnik aktywności* - służy do sygnalizacji nowych zdarzeń. Po wykryciu nowego zdarzenia alarmowego na przycisku przez pewien czas pojawia się migająca ikonka.

	Selekcja priorytetów, Selekcja grup, Ukrywanie potwierdzonych, Potwierdź, Wyklucz jednokrotnie, Wycisz lokalnie, Wycisz wszędzie, Wykonaj akcję, Wstrzymanie/ przywrócenie odświeżania, Odśwież, Edytor filtrów, Edytuj wykluczenia, Pierwszy alarm, Poprzednia strona, Poprzedni alarm, Następny alarm, Następna strona, Ostatni alarm, Szczegóły, Drukowanie	Alternatywnie, nowe alarmy mogą być sygnalizowane na wskaźniku wierszy. • *Selekcja priorytetów* - selekcja alarmów według priorytetów. • *Selekcja grup* - otwarcie okna selekcji alarmów według grup. • *Ukrywanie potwierdzonych* - wybór trybu pokazywania alarmów potwierdzonych. • *Potwierdź* - potwierdzenie alarmów pokazywanych aktualnie w tabeli. Alarm pilny i krytyczny zostanie potwierdzony tylko wtedy, gdy jego wiersz jest wybrany. Można również selektywnie potwierdzać alarmy poprzez podwójne kliknięcie klawiszem myszki na linii alarmu, lub wybranie alarmu i użycie klawisza *Enter.* • *Wyklucz jednokrotnie* - jednokrotne wykluczenie alarmu. Alarm zostanie usunięty z listy alarmów aktywnych, kolejne wykrycie alarmu zostanie normalnie pokazane. • *Wycisz lokalnie* - w przypadku gdy alarmy są sygnalizowane dźwiękiem, powoduje wyciszenie sygnału na lokalnym komputerze. • *Wycisz wszędzie* - w przypadku gdy alarmy są sygnalizowane dźwiękiem, powoduje wyciszenie sygnału na wszystkich komputerach domeny alarmów. • *Wykonaj akcję* - wykonanie akcji zdefiniowanej w bazie alarmów dla wybranego alarmu. • *Wstrzymanie/przywrócenie odświeżania* - przycisk pozwalający na blokowanie przez operatora odświeżania (przewijania) listy alarmów. • *Odśwież* - w przypadku pracy w trybie bez automatycznego usuwania zakończonych alarmów, przycisk pozwala na odświeżenie widoku usuwające z tabeli wszystkie nieaktualne zdarzenia. • *Edytor filtrów* - otwarcie okna zarządzania alarmami filtrowanymi. • *Edytuj wykluczenia* - otwarcie okna zarządzania alarmami wykluczonymi. • *Wyszukiwanie opisu* - otwarcie okna filtrowania lub wyszukiwania alarmów według opisów. • *Pierwszy alarm* - pozycjonowanie na pierwszym wierszu tabeli, identyczne z przyciśnięciem przycisku *Ctrl-Home*. • *Poprzednia strona* - przejście w górę o ilość wierszy równą wielkości strony, identyczne z przeciśnięciem przycisku PageUp.

		• *Poprzedni alarm* - pozycjonowanie o jeden wiersz w górę, identyczne z przyciśnięciem przycisku strzałki w górę. • *Ostatni alarm* - pozycjonowanie na ostatnim wierszu tabeli, identyczne z przyciśnięciem przycisku *Ctrl-End*. • *Następna strona* - przejście w dół o ilość wierszy równą wielkości strony, identyczne z przyciśnięciem przycisku *PageDown*. • *Następny alarm* - pozycjonowanie o jeden wiersz w dół, identyczne z przyciśnięciem przycisku strzałki w dół. • *Szczegóły* - otwarcie okna szczegółów alarmów. • *Drukowanie* - otwarcie okna wydruku alarmów.
Wysokość paska narzędziowego	Liczba	Określa wysokość paska narzędziowego w pikselach. Od wysokości paska zależy wielkość ikon przycisków.
Filtr priorytetu alarmu	Typ wyliczeniowy wielokrotny: Komunikat, Ostrzeżenie, Alarm, Pilny, Krytyczny	Określa początkowy stan priorytetów pokazywanych alarmów. Operator może zmienić ustawienie poprzez przycisk selekcji priorytetu paska narzędziowego.
Pokazuj nagłówki kolumn	Logiczna	Określa, czy należy pokazywać linię nagłówków kolumn.
Zezwalaj na zmianę szerokości kolumn	Logiczna	Określa, czy użytkownik aplikacji będzie mógł zmieniać szerokość kolumn tabeli alarmów przy pomocy myszki.
Przewiń do ostatniego alarmu	Logiczna	Określa, czy w momencie zmian w stanie alarmów należy automatycznie spozycjonować widok na ostatnim zdarzeniu.
Usuwaj zakończone alarmy	Logiczna	Określa sposób zachowania tabeli w momencie usuwania alarmów z logu alarmów aktywnych. Alarm zakończony jest usuwany natychmiast lub z pewnym opóźnieniem z logu alarmów aktywnych. Jeżeli właściwość ma wartość Tak, to w tym samym momencie alarm zostanie usunięty z tabeli. Jeżeli wartość jest równa Nie, to alarmy nie są automatycznie usuwane z tabeli. Będą pokazywane aż do

		momentu, gdy operator użyje przycisk *Odśwież* z paska narzędziowego tabeli.
Nie pokazuj potwierdzonych alarmów	Logiczna	Określa, czy alarmy potwierdzone powinny być usunięte z tabeli alarmów bez oczekiwania na zakończenie alarmu. Operator może zmienić tryb pokazywania alarmów potwierdzonych przyciskiem paska narzędziowego.
Odwróć kolejność alarmów	Logiczna	Określa porządek wyświetlania alarmów. Wartość Tak oznacza, że najnowszy alarm jest wyświetlany w pierwszej (górnej) linii tabeli.
Pokazuj wskaźnik wierszy	Logiczna	Powoduje wyświetlenie dodatkowej kolumny służącej do sygnalizacji wybranych wierszy i nowo pojawiających się alarmów. Po wykryciu nowego zdarzenia alarmowego we właściwym wierszu kolumny wskaźników pojawia się przez pewien czas migająca ikonka. Alternatywnie, nowe alarmy mogą być sygnalizowane na przycisku Wskaźnik aktywności paska narzędziowego.
Pokazuj pasek przewijania	Logiczna	Określa, czy należy pokazywać suwak przewijania alarmów.
Format czasu	Tekst	Określa format wyświetlania kolumn czasu alarmów. Domyślny format *yyyy'-'MM'-'dd' 'HH':'mm':'ss'.'fff* powoduje wyświetlanie czasów w postaci 2011-06-17 11:10:09.345. Można dowolnie skonstruować własny format używając wybranych oznaczeń pól z domyślnego formatu. Separatory pól mogą być dowolne, ale muszą być umieszczone pomiędzy znakami apostrofów.

Właściwości kolumny

Nazwa	**Typ wartości**	**Opis**
Typ kolumny	Typ wyliczeniowy: Identyfikator, Priorytet, Kategoria, Status, Opis, Czas zdarzenia,	Określa rodzaj informacji pokazywanej w kolumnie. Informacje dotyczące alarmu pochodzą z logu alarmów historycznych lub z bazy definicji alarmów. Tabela alarmów nie musi zawierać kolumn pokazujących wszystkie rodzaje informacji. Użytkownik może użyć okienka Szczegóły alarmu w celu uzyskania dostępu do wszystkich informacji związanych z alarmem. Dostępne są poniższe typy kolumn:

	Czas początku, Opis początku, Czas końca, Opis końca, Czas detekcji początku, Czas detekcji końca, Potwierdzony, Czas potwierdzenia, Potwierdzony przez, Potwierdzony na, Filtrowany, Notatka potwierdzenia	• Identyfikator - identyfikator alarmu; • Priorytet - priorytet alarmu (krytyczny, komunikat itd.) w postaci ikonki; • Kategoria- informacje: czy alarmy jest zwykły czy typu powiadomienie; • Status - status zdarzenia w postaci ikonki (rozpoczęty, rozpoczęty i potwierdzony, zakończony, zakończony i potwierdzony); • Opis - opis tekstowy alarmu - w zależności od tego czy pokazywane zdarzenie już się zakończyło, wyświetlany jest opis początku lub końca; • Czas zdarzenia - wyświetla jeden z zarejestrowanych czasów alarmów w zależności od typu zdarzenia. Dla alarmu rozpoczętego i niepotwierdzonego pokazuje czas początku alarmu, dla alarmów rozpoczętych i potwierdzonych pokazuje czas potwierdzenia, dla zakończonych i niepotwierdzonych czas końca alarmu, a dla zakończonych i potwierdzonych późniejszy z czasów końca lub potwierdzenia; • Czas początku- zarejestrowany czas początku alarmu; • Opis początku - opis tekstowy początku alarmu tworzony na podstawie definicji alarmu z uwzględnieniem parametrów zgłoszenia; • Czas końca - zarejestrowany czas zakończenia alarmu; • Opis końca - opis tekstowy końca alarmu tworzony na podstawie definicji alarmu z uwzględnieniem parametrów zgłoszenia; • Czas detekcji początku - zarejestrowany czas wykrycia początku alarmu; • Czas detekcji końca - zarejestrowany czas wykrycia zakończenia alarmu; • Potwierdzony - informacja określająca, czy alarm został potwierdzony przez operatora; • Czas potwierdzenie - zarejestrowany czas potwierdzenia alarmu; • Potwierdzony przez - identyfikator użytkownika, który potwierdził alarm; • Potwierdzony na - nazwa stanowiska, na którym alarm został potwierdzony; • Filtrowana - informacja określająca, czy w momencie zgłoszenia alarmu podlegał on mechanizmowi filtrów wykrywania alarmów. • • Notatka potwierdzenia - notatka wprowadzona przez operatora w momencie potwierdzenia alarmu.

Szerokość kolumny [%]	Procent	Określa procentową szerokość kolumny. Wartość 0 oznacza, że szerokość kolumny ma być dobrana automatycznie. Jeżeli tylko jedna kolumna będzie miała szerokość 0, to zajmie ona całe pozostałe miejsce po uwzględnieniu szerokości pozostałych kolumn.
Nagłówek kolumny	Tekst	Określa tekst nagłówka kolumny, jeżeli ma być inny od opisu domyślnego.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor tła	Kolor	Określa kolor tła tabeli - używany, gdy ilość alarmów jest mniejsza niż ilość wierszy w tabeli.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze tabeli.
Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.
Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.
Kolor komunikatu	Kolor	Określa kolor tekstów dla wierszy alarmów typu komunikat.
Kolor tła komunikatu	Kolor	Określa kolor tła dla wierszy alarmów typu komunikat. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor ostrzeżenia	Kolor	Określa kolor tekstów dla wierszy alarmów typu ostrzeżenie.
Kolor tła ostrzeżenia	Kolor	Określa kolor tła dla wierszy alarmów typu ostrzeżenie. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów*.

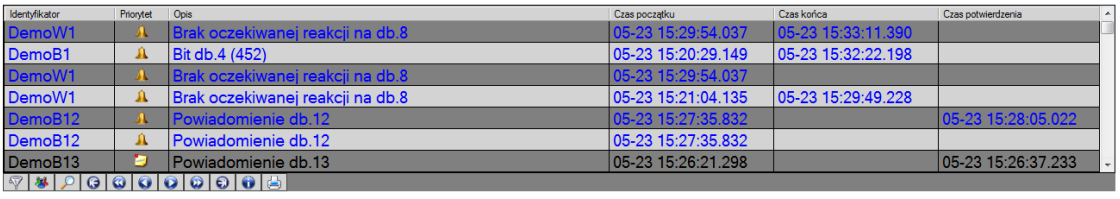
Kolor alarmu	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm zwykły.
Kolor tła alarmu	Kolor	Określa kolor tła dla wierszy alarmów typu alarm zwykły. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów*.
Kolor alarmu pilnego	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm pilny.
Kolor tła alarmu pilnego	Kolor	Określa kolor tła dla wierszy alarmów typu alarm pilny. Wybranie koloru Transparent oznacza, że kolor tła określany jest na podstawie właściwości Kolor tła alarmów i Naprzemienny kolor tła alarmów.
Kolor alarmu krytycznego	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm krytyczny.
Kolor tła alarmu krytycznego	Kolor	Określa kolor tła dla wierszy alarmów typu alarm krytyczny. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów*.
Kolor tła alarmów	Kolor	Określa kolor tła dla wierszy nieparzystych.
Naprzemienny kolor tła alarmów	Kolor	Określa kolor tła dla wierszy parzystych. Ustawienie innej wartości niż dla właściwości *Kolor tła alarmów* pozwala na uzyskanie efektu naprzemiennego wyświetlania alarmów w różnych kolorach.
Kolor czcionki wybranego alarmu	Kolor	Określa kolor tekstów dla aktualnie wybranego wiersza. Oznaczenie aktywnego wiersza można wyłączyć poprzez wybór koloru *Transparent*.
Kolor tła wybranego alarmu	Kolor	Określa kolor tła dla aktualnie wybranego wiersza. Oznaczenie aktywnego wiersza można wyłączyć poprzez wybór koloru *Transparent*.
Kolor tła wskaźnika wierszy	Kolor	Określa kolor tła kolumny wskaźników wierszy.

Kolor tła wskaźnika wybranego wiersza	Kolor	Określa kolor tła dla wybranego wiersza w kolumnie wskaźników wierszy.
Kolor przycisków	Kolor	Określa kolor tła przycisków paska narzędziowego.
Ramka tabeli widoczna	Logiczna	Określa, czy wyświetlać linię obramowania tabeli.

Obiekt Tabela alarmów historycznych

Funkcje obiektu

Obiekt *Tabela alarmów historycznych* jest obiektem umożliwiającym wyświetlanie i zaawansowane filtrowanie zarejestrowanych alarmów historycznych. Tabela pokazuje alarmy z wybranej domeny alarmowej. Zestaw danych pokazywanych w tabeli jest swobodnie definiowany. Możliwe jest użycie zintegrowanego paska narzędziowego - zestaw dostępnych funkcji jest parametryzowany.



Rys. Przykłady obiektu Tabela alarmów historycznych.

Obiekt *Tabela alarmów historycznych* jest obiektem typu kontrolkowego.

Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Domena	Nazwa domeny	Określa nazwę domeny alarmowej, której alarmy są pokazywane w tabeli.
Grupy	Tekst	Określa początkowy filtr selekcji grup pokazywanych alarmów. Użytkownik może zmienić w trakcie pracy ten filtr przy pomocy okienka *Filtrowanie grup alarmów*. Definicja właściwości ma postać tekstu złożonego z par *<identyfikator_atrybutu_grupującego>=<wartość_atrybutu>* rozdzielonych znakami średnika. Przykład: Strefa="B1";Strefa="B2”;Typ="Elektryczna"; Pokazywane będą alarmy, których atrybut *Strefa* jest równy *B1* lub *B2* i atrybut *Typ* jest równy *Elektryczna*.
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki komunikatów	Liczba	Określa rozmiar czcionki używanej do wyświetlania informacji w wierszach alarmów.
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, jeżeli rozmiar obiektu zmienia się w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba wierszy - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.

Elementy paska narzędziowego	Typ wyliczeniowy wielokrotny: Brak paska, Kryteria selekcji, Wyszukiwanie opisu, Wykonaj akcję, Pierwszy alarm, Poprzednia strona, Poprzedni alarm, Następny alarm, Następna strona, Ostatni alarm, Szczegóły, Drukowanie	*Określa zawartość paska narzędziowego tabeli. Dostępne są poniższe typy przycisków:* • *Kryteria selekcji* - otwarcie okna selekcji pokazywanych alarmów; • *Wyszukiwanie opisu* - otwarcie okna filtrowania lub wyszukiwania alarmów według opisów; • • Wykonaj akcję - wykonanie akcji zdefiniowanej w bazie alarmów dla wybranego alarmu; • *Pierwszy alarm*- pozycjonowanie na pierwszym wierszu tabeli, identyczne z przyciśnięciem przycisku *Ctrl-Home;* • *Poprzednia strona* - przejście w górę o ilość wierszy równą wielkości strony, identyczne z przyciśnięciem przycisku *PageUp;* • *Poprzedni alarm* - pozycjonowanie o jeden wiersz w górę, identyczne z przyciśnięciem przycisku strzałki w górę; • *Ostatni alarm*- pozycjonowanie na ostatnim wierszu tabeli, identyczne z przyciśnięciem przycisku *Ctrl-End;* • *Następna strona*- przejście w dół o ilość wierszy równą wielkości strony, identyczne z przyciśnięciem przycisku *PageDown;* • *Następny alarm* - pozycjonowanie o jeden wiersz w dół, identyczne z przyciśnięciem przycisku strzałki w dół; • *Szczegóły* - otwarcie okna szczegółów alarmów, identyczne z podwójnym kliknięciem na wierszu alarmu; • *Drukowanie* - otwarcie okna wydruku alarmów.
Wyrażenie filtrujące	Tekst	Określa początkowy filtr selekcji alarmów. Użytkownik może zmienić w trakcie pracy ten filtr przy pomocy okienka Selekcja alarmów historycznych. Filtr ma postać wyrażenie SQL służącego od odpytania bazy alarmów - najłatwiejszą metodą ustawienia tej właściwości jest skorzystanie z okienka edytora właściwości. Przykład: (Priority = 1 OR Priority = 0) Filtr wybiera tylko alarmy pilne i krytyczne. W przypadku użycia filtrów czasowych, w treści wyrażenia można używać składników dynamicznych postaci '$Time(czas_opc)'. Pozwala to uzyskać początkowy warunek filtrujący zależny od czasu otwarcia diagramu. Przykład: StartTime >= '$Time(NOW-2H)'

		Filtr wybiera alarmy, które rozpoczęły się w ciągu dwóch ostatnich godzin.
Wysokość paska narzędziowego	Liczba	Określa wysokość paska narzędziowego w pikselach. Od wysokości paska zależy wielkość ikon przycisków.
Pokazuj nagłówki kolumn	Logiczna	Określa czy należy pokazywać linię nagłówków kolumn.
Kolejność alarmu	Typ wyliczeniowy: Sekwencja detekcji, Według czasów początków, Według czasów detekcji alarmów, Sekwencja zdarzeń	Określa sposób uporządkowania alarmów w tabeli. W trybie sekwencji detekcji alarmy porządkowane są zgodnie z kolejnością pojawiania się zdarzeń w systemie. W trybie sekwencji zdarzeń o kolejności decydują czasy zdarzeń nadane przez strategie wykrywania alarmów.
Odwróć kolejność alarmów	Logiczna	Określa porządek wyświetlania alarmów. Wartość Tak oznacza, że najnowszy alarm jest wyświetlany w pierwszej (górnej) linii tabeli.
pokazuj wskaźnik wierszy	Logiczna	Powoduje wyświetlenie dodatkowej kolumny służącej do sygnalizacji wybranych wierszy.
Format czasu	Tekst	Określa format wyświetlania kolumn czasu alarmów. Domyślny format *yyyy'-'MM'-'dd' 'HH':'mm':'ss'.'fff* powoduje wyświetlanie czasów w postaci 2011-06-17 11:10:09.345.Można dowolnie skonstruować własny format, używając wybranych oznaczeń pól z domyślnego formatu. Separatory pól mogą być dowolne, ale muszą być umieszczone pomiędzy znakami apostrofów.
Okres aktualizacji danych [s]	Liczba	Określa, jak często odświeżana jest zawartość tabeli. Zwiększenie okresu może mieć korzystny wpływ na ogólną wydajność aplikacji, gdy system charakteryzuje się dużą liczbą zgłaszanych zdarzeń alarmowych.

Właściwości kolumny

Nazwa	Typ wartości	Opis
Typ kolumny	Typ wyliczeniowy: Identyfikator, Priorytet, Kategoria, Status, Opis, Czas zdarzenia, Czas początku, Opis początku, Czas końca, Opis końca, Czas detekcji początku, Czas detekcji końca, Potwierdzony, Czas potwierdzenia, Potwierdzony przez, Potwierdzony na, Filtrowany, Notatka potwierdzenia	Określa rodzaj informacji pokazywanej w kolumnie. Informacje dotyczące alarmu pochodzą z logu alarmów historycznych lub z bazy definicji alarmów. Tabela alarmów nie musi zawierać kolumn pokazujących wszystkie rodzaje informacji. Użytkownik może użyć okienka *Szczegóły alarmu* w celu uzyskania dostępu do wszystkich informacji związanych z alarmem. Dostępne są poniższe typy kolumn: • *Identyfikator* - identyfikator alarmu; • *Priorytet* - priorytet alarmu (krytyczny, komunikat itd.) w postaci ikonki; • *Kategoria* - informacje określające, czy alarmy jest zwykły czy typu powiadomienie; • *Status* - status zdarzenia w postaci ikonki (rozpoczęty, rozpoczęty i potwierdzony, zakończony, zakończony i potwierdzony); • *Opis* - opis tekstowy alarmu, w zależności od tego, czy pokazywane zdarzenie dotyczy początku czy końca alarmów, wyświetlany jest odpowiednio opis początku lub końca; • *Czas zdarzenia* - wyświetla jeden z zarejestrowanych czasów alarmów w zależności od typu zdarzenia. Dla alarmu rozpoczętego i niepotwierdzonego pokazuje czas początku alarmu, dla alarmów rozpoczętych i potwierdzonych pokazuje czas potwierdzenia, dla zakończonych i niepotwierdzonych - czas końca alarmu, a dla zakończonych i potwierdzonych - późniejszy z czasów końca lub potwierdzenia; • *Czas początku* - zarejestrowany czas początku alarmu; • *Opis początku* - opis tekstowy początku alarmu tworzony na podstawie definicji alarmu z uwzględnieniem parametrów zgłoszenia; • *Czas końca* - zarejestrowany czas zakończenia alarmu; • *Opis końca* - opis tekstowy końca alarmu tworzony na podstawie definicji alarmu z uwzględnieniem parametrów zgłoszenia;

		• *Czas detekcji początku* - zarejestrowany czas wykrycia początku alarmu; • *Czas detekcji końca* - zarejestrowany czas wykrycia zakończenia alarmu; • *Potwierdzony* - informacja określająca, czy alarm został potwierdzony przez operatora; • *Czas potwierdzenie* - zarejestrowany czas potwierdzenia alarmu; • *Potwierdzony przez* - identyfikator użytkownika, który potwierdził alarm; • *Potwierdzony na* - nazwa stanowiska, na którym alarm został potwierdzony; • *Filtrowana* - informacja określająca, czy w momencie zgłoszenia alarmu podlegał on mechanizmowi filtrów wykrywania alarmów. • *Notatka potwierdzenia* - notatka wprowadzona przez operatora w momencie potwierdzenia alarmu
Szerokość kolumny [%]	Procent	Określa procentową szerokość kolumny. Wartość 0 oznacza, że szerokość kolumny ma być dobrana automatycznie. Jeżeli tylko jedna kolumna będzie miała szerokość 0, to zajmie ona całe pozostałe miejsce po uwzględnieniu szerokości pozostałych kolumn.
Nagłówek kolumny	Tekst	Określa tekst nagłówka kolumny, jeżeli ma być inny od opisu domyślnego.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła tabeli - używany, gdy ilość alarmów jest mniejsza niż ilość wierszy w tabeli.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze tabeli.
Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.

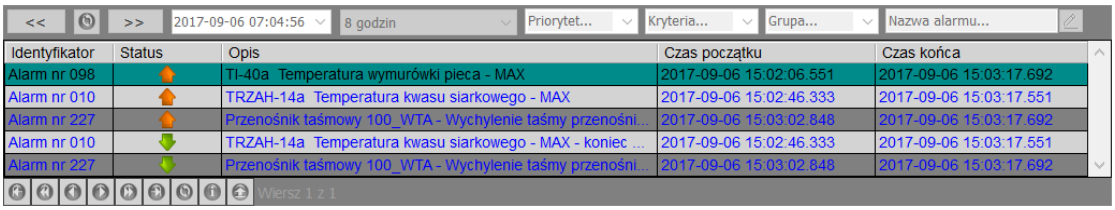
Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.
Kolor komunikatu	Kolor	Określa kolor tekstów dla wierszy alarmów typu komunikat.
Kolor tła komunikatu	Kolor	Określa kolor tła dla wierszy alarmów typu komunikat. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów*.
Kolor ostrzeżenia	Kolor	Określa kolor tekstów dla wierszy alarmów typu ostrzeżenie.
Kolor tła ostrzeżenia	Kolor	Określa kolor tła dla wierszy alarmów typu ostrzeżenie. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor alarmu	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm zwykły.
Kolor tła alarmu	Kolor	Określa kolor tła dla wierszy alarmów typu alarm zwykły. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor alarmu pilnego	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm pilny.
Kolor tła alarmu pilnego	Kolor	Określa kolor tła dla wierszy alarmów typu alarm pilny. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor alarmu krytycznego	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm krytyczny.
Kolor tła alarmu krytycznego	Kolor	Określa kolor tła dla wierszy alarmów typu alarm krytyczny. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor tła alarmów	Kolor	Określa kolor tła dla wierszy nieparzystych.

Naprzemienny kolor tła alarmów	Kolor	Określa kolor tła dla wierszy parzystych. Ustawienie innej wartości niż dla właściwości *Kolor tła alarmów* pozwala na uzyskanie efektu naprzemiennego wyświetlania alarmów w różnych kolorach.
Kolor czcionki wybranego alarmu	Kolor	Określa kolor tekstów dla aktualnie wybranego wiersza. Oznaczenie aktywnego wiersza można wyłączyć poprzez wybór koloru Transparent.
Kolor tła wybranego alarmu	Kolor	Określa kolor tła dla aktualnie wybranego wiersza. Oznaczenie aktywnego wiersza można wyłączyć poprzez wybór koloru Transparent.
Kolor tła wskaźnika wierszy	Kolor	Określa kolor tła dla komórek kolumny wskaźnika wierszy.
Kolor tła wskaźnika wybranego wiersza	Kolor	Określa kolor tła dla komórki wskaźnika aktualnie wybranego wiersza.
Kolor przycisków	Kolor	Określa kolor tła przycisków paska narzędziowego.
Ramka tabeli widoczna	Logiczna	Określa, czy wyświetlać linię obramowania tabeli.

Obiekt Tabela alarmów SQL

Funkcje obiektu

Obiekt *Tabela alarmów SQL* jest obiektem umożliwiającym wyświetlanie i zaawansowane filtrowanie alarmów historycznych zarejestrowanych w bazie MS SQL. Tabela pokazuje alarmy z wybranej domeny alarmowej. Aby alarmy zdarzenia były dostępne, to w domenie alarmów musi być zdefiniowana co najmniej jedna baza SQL, do której są zapisywane zdarzenia historyczne alarmów. Zestaw danych pokazywanych w tabeli jest swobodnie definiowany. Możliwe jest użycie zintegrowanego paska narzędziowego - zestaw dostępnych funkcji jest parametryzowany.



Rys. Przykłady obiektu Tabela alarmów SQL.

Obiekt *Tabela alarmów historycznych* jest obiektem typu kontrolkowego.

Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Domena	Nazwa domeny	Określa nazwę domeny alarmowej, której alarmy są pokazywane w tabeli.

Uporządkowanie alarmów	Typ wyliczeniowy: Sekwencja zdarzeń, Sekwencja detekcji zdarzeń, Czas alarmu, Czas detekcji alarmu	Określa sposób uporządkowania alarmów w tabeli. W trybie sekwencji zdarzeń i sekwencji detekcji zdarzeń wiersze zawierają pełną informacje o wystąpieniu danego alarmu (początek, koniec, potwierdzenie). W trybie czasu alarmu i czasu detekcji alarmu każdy wiersz oznacza osobne zdarzenie alarmu.
Horyzont czasu	Typ wyliczeniowy: od 5 minut do 1 roku	Określa długość odcinka czasu, za jaki wyświetlane są zdarzenia alarmów.
Czas	Data i czas	Określa czas początkowy odcinka czasu, za jaki wyświetlane są zdarzenia alarmów.
Filtr priorytetu alarmu	Typ wyliczeniowy wielokrotny: Komunikat, Ostrzeżenie, Alarm, Pilny, Krytyczny	Określa początkowy stan priorytetów pokazywanych alarmów. Operator może zmienić ustawienie poprzez przycisk selekcji priorytetu paska narzędziowego.
Kryteria	Typ wyliczeniowy: Zwykły, Powiadomienie, Początki, Końce, Potwierdzone, Niepotwierdzone	Określa dodatkowe kryteria filtrowaniazdarzeń alarmów
Grupa	Tekst	Określa początkowy filtr selekcji grupy pokazywanych alarmów. Użytkownik może zmienić w trakcie pracy ten filtr przy pomocy górnego paska narzędziowego. Definicja właściwości ma postać tekstu złożonego z par *<identyfikator_atrybutu_grupującego>=<wartość_atrybutu>* rozdzielonych znakami średnika. Przykład: Strefa="B1";Strefa="B2”;Typ="Elektryczna";

		Pokazywane będą alarmy, których atrybut *Strefa* jest równy *B1* lub *B2* i atrybut *Typ* jest równy *Elektryczna*.
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki komunikatów	Liczba	Określa rozmiar czcionki używanej do wyświetlania informacji w wierszach alarmów.
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, jeżeli rozmiar obiektu zmienia się w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba wierszy - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.
Wyrażenie filtrujące	Tekst	Określa początkowy filtr selekcji alarmów. Użytkownik może zmienić w trakcie pracy ten filtr przy pomocy okienka *Selekcja alarmów historycznych*. Filtr ma postać wyrażenia SQL służącego od odpytania bazy alarmów - najłatwiejszą metodą ustawienia tej właściwości jest skorzystanie z okienka edytora właściwości. Przykład: (Priority = 1 OR Priority = 0) Filtr wybiera tylko alarmy pilne i krytyczne. W przypadku użycia filtrów czasowych, w treści wyrażenia można używać składników dynamicznych postaci '$Time(czas_opc)'. Pozwala to uzyskać początkowy warunek filtrujący, zależny od czasu otwarcia diagramu. Przykład: StartTime >= '$Time(NOW-2H)' *Filtr wybiera alarmy, które rozpoczęły się w ciągu dwóch ostatnich godzin.*
Wysokość paska narzędziowego	Liczba	Określa wysokość paska narzędziowego w pikselach. Od wysokości paska zależy wielkość ikon przycisków.
Pokazuj nagłówki kolumn	Logiczna	Określa, czy należy pokazywać linię nagłówków kolumn.

Odwróć kolejność alarmów	Logiczna	Określa porządek wyświetlania alarmów. Wartość Tak oznacza, że najnowszy alarm jest wyświetlany w pierwszej (górnej) linii tabeli.
pokazuj wskaźnik wierszy	Logiczna	Powoduje wyświetlenie dodatkowej kolumny służącej do sygnalizacji wybranych wierszy.
Format czasu	Tekst	Określa format wyświetlania kolumn czasu alarmów. Domyślny format *yyyy'-'MM'-'dd' 'HH':'mm':'ss'.'fff* powoduje wyświetlanie czasów w postaci 2011-06-17 11:10:09.345.Można dowolnie skonstruować własny format, używając wybranych oznaczeń pól z domyślnego formatu. Separatory pól mogą być dowolne, ale muszą być umieszczone pomiędzy znakami apostrofów.
Pasek filtrów widoczny	*Logiczna*	Określa, czy widoczny jest górny pasek umożliwiający filtrowanie zdarzeń.
Wysokość paska narzędziowego	*Liczba*	Określa wysokość w pikselach dolnego paska narzędziowego (0 – brak paska)

Właściwości kolumny

Nazwa	Typ wartości	Opis
Typ kolumny	Typ wyliczeniowy: Identyfikator, Priorytet, Kategoria, Status, Opis, Czas zdarzenia, Czas początku, Opis początku, Czas końca, Opis końca, Czas detekcji początku, Czas detekcji końca, Potwierdzony, Czas potwierdzenia, Potwierdzony przez, Potwierdzony na, Filtrowany, Notatka potwierdzenia	Określa rodzaj informacji pokazywanej w kolumnie. Informacje dotyczące alarmu pochodzą z logu alarmów historycznych lub z bazy definicji alarmów. Tabela alarmów nie musi zawierać kolumn pokazujących wszystkie rodzaje informacji. Użytkownik może użyć okienka *Szczegóły alarmu* w celu uzyskania dostępu do wszystkich informacji związanych z alarmem. Dostępne są poniższe typy kolumn: • *Identyfikator* - identyfikator alarmu; • *Priorytet* - priorytet alarmu (krytyczny, komunikat itd.) w postaci ikonki; • *Kategoria* - informacje określające, czy alarmy jest zwykły czy typu powiadomienie; • *Status* - status zdarzenia w postaci ikonki (rozpoczęty, rozpoczęty i potwierdzony, zakończony, zakończony i potwierdzony); • *Opis* - opis tekstowy alarmu, w zależności od tego, czy pokazywane zdarzenie dotyczy początku czy końca alarmów, wyświetlany jest odpowiednio opis początku lub końca; • *Czas zdarzenia* - wyświetla jeden z zarejestrowanych czasów alarmów w zależności od typu zdarzenia. Dla alarmu rozpoczętego i niepotwierdzonego pokazuje czas początku alarmu, dla alarmów rozpoczętych i potwierdzonych pokazuje czas potwierdzenia, dla zakończonych i niepotwierdzonych - czas końca alarmu, a dla zakończonych i potwierdzonych - późniejszy z czasów końca lub potwierdzenia; • *Czas początku* - zarejestrowany czas początku alarmu; • *Opis początku* - opis tekstowy początku alarmu tworzony na podstawie definicji alarmu z uwzględnieniem parametrów zgłoszenia; • *Czas końca* - zarejestrowany czas zakończenia alarmu; • *Opis końca* - opis tekstowy końca alarmu tworzony na podstawie definicji alarmu z uwzględnieniem parametrów zgłoszenia;

		• *Czas detekcji początku* - zarejestrowany czas wykrycia początku alarmu; • *Czas detekcji końca* - zarejestrowany czas wykrycia zakończenia alarmu; • *Potwierdzony* - informacja określająca, czy alarm został potwierdzony przez operatora; • *Czas potwierdzenie* - zarejestrowany czas potwierdzenia alarmu; • *Potwierdzony przez* - identyfikator użytkownika, który potwierdził alarm; • *Potwierdzony na* - nazwa stanowiska, na którym alarm został potwierdzony; • *Filtrowana* - informacja określająca, czy w momencie zgłoszenia alarmu podlegał on mechanizmowi filtrów wykrywania alarmów. • *Notatka potwierdzenia* - notatka wprowadzona przez operatora w momencie potwierdzenia alarmu
Szerokość kolumny [%]	Procent	Określa procentową szerokość kolumny. Wartość 0 oznacza, że szerokość kolumny ma być dobrana automatycznie. Jeżeli tylko jedna kolumna będzie miała szerokość 0, to zajmie ona całe pozostałe miejsce po uwzględnieniu szerokości pozostałych kolumn.
Nagłówek kolumny	Tekst	Określa tekst nagłówka kolumny, jeżeli ma być inny od opisu domyślnego.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor tła	Kolor	Określa kolor tła tabeli - używany, gdy ilość alarmów jest mniejsza niż ilość wierszy w tabeli.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze tabeli.
Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.

Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.
Kolor komunikatu	Kolor	Określa kolor tekstów dla wierszy alarmów typu komunikat.
Kolor tła komunikatu	Kolor	Określa kolor tła dla wierszy alarmów typu komunikat. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów*.
Kolor ostrzeżenia	Kolor	Określa kolor tekstów dla wierszy alarmów typu ostrzeżenie.
Kolor tła ostrzeżenia	Kolor	Określa kolor tła dla wierszy alarmów typu ostrzeżenie. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor alarmu	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm zwykły.
Kolor tła alarmu	Kolor	Określa kolor tła dla wierszy alarmów typu alarm zwykły. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor alarmu pilnego	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm pilny.
Kolor tła alarmu pilnego	Kolor	Określa kolor tła dla wierszy alarmów typu alarm pilny. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor alarmu krytycznego	Kolor	Określa kolor tekstów dla wierszy alarmów typu alarm krytyczny.
Kolor tła alarmu krytycznego	Kolor	Określa kolor tła dla wierszy alarmów typu alarm krytyczny. Wybranie koloru *Transparent* oznacza, że kolor tła określany jest na podstawie właściwości *Kolor tła alarmów* i *Naprzemienny kolor tła alarmów.*
Kolor tła alarmów	Kolor	Określa kolor tła dla wierszy nieparzystych.

Naprzemienny kolor tła alarmów	Kolor	Określa kolor tła dla wierszy parzystych. Ustawienie innej wartości niż dla właściwości *Kolor tła alarmów* pozwala na uzyskanie efektu naprzemiennego wyświetlania alarmów w różnych kolorach.
Kolor czcionki wybranego alarmu	Kolor	Określa kolor tekstów dla aktualnie wybranego wiersza. Oznaczenie aktywnego wiersza można wyłączyć poprzez wybór koloru Transparent.
Kolor tła wybranego alarmu	Kolor	Określa kolor tła dla aktualnie wybranego wiersza. Oznaczenie aktywnego wiersza można wyłączyć poprzez wybór koloru Transparent.
Kolor tła wskaźnika wierszy	Kolor	Określa kolor tła dla komórek kolumny wskaźnika wierszy.
Kolor tła wskaźnika wybranego wiersza	Kolor	Określa kolor tła dla komórki wskaźnika aktualnie wybranego wiersza.
Kolor przycisków	Kolor	Określa kolor tła przycisków paska narzędziowego.
Ramka tabeli widoczna	Logiczna	Określa, czy wyświetlać linię obramowania tabeli.

Obiekt Tabela Asbase

Funkcje obiektu

Obiekt *Tabela Asbase* jest obiektem umożliwiającym wyświetlanie w postaci tabelarycznej danych pochodzących z bazy danych programu AsBase.

Obiekt pozwala też na prace w tzw. trybie mieszanym. W tym trybie, w tabeli wyświetlane są tylko niektóre pola zestawów rejestracji i receptur. Wartości innych pól mogą być wyświetlane za pomocą innych obiektów poprzez użycie zmiennych procesowych podanych w definicji zestawu zmiennych program AsBase. W trybie mieszanym, tabela wyświetla wszystkie rekordy danych spełniające zadane kryteria, a inne obiekty wyświetlają wartości pól pojedynczego rekordu, skojarzonego z aktualnie wybranym wierszem tabeli.

SN	PNC	Temperatura MAX	Obroty MAX	Stanowisko	Data	Godzina startu	Godzina końca
34600412	914906449	0.0	1152	L20_4	16 listopada 2013	11:25:46	11:45:16
34600454	914906449	0.0	1158	L20_3	16 listopada 2013	11:22:35	11:45:04
34600510	914906449	0.0	1152	L20_2	16 listopada 2013	11:21:54	11:36:13
34600541	914906449	41.1	1152	L20_1	16 listopada 2013	11:18:06	12:10:40
34600005	914907045	0.0	798	L10_1	16 listopada 2013	2013-11-16 11:18:06	11:22:18
34600073	914907045	0.0	942	L10_4	16 listopada 2013	11:05:08	11:19:16
34600004	914906614	0.0	1518	L10_8	16 listopada 2013	10:56:31	11:09:02
34600171	914907045	0.0	942	L10_6	16 listopada 2013	10:54:49	11:07:50
34600002	914907047	0.0	942	L10_10	16 listopada 2013	10:52:11	11:09:40
34600004	914907427	0.0	1152	L10_9	16 listopada 2013	10:51:24	10:57:56
34600002	914907034	0.0	1104	L10_3	16 listopada 2013	10:24:35	10:37:46
34600101	914907045	20.4	948	L10_5	16 listopada 2013	10:21:35	10:39:28
34500079	913101420	0.0	960	TOP_8	08 listopada 2013	13:42:07	13:57:59
34500079	913101420	25.1	960	TOP_7	08 listopada 2013	13:22:47	13:39:11
34500079	913101420	0.0	960	TOP_5	08 listopada 2013	13:03:18	13:19:45
34500079	913101420	0.0	960	TOP_4	08 listopada 2013	12:39:12	12:54:23

Rys. Przykłady obiektu Tabela Asbase.

Obiekt posiada własne mechanizmy interakcyjne pozwalające na poruszanie się w ramach wierszy tabeli, ich filtrowanie i sortowanie, ładowanie definicji receptury do zmiennych procesowych, zapisanie zawartości tabeli do pliku tekstowego. Za pośrednictwem akcji operatorskiej ObjectAction można także dopisywać, usuwać i modyfikować rekordy wyświetlane w tabeli.

Obiekt może wyświetlać własny pasek narzędziowy. Funkcje paska mogą być też wykonane za pośrednictwem akcji operatorskiej ObjectAction.

Właściwości

Zobacz: **Elementy wizualizacji >** **Standardowe właściwości obiektów**

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Typ źródła danych	Typ wyliczeniowy: Archiwum, Receptury, Ładowania receptur	Określa typ źródła danych pobieranych z bazy danych AsBase.
Identyfikator źródła danych	Tekst	Identyfikator podstawowego źródła danych, w zależności od wartości *Typ źródła danych*, jest to identyfikator zestawu rejestracji lub grupy receptur.
Identyfikator podrzędnego źródła danych	Tekst	Identyfikator podrzędnego źródła danych, czyli źródła, które w parametryzacji programu AsBase zostało zadeklarowane jako pozostające w relacji ze źródłem podstawowym.
Identyfikator zestawu zmiennych	Tekst	Opcjonalny identyfikator zestawu zmiennych dla źródła podstawowego, do którego zmiennych będą zapisywane wartości pól bieżącego wiersza tabeli.
Identyfikator podrzędnego zestawu zmiennych	Tekst	Opcjonalny identyfikator zestawu zmiennych dla źródła podrzędnego, do którego zmiennych będą zapisywane wartości pól bieżącego wiersza tabeli.
Dolny limit czasu	Tekst	Opcjonalne dolne ograniczenie czasu zapisu dla rekordów danych wyświetlanych w tabeli. Właściwości nie należy używać w przypadku, gdy typem źródła danych są receptury. Możliwe jest stosowanie dowolnego wyrażenia zwracającego wynik w postaci wartości typu DateTime.

		Można też deklarować ograniczenie w postaci tekstu konwertowalnego na DateTime, także w konwencji OPC. Na przykład, wartość „DAY-2D” oznacza wyświetlanie danych zapisanych maksymalnie dwa dni przed dniem aktualnym.
Górny limit czasu	Tekst	Opcjonalne górne ograniczenie czasu zapisu dla rekordów danych wyświetlanych w tabeli. Wszystkie zasady użycia są identyczne jak dla właściwości *Dolny limit czasu*.
Wyrażenie filtrujące	Tekst	Wyrażenie filtrujące, które pozwala na ograniczenie zakresu danych dostępnych w tabeli. Szczegóły dotyczące składni filtra znajdują się w rozdziale *Współpraca z programem AsBase*. W przypadku filtrowania danych pod względem czasu zapisu zalecane jest stosowanie właściwości *Dolny limit czasu* i *Górny limit czasu*.
Wyrażenie sortujące	Tekst	Wyrażenie określające początkowe uporządkowanie rekordów. Jeżeli właściwość nie jest zdefiniowana to dane są sortowane automatycznie według czasów rejestracji lub nazw receptur (w zależności od typu źródła danych). Wartość właściwość musi być zgodną ze składnią klauzuli ORDER BY języka SQL. Zasady nazewnictwa pól są identyczne jak w przypadku wyrażenia filtrującego. Przykład: V_PROD_LENGTH DESC Sortowanie malejące według wartość pola o identyfikatorze PROD_LENGTH
Limit wierszy	Liczba	Określa maksymalną liczbę wierszy wyświetlanych w tabeli. Wartość 0 oznacza, że liczba wierszy nie jest limitowana i wynika wyłącznie z liczby rekordów zapisanych w tabelach źródeł danych i podanego wyrażenia filtrującego.

Identyfikator zestawu zmiennych ładowań	Tekst	Opcjonalny identyfikator zestawu zmiennych, do którego będą zapisywane wartości receptury w efekcie wykonania komendy ładowania. Komenda ładowania dostępna jest w pasku narzędziowym lub można wykorzystać do tego celu akcję operatorską *ObjectAction*. Funkcjonalność jest dostępna tylko wtedy, gdy źródło podstawowe jest recepturą. Wykonanie operacji ładowania jest rejestrowane w tabeli ładowań w bazie danych AsBase.
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki danych	Liczba	Określa rozmiar czcionki używanej do wyświetlania informacji w wierszach wartości.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania wszystkich tekstów w tabeli.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, jeżeli rozmiar obiektu zmienia się w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba wierszy - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.
Pokazuj nagłówki kolumn	Logiczna	Określa, czy należy pokazywać linię nagłówków kolumn.
Automatyczna szerokość kolumn	Logiczna	Określa sposób wyliczenia szerokości kolumn. Dla wartości Tak wszystkie kolumny muszą się zmieścić w obszarze obiektu. Dla wartości Nie, jeżeli sumaryczna szerokość kolumn przekroczy szerokość obiektu, to wyświetlany jest poziomy suwak.

		W obu przypadkach znaczenie ma także właściwość kolumny *Szerokość kolumny [%]*.
Zezwalaj na filtrowanie	Logiczna	Określa, czy odblokowany zostanie mechanizm ustawiania filtrów kolumn, pozwalający użytkownikowi aplikacji na interaktywną selekcję wyświetlanych rekordów.
Wysokość paska narzędziowego	Liczba	Określa wysokość paska narzędziowego tabeli. Wysokość równa 0 oznacza wyłączenie pokazywania paska.
Pokazuj przycisk ładowania	Logiczna	Określa, czy na pasku narzędziowym wyświetlony zostanie przycisk załadowania rekordu.
Pokazuj przycisk eksportu	Logiczna	Określa, czy na pasku narzędziowym wyświetlony zostanie przycisk zapisu zawartości tabeli do pliku tekstowego.
Pokazuj przycisk zakresu czasu	Logiczna	Określa, czy na pasku narzędziowym wyświetlony zostanie przycisk, którego użycie spowoduje wyświetlenie okienka pozwalającego operatorowi na zmianę zakresu czasu danych wyświetlanych w tabeli. Pozwala to zmienić ograniczenie wstępnie ustawione za pomocą właściwości *Dolny limit czasu* i *Górny limit czasu*.

Właściwości kolumny

Nazwa	**Typ wartości**	**Opis**
Kolumna widoczna	Logiczna	Określa, czy kolumna jest widoczna. Właściwość pozwala na wybór wyświetlanych kolumn w trybie wykonania aplikacji.
Pole tablicy nadrzędnej	Tekst	Określa, czy pole pochodzie z podstawowego (nadrzędnego) źródła danych, czy ze źródła podrzędnego.
Identyfikator pola	Tekst	Identyfikator pola, zgodnie z definicja zapisaną w bazie danych AsBase, którego wartości będą wyświetlane w kolumnie.
Szerokość kolumny [%]	Procent	Określa procentową szerokość kolumny. Wartość 0 oznacza, że szerokość kolumny ma być wyliczona na podstawie szerokości obiektu, szerokości innych kolumn i wybranego trybu pracy we właściwości *Automatyczna szerokość kolumn*.

		W przypadku pracy w trybie automatycznej szerokości, suma wartości właściwości dla wszystkich kolumn nie może przekroczyć 100%. Kolumny o szerokości 0 zajmują całą pozostałą szerokość obiektu. W szczególności, jeżeli tylko jedna kolumna będzie miała szerokość 0, to zajmie ona całe pozostałe miejsce po uwzględnieniu szerokości pozostałych kolumn. W trybie bez automatycznego wyliczania szerokości, jeżeli suma szerokości kolumn przekroczy 100%, to wyświetlony zostanie poziomy suwak.
Nagłówek kolumny	Tekst	Określa tekst nagłówka kolumny. Jeżeli właściwość nie zostanie zdefiniowana, to treść nagłówka zostanie określona na podstawie definicji pola zapisanej w bazie danych AsBase.
Wyrównywanie tekstu w poziomie	Typ wyliczeniowy: Do lewej, Wyśrodkowanie, Do prawej	Określa sposób wyrównywania tekstów w komórkach wierszy tabeli.
Format wartości pola	Tekst	Określa sposób konwersji wartości pola na tekst. Napis formatujący musi być zgodny z formatem platformy .Net. Dokładniejszy opis sposobów formatowania znajduje się w opisie funkcji *Format*. Przykład: Numer {0:d4} Rezultatem powyższego formatowania jest połączenie teksu *Numer* z 4-znakową liczbą wypełnioną zerami od lewej strony, np. *Numer 0001*
Kolor czcionki	Kolor	Określa kolor tekstów wyświetlanych w komórkach wierszy tabeli. Jeżeli właściwość nie jest ustawiona, to obowiązuje wartość właściwości stanowej *Kolor czcionki wierszy*.
Kolor tła	Kolor	Określa kolor tła komórek wierszy tabeli. Jeżeli właściwość nie jest ustawiona, to obowiązuje wartość właściwości stanowej *Kolor tła wierszy*.
Akcja	Akcja operatorska	Pozwala zdefiniować akcję operatorską, która zostanie wykonana, gdy operator kliknie w zawartość komórki kolumny. Może służyć np. do wyświetlenia okienka ze szczegółami lub do

		drążenia danych. W akcji można odwołać się za pomocą funkcji EventContext do danych szczegółowych zwiazanych z kliknięciem: ColumnIndex, RowIndex, FieldName, MappedVariable, CellValue.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor tła	Kolor	Określa kolor tła tabeli - używany, gdy wiersze wartości nie zajmują całego obszaru tabeli.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze tabeli.
Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.
Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.
Kolor czcionki wierszy	Kolor	Określa kolor tekstów wyświetlanych w komórkach wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor czcionki* nie została ustawiona.
Kolor tła wierszy	Kolor	Określa kolor tła komórek nieparzystych wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor tła* nie została ustawiona.
Naprzemienny kolor tła wierszy	Kolor	Określa kolor tła komórek parzystych wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor tła* nie została ustawiona.
Kolor czcionki wybranego wiersza	Kolor	Określa kolor tekstów wyświetlanych w komórkach aktualnie wybranego wiersza tabeli.
Kolor tła wybranego wiersza	Kolor	Określa kolor tła komórek aktualnie wybranego wiersza tabeli.

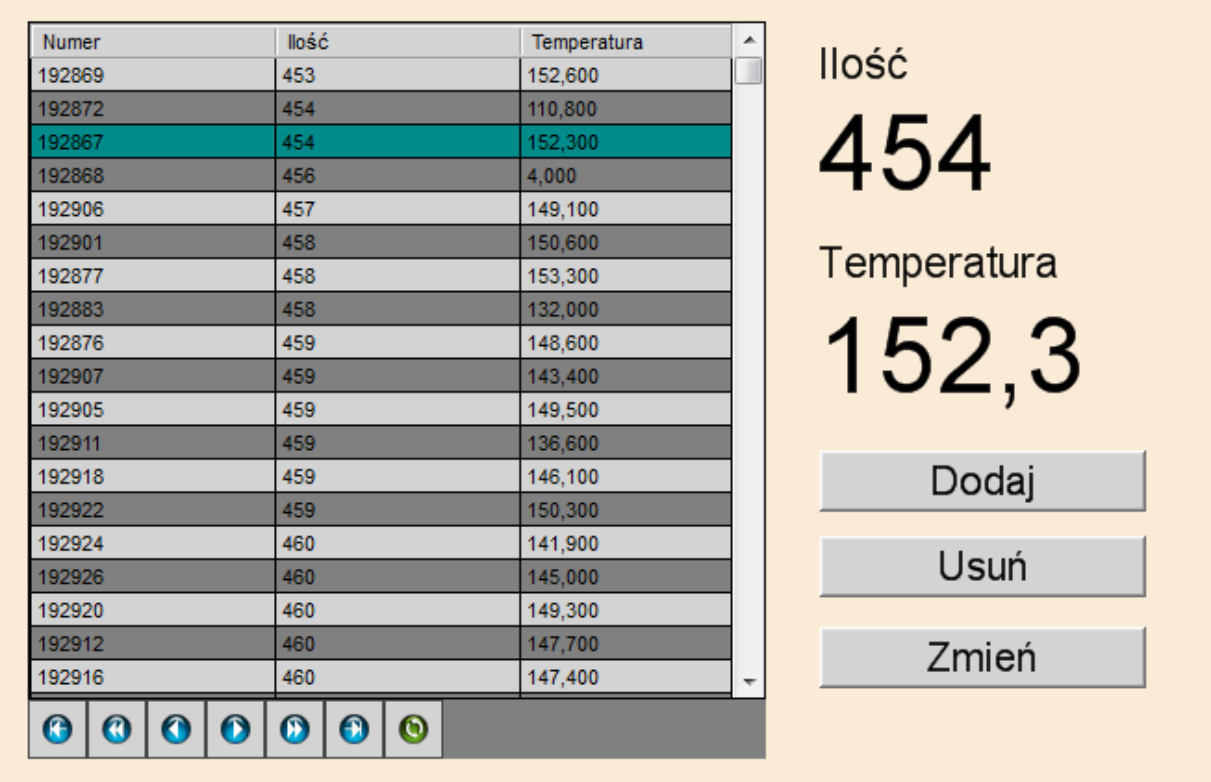
Kolor przycisków	Kolor	Określa kolor przycisków paska narzędziowego.

Obiekt Tabela Danych

Funkcje obiektu

Obiekt *Tabela Danych* jest obiektem umożliwiającym wyświetlanie w postaci tabelarycznej danych pochodzących z dowolnej bazy danych serwera MS SQL.

Obiekt pozwala też na prace w tzw. trybie mieszanym. W tym trybie, w tabeli wyświetlane są tylko niektóre, wybrane pola tabeli. Wartości innych pól mogą być wyświetlane za pomocą innych obiektów poprzez użycie zmiennych procesowych podłączonych w definicji obiektu do tych pól. W trybie mieszanym, tabela wyświetla wszystkie rekordy danych spełniające zadane kryteria, a inne obiekty wyświetlają wartości pól pojedynczego rekordu, skojarzonego z aktualnie wybranym wierszem tabeli.



Rys. Przykład obiektu Tabela Danych z elementami pomocniczymi.

Obiekt posiada własne mechanizmy interakcyjne pozwalające na poruszanie się w ramach wierszy tabeli, ich filtrowanie i sortowanie, zapisanie zawartości tabeli do pliku tekstowego.

Możliwa jest zmiana rekordów pokazywanej tabeli. Odbywa się to za pośrednictwem akcji operatorskiej ObjectAction w wariantach $Add, $Delete i $Update oraz zmiennych procesowych podłączonych do pól tabeli.

Obiekt może wyświetlać własny pasek narzędziowy. Funkcje paska mogą być też wykonane za pośrednictwem akcji operatorskiej ObjectAction.

Właściwości

Zobacz: **Elementy wizualizacji >** **Standardowe właściwości obiektów**

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Źródło danych	DataTable	Pozwala ustanowić ogólne źródło danych dla tabeli w postaci typu wartości DataTable dostarczonego za pomocą funkcji AppData(..) lub jako wartość zmiennej. Dana aplikacyjna (AppData) lub zmienna może być ustawiana za pomocą skryptu jako wartość typu DataTable, zawierająca zdefiniowane kolumny i wiersze. Każda kolumna tabeli może się odwołać do odpowiedniej kolumny obiektu DataTable.
Parametry połączenia	Tekst	Określa parametry połączenia z serwerem SQL i nazwę bazy danych (connection string). Poprawną wartość właściwości najłatwiej wprowadzić przy pomocy edytora właściwości.
Nazwa tabeli	Tekst	Nazwa tabeli, z której będą pochodzić pokazywane dane. Można też podać nazwę widoku.
Nazwa pola klucza	Tekst	Nazwa pola tabeli, które jest jej polem klucza. Podanie tej właściwości jest konieczne, jeżeli obiekt ma pozwolić na edycję danych.
Wyrażenie filtrujące	Tekst	Wyrażenie filtrujące, które pozwala na ograniczenie zakresu danych dostępnych w tabeli. Jest podawane w składni języka SQL (klauzula where).

Wyrażenie sortujące	Tekst	Wyrażenie określające początkowe uporządkowanie rekordów. Jest podawane w składni języka SQL (klauzula order by).
Limit wierszy	Liczba	Określa maksymalną liczbę wierszy wyświetlanych w tabeli. Wartość 0 oznacza, że liczba wierszy nie jest limitowana i wynika wyłącznie z liczby rekordów zapisanych w tabeli i podanego wyrażenia filtrującego.
Wyzwolenie przeładowania	Logiczna	Właściowość pozwala na implementację mechanizmu automatycznego odświeżania zawartości tabeli. Zmiana wartości z *false* na *true* powoduje ponowny odczyt danych z serwera SQL.
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki danych	Liczba	Określa rozmiar czcionki używanej do wyświetlania informacji w wierszach wartości.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania wszystkich tekstów w tabeli.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, jeżeli rozmiar obiektu zmienia się w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba wierszy - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.
Pokazuj nagłówki kolumn	Logiczna	Określa, czy należy pokazywać linię nagłówków kolumn.
Automatyczna szerokość kolumn	Logiczna	Określa sposób wyliczenia szerokości kolumn. Dla wartości Tak wszystkie kolumny muszą się zmieścić w obszarze obiektu. Dla wartości Nie, jeżeli sumaryczna

		szerokość kolumn przekroczy szerokość obiektu, to wyświetlany jest poziomy suwak. W obu przypadkach znaczenie ma także właściwość kolumny *Szerokość kolumny [%]*.
Zezwalaj na filtrowanie	Logiczna	Określa, czy odblokowany zostanie mechanizm ustawiania filtrów kolumn, pozwalający użytkownikowi aplikacji na interaktywną selekcję wyświetlanych rekordów.
Wysokość paska narzędziowego	Liczba	Określa wysokość paska narzędziowego tabeli. Wysokość równa 0 oznacza wyłączenie pokazywania paska.
Pokazuj przycisk eksportu	Logiczna	Określa, czy na pasku narzędziowym wyświetlony zostanie przycisk zapisu zawartości tabeli do pliku tekstowego.

Właściwości kolumny

Nazwa	**Typ wartości**	**Opis**
Kolumna widoczna	Logiczna	Określa, czy kolumna jest widoczna. Właściwość pozwala na wybór wyświetlanych kolumn w trybie wykonania aplikacji. Pozwala ponadto na zamapowanie nazwy zmiennej procesowej do pola tabeli danych bez pokazywania kolumny z danymi.
Nazwa pola	Tekst	Określa, nazwę pola w tabeli danych(kolumny SQL) , którego wartości będą pokazywane w kolumnie.
Nazwa tabeli słownika	Tekst	Służy do określenia nazwy tabeli SQL, która posłuży do translacji oryginalnej wartości pola kolumny na jej alternatywna postać tekstową.
Nazwa pola klucza słownika	Tekst	Jeżeli tabela słownikowa została podana, to właściwość określa nazwę jej kolumny klucza translacji. Na podstawie wartości pola, w tabeli słownika wyszukiwany jest rekord o zgodnej wartości w kolumnie klucza.

Nazwa pola wartości słownika	Tekst	Określa nazwę kolumny tabeli słownika, z której pobierana jest postać tekstowa wartości pola.
Zmienna mapowana	Tekst	Nazwa zmiennej procesowej, która jest ustawiana zgodnie z wartością pola kolumny w rekordzie bieżącym tabeli. Zmienne mapowana są niezbędne także w przypadku używania mechanizmów edycji danych tabeli.
Szerokość kolumny [%]	Procent	Określa procentową szerokość kolumny. Wartość 0 oznacza, że szerokość kolumny ma być wyliczona na podstawie szerokości obiektu, szerokości innych kolumn i wybranego trybu pracy we właściwości *Automatyczna szerokość kolumn*. W przypadku pracy w trybie automatycznej szerokości, suma wartości właściwości dla wszystkich kolumn nie może przekroczyć 100%. Kolumny o szerokości 0 zajmują całą pozostałą szerokość obiektu. W szczególności, jeżeli tylko jedna kolumna będzie miała szerokość 0, to zajmie ona całe pozostałe miejsce po uwzględnieniu szerokości pozostałych kolumn. W trybie bez automatycznego wyliczania szerokości, jeżeli suma szerokości kolumn przekroczy 100%, to wyświetlony zostanie poziomy suwak.
Nagłówek kolumny	Tekst	Określa tekst nagłówka kolumny. Jeżeli właściwość nie zostanie zdefiniowana, to treść nagłówka zostanie określona na podstawie nazwy pola.
Wyrównywanie tekstu w poziomie	Typ wyliczeniowy: Do lewej, Wyśrodkowanie, Do prawej	Określa sposób wyrównywania tekstów w komórkach wierszy tabeli.
Format wartości pola	Tekst	Określa sposób konwersji wartości pola na tekst. Napis formatujący musi być zgodny z formatem platformy .Net. Dokładniejszy opis sposobów formatowania znajduje się w opisie funkcji *Format*. Przykład: Numer {0:d4}

		Rezultatem powyższego formatowania jest połączenie teksu *Numer* z 4-znakową liczbą wypełnioną zerami od lewej strony, np. *Numer 0001*
Tekst zastępczy	Tekst	Tekst, który zostanie wyświetlony w komórkach kolumny dla których brak danych.
Kolor czcionki	Kolor	Określa kolor tekstów wyświetlanych w komórkach wierszy tabeli. Jeżeli właściwość nie jest ustawiona, to obowiązuje wartość właściwości stanowej *Kolor czcionki wierszy*.
Kolor tła	Kolor	Określa kolor tła komórek wierszy tabeli. Jeżeli właściwość nie jest ustawiona, to obowiązuje wartość właściwości stanowej *Kolor tła wierszy*.
Akcja	Akcja operatorska	Pozwala zdefiniować akcję operatorską, która zostanie wykonana, gdy operator kliknie w zawartość komórki kolumny. Może służyć np. do wyświetlenia okienka ze szczegółami lub do drążenia danych. W akcji można odwołać się za pomocą funkcji EventContext do danych szczegółowych zwiazanaych z kliknięciem: ColumnIndex, RowIndex, FieldName, MappedVariable, CellValue.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor tła	Kolor	Określa kolor tła tabeli - używany, gdy wiersze wartości nie zajmują całego obszaru tabeli.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze tabeli.
Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.
Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.
Kolor czcionki wierszy	Kolor	Określa kolor tekstów wyświetlanych w komórkach wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor czcionki* nie została ustawiona.

Kolor tła wierszy	Kolor	Określa kolor tła komórek nieparzystych wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor tła* nie została ustawiona.
Naprzemienny kolor tła wierszy	Kolor	Określa kolor tła komórek parzystych wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor tła* nie została ustawiona.
Kolor czcionki wybranego wiersza	Kolor	Określa kolor tekstów wyświetlanych w komórkach aktualnie wybranego wiersza tabeli.
Kolor tła wybranego wiersza	Kolor	Określa kolor tła komórek aktualnie wybranego wiersza tabeli.
Kolor przycisków	Kolor	Określa kolor przycisków paska narzędziowego.

Obiekt Tabela Zmiennych

Funkcje obiektu

Obiekt *Tabela zmiennych* jest obiektem umożliwiającym wyświetlanie w postaci tabelarycznej dowolnych zestawów zmiennych zdefiniowanych na etapie edycji lub pobieranych automatycznie w oparciu o wartości wybranych atrybutów zmiennych (również z użyciem wieloznaczników '*' i '?'). Każda komórka tabeli może zawierać dowolne wyrażenie zbudowane w oparciu o atrybuty i wartość zmiennej, stały napis, hiperłącze powiązane z akcją, wartości bitów zmiennych lub teksty i opisy stanów zmiennych. Każdy wiersz może posiadać zdefiniowaną nazwę zmiennej w kontekście której wyliczane są wszystkie wyrażenia określające zawartość i wygląd komórek. Wyrażenia mogą być zdefiniowane dla kolumn lub indywidualnie dla każdego wiersza. W każdej kolumnie można uzależnić kolor tekstu i tła komórek od dowolnego wyrażenia.

W trybie wykonywania aplikacji tabela umożliwia użytkownikowi wstawianie i usuwanie wierszy odpowiadających wybranym zmiennym, podmianę zmiennych w wierszach oraz zapamiętywanie tak zmodyfikowanego zestawu zmiennych. W trybie automatycznego wypełniania zmiennymi można dodatkowo wymusić sortowanie według nazw zmiennych.

Zawartość tabeli może być w dowolnym momencie zapisana w pliku tekstowym w formacie CSV. W tym celu należy wykorzystać akcję *ObjectAction*.

Nazwa	Wartość	Bit0	Akcja
?			Bity zmiennej ...
A100	12,8	True	Bity zmiennej A100 ...
A004	6	False	Bity zmiennej A004 ...
A084	21,0	True	Bity zmiennej A084 ...
?			Bity zmiennej ...
A032	70	False	Bity zmiennej A032 ...
A080	75,0	True	Bity zmiennej A080 ...
A052	254	False	Bity zmiennej A052 ...
A112	310	False	Bity zmiennej A112 ...
A046	302	False	Bity zmiennej A046 ...

Rys. *Przykłady obiektu Tabela Zmiennych.*

Właściwości

Zobacz: **Elementy wizualizacji** > **Standardowe właściwości obiektów**

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Rozmiar czcionki nagłówków	Liczba	Określa rozmiar czcionki używanej do wyświetlenia nagłówków kolumn.
Rozmiar czcionki komunikatów	Liczba	Określa rozmiar czcionki używanej do wyświetlania informacji w wierszach wartości.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania wszystkich tekstów w tabeli.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Liczba widocznych wierszy	Liczba	Jeżeli właściwość jest równa 0, to liczba wyświetlanych wierszy wynika z rozmiaru czcionki i wielkości obiektu. Zmienia się, jeżeli rozmiar obiektu zmienia się w wyniku przeskalowania diagramu. Jeżeli właściwość jest większa od 0, to zawsze wyświetlana jest ta sama podana liczba wierszy - rozmiar czcionki jest automatycznie modyfikowany przy zmianie rozmiaru obiektu.
Pokazuj nagłówki kolumn	Logiczna	Określa, czy należy pokazywać linię nagłówków kolumn.
Nazwa atrybutu selekcji	Nazwa atrybutu	Określa nazwę atrybutu zmiennych, na podstawie którego zostaną wybrane zmienne, których stan będzie monitorowany w tabeli. Właściwość działa w połączeniu z właściwością Wartość atrybutu selekcji.

		W przypadku dynamicznego określania zestawu wierszy wiersze zdefiniowane jawnie są ignorowane.
Wartość atrybutu selekcji	Tekst	Określa wzorzec służący do dynamicznego wyboru zmiennych monitorowanych w tabeli. Właściwość działa w połączeniu z właściwością *Nazwa atrybutu selekcji*. Przykład B11_* Zakładając, że nazwa atrybutu selekcji została określona jako *Name*, to w tabeli zostaną utworzone wiersze dla wszystkich zmiennych, których nazwa zaczyna się od *B11_*.
Nazwa atrybutu sortującego	Nazwa atrybutu	Jeżeli zestaw wierszy tabeli jest tworzony w sposób dynamiczny, to właściwość określa nazwę atrybutu służącego do posortowania wierszy wybranych zmiennych w tabeli.
Edytowalna	Logiczna	Określa, czy użytkownik może interakcyjnie zmieniać zawartość tabeli. Wartość *Tak* powoduje, że dostępne będzie menu kontekstowe z komendami służącymi do zarządzania informacjami pokazywanymi w tabeli, w szczególności wyboru monitorowanych zmiennych.
Skalowalna	Logiczna	Określa, czy użytkownik może interakcyjnie zmieniać kolejność wierszy wyświetlanych w tabeli. Zmiana kolejności odbywa się poprzez kliknięcie na nagłówku kolumny.
Dopasuj rozmiar okna	Logiczna	Określa, czy rozmiar okna ma być dopasowany do rozmiaru tabeli. Opcja przydatna dla tabel, dla których liczba wierszy wynika z zawartości bazy definicji zmiennych (liczby zmiennych zgodnych spełniających warunki selekcji). Tryb dopasowania działa tylko wtedy, gdy w oknie jest tylko jeden panel z diagramem, na którym jest tylko jeden obiekt w postaci tabeli zmiennych.

Właściwości kolumny

Nazwa	Typ wartości	Opis
Kolumna widoczna	Logiczna	Określa, czy kolumna jest widoczna. Właściwość pozwala na wybór wyświetlanych kolumn w trybie wykonania aplikacji.
Szerokość kolumny [%]	Procent	Określa procentową szerokość kolumny. Wartość 0 oznacza, że szerokość kolumny ma być dobrana automatycznie. Jeżeli tylko jedna kolumna będzie miała szerokość 0, to zajmie ona całe pozostałe miejsce po uwzględnieniu szerokości pozostałych kolumn.
Nagłówek kolumny	Tekst	Określa tekst nagłówka kolumny.
Wyrównywanie tekstu w poziomie	Typ wyliczeniowy: Do lewej, Wyśrodkowanie, Do prawej	Określa sposób wyrównywania tekstów w komórkach wierszy wartości.
Wartość wyświetlana	Tekst	Określa tekst, który zostanie wyświetlony w komórce wiersza. Właściwość stosowana wtedy, gdy sposób wyliczenia tekstu jest jednolity dla wszystkich wierszy tabeli. Wyrażenie użyte we właściwości jest wyliczane indywidualnie dla każdego wiersza - zmienną kontekstową jest zmienna zdefiniowana we właściwości *Nazwa zmiennej* wiersza. Przykład: # Wyświetlenie wartości zmiennej wiersza. =Attribute(Description) Wyświetlenie opisu zmiennej wiersza.
Źródło wartości	Typ wyliczeniowy: Wartość kolumny, Wartość wiersza, Opis wiersza	Określa metodę pozyskania tekstu wyświetlanego w komórce wiersza: • *Wartość kolumny* - tekst pobierany z właściwości *Wartość wyświetlania kolumny*. • *Wartość wiersza* - tekst pobierany z właściwości Wartość wyświetlana wiersza. Jeżeli nie została zdefiniowana, to z właściwości *Wartość wyświetlania kolumny*.

		• *Opis wiersza* - tekst pobierany z właściwości Opis wiersza. Jeżeli nie została zdefiniowana, to z właściwości *Wartość wyświetlania kolumny.*
Kolor czcionki	Kolor	Określa kolor tekstów wyświetlanych w komórkach wierszy wartości. Jeżeli właściwość nie jest ustawiona, to obowiązuje wartość właściwości stanowej *Kolor czcionki wierszy*. Przykład VarIsGood() ? Green : Red Kolor czcionki określany jest na podstawie statusu zmiennej wiersza (zdefiniowana we właściwości *Nazwa zmiennej* wiersza).
Kolor tła	Kolor	Określa kolor tła komórek wierszy wartości. Jeżeli właściwość nie jest ustawiona, to obowiązuje wartość właściwości stanowej *Kolor tła wierszy*.
Dymek	Tekst	Pozwala zdefiniować treść wyświetlaną w dymku dla komórki w kolumnie.
Podkreślenie hiperłącza	Logiczna	Określa, czy wyświetlać zawartość komórki w trybie hiperłącza (z podkreśleniem). Właściwość ma znaczenie tylko wtedy, gdy zdefiniowana została właściwość *Akcja* kolumny.
Akcja	Akcja operatorska	Treść akcji operatorskiej powiązanej z komórką wiersza wartości. Jeżeli wartość właściwości jest zdefiniowana, to kliknięcie na wyświetlanym tekście spowoduje wykonanie akcji. Akcja wykonywana jest w kontekście konkretnego wiersza, czyli w treści akcji można odwoływać się do zmiennej wiersza. Przykład SetVariable(””,1) Zmienna wiersza (zdefiniowana we właściwości *Nazwa zmiennej* wiersza) zostanie ustawiona na wartość 1.

Właściwości wiersza

Nazwa	Typ wartości	Opis
Nazwa zmiennej	Nazwa zmiennej	Nazwa zmiennej głównej wiersza. Wszystkie wyrażenia i akcje używane we właściwościach wiersza lub dziedziczone z ustawień kolumny wykonywane są w kontekście tej zmiennej.
Wartość wyświetlana	Tekst	Określa tekst, który zostanie wyświetlony w komórce wiersza, jeżeli właściwość *Źródło wartości* kolumny jest równe Wartość wiersza. Właściwość należy stosować wtedy, gdy sposób wyliczania tekstów kolumny nie jest jednolity dla wszystkich wierszy. Przykład =(Variable() &1)?Tak:Nie W definiowanym wierszu zostanie wyświetlony tekst zależny od stanu najmłodszego bitu zmiennej wiersza - w kolumnie, w której właściwość Źródło wartości jest równe *Wartość wiersza*. W innym wierszach można użyć wyrażeń, które np. testują stany innych bitów.
Opis	Tekst	Określa tekst, który zostanie wyświetlony w komórce wiersza, jeżeli właściwość *Źródło wartości* kolumny jest równe *Opis wiersza*. Właściwość należy stosować wtedy, gdy sposób wyliczania tekstów kolumny nie jest jednolity dla wszystkich wierszy. Właściwości wiersza *Opis* i *Wartość wyświetlania* pozwalają na zdefiniowanie dwóch indywidualnie wyliczanych kolumn. Przykład Bit statusu W definiowanym wierszu zostanie wyświetlony tekst *Bit statusu* - w kolumnie, w której właściwość *Źródło wartości* jest równe *Opis wiersza.*

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła tabeli - używany, gdy wiersze wartości nie zajmują całego obszaru tabeli.
Kolor siatki	Kolor	Określa kolor siatki rozdzielającej pola i wiersze tabeli.

Kolor czcionki nagłówków	Kolor	Określa kolor tekstów opisujących nagłówki.
Kolor tła nagłówków	Kolor	Określa kolor tła linii nagłówków kolumn.
Kolor czcionki wierszy	Kolor	Określa kolor tekstów wyświetlanych w komórkach wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor czcionki* nie została ustawiona.
Kolor tła wierszy	Kolor	Określa kolor tła komórek nieparzystych wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor tła* nie została ustawiona.
Naprzemienny kolor tła wierszy	Logiczna	Określa kolor tła komórek parzystych wierszy wartości. Ma znaczenie tylko dla tych kolumn, których właściwość *Kolor tła* nie została ustawiona.

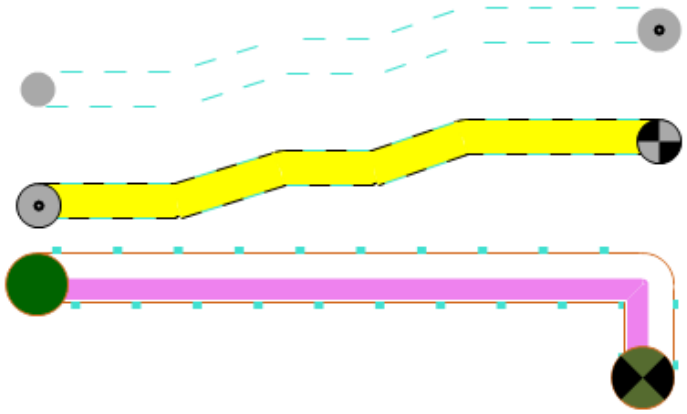
Obiekt Taśmociąg

Funkcje obiektu

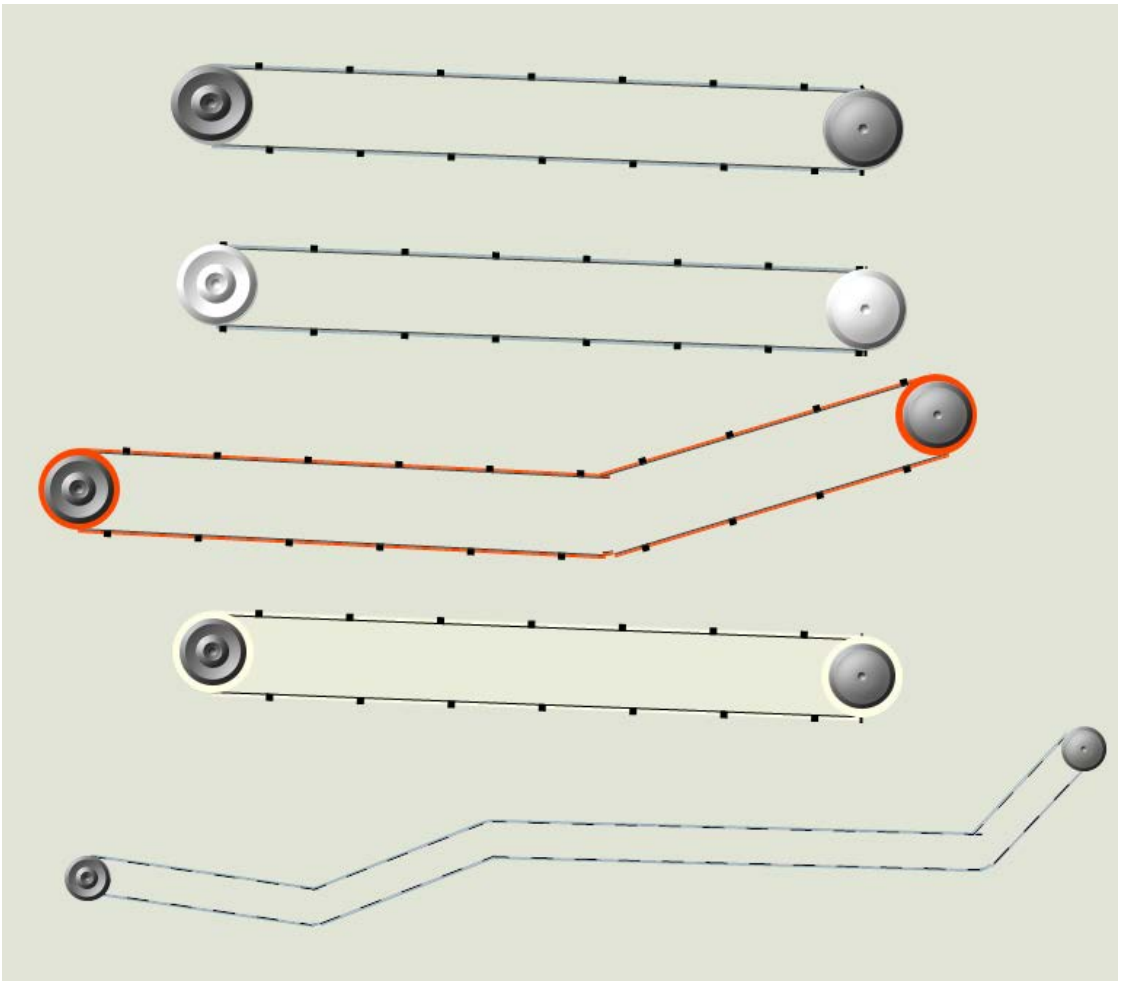
Obiekt *Taśmociąg* służy do prezentacji stanu rurociągów. Kształt obiektu jest definiowany serią koordynat poszczególnych przegubów taśmociągu. Definiując taśmociąg, można określić jego szerokość, typ (zgrzebłowy lub zwykły) oraz styl rolki początkowej i końcowej. Manipulując parametrami określającymi przesunięcie taśmy oraz obroty napędów, można uzyskać efekt animacji symulujący ruch taśmy w dowolną stronę i z dowolną prędkością.

Obiekt *Taśmociąg* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

Zobacz: ***Techniki budowy diagramów > Parametryzacja funkcji interaktywnych w obiektach pasywnych***



Rys. Przykłady obiektu Taśmociąg.

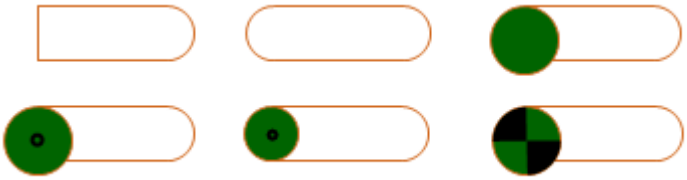


Rys. Przykłady zastosowania obiektu Taśmociąg (zgrzebłowy) w połączeniu z obiektem Obraz.

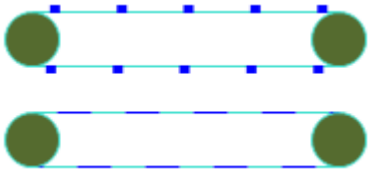
Właściwości

Zobacz: **Elementy wizualizacji** > **Standardowe właściwości obiektów**

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wierzchołki	Lista współrzędnych	Lista par X,Y współrzędnych kolejnych przegubów taśmociągu. Współrzędne X i Y pozycji przegubu wyrażone są pikselach i liczone względem lewego górnego narożnika diagramu. Kolejne pary wierzchołków są rozdzielone znakiem średnika.
Grubość	Liczba	Określa grubość taśmociągu wyrażoną w ilości pikseli.
Styl rolki początkowej	Typ wyliczeniowy: Płaski, Okrągły, Duża rolka, Mała rolka, Duża rolka z osią, Mała rolka z osią, Napęd	Określa styl rysowania rolki początkowej taśmociągu. 
Styl rolki końcowej	Typ wyliczeniowy: Płaski, Okrągły, Duża rolka, Mała rolka, Duża rolka z osią, Mała rolka z osią, Napęd	Określa styl rysowania rolki końcowej taśmociągu.
Taśmociąg zgrzebłowy	Logiczna	Określa, czy na taśmie należy rysować zgrzebła. 

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor	Kolor	Określa podstawowy kolor wypełniający wnętrze taśmociągu. 
Drugi kolor	Kolor	Określa alternatywny kolor wnętrza używany do wypełniania dolnej połowy taśmociągu. 
Kolor otoczki	Kolor	Określa kolor linii rysowanej ponad taśmą. 
Kolor brzegu	Kolor	Określa podstawowy kolor używany do rysowania taśmy. Służy także jako kolor zgrzebeł (jeżeli Kolor pasków brzegu nie został zdefiniowany). 
Kolor pasków brzegu	Kolor	W przypadku taśmociągu zgrzebłowego określa kolor zgrzebeł. Dla taśmociągów bez zgrzebeł określa alternatywny kolor taśmy. 
Kolor podkładu taśmy	Kolor	Określa kolor linii rysowanej pod taśmą. 
Kolor rolki początkowej	Kolor	Określa kolor rolki początkowej taśmociągu.

Kolor rolki końcowej	Kolor	Określa kolor rolki końcowej taśmociągu.
Przesunięcie taśmy	Liczba	W trybie wyświetlania taśmociągu zgrzebłowego lub paskowanego właściwość pozwala na przesunięcie wzorca rysowania, co powoduje efekt animacji ruchu. W przypadku taśmociągu zgrzebłowego jednostką ruchu jest szerokość zgrzebła, a w przypadku taśmociągu paskowanego jest to piksel. Przykład: =Variable(Counter) Wyrażenie animuje ruch rurociągu na bazie zmian zmiennej systemowej *Counter*. Kierunek ruchu można zmieniać poprzez zmianę znaku wyrażenia animującego ruch. Ewentualnie przemnożenie (podzielenie) wartości zmiennej pozwala na sterowanie szybkością i płynnością animacji.
Obrót napędu początkowego	Liczba	Jeżeli rolka początkowa jest typu *Napęd*, to właściwość pozwala na animację obrotu rolki. Jednostką obrotu jest stopień. Ogólne zasady animacji są identyczne jak w przypadku właściwości *Przesunięcie* taśmy.
Obrót napędu końcowego	Liczba	Jeżeli rolka końcowa jest typu *Napęd*, to właściwość pozwala na animację obrotu rolki. Jednostką obrotu jest stopień. Ogólne zasady animacji są identyczne jak w przypadku właściwości *Przesunięcie* taśmy.

Zdarzenia

Zobacz: ***Elementy wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Tekst

Funkcje obiektu

Obiekt *Tekst* służy do wyświetlania informacji w postaci tekstowej. Wyświetlane teksty mogą mieć charakter statyczny lub być zależne od wartości zmiennych procesowych. Obiekt pozwala na wykonywanie operacji sterujących. Dostępne są dwa tryby wprowadzania nowej wartości: bezpośrednia edycja tekstu lub selekcja z zestawu predefiniowanych stanów.



Rys. Przykłady obiektu Tekst.

Obiekt pozwala wyświetlać teksty jedno- i wieloliniowe. Dla każdego tekstu można określić kolor, grubość i kolor obwódki, głębokość i kolor cienia, głębokość i kolor wypukłości (rozmycia), kierunek i kolor gradientu, krój i styl czcionki oraz kąt obrotu.

Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wartość minimalna sterowania	Liczba	Ograniczenie dolne dopuszczalnych wartości sterujących.
Wartość maksymalna sterowania	Liczba	Ograniczenie górne dopuszczalnych wartości sterujących.
Limit czasu sterowania [ms]	Liczba	Ustawienie limitu pozwala na zamaskowanie opóźnienia pomiędzy wysłaniem sterowania, a faktycznym wykonaniem tego sterowania. W praktyce oznacza to, że po wykonaniu sterowania obiekt pokazuje wartość zadaną niezależnie od stanu faktycznego, a dopiero po upływie limitu czasu odpowiedzi zaczyna pokazywać wartość rzeczywistą. Limit powinien być na tyle długi, żeby w jego trakcie zdążyło się wykonać wysłanie wartości sterującej oraz odświeżenie wartości zmiennej głównej obiektu. Takie ustawienie pozwala ukryć efekt migotania stanu obiektu w trakcie wykonywania operacji sterowania.
Kąt obrotu	Liczba	Kąt w stopniach liczony względem kierunku poziomego, pod którym wyświetlany jest tekst. Kąt 90 oznacza kierunek pionowo w dół.
Kierunek gradientu	Typ wyliczeniowy: Poziomo, Pionowo, Ukośnie w górę, Ukośnie w dół	Określa sposób zastosowania gradientu do wyświetlenia napisu. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor gradientu we właściwości *Kolor gradientu*. 
Wysokość podczas obrotu	Liczba	Określa wysokość napisu w pikselach w przypadku stosowania kąta obrotu. 

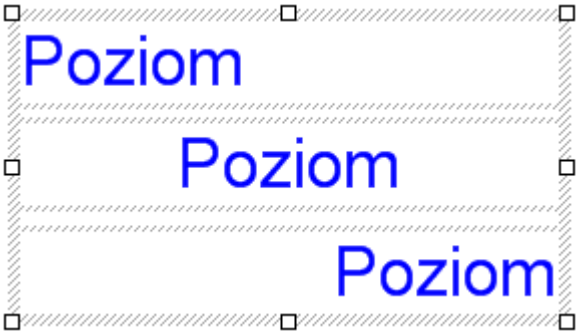
		W przypadku napisów poziomych wysokość czcionki jest określana automatycznie na podstawie wielkości obiektu i długości napisu.
Słownik wartości	Tekst	Właściwość pozwala na zdefiniowanie przyporządkowania wartości liczbowych zmiennej procesowej do ich odpowiedników tekstowych. W celu wyświetlenia właściwego tekstu, we właściwości stanowej Tekst należy wprowadzić wartość #. Jeżeli obiekt służy także do sterowania, to zdefiniowanie właściwości powoduje automatyczne przejście do trybu selekcji stanów. Jest to uproszczona wersja tego trybu (zobacz właściwość *Tryb selekcji stanów*), w której nie ma możliwości indywidualnego ustawiania atrybutów tekstów, ani wykonywania sterowań bitowych. Zaletą jest definiowanie całego przekodowania w pojedynczej właściwości (zamiast w osobnych stanach) oraz możliwość dynamicznej zmiany sposobu kodowania w trakcie pracy aplikacji (bez zmiany definicji diagramu). Wartość właściwości musi być podana w sposób zgodny ze wzorcem: *wartość1=tekst1;wartość2=tekst2;wartość=tekstN*. Możliwe jest także użycie funkcji *GetDictionary*. Funkcja ta pozwala zintegrować obiekt Tekst z kolumną słownikową obiektu *Tabela Asbase*. W ten sposób definicja kodowania jest pobierana z bazy danych.
Tryb selekcji stanów	Logiczna	Określa tryb wykonywania operacji sterujących. Wartość Tak oznacza, że wartość sterująca będzie określana poprzez wybór z predefiniowanego zestawu stanów (opisy i wartości stanów definiowane są we właściwościach stanowych). Wartość Nie oznacza, że użytkownik będzie bezpośrednio wprowadzał wartość sterującą. W każdym przypadku wymagane jest dodatkowo ustawienie właściwości *Aktywny* na wartość Tak oraz określenie nazwy zmiennej sterowanej.
Selekcja stanów przez menu	Logiczna	Właściwość ma znaczenie tylko wtedy, gdy właściwość *Tryb selekcji stanów* jest równa *Tak*. Określa sposób wyboru nowego stanu sterującego. Wartość Tak oznacza wybór z menu kontekstowego tworzonego automatycznie na podstawie zdefiniowanych stanów obiektu. Wartość *Nie* oznacza, że przy każdym kliknięciu na obiekcie, będzie następowało cykliczne przełączanie kolejnych stanów.
Edycja przy selekcji	Logiczna	Określa, czy obiekt wejdzie w tryb edycji natychmiast po jego wybraniu (selekcji), czy też wymagane będzie podwójne kliknięcie klawiszem myszki.

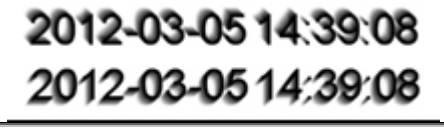
Początkowa wartość edycji	Typ wyliczeniowy: Pusta, Ostatnia, Bieżąca	Określa jaka jest początkowa wartość sterująca po wejściu w tryb edycji napisu. Wariant *Ostatnia* oznacza wartość wprowadzoną przy poprzedniej edycji, a wariant *Bieżąca* oznacza aktualną treść wyświetlanego napisu.
Rodzaj edytora	Typ wyliczeniowy: Tekst, Data i czas, Zakres czasu	Określa sposób wprowadzania nowej wartość sterującej. Wartość 'Tekst' oznacza swobodną edycję tekstu przez użytkownika. W pozostałych przypadkach wprowadzenie wartości odbywa się poprzez specjalizowane okienka dialogowe.
Zatwierdź po utracie selekcji	Logiczna	Ustawienie opcji powoduje, że obiekt automatycznie wysyła sterowanie (lub ustawia oczekujące) po utracie selekcji (np. po wyborze innego obiektu).
Zatwierdź po utracie selekcji	Logiczna	Ustawienie opcji powoduje, że obiekt automatycznie wysyła sterowanie (lub ustawia oczekujące) po utracie selekcji (np. po wyborze innego obiektu).
Liczba linii	Liczba	Steruje sposobem dopasowania wysokości czcionki do długości tekstu. Wartość 0 oznacza, że należy tak zmniejszyć czcionkę, aby cały tekst się zmieścił w obszarze obiektu. Skutkiem tego w zależności od długości tekstu może zostać użyta inna czcionka. Wartość większa od 0 oznacza, że o wysokości czcionki decyduje wyłącznie rozmiar obiektu. W razie konieczności tekst może zostać obcięty lub podzielony na kilka linii. 
Margines tekstu [%]	Logiczna	Podawana w procentach wielkość marginesu pozostawionego wokół tekstu. Właściwość przydatna w przypadku stosowania nieprzeźroczystego koloru tła lub wyświetlania tekstu w ramce.

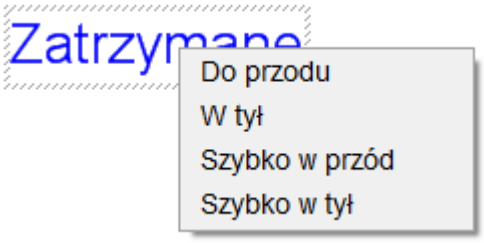
Sterowanie natychmiastowe	Logiczna	Określa, czy w przypadku wprowadzenia nowej wartości sterującej jest ona automatycznie wysyłana do zmiennej sterującej. Jeżeli nie, to do realizacji sterowania wymagane jest użycie akcji operatorskiej *SendControls*.
Grupa sterowania	Tekst	Określa nazwę grupy obiektów sterujących. Jeżeli obiekt wchodzi w stan oczekiwania na wysłanie sterowania (oczekiwanie na akcję *SendControls*), to wszystkie pozostałe obiekty grupy rezygnują z oczekiwania. Mechanizm pozwala zagwarantować, że jeżeli operator dysponował alternatywnymi metodami ustawienia tej samej wartości (np. obiekty *Słupek* i *Tekst*), to wykonane zostanie tylko ostatnie sterowanie.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Tekst	Tekst	Treść wyświetlanego napisu. Może to być statyczny tekst lub dowolne wyrażenie, którego wynikiem jest wartość tekstowa. Najczęściej wyświetlane napisy są tworzone na podstawie wartości zmiennych procesowych. Przykład: # Wyświetla wartość głównej zmiennej obiektu, stosując domyślną konwersję na tekst. =Attribute(Description) + ” ” + Variable() Wyświetla opis zmiennej głównej połączony z jej wartością.
Kolor	Kolor	Określa kolor napisu.
Kolor gradientu	Kolor	Określa kolor gradientu. Kolor napisu będzie płynnie zmieniany pomiędzy kolorem podstawowym a kolorem gradientu. Sposób użycia gradientu określa właściwość *Kierunek gradientu*.
Kolor edycji	Kolor	Określa kolor tekstu w trakcie wprowadzania nowej wartości.
Kolor przekreślenia	Kolor	Zdefiniowanie koloru innego niż *Transparent* powoduje narysowanie kreski przekreślającej napis. Właściwość stosowana do sygnalizacji stanu błędnego odczytu zmiennej. 

		Przykład: =VarIsNotGood()?Red:Transparent
Kolor otoczki	Kolor	Określa kolor konturu liter. 
Kolor cienia	Kolor	Określa kolor cienia liter.
Kolor tła	Kolor	Określa kolor tła napisu.
Kolor tła edycji	Kolor	Określa kolor tła napisu w trakcie edycji nowej wartości sterującej.
Wyrównanie w poziomie	Typ wyliczeniowy: Do lewej, Wyśrodkowanie, Do prawej	Określa sposób poziomego pozycjonowania napisów wewnątrz obszaru obiektu. 
Wyrównanie w pionie	Typ wyliczeniowy: Do góry, Wyśrodkowanie, Do dołu	Określa sposób pionowego pozycjonowania napisów wewnątrz obszaru obiektu. 
Szerokość otoczki	Liczba	Określa grubość konturu liter. 
Głębokość cienia	Liczba	Wartość większa od 0 powoduje wyświetlenie cienia wszystkich liter napisu. Cień rysowany jest z przesunięciem w dół i w lewo o podaną ilość pikseli. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor cienia we właściwości *Kolor cienia*.  Cień nie jest rysowany, jeżeli została ustawiona właściwość *Głębokość wypukłości*.
Głębokość wypukłości	Liczba	Wartość większa od 0 powoduje wyświetlenie napisu z efektem wypukłości (rozmycia). Kolor wypukłości jest zmodyfikowanym kolorem podstawowym.

		
Kierunek wytłoczenia	Typ wyliczeniowy: Lewa góra, Prawa góra, Lewy dół, Prawy dół	Określa kierunek rysowania efektu wypukłości. 
Styl ramki	Typ wyliczeniowy: Brak, Pojedyncza, Wklęsła, Wypukła	Określa sposób rysowania ramki wokół tekstu.
Kolor ramki	Kolor	Kolor ramki. Ma znaczenie tylko dla ramki w trybie *Pojedyncza*. W przypadku ramek trójwymiarowych kolory są dobierane automatycznie.
Czcionka	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania napisu.
Styl czcionki	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl używanej czcionki. Można stosować wiele styli jednocześnie.
Wartość stanu	Tekst	Wartość wysyłana do zmiennej sterowanej, gdy obiekt pracuje w trybie selekcji stanów, a użytkownik wybrał z memu kontekstowego pozycję odpowiadającą temu stanowi.
Maska bitowa stanu	Liczba	Modyfikator wartości wysyłanej do zmiennej sterowanej w trybie selekcji stanów. Nowa wartość wyliczana jest na podstawie aktualnej wartości zmiennej procesowej, właściwości Wartość stanu i podanej maski bitowej. Efekt działania algorytmu jest następujący: • zmieniane są tylko te bity wartości zmiennej sterowanej, które w masce mają wartość 1; • wartość zmienionych bitów jest zgodna z wartościami bitów w wartości stanu. Maskowanie bitów nie jest stosowane, gdy właściwość nie jest zdefiniowana (jest pusta). Przykład: Wartość stanu = 1 Maska stanu = 1 Najmłodszy bit zmiennej sterowanej zostanie ustawiony na 1, pozostałe bity nie zostaną zmienione. Wartość zmiennej = 1 Maska stanu = 3

		Dwa najmłodszy bit zmiennej sterowanej zostaną ustawione na wartości 01, pozostałe bity nie zostaną zmienione.
Stan wybieralny	Logiczna	Określa, czy stan pojawia się w menu selekcji stanów. Właściwość ma znaczenie tylko wtedy, gdy obiekt pracuje w trybie selekcji stanów. Dla wszystkich stanów, których właściwość *Stan wybieralny* jest równa *Tak*, tworzona jest pozycja w menu kontekstowym otwieranym, gdy użytkownik kliknie na obiekcie. Tekst w pozycji menu wynika z aktualnej wartości właściwości *Tekst* w definicji stanu odpowiadającej pozycji. 

Zdarzenia

Zobacz: ***Elementy_wizualizacji*** > ***Standardowe zdarzenia obiektów***

Obiekt Wielokąt

Funkcje obiektu

Obiekt*Wielokąt* pozwala na tworzenie dowolnych płaskich zamkniętych figur geometrycznych. Kształt obiektu jest definiowany serią koordynat poszczególnych wierzchołków figury. Dla każdej figury można określić kolor wnętrza, grubość i kolor obwódki. Możliwe jest też tworzenie figur o płynnie wygiętych krawędziach w trybie krzywych Beziera.



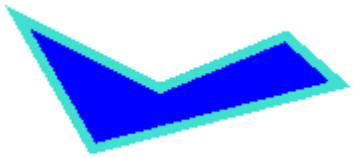
Obiekt *Wielokąt* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

Zobacz: ***Techniki budowy diagramów > Parametryzacja funkcji interaktywnych w obiektach pasywnych***

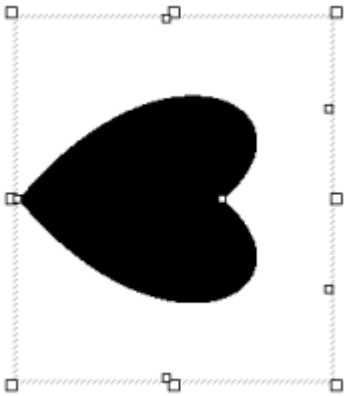
Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Wierzchołki	Lista współrzędnych	Lista par X,Y współrzędnych kolejnych wierzchołków figury. Współrzędne X i Y pozycji wierzchołka wyrażone są pikselach i liczone względem lewego górnego narożnika diagramu. Kolejne pary wierzchołków są rozdzielone znakiem średnika.
Szerokość otoczki	Liczba	Określa szerokość (w pikselach) otoczki rysowanej wzdłuż krawędzi figury. Właściwość ma znaczenie tylko wtedy, gdy ustawiony jest kolor otoczki we właściwości *Kolor otoczki*. 

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor	Kolor	Określa kolor wypełnienia kształtu. Wybranie koloru *Transparent* i ustawienie niezerowej szerokości otoczki pozwala na rysowanie jedynie konturu figury.
Kolor otoczki	Kolor	Określa kolor otoczki.
Styl otoczki	Typ wyliczeniowy: Brak, Kreska, Kropka, Kreska-Kropka, Kreska-Kropka-Kropka	Określa sposób rysowania otoczki. Wartość *Brak* oznacza rysowanie otoczki pełną linią.
Typ linii	Typ wyliczeniowy: Łamana, Krzywa Beziera	Określa czy boki figury są prostymi odcinkami czy krzywymi Beziera. W trybie Beziera każdy bok jest definiowany przez cztery wierzchołki. Pierwszy i czwarty określają punkty końcowe. Drugi i trzeci służą do kontroli krzywizny. 

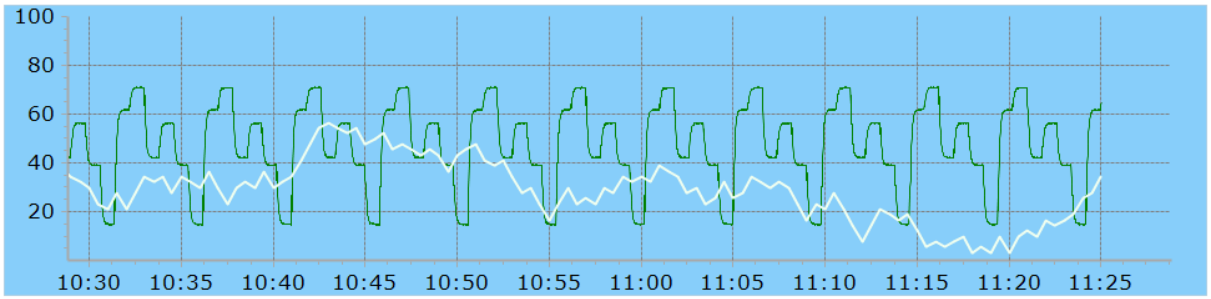
Zdarzenia

Zobacz: ***Elementy wizualizacji > Standardowe zdarzenia obiektów***

Obiekt Wykres

Funkcje obiektu

Obiekt *Wykres* jest obiektem umożliwiającym rysowanie przebiegów wartości zmiennych.



Rys. Przykłady obiektu Wykres.

W typowym przypadku wykres wyświetla wartości pobierane z archiwum wartości historycznych zmiennych procesowych. Może jednak pracować także w trybie wartości bieżących. Wtedy nie jest potrzebna archiwizacja wartości zmiennej, ponieważ wykres jest budowany na podstawie wartości bieżących dostępnych od chwili uruchomienia wykresu. Dla danych historycznych można wyświetlać również wartości zagregowane.

Na jednym wykresie możliwe jest rysowanie wielu przebiegów (serii). Każdej z serii można osobno określić zmienną, styl wykresu (linia lub słupek), kolor wartości poprawnych, kolor wartości niepewnych, grubość linii, styl znaczników oraz czy pracuje w tzw. trybie procentowym (w którym wartości są skalowane do zakresu 0 -100 w oparciu o atrybuty zmiennej określające zakres).

Oprócz serii zmiennych wykres umożliwia również wyświetlanie zdefiniowanych wcześniej przebiegów wzorcowych. Każdemu przebiegowi wzorcowemu możemy zdefiniować jego krzywą, punkt czasowy zaczepu oraz - podobnie jak seriom - kolor, styl, szerokość linii i styl znaczników. Przebiegi wzorcowe mogą być dostosowywane w trybie działania. Każdemu zdefiniowanemu przebiegowi można podmienić krzywą i modyfikować punkt zaczepu. Opcje te są dostępne w menu kontekstowym wykresów.

Kolejnym elementem wyświetlanym na wykresie są tzw. limity. Mogą być pokazywane w postaci linii lub pasma. Limity pokazywane są w postaci linii (pasm) poziomych i nie zależą od historii zmian zmiennych. Jeżeli wartości limitów byłyby zmienne w czasie, to należałoby je niezależnie archiwizować i pokazywać na wykresie w postaci zwykłych serii.

Obiekt *Wykres* posiada własne mechanizmy obsługi wykresu, typu zmiana skali, czy przesuwanie się w osi czasu. Można także sterować pracą wykresu przy pomocy obiektu *Kontroler Wykresu*. Możliwe jest synchroniczne sterowanie grupami obiektów *Wykres*.

Poniższa tabela opisuje skróty klawiszowe obsługiwane przy obiekt *Wykres*.

Skrót	***Znaczenie***
Klawisz strzałki	Przesunięcie kursora we wskazanym kierunku
Shift + klawisz strzałki	Przesunięcie wykresu w osi X lub Y we wskazanym kierunku
Ctrl + klawisz strzałki	Zwiększenie lub zmniejszenie zakresu osi X lub Y w zależności od użytego klawisza strzałki
PgUp, PgDn	Przesunięcie wykresu w osi X o cały zakres wykresu
Shift + PgUp, PgDn	Przesunięcie wykresu w osi Y o cały zakres wykresu
Ctrl-Home	Reset osi czasu do ustawień początkowych
Ctrl-End	Reset osi wartości do ustawień początkowych
Numeryczne 5	Wyśrodkowanie pozycji kursora
Ctrl + numeryczne 5	Reset wykresu do ustawień początkowych
Ctrl-h	Wyświetlenie okienka zmiany zakresu czasu
Ctrl-l	Wyświetlenie okienka legendy
Ctrl-F1	Wyświetlenie okienka pomocy pokazującego wszystkie skróty klawiszowe
Ins	Włączenie lub wyłączenie kursora

Obiekt może być też obsługiwany przy pomocy myszki.

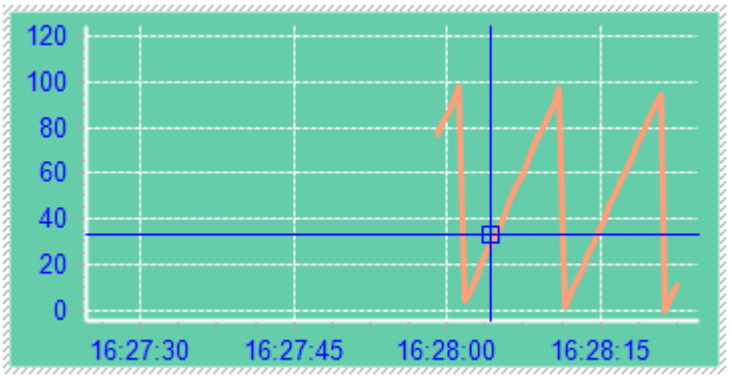
Operacja	***Znaczenie***
Zaznaczenie obszaru wykresu ruchem od lewej do prawej strony	Powiększenie wskazanego obszaru wykresu
Zaznaczenie obszaru wykresu ruchem od prawej do lewej strony	Reset wykresu do ustawień początkowych
Ctrl + obrót rolki	Powiększenie lub zmniejszenie horyzontu czasu wykresu
Przeciągnięcie kursora wykresu lub dowolnej z linii kursora wykresu	Zmiana pozycji kursora
Przeciągnięcie osi wartości	Przesuniecie zakresu wartości (pod warunkiem ustawienia właściwości *Przesuwne osie*)
Przeciągnięcie osi czasy	Przesuniecie horyzontu czasu wykresu (pod warunkiem ustawienia właściwości *Przesuwne osie*)
Podwójne kliknięcie	Reset wykresu do ustawień początkowych

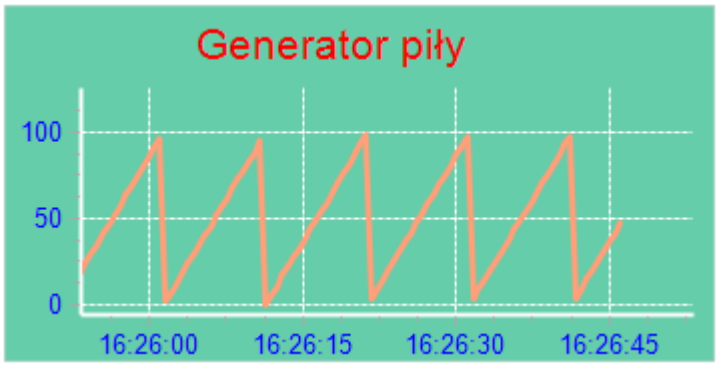
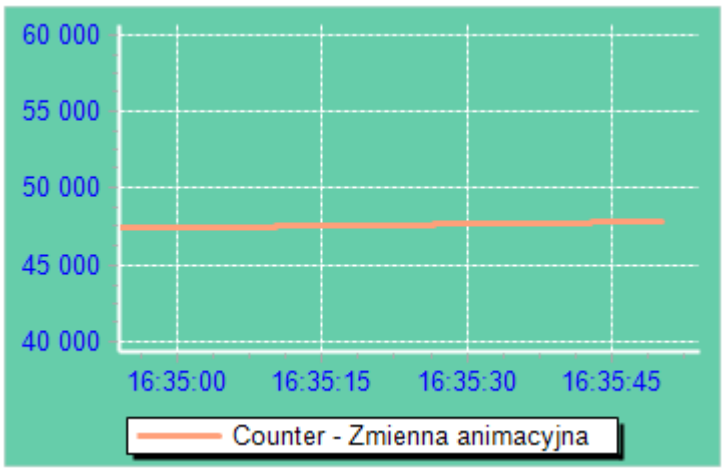
Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Dynamiczny zakres wartości	Logiczna	Określa, czy zakres osi Y jest dynamicznie wyliczany na podstawie odczytanych wartości punktów krzywych. Zakres osi może się zmieniać w trakcie wyświetlania wykresu w zależności od wartości przychodzących punktów. Jeżeli właściwość jest ustawiona na *true*, to poniższe właściwości związane z określeniem skali osi Y nie mają znaczenia.
Wartość minimalna automatyczna	Logiczna	Określa, czy początkowa wartość minimalna na osi pionowej jest wyliczana automatycznie. Algorytm wyliczania bazuje na wyszukiwaniu minimalnej wartości atrybutu *DisplayRangeFrom* dla wszystkich wyświetlanych zmiennych.
Wartość minimalna	Liczba	Określa początkową wartość minimalną na osi pionowej wykresu. Właściwość stosowana, gdy nie jest stosowany algorytm automatycznego wyliczania wartości minimalnej ani dynamiczny zakres wartości.
Wartość maksymalna automatyczna	Logiczna	Określa, czy początkowa wartość maksymalna na osi pionowej jest wyliczana automatycznie. Algorytm wyliczania bazuje na wyszukiwaniu maksymalnej wartości atrybutu *DisplayRangeTo* dla wszystkich wyświetlanych zmiennych.
Wartość maksymalna	Liczba	Określa początkową wartość maksymalną na osi pionowej wykresu. Właściwość stosowana, gdy nie jest stosowany algorytm automatycznego wyliczania wartości maksymalnej ani dynamiczny zakres wartości..
Ręczny czas bazowy	Tekst	Właściwość pozwala na forsowanie wyboru pokazywanego na wykresie zakresu czasu. Jeżeli właściwość nie jest ustawiona, to początkowy zakres określany jest na podstawie chwili bieżącej. Właściwość jest typu tekstowego. Dopuszczalny jest dowolny format rozpoznawalny przez system jako data i czas. Jawnie

		definiowany czas bazowy jest stosowany tylko wtedy, gdy właściwość *Tryb odświeżania* jest ustawiona na wartość *Ręcznie*.
Zakres czasu [min]	Liczba	Określa początkowy zakres czasu pokazywanego na wykresie.
Tylko dane bieżące	Logiczna	Jeżeli wartość właściwości jest równa Tak, to przebiegi zmiennych rysowane są na podstawie wartości bieżących tych zmiennych. Obiekt zaczyna samodzielnie gromadzić historię zmian od momentu otwarcia diagramu. Jeżeli wartość właściwości jest równa Nie, to przebiegi zmiennych rysowane są na podstawie danych historycznych przechowywanych w archiwach.
Tryb odświeżania	Typ wyliczeniowy: Skokowo, Płynnie, Ręcznie	Określa sposób przesuwania wykresu w momencie, gdy wykres dochodzi do prawej krawędzi obiektu. W trybie skokowym wykres przesuwany jest jednorazowo o około 20% rozmiaru obiektu, po czym przez pewien okres czasu pozostaje nieruchomy. W trybie płynnym, wykres jest cały czas przesuwany o minimalną odległość wynikającą z przyjętej skali czasu. W trybie ręcznym przesunąć wykres musi zawsze użytkownik.
Częstotliwość odświeżania zależna od zakresu	Logiczna	Określa, w jaki sposób odbywa się odświeżanie wykresu. Dla wartości *Tak*, częstotliwość odpytania o nowe dane jest wyliczana na podstawie zakresu czasu. Dla wykresów o dużym horyzoncie czasowym liczba zapytań jest ograniczana. Dla wartości Nie odpytanie o dane jest wykonywane zawsze, co jedną sekundę. Poprawia to czas reakcji na nowe dane, ale może skutkować pogorszeniem wydajności całej aplikacji.
Wyświetlaj kursor	Logiczna	Określa, czy należy na wykresie wyświetlać kursor ułatwiający określenie wartości zmiennych dla wskazanej chwili czasu. 
Przyciągaj kursor	Logiczna	Określa, czy kursor jest automatycznie pozycjonowany na wartości krzywej. Praca w trybie przyciągania zmienia także sposób obsługi klawiszy strzałek i przycisków Kursor w ... obiektu

		Kontroler wykresu. W trybie przyciągania służą one do poruszania się po punktach serii lub wyboru serii.
Format czasu kursora	Tekst	Określa sposób formatowanie czasu w polu wartości kursora.
Podgląd wartości kursora	Typ wyliczeniowy: Brak, W legendzie	Określa sposób podglądu wartości serii dla aktualnego położenia kursora. Dla wartości *W legendzie* wartość serii jest wyświetlana w legendzie obok opisu serii.
Tytuł wykresu	Tekst	Określa treść tytułu wykresu. 
Wyświetlaj legendę	Logiczna	Określa, czy należy wyświetlać legendę wykresu. W treści legendy domyślnie wyświetlana jest nazwa zmiennej i jej opis pobrany z bazy definicji zmiennych. Opis krzywej można zmienić we właściwościach serii. W legendzie może być też wyświetlana wartość serii dla pozycji kursora, jeżeli zostanie to włączone we właściwości *Podgląd wartości kursora*. 
Pokazuj podziałkę czasu	Logiczna	Określa, czy wyświetlać podziałkę osi czasu.

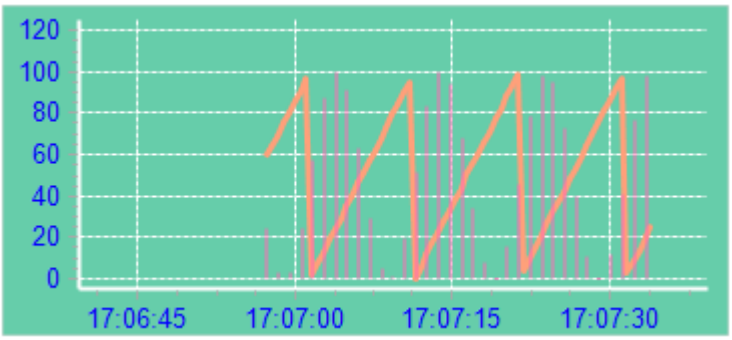
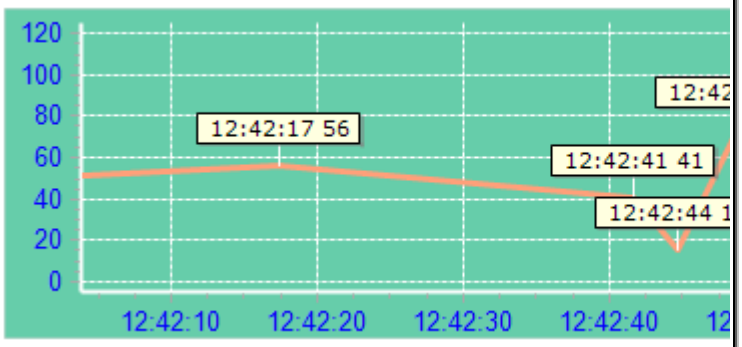
Pokazuj podziałkę wartości	Logiczna	Określa, czy wyświetlać podziałkę osi wartości.
Skok wartości	Liczba	Określa odstęp pomiędzy kolejnymi liniami podziałki wartości. Wartość 0 oznacza wybór automatyczny.
Liczba podpodziałek wartości	Liczba	Określa liczbę podpodziałek pomiędzy kolejnymi podziałkami głównymi osi wartości.
Liczba podpodziałek czasu	Liczba	Określa liczbę podpodziałek pomiędzy kolejnymi podziałkami głównymi osi czasu.
Format osi czasu	Tekst	Określa sugerowany sposób formatowania opisów podziałek na osi czasu. Na przykład, format HH:mm:ss tworzy znaczniki w postaci 13:30:45. Program może samodzielnie zmienić formatowanie, jeżeli w efekcie operacji zmiany zakresu czasu, formatowanie sugerowane będzie nieodpowiednie.
Format osi wartości	Tekst	Określa sposób formatowania opisów podziałek na osi wartości. Na przykład, format *f2* tworzy znaczniki w postaci liczb z dwoma miejscami po przecinku.
Siatka na wierzchu	Logiczna	Określa, czy linie siatki wykresu mają przesłaniać linie przebiegów i limitów.
Zezwalaj na zmianę zakresu wartości	Logiczna	Określa, czy użytkownik może zmieniać skalę wartości. Dotyczy operacji przesuwania skali i zmieniania jej zakresu.
Przesuwne osie	Logiczna	Określa, czy użytkownik może zmieniać skale czasu i wartości poprzez przeciągnie ich przy pomocy myszki.
Grupa wykresów	Tekst	Określa nazwę grupy wykresów. Wykresy o tej samej nazwie są obsługiwane w sposób synchroniczny - wszystkie operacje nawigacyjne na jednym z wykresów grupy są odtwarzane na pozostałych wykresach. Nazwa grupy służy też do połączenia obiektu *Wykres* z obiektem *Kontroler Wykresu*.
Obsługuj skróty klawiszowe	Logiczna	Określa, czy w trybie wykonania aplikacji użytkownik może używać klawiszy nawigacyjnych, między innymi pozwalających na zmianę zakresu wartości lub przemieszczanie się w osi czasu.

Modyfikowalne przebiegi wzorcowe	Logiczna	Określa, czy użytkownik może zmieniać wyświetlane krzywe wzorcowe oraz ich punkty zaczepienia. Wartość Tak powoduje, że dostępne będzie menu kontekstowe z komendami służącymi do zarządzania krzywymi wzorcowymi na wykresie.

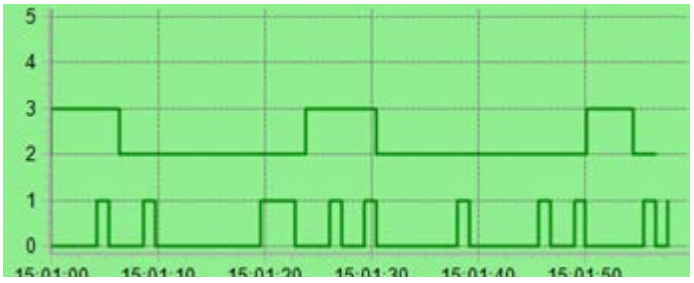
Właściwości przebiegów wzorcowych

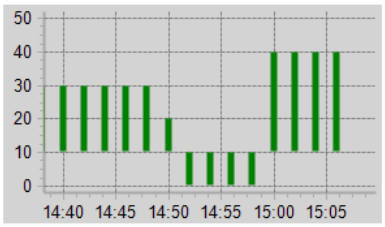
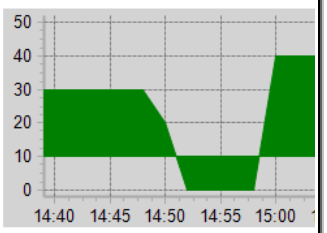
Nazwa	**Typ wartości**	**Opis**
Nazwa przebiegu	Nazwa przebiegu	Określa nazwę krzywej wzorcowej, którą należy wyświetlić.
Zaczep przebiegu	Tekst	Określa punkt zaczepienia(początek) krzywej wzorcowej. Właściwość jest typu tekstowego. Dopuszczalny jest dowolny format rozpoznawalny przez system jako data i czas. Jeżeli zaczep nie jest podany, to krzywa wzorcowa zaczyna się od prawej krawędzi wykresu. Użytkownik może ręcznie zmienić punkt zaczepienia, jeżeli właściwość *Modyfikowalne* przebiegi wzorcowe jest ustawiona na Tak.
Styl serii	Typ wyliczeniowy: Linia, Słupek, Obszarowy, Schodkowy	Określa, czy krzywa jest typu liniowego, słupkowego, obszarowego czy schodkowego.
Styl znaczników	Typ wyliczeniowy: Brak, Czas, Wartość, Czas i Wartość	Określa, czy przy każdym zarejestrowanym punkcie przebiegu należy wyświetlać znaczniki z informacją o czasie rejestracji i/lub zarejestrowanej wartości.
Kolor	Kolor	Określa kolor krzywej (lub słupków).
Szerokość	Liczba	Określa szerokość rysowanej krzywej (lub słupka).
Baza obszaru	Liczba	Właściwość stosowana tylko dla wykresów w stylu obszarowym. Określa poziom, względem którego zakreślane są obszary wartości.

Właściwości serii

Nazwa	Typ wartości	Opis
Nazwa zmiennej	Nazwa zmiennej	Określa nazwę zmiennej, dla której należy narysować wykres przebiegu.
Styl serii	Typ wyliczeniowy: Linia, Słupek, Linia schodkowa, Obszarowy, Obszarowy schodkowy	Określa, czy krzywa jest typu liniowego, słupkowego, obszarowego, liniowego schodkowego lub obszarowego schodkowego (dedykowany dla wykresów wartości binarnych). 
Styl znaczników	Typ wyliczeniowy: Brak, Czas, Wartość, Czas i Wartość	Określa, czy przy każdym zarejestrowanym punkcie przebiegu należy wyświetlać znaczniki z informacją o czasie rejestracji i/lub zarejestrowanej wartości. 
Punkty linii	Typ wyliczeniowy: Brak, Mały punkt, Punkt, Mały prostokąt, Prostokąt, Małe koło, Koło, Mały diament, Diament, Mała kula, Kula	Określa, czy należy (i w jakiej formie) wyświetlać, przy każdym zarejestrowanym punkcie przebiegu, symbol punktu. Szczególne znaczenie ma styl mały punkt. Dla przebiegu ciągłego jest on niewidoczny. Pozwala jednak pokazać wykres ”dziurawy”, gdy pojedyncze zarejestrowane punkty są otoczone obszarami braku danych.

		
Kolor	Kolor	Określa kolor krzywej (lub słupków).
Kolor statusu niepewnego	Kolor	Określa kolor krzywej (lub słupków) używany dla oznaczenia wartości niepewnych.
Szerokość	Liczba	Określa szerokość rysowanej krzywej (lub słupka).
Procentowy	Logiczna	Określa, czy dane archiwalne mają być przeliczane na wartości procentowe. Przeliczenie bazuje na zakresie zmiennej pobranym z atrybutów *DisplayRangeFrom* i *DisplayRangeTo*. Procentowe przeliczenie pozwala na umieszczenie na jednym wykresie krzywych, których zakresy zmienności są zupełnie inne (nie da się uzgodnić zakresu skali pionowej).
Agregat	Tekst	Określa nazwę agregatu, który należy zastosować do przeliczenia wartości historycznych zmiennej pokazywanej na wykresie. Dostępne nazwy agregatów to: *none, start, end, delta, min, max, range, total, average, average0, sumup, sumdown, prevknown, last, stdev, rms, avglk, totallk*. Użycie nazwy none oznacza, że należy stosować dane surowe (nieprzeliczane). Zobacz Rodzaje agregatów danych archiwalnych.
Interwał agregacji	Tekst	Określa interwał agregacji stosowany w trybie przeliczania surowych wartości historycznych na wartości zagregowane. Interwał przekazywany jest w postaci tekstowej, w formacie zgodnym ze stosowanym w funkcji *ToTimeSpan*. Przykład:

		00:01:00 Wyliczane są agregaty 1-minutowe 15M Wyliczane są agregaty 15-minutowe.
Maska bitowa	Tekst	Właściwość pozwalająca na utworzenie krzywej o charakterze binarnym. Maska ma postać tekstową, złożona jest ze znaków -, 0 i 1, oznaczających odpowiednio wartość nieistotną, wartość 0 i wartość 1. Na podstawie porównania wartości zmiennej i maski, wyliczana jest wartość punktu: *True* lub *False*. Przykład: Tekst "10-” oznacza, że jeżeli dla wartości zmiennej w punkcie bit numer 3 jest równy 1 i bit numer 2 jest równy 0, to wartość punktu jest *True*. W każdym innym przypadku jest *False*.
Wartość dla False	Liczba	Właściwość stosowana dla krzywych binarnych (wyliczanych na podstawie właściwości *Maska bitowa*). W połączeniu z właściwością Wartość dla True pozwala na dowolne umieszczenie krzywej binarnej w obszarze wykresu. W szczególności pozwala na rozmieszczenie na wykresie wielu krzywych binarnych, jedna nad drugą. Właściwość określa wartość punktu na wykresie odpowiadającą wartości *False* punktu serii. 
Wartość dla True	Liczba	Właściwość stosowana zawsze w parze z właściwością *Wartość dla False*. Określa wartość punktu na wykresie odpowiadającą wartości *True* punktu serii.
Opis	Tekst	Opis krzywej w legendzie wykresu. Wartość pusta oznacza użycie nazwy zmiennej serii wraz z jej opisem pobranym z bazy definicji zmiennych.

Baza obszaru/słupka	Liczba	Właściwość stosowana tylko dla wykresów w stylu obszarowym lub słupkowym. Określa poziom względem, którego zakreślane są obszary wartości lub rysowane słupki.  

Właściwości limitów

Nazwa	Typ wartości	Opis
Styl limitu	Typ wyliczeniowy: Linia, Pasmo	Określa sposób wyświetlenia limitu: w postaci linii lub pasma.
Wartość (początkowa)	Liczba	W przypadku pasma określa pierwszą wartość graniczną, w przypadku linii położenie linii limitu.
Wartość końcowa / grubość	Liczba	W przypadku pasma określa drugą wartość graniczną, a w przypadku linii jej szerokość.
Kolor limitu	Kolor	Określa kolor używany do wyświetlenia limitu.
Na wierzchu	Logiczna	Określa, czy limit wyświetlany jest nad krzywymi serii i innymi elementami wykresu.

UWAGA:

W przypadku wyświetlania limitu w postaci pasma, może zostać przesłonięta siatka wykresu i ewentualnie (w zależności od wartości właściwości *Na wierzchu*) krzywe serii. Uniknąć tego można stosując półprzezroczysty kolor limitu. W tym celu należy wprowadzić kolor w postaci liczbowej, np. kolor 120;238;130;238 oznacza półprzezroczysty kolor fioletowy. Można też ustawić właściwość *Siatka na wierzchu* z grupy właściwości podstawowych.

Właściwości stanowe

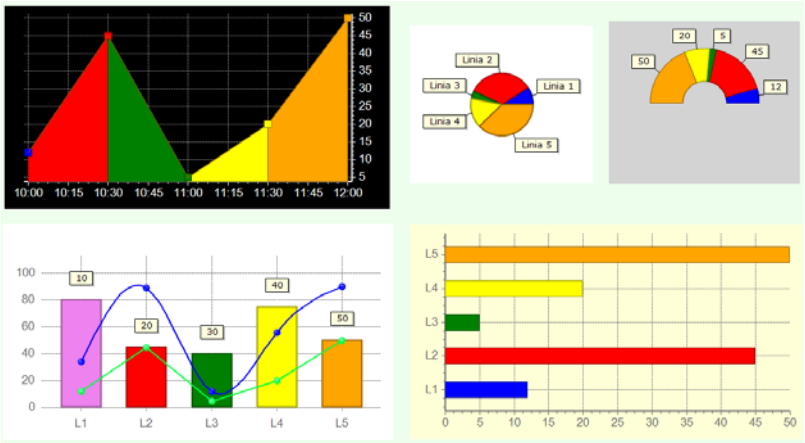
Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła wykresu
Kolor siatki	Kolor	Określa kolor siatki wykresu. Wybranie koloru *Transparent* powoduje, że siatka nie jest rysowana.
Kolor podpodziałek siatki wartości	Kolor	Określa kolor linii podpodziałek wartości na siatce wykresu. Wybranie koloru *Transparent* powoduje, że linie podpodziałek nie są rysowane.
Kolor podpodziałek siatki czasu	Kolor	Określa kolor linii podpodziałek czasu na siatce wykresu. Wybranie koloru Transparent powoduje, że linie podpodziałek nie są rysowane.
Kolor osi	Kolor	Określa kolor osi pionowej i poziomej wykresu.
Kolor opisu osi	Kolor	Określa kolor opisu skali osi pionowej i poziomej wykresu.
Nazwa czcionki osi	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlenia opisów osi.
Rozmiar czcionki osi	Liczba	Określa rozmiar czcionki używanej do wyświetlenia opisów osi.
Styl czcionki osi	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl czcionki używanej do wyświetlenia opisów osi. Można stosować wiele styli jednocześnie.
Kolor tytułu	Kolor	Określa kolor linii tytułu wykresu.
Nazwa czcionki tytułu	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlania tytułu wykresu.
Rozmiar czcionki tytułu	Liczba	Określa rozmiar czcionki używanej do wyświetlenia linii tytułu wykresu.
Styl czcionki tytułu	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie,	Określa styl czcionki używanej do wyświetlenia tytułu wykresu. Można stosować wiele styli jednocześnie.

	Kursywa, Podkreślenie, Przekreślenie	
Kolor czcionki legendy	Kolor	Określa kolor czcionki używanej do wyświetlenia legendy wykresu.
Kolor tła legendy	Kolor	Określa kolor tła obszaru legendy wykresu.
Nazwa czcionki legendy	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlenia legendy wykresu.
Rozmiar czcionki legendy	Liczba	Określa rozmiar czcionki używanej do wyświetlenia legendy wykresu.
Styl czcionki legendy	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl czcionki używanej do wyświetlenia legendy wykresu. Można stosować wiele styli jednocześnie.
Kolor kursora	Kolor	Określa kolor kursora. Do włączenia pokazywania kursora służy właściwość *Wyświetlaj kursor*.
Kolor tekstu kursora	Kolor	Określa kolor tekstu w polu wartości kursora.
Nazwa czcionki kursora	Nazwa czcionki	Systemowa nazwa czcionki używanej do wyświetlenia tekstu w polu wartości kursora.
Rozmiar czcionki kursora	Liczba	Określa rozmiar czcionki używanej do wyświetlenia tekstu w polu wartości kursora.
Styl czcionki kursora	Typ wyliczeniowy wielokrotny: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Określa styl czcionki używanej do wyświetlenia tekstu w polu wartości kursora. Można stosować wiele styli jednocześnie.

Obiekt Wykres Danych

Funkcje obiektu

Obiekt *Wykres Danych* jest obiektem umożliwiającym rysowanie dowolnych przebiegów i wykresów z danych dostarczanych przez skrypty za pomocą mechanizmu danych aplikacyjnych (AppData) lub za pomocą specjalnych funkcji (np. ToTable(..)). Obiekt nadaje się między innymi do tworzenia profesjonalnych wykresów OEE, zużycia energii, czasów pracy maszyn (Gantt) itp.



Rys. Przykłady obiektu Wykres Danych.

Na jednym wykresie możliwe jest rysowanie wielu przebiegów (serii). Każdej z serii można osobno określić dane, styl wykresu (punktowy, liniowy, liniowy gładki, liniowy schodkowy, słupkowy pionowy, słupkowy poziomy, obszarowy, obszarowy schodkowy, kołowy, pierścieniowy, Gantt'a), kolor, grubość linii, styl znaczników itp.

Kolejnym elementem wyświetlanym na wykresie są tzw. limity. Mogą być pokazywane w postaci linii lub pasma. Limity pokazywane są w postaci linii (pasm) poziomych i nie zależą od przebiegu serii.

W trybie operatorskim obiekt może być też obsługiwany przy pomocy myszki zależnie od wybranego trybu powiększania i przesuwania.

Operacja	*Znaczenie*
Zaznaczenie obszaru wykresu ruchem od lewej do prawej strony	Powiększenie wskazanego obszaru wykresu (zależnie od wybranego trybu przesuwania w osi X, Y lub obu).
Zaznaczenie obszaru wykresu ruchem od prawej do lewej strony	Reset wykresu do ustawień początkowych.
Przeciągnięcie osi X lub Y	Przesuniecie zakresu wartości osi X lub Y w żądanym kierunku (zależnie od wybranego trybu przesuwania w osi X, Y lub obu).
Przesuwanie z wciśniętym prawym klawiszem myszy	Dowolne przesuniecie obszaru wykresu (zależnie od wybranego trybu przesuwania w osi X, Y lub obu).

Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	**Typ wartości**	**Opis**
Źródło danych	DataTable	Pozwala ustanowić ogólne źródło danych dla serii w postaci typu wartości DataTable dostarczonego za pomocą funkcji AppData(..) lub jako wartość zmiennej. Dana aplikacyjna (AppData) lub zmienna może być ustawiana za pomocą skryptu jako wartość typu DataTable zawierająca zdefiniowane kolumny i wiersze. Każda seria wykresu może się odwołać do odpowiedniej kolumny (wartości X, Y, V etykiety, kolory).
Dziedzina czasu	Logiczna	Określa, czy oś pozioma X wykresu pokazuje wartości w postaci czasu lub liczbowe.
Tytuł wykresu	Tekst	Określa treść tytułu wykresu.

Styl legendy	Typ wyliczeniowy: Brak, Do góry, Do dołu, Do lewej, Do prawej	Określa widoczność i lokalizację legendy.
Wybór serii w legendzie	Logiczna	Określa, czy w legendzie zezwalamy operatorowi na sterowanie widocznością serii.
Pokazuj oś poziomą	Logiczna	Określa, czy wyświetlać podziałkę osi poziomej X.
Pokazuj osie pionowe	Logiczna	Określa, czy wyświetlać podziałkę osi pionowych (lewej i prawej) Y.
Dolna oś min	Liczba lub czas	Określa początkową wartość minimalną na osi dolnej poziomej.
Dolna oś max	Liczba lub czas	Określa końcową wartość maksymalną na osi dolnej poziomej.
Lewa oś min	Liczba	Określa początkową wartość minimalną na osi lewej pionowej.
Lewa oś max	Liczba	Określa końową wartość maksymalną na osi lewej pionowej.
Prawa oś min	Liczba	Określa początkową wartość minimalną na osi prawej pionowej.
Prawa oś max	Liczba	Określa końową wartość maksymalną na osi prawej pionowej.
Opisy dolnej osi	Logiczna	Określa, czy mają być wyświetlane opisy dolnej osi poziomej.
Opisy lewej osi	Logiczna	Określa, czy mają być wyświetlane opisy lewej osi pionowej.
Opisy prawej osi	Logiczna	Określa, czy mają być wyświetlane opisy prawej osi pionowej.
Skok dolnej osi	Liczba lub czas	Określa odstępy pomiędzy głównymi podziałkami dolnej osi.
Skok lewej osi	Liczba	Określa odstępy pomiędzy głównymi podziałkami lewej osi.
Skok prawej osi	Liczba	Określa odstępy pomiędzy głównymi podziałkami prawej osi.
Podpodziałki dolnej osi	Liczba	Określa liczbę podpodziałek pomiędzy głównymi podziałkami osi dolnej.

Podpodziałki lewej osi	Liczba	Określa liczbę podpodziałek pomiędzy głównymi podziałkami osi lewej.
Podpodziałki prawej osi	Liczba	Określa liczbę podpodziałek pomiędzy głównymi podziałkami osi prawej.
Obrót etykiet dolnej osi	Liczba	Określa kąt w stopniach, pod jakim wyświetlane są etykiety osi dolnej.
Obrót etykiet lewej osi	Liczba	Określa kąt w stopniach, pod jakim wyświetlane są etykiety osi lewej.
Obrót etykiet prawej osi	Liczba	Określa kąt w stopniach, pod jakim wyświetlane są etykiety osi prawej.
Format czasu	Tekst	Określa sugerowany sposób formatowania opisów podziałek na osi dolnej czasu. Na przykład, format HH:mm:ss tworzy znaczniki w postaci 13:30:45.
Stos procentowy	Logiczna	Określa, czy wartości serii połączonych w stos mają zostać przeskalowane na wartości procentowe.
Tryb powiekszania	Typ wyliczeniowy: Brak, W poziomie, W pionie, Oba kierunki	Określa, czy operator może zmieniać skalę wykresu w określonych kierunkach.
Tryb przesuwania	Typ wyliczeniowy: Brak, W poziomie, W pionie, Oba kierunki	Określa, czy operator może przesuwać wykres w określonych kierunkach.
Czcionka	Krój czcionki	Czcionka używana przez wykres.
Styl czcionki	Typ wyliczeniowy: Zwykły, Pogrubienie, Kursywa, Podkreślenie, Przekreślenie	Styl czcionki używanej przez wykres.

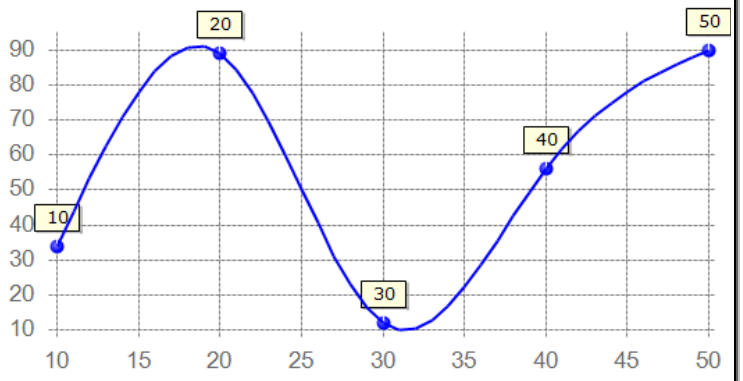
Rozmiar czcionki tytułu	Liczba	
Rozmiar czcionki osi	Liczba	Rozmiar czcionki używanej do wyświetlania etykiet osi.
Rozmiar czcionki legendy	Liczba	Rozmiar czcionki używanej do wyświetlania opisów w legendzie.
Lewy margines	Liczba	Rozmiar marginesu lewego pomiędzy krawędzią obiektu a obszarem wykresu.
Prawy margines	Liczba	Rozmiar marginesu prawego pomiędzy krawędzią obiektu a obszarem wykresu.
Górny margines	Liczba	Rozmiar marginesu górnego pomiędzy krawędzią obiektu a obszarem wykresu.
Dolny margines	Liczba	Rozmiar marginesu dolnego pomiędzy krawędzią obiektu a obszarem wykresu.

Właściwości limitów

Nazwa	**Typ wartości**	**Opis**
Styl limitu	Typ wyliczeniowy: Linia, Pasmo	Określa sposób wyświetlenia limitu: w postaci linii lub pasma.
Wartość (początkowa)	Liczba	W przypadku pasma określa pierwszą wartość graniczną, w przypadku linii położenie linii limitu.
Wartość końcowa / grubość	Liczba	W przypadku pasma określa drugą wartość graniczną, a w przypadku linii jej szerokość.
Kolor limitu	Kolor	Określa kolor używany do wyświetlenia limitu.
Na wierzchu	Logiczna	Określa, czy limit wyświetlany jest nad krzywymi serii i innymi elementami wykresu.

Właściwości serii

Nazwa	Typ wartości	Opis
Wartości X	Nazwa kolumny, tablica lub kolekcja wartości	Wartości współrzędnych osi poziomej serii lub początki odcinków serii Gantt'a. Jeżeli obiekt korzysta z ogólnego źródła danych (*Właściwości podstawowe - > Źródło danych*), to wystarczy nazwa kolumny tabeli. W innych przypadkach można się posługiwać funkcją ToTable(..) (np. =ToTable(Variable(V1), Variable(V2), Variable(V3)) do zdefiniowania punktów przebiegu albo korzystać z funkcji AppData(..) lub wartości zmiennej tablicowej do pozyskania tablicy punktów przebiegu.
Wartości Y	Nazwa kolumny, tablica lub kolekcja wartości	Wartości współrzędnych osi pionowej serii. Jeżeli obiekt korzysta z ogólnego źródła danych (*Właściwości podstawowe - > Źródło danych*), to wystarczy nazwa kolumny tabeli. W innych przypadkach można się posługiwać funkcją ToTable(..) (np. =ToTable(Variable(V1), Variable(V2), Variable(V3)) do zdefiniowania punktów przebiegu albo korzystać z funkcji AppData(..) lub wartości zmiennej tablicowej do pozyskania tablicy punktów przebiegu.
Wartości V	Nazwa kolumny, tablica lub kolekcja wartości	Końce odcinków serii Gantt'a. Jeżeli obiekt korzysta z ogólnego źródła danych (*Właściwości podstawowe - > Źródło danych*), to wystarczy nazwa kolumny tabeli. W innych przypadkach można się posługiwać funkcją ToTable(..) (np. =ToTable(Variable(V1), Variable(V2), Variable(V3)) do zdefiniowania punktów przebiegu albo korzystać z funkcji AppData(..) lub wartości zmiennej tablicowej do pozyskania tablicy punktów przebiegu.
Etykiety	Nazwa kolumny, tablica lub kolekcja tekstów	Teksty etykiet punktów serii. Jeżeli obiekt korzysta z ogólnego źródła danych (*Właściwości podstawowe - > Źródło danych*), to wystarczy nazwa kolumny tabeli zawierająca teksty. W innych przypadkach można się posługiwać funkcją ToTable(..) (np. =ToTable("Maszyna 1", "Maszyna 2", "Maszyna 3") do zdefiniowania tablicy etykiet albo korzystać z funkcji AppData(..) lub wartości zmiennej tablicowej do pozyskania tablicy tekstów.
Kolory	Nazwa kolumny, tablica lub kolekcja kolorów	Kolory punktów, odcinków lub obszarów serii. Jeżeli obiekt korzysta z ogólnego źródła danych (*Właściwości podstawowe - > Źródło danych*), to wystarczy nazwa kolumny tabeli zawierająca kolory. W innych przypadkach można się posługiwać funkcją ToTable(..) (np. =ToTable(Red,

		Green,Blue) do zdefiniowania tablicy etykiet albo korzystać z funkcji AppData(..) lub wartości zmiennej tablicowej do pozyskania tablicy kolorów.
Opis	Tekst	Nazwa lub opis serii, np. wyświetlana w legendzie.
Styl serii	Typ wyliczeniowy: punktowy, liniowy, liniowy gładki, liniowy schodkowy, słupkowy pionowy, słupkowy poziomy, obszarowy, obszarowy schodkowy, kołowy, pierścieniowy, Gantt'a	Określa sposób wyświetlania punktów serii.
Styl znaczników	Typ wyliczeniowy: Brak, Wartości X, Wartosci Y, Etykieta	Określa, czy przy każdym zarejestrowanym punkcie przebiegu należy wyświetlać znaczniki z informacją o punkcie. 
Punkty linii	Typ wyliczeniowy: Brak, Mały punkt, Punkt, Mały prostokąt, Prostokąt, Małe koło, Koło, Mały diament, Diament, Mała kula, Kula	Określa, czy należy (i w jakiej formie) wyświetlać przy każdym zarejestrowanym punkcie przebiegu symbol punktu.
Numer stosu	Liczba	Dla serii typu liniowego, obszarowego lub słupkowego pozwala zgrupować serie w stosy poprzez przypisanie im jednakowego numeru stosu.
Pierwszy kolor	Kolor	Określa główny kolor punktów, krzywej lub słupków.

Drugi kolor	Kolor	Określa dodatkowy kolor (np. obramowanie słupków).
Szerokość	Liczba	Określa szerokość rysowanej krzywej (lub obramowania słupka).
Baza obszaru/słupka lub kąt początkowy	Liczba	Dla wykresów w stylu obszarowym lub słupkowym określa poziom, względem którego zakreślane są obszary wartości lub rysowane słupki. Dla wykresów kołowych lub pierścieniowych określa kąt początkowy w stopniach.
Rozpiętość kątowa koła pierścienia	Liczba	Rozpiętość kąta (w stopniach) wykresów kołowych lub pierścieniowych
Prawa oś	Logiczna	Określa, czy seria jest przyczepiona do prawej lub lewej osi pionowej.
Akcja kliknięcia serii	Akcja operatorska	Pozwala zdefiniować akcję operatorską, która zostanie wykonana gdy operator kliknie w daną serię. Może służyć np. do wyświetlenia okienka ze szczegółami lub do drążenia danych. W akcji można odwołać się za pomocą funkcji EventContext do danych szczegółowych zwiazanaych z kliknięciem: SeriesIndex, PointIndex, PointX, PointY, PointLabel, MinX, MaxX, MinY, MaxY.

Właściwości stanowe

Nazwa	**Typ wartości**	**Opis**
Kolor tła	Kolor	Określa kolor tła wykresu.
Kolor kanwy	Kolor	Określa kolor tła obszaru roboczego.
Kolor tytułu	Kolor	Określa kolor linii tytułu wykresu.
Kolor czcionki legendy	Kolor	Określa kolor czcionki używanej do wyświetlenia legendy wykresu.
Kolor tła legendy	Kolor	Określa kolor tła obszaru legendy wykresu.
Kolor dolnej osi	Kolor	Kolor dolnej osi.
Kolor lewej osi	Kolor	Kolor lewej osi.
Kolor prawej osi	Kolor	Kolor prawej osi.

Kolor etykiet dolnej osi	Kolor	Kolor etykiet dolnej osi.
Kolor etykiet lewej osi	Kolor	Kolor etykiet lewej osi.
Kolor etykiet prawej osi	Kolor	Kolor etykiet prawej osi.
Kolor siatki dolnej osi	Kolor	Kolor siatki dolnej osi.
Kolor siatki lewej osi	Kolor	Kolor siatki lewej osi.
Kolor siatki prawej osi	Kolor	Kolor siatki prawej osi.
Kolor podpodziałek dolnej osi	Kolor	Kolor podpodziałek dolnej osi.
Kolor podpodziałek lewej osi	Kolor	Kolor podpodziałek lewej osi.
Kolor podpodziałek prawej osi	Kolor	Kolor podpodziałek prawej osi.

Zdarzenia

Nazwa	Typ wartości	Opis
Zmiana widoku	Akcja operatorska	Pozwala zdefiniować akcje operatorską wywoływaną po każdej zmianie skali wykresu lub każdym przesunieciu jego obszaru. W akcji można odwołać się za pomocą funkcji EventContext do danych szczegółowych zwiazanych ze zdarzeniem: MinX, MaxX, MinY, MaxY

Przykłady parametryzacji:

- Wykres ad-hoc z wartości bieżących – wykorzystanie funkcji ToTable(..) do tworzenia tablic wartości:

Wartości X	
Wartości Y	=ToTable(Variable(V1),Variable(V2),Variable(V3),Variable(V4),Variable(V5))
Wartości V	
Etykiety	=ToTable("Var1","Var2","Var3","Var4","Var5")
Kolory	=ToTable(Blue, Red, Green, Yellow,Orange)

- Wykres z tabeli danych (kolumny X,Y,V,L,C) tworzonej przez skrypt i zapisywanej w AppData – wykorzystanie funkcji AppData(..) do odczytu tej tabeli:

Źródło danych	**=AppData("gantt")**
Dziedzina czasu	**Tak**

Wartości X	**X**
Wartości Y	**Y**
Wartości V	**V**
Etykiety	**L**
Kolory	**C**

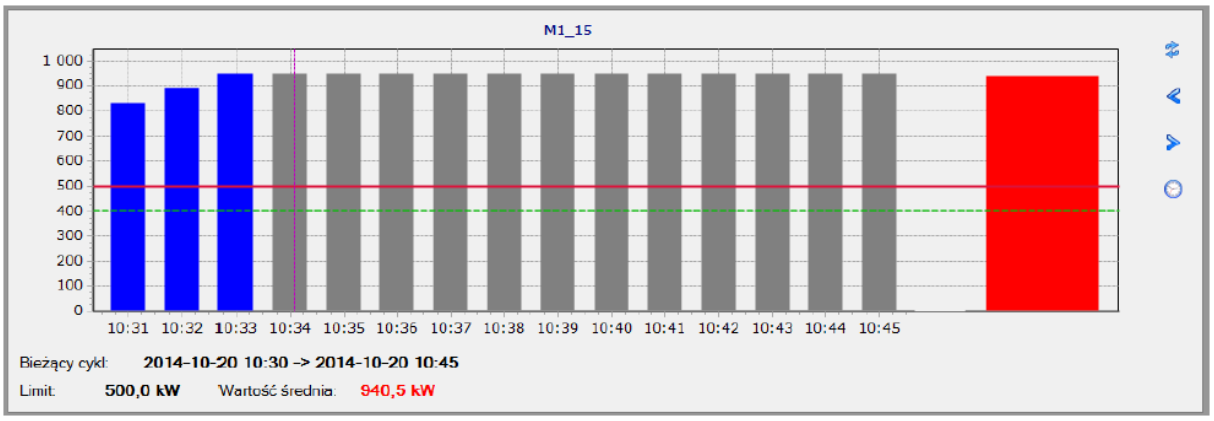
- Wyświetlenie stacyjki ze szczegółowymi danymi po kliknięciu w słupek serii – wykorzystanie funkcji EventContext(..) do pobrania informacji o współrzędnej X kliknięgo słupka:

Akcja kliknięcia serii	**^OpenDiagram("BarDetails", "BarX="+EventContext("PointX"), "Szczegóły", $Normal, $CursorLocation, $CursorLocation, "", $None)**

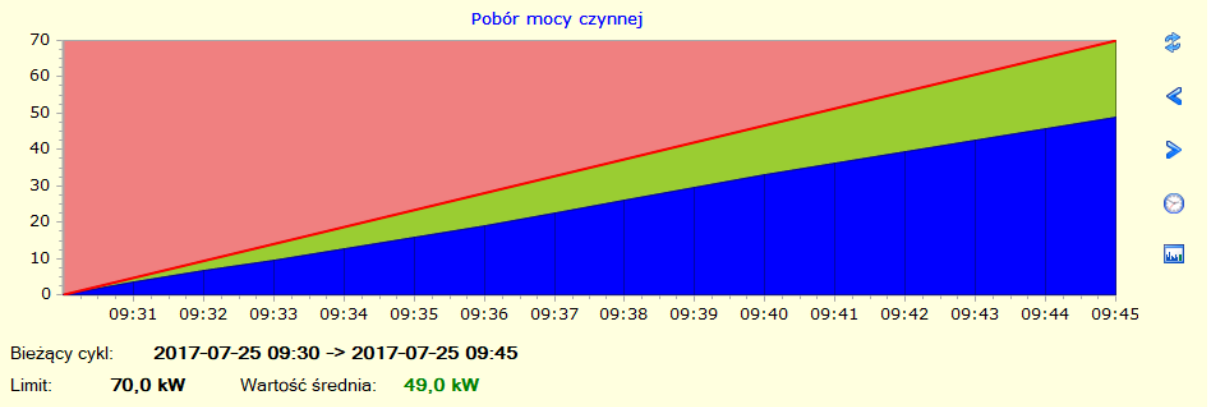
Obiekt Wykres strażnika mocy

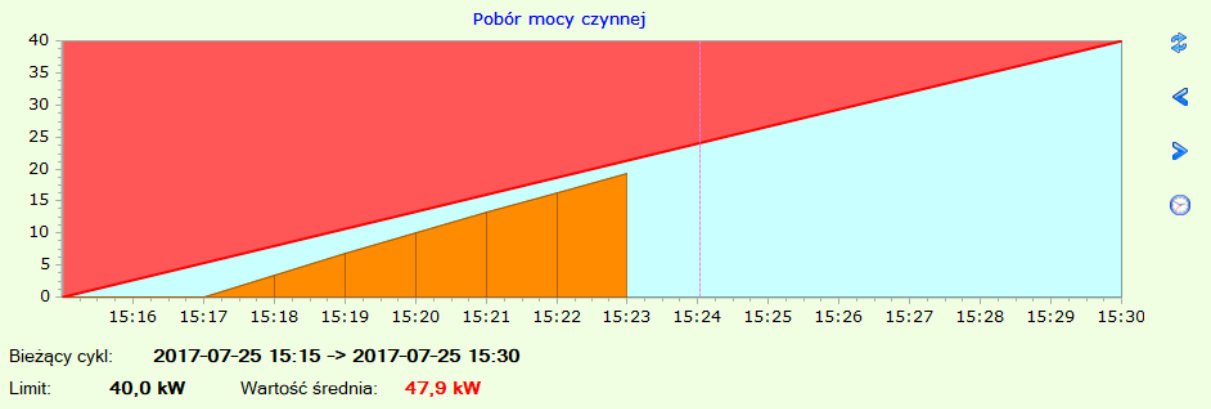
Funkcje obiektu

Obiekt *Wykres strażnika mocy* służy do wyświetlenia wykresu dla wybranego miernika modułu Strażnik mocy, pozwalającego obserwować interwały oraz średnią wartość prognozowaną w bieżącym cyklu. Ponadto, wykres wyświetla pionową linię czasu bieżącego, poziome linie limitu oraz poziomą linię, pokazującą jaka wartość jest bezpieczna dla pozostałych interwałów - tak aby całkowita średnia za cykl nie przekroczyła limitu. Wykres umożliwia także przegląd cyklów historycznych za pomocą przycisków nawigacyjnych z prawej strony.



Rys. Przykłady obiektu Wykres strażnika mocy (słupkowy).





Rys. Przykłady obiektu Wykres strażnika mocy (trójkątny).

Właściwości

Zobacz: ***Elementy wizualizacji > Standardowe właściwości obiektów***

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Identyfikator miernika	Tekst	Określa miernik modułu Strażnik mocy, którego dane będą pokazywane.
Tryb trójkątny	Logiczna	Umożliwia przełączenie sposobu wyświetlania wykresu na tryb trójkątny (przyrostowy).
Zezwalaj na zmiane trybu	Logiczna	Określa, czy wyświetlać z prawej strony wykresu przycisk umożliwiający operatorwi przełączanie trybu wyświetlania.
Pokazuj tytuł	Logiczna	Określa czy rysować linię tytułu. Jako treść tytułu używany jest opis miernika podany w jego definicji.
Pokazuj stopkę	Logiczna	Określa czy rysować stopkę zawierającą podstawowe dane dotyczące pokazywanego okresu pracy miernika.
Pokazuj przyciski	Logiczna	Określa czy rysować przyciski nawigacyjne.
Pokazuj przewidywane interwały	Logiczna	Określa czy rysować linię tytułu. Jako treść tytułu używany jest opis miernika podany w jego definicji.
Pokazuj czas bieżący	Logiczna	Określa czy rysować linię kursora pokazującego aktualną chwilę czasu.
Pokazuj linię bezpieczeństwa	Logiczna	Określa czy rysować linię pokazującą maksymalną wartość miernika, przy której nie nastąpi przekroczenie dopuszczalnego limitu na końcu cyklu.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor tła	Kolor	Określa kolor tła wykresu.
Kolor siatki	Kolor	Określa kolor siatki wykresu.
Kolor osi	Kolor	Określa kolor osi i krawędzi wykresu.
Kolor opisu osi	Kolor	Określa kolor opisów osi wykresu (podziałek).
Kolor tytułu	Kolor	Określa kolor linii tytułu.
Kolor tekstu	Kolor	Określa kolor tekstu w stopce wykresu.

Kolor interwałów	Kolor	Określa kolor słupków wartości zakończonych interwałów.
Kolor interwałów niepewnych	Kolor	Określa kolor słupków wartości zakończonych interwałów o niepewnym statusie wyliczenia.
Kolor interwałów przewidywanych	Kolor	Określa kolor słupków przewidywanych wartości niezakończonych interwałów.
Kolor niepewnych interwałów przewidywanych	Kolor	Określa kolor słupków przewidywanych wartości niezakończonych interwałów o niepewnym statusie wyliczenia.
Kolor wartości średniej	Kolor	Określa kolor słupka wartości średniej, gdy mieści się ona poniżej limitu.
Kolor przekroczonej wartości średniej	Kolor	Określa kolor słupka wartości średniej, gdy przewidywane jest przekroczenie limitu.
Kolor linii limitu	Kolor	Określa kolor linii limitu.
Kolor linii czasu	Kolor	Określa kolor linii kursora aktualnej chwili czasu.
Kolor linii bezpieczeństwa	Kolor	Określa kolor linii wartości miernika, przy której nie nastąpi przekroczenie dopuszczalnego limitu na końcu cyklu.

Zdarzenia

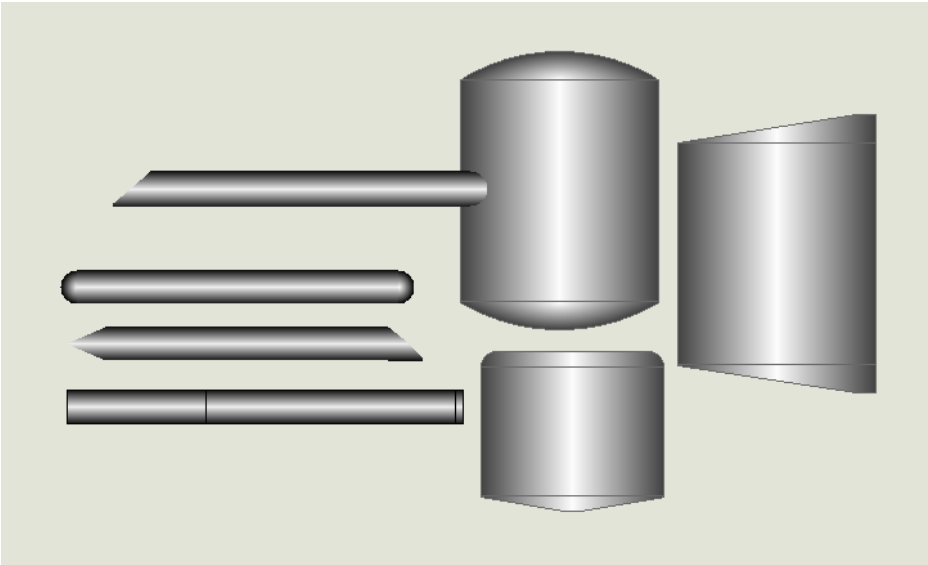
Obiekt nie obsługuje żadnych zdarzeń.

Obiekt Zbiornik

Funkcje obiektu

Obiekt *Zbiornik p*ozwala wyświetlać kształty zbiorników. Dla każdego zbiornika możemy zdefiniować rozmiary i styl jego górnego (prawego) oraz dolnego (lewego) ujścia. Dodatkowo można zdecydować,

czy ma być wyświetlana otoczka oraz separatory i określić ich kolor. Efekt 3D uzyskiwany jest poprzez odpowiednie manipulowanie kolorem gradientu.



Rys. Przykłady obiektu Zbiornik.

Obiekt *Zbiornik* nie posiada wbudowanych żadnych funkcji interaktywnych i sterujących. Można jednak taką funkcjonalność dodać poprzez użycie akcji operatorskich w obsłudze zdarzeń.

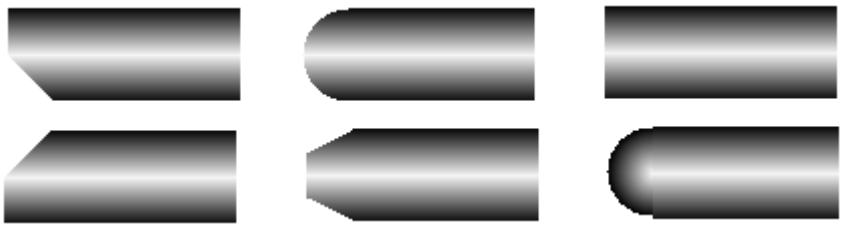
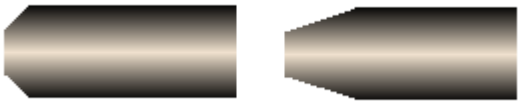
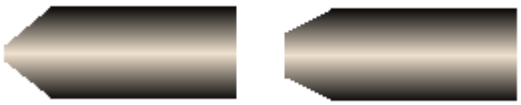
Zobacz: ***Techniki budowy diagramów > Parametryzacja funkcji interaktywnych w obiektach pasywnych***

Właściwości

Zobacz *Standardowe właściwości obiektów*

Właściwości podstawowe

Nazwa	Typ wartości	Opis
Układ poziomy	Logiczna	Określa, czy zbiornik jest rysowany w układzie pionowym lub poziomym.

Styl początku	Typ wyliczeniowy: Brak, Płaski, Kołnierz, Okrągły, Ścięty z lewej, Ścięty z prawej, Zaokrąglony, Stożkowy.	Określa sposób rysowania lewego (górnego) ujścia. 
Styl końca	Typ wyliczeniowy: Brak, Płaski, Kołnierz, Okrągły, Ścięty z lewej, Ścięty z prawej, Zaokrąglony, Stożkowy.	Określa sposób rysowania prawego (dolnego) ujścia.
Rozmiar początku [%]	Procent	Określa długość lewego (górnego) ujścia. Długość określana jest w procentach całkowitej długości (wysokości) zbiornika. 
Rozmiar końca [%]	Procent	Określa długość prawego (dolnego) ujścia. Długość określana jest w procentach całkowitej długości (wysokości) zbiornika.
Rozmiar wlotu [%]	Procent	Określa szerokość lewego (górnego) ujścia. Szerokość określana jest w procentach całkowitej szerokości zbiornika. 
Rozmiar wylotu [%]	Procent	Określa szerokość prawego (dolnego) ujścia. Szerokość określana jest w procentach całkowitej szerokości zbiornika.
Pokazuj separatory	Logiczna	Określa, czy linie separatorów wlotu mają być rysowane.  Kolor linii separatora określa właściwość *Kolor krawędzi*.

Właściwości stanowe

Nazwa	Typ wartości	Opis
Kolor	Kolor	Określa podstawowy kolor wypełnienia tła zbiornika.
Kolor gradientu	Kolor	Określa kolor stosowany do gradientowego wypełnienia tła. Kolor tła będzie płynnie zmieniany pomiędzy kolorem podstawowym, a kolorem gradientu.
Kolor krawędzi	Kolor	Określa kolor rysowania krawędzi zbiornika oraz separatorów. 

Zdarzenia

Zobacz: ***Elementy wizualizacji > Standardowe zdarzenia obiektów***

Asix.Evo - Obiekty zewnętrzne

1 Dostarczanie własnych obiektów do aplikacji Asix.Evo

Zewnętrzne obiekty dla aplikacji Asix Evo należy dostarczać w postaci skompilowanych bibliotek .NET jako pliki (*.dll*). Biblioteki te oraz wszystkie inne powiązane z nimi biblioteki i zasoby należy umieszczać w katalogu **Addons**, który jest podkatalogiem aplikacji Asixa Evo. Nowe lub zmienione obiekty i biblioteki będą widoczne po zrestartowaniu programu AsixEvo i aplikacji. Nazwa biblioteki zawierającej obiekty dla Asixa Evo musi rozpoczynać się przedrostkiem *„EvoObject.”* np.: EvoObject.TabeleSQL.dll.

UWAGA:

Ponieważ obiekty zewnętrzne dostarczane są w postaci bibliotek .NET, więc mogą zawierać dowolny kod, w szczególności kod niebezpieczny lub niestabilny. Kontrolki należy pisać unikając mocno obciążających operacji, zawieszających działanie wątku głównego (synchroniczne wywołania oczekujące), powodujących wycieki pamięci i inne niepożądane efekty. Za błędne lub niestabilne działanie dodatków do Asixa Evo dostarczonych przez użytkowników firma ASKOM nie ponosi odpowiedzialności.

2 Tworzenie kontrolek obiektów

2.1 Środowisko programistyczne

Do tworzenia własnych kontrolek obiektów zalecane jest używanie narzędzi programistycznych Microsoft: Visual Studio (2008 lub nowszy) lub darmowy Visual C# Express, zgodnych z platformą .NET Framework 3.5.

2.2 Klasa bazowa kontrolki

```
/// <summary>
/// Base class for Asix Evo diagram external objects
/// / Klasa bazowa dla zewnętrznych obiektów diagramowych
/// </summary>

public partial class EvoObjectBase : UserControl, AsixEvoAddon

{

  /// <summary>
   /// Override to return unique EvoObject ID. The proper ID is required.
   /// / Należy przeciążyć aby zwracać identyfukator obiektu. Wymagane!
   /// </summary>
   public virtual string EvoObjectID { get { return null; } }

   /// <summary>
   /// Override to return evo object description
   /// / Można prtzeciążyć aby zwracać nazwę (krótki opips) obiektu
   /// </summary>
   public virtual string EvoObjectDescription { get { return null; } }

   /// <summary>
   /// Override to initialize or reset control (e.g. register variable events)
   /// / Można przeciążyć aby dokonać inicjalizacji lub reinicjalizacji kontrolki
   /// (n.p. rejestrowanie zdarzeń zmiennych)
   /// </summary>
   public virtual void Reset() { }

   /// <summary>
   /// Override to cleanup when control is inactive/hidden etc. (e.g. unregister variable e
vents)
   /// / Można przeciążyć aby posprzątać (n.p. wyrejestrowanie zdarzeń zemiennych)
   /// </summary>
   public virtual void Disconnect() { }

   /// <summary>
   /// Override and return true to enable transparency for object in Control mode
   /// / Można przeciążyć i zwrócić true aby włączyć obsługę przeźroczystości dla
   /// obiektu kontrolkowego
   /// </summary>
   public virtual bool EvoObjectUsesTransparency { get { return false; } }

   /// <summary>
   /// Override and return true to determine if object is in OwnerDraw mode (vector, not co
ntrol)
   /// / Należy przeciążyć i zwrócić true aby określić że obiekt
   /// jest w trybie OwnerDraw (wektorowy, nie kontrolkowy)
   /// </summary>
   public virtual bool OwnerDraw { get { return false; } }

   /// <summary>
```

```
        /// Overide to draw object if object is in OwnerDraw mode
        /// / Należy przeciążyć aby malować obiekt gdy jest on w trybie OwnerDraw
        ///  (wektorowy, nie kontrolkowy)
        /// </summary>
        /// <param name="aGraphics">GDI + interface / Interfejs GDI+</param>
        /// <param name="aArea">Object bounds / Położenie i rozmiary obiektu</param>
        public virtual void Paint(Graphics aGraphics, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse click events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia klikniecia myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseClick(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle object focused events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia aktywacji obiektu
        /// </summary>
        public virtual void PerformFocus() { }

        /// <summary>
        /// Overide to handle object focus lost events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia dezaktywacji obiektu
        /// </summary>
        public virtual void PerformFocusLost() { }

        /// <summary>
        /// Overide to handle mouse double click events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia podwójnego klikniecia myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseDoubleClick(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse down events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia wciśniecia klawisza myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseDown(MouseEventArgs e, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse up events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia puszczenia myszy
        /// </summary>
        /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseUp(MouseEventArgs e, Rectangle aArea) { }


        /// <summary>
        /// Overide to handle mouse move events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia ruchu myszy
        /// </summary>
        /// <param name="aPoint">Cursor location on object/ Położenie kursora nad obiektam</para
m>
        /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
        public virtual void PerformMouseMove(System.Drawing.Point aPoint, Rectangle aArea) { }

        /// <summary>
        /// Overide to handle mouse enter events
        /// / Można przeciążyc aby obsłuyżyć zdarzenia najechania kursora myszy na obiekt
        /// </summary>
        public virtual void PerformMouseEnter() { }

        /// <summary>
```

```
    /// Overide to handle mouse leave events
    /// / Można przeciążyc aby obsłuyżyć zdarzenia opuszczenia przez kursor obszaru obiektu
    /// </summary>
    public virtual void PerformMouseLeave() { }

    /// <summary>
    /// Overide to handle mouse hold events
    /// / Można przeciążyc aby obsłuyżyć zdarzenia przytrzymania myszy nad obiektem
    /// </summary>
    /// <param name="e">Mouse events data / Szczegóły zdarzenia myszy</param>
    /// <param name="aArea">Current object area / Bieżący obszar obiektu</param>
    public virtual void PerformMouseHold(MouseEventArgs e, Rectangle aArea) { }

    /// <summary>
    /// Overide to handle key down events
    /// / Można przeciążyc aby obsłuyżyć zdarzenia wciśnięcia klawisza
    /// </summary>
    /// <param name="e">Key events data / Szczegóły zdarzenia klawisza</param>
    public virtual void PerformKeyDown(KeyEventArgs e) { }

    /// <summary>
    /// Overide to handle key up events
    /// / Można przeciążyc aby obsłuyżyć zdarzenia puszczenia klawisza
    /// </summary>
    /// <param name="e">Key events data / Szczegóły zdarzenia klawisza</param>
    public virtual void PerformKeyUp(KeyEventArgs e) { }

    /// <summary>
    /// Overide to handle key pressed events
    /// / Można przeciążyc aby obsłuyżyć zdarzenia wciśniecia klawisza znaku
    /// </summary>
    /// <param name="e">Key events data / Szczegóły zdarzenia klawisza</param>
    public virtual void PerformKeyPressed(KeyPressEventArgs e) { }




}
```

2.3 Tworzenie klasy kontrolki

Klasy kontrolek obiektów dla Asixa Evo najlepiej umieszczać w bibliotece utworzonej jako projekt typu Class Library. Klasa taka musi być publiczną klasą pochodną klasy *Askom.AddonManager.EvoObjectBase,* której definicja znajduje się w bibliotece *AsixEvoAddonManager.dll*. Referencja do biblioteki *AsixEvoAddonManager.dll* powinna być dodana do projektu. Plik *AsixEvoAddonManager.dll* znajduje się w katalogu instalacyjnym programu Asix Evo.

Najbardziej zalecaną metodą tworzenia kontrolki jest wykorzystanie w projekcie kreatora *Add NewItem->Windows Forms->Inherited User Control* , wskazanie w tym kreatorze (Browse…) pliku *AsixEvoAddonManager.dll,* a następnie wybranie z listy komponentu *EvoObjectBase*.

W klasie kontrolki (pochodnej klasy *Askom.AddonManager.EvoObjectBase)* należy obowiązkowo przeciążyć właściwość *EvoObjectID* , tak aby zwracała unikalny tekstowy identyfikator obiektu. Obiekty nie zwracające tego identyfikatora, zwracające pusty lub już wcześniej używany identyfikator, nie będą widoczne w aplikacji.

Opcjonalnie można dodatkowo przeciążyć właściwość *EvoObjectDescription*, tak aby zwracała krótką nazwę opisową obiektu, która będzie widoczna w przyborniku oraz jako nazwa typu obiektów na diagramach. Jeżeli nie zdefiniujemy tej nazwy, to będzie używany identyfikator z *EvoObjectID*.

Obiekt może pracować w trybie kontrolkowym lub wektorowym. W trybie kontrolkowym obiekt stanowi kontrolkę .NET, na której można umieszczać inne standardowe lub własne kontrolki .NET. W trybie wektorowym obiekt nie jest kontrolką, ale jest malowany za pomocą interfejsu GDI+ dla .NET, a odpowiednie zdarzenia obsługuje przeciążając właściwe metody wirtualne. Domyślnie obiekt pracuje w trybie kontrolkowym. Aby obiekt pracował w trybie wektorowym, to należy przeciążyć właściwość *OwnerDraw*, tak aby zwracała wartość *true*.

Jeżeli chcemy wykonywać w obiekcie jakieś czynności inicjalizacyjne czy czyszczące w chwilach, gdy obiekt jest podłączany do diagramu (np. podczas otwierania diagramu), to należy przeciążyć metodę *Reset()*.

Czyszczenie i zwalnianie zasobów przy zamknięciu i zwalnianiu kontrolki można wykonać podłączając się do zdarzenia *Disposed,* dostępnego w klasie bazowej, lub przeciążając metodę *Dispose*.

2.4 Definiowanie własnych właściwości obiektu

Jeżeli chcemy, aby pewne właściwości obiektu były dostępne dla aplikacji Asix Evo, należy je zadeklarować w klasie kontrolki jako zwykłe publiczne właściwości z obowiązkowymi akcesorami typu *set* oraz opcjonalnymi akcesorami typu *get*. Akcesory *get* mogą służyć do pozyskiwania wartości domyślnych dla właściwości obiektów Evo.

Każda właściwość komponentu, aby była widoczna w Aplikacji Evo, musi zostać oznakowana za pomocą atrybutu *Askom.AddonManager.EvoObjectProperty*. W atrybucie tym możemy opcjonalnie zdefiniować krótki opis tekstowy właściwości za pomocą parametru *Description*. Jeżeli tego nie zrobimy, opis ten będzie taki sam jak nazwa właściwości. Domyślnie właściwość będzie ulokowana w grupie właściwości podstawowych obiektu aplikacji Evo. Jeżeli chcemy, aby właściwość znajdowała się w grupie stanowej, to należy użyć w atrybucie opcjonalnego parametru *PropertyGroup* i nadać mu wartość *Askom.AddonManager.EvoObjectPropertyGroup.StateGroup*. Istnieje także możliwość umieszczenia właściwości we własnych grupach. W tym celu należy użyć w atrybucie opcjonalnego parametru *PropertyGroup* i nadać mu wartość *Askom.AddonManager.EvoObjectPropertyGroup.CustomGroup* oraz za pomocą parametru *CustomGroupName* określić nazwę własnej grupy. Jeżeli kilka właściwości będzie używało tej samej nazwy grupy, to będą one wspólnie znajdować się tej grupie.

Za pomocą parametru atrybutu *Askom.AddonManager.EditorType* możemy wskazać, jaki edytor wartości ma być używany w edytorze właściwości. Jeżeli nie użyjemy tego parametru, lub ustawimy go na wartość *Default,* to prosty edytor wartości zostanie określony na podstawie typu właściwości.

Edytor wartości	*Zalecany typ*
Default	string
Text	int
Logical	int
Color	int
Cursor	Int
Integer	Int
Double	bool
Variable	bool
FontName	int
FontStyle	int
PictureName	string
Shortcut	string
Attribute	string

2.5 Dostęp do standardowych właściwości obiektu

Dostęp do standardowych właściwości obiektu (takich jak np.: *Zmienna główna*, *Aktywność*, itp.) realizujemy poprzez zadeklarowanie w klasie kontrolki publicznych właściwości o dowolnej nazwie z obowiązkowymi akcesorami typu *set* oraz oznaczenie ich za pomocą atrybutu *Askom.AddonManager.EvoObjectStandardProperty*. Obowiązkowym parametrem atrybutu *EvoObjectStandardProperty* jest nazwa właściwości standardowej (typ wyliczeniowy

Askom.AddonManager.EvoObjectStandardPropertyName). Dla każdej z dostępnych właściwości standardowych powinien być używany odpowiedni typ właściwości:

Nazwa właściwości	*Wymagany typ*	*Opis*
Name	string	Nazwa obiektu
X	int	Współrzędna X (wartość względna dla diagramu od 0 do 1 000000)
Y	int	Współrzędna X (wartość względna dla diagramu od 0 do 1 000000)
Width	int	Szerokość (wartość względna dla diagramu od 0 do 1 000000)
Height	Int	Wysokość X (wartość względna dla diagramu od 0 do 1 000000)
Z	Int	Warstwa
Active	bool	Informuje czy obiekt jest aktywny
Visible	bool	Informuje czy obiekt jest widoczny
MinVisibleWidth	int	Minimalna szerokość widzialności (w pikselach)
MinVisibleHeight	int	Minimalna wysokość widzialności (w pikselach)
MainVar	string	Nazwa zmiennej głównej obiektu
ControlVar	string	Nazwa zmiennej sterowanej obiektu
Hint	string	Tekst dymka
CursorType	string	Nazwa typu kursora

Z punktu widzenia kontrolki zewnętrznej, właściwości standardowe są tylko do odczytu. Ustawianie ich wartości przez kontrolkę nie ma wpływu na działanie aplikacji.

2.6 Definiowanie własnych zdarzeń obiektu

Aby w aplikacji Asix Evo były widoczne określone zdarzenia obiektu, to należy je w klasie kontrolki zdefiniować jako publiczne zdarzenia (*event*) typu *Askom.AddonManager.EvoObjectEventHandler* oraz oznakować atrybutem *Askom.AddonManager.EvoObjectEvent*. W atrybucie tym, podobnie jak dla własnych właściwości, można opcjonalnie zdefiniować krótki opis za pomocą parametru *Description*.

2.7 Używanie przeźroczystości w obiektach kontrolkowych

Aby obiekt zewnętrzny w trybie kontrolkowym obsługiwał przeźroczystość (brak tła), to należy w klasie kontrolki (pochodnej klasy *Askom.AddonManager.EvoObjectBase*) przeciążyć właściwość *EvoObjectUsesTransprancy,* tak aby zawsze zwracała wartość *true*. Dodatkowo aby kontrolka sama nie malowała tła, konieczne jest przeciążenie właściwości systemowej *CreateParams* w sposób przedstawiony poniżej:

```
protected override CreateParams CreateParams
{
  get
   {
      const int WS_EX_TRANSPARENT = 0x20;
      CreateParams cp = base.CreateParams;
      cp.ExStyle |= WS_EX_TRANSPARENT;
      return cp;
   }
}
```

2.8 Pobieranie stanu i wartości zmiennych

Do pobrania wartości bieżącej zmiennej należy użyć funkcji:

```
VariableState GetVariableState(string aVarName);
```

aVarName - nazwa zmiennej

Funkcja te jest dostępna bezpośrednio w klasie bazowej kontrolki oraz w klasie *AsixEvo.Application*. Jeżeli zmienna istnieje, to funkcja zwraca obiekt deskryptora stanu zmiennej. Deskryptor zawiera informacje o aktualnej wartości (surowej, zwykłej i sformatowanej), o czasie zmiennej oraz status OPC. Gdy zmienna o danej nazwie nie istnieje, to funkcja zwróci *null*. Funkcja nie gwarantuje, że pobrany stan będzie odpowiadał rzeczywistej wartości aktualnej - jeżeli zmienna nie została zarejestrowana do nasłuchiwania (odświeżania).

2.9 Pobieranie wartości atrybutów zmiennych

Do pobrania wartości atrybutów zmiennej należy użyć funkcji:

```
object GetVariableAttribute(string aVarName, string aAttributeName);
```

aVarName	- nazwa zmiennej
aAttributeName	- nazwa atrybutu

Funkcja te jest dostępna bezpośrednio w klasie bazowej kontrolki oraz w klasie *AsixEvo.Application*. Jeżeli zmienna oraz atrybut istnieją, to funkcja zwraca wartość atrybutu zmiennej, w przeciwnym przypadku funkcja zwróci *null*.

2.10 Sterowanie zmiennych

Do sterowania zmiennych należy wykorzystać funkcję *AsixEvo.Application.SetVariableValue(&ldots;)*. Funkcja jest dostępna w następujących wersjach:

```
bool SetVariableValue(string aVarName, object aValue, bool aVerifyPrivileges);

bool SetVariableValue(string aVarName, object aValue, uint aStatus, bool aVerifyPrivileges);

bool SetVariableValue(string aVarName, object aValue, bool aVerifyPrivileges, out string
aErrorMsg);

bool SetVariableValue(string aVarName, object aValue, uint aStatus, bool aVerifyPrivileges,
out string aErrorMsg);
```

aVarName	- nazwa zmiennej
aValue	- nowa wartość
aStatus	- status OPC
aVarifyPrivileges	- określa, czy sprawdzać uprawnienia do sterowania. Jeżeli sterujemy ze sprawdzaniem uprawnień, to w przypadku braku uprawnień zostanie wyświetlone okienko umożliwiające tymczasowe (na czas sterowania) zalogowanie na użytkownika posiadającego uprawnienia.

Wersja funkcji bez argumentu aStatus domyślnie ustawia status OPC jako dobry.

Wersja funkcji z argumentem *aErrorMsg* zwraca za pomocą tego argumentu informacje o błędzie jeżeli sterowanie się nie powiedzie zamiast wyświetlać okienko dialogowe z opisem błędu.

Funkcje te są dostępne również bezpośrednio w klasie bazowej kontrolki.

2.11 Rejestrowanie i wyrejestrowywanie zdarzeń zmiany stanu zmiennych

Do zarejestrowania zdarzeń zmiany stanu zmiennej należy użyć funkcji:

```
bool RegisterVariable(string aVarName);
```

aVarName - nazwa zmiennej

Funkcja jest dostępna w klasie bazowej kontrolki. Jeżeli zmienna istnieje i udało się podłączyć do rejestrowania jej zdarzeń, to funkcja zwróci *true*.

Aby w kontrolce przechwytywać zdarzenia od zmiany stanu zarejestrowanych zmiennych, należy w klasie zdefiniować funkcję o dowolnej nazwie, ale zgodną z poniższym schematem:

```
void VarRefreshTest_VariableStateChangedEvent(string aVariableName, VariableState aVariableSta
te)
	{

	}
```

A następnie w konstruktorze podłączyć do niej zdarzenia korzystając ze znajdującego się w klasie bazowej kontrolki zdarzenia *VariableStateChangedEvent*:

```
	VariableStateChangedEvent += VarRefreshTest_VariableStateChangedEvent;
```

Tak zdefiniowana funkcja będzie przechwytywała zdarzenia zmiany wartości, stanu lub czasu wszystkich zarejestrowanych zmiennych. Pierwszy argument funkcji *aVariableName* informuje o tym, której zmiennej dotyczy zmiana. Drugi argument zawiera deskryptor stanu zmiennej.

Aby wyrejestrować przechwytywanie zdarzeń dla wskazanej zmiennej, należy wykorzystać funkcję:

```
void UnregisterVariable(string aVarName);
```

aVarName - nazwa zmiennej

Funkcja jest dostępna w klasie bazowej kontrolki.

Aby wyrejestrować przechwytywanie zdarzeń dla wszystkich zarejestrowanych zmiennych, należy użyć funkcji:

```
void UnregisterAllRegisteredVariables();
```

Funkcja jest dostępna w klasie bazowej kontrolki.

2.12 Dostęp do wartości archiwalnych zmiennych

Do wysłania zapytania o wartości archiwalne zmiennych służy funkcja:

```
IEvoAddonArcQueryResult OrderVariableArchData(string aVarName, string aAggregate, DateTime aStartTime,
DateTime aEndTime, TimeSpan aAggregateInterval);
```

aVarName	- nazwa zmiennej
aAggregate	- nazwa funkcji agregującej („none”, „average”, „start” itp.)
aStartTime	- czas początku zakresu danych archiwalnych
aEndTime	- czas końca zakresu danych archiwalnych
aAggregateInterval	- interwał agregacji

Powyższa metoda zwraca interfejs deskryptora wyniku *IEvoAddonArcQueryResult*, który umożliwia sprawdzenie statusu realizacji zapytania, pobranie jego wyników oraz rejestrację funkcji zwrotnej, która będzie wywołana po zmianie statusu, np. po wykonaniu zapytania lub jego błędzie). Po zakończeniu należy wywołać funkcję *Dispose()*.

```
public interface IEvoAddonArcQueryResult
{
	string VariableName { get; }
	EvoAddonArcResultStatus ResultStatus { get; }
	EvoAddonArcVariableState[] Values { get; }
	event EvoAddonArchQueryResultEvent ArchQueryResultEvent;
	void Dispose();
}
```

VariableName - nazwa zmiennej, na której wykonano zapytanie
ResultStatus - status realizacji zapytania (*Processing*, *Ready*, *Error*)
Values - tablica wartości wynikowych
ArchQueryResultEvent - zdarzenie zmiany statusu

2.13 Dostęp do tekstów wielojęzycznych aplikacji

Do pobierania napisów aplikacji w bieżącym języku służy funkcja:

```
string Askom.AddonManager.AsixEvo.Application.GetApplicationText(string aTextId);
```

Kod numeryczny bieżącego języka aplikacji można pobrać za pomocą właściwości:

```
int Askom.AddonManager.AsixEvo.Application.CurrentApplicationLanguageCode
```

Do przechwytywania zdarzeń zmiany języka aplikacji należy wykorzystać:

```
event EventHandler Askom.AddonManager.AsixEvo.Application.CurrentApplicationLanguageChanged;
```

2.14 Dostęp do obrazków aplikacji

Do pobierania obrazków z puli obrazków aplikacji należy używać funkcji:

```
Image Askom.AddonManager.AsixEvo.Application.GetImage(string aImageName)
```

2.15 Wykonywanie akcji operatorskich

Do wykonywania akcji złożonych należy używać funkcji funkcji:

```
bool Askom.AddonManager.AsixEvo.Application.ExecuteAction(string aAction, out string
aErrorMsg)
```

2.16 Dostęp do właściwości globalnych

Do pobierania wartości właściwości globalnych aplikacji należy używać funkcji funkcji:

```
 string GetGlobalProperty(string aPropertyId);
```

```
string Askom.AddonManager.AsixEvo.Application.GetGlobalProperty (string aPropertyId)
```

2.17 Obsługa zdarzeń w trybie wektorowym

Aby obiekt pracował w trybie wektorowym, to należy przeciążyć właściwość *OwnerDraw*, tak aby zwracała wartość *true*.

Podstawową metodą, którą należy przeciążyć i obsłużyć, gdy obiekt pracuje w trybie wektorowym, jest:

```
public virtual void Paint(Graphics aGraphics, Rectangle aArea) { }
```

W funkcji tej można malować zawartość obiektu za pomocą obiektu klasy Graphics, stanowiący w .NET odpowiednik GDI+. Należy zawsze uwzględniać aktualne rozmiary obiektu dostępne w parametrze *aArea,* tak aby obiekt wyglądał prawidłowo bez względu na skalę i położenie.

Pozostałe metody wirtualne dostępne w klasie bazowej (wypisane i opisane na str. 2) można przeciążać i wykorzystywać do obsługi zdarzeń interakcyjnych, jak np.: kliknięć myszą, wciśnięć klawiszy itp.

2.18 Przykład 1 – Definiowanie właściwości i zdarzeń

Poniżej zamieszczono przykładową definicję klasy własnego obiektu dla Asix Evo. Obiekt składa się z dwóch przycisków w układzie pionowym, dwóch pól edycyjnych tekstu oraz pola wyboru oraz pól etykiet. Pierwszy przycisk o stałej wysokości umieszczono na górze. Jego naciśnięcie powoduje wykonanie akcji określonej przez projektanta diagramu. Drugi przycisk pokrywa środkową część obszaru i służy do wykonywania sterowań zmiennych. Pierwsze pole edycyjne służy do wprowadzania nazwy zmiennej, domyślnie pobiera wartość z właściwości standardowej „Zmienna sterowana" obiektu. Drugi przycisk pozwala wprowadzać wartość dla zmiennych. Pole wyboru określa, czy sterowanie ma następować ze sprawdzaniem uprawnień. Dla kontrolki zdefiniowano własne właściwości i zdarzenia widoczne w aplikacji Asix Evo oraz odwołania do właściwości standardowych. Dodatkowo kontrolka pobiera z aplikacji określone napisy aplikacji (jeżeli istnieją w aplikacji) oraz reaguje na zmiany bieżącego języka.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using Askom.AddonManager;

namespace TestowaKontrolkaEvo
{
   /// <summary>
   /// Przykładowa klasa kontrolki obiektu dla Asix Evo.
   /// </summary>
   public class SampleEvoControl : Askom.AddonManager.EvoObjectBase
   {
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.Button button2;
      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.TextBox textBox2;
      private System.Windows.Forms.CheckBox checkBox1;
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.Label label2;

      public SampleEvoControl()
```

```
{
    // Inicjalizacja...

    SuspendLayout();

    // Przycisk 1
    button1 = new System.Windows.Forms.Button();
    button1.Dock = System.Windows.Forms.DockStyle.Top;
    button1.Name = "button1";
    button1.Text = "Rób coś";
    button1.UseVisualStyleBackColor = true;
    button1.Click += button1_Click;
    Controls.Add(button1);

    // Przycisk 2
    button2 = new System.Windows.Forms.Button();
    button2.Dock = System.Windows.Forms.DockStyle.Fill;
    button2.Name = "button2";
    button2.Text = "Steruj";
    button2.Click += button2_Click;
    Controls.Add(button2);

    // Etykieta 1
    label1 = new System.Windows.Forms.Label();
    label1.Dock = System.Windows.Forms.DockStyle.Bottom;
    label1.Name = "label1";
    label1.Text = "Zmienna:";
    Controls.Add(label1);

    // Pole tekstowe 1
    textBox1 = new System.Windows.Forms.TextBox();
    textBox1.Dock = System.Windows.Forms.DockStyle.Bottom;
    textBox1.Name = "textbox1";
```

```
            Controls.Add(textBox1);

            // Etykieta 2
            label2 = new System.Windows.Forms.Label();
            label2.Dock = System.Windows.Forms.DockStyle.Bottom;
            label2.Name = "label2";
            label2.Text = "Wartość:";
            Controls.Add(label2);

            // Pole tekstowe 2
            textBox2 = new System.Windows.Forms.TextBox();
            textBox2.Dock = System.Windows.Forms.DockStyle.Bottom;
            textBox2.Name = "textbox2";
            Controls.Add(textBox2);

            // Pole wyboru
            checkBox1 = new System.Windows.Forms.CheckBox();
            checkBox1.Dock = System.Windows.Forms.DockStyle.Bottom;
            checkBox1.Name = "checkBox1";
            checkBox1.Text = "Uprawnienia";
            checkBox1.Checked = true;
            Controls.Add(checkBox1);

            Name = "SampleEvoControl";

            ResumeLayout(false);

            if (AsixEvo.Application != null)
            {
               // Aktualizujemy opipsy na podstawie napisów aplikacji
               UpdateTexts();

               AsixEvo.Application.CurrentApplicationLanguageChanged += Application_CurrentApplic
ationLanguageChanged;
```

```
    }
}

public override string EvoObjectID
{
    get
    {
        return "SampleEvoControl";  // Unikalny identyfikator obiektu
    }
}

public override string EvoObjectDescription
{
    get
    {
        return "Kontrolka z przyciskami";  // Krótki opis obiektu
    }
}

public override void Reset()
{
    // Reinicjalizacja obiektu ...

    base.Reset();
}

/// <summary>
/// Przechwytywanie zdarzenia zmiany języka
/// </summary>
void Application_CurrentApplicationLanguageChanged(object sender, EventArgs e)
{
    UpdateTexts();
}
```

```
        /// <summary>
        /// Aktualizacja opipsów elemntów na podstawie napisów aplikacji
        /// </summary>
        void UpdateTexts()
        {
           if (AsixEvo.Application != null)
           {
              string txt = AsixEvo.Application.GetApplicationText("action");
              if (!string.IsNullOrEmpty(txt))
                 button1.Text = txt;

              txt = AsixEvo.Application.GetApplicationText("control");
              if (!string.IsNullOrEmpty(txt))
                 button2.Text = txt;

              txt = AsixEvo.Application.GetApplicationText("var");
              if (!string.IsNullOrEmpty(txt))
                 label1.Text = txt;

              txt = AsixEvo.Application.GetApplicationText("val");
              if (!string.IsNullOrEmpty(txt))
                 label2.Text = txt;

              txt = AsixEvo.Application.GetApplicationText("priv");
              if (!string.IsNullOrEmpty(txt))
                 checkBox1.Text = txt;
           }
        }

        /// <summary>
        /// Właściwość napisu na pierwszym przycisku. Grupa podstawowa. Widoczna w aplikacji jak
o 'Text1'.
        /// </summary>
        [Askom.AddonManager.EvoObjectProperty]
```

```
      public string Text1
      {
         get { return button1.Text; }
         set { button1.Text = value; }
      }

      /// <summary>
      /// Właściwość napisu na drugim przycisku. Grupa własna o nazwie "Grupa1". Widoczna w ap
likacji jako 'Napis 2'.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty(Description = "Napis 2",
         PropertyGroup = Askom.AddonManager.EvoObjectPropertyGroup.CustomGroup, CustomGroupNam
e="Grupa1")]
      public string Text2
      {
         get { return button2.Text; }
         set { button2.Text = value; }
      }

      /// <summary>
      /// Właściwość koloru tła pierwszego przycisku. Grupa stanowa. Widoczna w aplikacji jako
 'Kolor 1'.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty(Description = "Kolor 1", PropertyGroup = Askom.Add
onManager.EvoObjectPropertyGroup.StateGroup)]
      public Color Color1
      {
         get { return button1.BackColor; }
         set { button1.BackColor = value; }
      }

      /// <summary>
      /// Właściwość koloru tła drugiego przycisku. Grupa stanowa. Widoczna w aplikacji jako '
Kolor 2'.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty(Description = "Kolor 2", PropertyGroup = Askom.Add
onManager.EvoObjectPropertyGroup.StateGroup)]
```

```
      public Color Color2
      {
         get { return button2.BackColor; }
         set { button2.BackColor = value; }
      }

      /// <summary>
      /// Dostęp do właściwosci standardowej "Zmienna główna" - wartość wyświetlana w polu edy
cyjnym
      /// </summary>
      [Askom.AddonManager.EvoObjectStandardProperty(Askom.AddonManager.EvoObjectStandardProper
tyName.MainVar)]
      public string MainVariable
      {
         set;
         get;
      }

      /// <summary>
      /// Dostęp do właściwosci standardowej "Zmienna sterowana" - skojarzenie jej z polem tek
stowym textBox1
      /// </summary>
      [Askom.AddonManager.EvoObjectStandardProperty(Askom.AddonManager.EvoObjectStandardProper
tyName.ControlVar)]
      public string ControlVariable
      {
         get { return textBox1.Text; }
         set { textBox1.Text = value; }
      }

      /// <summary>
      /// Zdarzenie kliknięcia pierwszego przycisku. Grupa zdarzeń. Widoczne w aplikacji jako
'Wciśnięcie 1'.
      /// </summary>
      [Askom.AddonManager.EvoObjectEvent(Description = "Wciśnięcie 1")]
      public event Askom.AddonManager.EvoObjectEventHandler click1;
```

```
        /// <summary>
        /// Wciśnięcie pierwszego przycisku
        /// </summary>
        private void button1_Click(object sender, EventArgs e)
        {
            // Jeżeli jakieś funkcje są zarejestrowane to wywołujemy zdarzenie
            if (click1 != null)
                click1();
        }

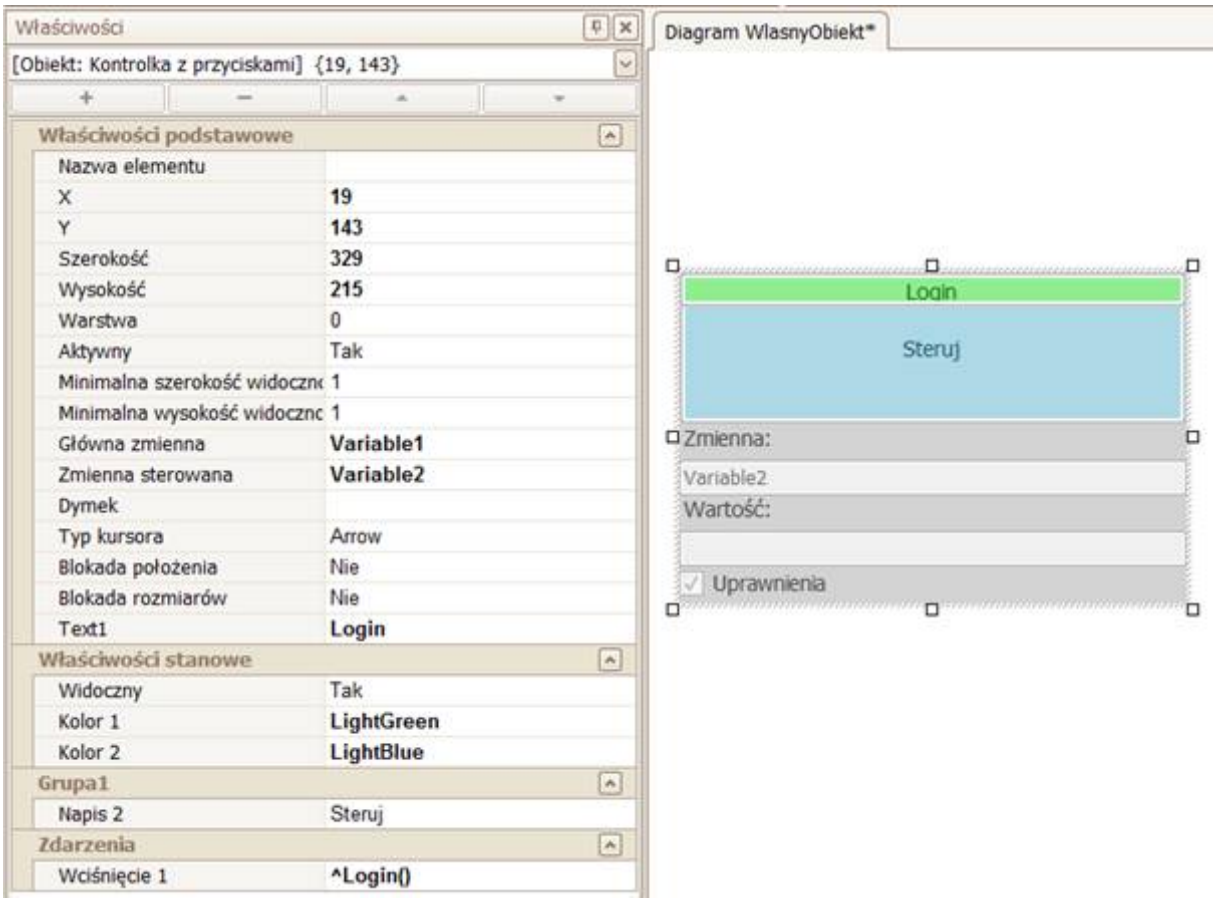

        /// <summary>
        /// Wciśnięcie drugiego przycisku
        /// </summary>
        private void button2_Click(object sender, EventArgs e)
        {
            // Wysłanie sterowania zmiennej
            if(AsixEvo.Application != null)
                AsixEvo.Application.SetVariableValue(ControlVariable, textBox2.Text, checkBox1.Che
cked);
        }


        protected override void Dispose(bool disposing)
        {
            // Sprzątamy...
            if (disposing)
            {
                if (AsixEvo.Application != null)
                {
                    AsixEvo.Application.CurrentApplicationLanguageChanged -
= Application_CurrentApplicationLanguageChanged;
                }


                button1.Click -= button1_Click;
                button2.Click -= button2_Click;
```

```
            }

            base.Dispose(disposing);
        }
    }
}
```



2.19 Przykład 2 – Praca ze zmiennymi

Poniższy kod prezentuje definicję obiektu dla AsixEvo, w którym wykorzystano opcję bezpośredniego dostępu do zmiennych, ich stanu, wartości i atrybutów. Obiekt składa się z trzech przycisków. Tekst pierwszego przycisku prezentuje wartości bieżące zmiennej systemowej „Counter”. Tekst drugiego przycisku pokazuje wartości bieżące zmiennej, której nazwa została podana jako wartość właściwości „Inna zmienna” obiektu. Dodatkowo, wartość ta jest poprzedzona tekstem pobranym z atrybutu „Opis” tej zmiennej. Tekst trzeciego przycisku pokazuje wartości bieżące zmiennej głównej obiektu.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using Askom.AddonManager;

namespace TestowaKontrolkaEvo
{
   /// <summary>
   /// Przykładowa klasa kontrolki obiektu dla Asix Evo korzystająca bezpośrednio ze zmiennych
.
   /// </summary>
   public class VarRefreshTest : Askom.AddonManager.EvoObjectBase
   {
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.Button button2;
      private System.Windows.Forms.Button button3;

      public VarRefreshTest()
      {
         // Inicjalizacja...

         SuspendLayout();

         // Przycisk 1
         button1 = new System.Windows.Forms.Button();
         button1.Dock = System.Windows.Forms.DockStyle.Top;
         button1.Name = "button1";
         button1.Text = "";
         Controls.Add(button1);

         // Przycisk 2
         button2 = new System.Windows.Forms.Button();
         button2.Dock = System.Windows.Forms.DockStyle.Fill;
         button2.Name = "button2";
         button2.Text = "";
         Controls.Add(button2);

         // Przycisk 3
         button3 = new System.Windows.Forms.Button();
         button3.Dock = System.Windows.Forms.DockStyle.Bottom;
         button3.Name = "button3";
         button3.Text = "";
         Controls.Add(button3);

         Name = "Odswiezanie";

         ResumeLayout(false);

         VariableStateChangedEvent += VarRefreshTest_VariableStateChangedEvent;
      }

      public override string EvoObjectID
      {
         get
         {
            return "Odswiezanie";  // Unikalny identyfikator obiektu
         }
      }

      public override string EvoObjectDescription
      {
         get
         {
            return "Nasluchiwanie zmiennych";  // Krótki opis obiektu
         }
      }

      string mDefaultVarName = "Counter";  // Counter jest zmienną systemową więc zawsze istni
eje.
```

```
      public override void Reset()
      {
         VariableState vs = GetVariableState(mDefaultVarName);
         if (vs != null)
         {
            RegisterVariable(mDefaultVarName); // Rejestrujemy zmienną wbudowaną
            button1.Text = vs.FormattedValue;  // Ustawiamy wartość aktualną
         }

         base.Reset();
      }

      public override void Disconnect()
      {
         UnregisterVariable(mDefaultVarName); // Wyrejestrowujemy zmienną wbudowaną
         base.Disconnect();
      }

      string mMainVarName = null;
      /// <summary>
      /// Dostęp do właściwosci standardowej "Zmienna główna" - wartość wyświetlana w polu edy
cyjnym
      /// </summary>
      [Askom.AddonManager.EvoObjectStandardProperty(Askom.AddonManager.EvoObjectStandardProper
tyName.MainVar)]
      public string MainVariable
      {
         set
         {
            UnregisterVariable(mMainVarName);  // Wyrejestrowujemy poprzednia zmienną
            mMainVarName = value;

            VariableState vs = GetVariableState(mMainVarName);
            if (vs != null)
            {
               RegisterVariable(mMainVarName);   // Rejestrujemy nową zmienną
               button3.Text = vs.FormattedValue; // Ustawiamy wartość aktualną
            }
         }
      }

      string mSecondVarName = null;
      string mSecondVarDescription = string.Empty;
      //// <summary>
      /// Właściwość napisu na drugim przycisku.
      /// </summary>
      [Askom.AddonManager.EvoObjectProperty(Description = "Inna zmienna", EditorType = EditorT
ype.Variable)]
      public string Var2
      {
         set
         {
            UnregisterVariable(mSecondVarName);  // Wyrejestrowujemy poprzednia zmienną
            mSecondVarName = value;

            VariableState vs = GetVariableState(mSecondVarName);
            if (vs != null)
            {
               object tob = GetVariableAttribute(mSecondVarName, "Description");
               if (tob != null)
                  mSecondVarDescription = tob.ToString();

               RegisterVariable(mSecondVarName); // Rejestrujemy nową zmienną
               button2.Text = mSecondVarDescription + " : " + vs.FormattedValue; // Ustawiamy
wartość aktualną
            }
         }

         get
         {
            return "DateTime"; // Wartość domyślna
         }
      }

      void VarRefreshTest_VariableStateChangedEvent(string aVariableName, VariableState aVaria
bleState)
```

```
      {
         if (string.Equals(aVariableName, mDefaultVarName, StringComparison.OrdinalIgnoreCase)
)
            button1.Text = aVariableState.FormattedValue;

         if (string.Equals(aVariableName, mSecondVarName, StringComparison.OrdinalIgnoreCase))
            button2.Text = mSecondVarDescription + " : " + aVariableState.FormattedValue;

         if (string.Equals(aVariableName, mMainVarName, StringComparison.OrdinalIgnoreCase))
            button3.Text = aVariableState.FormattedValue;
      }
   }
}
```

2.0 Przykład 3 – Obiekt wektorowy

Poniższy kod prezentuje definicję obiektu wektorowego dla AsixEvo. Obiekt jest wyświetlany jako prostokąt z ramką, zawierający czarny mniejszy prostokąt. Pozycja mniejszego prostokąta zależy od stanu obiektu. Stan obiektu można zmieniać, klikając w czarny prostokąt lub używając klawisza 'S'.

```
public class PaintedObjectTest : Askom.AddonManager.EvoObjectBase

   {

      bool switched = false;  // Stan obiektu


      public PaintedObjectTest()

      {

         // Inicjalizacja...

      }


      public override bool OwnerDraw

      {

         get

         {

            return true;  // Obiekt w trybie wektorowym (a nie kontrolkowym)

         }

      }
```

```
        public override string EvoObjectID
        {
            get
            {
                return "ObiektWektorowy";  // Unikalny identyfikator obiektu
            }
        }


        public override string EvoObjectDescription
        {
            get
            {
                return "Obiekt wektorowy";  // Krótki opis obiektu
            }
        }



        Color mColor1 = Color.Blue;
        //// <summary>
        /// Właściwość koloru tła
        /// </summary>
        [Askom.AddonManager.EvoObjectProperty(Description = "Kolor", EditorType = EditorType.Col
or, PropertyGroup = EvoObjectPropertyGroup.StateGroup)]
        public Color Color1
        {
            set
            {
                mColor1 = value;
                InvalidateObject();
            }


            get
            {
                return Color.Blue; // Wartość domyślna
            }
```

```
        }

        /// <summary>
        /// Obsługujemy zdarzenie malowania
        /// </summary>
        public override void Paint(Graphics aGraphics, Rectangle aArea)
        {
            using (Brush b = new SolidBrush(mColor1))
            {
                aGraphics.FillRectangle(b, aArea);  //Tło w kolorze właściwości
            }

            aGraphics.DrawRectangle(Pens.Red, aArea);  // Czerwona ramka

            // Obszar małego czarnego prostokąta zależy od stanu
            Rectangle swRect = Rectangle.Empty;
            if (switched)
            {
                swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.25d * aArea.Width),
                                       (int)Math.Round((double)aArea.Y + 0.25d * aArea.Heig
ht),
                                       (int)Math.Round((double)0.25d * aArea.Width),
                                       (int)Math.Round((double)0.25d * aArea.Height));
            }
            else
            {
                swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.5d * aArea.Width),
                                       (int)Math.Round((double)aArea.Y + 0.5d * aArea.Heigh
t),
                                       (int)Math.Round((double)0.25d * aArea.Width),
                                       (int)Math.Round((double)0.25d * aArea.Height));
            }

            aGraphics.FillRectangle(Brushes.Black, swRect); // Czarny prostokąt
```

```
        }

        /// <summary>
        /// Obsługujemy zdarzenie klikniecia myszą w obiekt
        /// </summary>
        public override void PerformMouseClick(System.Windows.Forms.MouseEventArgs e, Rectangle
aArea)
        {
            Rectangle swRect = Rectangle.Empty;  // Aktualny obszar małego czarnego prostokąta
            if (switched)
            {
                swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.25d * aArea.Width),
                                       (int)Math.Round((double)aArea.Y + 0.25d * aArea.Heig
ht),
                                       (int)Math.Round((double)0.25d * aArea.Width),
                                       (int)Math.Round((double)0.25d * aArea.Height));
            }
            else
            {
                swRect = new Rectangle((int)Math.Round((double)aArea.X + 0.5d * aArea.Width),
                                       (int)Math.Round((double)aArea.Y + 0.5d * aArea.Heigh
t),
                                       (int)Math.Round((double)0.25d * aArea.Width),
                                       (int)Math.Round((double)0.25d * aArea.Height));
            }

            if (swRect.Contains(e.Location))
            {  // Jeżeli kliknieto w obszar małego czarnego prostokąta to przełączamy stan
                switched = !switched;
                InvalidateObject();
            }
        }

        public override void PerformKeyDown(System.Windows.Forms.KeyEventArgs e)
        {
```

```
        if (e.KeyCode == System.Windows.Forms.Keys.S)
        {  // Jeżeli wciśnięto klawisz 'S' to przełączamy stan
           switched = !switched;
           InvalidateObject();
        }
    }
}
```

Asix.Evo – Diagramy mapowe GIS

1 Wstęp

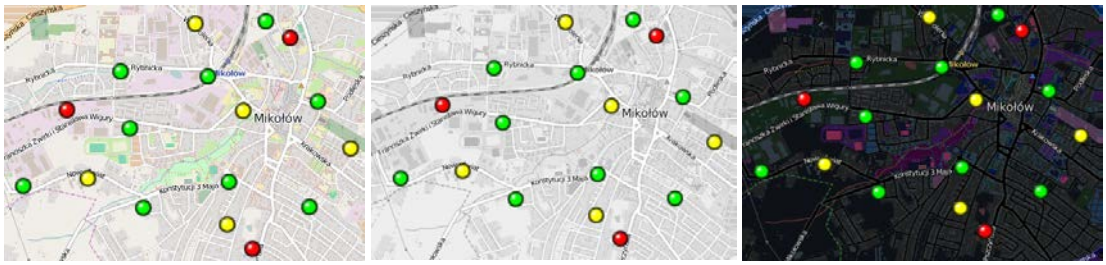
Zadaniem diagramów mapowych GIS jest wyświetlanie elementów na tle map geograficznych. Mapy pochodzą z serwisu OpenStreetMap i mogą być pobierane online z Internetu lub offline z lokalnego bufora uprzednio przygotowanego przez projektanta aplikacji. Na diagramach GIS mogą być nanoszone elementy wizualizacyjne w postaci: znaczników, linii (łamanych), obszarów, tekstów lub obrazków. Filozofia pracy z diagramami GIS odbiega nieco od tej znanej ze zwykłych diagramów. Przede wszystkim nacisk położono na warstwową organizację nanoszonych elementów. Zarówno projektant aplikacji jak i jej użytkownik (operator) mogą swobodnie kontrolować widoczność wybranych warstw diagramu. Istnieje również możliwość automatycznego wypełniania warstw elementami na podstawie danych geograficznych pozyskiwanych z baz MS SQL. Ten typ diagramów umożliwia ustawianie stopnia powiększenia widoku w szerokim zakresie.

1.1 Diagramy GIS

Diagramy GIS wykorzystują współrzędne geograficzne w systemie odniesienia WGS 84 (GPS) - a więc obszar roboczy jest ograniczony współrzędnymi z zakresu -180.0 do 180.0 stopni długości geograficznej (odpowiednik współrzędnych X) oraz -90.0 do 90.0 stopni szerokości geograficznej (odpowiednik współrzędnych Y), przy czym precyzja lokalizacji elementów wynika z precyzji liczb zmiennoprzecinkowych typu Double. Powiększenie diagramu (mapy) można zmieniać w zakresie od 2 do 25 w skali wykładniczej, ale istnieje możliwość narzucenia ograniczenia tych wartości do wskazanego zakresu. Skalowanie i pozycjonowanie widoku odbywa się za pomocą myszy lub można ustawić wskazane powiększenie i lokalizację za pomocą dedykowanych akcji operatorskich. Diagram umożliwia również skonfigurowanie kilku własnych przycisków wyświetlanych przy jednej z krawędzi i przypisanie im dowolnych akcji operatorskich.

1.2 Podkłady mapowe

Diagram może pracować w jednym z trzech trybów: bez mapy w tle, z mapą w tle pobieraną online z serwerów OSM (OpenStreetMap) lub z mapą w tle pobieraną z lokalnego bufora, w którym zawczasu projektant zapisał wybrane obszary dla określonych stopni powiększenia. Tryb bez mapy OSM w tle może być wykorzystany do projektowania diagramów, niekoniecznie geograficznych, o dużej rozpiętości skali powiększenia. Opcjonalnie można umieścić własne obrazy rastrowe lub wektorowe w jednej z warstw jako tło mapowe. Jeżeli korzystamy z map OSM online lub offline, możliwe jest ich wyświetlenie w trybie kolorowym, w odcieniach szarości lub w trybie negatywu.



Rysunek. Tryby kolorów mapy: kolorowa, w odcieniach szarości i negatyw.

1.2.1 Warstwy

Warstwy diagramów GIS funkcjonują inaczej niż warstwy diagramów zwykłych Asixa wersji Evo. W diagramach zwykłych warstwy nie stanowiły osobnych, definiowalnych bytów, a jedynie numer warstwy przypisany do obiektu decydował o położeniu tego obiektu względem innych obiektów (wzajemne przysłanianie). Diagramy GIS umożliwiają jawne zdefiniowanie warstw, ich nazwanie i opisanie. Możliwe jest ponadto uzależnienie widzialności warstwy GIS od dowolnych warunków lub od stopnia powiększenia diagramu. Do zarządzania widzialnością poszczególnych warstw przewidziano specjalne okienko w którym zarówno projektant aplikacji jak i operator (o ile mu pozwolimy) ma do dyspozycji listę wszystkich warstw. W trybie edycji projektant w tym okienku może wskazać aktualnie edytowaną, tzw. aktywną warstwę. Edycja odbywa się zawsze tylko w ramach aktywnej warstwy. Pozostałe warstwy (nieaktywne) mogą być wyświetlane, ale ich obiekty nie podlegają selekcji i edycji, w związku z czym nie przeszkadzają w edycji warstwy aktywnej.



Rysunek. Okienko warstw.

1.3 Elementy wizualizacyjne

Projektant aplikacji ma do dyspozycji pięć rodzajów obiektów wizualizacyjnych, które może umieścić na diagramach GIS:

1. Linia – skalowalny obiekt wierzchołkowy reprezentujący łamaną linię
2. Region – skalowalny obiekt wierzchołkowy reprezentujący obszar
3. Tekst – skalowalny obiekt wyświetlający statyczne lub dynamiczne napisy
4. Znacznik – obiekt punktowy o rozmiarze niezależnym od skali
5. Obraz – skalowalny obiekt prostokątny służący do wyświetlania dowolnego obrazu rastrowego lub wektorowego

Obiekty mogą mieć ustawione zakresy widzialności względem powiększenia diagramu, komunikat wyświetlany w dymku oraz można im przypisać akcje wykonywane po ich kliknięciu lewym lub prawym klawiszem myszy.

Obiekty typu Region, Linia i Tekst mogą posiadać indywidualne ustawienia sposobu wyświetlania i zachowania lub pobierać te ustawienia ze zdefiniowanych w aplikacji wzorców. Wzorce GIS pełnią rolę definiowanych przez projektanta zestawów właściwości i sposobów interakcji obiektów. Są przydatne, gdy chcemy wiele obiektów skonfigurować w podobny, powtarzalny sposób. Dzięki nim, gdy zachodzi potrzeba zmiany cech wielu obiektów, nie trzeba tego konfigurować wielokrotnie dla każdego obiektu, wystarczy odpowiednie zmiany wprowadzić we wzorcu. Opcjonalnie wzorce mogą posiadać parametry.

Obiekt typu Znacznik domyślnie wyświetlany jest jako kropka o określonym rozmiarze i kolorze, ale można mu nadać dowolny wygląd zdefiniowany przez projektanta aplikacji. Znaczniki, podobnie jak Linie i Regiony, również mogą korzystać ze wzorców, z tym że dodatkowo taki wzorzec Znacznika umożliwia jego graficzne zaprojektowanie. Zdefiniowane i nazwane wzorce Znaczników również mogą posiadać dowolne parametry.

1.4 Współpraca z geograficznymi bazami MS SQL lub plikami Shapefile

Alternatywnie do ręcznego rozmieszczania obiektów na warstwach diagramu GIS istnieje możliwość automatycznego wypełniania warstw obiektami na podstawie zawartości geograficznej bazy danych MS SQL lub pliku ESRI Shapefile. W tym celu w aplikacji Asix.Evo można zdefiniować tzw. *Źródła danych GIS* skojarzone z konkretną bazą danych i tabelą zawierającą kolumnę typu GEOMETRY lub plikiem Shapefile. Dane ze źródła będą mapowane na odpowiednie obiekty w zależności od typu geometrii. Baza danych geograficznych obiektów przygotowywana jest ręcznie według standardów definiujących obiekty. Źródło danych może wspierać geometrię w postaci punktów (POINT) - mapowanych na Znaczniki lub teksty, linii (LINESTRING) - mapowanych na obiekty typu Linia i Wielokątów (POLYGON) – mapowanych na obiekty typu Region. Pojedyncze źródło danych może obsługiwać tylko obiekty jednego typu. W źródle danych GIS można zdefiniować domyślny wzorzec Linii, Regionu, Tekstu lub Znacznika, używany do wizualizacji rekordów z tabeli lub można wskazać kolumnę tabeli, która będzie zawierała indywidualne nazwy wzorców dla każdego rekordu. Ponadto źródło danych może pobierać z

bazy dowolne inne niż geometria atrybuty, które będą automatycznie przypisywane do parametrów obiektów (np. teksty, opisy).

2 Praca z diagramami mapowymi GIS

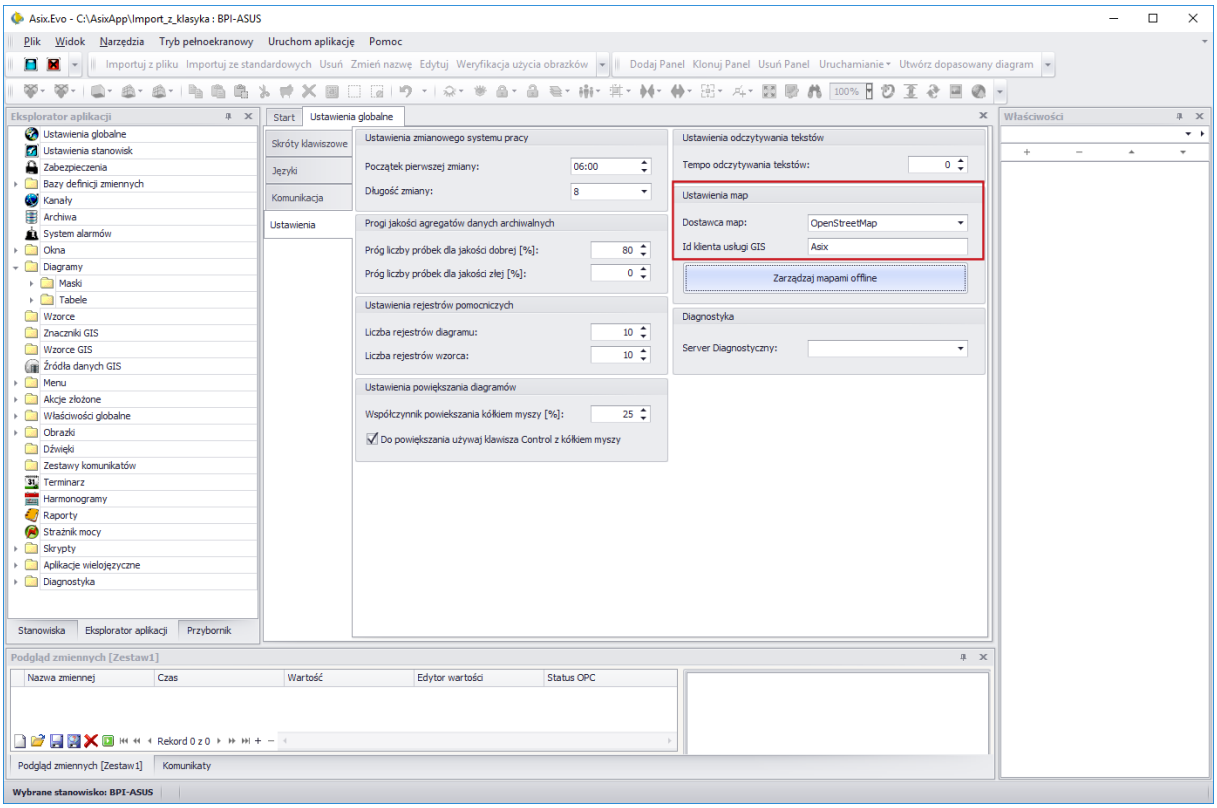
2.1 Konfiguracja źródła mapy

Przed przystąpieniem do projektowania diagramów GIS należy w ustawieniach globalnych aplikacji zadeklarować dostawcę map oraz identyfikator klienta usługi GIS.

Asix.Evo.exe > ***Ustawienia globalne*** > ***Ustawienia*** > ***Ustawienia map***:

Dostawca map – źródło map wykorzystywanych przez diagramy GIS; domyślnie diagramy pracują na darmowej i ogólnodostępnej mapie OpenStreetView;

Id Klienta usługi GIS – identyfikator klienta wymagany przez serwery map do pobrania mapy;

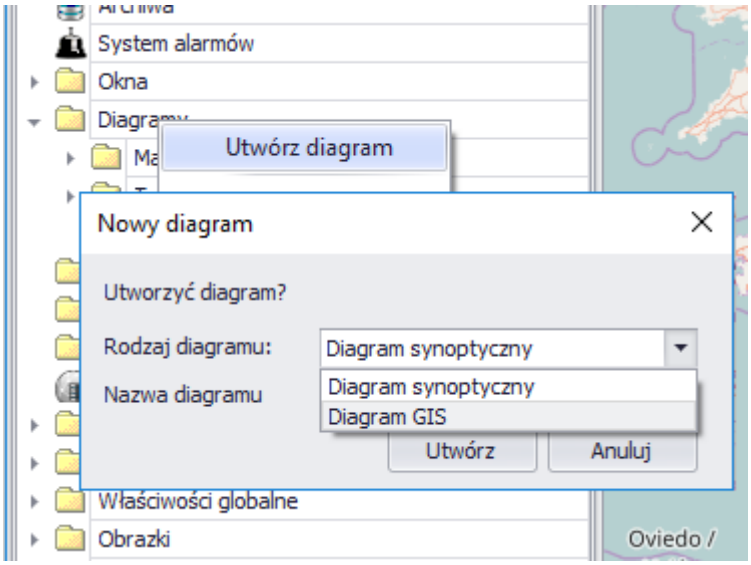


Rysunek. Deklaracja źródła map.

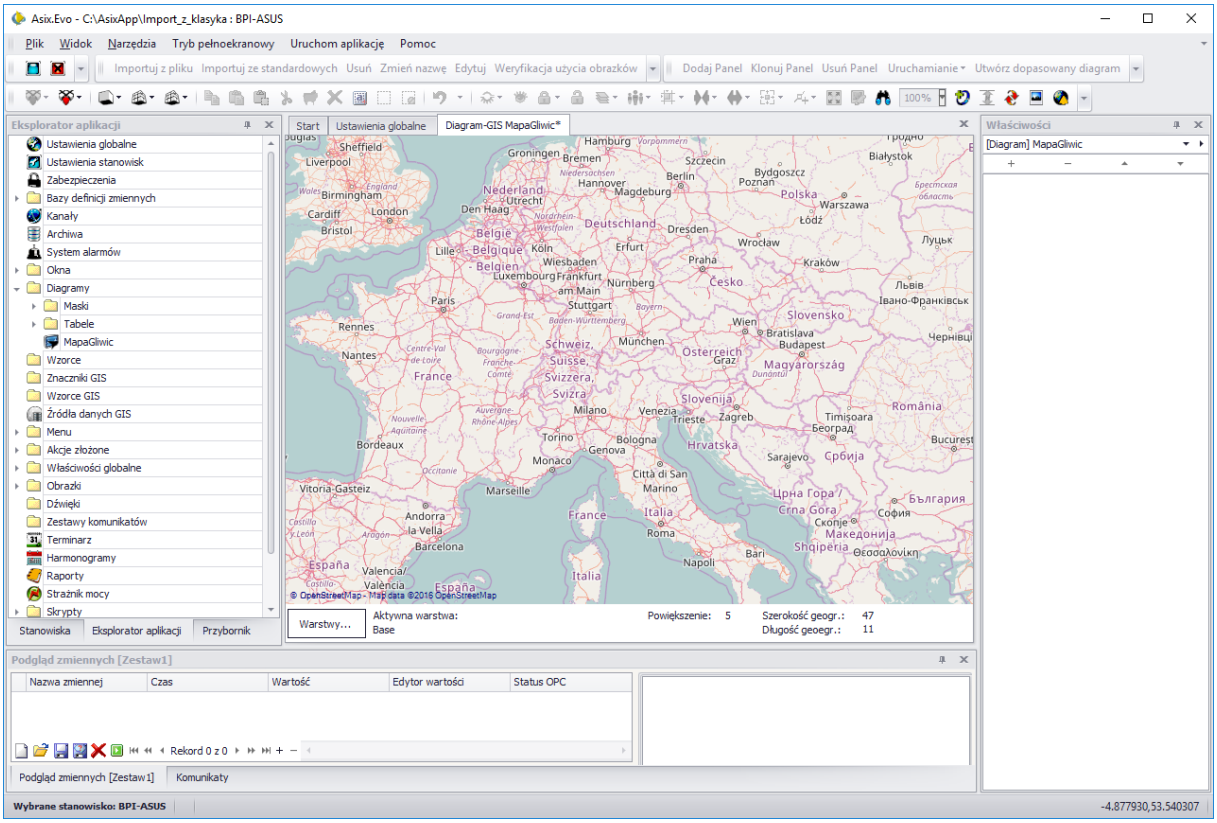
2.2 Tworzenie diagramu

Aby utworzyć diagram mapowy GIS, należy w eksploratorze aplikacji kliknąć prawym klawiszem myszy na pozycji *Diagramy* i z menu podręcznego wybrać polecenie *Utwórz diagram*.

Na nowym diagramie domyślnie załadowana jest mapa on-line.



Rysunek. Tworzenie diagramu GIS.



Rysunek. Diagram GIS z domyślną mapą OpenStreetView.

2.3 Definiowanie właściwości podstawowych diagramu

Właściwości podstawowe diagramu mapowego GIS:

Tryb mapy – offline lub online; wartość domyślna: online; zastosowanie map offline wymaga pobrania ich przez projektanta za pomocą okna *'Zarządzanie mapami offline'* (patrz: rozdział *2.14. Deklarowanie map offline*).

Tryb wyświetlania – zwykły, skala szarości lub negatyw;

Minimalne powiększenie / ***Maksymalne powiększenie*** – umożliwia ograniczenie możliwych stopni powiększeani diagramu/mapy;

Domyślne powiększenie – początkowy stopień powiększenia w skali od 2 do 25 ustawiany przy otwieraniu diagramu (w zakresie 2-25 każde kolejne powiększenie powoduje dwukrotne powiększenie (x2) widocznych na mapie elementów i jednocześnie dwukrotne zmniejszenie (÷2) wielkości (długości i szerokości) widocznego kawałka mapy);

Min. długość georgraficzna / ***Max. długość geograficzna*** / ***Min. szerokość geograficzna*** / ***Max. szerokość geograficzna*** - parametry określają zakres możliwych do wyświetlenia na diagramie współrzędnych geograficznych;

Domyślna długość geograficzna / ***Domyślna szerokość geograficzna*** – domyślne współrzędne geograficzne środka wyświetlanego obszaru mapy ustawiane przy otwieraniu diagramu – domyślne wartości te ustawione są na: 47^0N, 11^0N (47^0 szerokości geograficznej północnej i 11^0 długości geograficznej wschodniej);

Pokazuj środek mapy – załącza znacznik (w kształcie krzyżyka) środka mapy; domyślnie: nie; w trakcie powiększania/pomniejszania obszaru mapy punkt, w którym zostaje umieszczony kursor, staje się automatycznie środkiem powiększonego/pomniejszonego obszaru mapy;

Kolor środka mapy – definiuje kolor znacznika środka mapy; dostępne są predefiniowane kolory lub możliwość pobrania próbnikiem dowolnego koloru z ekranu monitora;

Pokazuj pasek narzędziowy – opcja decyduje o tym, czy operator będzie widział pasek narzędziowy na diagramie; domyślnie pasek jest dostępny dla operatora; niezależnie od ustawienia parametru pasek jest zawsze widoczny w trybie projektowania aplikacji; pasek narzędziowy obejmuje:

- przycisk ***Warstwy***, który rozwija okno ze zdefiniowanymi warstwami;
- informację na temat stopnia powiększenia mapy;
- szerokość i długość geograficzną środka mapy;

Kolor tła – kolor tła diagramu;

Położenie przycisków – określa położenie przycisków na mapie: brak – brak przycisków, do góry, do dołu, do lewej, do prawej;

Szerokość/wysokość przycisków – definiuje wielkość przycisków (w pikselach) na mapie;

Kolor przycisków – definiuje kolor przycisków na mapie;

Kolor tekstu przycisków – definiuje kolor tekstu wyświetlanego na przyciskach na mapie;

Czcionka przycisków – definiuje krój czcionki tekstu wyświetlanego na przyciskach na mapie.

Rozmiar czcionki przycisków – definiuje rozmiar czcionki tekstu wyświetlanego na przyciskach na mapie;

Styl czcionki przycisków – definiuje styl (zwykły, pogrubienie, kursywa, podkreślenie, przekreślenie) czcionki tekstu wyświetlanego na przyciskach na mapie;

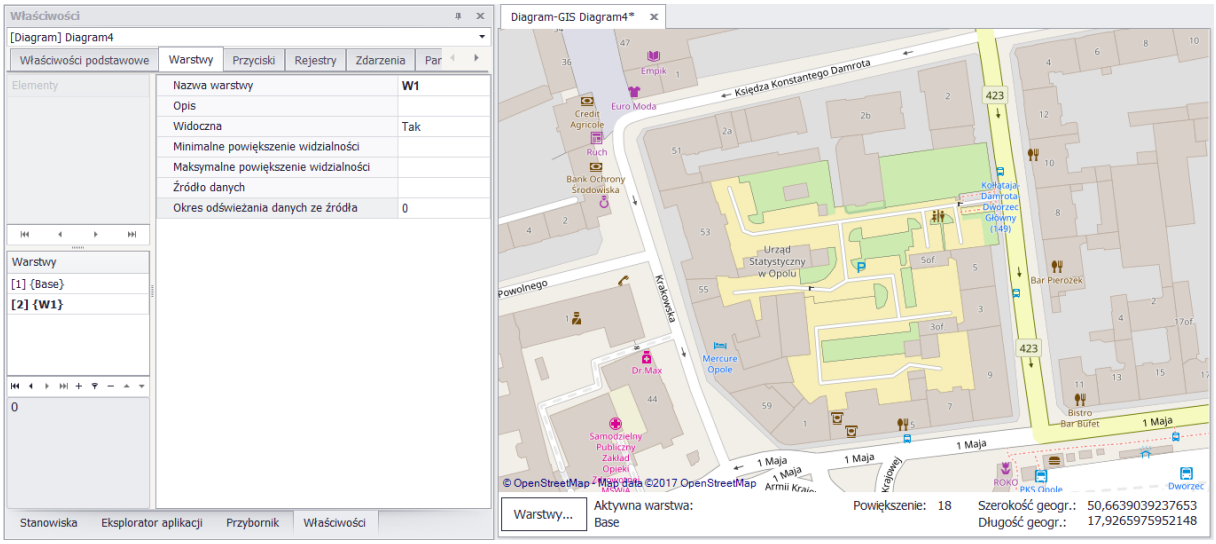
Przejrzystość przycisków – definiuje stopień transparentności przycisków – w skali od 0 do 1, gdzie 0 oznacza całkowitą przeźroczystość a 1 całkowite krycie.

2.4 Tworzenie i parametryzacja warstwy

Istnieje możliwość zdefiniowania dowolnej liczby warstw na diagramie.

Z chwilą utworzenia diagramu GIS automatycznie tworzona jest jedna warstwa o nazwie ‘Base’. Kolejne warstwy tworzone są za pomocą przycisku ‘+’ w:

okno właściwości diagramu > zakładka ***Warstwy*** > panel ***Warstwy***



Rysunek. Dodawanie nowej warstwy.

Aby sparametryzować warstwę, należy w oknie właściwości diagramu, na zakładce ***Warstwy***, w panelu ***Warstwy*** zaznaczyć wybraną warstwę i ustawić następujące właściwości:

Nazwa warstwy;

Opis;

Widoczna – parametr steruje widzialnością warstwy na diagramie;

Minimalne powiększenie widzialności – określa stopień powiększenia, przy którym warstwa (jej elementy) staje się widoczna na diagramie;

Maksymalne powiększenie widzialności – określa maksymalny stopień powiększenia, przy którym warstwa (jej elementy) jest jeszcze widzialna;

Źródło danych – umożliwia zdeklarowanie źródła danych GIS w postaci bazy MS SQL lub pliku Shapefile; opcja spowoduje załadowanie na warstwę obiektów określonych w źródle danych GIS; obiekty będą nanoszone na warstwę z źródłowej bazy dynamicznie, w zależności od obszaru wyświetlanego w danym momencie na diagramie GIS;

Istnieje możliwość zarządzania widocznością warstwy za pomocą okna wywoływanego z paska narzędziowego na diagramie (patrz: *2.6. Zarządzanie widocznością warstw / pasek narzędziowy*).

Po zdefiniowaniu i wybraniu warstwy można przystąpić do definiowania na niej elementów wizualizacyjnych takich jak: Linia, Region, Obrazek, Tekst, Znacznik.

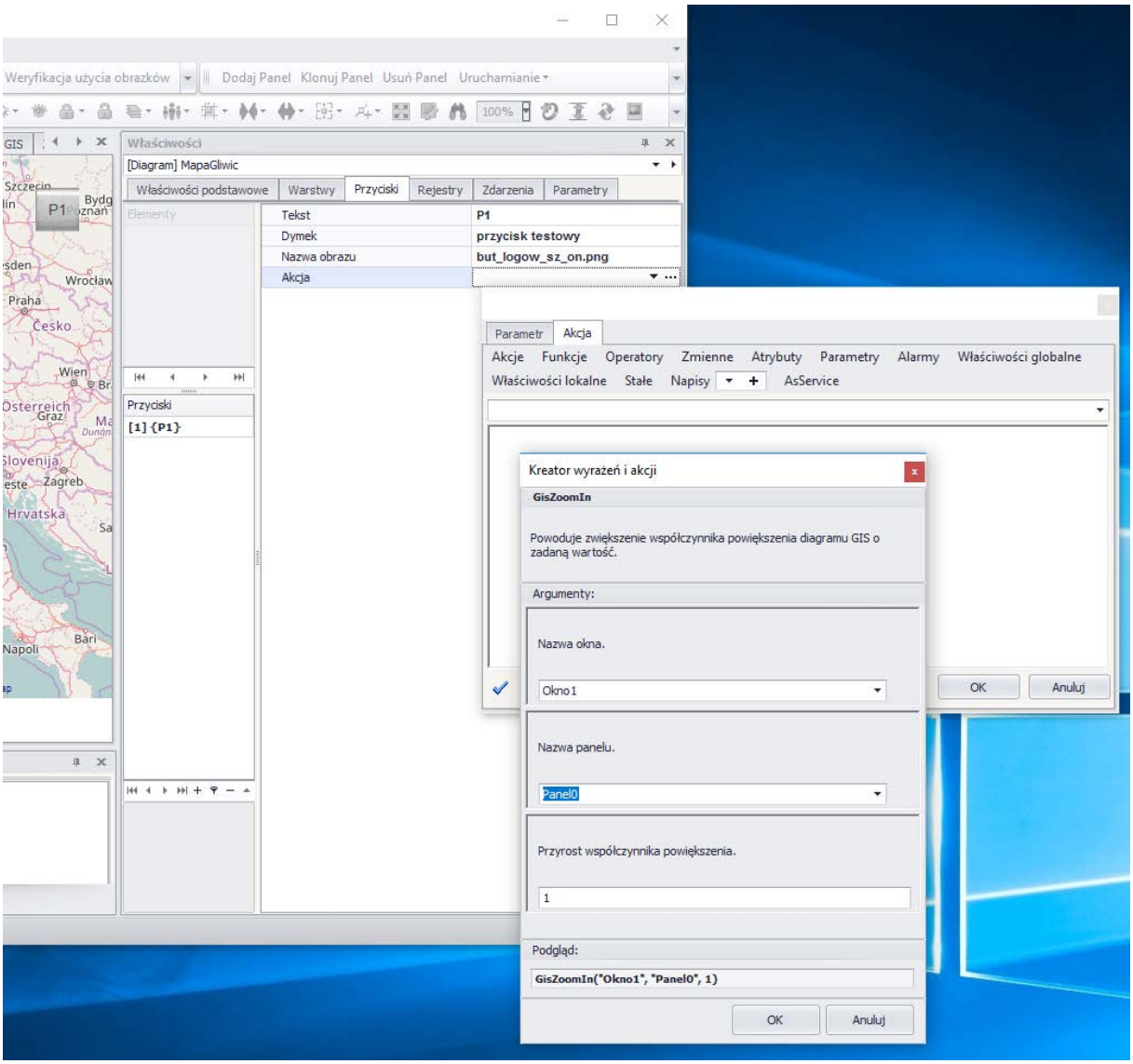
Okres odświeżania danych ze źródła – określa czas w sekundach automatycznej aktualizacji danych pochodzących ze źródła danych GIS;

2.5 Tworzenie i parametryzacja przycisków

Przyciski stanowią element diagramu GIS i realizują przypisane im akcje operatorskie.

Przyciski tworzone są za pomocą przycisku ‘+’ w:

okno właściwości diagramu > zakładka ***Przyciski*** > panel ***Przyciski***



Rysunek. Dodawanie przycisku.

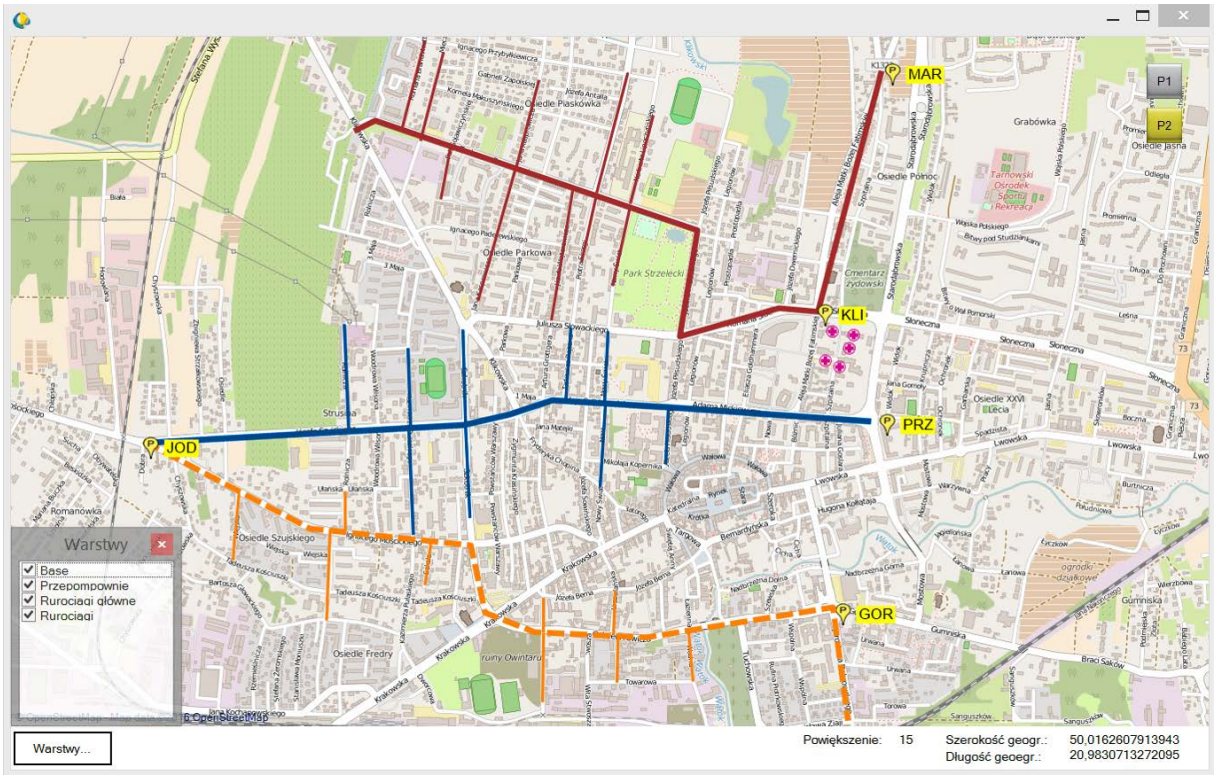
Aby sparametryzować przycisk, należy w oknie właściwości diagramu, na zakładce *Przyciski*, w panelu *Przyciski* zaznaczyć wybrany przycisk i ustawić dla niego następujące właściwości:

Tekst – określa tekst wyświetlany na przycisku;

Dymek - deklaruje dymek, który pojawi się w trybie wykonywania aplikacji po najechaniu myszą na przycisk;

Nazwa obrazu – deklaruje obraz, który zostanie załadowany jako tło przycisku;

Akcja – przypisuje akcję; poza standardowymi akcjami istnieje również grupa akcji typowo dedykowanych diagramom GIS – są to akcje, których nazwa zaczyna się od Gis….



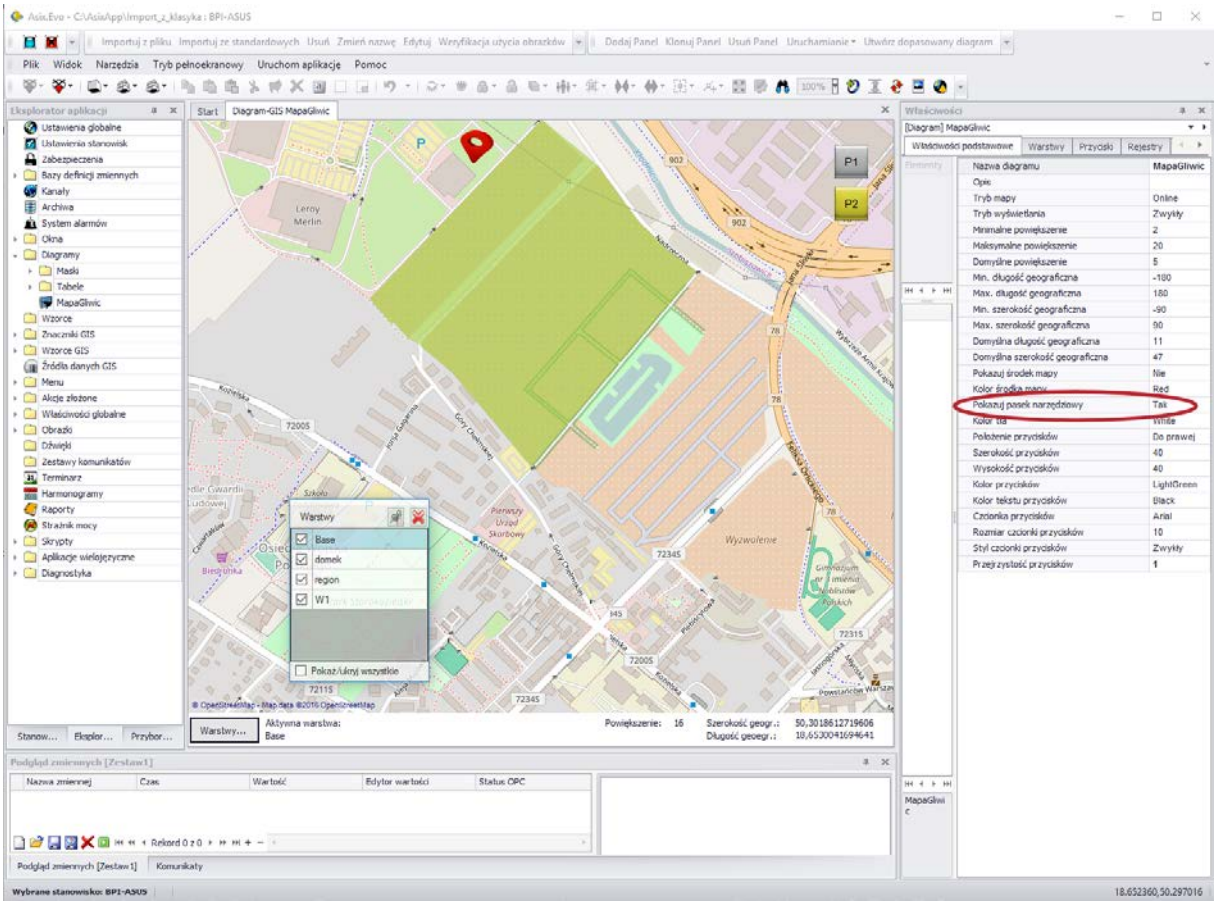
Rysunek. Przykład diagramu z Przyciskami.

2.6 Zarządzanie widocznością warstw / pasek narzędziowy

Pasek narzędziowy wyświetlany jest w dolnej części diagramu GIS. W trybie edycyjnym aplikacji jest on zawsze widoczny, natomiast w trybie realizacji aplikacji operator ma do dyspozycji pasek narzędziowy tylko wtedy, gdy projektant udostępni go za pomocą właściwości ***Pokazuj pasek narzędziowy*** we właściwościach podstawowych diagramu.

Pasek narzędziowy zawiera: przycisk ***Warstwy***, pole z informacją na temat aktywnej warstwy i pola wyświetlające stopień powiększenia oraz współrzędne geograficzne środka wyświetlanego obszaru mapy.

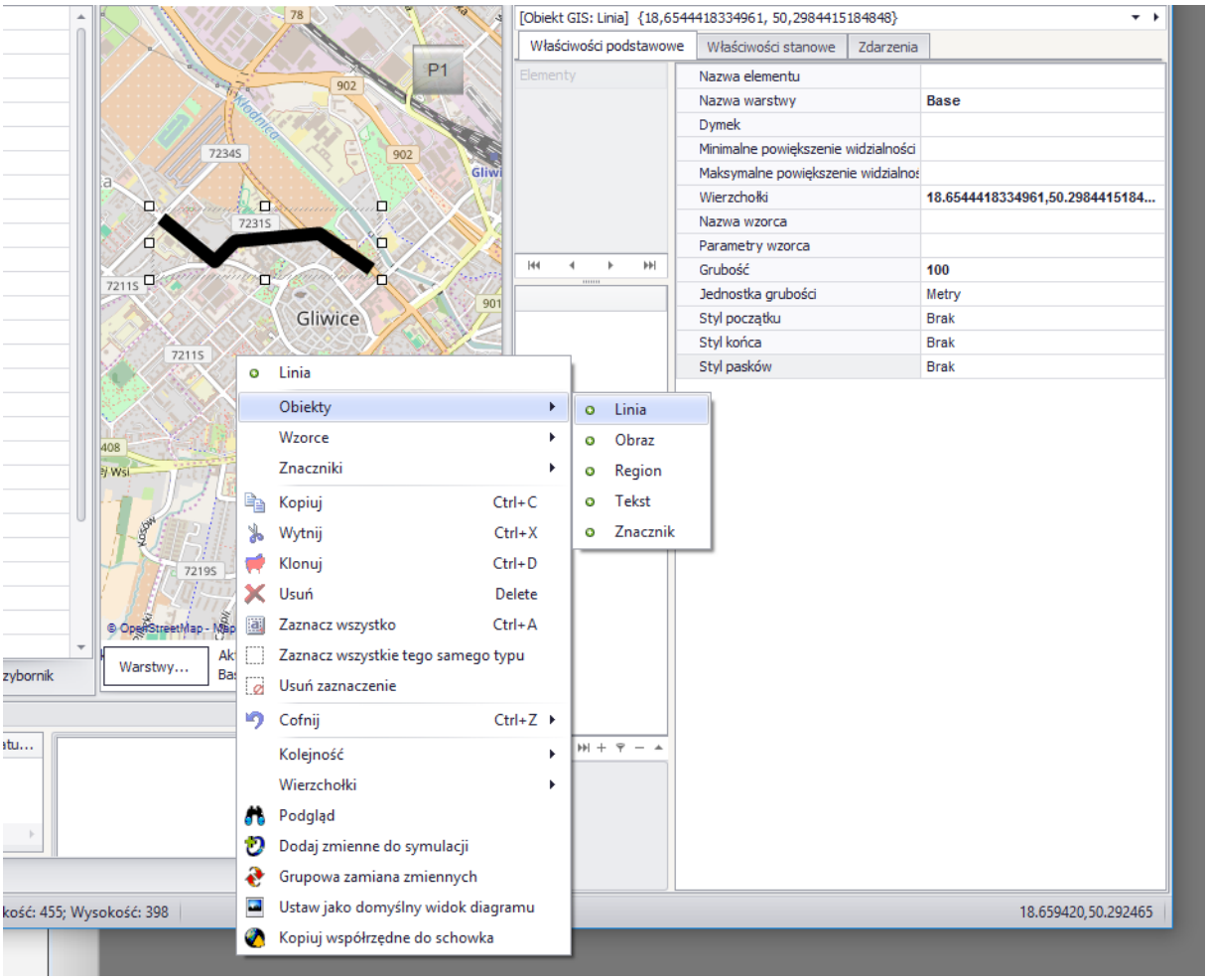
Przycisk ***Warstwy*** otwiera okno z listą zdefiniowanych warstw, na której można odznaczyć/zaznaczyć wybrane warstwy – spowoduje to ich ukrycie/wyświetlenie. Przycisk  blokuje ukrywanie okna w momencie, gdy jest ono nieaktywne.



Rysunek. Widok diagramu z paskiem narzędziowym.

2.7 Tworzenie i parametryzacja obiektu Linia

Obiekt Linia tworzony jest za pomocą polecenia ***Obiekty / Linia*** wywoływanego z menu podręcznego, rozwijanego po kliknięciu prawym klawiszem myszy na powierzchni diagramu GIS.



Rysunek. Tworzenie obiektu Linia.

Aby sparametryzować obiekt Linia, należy zaznaczyć obiekt Linia i w oknie właściwości obiektu określić następujące parametry:

Nazwa warstwy – nazwa warstwy, na której znajduje się obiekt;

Minimalne powiększenie widzialności / ***Maksymalne powiększenie widzialności*** – określa minimalny i maksymalny stopień powiększenia obszaru diagramu, od/do którego obiekt jest widzialny;

Wierzchołki – określa współrzędne geograficzne wierzchołków Linii; właściwość umożliwia dodawanie i odejmowanie wierzchołków Linii; (odejmowanie możliwe po kliknięciu na wiersz z współrzędnymi wierzchołków prawym klawiszem myszy);

Nazwa wzorca – umożliwia zdeklarowanie nazwy wzorca GIS, który zawiera zdefiniowane wybrane właściwości Linii;

Parametry wzorca – umożliwia zadeklarowanie wartości parametrom zdefiniowanym we wzorcu Linii;

Grubość – określa grubość Linii w jednostkach zdefiniowanych we właściwości *Jednostka grubości*;

Jednostka grubości – określa jednostkę, w której określana będzie grubość Linii; mogą to być metry lub piksele; gdy jednostką są metry, grubość Linii zwiększa się w zależności od stopnia powiększenia obszaru diagramu; gdy jednostką są piksele, grubość Linii pozostaje stała niezależnie od stopnia powiększenia powierzchni diagramu;

Styl początku – określa styl początku Linii; dostępne warianty: 'Brak', 'Płaski', 'Okrągły', 'Trójkątny', 'Strzałka';

Styl końca – określa styl końca linii; dostępne warianty: 'Brak', 'Płaski', 'Okrągły', 'Trójkątny', 'Strzałka';

Styl pasków – określa styl linii; dostępne warianty: 'Brak', 'Kreska', 'Kropka', 'Kreska-Kropka', 'Kreska-Kropka-Kropka'.

***

Istnieje możliwość zarządzania wierzchołkami Linii. Po wybraniu prawym klawiszem myszy obiektu Linia, w menu podręcznym dostępny jest zestaw poleceń ***Wierzchołki***:

- ***Edytuj wierzchołki***;

- ***Wstaw wierzchołek***;

- ***Usuń wierzchołek***;

- ***Zakończ edycję wierzchołków***;

- ***Transformacje wierzchołków***: 'Odbicie wierzchołków w poziomie', 'Odbicie wierzchołków w pionie', 'Obrót wierzchołków'.

Na zakładce *Właściwości stanowe* jest możliwość określenia następujących właściwości w ramach poszczególnych stanów obiektu:

- ***Opis stanu***;

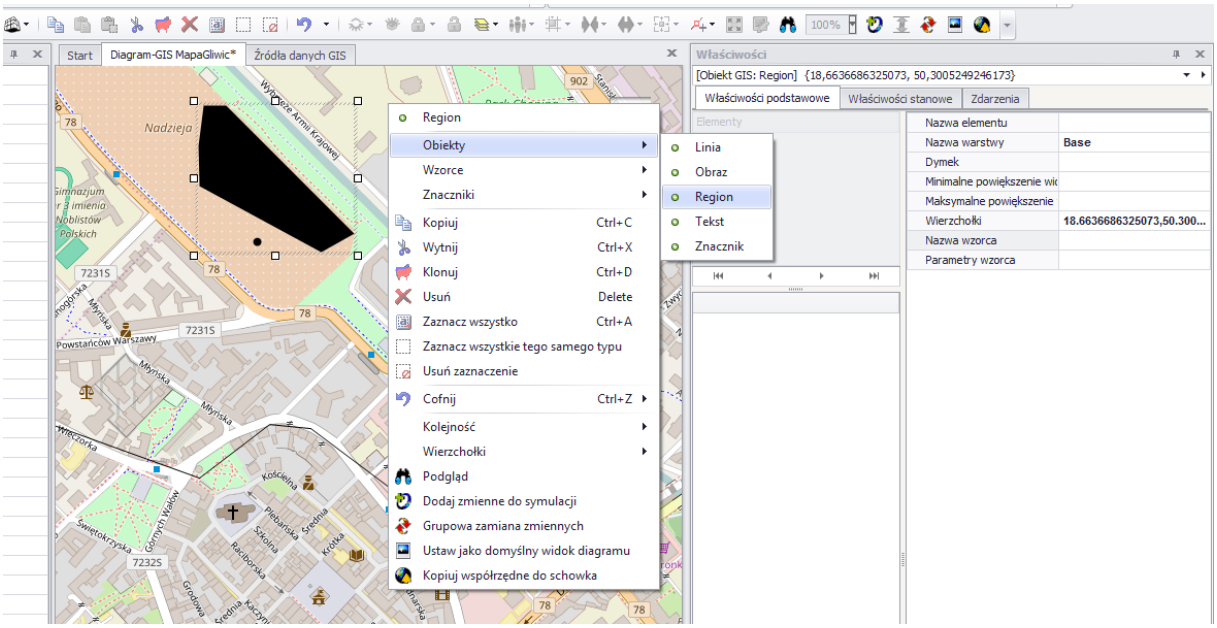
- ***Widoczny*** – deklarowanie widoczności obiektu;

- ***Kolor*** – kolor obiektu;

- ***Kolor pasków*** – kolor przerw pomiędzy paskami/kropkami w Linii, gdy Linia ma zdefiniowany styl z kreskami i/lub kropkami ('Kreska', 'Kropka', 'Kreska-Kropka', 'Kreska-Kropka-Kropka').

2.8 Tworzenie i parametryzacja obiektu Region

Obiekt Region tworzony jest za pomocą polecenia ***Obiekty/Region*** wywoływanego z menu podręcznego, rozwijanego po kliknięciu prawym klawiszem myszy na powierzchni diagramu GIS.



Rysunek. Tworzenie obiektu Region.

Aby sparametryzować obiekt Region, należy zaznaczyć obiekt Region i w oknie właściwości obiektu określić następujące parametry:

Nazwa warstwy – nazwa warstwy, na której znajduje się obiekt;

Minimalne powiększenie widzialności / ***Maksymalne powiększenie widzialności*** – określa minimalny i maksymalny stopień powiększenia obszaru diagramu, od/do którego obiekt jest widzialny.

Wierzchołki – określa współrzędne geograficzne wierzchołków Regionu; właściwość umożliwia dodawanie i odejmowanie wierzchołków Regionu; (odejmowanie możliwe po kliknięciu na wiersz z współrzędnymi wierzchołków prawym klawiszem myszy);

Nazwa wzorca – umożliwia zdeklarowanie nazwy wzorca GIS, który zawiera zdefiniowane wybrane właściwości Regionu;

Parametry wzorca - umożliwia zadeklarowanie wartości parametrom zdefiniowanym we wzorcu Regionu;

Na zakładce *Właściwości stanowe* jest możliwość określenia następujących właściwości w ramach poszczególnych stanów obiektu:

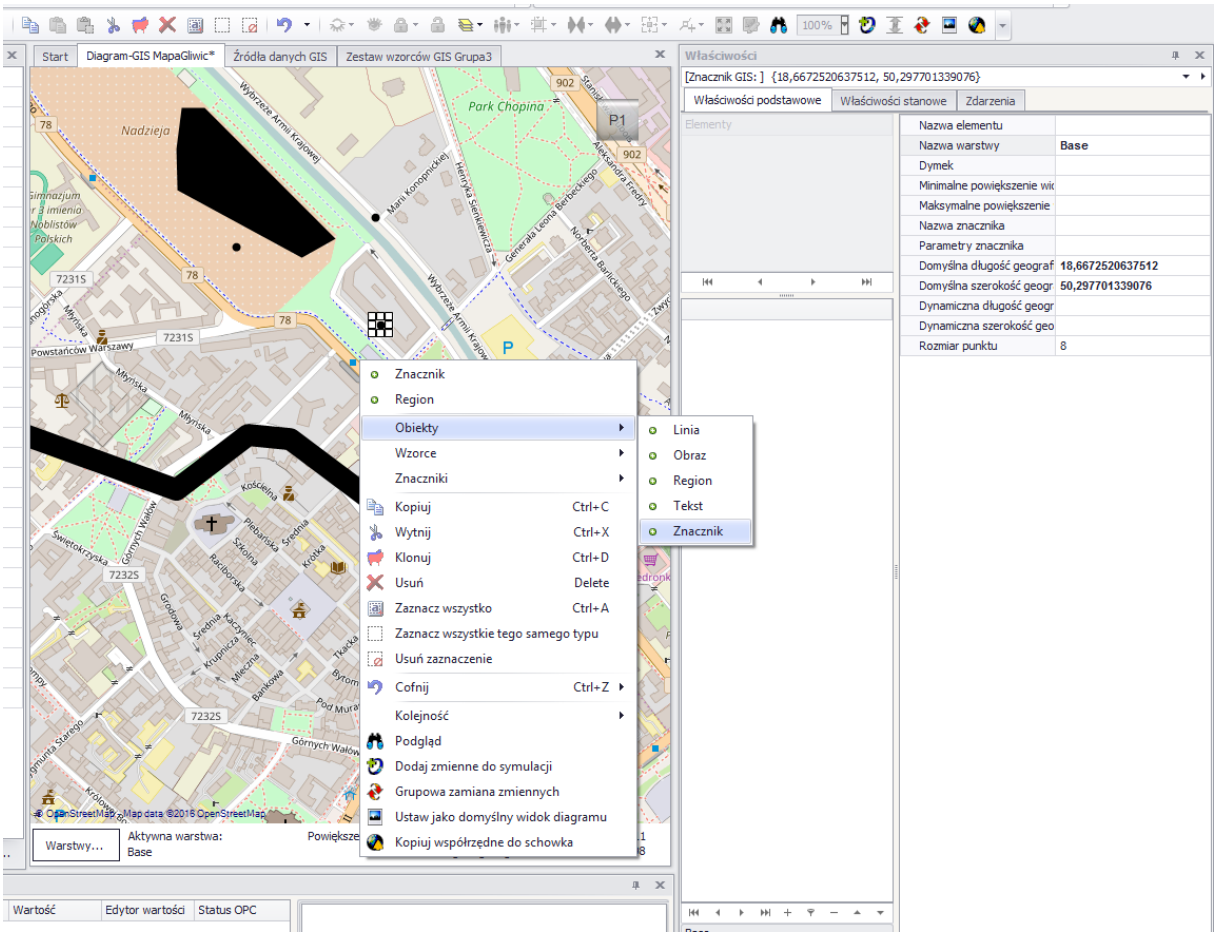
- ***Opis stanu***;

- ***Kolor*** – kolor obiektu;

- ***Przezroczystość*** – określa stopień przezroczystości obiektu;

2.9 Tworzenie i parametryzacja obiektu Znacznik

Obiekt Znacznik tworzony jest za pomocą polecenia ***Obiekt/Znacznik*** wywoływanego z menu podręcznego, rozwijanego po kliknięciu prawym klawiszem myszy na powierzchni diagramu GIS.



Rys. Tworzenie obiektu Znacznik.

Aby sparametryzować obiekt Znacznik, należy zaznaczyć obiekt Znacznik i w oknie właściwości obiektu określić następujące parametry:

Nazwa warstwy – nazwa warstwy, na której znajduje się obiekt;

Minimalne powiększenie widzialności / ***Maksymalne powiększenie widzialności*** – określa minimalny i maksymalny stopień powiększenia obszaru diagramu, od/do którego obiekt jest widzialny.

Nazwa znacznika – umożliwia zdeklarowanie nazwy Znacznika GIS, indywidualnie utworzonego przez projektanta; Znaczniki utworzone przez projektanta dostępne są w eksploratorze aplikacji, węźle *Znaczniki GIS*.

Parametry znacznika - umożliwia zadeklarowanie wartości parametrom zdefiniowanym w Znaczniku utworzonym indywidualnie przez projektanta;

Domyślna długość geograficzna / ***Domyślna szerokość geograficzna*** – domyślne współrzędne geograficzne środka obiektu Znacznik, odpowiadające położeniu Znacznika w momencie jego umieszczenia na diagramie;

Dynamiczna długość geograficzna / ***Dynamiczna szerokość geograficzna*** – właściwości umożliwiają podpięcie zmiennych procesowych, dzięki którym przy zmianie wartości tych zmiennych obiekt Znacznik będzie zmieniał swoje położenie;

Rozmiar punktu – wielkość określona w pikselach oznacza stały rozmiar Znacznika, który nie zmienia się bez względu na stopień powiększenia; domyślnie: 8 px;

Kształt Znacznika to czarna kropka. Jeśli użytkownik chce mieć inny Znacznik, ma możliwość zdefiniowania swoich własnych Znaczników (patrz: *2.12. Tworzenie wzorców / Znaczników GIS*).

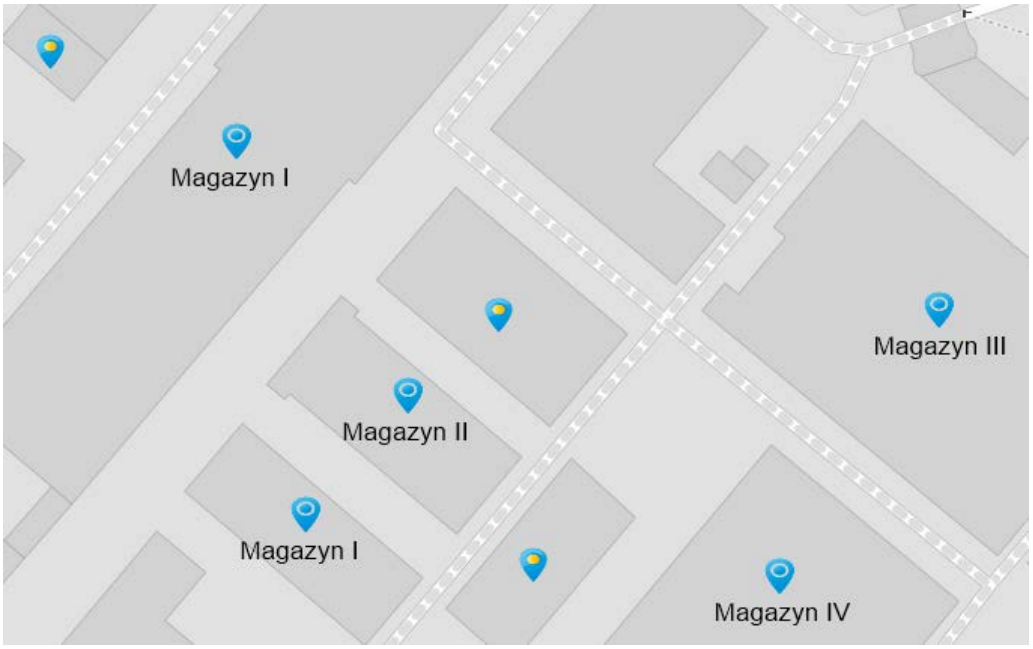
Na zakładce *Właściwości stanowe* jest możliwość określenia następujących właściwości w ramach poszczególnych stanów obiektu:

- ***Opis stanu***;

- ***Widoczny*** – deklarowanie widoczności obiektu;

- ***Kolor*** – kolor obiektu;

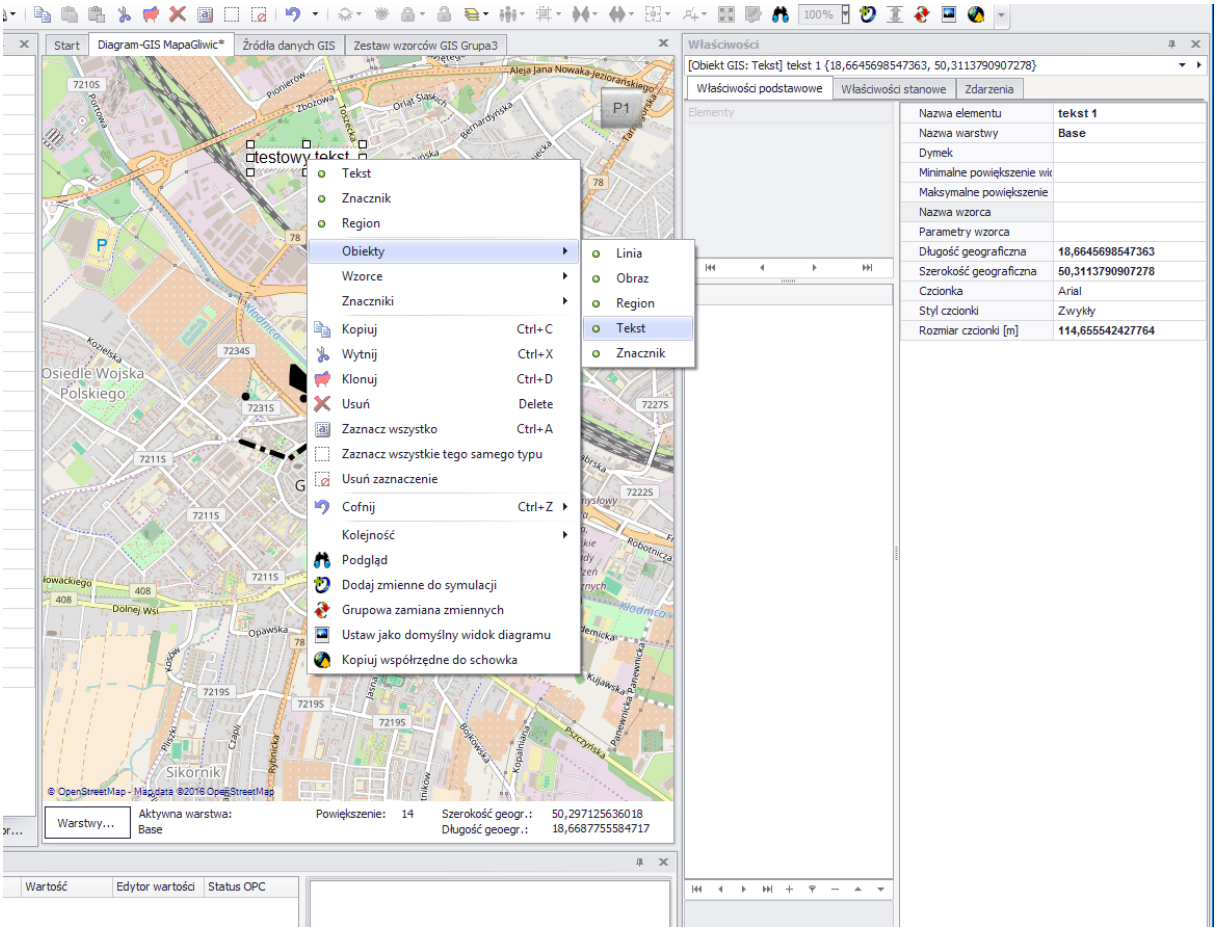
- ***Przezroczystość*** – określa stopień przezroczystości obiektu;



Rysunek. Znaczniki na mapie.

2.10 Tworzenie i parametryzacja obiektu Tekst

Obiekt Tekst tworzony jest za pomocą polecenia ***Obiekty/Tekst*** wywoływanego z menu podręcznego, rozwijanego po kliknięciu prawym klawiszem myszy na powierzchni diagramu GIS.



Rys. Tworzenie obiektu Tekst.

Nazwa warstwy – nazwa warstwy, na której znajduje się obiekt;

Minimalne powiększenie widzialności / ***Maksymalne powiększenie widzialności*** – określa minimalny i maksymalny stopień powiększenia obszaru diagramu, od/do którego obiekt jest widzialny.

Nazwa wzorca – umożliwia zdeklarowanie nazwy wzorca GIS, który zawiera zdefiniowane wybrane właściwości Tekstu;

Parametry wzorca - umożliwia zadeklarowanie wartości parametrom zdefiniowanym we wzorcu;

Długość geograficzna / ***Szerokość geograficzna*** – współrzędne geograficzne środka obiektu Tekst, odpowiadające położeniu Tekstu w momencie jego umieszczenia na diagramie;

Czcionka – definiuje krój czcionki tekstu wyświetlanego w obiekcie Tekst;

Styl czcionki – definiuje styl (zwykły, pogrubienie, kursywa, podkreślenie, przekreślenie) czcionki tekstu wyświetlanego w obiekcie Tekst;

Rozmiar czcionki – wielkość określona w metrach oznacza dynamiczny rozmiar, który zależy od stopnia powiększenia obszaru diagramu.

Tekst obiektu deklarowany jest na zakładce *Właściwości stanowe* okna właściwości obiektu. Na zakładce *Właściwości stanowe* jest możliwość określenia następujących właściwości w ramach poszczególnych stanów obiektu:

- ***Opis stanu***;

- ***Widoczny*** – deklarowanie widoczności obiektu;

- ***Tekst*** – tekst wyświetlany w obiekcie;

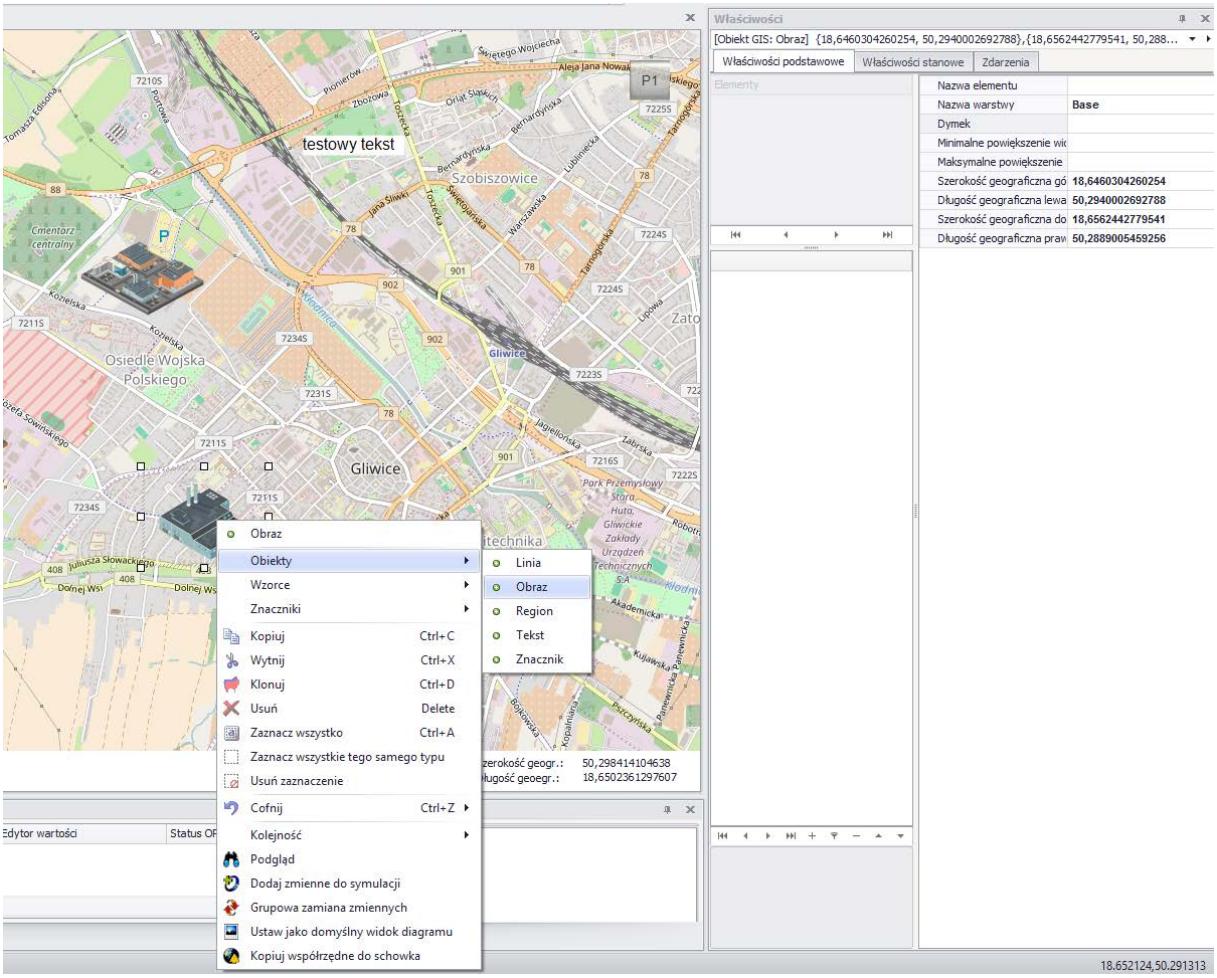
- ***Kolor*** – kolor tekstu;

- ***Kolor tła*** – kolor tła obszaru Tekstu;

- ***Kolor przekreślenia*** – obiekt jest przekreślony; istnieje możliwość ustawiania koloru tego przekreślenia; kolor 'Transparentny' pozwala ukryć przekreślenie.

2.11 Tworzenie i parametryzacja obiektu Obraz

Obiekt Obraz tworzony jest za pomocą polecenia ***Obiekty/Obraz*** wywoływanego z menu podręcznego, rozwijanego po kliknięciu prawym klawiszem myszy na powierzchni diagramu GIS.



Rys. Tworzenie obiektu Obraz.

Nazwa warstwy – nazwa warstwy, na której znajduje się obiekt;

Minimalne powiększenie widzialności / ***Maksymalne powiększenie widzialności*** – określa minimalny i maksymalny stopień powiększenia obszaru diagramu, od/do którego obiekt jest widzialny.

Szerokość geograficzna góry / ***Długość geograficzna lewa*** – współrzędne geograficzne lewego górnego narożnika pola Obrazu;

Szerokość geograficzna dołu / ***Długość geograficzna prawa*** – współrzędne geograficzne prawego dolnego narożnika pola Obrazu.

Obrazek obiektu deklarowany jest na zakładce *Właściwości stanowe* okna właściwości obiektu. Na zakładce *Właściwości stanowe* jest możliwość określenia następujących właściwości w ramach poszczególnych stanów obiektu:

- ***Opis stanu***;

- ***Widoczny*** – umożliwia deklarowanie widoczności obiektu;

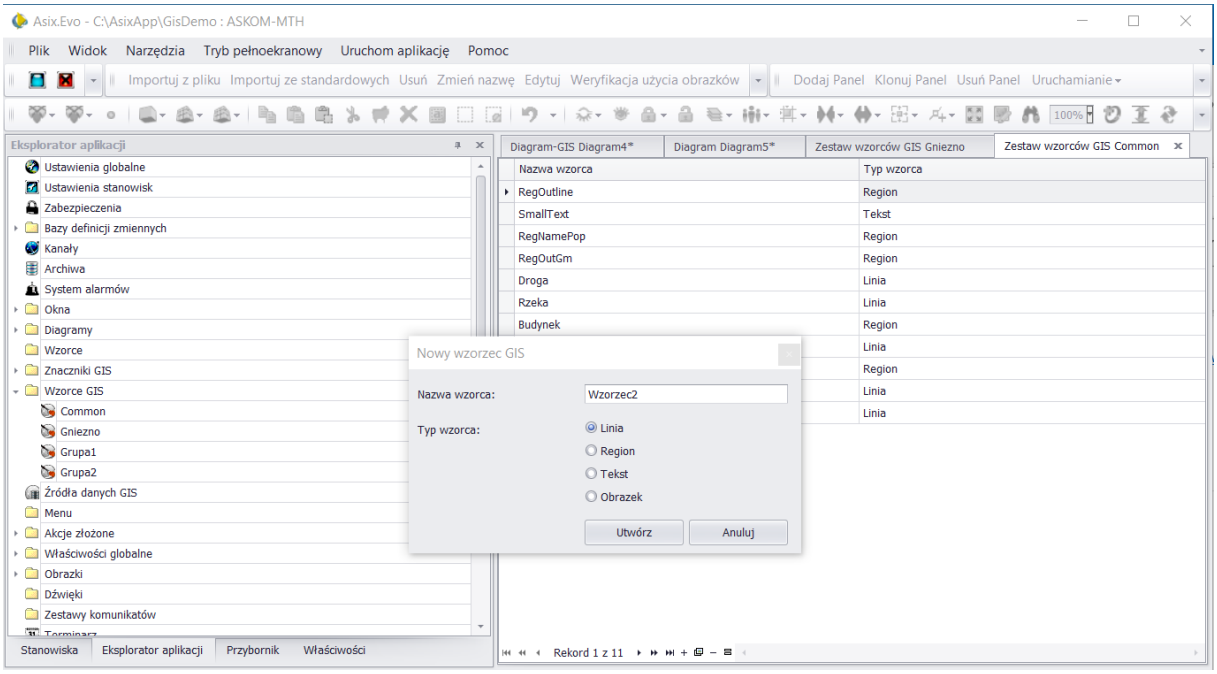
- ***Nazwa obrazu*** – umożliwia deklarowanie pliku graficznego, którego zawartość wyświetlana będzie w ramach obiektu Obraz;

2.12 Tworzenie wzorców / Znaczników GIS

Dla obiektów: Linia, Region, Obrazek i Tekst istnieje możliwość utworzenia wzorcowych zestawień właściwości tych obiektów - tzw. Wzorców GIS. Poza tym, projektant może utworzyć swoje własne indywidualnie zaprojektowane Znaczniki GIS.

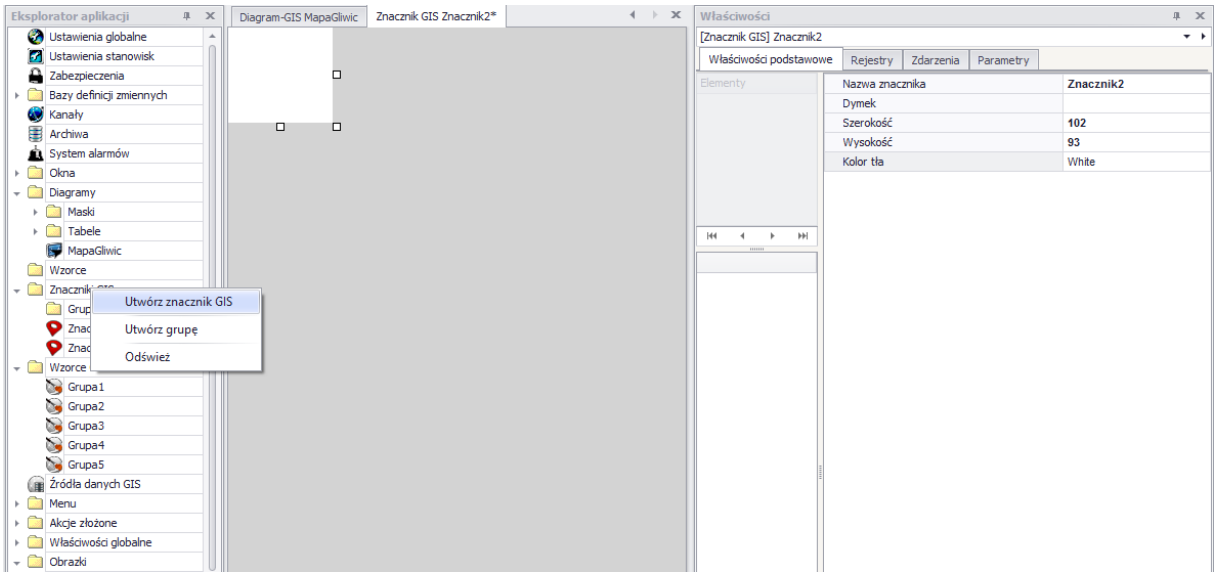
Wzorce GIS tworzone są poleceniem ***Utwórz zestaw wzorców GIS*** wywoływanym poprzez kliknięcie prawym klawiszem myszy na węźle ***Wzorce GIS*** w eksploratorze aplikacji. Potem, w oknie zakładki ***Zestaw wzorców GIS Grupa…*** uruchamianym po wybraniu zestawu wzorców GIS, należy dodać przyciskiem '***+***' wzorzec, deklarując przy tym typ wzorca (Linia, Region, Obrazek czy Tekst). Okno właściwości wzorca umożliwia zadeklarowanie indywidualnych wartości właściwości wzorcowego obiektu.

Gotowy wzorzec GIS przywoływany jest w polu ***Nazwa wzorca*** w oknie właściwości obiektu Linia, Region, Tekst, Obrazek.



Rysunek. Tworzenie wzorca obiektów GIS.

Znaczniki GIS tworzone są poleceniem ***Utwórz znacznik GIS*** wywoływanym poprzez kliknięcie prawym klawiszem myszy na węźle ***Znaczniki GIS*** w eksploratorze aplikacji. Potem, w oknie zakładki ***Znacznik GIS…*** uruchamianym po wybraniu Znacznika GIS, należy zaprojektować wygląd swojego Znacznika.



Rysunek. Tworzenie indywidualnego Znacznika GIS.

W zakresie projektowania Znacznika GIS projektant ma do dyspozycji zestaw właściwości podstawowych określających: ***Nazwę znacznika***, ***Dymek***, ***Szerokość***, ***Wysokość*** i ***Kolor tła*** oraz dwa obiekty Prostokąt i Tekst.

Prostokąt ma możliwość zadeklarowania zmiennej procesowej oraz stanów. Jedną z właściwości stanowych jest możliwość zadeklarowania pliku graficznego, którego zawartość zostanie wyświetlona w ramach Prostokąta. W szczególności, obiekt Prostokąt posiada następujące właściwości:

- Właściwości podstawowe:

    - ***Nazwa elementu***;

    - ***Główna zmienna***;

    - ***X,Y*** – współrzędne położenia Prostokąta na obszarze Znacznika GIS;

    - ***Szerokość*** / ***Wysokość*** – wymiary Prostokąta;

- Właściwości stanowe:

    - ***Opis stanu***;

    - ***Widoczny*** – pozwala sterować widocznością obiektu;

    - ***Kolor*** – kolor wypełnienia Prostokąta;

    - ***Kolor otoczki*** – kolor obramowania Prostokąta;

    - ***Szerokość otoczki*** – szerokość linii obramowania Prostokąta; wartość w pikselach;

    - ***Kolor przekreślenia*** - obiekt jest przekreślony; istnieje możliwość ustawiania koloru tego przekreślenia; kolor ‘Transparentny’ pozwala ukryć przekreślenie;

- ***Nawa obrazu*** – pozwala zadeklarować plik graficzny, którego zawartość będzie wyświetlana w ramach obiektu Prostokąt;

- ***Promień zaokrąglenia*** – pozwala na efekt zaokrąglonych wierzchołków Prostokąta; im wyższa wartość, tym wierzchołki są bardziej zaokrąglone;

Obiekt Tekst ma następujące właściwości;

- ***Nazwa elementu***;

- ***Główna zmienna***;

- ***X,Y*** – współrzędne położenia pola Tekstu na obszarze Znacznika GIS;

- Właściwości stanowe:

- ***Opis stanu***;

- ***Widoczny*** – pozwala sterować widocznością obiektu;

- ***Tekst*** – treść tekstu;

- ***Kolor*** – kolor tekstu;

- ***Kolor tła*** – kolor tła pola tekstu;

- ***Kolor przekreślenia*** - obiekt jest przekreślony; istnieje możliwość ustawiania koloru tego przekreślenia; kolor 'Transparentny' pozwala ukryć przekreślenie;

- ***Kolor ramki*** – kolor ramki okalającej pole tekstu; właściwość można określić, gdy określona jest właściwość *Styl ramki*;

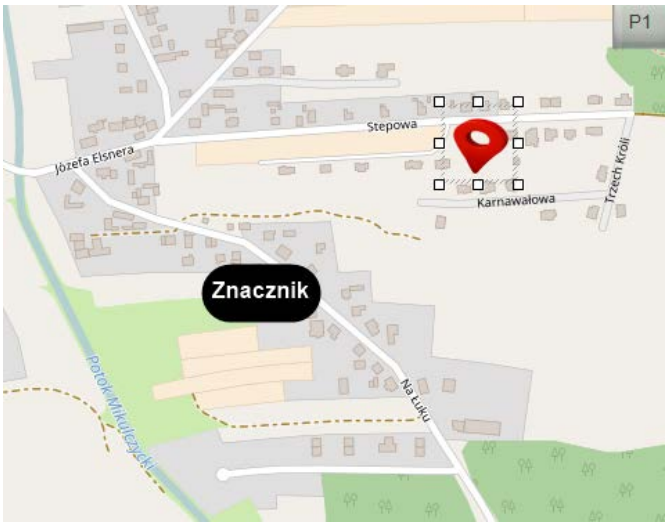
- ***Styl ramki*** – ramka okalająca pole tekstu może być: pojedynczą linią, linią dającą efekt wklęsłego obszaru tekstu lub linią dającą efekt wypukłego obszaru tekstu; tylko dla wariantu 'pojedyncza linia' można określić właściwość *Kolor ramki*;

- ***Wyrównanie w poziomie*** – wyrównanie tekstu: do lewej, do prawej wyśrodkowanie;

- ***Czcionka*** – określa krój czcionki tekstu;

- ***Styl czcionki*** – daje możliwość formatowania tekstu wg stylu: zwykły, pogrubienie, kursywa, podkreślenie i przekreślenie;

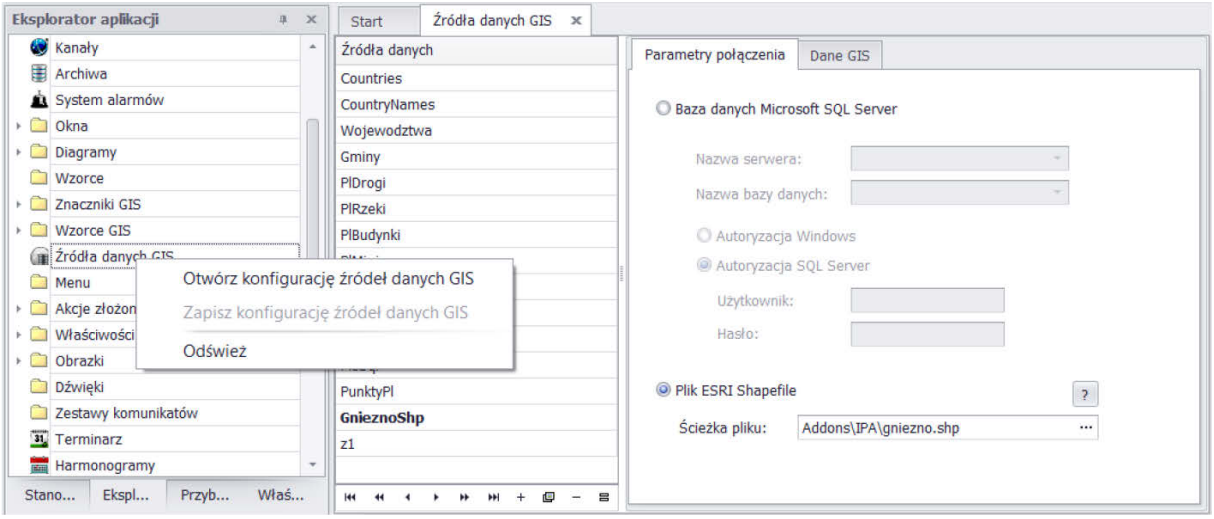
- ***Rozmiar czcionki*** – rozmiar czcionki w pikselach.



Rysunek. Przykłady Znaczników GIS.

2.13 Przygotowanie źródła danych GIS

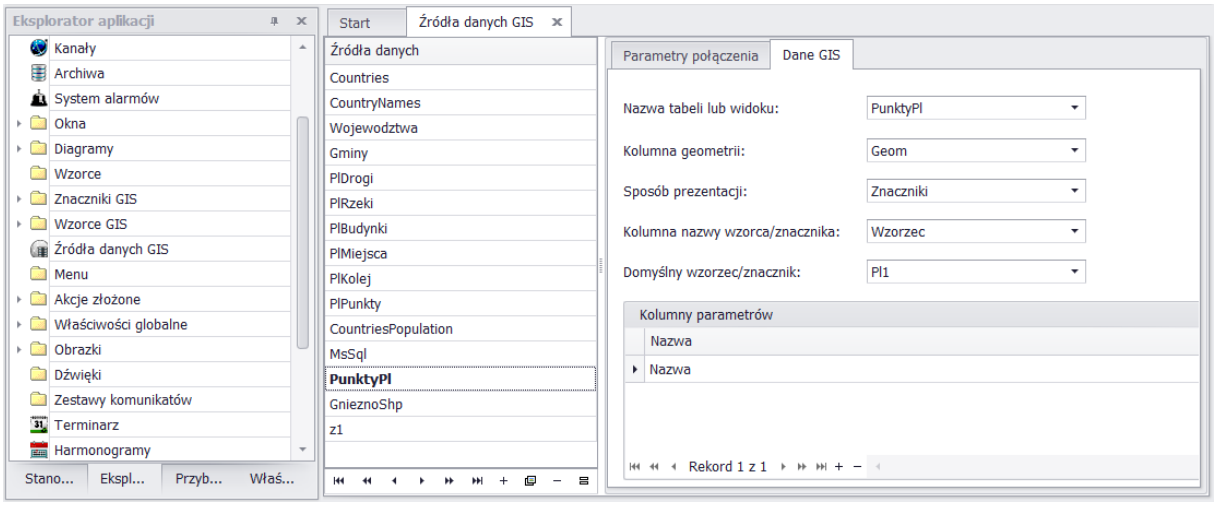
Źródła danych GIS tworzone są poleceniem ***Otwórz konfigurację źródeł danych GIS*** wywoływanym poprzez kliknięcie prawym klawiszem myszy na węźle ***Źródła danych GIS*** w eksploratorze aplikacji. Potem, w oknie zakładki ***Źródła danych GIS***, należy dodać przyciskiem '***+***' źródło danych w panelu ***Źródła danych*** i dokonać ustawień parametrów na zakładkach ***Parametry połączenia*** i ***Dane GIS***.



Rysunek. Deklarowanie źródła danych GIS – parametry połączenia.

Źródłem danych GIS może być baza MS SQL albo plik w formacie ESRI Shapefile.

Parametry połączenia do bazy MS SQL wymagają podania: ***Nazwy serwera***, ***Nazwa bazy danych*** oraz ***Autoryzacja Windows***/***Autoryzacja SQL Server***.



Dane GIS obejmują następujące elementy:

- ***Nazwa tabeli lub widoku*** – tylko dla trybu MS SQL;

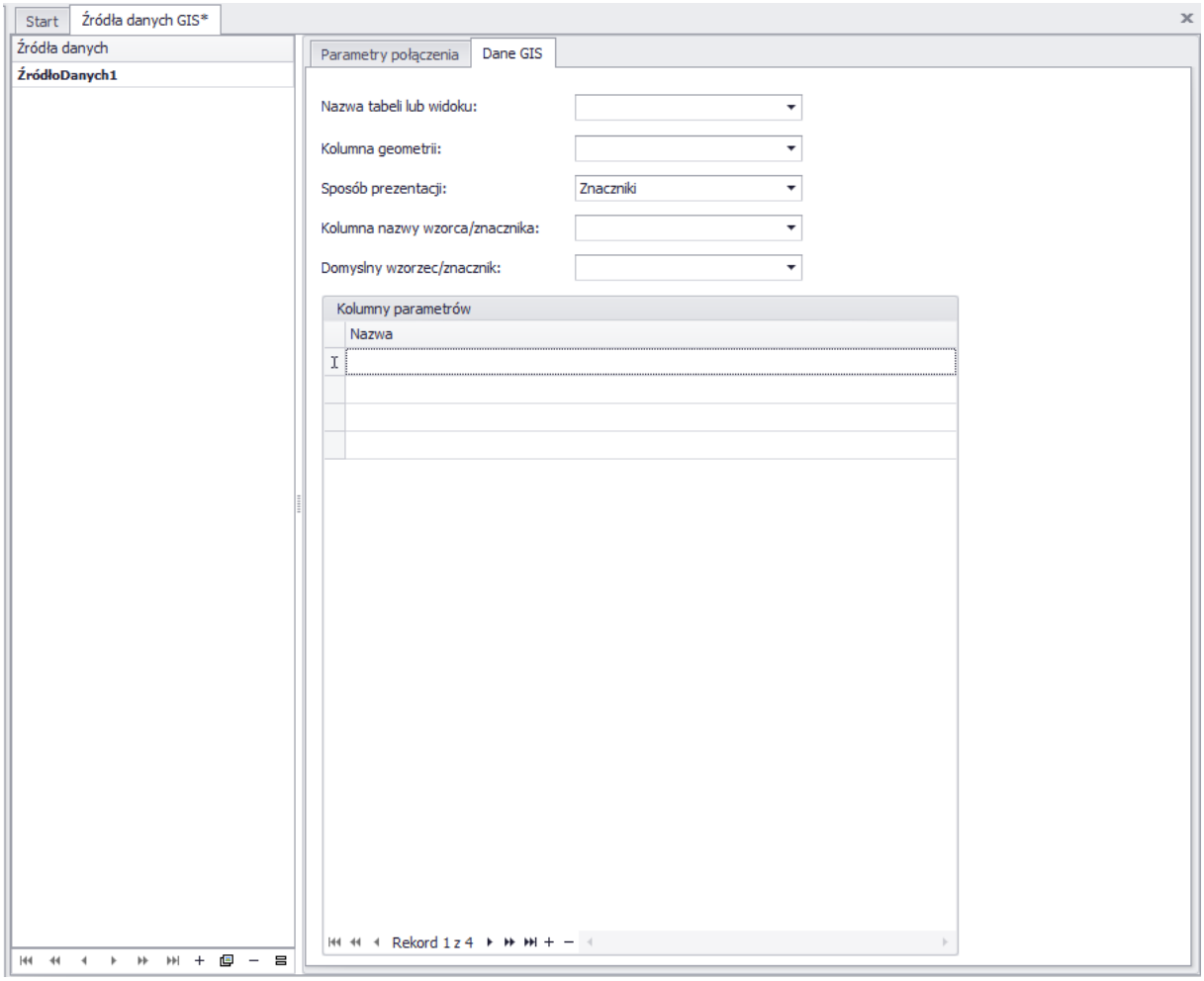
- ***Kolumna geometrii*** – tylko dla trybu MS SQL;

- ***Sposób prezentacji*** – określa obiekt, na który mają zostać przekonwertowane dane geograficzne z bazy; punkt (POINT) może być prezentowany jako Znacznik lub Tekst; linia (LINESTRING) jako obiekt Linia, a wielokąt (POLYGON) jako Region;

- ***Kolumna nazwy wzorca/znacznika*** – pozwala zadeklarować wzorzec, według którego mają zostać wyświetlone dane z bazy; wzorzec pobierany będzie z bazy ze wskazanej kolumny indywidualnie dla każdego wiersza;

- ***Domyślny wzorzec/znacznik*** – jeśli w bazie nie zostanie określony wzorzec, według którego wyświetlane będą dane z bazy, należy zadeklarować domyślny wzorzec;

- ***Kolumny parametrów*** **–** umożliwia zadeklarowanie nazwy kolumny z źródła danych GIS, zawierającej dowolne, inne niż geometria parametry, które zostaną automatycznie przypisane do zdefiniowanych parametrów wzorca; nazwa parametru musi być identyczna z nazwą kolumny.



Rysunek. Deklarowanie źródła danych GIS – dane GIS.

Przykład tworzenia bazy danych geometrycznych GIS:

Tworzenie przykładowej tabeli:

```
CREATE TABLE [dbo].[GeometrySample]
(
	[ID] [int] NOT NULL,
	[Name] [nvarchar](50) NULL,
	[Geom] [geometry] NULL,
	CONSTRAINT [PK_GeometrySample] PRIMARY KEY CLUSTERED ([ID] ASC )
);
```

Wstawianie do tabeli danych (kolejno: punkt, linia, wielokąt):

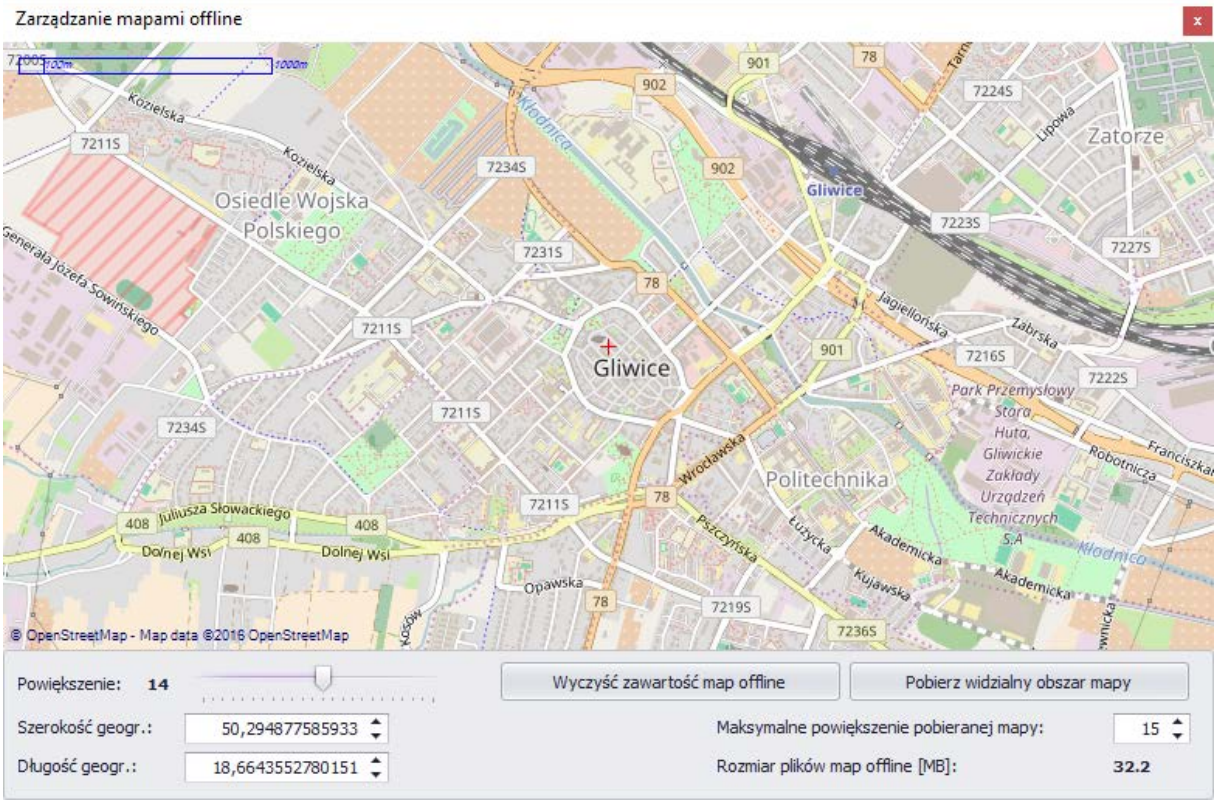
```
INSERT INTO GeometrySample (ID, Name, Geom)
  VALUES (1, 'P1', geometry::STGeomFromText('POINT (10 10)', 0));

INSERT INTO GeometrySample (ID, Name, Geom)
  VALUES (2, 'L1', geometry::STGeomFromText('LINESTRING (5 5, 25 25)', 0));

INSERT INTO GeometrySample (ID, Name, Geom)
  VALUES (3, 'R1', geometry::STGeomFromText('POLYGON ((10 10, 70 10, 80 30, 30 30, 10
10))', 0));
```

2.14 Deklarowanie map offline

Praca z podkładami mapowymi możliwa jest również w trybie offline. W tym celu należy zapisać w aplikacji widoki mapy, które wykorzystywane będą podczas pracy z aplikacją bez dostępu do Internetu. Służy do tego okno ***'Zarządzanie mapami offline'*** wywoływane za pomocą polecenia ***Zarządzanie mapami*** z menu głównego ***Narzędzia***. W oknie należy ustawić i powiększyć obszar mapy do pożądanej skali i zapisać ten fragment poleceniem ***Pobierz widzialny obszar mapy***. Na tym etapie należy przewidzieć potencjalne obszary, które będą oglądane na diagramach GIS aplikacji i pobrać je wszystkie. W przeciwnym razie, jeśli nie zostaną pobrane obszary w dużym powiększeniu, a końcowy użytkownik aplikacji będzie próbował powiększyć mapę z pobranych fragmentów o mniejszym stopniu powiększenia – zamiast poprawnego widoku zostanie mu wyświetlony obszar o niższym stopniu powiększenia, rozciągnięty do większych rozmiarów (powstanie efekt widocznych pikseli).



Rysunek. Okno zarządzania mapami offline.

Asix.Evo - Akcje operatorskie

1 Użycie akcji operatorskich

Akcje operatorskie są podstawowym środkiem sterowania pracą aplikacji przez operatora. Akcje są udostępniane operatorowi przez projektanta systemu przy pomocy poniższych mechanizmów:

- Obsługa zdarzeń obiektów i diagramów synoptycznych
- Menu użytkownika
- Terminarz
- Klawisze globalne

Ogólna postać akcji jest następująca:

nazwa_akcji (parametr1, parametr2, ...)

Liczba i znaczenie parametrów zależy od nazwy akcji. W najprostszym przypadku wszystkie parametry mogą być podane bezpośrednio, np.:

OpenWindow(OknoP1,PanelGlowny,Pokoj121,"")

Powyższa akcja otwiera okno o nazwie *OknoP1*, a w panelu *PanelGlowny* umieszczony zostanie diagram *Pokoj121*.

Każdy z parametrów akcji może być również dowolnym wyrażeniem. W akcjach obowiązują takie same zasady budowania wyrażeń jak we właściwościach obiektów synoptycznych. W momencie wykonania akcji wartości wszystkich użytych wyrażeń są wyliczane i dopiera tak skonstruowana treść akcji podlega wykonaniu, np.

OpenWindow(OknoP1,PanelGlowny,Pokoj +Variable(PokNr) ,"")

Parametr nazwy diagramu jest dynamicznie tworzony na podstawie wartości zmiennej *PokNr*. Jeżeli w momencie wykonania akcji wartość tej zmiennej będzie równa 101, to otwarty zostanie diagram o nazwie *Pokoj101*.

Istotne znaczenie dla wykonania akcji ma również kontekst jej użycia. Akcja wykonywana w obsłudze zdarzenia ma kontekstowy dostęp do właściwości i parametrów obiektów lub diagramów, dla których zdarzenie zostało aktywowane. Akcje wykonywane z terminarza lub klawisza globalnego takiej możliwości nie mają. Akcje wykonywane za pośrednictwem menu użytkownika mogą mieć dostęp do właściwości i parametrów obiektów i diagramów, o ile menu zostało otwarte poprzez akcję *ShowMenu* użytą w obsłudze zdarzenia - w takim przypadku następuje dziedziczenie kontekstu użycia.

OpenWindow(OknoP1,PanelGlowny,Pokoj +Variable() ,"")

W powyższym przypadku nazwa diagramu jest tworzona na podstawie wartości zmiennej głównej obiektu, dla którego zdarzenie zostało aktywowane - bezparametrowa funkcja *Variable* pobiera

wartość kontekstowej zmiennej głównej. Jeżeli w momencie wykonania akcji wartość tej zmiennej będzie równa 101, to otwarty zostanie diagram o nazwie *Pokoj101*.

```
OpenWindow(OknoP1,PanelGlowny,Pokoj +Parameter(PokNr) ,"")
```

W powyższym przypadku nazwa diagramu jest tworzona na podstawie wartości parametru o nazwie *PokNr*. Parametr ten jest parametrem diagramu lub wzorca (też zależnie od kontekstu użycia). Jeżeli w momencie wykonania akcji parametr będzie miał wartość równą 101, to otwarty zostanie diagram o nazwie *Pokoj101*.



Zobacz też

- *Wyrażenia*

- Elementy wizualizacji > *6.3. Standardowe zdarzenia obiektów*

4.2. Zdarzenia diagramów

5. Menu

- Parametryzacja aplikacji > *3. Konfiguracja Terminarza*

2 Akcje złożone

W przypadku, w którym zachodzi potrzeba wykonania sekwencji operacji można stosować akcje złożone. Są to po prostu zestawy zwykłych akcji operatorskich, które są kolejno wykonywane.

Akcje złożone są definiowane w panelu roboczym *Akcje złożone* otwieranym za pośrednictwem drzewa elementów panelu *Eksplorator aplikacji*. Akcja złożona jest uruchamiana przy pomocy akcji operatorskiej *ActionSet*.

Standardowo wykonanie akcji złożonej polega na sekwencyjnym wykonaniu wszystkich akcji składowych. Wykonanie akcji złożonej może być warunkowo przerwane przy pomocy akcji operatorskiej *Break*, np.:

```
Break(Variable(PokNr)==0)
```

Jeżeli wartość zmiennej *PokNr* będzie równa 0, to akcje składowe następujące po akcji *Break* nie zostaną wykonane

Akcje operatorskie wchodzące w skład akcji złożonej posiadają wszystkie cechy zwykłych akcji operatorskich. W szczególności, wartości wyrażeń użytych do zdefiniowania parametrów są wyliczane dopiero w momencie wykonania. Zachodzi też standardowe dziedziczenie kontekstu, jeżeli

akcja *ActionSet* została uruchomiona w kontekście obsługi zdarzenia, to w tym samym kontekście zostaną uruchomione wszystkie akcje składowe.

UWAGA:
W prostych przypadkach zamiast akcji złożonej można alternatywnie stosować akcję prostą *Actions*, która także pozwala sekwencyjnie wykonać kilka akcji składowych.

2.1 Parametry akcji złożonych

Cechą specyficzną akcji złożonych są parametry akcji. Nie należy ich mylić z parametrami diagramów i wzorców dostępnymi poprzez funkcję *Parameter*. W trakcie definiowania akcji złożonej można zdefiniować parametry akcji. Parametry identyfikowane są poprzez nazwę. Następnie w dowolnym miejscu akcji składowych można odwołać się do parametru poprzez wstawienie do treści akcji nazwy parametru poprzedzonej znakiem @.

Proces wykonania akcji złożonej wyjaśnia poniższy przykład.

Warunki wstępne:

- Akcja złożona *A1* ma zadeklarowane użycie parametrów o nazwie *Pokoj* i *Nastawa*
- Jedna z akcji składowych jest następującej postaci:

SetVariable(Nastawy+@Pokoj,@Nastawa)

- Akcja złożona jest uruchomiona poprzez akcję wykonaną w kontekście obsługi zdarzenia:

ActionSet(A1,PokNr+Parameter(PokNumer),"10+1”)

Wykonanie:

- W momencie wykonania akcji *ActionSet* wyliczana jest wartość drugiego parametru akcji poprzez połączenie tekstu "PokNr” i wartości parametru (diagramu lub wzorca) o nazwie *PokNumer*. Wartość wyliczonego parametru jest przekazywana w postaci tekstowej do akcji złożonej jako parametr *Pokoj*. Załóżmy, że wyliczona wartość jest równa "PokNr99”. Trzeci parametr akcji *ActionSet* przekazywany jest w niezmienionej formie "10+1" jako parametr *Nastawa*.

- W momencie wykonania akcji składowej *SetVariable* nazwa zmiennej wyliczana jest poprzez złożenie tekstu "Nastawy” i wartości parametru akcji złożonej *Pokoj* (uzyskana zostanie nazwa *NastawyPokNr99*). Wartość sterowana jest bezpośrednio wartością parametru *Nastawa* i będzie równa tekstowi "10+1”.
- W efekcie nastąpi ustawienie zmiennej *NastawyPokNr99* na wartość tekstową "10+1”.

Jeżeli celem było ustawienie wartości 11, to należało użyć akcję *ActionSet* w postaci:

ActionSet(A1,PokNr+Parameter(PokNumer),10+1)

W tym przypadku wyliczenie wartości wyrażenia *10+1* nastąpiłoby przed wykonaniem akcji złożonej.

3 Lista akcji operatorskich

Zarządzanie oknami synoptycznymi

CloseWindow - zamykanie okien synoptycznych

CopyWindow - utworzenie kopii okna synoptycznego

GisHideLayer – wyłącza widoczność wybranej warstwy diagramu GIS

GisRefreshLayer – odświeża (przeładowuje) wybraną warstwę diagramu GIS

GisShowLayer – włącza widoczność wybranej warstwy diagramu GIS

GisSwitchLayer – przełącza widoczność wybranej warstwy diagramu GIS

GisZoom – ustawia współczynnik powiększenia diagramu GIS

GisZoomIn – zwiększenie współczynnika powiększenia diagramu GIS

GisZoomOut – zmiejszenie współczynnika powiększenia diagramu GIS

HidePanel – ukrywa wskazany panel w oknie

MinimizeWindow - minimalizacja okien synoptycznych

NextDiagram - przełączenie na następny diagram zgodnie zapamiętaną historią wyświetlanych diagramów

OpenDiagram - otwarcie diagramu synoptycznego w dynamicznie tworzonym oknie

OpenWindow - otwarcie predefiniowanego okna synoptycznego lub podmiana diagramu w panelu okna wcześniej otwartego

PreviousDiagram - przełączenie na poprzednio wyświetlany diagram

SetWindowSize - zmiana rozmiaru okien synoptycznych

ShowPanel – przywraca ukryty panel w oknie

SwitchPanel – przełącza widoczność panelu w oknie

Zoom - bezwzględna zmiana skalowania diagramu

ZoomIn - względne powiększenie skalowanego diagramu

ZoomOut - względne pomniejszenie skalowanego diagramu

Dostęp do zmiennych procesowych i kontrola operacji sterujących

CancelControls - anulowanie oczekujących operacji sterujących

RefreshVariable - wymuszenie odświeżenia wartości zmiennej procesowej

SendControls - wykonanie oczekujących operacji sterujących

SetTableVariable - ustawienie wartości tablicowej zmiennej procesowej

SetVariable - ustawienie wartości zmiennej procesowej

ToggleBits - zmiana bitów w wartości zmiennej procesowej

Transfer - przepisuje wartość i status zmiennej źródłowej do zmiennej docelowej

VariableInfo - wyświetlenie okna z informacjami dotyczącymi wybranej zmiennej procesowej

Sterowanie pracą aplikacji

Restart – wykonuje pełny restart aplikacji

ShowControlPanel - otwarcie okna panelu kontrolnego

SwitchLanguage - zmiana aktualnego języka pracy aplikacji

SwitchWorkMode - przełączenie trybu pracy aplikacji

Terminate - zakończenie pracy aplikacji

Sterowanie wykonaniem akcji operatorskich

Actions - wykonanie sekwencji akcji składowych

ActionSet - uruchomienie akcji złożonej

Ask - wyświetlenie okienka z zapytaniem, akcja złożona jest przerywana w przypadku negatywnej odpowiedzi użytkownika

Break - warunkowe przerwanie wykonywania akcji złożonej

Execute - wykonanie akcji, której treść jest konstruowana dynamicznie

Nothing - akcja pusta

Perform - warunkowe wykonanie akcji

Sygnalizacja dźwiękowa

PlaySound - odtworzenie pliku dźwiękowego

PlaySoundLooping - odtworzenie pliku dźwiękowego z powtarzaniem

Speak – głosowe odczytanie podanego tekstu

StopSound - przerwanie odtwarzania dźwięku

Współpraca z systemem alarmów

AcceptAlarm - potwierdzenie alarmu

AcceptAlarmsOnTable - potwierdzenie alarmów pokazywanych w tabeli alarmów

EndAlarm - zakończenie alarmu

ExcludeAlarm - wykluczenie alarmu z obsługi

IncludeAlarm - ponowne włączenie alarmu do obsługi

StartAlarm - rozpoczęcie alarmu

Współpraca z systemem uprawnień

AddNote - wprowadzenie notatki operatora

ChangePassword - zmiana hasła użytkownika

ConfirmRole - weryfikacja przynależności użytkownika do roli

Login - zalogowanie użytkownika

Logout - wylogowanie użytkownika

SecurityBrowser - otwarcie okna przeglądania logów gromadzonych przez system zabezpieczeń

SecurityManager - otwarcie okna zarządzania uprawnieniami użytkowników

Uruchomienie składników aplikacji

AsreportDisplay - uruchomienie okna przeglądarki raportów opartych na technologii Reporting Services

PowerGuardManager - otwiera okno zarządzania modułu Strażnik Mocy

PowerGuardMeter - otwiera okno wykresu miernika modułu Strażnik Mocy

ReportGenerate - uruchamia proces generacji raportu skryptowego i/lub jego wyświetlenia

ReportsManager - otwiera okno zarządzania raportami skryptowymi

Run - uruchomienie zewnętrznego programu lub pliku

Script - uruchomienie skryptu użytkownika

ScriptClose - zakończenie pracy skryptu

ScriptRestart - zamknięcie i ponowne uruchomienie skryptu użytkownika

ShowMenu - otwarcie menu kontekstowego

SydelShow - przełączenie wyświetlanego węzła w programie SydelEko

Sterowanie pracą programu AsTrend

AstrendAsix6 - sterowanie programem AsTrend. Akcja używana tylko przy automatycznej konwersji aplikacji z systemu Asix6.

AstrendDisplay - wyświetlenie trendu lub raportu przy pomocy program AsTrend

AstrendPrint - wydrukowanie trendu lub raportu przy pomocy program AsTrend

AstrendPrintToPdf – eksport trendu lub raportu do pliku Pdf przy pomocy program AsTrend

Komunikacja z programem AsBase

AsbaseAdd - dodanie nowego rekordu w ramach połączenia

AsbaseClose - zamknięcie połączenia pomiędzy aplikacją Asix.Evo a programem AsBase

AsbaseDelete - usunięcie bieżącego rekordu w ramach połączenia

AsbaseLoad - załadowanie wartości z wybranej receptury do zestawu zmiennych procesowych. Wybór receptury wykonywany jest na podstawie jej nazwy.

AsbaseLoadByValue - załadowanie wartości z wybranej receptury do zestawu zmiennych procesowych. Wybór receptury wykonywany jest na podstawie wartości dowolnego pola.

AsbaseNavigate - sterowanie przeglądaniem i zarządzanie danymi w ramach połączenia.

AsbaseOpen - utworzenie połączenia pomiędzy aplikacją Asix.Evo a programem AsBase

AsbasePrint - wydrukowanie wskazanych danych z archiwów programu AsBase

AsbaseSave - dodanie nowego rekordu w trybie bezpołączeniowym

AsbaseShow - wyświetlenie okna programu AsBase z przełączeniem na żądany zestaw danych

AsbaseUpdate - zmiana wartości pól bieżącego rekordu w ramach połączenia

AsbaseWait - oczekiwanie na zakończenie wykonania wcześniejszych akcji rodziny Asbase...

Pomocnicze

ClearSelection - usunięcie stanu selekcji obiektu diagramu

EditPatterns - otwarcie okna edytora krzywych wzorcowych

EditTimetables - otwarcie okna edycji cykli harmonogramów

GetVarNames - zapisuje do schowka nazwy zmiennych używanych przez obiekty znajdujące się pod kursorem myszki

Message - zapisanie informacji do dziennika zdarzeń

ObjectAction - wykonanie operacji na wybranych obiektach

PrintScreen - zapisanie obrazu aktualnego okna, diagramu lub ekranu w schowku systemowym lub w pliku

ResetGlobalProperty - usuwa wykonane wcześniej modyfikacje wartości właściwości globalnej

SelectPattern - wybór krzywej wzorcowej i zapisanie jej nazwy do zmiennej procesowej

SetAppData – ustawienie lub zmiana wartości danej aplikacyjnej

SetBounds - zmiana położenia i rozmiaru obiektu, wzorca lub grupy obiektów

SetSelection – wymusza selekcję wskazanego obiektu diagramu

SetBounds - zmiana położenia i rozmiaru obiektu, wzorca lub grupy obiektów

SetGlobalProperty - zmienia wartość właściwości globalnej

SetParameter - zmiana wartości pojedynczego parametru diagramu

SetParameters - zmiana wartości parametrów diagramu

SetPosition - zmiana położenia obiektu, wzorca lub grupy obiektów

SetProperty - ustawienie właściwości obiektu lub diagramu synoptycznego

SetSelection – wymusza selekcję wskazanego obiektu diagramuSetSize - zmiana rozmiaru obiektu lub wzorca

ShowKeyboard - wyświetlenie okna klawiatury ekranowej

ShowNumericKeyboard - wyświetlenie okna numerycznej klawiatury ekranowej

4 Opisy akcji operatorskich

AcceptAlarm

Przeznaczenie

Akcja służy do potwierdzenia alarmu przez użytkownika. Alternatywną metodą programowego potwierdzenia alarmu jest użycie w funkcji *Accept* interfejsu *IAlarm* w kodzie skryptu użytkownika. Alarmy mogą być także potwierdzane z wykorzystaniem obiektu *Tabela alarmów aktywnych*.

Składnia

AcceptAlarm(*nazwa_domeny, identyfikator_alarmu*)

Parametry

nazwa_domeny

Nazwa domeny alarmów, której dotyczy akcja. Jeżeli nazwa nie jest podana, to akcja dotyczy domeny domyślnej (pierwszej).

identyfikator_alarmu

Identyfikator alarmu, który ma zostać potwierdzony.

 **Zobacz też**

- *System alarmów*

- obiekt *Tabela alarmów aktywnych*

- *Interfejs IAlarm*

AcceptAlarmsOnTable

Przeznaczenie

Akcja służy do potwierdzenia alarmów pokazywanych aktualnie w otwartych tabelach alarmów aktywnych. Jest to alternatywa do mechanizmów potwierdzania wbudowanych w interfejs użytkownika tabel alarmów. W parametrze akcji można określić, których tabel akcja dotyczy.

Składnia

AcceptAlarmsOnTable(*nazwa_tabeli*)

Parametry

nazwa_tabeli

Nazwa tabeli alarmów aktywnych, dla której należy potwierdzić alarmy. Można stosować znaki specjalne * i ?. W szczególności, nazwa * oznacza potwierdzenie wszystkich alarmów pokazywanych przez wszystkie otwarte tabele.

 **Zobacz też**

- *System alarmów*
- *Tabela alarmów aktywnych*
- akcja *AcceptAlarm*

Actions

Przeznaczenie

Akcja służy do sekwencyjnego wykonania zestawu akcji składowych. W swoim działaniu jest zbliżona do sposobu działania akcji złożonych. Można ją stosować w prostszych przypadkach, gdy ilość akcji składowych i ich złożoność nie są zbyt duże.

Składnia

Actions(*akcja1, akcja2, ...*)

Parametry

akcja1, ..., akcjaN

Akcje składowe, które zostaną kolejno wykonane.

Przykłady

Actions(SetVariable(v1,1), SetVariable(v2,1))

Akcja uruchamia kolejno dwie akcje składowe, które ustawiają wartości dwóch zmiennych procesowych.

 **Zobacz też**

2. Akcje złożone

ActionSet

Przeznaczenie

Akcja służy do uruchomienia akcji złożonej.

Składnia

ActionSet(*nazwa , parametr1, parametr2,...*)

Parametry

nazwa

Nazwa akcji złożonej.

parametr1,..., parametrN

Wartości parametrów akcji złożonej. Liczba parametrów użytych w wywołaniu musi być zgodna z liczbą parametrów w definicji akcji złożonej. Kolejne parametry wywołania zostaną podstawione do parametrów akcji złożonej zgodnie z kolejnością ich występowania w definicji akcji złożonej.

Przykłady

ActionSet(wyłacz_agregat, agr100, agr101)

Akcja uruchamia akcję złożoną o nazwie *wyłącz_agregat*. Zakładając, że akcja *wyłącz_agregat* posiada dwa parametry o nazwach *zmienna1* i *zmienna2* zdefiniowane w podanej kolejności, to w momencie wykonania akcji parametr *zmienna1* będzie równy *agr100*, a parametr *zmienna2* będzie równy *agr101*.

 **Zobacz też**

2. Akcje złożone

AddNote

Przeznaczenie

Akcja powoduje otwarcie okna dialogowego, w którym operator systemu może wprowadzić notatkę tekstową.

Składnia

AddNote (*identyfikator_segmentu*)

Parametry

Identyfikator_segmentu

W parametrze można podać identyfikator segmentu, z którym wprowadzona notatka będzie skojarzona. Wartość parametru może być pusta.

 **Zobacz też**

- Parametryzacja_aplikacji > *5. System uprawnień*

AsbaseAdd

Przeznaczenie

Akcja służy do dopisania nowego rekordu w połączeniu utworzonym wcześniej przy pomocy akcji *AsbaseOpen.*Wartości pól nowego rekordu wynikają z wartości zmiennych procesowych należących do zestawu zmiennych użytego w połączeniu.

Składnia

AsbaseAdd (*identyfikator_połączenia*)

Parametry

Identyfikator_połączenia

W parametrze należy podać identyfikator, który został wcześniej użyty w akcji *AsbaseOpen.*

Zobacz też

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

- akcja *AsbaseOpen*

AsbaseClose

Przeznaczenie

Akcja służy utworzenia zamknięcia połączenia utworzonego wcześniej przy pomocy akcji *AsbaseOpen.*

Akcję *AsbaseClose* należy użyć w obsłudze zdarzenia *Zamknięcie diagramu* diagramu, dla którego połączenie zostało otwarte.

Składnia

AsbaseClose (*identyfikator_połączenia*)

Parametry

Identyfikator_połączenia

W parametrze należy podać identyfikator, który został wcześniej użyty w akcji *AsbaseOpen*.

Zobacz też

- akcja *AsbaseOpen*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseDelete

Przeznaczenie

Akcja służy do usunięcia bieżącego rekordu w połączeniu utworzonym wcześniej przy pomocy akcji *AsbaseOpen.*

Składnia

AsbaseDelete (*identyfikator_połączenia*)

Parametry

Identyfikator_połączenia

W parametrze należy podać identyfikator, który został wcześniej użyty w akcji *AsbaseOpen.*



Zobacz też

- akcja *AsbaseOpen*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseLoad

Przeznaczenie

Akcja pozwala na załadowanie nastaw pochodzących z rekordu w wybranej grupie receptur do zmiennych procesowych skojarzonych z tą grupą w programie AsBase. Rekord nastaw może być wybierany poprzez bezpośrednie podanie nazwy receptury, lub może być wykonana interaktywna selekcja nazwy.

W przypadku korzystania z trybu połączeniowego jako akcję ładującą recepturę należy zawsze stosować akcję *AsbaseNavigate* z opcją *$Load*.

Składnia

AsbaseLoad(*identyfikator_grupy_receptur, identyfikator_zestawu_zmiennych, nazwa_receptury*)

Parametry

identyfikator_grupy_receptur

Identyfikator grupy receptur skąd pobierane są wartości pól receptury

identyfikator_zestawu_zmiennych

Identyfikator zestawu zmiennych, do którego zmiennych zostaną wysłane wartości pochodzące z wybranej receptury.

nazwa_receptury

Nazwa receptury, która ma zostać załadowana. Można podać nazwę dokładną lub wzorzec nazwy zawierający znaki specjalne * i ?. Jeżeli nazwa receptury nie zostanie podana lub zawiera znaki specjalne , to wyświetlone zostanie okno z listą receptur (wszystkich lub zgodnych z wzorcem nazwy) zdefiniowanych w grupie receptur, z której użytkownik będzie mógł wybrać recepturę, która ma być załadowana.

Przykłady

AsbaseLoad(Nastawy,Zbiornik1 ,"")

Akcja spowoduje wyświetlenie okienka z listą nazw wszystkich receptur w grupie Nastawy. Użytkownik będzie mógł wybrać recepturę, której wartości pól mają być załadowane do zmiennych procesowych z zestawu *Zbiornik1*.



Zobacz też

- akcja *AsbaseNavigate*, akcja *AsbaseLoadByValue*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseLoadByValue

Przeznaczenie

Akcja pozwala na załadowanie nastaw pochodzących z rekordu w wybranej grupie receptur do zmiennych procesowych skojarzonych z tą grupą w programie AsBase. Rekord nastaw jest wybierany poprzez podanie wartości referencyjnej dla dowolnego pola receptury.

Składnia

AsbaseLoadByValue(*identyfikator_grupy_receptur, identyfikator_zestawu_zmiennych, identyfikator_pola, wartość_pola*)

Parametry

identyfikator_grupy_receptur

Identyfikator grupy receptur, skąd pobierane są wartości pól receptury.

identyfikator_zestawu_zmiennych

Identyfikator zestawu zmiennych, do którego zmiennych zostaną wysłane wartości pochodzące z wybranej receptury.

identyfikator_pola

Identyfikator pola receptury, dla którego zostanie wykonany test zgodności wartości. Oprócz identyfikatorów pól zdefiniowanych przez użytkownika, można użyć identyfikator *Name*, oznaczający test zgodności nazwy receptury.

wartość_pola

Wartość referencyjna dla pola podanego w parametrze *identyfikator_pola*. Załadowana zostanie receptura, dla której zajdzie zgodność wartości referencyjnej z wartością zadeklarowanego pola.

UWAGA:

Istotną różnicą w sposobie wykonania akcji *AsbaseLoadByValue* w stosunku do akcji *AsbaseLoad* jest to, że fakt załadowania receptury do zmiennych nie jest rejestrowany w tabeli wykonanych ładowań dla grupy receptur. Akcja *AsbaseLoadByValue* z użyciem pola *Name* działa identycznie jak akcja *AsbaseLoad*, za wyjątkiem zarejestrowania faktu wykonania operacji.

Przykłady

AsbaseLoadByValue(Nastawy,Preview ,ProdId,Variable(PID))

Do zmiennych zestawu *Preview* zostaną załadowane wartości z rekordu grup receptur *Nastawy*, dla którego wartość pola *ProdId* będzie zgodna z wartością zmiennej PID.

 **Zobacz też**

- akcja *AsbaseNavigate*, akcja *AsbaseLoad*

- Parametryzacja_aplikacji > *Współpraca z programem AsBase*

AsbaseNavigate

Przeznaczenie

- sterowanie przeglądaniem I zarządzanie danymi w ramach połączenia.

Akcja służy do sterowania przeglądaniem i zarządzania danymi w ramach połączenia utworzonego wcześniej przy pomocy akcji *AsbaseOpen.*Akcja pozwala na zmianę bieżącego rekordu w połączeniu, wykonanie operacji załadowania receptury i zmianę parametrów połączenia.

Składnia

AsbaseNavigate (*identyfikator_połączenia, typ_operacji, parametry_operacji*)

Parametry

Identyfikator_połączenia

W parametrze należy podać identyfikator, który został wcześniej użyty w akcji *AsbaseOpen.*

typ_operacji

Parametr określa, jaka operacja ma być wykonana w ramach połączenia. Możliwe wartości to jedna z poniższych stałych:

$Count - powoduje przeliczenie ilości rekordów widocznych w połączeniu zgodnie z aktualnym filtrem i umieszczenie tej liczby w odpowiedniej zmiennej zestawu zmiennych.

$Filter - powoduje zdefiniowanie nowego filtru oraz przejście do ostatniego rekordu połączenia po zastosowaniu filtra. Szczegóły dotyczące składni filtra znajdują się rozdziale Współpraca z programem AsBase.

$First - powoduje przejście do pierwszego rekordu w ramach połączenia.

$Download - wysłanie zawartości bieżącego rekordu do zmiennych z zestawu zmiennych podanego jawnie w parametrze *parametry_operacji*.

$Last- powoduje przejście do ostatniego rekordu w ramach połączenia.

$Load - powoduje przejście do rekordu wskazanego nazwą receptury lub wybranego interaktywnie przez użytkownika. Operacja dostępna tylko dla połączeń dotyczących grupy receptur.

$Next- powoduje przejście do następnego rekordu w ramach połączenia.

$Previous- powoduje przejście do poprzedniego rekordu w ramach połączenia.

parametry_operacji

W przypadku operacji *$Filter* parametr określa nową treść filtra. W przypadku operacji $Download parametr określa nazwę zestawu zmiennych, do którego należy wysłać wartości z bieżącego rekordu.

W przypadku operacji $*Load* w parametrze należy podać nazwę receptury. Można podać nazwę dokładną lub wzorzec nazwy zawierający znaki specjalne * i ?. Jeżeli nazwa receptury nie zostanie podana lub zawiera znaki specjalne , to zostanie wyświetlona lista nazw receptur (wszystkich lub zgodnych z wzorcem nazwy), z której użytkownik będzie mógł wybrać recepturę.

Przykłady

AsbaseNavigate(conn1, $Next, null)

Akcja powoduje przejście do następnego rekordu w ramach danych wynikających z połączenia *conn1*. Do zmiennych należących do zestawu zmiennych połączenia zostaną wpisane wartości pobrane z nowego bieżącego rekordu.

AsbaseLoad(conn1, $Load, RC_123)

Akcja powoduje przejście do receptury o nazwie *RC*_123 w ramach grupy receptur wynikającej z połączenia *conn1*. Do zmiennych należących do zestawu zmiennych połączenia zostaną wpisane wartości pobrane z nowego bieżącego rekordu.

 **Zobacz też**

- akcja *AsbaseOpen*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseOpen

Przeznaczenie

Akcja służy utworzenia połączenia służącego do wymiany danych pomiędzy aplikacją Asix.Evo a programem AsBase. W parametrach akcji określa się, które dane będą wymieniane i przez jakie zmienne procesowe. Po wykonaniu akcji dane z bieżącego rekordu połączenia będą zapisywane w zmiennych interfejsowych. Użycie akcji wymaga wcześniejszego sparametryzowania aplikacji AsBase.

Akcję *AsbaseOpen* należy użyć w obsłudze zdarzenia *Otwarcie diagramu* dla diagramu który musi pozostać otwarty przez cały czas trwania wymiany danych. W zdarzeniu *Zamknięcie diagramu* w celu zakończenia połączenia należy użyć akcję *AsbaseClose*.

Składnia

AsbaseOpen (*identyfikator_połączenia, typ_danych, identyfikator_źródła, identyfikator_zestawu_zmiennych, identyfikator_podrzędnego_źródła, identyfikator_podrzędnego_zestawu_zmiennych, filtr*)

Parametry

Identyfikator_połączenia

W parametrze należy podać identyfikator, który będzie jednoznacznie identyfikował połączenie z określonym źródłem danych i zestawem zmiennych. Identyfikator wykorzystywany jest później w innych akcjach z rodziny *AsbaseXxx*.

typ_danych

Parametr określa typ źródła danych. Możliwe wartości to jedna z poniższych stałych:

$Archive- połączenie dotyczy zestawu rejestracji

$Recipes - połączenie dotyczy grupy receptur

$Loads - połączenie dotyczy historii ładowań receptur

identyfikator_źródła

Identyfikator źródła danych, w zależności od wartości parametru *typ_danych,* jest to identyfikator zestawu rejestracji lub grupy receptur.

identyfikator_zestawu_zmiennych

Identyfikator zestawu zmiennych, za pośrednictwem którego będzie wykonywana komunikacja pomiędzy aplikacją Asix.Evo a programem AsBase.

identyfikator_podrzędnego_źródła

Identyfikator podrzędnego źródła danych, czyli źródła, które w parametryzacji programu AsBase zostało zadeklarowane jako pozostające w relacji ze źródłem podstawowym.

identyfikator_podrzędnego_zestawu_zmiennych

Identyfikator zestawu zmiennych, za pośrednictwem którego będzie wykonywana komunikacja pomiędzy aplikacją Asix.Evo a programem AsBase. Zestaw przeznaczony dla komunikacji ze źródłem podrzędnym.

filtr

Wyrażenie filtrujące, które pozwala na ograniczenie zakresu danych dostępnych poprzez połączenie. Szczegóły dotyczące składni filtra znajdują się w rozdziale *Współpraca z programem AsBase*.

Przykłady

AsbaseOpen(conn1, $Archive, Farby, FarbyDgr, null, null, "V_Akt=1")

Akcja tworzy połączenie o nazwie *conn1*. W ramach tego połączenie dostępne będą rekordy danych z zestawu rejestracji o identyfikatorze *Farby*, spełniające warunek mówiący o tym, że wartość pola zestawu o identyfikatorze*Akt* będzie równa 1. Dane będą wymieniane poprzez zestaw zmiennych o identyfikatorze *FarbyDgr*. W zmiennych należących do tego zestawu będą zapisywane wartości aktywnego rekordu połączenia.

Zobacz też

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

- akcja *AsbaseClose*, akcja *AsbaseAdd*, akcja *AsbaseDelete*, akcja *AsbaseNavigate*, akcja *AsbaseUpdate*

AsbasePrint

Przeznaczenie

Akcja powoduje wydrukowanie przez program AsBase zawartości wybranych tabel.

Składnia

AsbasePrint (typ_danych, identyfikator_źródła , identyfikator_widoku)

Parametry

typ_danych

Parametr określa typ źródła danych. Możliwe wartości to jedna z poniższych stałych:

$Archive - połączenie dotyczy zestawu rejestracji

$Recipes - połączenie dotyczy grupy receptur

$Loads - połączenie dotyczy historii ładowań receptur

identyfikator_źródła

Identyfikator źródła danych, w zależności od wartości parametru *typ_danych,* jest to identyfikator zestawu rejestracji lub grupy receptur.

identyfikator_widoku

Identyfikator widoku skojarzonego w programie Asbase ze źródłem danych. Widok określa sposób wydrukowania rekordów, może też zawierać informacje służące do filtrowania danych.

Przykłady

AsbasePrint($Recipes, Nastawy, "")

Akcja spowoduje wydrukowanie wszystkich rekordów z grupy receptur *Nastawy* w układzie domyślnym.

**Zobacz też**

- *Parametryzacja_aplikacji* > *Współpraca z programem AsBase*

AsbaseSave

Przeznaczenie

Akcja powoduje dopisanie nowego rekordu do zestawu rejestracji lub grupy receptur. Wartości pól nowego rekordu wynikają z wartości zmiennych procesowych należących do zestawu zmiennych podanych w parametrze akcji.

Operacja wykonywana jest w trybie bezpołączeniowym bez konieczności spełniania żadnych warunków wstępnych. W trybie połączeniowym należy stosować akcję *AsbaseAdd*.

Składnia

AsbaseSave (*typ_danych, identyfikator_źródła, identyfikator_zestawu_zmiennych*)

Parametry

typ_danych

Parametr określa typ zapisywanych danych. Możliwe wartości to jedna z poniższych stałych:

$Archive - połączenie dotyczy zestawu rejestracji

$Recipes - połączenie dotyczy grupy receptur

identyfikator_źródła

Identyfikator źródła danych - w zależności od wartości parametru *typ_danych* jest to identyfikator zestawu rejestracji lub grupy receptur.

identyfikator_zestawu_zmiennych

Identyfikator zestawu zmiennych. Wartości zmiennych z tego zestawu zostaną zapisane do odpowiednich pól nowego rekordu danych.

**Zobacz też**

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseShow

Przeznaczenie

Akcja powoduje uaktywnienie okna programu AsBase (wyciągnięcie go na wierzch). W parametrach akcji można określić, które dane mają zostać pokazane.

Składnia

AsbaseShow (*typ_danych, identyfikator_źródła, identyfikator_zestawu_zmiennych, identyfikator_widoku*)

Parametry

typ_danych

Parametr określa typ źródła danych. Możliwe wartości to jedna z poniższych stałych:

$Archive - połączenie dotyczy zestawu rejestracji

$Recipes - połączenie dotyczy grupy receptur

$Loads - połączenie dotyczy historii ładowań receptur

identyfikator_źródła

Identyfikator źródła danych - w zależności od wartości parametru *typ_danych* jest to identyfikator zestawu rejestracji lub grupy receptur.

identyfikator_zestawu_zmiennych

Identyfikator zestawu zmiennych. Ustawiany tylko wtedy, gdy chce się wyświetlić okno podglądu wartości zmiennych dla zestawu zmiennych.

identyfikator_widoku

Identyfikator widoku skojarzonego w programie AsBase ze źródłem danych. Widok określa sposób wyświetlania rekordów, może też zawierać informacje służące do filtrowania danych.

Przykłady

AsbaseShow($Recipes, Nastawy, "",V1)

Akcja spowoduje wyświetlenie w oknie programu AsBase danych należących do grupy receptur *Nastawy*. Sposób wyświetlenia oraz zakres danych jest określony przez zdefiniowany w aplikacji AsBase widok o identyfikatorze *V1*.

**Zobacz też**

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseUpdate

Przeznaczenie

Akcja służy do zmiany wartości pól bieżącego rekordu w połączeniu utworzonym wcześniej przy pomocy akcji *AsbaseOpen.*Nowe *w*artości pól rekordu wynikają z wartości zmiennych procesowych należących do zestawu zmiennych użytego w połączeniu.

Składnia

AsbaseUpdate (*identyfikator_połączenia*)

Parametry

Identyfikator_połączenia

W parametrze należy podać identyfikator, który został wcześniej użyty w akcji *AsbaseOpen.*

**Zobacz też**

- akcja *AsbaseOpen*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsBase*

AsbaseWait

Przeznaczenie

Akcja oczekuje na zakończenie wykonywania wcześniej wywołanych akcji rodziny Asbase...

Operacje zlecone przy pomocy akcji rodziny Asbase... wykonywane są zawsze w trybie asynchronicznym. W efekcie wykonanie akcji operatorskiej z reguły kończy się wcześniej niż wykonanie operacji zleconych przez tą akcję. Może to stanowić problem w przypadku użycia akcji złożonych. Jeżeli jakieś akcje wymagają zakończenia operacji wykonywanych przez wcześniejsze akcje Asbase... (np. korzystają ze zmiennych ustawianych we wcześniejszej akcji), to w takim przypadku należy dodać do akcji złożonej wywołanie akcji AsbaseWait.

Składnia

AsbaseWait ()

Ask

Przeznaczenie

Akcja przeznaczona do użytku w akcjach złożonych. Wyświetla okienko komunikatów z zapytaniem podanym w parametrze akcji oraz dwoma przyciskami: *OK* i *Anuluj*. Jeżeli użytkownik wybierz przycisk *Anuluj*, to dalsze wykonanie akcji złożonej jest przerywane.

Składnia

Ask (*pytanie*)

Ask (*pytanie*, *czas_oczekiwania*, *wartość_domyślna*)

Parametry

pytanie

Treść zapytania wyświetlonego w okienku komunikatu.

czas_oczekiwania

Okres czasu wyrażony w sekundach, po którym okno zapytania zostanie zamknięte, jeżeli użytkownik nie wybierze wcześniej żadnej odpowiedzi. W wariancie akcji bez podania czasu oczekiwania, okienko pozostaje otwarte bezterminowo.

wartość_domyślna

Domyślny rezultat zapytania (*true*/*false*), który zostanie użyty, gdy użytkownik nie wybierze żadnej odpowiedzi.

Przykłady

Actions(Ask("Włączyć pompę ?"),SetVariable(v1,1))

Operacja ustawienia zmiennej v1, zostanie wykonana tylko wtedy, gdy użytkownik odpowie twierdząco na zapytanie.

 **Zobacz też**

Akcje złożone, *akcja Perform*, *akcja Actions*, *akcja Break*

AsreportDisplay

Przeznaczenie

Akcja służy do otwarcia okna przeglądarki raportów utworzonych w technologii Reporting Services z wykorzystaniem modułu AsRaport pakietu Asix. Okno pozwala na generowanie i przeglądanie wybranych raportów. Parametry generacji mogą być zadawane w akcji lub określane w sposób interaktywny.

Użycie akcji wymaga ustawienia parametrów dostępu do serwera usług raportowych. Parametry te podawane są w panelu roboczym *Ustawienia stanowisk* w zakładce *Źródła danych*.

Oprócz parametrów podanych bezpośrednio w treści akcji, na dostępność niektórych funkcji okna przeglądarki mają wpływ także uprawnienia(rola) aktualnie zalogowanego użytkownika.

Składnia

AsreportDisplay (*nazwa_raportu, parametry,pokazuj_parametry, współrzędna_X, współrzędna_Y, katalog_raportów*)

Parametry

nazwa_raportu

Parametr służy do podania nazwy raportu, który należy wyświetlić w oknie przeglądarki. Nazwa musi być zgodna z nazwą raportu opublikowanego na serwerze i może zawierać ścieżkę względną do katalogu raportu. Parametr może być pusty - użytkownik będzie mógł wybrać raport interaktywnie.

Parametry

Parametr służy do określenia wartości parametrów generacji raportu. Zestaw możliwych parametrów wynika z definicji raportu. Parametr ma postać tekstu złożonego z par postaci *@nazwa_parametry=wartość_parametru* rozdzielonych znakiem spacji, np.*"@month=12 @section=A"*.

pokazuj_raporty

Parametr typu logicznego *true/false*, który określa, czy wyświetlać panel edycji parametrów raportu.

współrzędna_X

Parametr określa współrzędną X okna przeglądarki raportów(lewego górnego narożnika).

współrzędna_Y

Parametr określa współrzędną Y okna przeglądarki raportów (lewego górnego narożnika).

katalog_raportów

Nazwa podkatalogu (lub ścieżka względna podkatalogu) na serwerze raportów, którego zawartość (nazwy raportów oraz podkatalogi) należy umieścić w polu wyboru raportu okna przeglądarki. Wartość * oznacza, że należy pokazać wszystkie raporty, a wartość pusta oznacza, że pole wyboru raportu nie będzie pokazywane.

AstrendAsix6

Przeznaczenie

Akcja służy do sterowania pracą programu AsTrend. Jest do dokładny odpowiednik akcji *AsTrend* systemu Asix6. Akcja jest używana w przypadku automatycznej konwersji aplikacji systemu Asix6. W aplikacjach tworzonych od podstaw należy używać akcji *AstrendDisplay* i *AstrendPrint*.

Składnia

AstrendAsix6 (*nazwa_trendu, czas_początku, współrzędna_X, współrzędna_Y, monitor, parametry_dodatkowe*)

Parametry

nazwa_trendu

Nazwa pliku z definicją trendu. Wpisanie znaku "?" powoduje otwarcie okna, w którym użytkownik może wybrać trend do wyświetlenia.

Po nazwie pliku trendu można podać nazwy zmiennych, które zostaną dodane do wykresu zmiennych zadeklarowanych w definicji trendu. Jeżeli nazwy zmiennych będą rozdzielone znakami #, to zmienne zostaną dodane do trendu. Jeżeli nazwy zmiennych będą rozdzielone znakami <, to zmienne zastąpią zmienne znajdujące się w definicji trendu.

czas_początku

Określenie czasu początku wyświetlanego trendu. Parametr może być czasem bezwzględnym w postaci *yyyymmddhhnnss*, czasem w formacie OPC lub nazwą zmiennej procesowej. W sytuacji, gdy parametrem jest nazwą zmiennej, to jej wartość interpretowana jest jako ilość sekund od 1.1.1970 dla zmiennych o wartościach liczbowych, a dla zmiennych tekstowych jako czas w postaci *yyyymmddhhnnss* lub w formacie OPC. Możliwe jest też podanie czasów początku i końca trendu, rozdzielając je znakiem średnika.

współrzędna_X

Parametr określa współrzędną X okna programu AsTrend (lewego górnego narożnika). Wartość -1 oznacza pobranie odpowiedniej współrzędnej z definicji trendu.

współrzędna_Y

Parametr określa współrzędną Y okna programu AsTrend (lewego górnego narożnika). Wartość -1 oznacza pobranie odpowiedniej współrzędnej z definicji trendu.

monitor

W parametrze należy podać numer monitora, na którym powinno otworzyć się okno programu AsTrend. Wartość 0 oznacza monitor, na którym aktualnie znajduje się kursor myszki, -1 oznacza monitor wynikający z przyjętych współrzędnych x, y.

parametry_dodatkowe

Dodatkowe parametry akcji zgodne z opisem akcji *AsTrend* systemu Asix6.



Zobacz też

- akcja *AsTrendDisplay*, akcja *AsTrendPrint*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsTrend*

AstrendDisplay

Przeznaczenie

Akcja służy do otwarcia okna programu AsTrend z wyświetlonym wybranym trendem lub raportem. W akcji można podać nazwę wzorca trendu(raportu) i czas początku danych. Opcjonalnie można podać nazwy zmiennych użytych na wykresie lub w raporcie.

Składnia

AstrendDisplay (*nazwa_trendu, czas_początku, współrzędna_X, współrzędna_Y, monitor*)
AstrendDisplay (*nazwa_trendu, czas_początku, czas_końca, współrzędna_X, współrzędna_Y, monitor*)

Parametry

nazwa_trendu

Nazwa pliku z definicją trendu(raportu). Wpisanie znaku "?" powoduje otwarcie okna, w którym użytkownik może wybrać plik trnx. Brak nazwy spowoduje otwarcie okna programu AsTrend z zachowaniem poprzednio używanego pliku.

Po nazwie pliku można podać nazwy zmiennych, które zostaną dodane do wykresu(raportu). Jeżeli nazwy zmiennych będą rozdzielone znakami #, to zmienne zostaną dodane. Jeżeli nazwy zmiennych będą rozdzielone znakami <, to zmienne zastąpią zmienne znajdujące się w definicji trendu(raportu).

czas_początku

Określenie czasu początku wyświetlanych danych. Parametr może być wartością typu *DateTime,* czasem bezwzględnym w postaci *yyyymmddhhnnss*, czasem w formacie OPC lub nazwą zmiennej procesowej. W sytuacji, gdy parametrem jest nazwą zmiennej, to jej wartość interpretowana jest jako ilość sekund od 1.1.1970 dla zmiennych o wartościach liczbowych, a dla zmiennych tekstowych jako czas w postaci *yyyymmddhhnnss* lub w formacie OPC.

czas_końca

Określnie czasu końca danych. Podawany w sposób analogiczny jak dla czasu początku. Jeżeli parametr nie zostanie użyty, to długość okresu danych wynika z zakresu zapisanego w pliku definicji.

współrzędna_X

Parametr określa współrzędną X okna programu AsTrend (lewego górnego narożnika). Wartość -1 oznacza pobranie odpowiedniej współrzędnej z definicji trendu.

współrzędna_Y

Parametr określa współrzędną Y okna programu AsTrend (lewego górnego narożnika). Wartość -1 oznacza pobranie odpowiedniej współrzędnej z definicji trendu.

monitor

W parametrze należy podać numer monitora, na którym powinno otworzyć się okno programu AsTrend. Wartość 0 oznacza monitor, na którym aktualnie znajduje się kursor myszki, -1 oznacza monitor wynikający z przyjętych współrzędnych x, y.

Przykłady

AstrendDisplay ("tr1.trnx#z1#z2”, zmienna_startu ,"","","")

Akcja wyświetla trend *tr1.trnx*. Do zmiennych zadeklarowanych w pliku trendu zostaną dodane zmienne *z1* i *z2*. Czas początku trendu zostanie ustalony na podstawie aktualnej wartości zmiennej *zmienna_startu*.

 **Zobacz też**

- akcja *AsTrendPrint*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsTrend*

AstrendPrint

Przeznaczenie

Akcja służy do wydrukowania wybranego trendu lub raportu przy pomocy programu AsTrend. W akcji można podać nazwę wzorca trendu(raportu) i czas początku danych. Opcjonalnie można podać nazwy zmiennych wyświetlonych użytych na wykresie lub w raporcie.

Składnia

AstrendPrint (*nazwa_trendu, czas_początku, tytuł, nagłówek, stopka*)
AstrendPrint (*nazwa_trendu, czas_początku, czas_końca, tytuł, nagłówek, stopka*)

Parametry

nazwa_trendu

Nazwa pliku z definicją trendu(raportu). Wpisanie znaku "?" powoduje otwarcie okna, w którym użytkownik może wybrać plik trnx.

Po nazwie pliku można podać nazwy zmiennych, które zostaną dodane do wykresu(raportu). Jeżeli nazwy zmiennych będą rozdzielone znakami #, to zmienne zostaną dodane. Jeżeli nazwy zmiennych będą rozdzielone znakami <, to zmienne zastąpią zmienne znajdujące się w definicji trendu(raportu).

czas_początku

Określenie czasu początku wyświetlanych danych. Parametr może być wartością typu *DateTime, c*zasem bezwzględnym w postaci *yyyymmddhhnnss*, czasem w formacie OPC lub nazwą zmiennej procesowej. W sytuacji, gdy parametrem jest nazwą zmiennej, to jej wartość interpretowana jest jako ilość sekund od 1.1.1970 dla zmiennych o wartościach liczbowych, a dla zmiennych tekstowych jako czas w postaci *yyyymmddhhnnss* lub w formacie OPC.

czas_końca

Określnie czasu końca danych. Podawany w sposób analogiczny jak dla czasu początku. Jeżeli parametr nie zostanie użyty, to długość okresu danych wynika z zakresu zapisanego w pliku definicji.

tytuł

Parametr określa nowy tytuł wydruku, zastępujący tytuł zdefiniowany w pliku trendu.

nagłówek

Parametr określa nowy nagłówek wydruku, zastępujący nagłówek zdefiniowany w pliku trendu.

stopka

Parametr określa nowy tekst stopki wydruku, zastępujący stopkę zdefiniowaną w pliku trendu.

Przykłady

AstrendPrint (tr1.trnx, "NOW-1H" ,"","","")

Akcja drukuje trend *tr1.trnx* z domyślnymi ustawieniami tytułu, nagłówka i stopki. Drukowany jest trend z ostatniej godziny.

 **Zobacz też**

- akcja *AsTrendDisplay, AsTrendPrintToPdf*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsTrend*

AstrendPrintToPdf

Przeznaczenie

Akcja służy do wyeksportowaniu wybranego trendu lub raportu do pliku typu PDF przy pomocy programu AsTrend. Działanie akcji jest analogiczne d akcji wydruku trendów(raportów) *AstrendPrint*. W akcji można podać nazwę wzorca trendu(raportu) i czas początku danych. Opcjonalnie można podać nazwy zmiennych wyświetlonych użytych na wykresie lub w raporcie.

Składnia

AstrendPrint (*nazwa_trendu, nazwa_pliku, czas_początku, tytuł, nagłówek, stopka*)
AstrendPrint (*nazwa_trendu, nazwa_pliku, czas_początku, czas_końca, tytuł, nagłówek, stopka*)

Parametry

nazwa_trendu

Nazwa pliku z definicją trendu(raportu). Wpisanie znaku "?" powoduje otwarcie okna, w którym użytkownik może wybrać plik trnx.

Po nazwie pliku można podać nazwy zmiennych, które zostaną dodane do wykresu(raportu). Jeżeli nazwy zmiennych będą rozdzielone znakami #, to zmienne zostaną dodane. Jeżeli nazwy zmiennych będą rozdzielone znakami <, to zmienne zastąpią zmienne znajdujące się w definicji trendu(raportu).

nazwa_pliku

Nazwa i ścieżka pliku wynikowego. Jeżeli ścieżka nie zostanie podana, to plik zostanie zapisany w katalogu roboczym aplikacji.

czas_początku

Określenie czasu początku wyświetlanych danych. Parametr może być wartością typu *DateTime, c*zasem bezwzględnym w postaci *yyyymmddhhnnss*, czasem w formacie OPC lub nazwą zmiennej procesowej. W sytuacji, gdy parametrem jest nazwą zmiennej, to jej wartość interpretowana jest jako ilość sekund od 1.1.1970 dla zmiennych o wartościach liczbowych, a dla zmiennych tekstowych jako czas w postaci *yyyymmddhhnnss* lub w formacie OPC.

czas_końca

Określnie czasu końca danych. Podawany w sposób analogiczny jak dla czasu początku. Jeżeli parametr nie zostanie użyty, to długość okresu danych wynika z zakresu zapisanego w pliku definicji.

tytuł

Parametr określa nowy tytuł wydruku, zastępujący tytuł zdefiniowany w pliku trendu.

nagłówek

Parametr określa nowy nagłówek wydruku, zastępujący nagłówek zdefiniowany w pliku trendu.

stopka

Parametr określa nowy tekst stopki wydruku, zastępujący stopkę zdefiniowaną w pliku trendu.



Zobacz też

- akcja *AsTrendDisplay, AsTrendPrint*

- Parametryzacja_aplikacji > *8. Współpraca z programem AsTrend*

Break

Przeznaczenie

Akcja pozwala na warunkowe przerywanie wykonywania akcji złożonej. Gdy testowane wyrażenie jest równe *true,* to akcja zostanie przerwana. Akcję można też używać jako jeden z elementów akcji *Actions*.

Składnia

Break (*warunek*)

Parametry

warunek

Warunek logiczny, którego wartość decyduje o tym, czy należy przerwać wykonywanie akcji złożonej.

Przykłady

Break (Variable(v3)==0)

SetVariable (v2, 0)

SetVariable (v1, 0)

Operacje sterujące zostaną wykonane tylko wtedy, gdy aktualna wartość zmiennej *v3* będzie różna od 0.

 **Zobacz też**

Akcje złożone, akcja *Perform*, akcja *Actions*, akcja *Ask*

CancelControls

Przeznaczenie

Akcja pozwala na anulowanie wybranych oczekujących operacji sterujących.

Niektóre obiekty na diagramach synoptycznych mogą pracować w trybie opóźnionego sterowania. Akcja *CancelControls* powoduje, że te obiekty wrócą do trybu normalnego działania, a operacje sterujące nie zostaną wykonane.

Składnia

CancelControls(*nazwa_obiektu*)

Parametry

nazwa_obiektu

Parametr określa nazwę obiektów, dla których sterowania mają zostać anulowane. W nazwie można stosować znaki specjalne wzorców nazw '*' i '?' .
Parametr może być też jedną z poniższych stałych:

$All - akcja dotyczy wszystkich obiektów we wszystkich oknach i diagramach.
$AllVisible - akcja dotyczy wszystkich obiektów we wszystkich widocznych oknach. Okno jest widoczne wtedy, gdy nie jest przesłonięte w żaden sposób przez jakiekolwiek inne okno.
$Window- akcja dotyczy wszystkich obiektów we wszystkich diagramach bieżącego okna (wynikającego z kontekstu użycia akcji).
$Diagram - akcja dotyczy wszystkich obiektów bieżącego diagramu (wynikającego z kontekstu użycia akcji).

Przykłady

CancelControls($Diagram)

Wszystkie oczekujące operacje sterujące na aktywnym (wynikającym z kontekstu użycia) diagramie zostaną zaniechane.



Zobacz też

- Techniki_budowy_diagramów > *6. Organizacja operacji sterujących*

- akcja *SendControls*

ChangePassword

Przeznaczenie

Akcja powoduje wyświetlenie okna służącego do zmiany hasła aktualnie zalogowanego użytkownika. Hasło można też zmienić w obiekcie typu *Pole autoryzacji* osadzonym na diagramie synoptycznym.

Składnia

ChangePassword()

 **Zobacz też**

- Parametryzacja_aplikacji > *5. System uprawnień*

- Obiekty > obiekt *Pole autoryzacji*

ClearSelection

Przeznaczenie

Akcja powoduje usunięcie selekcji z aktualnie wybranego obiektu diagramu.

Składnia

ClearSelection()

CloseWindow

Przeznaczenie

Akcja służy do zamykania otwartych okien synoptycznych. Możliwe jest grupowe zamykanie okien na podstawie zgodności ze wzorcem nazwy. Zamykanie okien może być ograniczone do wybranego monitora.

Składnia

CloseWindow (*nazwa_okna, wszystkie*)
CloseWindow (*nazwa_okna, wszystkie, monitor*)

Parametry

nazwa_okna

Parametr określa nazwę okien, których akcja dotyczy. Można podać nazwę dokładną lub wzorzec nazwy zawierający znaki specjalne * i ?. Stosowane są następujące zasady wyszukiwania okien:

- gdy nazwa okna jest pusta, to zamykane jest tylko okno wynikające z kontekstu użycia - okno z którego została wywołana akcja *CloseWindow.*
- gdy nazwa jest podana, to przeszukiwanie okien wykonywane jest w poniższej kolejności:

1 - okno kontekstowe
2 - okno aktywne
3 - pozostałe otwarte okna

- okna startowe aplikacji mogą być zamknięte tylko wtedy, gdy podana została dokładna nazwa okna.

wszystkie

Gdy parametr ma wartość *false*, to zamykana jest tylko pierwsze okno spełniające warunki określone parametrami *nazwa_okna* i *monitor*. Jeżeli parametr ma wartość *true*, to zamykane są wszystkie okna spełniające warunki.

monitor

Parametr określa, na którym monitorze okna mają zostać zamknięte. Parametr może przyjmować poniższe wartości:

-1 - akcja dotyczy wszystkich monitorów.

0 - zamykane są okna na monitorze, na którym znajduje się kursor myszki.

>0 - numer monitora zgodny z numeracją systemową - akcja dotyczy tylko podanego monitora.

Przykłady

CloseWindow ("A*”, true, 0)

Akcja zamyka wszystkie okna, których nazwy zaczynają się od znaku A i znajdują się na monitorze wskazywanym przez położenie kursora myszki.



Zobacz też

- akcja *OpenWindow*, akcja *OpenDiagram*, akcja *MinimizeWindow*

- Techniki_budowy_diagramów > *5. Zasady otwierania i zamykania okien synoptycznych*

ConfirmRole

Przeznaczenie

Akcja służy do sprawdzenia, czy użytkownik jest przypisany do sprawdzanej roli. Akcja jest przeznaczona do użycia w akcjach złożonych. W zależności od trybu sprawdzane są uprawnienia aktualnie zalogowanego użytkownika lub wyświetlane jest okno logowania, które wymusza identyfikację użytkownika w momencie wykonania akcji.

Wariant z wymuszaną autoryzacją pozwala wykonać akcję złożoną na prawach innego użytkownika niż aktualny. Akcja nigdy nie powoduje zmiany aktualnego użytkownika.
W przypadku błędnej autentyfikacji lub gdy użytkownik nie posiada wskazanej roli wykonywanie kolejnych akcji w akcji złożonej jest przerywane.
Akcję można też używać jako jeden z elementów akcji *Actions*.

Składnia

ConfirmRole(*nazwa_roli, tryb_weryfikacji*)

Parametry

nazwa_roli

Nazwa roli, przynależność do której jest weryfikowana.

tryb_weryfikacji

Parametr określający sposób weryfikacji uprawnień użytkownika. Możliwe są następujące warianty:

$Always - potwierdzenie tożsamości wymagane jest za każdym razem. Zapytanie o identyfikator i hasło nastąpi nawet wtedy, gdy aktualnie zalogowany użytkownik jest przypisany do sprawdzanej roli.

$Never- potwierdzenie tożsamości nigdy nie jest wykonywane. Akcja sprawdza jedynie, czy bieżący użytkownik jest przypisany do sprawdzanej roli.

$IfRequired - Potwierdzenie tożsamości wykonane będzie tylko wtedy, gdy aktualnie zalogowany użytkownik nie jest przypisany do danej roli.

Przykłady

ConfirmRole ("SuperOperator",$Never)

SetVariable (v2, 0)

SetVariable (v1, 0)

W akcji złożonej operacje sterujące zostaną wykonane tylko wtedy, gdy aktualnie zalogowany użytkownik jest przypisany do roli *SuperOperator*. Okno logowania nie będzie wyświetlone.

ConfirmRole ("SuperOperator",$Always)

SetVariable (v2, 0)

SetVariable (v1, 0)

Niezależnie od uprawnień aktualnie zalogowanego użytkownika, nastąpi weryfikacja użytkownika. Żeby dalsza część akcji została wykonana, należy podać identyfikator i hasło użytkownika, który jest przypisany do roli *SuperOperator*.

 **Zobacz też**

- Parametryzacja_aplikacji > *5. System uprawnień*

- Techniki_budowy_diagramów > *6.5. Kontrola uprawnień*

CopyWindow

Przeznaczenie

Akcja służy od otwarcia kopii aktualnie otwartego okna synoptycznego. Tworzone jest bliźniacze okno, dla którego we wszystkich panelach wyświetlany jest aktualnie używany diagram (łącznie z aktualnymi wartościami parametrów).

Składnia

CopyWindow (*nazwa_okna, współrzędna_X, współrzędna_Y*)

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji jest oknem, z którego akcja została użyta. Jeżeli nazwa jest jawnie podana, to takie okno musi być aktualnie otwarte.

współrzędna_X

Parametr określa współrzędną X otwieranego okna (lewego górnego narożnika). Parametr może być bezpośrednią współrzędną liczbową lub jedną z poniższych stałych:

$Default - współrzędna X wynika z definicji okna.

$CursorLocation - współrzędną X określa aktualne położenie kursora myszki. Współrzędna może zostać zmodyfikowana, aby zapobiec częściowemu wyjściu otwieranego okna poza obszar monitora.

$ActiveLocation - współrzędna X otwieranego okna jest identyczna jak współrzędna X okna, które było aktywne w momencie wykonania akcji.

współrzędna_Y

Parametr określa współrzędną Y otwieranego okna (lewego górnego narożnika). Zasady podawania parametru są identyczne jak dla parametru *współrzędna_X*.

EditPatterns

Przeznaczenie

Uruchamia edytor przebiegów wzorcowych. Edytor krzywych wzorcowych uruchomiony przy pomocy akcji *EditPatterns* pozwala na edycję tylko krzywych lokalnych.

Składnia

EditPatterns()

EditTimetables

Przeznaczenie

Akcja otwiera okno edycji cykli harmonogramów. Okno pozwala operatorowi na zmiany w cyklach realizacji harmonogramu w trakcie wykonywania aplikacji. Akcja zostanie wykonana pod warunkiem, że aktualnie zalogowany operator jest uprawniony do edycji cykli harmonogramów.

Składnia

EditTimetables ()

EditTimetables (identyfikator_harmonogramu)

Parametry

identyfikator_harmonogramu

Parametr określa harmonogram, którego cykle mają być edytowane. Możliwe jest użycie znaków specjalnych * i ?. Pozwala to jednoczesną edycję cykli dla grupy harmonogramów spełniających warunek wyboru. W wariancie akcji bez podania identyfikatora harmonogramu, w oknie wyświetlane i edytowane są cykle dla wszystkich harmonogramów.

EndAlarm

Przeznaczenie

Akcja służy do zgłoszenia zdarzenia zakończenia alarmu. Alarm będący obiektem akcji musi należeć do strategii zewnętrznej lub niezdefiniowanej. Alternatywną metodą zgłoszenia zakończenia alarmu jest użycie w funkcji *Cancel* interfejsu *IAlarm* w kodzie skryptu użytkownika.

Składnia

EndAlarm(*nazwa_domeny, identyfikator_alarmu*)
EndAlarm(*identyfikator_alarmu, identyfikator_alarmu, parametr_1, ...*)

Parametry

nazwa_domeny

Nazwa domeny alarmów, której dotyczy akcja. Jeżeli nazwa nie jest podana, to akcja dotyczy domeny domyślnej (pierwszej).

identyfikator_alarmu

Identyfikator alarmu, który ma zostać zakończony.

parametr_1, ..., parametr_n

Dodatkowe parametry zgłoszenie rejestrowane w logu alarmów.



Zobacz też

- System_alarmów > *Strategie wykrywania alarmów*

- Skrypty > *Interfejs IAlarm*

- akcja *StartAlarm*

- System_alarmów

ExcludeAlarm

Przeznaczenie

Akcja służy do wykluczenie alarmu z obsługi. Jeżeli alarm jest wykluczony, to ignorowane są wszystkie kolejne zdarzenia początku i końca alarmu. Alarmy mogą być także wykluczane z wykorzystaniem obiektu *Tabela alarmów aktywnych*.

Składnia

ExcludeAlarm(*nazwa_domeny, identyfikator_alarmu*)

Parametry

nazwa_domeny

Nazwa domeny alarmów, której dotyczy akcja. Jeżeli nazwa jest pusta, to akcja dotyczy domeny domyślnej (pierwszej).

identyfikator_alarmu

Identyfikator alarmu, który ma zostać wykluczony.

 **Zobacz też**

- System alarmów

- Obiekty > obiekt *Tabela alarmów aktywnych*

- akcja *IncludeAlarm*

Execute

Przeznaczenie

Akcja wykonuje inną akcję podaną w parametrze wywołania. Akcja pozwala na wykonywanie akcji definiowanych w bazie definicji zmiennych lub budowanych w zależności od stanu aplikacji. W tym drugim przypadku akcję *Execute* należy stosować tylko wtedy, gdy typ akcji nie jest z góry znany.

Składnia

Execute(*akcja*)

Parametry

akcja

Dynamicznie tworzona treść akcji, którą należy wykonać.

Przykłady

Execute(Attribute(Akcja))

W powyższym przykładzie treść akcji pobierana jest z atrybutu *Akcja* zmiennej kontekstowej (zmiennej głównej obiektu, dla którego akcja *Execute* została wykonana w reakcji na jakieś zdarzenie).

Execute ("SetVariable (v1," + Variable(v1)==0?1:0)

W tym przypadku tworzona jest dynamicznie akcja *SetVariable*, o treści zależnej od wartości zmiennej *v1*. Chociaż konstrukcja jest poprawna, to zalecane jest identycznie działająca prostsza forma:

SetVariable (v1,Variable(v1)==0?1:0)

Execute (Variable(v1)==0? "SetVariable (v1,1)":"Nothing()")

W zależności od wartości zmiennej v1 wykonywana jest akcja SetVariable lub Nothing. Identyczny efekt można uzyskać przy pomocy akcji Perform:

Perform (Variable(v1)==0, SetVariable (v1,1), Nothing())

 **Zobacz też**

akcja *Nothing*, akcja *Perform*

GetVarNames

Przeznaczenie

Akcja zapisuje do schowka nazwy zmiennych używanych w obiektach położonych pod aktualną pozycją kursora myszki. Nazwy zmiennych ze schowka mogą być następnie użyte w programie AsTrend jako źródło wyboru zmiennych.

Wykrywane są nazwy zmiennych podane we właściwościach o charakterze nazwy zmiennej oraz zmienne, do których odwołanie nastąpiło poprzez prefiksy właściwości # i &.

Akcja jest typowo używana jako akcja wykonawcza globalnego skrótu klawiszowego.

Składnia

GetVarNames ()

GisGoTo

Przeznaczenie

Akcja przesuwa środek diagramu GIS do żądanych współrzędnych geograficznych oraz opcjonalnie ustawia współczynnik powiększenia.

Składnia

GisGoTo(nazwa_okna, nazwa_panela, szerokość geograficzna, długość geograficzna)

GisGoTo(nazwa_okna, nazwa_panela, szerokość geograficzna, długość geograficzna, współczynnik_powiększenia)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panela, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panela kontekstowego.

szerokość_geograficzna

Szerokość geograficzna nowej pozycji diagramu GIS

długość_geograficzna

Długość geograficzna nowej pozycji diagramu GIS

współczynnik_powiększenia

Wartość nowego współczynnika powiększenia (liczba z zakresu od 2 do 25).

GisHideLayer

Przeznaczenie

Akcja wyłącza widoczność wskazanej warstwy diagramu GIS.

Składnia

GisHideLayer(nazwa_okna, nazwa_panela, nazwa_warstwy)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panela, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panela kontekstowego.

nazwa_warstwy

Nazwa warstwy diagramu GIS, która zostanie ukryta.

GisShowLayer

Przeznaczenie

Akcja włącza widoczność wskazanej warstwy diagramu GIS.

Składnia

GisShowLayer(nazwa_okna, nazwa_panela, nazwa_warstwy)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panelu, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panelu kontekstowego.

nazwa_warstwy

Nazwa warstwy diagramu GIS, która zostanie pokazana.

GisSwitchLayer

Przeznaczenie

Akcja przełącza widoczność wskazanej warstwy diagramu GIS. Jeżeli warstwa jest ukryta, to zostanie pokazana. Jeżeli jest pokazywana, to zostanie ukryta.

Składnia

GisSwitchLayer(nazwa_okna, nazwa_panela, nazwa_warstwy)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panelu, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panelu kontekstowego.

nazwa_warstwy

Nazwa warstwy diagramu GIS, której widoczność zostanie przełączona.

GisRefreshLayer

Przeznaczenie

Akcja służąca do ręcznego odświeżenia (przeładowania) zawartości warstwy diagramu GIS.

Składnia

GisRefreshLayer(nazwa_okna, nazwa_panela, nazwa_warstwy)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panelu, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panelu kontekstowego.

nazwa_warstwy

Nazwa warstwy diagramu GIS, która ma zostać odświeżona.

GisZoom

Przeznaczenie

Akcja powoduje ustawienie współczynnika powiększenia diagramu GIS na zadaną wartość.

Składnia

GisZoom(nazwa_okna, nazwa_panela, współczynnik_powiększenia)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panelu, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panelu kontekstowego.

współczynnik_powiększenia

Wartość nowego współczynnika powiększenia (liczba z zakresu od 2 do 25).

GisZoomIn

Przeznaczenie

Akcja powoduje zwiększenie współczynnika powiększenia diagramu GIS o zadaną wartość.

Składnia

GisZoomIn(nazwa_okna, nazwa_panela, zmiana_współczynnika)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panelu, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panelu kontekstowego.

zmiana_współczynnika

Wartość, o jaką zostanie zwiększony współczynnik powiększenia.

GisZoomOut

Przeznaczenie

Akcja powoduje zmniejszenie współczynnika powiększenia diagramu GIS o zadaną wartość.

Składnia

GisZoomOut(nazwa_okna, nazwa_panela, zmiana_współczynnika)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panelu

Nazwa panelu, w którym otwarty jest diagram GIS, którego dotyczy wykonanie akcji. Pusta nazwa oznacza odwołania do panelu kontekstowego.

zmiana_współczynnika

Wartość, o jaką zostanie zmniejszony współczynnik powiększenia.

HidePanel

Przeznaczenie

Akcja powoduje ukrycie wskazanego lub bieżącego panelu w oknie

Składnia

HidePanel()

HidePanel(nazwa_panela)

HidePanel(nazwa_okna, nazwa_panela)

Parametry

nazwa_okna

Nazwa okna, w którym ma zostać ukryty panel.

nazwa_panelu

Nazwa panelu, który ma zostać ukryty.

IncludeAlarm

Przeznaczenie

Akcja służy do ponownego włączenia do obsługi alarmu wcześniej wykluczonego. Alarmy mogą być także włączane od obsługi z wykorzystaniem obiektu *Tabela alarmów aktywnych*.

Składnia

IncludeAlarm(*nazwa_domeny, identyfikator_alarmu*)

Parametry

nazwa_domeny

Nazwa domeny alarmów, której dotyczy akcja. Jeżeli nazwa jest pusta, to akcja dotyczy domeny domyślnej (pierwszej).

identyfikator_alarmu

Identyfikator alarmu, który ma zostać włączony do obsługi.

 **Zobacz też**

- System_alarmów

- Obiekty > obiekt *Tabela alarmów aktywnych*

- akcja *ExcludeAlarm*

Login

Przeznaczenie

Akcja powoduje wyświetlenie okna służącego do zalogowania użytkownika. Inne dostępne metody logowania to:

- Okno panelu kontrolnego, które można w każdej aplikacji otworzyć przy pomocy kombinacji klawiszy *Ctrl-Shift-F1*
- W obiekcie typu *Pole autoryzacji* osadzonym na diagramie synoptycznym

Składnia

Login()

 **Zobacz też**

- Obiekty > obiekt *Pole autoryzacji*

- Parametryzacja_aplikacji > *System uprawnień*

Logout

Przeznaczenie

Akcja wylogowuje aktualnego użytkownika. Inne dostępne metody wylogowania to:

- Okno panelu kontrolnego, które można w każdej aplikacji otworzyć przy pomocy kombinacji klawiszy *Ctrl-Shift-F1*

- W obiekcie typu *Pole autoryzacji* osadzonym na diagramie synoptycznym

Składnia

Logout()

 **Zobacz też**

- Obiekty > obiekt *Pole autoryzacji*

- Parametryzacja_aplikacji > *System uprawnień*

Message

Przeznaczenie

Akcja pozwala na zapisanie w dzienniku zdarzeń aplikacji dowolnego komunikatu. Zarejestrowane komunikaty są widoczne w panelu komunikatów i w obiektach typu *Dziennik zdarzeń*.

Składnia

Message(*typ_komunikatu, źródło_komunikatu, treść_komunikatu*)

Parametry

typ_komunikatu

Parametr określa ważność komunikatu. Możliwe są poniższe wartości:

$Info - Komunikat informacyjny

$Error - Komunikat informujący o wystąpieniu błędu

$Warning - Komunikat ostrzegawczy

źródło_komunikatu

Krótki tekst identyfikujący źródło, które utworzyło komunikat.

treść_komunikatu

Dowolny tekst stanowiący treść komunikatu.

Przykłady

Message($Info, "Test", "Akcja wykonana dla użytkownika" + Variable(CurrentUser))

Akcja zapisuje do dziennika zdarzeń komunikat informacyjny, w którym przekazuje nazwę aktualnie zalogowanego użytkownika.



Zobacz też

- Obiekty > Obiekt *Dziennik zdarzeń*

MinimizeWindow

Przeznaczenie

Akcja służy do zminimalizowania otwartych okien synoptycznych. Możliwa jest grupowa minimalizacja okien na podstawie zgodności z wzorcem nazwy.

Składnia

MinimizeWindow(*nazwa_okna*)

Parametry

nazwa_okna

Parametr określa nazwę okien, których akcja dotyczy. Można podać nazwę dokładną lub wzorzec nazwy zawierający znaki specjalne * i ?. Gdy nazwa okna jest pusta, to minimalizowane jest tylko okno wynikające z kontekstu użycia - okno z którego została wywołana akcja *MinimizeWindow.*

Przykłady

MinimizeWindow("")

Akcja minimalizuje okno w kontekście, którego została użyta.



Zobacz też

- akcja *OpenWindow*, akcja *CloseWindow*, akcja *OpenDiagram*

- Techniki_budowy_diagramów > *Zasady otwierania i zamykania okien synoptycznych*

NextDiagram

Przeznaczenie

Akcja służy od przełączenia diagramu w wybranym panelu okna synoptycznego. Przełączenie bazuje na zapamiętanej historii wcześniej używanych diagramów. Akcja ma sens tylko w przypadku wcześniejszego użycia akcji *PreviousDiagram* dla wybranego panelu. Akcja *NextDiagram* wyświetla następny diagram z historii, odwracając działanie akcji *PreviousDiagram*.

Składnia

PreviousDiagram (*nazwa_okna*, *nazwa_panelu*)

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji ustalane jest na podstawie kontekstu użycia.

nazwa_panelu

Nazwa panelu, w którym diagram ma zostać przełączony. Może być pusta, a wtedy panel będący obiektem akcji ustalany jest na podstawie kontekstu użycia.

 **Zobacz też**

- akcja *OpenWindow* , akcja *PreviousDiagram*

Nothing

Przeznaczenie

Akcja nie wykonuje żadnych czynności. Przeznaczona jako pusta alternatywa w konstrukcjach wykorzystujących akcje *Perform* i *Execute*.

Składnia

Nothing()

Przykłady

Perform (HasRole(”SuperOperator”), SetVariable (v, 0), Nothing())

Jeżeli zalogowany użytkownik jest przypisany do roli *SuperOperator*, to zostanie wykonana operacja sterująca. Jeżeli nie, to nic nie zostanie wykonane.

Execute(Attribute(Akcja))

W powyższym przykładzie treść akcji pobierana jest z atrybutu *Akcja* zmiennej kontekstowej. Jeżeli dla niektórych zmiennych nie należy wykonywać żadnych operacji, to w bazie zmiennych trzeba użyć akcji *Nothing*.



Zobacz też

akcja *Execute*, akcja *Perform*

ObjectAction

Przeznaczenie

Akcja pozwala na wykonanie operacji na wybranych obiektach diagramu. Rodzaje dostępnych operacji zależą od typu wybranych obiektów.

Składnia

ObjectAction (*nazwa_obiektu, rodzaj_operacji, parametr1, parametr2,* ...)

Parametry

nazwa_obiektu

Nazwa obiektów, na których należy wykonać żądaną operację. W parametrze można używać znaków '*' i '?'.

W przypadku użycia akcji w kontekście obiektu będącego częścią wzorca wyszukiwanie obiektów zgodnych z podaną nazwą dotyczy standardowo tylko obiektów tego wzorca. Odwołanie do obiektu nie będącego częścią wzorca wymaga dodania prefiksu '!' do nazwy obiektu.

rodzaj operacji

Stała określająca rodzaj operacji wykonywanej na wskazanych obiektach. Dozwolony zestaw operacji zależy od typu obiektów.

Poniższa tabela opisuje dostępne rodzaje operacji.

Lista rozwijana	
$ColapseAll	Powoduje zwinięcie wszystkich węzłów listy.
$ExpandAll	Powoduje rozwinięcie wszystkich węzłów listy.
$Reload	Ponowne utworzenie listy. Może być użyte w celu ponownego odczytania struktury listy zapisanej w pliku.
Przeglądarka	
$Reload	Przeładowanie strony.
$Print	Otwarcie okienka wydruku strony.
Tabela Asbase	
$First	Przejście do pierwszego wiersza.
$Last	Przejście do ostatniego wiersza.
$Next	Przejście do następnego wiersza.
$NextPage	Przejście do następnej strony.
$Previous	Przejście do poprzedniego wiersza.
$PreviousPage	Przejście do poprzedniej strony.
$Download	Załadowanie wybranej receptury do zestawu zmiennych określonego we właściwości *Identyfikator zestawu zmiennych ładowań* obiektu. Operacja dozwolona tylko dla receptur.
$Reload	Ponowne odczytanie rekordów z bazy danych.
$Add	Dodanie nowego rekordu do tabeli. Wartości pól rekordu zostaną ustawione na podstawie wartości zmiennych z zestawu podanego we właściwości *Identyfikator zestawu zmiennych*.
$Delete	Usunięcie aktywnego rekordu tabeli.
$Update	Zmiana wartości pól aktywnego rekordu tabeli. Wartości pól rekordu zostaną ustawione na podstawie wartości zmiennych z zestawu podanego we właściwości *Identyfikator zestawu zmiennych*.
$Export	Zapis zawartości tabeli w pliku tekstowym w formacie CSV.
Słupek	

$Decrease	Wykonuje sterowanie ustawiające zmienną sterującą obiektu na wartość mniejszą o zadany krok względem aktualnej pozycji wskaźnika. Wielkość kroku określona jest pierwszym dodatkowym parametrem akcji.
$Increase	Wykonuje sterowanie ustawiające zmienną sterującą obiektu na wartość większą o zadany krok względem aktualnej pozycji wskaźnika. Wielkość kroku określona jest pierwszym dodatkowym parametrem akcji.
Tabela zmiennych	
$Export	Zapis zawartości tabeli w pliku tekstowym w formacie CSV.
Tekst	
$Decrease	Wykonuje sterowanie ustawiające zmienną sterującą obiektu na wartość mniejszą o zadany krok względem aktualnie wyświetlanej wartości. Wielkość kroku określona jest pierwszym dodatkowym parametrem akcji.
$Increase	Wykonuje sterowanie ustawiające zmienną sterującą obiektu na wartość większą o zadany krok względem aktualnie wyświetlanej wartości. Wielkość kroku określona jest pierwszym dodatkowym parametrem akcji.
Wykres	
$SetTimeRange	Otwiera okienko wyboru zakres osi czasu.
$ZoomInX	Zwiększenie dokładności w osi czasu.
$ZoomOutX	Zmniejszenie dokładności w osi czasu.
$ZoomInY	Zwiększenie dokładności w osi wartości.
$ZoomOutY	Zmniejszenie dokładności w osi wartości.
$MoveLeft	Przesunięcie wykresu w lewo.
$MoveRight	Przesunięcie wykresu w prawo.
$PageLeft	Przesunięcie wykresu w lewo o stronę.
$PageRight	Przesunięcie wykresu w prawo o stronę.
$MoveUp	Przesunięcie wykresu w górę.
$MoveDown	Przesunięcie wykresu w dół..

$PageUp	Przesunięcie wykresu w górę o stronę.
$PageDown	Przesunięcie wykresu w dół o stronę.
$CursorLeft	Przesunięcie kursora w lewo.
$CursorRight	Przesunięcie kursora w prawo.
$CursorUp	Przesunięcie kursora w górę.
$CursorDown	Przesunięcie kursora w dół.
$ResetX	Reset osi X do ustawień początkowych.
$ResetY	Reset osi Y do ustawień początkowych.
$Reset	Reset osi X i Y do ustawień początkowych.
$CenterCursor	Pozycjonowanie kursora na środku wykresu.
$SwitchCursor	Włączenie/wyłączenie kursora.
$Legend	Wyświetlenie legendy wykresu.
$Help	Wyświetlenie okienka pomocy.

parametr1, ..., parametrN

Wartości parametrów operacji wykonywanej na obiektach. Liczba i postać parametrów użytych w wywołaniu musi być zgodna z wybranym rodzajem operacji.

OpenDiagram

Przeznaczenie

Akcja pozwala na otwarcie diagramu w dynamicznie utworzonym nowym oknie. Jeżeli diagram o podanej nazwie i identycznym zestawie parametrów jest już otwarty, to nastąpi jedynie aktywizacja wcześniej utworzonego okna.

Składnia

OpenDiagram (*nazwa_diagramu, parametry, tytuł, tryb_otwarcia, współrzędna_X, współrzędna_Y, atrybuty , tryb_wymiany*)

Parametry

nazwa_diagramu

Parametr określa nazwę diagramu, który ma zostać wyświetlony.

parametry

Parametr służy do przekazania do otwieranego diagramu wartości jego parametrów. Zestaw dozwolonych parametrów zależy od definicji diagramu. Parametry są podawane jako sekwencja par *nazwa_parametru*=*wartość_parametru* rozdzielonych znakiem średnika, np.

Kolor=Red;Zmienna=v1

tytuł

Parametr określa tekst, który ma zostać wyświetlony na belce tytułowej okna.

tryb_otwarcia

Parametr określa sposób zachowania się okna po otwarciu. Możliwe wartości to jedna z poniższych stałych:

$Normal - okno zwykłe

$Dialog - okno dialogowe, nie będzie możliwości przejścia do innych otwartych okien aplikacji dopóki okno dialogowe nie zostanie zamknięte

$Temp - okno tymczasowe, w momencie przejścia do dowolnego innego okna nastąpi automatyczne zamknięcie okna otwartego w trybie tymczasowym

współrzędna_X

Parametr określa współrzędną X tworzonego okna (lewego górnego narożnika). Parametr może być bezpośrednią współrzędną liczbową lub jedną z poniższych stałych:

&CursorLocation - współrzędną X określa aktualne położenie kursora myszki. Współrzędna może zostać zmodyfikowana, aby zapobiec częściowemu wyjściu otwieranego okna poza obszar monitora.

&ActiveLocation - współrzędna X otwieranego jest identyczna jak współrzędna X okna, które było aktywne w momencie wykonania akcji.

współrzędna_Y

Parametr określa współrzędną Y tworzonego okna (lewego górnego narożnika). Zasady podawania parametru są identyczne jak dla parametru *współrzędna_X.*

atrybuty

Parametr określa wygląd tworzonego okna. Wartością parametry jest łańcuch tekstowy złożony z poniższych nazw atrybutów rozdzielonych znakiem przecinka.

NoFrame - okno nie będzie posiadało ramki.

NoTitleBar- okno nie będzie posiadało belki tytułowej.

NoButtons - na belce tytułowej okna nie będą wyświetlane przyciski systemowe.

NoMinButton - na belce tytułowej nie będzie wyświetlany przycisk minimalizacji.

NoMaxButton - na belce tytułowej nie będzie wyświetlany przycisk maksymalizacji.

FixedSize - nie będzie można zmienić rozmiaru okna.

ToolWindow - okno będzie wyświetlone w stylu systemowego okna narzędziowego (m.in. mniejsza wysokość belki tytułowej).

TopMost- okno będzie wyświetlane w trybie "zawsze na wierzchu", czyli zawsze będzie przykrywało okna zwykłe.

NoTaskBar - okno nie będzie pokazywane na systemowym pasku zadań.

Maximized - okno zostanie wyświetlone na całym obszarze monitora wynikającego ze współrzędnych x i y.

Center - okno zostanie wyśrodkowane na monitorze wynikającym ze współrzędnych x i y.

KeepAspectRadio - zapewnia zachowanie proporcji okna przy zmianie jego wielkości przez użytkownika.

FitToScreen - zapewnia dopasowanie okna do wielkości ekranu. Jeżeli okno jest większe od ekranu, to zostanie odpowiednio zmniejszone. Podobnie, jeżeli wystaje poza krawędź ekranu, to zostanie przesunięte. Opcja jest przydatna, gdy aplikacja jest uruchamiana na komputerach o różnych rozmiarach ekranów.

tryb_wymiany

Parametr określa jakie dodatkowe akcje są podejmowane po otwarciu okna. Możliwe wartości to jedna z poniższych stałych:

$None - żadne dodatkowe operacje nie są wykonywane.

$Closable - zamknięte zostaną wszystkie otwarte okna, które nie posiadają blokady zamykania i znajdują się na monitorze nowo otwartego okna.

$ExceptCurrent - zamknięte zostaną wszystkie otwarte okna, które nie posiadają blokady zamykania i znajdują się na monitorze nowo otwartego okna za wyjątkiem okna aktywnego w momencie wykonania akcji.

$CurrentOnly - zamknięte zostanie tylko okno aktywne w momencie wykonania akcji.

$ReplaceCurrent - zamknięte zostanie okno aktywne w momencie wykonania akcji. Współrzędne i rozmiar okna otwieranego zostaną ustawione identycznie jak dla okna zamykanego (parametry współrzędna_X i współrzędna_Y są ignorowane).

Przykłady

OpenDiagram (diag1,"pokoj=11”, null, $Dialog, $CursorLocation, $CursorLocation, "NoTitleBar,FixedSize”, $None)

Akcja otwiera w nowym oknie diagram *diag1,* przekazując mu wartość parametru *pokoj*. Okno jest typu dialogowego, a pozycja wynika z aktualnego położenia kursora myszki. Belka tytułowa nie jest pokazywana, a okno nie można zmienić swojego rozmiaru. Nie są zamykane żadne inne okna aplikacji.



Zobacz też

- akcja *OpenWindow*, akcja *CloseWindow*, akcja *MinimizeWindow*

- AsixEvo_Techniki_budowy_diagramów > *5. Zasady otwierania i zamykania okien synoptycznych*

OpenWindow

Przeznaczenie

Akcja służy od otwierania predefiniowanych okien synoptycznych lub do wymiany diagramów wyświetlanych w panelach okna. Jest to podstawowa akcja służącą do stworzenia schematu nawigacji w aplikacji.

Składnia

OpenWindow (*nazwa_okna, nazwa_panelu, nazwa_diagramu, parametry, nowe_okno, współrzędna_X, współrzędna_Y, tryb_wymiany*)

OpenWindow (*nazwa_okna, nazwa_panelu, nazwa_diagramu, parametry, nowe_okno*)

OpenWindow (*nazwa_okna, nazwa_panelu, nazwa_diagramu, parametry*)

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji ustalane jest na podstawie kontekstu użycia. Szczegóły użycia w sekcji *Algorytm*.

nazwa_panelu

Nazwa panelu, w którym diagram ma zostać otwarty. Może być pusta, a wtedy panel będący obiektem akcji ustalany jest na podstawie kontekstu użycia. Szczegóły użycia w sekcji *Algorytm*.

nazwa_diagramu

Nazwa diagramu, którego akcja dotyczy. Może być pusta, a wtedy akcja powoduje jedynie otwarcie okna bez zmiany diagramu lub z otwarciem domyślnych diagramów paneli. Szczegóły użycia w sekcji *Algorytm*.

parametry

Parametr służy do przekazania do otwieranego diagramu wartości jego parametrów. Zestaw dozwolonych parametrów zależy od definicji diagramu. Parametry są podawane jako sekwencja par *nazwa_parametru=wartość_parametru* rozdzielonych znakiem średnika, np.

Kolor=Red;Zmienna=v1

Jeżeli w oknie będącym obiektem akcji jest używane menu, to parametry akcji są także przekazywane do tego menu.

nowe_okno

Parametr typu logicznego *true/false*, który określa, czy należy utworzyć nowe okno. Szczegóły użycia w sekcji *Algorytm*.

współrzędna_X

Parametr określa współrzędną X otwieranego okna (lewego górnego narożnika). Parametr może być bezpośrednią współrzędną liczbową lub jedną z poniższych stałych:

$Default - współrzędna X wynika z definicji okna lub (dla okna już otwartego) jest to bieżąca pozycja. Jest to wartość domyślna w przypadku użycia wariantu akcji bez parametru *współrzędna_X*.

$CursorLocation - współrzędną X określa aktualne położenie kursora myszki. Współrzędna może zostać zmodyfikowana, aby zapobiec częściowemu wyjściu otwieranego okna poza obszar monitora.

$ActiveLocation - współrzędna X otwieranego okna jest identyczna jak współrzędna X okna, które było aktywne w momencie wykonania akcji.

współrzędna_Y

Parametr określa współrzędną Y otwieranego okna (lewego górnego narożnika). Zasady podawania parametru są identyczne jak dla parametru *współrzędna_X*.

tryb_wymiany

Parametr określa dodatkowe akcje, które są podejmowane po otwarciu okna. Możliwe wartości to jedna z poniższych stałych:

$None - żadne dodatkowe operacje nie są wykonywane (wartość domyślna).

$Closable - zamknięte zostaną wszystkie otwarte okna, które nie posiadają blokady zamykania i znajdują się na monitorze otwieranego okna.

$ExceptCurrent - zamknięte zostaną wszystkie otwarte okna, które nie posiadają blokady zamykania i znajdują się na monitorze otwieranego okna za wyjątkiem okna aktywnego w momencie wykonania akcji.

$CurrentOnly - zamknięte zostanie tylko okno aktywne w momencie wykonania akcji.

$ReplaceCurrent - zamknięte zostanie okno aktywne w momencie wykonania akcji. Współrzędne i rozmiar okna otwieranego zostaną ustawione identycznie jak dla okna zamykanego (parametry współrzędna_X i współrzędna_Y są ignorowane).

Algorytm

Jeżeli nazwa diagramu nie jest podana:

Jeżeli nazwa okna też nie została podana, to obiektem akcji jest okno wynikające z kontekstu użycia lub okno bieżące.
Jeżeli nazwa okna nie została podana, ale została podana nazwa panelu, a kontekst wywołania nie jest znany (np. wywołanie ze

skrótu klawiszowego) to: jeżeli panel nie jest częścią okna bieżącego(aktywnego), to jest wyszukiwany we wszystkich otwartych oknach (pierwszeństwo mają okna na ekranie, na którym jest kursor myszki).

Jeżeli nazwa okna została podana i parametr *nowe_okno* jest równy *true*, to otwierane jest nowe okno o podanej nazwie z domyślnymi diagramami.

Jeżeli nazwa okna została podana i parametr *nowe_okno* jest równy *false*, to wyszukiwane jest wśród otwartych okien okno o podanej nazwie (zaczynając od okna aktywnego). Jeżeli takie okno zostanie znalezione, to jest ono obiektem akcji. Jeżeli nie, to otwierane jest nowe okno o podanej nazwie z domyślnymi diagramami.

Jeżeli nazwa diagramu została podana:

- Jeżeli nazwa okna nie została podana, to w oknie kontekstowym (wynikającym z miejsca użycia akcji *OpenWindow*) i w podanym panelu (lub w panelu kontekstowym) otwierany jest podany diagram.
- Jeżeli nazwa okna została podana, to sprawdzane jest, czy wśród otwartych okien o podanej nazwie istnieje okno, które ma już otwarty diagram określony w wywołaniu akcji wraz z identycznym zestawem parametrów (w kolejności wyszukiwania: okno kontekstowe, okno bieżące, pozostałe okna). Jeżeli takie okno zostanie znalezione, to zostanie ono oknem aktywnym.
- Jeżeli w poprzednim punkcie właściwego okna nie znaleziono, a parametr *nowe_okno* jest równy *false*, to wyszukiwane jest wśród otwartych okien okno o podanej nazwie (w kolejności: okno kontekstowe, okno bieżące, pozostałe okna). Jeżeli takie okno zostanie znalezione, to w panelu podanym w treści akcji otwierany jest podany diagram. Jeżeli okna nie znaleziono lub parametr *nowe_okno* jest równy *true*, to otwierane jest nowe okno o podanej nazwie. W panelu podanym w treści akcji otwierany jest podany diagram, a w pozostałych panelach otwierane są diagramy domyślne.

W każdym przypadku, jeżeli udało się określić okno będące obiektem akcji, to okno to staje się oknem aktywnym, wykonywana jest w razie konieczności odpowiednia korekta pozycji okna i zamykane są okna wynikające z parametru *tryb_wymiany*.

Przykłady

OpenWindow (start, schemat, silnik, zmienna=s1)

Jeżeli okno o nazwie *start* jest już otwarte, to w panelu *schemat* zostanie otwarty diagram *silnik* z parametrami *zmienna=s1*. Jeżeli taki diagram z identycznymi parametrami był już otwarty, to nastąpi jedynie aktywizacja okna.

Jeżeli okno o nazwie *start* nie jest jeszcze otwarte, to zostanie teraz otworzone. W panelu *schemat* zostanie otwarty diagram *silnik* z parametrami *zmienna=s1*. W pozostałych panelach zostaną otwarte diagramy domyślne.

OpenWindow (null, schemat, silnik, zmienna=s1, false, $Default, $Default, $Closable)

W oknie kontekstowym, w panelu *schemat* zostanie otwarty diagram *silnik* z parametrami *zmienna=s1*. Pozycja okna nie zostanie zmieniona. Dodatkowo zostaną zamknięte wszystkie okna bez blokady zamknięcia.



Zobacz też

- akcja *OpenDiagram*, akcja *CloseWindow*, akcja *MinimizeWindow*

- AsixEvo_Techniki_budowy_diagramów > *5. Zasady otwierania i zamykania okien synoptycznych*

Perform

Przeznaczenie

Akcja pozwala na sprawdzenie warunku logicznego i w zależności od jego wartości wykonanie jednej z dwóch akcji. Akcja *Perform* w niektórych przypadkach może zastąpić akcje złożone z konstrukcją *Break*.

Składnia

Perform(*warunek, akcja_gdy true, akcja_gdy_false*)

Parametry

warunek

Wyrażenie warunkowe, którego wartość decyduje, która z akcji składowych zostanie wykonana.

akcja_gdy_true

Akcja, która zostanie wykonana, gdy parametr *warunek* będzie miał wartość *true*.

akcja_gdy_false

Akcja, która zostanie wykonana, gdy parametr *warunek* będzie miał wartość *false*.

Przykłady

Perform (Variable(AktywnaStrefa)==1, SetVariable(Strefa1,0), PlaySound(NieWolno))

Jeżeli wartość zmiennej *AktywnaStrefa* jest równa 1, to zostanie ustawiona zmienna *Strefa1* na 0. W przeciwnym wypadku zostanie odtworzony dźwięk o nazwie *NieWolno*.



Zobacz też

- *Akcje złożone*, akcja *Break*

PlaySound

Przeznaczenie

Akcja odtwarza jednokrotnie wybrany sygnał dźwiękowy. Sygnał dźwiękowy musi być wcześniej dodany do puli dźwięków aplikacji.

Składnia

PlaySound(*nazwa_dźwięku*)

Parametry

nazwa_dźwieku

Nazwa sygnału dźwiękowego, który należy odtworzyć.

Przykłady

PlaySound(alert)

Akcja odtwarza jednokrotnie plik dźwiękowy, który został zadeklarowany w puli dźwięków pod nazwą *alert*.



Zobacz też

- System_alarmów > *Sygnalizacja dźwiękowa*

- akcja *PlaySoundLooping*, akcja *StopSound*

PlaySoundLooping

Przeznaczenie

Akcja odtwarza wielokrotnie wybrany sygnał dźwiękowy. Odtwarzanie jest powtarzane do momentu wykonania akcji *StopSound* lub do rozpoczęcia odtwarzania innego sygnału dźwiękowego.

Sygnał dźwiękowy musi być wcześniej dodany do puli dźwięków aplikacji.

Składnia

PlaySoundLooping(*nazwa_dźwięku*)

Parametry

nazwa_dźwieku

Nazwa sygnału dźwiękowego, który należy odtworzyć.

Przykłady

PlaySoundLooping(alert)

Akcja odtwarza wielokrotnie plik dźwiękowy, który został zadeklarowany w puli dźwięków pod nazwą *alert*.



Zobacz też

- System_alarmów > *Sygnalizacja dźwiękowa*

- akcja *PlaySound*, akcja *StopSound*

PowerGuardManager

Przeznaczenie

Akcja otwiera okno zarządzania modułu Strażnik mocy.

Składnia

PowerGuardManager ()

PowerGuardMeter

Przeznaczenie

Akcja służy od otwarcia okna wykresu miernika modułu Strażnik mocy.

Składnia

PowerGuardMeter ***(identyfikator)***

Parametry

identyfikator

Identyfikator miernika strażnika mocy, dla którego należy otworzyć okno wykresu.

PreviousDiagram

Przeznaczenie

Akcja służy od przełączenia diagramu w wybranym panelu okna synoptycznego. Przełączenie bazuje na zapamiętanej historii wcześniej używanych diagramów. Wybierany jest poprzednio wyświetlany diagram. Kolejne wykonania akcji powodują coraz głębsze cofanie się w historii diagramów.

Składnia

PreviousDiagram*(nazwa_okna, nazwa_panelu)*

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji ustalane jest na podstawie kontekstu użycia.

nazwa_panelu

Nazwa panelu, w którym diagram ma zostać przełączony. Może być pusta, a wtedy panel będący obiektem akcji ustalany jest na podstawie kontekstu użycia.

Zobacz też

- akcja *OpenWindow*, akcja *NextDiagram*

PrintScreen

Przeznaczenie

Akcja rejestruje obraz wybranego obszaru ekranu. Możliwa jest rejestracja aktywnego okna lub diagramu. Można też zarejestrować kompletny obraz ekranu monitora. Zarejestrowany obraz jest wstawiany do schowka systemowego, drukowany lub zapisywany w pliku. W przypadku zapisu do pliku obraz jest zapisywany w formacie *png* w podkatalogu *Screenshots* katalogu roboczego aplikacji. Nazwa pliku jest tworzona automatycznie na podstawie aktualnej godziny i rodzaju rejestrowanego obrazu.

Składnia

PrintScreen (*obszar, tryb_wydruku*)

PrintScreen (*obszar, tryb_wydruku, tytuł_wydruku*)

PrintScreen (*obszar, tryb_wydruku, tytuł_wydruku, podmiana_kolorów*)

Parametry

obszar

Parametr określa obszar ekranu, który należy zarejestrować. Możliwe jest użycie poniższych wartości:

$Diagram - zarejestrowany zostanie obszar aktywnego diagramu.

$Screen - zarejestrowany zostanie obszar całego ekranu monitora, na którym znajduje się kursor myszki.

$Window - zarejestrowany zostanie obszar aktywnego okna.

nazwa_obiektu - nazwa obiektu umieszczonego na aktywnym diagramie. Zarejestrowany zostanie tylko obszar tego obiektu.

tryb_wydruku

Parametr określa sposób wykonania akcji. Możliwe jest użycie poniższych stałych:

$File - obraz jest zapisywany w pliku.

$Clipboard - obraz jest umieszczany w schowku systemowym.

$Printer - obraz jest drukowany na drukarce. Używana jest systemowa drukarka domyślna. W ustawienia stanowiska można zdefiniować dodatkowe opcje wydruku.

$PrinterFill - obraz jest drukowany na drukarce wypełniając całą stronę bez zachowania proporcji. Używana jest systemowa drukarka domyślna. W ustawienia stanowiska można zdefiniować dodatkowe opcje wydruku.

tytuł_wydruku

Parametr określa tytuł umieszczany w nagłówku wydruku.

podmiana_kolorów

Parametr pozwala na uzyskanie lepszych wydruków, poprzez podmianę niektórych kolorów używanych na ekranie na inne używane w trakcie wydruku. W szczególności, można podmienić ciemne tło diagramu na jaśniejsze. Parametr ma postać listy par *kolor_zastepowany=kolor_zastępujący* rozdzielonych przecinkami. Kolor może być podany przez jego nazwę lub w notacji RGB, na przykład: "Black=White,131;148;180=0;0;0"

Przykłady

```
PrintScreen($Window, $File)
```

Obszar aktywnego okna zostanie zarejestrowany w pliku umieszczonym w podkatalogu *Screenshots* katalogu roboczego aplikacji.

RefreshVariable

Przeznaczenie

Akcja pozwala na wymuszenie odczytu wartości zmiennej poza okresem jej planowanego cyklicznego odświeżania wynikającego z definicji.

Składnia

RefreshVariable(*nazwa zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej procesowej, której wartość należy odświeżyć. Pusta nazwa oznacza użycie zmiennej głównej obiektu wynikającego z kontekstu użycia akcji. Możliwa jest notacja sufiksowa nazwy zmiennej względem zmiennej głównej obiektu kontekstowego.

Przykłady

RefreshVariable ("")

Akcja wymusza odświeżenie wartości zmiennej głównej obiektu na konto, którego akcja jest wykonywana.

ReportGenerate

Przeznaczenie

Akcja służy do wygenerowania treści raportu i/lub jego wyświetlenia. Jest to alternatywa do ręcznej obsługi raportu za pośrednictwem okna zarządzania raportami.

Składnia

ReportGenerate(***identyfikator_raportu, data_raportu, opcje_generacji*****)**

Parametry

identyfikator_raportu

Identyfikator raportu, którego treść należy wygenerować/wyświetlić.

data_raportu

Data dnia, dla którego raport należy wygenerować. Można przekazać wartość typu DateTime, datę w postaci tekstowej lub datę w postaci tekstowej typu OPC (*zobacz opis funkcji OPCTime*).

opcje_generacji

Stała określająca szczegóły wykonania akcji. Możliwe wartości:

$Display - raport jest wyliczany, a następnie wyświetlany

$DisplayOnly - raport jest jedynie wyświetlany (zakłada się, że został wygenerowany wcześniej)

W wariancie wywołania akcji bez podania parametru *opcje_generacji* raport jest wyłącznie generowany (bez pokazywania).

Przykłady

ReportGenerate(RaportDobowy, "NOW”, $Display)

Akcja generuje treść raportu o identyfikatorze RaportDobowy dla dnia dzisiejszego. Po wygenerowaniu treść raportu zostanie wyświetlona.

ReportsManager

Przeznaczenie

Akcja otwiera okno zarządzania raportami skryptowymi.

Składnia

ReportsManager ()

ResetGlobalPropery

Przeznaczenie

Akcja ustawia wartość właściwości globalnej zgodnie z jej oryginalną definicją. Usuwa w ten sposób skutki wcześniejszego użycia akcji *SetGlobalProperty*.

Składnia

ResetGlobalProperty(*nazwa_właściwości*)

Parametry

nazwa_właściwości

Nazwa właściwości globalnej, której wartość należy zresetować.

Restart

Przeznaczenie

Akcja wykonuje pełny restart aplikacji: aktualnie wykonywana aplikacja jest zamykana, a następnie ponownie uruchomiana. Wykonanie akcji wymaga uprawnień do zamykania aplikacji.

Składnia

Restart ()

Run

Przeznaczenie

Akcja służy do uruchomienia zewnętrznego programu. Można uruchamiać program bezpośrednio podany w parametrach lub wykorzystać systemowe kojarzenie rozszerzeń nazw plików z programami.

Składnia

Run(*nazwa_pliku*)
Run(*nazwa_pliku, argumenty*)

Parametry

nazwa_pliku

Parametr określa program, który należy uruchomić. Może to być nazwa pliku typu exe lub nazwa pliku z danymi. W drugim przypadku zostanie uruchomiony program skojarzony w systemie operacyjnym z typem pliku danych.
W poszukiwaniu programu typu exe uwzględniana jest systemowa ścieżka programów.

argumenty

Tekstowy łańcuch argumentów przekazywany do uruchamianego programu. Format i znaczenie parametru jest specyficzne dla programu.

Przykłady

```
Run("plikipdf\plan.pdf")
```

Zostanie uruchomiona zadeklarowana w systemie przeglądarka plików pdf oraz automatycznie otwarty plik *plan.pdf* z podkatalogu *plikipdf* katalogu aplikacji.

Script

Przeznaczenie

Akcja służy do uruchomienia skryptu użytkownika.

Składnia

Script(*nazwa , parametr1, parametr2, ...*)

Parametry

nazwa

Nazwa skryptu użytkownika.

parametr1, ..., parametrN

Wartości parametrów skryptu. Liczba i postać parametrów użytych w wywołaniu musi być zgodna z wymaganiami skryptu.

Przykłady

Script(ObliczZuzycie, v1)

Akcja uruchamia skrypt o nazwie *ObliczZuzycie,* przekazując do niego pojedynczy parametr o treści *v1*.

 **Zobacz też**

- Skrypty

ScriptClose

Przeznaczenie

Akcja służy do zakończenia pracy uruchomionego skryptu użytkownika. Skrypt, który należy zakończyć jest identyfikowany przez nazwę oraz zestaw parametrów z jakimi został uruchomiony. Akcja kończy pracę wszystkich skryptów pasujących do parametrów wywołania.

Składnia

ScriptClose(*nazwa , parametr1, parametr2, ...*)

Parametry

nazwa

Nazwa skryptu użytkownika.

parametr1, ..., parametrN

Wartości parametrów skryptu. Liczba i postać parametrów użytych w wywołaniu. Musi być zgodna z parametrami użytymi w momencie uruchomienia skryptu.

Przykłady

Perform(IsScriptWorking(S1),ScriptClose(S1),Nothing())

Akcja kończy pracę skryptu o nazwie S1 uruchomionego wcześniej bez podania parametrów. Dodatkowo pokazane jest użycie funkcji *IsScriptWorking*, która pozwala sprawdzić, czy taki skrypt jest uruchomiony.



Zobacz też

- Skrypty

ScriptRestart

Przeznaczenie

Akcja służy do restartu (zakończenia i ponownego uruchomienia) pracy uruchomionego wcześniej skryptu użytkownika. Skrypt, który należy zrestartować jest identyfikowany przez nazwę oraz zestaw parametrów z jakimi został uruchomiony. Akcja dotyczy wszystkich skryptów pasujących do parametrów wywołania.

Składnia

ScriptRestart(*nazwa , parametr1, parametr2,...*)

Parametry

nazwa

Nazwa skryptu użytkownika.

parametr1, ..., parametrN

Wartości parametrów skryptu. Liczba i postać parametrów użytych w wywołaniu. Musi być zgodna z parametrami użytymi w momencie uruchomienia skryptu.

 **Zobacz też**

- Skrypty

SecurityBrowser

Przeznaczenie

Akcja powoduje otwarcie okna dialogowego służącego do przeglądania logów gromadzonych przez system zabezpieczeń. Użycie akcji ma sens tylko wtedy, gdy system zabezpieczeń pracuje w trybie centralnym. Dostęp do logów systemu zabezpieczeń może być ograniczony w zależności od uprawnień użytkownika oraz posiadanej licencji systemu Asix.

Składnia

SecurityBrowser(*typ_logu*)

SecurityBrowser(*typ_logu, filtr*)

Parametry

typ_logu

Parametr określa typ logu, którego zawartość należy pokazać w momencie otwarcia okna. Użytkownik będzie mógł przełączyć się interaktywnie na inne typy logów. Parametr może być też jedną z poniższych stałych:

$Messages - przełączenie na log komunikatów systemu zabezpieczeń
$Controls- przełączenie na log wykonanych operacji sterujących
$Actions - przełączenie na log działań wykonanych przez operatora
$Notes - przełączenie na log notatek operatora

filtr

Parametr określa filtr wyświetlanych danych. Dla logu operacji sterujących jest to nazwa zmiennej, a dla logu notatek identyfikator segmentu. W treści parametru można używać znaków specjalnych * i ?.
W przypadku logu działań operatora należy podać rodzaj operacji. Dopuszczalne rodzaje operacji operatora to:

OpenDiagram - otwarcie diagramu
CloseDiagram - zamknięcie diagramu
OpenTrend - otwarcie trendu
CloseTrend - zamknięcie trendu
RunAsReport - wykonanie akcji AsreportDisplay
RunFile - wykonanie akcji Run
ChangeMode - zmiana trybu pracy

 **Zobacz też**

- Parametryzacja_aplikacji > *5. System uprawnień*

SecurityManager

Przeznaczenie

Akcja otwiera okno dialogowe służące do zarządzania uprawnieniami użytkowników w trakcie wykonywania aplikacji.

Składnia

SecurityManager()

 **Zobacz też**

- Parametryzacja_aplikacji > *5. System uprawnień*

SelectPattern

Przeznaczenie

Akcja pozwala na interaktywny wybór krzywej wzorcowej. Nazwa wybranej krzywej jest zapisywana do zmiennej procesowej podanej w wywołaniu akcji.

Składnia

SelectPattern(*nazwa zmiennej*)

SelectPattern(*nazwa zmiennej, filtr_nazwy*)

Parametry

nazwa_zmiennej

Nazwa zmiennej procesowej, do której należy wysłać nazwę krzywej wzorcowej. Pusta nazwa oznacza użycie zmiennej sterowanej obiektu wynikającego z kontekstu użycia akcji. Możliwa jest notacja sufiksowa nazwy zmiennej względem zmiennej głównej obiektu wynikającego z kontekstu.

filtr_nazwy

Filtr nazw przebiegów wzorcowych, które należy wyświetlić w oknie wyboru zmiennej. W filtrze należy stosować znaki dzikich kart * i ?. W wariancie akcji bez parametru *filtr_nazwy* w oknie wyboru wyświetlane są nazwy wszystkich zdefiniowanych krzywych wzorcowych.

SendControls

Przeznaczenie

Niektóre obiekty na diagramach synoptycznych mogą pracować w trybie opóźnionego sterowania. Akcja *SendControls* powoduje, że te obiekty wykonają oczekujące operacje sterujące.

Składnia

SendControls(*nazwa_obiektu*)

Parametry

nazwa_obiektu

Parametr określa nazwę obiektów, dla których sterowania mają zostać wykonane. W nazwie można stosować znaki specjalne wzorców nazw '*' i '?' .
Parametr może być też jedną z poniższych stałych:

$All - akcja dotyczy wszystkich obiektów we wszystkich oknach i diagramach.
$AllVisible - akcja dotyczy wszystkich obiektów we wszystkich widocznych oknach. Okno jest widoczne wtedy, gdy nie jest przesłonięte w żaden sposób przez jakiekolwiek inne okno.
$Window- akcja dotyczy wszystkich obiektów we wszystkich diagramach bieżącego okna (wynikającego z kontekstu użycia akcji).
$Diagram - akcja dotyczy wszystkich obiektów bieżącego diagramu (wynikającego z kontekstu użycia akcji).

Przykłady

SendControls($Window)

Wszystkie oczekujące operacje sterujące na aktywnym (wynikającym z kontekstu użycia) oknie zostaną wykonane.

Zobacz też

- Techniki budowy diagramów > *Organizacja operacji sterujących*

- akcja *CancelControls*

SetAppData

Przeznaczenie

Akcja pozwala ustawić lub zmienić wartość danej aplikacyjnej. Jeżeli dana aplikacyjna o dana nazwie nie istnieje, to zostanie utworzona. Dane aplikacyjne są dostępne tylko lokalnie w obrębie jednego stanowiska aplikacji. Wartość danej aplikacyjnej może być skalarem (np. pojedyncza liczba), tablicą, tabelą, kolekcją lub innym obiektem złożonym. Dane aplikacyjne mogą być wykorzystywane do wymiany dowolnych danych pomiędzy skryptami lub różnymi modułami lub fragmentami aplikacji.

Składnia

SetAppData(*nazwa_danej, wartość*)

Parametry

nazwa_danej

Nazwa danej aplikacyjnej

wartość

Wartość danej aplikacyjnej

Przykłady

SetAppData("Dana1", 10)

Zapisanie liczby 10 jako wartości danej aplikacyjnej.

SetAppData("Dana2", ToTable(13,45,67,34))

Zapisanie tablicy czterech liczb jako wartości danej aplikacyjnej.

SetBounds

Przeznaczenie

Akcja pozwala na zmianę położenia i rozmiaru wybranego elementu diagramu. Służy do realizacji animacji ruchu i rozmiaru obiektów na diagramie. Dodatkowe informacje i przykład użycia znajdują się w opisie akcji *SetPosition.*

Składnia

SetBounds (*nazwa_obiektu, współrzędna_X, współrzędna_Y, szerokość, wysokość*)

SetBounds (*współrzędna_X, współrzędna_Y, szerokość, wysokość*)

Parametry

nazwa_obiektu

Nazwa elementu, którego położenie i rozmiar należy zmienić. Wariant akcji bez podania nazwy dotyczy zmiany położenia i rozmiaru elementu kontekstowego (obiektu, wzorca lub grupy). W parametrze można używać znaków''*' i''?'.

W przypadku użycia akcji w kontekście obiektu będącego częścią wzorca wyszukiwanie obiektów zgodnych z podaną nazwą dotyczy standardowo tylko obiektów tego wzorca. Odwołanie do obiektu nie będącego częścią wzorca wymaga dodania prefiksu '!' do nazwy obiektu.

współrzędna_X

Nowa wartość współrzędnej X kontrolowanego elementu w jednostkach bezwzględnych.

współrzędna_Y

Nowa wartość współrzędnej Y kontrolowanego elementu w jednostkach bezwzględnych.

szerokość

Nowa szerokość kontrolowanego elementu w jednostkach bezwzględnych.

wysokość

Nowa wysokość kontrolowanego elementu w jednostkach bezwzględnych.



Zobacz też

- Techniki_budowy_diagramów > *Animacja ruchu i zmiana rozmiaru obiektów*

- *a*kcje *SetPosition*, *akcja SetSize*

SetGlobalProperty

Przeznaczenie

Akcja pozwala na zmianę wartości właściwości globalnej. Zmiana dotyczy tylko lokalnego stanowiska. Zmiana zostanie zapamiętana - w przypadku restartu aplikacji właściwość będzie w dalszym ciągu posiadała zmienioną wartość. Powrót do oryginalnej definicji wymaga użycia akcji ResetGlobalProperty.

Składnia

SetGlobalProperty(*nazwa_właściwości, wartość*)

Parametry

nazwa_właściwości

Nazwa właściwości globalnej, której wartość należy zmienić.

wartość

Nowa wartość właściwości globalnej.

SetParameter

Przeznaczenie

Akcja pozwala na zmianę wartości wskazanego parametru diagramu kontekstowego.

Składnia

SetParameter (*nazwa_parametru, wartość_parametru*)

Parametry

nazwa_parametru

Nazwa parametru diagramu, którego wartość należy zmienić.

wartość_parametru

Nowa wartość parametru.

SetParameters

Przeznaczenie

Akcja pozwala na zmianę wartości parametrów diagramu kontekstowego.

Składnia

SetParameters (*parametry*)

Parametry

parametry

Parametr określa nowe wartości parametrów. Parametry są podawane jako sekwencja par *nazwa_parametru=wartość_parametru* rozdzielonych znakiem średnika, np.

Kolor=Red;Zmienna=v1

SetPosition

Przeznaczenie

Akcja pozwala na zmianę położenia i rozmiaru wybranego elementu diagramu. Służy do realizacji animacji ruchu obiektów na diagramie. Akcja jest typowo używana w obsłudze zdarzenia *Animacja* obiektu, wzorca lub grupy. Parametry położenia przekazywane do funkcji muszą być wyrażone we współrzędnych bezwzględnych (od 0 do 1 000 000, współrzędna bezwzględna 1 000 000 odpowiada prawej krawędzi diagramu). W przypadku posługiwania się współrzędnymi względnymi (pikselowymi) należy wykonać odpowiednie przeliczenia używając funkcji *RelToAbsX* i *RelToAbsY.*

Składnia

SetPosition (*nazwa_obiektu, współrzędna_X, współrzędna_Y*)

SetPosition (*współrzędna_X, współrzędna_Y*)

Parametry

nazwa_obiektu

Nazwa elementu, którego położenie należy zmienić. Wariant akcji bez podania nazwy dotyczy zmiany położenia elementu kontekstowego (obiektu, wzorca lub grupy). W parametrze można używać znaków '*' i '?'.

W przypadku użycia akcji w kontekście obiektu będącego częścią wzorca wyszukiwanie obiektów zgodnych z podaną nazwą dotyczy standardowo tylko obiektów tego wzorca. Odwołanie do obiektu nie będącego częścią wzorca wymaga dodania prefiksu '!' do nazwy obiektu.

współrzędna_X

Nowa wartość współrzędnej X kontrolowanego elementu w jednostkach bezwzględnych.

współrzędna_Y

Nowa wartość współrzędnej Y kontrolowanego elementu w jednostkach bezwzględnych.

Przykład

SetPosition(RelToAbsX(100+Variable(pos)),"")

Modyfikowana jest tylko współrzędna X elementu diagramu. Zmienna *pos* przekazuje położenie we współrzędnych pikselowych. Po dodaniu offsetu położenia wykonywana jest konwersja na współrzędne bezwzględne.



Zobacz też

- Techniki budowy diagramów > *Animacja ruchu i zmiana rozmiaru obiektów*

- *akcja SetBounds*, *akcja SetSize*

- Wyrażenia_i_funkcje > funkcja *RelToAbsX*

- Wyrażenia_i_funkcje > funkcja *RelToAbsY*

SetProperty

Przeznaczenie

Akcja pozwala na ustawienie dowolnej właściwości dla wskazanego obiektu lub diagramu. Pozwala to na sterowanie wyglądem/zachowaniem aplikacji.

Składnia

SetProperty (*nazwa_obiektu, nazwa_właściwości, wartość*)

SetProperty (*nazwa_właściwości, wartość*)

Parametry

nazwa_obiektu

Nazwa obiektu lub wzorca, którego właściwość należy zmienić. Jeżeli nazwa nie jest podana, to akcja dotyczy zmiany właściwości elementu kontekstowego (obiektu, wzorca, grupy lub diagramu). W parametrze można używać znaków'*' i '?'.

W przypadku użycia akcji w kontekście obiektu będącego częścią wzorca wyszukiwanie obiektów zgodnych z podaną nazwą dotyczy standardowo tylko obiektów tego wzorca. Odwołanie do obiektu nie będącego częścią wzorca wymaga dodania prefiksu '!' do nazwy obiektu.

nazwa_właściwości

Nazwa właściwości, którą należy zmienić. Należy używać nazw wewnętrznych właściwości. Najprostszą metodą wstawienia do treści akcji poprawnej nazwy właściwości jest skorzystanie z komendy *Właściwości lokalne* w menu edytora akcji.

wartość

Nowa wartość zmienianej właściwości.

Przykłady

SetProperty(Temp,MainVar,blok1Temp)

Akcja zmienia główną zmienną obiektu o nazwie *Temp* na wartość *blok1Temp*. Skutkuje to przełączeniem obiektu na wyświetlanie stanu nowej zmiennej procesowej.

SetSelection

Przeznaczenie

Akcja służy do wymuszenia przejścia w stan selekcji dla podanego obiektu diagramu. Może skutkować również wejściem w stan edycji, gdy obiekt jest ustawiony na tryb automatycznego przejścia do edycji po wyselekcjonowaniu.

Akcja użyta w zdarzeniu otwarcia diagramu pozwala na tworzenie formatek ustawiania parametrów z automatycznym rozpoczęciem wprowadzania ustawień.

Składnia

SetSelection (*nazwa_obiektu*)

Parametry

nazwa_obiektu

Nazwa obiektu, który powinien zostać wyselekcjonowany.

W przypadku użycia akcji w kontekście obiektu będącego częścią wzorca, wyszukiwanie obiektów zgodnych z podaną nazwą dotyczy standardowo tylko obiektów tego wzorca. Odwołanie do obiektu niebędącego częścią wzorca wymaga dodania prefiksu '!' do nazwy obiektu.

SetSize

Przeznaczenie

Akcja pozwala na zmianę rozmiaru wybranego elementu diagramu. Służy do realizacji animacji rozmiaru obiektów na diagramie. Dodatkowe informacje i przykład użycia znajdują się w opisie akcji *SetPosition.*

Składnia

SetSize (*nazwa_obiektu, szerokość, wysokość*)

SetSize(*szerokość, wysokość*)

Parametry

nazwa_obiektu

Nazwa elementu, którego rozmiar należy zmienić. Wariant akcji bez podania nazwy dotyczy zmiany rozmiaru elementu kontekstowego (obiektu, wzorca lub grupy). W parametrze można używać znaków '*' i '?'.

W przypadku użycia akcji w kontekście obiektu będącego częścią wzorca, wyszukiwanie obiektów zgodnych z podaną nazwą dotyczy standardowo tylko obiektów tego wzorca. Odwołanie do obiektu niebędącego częścią wzorca wymaga dodania prefiksu '!' do nazwy obiektu.

szerokość

Nowa szerokość kontrolowanego elementu w jednostkach bezwzględnych.

wysokość

Nowa wysokość kontrolowanego elementu w jednostkach bezwzględnych.



Zobacz też

- Techniki budowy diagramów > *Animacja ruchu i zmiana rozmiaru obiektów*

- *akcja SetPosition*, *akcja SetBounds*

SetTableVariable

Przeznaczenie

Akcja pozwala na bezpośrednie wysłanie do wskazanej zmiennej procesowej wartości sterującej w postaci tablicy wartości.

Wykonanie akcji jest kontrolowane przez system uprawnień.

Składnia

SetVariable(*nazwa zmiennej, element1, element2,...*)

Parametry

nazwa_zmiennej

Nazwa zmiennej procesowej, do której należy wysłać sterowanie. Pusta nazwa oznacza użycie zmiennej sterowanej obiektu wynikającego z kontekstu użycia akcji. Możliwa jest notacja sufiksowa nazwy zmiennej względem zmiennej głównej obiektu wynikającego z kontekstu.

element1,... elementN

Kolejne elementy, z których składana jest tablica wartości sterujących wysyłanych do zmiennej.



Zobacz też

- Techniki_budowy_diagramów > *Organizacja operacji sterujących*

SetVariable

Przeznaczenie

Akcja pozwala na bezpośrednie wysłanie do wskazanej zmiennej procesowej wartości sterującej.

Wykonanie akcji jest kontrolowane przez system uprawnień.

Składnia

SetVariable(*nazwa zmiennej, wartość*)
SetVariable(*nazwa zmiennej, wartość, maska_wartości*)

Parametry

nazwa_zmiennej

Nazwa zmiennej procesowej, do której należy wysłać sterowanie. Pusta nazwa oznacza użycie zmiennej sterowanej obiektu wynikającego z kontekstu użycia akcji. Możliwa jest notacja sufiksowa nazwy zmiennej względem zmiennej głównej obiektu wynikającego z kontekstu.

wartość

Wartość sterowania wysyłana do zmiennej. W wariancie akcji bez parametru *maska_wartości* podana wartość jest wysyłana bez żadnej modyfikacji. W drugim wariancie wartość jest modyfikowana zgodnie z algorytmem podanym w opisie parametru *maska_wartości*.

maska_wartości

Jeżeli parametr jest użyty, to w trakcie wykonywania akcji:

- odczytywana jest bieżąca wartość zmiennej;
- w odczytanej wartości zmieniane są tylko te bity, którym w *masce_wartości* odpowiadają bity o wartości 1;
- tak zmodyfikowana wartość wysyłana jest do zmiennej.

Przykłady

SetVariable ("",1)

Zmienna sterowana obiektu, na konto którego wykonywana jest akcja, zostanie ustawiona na wartość 1.

SetVariable("#_S”,8,0xc)

Ustawiona zostanie zmienna, której nazwa powstanie z połączenia nazwy głównej obiektu, na konto którego wykonywana jest akcja, oraz sufiksu "_S”. Bit 3 zmiennej zostanie ustawiony na 1 a bit 2 na 0. Pozostałe bity pozostaną niezmienione. Na przykład, jeżeli wartość oryginalna zmiennej była równa 0xff, to po wykonaniu sterowania będzie równa 0xfb.



Zobacz też

- Techniki_budowy_diagramów > *Organizacja operacji sterujących*

SetWindowSize

Przeznaczenie

Akcja służy do zmiany rozmiaru okien synoptycznych. Możliwa jest grupowa zmiana rozmiaru okien na podstawie zgodności ze wzorcem nazwy.

Akcja jest alternatywą do ręcznej zmiany rozmiaru okna przez użytkownika. Użytkownik może zmienić rozmiar okna, które nie ma założonej blokady i posiada ramkę. Należy pamiętać, że zmiana rozmiaru okna spowoduje zmianę rozmiarów paneli wchodzących w skład okna. Może też nastąpić przeskalowanie diagramów do nowych rozmiarów.

Składnia

SetWindowSize(*nazwa_okna, szerokość, wysokość*)

Parametry

nazwa_okna

Parametr określa nazwę okien, których akcja dotyczy. Można podać nazwę dokładną lub wzorzec nazwy zawierający znaki specjalne * i ?. Gdy nazwa okna jest pusta, to zmieniany jest rozmiar okna wynikającego z kontekstu użycia - okno z którego została wywołana akcja *SetWindowSize.*

Szerokość

Nowa szerokość okna.

Wysokość

Nowa wysokość okna.

Przykłady

SetWindowsSize (null,400,200)

Akcja zmienia rozmiar okna kontekstowego.



Zobacz też

- Techniki_budowy_diagramów > *4.2. Kontrola położenia i rozmiaru okien*

ShowControlPanel

Przeznaczenie

Akcja służy do otwarcia okna panelu kontrolnego. Panel kontrolny to predefiniowane okno pozwalające na wykonanie podstawowych czynności administracyjnych.

Okno panelu można w każdej aplikacji otworzyć także przy pomocy kombinacji klawiszy *Ctrl-Shift-F1*.

Składnia

ShowControlPanel()

ShowKeyboard

Przeznaczenie

Akcja służy do wyświetlenia okna alfanumerycznej klawiatury ekranowej.

Składnia

ShowKeyboard()

 Zobacz też

- Akcja *ShowNumericKeyboard*

ShowMenu

Przeznaczenie

Akcja służy do wyświetlenia wskazanego menu kontekstowego. Menu jest otwierane w miejscu, w którym znajduje się kursor myszki. Alternatywą do menu kontekstowego są menu podłączone na stałe do okien synoptycznych.

Składnia

ShowMenu(*nazwa_menu, parametry*)

Parametry

nazwa_menu

Nazwa menu zdefiniowanego w aplikacji.

parametry

Parametr służy do przekazania do otwieranego menu wartości jego parametrów. Zestaw dozwolonych parametrów zależy od definicji menu. Parametry są podawane jako sekwencja par *nazwa_parametru*=*wartość_parametru* rozdzielonych znakiem średnika, np.

Kolor=Red;Zmienna=v1

 **Zobacz też**

- Elementy_wizualizacji > *5. Menu*

ShowNumericKeyboard

Przeznaczenie

Akcja służy do wyświetlenia okna numerycznej klawiatury ekranowej.

Składnia

ShowNumericKeyboard()

 Zobacz też

Akcja *ShowKeyboard*

Speak

Przeznaczenie

Akcja powoduje głosowe odczytanie dowolnego podanego tekstu przez narratora systemu Windows.

Składnia

Speak(*tekst*)

Parametry

tekst

Treść komunikatu, który zostanie odczytany.

ShowPanel

Przeznaczenie

Akcja powoduje przywrócenie ukrytego wskazanego panelu w oknie.

Składnia

ShowPanel(nazwa_panela)

ShowPanel(nazwa_okna, nazwa_panela)

Parametry

nazwa_okna

Nazwa okna, w którym przywrócony ma zostać ukryty panel.

nazwa_panelu

Nazwa panelu, który ma zostać przywrócony.

StartAlarm

Przeznaczenie

Akcja służy do zgłoszenia zdarzenia rozpoczęcia alarmu. Alarm będący obiektem akcji musi należeć do strategii zewnętrznej lub niezdefiniowanej. Alternatywną metodą zgłoszenia początku alarmu jest użycie w funkcji R*aise* interfejsu *IAlarm* w kodzie skryptu użytkownika.

Składnia

StartAlarm(*nazwa_domeny, identyfikator_alarmu*)
StartAlarm(*nazwa_domeny, identyfikator_alarmu,parametr_1,...*)

Parametry

nazwa_domeny

Nazwa domeny alarmów, której dotyczy akcja. Jeżeli nazwa nie jest podana, to akcja dotyczy domeny domyślnej (pierwszej).

identyfikator_alarmu

Identyfikator alarmu, który ma zostać rozpoczęty.

parametr_1,..., parametr_n

Dodatkowe parametry rejestrowane w logu alarmów.



Zobacz też

- System alarmów

- System_alarmów > *Strategie wykrywania alarmów*

- Skrypty > *Interfejs IAlarm*

- akcja *EndAlarm*

StopSound

Przeznaczenie

Akcja przerywa aktualnie odtwarzany sygnał dźwiękowy. W wersji wywołania z podanym parametrem nazwy domeny alarmowej, akcja powoduje wyciszenie sygnału dźwiękowego

sygnalizacji alarmu na stanowisku wykonania akcji oraz na innych stanowiskach uczestniczących w pracy domeny.

Składnia

StopSound()
StopSound(*nazwa_domeny*)

Parametry

nazwa_domeny

Nazwa domeny alarmów, której dotyczy akcja. Można podać nazwę '*', co oznacza wyłączenie sygnału alarmowego dla wszystkich domen alarmów.

 **Zobacz też**

- System alarmów > *Sygnalizacja dźwiękowa*

- akcja *PlaySound*, akcja *PlaySoundLooping*

SwitchLanguage

Przeznaczenie

Akcja pozwala na przełączenie języka pracy. Przełączany jest aktualny język aplikacji i skojarzony z nim język programu. Aplikacja musi być skonfigurowana do pracy wielojęzycznej.

Język aplikacji można zmienić także poprzez użycie okna panelu kontrolnego. Okno panelu można w każdej aplikacji otworzyć przy pomocy kombinacji klawiszy *Ctrl-Shift-F1.*

Składnia

SwitchLanguage(*kod_języka*)

Parametry

kod_języka

Dwuliterowy kod języka.

Przykłady

SwitchLanguage("en”)

Akcja przełącza aplikację na pracę w języku angielskim.

 **Zobacz też**

- Parametryzacja aplikacji > *4. Parametryzacja aplikacji wielojęzycznych*

SwitchPanel

Przeznaczenie

Akcja powoduje przełączenmie widzialności wskazanego panelu w oknie.

Składnia

SwitchPanel(nazwa_panela)

SwitchPanel(nazwa_okna, nazwa_panela)

Parametry

nazwa_okna

Nazwa okna, w którym ma zostać przełąoczona widoczność panela.

nazwa_panelu

Nazwa panelu, którego widoczność ma zostać przełączona.

SwitchWorkMode

Przeznaczenie

Akcja pozwala na przejście z trybu wykonywania aplikacji do wybranego trybu edycyjnego.

Wykonanie akcji jest kontrolowane przez system uprawnień. Aby przejść do tryby edycji, trzeba się zalogować oraz posiadać uprawnienie do właściwego trybu edycyjnego.

Tryb pracy można zmienić także poprzez użycie okna panelu kontrolnego. Okno panelu można w każdej aplikacji otworzyć przy pomocy kombinacji klawiszy *Ctrl-Shift-F1.*

Składnia

SwitchWorkMode(*tryb_edycji*)

Parametry

tryb_edycji

Parametr określa żądany tryb edycji. Parametr musi być jedną z poniższych stałych:

$Architect - pełny tryb edycyjny z otwarciem okna Architekta.

$WindowEdit - uproszczony tryb edycji okien synoptycznych.

$DiagramEdit - uproszczony tryb edycji diagramów synoptycznych.

Przykłady

SwitchWorkMode($DiagramEdit)

Akcja przełącza aplikację do trybu uproszczonej edycji diagramów synoptycznych.

 **Zobacz też**

- Jak to zrobić > *Uruchamianie programu i zmiana trybów pracy*/Przejście z trybu wykonawczego do trybu edycyjnego

- Parametryzacja aplikacji > *5. System uprawnień*

SydelShow

Przeznaczenie

Akcja służy do sterowania pracą programu SydelEko firmy Technicon. Akcja pozwala na przełączanie informacji wyświetlanych w oknie programu SydelEKO.

Składnia

SydelShow(*opis_węzła*)

Parametry

opis_węzła

Parametr służy do identyfikacji węzła, którego okno należy wyświetlić. Należy podać pełną nazwę opisową węzła lub jego numeryczny identyfikator.

Terminate

Przeznaczenie

Akcja służy do zakończenia wykonywania aplikacji. Działa tylko wtedy, gdy aplikacja została uruchomiona od początku w trybie wykonawczym.

Wykonanie akcji jest kontrolowane przez system uprawnień. Aby zakończyć pracę aplikacji, trzeba się zalogować oraz posiadać uprawnienie do zamknięcia aplikacji.

Możliwe są alternatywne metody kończenia pracy aplikacji:

- Zamknięcie wszystkich otwartych okien.
- Użycie przycisku końca pracy w oknie panelu kontrolnego. Okno panelu można w każdej aplikacji otworzyć przy pomocy kombinacji klawiszy *Ctrl-Shift-F1*.

Składnia

Terminate()

 **Zobacz też**

- Parametryzacja aplikacji > *5. System uprawnień*

ToggleBits

Przeznaczenie

Akcja pozwala na zmianę wartości wybranych bitów zmiennej procesowej na przeciwną.

Wykonanie akcji jest kontrolowane przez system uprawnień.

Składnia

ToggleBits(*nazwa zmiennej, maska_bitów*)

Parametry

nazwa_zmiennej

Nazwa zmiennej procesowej, do której należy wysłać sterowanie. Pusta nazwa oznacza użycie zmiennej sterowanej obiektu wynikającego z kontekstu użycia akcji. Możliwa jest notacja sufiksowa nazwy zmiennej względem zmiennej głównej obiektu wynikającego z kontekstu.

maska_bitów

Parametr określa bity zmiennej, które ulegną zmianie. Zmieniane są tylko te bity, którym w masce_bitów odpowiadają bity o wartości 1.

Przykłady

ToggleBits ("",0x1)

Wartość najmłodszego bitu zmiennej sterowanej obiektu, na konto którego wykonywana jest akcja, zostanie zmieniona na przeciwną.

 **Zobacz też**

- Techniki budowy diagramów > *6. Organizacja operacji sterujących*

- Parametryzacja aplikacji > *5. System uprawnień*

Transfer

Przeznaczenie

Akcja pozwala na przepisanie aktualnej wartości i status zmiennej źródłowej do zmiennej docelowej.

Wykonanie akcji jest kontrolowane przez system uprawnień.

Składnia

Transfer(*zmienna_źródłowa, zmienna_docelowa, flaga_odczytu*)

Parametry

zmienna_źródłowa

Nazwa zmiennej procesowej, której wartość i status są transferowane do zmiennej docelowej.

zmienna_docleowa

Nazwa zmiennej docelowej.

flaga_odczytu

Wartość logiczna określająca, czy przed wykonaniem transferu wymusić odczyt wartości zmiennej źródłowej. Wymuszenie odczytu może znacząco wydłużyć czas wykonania akcji (w szczególności, gdy wykonywana jest sekwencja akcji Transfer). Z drugiej strony, wymuszenie może być konieczne w przypadku, gdy zmienna źródłowa nie jest odświeżana na żadnym diagramie i pochodzi z kanału innego niż *None*.

VariableInfo

Przeznaczenie

Akcja pozwala na wyświetlenie okienka informacyjnego dotyczącego wskazanej zmiennej. Możliwe jest określenie nazw atrybutów, których wartości zostaną pokazane.

Składnia

VariableInfo(*nazwa zmiennej, lista_atrybutów*)

Parametry

nazwa_zmiennej

Nazwa zmiennej procesowej, dla której należy wyświetlić okienko informacyjne. Pusta nazwa oznacza użycie zmiennej głównej obiektu wynikającego z kontekstu użycia akcji. Możliwa jest notacja sufiksowa nazwy zmiennej względem zmiennej głównej obiektu wynikającego z kontekstu.

lista_atrybutów

Lista nazw atrybutów zmiennej rozdzielonych znakami przecinka. Parametr określa, które atrybuty zmiennej należy wyświetlić w okienku informacyjnym. Pusta lista atrybutów oznacza wyświetlenie wszystkich atrybutów.

Preferowaną metodą określenia listy atrybutów jest wykorzystanie właściwości globalnej typu *Zestaw atrybutów*. Pozwala to jednokrotnie zdefiniować listę, a następnie zastosować ją we wszystkich wywołaniach akcji *VariableInfo*, używając funkcji *Property*.

Przykłady

VariableInfo ("", Property(InfoAttrs))

Wyświetlone zostanie okno informacyjne dla kontekstowej zmiennej głównej obiektu. Nazwy wyświetlanych atrybutów zostaną pobrane z definicji właściwości globalnej *InfoAttrs*.

Zoom

Przeznaczenie

Akcja ustawia stopień skalowania diagramu na wybraną wartość procentową względem wielkości początkowej diagramu.

Składnia

Zoom (*nazwa_okna, nazwa_panelu, stopień_przeskalowania*)

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji ustalane jest na podstawie kontekstu użycia.

nazwa_panelu

Nazwa panelu, w którym aktualnie otwarty diagram zostanie przeskalowany. Może być pusta, a wtedy panel będący obiektem akcji ustalany jest na podstawie kontekstu użycia.

Stopień_przeskalowania

Bezwzględna procentowa wartość przeskalowania diagramu w stosunku do jego wielkości początkowej. Wartość 100 oznacza powrót do początkowej wielkości diagramu.

Przykłady

Zoom (null, schemat, 150)

Diagram znajdujący się w panelu *Schemat* bieżącego okna zostanie przeskalowany do 150% jest wielkości początkowej.

 **Zobacz też**

- akcja *ZoomIn*, akcja *ZoomOut*

ZoomIn

Przeznaczenie

Akcja powoduje powiększenie skalowanego diagramu o zadaną wartość procentową względem aktualnego stanu.

Składnia

ZoomIn (*nazwa_okna, nazwa_panelu, stopień_przeskalowania*)

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji ustalane jest na podstawie kontekstu użycia.

nazwa_panelu

Nazwa panelu, w którym aktualnie otwarty diagram zostanie przeskalowany. Może być pusta, a wtedy panel będący obiektem akcji ustalany jest na podstawie kontekstu użycia.

stopień_przeskalowania

Procentowa wartość powiększenia diagramu w stosunku do jego wielkości aktualnej, np. wartość 10 oznacza powiększenie diagramu o 10%.

 **Zobacz też**

- akcja *Zoom*, akcja *ZoomOut*

ZoomOut

Przeznaczenie

Akcja powoduje pomniejszenie skalowanego diagramu o zadaną wartość procentową względem aktualnego stanu.

Składnia

ZoomOut (*nazwa_okna, nazwa_panelu, stopień_przeskalowania*)

Parametry

nazwa_okna

Nazwa okna, którego akcja dotyczy. Może być pusta, a wtedy okno będące obiektem akcji ustalane jest na podstawie kontekstu użycia.

nazwa_panelu

Nazwa panelu, w którym aktualnie otwarty diagram zostanie przeskalowany. Może być pusta, a wtedy panel będący obiektem akcji ustalany jest na podstawie kontekstu użycia.

stopień_przeskalowania

Procentowa wartość pomniejszenia diagramu w stosunku do jego wielkości aktualnej, np. wartość 10 oznacza pomniejszenie diagramu o 10%.

 **Zobacz też**

- akcja *Zoom*, akcja *ZoomIn*

Asix.Evo - Wyrażenia i funkcje

1 Użycie wyrażeń i funkcji

Podstawowym mechanizmem stosowanym w parametryzacji aplikacji Asix.Evo są wyrażenia. Pozwalają one na dynamiczne wyliczanie wartości w trakcie pracy aplikacji w zależności od aktualnego stanu różnych elementów systemu, w szczególności zmiennych procesowych.

Wyrażenia są używane między innymi w:

- Definicjach właściwości obiektów i elementów menu
- Definicjach właściwości globalnych
- Argumentach akcji operatorskich i funkcji
- Funkcjach przeliczających zmiennych procesowych
- Parametrach wykrywania alarmów

2 Składnia wyrażeń

2.1 Elementy wyrażeń

Konstrukcja wyrażeń Asix.Evo jest zbliżona do składni wyrażeń arytmetycznych języka C. Wyrażenia zbudowane są z operatorów określających wykonywane działania oraz elementów wyrażeń będących argumentami tych działań. Możliwe są poniższe rodzaje elementów, opisane w kolejnych podrozdziałach.

2.1.1 Liczby

Liczby całkowite, zmiennoprzecinkowe lub szesnastkowe. Liczby zmiennoprzecinkowe zawsze używają znaku kropki jako separatora części ułamkowej. Liczby szesnastkowe zapisywane są z prefiksem *0x*.

Przykłady:

77 -14 45.34 12e2 0x0f

2.1.2 Teksty

Napisy złożone z dowolnych znaków ograniczonych znakami ””. Wewnątrz napisu można stosować sekwencje specjalne \n, \r, \” , \\ oznaczające kolejno: znaki nowej linii, znak powrotu karetki, znak ”,

znak \. Można też wstawiać dowolne znaki w kodzie Unicode poprzez sekwencje *u<4-znakowa liczba szesnastkowa>*.

Przykład:

”Ko\u0144” = ”Koń”

2.1.3 Stałe

Stałe symboliczne, które reprezentują z góry określoną wartość. Są to:

- false, wartość logiczna false(fałsz)
- true, wartość logiczna true(prawda)
- null, wartość null(pusta)
- stałe $nazwa, gdzie *nazwa* określa wartość stałej. Są to wartości używane w wywołaniach funkcji i akcji. Zostały opisane w rozdziałach dotyczących konkretnych akcji i funkcji.

2.1.4 Identyfikatory

Identyfikatory są napisami, w których pierwszym znakiem jest litera, a kolejnymi znakami mogą być litery, cyfry i znak _. Znaczenie identyfikatora wynika z kontekstu jego użycia. W sensie wartości i zachowania w trakcie wyliczania wartości wyrażenia są identyczne z tekstami.

Przykłady

Variable (Limit_1) – w wywołaniu funkcji *Variable* identyfikator *Liimit_1* oznacza nazwę zmiennej

Limit_1 + ”_Hi” – operator *+* łączy identyfikator z napisem dając w efekcie napis *Limit_1_Hi*

2.1.5 Wywołania funkcji

Wywołania funkcji i akcji mają postać:

n*azwa(argument_1, …, argument_n)*

Ilość i znaczenie argumentów są zależne od nazwy funkcji. Argumenty wywołania mogą być również wyrażeniami. Funkcje zostały szczegółowo opisane w osobnych rozdziałach.

2.1.6 Komentarze

W treści wyrażeń można umieszczać komentarze. Jako komentarz traktowany jest każdy tekst umieszczony pomiędzy znakami /* i */. Komentarze nie mają wpływy na proces obliczenia wartości wyrażenia.

7 + /* to jest komentarz */ 10

2.2 Operatory

Operatory określają operacje, które należy wykonać na elementach wyrażeń. Kolejność wykonania operacji zależy od priorytetów operatorów i kolejności ich użycia (w przypadku operatorów o identycznym priorytecie). Można sterować kolejnością użycia operatorów przy pomocy nawiasów okrągłych (i). Poniższe rozdziały opisują operatory działań w kolejności malejących priorytetów:

~ ! - (unarny)

Operatory negacji bitowej, negacji logicznej i zmiany znaków.

Przykłady:

~0x000f = 0xfff0

!(1==0) = false

* / %

Operatory mnożenia, dzielenia i dzielenia modulo. Operacja dzielenia na liczbach całkowitych zwraca w wyniku liczbę całkowitą.

Przykłady

17/5 = 3

17%5 = 2

+ -

Operatory dodawania i odejmowania. Operator dodawania w przypadku argumentów tekstowych wykonuje operacje łączenia tekstów.

Przykłady:

5 + 9 = 14

”5” + 9 = ”59”

<< >>

Operatory przesunięć bitowych w lewo i prawo.

Przykłady:

0x10 <<2 = 0x1000

< <= > >=

Operatory porównania mniejsze, mniejsze lub równe, większe, większe lub równe.

== !=

Operatory porównania równe, różne. Operatory mogą być używane do porównywania tekstów.

&

Operator iloczynu bitowego.

Przykłady:

0xf0 & 0x11 = 0x10

^

Operator alternatywy wykluczającej (xor).

Przykłady:

0xf0 ^ 0x11 = 0xe1

|

Operator sumy bitowej.

Przykłady:

0xf0 & | 0x11 = 0xf1

&&

Operator iloczynu logicznego.

Przykłady:

1==0 && 2==2 = false

||

Operator sumy logicznej.

Przykłady:

1==0 || 2==2 = true

?:

Operator wyrażenia logicznego postaci *warunek*?*wyrażenie_gdy_prawda*:*wyrażenia_gdy_fałsz*. W zależności od wyniku sprawdzenia warunku wyliczane jest jedno z dwóch podwyrażeń.

Przykład

1==0 ? ”tak” : ”nie” = ”nie”

3 Wyliczanie wyrażeń

Asix.Evo stosuje optymalizację wyliczania wartości wyrażeń. Oznacza to, że wyrażenia są przeliczane, tylko wtedy, gdy jest to niezbędne. W przypadku wyrażeń użytych we właściwościach obiektów, obliczenie następuje, gdy zmieni się któryś z elementów wyrażenia.

Attribute(z1, LimitHi)

W tym przypadku obliczenie wyrażenia następuje tylko w momencie otwarcia diagramu lub przy zmianie bazy definicji zmiennych.

10 * Variable (z2)

Przeliczenie następuje tylko w momentach zmiany wartości zmiennej *z2*.

3.1 Typy i konwersja wartości

W trakcie wykonywania operacji następuje automatyczna konwersja wartości argumentów. Sposób konwersji zależy od użytego operatora lub funkcji oraz od typu argumentów.

Zasady konwersji są następujące:

- W przypadku działań arytmetycznych uzgadniane są typy argumentów do typu o większym zakresie wartości.

12 + 10.10
wynik 22.10, pierwszy argument konwertowany na liczbę zmiennoprzecinkową

- Teksty użyte w wyrażeniach są z reguły konwertowane na liczby, za wyjątkiem operacji wykonywanych bezpośrednio na tekstach.

  10 * Attribute(z1, LimitHi)
  atrybut *LimitHi* zmiennej zostanie skonwertowany na tekst i pomnożony przez 10

10 + 1

wynik 11, operacja dodania dwóch liczb

”10” + 1

wynik 101, operacja połączenia tekstów

- W miejscu użycia wartości logicznych, można używać wartości liczbowych. Wartości różne od zera oznaczają *true*, równe zero oznaczają *false*. Można też użyć tekstów *”true”* i *”false”*.

Variable(z1) ? red : blue

Jeżeli wartość zmiennej *z1* jest różna od zera, wynikiem będzie tekst *”red”*

- Identyfikatory są konwertowane na teksty.

  Ident1 + 1

  wynikiem jest *ident11* ze względu na wykonanie tekstowej operacji +

W niektórych wypadkach niezbędna może okazać się jawna konwersja typu wartości. Służą do tego funkcje rodziny *To….*

ToDouble(Variable(z1))/Variable(z2)

Jeżeli zmienne *z1* i z2 są typu całkowitego, to wynik zwykłego dzielenia byłby też liczba całkowitą. W wyrażeniu użyto jawnej konwersji pierwszej zmiennej, aby uzyskać dokładny wynik dzielenia.

ToUint32(Attribute(z1, LimitHi)) + 10

Jawna konwersja atrybutu na funkcje powoduje, że operator + ma znaczenie dodawania arytmetycznego, a nie łączenia tekstów.

3.2 Kontekst użycia

Istotne znaczenie dla procesu wyliczenia wyrażenia ma również kontekst jego użycia. Wyrażenie użyte w definicji właściwości obiektu ma kontekstowy dostęp do definicji obiektu i diagramu, w którym zostało użyte. Typową informacją kontekstową jest nazwa zmiennej głównej obiektu.

Zestaw danych kontekstowych zależy od miejsca użycia wyrażenia, np. wyrażenia funkcji przeliczających mają jedynie dostęp do zmiennej, w definicji której zostały użyte.

Informacje kontekstowe są dziedziczone. Wyrażenie użyte we właściwości globalnej nie posiada własnego kontekstu. Jednak, użycie tej właściwości globalnej w definicji właściwości obiektu diagramu spowoduje, że wyrażenie otrzyma kontekst obiektu.

Variable()

Powyższe wyrażenie jest legalne we właściwości obiektu i oznacza pobranie wartości zmiennej głównej obiektu. Nie jest jednak legalne w treści akcji operatorskiej użytej w definicji skrótu klawiszowego.

Variable(”#_hi”)

Powyższe wyrażenie też jest legalne, tylko w kontekście (bezpośrednim lub dziedziczonym) obiektu diagramu. Znak specjalny # oznacza nazwę zmiennej głównej, w efekcie funkcja pobiera wartość zmiennej o nazwie utworzonej przez dodanie sufiksu *_hi* do nazwy zmiennej głównej.

3.3 Diagnostyka wyrażeń

W przypadku problemów w prawidłowym skonstruowaniu wyrażenia polecana jest kontrola komunikatów rejestrowanych w panelu *Komunikaty*. W przypadku błędu w trakcie liczenia wyrażenia pojawiają się tam komunikaty opisujące rodzaj błędu i miejsce jego wystąpienia.

Inną użyteczną techniką jest użycie wyrażenia lub jego fragmentu we właściwości *Tekst* obiektu *Tekst* – pozwala to na wizualną kontrolę wyniku wyrażenia.

4 Spis i opis funkcji

Dostęp do elementów aplikacji

AppData - zwraca wartość wskazanej danej aplikacyjnej lub jej elementu składowego

Asix6Attribute - zwraca wartość wskazanego atrybutu zmiennej procesowej w postaci liczbowej. Jeżeli atrybut nie jest liczbą, to jest traktowany jako nazwa zmiennej i zwracana jest wartość tej zmiennej.

Attribute - zwraca wartość wskazanego atrybutu zmiennej procesowej

AttributeOrDefault - zwraca wartość wskazanego atrybutu zmiennej procesowej lub wartość zastępczą w przypadku błędu.

ExistsAttribute - sprawdza w bazie definicji zmiennych aplikacji, czy używany jest podany atrybut

GetDictionary – zwraca treść słownika wartości pobranego ze wskazanego obiektu tabelarycznego

GetDictionaryValue – zwraca wartość wyliczoną przy pomocy słownika wartości pobranego ze wskazanego obiektu tabelarycznego

GetMessage – zwraca tekst, kolor tekstu albo tła odpowiadający określonej wartości wskazanego zestawu komunikatów

LocalProperty - zwraca wartość właściwości obiektu lub diagramu

LocalPropertyStatus - zwraca status OPC związany z właściwością obiektu lub diagramu

Parameter - zwraca wartość parametru użytego w diagramie lub wzorcu

Parameters - zwraca ciąg nazw i wartości wszystkich parametrów używanych w diagramie, wzorcu lub menu

PowerGuardRead - odczyt danych mierników modułu Strażnik mocy

Property - zwraca wartość właściwości globalnej

Reg - zwraca wartość wskazanego rejestru diagramu lub wzorca

RegStatus - zwraca status OPC wartości wskazanego rejestru diagramu lub wzorca

Text - zwraca tekst z puli tekstów wielojęzycznych

TimetableRead – odczyt wybranych danych związanych z harmonogramem

Dostęp do wartości bieżących zmiennych procesowych

Asix6Limit - sprawdza warunek przekroczenia limitu wartości zmiennej procesowej zgodnie z algorytmem stosowanym w aplikacjach Asix6

CheckStateMask - testuje, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest wartościami liczbowymi maski i spodziewanej wartości.

CheckStateTemplate - testuje, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest w postaci tekstowej.

ExistsVariable - sprawdza, czy zmienna procesowa o podanej nazwie jest zdefiniowana

GetStateDescription - zwraca opis stanu zmiennej procesowej w oparciu o informacje znajdujące się w bazie zmiennych

GetStateText - zwraca nazwę stanu zmiennej procesowej w oparciu o informacje znajdujące się w bazie zmiennych

NewValue - zwraca nową wartość zmiennej procesowej. Funkcja używana tylko w trakcie wykonywania operacji sterującej w funkcji przeliczającej zapisu.

RawValue - zwraca wartość surową zmiennej procesowej, bez wykonania przeliczeń zdefiniowanych w bazie definicji zmiennych

TableElement - zwraca wskazany element z wartości zmiennej procesowej typu tablicowego

TableLength - zwraca liczbę elementów w zmiennej procesowej typu tablicowego

VarChanged - informuje, czy wartość zmiennej uległa zmianie w ostatnim okresie czasu

Variable - zwraca bieżąca wartość zmiennej procesowej po zastosowaniu przeliczeń zdefiniowanych w bazie definicji zmiennych

VariableOrDefault - zwraca bieżąca wartość zmiennej procesowej lub wartość domyślną, gdy zmienna nie jest zdefiniowana.

VarIsGood - informuje, czy status wartości zmiennej procesowej jest dobry.

VarIsNotGood - informuje, czy status wartości zmiennej procesowej nie jest dobry.

VarStatus - zwraca status wartości zmiennej procesowej

VarStringValue - zwraca wartość tekstową zmiennej procesowej, utworzoną na podstawie formatu zdefiniowanego w bazie definicji zmiennych

VarTime - zwraca czas ostatniej aktualizacji wartości zmiennej procesowej

VarTimeUtc - zwraca czas UTC ostatniej aktualizacji wartości zmiennej procesowej

VarValueChangeTime - zwraca czas ostatniej zmiany wartości lub statusu zmiennej procesowej

Dostęp do wartości historycznych zmiennych procesowych

HorizonAggregate - zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego horyzontu czasu liczonego względem chwili bieżącej.

RangeAggregate - zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego zakresu czasu.

Funkcje stanu obiektów i diagramów

Angle - wylicza kąt odpowiadający położeniu kursora w obrębie bieżącego obiektu względem środka tego obiektu.

EventContext – zwraca wskazaną daną kontekstu zdarzenia, której można użyć w argumentach akcji operatorskiej.

HasWaitingControl - informuje, czy znajduje się w stanie oczekiwania na wysłanie sterowania

IsActive - informuje, czy obiekt jest aktywny, czyli może wykonywać operacje sterujące

IsBlinkOff - informuje o aktualnej fazie migotania (zwraca cyklicznie wartości *true* lub *false)*

IsDiagramActive - informuje, czy diagram o podanej nazwie jest aktualnie diagramem aktywnym (bieżącym)

IsDiagramActivePar - informuje, czy diagram o podanej nazwie i zestawie parametrów jest aktualnie diagramem aktywnym (bieżącym)

IsDiagramOpened - informuje, czy diagram o podanej nazwie jest uruchomiony

IsGisLayerVisible - informuje, czy wskazana warstwa diagramu GIS jest widoczna

CurrentGisZoom - zwraca bieżącą skalę (powiększenie) diagramu GIS

IsSelected - informuje, czy obiekt jest aktualnie wybrany (jest obiektem bieżącym)

IsWindowOpened - informuje, czy okno o podanej nazwie jest uruchomione

Wsparcie dla systemu alarmów

AlarmsControler - funkcja zwraca nazwę aktywnego kontrolera

AlarmsGroupState - zwraca status grupy alarmów, pozwala sprawdzić, czy w grupie są alarmy aktywne lub/i niepotwierdzone.

AlarmsMode - informuje o aktualnym trybie pracy systemu alarmów

DeadBand - funkcja wykorzystywana w wykrywaniu alarmów, kontroluje przekraczanie limitów z histerezą

DelayedState - funkcja wykorzystywana w wykrywaniu alarmów, wykrywa brak przewidywanych zmian stanu

IsAlarm - informuje, czy alarm znajduje się w logu alarmów aktywnych i czy jest aktywny(niezakończony)

IsAlarmExcluded - informuje, czy wskazany alarm jest aktualnie wykluczony

IsAlarmUnaccepted - informuje, czy alarm znajduje się w logu alarmów aktywnych i czy jest niepotwierdzony

RateOfChange - funkcja wykorzystywana w wykrywaniu alarmów, monitoruje prędkość zmian wartości wyrażenia

Funkcje konwersji współrzędnych

AbsToRelX - konwertuje współrzędną poziomą lub szerokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach

AbsToRelY - konwertuje współrzędną pionową lub wysokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach

RelToAbsX - konwertuje współrzędną poziomą lub szerokość z wartości względnej w pikselach na wartość bezwzględną (0 do 1 000 000)

RelToAbsY - konwertuje współrzędną pionową lub wysokość z wartości względnej w pikselach na wartość bezwzględną (0 do 1 000 000)

XOnScreen - przelicza współrzędną poziomą z wartości względnej wskazanego monitora na wartość bezwzględną dla pełnego pulpitu

YOnScreen - przelicza współrzędną pionową z wartości względnej wskazanego monitora na wartość bezwzględną dla pełnego pulpitu

Kontrola stanu klawiatury i myszki

CursorX - zwraca współrzędną X aktualnego położenia kursora w ramach diagramu

CursorY - zwraca współrzędną Y aktualnego położenia kursora w ramach diagramu

IsControlPressed - informuje, czy klawisz *Control* jest wciśnięty

IsAltPressed - informuje, czy klawisz *Alt* jest wciśnięty

IsShiftPressed - informuje, czy klawisz *Shift* jest wciśnięty

IsMouseOver - informuje, czy kursor myszki znajduje się nad obiektem

IsMousePressed - informuje, czy przyciśnięty jest klawisz myszki i kursor myszki znajduje się nad obiektem

LastKeyPressed - zwraca opis ostatnio przyciśniętego klawisza na klawiaturze

Operacje na łańcuchach tekstowych

Concat - łączy wiele ciągów znaków w pojedynczy napis

ContainsString - funkcja informuje, czy w podanym łańcuchu tekstowym występuje określony ciąg znaków.

Format - zwraca napis utworzony zgodnie z podanym formatem

Replace - zamienia wszystkie wystąpienia wskazanego podciągu na inny ciąg znaków

SplitString - dzieli napis według podanego separatora na podciągi składowe

Substring - zwraca napis będący fragmentem napisu podanego w wywołaniu funkcji

ToLower - zwraca napis skonwertowany na małe litery

ToUpper - zwraca napis skonwertowany na wielkie litery

Funkcje arytmetyczne

Abs - zwraca wartość bezwzględną

Acos - zwraca wartość funkcji arcus cosinus

Add – sumuje dowolną liczbę argumentów numerycznych

And – wykonuje iloczyn logiczny na dowolnej liczbie argumentów

Asin - zwraca wartość funkcji arcus sinus

Atan - zwraca wartość funkcji arcus tangens

Atan2 - zwraca wartość funkcji arcus tangens dla ilorazu dwóch liczb (współrzędnych kartezjańskich)

Ceiling - zwraca najmniejszą liczbę całkowitą większą lub równą podanej liczbie

Cos - wylicza wartość funkcji cosinus

Div - wylicza iloraz dwóch liczb z kontrolą poprawności wykonania

E - zwraca wartość liczby e

Floor - zwraca największą liczbę całkowitą mniejszą lub równą podanej liczbie

Log - wylicza wartość logarytmu naturalnego

Log10 - wylicza wartość logarytmu dziesiętnego

Max - zwraca większą liczbę

Min - zwraca mniejszą liczbę

Or – wykonuje sumę logiczną na dowolnej liczbie argumentów

PI - zwraca wartość liczby pi

Pow - wylicza wartość liczby podniesionej do podanej potęgi

Round - zaokrągla liczbę do podanej liczby cyfr ułamkowych

Sin - wylicza wartość funkcji sinus

Sqrt - wylicza wartość pierwiastka kwadratowego

Tan - wylicza wartość funkcji tangens

Truncate - zwraca część całkowitą liczby

Funkcje konwersji wartości

CheckOPCStatus - pozwala na uproszczone testowanie wartości statusu OPC

Color - tworzy wartość koloru na podstawie podanych barw podstawowych RGB

FromAsix6Date - konwertuje liczbową wartość daty i czasu aplikacji Asix6 na format daty i czasu

FromAsix6Time - konwertuje liczbową wartość zakresu czasu aplikacji Asix6 na format zakresu czasu

OPCTime - tworzy wartość daty i czasu po zastosowaniu modyfikacji opisanej rozszerzonym wyrażeniem formatującym OPC

ToAsix6Date - konwertuje datę i czas na liczbową wartość daty i czasu aplikacji Asix6

ToAsix6Time - konwertuje zakres czasu na liczbową wartość zakresu czasu aplikacji Asix6

ToBool - konwersja na wartość logiczną *true/false*

ToDateTime - konwertuje wartość do postaci daty i czasu

ToDouble - konwersja na liczbę zmiennoprzecinkową podwójnej precyzji

ToInt8 - konwersja na 8-bitową liczbę całkowitą ze znakiem

ToInt16 - konwersja na 16-bitową liczbę całkowitą ze znakiem

ToInt32 - konwersja na 32-bitową liczbę całkowitą ze znakiem

ToInt64 - konwersja na 64-bitową liczbę całkowitą ze znakiem

ToLocalTime - konwersja wartość typu data/czas z czasu UTC na lokalny

ToSingle - konwersja na liczbę zmiennoprzecinkową pojedynczej precyzji

ToString - konwertuje wartość liczbową interpretowaną jako sekwencję znaków Ascii na tekst

ToTimeSpan - konwertuje wartość do postaci zakresu czasu

ToUInt8 - konwersja na 8-bitową liczbę całkowitą bez znaku

ToUInt16 - konwersja na 16-bitową liczbę całkowitą bez znaku

ToUInt32 - konwersja na 32-bitową liczbę całkowitą bez znaku

ToUInt64 - konwersja na 64-bitową liczbę całkowitą bez znaku

ToUTCTime - konwersja wartość typu data/czas z czasu lokalnego na UTC

ValueOrDefault - zwraca wartość lub wartość domyślną

XorColor - zwraca dopełnienie (XOR) podanego koloru

Pomocnicze

ApplicationChanged - informuje, czy aplikacja została zaktualizowana

AsbaseText - zamiana wartości liczbowej na tekst na podstawie słownika zdefiniowanego w aplikacji Asbase

AsserviceRead - odczyt wartości pola licznika programu AsService

ChannelState - zwraca status pracy kanału komunikacyjnego

Evaluate - wyznacza wartość wyrażenia podanego w parametrze funkcji

HasPrivilege - sprawdza w systemie uprawnień aplikacji, czy aktualny użytkownik posiada podane uprawnienie

HasRole - sprawdza w systemie uprawnień aplikacji, czy aktualny użytkownik pełni daną rolę

IsActiveNote - sprawdza, czy z określonym segmentem skojarzone są jakieś aktywne notatki operatora

IsBrowser - sprawdza, czy aplikacja jest uruchomiona w oknie przeglądarki

IsLiteBrowserTerminal - sprawdza, czy aplikacja jest uruchomiona w oknie przeglądarki w ograniczonym trybie *Lite*.

IsRunning - sprawdza, czy aplikacja jest aktualnie w trybie pracy

IsScriptWorking - sprawdza, czy jest uruchomiony skrypt o podanej nazwie

IsWorking - sprawdza, czy jest nawiązane lub można nawiązać połączenie z określoną stacją zdefiniowaną w ustawieniach aplikacji

RoleDescription - zwraca opis podanej roli systemu zabezpieczeń

Slide - symulacja płynnej zmiany wartości pomiarowej

StationDiagInfo - zwraca informacje diagnostyczne dla wskazanego stanowiska zgromadzone na serwerze diagnostyki

StationInfo - zwraca informacje dotyczące aktualnego stanowiska aplikacji

ToFullPath - zwraca bezwzględną ścieżkę do pliku dla ścieżki podanej względem katalogu aplikacji

ToTable - zwraca tablicę wartości przekazanych jako kolejne argumenty

Abs

Przeznaczenie

Funkcja zwraca wartość bezwzględną liczby.

Składnia

Abs(wartość)

Parametry

wartość

Argument funkcji

Acos

Przeznaczenie

Funkcja wylicza wartość funkcji arcus cosinus. Zwraca kąt mierzony w radianach z zakresu $0 \leq \theta \leq \pi$, którego cosinus wynosi wartość podaną w argumencie funkcji.

Składnia

Acos(wartość)

Parametry

wartość

Argument funkcji

AbsToRelX

Przeznaczenie

Funkcja konwertuje współrzędną poziomą lub szerokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach. Wartości bezwzględne są przechowywane w definicjach współrzędnych. Przejście na wartości względne wykonywane jest w trakcie wyświetlania obiektu i jest dostosowywane do aktualnych rozmiarów diagramu. Współrzędna bezwzględna 1 000 000 odpowiada prawej krawędzi diagramu. Funkcja używana jest w przypadku implementacji ruchu lub zmiany rozmiaru obiektów.

Składnia

AbsToRelX(*współrzędna_X*)

Parametry

współrzędna_X

Współrzędna X lub szerokość obiektu w jednostkach bezwzględnych, którą należy przeliczyć na wartość względną.

Zobacz też

Techniki budowy diagramów > *13.Animacja ruchu i zmiana rozmiaru obiektów*

funkcje: *AbsToRelY*, *RelToAbsX*, *RelToAbsY*

AbsToRelY

Przeznaczenie

Funkcja konwertuje współrzędną pionową lub wysokość z wartości bezwzględnej (0 do 1 000 000) na wartość względną w pikselach. Ogólne zasady użycia są takie same jak w przypadku funkcji *AbsToRelX*.

Składnia

AbsToRelY(*współrzędna_Y*)

Parametry

współrzędna_Y

Współrzędna Y lub wysokość obiektu w jednostkach bezwzględnych, którą należy przeliczyć na wartość względną.

Zobacz też

Techniki budowy diagramów > *13.Animacja ruchu i zmiana rozmiaru obiektów*

funkcje: *AbsToRelX*, *RelToAbsX*, *RelToAbsY*

AlarmsControler

Przeznaczenie

Funkcja zwraca nazwę aktywnego kontrolera w wybranej domenie alarmów. W przypadku terminali podpiętych za pośrednictwem pomostu zwracana jest nazwa użytego pomostu.

Składnia

AlarmsControler ()

AlarmsControler (*nazwa_domeny*)

Parametry

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny, używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *AlarmsMode*

AlarmsCount

Przeznaczenie

Funkcja zwraca ilość alarmów w wybranej grupie alarmów lub w całej domenie. W zależności od sposobu wywołania, można uzyskać ilość alarmów aktywnych, niepotwierdzonych lub aktywnych niepotwierdzonych.

Składnia

AlarmsCount (nazwa_domeny, selektor_grupy, rodzaj_alarmów)

AlarmsCount (selektor_grupy, rodzaj_alarmów)

Parametry

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

selektor_grupy

Parametr określa podzbiór alarmów, którego stan ma zostać zwrócony. Definicja parametru ma postać tekstu złożonego z par <identyfikator_atrybutu_grupującego>=<wartość_atrybutu> rozdzielonych znakami średnika. Użycie pustego parametru wywołania oznacza zwrócenie ilości alarmów dla całej domeny.

rodzaj_alarmów

Parametr określa typ alarmów, które mają zostać policzone. Parametr musi być jedną z poniższych stałych:

$Active - liczone są alarmy aktywne (niezakończone), niezależnie od ich stanu potwierdzenia.

$ActiveUnaccepted - liczone są alarmy aktywne (niezakończone), które nie zostały jeszcze potwierdzone przez operatora.

$Unaccepted - liczone są alarmy, które nie zostały jeszcze potwierdzone przez operatora.

Zobacz też

Funkcja AlarmsGroupState

AlarmsGroupState

Przeznaczenie

Funkcja zwraca zbiorczy stan wybranego zbioru alarmów lub całej domeny. Funkcja pozwala określić, czy w kontrolowanym zbiorze alarmów znajdują się alarmy aktywne (niezakończone) oraz alarmy niepotwierdzone. Interpretację zwracanej wartości opisuje poniższa tabelka.

Bit 2 **Istnieje alarm aktywny niepotwierdzony**	**Bit 1** **Istnieje alarm niezakończony**	**Bit 0** **Istnieje alarm niepotwierdzony**	**Wartość liczbowa**	**Interpretacja**
0	0	0	0	Brak alarmów w logu alarmów aktywnych
0	0	1	1	Istnieją tylko alarmy zakończone niepotwierdzone
0	1	0	2	Istnieją tylko alarmy aktywne potwierdzone
0	1	1	3	Mieszanka alarmów zakończonych niepotwierdzonych i aktywnych potwierdzonych
1	1	1	7	Mieszanka alarmów zakończonych niepotwierdzonych, aktywnych potwierdzonych i aktywnych niepotwierdzonych

W zależność od potrzeb zwracany wynik można testować pod kątem ustawionych bitów lub wartości liczbowych.

Należy pamiętać, że wartość funkcji wyliczana jest na podstawie logu alarmów aktywnych. Oznacza to, że obsługa alarmów zakończonych jest zależna od konfiguracji logu. Jeżeli alarmy zakończone są natychmiast usuwane z logu alarmów aktywnych, to funkcja nigdy nie zwróci stanu oznaczającego, że są niepotwierdzone alarmy zakończone. Analogicznie, jeżeli alarmy zakończone są usuwane po zadanym czasie, to informacja o ich niepotwierdzeniu zostanie utracona.

Składnia

AlarmsGroupState (*selektor_grupy*)

AlarmsGroupState (*nazwa_domeny*, *selektor_grupy*)

Parametry

selektor_grupy

Parametr określa podzbiór alarmów, którego stan ma zostać zwrócony. Definicja parametru ma postać tekstu złożonego z par <*identyfikator_atrybutu_grupującego*>=<*wartość_atrybutu*> rozdzielonych znakami średnika. Użycie pustego parametru wywołania oznacza zwrócenie stanu całej domeny alarmów.

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Przykład

=AlarmsGroupState("Strefa=B1;Strefa=B2;Typ=Elektryczna")&4

Funkcja *AlarmsGroupState* zwraca zbiorczy stan dla alarmów, których atrybut grupujący *Strefa* jest równy *B1* lub *B2* i atrybut *Typ* jest równy *Elektryczna.* Całe wyrażenie natomiast zwraca wartość *true*, gdy w zbiorze kontrolowanych alarmów jest przynajmniej jeden alarm aktywny i niepotwierdzony.

Zobacz też

Funkcja *IsAarm*, *IsAlarmUnaccepted*, *AlarmsCount*

AlarmsMode

Przeznaczenie

Funkcja zwraca tryb pracy stanowiska w ramach wybranej domeny alarmów. Funkcja może zwrócić poniższe wartości:

1. - stanowisko nie bierze udziału w pracy domeny
2. - stanowisko jest kontrolerem aktywnym
3. - stanowisko jest kontrolerem pasywnym
4. - stanowisko jest terminalem lub pomostem
5. - stan stanowiska jest nieustalony, występuje w trakcie startu domeny oraz poszukiwania kontrolera aktywnego

Składnia

AlarmsMode ()

AlarmsMode (nazwa_domeny)

Parametry

nazwa_domeny

Nazwa domeny, której dotyczy wywołanie funkcji. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *AlarmsControler*

Add

Przeznaczenie

Funkcja służy do wyliczenia sumy wielu argumentów numerycznych. Efekt jej działania jest identyczny do użycia konstrukcji z wieloma operatorami +, ale w przypadku dużej liczby argumentów funkcja *Add* jest bardziej efektywna.

W przypadku konieczności połączenia wielu argumentów tekstowych należy stosować funkcję *Concat*.

Składnia

Add (*arg_1, ... ,arg_n*)

Parametry

arg_1, ..., arg_n

Kolejne składniki sumy.

And

Przeznaczenie

Funkcja służy do wyliczenia iloczynu logicznego wielu argumentów. Efekt jej działania jest identyczny do użycia konstrukcji z wieloma operatorami &&, ale w przypadku dużej liczby argumentów funkcja *And* jest bardziej efektywna.

Składnia

And (*arg_1, ... ,arg_n*)

Parametry

arg_1, ..., arg_n

Kolejne argumenty iloczynu logicznego.

Zobacz też

Funkcja Or

Angle

Przeznaczenie

Funkcja zwraca kąt (0 do 360) odpowiadający położeniu kursora w obrębie bieżącego obiektu względem środka tego obiektu. Kąt 0 to kierunek „wschodni” – na prawo od środka obiektu. Kąt wzrasta zgodnie z ruchem wskazówek zegara.

Funkcja może być użyta we wszelkich scenariuszach sterowania kierunkiem za pomocą myszki, na przykład w obsłudze zdarzeń kliknięcia myszką lub animacji.

Składnia

Angle ()

AppData

Przeznaczenie

Funkcja zwraca wartość wskazanej danej aplikacyjnej lub jej elementu składowego. Dana aplikacyjna może być zapisana akcją SetAppData(..) lub przez skrypty. Dane aplikacyjne są dostępne tylko lokalnie w obrębie jednego stanowiska aplikacji. Wartość danej aplikacyjnej może być skalarem (np. pojedyncza liczba), tablicą, tabelą, kolekcją lub innym obiektem

złożonym. Funkcja może być wykorzystywana na przykład do wyświetlania wartości wyliczonych przez skrypty lub do przekazywania tabeli danych jako źródła danych obiektu Wykres Danych lub Tabela Danych.

Składnia

AppData(nazwa_danej)

AppData(nazwa_danej, nazwa_elementu1, nazwa_elementu2, …)

Parametry

nazwa_danej

Nazwa danej aplikacyjnej

nazwa_elementuN

Jeżeli dana aplikacyjna jest tablicą, to można pobrać jej element podając dodatkowo numer elementu tablicy. Jeżeli dana aplikacyjna jest tabelą, to można pobrać jej wskazany element lub tablice wartości dla wskazanej kolumny.

Przykłady

AppData("WspółczynnikX")

Zwraca całą wartość danej aplikacyjnej o nazwie „WspółczynnikX”

AppData("TablicaSterowan", 5)

Zwraca piąty element danej aplikacyjnej o nazwie „TablicaSterowan”, której wartością jest tablica

AppData("TabelaOEE", "Zalaczenia", 0)

Zwraca pierwszy element z kolumny „Zalaczenia” danej aplikacyjnej, której wartością jest tabela danych.

ApplicationChanged

Przeznaczenie

Funkcja zwraca wartość logiczną, informującą czy aplikacja została zaktualizowana poprzez mechanizm okresowej synchronizacji definicji aplikacji.

Składnia

ApplicationChanged()

ArchiveData

Przeznaczenie

Funkcja zwraca tablicę zagregowanych wartości archiwalnych zmiennej ze wskazanego zakresu czasu. Może być wykorzystana na przykład do zasilenia danymi obiekty Wykres Danych.

Składnia

RangeAggregate(*nazwa_agregatu, początek, koniec, interwał, okres_odświeżania*)

RangeAggregate(*nazwa_zmiennej, nazwa_agregatu, początek, koniec, interwał, okres_odświeżania*)

Parametry

nazwa_agregatu

Nazwa agregatu, który należy wyliczyć. Dostępne nazwy to: *none, start, end, delta, min, max, range, total, average, average0, sumup, sumdown, prevknown, last, stdev, rms, avglk, totallk*. Nazwa *none* oznacza odczyt danych surowych. Opis agregatów znajduje się w dokumentacji modułu archiwizacji Aspad (Patrz: Rodzaje agregatów danych archiwalnych).

początek

Początek horyzontu wyliczanego agregatu. Może być wartością typu data i czas (Date*Time)* lub tekstem konwertowalnym na datę i czas. Dopuszczalne są formaty systemowe i OPC (zobacz funkcje *ToDateTime* i *OPCTime).*

koniec

Koniec horyzontu wyliczanego agregatu. Może być wartością typu data i czas (Date*Time)* lub tekstem konwertowalnym na datę i czas. Dopuszczalne są formaty systemowe i OPC (zobacz funkcje *ToDateTime* i *OPCTime).*

interwał

Interwał wyliczanego agregatu. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan).*

okres_odświeżania

Okres ponawiania odczytu danych archiwalnych. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan).*

nazwa_zmiennej

Nazwa zmiennej, której wartości historyczne są pobierane. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej używana jest zmienna wynikająca z kontekstu użycia.

Przykład

ArchiveData(S1, "Average", "HOUR-1H", "HOUR", "5m", "5s")

Funkcja zwraca tablicę 12 średnich 5-cio minutowych zmiennej S1 z okresu od początku godziny poprzedniej do początku godziny bieżącej. Wartość funkcji jest przeliczana co 5 sekund.

ArchiveData("Max", "10.05.2018", "12.05.2018", "1h", "30s")

Funkcja zwraca tablicę 24 maksimów godzinowych z okresu od początku dnia 10.05.2018 do początku dnia 12.05.2018. Wartość funkcji jest przeliczana co 30 sekund.

Zobacz też

- Funkcje *RangeAggregate, HorizonAggregate, ToDateTime, OPCTime, ToTimeSpan,*

- Rodzaje agregatów danych archiwalnych

AsbaseText

Przeznaczenie

Funkcja pozwala zamienić wartość liczbową na jej odpowiednik tekstowy. Przyporządkowanie liczba – tekst zdefiniowane jest w słowniku utworzonym w aplikacji Asbase. W typowym przypadku wartość liczbowa pobierana jest ze zmiennej należącej do zestawu zmiennych, który jest użyty w akcjach operatorskich rodziny Asbase.

Składnia

AsbaseText (*nazwa_słownika*, wartość)

Parametry

nazwa_słownika

Nazwa słownika zdefiniowanego w aplikacji Asbase.

wartość

Wartość liczbową, którą należy zamienić na tekst.

Asin

Przeznaczenie

Funkcja wylicza wartość funkcji arcus sinus. Zwraca kąt mierzony w radianach z zakresu -π/2≤θ≤π/2, którego sinus wynosi wartość podaną w argumencie funkcji.

Składnia

Asin(wartość)

Parametry

wartość

Argument funkcji

Asix6Attribute

Przeznaczenie

Funkcja pobiera z bazy definicji zmiennych wartość tekstową wskazanego atrybutu zmiennej procesowej i konwertuje ją na wartość liczbową. Jeżeli atrybut nie jest liczbą, to jest traktowany jako nazwa zmiennej procesowej i zwracana jest wartość tej zmiennej. Funkcja typowo stosowana w warunkach stanów obiektów, szczególnie w trakcie konwersji aplikacji Asix6.

Składnia

Asix6Attribute (*nazwa_atrybutu*)

Asix6Attribute (*nazwa_zmiennej*, *nazwa_atrybutu*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych i przetworzyć zgodnie z opisanym wyżej algorytmem.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *Attribute*, *Asix6Limit*

Asix6Limit

Przeznaczenie

Funkcja zwraca wartość logiczną, której wartość jest rezultatem sprawdzenia warunku przekroczenia limitu wartości zmiennej procesowej zgodnie z algorytmem stosowanym w aplikacjach Asix6. Funkcja jest typowo stosowana w warunkach stanów obiektów. Szczegółowy algorytm wyliczania wyniku funkcji jest następujący:

- Jeżeli wartość atrybutu jest liczbą, to wartość zmiennej procesowej jest porównywana bezpośrednio z wartością limitu. Funkcja zwraca *true*, gdy wartość zmiennej jest większa od limitu (dla atrybutów *LimitHi* i *LimitHiHi)* lub wartość zmiennej jest mniejsza od limitu (dla atrybutów *LimitLo* i *LimitLoLo).*
- *Jeżeli wartość atrybutu jest nazwą zmiennej, to jako próg limitu pobierana jest wartość tej zmiennej. Dalsze działanie jest zgodne z punktem poprzednim.*
- *Jeżeli wartość atrybutu ma postać nazwa_zmiennej|maska_bitowa, to pobierana jest wartość zmiennej podanej w atrybucie i sprawdzane jest, czy w tej wartości ustawiony jest przynajmniej jeden z jedynkowych bitów maski bitowej.*
- Jeżeli wartość atrybutu ma postać *nazwa_zmiennej&maska_bitowa*, to pobierana jest wartość zmiennej podanej w atrybucie i sprawdzane jest, czy w tej wartości ustawione są wszystkie jedynkowe bity maski bitowej.

Składnia

Asix6Limit (nazwa_atrybutu)

Asix6Limit (nazwa_zmiennej, nazwa_atrybutu)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych i przetworzyć zgodnie z opisanym wyżej algorytmem. Dopuszczalne nazwy atrybutów to: *LimitLo, LimitLoLo, LimitHi* i *LimitHiHi*.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje *Asix6Attribute*

AsserviceRead

Przeznaczenie

Funkcja służy do odczytu wybranego pola wartości licznika programu AsService.

Składnia

AsserviceRead (*identyfikator_licznika*, *nazwa_pola*)

Parametry

Identyfikator_licznika

Identyfikator licznika program AsService, którego dotyczy wywołanie funkcji. Licznik o podanym identyfikatorze musi być zdefiniowany w bazie danych programu AsService.

nazwa_pola

Nazwa pola licznika, którego wartość ma zostać zwrócona przez funkcję. Dozwolone są poniższe nazwy pól:

Pola związane z definicją i aktualnym stanem licznika

Name	Tekstowy opis licznika
CounterType	Typ licznika
CounterTypeDesc	Tekstowy opis typu licznika
VarName	Nazwa zmiennej procesowej na podstawie której przeliczana jest wartość licznika
AggrName	Nazwa agregatu używanego do przeliczeń
MainCounter	Flaga True/False określająca, czy licznik jest licznikiem głównym urządzenia
RefreshFreq	Okres przeliczania wartości licznika
StartValue	Początkowa wartość licznika
Value	Aktualna wartość licznika
StartTime	Czas początku zliczania
NoValueTime	Okres czasu braku odczytu poprawnych wartości zmiennej źródłowej
LastReadTime	Czas ostatniego przeliczenia licznika
NextReadTime	Czas planowanego następnego przeliczenia licznika
OperatingLevel	Wartość limitu eksploatacyjnego
CyclicOperatingLevel	Flaga True/False określająca czy wartość limitu eksploatacyjnego jest odtwarzana po każdym resecie
WarningLevel	Wartość limitu ostrzegawczego
WarningLevel2	Wartość drugiego limitu ostrzegawczego
AutoResetReason	Przyczyna automatycznego resetu licznika
AutoResetReasonDesc	Tekstowy opis przyczyny automatycznego resetu licznika
OperatingTask	Czynność eksploatacyjna związana z przekroczeniem limitu licznika
OperatingTaskDesc	Tekstowy opis czynności eksploatacyjnej
Pola związane z definicją urządzenia, do którego licznik należy	

DeviceId	Identyfikator urządzenia
DeviceName	Nazwa urządzenia
DeviceDesc	Tekstowy opis urządzenia
DeviceSymbol	Alfanumeryczny symbol urządzenia
DeviceRegistryNum	Numer ewidencyjny urządzenia
DeviceFactoryNum	Numer fabryczny urządzenia
DeviceProdYear	Rok produkcji urządzenia
DevicePurchaseDate	Data zakupu urządzenia
DevicePowerInst	
DeviceDimensions	Rozmiary urządzenia
DeviceMass	Masa całkowita urządzenia
DeviceVirtual	Flaga True/False określająca czy urządzenie jest wirtualne
DeviceMachine	Nazwa maszyny, do której urządzenie należy
DeviceMachineDesc	Opis tekstowy maszyny
DeviceDepartment	Nazwa wydziału, do którego urządzenie należy
DeviceDepartmentDesc	Opis tekstowy wydziału
DeviceProducer	Producent urządzenia
DeviceProducerDesc	Opis tekstowy producenta
DeviceSupplier	Dostawca urządzenia
DeviceSupplierDesc	Opis tekstowy dostawcy
DeviceLine	Nazwa linii, do której urządzenie należy
DeviceLineDesc	Opis tekstowy linii
DeviceGroup	Nazwa grupy, do której urządzenie należy
DeviceGroupDesc	Opis tekstowy grupy
DevicePowerInst	Moc zasilania urządzania
DevicePowerType	Typ zasilania urządzenia

DevicePowerTypeDesc	Tekstowy opis typu zasilania
DeviceStatus	Status urządzenia
DeviceStatusDesc	Tekstowy opis statusu
DeviceTechCond	Stan techniczny urządzenia
DeviceTechCondDesc	Tekstowy opis stanu technicznego
Pola opisujące ostatni reset licznika	
ArcEndValue	Wartość licznika w momencie resetu
ArcStartValue	Początkowa wartość licznika
ArcStartTime	Moment uruchomienia licznika
ArcEndTime	Moment resetu licznika
ArcNoValueTime	Okres czasu, przez który brakowało odczytu poprawnych danych źródłowych
ArcProtocolNo	Numer protokołu związanego z resetem licznika
ArcExecutedBy	Osoba wykonująca reset
ArcResetReason	Przyczyna resetu licznika
ArcResetReasonDesc	Opis tekstowy przyczyny resetu

Atan

Przeznaczenie

Funkcja wylicza wartość funkcji arcus tangens. Zwraca kąt mierzony w radianach z zakresu -π/2≤θ≤π/2, którego tangens wynosi wartość podaną w argumencie funkcji.

Składnia

Atan(wartość)

Parametry

wartość

Argument funkcji

Atan2

Przeznaczenie

Funkcja wylicza wartość funkcji arcus tangens dla ilorazu dwóch liczb traktowanych, jako współrzędne układu kartezjańskiego. Zwraca kąt mierzony w radianach z zakresu -π≤θ≤π. Funkcja pozwala przeliczyć współrzędne kartezjańskie na kąt w układzie polarnym.

Składnia

Atan2(wartośćX, wartośćY)

Parametry

wartośćX, wartośćY

Współrzędne kartezjańskie punktu.

Przykład

Atan2(10,20)*(180/PI())

Przelicza współrzędne 10,20 na kąt wyrażony w stopniach.

Attribute

Przeznaczenie

Funkcja pobiera z bazy definicji zmiennej wartość wskazanego atrybutu zmiennej procesowej.

Składnia

Attribute (*nazwa_atrybutu*)

Attribute (*nazwa_zmiennej*, *nazwa_atrybutu*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

Przykład

Attribute(LimitLoLo)

Pobierany jest atrybut *LimitLoLo* kontekstowej zmiennej procesowej. Jeżeli w wyrażeniu definiującym właściwość obiektu nie jest wymagana żadna dodatkowa funkcjonalność, to powyższe wywołanie można zastąpić skrótową notacją:

@LimitLoLo

Zobacz też

Funkcja *ExistsAttribute*

AttributeOrDefault

Przeznaczenie

Funkcja pobiera z bazy definicji zmiennej wartość wskazanego atrybutu zmiennej procesowej. W przypadku błędu (brak zmiennej lub atrybutu) zwraca wartość domyślną.

Składnia

AttributeOrDefault (*nazwa_atrybutu, wartość_domyślna*)

AttributeOrDefault (*nazwa_zmiennej*, *nazwa_atrybutu, wartość_domyślna*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, który należy pobrać z bazy definicji zmiennych.

nazwa_zmiennej

Nazwa zmiennej, której atrybut jest pobierany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, pobierany jest atrybut zmiennej wynikającej z kontekstu użycia.

wartość_domyślna

Wartość zastępcza zwracana w przypadku błędu odczytu atrybutu.

Ceiling

Przeznaczenie

Funkcja zwraca najmniejszą liczbę całkowitą większą lub równą podanej liczbie.

Składnia

Ceiling(argument)

Parametry

argument

Argument funkcji

Zobacz też

Funkcje Floor

ChannelState

Przeznaczenie

Funkcja zwraca status pracy kanału komunikacyjnego. Możliwe są poniższe wartości statusu:

0 - kanał pracuje normalnie

1 - kanał w fazie inicjalizacji (nie pracuje)

2 - błąd komunikacji w kanale fizycznym (kanale Asmena) obsługiwanym za pośrednictwem drajwera Asix6

3 - praca w kanale redundantnym kanału Asmena obsługiwanego za pośrednictwem drajwera Asix6

4 - sterownik PLC, z którym komunikuje się drajwer kanału fizycznego (kanału Asmena) obsługiwanego za pośrednictwem drajwera Asix6, jest w stanie STOP

5 - kanał fizyczny (kanał Asmena) obsługiwany za pośrednictwem drajwera Asix6 jest wyłączony z działania

6 - kanał pracuje na drajwerze redundantnym

30 - inny błąd kanału

31 - niepoprawna nazwa kanału w wywołaniu funkcji

32- stan kanału nie został jeszcze określony

W przypadku użycia kanału Asmena w ramach serwera danych (drajwer Asix6), do określenia jego stanu można również wykorzystać funkcję przeliczającą Asmena STATUS_KANALU.

Składnia

ChannelState(*nazwa_kanału*)

Parametry

nazwa_kanału

Nazwa kanału komunikacyjnego, którego status należy zwrócić.

CheckOPCStatus

Przeznaczenie

Funkcja służy do testowania wartości statusu OPC. Funkcja zwraca wartość logiczną informującą o tym, czy status spełnia podany warunek. Rodzaj testu jest definiowany przez podanie w parametrze wywołania jednej ze stałych rodziny *$Opc....*

Alternatywą dla funkcji *CheckOPCStatus* są bezpośrednie testy bitowe na wartości statusu. Dla prostych testów poprawności wartości zmiennej procesowej można stosować funkcje *VarIsGood* i *VarIsNotGood*.

Składnia

CheckOPCStatus(wartość_statusu, rodzaj_testu)

Parametry

wartość_statusu

Status Opc, którego wartość należy sprawdzić. Typowo status Opc pozyskiwany jest za pomocą funkcji *VarStatus, RegStatus* lub *LocalPropertyStatus*.

rodzaj_testu

Jedna ze stałych rodziny *$Opc...*, która określa rodzaj wykonywanego testu wartości statusu. Pełną listę stałych i ich znaczenie można znaleźć w edytorze wyrażeń w menu *Stałe*.

Przykład

CheckOpcStatus(LocalPropertyStatus(Text), $OpcQualityGood)

Funkcja sprawdza, czy wartość wyliczona we właściwości *Text* obiektu jest poprawna.

Zobacz też

Funkcje *VarStatus VarIsGood VarIsNotGood LocalPropertyStatus*

CheckStateMask

Przeznaczenie

Funkcja zwraca wartość logiczną informującą o tym, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest wartościami liczbowymi maski i spodziewanej wartości. W pierwszym kroku na wartości testowanej zmiennej wykonywana jest operacja iloczynu bitowego z wartością maski. Jeżeli tak uzyskana wartość jest równa wartości spodziewanej, to zwracana jest wartość *true*.

Funkcja jest typowo stosowana w warunkach stanów obiektów.

Składnia

CheckStateMask(*maska_wartości, wartość_stanu*)

CheckStateMask (*nazwa_zmiennej, maska_wartości, wartość_stanu*)

Parametry

maska_wartości

Maska bitowa, wykonywana jest operacja bitowego iloczynu z wartością zmiennej monitorowanej

wartość_stanu

Spodziewana wartość po zastosowaniu maski

nazwa_zmiennej

Nazwa zmiennej, której wartość jest sprawdzana. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, używana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

CheckStateMask(0xf,3)

Funkcja zwraca *true,* gdy stan 4 najmłodszych bitów wartości zmiennej kontekstowej jest równy 0011. Pozostałe bity nie mają znaczenia.

CheckStateMask(z1, 4,0)

Funkcja zwraca *true,* gdy trzeci najmłodszy bit wartości zmiennej *z1* jest równy 0.

CheckStateMask(z1, 4,4)

Funkcja zwraca *true,* gdy trzeci najmłodszy bit wartości zmiennej *z1* jest równy 1. W tym przypadku identycznie zadziała prostsze wyrażenie:

Variable(z1)&4

Zobacz też

Funkcja *CheckStateTemplate*

CheckStateTemplate

Przeznaczenie

Funkcja zwraca wartość logiczną informującą o tym, czy wartość zmiennej procesowej jest zgodna z podanym stanem. Testowany stan opisany jest w postaci tekstowej. Opis stanu składa się z sekwencji znaków: **0** , **1** **i** **-** , odpowiadających spodziewanym wartościom bitów zmiennej testowanej. Znak **-** oznacza, że wartość odpowiadającego mu bitu nie ma znaczenia.

Składnia

CheckStateTemplate(*wzorzec_stanu*)

CheckStateMask (*nazwa_zmiennej, wzorzec_stanu*)

Parametry

wzorzec_stanu

Tekstowy opis spodziewanej wartości zmienne testowanej.

nazwa_zmiennej

Nazwa zmiennej, której wartość jest sprawdzana. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej używana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

CheckStateTemplate(”0110----”)

Funkcja zwraca *true,* gdy stan bitów od 7 do 4 wartości zmiennej kontekstowej jest równy 0110. Pozostałe bity nie mają znaczenia.

CheckStateTemplate(z1,”1--”)

Funkcja zwraca *true,* gdy trzeci najmłodszy bit wartości zmiennej *z1* jest równy 1. W tym przypadku identycznie zadziała prostsze wyrażenie:

Variable(z1)&4

Zobacz też

Funkcja *CheckStateMask*

Color

Przeznaczenie

Funkcja tworzy wartość koloru na podstawie podanych barw podstawowych RGB. Jest to alternatywa do bezpośredniego podania koloru przez jego nazwę.

Składnia

Color (r, g ,b)

Parametry

r

Czerwona składowa koloru, liczba z zakresu 0 - 255

g

Zielona składowa koloru, liczba z zakresu 0 - 255

b

Niebieska składowa kolor, liczba z zakresu 0 - 255

Przykład

Color(Variable(zR), Variable(zG), Variable(zB))

Funkcja wylicza kolor na podstawie wartości trzech zmiennych procesowych.

Zobacz też

Funkcja *XorColor*

Concat

Przeznaczenie

Funkcja pozwala na utworzenie napisu poprzez złączenie dowolnej liczby napisów składowych. Podobną funkcjonalność można uzyskać przy pomocy operatora +, ale w przypadku wielokrotnego łączenia funkcja C*oncat* jest bardziej efektywna. Inną alternatywą jest użycie funkcji *Format*.

Składnia

Concat (*napis_1, ... , napis_n*)

Parametry

napis_1, ..., napis_n

Kolejne napisy składowe, które zostaną złączone w napis wynikowy.

Przykład

Concat(VarTime() ," ", Variable(), " ", (VarIsGood()?OK,Błąd))

Funkcja łączy w pojedynczy napis stempel czasu, wartość i opis statusu kontekstowej zmiennej procesowej. Alternatywne implementacja pokazano poniżej:

VarTime() + " " + Variable() + " " + (VarIsGood()?OK,Błąd)

Format("r;{0} {1} {2}", VarTime(), Variable(), (VarIsGood()?OK,Błąd))

Zobacz też

Funkcja *Format*

ContainsString

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy w podanym łańcuchu tekstowym występuje określony ciąg znaków.

Składnia

ContainsString(napis_wejściowy, napis_wyszukiwany)

Parametry

napis_wejściowy

Napis, w którym wyszukiwany będzie ciąg znaków.

napis_wyszukiwany

Wyszukiwany ciąg znaków.

Cos

Przeznaczenie

Funkcja wylicza wartość funkcji cosinus.

Składnia

Cos (*kąt*)

Parametry

kąt

Argument funkcji cosinus wyrażony w radianach

Zobacz też

Funkcje *Sin*, *Tan*, *Pi*

CurrentGisZoom

Przeznaczenie

Funkcja zwraca bieżącą skalę (powiększenie) wskazanego (lub kontekstowego) diagramu GIS.

Składnia

IsGisLayerVisible()

IsGisLayerVisible(nazwa_okna, nazwa_panela)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wywołanie funkcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panela

Nazwa panela, w którym otwarty jest diagram GIS, którego dotyczy wywołanie funkcji. Pusta nazwa oznacza odwołania do panela kontekstowego.

CursorX

Przeznaczenie

Funkcja zwraca aktualną współrzędna X położenia kursora myszki w ramach diagramu. Zwracana jest w wartość bezwzględna z zakresu od 0 do 1000000.

Składnia

CursorX ()

Zobacz też

Funkcja *CursorY*

CursorY

Przeznaczenie

Funkcja zwraca aktualną współrzędna Y położenia kursora myszki w ramach diagramu. Zwracana jest w wartość bezwzględna z zakresu od 0 do 1000000.

Składnia

CursorY ()

Zobacz też

Funkcja *CursorX*

DeadBand

Przeznaczenie

Funkcja zwraca wartość logiczną, która informuje o przekroczeniu limitu. Sprawdzanie wykonywane jest z histerezą, próg zmiany statusu zależy od kierunku zmiany wartości. Typowo funkcja jest wykorzystywana w strategii warunkowej wykrywania alarmów jako część wyrażenia użytego w parametrach wykrywania.

Składnia

DeadBand (*wartość_monitorowana, limit_włączenia, limit_wyłączenia*)

Parametry

wartość_ monitorowana

Wyrażenie, którego prędkość jest testowana na przekroczenie limitów

limit_włączenia

Limit wartości monitorowanej, powyżej którego następuje zmiana wartości funkcji z *false* na *true*

limit_wyłączenia

Limit wartości monitorowanej, poniżej którego następuje zmiana wartości funkcji z *true* na *false.*

Przykład

DeadBand (Variable(s1), Variable(low), Variable(high))

Wartość zmiennej *s1* jest sprawdzana pod kątem przekroczenia limitów z uwzględnieniem histerezy. Wartości progowe są dynamicznie zmienne, zależne od aktualnych wartości zmiennych *low* i *high*.

Zobacz też

- System_alarmów > *5.2. Strategia warunkowa*

- funkcje *RateOfChange*, *DelayedState*

DelayedState

Przeznaczenie

Funkcja zwraca wartość logiczną, która informuje o tym, że w systemie nie zaszły przewidywane zmiany stanu. Funkcja monitoruje stan warunku początkowego. Jeżeli on zajdzie, to sprawdzane jest, czy w zadanym czasie zostanie spełniony warunek końcowy. Jeżeli nie, to wartość funkcji jest równa *true*. Typowo funkcja jest wykorzystywana w strategii warunkowej wykrywania alarmów jako część wyrażenia użytego w parametrach wykrywania.

Składnia

DelayedState (*warunek_startu, limit_czasu, warunek_zakończenia*)

Parametry

warunek_ startu

Warunek, którego zmiana na *true* powoduje początek cyklu sprawdzania warunku końcowego.

limit_czasu

Okres czasu w sekundach, w ciągu którego spodziewanie jest spełnienie warunku końcowego.

warunek_zakończenia

Warunek logiczny, który powinien zostać spełniony w zadanym czasie.

Przykład

DelayedState (Variable(st1)==1, 5, Variable(stop)==1)

Od momentu zmiany wartości zmiennej *st1* na wartość 1 przez 5 sekund sprawdzane jest, czy wartość zmiennej *stop* też zmieniła się na 1. Jeżeli nie, to funkcja przyjmie wartość *true.*

Zobacz też

- System_alarmów > *5.2. Strategia warunkowa*

- *f*unkcje *RateOfChange*, *DeadBand*

Div

Przeznaczenie

Funkcja zwraca wynik dzielenia dwóch liczb. Jeżeli wartość dzielnika jest ró•na 0 lub wystąpi inny błąd, to zwracana jest podana wartość domyślna.

Składnia

Div (*liczba1, liczba2, wartość_domyślna*)

Parametry

liczba1, liczba2

Argumenty operacji dzielenia.

Wartość_domyślna

Warunek logiczny, który powinien zostać spełniony w zadanym czasie.

Przykład

Div (Variable(v1), Variable(v2),"błąd")

Funkcja zwraca iloraz wartości zmiennych o nazwach *v1* i v2. W przypadku wystąpienia błędu dzielenia zwracany jest tekst *błąd.*

E

Przeznaczenie

Zwraca wartość liczby e.

Składnia

E ()

Zobacz też

Funkcja *Log*

Evaluate

Przeznaczenie

Funkcja wylicza wartość wyrażenia (ewaluatora) podanego w parametrze funkcji. Parametr jest tekstem, w którym należy stosować składnię definicji właściwości. Funkcja jest używana wtedy, gdy nie jest z góry znana treść wyliczanego wyrażenia, np. wyrażenie jest odczytywane z atrybutów zmiennej procesowej.

Składnia

Evaluate (wyrażenie)

Parametry

wyrażenie

Treść obliczanego wyrażenia

Przykład

Evaluate(Attribute(a1))

Z atrybutu *a1* zmiennej kontekstowej pobierana jest treść ewaluatora, który zostanie następnie wyliczony. Poniższe przykłady pokazują efekt obliczenia w zależności od definicji atrybutu:

#_s1	wartość zmiennej o nazwie zmiennej kontekstowej z sufiksem *s1*
@Name	nazwa zmiennej głównej
!g1	wartość właściwości globalnej *g1*
=7+4	11
7+4	tekst *7+4*

EventContext

Przeznaczenie

Funkcja zwraca wskazaną informacje dotyczącą kontekstu zdarzenia, której można użyc w argumentach akcji operatorskiej.

Składnia

EventContext (dana)

Parametry

dana

Nazwa danej. Nazwy dostępnych danych dla określonych zdarzeń obiektów są wymienione w opisach obiektów i ich zdarzeń.

Przykład

OpenDiagram ("SeriesDetails”,"index=”+EventContext("SeriesIndex"), null, $Dialog, $CursorLocation, $CursorLocation, "”, $None)

W powyższym przykładzie Funkcja EventContext została użyta w akcji OpenDiagram wywołanej w zdarzeniu kliknięcia na serię obiektu Wykres danych. Pozwala ona przekazać jako parametr wywołania diagramu numer serii, którą kliknął użytkownik i dzięki temu otworzyć diagram ze szczegółami dotyczącymi konkretnej serii danych.

ExistsAttribute

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy w bazie definicji zmiennych aplikacji używany jest atrybut o podanej nazwie.

Składnia

ExistsAttribute (*nazwa_atrybutu*)

Parametry

nazwa_atrybutu

Nazwa atrybutu, którego obecność w bazie definicji zmiennych jest sprawdzana.

Zobacz też

Funkcja *Attribute*

ExistsVariable

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy w bazie definicji zmiennych aplikacji zdefiniowana jest zmienna o podanej nazwie.

Składnia

ExistsVariable (nazwa_zmiennej)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której obecność w bazie definicji zmiennych jest sprawdzana. Można stosować notację sufiksową nazw zmiennych z użyciem przedrostka #.

Floor

Przeznaczenie

Funkcja zwraca największą liczbę całkowitą mniejszą lub równą podanej liczbie.

Składnia

Floor (argument)

Parametry

argument

Argument funkcji

Zobacz też

Funkcje Ceiling

Format

Przeznaczenie

Funkcja tworzy napis, który powstaje na podstawie napisu formatującego i zestawu argumentów. Argumenty te są konwertowane na tekst zgodnie z instrukcjami zawartymi w napisie formatującym i wstawiane we wskazane miejsca napisu docelowego. Działanie funkcji jest identyczne z działaniem funkcji *String.Format* platformy .NET. Na witrynie *msdn.microsoft.com* można znaleźć szczegółowy opis działania funkcji *Format* łącznie z opisem struktury napisu formatującego. W przykładach użycia pokazano typowe przypadki działania funkcji.

Składnia

Format(*napis_formatujący, argument_0, argument_1, ...*)

Parametry

napis_formatujący

Napis formatujący zgodny z formatem platformy .NET. Fragmenty napisu postaci {x:yyy} oznaczają miejsca wstawienia pozostałych argumentów funkcji, gdzie *x* oznacza numer kolejny argumentu (pierwszy argument ma numer 0), a *yyy* określa szczegóły formatowania obiektu.

argument_1, argument_2, ...

Kolejne argumenty wstawiane do napisu wynikowego zgodnie z zasadami podanymi w napisie formatującym.

Przykład

Format (”{0} {1}”,Variable(),Attribute(Unit))

Tworzy połączenie wartości zmiennej, skonwertowanej na tekst w sposób domyślny wynikający z typu wartości, z opisem jednostki pomiaru pobranym z bazy definicji zmiennych, np. "122 kg”

Format("{0:f4}", Variable())

Konwertuje liczbę zmiennoprzecinkową na tekst z zaokrągleniem do 4 cyfr ułamkowych.

Format("{0:x}", Variable())

Konwertuje liczbę na postać tekstową szesnastkową.

Format("{0:hh:mm:ss}", Variable(DateTime))

Tworzy napis zawierający aktualną godzinę, np. 11:30:00

Format("{0:# ###}", Variable())

Formatuje duże liczby wstawiając znak spacji przed cyfrą setek.

Zobacz też

Funkcje *Concat*, *VarStringValue*

FromAsix6Date

Przeznaczenie

Funkcja przelicza datę i czas wyrażoną liczbą sekund, która upłynęła od 1.1.1970 na wartość typu data i czasu (typ DateTime platformy .NET). Funkcja używana głównie przy konwersji aplikacji z Asix6.

Składnia

FromAsix6Date (*sekundy*)

Parametry

sekundy

Liczba sekund od 1.1.1970.

Zobacz też

Funkcje ToDateTime, ToAsix6Date, OPCTime

FromAsix6Time

Przeznaczenie

Funkcja przelicza czas wyrażony liczbą milisekund na wartość typu zakres czasu (typ TimeSpan platformy .NET).

Składnia

FromAsix6Time (*milisekundy*)

Parametry

milisekundy

Liczba milisekund.

Zobacz też

Funkcje ToTimeSpan, ToAsix6Time, OPCTime

GetDictionary

Przeznaczenie

Funkcja zwraca definicję słownika wartości pobranego z obiektu klasy *Tabela Asbase*. Słownik wartości pozwala na przekodowanie wartości liczbowych na ich odpowiedniki tekstowe. Funkcja jest przeznaczona do użycia we właściwości *Słownik wartości* obiektów klasy *Tekst*.

Składnia

GetDictionary (*nazwa_obiektu, identyfikator_słownika*)

Parametry

nazwa_obiektu

Nazwa obiektu klasy *Tabela Asbase,* z którego zostanie pobrana definicja słownika.

identyfikator_słownika

Identyfikator słownika. W przypadku obiektu *Tabela Asbase* jest to identyfikator pola kolumny tabeli, która podlega przekształceniu wartość-tekst na podstawie słownika zdefiniowanego w bazie modułu Asbase.

Zobacz też

Funkcja *GetDictionary,* Obiekt *Text*

GetDictionaryValue

Przeznaczenie

Funkcja zwraca wartość wyliczoną przy pomocy słownika wartości pobranego z obiektu klasy *Tabela Asbase*. Słownik wartości pozwala na przekodowanie wartości liczbowych na ich odpowiedniki tekstowe.

Składnia

GetDictionaryValue (*nazwa_obiektu, identyfikator_słownika, klucz_słownika*)

Parametry

nazwa_obiektu

Nazwa obiektu klasy *Tabela Asbase,* z którego zostanie pobrana definicja słownika.

identyfikator_słownika

Identyfikator słownika. W przypadku obiektu *Tabela Asbase* jest to identyfikator pola kolumny tabeli, która podlega przekształceniu wartość-tekst na podstawie słownika zdefiniowanego w bazie modułu Asbase.

klucz_słownika

Wartość klucza. Słownik wartości zostanie przeszukany pod kątem obecności w nim pozycji o podanym kluczu – zostanie zwrócona odpowiadająca mu wartość tekstowa.

Przykład

GetDictionaryValue(TabRej, SK2_KOD, Variable(V_SK2_KOD))

Funkcja pobiera słownik wartości z obiektu klasy *Tabela Asbase* z kolumny o identyfikatorze pola *SK2_KOD*. Następnie na podstawie wartości zmiennej *V_SK2*_KOD, ze słownika jest pobierana właściwa wartość tekstowa. Jeżeli zmienna jest częścią zestawu zmiennych podpiętych do tabeli, to uzyskany zostanie efekt synchronicznej zmiany wyświetlanego tekstu zgodnie ze zmianami aktywnego wiersza tabeli.

Podobny efekt można uzyskać przy pomocy funkcji *GetDictionary* i właściwości *Słownik wartości* obiektu klasy *Tekst*.

Zobacz też

Funkcja Get*Dictionary*

GetMessage

Przeznaczenie

Funkcja zwraca tekst, kolor tekstu albo tła odpowiadający określonej wartości klucza wskazanego zestawu komunikatów.

Składnia

GetMessage (*nazwa_zestawu_komunikatów, wartość_klucza, typ_danej*)

Parametry

nazwa_*zestawu_komunikatów*

Nazwa zdefiniowanego zestawu komunikatów

wartość_klucza

Wartość liczbowa klucza

typ_danej

Stała: $Text – tekst komunikatu, $Color – kolor komunikatu, $BackgroundColor – kolor tła komunikatu

Przykład

GetMessage (”Zestaw1”, 5, $Text)

Dla zestawu „Zestaw1” funkcja pobiera tekst komunikatu odpowiadający wartości 5.

GetStateDescription

Przeznaczenie

Funkcja zwraca opis stanu wyrażenia (zmiennej procesowej) w oparciu o informacje znajdujące się w bazie definicji zmiennych. Algorytm działania funkcji jest następujący:

- Określana jest nazwa grupy opisującej stany poprzez odczyt wartości atrybutu *Nazwy stanów* dla zmiennej podanej w wywołaniu funkcji.
- W bazie definicji zmiennych wyszukiwane są wszystkie zmienne, których atrybut *Zestaw stanów*, jest identyczny z nazwą grupy określoną w poprzednim punkcie. Typowo są to zmienne nieaktywne służące tylko do definiowania nazw stanów.

- Wśród zmiennych wybranych w poprzednim punkcie wyszukiwana jest ta, której wartość atrybutu *Wartość_stanu* jest identyczna z wartością stanu przekazanego do wywołania funkcji *GetStateDescription*.
- Wynikiem funkcji jest wartość atrybutu *Opis* zmiennej wybranej w poprzednim punkcie.

Efekt podobny do działania funkcji *GetStateDescription* można uzyskać stosując wzorzec bazujący na obiekcie *Tekst*, w którym zostały zdefiniowane odpowiednie stany.

Składnia

GetStateDescription (*nazwa_zmiennej, wartość_stanu*)

Parametry

nazwa_zmiennej

Nazwa zmiennej służącej do pozyskania opisów stanów z bazy definicji zmiennych . Można używać notacji sufiksowej. Pusta nazwa oznacza użycie zmiennej kontekstowej.

wartość_stanu

Wartość określająca stan, dla którego należy uzyskać opis. W typowym przypadku jest to wyrażenie wykorzystujące wartość zmiennej przekazanej w parametrze *nazwa_zmiennej* (nie jest to jednak obowiązkowe).

Przykład

GetStateDescription (””,Variable())

Funkcja pobiera opisy stanów dla zmiennej kontekstowej, a następnie określa odpowiedni opis na podstawie wartości zmiennej kontekstowej.

Zobacz też

Funkcja *GetStateText*

GetStateText

Przeznaczenie

Funkcja zwraca tekstowy opis stanu na podstawie dwóch parametrów liczbowych: stanu i podstanu oraz informacji znajdujących się w bazie definicji zmiennych. Typowym zastosowaniem jest tworzenie opisów dla wartości pojedynczych bitów, gdzie stan to numer bitu, a podstan to aktualna wartość bitu. Algorytm działania funkcji jest następujący:

- Pobierana jest wartość atrybutu *Nazwy stanów* dla zmiennej podanej w wywołaniu funkcji.
- Jeżeli pobrany atrybut to ciąg postaci 0=stan1;1=stan2;... , to zwracana jest nazwa stanu odpowiadająca wartości równej parametrowi podstanu. Ten wariant zbliżony jest działaniu do funkcji *GetStateDescription*, ale nie wymaga dodatkowych zmiennych.
- Jeżeli działanie funkcji nie zostało jeszcze zakończone, to w bazie definicji zmiennych wyszukiwane są wszystkie zmienne, których atrybut Zestaw stanów, jest identyczny z wartością atrybutu *Nazwy stanów* pobranego w punkcie pierwszym. Typowo są to zmienne nieaktywne służące tylko do definiowania nazw stanów.
- Wśród zmiennych wybranych w poprzednim punkcie wyszukiwana jest ta, której wartość atrybutu *Wartość_stanu* jest identyczna z wartością stanu przekazanego do wywołania funkcji *GetStateText*.
- Jeżeli atrybut *Nazwy stanów* zmiennej wybranej w poprzednim punkcie stanowi ciąg postaci *0=stan1;1=stan2;...,* to zwracana jest nazwa stanu odpowiadająca wartości równej parametrowi podstanu. W przeciwnym przypadku wartością funkcji jest kompletna wartość atrybutu.

Składnia

GetStateText (*nazwa_zmiennej, wartość_stanu, wartość_podstanu*)

Parametry

nazwa_zmiennej

Nazwa zmiennej służącej do pozyskania opisów stanów z bazy zmiennych . Można używać notacji sufiksowej. Pusta nazwa oznacza użycie zmiennej kontekstowej.

wartość_stanu

Wartość określająca stan, dla którego należy uzyskać opis w przypadku stosowania Wirtualnych zmiennych definiujących opisy.

wartość_podstanu

Parametr określający wartość podstan, stosowana przy wyciąganiu opisu z teksów w postaci *0=stan1;1=stan2;...* W typowym przypadku jest to wyrażenie wykorzystujące wartość zmiennej przekazanej w parametrze *nazwa_zmiennej* (nie jest to jednak obowiązkowe).

Przykład

GetStateText(z1, 3 , Variable(z1)&8)

Zadaniem funkcji jest uzyskanie opis stanu pojedynczego bitu. Przykładowe dane w bazie definicji zmiennych:

- Atrybut Nazwy stanów zmiennej z1 jest równy *OpisyBitów*
- Nieaktywna zmienna Bit3 ma atrybut *Zestaw stanów* równy *OpisyBitów*, a atrybut *Wartość stanu* równy 3.

- Atrybut *Nazwy stanów* zmiennej *Bit3* ma postać *0=OK;8=Awaria*

W zależności od wartości bitu 3 zmiennej *z1* zostanie zwrócony opis *OK* lub *Awaria*.

Zobacz też

Funkcja *GetStateDescription*

HasPrivilege

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aktualnie zalogowany użytkownik posiada podane uprawnienie systemowe lub aplikacyjne. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektu, właściwościach *Aktywny* lub *Widoczny* lub operatorskich akcjach warunkowych *Break* i *Perform*. Funkcja jest alternatywą dla zalecanej funkcji *HasRole.*

Składnia

HasPrivilege (*identyfikator_uprawnienia*)

Parametry

Identyfikator_uprawnienia

Tekstowy identyfikator uprawnienia systemowego lub aplikacyjnego, którego posiadanie jest sprawdzane. Lista identyfikatorów wszystkich używanych uprawnień dostępna jest w okienku kreatora wyrażeń.

Zobacz też

Parametryzacja_aplikacji > *5. System uprawnień*, Funkcja *HasRole*

HasRole

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aktualnie zalogowany użytkownik pełni podaną rolę. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektu,

właściwościach *Aktywny* lub *Widoczny* lub operatorskich akcjach warunkowych *Break* i *Perform*.

Składnia

HasRole (*nazwa_roli*)

Parametry

nazwa_roli

Nazwa roli, do której przynależność jest sprawdzana. W nazwie roli można stosować znaki specjalne * i ?.

Zobacz też

Parametryzacja_aplikacji > *5. System uprawnień*, Funkcja *HasPrivilege*

HasWaitingControl

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy obiekt wynikający z kontekstu użycia lub obiekty o zgodnej nazwie znajdują się w stanie oczekiwania na wysłanie sterownia. Funkcja dotyczy obiektów na diagramach synoptycznych, które mogą pracować w trybie opóźnionego sterowania. Funkcja typowo używana w warunku stanu w celu modyfikacji wyglądu obiektu oczekującego na wysłanie sterowania.

Składnia

HasWaitingControl ()

HasWaitingControl (*wzorzec_nazwy*)

Parametry

wzorzec_nazwy

Wzorzec nazwy obiektów, których stan jest kontrolowany. We wzorcu można stosować znaki specjalne * i ?. Parametr ”r;*” sprawdza stan wszystkich obiektów diagramu, nawet tych nieposiadających nazwy. Możliwe jest też użycie jednej z poniższych stałych:

$All - test dotyczy wszystkich obiektów umieszczonych na wszystkich otwartych diagramach

$Window - test dotyczy wszystkich obiektów umieszczonych na diagramach bieżącego okna

$Diagram - test dotyczy wszystkich obiektów umieszczonych na bieżącym diagramie

Użycie bezparametrowego wariantu oznacza test stanu obiektu kontekstowego

Zobacz też

- funkcje *IsActive*, *IsSelected*

- Akcje > akcje *SendControls*, *CancelControls*

HorizonAggregate

Przeznaczenie

Funkcja zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego horyzontu czasu liczonego względem chwili bieżącej (tzw. agregat kroczący)

Składnia

HorizonAggregate(*nazwa_agregatu, horyzont_czasu, okres_odświeżania*)

HorizonAggregate(*nazwa_zmiennej, nazwa_agregatu, horyzont_czasu, okres_odświeżania*)

Parametry

nazwa_agregatu

Nazwa agregatu, który należy wyliczyć. Dostępne nazwy to: *none, start, end, delta, min, max, range, total, average, average0, sumup, sumdown, prevknown, last, stdev, rms, avglk, totallk*. Nazwa *none* oznacza odczyt danych surowych.

horyzont_czasu

Okres wyliczanego agregatu. Liczony zawsze względem chwili bieżącej. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan)*

okres_odświeżania

Okres ponawiania odczytu danych archiwalnych. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan)*

nazwa_zmiennej

Nazwa zmiennej, której wartości historyczne są pobierane. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, używana jest zmienna wynikająca z kontekstu użycia.

Przykład

HorizonAggregate(max,”00:15:00”,”00:00:10)

Funkcja zwraca maksymalną wartość zmiennej kontekstowej z okresu ostatnich 15 minut. Wartość funkcji jest przeliczana co 10 sekund.

Zobacz też

- Funkcje *RangeAggregate*, *ToTimeSpan*

- *Rodzaje agregatów danych archiwalnych*

IsActive

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy obiekt wynikający z kontekstu użycia jest aktywny, czyli może wykonywać operacje sterujące (interaktywne). Wynik funkcji jest bezpośrednio związany z wartością właściwości *Aktywny* obiektu. Funkcja typowo używana w warunku stanu w celu modyfikacji wyglądu obiektu.

Składnia

IsActive ()

Zobacz też

Funkcje *IsSelected*, *HasWaitingControl*

IsActiveNote

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy z określonym segmentem skojarzone są aktualnie jakieś aktywne notatki operatora.

Składnia

IsActiveNote (*identyfikator_segmentu*)

Parametry

identyfikator_segmentu

Identyfikator segmentu, dla którego sprawdzana jest obecność aktywnych notatek. Można stosować znaki specjalne * i ?.

IsAlarm

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazany alarm jest aktualnie aktywny (rozpoczęty i niezakończony).

Składnia

IsAlarm (*identyfikator_alarmu*)

IsAlarm (*nazwa_domeny, identyfikator_alarmu*)

Parametry

identyfikator_alarmu

Identyfikator alarmu, którego status aktywności ma być zwrócony.

nazwa_domeny

Nazwa domeny , do której należy kontrolowany alarm. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *IsAlarmUnaccepted*, *AlarmsGroupState*

IsAlarmExcluded

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazany alarm jest aktualnie na liście alarmów wykluczonych z wykrywania.

Składnia

IsAlarmExcluded (*identyfikator_alarmu*)

IsAlarmExcluded (*nazwa_domeny, identyfikator_alarmu*)

Parametry

identyfikator_alarmu

Identyfikator alarmu, którego status wykluczenia ma być zwrócony.

nazwa_domeny

Nazwa domeny, do której należy kontrolowany alarm. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Akcje > akcje *ExcludeAlarm*, *IncludeAlarm*

IsAlarmUnaccepted

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazany alarm występuje w logu alarmów aktywnych i nie został jeszcze potwierdzony przez operatora.

Składnia

IsAlarmUnaccepted (*identyfikator_alarmu*)

IsAlarmUnaccepted (*nazwa_domeny, identyfikator_alarmu*)

Parametry

identyfikator_alarmu

Identyfikator alarmu, którego status potwierdzenia ma być zwrócony.

nazwa_domeny

Nazwa domeny , do której należy kontrolowany alarm. W przypadku wariantu funkcji bez podania nazwy domeny używana jest domena domyślna (pierwsza zdefiniowana).

Zobacz też

Funkcja *IsAlarm*, *AlarmsGroupState*

IsAltPressed

Przeznaczenie

Funkcja służy do sprawdzenia, czy na klawiaturze jest wciśnięty klawisz *Alt*. Typowe zastosowania funkcji to uwarunkowanie pewnych działań lub wyglądu diagramu od przyciśnięcia klawisza *Alt*. Funkcja typowa używana w warunkach stanów obiektu, właściwościach *Aktywny* lub *Widoczny* lub operatorskich akcjach warunkowych *Break* i *Perform*.

Składnia

IsAltPressed ()

Przykład

Perform(IsAltPressed(), SetVariable(z1,1), Nothing())

Zmienna *z1* jest sterowana tylko wtedy, gdy w momencie wykonania akcji klawisz *Alt* jest przyciśnięty.

Zobacz też

Funkcje *IsControlPressed*, *IsShiftPressed*

IsBlinkOff

Przeznaczenie

Funkcja zwraca cyklicznie wartości *true* lub *false* informujące o aktualnej fazie migotania. Funkcja typowa używana w warunkach stanów obiektu lub wyrażeniach warunkowych we

właściwościach obiektów. Pozwala to zmieniać wygląd obiektów w zależności od fazy migotania.

Składnia

IsBlinkOff ()

Przykład

IsBlinkOff()?Blue:Red

Wyrażenie zwraca cyklicznie wartości kolorów *Blue* i R*ed*.

Zobacz też

Techniki_budowy_diagramów > 4.3. Migotanie poprzez zmianę właściwości obiektów

IsBrowser

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aplikacja jest uruchomiona w oknie przeglądarki www.

Składnia

IsBrowser ()

IsLiteBrowserTerminal

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aplikacja jest uruchomiona w oknie przeglądarki www w ograniczonym trybie *Lite*.

Składnia

IsLiteBrowserTerminal ()

IsControlPressed

Przeznaczenie

Funkcja służy do sprawdzenia, czy na klawiaturze jest wciśnięty klawisz *Control*. Zastosowania funkcji są identyczne jak w przypadku funkcji *IsAltPressed*.

Składnia

IsAltPressed ()

Zobacz też

Funkcje *IsAltPressed*, *IsShiftPressed*

IsDiagramActive

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy diagram o podanej nazwie (lub diagram kontekstowy) jest aktualnie diagramem aktywnym (bieżącym). Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsDiagramActive()

IsDiagramActive(*wzorzec_nazwy*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy diagramu. Można używać znaków * i ?. Jeżeli nazwa nie jest podana, to funkcja sprawdza aktywność diagramu kontekstowego.

Przykład

IsDiagramActive(”Stac*”)

Funkcja zwraca *true*, jeżeli aktywny jest diagram o nazwie rozpoczynającej się od *Stac*.

Zobacz też

Funkcja *IsDiagramOpened*

IsDiagramActivePar

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy diagram o podanej nazwie i zestawie parametrów otwarcia jest aktualnie diagramem aktywnym (bieżącym). Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsDiagramActivePar(*wzorzec_nazwy*, *parametry*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy diagramu. Można używać znaków * i ?.

parametry

Parametry otwarcia diagramu. Podawane w sposób określony w opisie akcji *OpenDiagram*.

Przykład

IsDiagramActivePar(”Hydrant”,”nr=7”)

Funkcja zwraca *true*, jeżeli aktywny jest diagram o nazwie *Hydrant* otwarty z parametrem *nr* równym 7.

Zobacz też

Funkcja *IsDiagramOpened*, *IsDiagramActive*

IsDiagramOpened

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy jest otwarty diagram o podanej nazwie i parametrach otwarcia. Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsDiagramOpened(*wzorzec_nazwy*)

IsDiagramOpened(*wzorzec_nazwy, parametry_otwarcia*)

IsDiagramOpened(*wzorzec_nazwy, parametry_otwarcia, ekran*)

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy diagramu. Można używać znaków * i ?.

parametry_otwarcia

Parametry otwarcia diagramu, których zgodność jest weryfikowana przez funkcję. Sprawdzana jest zgodność tylko parametrów podanych w wywołaniu. Jeżeli parametry nie są podane, to funkcja sprawdza tylko nazwę diagramu.

ekran

Numer ekranu, na którym sprawdzane jest otwarcie diagramu. Możliwe są poniższe wartości:

-1 - ekran jest dowolny (działanie identyczne jak dla wywołania bez podania

numeru ekranu)

0 - ekran na którym znajduje się kursor myszki

większe od 1 - systemowy numer ekranu

Przykład

IsDiagramOpened(”StacX”,”pokoj=11”**)**

Funkcja zwraca *true,* jeżeli otwarty jest diagram o nazwie *StacX* i parametrach otwarcia identycznych z *pokoj=11*.

Zobacz też

Funkcja *IsDiagramActive*

IsGisLayerVisible

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy wskazana warstwa diagramu GIS jest widoczna.

Składnia

IsGisLayerVisible(nazwa_okna, nazwa_panela, nazwa_warstwy)

Parametry

nazwa_okna

Nazwa okna, w którym otwarty jest diagram GIS, którego dotyczy wywołanie funkcji. Pusta nazwa oznacza odwołania do okna kontekstowego.

nazwa_panela

Nazwa panela, w którym otwarty jest diagram GIS, którego dotyczy wywołanie funkcji. Pusta nazwa oznacza odwołania do panela kontekstowego.

nazwa_warstwy

Nazwa warstwy diagramu GIS, którego dotyczy wywołanie funkcji.

IsMouseOver

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy kursor myszki znajduje się nad obiektem. Można sprawdzać pozycję nad obiektem kontekstowym lub wskazanym przez nazwę. Funkcja typowo używana w warunkach stanów obiektu w celu zmiany wyglądu obiektu po najechaniu na niego kursorem myszki.

Funkcja wymaga, żeby sprawdzany obiekt miał właściwość *Aktywny* ustawioną na *true*.

Składnia

IsMouseOver (*wzorzec_nazwy*)

IsMouseOver ()

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy obiektów, dla których sprawdzany jest warunek najechania kursorem myszki. We wzorcu można znaki specjalne * i ?. Jako pierwszy znak można też podać prefiks !.

Algorytm określania obiektów podlegających kontroli najechania kursorem myszki jest następujący:

- brak parametru oznacza sprawdzanie wyłącznie obiektu kontekstowego.

- podanie tylko znaku ! oznacza sprawdzenie dla wszystkich obiektów (także tych bez nazwy) będących częścią wzorca, do którego należy obiekt kontekstowy.

- dodanie prefiksu ! przed wzorcem nazwy oznacza sprawdzanie obiektów bezpośrednio osadzonych w diagramie kontekstowym.

- brak prefiksu ! przed wzorcem nazwy oznacza sprawdzenia tylko obiektów będących częścią wzorca, do którego należy obiekt kontekstowy lub obiektów bezpośrednio osadzonych w diagramie, gdy obiekt kontekstowy nie jest częścią wzorca.

Zobacz też

- Techniki_budowy_diagramów > *7.1. Nawigacja poprzez tekstowe łącza*

- Funkcje *IsMousePressed*

IsMousePressed

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy dowolny klawisz myszki jest wciśnięty, a kursor myszki znajduje się nad obiektem wynikającym z kontekstu użycia.

Funkcja wymaga, żeby sprawdzany obiekt miał właściwość *Aktywny* ustawioną na *true*.

Składnia

IsMousePressed ()

Zobacz też

- - Techniki_budowy_diagramów > *7.1. Nawigacja poprzez tekstowe łgcza*

- Funkcje *IsMouseOver*

IsRunning

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy aplikacja jest aktualnie w trybie pracy. Funkcja pozwala np. na zmianę wyglądu obiektów po przejściu do trybu edycji.

Składnia

IsRunning ()

IsSelected

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy obiekt wynikający z kontekstu użycia jest aktualnie wybrany (jest obiektem bieżącym diagramu). Funkcja typowo używana w warunku stanu w celu modyfikacji wyglądu wyselekcjonowanego obiektu.

Składnia

IsSelected ()

Zobacz też

Funkcje *IsActive*, *HasWaitingControl*

IsShiftPressed

Przeznaczenie

Funkcja służy do sprawdzenia, czy na klawiaturze jest wciśnięty klawisz *Shift*. Zastosowania funkcji są identyczne jak w przypadku funkcji *IsAltPressed*.

Składnia

IsAltPressed ()

Zobacz też

Funkcje *IsControlPressed*, *IsAltPressed*

IsScriptWorking

Przeznaczenie

Funkcja służy do sprawdzenia czy jest aktualnie uruchomiony skrypt o podanej nazwie i liście parametrów startowych.

Składnia

IsScriptWorking (*nazwa_skryptu*))

IsScriptWorking (*nazwa_skryptu*, *parametr_1*, ... , *parametr_n*)

Parametry

nazwa_skryptu

Nazwa skryptu, którego status pracy jest zwracany.

parametr_1, ... , parametr_n

Opcjonalna lista parametrów uruchomienia skryptu.

IsWindowOpened

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy jest otwarte okno o podanej nazwie. Funkcja pozwala na uzyskanie funkcjonalności, którą w aplikacjach Asix6 daje obiekt *Prezenter*.

Składnia

IsWindowOpened*(wzorzec_nazwy)*

IsWindowOpened*(wzorzec_nazwy, ekran)*

Parametry

wzorzec_nazwy

Parametr określa wzorzec nazwy okna. Można używać znaków * i ?.

ekran

Numer ekranu, na którym sprawdzane jest otwarcie okna. Możliwe są poniższe wartości:

-1	- ekran jest dowolny (działanie identyczne jak dla jednoargumentowej wersji wywołania)
0	- ekran na którym znajduje się kursor myszki
większe od 1	- systemowy numer ekranu

Zobacz też

Funkcja *IsDiagramOpened*

IsWorking

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy jest nawiązane lub można nawiązać połączenie z określoną stacją zdefiniowaną w ustawieniach aplikacji. Wartość funkcji jest odświeżana cyklicznie.

Składnia

IsWorking(*nazwa_stanowiska*)

Parametry

nazwa_stanowiska

Nazwa testowanego stanowiska. Nazwa musi być zdefiniowana w ustawieniach aplikacji.

LastKeyPressed

Przeznaczenie

Funkcja zwraca tekstowy opis ostatnio przyciśniętego klawisza na klawiaturze. Postać opisu jest zgodna z opisami stosowanymi w definicji skrótów klawiszowych. Funkcja jest przeznaczona do użycia w obsłudze zdarzenia *Wciśnięcie klawisza* obiektów. Pozwala uzależnić działanie od rodzaju przyciśniętego klawisza.

Składnia

LastKeyPressed ()

Przykład

```
Perform ( LastKeyPressed()==”Alt+G”, SetVariable ( v1, 1), Nothing())
```

Zmienna v1 jest ustawiana na 1, jeżeli zostały przyciśnięte klawisze Alt i g. Funkcja powinna być użyta w obsłudze zdarzenia *Wciśnięcie klawisza*.

Zobacz też

Elementy_wizualizacji > *Standardowe zdarzenia obiektów*

Log

Przeznaczenie

Funkcja wylicza wartość logarytmu naturalnego.

Składnia

Log (*argument*)

Parametry

argument

Argument funkcji logarytmu naturalnego

Zobacz też

Funkcje *E*, *Log10*

LocalProperty

Przeznaczenie

Funkcja zwraca wartość właściwości lokalnej obiektu diagramu. Można pobrać właściwość obiektu kontekstowego lub dowolnego innego wskazanego nazwą.

Składnia

LocalProperty (*nazwa_właściwości*)

LocalProperty (*nazwa_obiektu, nazwa_właściwości*)

Parametry

nazwa_właściwości

Nazwa właściwości lokalnej, którą należy pobrać.

*n**azwa_obiektu***

Nazwa obiektu, którego właściwość lokalną należy pobrać. W przypadku wariantu funkcji bez podania nazwy obiektu, pobierana jest wartość właściwości obiektu wynikającego z kontekstu użycia. Jeżeli funkcja użyta jest w obiekcie znajdującym się w osadzonym wzorcu, to obiekt o danej nazwie najpierw jest szukany w tym samym wzorcu. Dopiero gdy takiego nie ma, to obiekt szukany jest na diagramie, ale z pominięciem obiektów wchodzących w skład innych wzorców.

Przykład

LocalProperty(o2,"Color")

Pobierana jest definicja właściwości lokalnej *Color* z obiektu o nazwie *o2.*

Zobacz też

Funkcje *Property*

LocalPropertyStatus

Przeznaczenie

Funkcja zwraca wartość statusu dla właściwości lokalnej obiektu diagramu. Można pobrać status właściwości obiektu kontekstowego lub dowolnego innego wskazanego nazwą.

Z wartością każdej właściwości obiektu związany jest jej status Opc. Status ten w większości przypadków jest stale ustawiony na QualityGood. Jeżeli jednak definicja właściwości odwołuje się do jakiegoś elementu aplikacji, który status posiada, to ten status jest zwracany. Typowe przypadki to:

a) Odwołania do wartości zmiennych poprzez prefiksy & i # lub samodzielne (nie w złożonych wyrażeniach) wywołania funkcji Variable lub VarStringValue.

b) Status wyliczenia agregatów historycznych funkcjami HorizonAggregate lub RangeAggregate (dla samodzielnych wywołań).

c) Odwołania do wartości liczników programu AsService funkcją AsserviceRead.

W przypadku łączenia przy pomocy operatorów odwołań do elementów aplikacji posiadających status, to stosowana jest zasada, że status jest równy QualityGood wtedy, gdy wszystkie elementy wyrażenia posiadają status QualityGood.

Składnia

LocalPropertyStatus *(nazwa_właściwości)*

LocalPropertyStatus *(nazwa_obiektu, nazwa_właściwości)*

Parametry

nazwa_właściwości

Nazwa właściwości lokalnej, której status należy pobrać.

nazwa_obiektu

Nazwa obiektu, dla którego status właściwości lokalnej należy pobrać. W przypadku wariantu funkcji bez podania nazwy obiektu, pobierany jest status właściwości obiektu wynikającego z kontekstu użycia. Jeżeli funkcja użyta jest w obiekcie znajdującym się w osadzonym wzorcu, to obiekt o danej nazwie najpierw jest szukany w tym samym wzorcu. Dopiero gdy takiego nie ma, to obiekt szukany jest na diagramie, ale z pominięciem obiektów wchodzących w skład innych wzorców.

Przykład

LocalPropertyStatus(o2,"Text")

Pobierany jest status właściwości lokalnej Text z obiektu o nazwie o2. W przypadku gdyby właściwość Text była obliczana np. wyrażeniem =HorizonAggregate(max,”00:15:00”,”00:00:10”) , to pobrany byłby status wyliczenia agregatu historycznego zmiennej procesowej.

Zobacz też

Funkcja *CheckOPCStatus*

Log10

Przeznaczenie

Funkcja wylicza wartość logarytmu dziesiętnego.

Składnia

Log 10(*argument*)

Parametry

argument

Argument funkcji logarytmu dziesiętnego

Zobacz też

Funkcje *Log*

Max

Przeznaczenie

Funkcja zwraca większą z dwóch podanych wartości.

Składnia

Max *(wartość_1, wartość_2)*

Parametry

wartość_1

Pierwsza z porównywanych wartości.

wartość_2

Druga z porównywanych wartości.

Zobacz też

Funkcja Min

Min

Przeznaczenie

Funkcja zwraca najmniejszą z dwóch podanych wartości.

Składnia

Max *(wartość_1, wartość_2)*

Parametry

wartość_1

Pierwsza z porównywanych wartości.

wartość_2

Druga z porównywanych wartości.

Zobacz też

Funkcja Max

NewValue

Przeznaczenie

Funkcja zwraca wartość sterującą zmiennej procesowej, która została przekazana do wysłania w efekcie działań operatora lub skryptów użytkownika. Funkcja używana jest tylko w trakcie wykonywania operacji sterującej w wyrażeniach atrybutu *Funkcja przeliczająca zapisu*. Wynik obliczenia takiego wyrażenia jest wartością przeliczoną wysyłaną do drajwera kanału komunikacyjnego. Funkcja działa na zmiennej kontekstowej, której dotyczy operacja sterująca.

UWAGA: w przypadku korzystania z kanału komunikacyjnego *Asix6,* wartość sterująca jest jeszcze dodatkowo przeliczana przez moduł komunikacyjny Asmen na podstawie atrybutu *Funkcja przeliczająca*.

Składnia

NewValue ()

Zobacz też

Funkcje *RawValue*

OPCTime

Przeznaczenie

Funkcja tworzy wartość daty i czasu wyliczaną względem chwili aktualnej lub dowolnego innego momentu czasu. Przesunięcie opisane jest rozszerzonym wyrażeniem formatującym OPC.

Składnia

OPCTime (*moment_bazowy, przesunięcie_czasu*)

OPCTime (*przesunięcie_czasu*)

Parametry

moment_bazowy

Chwila czasu, względem którego liczone jest przesunięcie. W przypadku wariantu wywołania bez momentu bazowego przesunięcie jest liczone względem chwili bieżącej.

przesunięcie_czasu

Określa modyfikację momentu bazowego służącą do wyliczenia wyniku funkcji. Przesunięcie opisane jest rozszerzonym wyrażeniem formatującym OPC postaci:

[NOW|SECOND|MINUTE|HOUR|DAY|WEEK|MONTH|YEAR][cykl[/przesunięcie]] [[+|-][liczba][Y|MO|W|D|H|M|S]...]

Pierwsza część wyrażenia określa początkowe przesunięcie, np. NOW oznacza dokładnie moment bazowy, HOUR początek godziny. Klauzule *cykl* i *przesunięcie* pozwalają na implementacje cykli, np. HOUR8/6 pozwala na wyliczenie początku 8-godzinnej zmiany, gdzie pierwsza zmiana zaczyna się o godzinie 8. Końcowa część wyrażenia pozwala dodatkowo przesunąć czas, np. HOUR-15M , oznacza 15 minut przed początkiem godziny.

Przykład

Poniższe przykłady zakładają, że moment bazowy (zadeklarowany jawnie lub domyślnie) to 2011//05/09 11:06:30

NOW (lub pusty) 2011//05/09 11:06:30

NOW-1H	2011//05/09 10:06:30
HOUR	2011//05/09 11:00:00
HOUR8/6	2011//05/09 06:00:00
DAY+12H+15M	2011//05/09 12:15:00
MONTH-1D	2011//04/30 00:00:00

Or

Przeznaczenie

Funkcja służy do wyliczenia sumy logicznej wielu argumentów. Efekt jej działania jest identyczny do użycia konstrukcji z wieloma operatorami ||, ale w przypadku dużej liczby argumentów funkcja *Or* jest bardziej efektywna.

Składnia

Or (*arg_1, ... ,arg_n*)

Parametry

arg_1, ..., arg_n

Kolejne argumenty sumy logicznej.

Zobacz też

Funkcja And

Parameter

Przeznaczenie

Funkcja zwraca wartość parametru użytego w diagramie lub wzorcu. Działanie zależy od kontekstu użycia. Funkcja użyta wewnątrz wzorca pobiera zawsze wyłącznie parametry wzorca.

Składnia

Parameter (*nazwa_parametru*)

Parametry

nazwa_parametru

Nazwa parametru, który należy pobrać z zestawu parametrów diagramu lub wzorca

Przykład

Parameter(pokoj)

Pobierany jest parametr *pokoj.* Jeżeli w wyrażeniu definiującym właściwość obiektu nie jest wymagana żadna dodatkowa funkcjonalność, to powyższe wywołanie można zastąpić skrótową notacją:

%pokoj

Parameters

Przeznaczenie

Funkcja zwraca ciąg nazw i wartości wszystkich parametrów używanych w bieżącym diagramie, wzorcu lub menu. Format ciągu parametrów jest zgodny z formatem używanym w akcjach *OpenDiagram*, *OpenWindow* i *ShowMenu*. Funkcja ma za zadanie ułatwić przekazywanie parametrów pomiędzy sekwencyjnie otwieranymi diagramami i menu.

Składnia

Parameters ()

Przykład

ShowMenu((menu1,Parameters())

Akcja otwiera menu kontekstowe diagramu o nazwie *menu1*. Jednocześnie do menu przekazywany jest kompletny zestaw parametrów zdefiniowanych w aktywnym diagramie.

PI

Przeznaczenie

Zwraca wartość liczby pi.

Składnia

PI ()

Zobacz też

Funkcje *Sin*, *Cos*, *Tan*

Pow

Przeznaczenie

Funkcja wylicza wartość liczby podniesionej do podanej potęgi.

Składnia

Pow (*podstawa*, *potęga*)

Parametry

podstawa

Wartość, która ma być podniesiona do potęgi.

potęga

Wartość potęgi.

Zobacz też

Funkcje *Sqrt*

PowerGuardRead

Przeznaczenie

Funkcja zwraca wartości związane z bieżącym stanem wskazanego miernika modułu Strażnik mocy.

Składnia

PowerGuardRead (*identyfikator*, *typ_danej*)

Parametry

identyfikator

Identyfikator miernika, którego dane należy zwrócić.

typ_danej

Określa, jaką daną dotyczącą miernika należy zwrócić. Możliwe jest użycie jednej z poniższych stałych:

$AverageValue – przewidywana średnia wartość w bieżącym cyklu

$Limit– bieżący limit

$CycleStartTime – czas początku cyklu

$CycleEndTime – czas końca cyklu

$IsPredictionSafe – informacja czy przewidywana wartość średnia przekracza limit

$SafeRemainingValue - średnia wartość bezpieczna dla pozostałych interwałów

Property

Przeznaczenie

Funkcja zwraca wartość właściwości globalnej o podanej nazwie.

Składnia

Property (*nazwa_właściwości*)

Parametry

nazwa_właściwości

Nazwa właściwości globalnej, którą należy pobrać.

Przykład

Property(g1)

Pobierana jest definicja właściwości globalnej o nazwie *g1.* Jeżeli w wyrażeniu definiującym właściwość obiektu nie jest wymagana żadna dodatkowa funkcjonalność, to powyższe wywołanie można zastąpić skrótową notacją:

!g1

Zobacz też

Funkcje *LocalProperty*

RangeAggregate

Przeznaczenie

Funkcja zwraca wartość agregatu wartości historycznych zmiennej procesowej ze wskazanego horyzontu czasu.

Składnia

RangeAggregate(*nazwa_agregatu, horyzont_początek,horyzont_koniec, okres_odświeżania*)

RangeAggregate(*nazwa_zmiennej, nazwa_agregatu, horyzont_początek, horyzont_koniec, okres_odświeżania*)

Parametry

nazwa_agregatu

Nazwa agregatu, który należy wyliczyć. Dostępne nazwy to: *none, start, end, delta, min, max, range, total, average, average0, sumup, sumdown, prevknown, last, stdev, rms, avglk, totallk*. Nazwa *none* oznacza odczyt danych surowych. Opis agregatów znajduje się w dokumentacji modułu archiwizacji Aspad (Patrz: Rodzaje agregatów danych archiwalnych; Asix.PDF/CHM, rozdz. *7. Aspad - Program Archiwizacji Danych*).

horyzont_początek

Początek horyzontu wyliczanego agregatu. Może być wartością typu data i czas (Date*Time)* lub tekstem konwertowalnym na datę i czas. Dopuszczalne są formaty systemowe i OPC (zobacz funkcje *ToDateTime* i *OPCTime).*

horyzont_koniec

Koniec horyzontu wyliczanego agregatu. Może być wartością typu data i czas (Date*Time)* lub tekstem konwertowalnym na datę i czas. Dopuszczalne są formaty systemowe i OPC (zobacz funkcje *ToDateTime* i *OPCTime).*

okres_odświeżania

Okres ponawiania odczytu danych archiwalnych. Może być wartością typu zakres czasu (*TimeSpan)* lub tekstem konwertowalnym na zakres czasu (zobacz funkcję *ToTimeSpan).*

nazwa_zmiennej

Nazwa zmiennej, której wartości historyczne są pobierane. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej używana jest zmienna wynikająca z kontekstu użycia.

Przykład

RangeAggregate(min,"HOUR","NOW","00:00:10")

Funkcja zwraca minimalną wartość zmiennej kontekstowej z okresu od początku godziny do chwili bieżącej. Wartość funkcji jest przeliczana co 10 sekund.

RangeAggregate(z1, average,"HOUR-1H","HOUR","00:01:00")

Funkcja zwraca średnią wartość zmiennej *z1* z okresu ostatniej zakończonej godziny. Wartość funkcji jest przeliczana co 1 minutę.

Zobacz też

- Funkcje *HorizonAggregate, ToDateTime, OPCTime, ToTimeSpan,*

- Rodzaje agregatów danych archiwalnych

- Asix.PDF/CHM, rozdz. *7. Aspad - Program Archiwizacji Danych*

RateOfChange

Przeznaczenie

Funkcja wylicza prędkość zmian wartości dowolnego wyrażenia. Typowo jest wykorzystywana w strategii warunkowej wykrywania alarmów jako część wyrażenia użytego w parametrach wykrywania.

Składnia

RateOfChange(*wartość, okres_przeliczania*)

Parametry

wartość

Wyrażenie, którego prędkość zmian wartości jest monitorowana.

okres_przeliczania

Okres czasu wyrażony w sekund określający, jak często wyliczana jest prędkość zmiany wartości monitorowanej.

Przykład

RateOfChange(Variable(s1),5)>20

Co 5 sekund wyliczana jest średnia prędkość zmiany wartości w ostatnim okresie obliczeniowym. Następnie sprawdzane jest, czy ta prędkość jest większa niż 20 jednostek/sekundę.

Zobacz też

- funkcje *DeadBand*, *DelayedState*

- System_alarmów > *5.2 Strategia warunkowa*

RelToAbsY

Przeznaczenie

Funkcja konwertuje współrzędną pionową lub wysokość z wartości względnej wyrażonej w pikselach na wartość bezwzględną (0 do 1 000 000). Ogólne zasady użycia są takie same jak w przypadku funkcji *RelToAbsX*.

Składnia

RelToAbsY(*współrzędna_Y*)

Parametry

współrzędna_Y

Współrzędna Y lub wysokość obiektu w jednostkach względnych(pikselach), którą należy przeliczyć na wartość bexwzględną.

Zobacz też

- funkcje *AbsToRelX*, *AbsToRelY*, *RelToAbsX*

- Techniki_budowy_diagramów > *13 Animacja ruchu i zmiana rozmiaru obiektów*

RawValue

Przeznaczenie

Funkcja zwraca bieżącą wartość surową zmiennej procesowej. Jest to oryginalna wartość zmiennej odczytana przez drajwer kanału komunikacyjnego przed zastosowaniem funkcji przeliczającej zdefiniowanej w bazie zmiennych w atrybucie *Funkcja przeliczająca odczytu*. Głównym zastosowaniem funkcji jest dostęp do wartości zmiennej w wyrażeniach użytych w atrybucie *Funkcja przeliczająca odczytu* w celu przeliczenia wartości surowej zmiennej. Tak przeliczona wartość zmiennej jest potem dostępna poprzez funkcję *Variable*.

UWAGA: w przypadku korzystania z kanału komunikacyjnego *Asix6,* wartość zwracana przez funkcję jest już wstępnie przeliczona przez moduł komunikacyjny Asmen na podstawie atrybutu *Funkcja przeliczająca*.

Składnia

RawValue ()

RawValue (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość surową należy zwrócić. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej odczytywana jest wartość zmiennej wynikającej z kontekstu użycia. W przypadku użycia funkcji w wyrażeniu atrybutu *Funkcja przeliczająca odczytu*, zmienną kontekstową jest zmienna, której definicja atrybutu dotyczy.

Zobacz też

Funkcje *Variable*, *NewValue*

Reg

Przeznaczenie

Funkcja zwraca wartość wybranego rejestru diagramu lub wzorca. Rejestry pozwalają na optymalizację działania aplikacji poprzez centralne wyliczanie wyrażeń, których wartość jest potrzebna w wielu miejscach.

Składnia

Reg(nazwa_rejestru)

Parametry

nazwa_rejestru

Nazwa rejestru, którego wartość ma zostać zwrócona. Nazwa rejestru składa się z litery X i numeru kolejnego rejestru (licząc od 1). Jeżeli nazwa zostanie poprzedzona znakiem !, to oznacza to, że w przypadku użycia funkcji wewnątrz wzorca odwołanie dotyczy rejestru diagramu, na którym wzorzec został osadzony.

Zobacz też

- funkcja RegStatus

RegStatus

Przeznaczenie

Funkcja zwraca wartość statusu dla wskazanego rejestru diagramu lub wzorca.

Z wartością każdego rejestru diagramu lub wzorca związany jest jego status Opc. Zasady wyliczania statusów rejestrów są identyczne jak dla statusów właściwości lokalnych i zostały podane w opisie funkcji *LocalPropertyStatus*.

Składnia

RegStatus (nazwa_rejestru)

Parametry

nazwa_właściwości

Nazwa właściwości lokalnej, której status należy pobrać.

nazwa_obiektu

Nazwa rejestru X1, X2, … . Jeżeli w wywołaniu funkcji użytym wewnątrz wzorca, nazwa rejestru zostanie poprzedzona znakiem !, to oznacza to, odwołanie do rejestru diagramu, na którym obiekt został osadzony.

Zobacz też

Funkcja *LocalPropertyStatus*

RelToAbsX

Przeznaczenie

Funkcja konwertuje współrzędną poziomą lub szerokość z wartości względnej wyrażonej w pikselach na wartość bezwzględną (0 do 1 000 000). Wartości bezwzględne są przechowywane w definicjach współrzędnych. Przejście na wartości względne wykonywane jest w trakcie wyświetlania obiektu i jest dostosowywane do aktualnych rozmiarów diagramu. Współrzędna bezwzględna 1 000 000 odpowiada prawej krawędzi diagramu. Funkcja używana jest w przypadku implementacji ruchu lub zmiany rozmiaru obiektów.

Składnia

RelToAbsX(*współrzędna_X*)

Parametry

współrzędna_X

Współrzędna X lub szerokość obiektu w jednostkach względnych (pikselach), którą należy przeliczyć na wartość bezwzględną.

Zobacz też

- funkcje *AbsToRelX*, *AbsToRelY*, *RelToAbsY*

- Techniki_budowy_diagramów > *13 Animacja ruchu i zmiana rozmiaru obiektów*

Replace

Przeznaczenie

Funkcja tworzy napis poprzez zamianę wszystkich wystąpień wskazanego podciągu na inny ciąg znaków.

Składnia

Replace(*napis, stary_tekst, nowy_tekst*)

Parametry

napis

Napis, na podstawie którego będzie tworzony napis zmodyfikowany.

stary_tekst

Ciąg znaków, który należy zastąpić nową zawartością.

nowy_tekst

Nowy ciąg znaków, który zamieni poprzednią zawartość.

Zobacz też

Funkcje *Substring*, *Concat*

RoleDescription

Przeznaczenie

Funkcja zwraca napis opisujący rolę systemu zabezpieczeń.

Składnia

RoleDescription(rola)

Parametry

rola

Identyfikator roli systemu zabezpieczeń, której opis należy zwrócić.

Zobacz też

Funkcje HasRole

Round

Przeznaczenie

Funkcja zaokrągla liczbę zmiennoprzecinkową do podanej liczby cyfr ułamkowych.

Składnia

Round (*wartość, liczba_cyfr*)

Parametry

wartość

Wartość, która ma być zaokrąglona.

liczba_cyfr

Liczba cyfr ułamkowych w zwracanej wartości.

Przykład

Round (Variable(), 3)

Wywołanie zwraca wartość zmiennej kontekstowej zaokrągloną do 3 miejsc po przecinku. Jeżeli celem byłoby jedynie wyświetlenie zaokrąglonej wartości na ekranie, to można też użyć funkcji formatującej teksty *Format* - w poniższy sposób:

Format(”{0:f3}”,Variable())

Zobacz też

Funkcja *Format*

Sequence

Przeznaczenie

Funkcja zwraca tablicę elementów wypełnioną sekwencją liczb lub czasów. Może być wykorzystywana na przykład do przekazywania wartości serii dla obiektu Wykres Danych lub parametrów obiektu Kontener Wzorców.

Składnia

Sequence (first, last, step)

Parametry

first

Pierwszy element sekwencji. Może być liczbą całkowitą, liczbą dziesiętną lub punktem w czasie (DateTime).

last

Ostatni element sekwencji. Może być liczbą całkowitą, liczbą dziesiętną lub punktem w czasie (DateTime).

step

Krok sekwencji. Może być liczbą całkowitą, liczbą dziesiętną lub czasem trwania (TimeSpan).

Przykład

Sequence(0, 200, 20)

Zwraca tablicę 11 liczb z zakresu od 0 do 200 tworzących ciąg <0,20, 40, …, 180, 200>.

Sequence(ToDateTime("14.05.2018 08:00:00"), ToDateTime("14.05.2018 12:00:00"), ToTimeSpan("30m"))

Zwraca tablicę 9 punktów w czasie (DateTime) tworzących ciąg <„14.05.2018 08:00:00”, „14.05.2018 08:30:00”, „14.05.2018 09:00:00”, …, „14.05.2018 12:00:00”>

Sin

Przeznaczenie

Funkcja wylicza wartość funkcji sinus.

Składnia

Sin (*kąt*)

Parametry

kąt

Argument funkcji sinus wyrażony w radianach

Przykład

Sin(PI() * Variable(z1) / 180)

Funkcja liczy sinus kąta podanego w stopniach i pobranego z wartości zmiennej *z1*.

Zobacz też

Funkcje Cos, *Tan*, *Pi*

Slide

Przeznaczenie

Funkcja symuluje płynną zmianę wartości pomiarowej. Funkcja reaguje na zmianę wartości wejściowej, stopniowo zwiększając wartość wynikową, aż do osiągnięcia wartości wejściowej. Funkcja może być stosowana do płynnej animacji ruchu elementów. Należy jednak pamiętać o tym, że jej użycie powoduje pewne opóźnienie wizualizacji stanu bieżącego kontrolowanego obiektu.

Składnia

Slide (wartość, czas_trwania)

Parametry

wartość

Docelowa wartość funkcji.

czas_trwania

Czas podany w milisekundach, po którym wartość funkcji osiągnie docelową wartość przekazaną w pierwszym parametrze.

Przykład

SetPosition(RelToAbsX(100 + Slide(Variable(pos),1000)),null)

Akcja pozwala na płynny przesuw położenia obiektu w osi X. Należy użyć ją w obsłudze zdarzenia *Animacja*. Zmienna pos przekazuje żądaną pozycję w pikselach. Użycie funkcji Slide spowoduje, że przesunięcie obiektu na diagramie odbędzie się w sposób płynny.

SplitString

Przeznaczenie

Funkcja dzieli podany napis wejściowy na elementy składowe. Elementy składowe są rozdzielone podanym w wywołaniu separatorem. W zależności od sposobu wywołania, funkcja zwraca wskazany podciąg lub tablice wszystkich podciągów.

Składnia

SplitString*(napis, separator)*

SplitString*(napis, separator, numer_elementu)*

Parametry

napis

Napis, który należy podzielić na elementy składowe.

separator

Separator służący do rozdzielenia napisu wejściowego ma elementy składowe.

numer_elementu

Numer podciągu składowego, który zostanie zwrócony w wyniku wykonania funkcji. Pierwszy podciąg ma numer 0. W przypadku wywołania funkcji bez tego parametru, zwrócona zostanie tablica wszystkich podciągów składowych.

Sqrt

Przeznaczenie

Funkcja wylicza wartość pierwiastka kwadratowego.

Składnia

Sqrt (*argument*)

Parametry

argument

Argument funkcji pierwiastka kwadratowego.

Zobacz też

Funkcja *Pow*

StationDiagInfo

Przeznaczenie

Funkcja zwraca wybrane informacje diagnostyczne dotyczącego wskazanego stanowiska zgromadzone na serwerze diagnostyki, np. status kanału komunikacyjnego, nazwę zalogowanego użytkownika itp.

Działanie funkcji wymaga zdefiniowania we właściwościach globalnych aplikacji nazwy serwera diagnostyki.

Składnia

StationDiagInfo(nazwa_stanowiska, *typ_informacji*)

StationDiagInfo(nazwa_stanowiska, *typ_informacji, nazwa zasobu*)

Parametry

nazwa_stanowiska

Nazwa stanowiska, którego dane diagnostyczne mają zostać zwrócone. Dozwolone warianty przekazania parametru są następujące:

Nazwa_stanowiska – dotyczy konkretnego unikalnie zdefiniowanego w aplikacji stanowiska

Nazwa_stanowiska&Adres_IP – dotyczy stanowiska wielokrotnego (np. terminalowego) o konkretnym adresie IP

Nazwa_stanowiska&Nazwa_komputera – dotyczy stanowiska wielokrotnego (np. terminalowego) o konkretnej nazwie systemowej komputera

Nazwa_stanowiska&Nazwa_serwera_RDS#Numer_sesji – dotyczy stanowiska wielokrotnego (np. terminalowego) pracującego na serwerze RDS w sesji o podanym numerze

typ_informacji

Nazwa typu informacji, które zostaną zwrócone przez funkcję. Lista dozwolonych nazw dostępna jest w okienku kreatora wyrażeń.

nazwa_zasobu

Dodatkowy parametr, którego podanie wymagane jest dla niektórych typów informacji, np. nazwa kanału komunikacyjnego, nazwa archiwum, nazwa domeny alarmów.

Przykład

StationDiagInfo(SERV2, ChannelState, Silos1)

Wywołanie zwraca aktualny status pracy dla kanału o nazwie *Silos1* na stanowisku o nazwie *SERV2*.

StationInfo

Przeznaczenie

Funkcja zwraca wybrane informacje o stanowisku aplikacji lub komputerze, np. nazwę i adres sieciowy komputera, nazwę użytkownika systemowego itp.

Składnia

StationInfo(*typ_informacji*)

Parametry

Typ_informacji

Stała określająca, jakie informacje dotyczące stanowiska należy zwrócić. Lista dozwolonych stałych dostępna jest w okienku kreatora wyrażeń.

Przykład

StationInfo($ApplicationName)

Wywołanie zwraca nazwę wykonywanej aplikacji.

Substring

Przeznaczenie

Funkcja zwraca napis będący fragmentem napisu podanego w wywołaniu funkcji.

Składnia

Substring(*napis, początek*)

Substring(*napis, początek, długość*)

Parametry

napis

Napis, z którego zostanie wycięty fragment.

początek

Indeks pierwszego znaku wycinanego fragmentu. Pierwszy znak ma numer 0.

długość

Ilość wycinanych znaków. Brak parametru oznacza wycięcie wszystkich znaków aż do końca napisu.

Przykład

Substring(Attribute(Description), 0, 10)+"..."
Wyrażenie zwraca pierwsze 10 znaków opisu zmiennej kontekstowej uzupełnione o znaki ....

Zobacz też

Funkcje *Replace*, *Concat*

TableElement

Przeznaczenie

Funkcja zwraca wskazany element z wartości tablicowej. Funkcja przeznaczona głównie do uzyskania dostępu do elementów tablicowych zmiennych procesowych w wyrażeniach użytych we właściwościach obiektów.

Składnia

TableElement(*tablica, indeks*)

Parametry

tablica

Wartość tablicowa, której element należy pobrać. Typowo jest to wartość tablicowej zmiennej procesowej. Można również użyć napis - w tym przypadku zwrócony jest pojedynczy znak.

Gdy tablica jest dwuwymiarowa, zwracana jest podtablica zawierająca wszystkie elementy w drugim wymiarze odpowiadające wskazanemu indeksowi w pierwszym wymiarze.

indeks

Indeks elementu, który należy zwrócić. Pierwszy element ma indeks równy 0.

Przykład

TableElement(Variable(),0)

Funkcja zwraca pierwszy element z wartości tablicowej zmiennej kontekstowej.

Zobacz też

funkcja *TableLength*

TableLength

Przeznaczenie

Funkcja zwraca liczbę elementów w wartości tablicowej.

Składnia

TableLength (*tablica*)

Parametry

tablica

Wartość tablicowa, której ilość elementów zostanie zwrócona. Typowo jest to wartość tablicowej zmiennej procesowej. Można również użyć napis - w tym przypadku zwrócona zostanie długość napisu.

Przykład

TableLength(Variable())

Funkcja zwraca ilość elementów w wartości zmiennej kontekstowej.

TableLength(Attribute(Name))

Zwraca długość nazwy zmiennej kontekstowej.

Zobacz też

funkcja *TableElement*

Tan

Przeznaczenie

Funkcja wylicza wartość funkcji tangens.

Składnia

Tan (*kąt*)

Parametry

kąt

Argument funkcji tangens wyrażony w radianach.

Zobacz też

Funkcje *Sin*, Cos,*Pi*

Text

Przeznaczenie

Funkcja zwraca napis z puli napisów wielojęzycznych. Zwracany jest wariant napisu odpowiedni dla aktualnego języka pracy. Użycie funkcji w definicji właściwości *Tekst* obiektu *Tekst* spowoduje, że w momencie zmiany języka obiekt automatycznie zmieni wyświetlany napis.

Składnia

Text (*identyfikator_napisu*)

Parametry

identyfikator_napisu

Identyfikator napisu wielojęzycznego zdefiniowanego w aplikacji.

Zobacz też

Asix.Evo_Parametryzacja_aplikacji, 4 Parametryzacja aplikacji wielojęzycznych

TimetableRead

Przeznaczenie

Funkcja zwraca informacje dotyczące wskazanego harmonogramu.

Składnia

TimetableRead (*identyfikator_harmonogramu, rodzaj_danej*)

Parametry

identyfikator_harmonogramu

Identyfikator harmonogramu, którego dotyczy wywołanie funkcji.

rodzaj_danej

Parametr określa rodzaj zwracanej informacji. Możliwe jest użycie jednej z poniższych stałych:

$NextEventTime – żąda zwrócenia czasu następnego planowanego zdarzenia w postaci wartości typu *DateTime*.

$HasNextEvents – żąda zwrócenia informacji czy są jeszcze jakieś planowane zdarzenia dla harmonogramu. Zwracana jest wartość logiczna.

ToAsix6Date

Przeznaczenie

Funkcja przelicza wartość typu data i czas (typ DateTime platformy .NET) lub tekstowy opis daty i czasu na liczbę sekund, która upłynęła od 1.1.1970. Zwracana wartość jest liczbą 32-bitową bez znaku. Funkcja używana głównie przy konwersji aplikacji Asix6.

Składnia

ToAsix6Time (*data_czas*)

Parametry

data_czas

Przeliczana data i czas, która może być:

- wartością typu data i czas (DateTime)
- tekstem, którego format jest zgodny z formatem stosowanym w funkcji *ToDateTime*

Zobacz też

Funkcje ToDateTime, FromAsix6Date, OPCTime

ToAsix6Time

Przeznaczenie

Funkcja przelicza wartość typu zakres czasu (typ TimeSpan platformy .NET) lub tekstowy opis okresu czasu na liczbę milisekund. Zwracana wartość jest liczbą 32-bitową bez znaku.

Składnia

ToAsix6Time (*zakres_czasu*)

Parametry

zakres_czasu

Przeliczany zakres czasu, który może być:

- wartością typu zakres czasu (TimeSpan)
- tekstem, którego format jest zgodny z formatem stosowanym w funkcji *ToTimeSpan*

Przykład

ToAsix6Time(”30:00”)

Uzyskana wartość jest równa ilości milisekund w 30-godzinnym okresie czasu.

Zobacz też

Funkcje *ToTimeSpan*, *FromAsix6Time*, *OPCTime*

ToBool

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na wartość logiczną *true/false*.

Składnia

ToBool (*wartość*)

Parametry

wartość

Konwertowana wartość. W przypadku wartości liczbowych, jeżeli wartość jest różna 0, to zwracana jest wartość *true*, w przeciwnym przypadku zwracane jest *false*. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę.

Przykład

ToBool(Attribute(*wazna*))

Z bazy definicji zmiennych pobierana jest treść atrybutu *wazna* zmiennej kontekstowej. Następnie treść atrybutu jest konwertowana na liczbę i zamieniana na wartość logiczną. Dopuszczalne jest także bezpośrednie umieszczenie w atrybucie tekstów true lub false.

ToDateTime

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na format liczbowy daty i czasu (typ DateTime platformy .NET)

Składnia

ToDateTime (*data_czas*)

Parametry

data_czas

Data i czas w formacie tekstowym lub w postaci liczby zmiennoprzecinkowej. Dopuszczalny jest dowolny format tekstowy rozpoznawalny przez system jako data i czas. Jeżeli podany jest tylko czas, to jako data zostanie przyjęty dzień bieżący. W przypadku liczby zmiennoprzecinkowej w trakcie konwersji stosowana jest konwencja OLE Automation.

Przykład

DateTime(”15:10”)

Uzyskana zostanie wartość równoważna godzinie 15:10 i dacie dnia bieżącego.

DateTime(Variable(z1))

Wartość zmiennej *z1* zostanie zwrócona w formacie daty i czasu. Wartość zmiennej musi być tekstem konwertowalnym na datę i czas lub liczbą zmiennoprzecinkową w konwencji OLE Automation.

Zobacz też

Funkcje ToAsix6Date, FromAsix6Date, OPCTime

ToDouble

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na liczbę zmiennoprzecinkową podwójnej precyzji.

Składnia

ToDouble (*wartość*)

Parametry

wartość

Konwertowana wartość.

Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Jeżeli wartość jest typu *DateTime*, to jest konwertowana na odpowiednik daty OLE Automation.

Przykład

ToDouble(Variable(i1))/Variable(i2)

Jeżeli zmienne *i1* I *i2* są całkowitoliczbowe, to dzielenie wartości tych zmiennych zwracało by także liczbę całkowitą, np. 4/8 = 0. Konwersja jednego z argumentów dzielenia na liczbę zmiennoprzecinkową powoduje, że wynik będzie dokładny, 4/8 = 0.5

Zobacz też

Funkcja *ToSingle*

ToFullPath

Przeznaczenie

Funkcja zwraca bezwzględną ścieżkę do pliku dla ścieżki podanej względem katalogu definicyjnego aplikacji.

Składnia

ToFullPath (*nazwa_pliku*)

Parametry

nazwa_pliku

Nazwa pliku lub katalogu, która zostanie uzupełniona z uwzględnieniem pełnej ścieżki katalogu aplikacji. Pusta nazwa pliku spowoduje zwrócenie ścieżki katalogu aplikacyjnego.

ToInt8

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 8-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt8 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje ToInt16, ToInt32, ToInt64, ToUInt8, ToUInt16, ToUInt32, ToUInt64

ToInt16

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 16-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt16 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Przykład

ToInt16(Variable(z1)&0xffff)

Operacja & powoduje, że wartość zmiennej *z1* jest ograniczana do zakresu liczby 16-bitowej.

Zobacz też

Funkcje ToInt8, ToInt32, ToInt64, ToUInt8, ToUInt16, ToUInt32, ToUInt64

ToInt32

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 32-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt32 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje ToInt8, ToInt16, ToInt64, ToUInt8, ToUInt16, ToUInt32, ToUInt64

ToInt64

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 64-bitową liczbę całkowitą ze znakiem.

Składnia

ToInt64 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Jeżeli wartość jest typu *DateTime*, to jest konwertowana na odpowiednik typu *FileTime* systemu Windows.

Zobacz też

Funkcje ToInt8, ToInt16, ToInt32, ToUInt8, ToUInt16, ToUInt32, ToUInt64

ToLocalTime

Przeznaczenie

Funkcja konwertuje wartość typu data i czas z czasu UTC na czas lokalny.

Składnia

ToLocalTime(czas_UTC)

Parametry

czas_UTC

Wartość konwertowana. Musi to być wartość typu DateTime lub tekst, który można poprawnie zinterpretować, jako datę i czas.

Zobacz też

Funkcje ToUTCTime

ToLower

Przeznaczenie

Funkcja zwraca napis, który powstał przez zamianę w napisie (przekazanym w parametrze) wszystkich wielkich liter na małe.

Składnia

ToLower(napis)

Parametry

napis

Napis, który zostanie skonwertowany na małe litery.

Zobacz też

Funkcje ToUpper

ToSingle

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na liczbę zmiennoprzecinkową pojedynczej precyzji.

Składnia

ToSingle (*wartość*)

Parametry

wartość

Konwertowana wartość.

Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcja *ToDouble*

ToString

Przeznaczenie

Funkcja konwertuje wartość liczbową interpretowaną jako sekwencję kodów ASCII na tekst Unicode.

Składnia

ToString (*wartość*)

Parametry

wartość

Konwertowana wartość liczbowa. Obsługiwane są także tablice wartości.

Przykład

ToString(Variable(msgtext))

Wartość zmiennej *msgtext* jest konwertowana na tekst. Zakładając, że ta wartość (typu 32-bitowa liczba całkowita) była równa 0x414243, to zwrócony zostanie tekst "CAB". Kolejność znaków wynika z fizycznego ułożenia kolejnych bajtów liczby w pamięci - młodsze bajty występują jako pierwsze.

ToTable

Przeznaczenie

Funkcja zwraca tablicę elementów przekazanych jako kolejne argumenty jej wywołania. Może być wykorzystywana na przykład do przekazywania wartości serii dla obiektu Wykres Danych.

Składnia

ToTable (*el1, el2, el3, …*)

Parametry

Elementy…

Wartości kolejnych elementów tablicy

Przykład

ToTable(Variable("V1"), Variable("V2"), Variable("V3"))

Zwraca tablicę zbudowaną z bieżących wartości trzech wskazanych zmiennych procesowych.

ToTimeSpan

Przeznaczenie

Funkcja konwertuje tekstowy opis zakresu czasu na format zakresu czasu (typ TimeSpan platformy .NET)

Składnia

ToTimeSpan (*zakres*)

Parametry

zakres

Zakres czasu przekazywany jest w formacie tekstowym. Obsługiwane są poniższe konwencje:

- d.h:m:s h:m:s h:m , gdzie d, h, m, s są liczbami nieujemnymi oznaczającymi kolejno dni, godziny, minuty i sekundy, np.: "3.10:5:4" lub "123:00"
- ciąg złożony ze składników ze zbioru (nd, nD, nh, nH, nm, nM, ns, nS), gdzie n jest liczbą nieujemną, a d, h, m, s oznaczają kolejno dni, godziny, minuty i sekundy, np.: "20d" lub "3D 4H 2M 1S" lub "15M10s"

Zobacz też

Funkcje ToAsix6Time, FromAsix6Time

ToUInt8

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 8-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt8 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje ToInt8, ToInt16, ToInt32, ToInt64, ToUInt8, ToUInt16, ToUInt32, ToUInt64

ToUInt16

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 16-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt16 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje ToInt8, ToInt16, ToInt32, ToInt64, ToUInt8, ToUInt16, ToUInt32, ToUInt64

ToUInt64

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 64-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt64 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Jeżeli wartość jest typu *DateTime*, to jest konwertowana na odpowiednik typu *FileTime* systemu Windows.

Zobacz też

Funkcje ToInt8, ToInt16, ToInt32, ToInt64, ToUInt8, ToUInt16, ToUInt32

ToUInt32

Przeznaczenie

Funkcja konwertuje wartość przekazaną w parametrze na 32-bitową liczbę całkowitą bez znaku.

Składnia

ToUInt32 (*wartość*)

Parametry

wartość

Konwertowana wartość. Jeżeli parametr jest tekstem, to najpierw wykonywana jest próba konwersji na liczbę. W przypadku konwersji z liczb zmiennoprzecinkowych wartość jest zaokrąglana. Nie można konwertować liczb zbyt dużych dla docelowego formatu.

Jeżeli wartość jest typu *TimeSpan*, to jest konwertowana na liczbę sekund.

Zobacz też

Funkcje ToInt8, ToInt16, ToInt32, ToInt64, ToUInt8, ToUInt16, ToUInt64

ToUpper

Przeznaczenie

Funkcja zwraca napis, który powstał przez zamianę w napisie (przekazanym w parametrze) wszystkich małych liter na wielkie.

Składnia

ToUpper(napis)

Parametry

napis

Napis, który zostanie skonwertowany na wielkie litery.

Zobacz też

Funkcje ToLower

ToUTCTime

Przeznaczenie

Funkcja konwertuje wartość typu data i czas z czasu lokalnego na czas UTC.

Składnia

ToUTCTime*(czas_lokalny)*

Parametry

czas_lokalny

Wartość konwertowana. Musi to być wartość typu DateTime lub tekst, który można poprawnie zinterpretować, jako datę i czas.

Zobacz też

Funkcje ToLocalTime

Truncate

Przeznaczenie

Funkcja zwraca część całkowitą liczby.

Składnia

Truncate ***(wartość)***

Parametry

wartość

Argument funkcji

Zobacz też

Funkcja Round

ValueOrDefault

Przeznaczenie

Funkcja zwraca wartość przekazaną w parametrze wywołania lub wartość domyślną (zastępczą) w przypadku, gdy wartość podstawowa jest niezdefiniowana, niepoprawna lub posiada status QualityBad. Funkcja pozwala uprościć niektóre wyrażenia wykorzystujące konstrukcje warunkowe.

Składnia

ValueOrDefault ***(wartość, wartość_domyślna)***

Parametry

wartość

Wartość podstawowa, zwracana przez funkcję, gdy jest różna od null (jest ustawiona), posiada status różny od QualityBad i w przypadku liczby zmiennoprzecinkowej jest liczbą poprawną.

wartość_domyślna

Wartość zastępcza zwracana, gdy pierwszy argument wywołania funkcji jest równy null (nieustawiony) lub jest niepoprawną liczbą zmiennoprzecinkową lub posiada status QualityBad.

Przykład

ValueOrDefault(Variable(v1), -1)

Wyrażenie zwraca wartość zmiennej v1, jeżeli została ona już odczytana i jest poprawna. W przeciwnym wypadku zwraca wartość -1. Powyższe wyrażenie jest równoważne wyrażeniu warunkowemu *Variable(v1)!=null && VarIsGood(v1) ? Variable(v1) : -1*

Zobacz też

Funkcja *LocalPropertyStatus*

VarChanged

Przeznaczenie

Funkcja zwraca wartość logiczną informującą o zmianie wartości zmiennej procesowej w ostatnim okresie czasu. Funkcja ma dwa przeznaczenia. Jeżeli wartość okresu sprawdzania jest większa od 0, to funkcja jest typowo używana w warunkach stanów obiektów (lub wyrażeniach warunkowych) - służy do sygnalizowania na diagramie faktu niedawnej zmiany wartości. Jeżeli okres sprawdzania jest równy 0, to funkcja używana jest w wyrażeniach warunków terminarza akcji. Pozwala to na uruchamianie akcji operatorskiej w reakcji na każdorazową zmianę wartości zmiennej.

Składnia

VarChanged (*okres*)

VarChanged (*nazwa_zmiennej, okres*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, dla której wyliczany jest status zmiany wartości. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, kontrolowana jest wartość zmiennej wynikającej z kontekstu użycia.

okres

Okres w sekundach, przez jaki wartość funkcji będzie równa *true* po wykryciu zmiany wartości zmiennej. Wartość 0 oznacza chwilowe ustawienie wartości *true* i niezwłoczny powrót do stanu *false*.

Zobacz też

Funkcja *VarTime*, Terminarz

Variable

Przeznaczenie

Funkcja zwraca bieżącą wartość zmiennej procesowej po zastosowaniu wszystkich przeliczeń wynikających z definicji zmiennej w bazie definicji zmiennych. Dostęp do wartości nieprzeliczonej dostępny jest poprzez funkcję *RawValue*.

Składnia

Variable ()

Variable (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość należy zwrócić. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, odczytywana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

Variable(”r;#_lmt”)

Odczytywana jest wartość zmiennej procesowej o nazwie zbudowanej przez dodanie sufiksu *_lmt* do nazwy zmiennej wynikającej z kontekstu użycia. Jeżeli wyrażenie jest użyte w miejscu, które będzie wymagało przejścia na postać tekstową, to konwersja zostanie

wykonana w sposób domyślny dla typu wartości. Jeżeli wymagana jest kontrola formatowania, to należy stosować funkcje *VarStringValue* lub *Format*.

Zobacz też

Funkcje ***VarIsGood*, *VarIsNotGood*, *VarStatus*, *VarTime*, *RawValue*, *VarStringValue*, *Format***

VariableOrDefault

Przeznaczenie

Funkcja zwraca bieżącą wartość zmiennej procesowej po zastosowaniu wszystkich przeliczeń wynikających z definicji zmiennej w bazie definicji zmiennych (identycznie jak funkcja *Variable*). Jeżeli zmienna o podanej nazwie nie istnieje, to zwracana jest wartość domyślna.

Składnia

VariableOrDefault (*nazwa_zmiennej, wartość_domyślna*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość należy zwrócić. Można stosować sufiksowe nazwy zmiennych.

wartość_domyślna

Wartość zastępcza zwracana, gdy zmienna nie istnieje.

VarIsGood

Przeznaczenie

Funkcja zwraca wartość logiczną informującą czy status wartości zmiennej jest poprawny. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektów.

Składnia

VarIsGood ()

VarIsGood (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której status poprawności jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, sprawdzany jest status zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje ***VarIsNotGood*, *VarStatus*, *VarTime***

VarIsNotGood

Przeznaczenie

Funkcja zwraca wartość logiczną informującą, czy status wartości zmiennej jest niepoprawny - czyli aktualna wartość zmiennej jest niewiarygodna. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektów.

Składnia

VarIsNotGood ()

VarIsNotGood (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której status poprawności jest zwracany. W przypadku wariantu funkcji bez podania nazwy zmiennej, sprawdzany jest status zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje ***VaIsGood*, *VarStatus*, *VarTime***

VarName

Przeznaczenie

Zwraca nazwę zmiennej o wskazanej wartości atrybutu nazwy zastępczej (alias). Atrybut zastępczy nazwy zmiennej można skonfigurować w ustawieniach globalnych aplikacji.

Składnia

VarName(alias)

Parametry

alias

Wartość atrybutu nazwy zastępczej zmiennej

VarNames

Przeznaczenie

Zwraca tablicę nazw zmiennych spełniających określone kryteria. Przydatna na przykład do wypełniania listy wartości parametrów obiektu Kontener wzorców.

Składnia

VarNames(criteria…)

Parametry

Criteria…

Jeden lub więcej warunków selekcji zmiennych rozdzielone przecinkami. Każdy warunek jest postaci <nazwa atrybutu>=<wartość atrybutu>. Można stosować wieloznaczniki ’*’ i ’?’.

Przykłady

VarNames("Channel=S7")

Wszystkie zmienne z kanału S7

VarNames("Channel=S7", "Channel=MODBUS")

Wszystkie zmienne z kanału S7_TCP lub MODBUS

VarNames("Linia=Linia1", "Unit=kV")

Wszystkie zmienne z linii 1 i z jednostką kV

VarNames("Name=IPC")*

Wszystkie zmienne o nazwie rozpoczynającej się od liter „IPC”

*VarNames("Channel=S7", "Name=*_Min")*

Wszystkie zmienne z kanału S7 o nazwie kończącej się na „_Min”

VarStatus

Przeznaczenie

Funkcja zwraca pełny numeryczny status wartości zmiennej zgodny ze standardem OPC. W typowym zastosowaniu funkcja jest używana w warunkach stanów obiektów.

Składnia

VarStatus ()

VarStatus (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której status wartości jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, zwracany jest status zmiennej wynikającej z kontekstu użycia.

Przykład

(VarStatus()&0xc0)==0xc0

Działanie powyższego wywołania jest identyczne z użyciem funkcji *VarIsGood()*

Zobacz też

Funkcje *VarIsGood*, *VarIsNotGood*, *VarTime*

VarStringValue

Przeznaczenie

Funkcja zwraca wartość tekstową zmiennej procesowej utworzoną na podstawie formatu zdefiniowanego w bazie definicji zmiennych. Alternatywną metodą formatowania wartości jest funkcja *Format*.

Składnia

VarStringValue ()

VarStringValue (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której bieżącą wartość należy zwrócić w formacie tekstowym. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej, odczytywana jest wartość zmiennej wynikającej z kontekstu użycia.

Przykład

Concat (VarStringValue(), ” ”, Attribute(Unit))

Zwracany jest napis złożony ze sformatowanej wartości zmiennej procesowej i nazwy jednostki pobranej z bazy definicji zmiennych.

VarStringValue(z1)

Zwraca wartość zmiennej *z1* w tekstowej formie. Jeżeli nie jest wymagana dodatkowa funkcjonalność, to w definicji właściwości obiektu identyczny efekt można uzyskać poprzez skrótową notację:

&z1

Zobacz też

Funkcje *Variable*, *Format*

VarTime

Przeznaczenie

Funkcja zwraca czas ostatniej aktualizacji wartości zmiennej procesowej. W przypadku typowych kanałów komunikacyjnych jest to czas ostatniego odświeżenia. Czas zmiennej może się zmienić nawet wtedy, gdy nie zmienia się wartość zmiennej. W przypadku kanału wirtualnego czas aktualizacji jest czasem ostatniego zapisu zmiennej.

Czas ostatniej zmiany wartości zmiennej może być pobrany przy pomocy funkcji *GetValueChangeTime*.

Składnia

VarTime ()

VarTime (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której czas aktualizacji wartości jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej zwracany jest czas aktualizacji zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje ***VaIsGood*, *VarIsNotGood*, *VarStatus*, *Variable, VarValueChangeTime, VarTimeUtc***

VarTimeUtc

Przeznaczenie

Funkcja zwraca czas UTC ostatniej aktualizacji wartości zmiennej procesowej. W przypadku typowych kanałów komunikacyjnych jest to czas ostatniego odświeżenia. Czas zmiennej może się zmienić nawet wtedy, gdy nie zmienia się wartość zmiennej. W przypadku kanału wirtualnego czas aktualizacji jest czasem ostatniego zapisu zmiennej.

Składnia

VarTimeUtc ()

VarTimeUtc (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której czas UTC aktualizacji wartości jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej zwracany jest czas UTC aktualizacji zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje ***VarIsGood, VarIsNotGood, VarStatus, Variable, VarTime***

VarValueChangeTime

Przeznaczenie

Funkcja zwraca czas ostatniej zmiany wartości lub statusu zmiennej procesowej.

Składnia

VarValueChangeTime ()

VarValueChangeTime (*nazwa_zmiennej*)

Parametry

nazwa_zmiennej

Nazwa zmiennej, której czas zmiany wartości lub statusu jest zwracany. Można stosować sufiksowe nazwy zmiennych. W przypadku wariantu funkcji bez podania nazwy zmiennej zwracany jest czas zmiany zmiennej wynikającej z kontekstu użycia.

Zobacz też

Funkcje ***VarTime***

XOnScreen

Przeznaczenie

Funkcja przelicza współrzędną poziomą z wartości względnej dla wskazanego monitora na wartość bezwzględną dla pełnego pulpitu. Funkcja przeznaczona do użycia w akcjach *OpenWindow* i *OpenDiagram*. Pozwala na pozycjonowanie otwieranych okien na wybranych monitorach bez konieczności znajomości dokładnej konfiguracji systemu.

Składnia

XOnScreen(*współrzędna_X*, *monitor*)

Parametry

współrzędna_X

Wartość względnej koordynaty poziomej.

monitor

Numer monitora, względem którego nastąpi przeliczenie koordynaty względnej *współrzędna_X* na koordynatę bezwzględną pulpitu. Wartość 0 oznacza użycie monitora, na którym znajduje się kursor myszki. Wartości większe od 0 to numery monitorów zgodne z numeracją systemu operacyjnego.

Przykład

OpenDiagram (diag1, pokoj=11, null, $Dialog, XOnScreen(0,2), YOnScreen(0,2), ”NoTitleBar,FixedSize”, $None)

Akcja spowoduje otwarcie okna diagramu w lewym górnym narożniku drugiego monitora.

Zobacz też

- funkcja *YOnScreen*

- Asix.Evo_Akcje.PDF/CHM, akcje *OpenDiagram*, *OpenWindow*

XorColor

Przeznaczenie

Funkcja zwraca dopełnienie (XOR) podanego koloru.

Składnia

XorColor (*kolor*)

Parametry

kolor

Parametr koloru może być wartością numeryczną pobraną z właściwości obiektu lub uzyskaną przy pomocy funkcji Color. Kolor można podać też w postaci tekstowej, jako nazwę koloru lub wartości liczbowe składowych rozdzielone znakami średnika.

Przykład

XorColor (red)

Wywołanie zwraca dopełnienie koloru czerwonego.

XorColor(LocalProperty("Color"))

Wywołanie zwraca dopełnienie koloru pobranego z właściwości *Color* obiektu.

Zobacz też

Funkcja *Color*

YOnScreen

Przeznaczenie

Funkcja przelicza współrzędną pionową z wartości względnej dla wskazanego monitora na wartość bezwzględną dla pełnego pulpitu. Ogólne zasady użycia są identyczne jak dla funkcji *XOnScreen.*

Składnia

YOnScreen(*współrzędna_Y, monitor*)

Parametry

współrzędna_Y

Wartość względnej koordynaty pionowej.

monitor

Numer monitora, względem którego nastąpi przeliczenie koordynaty względnej *współrzędna_Y* na koordynatę bezwzględną pulpitu. Wartość 0 oznacza użycie monitora, na którym znajduje się kursor myszki. Wartości większe od 0 to numery monitorów zgodne z numeracją systemu operacyjnego.

Zobacz też

- funkcja *XOnScreen*

- Asix.Evo_Akcje.PDF/CHM, akcje *OpenDiagram*, *OpenWindow*

Asix.Evo - System alarmów

1 Podstawowe pojęcia

System Asix.Evo posiada wbudowany elastyczny, konfigurowalny system obsługi alarmów, który pozwala na pełne zaspokojenie potrzeb związanych z kompleksową obsługą sytuacji awaryjnych i zdarzeń w procesie technologicznym.

System alarmów Asix.Evo jest zorganizowany w tzw. domeny. Domena identyfikowana jest przez nazwę i w praktyce stanowi w pełni funkcjonalny i niezależny od innych domen samodzielny system. Domeny pozwalają na logiczny i funkcjonalny podział alarmów na pewne niezależne obszary. W typowym zastosowaniu, w aplikacji stosowana jest pojedyncza domena, służąca do obsługi wszystkich alarmów aplikacji. W bardziej złożonych przypadkach, w których aplikacja obsługuje wiele niezależnych instalacji, może być przydatne rozdzielenie alarmów na kilka domen.

Uwaga: W aplikacjach Asix, domenie odpowiadała bezpośrednio tzw. nazwa sieciowa zasobu alarmowego. Podstawowa różnica, to możliwość użycia wielu, jednocześnie aktywnych domen Asix.Evo.

Stany alarmowe rozpoznawane są przy pomocy strategii wykrywania alarmów. Istnieją różne typy strategii, które wykrywają alarmy różnymi metodami.

Wykryte alarmy gromadzone są w dwóch logach:

- Log alarmów aktywnych

  Przechowuje informacje o wszystkich aktualnie aktywnych alarmach. Przy odpowiedniej parametryzacje może też posiadać informacje dotyczące alarmów niedawno zakończonych.

- Log alarmów historycznych

  Przechowuje długoterminowo informacje o wszystkich wykrytych alarmach.

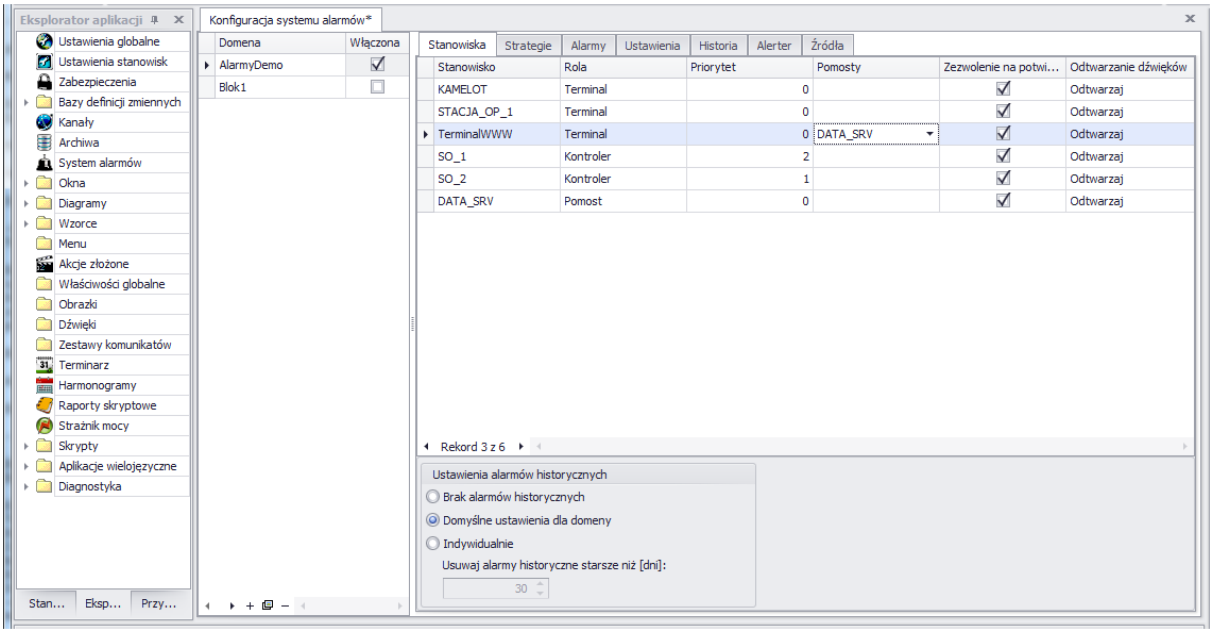
Zarejestrowane alarmy mogą być przeglądane przy pomocy tabel alarmów aktywnych i historycznych. Stan alarmów aktywnych może być także prezentowany bezpośrednio na diagramach synoptycznych.

W przypadku aplikacji wykorzystujących wiele stanowisk Asix.Evo, stan systemu alarmów jest synchronizowany pomiędzy tymi stanowiskami. Stanowiska rozpoczynające pracę uzgadniają stan alarmów aktywnych oraz uzupełniają archiwum alarmów historycznych. Obsługa alarmów przez operatora jest także synchronizowana, np. potwierdzenie alarmu na jednym stanowisku jest natychmiast widoczne na stanowiskach pozostałych.

Alarmy w aplikacji Asix.Evo identyfikowane są przez nazwę. Wszystkie parametry alarmu (opis, kategoria, metoda wykrywania, itp.) są przechowywane w bazie definicji alarmów. Możliwe jest importowanie definicji z arkuszy programu Excel.

2 Parametryzacja systemu alarmów

Parametryzacja systemu alarmów wykonywana jest w panelu *Konfiguracja systemu alarmów* otwieranym poprzez wybór węzła *System alarmów* w panelu *Eksplorator aplikacji*.



Rys. Konfiguracja systemu alarmów.

Panel konfiguracyjny składa się dwóch części. Lewa służy do tworzenia i wyboru domen. W prawej części, w przełączanych zakładkach wykonywana jest parametryzacja domeny. Poszczególne zakładki pozwalają na:

- Stanowiska

  Określenie roli pełnionej w systemie alarmów przez poszczególne stanowiska zdefiniowane w aplikacji.

- Strategie

  Definiowanie strategii wykrywania alarmów.

- Alarmy

  Definiowanie alarmów domeny.

- Ustawienia

Parametryzowanie różnych elementów domeny, np. sposobu pracy logów alarmów.

- Historia

  Określenie zasad tworzenia plików archiwów alarmów.

- Alerter

  Określenie zasad współpracy z modułem AsAlert.

- Źródła

  Określenie źródeł definicji alarmów (arkuszy programu Excel) oraz wykonywanie importu definicji z tych źródeł.

3 Role stanowisk

Każde ze stanowisk uczestniczących w pracy domeny pełni w nim określoną rolę. Stanowiska mogą pełnić jedną z następujących ról:

- Kontroler

  Stanowisko uczestniczy aktywnie w obsłudze alarmów. Kontrolery odpowiedzialne są za wykrywanie alarmów oraz przesyłanie informacji o stanie alarmów pomiędzy stanowiskami. W każdej domenie musi być przynajmniej jedno stanowisko kontrolera. W celu zapewnienia redundancji można stosować większą liczbę kontrolerów. W tym przypadku, w trakcie pracy następuje automatyczne wybranie tzw. kontrolera aktywnego. Pozostałe stanowiska kontrolerów przechodzą chwilowo w tryb pracy zbliżony do stanowisk terminalowych. Pozostają jednak w gotowości do przejęcia funkcji kontrolera aktywnego. Mogą też mieć uruchomione strategie wykrywania alarmów.

  Każdy kontroler posiada priorytet. W sytuacji, gdy dochodzi do uzgodnienia kontrolera aktywnego, jeżeli jest to możliwe, wybierany jest kontroler o najwyższym priorytecie.

- Terminal

  Stanowiska terminali służą jedynie do pokazywania stanu alarmów i obsługi operatorskiej. Terminal nie może działać samodzielnie, musi być podłączony do kontrolera. Połączenie może być bezpośrednie lub za pomocą pomostu. W tym drugim przypadku, w parametryzacji stanowiska, w kolumnie *Pomosty*, należy podać nazwy stanowisk pomostowych. Jeżeli dla danego terminala nie wskazano pomostów, to będzie on się łączyć tylko z aktywnym kontrolerem domeny.

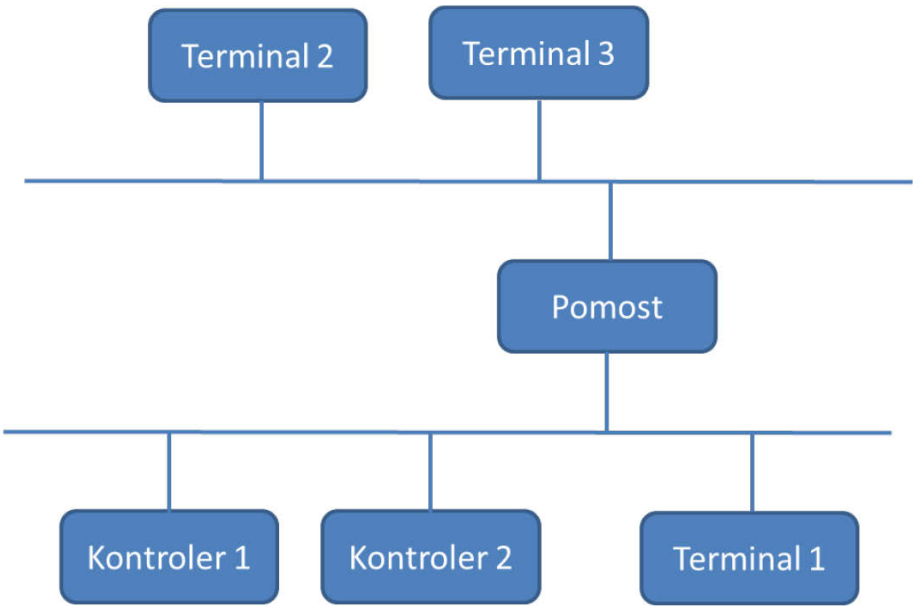
- Pomost

  Funkcjonalność dotycząca obsługi alarmów jest identyczna jak w przypadku terminali. Pomost dodatkowo może służyć jako pośrednik w wymianie danych pomiędzy stanowiskami terminali, a kontrolerem. Pomosty stosuje się wtedy, gdy terminale znajdują się w innej sieci niż stanowiska kontrolerów. Innym zastosowaniem jest użycie pomostu w celu redukcji obciążenia kontrolerów (w szczególności, gdy liczba terminali jest duża).

- Brak

  Jeżeli żadna rola nie została wybrana, to stanowisko nie uczestniczy w pracy domeny i nie ma dostępu do alarmów domeny.

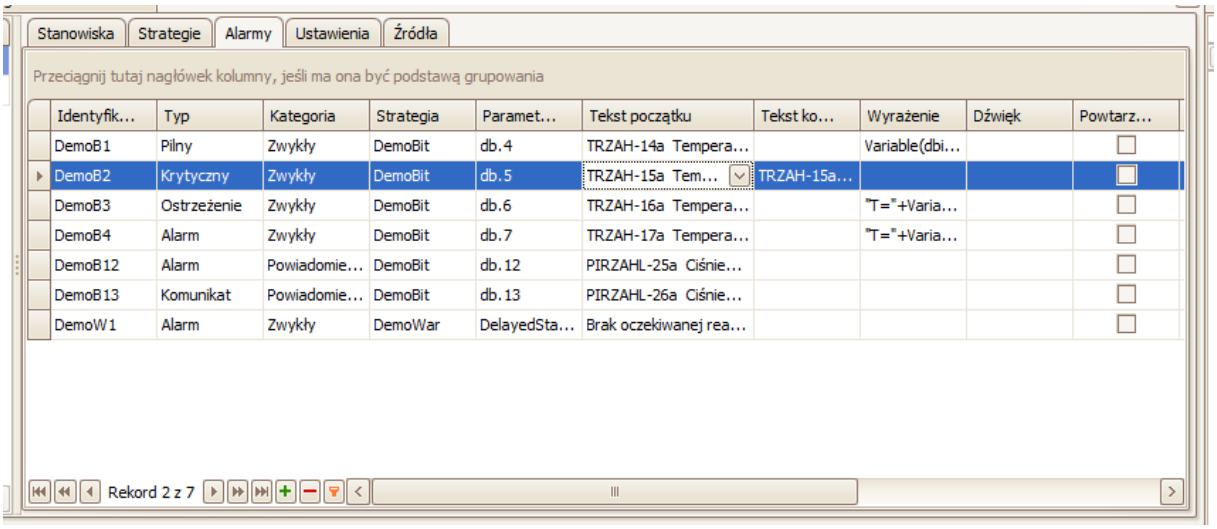
W najprostszym przypadku aplikacja może składać się z pojedynczego stanowiska pełniącego rolę kontrolera domeny. Poniższy schemat pokazuje konfigurację z dwoma redundantnymi kontrolerami. Jeden terminal pobiera dane o alarmach bezpośrednio z kontrolerów. Dwa inne terminale znajdujące się w sieci, która nie ma bezpośredniego kontaktu z kontrolerami, wykorzystują stanowisko pełniące rolę pomostu.



Pewne ograniczenia na możliwe role w systemie alarmów narzuca rola stanowiska w systemie komunikacyjnym. Kontrolerem lub pomostem może być jedynie stanowisko, które jest serwerem komunikacyjnym.

4 Baza definicji alarmów

Wszystkie alarmy używane w aplikacji Asix.Evo muszą zostać wcześniej zdefiniowane. Służy do tego tabela alarmów w zakładce *Alarmy*.



Rys. Tabela definicji alarmów.

4.1 Atrybuty

Definicja alarmu składa się z zestawu atrybutu. Znaczenie poszczególnych atrybutów wyjaśnia poniższa tabelka.

Nazwa	Znaczenie
Identyfikator	Tekstowy identyfikator alarmu
Priorytet	Priorytet alarmy określający ważność alarmu. Możliwe są poniższe priorytety (w kolejności malejącej ważności): • Krytyczny • *Pilny* • *Alarm* • *Ostrzeżenie* • Komunikat

	Alarmy o różnych priorytetach wyświetlane są na tabelach alarmów różnymi kolorami. Alarmy krytyczne i pilne są wymagają selektywnego potwierdzania.
Kategoria	Określa kategorię alarmu. Dostępne są dwie kategorie: alarmy zwykły i powiadomienie. Powiadomienie różni się od alarmu zwykłego tym, że wykrywane jest dla niego tylko zdarzenie początku. Powiadomienia nie maja końców.
Strategia	Określa nazwę strategii używanej do wykrywania alarmu. Sposób parametryzacji strategii wykrywania znajduje się osobnym rozdziale.
Parametry wykrywania	Określa parametry służące do wykrywania alarmu. Postać atrybutu zależy od typu użytej strategii. Sposób parametryzacji strategii wykrywania znajduje się osobnym rozdziale.
Tekst początku	Określa tekst opisujący zdarzenie początku alarmu. Sposób tworzenia tekstu i jego użycia opisuje osobny rozdział.
Tekst końca	Określa opcjonalny tekst opisujący zdarzenie zakończenia alarmu. Jeżeli tekst nie jest zdefiniowany, to koniec alarmu jest opisywany w identyczny sposób jak początek. Sposób tworzenia tekstu i jego użycia opisuje osobny rozdział.
Formatowanie parametrów	W momencie wykrycia alarmu strategia alarmowa może przekazać razem ze zgłoszeniem alarmu pewną ilość dodatkowych parametrów wykrycia. Zestaw i znaczenie parametrów jest specyficzne dla konkretnego typu strategii. W momencie zgłoszenia wartości parametrów są konwertowane na postać tekstową. Atrybut *Formatowanie parametrów* pozwala kontrolować tą konwersję. Atrybut jest kolekcją tekstów formatujących, kolejne elementy służą do formatowania kolejnych parametrów wykrycia. Postać tekstu formatującego jest następująca: [-]*Szerokość:Format* *Szerokość* określa minimalną szerokość wygenerowanego tekstu. W razie konieczności tekst uzupełniany jest spacjami. Opcjonalny znak minusa oznacza wyrównanie pola do lewej strony. *Format* określa sposób konwersji w sposób zgodny z przyjętym w platformie .NET. Przykład: 0:f3

	Konwersja na format zmiennoprzecinkowy z trzema cyframi ułamkowymi. Długość tekstu wynika z wartości konwertowanego parametru. 6:d Konwersja na liczbę całkowitą. Pole będzie miało minimalnie 6 znaków, a liczna będzie wyrównana do prawej strony.
Wyrażenie	Określa wyrażenie, którego wartość w momencie początku i końca alarmu jest rejestrowana w logu. Wartość wyrażenia może być wyświetlona w treści tekstu początku lub końca alarmu. Jest również pokazywana w okienku informacyjnym alarmu dostępnym poprzez przyciski tabel alarmów.
Akcja	Określa akcję skojarzoną z alarmem. Operator może uruchomić tą akcję przyciskiem dostępnym w tabeli alarmów aktywnych.
Bit wykluczenia	Służy do deklaracji bitu zmiennej procesowej, którego stan 0/1 będzie odzwierciedlał aktualny stan wykluczenia alarmu. Mechanizm pozwala na przekazanie do sterownika PLC informacji o wykluczeniu - w efekcie tego sterownik może np. nie generować ostrzegawczych sygnałów dźwiękowych dla alarm wykluczonego. Atrybut ma postać *nazwa_zmiennej.numer_bitu*. Najmłodszy bit zmiennej ma numer 0.
Dźwięk	Określa nazwę dźwięku, które należy odtworzyć w momencie wystąpienia alarmu. Dostępne są wszystkie dźwięki zdefiniowane w puli dźwięków aplikacji.
Powtarzanie dźwięku	Określa czy dźwięk sygnalizujący wystąpienie alarmu ma być powtarzany aż do momentu wykonania przez operatora funkcji wyciszenia dźwięku. Wyciszyć sygnał można przyciskiem w tabeli alarmów aktywnych lub przy pomocy akcji operatorskiej StopSound.
Odczytywanie tekstu	Określa, czy w momencie wykrycia alarmu jego opis tekstowy będzie odczytany przez narratora systemu Windows. Tempo odczytywania tekstów może być ustawione we właściwościach globalnych aplikacji.
Notatka przy potwierdzaniu	Określa, czy w chwili potwierdzenia alarmu przez operatora należy wyświetlić okienko, w którym będzie możliwość wprowadzenia notatki do alarmu. Notatki mogą być przeglądane w obiektach tabel alarmów.

Tryb alertów	Określa czy alarm ma być sygnalizowany przy pomocy sms'ów lub email'i za pośrednictwem programu Alerter. Sposób użycia Alertera opisuje osobny rozdział.
Adresaci alertów	Określa adresatów, do których należy wysłać powiadomienie za pośrednictwem programu Alerter. Sposób użycia Alertera opisuje osobny rozdział.
Atrybuty grupowania	Nazwy i ilość atrybut grupujących jest określana przez projektanta aplikacji. Zasady grupowania opisuje osobny rozdział.

4.2 Teksty opisu alarmów

Definicja alarmu zawiera dwa teksty opisujące alarm – tekst dla zdarzenia początku oraz tekst dla zdarzenia końca. Tekst końca jest opcjonalny. Teksty alarmów używane są w obiektach tabel alarmów. Projektant aplikacji wybiera, które teksty należy wyświetlić w linii alarmów. Do wyboru jest także opcja, w której wyświetlana jest tylko jedna kolumna tekstu, ale jej zawartość jest określana w sposób kontekstowy. Jeżeli zdarzenie dotyczy początku alarmu, to wyświetlany jest tekst początku. Jeżeli zdarzenie dotyczy końca alarmu, to wyświetlany jest tekst końca (o ile został zdefiniowany).

W tekście alarmu można umieścić dodatkowe informacje. Może to być wartość wyrażenia określona w atrybucie *Wyrażenie*, lub wartości parametrów wykrycia alarmu (zgłaszane przez strategię alarmową, skrypty lub akcje operatorskie). Aby wstawić te dodatkowe informacje do tekstu alarmu należy wstawić do tekstu znaczniki postaci {*numer*}. *Numer* 0 oznacza wstawienie wartości wyrażenia podanego w atrybucie *Wyrażenie*. Numery od 1 w górę, oznaczają kolejne parametry wykrycia alarmu.

Przykład:

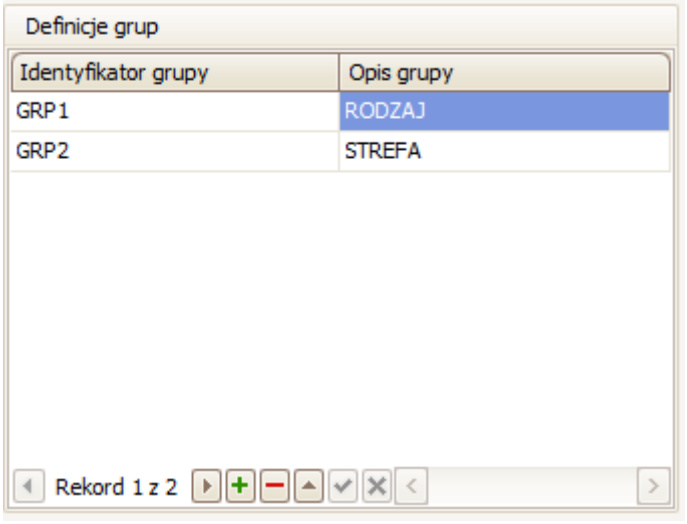
Przekroczony limit {2}, wartość {1}, stan maszyny {0}

Stan maszyny pobierany jest z atrybutu *Wyrażenie*, a limit oraz wartość pochodzą ze zgłoszenia alarmu (np. w skrypcie użytkownika).

W tekście alarmów nie można używać dodatkowego formatowania wstawianych elementów. Wynika to z faktu, że zarówno wyrażenie, jak i parametry wykrycia są konwertowane na tekst już w momencie zgłoszenia alarmu, a tekst alarmu używany jest tylko w momencie jego wyświetlenia. Jeżeli konieczna jest konwersja inna niż domyślna, to w przypadku parametrów wykrycia należy odpowiednio zdefiniować atrybut *Formatowanie parametrów,* a w przypadku wyrażenia zapewnić właściwą konwersję już w samym wyrażeniu (np. przy pomocy funkcji *Format*).

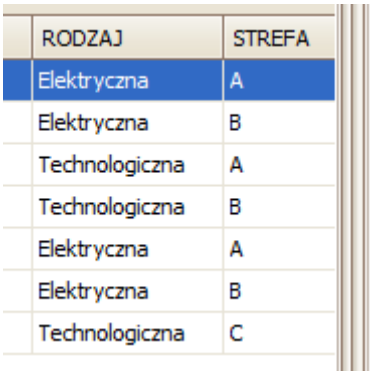
4.3 Grupy alarmów

System alarmów Asix.Evo posiada elastyczny mechanizm selekcji alarmów wykorzystujący tzw. atrybuty grupujące. Pierwszym krokiem w parametryzacji grupowania alarmów jest utworzenie zestawu atrybutów grupujących. Poniższy obrazek pokazuje fragment zakładki *Ustawienia*, w którym można dodać dowolną ilość atrybutów służących do grupowania. Każdy atrybut zdefiniowany jest przez identyfikatorem grupy oraz tekst opisowy. Identyfikator ma znaczenie tylko organizacyjne, tekst opisowy jest używany we wszystkich interfejsach użytkownika.



Rys. Tworzenie zestawów atrybutów grupujących.

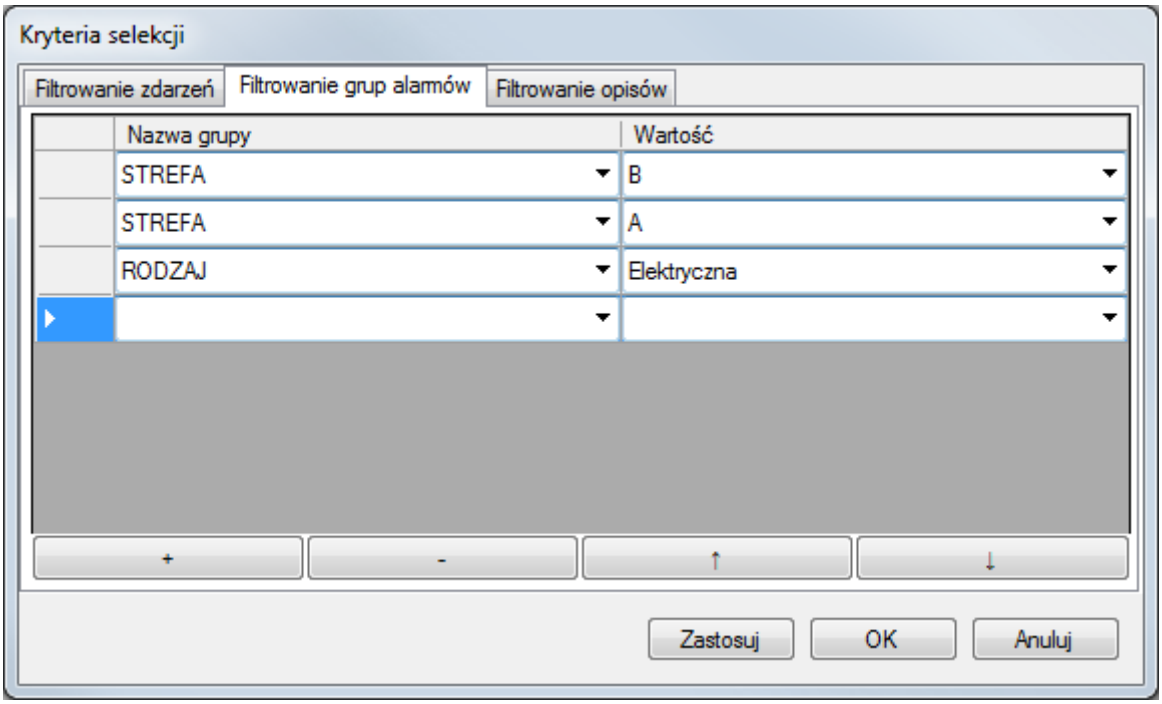
Po zdefiniowaniu atrybutów grupujących, definicje alarmów w zakładce *Alarmy* zostaną rozszerzone o dodatkowe kolumny.



RODZAJ	STREFA
Elektryczna	A
Elektryczna	B
Technologiczna	A
Technologiczna	B
Elektryczna	A
Elektryczna	B
Technologiczna	C

Dla każdego alarmu należy podać wartości atrybutów grupujących. Alarmy o tej samej wartości atrybutu należą do tej samej grupy.

W trybie wykonania aplikacji użytkownik będzie mógł wybierać alarmy pokazywane w tabeli alarmów historycznych, wykorzystując zdefiniowane wcześniej grupy. Po użyciu przycisku *Kryteria selekcji* otwarte zostanie okienko, w którym użytkownik może wybrać pokazywane grupy.



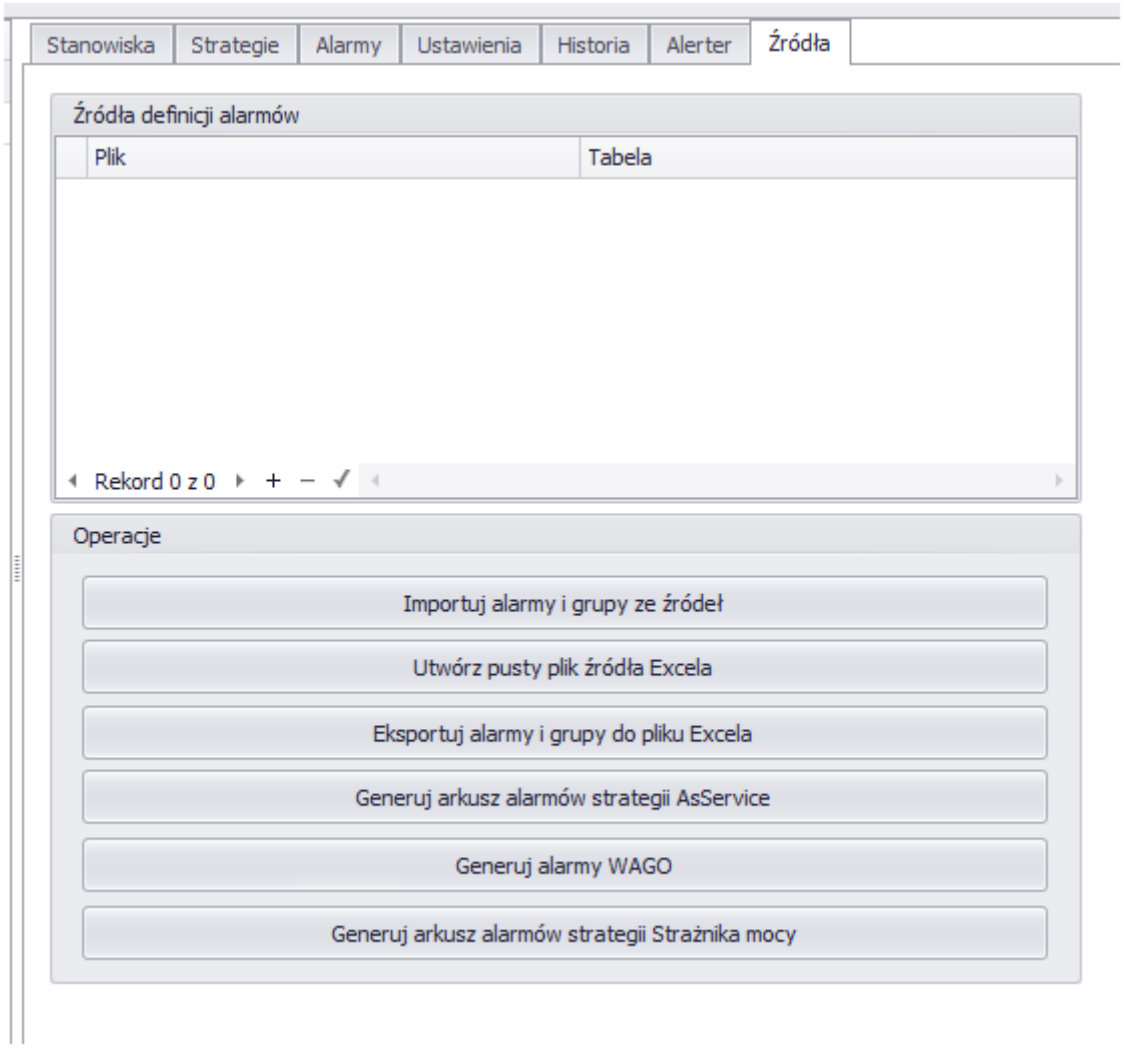
Rys. Kryteria selekcji wyświetlanych grup alarmów.

Powyższy przykład należy interpretować następująco: pokazane zostaną alarmy, których atrybut *Strefa* jest równy *A* lub *B*, a atrybut *Rodzaj* jest równy *Elektryczna*.

I.4 Import z arkuszy Excel

W przypadku aplikacji z większa ilością alarmów korzystne może okazać się definiowanie alarmów w arkuszach programu MS Excel i generowanie bazy definicji alarmów na ich podstawie.

Obsługę funkcji importu z arkuszu Ms Excel zapewnia zakładka *Źródła*.



Rys. Dostęp do funkcji import z arkusza MS Excel.

Można zdefiniować dowolną ilość arkuszy źródłowych. Po naciśnięciu przycisku *Importuj alarmy i grupy ze źródeł* baza definicji alarmów zostanie utworzona od nowa na podstawie aktualnej zawartości arkuszy.

Poniższy obrazek pokazuje przykładowy arkusz definicji alarmów.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Id	Priorytet	Kategoria	TekstPoczatku	TekstKonca	Strategia	Adres	Wyrazenie	Akcja	Dzwiek	Petla	{grp1}Rodzaj	{grp2}Strefa
2	DemoB1	al	N	Bit db.4 ({0})		DemoBit	db.4	Variable(dbinfo1)				Elektryczna	A
3	DemoB2	os	N	Bit db.5		DemoBit	db.5					Elektryczna	B
4	DemoB3	os	N	Bit db.6, {0}		DemoBit	db.6	"T="+Variable(dbinfo1)+", C="+Variable(dbinfo2)				Technologiczna	A
5	DemoB4	al	N	Bit db.7		DemoBit	db.7	"T="+Variable(dbinfo1)+", C="+Variable(dbinfo2)				Technologiczna	B
6	DemoB12	al	BK	Powiadomienie db.12		DemoBit	db.12					Elektryczna	A
7	DemoB13	ko	BK	Powiadomienie db.13		DemoBit	db.13					Elektryczna	B
8	DemoW1	al	N	Brak oczekiwanej reakcji na db.8		DemoWar	DelayedState(Variable(db)&0x100,15,Variable(db)&0x200)					Technologiczna	C
9													
10													

Rys. Arkusz definicji alarmów.

Najprostszą metodą utworzenia poprawnego wzorca arkusza definicji alarmów jest użycie przycisku *Utwórz pusty plik źródła Excela*. Utworzony wzorzec będzie uwzględniał zdefiniowane języki aplikacji oraz dodane atrybuty grupujące.

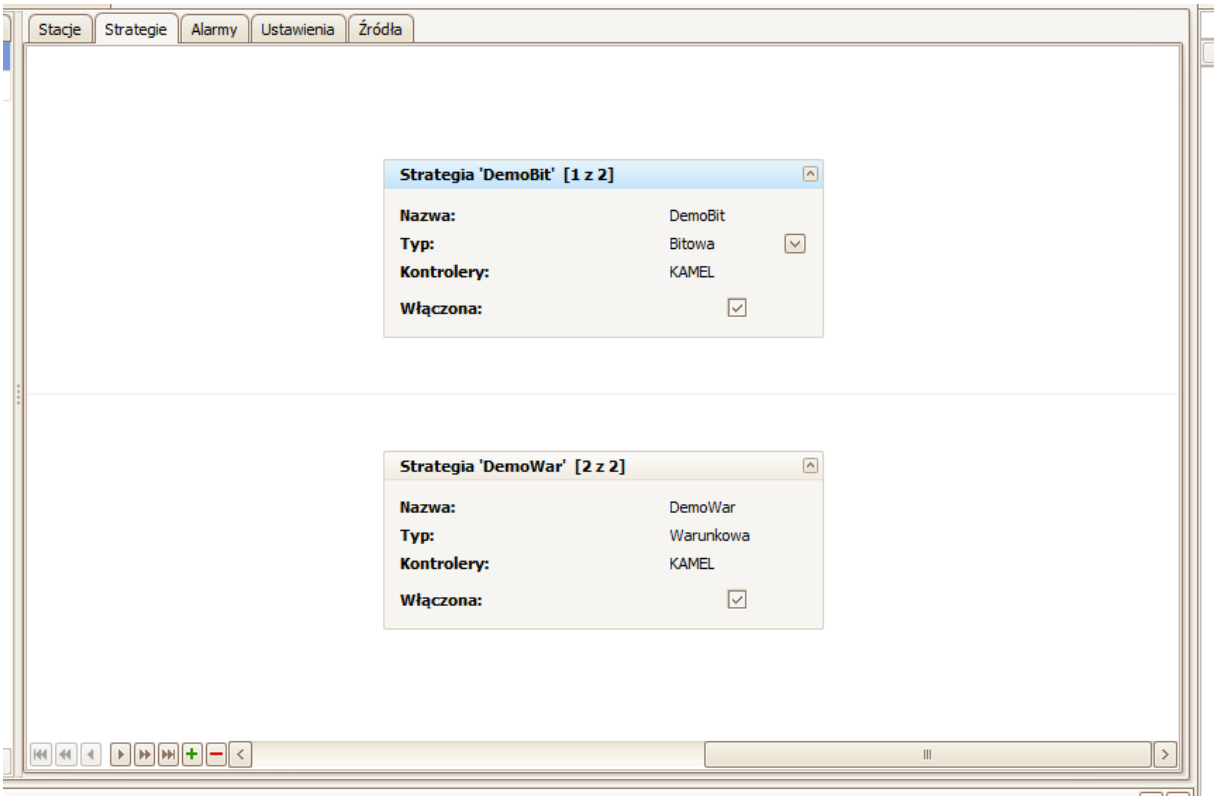
Można też utworzyć plik Excela łącznie z wszystkimi aktualnie zdefiniowanymi alarmami, korzystając z przycisku *Eksportuj alarmy i grupy do pliku Excela*.

W przypadku, gdy aplikacja współpracuje z programem AsService, można użyć przycisk *Generuj arkusz alarmów strategii AsService* w celu utworzenia wzorcowego arkusza alarmów, zawierającego wstępne definicje alarmów dla strategii AsService. Arkusz będzie zawierał definicje alarmów dla wszystkich liczników zdefiniowanych w bazie danych programu AsService, które mają niezerowe wartości limitów eksploatacyjnych lub ostrzegawczych. Po uzupełnieniu o dodatkowe dane arkusz może służyć jako arkusz źródłowy funkcji importowania alarmów.

Można też wygenerować arkusze alarmów w przypadku tworzenia aplikacji typu Asix4WAGO lub wykorzystywaniu alarmów modułu Strażnika Mocy.

5 Strategie wykrywania alarmów

W każdej domenie alarmów można stosować dowolną ilość strategii wykrywania alarmów. Do parametryzowania strategii służy zakładka *Strategie* panelu parametryzacji systemu alarmów.



Strategię wykrywania alarmów, można przez analogię, traktować jak kanał komunikacyjny, którego efektem działania jest zgłaszanie alarmów. Każda strategia może pracować tylko na jednym kontrolerze (aktywnym lub pasywnym) w ramach domeny. O tym, na którym kontrolerze ma zostać uruchomiona dana strategia decyduje zawsze aktywny kontroler. Każda strategia określona jest przez:

- Nazwa strategii

  Nazwa identyfikuje strategię. Alarmy, które w swojej definicji, w atrybucie *Strategia* mają podaną zgodną nazwę, są wykrywane przez tą strategię.

- Typ strategii

  Określa typ strategii, a co za tym idzie sposób wykrywania alarmów. Dostępne typy strategii zostały opisane w dalszej części dokumentacji.

- Lista kontrolerów

  Określa listę nazw stanowisk kontrolerów, na których można strategię uruchomić. Kontroler aktywny, jeżeli będzie miał zezwolenie, to uruchomi strategię na własnym stanowisku. Jeżeli nie, to wybierze jeden z wymienionych kontrolerów.

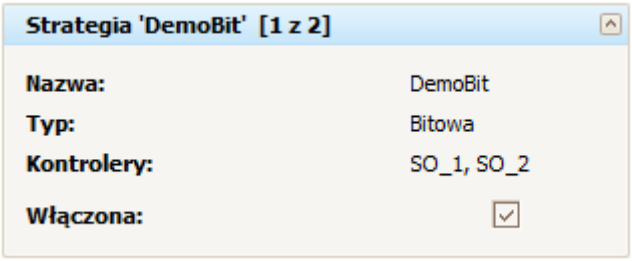
- Flaga aktywności

  Określa czy strategia ma być używana.

- Inne parametry

  W zależności od wybranego typu strategii może być wymagane ustawienie dodatkowych parametrów.

5.1 Strategia bitowa



Rys. Definicja strategii bitowej.

Strategia bitowa przeznaczona jest do wykrywania alarmów na podstawie pojedynczego bitu określonej zmiennej procesowej.

Dla alarmów kontrolowanych przez strategię bitową, atrybut *Parametry wykrywania* w definicji alarmu musi mieć poniższą postać:

nazwa_zmiennej.nr_bitu[r]

, nazwa_zmiennej określa nazwę monitorowanej zmiennej procesowej, *nr_bitu* określa numer kontrolowanego bitu (najmłodszy bit ma numer 0), a opcjonalna litera *r* oznacza użycie odwrotnej logiki wykrywania (zmiana wartości bitu z 1 na 0 oznacza początek alarmu).

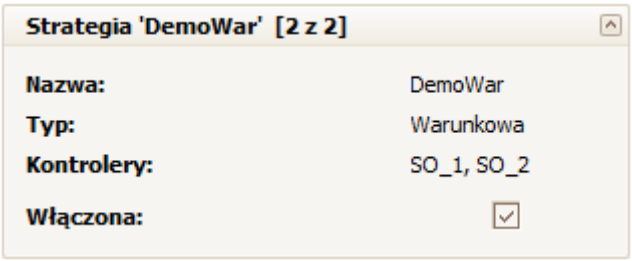
W ramach pojedynczej strategii można kontrolować stan wielu zmiennych procesowych. Nie ma ograniczenia na rozmiar kontrolowanych zmiennych.

Przykład:

al100.1r

Początek alarmu będzie wykrywany w momencie zmiany bitu numer 1 (maska bitowa 0x0002) zmiennej *al100* z wartości 1 na 0.

5.2 Strategia warunkowa



Rys. Definicja strategii warunkowej.

Strategia warunkowa przeznaczona jest do wykrywania alarmów na podstawie monitorowania wartości dowolnego wyrażenia podanego jako parametr wykrywania alarmu.

Dla alarmów kontrolowanych przez strategię warunkową, atrybut *Parametry wykrywania* w definicji alarmu musi mieć postać wyrażenia arytmetycznego. Monitorowane wyrażenie powinno mieć wynik typu logicznego. Zmiana wartości wyrażenia z *false* na *true* oznacza rozpoczęcie alarmu, a zmiana odwrotna koniec alarmu.

Jeżeli wyrażenie służące do określenia wystąpienia alarmu ma konstrukcję wykorzystującą operator porównania (wyliczany jako ostatni element wyrażenia), np. *Variable(v1)<Variable(limit)*, to obie

strony porównania są zgłaszane jako dodatkowe parametry wykrycia alarmu i mogą być wyświetlone w tekście alarmu.

W zestawie funkcji udostępnianych przez Asix.Evo znajdują się funkcje dedykowane specjalnie do użycia w strategii warunkowej:

- DeadBand - kontrola przekraczania limitów z histerezą
- DelayedState - wykrywanie braku przewidywanych zmian stanu procesu
- RateOfChange - monitorowanie prędkości zmian wartości wyrażenia

Przykład:

Variable(Speed1)>1000 && (Variable(SMode)&1)

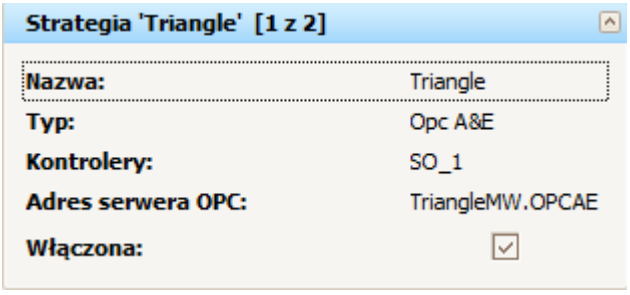
Alarm będzie aktywny, gdy zmienna *Speed1* będzie miała wartość większą od 0 i najmłodszy bit zmiennej *SMode* będzie ustawiony.

DelayedState(Variable(MState)&0x100,15,Variable(MState)&0x200)

Alarm zostanie włączony wtedy, gdy po ustawieniu bitu numer 8 zmiennej *MState*, w ciągu 15 sekund nie dojdzie do ustawienie bitu numer 9.

5.3 Strategia OPC

W strategii alarmów OPC dane o alarmach odbierane są z niezależnego serwera alarmów OPC (ang. OPC Alarm and Event Server). Strategia pozwala zaimplementować w aplikacji systemu Asix sygnalizację alarmów pochodzących z innych systemów.

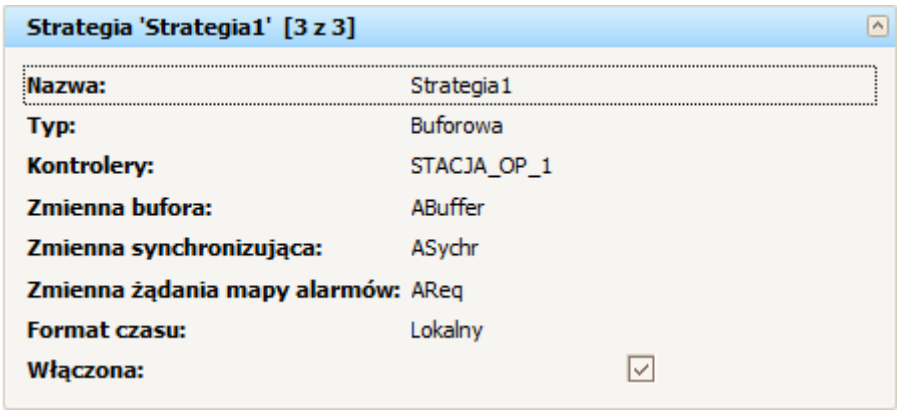


Rys. Definicja strategii OPC.

Parametr Adres serwera OPC służy do podania nazwy serwera OPC, z którego będą pobierane alarmy.

Dla poprawnego działania strategii istotne jest właściwe przemapowanie warunków i podwarunków alarmowych serwera OPC na identyfikatory alarmów systemu Asix. Sposób mapowania określony jest w atrybucie Parametry wykrywania definicji poszczególnych alarmów.

5.4 Strategia buforowa



Rys. Definicja strategii buforowej.

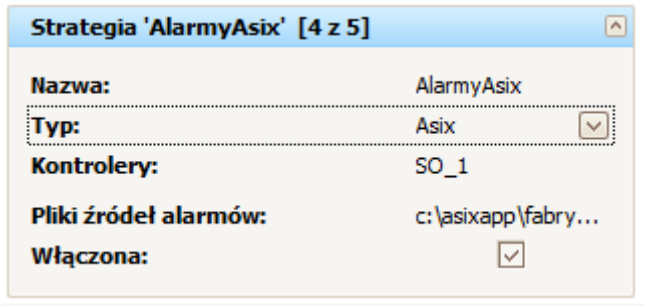
Strategia buforowa jest specjalną strategią, która pozwala na precyzyjne wykrywanie alarmów z rozdzielczością milisekundową i gwarancją zachowania sekwencji zdarzeń. Strategia wymaga jednak współpracy z oprogramowaniem sterownika. Sterownik odpowiedzialny jest za wykrywanie alarmów oraz nadawanie im czasów. Dokładny opis protokołu wymiany danych opisany jest w pliku pomocy systemu Asix 6.

Strategia buforowa wymaga podania dodatkowych parametrów, które związane z protokołem wymiany danych pomiędzy sterownikiem, a aplikacją Asix.Evo:

- Zmienna buforowa – zmienna poprzez, którą są przekazywane informacje o stanie alarmów. Wartość zmiennej musi być tablicą 8- lub 16-bitowych liczb całkowitych bez znaku.
- Zmienna synchronizująca – zmienna służącą do synchronizacji dostępu do bufora alarmów. Wartość zmiennej musi być 16-bitową liczbą całkowitą bez znaku.
- Zmienna żądania mapy alarmów – zmienna poprzez, która Asix.Evo może zażądać przesłanie aktualnego stanu wszystkich alarmów. Wartość zmiennej musi być 16-bitową liczbą całkowitą bez znaku.
- Format czasu – deklaracja w jaki sposób przekazywane są czasy alarmów: w czasie lokalnym czy w czasie UTC.

Dla alarmów kontrolowanych przez strategię buforową, atrybut *Parametry wykrywania* w definicji alarmu musi być numerem kolejnym alarmu w puli alarmów obsługiwanych przez strategię. Pierwszy alarm strategii ma numer 0.

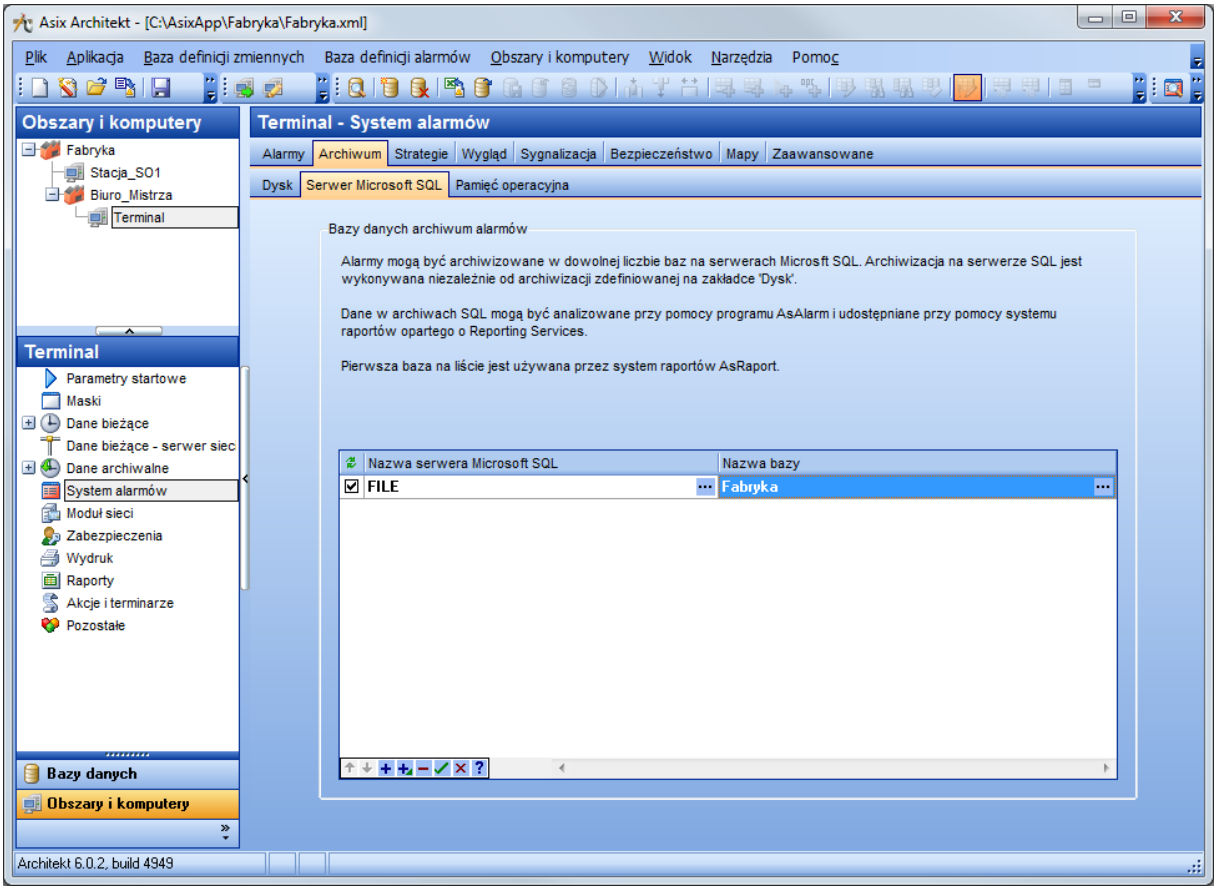
5.5 Strategia Asix



Rys. Definicja strategii Asix.

Strategia Asix jest specjalna strategią, która pozwala przejąć alarmy z klasycznej aplikacji Asix. Stosuje się ją w konfiguracjach, w których aplikacja Asix.Evo służy jako wersja przeglądarkowa klasycznej aplikacji Asix.

Pierwszym krokiem w parametryzacji strategii jest skonfigurowanie aplikacji Asix w taki sposób, aby tworzyła ona plik eksportu alarmów. Wykonać to można przy pomocy programu *Architekt* z pakietu Asix, w zakładce konfiguracji archiwum alarmów typu SQL. Pokazuje to poniższy obrazek:



Rys. Architekt - konfiguracja archiwum alarmów typu SQL.

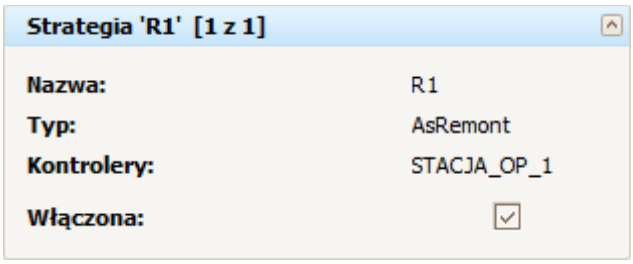
Istotne jest aby w kolumnie *Nazwa serwera Microsoft SQL* podać nazwę *FILE*.

Strategia Asix posiada jeden dodatkowy parametr *Pliki źródeł alarmów*. Należy w nim podać ścieżkę wskazującą na plik, do którego aplikacja Asix zapisuje zdarzenia alarmów. W przypadku podłączenia do aplikacji z redundantnymi serwerami alarmów, należy podać ścieżki do plików eksportowych wszystkich serwerów. Plik eksportowy tworzony jest katalogu logu alarmów aplikacji Asix. W przypadku pokazanym na obrazku prawidłowa nazwa to:

C:\AsixApp\Fabryka\Alarms\FILE_Fabryka.xml

Dodatkowym wymaganiem jest zgodność pomiędzy numerem alarmu w systemie Asix (wersji klasycznej), a identyfikatorem alarmu w Asix.Evo, np. zgłoszenie alarmu numer 100 w Asix, powoduje w Asix.Evo zgłoszenie alarmu o identyfikatorze *100*.

5.6 Strategia AsService



Rys. Definicja strategii AsService.

Strategia AsService przeznaczona jest do wykrywania alarmów na podstawie aktualnego stanu liczników programu AsService. Strategia pozwala wykrywać przekroczenia limitów eksploatacyjnych lub ostrzegawczych wartości liczników.

Dla alarmów kontrolowanych przez strategię AsService, atrybut *Parametry wykrywania* w definicji alarmu musi mieć poniższą postać:

identyfikator_licznika.typ_limitu

-*identyfikator_licznika* określa identyfikator licznika zdefiniowanego w bazie danych programu AsService, a *typ_limitu* określa, przekroczenie którego limitu powoduje zgłoszenie alarmu. Możliwe są następujące wartość: *E* dla limitu eksploatacyjnego, *W* dla limitu ostrzegawczego lub *W2* dla drugiego limitu ostrzegawczego.

W zakładce *Źródła* panelu konfiguracyjnego systemu alarmów znajduje się przycisk, który pozwala na wygenerowanie arkusza Excel z utworzonymi definicjami alarmów dla wszystkich liczników aplikacji AsService.

5.7 Strategia Strażnik Mocy



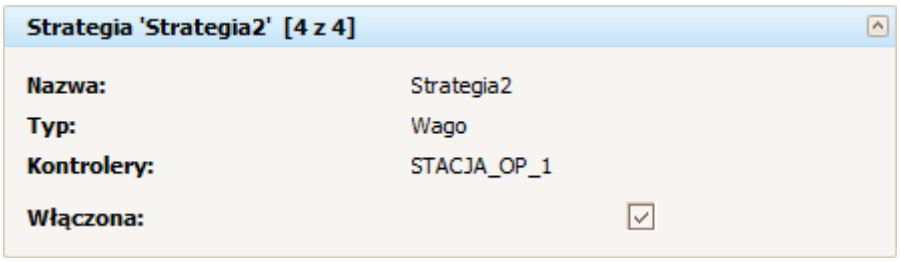
Rys. Definicja strategii Strażnik Mocy.

Strategia*Strażnik Mocy* pozwala na integrację systemu alarmu z modułem Strażnika Mocy. Dla każdego utworzonego miernika można zdefiniować jeden alarm. Alarm ten będzie wszczynany za każdym razem, gdy w bieżącym cyklu miernik wykryje zagrożenie, że średnia wartość śledzonego medium na koniec cyklu przekroczy zakładany limit (np. moc zamówioną) i będzie tak długo utrzymywany, aż zagrożenie nie minie w cyklu bieżącym lub kolejnych cyklach.

Dla alarmów kontrolowanych przez strategię Strażnik Mocy w atrybucie *Parametry wykrywania* należy podać identyfikator miernika.

W zakładce *Źródła* panelu konfiguracyjnego systemu alarmów znajduje się przycisk, który pozwala na wygenerowanie arkusza Excel z utworzonymi definicjami alarmów dla wszystkich aktualnie zdefiniowanych mierników Strażnika Mocy.

5.8 Strategia Wago

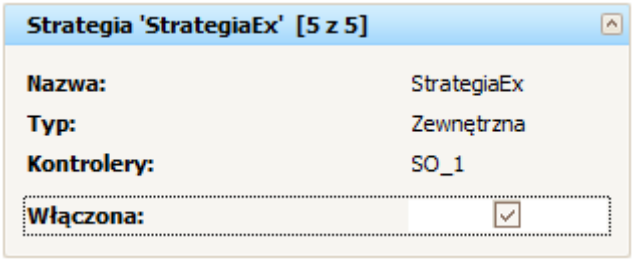


Rys. Definicja strategii Wago.

Strategia Wago przeznaczona jest do obsługi alarmów zgłaszanych przez oprogramowanie sterowników WAGO. Stosowana jest w aplikacjach, które tworzone są przez kreatora aplikacji typu WAGO. W trakcie pracy kreator parametryzuje domenę alarmową, zakłada strategię oraz definiuje alarmy należące do strategii.

Atrybut *Parametry wykrywania* w definicji alarmów jest nazwą instancji struktury Wago. W praktyce jest to prefiks nazwy zmiennych stosowanych do rozpoznawania alarmów (*prefiks_wError_Status* i *prefiks_xError*).

5.9 Strategia zewnętrzna



Rys. Definicja strategii zewnętrznej.

Strategia zewnętrzna przeznaczona jest do obsługi alarmów, których zmiany stanu są zgłaszane przy pomocy akcji operatorskich *StartAlarm* i *EndAlarm,* lub poprzez skrypty przy pomocy funkcji *Raise* i *Cancel* interfejsu *IAlarm.* Strategia sama w sobie nie posiada żadnych metod wykrywania alarmów. Służy jako kontener dla alarmów zgłaszanych metodami zewnętrznymi. Pomimo tego, że strategia sama nie zgłasza alarmów, podlega normalnemu zarządzaniu przez kontroler aktywny. W danym momencie strategia jest aktywna tylko na jednym stanowisku - oznacza to, że jeżeli skrypt działający na różnych stanowiskach zgłosi wielokrotnie ten sam alarm, to zostanie on zaakceptowany tylko na jednym stanowisku (na którym strategia jest uruchomiona).

5.10 Globalna strategia zewnętrzna

Globalna strategia zewnętrzna działa w sposób zbliżony do zwykłej strategii zewnętrznej. Strategia globalna nie jest nigdzie definiowana. Jest tworzona zawsze. Alarmy, które w swojej definicji nie mają podanej nazwy strategii są automatycznie przydzielane do domyślnej strategii zewnętrznej.

Jedyną różnicą w działaniu w stosunku do zwykłej strategii zewnętrznej jest brak kontroli aktywności. Strategia globalna jest zawsze aktywna - zgłoszenie alarmu należącego do niej jest zawsze akceptowane. Projektant aplikacji musi sam zapewnić, że zgłoszenia alarmu nie są duplikowane na różnych stanowiskach.

6 Obsługa alarmów zgłaszanych przez drajwery komunikacyjne typu Asix6

Drajwery typu Asix6 w wyniku swojego działania mogą bezpośrednio zgłaszać alarmy. Typowym przykładem mogą być alarmy od braku komunikacji, ale pełny zestaw możliwości zależy od użytego drajwera Asmena.

W celu przejęcia alarmów tego typu należy:

a) Sparametryzować drajwery Asmena przy pomocy programu Architekt w sposób identyczny jak dla aplikacji typu klasycznego. W typowym przypadku parametryzacja polega na podaniu identyfikatora alarmu związanego z konkretnym zdarzeniem.

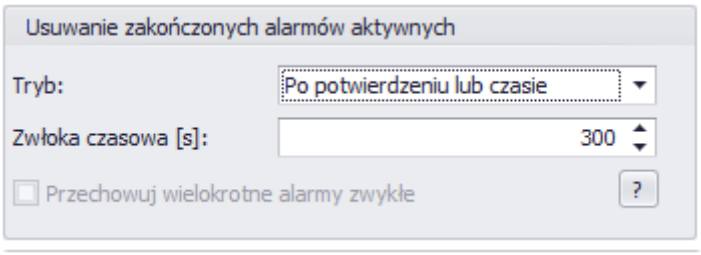
b) Zdefiniować alarmy o identyfikatorach użytych w trakcie parametryzacji drajwerów. Alarmy mogą być zdefiniowane w dowolnej domenie, ale należy zapewnić unikalność identyfikatorów alarmów w ramach wszystkich domen aplikacji. W momencie zgłoszenia alarmu, na podstawie identyfikatora zostanie określone, którą domenę alarmów należy użyć. Alarmy należy skojarzyć ze strategią typu zewnętrznego lub pozostawić atrybut strategii pusty, co oznacza użycie globalnej strategii zewnętrznej.

Strategię nazwaną należy używać wtedy, gdy w konfiguracji sieciowej te same alarmy mogą być zgłaszane przez różne stanowiska – alarmy będą przyjmowane wtedy tylko na jednym stanowisku.

7 Log alarmów aktywnych

Log alarmów aktywnych przechowuje informacje o wszystkich aktualnie rozpoczętych alarmach. W zależności od konfiguracji może też przechowywać informacje o alarmach niedawno zakończonych. Alarmy znajdujące się logu pokazywane są w tabelach alarmów aktywnych.

Tryb pracy logu alarmów aktywnych jest parametryzowany w pokazanym niżej fragmencie zakładki *Ustawienia*.



Rys. Parametryzacja usuwania zakończonych alarmów aktywnych.

Możliwe tryby pracy opisuje poniższa tabela.

Tryb	**Sposób pracy**
Natychmiast	Alarm jest usuwany z logu natychmiast po zakończeniu, niezależnie od jakichkolwiek innych warunków. Należy zwrócić uwagę, że w tym trybie alarmy kategorii *Powiadomienie* nigdy nie znajdą się logu alarmów aktywnych, ponieważ taktowane są jako od razu zakończone.
Po potwierdzeniu	Zakończony alarm jest usuwany dopiero wtedy, gdy został potwierdzony przez operatora.
Po potwierdzeniu lub czasie	Zakończony alarm jest usuwany dopiero wtedy, gdy został potwierdzony przez operatora lub od momentu zakończenia upłynął okres czasu podany w parametrze *Zwłoka czasowa*.

Dla trybu *Po potwierdzeniu* istnieje dodatkowa opcja *Przechowuj wielokrotne alarmy zwykłe*. W ustawieniu domyślnym w logu alarmów aktywnych nigdy nie ma dwóch alarmów zwykłych o tym samym identyfikatorze. Nowy alarm zwykły zawsze usuwa alarm poprzedni nawet, gdy poprzedni nie był jeszcze potwierdzony. Ustawienie opcji *Przechowuj...* spowoduje, że wielokrotne alarmy zwykłe będą przechowywane, aż do momentu ich potwierdzenia. Wymusza to na operatorach potwierdzanie wszystkich alarmów (z ewentualnym dodaniem notatki).

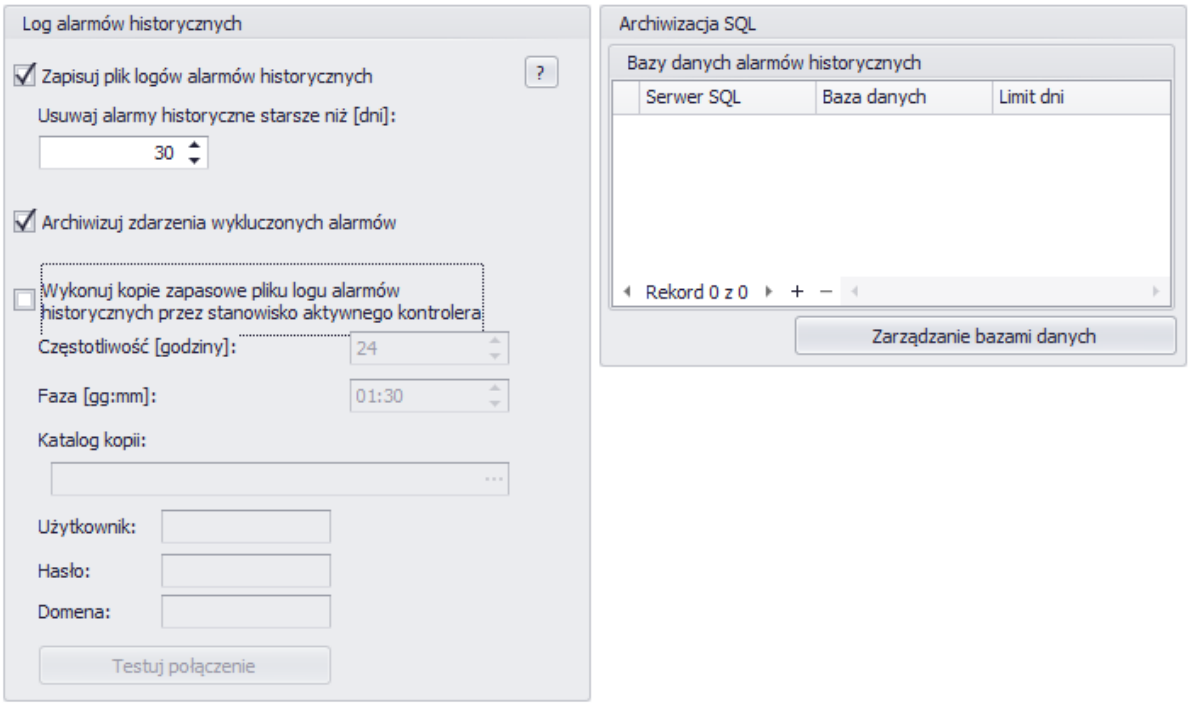
Uwaga:
Niezależnie od użytego trybu usuwania zakończonych alarmów, obiekty tabel alarmów aktywnych mogą wprowadzać własne buforowanie (właściwość *Usuwaj zakończone alarmy*), które może spowodować pokazywanie alarmu usuniętego już z logu alarmów aktywnych. Musi być jednak

spełniony warunek, że tabela była otwarta w czasie przebywania alarmu w logu alarmów aktywnych.

8 Log alarmów historycznych

Log alarmów historycznych przechowuje informacje o wszystkich zdarzeniach alarmowych, zarówno początkach jak i końcach alarmów. Log może przechowywać historię zdarzeń z wielu dni.

Globalny tryb pracy logu alarmów historycznych domeny jest parametryzowany w pokazanym niżej fragmencie zakładki *Historia*.



Rys. Parametryzacja globalnego trybu pracy logu alarmów historycznych domeny.

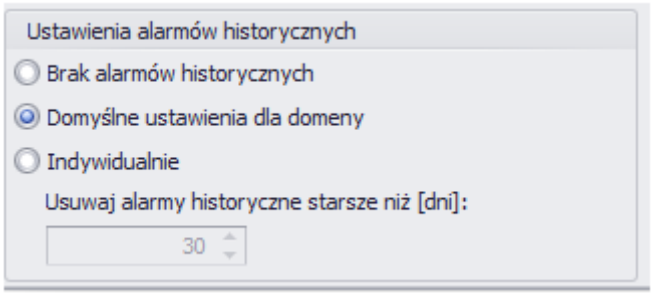
Podstawowy log alarmów, wykorzystywany do bieżącej obsługi alarmów, tworzony jest na każdym stanowisku domeny. Zawartość logów historycznych poszczególnych stanowisk jest wzajemnie synchronizowana. Standardowo log przechowywany jest w lokalnym pliku. W parametrze *Usuwaj*

alarmy historyczne starsze niż można określić okres przechowywania informacji o zdarzeniach alarmowych. Jeżeli opcja *Zapisuj plik logów alarmów historycznych* nie zostanie zaznaczona, to log historyczny jest przechowywany tylko w pamięci w trakcie pracy aplikacji. Po restarcie aplikacji log będzie tworzony od nowa.

Opcja *Archiwizuj zdarzenia wykluczonych alarmów* określa, czy alarmy wykluczone przez operatora mają być archiwizowane w logu alarmów. Jeżeli opcja nie będzie zaznaczona, to w systemie nie zostaną żadne ślady po wystąpieniach wykluczonych alarmów.

Opcja *Wykonuj kopie zapasowe...* i położone niżej elementy panelu służą do zdefiniowania sposobu tworzenia kopii zapasowej pliku logu alarmów historycznych.

Niektóre ustawienia mogą być deklarowane indywidualnie dla poszczególnych stanowisk - w tym przypadku należy użyć ustawień w dolnej części zakładki *Stanowiska*.



Rys. Ustawienia alarmów historycznych.

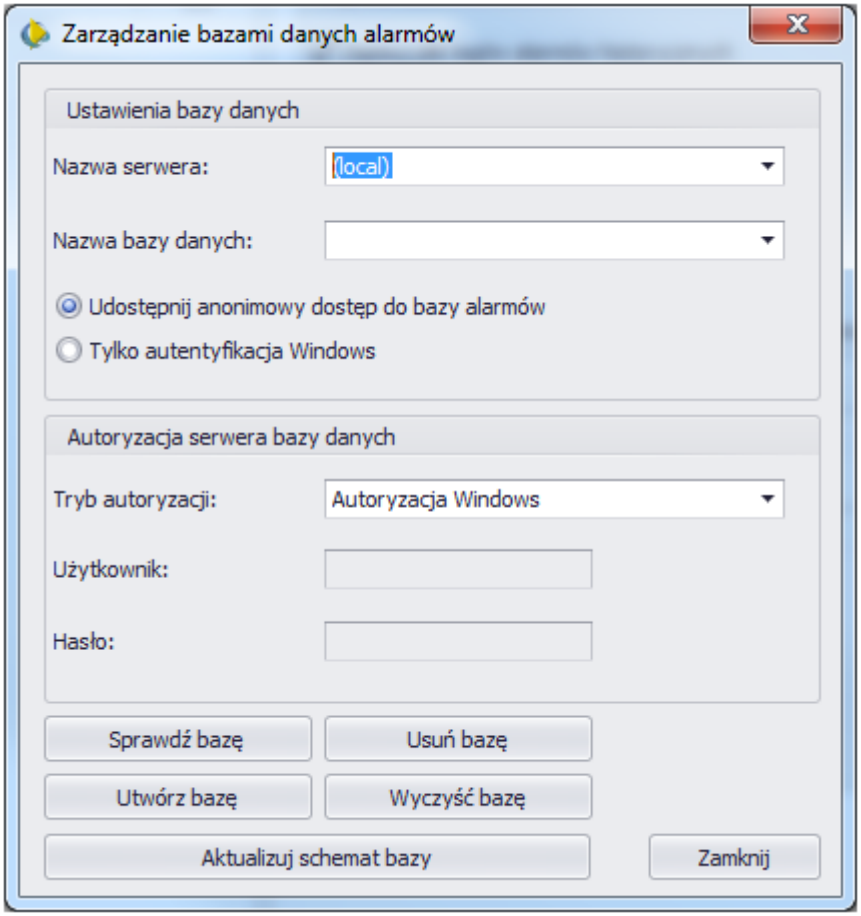
Można w ten sposób, na przykład, ograniczyć zakres logu alarmów na stanowiskach terminalowych.

8.1 Archiwum alarmów SQL

Uzupełnieniem standardowego logu alarmów historycznych jest archiwum alarmów typu SQL. Archiwum to jest przeznaczone do analizy alarmów przy pomocy programu AsAlarm, a także w systemach raportowania.

Archiwa typu SQL parametryzuje się w sekcji *Archiwizacja SQL* zakładki *Historia*. W celu podniesienia niezawodności, można równolegle tworzyć archiwum na kilku serwerach. Jednak nawet w przypadku użycia pojedynczego serwera, Asix.Evo dysponuje mechanizmami buforowania danych pozwalającymi na pracę z chwilową utratą połączenia z serwerem SQL.

Przed rozpoczęciem archiwizacji alarmowa baza danych SQL musi być utworzona. Służy do tego poniższe okno otwierane przyciskiem *Zarządzanie bazami danych*.



Rys. Zarządzanie bazami danych alarmów.

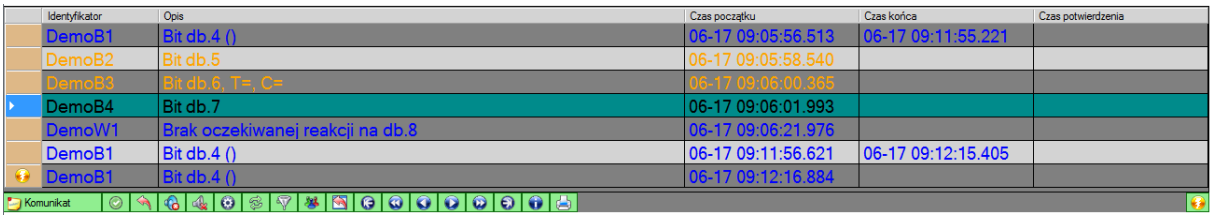
Okno pozwala również na wykonywanie innych zadań administracyjnych. Poświadczenia z grupy *Autoryzacja serwera bazy danych* muszą zapewnić uzyskanie w serwerze SQL uprawnień wystarczających do wykonywania operacji tworzenia i usuwania baz danych.

9 Wizualizacja stanu alarmów

Podstawowym elementem służącym do wizualizacji stanu alarmów są obiekty tabel alarmów. Dostępne są dwa typy tabel, przeznaczone do obsługi alarmów aktywnych i historycznych. Jako uzupełnienie można stosować prezentację stanu alarmów przy pomocy innych obiektów ogólnego przeznaczenia.

9.1 Tabele alarmów aktywnych

Obiekt *Tabela alarmów aktywnych* służy do wyświetlania informacji o alarmach znajdujących się logu alarmów aktywnych.

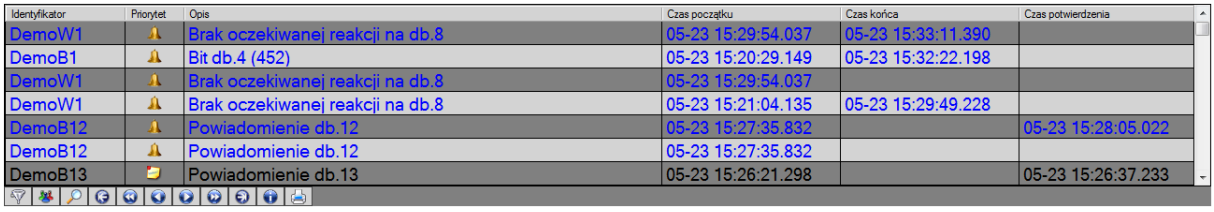


Rys. Tabela alarmów aktywnych.

Obiekt pozwala także na obsługę operatorską. Wykorzystywane są do tego celu przyciski ze zintegrowanego paska narzędziowego.

9.2 Tabele alarmów historycznych

Obiekt *Tabela alarmów historycznych* służy do przeglądania informacji o zdarzeniach alarmach zarejestrowanych w logu alarmów historycznych.

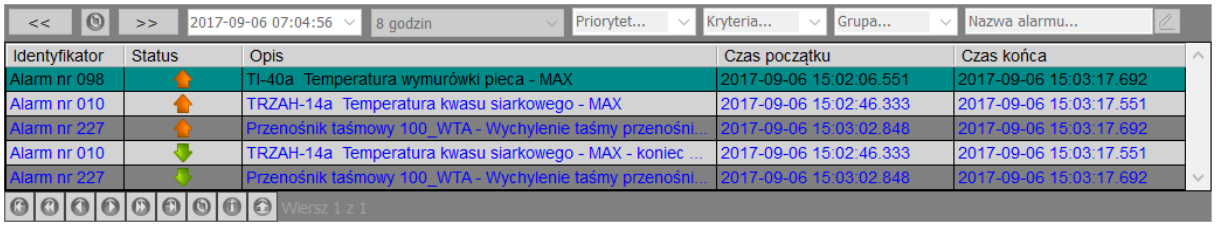


Rys. Tabela alarmów historycznych.

Obiekt posiada wbudowane mechanizmy swobodnego określania zestawu alarmów pokazywanych w tabeli.

9.3 Tabele alarmów SQL

Obiekt *Tabela alarmów* SQL służy do przeglądania informacji o historycznych zdarzeniach alarmów zarejestrowanych w bazie SQL.



Rys. Tabela alarmów SQL.

Obiekt posiada wbudowane mechanizmy swobodnego określania zestawu alarmów pokazywanych w tabeli.

9.4 Inne obiekty

Stan alarmów może być pokazywany na zwykłych obiektach. W tym celu należy użyć funkcji *IsAlarm*, *IsAlarmUnaccepted* lub *IsAlarmExcluded*. Wszystkie te funkcje zwracają informacje na temat stanu wskazanego alarmu. Dostępna jest także funkcja *AlarmsGroupState*, która pozwala monitorować stan alarmów na poziomie całych grup.

Poniższy przykład pokazuje parametryzację obiektu *Tekst* wyświetlającego wartość zmiennej *VA001*. Jeżeli alarm o identyfikatorze *A_VA001* jest aktywny, to wartość zmiennej wyświetlana jest w kolorze czerwonym.

Właściwości podstawowe		
	Główna zmienna	VA001
Właściwości stanowe, stan podstawowy		
	Tekst	#
	Kolor	Black
Właściwości stanowe, stan numer 1		
	Warunek stanu	=IsAlarm(A_VA001)
	Kolor	Red

10 Obsługa alarmów

Większość funkcji związanych z operatorską obsługą alarmów dostępna jest poprzez interfejs obiektu *Tabela alarmów aktywnych*. Niektóre funkcje mogą być wykonane przy pomocy akcji operatorskich. Pozwala to projektantowi aplikacji na zintegrowanie funkcji obsługowych bezpośrednio z obiektami diagramów synoptycznych.

10.1 Potwierdzanie alarmów

Podstawowym mechanizmem potwierdzenia alarmu jest przycisk w pasku narzędziowym obiektu *Tabela alarmów aktywnych*. Przyciśnięcie przycisku potwierdza wszystkie widoczne alarmy. Jedynie w przypadku alarmów o priorytetach pilny i krytyczne wymagana jest wcześniejsza selekcja wiersza alarmu. Alternatywna metoda potwierdzenia to wykonanie akcji *AcceptAlarm* użytej w obsłudze zdarzenia obiektu diagramu.

Funkcja potwierdzania alarmów jest kontrolowana przez system uprawnień. Pierwszym warunkiem jest zezwolenie na potwierdzanie alarmów na stanowisku. Jest ono deklarowane w ustawienia stanowiska w panelu parametryzacji systemu alarmów. Drugi warunek to indywidualne uprawnienia aktualnie zalogowanego użytkownika. Musi on pełnić rolę, która posiada uprawnienie *Prawo potwierdzania alarmów*.

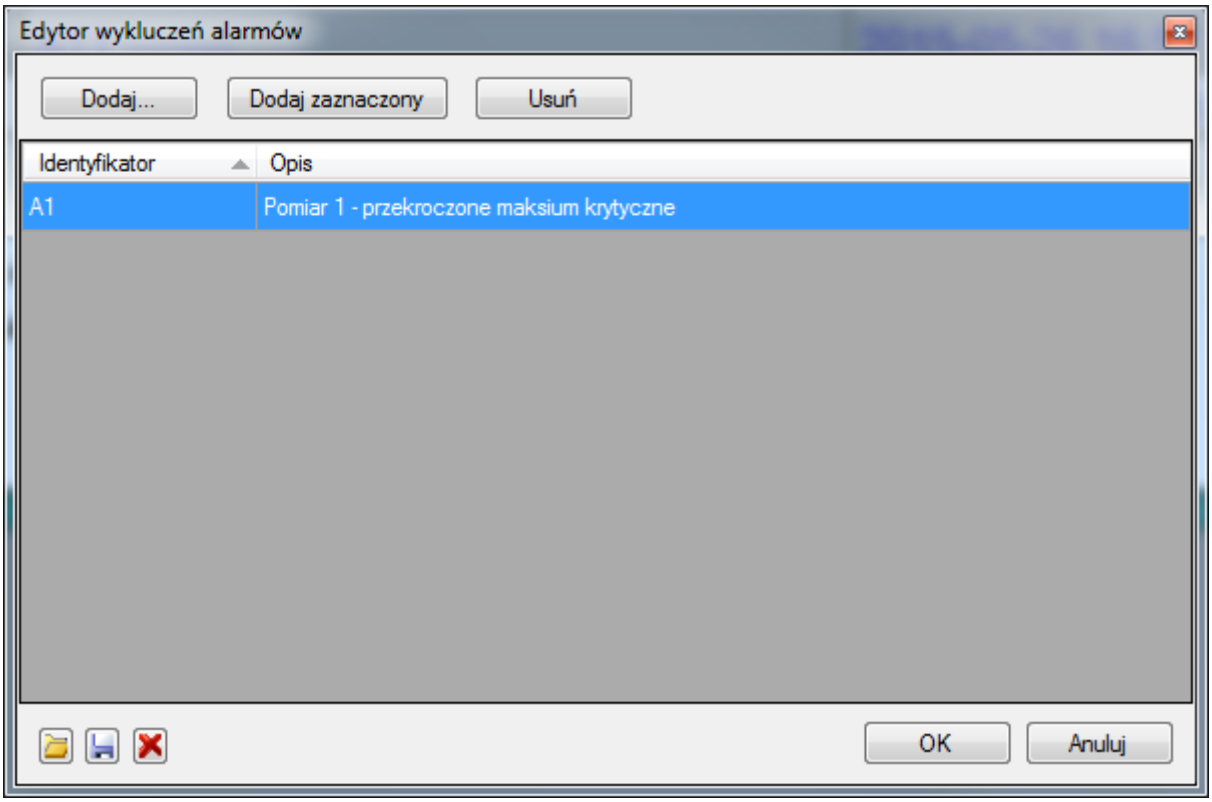
Jeżeli wykorzystywana jest aplikacja Asix Mobile, to potwierdzanie alarmów wykonywane z tej aplikacji, wymaga dodatkowego zezwolenia w ustawieniach domeny alarmowej *Zezwalaj na potwierdzanie alarmów z aplikacji Asix Mobile.*

Potwierdzenie alarmu może być powiązane z koniecznością wprowadzania notatki przez operatora, jeżeli alarm został zdefiniowany, jako tego wymagający (atrybut alarmu *Notatka przy potwierdzeniu*).

10.2 Wykluczenia alarmów

Mechanizm wykluczania alarmów pozwala wyłączyć z obsługi dowolny alarm. Wszystkie zdarzenia dotyczące alarmu wykluczonego są ignorowane. Wykluczenia są przydatne w celu eliminacji alarmów, które ze względu na jakąś awarię są nieprawidłowo wykrywane.

Do obsługi wykluczeń służy poniższe okno otwierane przyciskiem paska narzędziowego obiektu *Tabela alarmów aktywnych.*



Rys. Edytor wykluczeń alarmów.

W tabeli pokazane są wszystkie alarmy, które są aktualnie wykluczone. Przycisk *Dodaj zaznaczony* pozwala na łatwe wykluczenie alarmu aktualnie wybranego w tabeli alarmów aktywnych, dla której okno wykluczeń zostało otwarte.

Okno edytora wykluczeń pozwala także na zapis aktualnych ustawień do pliku oraz późniejszy odczyt tych plików.

Funkcja wykluczania alarmów jest kontrolowana przez system uprawnień. Aktualnie zalogowany użytkownik musi pełnić rolę, która posiada uprawnienie *Prawo edycji wykluczeń alarmów*.

10.3 Wykluczenia jednorazowe

Mechanizm wykluczeń jednorazowych pozwala operatorowi na usunięcie z tabeli dowolnego aktywnego alarmu. W przeciwieństwie jednak do standardowych wykluczeń, ponowne wykrycie alarmu zostanie zgłoszone w normalny sposób.

Jednokrotne wykluczenie wykonywane jest poprzez wskazanie alarmu w tabeli alarmów aktywnych i przyciśnięcie odpowiedniego przycisku paska narzędziowego.

Funkcja jednorazowego wykluczania alarmów jest kontrolowana przez system uprawnień. Aktualnie zalogowany użytkownik musi pełnić rolę, która posiada uprawnienie *Prawo jednokrotnego wykluczania alarmów*.

10.4 Filtrowanie alarmów

Mechanizm filtrowania alarmów pozwala ograniczyć liczbę zgłaszanych alarmów. Mechanizm eliminuje krótkotrwałe zdarzenia alarmowe, które mogą powstawać w wyniku złego działania czujników.

Możliwe są dwa tryby filtrowania:

- Filtrowanie krótkich alarmów

  Ignorowane są krótkotrwałe zmiany stanu alarmu. Stan musi utrzymać się przez określony czas T1, aby zmiana została zaakceptowana.

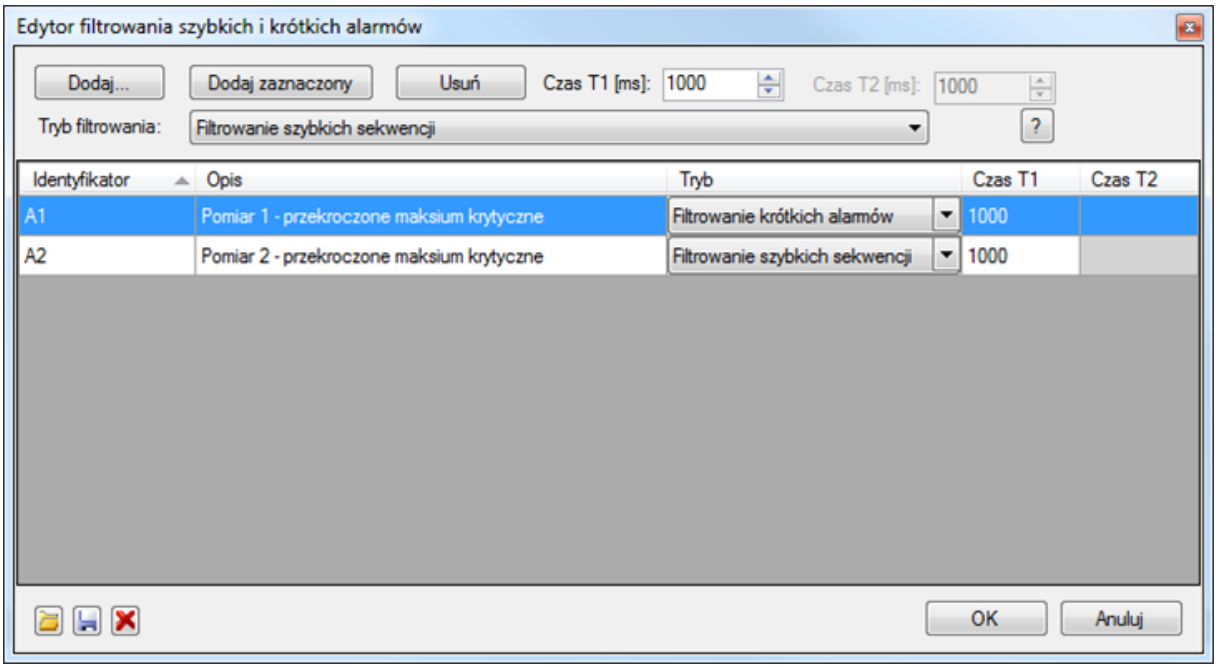
- Filtrowanie szybkich sekwencji

  Ignorowane są sekwencje szybko następujących po sobie zmian stanu alarmu (tzw. „drgające styki"). Pierwsza, nawet krótka, zmiana jest natychmiast akceptowana, ale następne zostaną przyjęte dopiero po upływie czasu karencji T1.

- Filtrowanie krótkich i rzadko występujących alarmów

  Ignorowane są sekwencje krótkotrwałych wystąpień warunku alarmu, jeżeli trwale zanikną przed upływem czasu karencji T1. Wyłączenie zostanie przyjęte dopiero po upływie czasu karencji T2 od ostatniego zaniku warunku alarmu.

Do obsługi filtrów służy poniższe okno otwierane przyciskiem paska narzędziowego obiektu *Tabela alarmów aktywnych.*



Rys. Edytor filtrowania szybkich i krótkich alarmów.

Okno edytora filtrów pozwala także na zapis aktualnie ustawionych filtrów do pliku oraz odczyt tych plików.

Funkcja filtrowania alarmów jest kontrolowana przez system uprawnień. Aktualnie zalogowany użytkownik musi pełnić rolę, która posiada uprawnienie *Prawo edycji filtrów alarmów*.

10.5 Sygnalizacja dźwiękowa

Moment wykrycia alarmu może być sygnalizowany sygnałem dźwiękowym. Nastąpi to, gdy w definicji alarmu, w atrybucie *Dźwięk* została podana nazwa pliku dźwiękowego. Możliwe jest też odczytywanie tekstów alarmów przez narratora systemu Windows, gdy zostanie ustawiony atrybut *Odczytywanie tekstu*. W zależności od ustawienia atrybutu *Powtarzanie dźwięku* sygnał jest odtwarzany jednokrotnie lub powtarzany do momentu wyciszenia.

Powtarzający się sygnał może zostać wyciszony automatycznie. Dochodzi do tego, gdy system podejmie decyzję o odtworzeniu innego sygnału (np. ze względu na pojawienie się alarmu o wyższym priorytecie) lub alarm powiązany z sygnałem został zakończony lub potwierdzony. W drugim przypadku konieczne jest wybranie w ustawieniach domeny alarmowej trybu wyciszania alarmów *Po potwierdzeniu lub usunięciu*.

W przypadku wyciszania ręcznego operator ma do dyspozycji dwie metody. Pierwszą jest użycie przycisku wyciszenia na pasku narzędziowym obiektu *Tabela alarmów aktywnych*. Druga metoda to użycie akcji operatorskiej *StopSound*.

W przypadku aplikacji złożonych z kilku stanowisk, sygnał dźwiękowy może być odtwarzany na kilku stanowiskach. Pasek narzędziowy tabeli alarmów ma dwa różne przyciski wyciszenia. Pierwszy

wycisza dźwięk tylko na lokalnym stanowisku, drugi wykonuje to na wszystkich stanowiskach domeny.

To czy wybrane stanowisko odtwarza dźwięki alarmowe zależy od ustawień w zakładce *Stanowiska* panelu parametryzacji systemu alarmów. Możliwe są trzy tryby pracy:

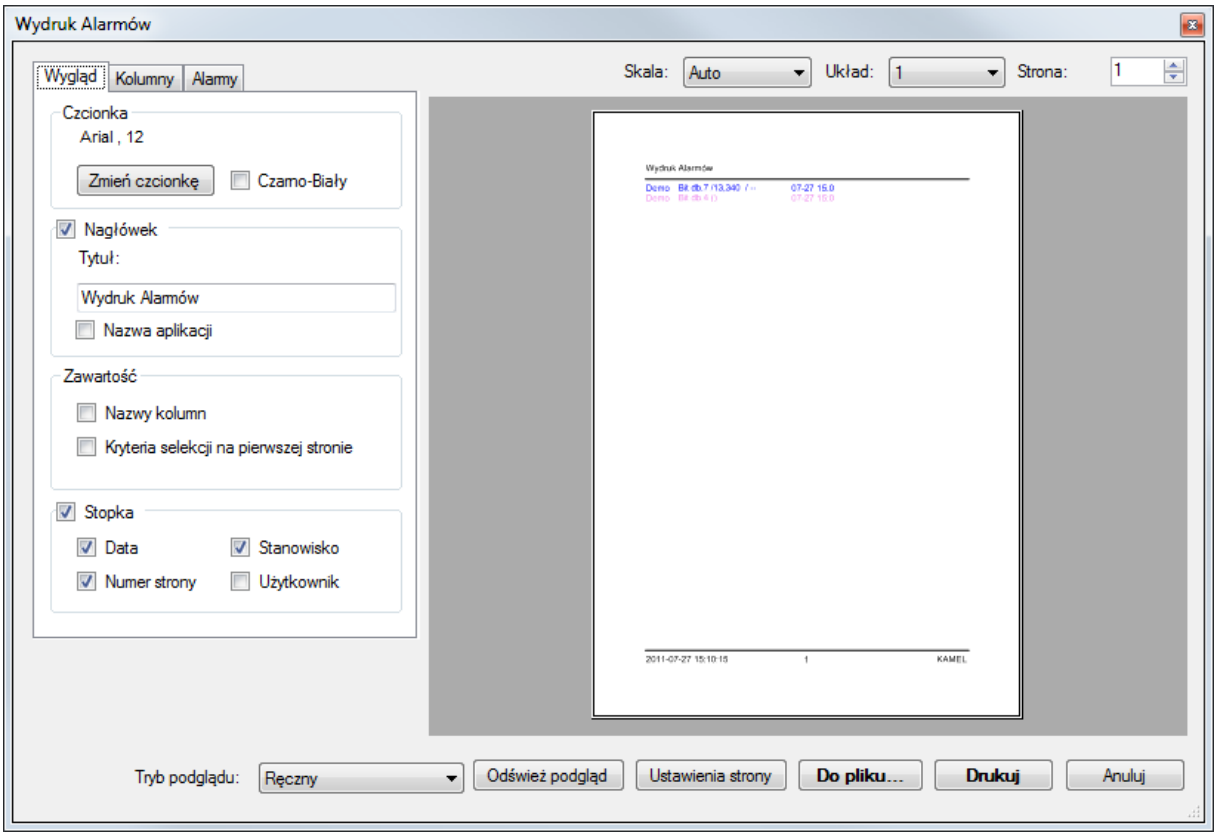
a) *Odtwarzaj* – stanowisko zawsze odtwarza dźwięki alarmowe

b) *Nie odtwarzaj* – stanowisko nigdy nie odtwarza dźwięków alarmowych

c) *Gdy aktywny kontroler* – stanowisko odtwarza dźwięki tylko wtedy, gdy jest kontrolerem aktywnym. Pozwala to zapewnić, że w przypadku stosowania kilku redundantnych stanowisk operatorskich dźwięki są odtwarzane tylko przez jedno stanowisko.

10.6 Drukowanie

Funkcja wydruku alarmów dostępna jest w obiektach *Tabela alarmów aktywnych* i*Tabela alarmów historycznych*. Przyciśnięcie przycisku wydruku powoduje otwarcie poniższego okna, w którym operator może ustawić opcje wydruku.



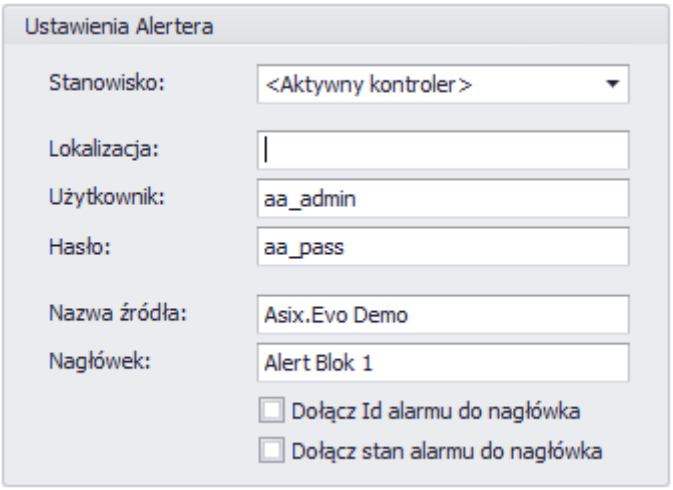
Rys. Okno wydruku alarmów.

W oknie wydruku można określić wygląd wydruku, zestaw drukowanych danych oraz kryteria wyboru alarmów. Oprócz wydruku na drukarce, można zapisać dane o alarmach w pliku tekstowym.

11 Współpraca z AsAlert

Program AsAlert w aplikacjach systemu Asix służy do wysyłania powiadomień o alarmach. W zależności od konfiguracji programu, powiadomienia mogą być wysyłane przy pomocy komunikatów SMS lub poczty elektronicznej.

System alarmów Asix.Evo może współpracować z programem AsAlert. Parametryzacja sprzęgu wykonywana jest dwustopniowo. W pierwszym kroku deklarowane są podstawowe parametry komunikacyjne. Służy do tego zakładka *Alerter*.



Rys. Ustawienia Alertera.

Znaczenie parametrów jest następujące:

- *Stanowisko*

  Określenie stanowiska aplikacji Asix.Evo, które będzie inicjowało wysłanie powiadomienia. Może to być nazwa dowolnego stanowiska uczestniczącego w pracy domeny. Można też wybrać opcję, w której powiadomienia wysyła zawsze aktywny kontroler. Właściwy wybór zależy od konfiguracji systemu. W każdym przypadku stanowisko powinno być włączone przez cały czas. Dlatego zazwyczaj alerty wysyłane są ze stanowisk kontrolerów. Jeżeli AsAlert jest zainstalowany na komputerze jednego z kontrolerów, to z reguły

powiadomienia powinno wysyłać to stanowisko. W przeciwnym wypadku z reguły lepsza jest opcja aktywnego kontrolera.

- *Lokalizacja*

  Systemowa nazwa komputera, na którym zainstalowany jest program AsAlert. W przypadku korzystania z lokalnej instalacji należy parametr pozostawić pusty.

- *Użytkownik*

  Nazwa użytkownika programu AsAlert na konto, którego będą wysyłane powiadomienia.

- *Hasło*

  Hasło użytkownika programu AsAlert na konto, którego będą wysyłane powiadomienia.

- *Nazwa źródła*

  Dowolny tekst, który w wysłanym alercie zostanie użyty, jako nadawca wiadomości.

- *Nagłówek*

  Dowolny tekst, który w wysłanym alercie zostanie użyty, jako tytuł wiadomości.

- *Dołącz Id alarmu do nagłówka*

  Określa czy w nagłówku alertu ma zostać umieszczony identyfikator alarmu.

- *Dołącz stan alarmu do nagłówka*

  Określa czy w nagłówku alertu ma zostać umieszczona informacja o tym, jaki rodzaj zdarzenia wystąpił: początek lub koniec alarmu.

Drugim krokiem w parametryzacji sprzęgu jest odpowiednie zdefiniowanie atrybutów *TrybAlertów* i *Adresaci Alertów* w definicji każdego alarmu, dla którego mają być wysyłane alerty. Tryb alertów pozwala wybrać, kiedy alerty są wysyłane: możliwe jest wysyłanie w momentach początku i/lub końcu alarmu. W atrybucie *AdresaciAlertów* należy podać nazwy adresatów zdefiniowanych w programie AsAlert, do których powiadomienie ma być wysłane.

Asix.Evo - Harmonogramy

1 Cechy modułu harmonogramów

Moduł harmonogramów pozwala na zaawansowane planowanie automatycznego wykonywania akcji operatorskich. W pewnym zakresie w swoim działaniu jest zbliżony do modułu terminarza. Podstawowe różnice pomiędzy tymi modułami są następujące:

a) Harmonogramy posiadają bogatsze możliwości precyzyjnego określenia momentów wykonania akcji. Terminarz opiera się na stałym cyklu wykonywania.

b) Terminarz może synchronizować wykonywanie akcji z zajściem innym warunków niż czasowe (np. konkretną wartością zmiennej procesowej). Harmonogramy nie mają takiej możliwości - aczkolwiek wykonanie akcji może być uzależnione od tego, czy w danym momencie zajdzie pewien warunek.

c) Praca harmonogramów może być sterowana przez operatora w trybie wykonywania aplikacji. Moduł terminarza jest całości konfigurowany przez projektanta aplikacji.

1.1 Podstawowe pojęcia

Dla zrozumienia zasad działania modułu harmonogramów konieczna jest znajomość następujących terminów:

a) **Harmonogram**

Harmonogram to podstawowe pojęcie. Można go opisać jako zadanie, które ma być wykonywane co jakiś czas. Składa się z definicji oraz zestawu cykli wykonania. Definicja harmonogramu opisuje, jakie akcje operatorskie mają być wykonywane (wraz z ich parametrami). W ramach definicji określane są też zasady pracy w konfiguracjach sieciowych.

b) **Cykl harmonogramu**

Cykl harmonogramu określa, w jakich momentach czasu mają być wykonywane akcje operatorskie harmonogramu. Z każdym harmonogramem można powiązać dowolną liczbę cykli. Cykle mogą być definiowane przez użytkownika aplikacji w trybie wykonywania.

c) **Kategoria dni**

Kategorie dni służą do powiązania wybranych dni w jedną całość. Tak zdefiniowane kategorie mogą być następnie użyte w definicjach cykli. Istnieją kategorie predefiniowane, które nie mogą być zmieniane przez użytkownika, oraz kategorie, które tworzy użytkownik aplikacji.

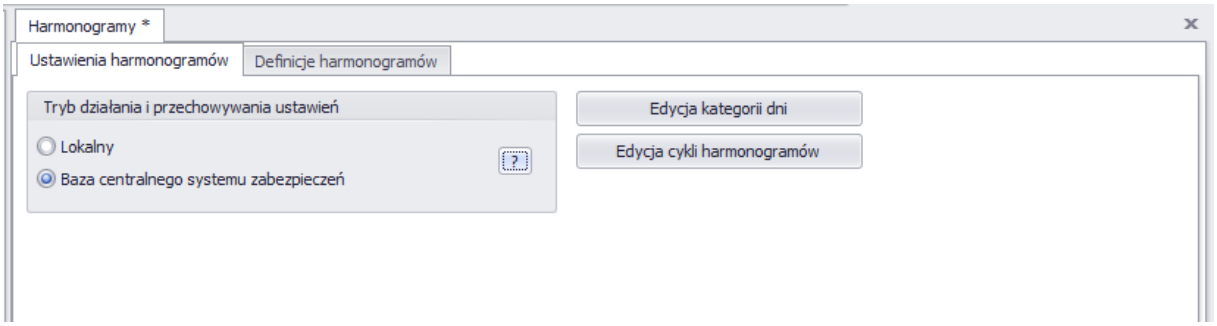
1.2 Praca w trybie sieciowym

Moduł harmonogramów w pełni wspiera pracę wielostanowiskową. W takich konfiguracjach edycja cykli harmonogramów jest możliwa na wszystkich stanowiskach. Wszelkie zmiany są natychmiast widoczne w całej sieci stanowisk. Harmonogram jest wykonywany w danym momencie na pojedynczym stanowisku, a w razie wyłączenia tego stanowiska obsługę automatycznie przejmuje inne stanowisko. Rola poszczególnych stanowisk jest niezależnie definiowana dla każdego harmonogramu.

2 Parametryzacja harmonogramów

Panel definicyjny modułu harmonogramów otwierany jest za pośrednictwem węzła Harmonogramy w panelu eksploratora aplikacji.

2.1 Tryb pracy modułu



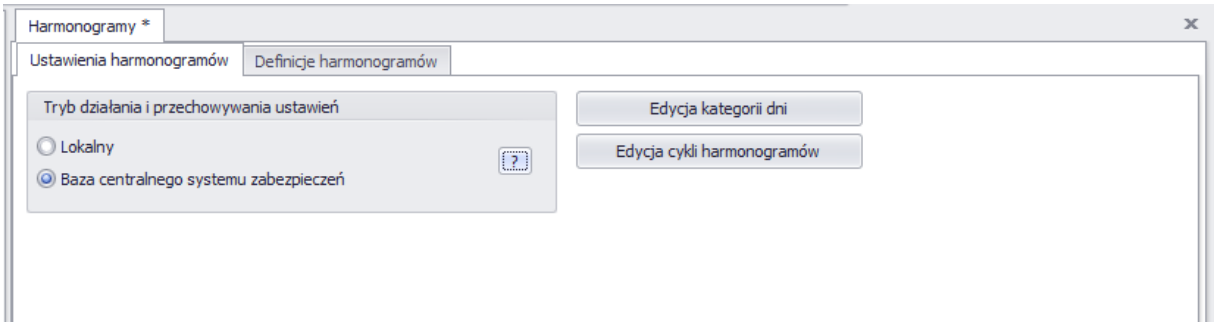
Rys. Panel definicyjny modułu harmonogramów.

W zakładce *Ustawienia harmonogramów* należy zadeklarować miejsce zapisu konfiguracji modułu, a co za tym idzie sposób działania w konfiguracjach wielostanowiskowych. Przechowywanie w lokalnym pliku aplikacji jest dobre dla prostych systemów jednostanowiskowych lub wtedy, gdy harmonogramy będą obsługiwane tylko na jednym stanowisku. W przypadku innych konfiguracji należy wybrać zapis w centralnej bazie zabezpieczeń (**należy wcześniej założyć bazę w ustawieniach systemu zabezpieczeń aplikacji**). Opcja ta zapewnia niemal natychmiastową propagację zmian konfiguracji na wszystkie stanowiska w ramach aplikacji.

Przyciski *Edycja kategorii dni* i *Edycja cykli harmonogramów* otwierają okna przeznaczone do wstępnego definiowania kategorii dni i cykli utworzonych harmonogramów. Okna te dostępne są także użytkownikom aplikacji w trybie wykonywania. Zostaną one opisane w rozdziale dotyczącym obsługi modułu harmonogramów.

2.2 Definiowanie harmonogramu

Zakładka *Definicje harmonogramów* służy do definiowania harmonogramów.



Rys. Panel definicyjny modułu harmonogramów - definicje harmonogramów.

Lista z lewej strony panelu wraz z jej paskiem narzędziowym służy do zakładania, usuwania i zarządzania harmonogramami. Kolumna *Aktywny* określa, czy harmonogram ma być realizowany w trybie wykonywania aplikacji.

Prawa część panelu służy do ustawiania parametrów harmonogramu. Podstawowe znaczenie ma wybór typu harmonogramu. Typ *Pojedyncza akcja* oznacza, że w momentach określonych przez cykl harmonogramu będzie wykonywana akcja operatorska (definiowana w polu *Akcja początku*) i na tym skończy się obsługa tego cyklu. W przypadku typu *Akcje początku i końca*, w momentach określonych przez cykl harmonogramu, będzie wykonywana akcja operatorska (definiowana w polu *Akcja początku*), a następnie po upłynięciu czasu określonego w cyklu wykonana będzie druga, kończąca akcja operatorska (definiowana w polu Akcja końca).

W sekcji *Definicje parametrów* definiowane są opcjonalne parametry harmonogramu. Mechanizm ten pozwala tworzyć harmonogramy, których działanie może być sterowane w definicjach cyklów wykonawczych (w ten sposób każdy cykl może wykonywać akcje harmonogramu w inny sposób). W treści akcji harmonogramu można pobrać wartość parametru przy pomocy funkcji *Parameter*, np. *SetVariable*(*WTEMP*, *Parameter*(*Temperatura*).

W definicji harmonogramu można podać tylko jedną akcję początku i jedną akcję końca. Jeżeli zachodzi konieczność wykonania większej liczby operacji, należy użyć akcji operatorskich złożonych *ActionSet* lub *Actions*.

Zasadniczo, praca harmonogramu jest uzależniona wyłącznie od cykli czasowych. Jeżeli zachodzi konieczność dodatkowych uwarunkowań, to należy te dodatkowe testy zawrzeć bezpośrednio w akcjach operatorskich. Można zastosować akcję operatorską *Perform*, np. *Perform(Variable(WSERVICE)==0, SetVariable(WCONTROL,0), Nothing())*

Konfiguracja roli stanowisk

W sekcji *Stanowiska* należy określić rolę, jaką pełnią poszczególne stanowiska w obsłudze definiowanego harmonogramu. Możliwe są poniższe ustawienia trybu uruchamiania:

a) Brak - stanowisko nigdy nie zajmuje się wykonywaniem harmonogramu. Może jednak służyć do definiowania cykli harmonogramów.
b) Lokalnie - stanowisko zawsze zajmuje się wykonywaniem harmonogramu. Ten tryb powinien być wybrany dla tylko jednego stanowiska i nie powinno być wtedy stanowisk pracujących *Centralnie*.
c) Centralnie - stanowisko może wykonywać harmonogram. Nastąpi automatyczny wybór aktywnego stanowiska z puli wszystkich stanowisk z ustawionym trybem *Centralnie*.

Parametr *Priorytet centralny* służy do określenia priorytetów wyboru stanowiska wykonującego harmonogram, w sytuacji, gdy kilka stanowisk może objąć taką rolę.

2.3 Predefiniowane kategorie dni

W każdej aplikacji istnieje kilka predefiniowanych kategorii dni. Użytkownik nie ma możliwości zmiany definicji tych kategorii.

Część kategorii predefiniowanych, np. *Dni robocze* jest tworzona programowo i nie podlega żadnym modyfikacjom. Inne zapisane są w katalogu definicyjnym aplikacji w pliku *TimetableCategorySets.csv*. Plik ten jest umieszczany w katalogu definicyjnym aplikacji w momencie jej tworzenia - wersja wzorcowa pliku pochodzi z podkatalogu *Prototypes* katalogu instalacyjnego.

Plik *TimetableCategorySets.csv* jest plikiem tekstowym w formacie *csv*. Projektant aplikacji może zmodyfikować jego zawartość, a w szczególności dodać (w osobnej kolumnie) własną kategorię predefiniowaną.

2.4 Uruchomienie okna edycji cykli harmonogramów

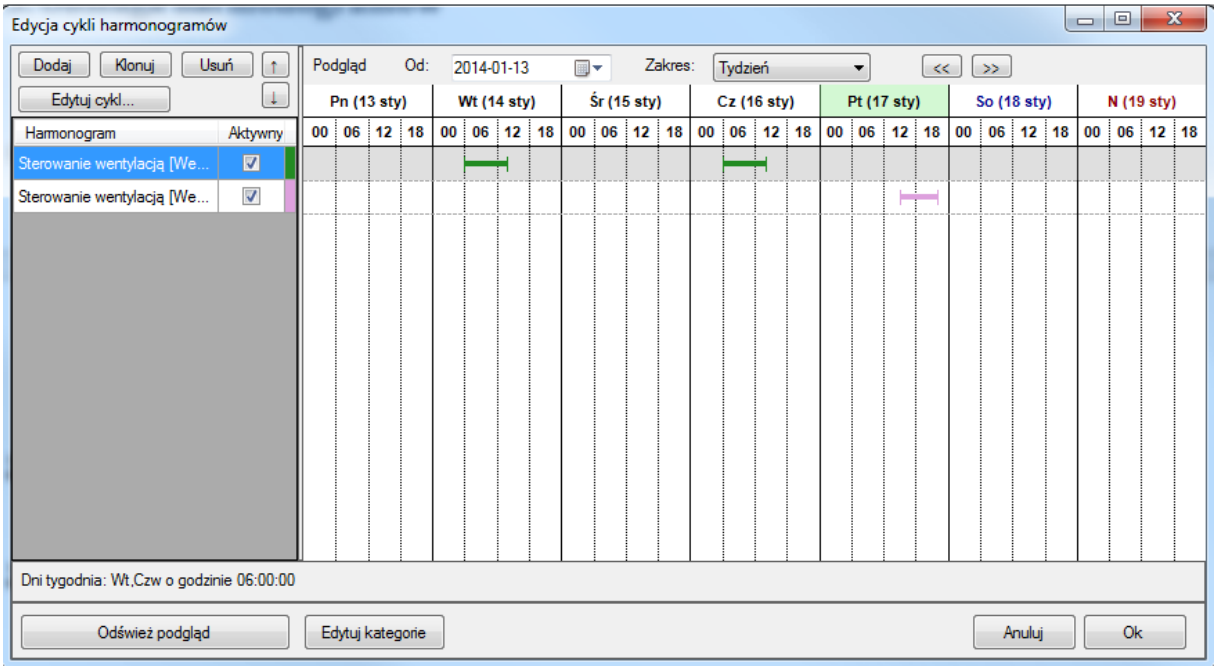
W trybie wykonywania aplikacji możliwa jest edycja kategorii dni oraz cykli harmonogramów przez użytkowników aplikacji. Edycja jest możliwa na każdym stanowisku, wymagane jest jednak, aby zalogowany użytkownik posiadał uprawnienie *Prawo edycji cykli harmonogramów*.

Do otwierania okna edycji kategorii i cykli służy akcja operatorska *EditTimetables*. W parametrach akcji można określić identyfikator harmonogramu, którego cykle będą definiowane. Możliwa jest też wspólna edycja cykli dla grupy harmonogramów (poprzez użycie w treści identyfikatora znaków specjalnych * i ?).

3 Obsługa harmonogramów

3.1 Edycja cykli harmonogramów

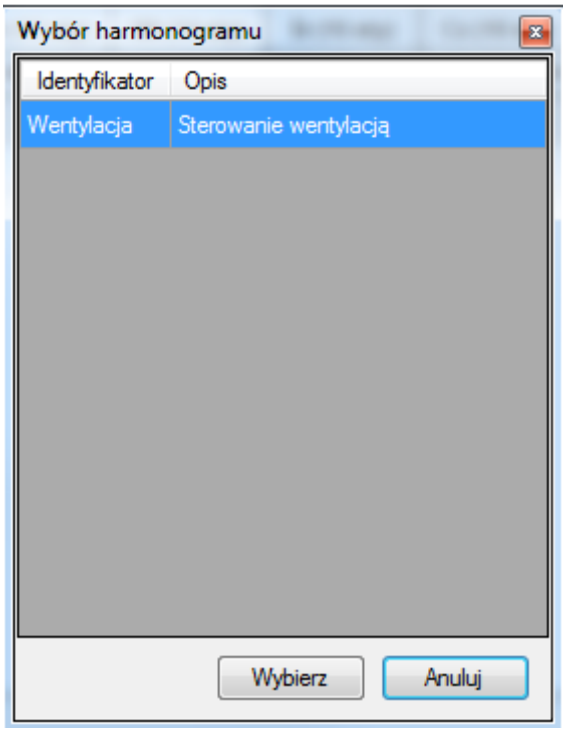
Okno edycji cykli harmonogramów służy do określenia cykli czasowych dla wykonywania zadań określonych w definicji harmonogramu. Dla każdego harmonogramu można deklarować dowolną liczbę cykli czasowych. Efektywne działanie harmonogramu jest sumą wszystkich cykli przyporządkowanych do harmonogramu.



Rys. Okno edycji cykli harmonogramów.

Lewa strona okna służy do definiowania cykli. Po prawej stronie znajduje się sekcja służąca do wizualnej prezentacji momentów wykonywania akcji harmonogramu.

Aby dodać nowy cykl, należy użyć przycisk *Dodaj* (lub *Klonuj*, jeżeli nowy cykl ma być kopią innego cyklu). W wyniku operacji *Dodaj* otworzy się lista zdefiniowanych harmonogramów, z której należy wybrać harmonogram, do którego zostanie dodany cykl.



Rys. Okno listy zdefiniowanych harmonogramów.

Po wybraniu harmonogramu i użyciu przycisku *Wybierz* zostanie utworzony nowy cykl i otwarte okno parametrów cyklu.
Okno parametrów cyklu może być też otwarte poprzez podwójne kliknięcie na linii cyklu lub poprzez przycisk *Edytuj cykl*.

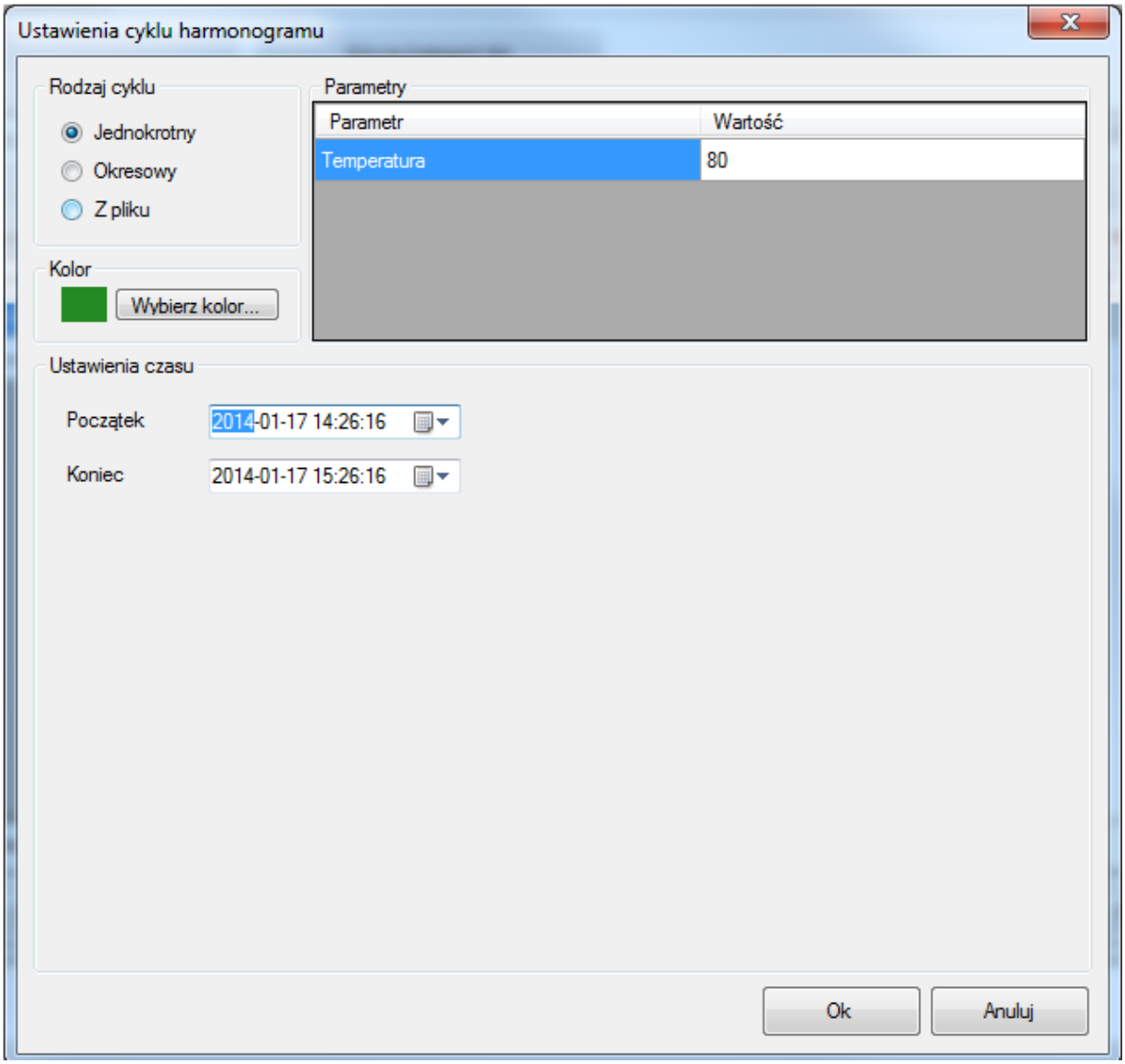
Żeby usunąć zaznaczony cykl, należy użyć przycisku *Usuń*. Można też chwilowo wyłączyć wykonywanie cyklu bez jego usunięcia, poprzez odznaczenie opcji w kolumnie *Aktywny*.

Przyciski strzałek służą wyłącznie do porządkownia kolejności cykli. Kolejność wyświetlania nie ma żadnego znaczenia dla sposobu wykonywania harmonogramu.

Poniżej listy cykli znajduje się linia informacyjna, która w skrótowy sposób informuje o sposobie działania wybranego cyklu.

Okno parametrów cyklu

Okno parametrów cyklu składa się z dwóch części. Górna część wygląda zawsze identycznie, natomiast postać części dolnej zależy od wybranego rodzaju cyklu.



Rys. Okno parametrów cyklu harmonogramu - cykl jednokrotny.

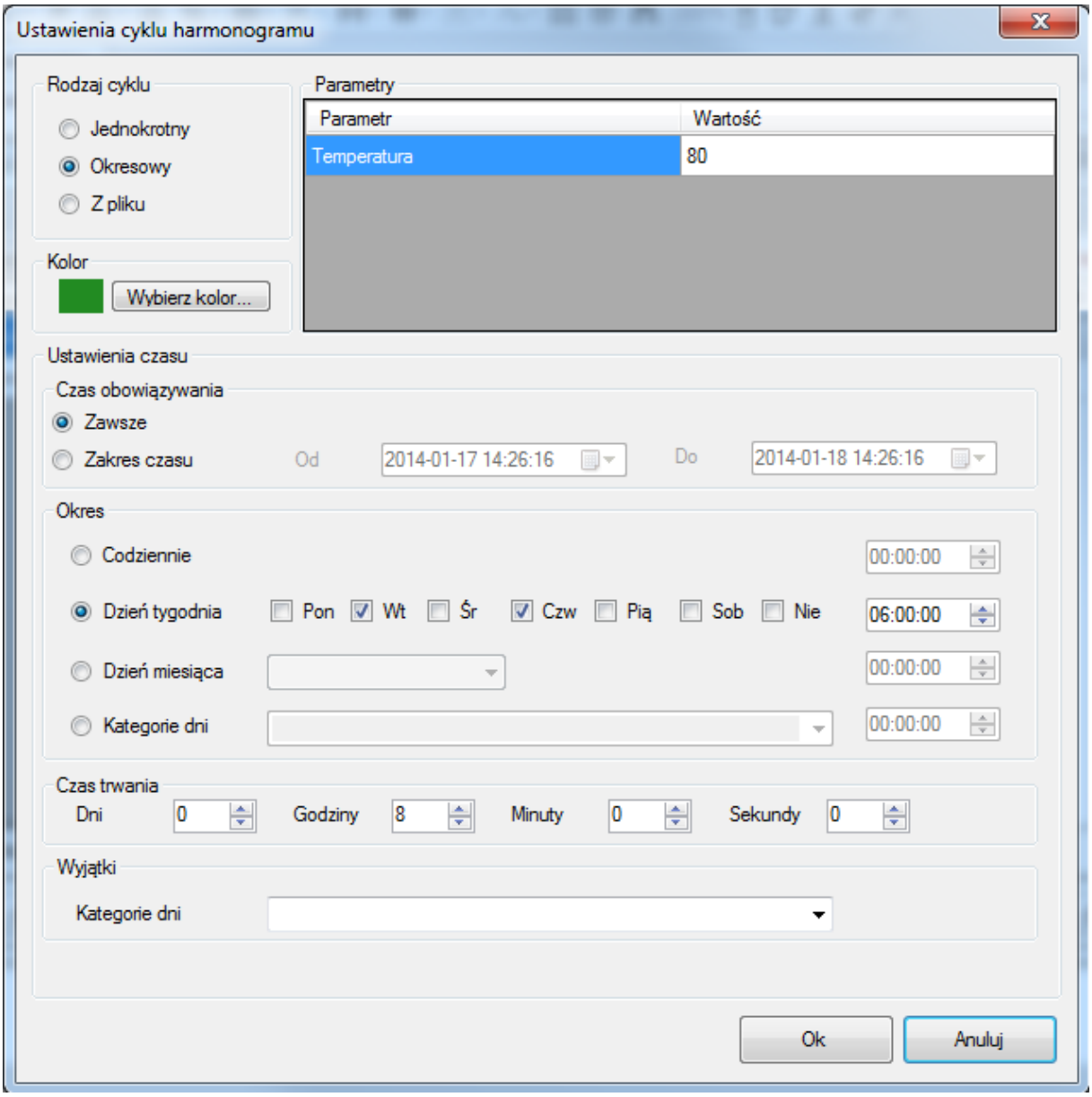
Górna część okna składa się z:

a) Sekcji służącej do wyboru rodzaju cyklu.
b) Tabeli parametrów - w tabeli umieszczone są nazwy wszystkich dostępnych parametrów harmonogramu. W kolumnie *Wartość* należy podać wartości parametrów obowiązujące dla definiowanego cyklu.
c) Przycisku *Wybierz kolor* pozwalającego zmienić kolor, używany do wyświetlania momentów działania cyklu na wykresie podglądowym.

W przypadku cyklu jednokrotnego jedyne parametry to:

a) Czas początku - moment jednorazowego wykonania akcji harmonogramu.
b) Czas końca - moment jednorazowego wykonania akcji kończącej harmonogramu (o ile harmonogram ma taką akcję zdefiniowaną).

Tryb okresowy to podstawowy rodzaj pracy harmonogramów. Pozwala on wykonywać akcje harmonogramów w cyklicznych momentach czasu.



Rys. Okno parametrów cyklu harmonogramu - cykl okresowy.

Cykl okresowy jest definiowany przez:

a) Czas obowiązywania
Sekcja pozwala ograniczyć okres wykonywania cyklu do wybranego zakresu czasu.

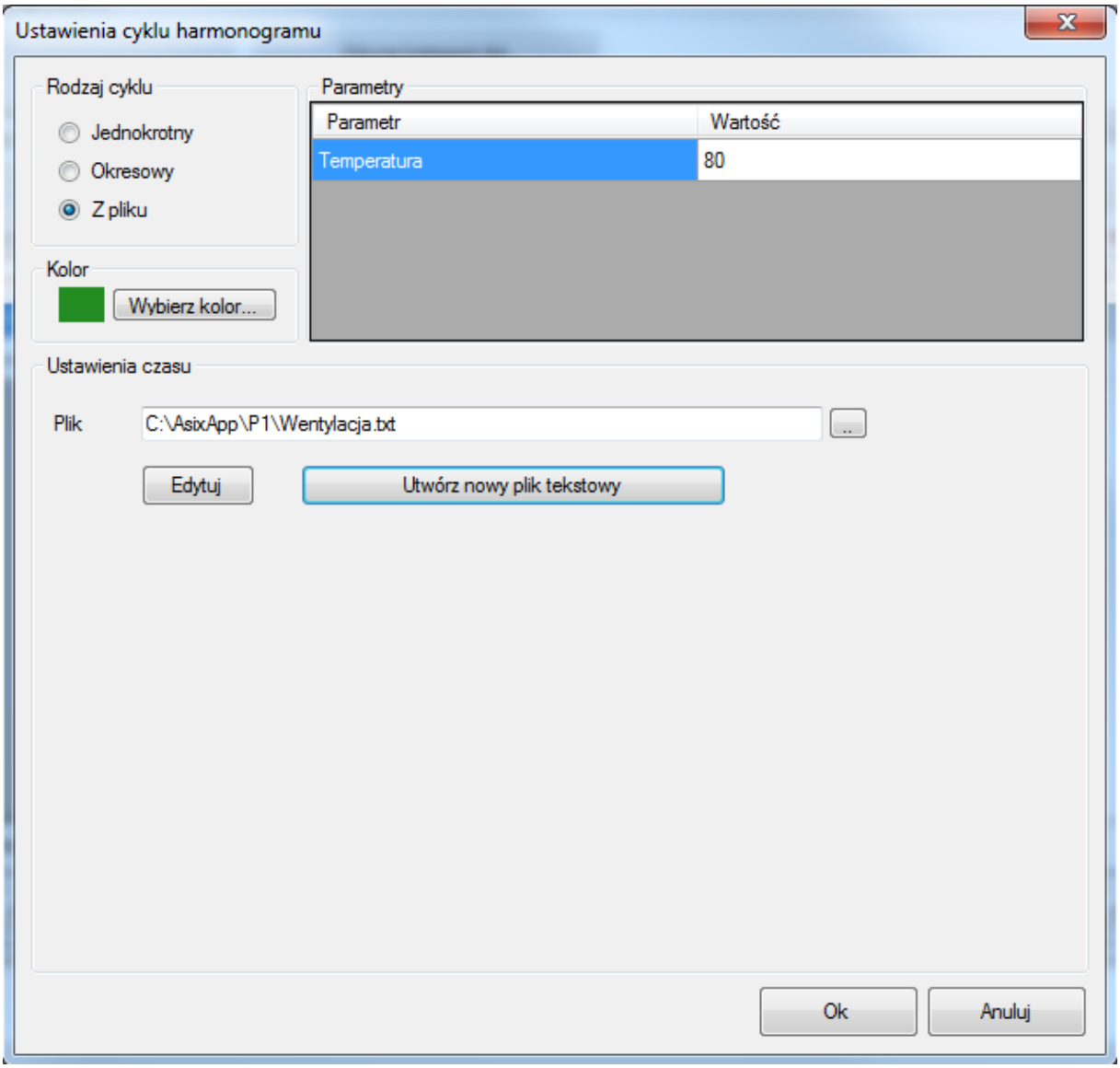
b) Okres
Sekcja określa momenty wykonania harmonogramu dla definiowanego cyklu (a dokładnie momenty wykonania akcji początku zdefiniowanej dla harmonogramu).
Okres składa się zawsze z określenia dni wykonywania oraz dokładnej godziny. Jeżeli wymagane jest wykonanie harmonogramu więcej niż raz dziennie, to należy zdefiniować dodatkowe cykle.

Wybór dnia może być wykonany przez: wybór dni tygodnia, wybór konkretnego dnia miesiąca lub wybór kategorii dni.

c) Czas trwania
Czas trwania jest używany tylko dla harmonogramów, które mają zdefiniowaną akcję końca. Definiuje on po jakim okresie czasu zostanie wykonana akcja końca. Czas ten jest odliczany względem momentów wykonania cyklu.

d) Wyjątki
Sekcja pozwala na określenie dni, dla których cykl ma nie być wykonywany. Należy podać identyfikatory kategorii dni. Mechanizm pozwala deklarować na przykład cykl typu: codziennie o wybranej godzinie, ale za wyjątkiem świąt.

Tryb pracy Z pliku pozwala na pracę harmonogramu w oparciu o dostarczany z zewnątrz plik określający momenty wykonania akcji harmonogramu. W szczególności plik może być tworzony przez inny system, generujący zlecenia dla aplikacji Asix.



Rys. Okno parametrów cyklu harmonogramu - cykl z pliku.

W polu *Plik* należy podać ścieżkę i nazwę pliku z momentami czasu wykonania harmonogramu. W przypadku pracy wielostanowiskowej plik powinien być umieszczony w lokalizacji dostępnej dla wszystkich stanowisk, które mogą przejąć kontrolę wykonywania harmonogramu (z reguły na jakimś zasobie sieciowym).

Przycisk *Edytuj* pozwala przejść do ręcznej edycji zawartości pliku. Przycisk *Utwórz nowy plik tekstowy* służy do tworzenia wzorcowego pliku cykli dla używanego harmonogramu.

Plik cykli jest plikiem tekstowym w formacie CSV. Pierwsza linia to nagłówki kolumn, a w następnych umieszczane są definicje cykli.

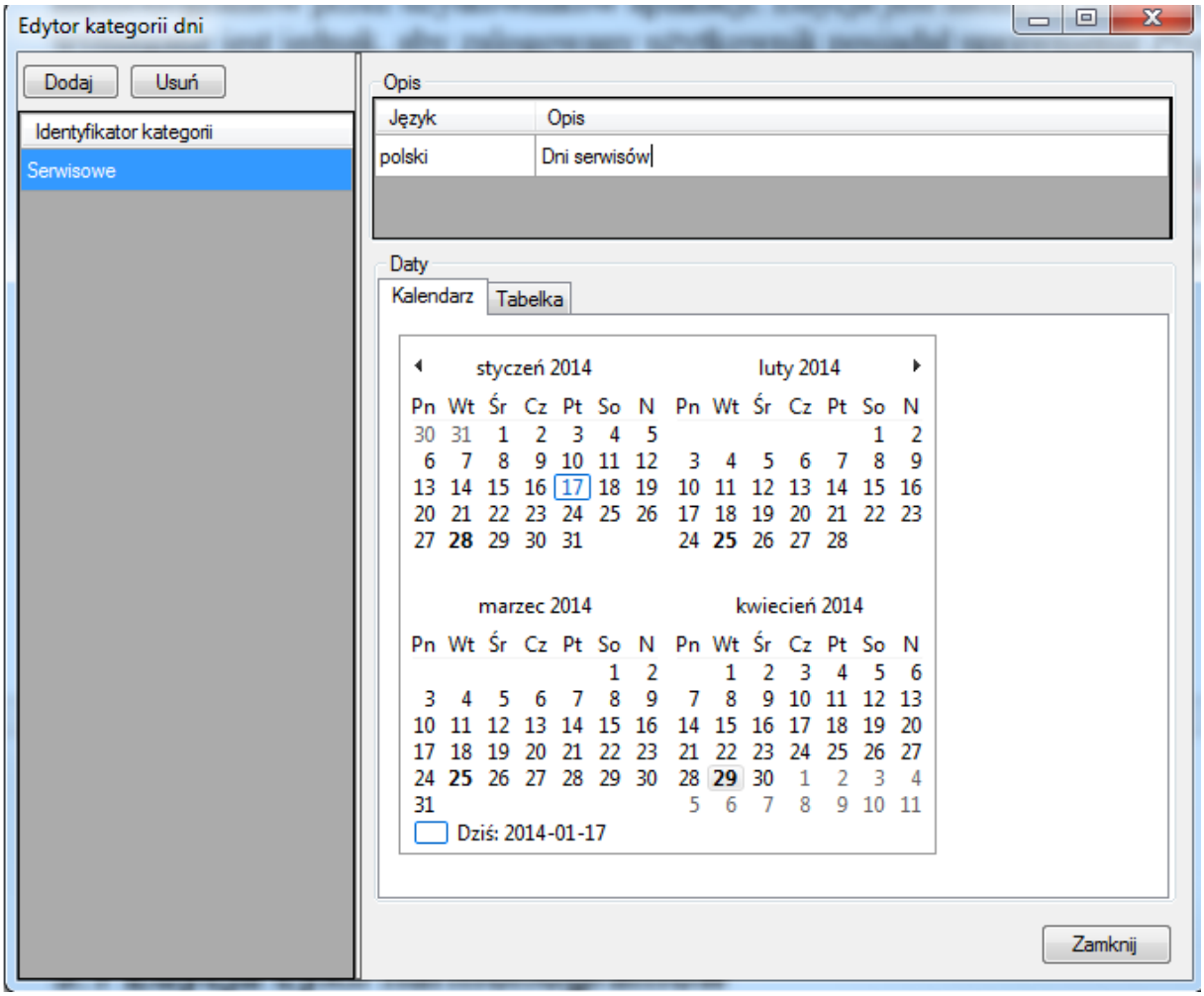
Poniżej pokazana jest zawartość przykładowego pliku.

StartTime, EndTime, Temperatura
2013.01.01 12:00:00, 2013.01.01 14:00:00, 20
2013.01.15 14:30:00, 2013.01.01 16:30:00, 25

System wykonywania harmonogramów wykrywa zmiany zachodzące w plikach cykli. Po wgraniu nowej wersji pliku nie są wymagane żadne czynności administracyjne - nastąpi automatyczne załadowanie nowego pliku.

3.2 Edycja kategorii dni

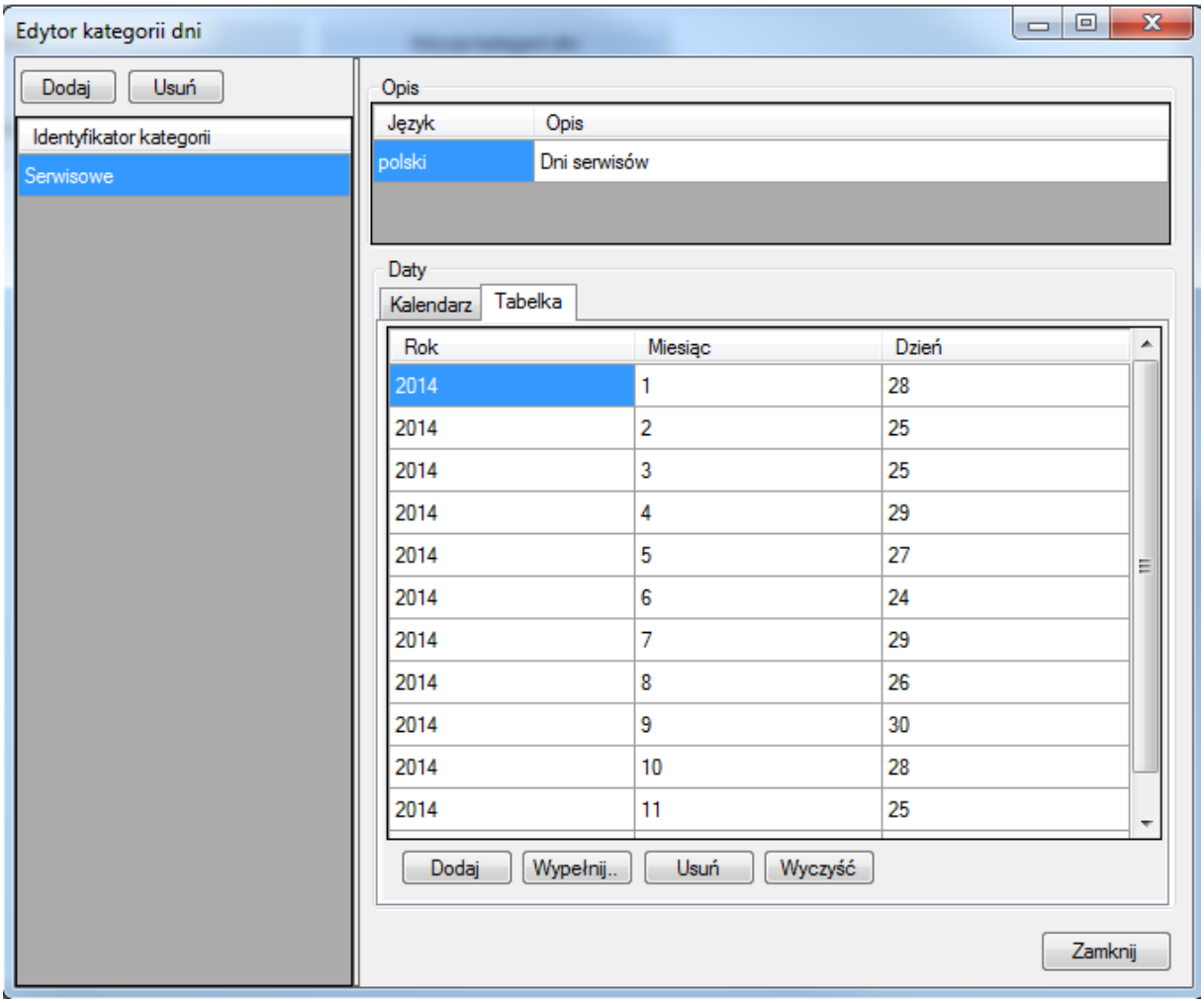
Okno edycji kategorii dni służy do definiowania nowych kategorii dni i zmiany zestawu dni wchodzących w skład kategorii dni. Okno może być otwarte poprzez użycie przycisku *Edytuj kategorie*, umieszczonego w oknie edycji cykli harmonogramu. Okno nie pozwala na zmiany kategorii predefiniowanych.



Rys. Okno edycji kategorii dni.

Lewa część okna służy do zarządzania kategoriami: ich tworzenia i usuwania. Prawa część służy do określenia zestawu dni wchodzących w skład wybranej kategorii.

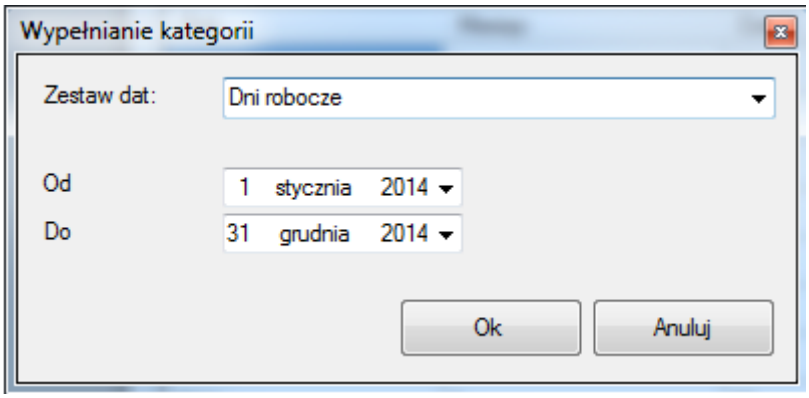
Możliwe są dwa tryby wyboru dni. W trybie *Kalendarz* należy kliknąć na numerze dnia, który ma być dołączony (lub usunięty) do kategorii. Dni wchodzące w skład kategorii wyświetlone są wytłuszczoną czcionką.



Rys. Okno edycji kategorii dni - kalendarz.

W trybie *Tabelka* wyświetlana jest lista dni wchodzących w skład kategorii. Przyciski poniżej tabelki służą do dodawania i usuwania dni. Możliwa jest także zmiana wybranej daty - w tym celu należy podwójnie kliknąć w polu, które ma być zmienione.

Przycisk *Wypełnij* pozwala jednorazowo dodać do kategorii grupę dni.



Rys. Okno dodawania grupy dni do kategorii.

W polu *Zestaw dat* należy wybrać identyfikator jednej z wcześniej utworzonych kategorii. W polach *Od* i *Do* należy podać daty graniczne. Po użyciu przycisku *Ok* do kategorii zostaną dodane wszystkie dni spełniające podane kryteria.

Asix.Evo - Raporty skryptowe

1 System raportów skryptowych

System raportów skryptowych stanowi alternatywny, w stosunku do systemu opartego o Microsoft Reporting Services, system generacji i przeglądania raportów w aplikacjach Asix.Evo. Odpowiada on w pewnym stopniu systemowi Raportera aplikacji klasycznych.

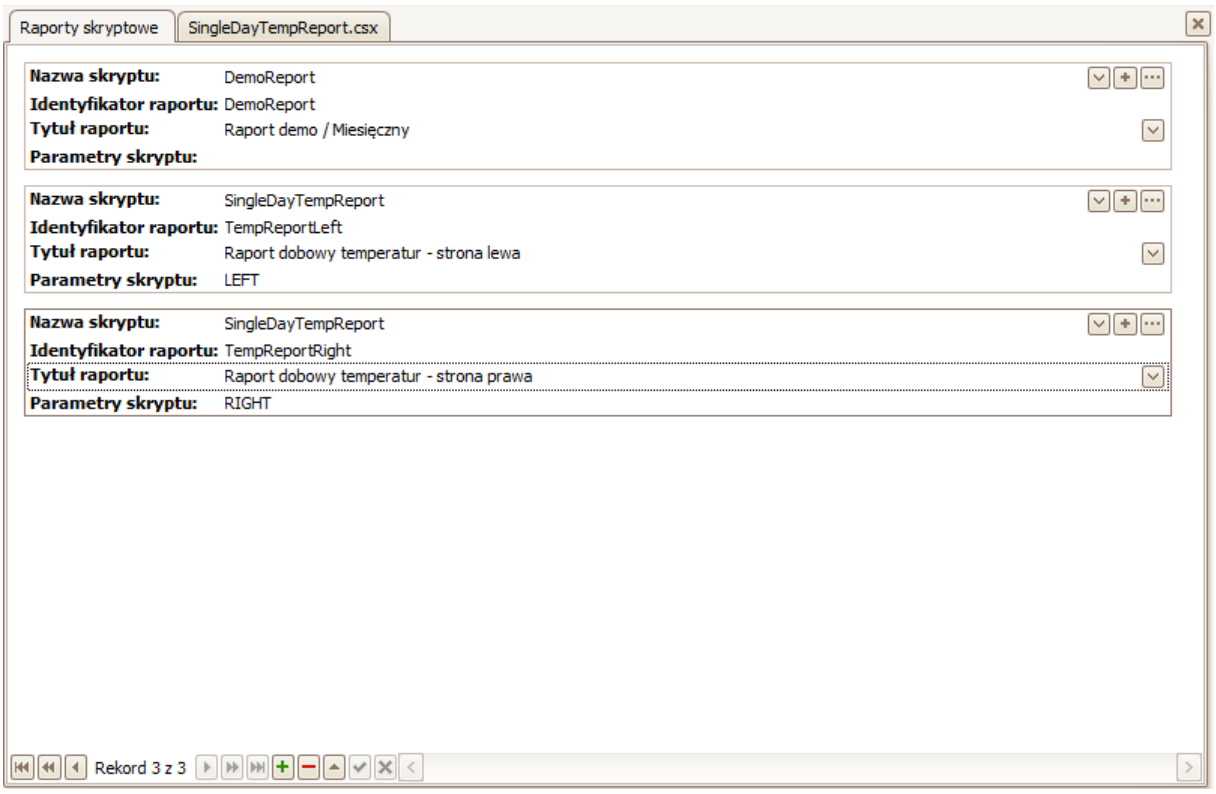
Zasadniczo system przeznaczony jest dla aplikacji prostszych, o mniejszych wymaganiach względem postaci i sposobów dystrybucji raportów. Sposób konfiguracji systemu jest łatwiejszy. Raporty skryptowe nie wymagają instalacji żadnego dodatkowego specjalistycznego oprogramowania, poza programem przeglądarki dokumentów PDF.

Raporty w systemie skryptowym tworzone są w sposób programowy. W związku z tym wymagane są od projektanta aplikacji pewne umiejętności programistyczne. Aby jednak to zadanie było możliwie proste, model obiektowy systemu skryptów został wzbogacony o grupę klas obiektów dedykowanych do tworzenia raportów.

W systemie raportów skryptowych raporty wynikowe tworzone są w formacie PDF. Mogą być one przeglądane i drukowane przez dowolne standardowe narzędzia do obsługi plików PDF. Do wyświetlania raportów używany jest program, który został w systemie operacyjnym zadeklarowany jako program do otwierania plików z rozszerzeniem PDF (zwykle Adobe Reader).

2 Konfiguracja raportów skryptowych

Każdy raport skryptowy musi zostać wstępnie zarejestrowany w aplikacji. Służy do tego panel ***Raporty skryptowe***, który otwierany jest poprzez węzeł ***Raporty skryptowe*** panelu ***Eksplorator aplikacji***.



Rys. Panel 'Raporty skryptowe'.

Definicja raportu skryptowego składa się z:

Nazwa skryptu - nazwa skryptu, który będzie używany do wygenerowania treści raportu.

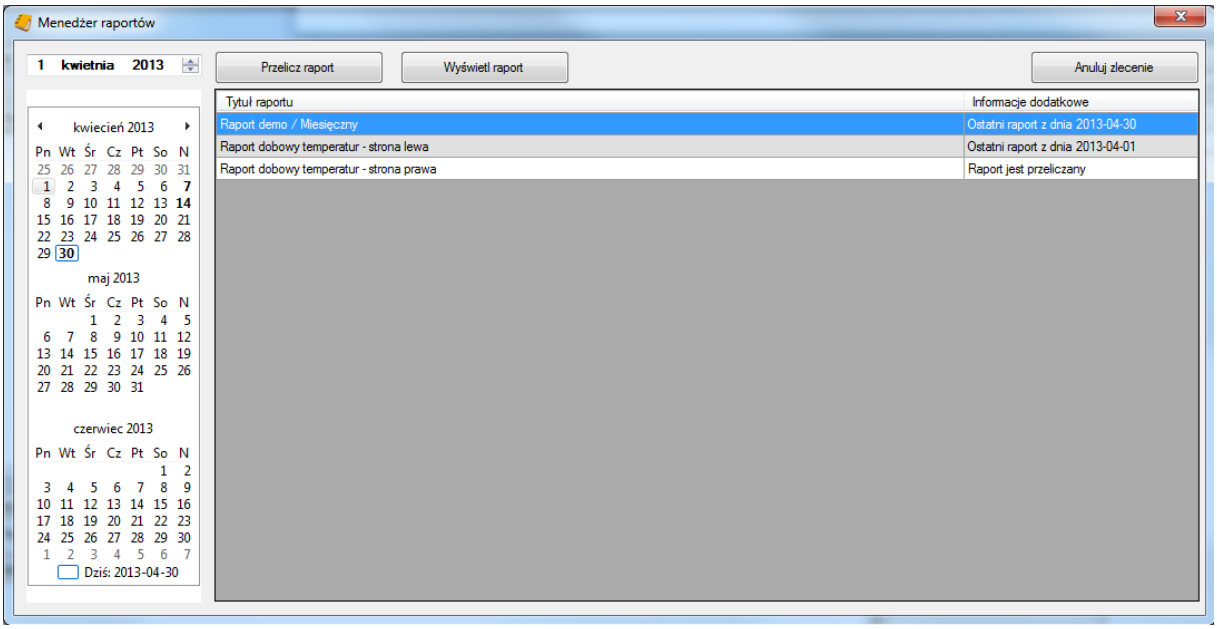
Identyfikator raportu - unikalny tekstowy identyfikator raportu. Używany jest w akcjach operatorskich do identyfikacji raportu oraz jako część nazwy plików wygenerowanych raportów.

Tytuł raportu - wielojęzyczny tekst opisujący przeznaczenie raportu.

Parametry skryptu - opcjonalne parametry przekazywane do skryptu w momencie generowania treści raportu. Właściwość typowo stosowana w przypadku, gdy ten sam skrypt stosowany jest do generowania różnych raportów. Poprzez przekazywane mu różne parametry (np. różne nazwy zmiennych) skrypt może dostosowywać zawartość raportu.

3 Obsługa raportów skryptowych

Proces generacji i wyświetlania może być kontrolowany przez użytkownika za pośrednictwem okna ***Menedżer raportów***. Okno otwierane jest komendą ***Menedżer raportów skryptowych*** z menu *Widok*, a w trybie wykonania aplikacji poprzez akcję ***ReportsManager***.



Rys. Okno 'Menedżer raportów'.

Okna składa się z dwóch części. Po lewej znajduje się kalendarz służący do wyboru daty generacji/wyświetlenia raportu wybranego w prawej części okna. Podświetlone daty oznaczają dni, dla których raport został już wcześniej wyliczony. Poprzez podwójne kliknięcie na takiej dacie, można wyświetlić utworzony raport.

W prawej części okna znajduje się lista zdefiniowanych w aplikacji raportów oraz kilka przycisków służących do zarządzania raportami. Przycisk ***Przelicz raport*** inicjuje proces wyliczenia. Można inicjować przeliczenie kolejnego raportu bez oczekiwania na zakończenie poprzedniej operacji (również tego samego raportu dla innej daty). Na liście raportów wyświetlana jest informacja - w jakim stanie jest aktualnie raport. Przycisk ***Wyświetl raport*** pozwala na wyświetlenie raportu dla daty wybranej w kalendarzu. Kliknięcie prawym klawiszem myszy na przycisku powoduje wyświetlenie listy dat, dla których raport został już wyliczony. Przycisk ***Anuluj zlecenie*** pozwala na usunięcie oczekujących zleceń generacji.

Alternatywą do użycia okna menedżera raportów jest użycie w aplikacji odpowiednich wywołań akcji operatorskiej ***ReportGenerate***.

4 Konstrukcja skryptów raportowych

Raporty w systemie skryptowym tworzone są w sposób programowy. Model obiektowy systemu skryptów został wzbogacony o grupę klas dedykowanych do tworzenia raportów. Klasy te są w rzeczywistości nadbudową nad bibliotekami PDFsharp i MigraDoc, które pozwalają na tworzenie dokumentów w formacie PDF. Klasy raportowe Asix.Evo mają za zadanie ułatwienie wykonania typowych operacji związanych z generacją raportów w postaci tabelarycznej lub wykresów. Podstawowe klasy związane z generacją raportów to: ***Report***, ***ReportTable*** i ***ReportChart***. Oryginalne klasy bibliotek PDFsharp i MigraDoc są również dostępne - można ich używać w razie konieczności zastosowania funkcji nie obsługiwanych przez klasy raportowe Asix.Evo.

4.1 Rodzaje raportów

Klasy raportowe Asix.Evo zorientowane są na tworzenie raportów w postaci tabelarycznej lub w postaci wykresów. Istnieje także możliwość łączenia w ramach pojedynczego raportu dowolnej liczby tabel i wykresów. Rozróżniane są dwa typy tabel, różniące się od siebie sposobem uporządkowania danych. W tabelach poziomych dane dla poszczególnych źródeł (pomiarów) są umieszczane w pojedynczym wierszu. W tabelach pionowych dane te są umieszczane w pojedynczej kolumnie. Tabele poziome stosuje się wtedy, gdy źródeł danych jest dużo, ale ilość danych jest niewielka. Tabele pionowe stosowane są, gdy źródeł danych jest niewiele, ale za to ilość raportowanych danych dla każdego źródła jest duża.

Typowym źródłem danych do raportu są dane archiwalne zmiennych procesowych. W tym przypadku autor raportu musi określić nazwy zmiennych wraz z nazwami pobieranych agregatów, oraz zakres czasowy wykonywanego zapytania. W przypadku takich raportów odpytanie o dane archiwalne wykonywane jest automatycznie w trakcie generacji zawartości tabeli (wykresu).

Alternatywnym źródłem danych jest dostarczenie do tabeli (wykresu) przygotowanej wcześniej kolekcji danych. W tym przypadku dane mogą pochodzić z dowolnego źródła. Dane mogą być dostarczone w obiekcie klasy ***DataTable***. Jest to wygodny format w przypadku danych pochodzących z bazy danych. Dane mogą być też dostarczone w dowolnej kolekcji spełniającej interfejs ***System.Collections.ICollection*** (w szczególności w zwykłej tablicy obiektów dowolnej klasy).

4.2 Szkielet kodu generacji raportu

Jeżeli skrypt raportowy został utworzony w oknie konfiguracji raportów skryptowych (lub została wybrana opcja skryptu raportowego w standardowym oknie tworzenia skryptu), to został dla niego zbudowany typowy szkielet kodu generującego raport. Kod ten pokazuje typowy sposób generacji raportu tabelarycznego dla danych archiwalnych zmiennych procesowych. Praktycznie każdy skrypt raportowy opiera się na poniższym schemacie działania.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class ReportTemplate : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         DateTime reportDate = DateTime.Parse(aParameters[0]).Date;
         string outputFile = aParameters[1];


         // Create report
         Report report = aApplication.CreateReport();
         report.SetPageSetup(false, PageFormat.A4);


         // Define standard and custom styles
         report.SetStyle(ReportStyleStandardNames.Normal, "Arial", 10, Colors.Black, false,
                         false, ParagraphAlignment.Left, Color.Empty);
         report.SetStyle(ReportStyleStandardNames.Header, "Arial", 10, Colors.SeaGreen, false,
                         false, ParagraphAlignment.Left, Color.Empty);
         report.SetStyle(ReportStyleStandardNames.Footer, "Arial", 10, Colors.SeaGreen, false,
                         false, ParagraphAlignment.Left, Color.Empty).
                         AddTabStop(150, TabAlignment.Right);
         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
                         true, true, ParagraphAlignment.Center, Colors.LightYellow);
         report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue, false,
                         false, ParagraphAlignment.Left, Color.Empty);
         report.SetStyle("Title", "Arial", 14, Colors.DarkBlue, false, false,
                         ParagraphAlignment.Center, Color.Empty);

         // Define header
         report.Header.AddParagraph("Header text");


         // Define footer
         report.Footer.AddParagraph("Report generated ").
            AddText(DateTime.Now.ToString()).AddText("\tPage ").AddPageField().
                                                    AddText("/").AddNumPagesField();


         // Add title line
         report.AddParagraph("Example report for date " + reportDate.ToString(), "Title");
         report.AddParagraph("");


         // Vertical table example
         ReportTable vt = report.AddVerticalTable(true);
         vt.LeftIndent(20);
```

```
        vt.AddColumn("Start Time", ReportTable.StartTimeColumn, null, 40, null,
                     ReportTableCellErrorBehavior.Normal, null);
        vt.AddColumn("End Time", ReportTable.EndTimeColumn, null, 40, null,
                     ReportTableCellErrorBehavior.Normal, null);
        vt.AddColumn("Var1 Value", "Var1", "average", 25, "f2",
                     ReportTableCellErrorBehavior.Empty, null);
        vt.AddColumn("Var2 Value", "Var2", "average", 25, "f2",
                     ReportTableCellErrorBehavior.Empty, null);
        vt.SetQueryData(reportDate, reportDate.AddDays(1), "1H"));
        vt.SetEdgeStyle(ReportTableEdgeType.Outline, Colors.Cyan);


        // Horizontal table example
        //ReportTable ht = report.AddHorizontalTable(true, 30, 25, "Signals", "HH:mm");
        //ht.AddRow("Var1 Value", "Var1", "average", "f2",
        //          ReportTableCellErrorBehavior.Empty, null);
        //ht.AddRow("Var2 Value", "Var2", "average", "f2",
        //          ReportTableCellErrorBehavior.Empty, null);
        //ht.AddRow("Var3 Value", "Var3", "average", "f2",
        //          ReportTableCellErrorBehavior.Empty, null);
        //ht.SetQueryData(reportDate, reportDate.AddDays(1), "1H");
        //ht.SetEdgeStyle(ReportTableEdgeType.Outline, Colors.Cyan);

        report.Render(outputFile);
     }
     catch (Exception e)
     {
        aApplication.ReportMessage(EvoMessageType.Error, "Error generating report: " +
             e.ToString());
     }

     return true;
  }

  public void FinalizeScript()
  {
  }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class ReportTemplateVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize
        Try


            Dim reportDate As DateTime = DateTime.Parse(aParameters(0)).Date
            Dim outputFile As String = aParameters(1)

            ' Create report
            Dim report As Report = aApplication.CreateReport()
            report.SetPageSetup(False, PageFormat.A4)

            ' Define standard and custom styles
            report.SetStyle(ReportStyleStandardNames.Normal, "Arial", 10, Colors.Black, _
                            False, False, ParagraphAlignment.Left, Color.Empty)
            report.SetStyle(ReportStyleStandardNames.Header, "Arial", 10, Colors.SeaGreen, _
                            False, False, ParagraphAlignment.Left, Color.Empty)
```

```
            report.SetStyle(ReportStyleStandardNames.Footer, "Arial", 10, Colors.SeaGreen, _
                            False, False, ParagraphAlignment.Left, Color.Empty). _
                            AddTabStop(150, TabAlignment.Right)
            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,_

                    True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue, _
                            False, False, ParagraphAlignment.Left, Color.Empty)
            report.SetStyle("Title", "Arial", 14, Colors.DarkBlue, _
                            False, False, ParagraphAlignment.Center, Color.Empty)


            ' Define header
            report.Header.AddParagraph("Header text")


            ' Define footer
            report.Footer.AddParagraph("Report generated "). _
             AddText(DateTime.Now.ToString()).AddText("\tPage ").AddPageField().AddText("/")._
             AddNumPagesField()


            ' Add title line
            report.AddParagraph("Example report for date " + reportDate.ToString(), "Title")
            report.AddParagraph("")


            ' Vertical table example
            Dim vt As ReportTable = report.AddVerticalTable(True)
            vt.LeftIndent(20)
            vt.AddColumn("Start Time", ReportTable.StartTimeColumn, Nothing, 40, Nothing,_
                         ReportTableCellErrorBehavior.Normal, Nothing)
            vt.AddColumn("End Time", ReportTable.EndTimeColumn, Nothing, 40, Nothing,_
                         ReportTableCellErrorBehavior.Normal, Nothing)
            vt.AddColumn("Var1 Value", "Var1", "average", 25, "f2",_
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.AddColumn("Var2 Value", "Var2", "average", 25, "f2",_
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.SetQueryData(reportDate, reportDate.AddDays(1), "1H"))
            vt.SetEdgeStyle(ReportTableEdgeType.Outline, Colors.Cyan)


            ' Horizontal table example
            'Dim ht As ReportTable = report.AddHorizontalTable(true, 30, 25, "Signals", _
            '                                                  "HH:mm")
            'ht.AddRow("Var1 Value", "Var1", "average", "f2", _
            '          ReportTableCellErrorBehavior.Empty, Nothing)
            'ht.AddRow("Var2 Value", "Var2", "average", "f2", _
            '          ReportTableCellErrorBehavior.Empty, Nothing)
            'ht.AddRow("Var3 Value", "Var3", "average", "f2", _
            '          ReportTableCellErrorBehavior.Empty, Nothing)
            'ht.SetQueryData(reportDate, reportDate.AddDays(1), "1H")
            'ht.SetEdgeStyle(ReportTableEdgeType.Outline, Colors.Cyan)

            report.Render(outputFile)

        Catch e As Exception


            aApplication.ReportMessage(EvoMessageType.Error,_
                                       "Error generating report: " + e.ToString())


        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

End Class
```

Całość kodu generującego raport ujęta została w bloku *try catch*. Jest to zalecana konstrukcja, ponieważ w przypadku wystąpienia błędów w trakcie generacji raportu, zostaną one zgłoszone za pośrednictwem wyjątków. Blok *try catch* pozwala na kontrolowanie poprawności wykonania skryptu.

Pierwszą operacją skryptu jest pobranie parametrów uruchomienia skryptu i umieszczeniu ich w pomocniczych zmiennych *reportDate* i *outputFile*. Do skryptu raportowego przekazywane są zawsze dwa parametry. Pierwszy określa datę, dla której raport ma być wygenerowany, a drugi nazwę pliku wynikowego. Jeżeli w konfiguracji raportu skryptowego były zdefiniowane dodatkowe parametry, to zostaną one przekazane do skrypty jako kolejne, po parametrze nazwy pliku wynikowego.

Kolejnym krokiem, obowiązkowym w każdym skrypcie raportowym, jest utworzenie obiektu klasy *Report* poprzez wywołanie metody *CreateReport* obiektu *aApplication*. Dalsze operacje związane z generowaniem treści raportu będą wykorzystywały utworzony obiekt.

Opcjonalne wywołanie metody *SetPageSetup* służy do określenia parametrów strony wydruku. W szczególności pierwszy parametr ustawiony na *true* pozwala przełączyć układ strony na poziomy. Istnieje też alternatywna wersja metody *SetPageSetup* pozwalająca na ustawienie wielkości marginesów.

Kolejna sekcja skryptu służy do zdefiniowania stylów. Style służą do określenia wyglądu raportu (głównie tekstów). Definiują one takie parametry jak kształt i wielkość czcionki, jej kolor oraz kolor tła. Każdy styl posiada nazwę. Istnieje pewien zestaw nazw standardowych (zdefiniowanych w klasie *ReportStyleStandardNames*). Style o takich nazwach są używane w sposób automatyczny, np. styl *ReportStyleStandardNames.Normal* jest domyślnym stylem dla wszystkich elementów, dla których styl nie został w żaden sposób określony. Możliwe jest definiowanie stylów o nazwach własnych, ale wtedy do ich użycia konieczna jest jawna deklaracja stylu w metodach tworzących elementy raportu.

Po zdefiniowaniu stylów, skrypt przechodzi do tworzenia treści nagłówka i stopki raportu. Nie są to elementy obowiązkowe, ale zazwyczaj są stosowane. Zawartość nagłówka i stopki jest wypełniana przy pomocy uniwersalnej metody *AddParagraph* służącej do wstawiania tekstu do treści raportu. W przypadku stopki tekst jest wieloelementowy, złożony z części tekstowych oraz pól specjalnych: numeru bieżącej strony i sumarycznej liczby stron. Wygląd nagłówka i stopki zależy od standardowych stylów *ReportStyleStandardNames.Header* i *ReportStyleStandardNames.Footer*.

Kolejnym krokiem jest wstawienie opcjonalnego tytułu raportu. W tym przypadku zastosowany jest wariant metody *AddParagraph* z jawnym określeniem nazwy stylu. W definicji użytego stylu zostało zadeklarowane wyrównywanie typu *ParagraphAlignment.Center* - powoduje to, że treść tytułu zostanie umieszczona na środku wiersza.

W sposób podobny do wstawiania tekstów, do raportu mogą być też wstawione elementy graficzne przy pomocy metody *AddImage*.

Kolejna sekcja skryptu, rozpoczęta użyciem metody *AddVerticalTable*, odpowiada za umieszczenie w raporcie tabeli wartości archiwalnych zmiennych procesowych. Po utworzeniu tabeli należy do niej jeszcze dodać zestaw kolumn (wywołanie metody *AddColumn*), określić parametry czasowe odpytania o dane (metoda *SetQueryData*), oraz opcjonalnie określić sposób rysowania krawędzi komórek tabeli (metoda *SetEdgeStyle*). Należy zwrócić uwagę, że w wywołaniu metody *SetQueryData*

wykorzystano zmienną *reportDate,* zawierającą datę wykonania raportu, przekazaną w parametrach uruchomienia skryptu.

Poniżej kodu generującego tabelę pionową znajduje się zakomentowany szkielet kodu dla tabeli poziomej. Podstawowa różnica to użycie metody *AddRow* zamiast *AddColumn*.

Praca skryptu obowiązkowo kończona jest wywołaniem metody *Render* z przekazaniem nazwy pliku wynikowego, podanego w parametrach uruchomienia skryptu. Dopiera ta metoda wykonuje generację wynikowego pliku PDF, na podstawie wcześniej wykonanych operacji i odczytanych danych archiwalnych.

4.3 Tabela pionowa dla danych archiwalnych

W tym rozdziale omówimy dokładniej proces budowania tabeli pionowej w oparciu o dane archiwalne zmiennych procesowych. Pokazany dalej skrypt tworzy tabelę, która pokazuje wartość średnią, maksymalną i minimalną zmiennej procesowej w rozbiciu na okresy trzygodzinne. Wygląd generowanej tabeli pokazuje ilustracja.

Godzina	V1 średnia	V1 minimum	V1 maksimum
2012-11-01 00:00:00	26,19	25,50	26,25
2012-11-01 03:00:00	24,87	24,00	25,50
2012-11-01 06:00:00	23,45	22,50	24,00
2012-11-01 09:00:00	21,84	21,00	22,50
2012-11-01 12:00:00	20,52	20,25	21,00
2012-11-01 15:00:00	20,04	19,50	20,25
2012-11-01 18:00:00	19,50	19,50	19,50
2012-11-01 21:00:00	19,11	18,75	19,50

Rys. Tabela pionowa dla danych archiwalnych.

Ze względu na chęć skrócenia kodu, poniższe skrypty przykładowe zostały pozbawione fragmentów dodających do raportu elementy typu: nagłówek, stopka lub linia tytułu.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class VertTable : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         DateTime reportDate = DateTime.Parse(aParameters[0]).Date;
         string outputFile = aParameters[1];

         Report report = aApplication.CreateReport();

         // Define standard and custom styles
         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
                         true, true, ParagraphAlignment.Center, Colors.LightYellow);
         report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.Gray);
         report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.DarkGray);

         // Build table
         ReportTable vt = report.AddVerticalTable(true);
         vt.LeftIndent(10);
         vt.AddColumn("Godzina", ReportTable.StartTimeColumn, null, 40, null,
                      ReportTableCellErrorBehavior.Normal, null);
         vt.AddColumn("V1 średnia", "sym01", "average", 30, "f2",
                      ReportTableCellErrorBehavior.Empty, null);
         vt.AddColumn("V1 minimum", "sym01", "min", 30, "f2",
                      ReportTableCellErrorBehavior.Empty, null);
         vt.AddColumn("V1 maksimum", "sym01", "max", 30, "f2",
                      ReportTableCellErrorBehavior.Empty, null);
         vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan);
         vt.SetQueryData(reportDate, reportDate.AddDays(1), "3H");

         report.Render(outputFile);
      }
      catch (Exception e)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
                                    "Error generating report: " + e.ToString());
      }

      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
```

```
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class VertTableVb
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize
        Try

            Dim reportDate As DateTime = DateTime.Parse(aParameters(0)).Date
            Dim outputFile As String = aParameters(1)

            Dim report As Report = aApplication.CreateReport()

            ' Define standard And custom styles
            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,_
                            True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,_
                            False, False, ParagraphAlignment.Right, Colors.Gray)
            report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,_
                            False, False, ParagraphAlignment.Right, Colors.DarkGray)

            ' Build table
            Dim vt As ReportTable = report.AddVerticalTable(True)
            vt.LeftIndent(10)
            vt.AddColumn("Godzina", ReportTable.StartTimeColumn, Nothing, 40, Nothing,_
                         ReportTableCellErrorBehavior.Normal, Nothing)
            vt.AddColumn("V1 średnia", "sym01", "average", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.AddColumn("V1 minimum", "sym01", "min", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.AddColumn("V1 maksimum", "sym01", "max", 30, "f2",_
                          ReportTableCellErrorBehavior.Empty, Nothing)
            vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan)
            vt.SetQueryData(reportDate, reportDate.AddDays(1), "3H")

            report.Render(outputFile)


        Catch e As Exception

            aApplication.ReportMessage(EvoMessageType.Error, _
                                       "Error generating report: " + e.ToString())

        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
     End Sub

End Class
```

W przykładowym kodzie wygląd tabeli określa kilka stylów o predefiniowanych nazwach. Są to przede wszystkim style *TableHeader* i *TableRow*, określające sposób tworzenia wiersza nagłówka tabeli oraz wierszy danych. Ponieważ w naszym przykładzie chcemy inaczej formatować parzyste i nieparzyste wiersze danych, to zdefiniowany został też styl *TableAltRow*. Style dla wierszy danych różnią się ostatnim parametrem, określającym kolor tła komórek tabeli.

Oprócz używanej czcionki oraz kolorów styl określa także sposób wyrównywania zawartości komórek. W linii nagłówka tekst będzie środkowany a w liniach danych wartości pomiarów będą wyrównywane do prawej strony.

Po zdefiniowaniu stylów tworzona jest tabela pionowa przy pomocy metody *AddVerticalTable*. Parametr wywołania metody określa, czy do tabeli zostanie dodany wiersz nagłówka.

Kolejnym krokiem jest dodanie do tabeli zestawów kolumn, wykorzystując metodę *AddColumn*. Parametry wywołania mają następujące znaczenie: nazwa kolumny umieszczana w linii nagłówka, nazwa zmiennej procesowej i agregatu - służące do pobrania danych archiwalnych, szerokość kolumny, sposób formatowania wartości (jeżeli nie zostanie podany, to użyty zostanie atrybut formatu z bazy definicji zmiennych lub nastąpi formatowanie standardowe zależne od typu wartości), sposób postępowania w przypadku braku lub niepoprawnej wartości, funkcja tworzenia zawartości komórki (jej użycie zostanie opisane w rozdziale *4.5 Kontrola zawartości komórek*).

W wywołaniach *AddColumn* zamiast nazwy zmiennych można użyć jednej z dwóch specjalnych nazw: *ReportTable.StartTimeColumn* lub *ReportTable.EndTimeColumn*. Nazwy te oznaczają użycie w kolumnie, zamiast wartości zmiennych, czasów początków lub końców poszczególnych okresów agregacji zmiennych.

W przykładzie użyto stałej *ReportTableCellErrorBehavior.Empty* do określenia trybu postępowania w wypadku braku poprawnej wartości. Stała ta oznacza pozostawienie pustej komórki. Inne standardowe tryby są wyjaśnione w opisie metody *AddColumn* w części referencyjnej podręcznika.

Przykładowy skrypt używa prostszego wariantu metody *AddColumn*. Dostępny jest także wariant pozwalający podać nazwy stylów używanych w kolumnie. Style podane jawnie mają pierwszeństwo przed stylami standardowymi. Rozszerzony wariant metody pozwala indywidualnie określić sposób tworzenia poszczególnych kolumn tabeli.

Po dodaniu kolumn pozostaje jeszcze określić sposób tworzenia krawędzi komórek tabeli (metoda *SetEdgeStyle*) oraz parametry zapytania o dane archiwalne (metoda *SetQueryData*).

Końcowe wywołanie metody *Render* powoduje wykonanie wszystkich niezbędnych zapytań o dane archiwalne oraz wygenerowanie treści raportu.

4.4 Tabela pozioma dla danych archiwalnych

W tym rozdziale omówimy dokładniej proces budowania tabeli w układzie poziomym w oparciu o dane archiwalne zmiennych procesowych. Pokazany dalej skrypt tworzy tabelę, która pokazuje wartość średnią, maksymalną i minimalną zmiennej procesowej w rozbiciu na okresy dwugodzinne. Wygląd generowanej tabeli pokazuje ilustracja.

Godzina	00:00	02:00	04:00	06:00	08:00	10:00	12:00	14:00	16:00	18:00	20:00	22:00
V1 średnia	26,25	25,77	24,58	23,89	22,62	21,62	20,66	20,25	19,94	19,50	19,50	18,92
V1 minimum	26,25	24,75	24,00	23,25	21,75	21,00	20,25	20,25	19,50	19,50	19,50	18,75
V1 maksimum	26,25	26,25	24,75	24,00	23,25	21,75	21,00	20,25	20,25	19,50	19,50	19,50

Rys. Tabela pozioma dla danych archiwalnych.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class HorzTable : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         DateTime reportDate = DateTime.Parse(aParameters[0]).Date;
         string outputFile = aParameters[1];

         Report report = aApplication.CreateReport();
         report.SetPageSetup(true, PageFormat.A4);

         // Define standard and custom styles
         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
                         true, true, ParagraphAlignment.Center, Colors.LightYellow);
         report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.Gray);
         report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.DarkGray);

         // Build table
         ReportTable ht = report.AddHorizontalTable(true, 25, 15, "Godzina", "HH:mm");
         ht.AddRow("V1 średnia", "sym01", "average", "f2",
                   ReportTableCellErrorBehavior.Empty, null);
         ht.AddRow("V1 minimum", "sym01", "min", "f2",
                   ReportTableCellErrorBehavior.Empty, null);
         ht.AddRow("V1 maksimum", "sym01", "max", "f2",
                   ReportTableCellErrorBehavior.Empty, null);
         ht.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan);
         ht.SetQueryData(reportDate.AddHours(0), reportDate.AddDays(1), "2H");


         report.Render(outputFile);
      }
      catch (Exception e)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
                                    "Error generating report: " + e.ToString());
      }

      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class HorzTableVb
    Implements IScript
```

```
    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize
        Try

            Dim reportDate As DateTime = DateTime.Parse(aParameters(0)).Date
            Dim outputFile As String = aParameters(1)

            Dim report As Report = aApplication.CreateReport()
            report.SetPageSetup(True, PageFormat.A4)

            ' Define standard And custom styles
            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,_
                            True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,_
                            False, False, ParagraphAlignment.Right, Colors.Gray)
            report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,_
                            False, False, ParagraphAlignment.Right, Colors.DarkGray)


            ' Build table
            Dim ht As ReportTable = report.AddHorizontalTable(True,25,15,"Godzina","HH:mm")
            ht.AddRow("V1 średnia", "sym01", "average", "f2",
                      ReportTableCellErrorBehavior.Empty, Nothing)
            ht.AddRow("V1 minimum", "sym01", "min", "f2",
                      ReportTableCellErrorBehavior.Empty, Nothing)
            ht.AddRow("V1 maksimum", "sym01", "max", "f2",
                      ReportTableCellErrorBehavior.Empty, Nothing)
            ht.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan)
            ht.SetQueryData(reportDate.AddHours(0), reportDate.AddDays(1),"2H")




            report.Render(outputFile)

        Catch e As Exception

            aApplication.ReportMessage(EvoMessageType.Error, "Error generating report: " + e.T
oString())



        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

End Class
```

W przypadku raportów poziomych z reguły konieczna jest zmiana orientacji strony raportu na poziomą. Dlatego bezpośrednio po utworzeniu obiektu *Report* użyta jest metoda *SetPageSetup*. Pierwszy parametr wywołania ustawiony na *true* oznacza poziome ułożenie strony.

Kolejny fragment kodu definiuje klika stylów o predefiniowanych nazwach. Zasady użycia tych stylów zostały opisane w rozdziale dotyczącym tabel pionowych.

Po zdefiniowaniu stylów tworzona jest tabela pionowa przy pomocy metody *AddHorizontalTable*. Parametry wywołania strony dotyczą głównie linii nagłówka tabeli. Ich znaczenie jest następujące:

informacja czy linia nagłówka ma być dodana do tabeli, szerokość pierwszej kolumny tabeli, szerokość następnych kolumn tabeli, tekst umieszczony w pierwszej kolumnie linii nagłówka, sposób formatowanie czasu w następnych kolumnach nagłówka.

Pierwsza kolumna tabeli poziomej jest kolumną opisową, w której umieszczany jest tekst opisujący znaczenie danych w kolumnie. W linii nagłówka zawsze wyświetlane są czasy początków okresów agregacji, wynikające z wykonanego zapytania o dane archiwalne.

Istnieje również rozszerzona wersja metody *AddHorizonatlTable* pozwalająca jawnie wybrać style dla pierwszej i następnych kolumn nagłówka.

Kolejnym krokiem jest dodanie do tabeli zestawu wierszy, wykorzystując metodę *AddRow*. Parametry wywołania mają następujące znaczenie: nazwa wiersza umieszczana w pierwszej kolumnie wiersza, nazwa zmiennej procesowej i agregatu służące do pobrania danych archiwalnych, sposób formatowania wartości (jeżeli nie zostanie podany, to użyty zostanie atrybut formatu z bazy definicji zmiennych lub nastąpi formatowanie standardowe zależne od typu wartości), sposób postępowania w przypadku braku lub niepoprawnej wartości, funkcja tworzenia zawartości komórki (jej użycie zostanie opisane w rozdziale *Kontrola zawartości komórki*).

W przykładzie użyto stałej *ReportTableCellErrorBehavior.Empty* do określenia trybu postępowania w wypadku braku poprawnej wartości. Stała ta oznacza pozostawienie pustej komórki. Inne standardowe tryby są wyjaśnione w opisie metody *AddColumn* w części referencyjnej podręcznika.

Przykładowy skrypt używa prostszego wariantu metody *AddRow*. Dostępny jest także wariant pozwalający podać nazwy stylów używanych w kolumnie. Style podane jawnie mają pierwszeństwo przed stylami standardowymi. Rozszerzony wariant metody pozwala indywidualnie określić sposób tworzenia poszczególnych tabel.

Po dodaniu kolumn pozostaje jeszcze określić sposób tworzenia krawędzi komórek tabeli (metoda *SetEdgeStyle*) oraz parametry zapytania o dane archiwalne (metoda *SetQueryData*).

Końcowe wywołanie metody *Render* powoduje wykonanie wszystkich niezbędnych zapytań o dane archiwalne oraz wygenerowanie treści raportu.

4.5 Kontrola zawartości komórek

W niektórych przypadkach konieczna jest dodatkowa kontrola nad zawartością komórek, np. potrzeba wykrywania przekroczenia wartości granicznych, niestandardowa reakcja na niepoprawne wartości, potrzeba nietypowej generacji treści komórki. W takim przypadku można zastosować mechanizm funkcji generacji treści komórek, który zostanie omówiony w tym rozdziale. Celem przykładowego skryptu jest uzyskanie poniższej tabeli, w której przekroczenie wartości granicznej jest sygnalizowane dodaniem do wartości znaku **!** oraz zmianą koloru komórki.

Godzina	V1 średnia	V1 minimum	V1 maksimum
2012-11-01 00:00:00	! 26,19	25,50	26,25
2012-11-01 03:00:00	! 24,87	24,00	25,50
2012-11-01 06:00:00	23,45	22,50	24,00
2012-11-01 09:00:00	21,84	21,00	22,50
2012-11-01 12:00:00	20,52	20,25	21,00
2012-11-01 15:00:00	20,04	19,50	20,25
2012-11-01 18:00:00	19,50	19,50	19,50
2012-11-01 21:00:00	19,11	18,75	19,50

Rys. Tabela z kontrolą zawartości komórek.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class VertTable : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         DateTime reportDate = DateTime.Parse(aParameters[0]).Date;
         string outputFile = aParameters[1];

         Report report = aApplication.CreateReport();


         // Define standard and custom styles
         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
                         true, true, ParagraphAlignment.Center, Colors.LightYellow);
         report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.Gray);
         report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.DarkGray);
         report.SetStyle("OverLimit", "Calibri", 10, Colors.Red, false, false,
                         ParagraphAlignment.Right, Colors.Yellow);

         // Build table
         ReportTable vt = report.AddVerticalTable(true);
         vt.LeftIndent(10);
         vt.AddColumn("Godzina", ReportTable.StartTimeColumn, null, 40, null,
                      ReportTableCellErrorBehavior.Normal, null);
         vt.AddColumn("V1 średnia", "sym01", "average", 30, "f2",
                      ReportTableCellErrorBehavior.Empty, CheckLimit);
         vt.AddColumn("V1 minimum", "sym01", "min", 30, "f2",
                      ReportTableCellErrorBehavior.Empty, null);
         vt.AddColumn("V1 maksimum", "sym01", "max", 30, "f2",
                      ReportTableCellErrorBehavior.Empty, null);
         vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan);
         vt.SetQueryData(reportDate, reportDate.AddDays(1), "3H");

         report.Render(outputFile);
      }
      catch (Exception e)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
                                    "Error generating report: " + e.ToString());
      }
```

```
        return true;
    }

    bool CheckLimit(ReportTableCell c)
    {
        if ((c.ValueStatus & Askom.Ascent.StatusOpc.QualityMask) ==
                                                    Askom.Ascent.StatusOpc.Good &&
            c.RawValue != null && (double)c.RawValue >= 24)
        {
            c.Content = "! " + c.Content;
            c.Style = "OverLimit";
        }
        return false;
    }

    public void FinalizeScript()
    {
    }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class VertTableVb
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize
        Try


            Dim reportDate As DateTime = DateTime.Parse(aParameters(0)).Date
            Dim outputFile As String = aParameters(1)

            Dim report As Report = aApplication.CreateReport()


            ' Define standard And custom styles
            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,_
                            True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,_
                            False, False, ParagraphAlignment.Right, Colors.Gray)
            report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,_
                            False, False, ParagraphAlignment.Right, Colors.DarkGray)
            report.SetStyle("OverLimit", "Calibri", 10, Colors.Red, False, False, _
                            ParagraphAlignment.Right, Colors.Yellow)

            ' Build table
            Dim vt As ReportTable = report.AddVerticalTable(True)
            vt.LeftIndent(10)
            vt.AddColumn("Godzina", ReportTable.StartTimeColumn, Nothing, 40, Nothing, _
                         ReportTableCellErrorBehavior.Normal, Nothing)
            vt.AddColumn("V1 średnia", "sym01", "average", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, AddressOf CheckLimit)
            vt.AddColumn("V1 minimum", "sym01", "min", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.AddColumn("V1 maksimum", "sym01", "max", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, Nothing)

            vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan)
            vt.SetQueryData(reportDate, reportDate.AddDays(1), "3H")

            report.Render(outputFile)

        Catch e As Exception
```

```
            aApplication.ReportMessage(EvoMessageType.Error, _
                                "Error generating report: " + e.ToString())


        End Try

        Return True

    End Function

    Function CheckLimit(c As ReportTableCell) As Boolean

        If ((c.ValueStatus And Askom.Ascent.StatusOpc.QualityMask) = _
                                                    Askom.Ascent.StatusOpc.Good And _
             c.RawValue IsNot Nothing And CType(c.RawValue, Double) >= 24) Then

            c.Content = "! " + c.Content
            c.Style = "OverLimit"

        End If
        Return False


    End Function


    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

End Class
```

Przykładowy kod to modyfikacja skryptu generującego tabelę pionową. Zmienione fragmenty zostały oznaczone wytłuszczoną czcionką.

Pierwszy nowy element to dodanie stylu *OverLimit*, który będzie używany w komórkach z przekroczoną wartością. Druga nowość to użycie w jednym z wywołań metody *AddColumn* odwołania do funkcji *CheckLimit*.

Najważniejszą częścią skryptu jest treść funkcji *CheckLimit*. Funkcja ta jest wywoływana dla każdej komórki kolumny, a w wywołaniu przekazywany jest jej obiekt klasy *ReportTableCell*. Poprzez ten obiekt funkcja uzyskuje dostęp do danych opisujących komórkę. Obiekt służy także do zmiany zawartości komórki.

W typowych scenariuszach funkcja kontroli zawartości komórki używa pól *ValueStatus* i *RawValue* obiektu *ReportTableCell* do podjęcia decyzji dotyczącej ewentualnej ingerencji w zawartość komórki. Pola te przechowują status wartości i oryginalną wartość numeryczną komórki (w tym przypadku pochodzącą z zapytania o dane archiwalne). W przypadku gdy zachodzi konieczność zmiany zawartości komórki, można to wykonać przez ustawienie w polu *Style* nowej nazwy stylu i w polu *Content* nowej tekstowej treści komórki. Pole *Content* w momencie wywołania funkcji jest ustawione na treść, która byłaby umieszczona w komórce, gdyby nie była użyta funkcja kontroli. W przykładowym kodzie wykorzystano to, żeby wyświetlić standardową zawartość z dodanym znakiem wykrzyknika. Zwrócenie wartości *false* oznacza, że system generacji ma wygenerować treść komórki w sposób normalny, ale z ewentualnym wykorzystaniem zmienionych pól *Content* i *Style*.

Bardziej zaawansowaną metodą zmiany treści komórki jest własna generacja kompletnej zawartości, wykorzystując obiekt bazowy komórki (dostępny poprzez właściwość *MigraCell*) oraz jego metody. Ten tryb pozwala na większy wpływ na postać komórki (np. dodanie obrazka), ale wymaga działania

bezpośrednio na obiektach i metodach biblioteki *MigraDoc*. Jeżeli funkcja kontroli zawartości komórki tworzy treść bezpośrednio, to musi ona zwrócić wartość *true*.

4.6 Kolumny i wiersze wyliczane

W tym rozdziale przedstawiony zostanie sposób dodania do tabe raportowych wierszy i kolumn wyliczających podsumowania danych. Pokazany dalej skrypt tworzy typową tabelę pionową uzupełnioną o trzy dodatkowe wiersze wyliczane. Wygląd generowanej tabeli pokazuje ilustracja.

Godzina	V1 średnia	V1 minimum	V1 maksimum
2016-03-17 00:00:00	21,48	19,14	24,31
2016-03-17 03:00:00	20,85	18,96	! 25,75
2016-03-17 06:00:00	22,44	18,69	24,18
2016-03-17 09:00:00	22,92	18,58	24,15
2016-03-17 12:00:00	20,96	18,22	! 25,87
2016-03-17 15:00:00	20,53	18,89	24,11
2016-03-17 18:00:00	22,22	19,13	! 25,14
2016-03-17 21:00:00	20,93	18,19	! 25,18
Minimum	20,53	18,19	24,11
Maksimum	22,92	19,14	25,87
Przekroczenia			4

Rys. Tabela pionowa dla danych archiwalnych z wierszami wyliczanymi.

Dwa pierwsze wiersze wyliczane służą do pokazania wartości minimalnej i maksymalnej w poszczególnych kolumnach tabeli. W trzecim wierszu pokazywana jest liczba przekroczeń wartości 25 w kolumnie *V1 Maksimum.*

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class VertTable : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         DateTime reportDate = DateTime.Parse(aParameters[0]).Date;
         string outputFile = aParameters[1];

         Report report = aApplication.CreateReport();

         // Define standard and custom styles
         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
```

```
                        true, true, ParagraphAlignment.Center, Colors.LightYellow);
        report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,
                        false, false, ParagraphAlignment.Right, Colors.Gray);
        report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10, Colors.Blue,
                        false, false, ParagraphAlignment.Right, Colors.DarkGray);
        report.SetStyle("OverLimit", "Calibri", 10, Colors.Red, false, false,
                        ParagraphAlignment.Right, Colors.Yellow);

        // Build table
        ReportTable vt = report.AddVerticalTable(true);
        vt.LeftIndent(10);
        vt.AddColumn("Godzina", ReportTable.StartTimeColumn, null, 40, null,
                     ReportTableCellErrorBehavior.Normal, null);
        vt.AddColumn("V1 średnia", "sym01", "average", 30, "f2",
                     ReportTableCellErrorBehavior.Empty, null);
        vt.AddColumn("V1 minimum", "sym01", "min", 30, "f2",
                     ReportTableCellErrorBehavior.Empty, null);
        vt.AddColumn("V1 maksimum", "sym01", "max", 30, "f2",
                     ReportTableCellErrorBehavior.Empty, CheckLimit);

        vt.AddCalculatedRow("Minimum", CalculationType.Min, "f2", null);
        vt.AddCalculatedRow("Maksimum", CalculationType.Max, "f2", null);
        vt.AddCalculatedRow("Przekroczenia", CalculationType.Custom, "f2", OverLimit);

        vt.SetQueryData(reportDate, reportDate.AddDays(1), "3H");
        vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan);

        report.Render(outputFile);
     }
     catch (Exception e)
     {
        aApplication.ReportMessage(EvoMessageType.Error,
                                   "Error generating report: " + e.ToString());
     }

     return true;
  }

  bool OverLimit(ReportTableCell c)
  {
     int overCount = 0;
     if (c.Index == 3)
     {
        foreach (CellValue v in c.SourceValues)
        {
           if ((v.Status & Askom.Ascent.StatusOpc.QualityMask) ==
                     Askom.Ascent.StatusOpc.Good && v.Value != null && (double)v.Value >= 25)

              overCount++;
        }
        c.Content = overCount.ToString();
     }
     return false;
  }

  bool CheckLimit(ReportTableCell c)
  {

     if ((c.ValueStatus & Askom.Ascent.StatusOpc.QualityMask) ==
              Askom.Ascent.StatusOpc.Good && c.RawValue != null && (double)c.RawValue >= 25)
     {
        c.Content = "! " + c.Content;
        c.Style = "OverLimit";
     }
     return false;
  }

  public void FinalizeScript()
  {

  }
}
```

VB

```vb
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class VertTableVb
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize
        Try

            Dim reportDate As DateTime = DateTime.Parse(aParameters(0)).Date
            Dim outputFile As String = aParameters(1)

            Dim report As Report = aApplication.CreateReport()

            ' Define standard And custom styles
            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12,Colors.Blue, _
               True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue, _
               False, False, ParagraphAlignment.Right, Colors.Gray)
            report.SetStyle(ReportStyleStandardNames.TableAltRow, "Calibri", 10,Colors.Blue, _
               False, False, ParagraphAlignment.Right, Colors.DarkGray)
            report.SetStyle("OverLimit", "Calibri", 10, Colors.Red, False, False, _
               ParagraphAlignment.Right, Colors.Yellow)

            ' Build table
            Dim vt As ReportTable = report.AddVerticalTable(True)
            vt.LeftIndent(10)
            vt.AddColumn("Godzina", ReportTable.StartTimeColumn, Nothing, 40, Nothing, _
                         ReportTableCellErrorBehavior.Normal, Nothing)
            vt.AddColumn("V1 średnia", "sym01", "average", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.AddColumn("V1 minimum", "sym01", "min", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, Nothing)
            vt.AddColumn("V1 maksimum", "sym01", "max", 30, "f2", _
                         ReportTableCellErrorBehavior.Empty, AddressOf CheckLimit)

            vt.AddCalculatedRow("Minimum", CalculationType.Min, "f2", Nothing)
            vt.AddCalculatedRow("Maksimum", CalculationType.Max, "f2", Nothing)
            vt.AddCalculatedRow("Przekroczenia", CalculationType.Custom, "f2", _
                                AddressOf OverLimit)

            vt.SetQueryData(reportDate, reportDate.AddDays(1), "3H")
            vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan)

            report.Render(outputFile)

        Catch e As Exception

            aApplication.ReportMessage(EvoMessageType.Error, _
                                       "Error generating report: " + e.ToString())

        End Try

        Return True

    End Function

    Function OverLimit(c As ReportTableCell) As Boolean

        Dim overCount As Integer = 0
        Dim v As CellValue
        If c.Index = 3 Then

            For Each v In c.SourceValues
                If ((v.Status And Askom.Ascent.StatusOpc.QualityMask) = _
                         Askom.Ascent.StatusOpc.Good And _
                         v.Value IsNot Nothing And CType(v.Value, Double) >= 25) Then

                    overCount = overCount + 1
```

```
                End If
            Next
            c.Content = overCount

        End If

        Return False

    End Function

    Function CheckLimit(c As ReportTableCell) As Boolean

        If ((c.ValueStatus And Askom.Ascent.StatusOpc.QualityMask) = _
                    Askom.Ascent.StatusOpc.Good And _
                    c.RawValue IsNot Nothing And CType(c.RawValue, Double) >= 25) Then

            c.Content = "! " + c.Content
            c.Style = "OverLimit"

        End If

        Return False

    End Function


    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Przykładowy kod tworzy w standardowy sposób tabelę pionową. Wiersze wyliczane dodawane są do tabeli przy pomocy wywołań metody *AddCalculatedRow*. Nie ma ograniczenia na liczbę wierszy wyliczanych. W podobny sposób można by dodać kolumny wyliczane przy pomocy metody *AddCalculatedColumn.* Wiersze i kolumny wyliczane mogą być dodawane do tabel dowolnego typu.

Dwa pierwsze użycia *AddCalculatedRow* dodają standardowe wiersze wyliczane, służące do pokazanie wartości minimalnej i maksymalnej w poszczególnych kolumnach tabeli. Trzeci wiersz jest wyliczany niestandardowo. W tym celu w parametrach wywołania przekazywany jest typ wyliczenia *Custom* oraz adres funkcji *OverLimit* wykonującej wyliczenie wartości komórek wiersza.

Funkcja *OverLimit* sprawdza indeks kolumny, której dotyczy wywołanie. Wyłącznie dla kolumny wartości maksymalnych, wyliczana jest liczba przekroczeń wartości 25. Dostęp do wartości wszystkich komórek kolumny odbywa się za pośrednictwem właściwości *SourceValues* komórki wyliczanej.

4.7 Raport na podstawie danych przekazanych w kolekcji ICollection

Nie zawsze źródłem danych, na podstawie których należy wygenerować raport są wartości historyczne zmiennych procesowych. Jeżeli dane pochodzą z innego źródła, to można je zgromadzić w dowolnej kolekcji spełniającej interfejs *ICollection*. W praktyce oznacza to użycie dowolnej kolekcji platformy .Net, np. *List<T>*, lub nawet zwykłej tablicy.

Omówiony w tym rozdziale skrypt będzie tworzył raport pokazujący wartości bieżące zmiennych procesowych z chwili wykonania raportu.

Zmienna	Wartość
sym01	30,00
sym02	130,00
sym03	2,70

Rys. Raport pokazujący wartości bieżące zmiennych procesowych z chwili wykonania raportu.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class CollectionReport : IScript
{
   class VData
   {
      public VData(string name, double value)
      {
         varname = name; varvalue = value;
      }
      public string varname { get; private set; }
      public double varvalue { get; private set; }
   }

   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         string outputFile = aParameters[1];

         VData[] values = {
             new VData("sym01",
                   Convert.ToDouble(aApplication.GetVariableValue("sym01", true).RawValue)),
             new VData("sym02",
                   Convert.ToDouble(aApplication.GetVariableValue("sym02", true).RawValue)),
             new VData("sym03",
                   Convert.ToDouble(aApplication.GetVariableValue("sym03", true).RawValue)),
         };

         Report report = aApplication.CreateReport();

         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
                         true, true, ParagraphAlignment.Center, Colors.LightYellow);
         report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.Gray);

         ReportTable vt = report.AddVerticalTable(values, true);
         vt.LeftIndent(30);
         vt.AddColumn("Zmienna", "varname", 50, null, null);
         vt.AddColumn("Wartość", "varvalue", 50, "f2", null);
         vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan);

         report.Render(outputFile);
      }
      catch (Exception e)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
                                    "Error generating report: " + e.ToString());
      }



      return true;
```

```
    }

    public void FinalizeScript()
    {
    }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class CollectionReportVb
    Implements IScript

    Class VData
        Dim _varname As String
        Public Property varname As String
            Get
                Return _varname
            End Get
            Set(value As String)
                _varname = Value
            End Set
        End Property

        Dim _varvalue As Double
        Public Property varvalue As Double
            Get
                Return _varvalue
            End Get
            Set(value As Double)
                _varvalue = Value
            End Set
        End Property

        Public Sub New(ByVal name As String, ByVal value As Double)
            varname = name
            varvalue = value
        End Sub
    End Class

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize
        Try

            Dim outputFile As String = aParameters(1)

            Dim values As VData() = New VData(2) _
                { _
                New VData("sym01", Convert.ToDouble(aApplication.GetVariableValue _

                                                                ("sym01", True).RawValue)), _
                New VData("sym02", Convert.ToDouble(aApplication.GetVariableValue _
                                                                ("sym02", True).RawValue)), _
                New VData("sym03", Convert.ToDouble(aApplication.GetVariableValue _
                                                                 ("sym03", True).RawValue)) _
                }

           ' Create report
            Dim report As Report = aApplication.CreateReport()

            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, _
                       Colors.Blue, True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, _
                       Colors.Blue, False, False, ParagraphAlignment.Right, Colors.Gray)
```

```
            Dim vt As ReportTable = report.AddVerticalTable(values, True)
            vt.LeftIndent(30)
            vt.AddColumn("Zmienna", "varname", 50, Nothing, Nothing)
            vt.AddColumn("Wartość", "varvalue", 50, "f2", Nothing)
            vt.SetEdgeStyle(ReportTableEdgeType.Columns, Colors.Cyan)

            report.Render(outputFile)

        Catch e As Exception
            aApplication.ReportMessage(EvoMessageType.Error, _
                                       "Error generating report: " + e.ToString())
        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Suk

End Class
```

Poniższe wyjaśnienia dotyczą tylko elementów specyficznych dla raportów z kolekcji danych.

W przypadku tworzenia tabeli na podstawie danych z kolekcji pierwszym krokiem jest utworzenie klasy służącej do przechowania danych. Obiekty tej klasy będą następnie dodawane do kolekcji.

W przykładowym skrypcie zdefiniowana jest klasa *VData*. Jedynym wymaganiem co do konstrukcji klasy używanej w systemie raportowania jest zdefiniowanie grupy publicznych właściwości, które będą służyły do odczytu wartości umieszczanych w tabeli. W naszym przykładzie są to właściwości *varname* i *varvalue*.

W początkowej fazie skryptu tworzona jest tablica obiektów klasy *VData*. Następnie do tablicy wstawiane są trzy obiekty *VData*, w których przechowana jest nazwa zmiennej procesowej oraz jej bieżąca wartość odczytywana metodą *GetVariableValue*.

Dalsza część skryptu to właściwy kod tworzący raport zawierający pojedynczą tabelę pionową. Kolekcja danych, na podstawie których powstanie tabela, jest przekazywana w momencie jej utworzenia metodą *AddVerticalTable*. Metoda *SetQueryData* w tym schemacie działania nie jest używana.

Kolejnym krokiem jest dodanie do tabeli zestawu kolumn, wykorzystując metodę *AddColumn*. W przypadku tabel korzystających z danych kolekcji znaczenie parametrów jest następujące: nazwa kolumny umieszczana w linii nagłówka, nazwa publicznej właściwości zdefiniowanej w klasie służącej do przechowania danych kolekcji, szerokość kolumny, sposób formatowania wartości (jeżeli nie zostanie podany, to nastąpi formatowanie standardowe zależne od typu wartości), funkcja tworzenia zawartości komórki (jej użycie jest identyczne jak dla tabel z wartościami historycznymi i zostało opisane w rozdziale *4.5 Controla zawartości komórek*).

4.8 Raport na podstawie danych przekazywanych w obiekcie typu DataTable

W przypadku gdy dane źródłowe raportu są przechowywane w bazie danych najwygodniejszym mechanizmem jest bezpośrednie wykorzystanie systemowej klasy *DataTable*. Przykładowy skrypt będzie tworzył prosty raport tabelaryczny na podstawie danych odczytywanych z bazy danych SQL.

Produkt	Jakość	Waga
A12_1000	1	12,45
A12_1001	2	11,27
A10_178	1	11,10

Rys. Raport tabelaryczny na podstawie danych odczytywanych z bazy danych SQL.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;
using System.Data;
using System.Data.SqlClient;

public class DataTableRep : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         string outputFile = aParameters[1];

         DataSet dataSet;
         using (SqlConnection connection = new SqlConnection(
                  "Data Source=(local);Initial Catalog=ReportTest;Integrated Security=SSPI"))
         {
            SqlDataAdapter adapter = new SqlDataAdapter();
            connection.Open();
            SqlCommand command =
                new SqlCommand("SELECT ProdID, Quality, Weight FROM ProdInfo", connection);
            adapter.SelectCommand = command;
            dataSet = new DataSet();
            adapter.Fill(dataSet);
         }

         Report report = aApplication.CreateReport();

         report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12, Colors.Blue,
                         true, true, ParagraphAlignment.Center, Colors.LightYellow);
         report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue,
                         false, false, ParagraphAlignment.Right, Colors.Gray);


         ReportTable vt = report.AddVerticalTable(dataSet.Tables[0], true);
         vt.LeftIndent(20);
         vt.AddColumn("Produkt", "ProdId", 25, null, null);
         vt.AddColumn("Jakość", "Quality", 25, null, null);
         vt.AddColumn("Waga", "Weight", 25, "f2", null);
```

```
         vt.SetEdgeStyle(ReportTableEdgeType.Outline, Colors.Cyan);
         report.Render(outputFile);

         dataSet.Dispose();
      }
      catch (Exception e)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
                                          "Error generating report: " + e.ToString());
      }

      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts
Imports System.Data
Imports System.Data.SqlClient

Public Class DataTableVb
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        Try

            Dim outputFile As String = aParameters(1)

            Dim dataSet As DataSet
            Using connection As SqlConnection = New SqlConnection( _
                   "Data Source=(local);Initial Catalog=ReportTest;Integrated Security=SSPI")

                Dim adapter As SqlDataAdapter = New SqlDataAdapter()
                connection.Open()
                Dim command As SqlCommand = _
                   New SqlCommand("SELECT ProdID, Quality, Weight FROM ProdInfo", connection)
                adapter.SelectCommand = command
                dataSet = New DataSet()
                adapter.Fill(dataSet)

            End Using



            Dim report As Report = aApplication.CreateReport()

            report.SetStyle(ReportStyleStandardNames.TableHeader, "Calibri", 12,Colors.Blue, _
                            True, True, ParagraphAlignment.Center, Colors.LightYellow)
            report.SetStyle(ReportStyleStandardNames.TableRow, "Calibri", 10, Colors.Blue, _
                            False, False, ParagraphAlignment.Right, Colors.Gray)


            Dim vt As ReportTable = report.AddVerticalTable(dataSet.Tables(0), True)
            vt.LeftIndent(20)
            vt.AddColumn("Produkt", "ProdId", 25, Nothing, Nothing)
            vt.AddColumn("Jakość", "Quality", 25, Nothing, Nothing)
            vt.AddColumn("Waga", "Weight", 25, "f2", Nothing)

            vt.SetEdgeStyle(ReportTableEdgeType.Outline, Colors.Cyan)
```

```
            report.Render(outputFile)

            dataSet.Dispose()

        Catch e As Exception

            aApplication.ReportMessage(EvoMessageType.Error, _
                                                "Error generating report: " + e.ToString())

        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

End Class
```

Poniższe wyjaśnienia dotyczą tylko elementów specyficznych dla raportów z danych obiektu *DataTable*.

Początkowy fragment skryptu jest standardowym kodem służącym do odczytu danych przy pomocy zapytania SQL z bazy danych. Rezultat zapytania zostaje zapamiętany w obiekcie klasy *DataSet*.

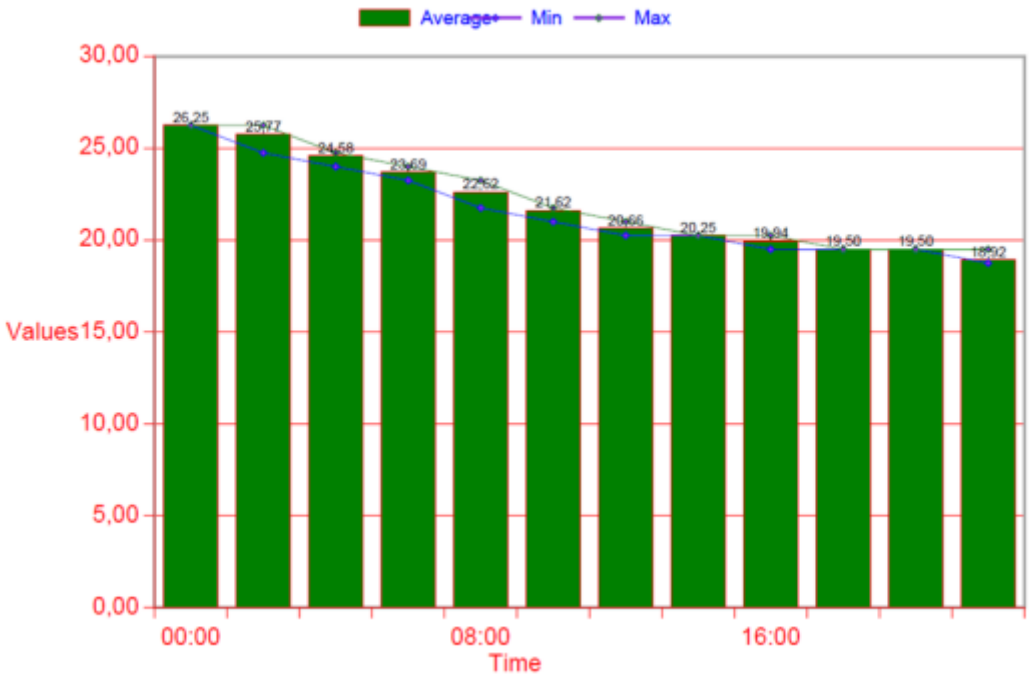
Dalsza część skryptu to właściwy kod tworzący raport zawierający pojedynczą tabelę pionową. W wywołaniu metody *AddVerticalTable* przekazywane jest odwołanie do obiektu *DataTable,* znajdującego się wewnątrz obiektu *DataSet,* wypełnionego danymi w początkowej fazie skryptu. Metoda *SetQueryData* w tym schemacie działania nie jest używana.

Kolejnym krokiem jest dodanie do tabeli zestawu kolumn, wykorzystując metodę *AddColumn*. W przypadku tabel korzystających z danych z obiektu *DataTable* znaczenie parametrów jest następujące: nazwa kolumny tabeli umieszczana w linii nagłówka, nazwa kolumny w obiekcie DataTable (nazwy te zależą od wykonanego zapytania SQL i struktury bazy danych), szerokość kolumny, sposób formatowania wartości (jeżeli nie zostanie podany, to nastąpi formatowanie standardowe zależne od typu wartości), funkcja tworzenia zawartości komórki (jej użycie jest identyczne jak dla tabel z wartościami historycznymi i zostało opisane w rozdziale *4.5 Kontrola zawartości komórek*).

4.9 Użycie wykresu w treści raportu

Oprócz tabel do treści raportu mogą być także wstawiane wykresy. Podobnie jak w przypadku tabel, wykresy mogą być tworzone na podstawie danych pochodzących z odczytu danych archiwalnych zmiennych procesowych lub danych dostarczonych w kolekcji lub obiekcie *DataTable*.

Przykładowy skrypt służy do generacji raportu pokazanego na ilustracji.



Rys. Raport z wykresem.

C#

```
using System;
using Askom.AsixEvo.Scripting;
using MigraDoc.DocumentObjectModel;
using MigraDoc.DocumentObjectModel.Shapes;
using MigraDoc.DocumentObjectModel.Shapes.Charts;

public class Wykres : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      try
      {
         DateTime reportDate = DateTime.Parse(aParameters[0]).Date;
         string outputFile = aParameters[1];



         // Create report
         Report report = aApplication.CreateReport();

         // Define standard and custom styles
         report.SetStyle(ReportStyleStandardNames.Normal, "Arial", 10, Colors.Black,
                         false, false, ParagraphAlignment.Left, Color.Empty);
         report.SetStyle("Axis", "Arial", 10, Colors.Red, false, false,
                         ParagraphAlignment.Left, Color.Empty);
         report.SetStyle("Legend", "Arial", 8, Colors.Blue, false, false,
                         ParagraphAlignment.Left, Color.Empty);
         report.SetStyle("Labels", "Arial", 6, Colors.Black, false, false,
                         ParagraphAlignment.Left, Color.Empty);

         ReportChart rc = report.AddChart(0, 150, 100);

         rc.SetXAxisTitle("Time", "Axis");
```

```
         rc.CustomizeXAxis(Colors.Red, "Axis", "HH:mm", 4);

         rc.SetYAxisTitle("Values", "Axis");
         rc.CustomizeYAxis(null, null, null, true, Colors.Red, "Axis", "f2");

         rc.AddLegend(ReportChartLegendPosition.Top, "Legend");

         ReportChartSeries s = rc.AddSeries("Average", ChartType.Column2D,
                                            "sym01", "average", Colors.Red, Colors.Green);
         s.SetDataLabels("Labels", "f2", DataLabelPosition.OutsideEnd);
         s = rc.AddSeries("Min", ChartType.Line, "sym01", "min", Colors.Blue, Color.Empty);
         s.SetMarkersStyle(MarkerStyle.Diamond, 1, Colors.Blue, Colors.BlueViolet);

         rc.SetQueryData(reportDate, reportDate.AddDays(1), TimeSpan.FromHours(2));
         report.Render(outputFile);
      }
      catch (Exception e)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
                                    "Error generating report: " + e.ToString());
      }

      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports Askom.AsixEvo.Scripting
Imports MigraDoc.DocumentObjectModel
Imports MigraDoc.DocumentObjectModel.Shapes
Imports MigraDoc.DocumentObjectModel.Shapes.Charts

Public Class WykresVb
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                               As IApplication) As Boolean Implements IScript.Initialize
        Try

            Dim reportDate As DateTime = DateTime.Parse(aParameters(0)).Date
            Dim outputFile As String = aParameters(1)

            ' Create report
            Dim report As Report = aApplication.CreateReport()


            report.SetStyle(ReportStyleStandardNames.Normal, "Arial", 10, Colors.Black, _
                            False, False, ParagraphAlignment.Left, Color.Empty)
            report.SetStyle("Axis", "Arial", 10, Colors.Red, False, False, _
                            ParagraphAlignment.Left, Color.Empty)
            report.SetStyle("Legend", "Arial", 8, Colors.Blue, False, False, _
                            ParagraphAlignment.Left, Color.Empty)
            report.SetStyle("Labels", "Arial", 6, Colors.Black, False, False, _
                            ParagraphAlignment.Left, Color.Empty)

            Dim rc As ReportChart = report.AddChart(0, 150, 100)

            rc.SetXAxisTitle("Time", "Axis")
            rc.CustomizeXAxis(Colors.Red, "Axis", "HH:mm", 4)

            rc.SetYAxisTitle("Values", "Axis")
            rc.CustomizeYAxis(Nothing, Nothing, Nothing, True, Colors.Red, "Axis", "f2")
```

```
            rc.AddLegend(ReportChartLegendPosition.Top, "Legend")

            Dim s As ReportChartSeries = rc.AddSeries("Average", ChartType.Column2D, _
                                              "sym01", "average", Colors.Red, Colors.Green)
            s.SetDataLabels("Labels", "f2", DataLabelPosition.OutsideEnd)
            s = rc.AddSeries("Min", ChartType.Line, "sym01", "min", Colors.Blue, Color.Empty)
            s.SetMarkersStyle(MarkerStyle.Diamond, 1, Colors.Blue, Colors.BlueViolet)

            rc.SetQueryData(reportDate, reportDate.AddDays(1), TimeSpan.FromHours(2))
            report.Render(outputFile)

        Catch e As Exception

            aApplication.ReportMessage(EvoMessageType.Error, _
                                       "Error generating report: " + e.ToString())

        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
     End Sub

End Class
```

W początkowej części skryptu definiowany jest zestaw stylów, które zostaną użyte w różnych częściach składowych tworzonego wykresu.

Pierwszym obowiązkowym fragmentem kodu jest wywołanie metody *AddChart*, które wstawia wykres do raportu oraz tworzy obiekt, za pośrednictwem którego będą wykonywane wszystkie następne operacje. W parametrach wywołania podawane jest położenie i rozmiar wykresu.

Kolejne wywołania dotyczą opcjonalnych parametryzacji osi wykresu oraz legendy.

Metoda *SetXAxisTitle* dodaje opis osi X jednocześnie określając nazwę stylu. Metoda *CustomizeXAxis* określa sposób rysowania skali osi X. Zdefiniowany jest kolor linii, styl elementów tekstowych, sposób formatowania opisów (dla wykresów danych archiwalnych oś X jest zawsze osią czasu) oraz gęstość opisów osi (wartość 1 oznacza opis wszystkich punktów, użyta wartość 4 oznacza opisywanie co czwartego punktu).

Metoda *SetYAxisTitle* dodaje opis osi Y jednocześnie określając nazwę stylu. Metoda *CustomizeYAxis* określa sposób rysowania skali osi Y. Pierwsze 3 parametry służą do zdefiniowania zakresu i podziału skali - użyte wartości *null* (*Nothing*) oznaczają automatyczne skalowanie. Kolejne parametry określają, czy rysować siatkę wykresu, kolor linii, styl elementów tekstowych oraz sposób formatowania opisów skali.

Opcjonalne wywołanie metody AddLegend dodaje do wykresu legendę, jednocześnie określając jej położenie oraz używany styl.

Następna część skryptu to właściwy kod budujący zawartość wykresu. Wywołania metody AddSeries dodają do wykresu kolejne serie danych. Parametry wywołań oznaczają: nazwa serii (używana w legendzie wykresu), typ graficzny serii, nazwa zmiennej procesowej i odczytywanego agregatu wartości archiwalnych, kolory używane do rysowania punktów serii (ich dokładne znaczenie zależy od użytego typu serii). Możliwe jest mieszanie na jednym wykresie serii różnych typów, ale nie wszystkie kombinacje są dozwolone.

Po utworzeniu obiektu serii danych można na nim wykonywać dalsze parametryzacje. Pokazane wywołania metod *SetDataLabels* i *SetMarkerStyle* pozwalają dodać do serii etykiety wartości i znaczniki punktów. Dostępność tych metod zależy jednak od użytego typu serii.

W przypadku wykresów tworzonych na podstawie danych kolekcji lub zapisanych w obiekcie *DataTable* sposób postępowania jest zbliżony do pokazanego w przykładzie. W wywołaniu metody *AddChart* należy dodatkowo podać odwołanie do obiektu, w którym dane są przechowywane. W wywołaniach *AddSeries* należy podać nazwę pola lub kolumny zamiast nazwy zmiennej i agregatu. Dodatkowo można użyć metodę *AddXSeries* w celu określenia serii danych służących do opisu skali osi X (brak wywołania spowoduje, że punkty skali będą opisane kolejnymi numerami).

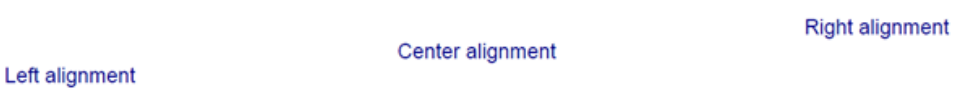
4.10 Metody pozycjonowania tekstów

Jak już zostało wcześniej wspomniane głównym mechanizmem pozycjonowania tekstów jest wykorzystanie stylów definiowanych metodą *SetStyle*, a dokładnie parametru wyrównania. Dotyczy to wszystkich tekstów: umieszczanych wewnątrz komórek tabeli, używanych w nagłówku lub stopce, wstawianych do raportu metodą *AddParagraph*.

```
report.SetStyle("Left", "Arial", 10, Colors.DarkBlue, false, false,
                ParagraphAlignment.Left, Color.Empty);
report.SetStyle("Right", "Arial", 10, Colors.DarkBlue, false, false,
                ParagraphAlignment.Right, Color.Empty);
report.SetStyle("Center", "Arial", 10, Colors.DarkBlue, false, false,
                ParagraphAlignment.Center, Color.Empty);

report.AddParagraph("Right alignment", "Right");
report.AddParagraph("Center alignment", "Center");
report.AddParagraph("Left alignment", "Left");
```

Wykonanie kodu pokazanego powyżej wygeneruje następujący fragment raportu:

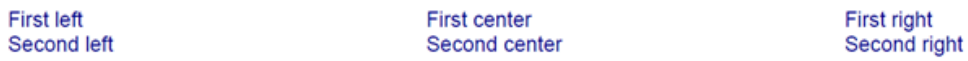


Bardziej skomplikowany jest przypadek, gdy w tej samej linii chcemy umieścić fragmenty tekstu z różnym wyrównaniem lub zachodzi potrzeba wyświetlenia tekstu w kilku szpaltach. Można w tym

celu do tekstu wstawić odpowiednią ilość spacji, ale bardziej skuteczną metodą jest wykorzystanie znaków tabulacji połączone z deklarowaniem pozycji tabulacji w ramach stylu.

```
      report.SetStyle("Tabs", "Arial", 10, Colors.DarkBlue, false, false,
                           ParagraphAlignment.Left, Color.Empty).AddTabStops(70, 140);
     report.AddParagraph("First left\tFirst center\tFirst right", "Tabs");
     report.AddParagraph("Second left\tSecond center\tSecond right", "Tabs");
```

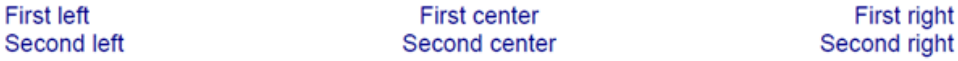
W powyższym kodzie do stylu *Tabs* została dodana deklaracja pozycji znaków tabulacji. Użyta została metoda *AddTabStops*, która określa pozycję dowolnej ilości znaków tabulacji z dokładnością do milimetra. Efekt wykonania kodu pokazuje ilustracja:



Jeżeli zachodzi potrzeba jeszcze dokładniejszej kontroli pozycjonowania, można posłużyć się metodą *AddTabStop*.

```
report.SetStyle("Tabs", "Arial", 10, Colors.DarkBlue, false, false,
              ParagraphAlignment.Left, Color.Empty).
            AddTabStop(70, TabAlignment.Center).AddTabStop(140, TabAlignment.Right);
     report.AddParagraph("First left\tFirst center\tFirst right", "Tabs");
     report.AddParagraph("Second left\tSecond center\tSecond right", "Tabs");
```

Metoda *AddStop* dodaje deklarację pojedynczego znaku tabulacji łącznie ze sposobem wyrównania. Przykład pokazuje, jak można w jednej instrukcji umieścić deklarację sekwencji znaków tabulacji. Efekt wykonania kodu pokazuje ilustracja.



5 Model obiektowy skryptów raportowych

5.1 Interfejs IApplication

Interfejs *IApplication* udostępnia jedną metodę związaną z systemem raportów skryptowych.

5.1.1 Metody

CreateReport

```
Report CreateReport();
```

Function **CreateReport** () As **Report**

Metoda tworzy i zwraca obiekt klasy *Report,* służący jako główny obiekt generowania raportu.

5.2 Klasa Report

Podstawowa klasa systemu raportów skryptowych. Reprezentuje dokument raportu jako całość i udostępnia metody służące do budowy treści raportu.

Obiekty klasy *Report* tworzone są za pośrednictwem metody *CreateReport* interfejsu *IApplication*.

5.2.1 Właściwości

Footer

```
ReportFooter Footer { get; }
```

ReadOnly Property **Footer** As **ReportFooter**

Właściwość zwraca obiekt stopki strony raportu.

Header

```
ReportHeader Header { get; }
```

ReadOnly Property **Header** As **ReportHeader**

Właściwość zwraca obiekt nagłówka strony raportu.

MigraDocument

```
MigraDoc.DocumentObjectModel.Document MigraDocument { get; }
```

ReadOnly Property **MigraDocument** As **MigraDoc.DocumentObjectModel.Document**

Właściwość zwraca obiekt *Document* biblioteki MigraDoc. Obiekt jest tworzony w momencie tworzenia obiektu klasy *Report*.

MigraSection

```
MigraDoc.DocumentObjectModel.Section MigraSection { get; }
```

ReadOnly Property **MigraSection** As **MigraDoc.DocumentObjectModel.Section**

Właściwość zwraca obiekt *Section* biblioteki *MigraDoc*. Sekcja jest tworzona w momencie tworzenia obiektu klasy *Report*. Jest to główna sekcja dokumentu, w której zostaną umieszczone wszystkie elementy raportu.

5.2.2 Metody

AddChart

```
ReportChart AddChart(int aLeft, int aWidth, int aHeight,
MigraDoc.DocumentObjectModel.Color aLineColor, MigraDoc.DocumentObjectModel.Color
```

```
aFillColor);

ReportChart AddChart(int aLeft, int aWidth, int aHeight);
```

```
Function AddChart(ByVal aLeft As Integer, ByVal aWidth As Integer, ByVal aHeight As Integer,
ByVal aLineColor As MigraDoc.DocumentObjectModel.Color, ByVal aFillColor
AsMigraDoc.DocumentObjectModel.Color) As ReportChart

Function AddChart(ByVal aLeft As Integer, ByVal aWidth As Integer, ByVal aHeight As Integer) As
ReportChart
```

Metoda dodaje do raportu wykres. Dane wyświetlanych na wykresie serii będą pochodziły z zapytania o dane archiwalne, którego parametry należy ustawić przy pomocy wywołania metody *SetQueryData* obiektu wykresu.

Parametr *aLeft* określa wyrażone w milimetrach przesunięcie całego wykresu w lewo. Parametry *aWidth* i *aHeight* określają szerokość i wysokość obszaru zajmowanego przez wykres.

Parametr *aLineColor* określa kolor linii obramowania wykresu, a parametr *aFillColor* kolor wypełnienia tła wykresu.

Metoda zwraca obiekt utworzonego wykresu.

```
ReportChart AddChart(object aData, int aLeft, int aWidth, int aHeight,
MigraDoc.DocumentObjectModel.Color aLineColor, MigraDoc.DocumentObjectModel.Color
aFillColor);

ReportChart AddChart(object aData, int aLeft, int aWidth, int aHeight);
```

```
Function AddChart(ByVal aData As Object, ByVal aLeft As Integer, ByVal aWidth As Integer, ByVal
aHeight As Integer, ByVal aLineColor As MigraDoc.DocumentObjectModel.Color, ByVal
aFillColor As MigraDoc.DocumentObjectModel.Color) As ReportChart

Function AddChart(ByVal aData As Object, ByVal aLeft As Integer, ByVal aWidth As Integer, ByVal
aHeight As Integer) As ReportChart
```

Metoda dodaje do raportu wykres, w którym wartości pokazywanych serii danych pochodzą z kolekcji przekazanej poprzez parametr aData. Obiekt danych serii musi być typu *System.Collections.ICollection* lub *System.Data.DataTable*.

Pozostałe parametry mają identyczne znaczenie, jak w przypadku metod tworzących wykres dla danych archiwalnych.

AddHorizontalTable

```
ReportTable AddHorizontalTable(bool aShowHeader, int aFirstWidth, int aNextWidth, string
aFirstHeader, string  aNextFormat);
ReportTable AddHorizontalTable(bool aShowHeader, int aFirstWidth, int aNextWidth,
string  aFirstHeader, string  aNextFormat, string  aFirstStyle, string  aNextStyle);
```

Function **AddHorizontalTable**(ByVal aShowHeader As **Boolean**, ByVal aFirstWidth As **Integer**, ByVal aNextWidth As **Integer**, ByVal aFirstHeader As **String**, ByVal aNextFormat As **String**) As **ReportTable**

Function **AddHorizontalTable**(ByVal aShowHeader As **Boolean**, ByVal aFirstWidth As **Integer**, ByVal aNextWidth As **Integer**, ByVal aFirstHeader As **String**, ByVal aNextFormat As **String**, ByVal aFirstStyle As **String**, ByVal aNextStyle As **String**) As **ReportTable**

Metoda dodaje do raportu tabelę w układzie poziomym, w którym serie danych pokazane są w pojedynczych wierszach. Dane serii będą pochodziły z zapytania o dane archiwalne, którego parametry należy ustawić przy pomocy wywołania metody *SetQueryData* obiektu tabeli.

Parametr *aShowHeader* określa, czy tabela będzie posiadała wiersz nagłówka. Parametry *aFirstWidth* i *aNextWidth* określają szerokość kolumn (w milimetrach), odpowiednio: pierwszej i wszystkich kolejnych kolumn tabeli.

Parametr *aFirstHeader* określa tekst w pierwszej kolumnie wiersza nagłówka. Parametr *aNextFormat* określa sposób formatowania wartości w kolumnach linii nagłówka. W kolumnach linii nagłówka tabeli poziomej wyświetlane są czasy przyporządkowane do poszczególnych kolumn. Na przykład, format *hh:mm* powoduje wyświetlenie czasu w formacie godziny:minuty (np. 10:45).

Parametry *aFirstStyle* i *aNextStyle* określają nazwy stylów stosowanych dla pierwszej i wszystkich kolejnych kolumn nagłówka. Jeżeli parametry nie zostaną podane, to stosowany jest standardowy styl nagłówka.

Metoda zwraca obiekt utworzonej tabeli.

```
ReportTable AddHorizontalTable(object aData, int aFirstWidth, int aNextWidth);
```

Function **AddHorizontalTable**(ByVal aData As **Object**, ByVal aFirstWidth As **Integer**, ByVal aNextWidth As **Integer**)

Metoda dodaje do raportu tabelę w układzie poziomym, w którym serie danych pokazane są w pojedynczych wierszach. Dane serii przekazane są poprzez parametr *aData*. Obiekt danych serii musi być typu *System.Collections.ICollection* lub *System.Data.DataTable*.

Parametry *aFirstWidth* i *aNextWidth* określają szerokość kolumn (w milimetrach), odpowiednio: pierwszej i wszystkich kolejnych kolumn tabeli.

Metoda zwraca obiekt utworzonej tabeli.

AddImage

```
ReportImage  AddImage(string  aImageName, MigraDoc.DocumentObjectModel.WrapStyle
aWrap, MigraDoc.DocumentObjectModel.ShapePosition aLeft,
MigraDoc.DocumentObjectModel.ShapePosition aTop);
ReportImage  AddImage(string  aImageName, MigraDoc.DocumentObjectModel.WrapStyle
aWrap, MigraDoc.DocumentObjectModel.ShapePosition aLeft);
ReportImage  AddImage(string  aImageName, MigraDoc.DocumentObjectModel.WrapStyle
aWrap, int  aLeft, int  aTop);

Function AddImage(ByVal aImageName As  String, aWrap As
MigraDoc.DocumentObjectModel.WrapStyle , aLeft As
MigraDoc.DocumentObjectModel.ShapePosition, aTop As
MigraDoc.DocumentObjectModel.ShapePosition)  As ReportImage
Function AddImage(ByVal aImageName As  String, aWrap As
MigraDoc.DocumentObjectModel.WrapStyle, aLeft As
MigraDoc.DocumentObjectModel.ShapePosition)  As ReportImage
Function AddImage(ByVal aImageName As  String, aWrap As
MigraDoc.DocumentObjectModel.WrapStyle, aLeft As Integer, aTop As Integer)  As
ReportImage
```

Metoda wstawia obrazek do treści raportu. Parametr *aImageName* określa nazwę pliku graficznego obrazka. Zalecane jest podawanie tylko nazwy pliku, bez podawania pełnej ścieżki dostępu. W takim przypadku zakłada się, że plik obrazka znajduje się w podkatalogu *Addons* katalogu definicyjnego aplikacji.

Parametr *aWrap* określa, w jaki sposób obrazek powinien być umieszczony między innymi elementami. Parametry *aLeft* i *aTop* określają położenie obrazka, podawane jako stałe typu *ShapePosition* lub bezwzględnie w milimetrach.

Metoda zwraca obiekt wstawionego obrazka.

AddPageBreak

```
void AddPageBreak();
Sub AddPageBreak( )
```

Metoda służy do przejścia do kolejnej strony raportu.

AddParagraph

```
ReportParagraph AddParagraph(string  aContent);
ReportParagraph AddParagraph(string  aContent, string  aStyle);
Function AddParagraph(ByVal aContent As  String) As ReportParagraph
Function AddParagraph(ByVal aContent As  String, ByVal aStyle As  String) As
ReportParagraph
```

Metoda wstawia paragraf tekstowy do treści raportu. Parametr aContent określa treść paragrafu, a parametr aStyle nazwę stylu użytego w paragrafie. Jeżeli styl nie jest podany, to stosowany jest styl domyślny (*Normal*).

Metoda zwraca obiekt utworzonego paragrafu. Korzystając z tego obiektu, można do paragrafu dodać dodatkowy tekst lub pola specjalne.

AddVerticalTable

```
ReportTable  AddVerticalTable(bool aShowHeader);
Function AddVerticalTable(ByVal aShowHeader As  Boolean) As ReportTable
```

Metoda dodaje do raportu tabelę w układzie pionowym, w którym serie danych pokazane są w kolumnach. Dane serii będą pochodziły z zapytania o dane archiwalne, którego parametry należy ustawić przy pomocy wywołania metody *SetQueryData* obiektu tabeli.

Parametr *aShowHeader* określa, czy tabela będzie posiadała wiersz nagłówka.

Metoda zwraca obiekt utworzonej tabeli.

```
ReportTable  AddVerticalTable(object aData, bool aShowHeader);
Function AddVerticalTable(ByVal aData As  Object, ByVal aShowHeader As  Boolean) As
ReportTable
```

Metoda dodaje do raportu tabelę w układzie pionowym, w którym serie danych pokazane są w kolumnach. Dane serii przekazane są poprzez parametr *aData*. Obiekt danych serii musi być typu *System.Collections.ICollection* lub *System.Data.DataTable*.

Parametr *aShowHeader* określa, czy tabela będzie posiadała wiersz nagłówka.

Metoda zwraca obiekt utworzonej tabeli.

Render

```
void Render(string  aOutputFileName)
```

```
Sub Render(ByVal aOutputFileName As  String)
```

Metoda służy do wygenerowania treści raportu i zapisania go do pliku o nazwie określonej przez parametr *aOutputFileName*. Raport jest tworzony w formacie PDF.

SetPageSetup

```
void SetPageSetup(int aLeftMargin, int aRightMargin, int aTopMargin, int aBottomMargin, int
aHeaderDistance, int aFooterDistance);
```

```
Sub SetPageSetup(ByVal aLeftMargin As  Integer, ByVal aRightMargin As  Integer, ByVal
aTopMargin As  Integer, ByVal aBottomMargin As  Integer, ByVal aHeaderDistance
As  Integer, ByVal aFooterDistance As  Integer )
```

Metoda ustawia parametry strony raportu poprzez określenie wielkości marginesów, oraz odległości nagłówka i stopki od krawędzi strony. Wszystkie argumenty podawane są w milimetrach.

void SetPageSetup(bool aLandscapeOrientation,
MigraDoc.DocumentObjectModel.PageFormat aFormat);

Sub **SetPageSetup**(ByVal aLandscapeOrientation As **Boolean**, ByVal aFormat As **MigraDoc.DocumentObjectModel.PageFormat**)

Metoda ustawia parametry strony raportu poprzez określenie orientacji oraz formatu strony. Wartość *true* parametru *aLandscapeOrientation* oznacza poziome ułożenie strony.

Przykład:

```
report.SetPageSetup(true, PageFormat.A4)
```

Powyższe wywołanie wybiera stronę w formacie A4 i orientacji poziomej.

SetStyle

ReportStyle SetStyle(string aName, string aFontName, double aFontSize,
MigraDoc.DocumentObjectModel.Color aFontColor, boolaBold, boolaItalic,
MigraDoc.DocumentObjectModel.ParagraphAlignment aAlignment,
MigraDoc.DocumentObjectModel.Color aShaderColor);

Function **SetStyle**(ByVal aName As **String,** ByVal aFontName As **String**, ByVal aFontSize As **Double**, ByVal aFontColor As **MigraDoc.DocumentObjectModel.Color**, ByVal aBold As Boolean, ByVal aIatlic As **Boolean**, ByVal aAlignment As **MigraDoc.DocumentObjectModel.ParagraphAlignment**, ByVal aShaderColor As **MigraDoc.DocumentObjectModel.Color**) As **ReportStyle**

Metoda służy do zdefiniowania stylu, który może być później użyty przy tworzeniu elementów raportu.

Parametr *aName* określa nazwę stylu. Można używać nazw predefiniowanych, zdefiniowanych w klasie *ReportStyleStandardNames* lub dowolnych własnych nazw. Style o nazwach predefiniowanych mają z góry określone znaczenie i po ich utworzeniu są automatycznie używane przy konstruowaniu odpowiednich elementów raportu.

Parametr *aFontName* określa nazwę stylu czcionki, *aFontSize* rozmiar czcionki w punktach, *aFontColor* kolor czcionki, *aBold* pozwala wybrać czcionkę pogrubioną, a *aItalic* wybiera czcionkę pochyloną. Określając kolor czcionki można stosować predefiniowane kolory zdefiniowane w klasie *MigraDoc.DocumentObjectModel.Colors* (np. *Colors.Orange*).

Parametr *aAlignment* określa sposób pozycjonowania tekstu w ramach elementu, w którym tekst został umieszczony (np. wewnątrz paragrafu).

Parametr *aShaderColor* określa kolor tła elementu, dla którego styl zostanie zastosowany. Dla tła przeźroczystego (pustego) należy podać stałą *Color.Empty*.

Metoda zwraca obiekt utworzonego stylu.

5.3 Klasa ReportHeader

Klasa reprezentuje nagłówek raportu. Obiekt tej klasy jest automatycznie tworzony w momencie tworzenia obiektu klasy *Report*. Dostęp do obiektu jest możliwy poprzez właściwość *Header* klasy *Report*.

Większość metod i właściwości klasy nagłówka jest dziedziczona z klasy *ReportHeaderFooter*.

5.3.1 Właściwości

MigraHeader

MigraDoc.DocumentObjectModel.HeaderFooter `MigraHeader { get; }`

ReadOnly Property **MigraHeader** As **MigraDoc.DocumentObjectModel.HeaderFooter**

Właściwość zwraca bazowy obiekt nagłówka raportu biblioteki MigraDoc.

5.4 Klasa ReportFooter

Klasa reprezentuje stopkę raportu. Obiekt tej klasy jest automatycznie tworzony w momencie tworzenia obiektu klasy *Report*. Dostęp do obiektu jest możliwy poprzez właściwość *Footer* klasy *Report*.

Większość metod i właściwości klasy stopki jest dziedziczona z klasy ReportHeaderFooter.

5.4.1 Właściwości

MigraFooter

MigraDoc.DocumentObjectModel.HeaderFooter `MigraFooter { get; }`

ReadOnly Property **MigraFooter** As **MigraDoc.DocumentObjectModel.HeaderFooter**

Właściwość zwraca bazowy obiekt stopki raportu biblioteki MigraDoc.

5.5 Klasa ReportHeaderFooter

5.5.1 Metody

AddImage

```
ReportImage AddImage(string aImageName, MigraDoc.DocumentObjectModel.WrapStyle
aWrap, MigraDoc.DocumentObjectModel.ShapePosition aLeft,
MigraDoc.DocumentObjectModel.ShapePosition aTop);

ReportImage AddImage(string aImageName, MigraDoc.DocumentObjectModel.WrapStyle
aWrap, MigraDoc.DocumentObjectModel.ShapePosition aLeft);

ReportImage AddImage(string aImageName, MigraDoc.DocumentObjectModel.WrapStyle
aWrap, int aLeft, int aTop);
```

Function **AddImage**(ByVal aImageName As **String**, aWrap As **MigraDoc.DocumentObjectModel.WrapStyle** , aLeft As **MigraDoc.DocumentObjectModel.ShapePosition**, aTop As **MigraDoc.DocumentObjectModel.ShapePosition**) As **ReportImage**

Function **AddImage**(ByVal aImageName As **String**, aWrap As **MigraDoc.DocumentObjectModel.WrapStyle**, aLeft As **MigraDoc.DocumentObjectModel.ShapePosition**) As **ReportImage**

Function **AddImage**(ByVal aImageName As **String**, aWrap As **MigraDoc.DocumentObjectModel.WrapStyle**, aLeft As **Integer**, aTop As **Integer**) As **ReportImage**

Metoda wstawia obrazek do nagłówka lub stopki raportu. Parametr *aImageName* określa nazwę pliku graficznego obrazka. Zalecane jest podawanie tylko nazwy pliku, bez podawania pełnej ścieżki dostępu. W takim przypadku zakłada się, że plik obrazka znajduje się w podkatalogu *Addons* katalogu definicyjnego aplikacji.

Parametr *aWrap* określa, w jaki sposób obrazek powinien być umieszczony między innymi elementami. Parametry *aLeft* i *aTop* określają położenie obrazka, podawane jako stałe typu *ShapePosition* lub bezwzględnie w milimetrach.

Metoda zwraca obiekt wstawionego obrazka.

5.6 Klasa ReportParagraph

Klasa reprezentuje paragraf tekstowy umieszczony w treści raportu, nagłówka lub stopki. Obiekty klasy *ReportParagraph* tworzone są przy pomocy metody *AddParagraph*, udostępnianej przez klasy *Report* i *ReportHeaderFooter*.

5.6.1 Właściwości

MigraParagraph

MigraDoc.DocumentObjectModel.Paragraph `MigraParagraph { get; }`

ReadOnly Property **MigraParagraph** As **MigraDoc.DocumentObjectModel.Paragraph**

Właściwość zwraca bazowy obiekt paragrafu tekstowego biblioteki MigraDoc.

5.6.2 Metody

AddNumPagesField

ReportParagraph AddNumPagesField();
Function **AddNumPagesField**() As **ReportParagraph**

Metoda dodaje do paragrafu pole specjalne oznaczające sumaryczną liczbę stron raportu. Typowo metoda jest stosowana dla paragrafów wstawionych do nagłówka lub stopki raportu.

Metoda zwraca obiekt paragrafu.

Przykład:

report.Footer.AddParagraph(”Strona ”).AddPageField().AddText(” z ”).AddNumPagesField()

Powyższe wywołanie wstawia do stopki raportu tekst zawierający bieżący numer strony i liczbę stron raportu. Sekwencyjne wywołanie kolejnych metod klasy *ReportParagraph* jest możliwe, ponieważ każda z tych metod zwraca własny obiekt paragrafu.

AddPageField

ReportParagraph AddPageField();
Function **AddPageField**() As **ReportParagraph**

Metoda dodaje do paragrafu pole specjalne oznaczające bieżący numer strony raportu. Typowo metoda jest stosowana dla paragrafów wstawionych do nagłówka lub stopki raportu.

Metoda zwraca obiekt paragrafu.

AddText

```
ReportParagraph AddText(string aText);
Function AddText(ByVal aText As String) As ReportParagraph
```

Metoda dodaje do paragrafu tekst określony parametrem *aText*. Tekst jest dopisywany do tekstu wcześniej wstawionego do paragrafu (zazwyczaj w momencie tworzenia paragrafu).

Metoda zwraca obiekt paragrafu.

5.7 Klasa ReportImage

Klasa reprezentuje obrazek umieszczony w treści raportu, nagłówka lub stopki. Obiekty klasy *ReportImage* tworzone są przy pomocy metody *AddImage* udostępnianej przez klasy *Report* i *ReportHeaderFooter*.

5.7.1 Właściwości

MigraImage

```
MigraDoc.DocumentObjectModel.Shapes.Image MigraImage { get; }
ReadOnly Property MigraImage As MigraDoc.DocumentObjectModel.Shapes.Image
```

Właściwość zwraca bazowy obiekt obrazka biblioteki MigraDoc.

5.8 Klasa ReportStyleStandardNames

5.8.1 Właściwości

Footer

```
string Footer { get; }
```

ReadOnly Property **Footer** As **String**

Styl używany dla stopki stron raportu.

Header

```
string Header { get; }
```

ReadOnly Property **Header** As **String**

Styl używany dla nagłówka stron raportu.

Normal

```
string Normal { get; }
```

ReadOnly Property **Normal** As **String**

Styl domyślny wszystkich elementów raportu. Jest używany, gdy żaden inny styl dla elementu nie został zdefiniowany.

TableAltRow

```
string TableAltRow { get; }
```

ReadOnly Property **TableAltRow** As **String**

Styl używany w komórkach parzystych wierszy tabeli.

TableErrorCell

```
string TableErrorCell { get; }
```

ReadOnly Property **TableErrorCell** As **String**

Styl używany dla komórek tabeli danych archiwalnych zawierających niepoprawną wartość.

TableHeader

```
string TableHeader { get; }
```

ReadOnly Property **TableHeader** As **String**

Styl używany dla komórek nagłówka tabeli.

TableRow

```
string TableRow { get; }
```

ReadOnly Property **TableRow** As **String**

Styl używany w komórkach wierszy tabeli. Jeżeli styl *TableAltRow* jest zdefiniowany, to styl *TableRow* dotyczy tylko wierszy nieparzystych.

5.9 Klasa ReportStyle

Klasa definicji stylu wyświetlania elementów raportu. Obiekty klasy *ReportStyle* tworzone są przy pomocy metody *SetStyle* klasy *Report*.

Użycie stylów o nazwach standardowych (zdefiniowanych w klasie *ReportStyleStandardNames*) jest w pełni automatyczne. Użycie pozostałych stylów wymaga podania nazwy stylu w metodach tworzących elementy raportu.

5.9.1 Właściwości

Alignment

```
MigraDoc.DocumentObjectModel.ParagraphAlignment Alignment { get; set;}
```

Property **Alignment** As **MigraDoc.DocumentObjectModel.ParagraphAlignment**

Właściwość określa sposób pozycjonowania tekstu w ramach elementu, w którym tekst został umieszczony (np. wewnątrz paragrafu).

FontBold

```
bool FontBold { get; set;}
```

Property **FontBold** As **Boolean**

Właściwość określa, czy czcionka jest pogrubiona.

FontColor

```
MigraDoc.DocumentObjectModel.Color FontColor { get; set;}
```

Property **FontColor** As **MigraDoc.DocumentObjectModel.Color**

Właściwość określa kolor czcionki. Określając kolor czcionki, można stosować predefiniowane kolory zdefiniowane w klasie *MigraDoc.DocumentObjectModel.Colors* (np. *Colors.Orange*).

FontItalic

```
bool FontItalic { get; set;}
```

Property **FontItalic** As **Boolean**

Właściwość określa, czy czcionka jest pochyła.

FontName

```
string FontName { get; set;}
```

Property **FontName** As **String**

Właściwość określa nazwę stylu czcionki.

FontSize

```
double FontSize { get; set;}
```

Property **FontName** As **Double**

Właściwość określa rozmiar czcionki wyrażony w punktach.

MigraStyle

```
MigraDoc.DocumentObjectModel.Style MigraStyle { get; }
```

ReadOnly Property **MigraStyle** As **MigraDoc.DocumentObjectModel.Style**

Właściwość zwraca bazowy obiekt stylu biblioteki MigraDoc.

Name

```
string Name { get; }
```

ReadOnly Property **Name** As **String**

Właściwość przechowuje nazwę stylu.

ShaderColor

```
MigraDoc.DocumentObjectModel.Color ShaderColor { get; set;}
```

Property **ShaderColor** As **MigraDoc.DocumentObjectModel.Color**

Właściwość określa kolor tła elementu, dla którego styl zostanie zastosowany. Określając kolor można stosować predefiniowane kolory zdefiniowane w klasie *MigraDoc.DocumentObjectModel.Colors* (np. *Colors.Orange*).

5.9.2 Metody

AddTabStops

ReportStyleAddTabStops(params int[ ] aPositions);

Function **AddTabStops**(ByVal ParamArray aPositions( ) As **Object**) As **ReportStyle**

Metoda dodaje do stylu położenie kolejnych znaków tabulacji. Położenie określone jest w milimetrach.

Przykład:

```
report.SetStyle(”szpalty”, ”Arial”, 10, Colors.Black, false, false, ParagraphAlignment.Left,
Color.Empty).AddTabStops(50,100);
report.AddParagraph("1\tfirst\t11111”, "szpalty”);
report.AddParagraph("2\tsecond\t22222”, "szpalty”);
```

Powyższy kod powoduje wyświetlenie tekstu w trzech kolumnach.

AddTabStop

ReportStyleAddTabStop(int aPosition, MigraDoc.DocumentObjectModel.TabAlignment aAlignmnet);

Function **AddTabStop**(ByVal aPosition As **Integer**, ByVal aAlignment As **MigraDoc.DocumentObjectModel.TabAlignment**) As **ReportStyle**

Metoda dodaje do stylu położenie kolejnego, pojedynczego znaku tabulacji wraz ze sposobem wyrównania tekstu w punkcie tabulacji. Położenie określa parametr *aPosition*, podawany w milimetrach. Sposób wyrównania określa parametr *aAlignment*.

5.10 Klasa ReportTable

Klasa reprezentuje obiekt tabeli wstawionej do treści raportu. Obiekty klasy *ReportTable* tworzone są przy pomocy metod *AddVerticalTable* i *AddHorizontalTable* klasy *Report*. Po utworzeniu tabela wymaga jeszcze dodania do niej zestawu kolumn (dla tabel pionowych - metoda *AddColumn*) lub wierszy (dla tabel poziomych - metoda *AddRow*). W przypadku tabel tworzonych na podstawie wartości archiwalnych zmiennych procesowych konieczne jest również określenie parametrów odczytu danych archiwalnych przy pomocy metody *SetQueryData*.

5.10.1 Właściwości

EndTimeColumn

```
string EndTimeColumn { get; }
```

ReadOnly Property **EndTimeColumn** As **String**

Nazwa kolumny specjalnej tabeli pionowej danych archiwalnych oznaczająca użycie w kolumnie czasu końcowego okresu agregacji wartości zmiennej. Właściwość należy użyć w metodzie *AddColumn* w miejscu parametru *aVarName*.

FirstStyle

```
string FirstStyle { get; }
```

ReadOnly Property **FirstStyle** As **String**

Właściwość przechowuje nazwę stylu dla pierwszej kolumny nagłówka tabeli poziomej. Styl ustawiany jest w momencie tworzenia tabeli metodą *AddHorizontalTable* klasy *Report*.

MigraTable

```
MigraDoc.DocumentObjectModel.Table MigraTable { get; }
```

ReadOnly Property **MigraTable** As **MigraDoc.DocumentObjectModel.Table**

Właściwość zwraca bazowy obiekt tabeli biblioteki *MigraDoc*.

NextStyle

```
string NextStyle { get; }
```

ReadOnly Property **NextStyle** As **String**

Właściwość przechowuje nazwę stylu dla drugiej i następnych kolumn nagłówka tabeli poziomej. Styl ustawiany jest w momencie tworzenia tabeli metodą *AddHorizontalTable* klasy *Report*.

StartTimeColumn

```
string StartTimeColumn { get; }
```

ReadOnly Property **StartTimeColumn** As **String**

Nazwa kolumny specjalnej tabeli pionowej danych archiwalnych, oznaczająca użycie w kolumnie czasu początkowego okresu agregacji wartości zmiennej. Właściwość należy użyć w metodzie *AddColumn* w miejscu parametru *aVarName*.

5.10.2 Metody

AddCalculatedColumn

ReportTableCalcColumn AddCalculatedColumn(string aHeaderText, CalculationType aCalcType, int aWidth, string aFormat = null, CellCallback aCallback = null, string aHeaderStyle = null, string aColumnStyle = null, string aColumnAltStyle = null, string aErrorStyle = null);

Function **AddCalculatedColumn**(ByVal aHeaderText As **String**, , ByVal aCalclType As **CalculationType**, ByVal aWidth As **Integer**, ByVal aFormat As **String** = Nothing, ByVal aCallback As **CellCalback** = Nothing, ByVal aHeaderStyle As **String** = Nothing, ByVal aColumnStyle As **String** = Nothing, ByVal aColumnAltStyle As **String** = Nothing, ByVal aErrorStyle As **String** = Nothing) As **ReportTableCalcColumn**

Metoda służy do dodania wyliczanej kolumny do tabeli raportowej. Wyliczenie odbywa się na podstawie wartości komórek poszczególnych wierszy tabeli.

Parametr *aHeaderText* określa tekst wyświetlany w linii nagłówka tabeli.

Parametr *aCalcType* określa sposób wyliczenia zawartości komórek kolumny. Możliwe są następujące wartości:

CalculationType.Custom – wyliczenie niestandardowe wykonywane w funkcji zwrotnej a*Callback*
CalculationType.Sum – suma wartości komórek wiersza
CalculationType.Min – minimalna wartość komórek wiersza
CalculationType.Max – maksymalna wartość komórek wiersza
CalculationType.Avg – średnia wartość komórek wiersza

Parametr *aWidth* określa szerokość kolumny w milimetrach, a parametr *aFormat* sposób formatowania wartości wyświetlanych w komórkach kolumny.

Parametr *aCallback* pozwala na zadeklarowanie funkcji zwrotnej, która będzie wykonywana dla każdej komórki kolumny w momencie generowania jej zawartości. Mechanizm ten pozwala na implementację własnej metody wyliczania wartości, kontroli i dodatkowego formatowania wartości wynikowej. Funkcja zwrotna może zmienić zarówno treść komórki jak i styl generacji. Funkcja zwrotna musi być zgodna z poniższym delegatem:

```
delegate bool CellCallback(ReportTableCell aCell);
```

Delegate Sub **CellCallback(**ByVal aCell As **IReportTableCell)**

Parametr *aCell* służy do identyfikacji komórki. Pozwala on też na dostęp do proponowanej zawartości komórki i jej ewentualnej modyfikacji. Poprzez parametr *aCall* można także dostać się do wartości wszystkich komórek wiersza, którego dane wywołanie funkcji zwrotnej dotyczy. Na podstawie tych wartości można zaimplementować własny sposób wyliczania wartości agregowanej wiersza. Funkcja zwrotna zwraca wartość *true*, gdy utworzyła samodzielnie

zawartość komórki. Wartość powrotna *false* oznacza, że należy wygenerować zawartość komórki na podstawie danych zawartych w obiekcie komórki *aCell*.

Opcjonalne parametry *aHeaderStyle*, *aColumnStyle*, *aColumnAltStyle*, *aErrorStyle* pozwalają na indywidualne określenie nazwy stylów dla: komórki nagłówka, komórek wartości, komórek wartości w wierszach parzystych, komórek wartości z niepoprawnym statusem danych.

AddCalculatedRow

```
ReportTableCalcRow AddCalculatedRow(string aFirstColumnText, CalculationType
aCalcType, string aFormat = null, CellCallback aCallback = null, string aFirstStyle = null,
string aNextStyle = null, string aErrorStyle = null);
```

```
Function AddCalculatedRow(ByVal aFirstColumnText As String, , ByVal aCalclType As
CalculationType,, ByVal aFormat As String = Nothing, ByVal aCallback As CellCalback =
Nothing, ByVal aFirstStyle As String = Nothing, ByVal aNextStyle As String = Nothing, ByVal
aErrorStyle As String = Nothing) As ReportTableCalcRow
```

Metoda służy do dodania wyliczanego wiersza do tabeli raportowej. Wyliczenie odbywa się na podstawie wartości komórek poszczególnych kolumn tabeli.

Parametr *aFirstColumnText* określa tekst wyświetlany w pierwszej kolumnie wiersza.

Parametr *aCalcType* określa sposób wyliczenia zawartości komórek wiersza. Możliwe są następujące wartości:

CalculationType.Custom – wyliczenie niestandardowe wykonywane w funkcji zwrotnej a*Callback*
CalculationType.Sum – suma wartości komórek kolumny
CalculationType.Min – minimalna wartość komórek kolumny
CalculationType.Max – maksymalna wartość komórek kolumny
CalculationType.Avg – średnia wartość komórek kolumny

Parametr parametr *aFormat* określa sposób formatowania wartości wyświetlanych w komórkach wiersza.

Parametr *aCallback* pozwala na zadeklarowanie funkcji zwrotnej, która będzie wykonywana dla każdej komórki wiersza w momencie generowania jej zawartości. Mechanizm ten pozwala na implementację własnej metody wyliczania wartości oraz kontroli i dodatkowego formatowania wartości wynikowej. Funkcja zwrotna może zmienić zarówno treść komórki jak i styl generacji. Funkcja zwrotna musi być zgodna z poniższym delegatem:

```
delegate bool CellCallback(ReportTableCell aCell);
```

```
Delegate Sub CellCallback(ByVal aCell As IReportTableCell)
```

Parametr *aCell* służy do identyfikacji komórki. Pozwala on też na dostęp do proponowanej zawartości komórki i jej ewentualnej modyfikacji. Poprzez parametr *aCall* można także dostać się do wartości wszystkich komórek kolumny, którego dane wywołanie funkcji zwrotnej dotyczy. Na podstawie tych wartości można zaimplementować własny sposób wyliczania wartości agregowanej wiersza. Funkcja zwrotna zwraca wartość *true*, gdy utworzyła samodzielnie zawartość komórki. Wartość powrotna *false* oznacza, że należy wygenerować zawartość komórki na podstawie danych zawartych w obiekcie komórki *aCell*.

Opcjonalne parametry *aFirstStyle*, *aNextStyle*, *aErrorStyle* pozwalają na indywidualne określenie nazwy stylów dla: pierwszej komórki wiersza(opisu wiersza), komórek wartości, komórek wartości z niepoprawnym statusem.

AddColumn

```
ReportTableColumn AddColumn(string aHeaderText, string aVarName,
string aAggrName, int aWidth, string aFormat = null,
ReportTableCellErrorBehavior aErrorMode =
ReportTableCellErrorBehavior.Normal, CellCallback aCallback = null,
string aHeaderStyle = null, string aColumnStyle = null, string aColumnAltStyle = null,
string aErrorStyle = null);
```

```
Function AddColumn(ByVal aHeaderText As String, ByVal aVarName As String, ByVal
aAggrName As String, ByVal aWidth As Integer, ByVal aFormat As String = Nothing,
ByVal aErrorMode As ReportTableCellErrorBehavior =
ReportTableCellErrorBehavior.Normal, ByVal aCallback As CellCalback = Nothing, ByVal
aHeaderStyle As String = Nothing, ByVal aColumnStyle As String = Nothing, ByVal
aColumnAltStyle As String = Nothing, ByVal aErrorStyle As String = Nothing) As
ReportTableColumn
```

Metoda służy do dodania kolumny do tabeli pionowej tworzonej na podstawie danych archiwalnych zmiennych procesowych.

Parametr *aHeaderText* określa tekst wyświetlany w linii nagłówka tabeli.

Parametry *aVarName* i *aAggrName* określają nazwę zmiennej procesowej oraz nazwę wyliczanego agregatu. W miejscu atrybutu *aVarName* można użyć dwóch specjalnych nazw: *ReportTable.StartTimeColumn* lub *ReportTableEndTimeColumn*. Pozwala to na zdefiniowanie kolumn, w których pokazywane będą czasy początków i końców poszczególnych okresów wyliczania wartości agregatów.

Parametr *aWidth* określa szerokość kolumny w milimetrach, a parametr *aFormat* sposób formatowania wartości wyświetlanych w komórkach kolumny. Jeżeli parametr *aFormat* nie zostanie podany, to zostanie użyty atrybut formatu wartości pobrany z bazy definicji zmiennych.

Parametr *aErrorMode* określa sposób tworzenia zawartości komórek wartości, dla których brak danych lub ich status jest niepoprawny. Możliwe są następujące wartości:

ReportTableCellErrorBehavior.Empty - pusta zawartość komórki
ReportTableCellErrorBehavior.Normal - ignoruj wszystkie błędy
ReportTableCellErrorBehavior.Normal_EmptyIfNoData - ignoruj błędy statusu, pusta zawartość gdy brak danych
ReportTableCellErrorBehavior.AddQuestion - dodaj znak zapytania
ReportTableCellErrorBehavior.AddQuestion_EmptyIfNoData - dodaj znak zapytania przy błędach statusu, pusta zawartość gdy brak danych

Parametr *aCallback* pozwala na zadeklarowanie funkcji zwrotnej, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Mechanizm ten pozwala na implementację własnej metody reakcji na błędy lub kontroli wartości zmiennych. Funkcja zwrotna może zmienić zarówno treść komórki jak i styl generacji. Funkcja zwrotna musi być zgodna z poniższym delegatem:

```
delegate bool CellCallback(ReportTableCell aCell);
```

Delegate Sub **CellCallback(**ByVal aCell As **IReportTableCell)**

Parametr *aCell* służy do identyfikacji komórki. Pozwala on też na dostęp do proponowanej zawartości komórki i jej ewentualnej modyfikacji. Funkcja zwrotna zwraca wartość *true*, gdy utworzyła samodzielnie zawartość komórki. Wartość powrotna *false* oznacza, że należy wygenerować zawartość komórki na podstawie danych zawartych w obiekcie komórki *aCell*.

Opcjonalne parametry *aHeaderStyle*, *aColumnStyle*, *aColumnAltStyle*, *aErrorStyle* pozwalają na indywidualne określenie nazwy stylów dla: komórki nagłówka, komórek wartości, komórek wartości w wierszach parzystych, komórek wartości z niepoprawnym statusem danych.

```
ReportTableColumn AddColumn(string aHeaderText, string aFieldName, int aWidth,
string aFormat = null, CellCallback aCallback = null, string aHeaderStyle = null,
string aColumnStyle = null, string aColumnAltStyle = null);
```

Function **AddColumn**(ByVal aHeaderText As **String**, ByVal aFieldName As **String**, ByVal aWidth As **Integer**, ByVal aFormat As **String** = Nothing, ByVal aCallback As **CellCalback** = Nothing, ByVal aHeaderStyle As **String** = Nothing, ByVal aColumnStyle As **String** = Nothing, ByVal aColumnAltStyle As **String** = Nothing) As **ReportTableColumn**

Metoda służy do dodania kolumny do tabeli pionowej tworzonej na podstawie danych przekazanych w momencie tworzenia tabeli (poprzez obiekt typu *DataTable* lub *ICollection*).

Parametr *aHeaderText* określa tekst wyświetlany w linii nagłówka tabeli.

Parametry *aFieldName* określa nazwę elementu używanego przez kolumnę. Jest to nazwa kolumny w obiekcie typy *DataTable* lub nazwa właściwości obiektów przekazanych w kolekcji *ICollection*.

Parametr *aWidth* określa szerokość kolumny w milimetrach, a parametr *aFormat* sposób formatowania wartości wyświetlanych w komórkach kolumny.

Parametr *aCallback* pozwala na zadeklarowanie funkcji zwrotnej, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Mechanizm ten pozwala na implementację własnej metody reakcji na błędy lub kontroli wartości zmiennych.

Opcjonalne parametry *aHeaderStyle*, *aColumnStyle*, *aColumnAltStyle* pozwalają na indywidualne określenie nazwy stylów dla: komórki nagłówka, komórek wartości, komórek wartości w wierszach parzystych.

AddRow

```
ReportTableColumn AddRow(string aFirstColumnText, string aVarName,
string aAggrName, string aFormat = null, ReportTableCellErrorBehavior
aErrorMode = ReportTableCellErrorBehavior.Normal, CellCallback aCallback =
null, string aFirstStyle = null, string aNextStyle = null, string aErrorStyle = null);
```

```
Function AddRow(ByVal aFirstColumnText As String, ByVal aVarName As String, ByVal
aAggrName As String, ByVal aFormat As String = Nothing, ByVal aErrorMode As
ReportTableCellErrorBehavior = ReportTableCellErrorBehavior.Normal, ByVal aCallback As
CellCalback = Nothing, ByVal aFirstStyle As String = Nothing, ByVal aNextStyle As String
= Nothing, ByVal aErrorStyle As String = Nothing) As ReportTableColumn
```

Metoda służy do dodania wiersza do tabeli poziomej tworzonej na podstawie danych archiwalnych zmiennych procesowych.
Parametr *aFirstColumnText* określa tekst wyświetlany w pierwszej kolumnie wiersza.

Parametry *aVarName* i *aAggrName* określają nazwę zmiennej procesowej oraz nazwę wyliczanego agregatu.

Parametr *aFormat* określa sposób formatowania wartości wyświetlanych w komórkach wiersza. Jeżeli parametr *aFormat* nie zostanie podany, to zostanie użyty atrybut formatu wartości pobrany z bazy definicji zmiennych.

Parametr *aErrorMode* określa sposób tworzenia zawartości komórek wartości, dla których brak danych lub ich status jest niepoprawny. Szczegóły użycia znajdują się w opisie metody *AddColumn*.

Parametr *aCallback* pozwala na zadeklarowanie funkcji zwrotnej, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Szczegóły użycia znajdują się w opisie metody *AddColumn*.

Opcjonalne parametry *aFirstStyle*, *aNextStyle*, *aErrorStyle* pozwalają na indywidualne określenie nazwy stylów dla: komórek w pierwszej kolumnie wiersza, komórek w drugiej i następnych kolumnach wiersza, komórek wartości z niepoprawnym statusem danych.

```
ReportTableColumn AddRow(string aFirstColumnText, string aFieldName,
string aFormat = null, CellCallback aCallback = null, string aFirstStyle = null,
string aNextStyle = null);
```

```
Function AddRow(ByVal aFirstColumnText As String, ByVal aFieldName As String, ByVal
aFormat As String = Nothing, ByVal aCallback As CellCalback = Nothing, ByVal aFirstStyle
As String = Nothing, ByVal aNextStyle As String = Nothing) As ReportTableColumn
```

Metoda służy do dodania wiersza do tabeli poziomej tworzonej na podstawie danych przekazanych w momencie tworzenia tabeli (poprzez obiekt typu *DataTable* lub *ICollection*).

Metoda służy do dodania wiersza do tabeli poziomej tworzonej na podstawie danych archiwalnych zmiennych procesowych.

Parametr *aFirstColumnText* określa tekst wyświetlany w pierwszej kolumnie wiersza.

Parametry *aVarName* i *aAggrName* określają nazwę zmiennej procesowej oraz nazwę wyliczanego agregatu.

Parametry *aFieldName* określa nazwę elementu używanego w wierszu. Jest to nazwa kolumny w obiekcie typy *DataTable* lub nazwa właściwości obiektów przekazanych w kolekcji *ICollection*.

Parametr *aFormat* określa sposób formatowania wartości wyświetlanych w komórkach wiersza.

Parametr *aCallback* pozwala na zadeklarowanie funkcji zwrotnej, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Szczegóły użycia znajdują się w opisie metody *AddColumn*.

Opcjonalne parametry *aFirstStyle*, *aNextStyle* pozwalają na indywidualne określenie nazwy stylów dla: komórek w pierwszej kolumnie wiersza, komórek w drugiej i następnych kolumnach wiersza.

GenerateContent

```
void GenerateContent ();
```

```
Sub GenerateContent ()
```

Metoda służy do wygenerowanie kompletnej zawartości tabeli na podstawie dostępnych danych. Użycie metody nie jest obowiązkowe. Metoda *Render* klasy *Report* automatycznie wywołuje metodę *GenerateContent* dla wszystkich tabel, dla których metoda nie była jeszcze wykonana.

LeftIndent

```
void LeftIndent(int aIndent);
```

```
Sub LeftIndent(ByVal aIndent As Integer)
```

Metoda pozwala przesunąć położenie całej tabeli w prawo o liczbę milimetrów podaną w parametrze *aIndent*.

SetEdgeStyle

```
void SetEdgeStyle(ReportTableEdgeType aEdgeType, MigraDoc.DocumentObjectModel.Color
aColor);
```

```
Sub SetEdgeStyle(ByVal aEdgeType As ReportTableEdgeType, ByVal aColor As
MigraDoc.DocumentObjectModel.Color)
```

Metoda służy do ustawienia sposobu rysowania krawędzi komórek tabeli. Parametr *aEdgeType* określa, które krawędzie należy rysować. Możliwe są poniższe wartości:

ReportTableEdgeStyle.None - brak krawędzi
ReportTableEdgeStyle.Outline - tylko zewnętrzne krawędzie tabeli
ReportTableEdgeStyle.Columns - krawędzie wokół kolumn
ReportTableEdgeStyle.Rows - krawędzie wokół wierszy
ReportTableEdgeStyle.Cell - krawędzie wokół każdej komórki

Parametr *aColor* określa kolor krawędzi. Określając kolor czcionki, można stosować predefiniowane kolory zdefiniowane w klasie *MigraDoc.DocumentObjectModel.Colors* (np. *Colors.Orange*).

SetQueryData

```
void SetQueryData(DateTime aStartTime, DateTime aStartTime, string aAggrInterval);
```

```
Sub SetQueryData(ByVal aStartTime As Date, ByVal aEndTime As Date, ByVal aAggrInterval
As String)
```

```
void SetQueryData(DateTime aStartTime, DateTime aStartTime, TimeSpan aAggrInterval);
```

```
Sub SetQueryData(ByVal aStartTime As Date, ByVal aEndTime As Date, ByVal aAggrInterval
As System.TimeSpan)
```

Metoda określa parametry zapytania o dane w przypadku tabel tworzonych na podstawie odczytu danych archiwalnych zmiennych procesowych.

Parametry *aStartTime* i *aEndTime* określają czasy początku i końca zapytania o dane. Parametr *aAggrInterval* określa okres agregacji danych.

Zalecane jest stosowanie wersji z parametrem *aAggrInterval* w postaci tekstowej. Okres podawany jest w standardzie OPC, np. ”1D”, ”3H”. Tylko ta wersja metody rozwiązuje problemy z określeniem prawidłowego okresu czasu i przedziałów wartości w przypadku wykonywania zapytań obejmujących moment zmiany czasu zimowy-letni.

5.11 Klasa ReportTableColumn

Klasa reprezentuje kolumnę dodaną do tabeli pionowej. Obiekty klasy *ReportTableColumn* są tworzone przy pomocy funkcji *AddColumn* klasy *ReportTable*.

5.11.1 Właściwości

AggrName

```
string AggrName { get; set;}
```

```
Property AggrName As String
```

Właściwość określa nazwę agregatu wartości historycznych zmiennej procesowej.

Callback

```
CellCallback Callback { get; set; }
Property Callback As CellCallback
```

Właściwość określa funkcję zwrotną, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ColumnAltStyle

```
string ColumnAltStyle { get; set;}
```

Property **ColumnAltStyle** As **String**

Właściwość określa nazwę stylu dla komórki wartości w parzystych wierszach tabeli.

ColumnStyle

```
string ColumnStyle { get; set;}
```

Property **ColumnStyle** As **String**

Właściwość określa nazwę stylu dla komórki wartości.

ErrorMode

```
ReportTableCellErrorBehavior ErrorMode { get; set; }
```

Property **ErrorMode** As **ReportTableCellErrorBehavior**

Właściwość określa sposób tworzenia zawartości komórek wartości, dla których brak danych lub ich status jest niepoprawny. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ErrorStyle

```
string ErrorStyle { get; set;}
```

Property **Errortyle** As **String**

Właściwość określa nazwę stylu dla komórek wartości z niepoprawnym statusem danych.

Format

```
string Format { get; set;}
```

Property **Format** As **String**

Właściwość określa sposób formatowania wartości wyświetlanych w komórkach kolumny.

HeaderStyle

```
string HeaderStyle { get; set;}
```

Property **HeaderStyle** As **String**

Właściwość określa nazwę stylu dla komórki nagłówka kolumny.

HeaderText

```
string HeaderText { get; set;}
```

Property **HeaderText** As **String**

Właściwość określa tekst wyświetlany w linii nagłówka kolumny.

ItemName

```
string ItemName { get; set;}
```
Property **ItemName** As **String**

W zależności od kontekstu użycia właściwość określa:

- nazwę zmiennej procesowej, której wartości archiwalne są odczytywane,
- nazwę kolumny specjalnej,
- nazwę pola kolekcji danych przekazanej do tabeli.

MigraColumn

```
MigraDoc.DocumentObjectModel.Column MigraColumn { get; }
```

ReadOnly Property **MigraColumn** As **MigraDoc.DocumentObjectModel.Column**

Właściwość zwraca bazowy obiekt kolumny tabeli biblioteki MigraDoc.

ReportTable

```
ReportTable ReportTable { get; }
```

ReadOnly Property **ReportTable** As **ReportTable**

Właściwość zwraca obiekt tabeli, do której należy kolumna.

Width

```
int Width { get; set;}
```

Property **Width** As **Integer**

Właściwość określa szerokość kolumny w milimetrach.

5.12 Klasa ReportTableRow

Klasa reprezentuje wiersz dodany do tabeli pionowej. Obiekty klasy *ReportTableRow* są tworzone przy pomocy funkcji *AddRow* klasy *ReportTable*.

5.12.1 Właściwości

AggName

```
string AggName { get; set;}
```

Property **AggName** As **String**

Właściwość określa nazwę agregatu wartości historycznych zmiennej procesowej.

Callback

```
CellCallback Callback { get; set; }
```

Property **Callback** As **CellCallback**

Właściwość określa funkcję zwrotną, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ErrorMode

```
ReportTableCellErrorBehavior ErrorMode { get; set; }
```

Property **ErrorMode** As **ReportTableCellErrorBehavior**

Właściwość określa sposób tworzenia zawartości komórek wartości, dla których brak danych lub ich status jest niepoprawny. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ErrorStyle

```
string ErrorStyle { get; set;}
```

Property **Errortyle** As **String**

Właściwość określa nazwę stylu dla komórek wartości z niepoprawnym statusem danych.

FirstStyle

```
string FirstStyle { get; set;}
```

Property **FirstStyle** As **String**

Właściwość określa nazwę stylu dla komórki w pierwszej kolumnie wiersza.

Format

```
string Format { get; set;}
```

Property **Format** As **String**

Właściwość określa sposób formatowania wartości wyświetlanych w komórkach wiersza.

HeaderText

```
string HeaderText { get; set;}
```

Property **HeaderText** As **String**

Właściwość określa tekst wyświetlany w pierwszej kolumnie wiersza.

ItemName

```
string ItemName { get; set;}
```

Property **ItemName** As **String**

W zależności od kontekstu użycia właściwość określa:

- nazwę zmiennej procesowej, której wartości archiwalne są odczytywane;
- nazwę pola kolekcji danych przekazanej do tabeli.

MigraRow

```
MigraDoc.DocumentObjectModel.Row MigraRow { get; }
```

ReadOnly Property **MigraRow** As **MigraDoc.DocumentObjectModel.Row**

Właściwość zwraca bazowy obiekt wiersza tabeli biblioteki MigraDoc.

NextStyle

```
string NextStyle { get; set;}
```

Property **NextStyle** As **String**

Właściwość określa nazwę stylu dla komórek w drugiej i następnych kolumnach wiersza.

ReportTable

```
ReportTable ReportTable { get; }
```

ReadOnly Property **ReportTable** As **ReportTable**

Właściwość zwraca obiekt tabeli, do której należy wiersz.

5.13 Klasa ReportTableCalcColumn

Klasa reprezentuje kolumnę wyliczaną dodaną do tabeli. Obiekty klasy *ReportTableColumn* są tworzone przy pomocy funkcji *AddCalculatedColumn* klasy *ReportTable*.

5.13.1 Właściwości

CalcType

```
CalculationType CalcType { get; set; }
```

Property **CalcType** As **CalculationType**

Właściwość określa rodzaj wyliczanego agregatu.

Callback

```
CellCallback Callback { get; set; }
```

Property **Callback** As **CellCallback**

Właściwość określa funkcję zwrotną, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ColumnStyle

```
string ColumnStyle { get; set;}
```

Property **ColumnStyle** As **String**

Właściwość określa nazwę stylu dla komórki wartości.

ColumnAltStyle

```
string ColumnAltStyle { get; set;}
```

Property **ColumnAltStyle** As **String**

Właściwość określa nazwę stylu dla komórki wartości w parzystych wierszach tabeli.

ErrorMode

```
ReportTableCellErrorBehavior ErrorMode { get; set; }
```

Property **ErrorMode** As **ReportTableCellErrorBehavior**

Właściwość określa sposób tworzenia zawartości komórek wartości, dla których brak danych lub ich status jest niepoprawny. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ErrorStyle

```
string ErrorStyle { get; set;}
```

Property **Errortyle** As **String**

Właściwość określa nazwę stylu dla komórek wartości z niepoprawnym statusem danych.

Format

```
string Format { get; set;}
```

Property **Format** As **String**

Właściwość określa sposób formatowania wartości wyświetlanych w komórkach kolumny.

HeaderText

```
string HeaderText { get; set;}
```

Property **HeaderText** As **String**

Właściwość określa tekst wyświetlany w linii nagłówka kolumny.

HeaderStyle

```
string HeaderStyle { get; set;}
```

Property **HeaderStyle** As **String**

Właściwość określa nazwę stylu dla komórki nagłówka kolumny.

MigraColumn

```
MigraDoc.DocumentObjectModel.Column MigraColumn { get; }
```

ReadOnly Property **MigraColumn** As **MigraDoc.DocumentObjectModel.Column**

Właściwość zwraca bazowy obiekt kolumny tabeli biblioteki MigraDoc.

ReportTable

```
ReportTable ReportTable { get; }
```

ReadOnly Property **ReportTable** As **ReportTable**

Właściwość zwraca obiekt tabeli, do której należy kolumna.

Width

```
int Width { get; set;}
```

Property **Width** As **Integer**

Właściwość określa szerokość kolumny w milimetrach.

5.14 Klasa ReportTableCalcRow

Klasa reprezentuje wiersz wyliczany dodany do tabeli. Obiekty klasy *ReportTableCalcRow* są tworzone przy pomocy funkcji *AddCalculatedRow* klasy *ReportTable*.

5.14.1 Właściwości

CalcType

```
CalculationType CalcType { get; set; }
```

Property **CalcType** As **CalculationType**

Właściwość określa rodzaj wyliczanego agregatu.

Callback

```
CellCallback Callback { get; set; }
```

Property **Callback** As **CellCallback**

Właściwość określa funkcję zwrotną, która będzie wykonywana dla każdej komórki wartości w momencie generowania jej zawartości. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ErrorMode

```
ReportTableCellErrorBehavior ErrorMode { get; set; }
```

Property **ErrorMode** As **ReportTableCellErrorBehavior**

Właściwość określa sposób tworzenia zawartości komórek wartości, dla których brak danych lub ich status jest niepoprawny. Szczegóły użycia znajdują się w opisie metody *AddColumn* klasy *ReportTable*.

ErrorStyle

```
string ErrorStyle { get; set;}
```

Property **Errortyle** As **String**

Właściwość określa nazwę stylu dla komórek wartości z niepoprawnym statusem danych.

FirstStyle

```
string FirstStyle { get; set;}
```

Property **FirstStyle** As **String**

Właściwość określa nazwę stylu dla komórki w pierwszej kolumnie wiersza.

Format

```
string Format { get; set;}
```

Property **Format** As **String**

Właściwość określa sposób formatowania wartości wyświetlanych w komórkach wiersza.

HeaderText

```
string HeaderText { get; set;}
```

Property **HeaderText** As **String**

Właściwość określa tekst wyświetlany w pierwszej kolumnie wiersza.

MigraRow

```
MigraDoc.DocumentObjectModel.Row MigraRow { get; }
```

ReadOnly Property **MigraRow** As **MigraDoc.DocumentObjectModel.Row**

Właściwość zwraca bazowy obiekt wiersza tabeli biblioteki MigraDoc.

NextStyle

```
string NextStyle { get; set;}
```

Property **NextStyle** As **String**

Właściwość określa nazwę stylu dla komórek w drugiej i następnych kolumnach wiersza.

ReportTable

```
ReportTable ReportTable { get; }
```

ReadOnly Property **ReportTable** As **ReportTable**

Właściwość zwraca obiekt tabeli, do której należy wiersz.

5.15 Klasa ReportTableCell

Klasa reprezentuje obiekt pojedynczej komórki tabeli. Obiekty klasy *ReportTableCell* są tworzone automatycznie. Wykorzystywane są jako parametr funkcji zwrotnej typu *CellCalback*, pozwalającej na kontrolowanie zawartości komórki. Mechanizm ten jest wyjaśniony w opisie metody *AddColumn* klasy *ReportTable*.

Część właściwości klasy służy do identyfikacji komórki, a inne pozwalają na modyfikację końcowej zawartości komórki.

5.15.1 Właściwości

CalcColumn

```
ReportTableCalcColumn CalcColumn { get; }
```

ReadOnly Property **CalcColumn** As **ReportTableCalcColumn**

Właściwość przechowuje odwołanie do obiektu kolumny wyliczanej, do której należy komórka. Wartość null oznacza, że komórka nie należy do kolumny wyliczanej.

CalcRow

```
ReportTableCalcRow CalcRow { get; }
```

ReadOnly Property **CalcRow** As **ReportTableCalcRow**

Właściwość przechowuje odwołanie do obiektu wiersza wyliczanego, do którego należy komórka. Wartość null oznacza, że komórka nie należy do wiersza wyliczanego.

Column

```
ReportTableColumn Column { get; }
```

ReadOnly Property **Column** As **ReportTableColumn**

Właściwość przechowuje odwołanie do obiektu kolumny, do której należy komórka. Właściwość ma znaczenie tylko dla tabel pionowych.

Content

```
string Content { get; set;}
```

Property **Content** As **String**

Właściwość określa tekst, który zostanie umieszczony w zawartości komórki. Tekst ten powstał na podstawie wartości komórki *RawValue*, jej statusu *ValueStatus* oraz innych właściwości określonych w definicji wiersza lub komórki. Funkcja zwrotna *CellCallback* może zmienić wartość właściwości w celu modyfikacji zawartości komórki.

Index

```
int Index { get;}
```

ReadOnly Property **Index** As **Integer**

Właściwość określa, w zależności od typu tabeli, indeks wiersza lub kolumny, do której należy komórka.

MigraCell

```
MigraDoc.DocumentObjectModel.Cell MigraCell { get; }
```

ReadOnly Property **MigraCell** As **MigraDoc.DocumentObjectModel.Cell**

Właściwość zwraca bazowy obiekt komórki tabeli biblioteki MigraDoc.

Row

```
ReportTableRow Row { get; }
```

ReadOnly Property **Row** As **ReportTableRow**

Właściwość przechowuje odwołanie do obiektu wiersza, do której należy komórka. Właściwość ma znaczenie tylko dla tabel poziomych.

RowIdx

```
int RowIdx { get;}
```

ReadOnly Property **RowIdx** As **Integer**

Właściwość określa indeks wiersza, do którego należy komórka. Dotyczy tylko komórek z kolumn wyliczanych.

RawValue

```
object RawValue { get;}
```

ReadOnly Property **RavValue** As **Object**

Właściwość przechowuje surową wartość komórki. Wartość pochodzi z zapytania o dane archiwalne lub z dostarczonej do tabeli kolekcji danych.

SourceValues

```
System.Collections.ICollection<CellValue> SourceValues { get;}
```

ReadOnly Property **SourceValues** As **System.Collections.ICollection**(Of **CellValue**)

Właściwość przekazuje kolekcję wartości wszystkich komórek wiersza lub kolumny. Używana tylko w komórkach należących do kolumn lub wierszy wyliczanych. Pozwala na implementację własnych algorytmów wyliczania wartości agregowanych.

Style

```
string Style { get; set;}
```

Property **Style** As **String**

Właściwość określa nazwę stylu komórki. Funkcja zwrotna *CellCallback* może zmienić nazwę stylu w celu modyfikacji wyglądu komórki.

ValueStatus

```
Askom.Ascent.StatusOpc ValueStatus { get; }
```

ReadOnly Property **ValueStatus** As **Askom.Ascent.StatusOpc**

Właściwość przechowuje status wartości komórki, pochodzący z zapytania o dane archiwalne zmiennej procesowej.

5.15.2 Metody

AvgValue

object AvgValue();

Function **AvgValue** () As **Object**

Metoda wylicza wartość średnią komórek wiersza lub kolumny, dla komórki należącej do wyliczanej kolumny lub wyliczanego wiersza.

MaxValue

object MaxValue();

Function **MaxValue** () As **Object**

Metoda wylicza wartość maksymalną komórek wiersza lub kolumny, dla komórki należącej do wyliczanej kolumny lub wyliczanego wiersza

MinValue

object MaxValue();

Function **MinValue** () As **Object**

Metoda wylicza wartość minimalną komórek wiersza lub kolumny, dla komórki należącej do wyliczanej kolumny lub wyliczanego wiersza

SumValue

```
object SumValue();
```

Function **SumValue** () As **Object**

Metoda wylicza sumę wartości komórek wiersza lub kolumny, dla komórki należącej do wyliczanej kolumny lub wyliczanego wiersza.

5.16 Klasa CellValue

Klasa reprezentuje obiekt służący do przechowania wartości komórki tabeli. Stosowana jest w trakcie wyliczania wartości agregowanych dla komórek należących do kolumn lub wierszy wyliczanych. Kolekcje obiektów *CellValue* tworzone są automatycznie. Dostęp do nich odbywa się poprzez właściwość *SourceValues* obiektów klasy *ReportTableCell.*

5.16.1 Właściwości

Content

```
string Content { get;}
```

ReadOnly Property **Content** As **String**

Właściwość udostępnia zawartość komórki tabeli po konwersji na postać tekstową.

Status

```
StatusOpc Status { get; }
```

ReadOnly Property **Status** As **Askom.Ascent.StatusOpc**

Właściwość udostępnia status wartości komórki tabeli.

TimeStamp

```
DateTime TimeStamp { get; }
```

ReadOnly Property **TimeStamp** As **DateTime**

Właściwość udostępnia stempel czasu wartości komórki tabeli. Dotyczy tylko tabel tworzonych na podstawie zapytań o dane archiwalne.

Value

```
object Value { get;}
```

ReadOnly Property **Value** As **Object**

Właściwość udostępnia wartość komórki tabeli w postaci źródłowej.

5.17 Klasa ReportChart

Klasa reprezentuje obiekt wykresu umieszczonego w treści raportu. Obiekt klasy *ReportChart* tworzone sa przy pomcy metody *AddChart* klasy *Report*.

5.17.1 Właściwości

MigraChart

```
MigraDoc.DocumentObjectModel.Shapes.Charts.Chart MigraChart { get; }
```

ReadOnly Property **MigraChart** As **MigraDoc.DocumentObjectModel.Shapes.Charts.Chart**

Właściwość zwraca bazowy obiekt wykresu biblioteki MigraDoc.

MigraLegend

```
MigraDoc.DocumentObjectModel.Legend MigraLegend { get; }
```

ReadOnly Property **MigraLegend** As **MigraDoc.DocumentObjectModel.Legend**

Właściwość zwraca obiekt legendy wykresu biblioteki MigraDoc. Legenda dodawana jest do wykresu metodą *AddLegend*.

5.17.2 Metody

AddLegend

```
void AddLegend(ReportChartLegendPosition aPosition, string aStyle);
```

Sub **AddLegend**(ByVal aPosition As **ReportChartLegendPosition**, ByVal aStyle As **String**)

Metoda służy do dodania do wykresu legendy opisującej serie danych umieszczonych na wykresie.

Parametr *aPosition* określa lokalizację legendy. Możliwe są poniższe wartości:

ReportChartLegendPosition.Bottom - poniżej obszaru wykresu

ReportChartLegendPosition.Top - powyżej obszaru wykresu

ReportChartLegendPosition.Left - na lewo od obszaru wykresu

ReportChartLegendPosition.Right - na prawo od obszaru wykresu

Parametr *aStyle* określa nazwę stylu dla tekstów umieszczonych w wygenerowanej legendzie.

AddSeries

```
ReportChartSeries AddSeries(string aName,
MigraDoc.DocumentObjectModel.Shapes.Chart.ChartType aChartType, string aVarName,
string aAggrName, MigraDoc.DocumentObjectModel.Color aColor,
MigraDoc.DocumentObjectModel.Color aFillColor);
```

Function **AddSeries**(ByVal aName As **String**, ByVal aChartPar As **MigraDoc.DocumentObjectModel.Shapes.Charts.ChartType**, ByVal aVarName As **String**, ByVal aAggrName As **String**, ByVal aColor As **MigraDoc.DocumentObjectModel.Color**, ByVal aFillColor As **MigraDoc.DocumentObjectModel.Color**) As **ReportChartSeries**

Metoda służy do dodania serii danych do wykresu tworzonego na podstawie danych archiwalnych zmiennych procesowych.

Parametr *aName* określa nazwę serii danych. Nazwa serii jest używana w legendzie wykresu.

Parametr *aChartType* określa sposób wyświetlania danych serii. W ramach pojedynczego wykresu można stosować różne typy serii, ale nie wszystkie kombinacje są dozwolone.

Parametry *aVarName* i *aAggrName* określają nazwę zmiennej procesowej oraz nazwę wyliczanego agregatu.

Parametr *aColor* określa kolor linii dla rysowania serii, a parametr *aFillColor* określa kolor wypełniania (stosowany dla niektórych typów serii, np. słupkowego).

```
ReportChartSeries AddSeries(string aName,
MigraDoc.DocumentObjectModel.Shapes.Chart.ChartType aChartType, string aFieldName,
MigraDoc.DocumentObjectModel.Color aColor,
MigraDoc.DocumentObjectModel.Color aFillColor);
```

Function **AddSeries**(ByVal aName As **String**, ByVal aChartPar As **MigraDoc.DocumentObjectModel.Shapes.Charts.ChartType**, ByVal aFieldName As **String**, ByVal aColor As **MigraDoc.DocumentObjectModel.Color**, ByVal aFillColor As **MigraDoc.DocumentObjectModel.Color**) As **ReportChartSeries**

Metoda służy do dodania serii danych do wykresu tworzonego na podstawie danych przekazanych w momencie tworzenia wykresu (poprzez obiekt typu *DataTable* lub *ICollection*).

Parametr *aChartType* określa sposób wyświetlania danych serii. W ramach pojedynczego wykresu można stosować różne typy serii, ale nie wszystkie kombinacje są dozwolone.

Parametr *aFieldName* określa nazwę elementu używanego przez serię. Jest to nazwa kolumny w obiekcie typy *DataTable* lub nazwa właściwości obiektów przekazanych w kolekcji *ICollection*.

Parametr *aColor* określa kolor linii dla rysowania serii, a parametr *aFillColor* określa kolor wypełniania (stosowany dla niektórych typów serii, np. słupkowego).

AddXSeries

```
void AddXSeries(string aFieldName);
```

```
Sub AddXSeries(ByVal aFieldName As String)
```

Metoda określa serię danych używanych do opisu osi X wykresu. Parametr *aFieldName* określa nazwę elementu używanego przez serię osi X. Jest to nazwa kolumny w obiekcie typy *DataTable* lub nazwa właściwości obiektów przekazanych w kolekcji *ICollection*.

Metoda jest używana tylko dla wykresów z kolekcji danych. Dla wykresów z danych historycznych oś X jest zawsze osią czasu.

CustomizeXAxis

```
void CustomizeXAxis(MigraDoc.DocumentObjectModel.Color aLineColor, string aStyle, string aFormat, int aTickDensity);
```

```
Sub CustomizeXAxis(ByVal aLine As MigraDoc.DocumentObjectModel.Color, ByVal aStyle As String, ByVal aFormat As String, ByVal aTickDensity As Integer)
```

Metoda służy do określenia sposobu utworzenia osi X wykresu.

Parametr *aLineColor* określa kolor linii osi. Parametr *aStyle* definiuje nazwę stylu używanego do generacji tekstów opisujących skalę osi. Parametr *aFormat* określa sposób formatowania (konwersji wartość na tekst) wartości skali osi X. Parametr *aTickDensity* określa gęstość opisu skali osi. Wartość 1 oznacza, że będzie opisany każdy punkt wykresu. Wartości większe od 1 oznaczają: co który punkt ma być opisany.

CustomizeYAxis

```
void CustomizeYAxis(double? aMinValue, double? aMaxValue, double? aTickValue, bool aHasGridLines, MigraDoc.DocumentObjectModel.Color aLineColor, string aStyle, string aFormat);
```

```
Sub CustomizeYAxis(ByVal aMinValue As Double?, ByVal aMaxValue As Double?, ByVal aTickValue As Double?, ByVal aHasGridLines As Boolean, ByVal aLine As MigraDoc.DocumentObjectModel.Color, ByVal aStyle As String, ByVal aFormat As String)
```

Metoda służy do określenia sposobu utworzenia osi Y wykresu.

Parametry *aMinValue*, *aMaxValue* i *aTickValue* służą do definiowania sposobu opisu skali osi Y. Oznaczają one odpowiednio: wartość minimalną skali, wartość maksymalną oraz odstęp pomiędzy kolejnymi opisami na skali osi. Przekazanie wartości *null* (*Nothing*) oznacza automatyczne skalowanie osi Y.
Parametr *aHasGrdiLines* określa, czy należy rysować poziome linie siatki wartości na wykresie. Parametr *aLineColor* określa kolor linii osi i siatki.
Parametr *aStyle* definiuje nazwę stylu używanego do generacji tekstów opisujących skalę osi. Parametr *aFormat* określa sposób formatowania (konwersji wartości na tekst) wartości skali osi X.

GenerateContent

```
void GenerateContent ();
```

```
Sub GenerateContent ()
```

Metoda służy do wygenerowanie kompletnej zawartości wykresu na podstawie dostępnych danych. Użycie metody nie jest obowiązkowe. Metoda *Render* klasy *Report* automatycznie wywołuje metodę *GenerateContent* dla wszystkich wykresów, dla których metoda nie była jeszcze wykonana.

SetQueryData

```
void SetQueryData(DateTime aStartTime, DateTime aStartTime, TimeSpan aAggrInterval);
```

```
Sub SetQueryData(ByVal aStartTime As Date, ByVal aEndTime As Date, ByVal aAggrInterval As System.TimeSpan)
```

Metoda określa parametry zapytania o dane w przypadku wykresów tworzonych na podstawie odczytu danych archiwalnych zmiennych procesowych.

Parametry aStartTime i aEndTime określają czasy początku i końca zapytania o dane. Parametr aAggrInterval określa okres agregacji danych.

SetXAxisTitle

```
void SetXAxisTitle(string aTitle, string aStyle);
```

Sub **SetXAxisTitle**(ByVal aTitle As **String**, ByVal aStyle As **String**)

Metoda służy do określenia opisu osi X wykresu. Parametr *aTitle* definiuje opis osi, a parametr *aStyle* nazwę stylu.

SetYAxisTitle

```
void SetYAxisTitle(string aTitle, string aStyle);
```

Sub **SetYAxisTitle**(ByVal aTitle As **String**, ByVal aStyle As **String**)

Metoda służy do określenia opisu osi Y wykresu. Parametr aTitle definiuje opis osi, a parametr aStyle nazwę stylu.

5.18 Klasa ReportChartSeries

Klasa reprezentuje obiekt serii danych wstawionej do wykresu. Obiekty klasy *ReportChartSeries* tworzone są przy pomocy metody *AddSeries* klasy *ReportChart*.

5.18.1 Właściwości

AggrName

```
string AggrName { get; set;}
```

Property **AggrName** As **String**

Właściwość określa nazwę agregatu wartości historycznych zmiennej procesowej.

ItemName

```
string ItemName { get; set;}
```

Property **ItemName** As **String**

W zależności od kontekstu użycia właściwość określa:

- nazwę zmiennej procesowej, której wartości archiwalne są odczytywane;
- nazwę pola kolekcji danych przekazanej do wykresu.

MigraSeries

```
MigraDoc.DocumentObjectModel.Shapes.Charts.Series MigraSeries { get; }
```

ReadOnly Property **MigraSeries** As **MigraDoc.DocumentObjectModel.Shapes.Charts.Series**

Właściwość zwraca bazowy obiekt serii biblioteki MigraDoc.

5.18.2 Metody

SetDataLabels

```
void SetDataLabels(string aStyle, string aFormat,
MigraDoc.DocumentObjectModel.Shapes.Charts.DataLabelPosition aPosition);
```

Sub **SetDataLabels**(ByVal aStyle As **String**, ByVal aFormat As **String**, ByVal aPosition As **MigraDoc.DocumentObjectModel.Shapes.Charts.DataLabelPosition**)

Metoda służy do określenia sposobu tworzenia etykiet danych serii. Jeżeli metoda nie zostanie użyta, to etykiety nie są tworzone. Etykiety nie są dostępne dla wszystkich typów serii.
Parametr *aStyle* określa nazwę stylu używanego do tworzenia etykiet. Parametr *aFormat* określa sposób formatowania (konwersji wartość na tekst) wartości etykiet. Parametr *aPosition* definiuje położenie etykiet względem punktów serii.

SetMarkersStyle

```
void SetMarkersStyle(MigraDoc.DocumentObjectModel.Shapes.Charts.MarkerStyle aStyle, int
aSize, MigraDoc.DocumentObjectModel.Color aForegroundColor,
MigraDoc.DocumentObjectModel.Color aBackgroundColor);
```

```
Sub SetMarkersStyle(ByVal aStyle As
MigraDoc.DocumentObjectModel.Shapes.Charts.MarkerStyle, ByVal aSize As  Integer,
ByVal aForegroundColor As  MigraDoc.DocumentObjectModel.Color, ByVal
aBackgroundColor As  MigraDoc.DocumentObjectModel.Color)
```

Metoda służy do określenia sposobu tworzenia znaczników punktów serii. Jeżeli metoda nie zostanie użyta, to znaczniki nie są tworzone. Znaczniki nie są dostępne dla wszystkich typów serii.

Parametr *aStyle* określa wygląd znaczników. Parametr *aSize* określa rozmiar znaczników w milimetrach. Parametry *aForegroundColor* i *aBackgroundColor* określają kolorystykę znaczników.

Asix.Evo - Skrypty

1 Podstawy użycia skryptów

W wielu aplikacjach zachodzi potrzeba zaimplementowania funkcjonalności, których nie można uzyskać używając tylko standardowych mechanizmów diagramów i obiektów wizualizacyjnych. Typowe przykłady takich funkcji to:

- Wykonywanie nietypowych przeliczeń wartości zmiennych procesowych, szczególnie z użyciem wartości archiwalnych.
- Zaprogramowanie nietypowych reakcji na zdarzenia zachodzące w kontrolowanym procesie.
- Przekazywanie na zewnątrz aplikacji Asix.Evo wartości zmiennych i innych informacji o działaniu systemu.
- Zaawansowane strategie wykrywania sytuacji alarmowych.
- Pozyskiwanie danych z nietypowych źródeł danych, np. z baz danych firm trzecich.
- Wykonywanie raportów.
- Tworzenie specjalizowanych interfejsów użytkownika opartych o swobodnie tworzone okna.

W celu realizacji powyższych funkcji (i innych podobnych) Asix.Evo został wyposażony w moduł obsługi skryptów użytkownika. Asix.Evo zapewnia zintegrowane środowisko do tworzenia i uruchamiania skryptów.

Asix.Evo wykorzystuje możliwości stwarzane przez platformę .Net. W rzeczywistości skrypty aplikacyjne są w pełni kompilowanymi modułami .Net. Możliwe jest użycie języków programowania C# i Visual Basic. Skrypty użytkownika mogą korzystać z pełnego zestawu obiektów i funkcji platformy .Net. Dodatkowo środowisko wykonania skryptu jest rozszerzane o funkcje dostępu do elementów aplikacji Asix.Evo.

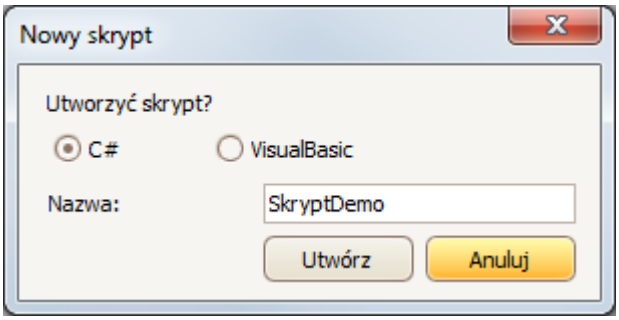
Skrypty użytkownika są uruchamiane w osobnych domenach aplikacji oraz we własnych wątkach. Skrypty i główny program są od siebie wzajemnie izolowane. Dzięki temu minimalizowane jest ryzyko, że błędne działanie skryptu spowoduje upadek całej aplikacji.

1.1 Utworzenie skryptu

Aby utworzyć nowy skrypt należy:

- Otworzyć panel *Eksplorator aplikacji.*

- Podwójnie kliknąć myszką na węźle *Skrypty* lub wykonać komendę *Utwórz skrypt* z menu kontekstowego węzła *Skrypty*. Spowoduje to otwarcie poniższego okienka:



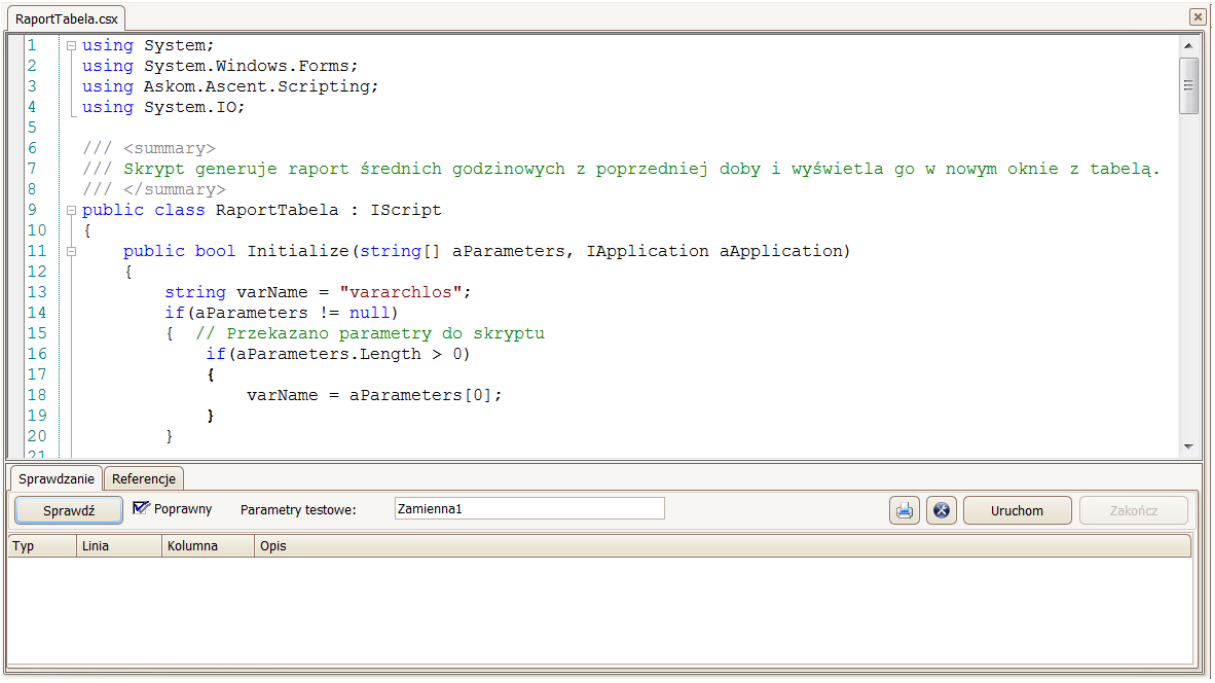
Rys. Okno deklaracji nowego skryptu.

- Określić język i nazwę skryptu oraz użyć przycisk *Utwórz*.

Efektem powyższych działań będzie wygenerowanie początkowego wzorca skryptu oraz otwarcie panelu edycji skryptów.

1.2 Edycja skryptów

Panel edycji skryptów jest podstawowym narzędziem służącym do pisania i testowego uruchamiania skryptów w aplikacji Asix.Evo. Dopuszczalne jest także użycie dowolnego innego edytora programistycznego – należy tylko zachować lokalizację pliku skryptu w katalogu definicyjnym aplikacji.



Rys. Panel edycji skryptów.

Edytor skryptów zapewnia kolorowanie kodu na podstawie składni, automatyczne formatowanie, natychmiastowe wykrywanie błędów, podpowiadanie i uzupełnianie nazw, rozwijanie składowych klas.

Niepoprawne fragmenty kodu są zaznaczane przez czerwone podkreślenie. Dodatkowo w dymku można wyświetlić skrócony opis błędu. Można też wykonać próbę pełnej kompilacji skryptu używając przycisk *Sprawdź*. Ewentualne błędy zostaną wyświetlone w tabelce komunikatów.

Jedną z najbardziej użytecznych funkcji edytora jest rozwijanie składowych i uzupełnianie nazw. Edytor sam rozpoznaje kontekst edycji i w odpowiednich momentach wyświetla okienko z proponowanymi alternatywami – mogą to być nazwy składowych klasy , nazwy zmiennych, elementy wyliczeniowe. Użytkownik może wymusić otwarcie okna podpowiedzi skrótem klawiaturowym *Ctrl-spacja*.

Panel edytora pozwala także na proste zarządzanie wykonaniem skryptu. Skrypt może zostać uruchomiony (z podaniem parametrów lub bez), można też wymusić zakończenie jego pracy.

1.3 Metody uruchamiania skryptów

Skrypty w aplikacji Asix.Evo uruchamiane są zawsze przy pomocy akcji operatorskiej *Script*.

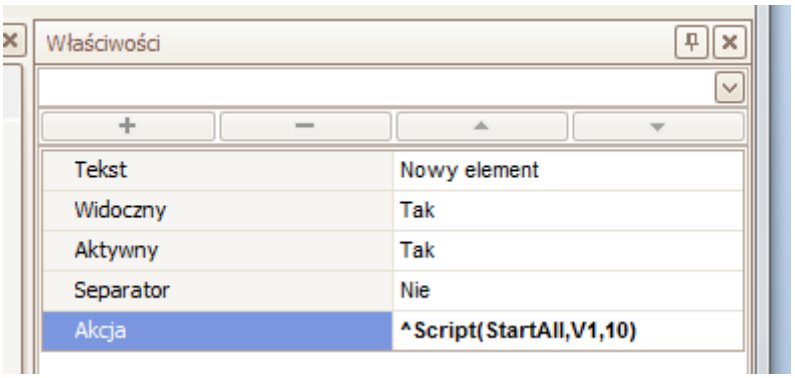
```
Script(nazwa , parametr1, parametr2, ...)
```

W treści wywołania należy podać nazwę skryptu oraz odpowiednią liczbę parametrów uruchomienia. Akcję *Script* można używać wszędzie tam, gdzie dopuszczalne jest użycie akcji operatorskiej.

Można wyróżnić dwie główne kategorie skryptów: skrypty interakcyjne uruchamiane w wyniku działań operatora oraz skrypty autonomiczne, działające w tle aplikacji, wykonujące funkcje nie związane bezpośrednio z działaniami operatora.

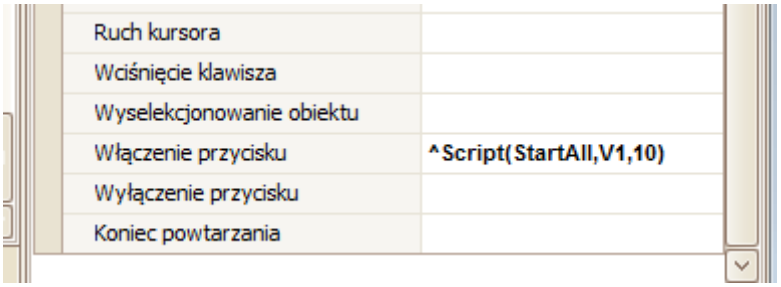
1.3.1 Skrypty interakcyjne

W przypadku skryptów interakcyjnych akcja operatorska *Script* jest używana jako akcja podpięta do zdarzenia obiektu wizualizacyjnego lub jako akcja menu użytkownika.



Rys. Przykład akcji operatorskiej w przypadku skryptów interakcyjnych.

Powyższy przykład pokazuje definicję elementu menu użytkownika, którego użycie będzie powodowało wykonanie skryptu *StartAll*.



Rys. Definicja elementu menu użytkownika, którego użycie będzie powodowało wykonanie skryptu StartAll.

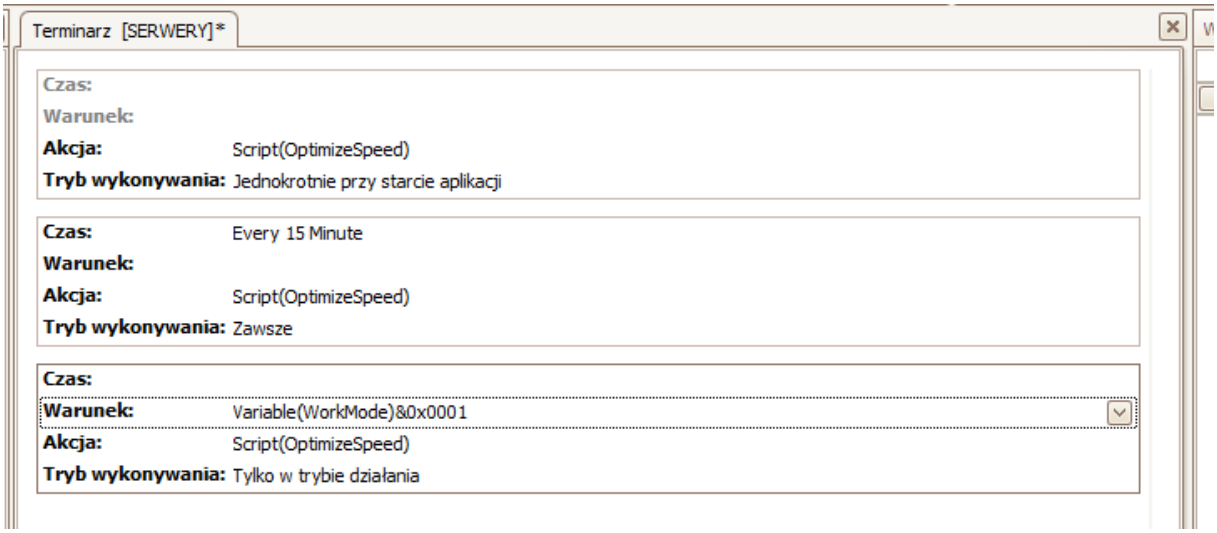
Kolejny przykład pokazuje identyczną akcję, ale tym razem użytą w obsłudze zdarzenia wciśnięcia przycisku w obiekcie klasy *Przycisk.*

1.3.2 Skrypty autonomiczne

Skrypty autonomiczne uruchamiane są z reguły poprzez mechanizm terminarza. Najczęściej skrypty tego typu używane są w elementach terminarza, których tryb wykonania jest równy *Jednokrotnie przy starcie aplikacji.* W tym trybie skrypt jest uruchomiony przy starcie i kontynuuje swoją pracę aż do zakończenia aplikacji. Możliwe jest też użycie trybu okresowego uruchamiania lub uruchamiania przy zajściu określonego zdarzenia. W tych trybach skrypt jest uruchamiany wielokrotnie – powinien wykonać swoje zadanie, a następnie się zakończyć.

Uwaga:

Funkcjonalność wykonywania pewnych funkcji okresowo lub w reakcji na zajście określonego zdarzenia, można zrealizować w łatwy sposób wewnątrz skryptu uruchamianego jednokrotnie. Taka technika jest bardziej efektywna – nie wymaga kosztownego procesu każdorazowego uruchomienia skryptu.

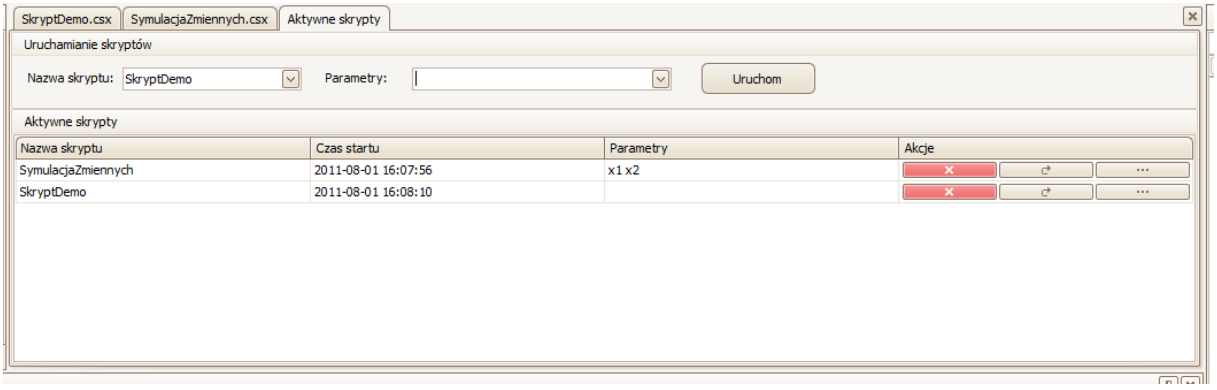


Rys. Terminarz - tryby uruchomienia skryptu OptimizeSpeed.

Powyższy przykład pokazuje różne tryby uruchomienia skryptu *OptimizeSpeed.*

1.4 Zarządzanie aktywnymi skryptami

Interaktywne zarządzanie pracującymi skryptami jest możliwe za pośrednictwem panelu diagnostycznego skryptów. Panel *Aktywne skrypty* otwierany jest poprzez wybranie węzła *Diagnostyka/Aktywne skrypty* w panelu *Eksplorator aplikacji* lub w trybie wykonania aplikacji za pośrednictwem panelu kontrolnego.



Rys. Panel aktywnych skryptów.

Okno aktywnych skryptów pozwala obserwować, zamykać i restartować wszystkie aktualnie pracujące skrypty. Umożliwia również uruchamianie nowych skryptów.

Alternatywne metody kontroli pracy skryptów oparte są na wykorzystaniu akcji operatorskich *ScriptRestart* i *ScriptClose* oraz funkcji *IsScriptWorking*. Skrypt może wymusić też własny restart, stosując metodę *RestartScript*.

1.5 Struktura kodu skryptu

Skrypty Asix.Evo mogą być tworzone w obiektowych językach C# oraz Visual Basic. W momencie utworzenia skryptu jest generowany szkielet jego kodu:

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class SkryptT1 : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System;
Imports System.Windows.Forms;
Imports Askom.AsixEvo.Scripting;

Public Class SkryptT2
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As IApplicati
on) As Boolean Implements IScript.Initialize

        Return True
    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Główna klasa skryptu musi implementować interfejs *IScript* zdefiniowany w przestrzeni *Askom.AsixEvo.Scripting*.

C#

```
public interface IScript
{
   bool Initialize(string[] aParameters, IApplication aAscent);
```

```
    void FinalizeScript();
}
```

VB

```
Public Interface IScript
    Function Initialize(ByVal aParameters() As String, ByVal aApplication As IApplication) As Boolean
    Sub FinalizeScript()
End Interface
```

Funkcja *Initialize*, to główna funkcja skryptu wywoływana w momencie jego uruchomienia. Funkcja otrzymuje na wejściu dwa parametry:

- *aParameters* – jest to tablica zawierająca wszystkie parametry przekazane podczas uruchomienia skryptu
- *aApplication* – przekazuje odwołanie do interfejsu *IApplication*, który pozwala na dostęp do stanu wykonywanej aplikacji. Jest to podstawowy mechanizm współpracy skryptu ze środowiskiem aplikacji Asix.Evo.

Wartość zwracana przez funkcję *Initialize* określa, czy skrypt zakończył pracę. Wartość *true* oznacza, że skrypt zrealizował wszystkie swoje funkcje i należy go rozładować. Jest to typowe działanie tzw. skryptów jednorazowych. Skrypty rezydentne powinny zwrócić wartość *false*. Skrypt pozostanie załadowany w pamięci i będzie mógł kontynuować swoją działalność.

Funkcja *FinalizeScript* jest wywoływana bezpośrednio przed zakończeniem pracy skryptu. Można w niej wykonać operacje wymagane do poprawnego zakończenia pracy skryptu, np. zwolnienie użytych zasobów systemowych, zamknięcie połączeń z bazą danych itp.

2 Przykłady typowych skryptów

2.1 Skrypty jednorazowe

Skrypty jednorazowe uruchamiane są najczęściej w sposób interaktywny przez operatora lub okresowo przy pomocy terminarza. Skrypt jednorazowy działa z reguły krótki czas. Całość jego funkcji znajduje się wewnątrz metody *Initialize* (lub innych metodach wywoływanych przez metodę *Initialize*). W przypadku skryptu jednorazowego metoda *Initialize* **musi** zwracać wartość *true*.

Przykład pokazuje skrypt, którego zadaniem jest rejestrowanie w pliku tekstowym obecności operatora. Operator ma za zadanie okresowo uruchamiać skrypt (np. poprzez użycie odpowiednio skonfigurowanego obiektu *Przycisk*), a każdorazowe uruchomienie spowoduje zapis komunikatu w pliku rejestru.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;
using System.IO;

public class Single : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      StreamWriter writer = new StreamWriter(
         Path.Combine(aApplication.ApplicationWorkingDirectory, "logs\\rejestr.txt"), true,
         System.Text.ASCIIEncoding.GetEncoding(1250));
      String msg = String.Format("{0} Zgłoszenie obecności: operator = {1}, stanowisko = {2}",
         DateTime.Now, aApplication.CurrentUserName, aApplication.CurrentStationName);
      writer.WriteLine(msg);
      writer.Close();

      aApplication.ReportMessage(EvoMessageType.Info, "Zgłoszenie operatora zarejestrowane");

      return true;
   }

   public void FinalizeScript()
   {

   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting
Imports System.IO

Public Class SingleVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize

        Dim writer As StreamWriter
        writer = New StreamWriter(Path.Combine(aApplication.ApplicationWorkingDirectory,
            "logs\\rejestr.txt"), True, System.Text.ASCIIEncoding.GetEncoding(1250))

        Dim msg As String
        msg = String.Format("{0} Zgłoszenie obecności: operator = {1}, stanowisko = {2}", _
            Date.Now, aApplication.CurrentUserName, aApplication.CurrentStationName)
        writer.WriteLine(msg)
        writer.Close()

        aApplication.ReportMessage(EvoMessageType.Info, "Zgłoszenie operatora zarejestrowane")

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Całość kodu znajduje się w funkcji *Initialize*. Na początku tworzony jest obiekt klasy *StreamWriter*. W jego konstruktorze przekazywana jest nazwa pliku, wybierana opcja dopisywania do pliku oraz wybierany sposób kodowania tekstów (domyślnie stosowane byłoby kodowanie UTF8). Nazwa pliku tworzona jest na podstawie nazwy katalogu roboczego aplikacji zwracanego przez właściwość *ApplicationWorkingDirectory*. Następnie tworzony i zapisywany jest tekst komunikatu, w którego treści umieszczana jest nazwa aktualnie zalogowanego użytkownika *CurrentUserName* oraz nazwa stanowiska *CurrentStationName*. Na końcu funkcja *Initialize* zwraca wartość *true* informującą o zakończeniu działania skryptu.

Wywołanie funkcji *ReportMessage* nie jest niezbędne. Dobrym zwyczajem jest jednak rejestrowanie w standardowym logu komunikatów informujących o pracy skryptu oraz ewentualnych błędach wykonania.

2.2 Skrypty rezydentne

Skrypty rezydentne charakteryzują się tym, że po uruchomieniu, pozostają w trybie działania aż do momentu zakończenia pracy aplikacji. Z reguły uruchamiane są przy starcie aplikacji przy pomocy mechanizmu terminarza.

W skryptach rezydentnych metoda *Initialize* służy z reguły jedynie do inicjacji skryptu, a główna funkcjonalność jest realizowana w wykreowanych wątkach lub zadeklarowanych funkcjach zwrotnych. W przypadku skryptu rezydentnego metoda *Initialize* **musi** zwracać wartość *false*.

Istnieją różne wzorce programowe służące do tworzenia skryptów rezydentnych, opisane w następnych rozdziałach.

2.2.1 Pętle w metodzie Initialize

Metoda niekończącej się pętli w metodzie *Initialize*, to najprostsza konstrukcja mająca wiele wspólnego ze skryptami jednorazowymi. Główna funkcjonalność realizowana jest w metodzie *Initialize* . Działanie metody jednak nigdy się nie kończy (a dokładnie zostanie automatycznie zakończone w momencie końca pracy aplikacji). Jest to jedyny przypadek, gdzie z oczywistych powodów, wartość zwracana przez metodę *Initialize* nie jest istotna.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Loop : IScript
{
   IApplication application;

   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      application = aApplication;
```

```
      while (true)
      {
         DoTask();
         aApplication.Sleep(TimeSpan.FromSeconds(1));
      }

      return true;
   }

   public void FinalizeScript()
   {

   }

   void DoTask()
   {
      application.ReportMessage(EvoMessageType.Info, "Wykonano cykl pracy skryptu");
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class LoopVB
    Implements IScript

    Dim application As IApplication
    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        While (True)

            DoTask()
            aApplication.Sleep(TimeSpan.FromSeconds(1))

        End While

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

    Sub DoTask()

        application.ReportMessage(EvoMessageType.Info, "Wykonano cykl pracy skryptu")

    End Sub

End Class
```

Główna funkcjonalność skryptu realizowana jest w metodzie *DoTask*. W metodzie *Initialize* znajduje się nieskończona pętla typu *While*, w której metoda *DoTask* jest cyklicznie wywoływana. Pomiędzy kolejnymi wywołaniami następuje 1-sekundowe zawieszenie pracy cyklu. Użyta metoda *Sleep* interfejsu *IApplication* jest zalecaną metodą zawieszania działania skryptu na określony okres czasu.

2.2.2 Wykorzystanie funkcji RegisterTimeEvent

Metoda funkcji *RegisterTimeEvent* interfejsu *IApplication* polega na zadeklarowaniu funkcji zwrotnej, które będzie okresowo wykonywana. Funkcja *RegisterTimeEvent* pozwala na bardzo szczegółowe określenie okresu i momentów wykonywania funkcji zwrotnej.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Timer : IScript
{
   IApplication application;
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      application = aApplication;
      application.RegisterTimeEvent(Report1Hour, TimeSpan.FromHours(1),
                                    TimeSpan.FromMinutes(15));
       return false;
   }

   public void FinalizeScript()
   {

   }

   void Report1Hour()
   {
      application.ReportMessage(EvoMessageType.Info, "Wykonanie raportu o godzinie " +
                                DateTime.Now);
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class TimerVB
    Implements IScript

    Dim application As IApplication
    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
        As IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        application.RegisterTimeEvent(AddressOf Report1Hour, TimeSpan.FromHours(1), _
                                      TimeSpan.FromMinutes(15))
        Return False

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

    Sub Report1Hour()

        application.ReportMessage(EvoMessageType.Info, "Wykonanie raportu o godzinie " _
```

```
                                    & DateTime.Now)
    End Sub

End Class
```

W metodzie *Initialize* rejestrowana jest funkcja zwrotna *Report1Hour.* Zwrócenie wartości *false* powoduje, że skrypt pozostanie załadowany w pamięci.

Drugi parametr metody *RegisterTimeEvent* określa okres czasu pomiędzy kolejnymi wywołaniami funkcji zwrotnej, a parametr trzeci przesunięcie chwili początkowej. Metoda *Report1Hour b*ędzie wywoływana co 1 godzinę, 15 minut po każdej pełnej godzinie.

2.2.3 Tworzenie wątków

W przypadku stosowania metody wielu wątków wykorzystywana jest metoda *CreateThread* interfejsu *IApplication*. Możliwe jest utworzenie wielu wątków, które będą wykonywane niezależnie. Sposób działania wątków oraz ich wzajemna synchronizacja jest całkowicie dowolne – należy stosować metody synchronizacji udostępniane przez platformę .NET.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;


public class ThreadDemo : IScript
{
   IApplication application;
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      application = aApplication;

      System.Threading.Thread t1 = aApplication.CreateThread(ThreadFun);
      t1.Start();

      return false;
   }

   public void FinalizeScript()
   {

   }

   void ThreadFun()
   {
      while (true)
      {
         application.ReportMessage(EvoMessageType.Info, "Wykonano cykl pracy skryptu");
         application.Sleep(TimeSpan.FromSeconds(1));
      }
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class ThreadDemoVB
    Implements IScript

    Dim application As IApplication
    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        Dim t As System.Threading.Thread
        t = aApplication.CreateThread(AddressOf ThreadFun)
        t.Start()
        Return False

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

    Sub ThreadFun()

        While True
            application.ReportMessage(Askom.MessageLogging.MessageType.Warning, _
                                      "Wykonano cykl pracy skryptu")
            application.Sleep(TimeSpan.FromSeconds(1))
        End While

    End Sub

End Class
```

W metodzie *Initialize* tworzony jest nowy wątek roboczy przy pomocy metody *CreateThread*. Jako funkcje wątku podano metodę *ThreadFun*. Po utworzeniu nowy wątek jest uruchamiany. Zwrócenie wartości *false* powoduje, że skrypt pozostanie załadowany w pamięci i wątek będzie mógł kontynuować pracę.

Uruchomienie wątku powoduje wywołanie metody *ThreadFun*. Główna funkcjonalność skryptu realizowana jest wewnątrz nieskończonej pętli typu *While*. Pomiędzy kolejnymi cyklami pracy następuje 1-sekundowa przerwa. Należy unikać tworzenia wątków, które działają bez żadnych przerw, ponieważ powoduje to nadmierne obciążenie pracy procesora.

Metodę niezależnych wątków należy stosować wtedy, gdy istnieje konieczność uruchomienia więcej niż jednego wątku. Pokazany przykład można byłoby prościej zaimplementować, stosując metodę nieskończonej pętli w metodzie *Initialize*.

2.2.4 Monitorowanie zmian zmiennej procesowej

Metoda tworzenia skryptów rezydentnych synchronizowanych zmianami wartości bieżącej zmiennej procesowej, polega na zadeklarowaniu funkcji zwrotnej, która będzie wywoływana przy każdej zmianie stanu zmiennej. Do rejestracji funkcji zwrotnej wykorzystywana jest metoda

RegisterStateEvent interfejsu *IVariable.* Przykładowy kod znajduje się w rozdziale dotyczącym odczytu wartości bieżących zmiennych procesowych (patrz: **2.4. Odczyt wartości bieżących zmiennych procesowych**).

2.2.5 Monitorowanie stanu alarmów

Metoda tworzenia skryptów rezydentnych synchronizowanych zmianami stanu wybranego alarmu, polega na zadeklarowaniu funkcji zwrotnej, która będzie wywoływana przy każdej zmianie stanu alarmu. Do rejestracji funkcji zwrotnej wykorzystywana jest metoda *RegisterStateEvent* interfejsu *IAlarm.* Przykładowy kod znajduje się w rozdziale dotyczącym kontrolowania stanu alarmów (patrz: **2.9. Kontrola stanu alarmów**).

2.3 Odczyt parametrów uruchomienia skryptu

W momencie uruchomienia skryptu do funkcji *Initialize* przekazywana jest tablica *aParameters,* z której można odczytać wartości wszystkich parametrów wywołania. Mechanizm parametrów pozwala na pisanie uniwersalnych skryptów, których szczegóły działania zależą od sposobu wywołania. Typowe przykłady, to przekazanie nazwy zmiennej procesowej, identyfikatora alarmu, nazwy pliku itd..

Istnieją dwie metody odczytu parametrów wywołania. Pierwsza wykorzystuje przeglądanie wszystkich elementów tablicy parametrów. W drugiej stosowany jest dostęp do argumentu poprzez podanie jego numeru kolejnego.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Argumenty : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      int i = 0;
      if (aParameters != null)
      {
         foreach (string p in aParameters)
            aApplication.ReportMessage(EvoMessageType.Info,
                                       String.Format("Parametr {0} = {1}", ++i, p));
      }

      return true;
   }

   public void FinalizeScript()
   {

   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class ArgumentyVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
        As IApplication) As Boolean Implements IScript.Initialize

        Dim i As Integer
        i = 0

        If (Not aParameters Is Nothing) Then
            For Each p As String In aParameters
                i += 1
                aApplication.ReportMessage(EvoMessageType.Info, _
                                           String.Format("Parametr {0} = {1}", i, p))
            Next p
        End If

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Powyższy kod pokazuje sposób iteracji po wszystkich elementach tablicy parametrów *aParameters*.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Argumenty : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      if (aParameters == null || aParameters.Length < 2)
         aApplication.ReportMessage(EvoMessageType.Info, "Zbyt mało parametrów");
      else
      {
         aApplication.ReportMessage(EvoMessageType.Info, "Parametr 1 = " + aParameters[0]);
         aApplication.ReportMessage(EvoMessageType.Info, "Parametr 2 = " + aParameters[1]);
      }

      return true;
   }

   public void FinalizeScript()
   {

   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class ArgumentyVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize


        If (aParameters Is Nothing) Then
            aApplication.ReportMessage(EvoMessageType.Info, "Brak parametrów")
        ElseIf (aParameters.Length < 2) Then
            aApplication.ReportMessage(EvoMessageType.Info, "Zbyt mało parametrów")
        Else
            aApplication.ReportMessage(EvoMessageType.Info, "Parametr 1 = " + aParameters(0))
            aApplication.ReportMessage(EvoMessageType.Info, "Parametr 2 = " + aParameters(1))
        End If

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Powyższy kod pokazuje sposób odczytu parametru przez podanie indeksu elementu w tablicy *aParameters*. Przed próbą odczytu należy koniecznie sprawdzić, czy do skryptu przekazano odpowiednią liczbę parametrów.

2.4 Odczyt wartości bieżących zmiennych procesowych

Odczyt wartości bieżących zmiennych możliwy jest dwoma metodami. Pierwsza z nich to użycie metody *GetVariableValue lub ReadVariables* interfejsu *IApplication*. Druga metoda to odczyt za pomocą obiektów interfejsu *IVariable*. Obiekt *IVariable* dla wybranej zmiennej można uzyskać za pośrednictwem funkcji *GetVariable* interfejsu *IApplication.*

Metoda *GertVariableValue* powinna być wykorzystywana tylko w najprostszych skryptach, w których wartość zmiennej jest odczytywana jednokrotnie. W innych przypadkach należy stosować metodę z obiektem *IVariable* ze względu na jej większą wydajność.

Metoda *ReadVariables* jest podobna do *GetVariableValue*. Pozwala ona odczyt wielu zmiennych jednocześnie I jest z reguły bardziej efektywna niż wielokrotne wywołanie *SetVariableValue*.

Przykład pokazuje najprostszy przypadek, w którym bieżąca wartość zmiennej jest jednokrotnie zapisywana do logu komunikatów.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class CurrValues : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      VariableState v = aApplication.GetVariableValue(aParameters[0], false);
      if (v == null)
         aApplication.ReportMessage(EvoMessageType.Error,
                         string.Format("Zmienna {0} nie jest zdefiniowana", aParameters[0]));
      else
      {
         if (v.IsGood)
         {
            aApplication.ReportMessage(EvoMessageType.Error,
               string.Format("Wartość zmiennej {0}, Stempel czasu: {1}",
                                 v.Value, v.TimeStamp));
         }
         else
            aApplication.ReportMessage(EvoMessageType.Error,
               string.Format("Niepoprawny status wartości {0} ", v.Status));
       }

      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class CurrValuesVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize

        Dim v As VariableState
        v = aApplication.GetVariableValue(aParameters(0), False)
        If v Is Nothing Then
            aApplication.ReportMessage(EvoMessageType.Error, _
                    String.Format("Zmienna {0} nie jest zdefiniowana", aParameters(0)))
        Else
            If (v.IsGood) Then
                aApplication.ReportMessage(EvoMessageType.Error, _
                          String.Format("Wartość zmiennej {0}, Stempel czasu: {1}", _
                          v.Value, v.TimeStamp))
            Else
                 aApplication.ReportMessage(EvoMessageType.Error, _
                 String.Format("Niepoprawny status wartości {0} ", v.Status))
            End If
        End If

        Return True
```

```
    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Pierwszy wariant skryptu używa zalecaną w tym przypadku funkcję *GetVariableValue*. Po uzyskaniu poprawnego obiektu *VariableState* dla zmiennej o nazwie przekazanej w parametrach wywołania, wartość zmiennej jest zapisywana do logu. Aktualna wartość, status i stempel czasu jest dostępna poprzez właściwości obiektu *VariableState*. Należy zwrócić uwagę na sposób weryfikacji poprawności zmiennej poprzez wywołanie metody *IsGood*.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class CurrValues : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      IVariable v = aApplication.GetVariable(aParameters[0], false);
      if (v == null)
         aApplication.ReportMessage(EvoMessageType.Error,
                        string.Format("Zmienna {0} nie jest zdefiniowana", aParameters[0]));
      else
      {
         if (v.VariableState.IsGood)
         {
            aApplication.ReportMessage(EvoMessageType.Error,
               string.Format("Wartość zmiennej {0}, Stempel czasu: {1}",
                                 v.VariableState.Value, v.VariableState.TimeStamp));
         }
         else
            aApplication.ReportMessage(EvoMessageType.Error,
               string.Format("Niepoprawny status wartości {0} ", v.VariableState.Status));

         v.Dispose();
      }

      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class CurrValuesVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
```

```
        IApplication) As Boolean Implements IScript.Initialize

        Dim v As IVariable
        v = aApplication.GetVariable(aParameters(0), False)
        If v Is Nothing Then
            aApplication.ReportMessage(EvoMessageType.Error, _
                    String.Format("Zmienna {0} nie jest zdefiniowana", aParameters(0)))
        Else
            If (v.VariableState.IsGood) Then
                aApplication.ReportMessage(EvoMessageType.Error, _
                          String.Format("Wartość zmiennej {0}, Stempel czasu: {1}", _
                          v.VariableState.Value, v.VariableState.TimeStamp))
            Else
                 aApplication.ReportMessage(EvoMessageType.Error, _
                 String.Format("Niepoprawny status wartości {0} ", v.VariableState.Status))
            End If
            v.Dispose()
        End If

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

W drugim wariancie, dla celów porównawczych użyty jest obiekt klasy *IVariable*. Po uzyskaniu poprawnego obiektu *IVariable* dla zmiennej o nazwie przekazanej w parametrach wywołania, wartość zmiennej jest zapisywana do logu. Aktualna wartość, status i stempel czasu jest dostępna poprzez właściwość *VariableState* obiektu zmiennej. Należy zwrócić uwagę na sposób weryfikacji poprawności zmiennej poprzez wywołanie metody *IsGood*. Po wykonaniu zapisu informacji do logu, obiekt zmiennej jest zwalniany poprzez wywołnie metody *Dispose*.

Wartość zmiennej uzyskana w poprzednich przykładach pochodzi z ostatniego planowego odświeżenia zmiennej. Przy rzadko odświeżanych zmiennych może być ona nieaktualna. Jeżeli skrypt wymaga działania na jak najbardziej aktualnej wartości, należy ustawić drugi parametr funkcji *GetVariableValue* na *true* lub użyć funkcję *Read* interfejsu *IVariable.*

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class CurrValuesRead : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      IVariable v = aApplication.GetVariable(aParameters[0], false);
      if (v == null)
      {
         aApplication.ReportMessage(EvoMessageType.Error,
            string.Format("Zmienna {0} nie jest zdefiniowana", aParameters[0]));
      }
      else
      {
         VariableState vValue = v.Read();
```

```
         if (vValue.IsGood)
         {
            aApplication.ReportMessage(EvoMessageType.Error,
               string.Format("Wartość zmiennej {0}, Stempel czasu: {1}",
               vValue.Value, vValue.TimeStamp));
         }
         else
            aApplication.ReportMessage(EvoMessageType.Error,
               string.Format("Niepoprawny status wartości {0} ", vValue.Status));

         v.Dispose();
      }
       return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class CurrValuesReadVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize

        Dim v As IVariable
        v = aApplication.GetVariable(aParameters(0), False)
        If v Is Nothing Then
            aApplication.ReportMessage(EvoMessageType.Error, _
                  String.Format("Zmienna {0} nie jest zdefiniowana", aParameters(0)))
        Else
            Dim vValue As VariableState
            vValue = v.Read()
            If (vValue.IsGood) Then
                aApplication.ReportMessage(EvoMessageType.Error, _
                       String.Format("Wartość zmiennej {0}, Stempel czasu: {1}", _
                       vValue.Value, vValue.TimeStamp))
            Else
                aApplication.ReportMessage(EvoMessageType.Error, _
                       String.Format("Niepoprawny status wartości {0} ", vValue.Status))
            End If
            v.Dispose()
        End If

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Należy jednak pamiętać o tym, że intensywne wykorzystywanie forsowanego odczytu wartości zmiennej może mocno obciążyć łącze komunikacyjne i pogorszyć działanie całej aplikacji.

2.4.1 Cykliczny odczyt wartości bieżących zmiennych procesowych

Częstym przypadkiem jest konieczność cyklicznego odczytu i przetwarzania wartości zmiennej procesowej. W tego typu przypadkach można wykorzystać mechanizm okresowego wywołania wybranej funkcji.

Poniższy przykładowy skrypt zapisuje do pliku tekstowego wartości bieżące dwóch zmiennych. Zapis realizowany jest co 15 minut.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;
using System.IO;

public class CurrValuesReport : IScript
{
   IApplication application;
   IVariable v1, v2;

   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      application = aApplication;
      v1 = aApplication.GetVariable(aParameters[0], true);
      v2 = aApplication.GetVariable(aParameters[1], true);
      aApplication.RegisterTimeEvent(Report15Min, TimeSpan.FromMinutes(15),
                                     TimeSpan.FromMinutes(0));
      return false;
   }

   public void FinalizeScript()
   {
   }

   void Report15Min()
   {
      StreamWriter writer = new StreamWriter(
         Path.Combine(application.ApplicationWorkingDirectory, "logs\\reports.txt"), true,
         System.Text.ASCIIEncoding.GetEncoding(1250));
      String msg = String.Format("{0},{1},{2}",
         DateTime.Now, v1.VariableState.IsGood ? v1.VariableState.Value : "?",
                       v2.VariableState.IsGood ? v2.VariableState.Value : "?");
      writer.WriteLine(msg);
      writer.Close();
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting
Imports System.IO

Public Class CurrentValuesReportVB
    Implements IScript
```

```
    Dim application As IApplication
    Dim v1, v2 As IVariable
    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        v1 = aApplication.GetVariable(aParameters(0), True)
        v2 = aApplication.GetVariable(aParameters(1), True)
        application.RegisterTimeEvent(Report15Min, TimeSpan.FromMinutes(15), _
                                      TimeSpan.FromMinutes(0))
        Return False

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

    Sub Report15Min()

        Dim writer As StreamWriter
        writer = New StreamWriter( _
           Path.Combine(application.ApplicationWorkingDirectory, "logs\\reports.txt"), True, _
           System.Text.ASCIIEncoding.GetEncoding(1250))
        Dim msg, v1v, v2v As String
        If v1.VariableState.IsGood Then v1v = v1.VariableState.Value Else v1v = "?"
        If v2.VariableState.IsGood Then v2v = v2.VariableState.Value Else v2v = "?"
        msg = String.Format("{0},{1},{2}", DateTime.Now, v1v, v2v)
        writer.WriteLine(msg)
        writer.Close()

    End Sub

End Class
```

W funkcji *Initialize* tworzone są dwa obiekty *IVariable*, które będą służyły do odczytu wartości zmiennych. Istotne jest ustawienie drugiego parametru funkcji *GetVariable* na *True*. Powoduje to włączenia odświeżania zmiennej procesowej – brak odświeżania mógłby spowodować, że w momencie pobrania wartości zmiennej w funkcji okresowej, wartość ta byłaby nieaktualna.

Funkcja *RegisterTimeEvent* rejestruje funkcję okresową *Report15Min.* Funkcja będzie wykonywana co 15 minut, w minucie 0, 15, 30 i 45.

Funkcja *Initialize* zwraca wartość *False*, co powoduje kontynuację pracy skryptu.

Wewnątrz funkcji *Report15Min* wykonywany jest zapis danych do pliku – wartości zmiennych są pobierane w standardowy sposób, za pośrednictwem właściwości *VariableState*.

2.4.2 Odczyt wartości bieżących sterowany odświeżaniem zmiennej

Inną metodą cyklicznego przetwarzania wartości zmiennych jest wykorzystanie funkcji *RegisterStateEvent,* która powoduje zarejestrowanie funkcji zwrotnej wywoływanej natychmiast po każdorazowym odświeżeniu wartości zmiennej. Pozwala to na natychmiastową reakcję na zmianę stanu zmiennej. Metoda funkcji zwrotnej wykorzystywana jest w takich scenariuszach jak automatyczne wyliczanie wartości pochodnych lub rozpoznawanie sytuacji alarmowych.

Przykład skryptu wykorzystującego funkcję *RegisterStateEvent* jest pokazany w rozdziale dotyczącym ustawiania wartości zmiennych (patrz: **2.5. Ustawianie wartości zmiennych procesowych**).

2.5 Ustawianie wartości zmiennych procesowych

Podobnie jak w przypadku odczytu wartości zmiennych, ustawienie wartości można wykonać dwoma metodami. W prostych przypadkach, jednokrotnego zapisu wartości, należy stosować metodę *SetVariableValue*, *SetVariableBits* lub *ToggleBits* interfejsu *IApplication*. Bardziej uniwersalnym sposobem działania jest wykorzystanie obiektów interfejsu *IVariable*. Wartość zmiennej można zmienić przy pomocy metod *SetVariableValue*, *SetVariableBits* i *ToggleBits* tego interfejsu. Metoda SetVariableBits pozwala na dowolną zmianę pojedynczych bitów wartości zmiennej, a metoda *ToggleBits* na zmianę wartości wybranych bitów na przeciwną.

Interfejs *IApplication* posiada także metodę *SetVariables*, która przy pomocy jednego wywołania pozwala ustawić wartości wielu zmiennych - użycie tej metody jest z reguły szybsze niż wielokrotne wywołanie *SetVariableValue*.

Wszytkie powyższe funkcje działają w sposób asynchroniczny, czyli nie oczekują na rzeczywiste wykonania sterownia (wysłanie nastawy do PLC). Wyjątkiem jest zapis do zmiennych wirtualnych w kanale typu None. Jeżeli konieczna jest kontrola rzeczywistego wykonania sterowania, to można użyć metody *SetVariablesSync* interfejsu *IApplication*, lub metod *SetVariableValueSync*, *SetVariableBitsSync* i *ToggleBitsSync* interfejsu *IVariable*.

2.5.1 Wyliczanie wartości pochodnych zmiennych

Poniższy skrypt pokazuje, w jaki sposób na podstawie wartości dwóch zwykłych zmiennych wyliczana jest wartość trzeciej zmiennej. Skrypt pokazuje także zastosowanie funkcji zwrotnej odświeżania do monitorowania wartości zmiennych wejściowych.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class VarCalc : IScript
{
```

```
   IApplication application;
   IVariable vIn1, vIn2, vOut;

   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      application = aApplication;
      vIn1 = aApplication.GetVariable(aParameters[0], true);
      vIn2 = aApplication.GetVariable(aParameters[1], true);
      vOut = aApplication.GetVariable(aParameters[2], false);
      vIn1.RegisterStateEvent(RefreshedEvent);
      vIn2.RegisterStateEvent(RefreshedEvent);
      return false;
   }

   public void FinalizeScript()
   {

   }

   void RefreshedEvent(string aVariableName, VariableState aVariableState)
   {
      try
      {
         if (vIn1.VariableState.IsGood && vIn2.VariableState.IsGood)
         {
            vOut.SetVariableValue(Recalculate(Convert.ToDouble(vIn1.VariableState.Value),
               Convert.ToDouble(vIn2.VariableState.Value)));
         }
      }
      catch (Exception e)
      {
         application.ReportMessage(EvoMessageType.Error, e.Message);

      }
   }

   double Recalculate(double v1, double v2)
   {
      return v1 + v2;
   }

}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class VarCalcVB
    Implements IScript

    Dim application As IApplication
    Dim vIn1, vIn2, vOut As IVariable

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                               IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        vIn1 = aApplication.GetVariable(aParameters(0), True)
        vIn2 = aApplication.GetVariable(aParameters(1), True)
        vOut = aApplication.GetVariable(aParameters(2), False)
        vIn1.RegisterStateEvent(AddressOf RefreshedEvent)
        vIn2.RegisterStateEvent(AddressOf RefreshedEvent)
        Return False
```

```
    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

    Sub RefreshedEvent(ByVal aVariableName As String, ByVal aVariableState As VariableState)

        Try
            If (vIn1.VariableState.IsGood And vIn2.VariableState.IsGood) Then

                vOut.SetVariableValue(
                       Recalculate(Convert.ToDouble(vIn1.VariableState.Value), _
                                   Convert.ToDouble(vIn2.VariableState.Value)))
            End If
        Catch e As Exception
            application.ReportMessage(EvoMessageType.Error, e.Message)
        End Try

    End Sub


    Function Recalculate(ByVal v1 As Double, ByVal v2 As Double) As Double

        Recalculate = v1 + v2

    End Function

End Class
```

W funkcji *Initialize* tworzone są dwa obiekty *IVariable*, które będą służyły do odczytu wartości zmiennych oraz jeden obiekt *IVariable* zmiennej wynikowej. Istotne jest ustawienie drugiego parametru funkcji *GetVariable* na *True* dla obu zmiennych wejściowych. Następnie dla zmiennych wejściowych, przy pomocy metody *registerStateEvent*, deklarowana jest funkcja zwrotna *RefresehedEvent*. Będzie ona wywoływana przy każdej zmianie wartości którejkolwiek ze zmiennych wejściowych.

Funkcja *Initialize* zwraca wartość *false*, co powoduje kontynuację pracy skryptu.

Wewnątrz funkcji R*efreshedEvent* następuje wyliczenie nowej wartości wynikowej, a następnie zmienna wynikowa jest ustawiana przy pomocy funkcji *SetVariableValue*. Parametry funkcji zwrotnej *aVariableName* i *aVariableState* mogą być użyte do określenia, która zmienna uległa zmianie i na jaką wartość, ale w przykładowym skrypcie nie było to potrzebne.

Należy zwrócić uwagę na sposób zabezpieczenia kodu przy pomocy konstrukcji *try*. Kod jest odporny na brak lub nieprawidłowe wartości zmiennych wejściowych.

2.5.2 Pozyskiwanie wartości ze źródeł zewnętrznych

Poniższy skrypt okresowo wykonuje operację ping do określonego urządzenia podanego adresem lub nazwą. Na podstawie wyniku ustawia wartość wskazanej zmiennej na *true* lub *false*. Zmienna wynikowa powinna znajdować się kanale typu *None*.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;
using System.Net.NetworkInformation;

public class Ping : IScript
{
   IEventInfo mTimeTickInfo;
   System.Net.NetworkInformation.Ping mPing;
   IVariable mPingVar;
   string mHost;

   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      mHost = aParameters[0];

      mPing = new System.Net.NetworkInformation.Ping();
      mPingVar = aApplication.GetVariable(aParameters[1], false);

      if (mPingVar != null && mPing != null)
      {
         mTimeTickInfo = aApplication.RegisterTimeEvent(PingCheck, TimeSpan.FromSeconds(5),
                                                        TimeSpan.MinValue);
         return false;
      }
      else
         return true;
   }

   public void FinalizeScript()
   {
   }

   void PingCheck()
   {
      System.Net.NetworkInformation.PingReply reply = mPing.Send(mHost, 500);
      mPingVar.SetVariableValue(reply.Status==System.Net.NetworkInformation.IPStatus.Success);
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class PingVB
    Implements IScript

    Dim mTimeTickInfo As IEventInfo
    Dim mPing As System.Net.NetworkInformation.Ping
    Dim mPingVar As IVariable
    Dim mHost As String

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        mHost = aParameters(0)

        mPing = New System.Net.NetworkInformation.Ping()
        mPingVar = aApplication.GetVariable(aParameters(1), False)

        If (mPingVar Is Nothing Or mPing Is Nothing) Then
            Return True
```

```
        Else
            mTimeTickInfo = aApplication.RegisterTimeEvent(AddressOf PingCheck, _
                                                    TimeSpan.FromSeconds(5), TimeSpan.MinValue)
            Return False
        End If

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

    Sub PingCheck()

        Dim reply As System.Net.NetworkInformation.PingReply

        reply = mPing.Send(mHost, 500)
        mPingVar.SetVariableValue(reply.Status=System.Net.NetworkInformation.IPStatus.Success)

    End Sub

End Class
```

Funkcja *PingCheck* wykonywana jest co 5 sekund. Na podstawie rezultatu funkcji *Send* obiektu klasy *System.Net.NetworkInformation.Ping* ustawiana jest wartość zmiennej procesowej.

2.6 Odczyt wartości archiwalnych zmiennych procesowych

Oprócz dostępu do wartości bieżących zmiennych, system skryptów pozwala także na odczyt zgromadzonych wartości archiwalnych. Typowe zastosowanie to generacja raportów oraz wyliczanie wskaźników produkcyjnych bazujących na danych archiwalnych.

Odczyt danych archiwalnych może być wykonany za pośrednictwem dwóch mechanizmów. Bardziej uniwersalne jest wykorzystanie interfejsu *IVariable*. Metoda ta pozwala na bardziej precyzyjną kontrolę procesu odczytu. W prostych przypadkach można wykorzystać metodę *GetVariableArcData* interfejsu *IApplication*.

W poniższym przykładzie pokazano, w jaki sposób utworzyć prosty raport w formacie HTML, korzystając z interfejsu *IVariable*. Przykład użycia metody *GetVariableArcData* znajduje się w opisie tej metody. Sposób wykorzystania odczytanych danych jest identyczny dla obu mechanizmów.

C#

```
using System;
using System.IO;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class RaportHtml : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
       IVariable v = aApplication.GetVariable(aParameters[0], false);
       if(v == null)
```

```
      {
         aApplication.ReportMessage( EvoMessageType.Error,
                         "Błąd generowania raportu: Brak zmiennej " + aParameters[0]);
         return true;
      }

      DateTime day = DateTime.Today;
      DateTime day0 = day.AddDays(-1);
      ArcQuery q = new ArcQuery(day0, day, "average", TimeSpan.FromHours(1));

      IArcQueryResult res = v.OrderArcData(q);

      int loopCnt = 0;
      while(res.ResultStatus == ArcResultStatus.Processing)
      {
         if(loopCnt > 200)
         {
            aApplication.ReportMessage(EvoMessageType.Error,
                  "Błąd generowania raportu! Przekroczono czas oczekiwania na odpowiedź.");
            v.Dispose();
            return true;
         }

         aApplication.Sleep(TimeSpan.FromMilliseconds(100));
         loopCnt++;
      }

      if(res.ResultStatus == ArcResultStatus.Ready)
         {
            StreamWriter sw = null;
            System.Text.StringBuilder sb = new System.Text.StringBuilder();
            try
            {
                sb.AppendLine(
"<html><head><title>Raport - Srednie godzinowe z poprzedniego dnia</title></head><body>");
             sb.AppendLine("<table width=\"25%\" align=\"center\" border=\"1\" >");
               sb.AppendLine("<tr><th>Godzina</th><th>Srednia</th></tr>");

               if(res.Values != null)
               {
                   foreach(ArcVariableState p in res.Values)
                   {
                        sb.AppendLine(
                "<tr><td>" + p.TimeStamp.ToString() + "</td><td>" + p.Value + "</td></tr>");
                }
               }
               sb.AppendLine("</table>");
               sb.AppendLine("</body></html>");

                string file = Path.Combine(aApplication.ApplicationWorkingDirectory,
                                           "reports\\report.html");
               sw = new StreamWriter(file);
               sw.Write(sb.ToString());
               System.Diagnostics.Process.Start(file);
            }
            catch(Exception ex)
            {
               aApplication.ReportMessage(EvoMessageType.Error,
                                                     "Błąd zapisu pliku: " + ex.Message);
            }
            finally
            {
               if(sw != null)
               {
                   sw.Close();
                   sw.Dispose();
               }
```

```
            }
         }
         else if(res.ResultStatus == ArcResultStatus.Error)
            aApplication.ReportMessage(EvoMessageType.Error,
                                       "Błąd generowania raportu! " + res.ErrorDescription);

      v.Dispose();
      return true;
   }

   public void FinalizeScript()
   {
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class RaportHtmlVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        Dim v As IVariable
        v = aApplication.GetVariable(aParameters(0), False)
        If (v Is Nothing) Then
            aApplication.ReportMessage(EvoMessageType.Error, _
                               "Błąd generowania raportu: Brak zmiennej " & aParameters(0))
            Return True
        End If

        Dim day, day0 As DateTime
        day = DateTime.Today
        day0 = day.AddDays(-1)
        Dim q As ArcQuery
        q = New ArcQuery(day0, day, "average", TimeSpan.FromHours(1))

        Dim res As IArcQueryResult
        res = v.OrderArcData(q)

        Dim loopCnt As Integer
        loopCnt = 0
        While (res.ResultStatus = ArcResultStatus.Processing)

            If (loopCnt > 200) Then
                aApplication.ReportMessage(EvoMessageType.Error, _
                 "Błąd generowania raportu! Przekroczono czas oczekiwania na odpowiedź.")
                v.Dispose()
                Return True
            End If

            aApplication.Sleep(TimeSpan.FromMilliseconds(100))
            loopCnt += 1

        End While

        If (res.ResultStatus = ArcResultStatus.Ready) Then

            Dim sw As System.IO.StreamWriter
            sw = Nothing
            Dim sb As System.Text.StringBuilder
```

```
            sb = New System.Text.StringBuilder()
            Try
                sb.AppendLine( _
    "<html><head><title>Raport - Srednie godzinowe z poprzedniego dnia</title></head><body>")
                sb.AppendLine("<table width=""25%"" align=""center"" border=""1"" >")
                sb.AppendLine("<tr><th>Godzina</th><th>Srednia</th></tr>")

                If (Not res.Values Is Nothing) Then

                    For Each p As ArcVariableState In res.Values
                        sb.AppendLine(
_                   "<tr><td>" & p.TimeStamp.ToString() & "</td><td>" & p.Value & "</td></tr>")
                    Next p
                    sb.AppendLine("</table>")
                    sb.AppendLine("</body></html>")

                    Dim file As String
                    file = System.IO.Path.Combine( _
                            aApplication.ApplicationWorkingDirectory, "reports\\report.html")
                    sw = New System.IO.StreamWriter(file)
                    sw.Write(sb.ToString())
                    System.Diagnostics.Process.Start(file)

                End If

            Catch ex As Exception
                aApplication.ReportMessage(EvoMessageType.Error, _
                                           "Błąd zapisu pliku: " & ex.Message)
            Finally
                If (Not sw Is Nothing) Then
                    sw.Close()
                    sw.Dispose()
                End If
            End Try
        ElseIf (res.ResultStatus = ArcResultStatus.Error) Then
            aApplication.ReportMessage(EvoMessageType.Error, _
                                       "Błąd generowania raportu! " & res.ErrorDescription)
        End If

        v.Dispose()
        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Odczyt wartości archiwalnych inicjowany jest wywołaniem funkcji *OrderArcData* interfejsu *IVariable*. Parametrem wywołania jest obiekt klasy *ArcQuery* określający, jakie dane i z jakiego okresu mają zostać pobrane. W przykładzie dane pobierane są z okresu poprzedniego dnia, a zwracane wartości to średnie 1-godzinowe.

Po zainicjowaniu odczytu należy zaczekać na koniec operacji poprzez testowanie statusu wykonania operacji *ResultStatus* (alternatywną metodą oczekiwania jest zadeklarowanie funkcji zwrotnej wywoływanej w momencie zakończenia odczytu). Po stwierdzeniu, że operacja zakończyła się poprawnie skrypt wykonuje generację raportu wynikowego. Wartości archiwalne zwracane są w kolekcji *Values* dostępnej poprzez obiekt statusu wykonania. Kolekcja składa się z obiektów

ArcVariableState - każdy obiekt określa wartość, status i stempel czasu pojedynczej wartości archiwalnej.

Po wygenerowaniu raportu, wywołanie metody *Start* obiektu *Process* powoduje otwarcie raportu w oknie domyślnej przeglądarki www.

2.7 Użycie zmiennych tablicowych

Sposób użycia zmiennych tablicowych niewiele różni się od obsługi zwykłych zmiennych. Inny jest tylko sposób interpretacji właściwości *Value* obiektu *VariableState.*

2.7.1 Odczyt zmiennych tablicowych

Poniższy przykład zapisuje do logu komunikatów wartości wszystkich elementów zmiennej tablicowej o nazwie podanej w parametrach uruchomienia.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class TableGet : IScript
{
    public bool Initialize(string[] aParameters, IApplication aApplication)
    {
        VariableState v = aApplication.GetVariableValue(aParameters[0], true);
        int[] valTable = (int[])(v.Value);

        System.Text.StringBuilder sb = new System.Text.StringBuilder();
        foreach (int val in valTable)
        {
            sb.Append(val);
            sb.Append("  ");
        }
        aApplication.ReportMessage(EvoMessageType.Info, sb.ToString());

        return true;
    }

    public void FinalizeScript()
    {
    }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class TableGetVB
```

```
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                                    As IApplication) As Boolean Implements IScript.Initialize

        Dim v As VariableState = aApplication.GetVariableValue(aParameters(0), True)
        Dim valTable() As Integer = v.Value

        Dim sb As System.Text.StringBuilder = New System.Text.StringBuilder()
        For Each val As Integer In valTable
            sb.Append(val)
            sb.Append("  ")
        Next val

        aApplication.ReportMessage(EvoMessageType.Info, sb.ToString())

        Return True
    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Kluczowym elementem skryptu jest podstawienie do zmiennej lokalnej *ValTable* właściwości *Value* obiektu stanu zmiennej procesowej połączone z odpowiednią deklaracją typu wartości. Istotne jest zachowanie zgodności typu zmiennej *valTable* z typem zmiennej procesowej wynikającym z jej definicji i sposobu pozyskania.

Oprócz pokazanego w przykładzie iterowania po wszystkich elementach tablicy, możliwy jest także bezpośredni odczyt dowolnego elementu poprzez użycie notacji indeksowej, np. *valTable[5] (C#)* lub *valTable(5) (VB)*.

2.7.2 Zapis zmiennych tablicowych

Następny przykład pokazuje sposób wykorzystania zmiennych tablicowych do przechowywania wielu informacji w pojedynczej zmiennej. Skrypt odczytuje zawartość pliku tekstowego zawierającego komunikaty, które zostaną wyświetlone na diagramie. Odczytywanych jest pierwsze pięć linii pliku. Odczytane teksty zapisywane są do kolejnych elementów zmiennej procesowej.

Zakłada się, że wykorzystywana zmienna znajduje się w kanale typu *NONE*, a jej atrybut *Typ wartości* jest równy *Table*.

C#

```
using System;
using System.IO;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Messages : IScript
{
    public bool Initialize(string[] aParameters, IApplication aApplication)
    {
```

```
        try
        {

            StreamReader file = new StreamReader(aParameters[0],System.Text.Encoding.Default);
            string[] msgs = new string[5];
            msgs[0] = file.ReadLine();
            msgs[1] = file.ReadLine();
            msgs[2] = file.ReadLine();
            msgs[3] = file.ReadLine();
            msgs[4] = file.ReadLine();

            aApplication.SetVariableValue(aParameters[1], msgs);
            file.Close();
        }
        catch (Exception e)
        {
            aApplication.ReportMessage(EvoMessageType.Info,"Error in Messages: " + e.Message);
        }
        return true;
    }

    public void FinalizeScript()
    {
    }
}
```

VB

```
Imports System
Imports System.IO
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class MessagesVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                                    As IApplication) As Boolean Implements IScript.Initialize

        Try
            Dim file As StreamReader = _
                            New StreamReader(aParameters(0), System.Text.Encoding.Default)
            Dim msgs(4) As String
            msgs(0) = file.ReadLine()
            msgs(1) = file.ReadLine()
            msgs(2) = file.ReadLine()
            msgs(3) = file.ReadLine()
            msgs(4) = file.ReadLine()

            aApplication.SetVariableValue(aParameters(1),msgs)
            file.Close()
        Catch e As Exception
            aApplication.ReportMessage(EvoMessageType.Info,"Error in Messages: " + e.Message)
        End Try

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

End Class
```

W skrypcie tworzona jest lokalna 5-elementowa zmienna tablicowa. Następnie na podstawie zawartości pliku tekstowego ustawiane są jej kolejne elementu. Działanie skryptu kończy standardowe wywołanie metody *SetVariableValue* interfejsu *IApplication*.

W obiektach diagramu można odwołać się do elementu zmiennej tablicowej przy pomocy funkcji *TableElement*, np. *TableElement(Variable(zmienna),2)*.

2.8 Transfer przebiegów wzorcowych

Typowym zadaniem w przypadku aplikacji wykorzystujących przebiegi wzorcowe jest transfer definicji przebiegu do sterownika. Można to wykonać sekwencją operacji wysyłających kolejne punkty poprzez pojedynczą zmienną, ale z reguły lepszym wyjściem jest wysłanie całej definicji pojedynczą transmisją z wykorzystaniem zmiennej tablicowej.

Do odczytu przebiegów wzorcowych wykorzystywana jest metoda *GetTrendPatternPoints* interfejsu *IApplication*.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Curve : IScript
{
    public bool Initialize(string[] aParameters, IApplication aApplication)
    {
        IVariable v = aApplication.GetVariable(aParameters[1], false);
        int varSize = Convert.ToInt32(v.VariableAttribute("ElementsCount"));

        int[] transferBuffer = new int[varSize];
        int pointsCount = 0;
        foreach (TrendPatternPoint point in
                                    aApplication.GetTrendPatternPoints(aParameters[0]))
        {
            if (pointsCount <= varSize - 2)
            {
                transferBuffer[pointsCount] = (int)point.TimeStamp.TotalSeconds;
                transferBuffer[pointsCount + 1] = (int)point.Value;
                pointsCount += 2;
            }
            else
            {
                if (pointsCount <= varSize)
                    transferBuffer[pointsCount] = 0;
                break;
            }
        }
        v.SetVariableValue(transferBuffer);

        v.Dispose();
        return true;
    }

    public void FinalizeScript()
    {
    }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class CurveVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                                    As IApplication) As Boolean Implements IScript.Initialize

        Dim v As IVariable = aApplication.GetVariable(aParameters(1), False)
        Dim varSize As Integer = Convert.ToInt32(v.VariableAttribute("ElementsCount"))
        Dim pointsCount As Integer = 0

        Dim transferBuffer() As Integer = New Integer(varSize) {}

        For Each point As TrendPatternPoint In _
                                        aApplication.GetTrendPatternPoints(aParameters(0))

            If pointsCount <= varSize - 2 Then

                transferBuffer(pointsCount) = CInt(point.TimeStamp.TotalSeconds)
                transferBuffer(pointsCount + 1) = CInt(point.Value)
                pointsCount += 2

            Else

                If (pointsCount <= varSize) Then transferBuffer(pointsCount) = 0
                Exit For

            End If
        Next point

        v.SetVariableValue(transferBuffer)

        v.Dispose()
        Return True
    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

End Class
```

Pierwszą operacją w skrypcie jest odczyt atrybutu *ElementsCount* zmiennej docelowej w celu określenia wielkości dostępnego bufora. Zakłada się, że typem elementów zmiennej jest 32-bitowa liczba całkowita. Dodatkowo tworzona jest tablica *transferBuffer*, o rozmiarze zgodnym z wielkością zmiennej procesowej.

Następnie kolejne punkty przebiegu wzorcowego (pobrane metodą *GetTrendPatternPoints*) są zapisywane do kolejnych elementów tablicy *transferBuffer,* parami przesunięcie czasowe i wartość punktu. Jeżeli przebieg nie wypełnia całej tablicy, to dodawana jest pojedyncza wartość zerowa oznaczająca koniec przebiegu.

Działanie skryptu kończy standardowe wywołanie metody *SetVariableValue*.

2.9 Kontrola stanu alarmów

Usługi modelu obiektowego skryptu pozwalają na pełną kontrolę pracy systemu alarmów aplikacji. Dotyczy to zarówno monitorowania stanu alarmów, jak i programowego wykrywania zdarzeń alarmowych.

2.9.1 Odczyt stanu alarmów aktywnych

Dostęp do aktualnego stanu systemu alarmów zapewnia grupa metod interfejsu *IApplication*. Podstawowe funkcje to: *GetActiveAlarms* zwracająca kolekcję identyfikatorów alarmów znajdujących się aktualnie w logu alarmów aktywnych, *GetDefinedAlarms* pozwalająca na dostęp do wszystkich zdefiniowanych alarmów, oraz *GetAlarm* zwracająca referencją do interfejsu *IAlarm*. Właściwości i metody tego interfejsu pozwalają na odczyt wszystkich danych dotyczących konkretnego alarmu – jego definicji i aktualnego stanu.

Przedstawiony przykład pokazuje sposób stworzenia prostego raportu, w którym zapisywane są podstawowe informacje o wszystkich alarmach aktywnych w momencie wykonania skryptu.

Skrypt wymaga podania dwóch parametrów: nazwy domeny i nazwy pliku wynikowego.

C#

```
using System;
using System.IO;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class AlarmReport : IScript
{
    public bool Initialize(string[] aParameters, IApplication aApplication)
    {
        try
        {
            StreamWriter sw = new StreamWriter
                                    (aParameters[1], false, System.Text.Encoding.UTF8);
            foreach (string aid in aApplication.GetActiveAlarms(aParameters[0]))
            {
                IAlarm alarm = aApplication.GetAlarm(aid);
                if (alarm.AlarmState.IsActive && !alarm.AlarmState.IsFinished)
                {
                    String desc = String.Format
                        ("{0:dd-MM-yyyy hh:mm:ss.fff} {1,10} {2,-40} {3}",
                        alarm.AlarmState.BeginTimeStamp, alarm.Id,
                        alarm.AlarmState.FormattedBeginText, alarm.AlarmState.BeginExprValue);
                    sw.WriteLine(desc);
                }
            }
            sw.Close();
        }
        catch (Exception)
        {
        }
```

```
        return true;
    }

    public void FinalizeScript()
    {
    }
}
```

VB

```
Imports System
Imports System.IO
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class AlarmReportVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                                    As IApplication) As Boolean Implements IScript.Initialize

        Try
            Dim sw As StreamWriter = New StreamWriter _
                                        (aParameters(1), False, System.Text.Encoding.UTF8)
            For Each aid As String In aApplication.GetActiveAlarms(aParameters(0))

                Dim alarm As IAlarm = aApplication.GetAlarm(aid)
                If alarm.AlarmState.IsActive And Not alarm.AlarmState.IsFinished Then

                    Dim desc As String = String.Format _
                        ("{0:dd-MM-yyyy hh:mm:ss.fff} {1,10} {2,-40} {3}", _
                        alarm.AlarmState.BeginTimeStamp, alarm.Id, _
                        alarm.AlarmState.FormattedBeginText, alarm.AlarmState.BeginExprValue)
                    sw.WriteLine(desc)

                End If

            Next aid
            sw.Close()
        Catch a As Exception
        End Try

        Return True
    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Funkcja *GetActiveAlarms* zwraca kolekcję identyfikatorów wszystkich alarmów z wybranej domeny, które znajdują się w danym momencie w logu alarmów aktywnych. Następnie dla każdego z tych alarmów, funkcją *GetAlarm* pobierana jest referencja do obiektu *IAlarm*. Poprzez tę referencję pobierane są wszystkie dane o alarmie. W instrukcji *if* sprawdzane jest, czy alarm jest aktywny. Test *AlarmState.IsActive* sprawdza, czy alarm jest w logu alarmów aktywnych. Druga część testu używa właściwości *AlarmState.IsFinished* w celu określenia, czy alarm nie jest już zakończony – w logu alarmów aktywnych mogą być przez pewien czas przechowywane dane o alarmie niedawno zakończonym. Po stwierdzeniu, że alarm jest na pewno aktywny, wybrane informacje o alarmie są zapisywane do pliku.

2.9.2 Aktywne monitorowanie stanu alarmów

Oprócz pokazanego w poprzednim przykładzie okresowego odczytywania stanu alarmów, możliwe jest również monitorowanie aktywne. W tym wariancie można natychmiast zareagować na pojawienie się alarmu lub inną zmianę jego stanu. Funkcję monitorującą stan konkretnego alarmu można zadeklarować przy pomocy metody *RegisterStateEvent* interfejsu *IAlarm*. Przykładowy kod znajduje się w opisie metody *RegisterStateEvent*. (Patrz: **RegisterStateEvent**). Możliwe jest też monitorowanie stanu całej domeny alarmów. W tym celu należy użyć metodę *RegisterAlarmDomainEvents* interfejsu *IApplication*.

2.9.3 Zgłaszanie alarmów

Skrypty mogą realizować pełną funkcjonalność strategii wykrywania alarmów. Przykład z tego rozdziału realizuje taką strategię. Jej działanie oparte jest na cyklicznym odczycie pliku tekstowego, w którym zapisywane są (przez zewnętrzny program kontrolujący stan systemu) tekstowe identyfikatory alarmów rozdzielone znakami średnika. Obecność identyfikatora w pliku oznacza, że alarm jest aktywny. Zadaniem skryptu jest przekazanie informacji z pliku do systemu alarmów aplikacji. Wstępnym krokiem, który należy wykonać jest zdefiniowanie strategii alarmowej typu „Zewnętrzna” oraz definicji alarmów należących do tej strategii. W definicji każdego z alarmów, w atrybucie „Parametry wykrywania”, należy podać identyfikator, który w pliku tekstowym będzie odpowiadał temu alarmowi.

Przykładowy skrypt w momencie uruchomienia wymaga podania trzech parametrów: nazwy pliku tekstowego, nazwy domeny alarmowej oraz nazwy strategii, na konto której skrypt będzie działał. Uruchomienie skryptu należy zlecić w terminarzu aplikacji przy pomocy akcji *Script.* Akcję należy uruchamiać w trybie „Jednokrotnie przy starcie aplikacji”.

C#

```
using System;
using System.IO;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;
using System.Collections.Generic;

public class AlarmStrat : IScript
{
    IApplication application;
    string alarmStateFile;
    string domain;
    string strategy;
    List<IAlarm> myAlarms = new List<IAlarm>();

    public bool Initialize(string[] aParameters, IApplication aApplication)
    {
        application = aApplication;
        alarmStateFile = aParameters[0];
        domain = aParameters[1];
        strategy = aParameters[2];

        foreach (string name in application.GetDefinedAlarms(domain))
        {
```

```
            IAlarm ia = application.GetAlarm(domain, name);
            if (ia.Definition.Strategy == strategy)
                myAlarms.Add(ia);
        }
        aApplication.RegisterTimeEvent(StrategyFun, TimeSpan.FromSeconds(10),
                                       TimeSpan.FromSeconds(0));

        return false;
    }

    public void FinalizeScript()
    {
    }

    char[] separator = { ';' };

    void StrategyFun()
    {
        if (application.GetAlarmDomainStationMode(domain)!=AlarmStationMode.ActiveController)
            return;

        try
        {
            StreamReader file = new StreamReader(alarmStateFile,System.Text.Encoding.Default);
            List<string> content = new List<string>(file.ReadToEnd().Split(separator));
            foreach (IAlarm alarm in myAlarms)
            {
                bool isOn = content.IndexOf(alarm.Definition.DetectionParameters) != -1;
                if (isOn)
                {
                    if (!alarm.AlarmState.IsActive || alarm.AlarmState.IsFinished)
                    {
                        alarm.Raise();
                    }
                }
                else
                {
                    if (alarm.AlarmState.IsActive && !alarm.AlarmState.IsFinished)
                    {
                        alarm.Cancel();
                    }
                }
            }
            file.Close();
        }
        catch (Exception e)
        {
            application.ReportMessage(EvoMessageType.Info, "Error in AlarmStrat: "+e.Message);
        }
    }
}
```

VB

```
Imports System
Imports System.IO
Imports System.Windows.Forms
Imports System.Collections.Generic
Imports Askom.AsixEvo.Scripting

Public Class AlarmStratVb
    Implements IScript

    Dim application As IApplication
    Dim alarmStateFile As String
    Dim domain As String
```

```
    Dim strategy As String
    Dim myAlarms As List(Of IAlarm) = New List(Of IAlarm)

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                               As IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        alarmStateFile = aParameters(0)
        domain = aParameters(1)
        strategy = aParameters(2)
        Dim ia As IAlarm

        For Each name As String In application.GetDefinedAlarms(domain)

            ia = application.GetAlarm(domain, name)
            If (ia.Definition.Strategy = strategy) Then
                myAlarms.Add(ia)
            End If

        Next name

        aApplication.RegisterTimeEvent(AddressOf StrategyFun, TimeSpan.FromSeconds(10),_
                                       TimeSpan.FromSeconds(0))
        Return False

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

    Dim separator() As Char = {";"}

    Public Sub StrategyFun()

        If application.GetAlarmDomainStationMode(domain) <> _
                                                    AlarmStationMode.ActiveController Then
            Return
        End If

        Try
            Dim file As StreamReader = New StreamReader(alarmStateFile, _
                                                        System.Text.Encoding.Default)
            Dim content As List(Of String) = New List(Of String) _
                                                        (file.ReadToEnd().Split(separator))
            For Each alarm As IAlarm In myAlarms

                Dim isOn As Boolean = _
                              Not content.IndexOf(alarm.Definition.DetectionParameters) = -1
                If isOn Then
                    If (Not alarm.AlarmState.IsActive) Or alarm.AlarmState.IsFinished Then
                        alarm.Raise()
                    End If
                Else
                    If alarm.AlarmState.IsActive And (Not alarm.AlarmState.IsFinished) Then
                        alarm.Cancel()
                    End If
                End If
            Next alarm
            file.Close()

        Catch e As Exception
            application.ReportMessage(EvoMessageType.Info, "Error in AlarmStrat: "+e.Message)
        End Try

    End Sub

End Class
```

W funkcji *Initialize* tworzona jest lista wszystkich alarmów należących do wskazanej parametrami domeny i strategii alarmowej. Funkcja *GetDefinedAlarms* zwraca wszystkie alarmy domeny. Następnie dla każdego alarmu domeny pobierana jest referencja do interfejsu *IAlarm*, a poprzez nią sprawdzane jest, czy alarm należy do właściwej strategii. Po odczycie danych konfiguracyjnych rejestrowana jest funkcja *StrategyFun* – będzie ona wykonywana co 10 sekund.

Funkcja *StrategyFun* sprawdza na początku tryb pracy domeny. Alarmy są wykrywane tylko wtedy, gdy stanowisko jest kontrolerem aktywnym domeny. Następnie odczytywany jest plik tekstowy i parsowana lista identyfikatorów. Dla każdego alarmu strategii określany jest aktualny stan alarmu wg danych odczytanych z pliku oraz status w aplikacji. W przypadku wykrycia niezgodności alarm jest zgłaszany przy pomocy metody *Raise* lub wygaszany metodą *Cancel*. Obie metody pozwalają potencjalnie na zgłoszenie dodatkowych danych związanych ze zdarzeniem. Dane te byłyby zapisane w logu alarmów. Możliwe jest też jawne podanie czasu zdarzenia – w przykładowym skrypcie czas zdarzenia jest identyczny z czasem wykonania funkcji *Raise* lub *Cancel*.

Wykrywanie aktualnego stanu alarmu wykonywane jest zgodnie z opisem podanym w przykładzie dotyczącym odczytu stanu alarmów.

2.10 Użycie kontekstu uruchomienia skryptu

Jedną z metod uruchomienia skryptu jest użycie akcji *Script* podpiętej do obsługi zdarzenia obiektu wizualizacyjnego na diagramie. W takim przypadku możliwe jest sprawdzenie kontekstu uruchomienia – pozwala to napisać skrypt, który dostosowuje się do miejsca wywołania.

Kontekst uruchomienie opisuje obiekt interfejsu *IContext*, który można uzyskać poprzez odczyt właściwości *ExecutionContext* interfejsu *IApplication*. Najbardziej użyteczną właściwością obiektu *IContext* jest *MainVar* zwracająca nazwę zmiennej głównej obiektu, dla którego skrypt został uruchomiony.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Context : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      string mainVar = aApplication.ExecutionContext.MainVariable;
      if (!string.IsNullOrEmpty(mainVar))
      {
         IVariable v = aApplication.GetVariable(mainVar, false);
         if (v != null)
         {
            try
            {
               if (v.VariableState.IsGood)
               {
                  v.SetVariableValue((UInt32)v.VariableState.Value + 1);
```

```
            }
         }
         catch (Exception e)
         {
            aApplication.ReportMessage(EvoMessageType.Error, e.Message);

         }
         finally
         {
            v.Dispose();
         }
      }
   }
   return true;
}

public void FinalizeScript()
{
}
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class ContextVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        Dim mainVar As String
        mainVar = aApplication.ExecutionContext.MainVariable
        If Not String.IsNullOrEmpty(mainVar) Then
            Dim v As IVariable
            v = aApplication.GetVariable(mainVar, False)
            If Not v Is Nothing Then
                Try
                    If (v.VariableState.IsGood) Then
                        v.SetVariableValue(v.VariableState.Value + 1)
                    End If
                Catch e As Exception
                    aApplication.ReportMessage(EvoMessageType.Error, e.Message)
                Finally
                    v.Dispose()
                End Try
            End If
        End If
        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

W powyższym przykładzie z kontekstu uruchomienia pobierana jest nazwa zmiennej głównej. Następnie, w przypadku poprawnego odczytu wartości aktualnej, wartość zmiennej jest inkrementowana.

Alternatywną metodą kontekstowego uruchomienia skryptu w obsłudze zdarzenia obiektu, jest dynamiczna konstrukcja treści akcji operatorskiej i przekazanie danych kontekstowych w parametrach skryptu.

Przykład:

^Script(Context,LocalProperty(ControlVar))

Powyższa akcja uruchamia skrypt *Context* z przekazaniem jako parametru nazwy zmiennej sterowanej obiektu, w kontekście którego akcja jest wykonywana.

2.11 Skrypty wielojęzyczne

W przypadku skryptów pracujących w ramach aplikacji wielojęzycznej istotne staje się użycie tekstów zgodnych z aktualnym językiem pracy aplikacji. W tego typu skryptach możliwe jest użycie dwóch mechanizmów. Można wykorzystać pulę wielojęzycznych napisów aplikacji lub wyliczać właściwą postać napisu w kodzie skryptu.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class LangMessage : IScript
{
   IApplication application;
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      application = aApplication;
      aApplication.ReportMessage(EvoMessageType.Info,
                                 aApplication.ApplicationText("ScriptStartInfo"));
      aApplication.ReportMessage(EvoMessageType.Info, StartMsg());
      return true;
   }

   public void FinalizeScript()
   {
   }

   string StartMsg()
   {
      switch (application.CurrentApplicationLanguage)
      {
         case "pl":
            return "Skrypt uruchomiony";

         case "en":
         default:
            return "Script started";
      }
```

```
    }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class LangMessageVB
    Implements IScript

    Dim application As IApplication
    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        application = aApplication
        aApplication.ReportMessage(EvoMessageType.Info, _
                                   aApplication.ApplicationText("ScriptStartInfo"))
        aApplication.ReportMessage(EvoMessageType.Info, StartMsg())
        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript
    End Sub

    Function StartMsg() As String

        Select Case application.CurrentApplicationLanguage

            Case "pl"
                Return "Skrypt uruchomiony"

            Case "en"
                Return "Script started"
            Case Else
                Return "Script started"

        End Select

    End Function

End Class
```

W powyższym kodzie w pierwszym wywołaniu metody *ReportMessage* treść komunikatu pobierana jest z puli napisów aplikacji przy pomocy metody *ApplicationText*. Parametr „ScriptStartInfo” jest identyfikatorem napisu użytym w puli napisów aplikacji. Metoda *ApplicationText* zwraca wersję napisu zgodną z aktualnym językiem pracy.

W drugim wywołaniu *ReportMessage* treść komunikatu jest określona przez funkcję *StartMsg*. Funkcja ta wylicza właściwą wersję językową na podstawie aktualnej wartości właściwości *CurrentApplicationLanguage* interfejsu *IApplication*.

2.12 Tworzenie interfejsu okienkowego

Skrypty systemu Asix.Evo pozwalają także na budowę własnych okienek interakcji z użytkownikiem aplikacji. Asix.Evo nie udostępnia żadnych specjalnych mechanizmów tworzenia okien – projektant aplikacji może za to wykorzystywać wszystkie środki udostępniane przez platformę programową .NET.

2.12.1 Wyświetlanie komunikatów

Najprostszym typem okienka są okienka komunikatów, rozszerzone ewentualnie o możliwość zapytania użytkownika o proste decyzje typu Tak/Nie. Najłatwiej taką funkcjonalność zrealizować używają metodę *Show* klasy *System.Windows.Forms.MessageBox.*

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class Message : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      if (aApplication.HasRole("SuperUser"))
      {
         DialogResult result = MessageBox.Show("Czy włączyć pracę linii ?", "Start",
                                                                  MessageBoxButtons.YesNo);
         if (result == DialogResult.Yes)
         {
            IVariable v = aApplication.GetVariable("LineState", false);
            v.SetVariableBits(1, 1);
            v.Dispose();
         }
      }
      else
         MessageBox.Show("Nie masz wystarczających uprawnień !", "Brak zezwolenia");
      return true;
   }

   public void FinalizeScript()
   {

   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class MessageVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
```

```
                                    IApplication) As Boolean Implements IScript.Initialize

        If aApplication.HasRole("SuperUser") Then

            Dim result As DialogResult
            result = MessageBox.Show("Czy włączyć pracę linii ?", "Start", _
                                                    MessageBoxButtons.YesNo)
            If result = DialogResult.Yes Then

                Dim v As IVariable
                v = aApplication.GetVariable("LineState", False)
                v.SetVariableBits(1, 1)
                v.Dispose()

            End If

        Else
            MessageBox.Show("Nie masz wystarczających uprawnień !", "Brak zezwolenia")
        End If

        Return True


    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

Skrypt sprawdza, czy zalogowany użytkownik należy do roli *SuperUser*. Jeżeli tak, to wyświetlane jest okno komunikatu z zapytaniem, czy uruchomić pracę linii. Po pozytywnej odpowiedzi użytkownika najmłodszy bit zmiennej *LineState* ustawiany jest na wartość 1.

2.12.2 Okna budowane w treści skryptu

W treści skryptu można także skonstruować programowo okno dialogowe. Można zbudować okno dowolnie skomplikowane, jednakże ta technika sprawdza się dla okien niezbyt dużych.

W poniższym przykładzie tworzone jest okno złożone z dwóch tekstowych pól edycyjnych oraz dwóch przycisków.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;

public class FormBuilder : IScript
{
   Form entryForm;
   TextBox xPos, yPos;

   public bool Initialize(string[] aParameters, IApplication aApplication)
   {

      IVariable varX, varY;
```

```
    varX = aApplication.GetVariable(aParameters[0], true);
    varY = aApplication.GetVariable(aParameters[1], true);

    CreateForm();

    xPos.Text = varX.VariableState.FormattedValue;
    yPos.Text = varY.VariableState.FormattedValue;
    DialogResult res = entryForm.ShowDialog();
    if (res == DialogResult.OK)
    {
       Int32 newX, newY;
       if (Int32.TryParse(xPos.Text, out newX) && Int32.TryParse(yPos.Text, out newY))
       {
          varX.SetVariableValue(newX);
          varY.SetVariableValue(newY);
       }
       else
          MessageBox.Show("Niepoprawne wartości");
    }

    varX.Dispose();
    varX.Dispose();
    return true;
}

public void FinalizeScript()
{
}

void CreateForm()
{
    Label label1 = new Label();
    label1.AutoSize = true;
    label1.Location = new System.Drawing.Point(24, 29);
    label1.Size = new System.Drawing.Size(54, 13);
    label1.Text = "Pozycja X";

    xPos = new TextBox();
    xPos.Location = new System.Drawing.Point(97, 26);
    xPos.Size = new System.Drawing.Size(132, 20);

    Label label2 = new Label();
    label2.AutoSize = true;
    label2.Location = new System.Drawing.Point(24, 68);
    label2.Size = new System.Drawing.Size(54, 13);
    label2.Text = "Pozycja Y";

    yPos = new System.Windows.Forms.TextBox();
    yPos.Location = new System.Drawing.Point(97, 65);
    yPos.Size = new System.Drawing.Size(132, 20);

    Button buttonOk = new Button();
    buttonOk.DialogResult = DialogResult.OK;
    buttonOk.Location = new System.Drawing.Point(280, 20);
    buttonOk.Size = new System.Drawing.Size(142, 30);
    buttonOk.Text = "&OK";
    buttonOk.UseVisualStyleBackColor = true;

    Button buttonCancel = new Button();
    buttonCancel.DialogResult = DialogResult.Cancel;
    buttonCancel.Location = new System.Drawing.Point(280, 59);
    buttonCancel.Size = new System.Drawing.Size(142, 30);
    buttonCancel.Text = "&Anuluj";
    buttonCancel.UseVisualStyleBackColor = true;

    entryForm = new Form();
```

```
      entryForm.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
      entryForm.AutoScaleMode = AutoScaleMode.Font;
      entryForm.CancelButton = buttonCancel;
      entryForm.ClientSize = new System.Drawing.Size(466, 103);
      entryForm.Controls.Add(buttonCancel);
      entryForm.Controls.Add(buttonOk);
      entryForm.Controls.Add(yPos);
      entryForm.Controls.Add(label2);
      entryForm.Controls.Add(xPos);
      entryForm.Controls.Add(label1);
      entryForm.FormBorderStyle = FormBorderStyle.FixedSingle;
      entryForm.MaximizeBox = false;
      entryForm.MinimizeBox = false;
      entryForm.StartPosition = FormStartPosition.CenterScreen;
      entryForm.Text = "Koordynaty";
      entryForm.TopMost = true;
   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting

Public Class FormBuilderVB
    Implements IScript

    Dim entryForm As Form
    Dim xPos, yPos As TextBox

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        Dim varX, varY As IVariable

        varX = aApplication.GetVariable(aParameters(0), True)
        varY = aApplication.GetVariable(aParameters(1), True)

        CreateForm()

        xPos.Text = varX.VariableState.FormattedValue
        yPos.Text = varY.VariableState.FormattedValue
        If (entryForm.ShowDialog() = DialogResult.OK) Then

            Dim newX, newY As Int32
            If (Int32.TryParse(xPos.Text, newX) And Int32.TryParse(yPos.Text, newY)) Then
                varX.SetVariableValue(newX)
                varY.SetVariableValue(newY)
            Else
                MessageBox.Show("Niepoprawne wartości")
            End If

        End If

        varX.Dispose()
        varX.Dispose()

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub
```

```
    Sub CreateForm()

        Dim label1 As Label
        label1 = New Label()
        label1.AutoSize = True
        label1.Location = New System.Drawing.Point(24, 29)
        label1.Size = New System.Drawing.Size(54, 13)
        label1.Text = "Pozycja X"

        xPos = New TextBox()
        xPos.Location = New System.Drawing.Point(97, 26)
        xPos.Size = New System.Drawing.Size(132, 20)

        Dim label2 As Label
        label2 = New Label()
        label2.AutoSize = True
        label2.Location = New System.Drawing.Point(24, 68)
        label2.Size = New System.Drawing.Size(54, 13)
        label2.Text = "Pozycja Y"

        yPos = New System.Windows.Forms.TextBox()
        yPos.Location = New System.Drawing.Point(97, 65)
        yPos.Size = New System.Drawing.Size(132, 20)

        Dim buttonOk As Button
        buttonOk = New Button()
        buttonOk.DialogResult = DialogResult.OK
        buttonOk.Location = New System.Drawing.Point(280, 20)
        buttonOk.Size = New System.Drawing.Size(142, 30)
        buttonOk.Text = "&OK"
        buttonOk.UseVisualStyleBackColor = True

        Dim buttonCancel As Button
        buttonCancel = New Button()
        buttonCancel.DialogResult = DialogResult.Cancel
        buttonCancel.Location = New System.Drawing.Point(280, 59)
        buttonCancel.Size = New System.Drawing.Size(142, 30)
        buttonCancel.Text = "&Anuluj"
        buttonCancel.UseVisualStyleBackColor = True

        entryForm = New Form()
        entryForm.AutoScaleDimensions = New System.Drawing.SizeF(6.0F, 13.0F)
        entryForm.AutoScaleMode = AutoScaleMode.Font
        entryForm.CancelButton = buttonCancel
        entryForm.ClientSize = New System.Drawing.Size(466, 103)
        entryForm.Controls.Add(buttonCancel)
        entryForm.Controls.Add(buttonOk)
        entryForm.Controls.Add(yPos)
        entryForm.Controls.Add(label2)
        entryForm.Controls.Add(xPos)
        entryForm.Controls.Add(label1)
        entryForm.FormBorderStyle = FormBorderStyle.FixedSingle
        entryForm.MaximizeBox = False
        entryForm.MinimizeBox = False
        entryForm.StartPosition = FormStartPosition.CenterScreen
        entryForm.Text = "Koordynaty"
        entryForm.TopMost = True

    End Sub

End Class
```

Funkcja *CreateForm* tworzy programowo obiekt okienka dialogowego. Przed otwarciem okienka, pola edycyjne ustawiane są zgodnie z aktualnymi wartościami zmiennych przekazanych w

parametrach wywołania skryptu. Następnie okienko jest otwierane w trybie dialogowym przy pomocy metody *ShowDialog* obiektu okienka. Powrót do kodu skryptu następuje po zamknięciu okienka. Jeżeli zostało zamknięte klawiszem *OK,* to wykonywana jest próba konwersji wprowadzonych wartości na liczby typu całkowitego i wysłanie nowych wartości do zmiennych procesowych.

2.12.3 Użycie dołączanych bibliotek

Alternatywną do bezpośredniego programowego tworzenia okna dialogowego jest budowanie okien w zewnętrznym środowisku typu VisualStudio, kompilowaniu ich do postaci plików dll, a następnie użyciu w skrypcie Asix.Evo jedynie odwołania do gotowego kodu.

W poniższym przykładzie użyta zostanie biblioteka o nazwie *EvoDialog.dll,* w której znajduje się kod klasy *EvoDialog* dziedziczącej po *System.Windows.Forms.Form*. Klasa *EvoDialog* udostępnia dwie właściwości służące do ustawienia wartości wejściowych oraz jedną właściwość, w której po zamknięciu okna zostanie zapisana nazwa diagramu.

C#

```
using System;
using System.Windows.Forms;
using Askom.AsixEvo.Scripting;
using EvoAddOns;

public class Dialog : IScript
{
   public bool Initialize(string[] aParameters, IApplication aApplication)
   {
      IVariable varX, varY;

      varX = aApplication.GetVariable(aParameters[0], true);
      varY = aApplication.GetVariable(aParameters[1], true);

      EvoDialog dialog = new EvoDialog();
      dialog.XPos = varX.VariableState.FormattedValue;
      dialog.YPos = varY.VariableState.FormattedValue;

      if (dialog.ShowDialog() == DialogResult.OK)
      {
         if (!String.IsNullOrEmpty(dialog.Diagram))
            aApplication.ExecuteAction(EvoMessageType.Info,
                                       "OpenWindow(Main,P1," + dialog.Diagram + ",null)");
      }

      varX.Dispose();
      varY.Dispose();
      return true;
   }

   public void FinalizeScript()
   {

   }
}
```

VB

```
Imports System
Imports System.Windows.Forms
Imports Askom.AsixEvo.Scripting
Imports EvoAddOns

Public Class DialogVB
    Implements IScript

    Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                                    IApplication) As Boolean Implements IScript.Initialize

        Dim varX, varY As IVariable

        varX = aApplication.GetVariable(aParameters(0), True)
        varY = aApplication.GetVariable(aParameters(1), True)

        Dim dialog As EvoDialog
        dialog = New EvoDialog()
        dialog.XPos = varX.VariableState.FormattedValue
        dialog.YPos = varY.VariableState.FormattedValue

        If dialog.ShowDialog() = DialogResult.OK Then

            If Not String.IsNullOrEmpty(dialog.Diagram) Then

                aApplication.ExecuteAction(EvoMessageType.Info, _
                                    "OpenWindow(Main,P1," & dialog.Diagram & ",null)")

            End If

        End If

        varX.Dispose()
        varY.Dispose()

        Return True

    End Function

    Public Sub FinalizeScript() Implements IScript.FinalizeScript

    End Sub

End Class
```

W powyższym kodzie tworzony jest obiekt okienka *EvoDialog*. Następnie właściwości okienka *XPos* i *YPos* ustawiane są zgodnie z aktualnymi wartościami dwóch zmiennych o nazwach przekazanych w parametrach startowych skryptu. Okno dialogowe jest otwierane przy pomocy funkcji *ShowDialog*, a w przypadku zamknięcia okna ze statusem *DialogResult.OK*, konstruowana i wykonywana jest akcja operatorska otwierająca diagram o nazwie przekazanej we właściwości *Diagram* okienka.

Ażeby przykładowy skrypt wykonał się poprawnie, muszą być spełnione dwa dodatkowe warunki:

- plik *EvoDialog.dll* (wraz z ewentualnymi dodatkowymi bibliotekami) musi być skopiowany do podkatalogu *Addons* katalogu definicyjnego aplikacji.

- w zakładce *Referencje* panelu edycyjnego skryptu należy dodać referencję do pliku *EvoDialog.dll*.

Opisana technika rozszerzania funkcjonalności skryptu poprzez dołączanie bibliotek, może być stosowana również do realizacji innych zadań, niezwiązanych z interfejsem użytkownika.

e.

3 Model obiektowy skryptów

3.1 Interfejs IApplication

Interfejs *IApplication* jest podstawowym mechanizmem interakcji skryptu ze środowiskiem aplikacji Asix.Evo. Interfejs udostępnia różnorodne usługi. Część ma znaczenie czysto informacyjne. Inne służą do organizacji pracy skryptu. Interfejs służy także do tworzenia innych obiektów, które pozwalają na dostęp do takich danych aplikacji jak wartości zmiennych procesowych, czy stan alarmów.

Referencja do obiektu realizującego interfejs *IApplication* przekazywana jest do skryptu w momencie wywołania funkcji *Initialize* skryptu.

3.1.1 Właściwości

ApplicationAddonsDirectory

```
string ApplicationAddonsDirectory { get; }
```

ReadOnly Property **ApplicationAddonsDirectory** As **String**

Właściwość zwraca pełną ścieżkę dostępu do katalogu, w którym znajdują się wszystkie pliki rozszerzeń programowych aplikacji.

ApplicationDirectory

```
string ApplicationDirectory { get; }
```

ReadOnly Property **ApplicationDirectory** As **String**

Właściwość zwraca pełną ścieżkę dostępu do katalogu definicyjnego aplikacji. Jeżeli skrypt wymaga do pracy plików z dodatkowymi parametrami, to zaleca się, żeby były one przechowywane w katalogu definicyjnym aplikacji.

ApplicationName

```
string ApplicationName { get; }
```

ReadOnly Property **ApplicationName** As **String**

Właściwość zwraca nazwę aplikacji zdefiniowaną w ustawieniach stanowiska.

ApplicationWorkingDirectory

```
string ApplicationWorkingDirectory { get; }
```

ReadOnly Property **ApplicationWorkingDirectory** As **String**

Właściwość zwraca pełną ścieżkę dostępu do katalogu roboczego aplikacji. Jeżeli wynikiem działania skryptu jest tworzenie i zapis informacji do plików, to zaleca się żeby były one zapisywane w katalogu roboczym aplikacji.

Asalert

```
AsalertAccess Asalert { get; }
```

ReadOnly Property **Asalert** As **AsalertAccess**

Właściwość zwraca referencję do obiektu klasy *AsalertAccess,* służącego do komunikacji z serwerem programu AsAlert. Zwrócony obiekt pozwala na wykonanie operacji wysłania alertu przy pomocy sms'u lub email'a.

Asbase

AsbaseAccess Asbase { get; }

ReadOnly Property **Asbase** As **AsbaseAccess**

Właściwość zwraca referencję do obiektu klasy *AsbaseAccess* służącego do dostępu do danych archiwizowanych w bazie danych modułu AsBase.

Aslogger

AsloggerAccess Aslogger { get; }

ReadOnly Property **Aslogger** As **AsloggerAccess**

Właściwość zwraca referencję do obiektu klasy *AsloggerAccess* służącego do dostępu do danych archiwizowanych w bazie danych modułu AsLogger.

Asservice

AsserviceAccess Asservice { get; }

ReadOnly Property **Asservice** As **AsserviceAccess**

Właściwość zwraca referencję do obiektu klasy *AsserviceAccess* służącego do dostępu do danych archiwizowanych w bazie danych modułu AsService.

AsixDataServerStarted

```
bool AsixDataServerStarted { get; }
```

ReadOnly Property **AsixDataServerStarted** As **Boolean**

Właściwość zwraca informację, czy Serwer Danych został już uruchomiony. Właściwość może być użyta do kontrolowania, w trakcie startu aplikacji, czy system odczytu danych procesowych został w pełni uruchomiony.

ComputerName

```
string ComputerName { get; }
```

ReadOnly Property **ComputerName** As **String**

Właściwość zwraca nazwę komputera, na którym aplikacja jest wykonywana.

ComputerAddress

```
string ComputerAddress { get; }
```

ReadOnly Property **ComputerAddress** As **String**

Właściwość zwraca adres IP komputera, na którym aplikacja jest wykonywana.

CurrentApplicationLanguage

```
string CurrentApplicationLanguage { get; }
```

ReadOnly Property **CurrentApplicationLanguage** As **String**

Właściwość zwraca nazwę języka, w którym aplikacja aktualnie pracuje. Nazwy języka są dwuliterowych skrótami, np. *pl* dla języka polskiego, *en* dla języka angielskiego.

CurrentStationName

```
string CurrentStationName { get; }
```

ReadOnly Property **CurrentStationName** As **String**

Właściwość zwraca nazwę stanowiska, w kontekście którego aplikacja jest wykonywana.

CurrentUserId

```
string CurrentUserId { get; }
```

ReadOnly Property **CurrentUserId** As **String**

Właściwość zwraca identyfikator aktualnie zalogowanego użytkownika.

CurrentUserName

```
string CurrentUserName { get; }
```

ReadOnly Property **CurrentUserName** As **String**

Właściwość zwraca nazwę aktualnie zalogowanego użytkownika.

CurrentUserRoles

```
string CurrentUserRoles { get; }
```

ReadOnly Property **CurrentUserRoles** As **String**

Właściwość zwraca nazwy wszystkich ról, które pełni aktualnie zalogowany użytkownik. Nazwy ról zwracane są w postaci pojedynczego tekstu, w którym poszczególne nazwy oddzielone są przecinkami. Jeżeli wymagane jest jedynie sprawdzenie, czy użytkownik pełni konkretną rolę, to łatwiejsze w użyciu jest zastosowanie funkcji *HasRole*.

ExecutionContext

```
IContext ExecutionContext { get; }
```

ReadOnly Property **ExecutionContext** As **IContext**

Właściwość zwraca referencję do obiektu realizującego interfejs *IContext*. Interfejs ten zawiera właściwości, które określają kontekst wywołania skryptu. Kontekst wywołania jest szczególnie istotny w przypadku skryptów interakcyjnych uruchamianych w obsłudze zdarzeń obiektów wizualizacyjnych. Na podstawie kontekstu można np. pobrać nazwę zmiennej głównej obiektu.

IsAsixUser

```
bool IsAsixUser { get; }
```

ReadOnly Property **IsAsixUser** As **Boolean**

Właściwość zwraca wartość logiczną, informująca czy aktualnie zalogowany użytkownik aplikacji został zautoryzowany na podstawie wewnętrznej bazy użytkowników.

IsDomainUser

```
bool IsDomainUser { get; }
```

ReadOnly Property **IsDomainUser** As **Boolean**

Właściwość zwraca wartość logiczną, informująca czy aktualnie zalogowany użytkownik aplikacji został zautoryzowany za pośrednictwem domeny Windows.

ProgramVersion

```
string ProgramVersion { get; }
```

ReadOnly Property **ProgramVersion** As **String**

Właściwość zwraca numer wersji program *AsixEvo*. Numer wersji zwracany jest w postaci pojedynczego tekstu złożonego z czterech liczb rozdzielonych znakami kropki.

RDSComputerName

```
string RDSComputerName { get; }
```

ReadOnly Property **RDSComputerName** As **String**

W przypadku pracy w sesji RDS, właściwość zwraca nazwę komputera, na którym sesja RDS jest używana.

RDSComputerAddress

```
string RDSComputerAddress { get; }
```

ReadOnly Property **RDSComputerAddress** As **String**

W przypadku pracy w sesji RDS, właściwość zwraca adres IP komputera, na którym sesja RDS jest używana.

ScriptName

```
string ScriptName { get; }
```

ReadOnly Property **ScriptName** As **String**

Właściwość zwraca nazwę wykonywanego skryptu.

ScriptSettings

```
IScriptSettings ScriptSettings { get; }
```

ReadOnly Property **ScriptSettings** As **IScriptSettings**

Właściwość zwraca referencję do obiektu realizującego interfejs *IScriptSettings*. Interfejs ten pozwala na określenie, w jaki sposób program główny kontroluje pracę i uruchamianie skryptu. W szczególności dotyczy to sposobu reakcji na błąd skryptu.

SystemUser

```
string SystemUser { get; }
```

```
ReadOnly Property SystemUser As String
```

Właściwość zwraca identyfikator aktualnie zalogowanego użytkownika systemowego.

3.1.2 Metody

ApplicationText

```
string ApplicationText(string aTextId);
```

```
Function ApplicationText (ByVal aTextId As  String) As String
```

Funkcja zapewnia wsparcie dla aplikacji wielojęzycznych. Zwraca napis z puli napisów aplikacji. Zwracany jest napis o identyfikatorze podanym w parametrze *aTextId* w wersji językowej zgodnej z aktualnym jezykiem pracy aplikacji.

CreateThread

```
Thread CreateThread(ThreadStart aThreadFunction);
```

```
Function CreateThread (ByVal aThreadFunction As  System.Threading.ThreadStart) As
System.Threading.Thread
```

Funkcja tworzy nowy wątek wykonawczy skryptu. Parametr *aThreadFunction* określa funkcję, która zostanie wykonana w nowym wątku. Funkcja wątku musi być zgodna z systemowym delegatem *ThreadStart.* Funkcja *CreateThread* zwraca systemowy obiekt klasy *Thread,* za pośrednictwem którego można kontrolować pracę skryptu.

W skryptach Asix.Evo zalecane jest tworzenie wątków za pośrednictwem funkcji *CreateThread*, zamiast bezpośrednich metod platformy .Net.

Przykład:

C#

```
        void ThreadFnc()
        {
        //
        // kod wątku
//
}
public bool Initialize(string[] aParameters, IApplication aApplication)
{
   Thread t1 = aApplication.CreateThread(ThreadFnc);
          t1.Start();
   return false;
}
```

VB

```
Sub ThreadFun()
'
' kod wątku
'
End Sub

Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
               IApplication) As Boolean Implements IScript.Initialize

   Dim t1 As System.Threading.Thread
   t1 = aApplication.CreateThread(AddressOf ThreadFun)
   t1.Start()
   Return False

End Function
```

Funkcja ThreadFnc stanowi kod wątku. Wywołanie *CreateThread* tworzy wątek, a funkcja *Start* obiektu *Thread* uruchamia wątek.

ExecuteAction

bool ExecuteAction(string aAction);

bool ExecuteAction(string aAction, bool aWaitForFinish);

Function **ExecuteAction** (ByVal aAction As **String**) As **Boolean**

Function **ExecuteAction** (ByVal aAction As **String**, , ByVal aWaitForFinishRefresh As **Boolean**) As **Boolean**

Funkcja uruchamia akcję operatorską określoną parametrem *aAction*. Parametr *aWaitForFinish* określa czy należy oczekiwać na zakończenie wykonywania akcji operatorskiej. Wartość *false* lub brak parametru oznacza, że akcja operatorska zostanie wykonana asynchronicznie. Funkcja zwraca wartość *false*, gdy wystąpił błąd wykonania akcji.

Funkcja *ExecuteAction* rozszerza możliwości skryptów Asix.Evo o pełny zestaw operacji realizowanych w aplikacjach przy pomocy akcji operatorskich. Należy jednak zwrócić uwagę, że niektóre akcje wykonane w kontekście skryptu rezydentnego mogą mieć niekorzystne działanie. Na przykład, wykonanie akcji *OpenWindow* w przypadkowej chwili czasu może stanowić zaskoczenie dla użytkownika.

GetAlarm

```
IAlarm GetAlarm(string aAlarmName);
```

```
IAlarm GetAlarm(string aDomainName, string aAlarmName);
```

```
Function GetAlarm (ByVal aAlarmName As String) As IAlarm
```

```
Function GetAlarm (ByVal aDomain As String, ByVal aAlarmName As String) As IAlarm
```

Funkcja zwraca referencję do obiektu realizującego interfejs *IAlarm* dla alarmu o identyfikatorze określonym przez parametr *aAlarmName* z domeny o nazwie *aDomain*. Wersja wywołania bez podania nazwy domeny dotyczy domeny domyślnej (pierwszej zdefiniowanej) aplikacji.

Interfejs *IAlarm* pozwala na uzyskanie informacji na temat definicji alarmu i jego aktualnego stanu. Pozwala też na kontrolę stanu alarmu: zgłaszanie, kończenie i potwierdzanie alarmu.

GetAlarmDomainController

```
string[] GetAlarmDomainController(string aDomain);
```

```
Function GetAlarmDomainController (ByVal aDomain As String) As String()
```

Funkcja zwraca nazwę stanowiska aktywnego kontrolera domenie alarmowej określonej parametrem *aDomain*. Jeżeli nazwa domeny jest pusta, to wywołanie dotyczy domeny domyślnej (pierwszej zdefiniowanej) aplikacji.

GetAlarmDomains

```
string[] GetAlarmDomains();
```

```
Function GetAlarmDomains () As String()
```

Funkcja zwraca tablicę nazw wszystkich aktywnych domen alarmowych zdefiniowanych w aplikacji.

GetAlarmDomainStationMode

```
AlarmStationMode GetAlarmDomainStationName(string aDomain);
```

```
Function GetAlarmDomainStationName (ByVal aDomain As String) As
AlarmStationMode
```

Funkcja zwraca aktualny tryb pracy systemu alarmów w domenie alarmowej określonej parametrem *aDomain*. Jeżeli nazwa domeny jest pusta, to wywołanie dotyczy domeny domyślnej (pierwszej zdefiniowanej) aplikacji.

Typ wyliczeniowy *AlarmStationMode* może przyjmować następujące wartości: *ActiveController*, *PasiveController*, *Terminal* i *Undetermined*. Tryb *Undetermined* jest trybem przejściowym, który może zostać zwrócony w stanach nieustalonych, np. w trakcie uruchamiania domeny.

GetAlarmDomainStatistics

```
AlarmDomainStatistics GetAlarmDomainStatistics(string aDomain);
```

```
Function GetAlarmDomainStatistics (ByVal aDomain As String) As AlarmDomainStatistics
```

Funkcja zwraca obiekt klasy *AlarmDomainStatistics*, za pośrednictwem którego można sprawdzić szereg informacji statystycznych dotyczących aktualnego stanu domeny alarmowej określonej parametrem *aDomain*. Jeżeli nazwa domeny jest pusta, to wywołanie dotyczy domeny domyślnej (pierwszej zdefiniowanej) aplikacji.

GetActiveAlarms

```
string[] GetActiveAlarms(string aDomain);
```

```
Function GetActiveAlarms (ByVal aDomain As  String) As String()
```

Funkcja zwraca tablicę identyfikatorów wszystkich aktualnie aktywnych alarmów w domenie alarmowej określonej parametrem *aDomain*. Jeżeli nazwa domeny jest pusta, to wywołanie dotyczy domeny domyślnej (pierwszej zdefiniowanej) aplikacji.

GetAppData

```
string[] GetAppData(string aId);
```

```
Function GetAppData (ByVal aId As  String) As Object
```

Funkcja zwraca wartość wskazanej danej aplikacyjnej.

GetChannelState

```
ChannelState GetChannelState(string aChannelName);
```

```
Function GetChannelName (ByVal aChannelName As  String) As ChannelState
```

Funkcja zwraca aktualny status pracy kanału komunikacyjnego o nazwie podanej w parametrze *aChannelName*.

Typ wyliczeniowy *ChannelState* może przyjmować następujące wartości:

Correct - Kanał pracuje poprawnie

Initializing - Inicjalizacja kanału

MainChannelCommunicationError - Błąd komunikacji

MainChannelCommunicationErrorRedundantCommunication - Błąd komunikacji redundantnej

ControllerSTOP - Stan STOP sterownika

TempraryOff - Tymczasowo wyłączony

RedundantEvoDriver - W kanale Asix.Evo jest aktywny drajwer redundantny

OtherError - Inny błąd kanału

Unknown Channel - Nieznany kanał

Undetermined - Nieokreślony stan kanału

GetDefinedAlarms

```
string[] GetDefinedAlarms(string aDomain);
```

```
Function GetDefinedAlarms (ByVal aDomain As String) As String()
```

Funkcja zwraca tablicę identyfikatorów wszystkich alarmów zdefiniowanych w domenie alarmowej określonej parametrem *aDomain*. Jeżeli nazwa domeny jest pusta, to wywołanie dotyczy domeny domyślnej (pierwszej zdefiniowanej) aplikacji.

GetPowerGuardMeter

```
IPowerGuardMeter GetPowerGuardMeter(string aMeterName);
```

```
Function GetPowerGuardMeter(ByVal aMeterName As String) As IPowerGuardMeter
```

Funkcja zwraca interfejs miernika strażnika mocy o nazwie określonej parametrem *aMeterName*. Jeżeli miernik o danej nazwie nie istnieje to zwraca null.

GetTrendPatternPoints

```
TrendPatternPoint[] GetTrendPatternPoints(string aPatternName);
```

```
Function GetTrendPatternPoints(ByVal aPatternName As String) As TrendPatternPoint()
```

Funkcja zwraca tablicę punktów dla przebiegu wzorcowego o nazwie podanej w parametrze *aPatternName*.

Funkcja używana w skryptach, których zadaniem jest transfer przebiegu krzywej wzorcowej do sterownika.

GetVariable

```
IVariable GetVariable(string aVarName, bool aRefresh, bool aWithTimeStampsEvent);
```

```
IVariable GetVariable(string aVarName, bool aRefresh);
```

```
Function GetVariable(ByVal aVarName As String, ByVal aRefresh As Boolean) As IVariable
```

```
Function GetVariable(ByVal aVarName As String, ByVal aRefresh As Boolean, ByVal aWithTimeStampsEvent As Boolean) As IVariable
```

Funkcja zwraca referencję do obiektu realizującego interfejs *IVariable* dla zmiennej o nazwie określonej przez parametr *aVarName*. Parametr *aRefresh* określa, czy należy rozpocząć odświeżanie(okresowy odczyt) wartości zmiennej. Odświeżanie należy włączyć w przypadku monitorowania zmian wartości bieżącej zmiennej. Jeżeli skrypt będzie korzystał tylko z danych archiwalnych lub jedynie sterował wartością zmiennej, to włączenie odświeżania jest zbędne.

Parametr *aWithTimeStampsEvent* określa w jaki sposób wywoływana będzie funkcja monitorowania wartości zmiennych (deklarowana metodą *RegisterStateEvent* interfejsu IVariable). Wartość *false* lub brak parametru oznacza, że funkcja moniotorowania będzie wykonywana tylko wtedy, gdy zmieni się wartość lub status zmiennej. Wartość *true* oznacza, że kontrolowane będą także zmiany stempla czasu wartości.

Interfejs *IVariable* pozwala na uzyskanie informacji na temat definicji zmiennej oraz jej aktualnej wartości. Pozwala też na odczyt wartości archiwalnych oraz udostępnia metody zapisu wartości zmiennej.

GetVariableArcData

ArcVariableState[ ] GetVariableArcData(string aVarName,DateTime aStartTime, DateTime aEndTime, string aAggregate, TimeSpan aAggregateInterval);

Function **GetVariableArcData**(ByVal aVarName As **String**,ByVal aStartTime As **Date,** ByVal aEndTime As **Date,** ByVal aAggregate As **String,** ByVal aAggregateInterval As **System.TimeSpan**) As **ArcVariableState**( )

Funkcja pozwala na odczyt danych archiwalnych dla zmiennej określonej przez parametr *aVarName*. Parametry *aStartTime* i *aEndTime* oznaczają kolejno początek i koniec okresu czasu, z jakiego dane należy odczytać. Parametr *aAggregate* określa nazwę zastosowanego agregatu, a parametr *aAggregateInterval* okres jego wyliczania.

W momencie wykonania metody działanie skryptu jest zawieszane do chwili zakończenia odczytu danych archiwalnych. Metoda zwraca tablicę punktów (obiektów *ArcVariableState*) odczytanego przebiegu archiwalnego. W przypadku błędy zwracane jest wartość *null*.

Alternatywnym sposobem odczytu danych archiwalnych jest metoda *OrderArcData* interfejsu *IVariable*.

Zobacz też

Rodzaje agregatów danych archiwalnych - patrz: Rodzaje agregatów danych archiwalnych

Przykład:

C#

```
public bool Initialize(string[] aParameters, IApplication aApplication)
{
        ArcVariableState[] values = aApplication.GetVariableArcData("Pomiar1",
                DateTime.Today, DateTime.Now, "average", TimeSpan.FromMinutes(15));

      if (values == null)
             aApplication.ReportMessage(EvoMessageType.Error, "Read error");
      else
      {
             foreach (ArcVariableState v in values)
                      aApplication.ReportMessage(EvoMessageType.Info,
                             v.TimeStamp.ToString() + ": " + v.Value.ToString());
      }

        return true;
}
```

VB

```
Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
        IApplication) As Boolean Implements IScript.Initialize

        Dim values As ArcVariableState()
       values = aApplication.GetVariableArcData("Pomiar1", Date.Today, Date.Now, _
             "average", TimeSpan.FromMinutes(15))

      If (values Is Nothing)
             aApplication.ReportMessage(EvoMessageType.Error, "Read Error")
      Else
             For Each v As ArcVariableState In values
                    aApplication.ReportMessage(EvoMessageType.Error, _
                           v.TimeStamp.ToString() + ": " + v.Value.ToString())
             Next v
      End If
        Return True
End Function
```

Wywołanie metody *GetVariableArcData* powoduje odczyt 15-minutowych wartości średnich z okresu od początku dnia do chwili bieżącej. Jeżeli odczyt zakończył się poprawnie, to do logu komunikatów zapisywane są wszystkie wartości archiwalne, które zostały zwrócone w tablicy elementów klasy *ArcVariableState*.

GetVariableAttribute

```
object GetVariableAttribute(string aVarName, string aAttributeName);
```

Function **GetVariableAttribute** (ByVal aVarName As **String**, ByVal aAttributeName As **String**) As **Object**

Funkcja zwraca definicję atrybutu o nazwie *aAttributeName,* dla zmiennej procesowej określonej parametrem *aVarName*.

GetVariableNames

String[] GetVariableNames(string aAttributeName, string aAttributeValue);

Function **GetVariableNames**(ByVal aAttributeName As **String**, ByVal aAttributeValue As **String**) As **String()**

Funkcja zwraca tablicę nazw wszystkich zmiennych, dla których atrybut o nazwie podanej w parametrze *aAttributeName* jest zgodny z wartością poodaną w parametrze *aAttributeValue*. W parametrze *aAttributeValue* można stosować znaki specjalne * i ?.

Przykład:

C#

```
sring[] n = aApplication.GetVariableNames("Channel", "K1");
```

VB

```
Dim n As String() = aApplication.GetVariableNames("Channel", "K1")
```

Powyższe wywołanie zwraca tablicę nazw wszystkich zmiennych z kanału komunikacyjnego o nazwie *K1*.

GetVariableValue

VariableState GetVariableValue(string aVariableName, bool aForceRead);

Function **GetVariableValue**(ByVal aVariableName As **String**, ByVal aForceRead As **Boolean**) As **VariableState**

Funkcja zwraca obiekt klasy *VariableState* z bieżąca wartością zmiennej o nazwie podanej w parametrze *aVariableName*. Wartość *false* parametru *aForceRead* oznacza, że zwrócona wartość pochodzi z ostatniego planowego odświeżenia wartości zmiennej. Wartość *true* powoduje bezpośredni odczyt aktualnej wartości zmiennej w chwili wykonania funkcji.

GlobalProperty

object GlobalProperty(string aPropertyName);

Function **GlobalProperty** (ByVal aPropertyName As **String**) As **Object**

Funkcja zwraca definicję właściwości globalnej aplikacji o nazwie określonej parametrem *aPropertyName*.

HasRole

bool HasRole(string aRoleName);

Function **HasRole** (ByVal aRoleName As **String**) As **Boolean**

Funkcja zwraca wartość logiczną, która określa, czy aktualnie zalogowany użytkownik pełni rolę o nazwie przekazanej w parametrze *aRoleName*.

LoginUser

bool LoginUser(string aUser, string aPassword);

Function **LoginUser** (ByVal aUser As **String**, , ByVal aPassword As **Boolean**) As **Boolean**

Funkcja powoduje zalogowanie użytkownika o identyfikatorze i haśle podanym w parametrach wywołania. Funkcja zwraca wartość *true*, gdy operacja zakończyła się poprawnie.

Funkcja *LoginUser* może być użyta w przypadku tworzenia własnych mechanizmów logowania użytkownika, także wykorzystujących mechanizmy sprzętowe.

OPCTime

DateTime OPCTime(string aOPCTimeString);

DateTime OPCTime(string aOPCTimeString, DateTime aBaseTime);

Function **OPCTime**(ByVal aOPCTimeString As **String**) As **Date**

Function **OPCTime** (ByVal aOPCTimeString As **String**, ByVal aBaseTime As **Date**) As **Date**

Funkcja zwraca chwilę czasu wyliczoną na podstawie czasu bazowego i przesunięcia określonego wyrażeniem formatującym OPC. Czas bazowy jest określony parametrem *aBaseTime* lub jest czasem aktualnym dla wywołania jednoargumentowego.

Przesunięcie przekazywane w parametrze *aOPCTimeString* opisane jest rozszerzonym wyrażeniem formatującym OPC postaci:

[NOW|SECOND|MINUTE|HOUR|DAY|WEEK|MONTH|YEAR][cykl[/przesunięcie]] [[+|-][liczba][Y|MO|W|D|H|M|S]...]

Pierwsza część wyrażenia określa początkowe przesunięcie, np. HOUR początek godziny. Klauzule *cykl* i *przesunięcie* pozwalają na implementacje cykli, np. HOUR8/6 pozwala na wyliczenie początku 8-godzinnej zmiany, gdzie pierwsza zmiana zaczyna się o godzinie 6. Końcowa część wyrażenia pozwala dodatkowo przesunąć czas, np. HOUR-15M , oznacza 15 minut przed początkiem godziny.

ReadVariables

VariableState() ReadVariables(params string[] aVarNames);

Function **ReadVariables**(ByVal ParamArray aVarNames As **String()**) As **VariableState()**

Funkcja wykonuje grupowy odczyt wartości wielu zmiennych procesowych. Nazwy zmiennych podawane są tablicy przekazywanej w parametrze *aVarNames*. Funkcja zwraca tablicę obiektów klasy *VariableState* z bieżącymi wartościami zmiennych. Funkcja wykonuje rzeczywisty odczyt wartości zmiennych (w taki sam sposób jak metoda *GetVariableValue* z parametrem *aForceRead* ustawionym na *true*). Czas wykonania funkcji zależy od liczby zmiennych i stosowanych kanałów komunikacyjnych. Jest jednak z reguły znacznie krótszy niż dla wielokrotnego wywołania metody *GetVariableValue*.

RegisterAppDataEvent

IEventInfo RegisterAppDataEvent(SimpleEvent aFunction);

Function RegisterAppDataEvent(ByVal aFunction As SimpleEvent) As IEventInfo

Funkcja służy do zarejestrowania funkcji zwrotnej, która będzie wywoływana przy każdym zdarzeniu związanym ze zmianą wartości którejś z danych aplikacyjnych (AppData).

Parametr *aFunction* określa funkcję monitorującą stan danych aplikacyjnych - musi ona być zgodna z delegatem *AppDataEvent:*

delegate void AppDataEvent(string aAppDataId);

Delegate Sub **AppDataEvent(**ByVal aAppDataId As **String**)

RegisterAlarmDomainEvents

IAlarmEventInfo RegisterAlarmDomainEvents(string aDomain, AlarmEvent aFunction, AlarmDomainEvents aEvent);

Function **RegisterAlarmDomainEvents**(ByVal aDomain As **String**, ByVal aFunction As **AlarmEvent**, ByVal aEvent As **AlarmDomainEvents**) As **IAlarmEventInfo**

Funkcja służy do zarejestrowania funkcji zwrotnej, która będzie wywoływana przy każdym zdarzeniu związanym ze zmianą stanu domeny. Parametr *aEvent* określa rodzaj zdarzeń, które należy monitorować. Typ wyliczeniowy *AlarmDomainEvents* przybiera następujące wartości: Alarm*Begin* (wykrycie początku alarmu)*, AlarmEnd* (wykrycie końca alarmu), *AlarmManagement* (zmiana stanu wynikająca z działań operatora, np. potwierdzenia alarmu), *DomainStart* (uruchomienie pracy domeny), *DomainStop* (zatrzymanie pracy domeny), *All* (wszystkie typy zdarzeń). Możliwe jest łączenie typów zdarzeń operatorem sumy logicznej.

Parametr *aFunction* określa funkcję monitorującą stan alarmu - musi ona być zgodna z delegatem *AlarmEvent.*

delegate void AlarmEvent(string aDomain, string aAlarm, AlarmState aAlarmState, AlarmEventType aEventType);

Delegate Sub **AlarmEvent(**ByVal aDomain As **String**, ByVal aAlarm As **String**, ByVal aAlarmState As **AlarmState** ,ByVal aEventType As **AlarmEventType**).

W momencie wywołania funkcji monitorującej przekazywane są do niej trzy parametry. Parametry *aDomain* i *aAlarm* określają odpowiednio nazwę domeny i identyfikator alarmu, którego wywołanie funkcji zwrotnej dotyczy. Parametr *aAlarmState* przekazuje obiekt, z którego można odczytać aktualny stan alarmu. Parametr *aEventType* określa rodzaj zdarzenia. Typ wyliczeniowy *AlarmEventType* przybiera następujące wartości: Alarm*Begin* (wykrycie początku alarmu) *, AlarmEnd* (wykrycie końca alarmu), *AlarmManagement* (zmiana stanu wynikająca z działań operatora, np. potwierdzenia alarmu), *DomainStart* (uruchomienie pracy domeny), *DomainStop* (zatrzymanie pracy domeny).

Metoda *RegisterAlarmDomainEvents* zwraca obiekt interfejsu *IAlarmEvcentInfo*. Obiekt ten udostępnia jedną metodę *UnregisterStateEvent.* Służy ona do wyrejestrowania funkcji monitorującej.

Przykład:

C#

```
IAlarmEventInfo alarmInfo;
IApplication application;

public bool Initialize(string[] aParameters, IApplication aApplication)
{
       application = aApplication;

       alarmInfo = aApplication.RegisterAlarmDomainEvents("DemoB1", AlarmFun,
                    AlarmDomainEvents.AlarmBegin | AlarmDomainEvents.AlarmEnd);
       return false;
}

public void FinalizeScript()
{
       if (alarmInfo != null)
             alarmInfo.UnregisterStateEvent();
}

void AlarmFun(string aDoamin,string aId, AlarmState aState, AlarmEventType aType)
{
       if (aType == AlarmEventType.AlarmBegin)
              application.ReportMessage(EvoMessageType.Info,
                     "Wykryty początek alarmu " + aId + ", czas detekcji " +
                     aState.BeginDetectionTimeStamp);
       else if (aType == AlarmEventType.AlarmEnd)
             application.ReportMessage(EvoMessageType.Info,
                    "Wykryty koniec alarmu " + aId + ", czas detekcji " +
                      aState.EndDetectionTimeStamp);
}
```

VB

```
Dim alarmInfo As IAlarmEventInfo
Dim application As IApplication

Public Function Initialize(ByVal aParameters() As String, ByVal aApplication _
                     As IApplication) As Boolean Implements IScript.Initialize

       application = aApplication
      alarmInfo = aApplication.RegisterAlarmDomainEvents("DemoB1", _
                AddressOf AlarmFun, _
             AlarmDomainEvents.AlarmBegin Or AlarmDomainEvents.AlarmEnd)
      Return False
End Function

Public Sub FinalizeScript() Implements IScript.FinalizeScript

      If Not alarmInfo Is Nothing
             alarmInfo.UnregisterStateEvent()
      End If

End Sub
```

```
Sub AlarmFun(aDomain As String, aId As String, aState As AlarmState, _
               aType As AlarmEventType)

      If aType = AlarmEventType.AlarmBegin
            application.ReportMessage(EvoMessageType.Info, _
                  "Wykryty początek alarmu " + aId + ", czas detekcji " + _
                    aState.BeginDetectionTimeStamp)
      Else If aType = AlarmEventType.AlarmEnd
            application.ReportMessage(EvoMessageType.Info, _
                  "Wykryty koniec alarmu " + aId + ", czas detekcji " + _
                    aState.EndDetectionTimeStamp)
      End If

End Sub
```

Powyższy kod rejestruje funkcję monitorującą w domenie *DemoB1* dla zdarzeń początków i końców alarmów. Funkcja monitorująca zapisuje w logu komunikatów informację o każdym rozpoczętym i zakończonym alarmie.

RegisterTimeEvent

IEventInfo RegisterTimeEvent(SimpleEvent aFunction, TimeSpan aPeriod, TimeSpan aAlign);

Function RegisterTimeEvent(ByVal aFunction As SimpleEvent, ByVal aPeriod As System.TimeSpan, ByVal aAlign As System.TimeSpan) As IEventInfo

Funkcja służy do zarejestrowania funkcji zwrotnej, która będzie cyklicznie wywoływana w określonych odstępach czasu. Parametr *aFunction* określa funkcję, która będzie wykonywana okresowo - musi ona być zgodna z delegatem *SimpleEvent.*

delegate void SimpleEvent();

Delegate Sub **SimpleEvent()**

Parametr *aPeriod* określa okres czasu z jakim funkcja zwrotna będzie wykonywana. Parametr *aAlign* określa wyrównanie czasu uruchomienia względem początku doby. Jeżeli parametr jest mniejszy niż zero (np. *TimeSpan.FromSeconds(-1)*), to wyrównanie nie jest wykonywane. Jeżeli *aAlign* jest równy zero, to funkcja zwrotna będzie wywoływana co okres *aPeriod,* zaczynając od godziny 00:00. Jeżeli *aAlign* jest większy niż zero, to funkcja zwrotna będzie wywoływana co okres *aPeriod,* zaczynając od godziny 00:00 + *aAlign*.

Funkcja *RegisterTimeEvent* zwraca wartość, którą można użyć do wyrejestrowania funkcji zwrotnej przy pomocy metody *UnregisterEvent*.

Przykład:

C#

```
        void CallbackFun()
        {
        //
        // kod funkcji zwrotnej
//
}
public bool Initialize(string[] aParameters, IApplication aApplication)
{
   aApplication.RegisterTimeEvent(CallbackFun, TimeSpan.FromHours(1),
                                  TimeSpan.FromMinutes(15));
   return false;
}
```

VB

```
Sub CallbackFun()
        '
        ' kod funkcji zwrotnej
        '
End Sub

Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
               IApplication) As Boolean Implements IScript.Initialize

          aApplication.RegisterTimeEvent(AddressOf CallbackFun, TimeSpan.FromHours(1),_
                                  TimeSpan.FromMinutes(15))
   Return False

        End Function
```

Funkcja zwrotna *CallbackFun* będzie wykonywana co godzinę z przesunięciem 15 minut, czyli o godzinach 0:15, 1:15, 2:15 itd.

RemoveAppData

void RemoveAppData(string aId);

Sub **RemoveAppData** (ByVal aId As **String**)

Funkcja usuwa wskazaną daną aplikacyjną.

ReportMessage

void ReportMessage(EvoMessageType aType, string aText);

Sub **ReportMessage** (ByVal aType As **EvoMessageType**, ByVal aText As **String**)

Funkcja zapisuje w aplikacyjnym logu komunikat o treści podanej w parametrze *aText*. Parametr *aType* określa ważność komunikatu. Wyliczeniowy typ *EvoMessageType* przyjmuje następujące wartości: *Exception*, *Error*, *Warning*, *Info*, *Trace*. Komunikaty typu *Trace* nie są wyświetlane w okienku komunikatów, są jedynie zapisywane w pliku logu.

Poziom raportowania pracy skryptu w logu komunikatów jest w gestii projektanta skryptu. Należy jednak pamiętać o dokładnym raportowaniu wszystkich błędów, które wystąpiły w trakcie pracy. Dotyczy to szczególnie skryptów rezydentnych.

RestartScript

void RestartScript();

Sub RestartScript()

Metoda pozwala na wykonanie pełnego restart skryptu. Należy stosować wtedy, gdy własny kod kontrolny skryptu stwierdzi jego nieprawidłowe działanie wymagające restartu. Niezależnie od tego mechanizmu, w przypadku niekontrolowanego upadku skrytpu, możliwy jest jego automatyczny restart (jeżeli została ustawiona właściwość *RestartIfFailed* interfejsu *IScriptSettings*).

SetAppData

void SetAppData(string aId);

Sub **SetAppData** (ByVal aId As **String**, ByVal aData As **Object**)

Funkcja ustawia lub zmienia wartość wskazanej danej aplikacyjnej.

Przykład 1:

```
public bool Initialize(string[] aParameters, IApplication aApplication)
{
	aApplication.SetAppData("Number_1", 12);

	return true;
}
```

Przykład 2:

```
public bool Initialize(string[] aParameters, IApplication aApplication)
{
	DataTable dt = new DataTable();

	dt.Columns.Add("X", typeof(double));
	dt.Columns.Add("Y", typeof(double));
	dt.Columns.Add("Label", typeof(string));
	dt.Columns.Add("Color", typeof(string));

	for(int i = 0; i < 10; i++)
	{
		Color c = Color.FromArgb(mRnd.Next(255), mRnd.Next(255), mRnd.Next(255));
		dt.Rows.Add(i, mRnd.Next(100), "L" + mRnd.Next(100),
		Askom.Ascent.StringHelper.ColorToHex(c) );
	}

	mApplication.SetAppData("DataTable_1", dt);
}
```

SetVariableBits

bool SetVariableBits(string aVariableName, long aValue, long aMask);

bool SetVariableValue(string aVariableName, long aValue, long aMask, StatusOpc aStatus);

bool SetVariableValue(string aVariableName, long aValue, long aMask, StatusOpc aStatus, DateTime aTimeStamp);

Function **SetVariableBits**(ByVal aVariableName As **String**, ByVal aValue As **Long**, ByVal aMask As **Long**) As **Boolean**

Function **SetVariableBits**(ByVal aVariableName As **String**, ByVal aValue As **Long**, ByVal aMask As **Long**, ByVal aStatus As **Askom.Ascent.StatusOpc**) As **Boolean**

Function **SetVariableBits**(ByVal aVariableName As **String**, ByVal aValue As **Long**, ByVal aMask As **Long**, ByVal aStatus As **Askom.Ascent.StatusOpc**, ByVal aTimeStamp As **Date**) As **Boolean**

Funkcja powoduje wysterowanie zmiennej procesowej o nazwie podanej w parametrze *aVariableName* na wartość podaną w parametrze *aValue* z dodatkowym maskowaniem zmienianych bitów przez parametr *aMask*. W pierwszej fazie wykonania odczytywana jest bieżąca wartość zmiennej, a następnie bity, którym w masce *aMask* odpowiada wartość 1, zmieniane są zgodnie z wartościami w parametrze *aValue*.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennej. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennej. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

SetVariables

```
bool SetVariables(string[] aVarNames, object[] aValues);

bool SetVariables(string[] aVarNames, object[] aValues, StatusOpc aStatus);

bool SetVariables(string[] aVarNames, object[] aValues, StatusOpc aStatus, DateTime
aTimeStamp );

Function SetVariables(ByVal aVarNames As String(), ByVal aValues As  Object()) As
Boolean

Function SetVariables(ByVal aVarNames As String(), ByVal aValues As  Object(),
ByVal aStatus As  Askom.Ascent.StatusOpc) As Boolean

Function SetVariables(ByVal aVarNames As String(), ByVal aValues As  Object(), ByVal
aStatus As  Askom.Ascent.StatusOpc,ByVal aTimeStamp As Date) As Boolean
```

Funkcja powoduje wysterowanie zmiennych procesowej o nazwach podanych w tablicy *aVarNames* na wartości podane w tablicy *aValues*. Rozmiary tablic nazw i wartości muszą być identyczne. Czas wykonania sterowań przez funkcję *SetVariables* jest z reguły krótszy niż wykonanie kilku wywołań metody *SetVariableValue*.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennych. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennych. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Możliwość ustawienia czasu i statusu zależy od typu kanału, do którego zmienna należy. W szczególności, tej funkcjonalności nie mają zazwyczaj kanały fizyczne.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

SetVariablesSync

```
IVariableControlState[] SetVariablesSync(string[] aVarNames, object[] aValues);

IVariableControlState[]SetVariablesSync(string[] aVarNames, object[]
aValues, StatusOpc aStatus);

IVariableControlState[]SetVariablesSync(string[] aVarNames, object[]
aValues, StatusOpc aStatus, DateTime aTimeStamp );

Function SetVariablesSync(ByVal aVarNames As String(), ByVal aValues As  Object())
As IVariableControlState()

Function SetVariablesSync(ByVal aVarNames As String(), ByVal aValues As  Object(),
ByVal aStatus As  Askom.Ascent.StatusOpc) As IVariableControlState()

Function SetVariablesSync(ByVal aVarNames As String(), ByVal aValues As  Object(),
ByVal aStatus As  Askom.Ascent.StatusOpc, ByVal aTimeStamp As Date) As
IVariableControlState()
```

Funkcja służy do grupowego sterownia wartościami zmiennych z kontrolą rzeczywistego wykonania wszystkich sterowań. Poza tym działa identycznie jak metoda *SetVariables*.

Funkcja powoduje wysterowanie zmiennych procesowej o nazwach podanych w tablicy *aVarNames* na wartości podane w tablicy *aValues*. Rozmiary tablic nazw i wartości muszą być identyczne. Czas wykonania sterowań przez funkcję *SetVariables* jest z reguły krótszy niż wykonanie kilku wywołań metody *SetVariableValue*.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennych. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennych. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Możliwość ustawienia czasu i statusu zależy od typu kanału, do którego zmienna należy. W szczególności, tej funkcjonalności nie mają zazwyczaj kanały fizyczne.

Funkcja zwraca referencje do obiektu klasy *IVariableControlState*. Obiekt ten może służyć do kontroli procesu wykonania operacji.

Przykład:

C#

```
string[] varNames = new string[]{"var1","var2"};
object[] varVals = new object[]{101,102};

IVariableControlState[] rs = aApplication.SetVariablesSync(varNames, varVals);
bool allFinished = false;
while (!allFinished)
{
       aApplication.Sleep(TimeSpan.FromMilliseconds(100)); // or do something

      allFinished = true;
      foreach(IVariableControlState s in rs)
      {
            if (s.ControlStatus == VariableControlStatus.Pending)
            {
                  allFinished = false;
                  break;
            }
      }
}
foreach(IVariableControlState s in rs)
{
      if (s.ControlStatus == VariableControlStatus.Error)
            aApplication.ReportMessage(EvoMessageType.Error,
                                          "Write failed: " + s.ErrorMsg);
}
```

VB

```
Dim varNames As String() = {"var1","var2"}
Dim varVals As Object() = {101,102}

Dim s As IVariableControlState
Dim rs As IVariableControlState() = aApplication.SetVariablesSync(varNames, varVals)

Dim allFinished As Boolean = False
While Not allFinished

      aApplication.Sleep(TimeSpan.FromMilliseconds(100)) ' or do something

      allFinished = True
      For Each s In rs
            If s.ControlStatus = VariableControlStatus.Pending Then
                  allFinished = False
                  Exit For
            End If
      Next

End While

For Each s In rs
      If s.ControlStatus = VariableControlStatus.Error Then
            aApplication.ReportMessage(EvoMessageType.Error, "Write failed: " + _
                                          s.ErrorMsg)
      End If
Next
```

SetVariableValue

```
bool SetVariableValue(string aVariableName, object aValue);

bool SetVariableValue(string aVariableName, object aValue, StatusOpc aStatus);

bool SetVariableValue(string aVariableName, object aValue, StatusOpc aStatus,
DateTime aTimeStamp );

Function SetVariableValue(ByVal aVariableName As String, ByVal aValue As  Object)
As Boolean

Function SetVariableValue(ByVal aVariableName As String, ByVal aValue As  Object,
ByVal aStatus As  Askom.Ascent.StatusOpc) As Boolean

Function SetVariableValue(ByVal aVariableName As  String,  ByVal aValue As  Object,
ByVal aStatus As  Askom.Ascent.StatusOpc,  ByVal aTimeStamp As  Date) As
Boolean
```

Funkcja powoduje wysterowanie zmiennej procesowej o nazwie podanej w parametrze *aVariableName* na wartość podaną w parametrze *AValue*.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennej. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennej. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Możliwość ustawienia czasu i statusu zależy od typu kanału, do którego zmienna należy. W szczególności, tej funkcjonalności nie mają zazwyczaj kanały fizyczne.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

Sleep

```
void Sleep(TimeSpan aTimeSpan);

Sub Sleep (ByVal aTimeSpan As  System.TimeSpan)
```

Funkcja zawiesza działanie bieżącego wątku skryptu na okres czasu podany w parametrze *aTimeSpan*. Funkcję należy stosować wszędzie tam, gdzie wymagane jest odczekanie na zajście jakiegoś zdarzenia

lub upłynięcie określonego odcinka czasu. W żadnym przypadku nie należy stosować aktywnej pętli testującej warunki - prowadzi to do nadmiernego obciążenia procesora).

Przykład:

C#

```
aApplication.Sleep(TimeSpan.FromSeconds(5));
```

VB

```
aApplication.Sleep(TimeSpan.FromSeconds(5))
```

Wywołanie zawiesza działania skryptu na 5 sekund.

TerminateScript

```
void TerminateScript(bool aWithFinalize);
```

```
Sub TerminateScript (ByVal aWithFinalize As Boolean)
```

Wywołanie funkcji powoduje zakończenie pracy skryptu. Parametr *aWithFinalize* określa, czy w trakcie procesu rozładowania skryptu należy wykonać metodę *Finalize* skryptu.

Funkcja *TerminateScript* jest używana tylko w szczególnych przypadkach. Skrypty interakcyjne (jednorazowe) zazwyczaj kończone są przez wyjście z funkcji *Initialize* z wartością zwracaną *true*. Skrypty rezydentne są kończone automatycznie w momencie końca pracy aplikacji.

ToggleBits

```
bool ToggleBits(string aVariableName, long aMask);
```

```
bool SetVariableToggleBits(string aVariableName, long aMask, StatusOpc aStatus);
```

```
bool ToggleBits(string aVariableName, long aMask, StatusOpc aStatus, DateTime
aTimeStamp);
```

```
Function ToggleBits(ByVal aVariableName As String, ByVal aMask As  Long) As
Boolean
```

```
Function ToggleBits(ByVal aVariableName As String, ByVal aMask As  Long, ByVal
aStatus As  Askom.Ascent.StatusOpc) As Boolean
```

```
Function ToggleBits(ByVal aVariableName As String, ByVal aMask As  Long, ByVal
aStatus As  Askom.Ascent.StatusOpc, ByVal aTimeStamp As Date) As Boolean
```

Funkcja powoduje zmianę wartości wybranych bitów zmiennej procesowej o nazwie podanej w parametrze *aVariableName* na ich wartość przeciwną. W pierwszej fazie wykonania odczytywana jest bieżąca wartość zmiennej, a następnie bity, którym w parametrze *aMask* odpowiada wartość 1, zmieniane są na wartość przeciwną. Pozostałe bity zmiennej pozostają bez zmiany.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennej. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennej. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

UnregisterEvent

```
void UnregisterEvent(IEventInfo aCallbackInfo);
```

```
Sub UnregisterEvent(ByVal aCallbackInfo As IEventInfo)
```

Wyrejestrowanie funkcji uruchamianej czasowo, która została wcześniej zarejestrowana przy pomocy funkcji *RegisterTimeEvent*. Jako parametr *aCallbackInfo* należy przekazać wartość zwróconą w momencie rejestrowania funkcji.

3.2 Interfejs IContext

Interfejs *IContext* zawiera właściwości, które określają kontekst wywołania skryptu. Kontekst wywołania jest szczególnie istotny w przypadku skryptów interakcyjnych uruchamianych w obsłudze zdarzeń obiektów wizualizacyjnych. Interfejs pozwala uzależnić działanie skryptu od sposobu jego uruchomienia.

Referencję do interfejsu IContext można uzyskać poprzez odczyt właściwości *ExecutionContext* interfejsu *IApplication*.

3.2.1 Właściwości

Diagram

```
string Diagram { get; }
```

```
ReadOnly Property Diagram As String
```

Właściwość zwraca nazwę diagramu, w kontekście którego skrypt został uruchomiony.

DiagramObject

```
string DiagramObject { get; }
```

```
ReadOnly Property DiagramObject As String
```

Właściwość zwraca nazwę obiektu wizualizacyjnego, w kontekście którego skrypt został uruchomiony.

IsWebBrowser

```
bool IsWebBrowser { get; }
```

```
ReadOnly Property IsWebBrowser As Boolean
```

Właściwość zwraca informację, czy aplikacja pracuje w środowisku przeglądarki.

MainVariable

```
string MainVariable { get; }
```

ReadOnly Property **MainVariable** As **String**

Właściwość zwraca nazwę głównej zmiennej obiektu wizualizacyjnego, w kontekście którego skrypt został uruchomiony.

ScriptStartTime

```
DateTime ScriptStartTime { get; }
```

ReadOnly Property **ScriptStartTime** As **Date**

Właściwość zwraca czas momentu uruchomienia skryptu.

SourceFilePath

```
string SourceFilePath { get; }
```

ReadOnly Property **SourceFilePath** As **String**

Właściwość zwraca pełną ścieżkę pliku źródłowego wykonywanego skryptu.

WebServerAddress

```
string WebServerAddress { get; }
```

```
ReadOnly Property WebServerAddress As String
```

Właściwość zwraca adres serwera WWW, z którego została załadowana aplikacja przeglądarkowa.

3.2.2 Metody

IsDiagramActive

```
bool IsDiagramActive(string aDiagramName,string aParameters);
```

```
Function IsDiagramActive (ByVal aDiagramName As String, ByVal aParameters As String) As Boolean
```

Funkcja zwraca wartość logiczną, informującą czy jest aktualnie otwarty i aktywny(bieżący) diagram określony przez nazwę *aDiagramName* i parametry otwarcia *aParameters*.

IsDiagramOpened

```
bool IsDiagramOpened(string aDiagramName, string aParameters, int aScreen);
Function IsDiagramOpened (ByVal aDiagramName As String, ByVal aParameter As String,
ByVal aScreen As Integer) As Boolean
```

Funkcja zwraca wartość logiczną, informującą czy na ekranie określonym przez parametr *aScreen* jest otwarty diagram o nazwie określonej przez parametr *aWindowName* i parametrach otwarcia *aParameters*. Parametr *aScreen* może mieć wartość -1 (oznacza to dowolny monitor) lub wartość od 1 wzwyż, będącą numerem monitora zgodnie z numeracją systemu operacyjnego.

IsRuntime

```
bool IsRuntime();
```

```
Function IsRuntime () As Boolean
```

Funkcja zwraca wartość logiczną, informującą czy aplikacja jest aktualnie w trybie pracy operatorskiej.

IsScriptWorking

```
bool IsScriptWorking(string aScriptName,params string[] aParameters);
```

```
Function IsScriptWorking (ByVal aScriptName As String, ByVal ParamArray aParameters As String()) As Boolean
```

Funkcja zwraca wartość logiczną, informującą czy jest aktualnie uruchomiony skrypt o podanej nazwie i parametrach startowych.

IsWindowOpened

```
bool IsWindowOpened(string aWindowName, int aScreen);
```

```
Function IsWindowOpened (ByVal aWindowName As String, ByVal aScreen As Integer) As Boolean
```

Funkcja zwraca wartość logiczną, informującą czy na ekranie określonym przez parametr *aScreen* jest otwarte okno o nazwie określonej przez parametr *aWindowName*. Parametr *aScreen* może mieć wartość -1 (oznacza to dowolny monitor) lub wartość od 1 wzwyż, będącą numerem monitora zgodnie z numeracją systemu operacyjnego.

RestartApplication

```
void RestartApplication();
```

```
Sub RestartApplication()
```

Metoda wykonuje pełny restart aplikacji: program jest kończony, a następnie ponownie uruchomiany. Nie są sprawdzane uprawniania zamykania aplikacji przez aktualnie zalogowanego użytkownika.

3.3 Interfejs IScriptSettings

3.3.1 Właściwości

AllowMultipleInstances

```
MultipleInstanceMode AllowMultipleInstances { get; set; }
```

```
Property AllowMultipleInstances As MulitipleInstanceMode
```

Właściwość określa sposób kontroli prób wielokrotnego uruchomienia skryptu. Właściwość może przyjmować następujące wartości:

Never - nie można uruchomić wielu instancji skryptu

Always -można uruchomić dowolną liczbę instancji skryptu

IfDifferentParameters -można uruchomić wiele instancji skryptu, jeżeli różnią się parametrami startowymi (wartość domyślna)

RestartIfFailed

```
bool RestartIfFailed { get; set; }
```

```
Property RestartIfFailed As Boolean
```

Właściwość określa sposób reakcji na błąd wykonania skryptu. Wartość *true* oznacza, że w przypadku upadku skryptu zostanie on automatycznie ponownie uruchomiony. Wartość domyślna jest równa *false*.

Typowo, właściwość jest ustawiana na *true* dla skryptów autonomicznych po zakończeniu fazy inicjalizacji.

3.4 Interfejs IVariable

3.4.1 Właściwości

Name

```
string Name { get; }
```

ReadOnly Property **Name** As **String**

Właściwość zwraca nazwę zmiennej procesowej kontrolowanej przez posiadaną referencję do interfejsu *IAlarm*.

IsRefreshed

```
bool IsRefreshed { get; }
```

ReadOnly Property **IsRefreshed** As **Boolean**

Właściwość zwraca informację, czy wykonywane jest cykliczne odświeżanie wartości zmiennej. Tryb odświeżania deklarowany jest w momencie tworzenia interfejsu IVariable funkcją GetVariable interfejsu IApplication.

VariableState

```
VariableState VariableState { get; }
```

ReadOnly Property **VariableState** As **VariableState**

Właściwość zwraca obiekt klasy *VariableState*, z którego można odczytać aktualną wartość i status zmiennej procesowej. Wartość zmiennej pochodzi z chwili ostatniego odświeżenia zmiennej. Okres odświeżania jest określony w definicji zmiennej procesowej.

3.4.2 Metody

OrderArcData

IArcQueryResult `OrderArcData(`ArcQuery `aQuery);`

Function **OrderArcData** (ByVal aQuery As **ArcQuery**) As **IArcQueryResult**

Funkcja służy do odczytu danych archiwalnych zmiennej procesowej. Obiekt klasy *ArcQuery* przekazywany jako parametr *aQuery* określa parametry odczytu, w szczególności okres czasu i sposób agreagacji. Funkcja *OrderArcData* jedynie inicjuje proces odczytu danych historycznych. Zwracany obiekt interfejsu *IArcQueryResult* służy do kontroli procesu pozyskania danych oraz odczytu danych wynikowych.

Przykład:

C#

```
public bool Initialize(string[] aParameters, IApplication aApplication)
{
   IVariable iv = aApplication.GetVariable("A004", false);

    IArcQueryResult qr = iv.OrderArcData(new ArcQuery(DateTime.Today, DateTime.Now,
                                        "average", TimeSpan.FromMinutes(15)));
   if (qr.ResultStatus == ArcResultStatus.Processing)
       qr.Wait();


   if (qr.ResultStatus == ArcResultStatus.Error)
      aApplication.ReportMessage(EvoMessageType.Error, "OrderArcData error: " +
                                 qr.ErrorDescription);
   else
   {
       foreach (ArcVariableState v in qr.Values)
         aApplication.ReportMessage(EvoMessageType.Info, v.TimeStamp.ToString() +
                                    ": " + v.Value.ToString());

   }

   return true;
```

```
    }
```

VB

```
Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
               IApplication) As Boolean Implements IScript.Initialize

   Dim iv As IVariable
   iv = aApplication.GetVariable("A004", False)

   Dim qr As IArcQueryResult
   qr = iv.OrderArcData(New ArcQuery(Date.Today, Date.Now, _
        "average", TimeSpan.FromMinutes(15)))

   If (qr.ResultStatus = ArcResultStatus.Processing)
      Qr.Wait()
   End If


   If (qr.ResultStatus = ArcResultStatus.Error)
      aApplication.ReportMessage(EvoMessageType.Error, _
                      "OrderArcData error: " + qr.ErrorDescription)
   Else
      For Each v As ArcVariableState In qr.Values
         aApplication.ReportMessage(EvoMessageType.Error, _
                                         v.TimeStamp.ToString() + ": " + v.Value.ToString())
      Next v
   End If
   Return True
       End Function
```

Przekazany w wywołaniu metody *OrderArcData* obiekt klasy *ArcQuery* powoduje odczyt 15-minutowych wartości średnich z okresu od początku dnia do chwili bieżącej. Po inicjacji odczytu następuje oczekiwanie na koniec operacji poprzez wywołanie funkcji *Wait* obiektu *IArcQueryResult,* zwróconego w wyniku wykonania metody *OrderArcData*. Jeżeli odczyt zakończył się poprawnie, to do logu komunikatów zapisywane są wszystkie wartości archiwalne, które zostały zwrócone poprzez właściwość *Values* obiektu klasy *IArcQueryResult*.

Read

VariableState `Read();`

Function **Read** () As **VariableState**

Funkcja zwraca obiekt klasy *VariableState*, z którego można odczytać aktualną wartość i status zmiennej procesowej. Funkcja wymusza rzeczywisty odczyt zmiennej, wartość zmiennej pochodzi z momentu wykonania funkcji.

Czas wykonania funkcji zależy od czasu wykonania operacji odczytu wartości zmiennej ze źródła danych.

RegisterStateEvent

```
void RegisterStateEvent(VariableEvent aFunction);
```

```
Sub RegisterStateEvent(ByVal aFunction As VariableEvent)
```

Funkcja służy do zarejestrowania funkcji zwrotnej, która będzie wywoływana przy każdej zmianie wartości, statusu lub czasu kontrolowanej zmiennej. Parametr *aFunction* określa funkcję monitorującą stan zmiennej – musi ona być zgodna z delegatem *VariableEvent.*

```
delegate void VariableEvent(string aVariableName, VariableState aVariableState);
```

```
Delegate Sub VariableEvent(ByVal aVariableName As String, ByVal aVariableState As VariableState)
```

W momencie wywołania funkcji monitorującej przekazywane są do niej dwa parametry. Parametr *aVariableName* określa nazwę zmiennej. Możliwe jest podłączenie tej samej funkcji monitorującej do wielu zmiennych, a poprzez parametr *aVariableName* można rozróżnić, której zmiennej wywołanie dotyczy. Parametr *aVariableState* przekazuje obiekt, z którego można odczytać aktualny stan zmiennej.

Do poprawnego działania funkcji monitorującej wymagane jest włączenie odświeżania zmiennej w momencie tworzenia obiektu interfejsu *IVariable*. Funkcję monitorującą można wyrejestrować przy pomocy metody *UnregisterStateEvent*.

Przykład:

C#

```
IApplication application;
void ValueFun(string aVariableName, VariableState aVariableState)
{
// kod funkcji zwrotnej
   application.ReportMessage(EvoMessageType.Info, aVariableName + " = " +
```

```
                        aVariableState.FormattedValue);
}
public bool Initialize(string[] aParameters, IApplication aApplication)
{
   application = aApplication;
   IVariable iv = aApplication.GetVariable("varTest", true);
   iv.RegisterStateEvent(ValueFun);
   return false;
}
```

VB

```
Dim application As IApplication
Sub ValueFun(aVariableName As String, aVariableState As VariableState)
   ' kod funkcji zwrotnej
   application.ReportMessage(EvoMessageType.Info, aVariableName & _
                                    " = " & aVariableState.FormattedValue)
End Sub

Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                     IApplication) As Boolean Implements Initialize

   application = aApplication
   Dim iv As IVariable
   iv = application.GetVariable("varTest", True)
   iv.RegisterStateEvent(AddressOf ValueFun)
   Return False

       End Function
```

Powyższy kod tworzy obiekt *IVariable* dla zmiennej *varTest* z włączonym odświeżaniem zmiennej. Następnie rejestrowana jest funkcja monitorująca *ValueFun*, która przy każdej zmienie stanu zmiennej zapisuje w logu komunikatów informację złożoną z nazwy zmiennej i jej aktualnej wartości.

SetVariableBits

bool SetVariableBits(long aValue, long aMask);

bool SetVariableBits(long aValue, long aMask, StatusOpc aStatus);

bool SetVariableBits(long aValue, long aMask, StatusOpc aStatus, DateTime aTimeStamp);

Function **SetVariableBits**(ByVal aValue As **Long**, ByVal aMask As **Long**) As **Boolean**

Function **SetVariableBits**(ByVal aValue As **Long**, ByVal aMask As **Long**, ByVal aStatus As **Askom.Ascent.StatusOpc**) As **Boolean**

Function **SetVariableBits**(ByVal aValue As **Long**, ByVal aMask As **Long**, ByVal aStatus As **Askom.Ascent.StatusOpc**, ByVal aTimeStamp As **Date**) As **Boolean**

Funkcja powoduje wysterowanie zmiennej procesowej na wartość podaną w parametrze *aValue* z dodatkowym maskowaniem zmienianych bitów przez parametr *aMask*. W pierwszej fazie wykonania odczytywana jest bieżąca wartość zmiennej, a następnie bity, którym w masce *aMask* odpowiada wartość 1, zmieniane są zgodnie z wartościami w parametrze *aValue.*

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennej. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennej. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

SetVariableBitsSync

IVariableControlResultSetVariableBitsSync(long aValue, long aMask, StatusOpc aStatus, DateTime aTimeStamp);

Function **SetVariableBitsSync**(ByVal aValue As **Long**, ByVal aMask As **Long**, ByVal aStatus As **Askom.Ascent.StatusOpc**, ByVal aTimeStamp As **Date**) As **IVariableControlResult**

Funkcja powoduje wysterowanie zmiennej procesowej na wartość podaną w parametrze *aValue* z dodatkowym maskowaniem zmienianych bitów przez parametr *aMask*. W pierwszej fazie wykonania odczytywana jest bieżąca wartość zmiennej, a następnie bity, którym w masce *aMask* odpowiada wartość 1, zmieniane są zgodnie z wartościami w parametrze *aValue.*

Parametr *aStatus* ustawia nowy status zmiennej, a parametr *aTimeStamp* nowy stempel czasu zmiennej.

Funkcja pozwala na kontrolę rzeczywistego wykonania sterowania. Poza tym działa identycznie jak metoda *SetVariableBits*.

Funkcja zwraca referencje do obiektu klasy *IVariableControlResult*. Obiekt ten służy do kontroli procesu wykonania operacji.

Przykład zastosowania znajduje się w opisie metody *SetVariableValueSync*.

SetVariableValue

```
bool SetVariableValue(object aValue);

bool SetVariableValue(object aValue, StatusOpc aStatus);

bool SetVariableValue(object aValue, StatusOpc aStatus, DateTime aTimeStamp );

Function SetVariableValue(ByVal aValue As  Object) As Boolean

Function SetVariableValue(ByVal aValue As  Object, ByVal aStatus
As  Askom.Ascent.StatusOpc) As Boolean

Function SetVariableValue(ByVal aValue As  Object, ByVal aStatus
As  Askom.Ascent.StatusOpc,  ByVal aTimeStamp As  Date) As  Boolean
```

Funkcja powoduje wysterowanie zmiennej procesowej na wartość podaną w parametrze *AValue*.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennej. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennej. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Możliwość ustawienia czasu i statusu zależy od typu kanału, do którego zmienna należy. W szczególności, tej funkcjonalności nie mają zazwyczaj kanały fizyczne.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

SetVariableValueSync

IVariableControlResultSetVariableValueSync(object aValue, StatusOpc aStatus, DateTime aTimeStamp);

Function **SetVariableValueSync**(ByVal aValue As **Object**, ByVal aStatus As **Askom.Ascent.StatusOpc**, ByVal aTimeStamp As **Date**) As **IVariableControlResult**

Funkcja powoduje wysterowanie zmiennej procesowej na wartość podaną w parametrze *AValue*.

Parametr *aStatus* ustawia nowy status zmiennej, a parametr *aTimeStamp* nowy stempel czasu zmiennej.

Możliwość ustawienia czasu i statusu zależy od typu kanału, do którego zmienna należy. W szczególności, tej funkcjonalności nie mają zazwyczaj kanały fizyczne.

Funkcja pozwala na kontrolę rzeczywistego wykonania sterowania. Poza tym działa identycznie jak metoda *SetVariableValue*.

Funkcja zwraca referencje do obiektu klasy *IVariableControlResult*. Obiekt ten służy do kontroli procesu wykonania operacji.

Przykład:

```
C#
 IVariable v = aApplication.GetVariable("var1", true);
...
 IVariableControlResult r = v.SetVariableValueSync(100,
Askom.Ascent.StatusOpc.Good,
                                                   DateTime.Now );
 r.Wait();
 if (r.ControlStatus == VariableControlStatus.Error)
 {
    aApplication.ReportMessage(EvoMessageType.Error, "Write failed: "
+ r.ErrorMsg);
 }
 else
 {
    // write successful
 }
 VB
 Dim v As IVariable = aApplication.GetVariable("m1_czas", True)
 ...
 Dim r As IVariableControlResult = v.SetVariableValueSync(100,
                          Askom.Ascent.StatusOpc.Good, DateTime.Now)
 r.Wait()
```

```
       If (r.ControlStatus = VariableControlStatus.Error)
              aApplication.ReportMessage(EvoMessageType.Error, "Write
failed: " + r.ErrorMsg)
       Else
             ‘ write successful
       End If
```

ToggleBits

```
bool ToggleBits(long aMask);

bool SetVariableToggleBits(long aMask, StatusOpc aStatus);

bool ToggleBits(long aMask, StatusOpc aStatus, DateTime aTimeStamp);

Function ToggleBits(ByVal aMask As  Long) As Boolean

Function ToggleBits(ByVal aMask As  Long, ByVal aStatus
As  Askom.Ascent.StatusOpc) As Boolean

Function ToggleBits(ByVal aMask As  Long, ByVal aStatus
As  Askom.Ascent.StatusOpc, ByVal aTimeStamp As Date) As Boolean
```

Funkcja powoduje zmianę wartości wybranych bitów zmiennej procesowej na wartość przeciwną. W pierwszej fazie wykonania odczytywana jest bieżąca wartość zmiennej, a następnie bity, którym w parametrze *aMask* odpowiada wartość 1, zmieniane są na wartość przeciwną. Pozostałe bity zmiennej pozostają bez zmiany.

Wersja wywołania z parametrem *aStatus* dodatkowo ustawia nowy status zmiennej. Wersje bez parametru *aStatus* ustawiają zawsze status na *StatusOpc.Good*.

Wersja wywołania z parametrem *aTimeStamp* dodatkowo ustawia nowy stempel czasu zmiennej. Wersje bez parametru *aTimeStamp* ustawiają zawsze stempel czasu i na chwilę wykonania funkcji.

Funkcja zwraca wartość *false* w przypadku niepowodzenia. Ponieważ funkcja wykonywana jest w sposób asynchroniczny, to test poprawności dotyczy tylko spełnienia warunków wstępnych. W przypadku błędu komunikacji zostanie on zgłoszony w panelu komunikatów.

ToggleBitsSync

```
IVariableControlResultToggleBitsSync(long aMask, StatusOpc aStatus, DateTime
aTimeStamp);
```

```
Function ToggleBitsSync(ByVal aMask As  Long, ByVal aStatus
As  Askom.Ascent.StatusOpc, ByVal aTimeStamp As Date) As
IVariableControlResult
```

Funkcja powoduje zmianę wartości wybranych bitów zmiennej procesowej na wartość przeciwną. W pierwszej fazie wykonania odczytywana jest bieżąca wartość zmiennej, a następnie bity, którym w parametrze *aMask* odpowiada wartość 1, zmieniane są na wartość przeciwną. Pozostałe bity zmiennej pozostają bez zmiany.

Parametr *aStatus* ustawia nowy status zmiennej, a parametr *aTimeStamp* nowy stempel czasu zmiennej.

Funkcja pozwala na kontrolę rzeczywistego wykonania sterowania. Poza tym działa identycznie jak metoda Toggle*Bits*.

Funkcja zwraca referencje do obiektu klasy *IVariableControlResult*. Obiekt ten służy do kontroli procesu wykonania operacji.

Przykład zastosowania znajduje się w opisie metody *SetVariableValueSync*.

UnregisterStateEvent

```
void UnregisterStateEvent();
```

```
Sub UnregisterStateEvent()
```

Wyrejestrowanie funkcji monitorującej wartości zmiennej, która została wcześniej zarejestrowana przy pomocy funkcji *RegisterStateEvent*.

VariableAttribute

```
object VariableAttribute(string aAttributeName);
```

Function **VariableAttribute** (ByVal aAttributeName As **String**) As **Object**

Funkcja zwraca definicję atrybutu określonego parametrem *aAttributeName*.

3.5 Klasa VariableState

3.5.1 Właściwości

AsDouble

```
double AsDouble { get; }
```

ReadOnly Property **AsDouble** As **Double**

Właściwość zwraca wartość zmiennej procesowej po konwersji na liczbę zmiennoprzecinkową. Wymagane jest spełnienie warunku, że typ zmiennej procesowej dopuszcza konwersję na liczbę zmiennoprzecinkową. Wartość nieskonwertowana dostępna jest za pośrednictwem właściwości *Value*.

AsInteger

```
int AsInteger { get; }
```

ReadOnly Property **AsInteger** As **Integer**

Właściwość zwraca wartość zmiennej procesowej po konwersji na liczbę 32-bitową liczbę całkowitą. Wymagane jest spełnienie warunku, że typ zmiennej procesowej dopuszcza konwersję na liczbę całkowitą. Wartość nieskonwertowana dostępna jest za pośrednictwem właściwości *Value*.

FormattedStatus

```
string FormattedStatus { get; }
```

ReadOnly Property **FormattedStatus** As **String**

Właściwość zwraca szczegółowy status wartości zmiennej procesowej w konwencji statusu interfejsu komunikacyjnego OPC po skonwertowaniu na postać tekstową, opisującą w przyjazny sposób wszystkie ustawione statusy częściowe.

FormattedValue

```
string FormattedValue { get; }
```

ReadOnly Property **FormattedValue** As **String**

Właściwość zwraca wartość tekstową zmiennej procesowej utworzoną na podstawie formatu zdefiniowanego w bazie definicji zmiennych.

IsBad

```
bool IsBad { get; }
```

ReadOnly Property **IsBad** As **Boolean**

Właściwość określa, czy status wartości zmiennej jest niepoprawny.

IsGood

```
bool IsGood { get; }
```

ReadOnly Property **IsGood** As **Boolean**

Właściwość określa, czy status wartości zmiennej jest poprawny.

IsUncertain

```
bool IsUncertain { get; }
```

ReadOnly Property **IsUncertain** As **Boolean**

Właściwość określa, czy status wartości zmiennej jest niepewny.

RawValue

```
object RawValue { get; }
```

ReadOnly Property **RawValue** As **Object**

Właściwość zwraca wartość surową zmiennej procesowej. Jest to oryginalna wartość zmiennej odczytana przez drajwer kanału komunikacyjnego, przed zastosowaniem funkcji przeliczającej zdefiniowanej w bazie zmiennych w atrybucie *Funkcja przeliczająca odczytu*.

UWAGA: w przypadku korzystania z kanału komunikacyjnego *Asix,* wartość właściwości jest już wstępnie przeliczona przez moduł komunikacyjny Asmen na podstawie atrybutu *Funkcja przeliczająca*.

Status

```
StatusOpc Status { get; }
```

ReadOnly Property **Status** As **Askom.Ascent.StatusOpc**

Właściwość zwraca szczegółowy status wartości zmiennej procesowej w konwencji statusu interfejsu komunikacyjnego OPC. Do prostych testów statusu zmiennej zalecane jest korzystanie z właściwości *IsGood*, *IsBad* i *IsUncertain*. Do konwersji na postać tekstową można użyć właściwości *FormattedStatus*.

Więcej informacji dotyczących testowania statusu znajduje się w opisie właściwości *Status* klasy *ArcVariableState*.

TimeStamp

```
DateTime TimeStamp { get; }
```

ReadOnly Property **TimeStamp** As **Date**

Właściwość zwraca lokalny czas odczytu wartości zmiennej.

TimeStampUtc

```
DateTime TimeStampUtc { get; }
```

ReadOnly Property **TimeStampUtc** As **Date**

Właściwość zwraca czas UTC odczytu wartości zmiennej.

Value

```
object Value { get; }
```

ReadOnly Property **Value** As **Object**

Właściwość zwraca wartość zmiennej procesowej po zastosowaniu wszystkich przeliczeń wynikających z definicji zmiennej w bazie definicji zmiennych.

3.6 Klasa ArcQuery

Klasa *ArcQuery* służy do określenia parametrów odczytu wartości archiwalnych zmiennej procesowej. Obiekty *ArcQuery* używane są w wywołaniach metody *OrderArcData* interfejsu *IVariable*.

3.6.1 Konstruktor

ArcQuery (DateTime aStartTime,DateTime aEndTime)

Public Sub **New**(ByVal aStartTime As **Date**, ByVal aEndTime As **Date**)

Konstruktor tworzy obiekt parametrów dla zapytania dotyczącego odczytu danych surowych. Parametry *aStartTime* i *aEndTime* oznaczają kolejno początek i koniec okresu czasu, z jakiego dane należy odczytać.

ArcQuery (DateTime aStartTime, DateTime aEndTime, string aAggregate, TimeSpan aAggregateInterval)

Public Sub **New**(ByVal aStartTime As **Date**, ByVal aEndTime As **Date**, ByVal aAggregate As **String**, ByVal aAggregateInterval As **System.TimeSpan**)

Konstruktor tworzy obiekt parametrów dla zapytania dotyczącego odczytu danych agregowanych. Parametry *aStartTime* i *aEndTime* oznaczają kolejno początek i koniec okresu czasu, z jakiego dane

należy odczytać. Parametr *aAggregate* określa nazwę zastosowanego agregatu, a parametr *aAggregateInterval* okres jego wyliczania.

Zobacz też:

Rodzaje agregatów danych archiwalnych - Załącznik 1

3.6.2 Właściwości

Aggregate

```
string Aggregate { get; }
```

ReadOnly Property **Aggregate** As **String**

Właściwość zwraca nazwę agregatu użytego w zapytaniu o dane archiwalne.

AggregateInterval

```
TimeSpan TimeStamp { get; }
```

ReadOnly Property **AggregateInterval** As **System.TimeSpan**

Właściwość zwraca okres wyliczania agregatu użyty w zapytaniu o dane archiwalne.

EndTime

```
DateTime EndTime { get; }
```

ReadOnly Property **EndTime** As **Date**

Właściwość zwraca koniec okresu czasu, z którego dane archiwalne mają zostać odczytane.

ProcessedDataQuery

```
bool ProcessedDataQuery { get; }
```

ReadOnly Property **ProcessedDataQuery** As **Boolean**

Właściwość zwraca informację, jakiego rodzaju danych archiwalnych zapytanie dotyczy. Wartość *true* oznacza odczyt danych agregowanych, a wartość *false* odczyt danych surowych.

StartTime

```
DateTime StartTime { get; }
```

ReadOnly Property **StartTime** As **Date**

Właściwość zwraca początek okresu czasu, z którego dane archiwalne mają zostać odczytane.

3.7 Interfejs IArcQueryResult

Interfejs *IArcQueryResult* służy do kontroli wykonania operacji odczytu danych archiwalnych zainicjowanej metodą *OrderArcData* interfejsu *IVariable*. Właściwości i metody interfejsu pozwalają na sprawdzenie, czy i w jaki sposób operacja się zakończyła oraz udostępniają odczytane wartości.

3.7.1 Właściwości

ErrorDescription

```
string ErrorDescription { get; }
```

ReadOnly Property **ErrorDescription** As **String**

Właściwość zwraca tekst opisujący błąd odczytu danych archiwalnych (gdy właściwość *ResultStatus* jest równa *ArcResultStatus.Error*).

Query

ArcQuery Query { get; }

ReadOnly Property **Query** As **ArcQuery**

Właściwość zwraca obiekt klasy *ArcQuery*, z którego można odczytać parametry odczytu danych archiwalnych, użyte w funkcji *OrderArcData*.

ResultStatus

ArcResultStatus ResultStatus { get; }

ReadOnly Property **ResultStatus** As **ArcResultStatus**

Właściwość zwraca aktualny status wykonywania operacji odczytu danych archiwalnych. Typ wyliczeniowy *ArcResultStatus* może przyjmować następujące wartości:

- **Processing – odczyt danych jest w trakcie wykonywania**
- **Ready – odczyt danych został zakończony poprawnie**
- **Error – odczyt danych zakończył się błędem**

Values

ArcVariableState[ ] Values { get; }

ReadOnly Property **ArcVariableState** As **ArcVariableState**( )

Właściwość zwraca tablicę punktów przebiegu archiwalnego uzyskaną w wyniku wykonania funkcji *OrderArcData*.

VariableName

```
string VariableName { get; }
```

```
ReadOnly Property VariableName As String
```

Właściwość zwraca nazwę zmiennej, dla której odczytywane są dane archiwalne.

3.7.2 Metody

CancelQuery

```
void CancelQuery( );
```

```
Sub CancelQuery ( )
```

Metoda *CancelQuery* pozwala na przerwanie wykonywanej operacji odczytu danych archiwalnych.

RegisterQueryResultEvent

```
void RegisterQueryResultEvent(ArcQueryResultEvent aFunction);
```

```
Sub RegisterQueryResultEvent(ByVal aFunction As  ArcQueryResultEvent)
```

Funkcja służy do zarejestrowania funkcji zwrotnej, która zostanie wywoływana po zakończeniu operacji odczytu danych archiwalnych. Parametr *aFunction* określa funkcję zwrotną – musi ona być zgodna z delegatem *ArcQueryResultEvent.*

delegate void ArcQueryResultEvent(IArcQueryResult aQueryResult);

Delegate Sub **ArcQueryResultEvent(**ByVal aQueryResult As **IArcQueryResult)**

W parametrze *aQueryResult* do funkcji zwrotnej przekazywana jest referencja do interfejsu *IArcQueryResult,* w kontekście którego rejestracja została wykonana.

Funkcję oczekiwania na koniec odczytu można wyrejestrować przy pomocy metody *UnregisterQueryResultEvent*.

Przykład:

C#

```
IApplication application;
void ArcQueryFun(IArcQueryResult aQueryResult)
{
   if (aQueryResult.ResultStatus == ArcResultStatus.Error)
      application.ReportMessage(EvoMessageType.Error, "OrderArcData error: " +
                                aQueryResult.ErrorDescription);
   else
   {
      foreach (ArcVariableState v in aQueryResult.Values)
         application.ReportMessage(EvoMessageType.Error, v.TimeStamp.ToString() +
                                           ": " + v.Value.ToString());

   }
   application.TerminateScript(true);
}

public bool Initialize(string[] aParameters, IApplication aApplication)
{
   application = aApplication;
   IVariable iv = aApplication.GetVariable("A004", false);

   IArcQueryResult qr = iv.OrderArcData(new ArcQuery(DateTime.Today,
                            DateTime.Now,"average",TimeSpan.FromMinutes(15)));
   if (qr.ResultStatus == ArcResultStatus.Error)
      aApplication.ReportMessage(EvoMessageType.Error, "OrderArcData error: " +
                                qr.ErrorDescription);
   else
     qr.RegisterQueryResultEvent(ArcQueryFun);

   return false;
}
```

VB

```
Dim application As IApplication
Sub QueryArcFun(aQueryResult As IArcQueryResult)

   If (aQueryResult.ResultStatus = ArcResultStatus.Error)
      application.ReportMessage(EvoMessageType.Error, _
                        "OrderArcData error: " + aQueryResult.ErrorDescription)
   Else
      For Each v As ArcVariableState In aQueryResult.Values
         application.ReportMessage(EvoMessageType.Error, _
```

```
                                    v.TimeStamp.ToString() + ": " + v.Value.ToString())
      Next v
   End If
   application.TerminateScript(True)
End Sub

Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
               IApplication) As Boolean Implements IScript.Initialize

   application = aApplication
   Dim iv As IVariable
   iv = aApplication.GetVariable("A004", False)

   Dim qr As IArcQueryResult
   qr = iv.OrderArcData(New ArcQuery(Date.Today, Date.Now, _
                     "average", TimeSpan.FromMinutes(15)))

   If (qr.ResultStatus = ArcResultStatus.Error)
      aApplication.ReportMessage(EvoMessageType.Error, _
                                    "OrderArcData error: " + qr.ErrorDescription)
   Else
      qr.RegisterQueryResultEvent(AddressOf QueryArcFun)
   End If
   Return False

End Function
```

Przekazany w wywołaniu metody *OrderArcData* obiekt klasy *ArcQuery* powoduje odczyt 15-minutowych wartości średnich z okresu od początku dnia do chwili bieżącej. Jeżeli inicjacja odczytu wykonała się poprawnie, to rejestrowana jest funkcja *ArcQueryFun,* która zostanie wykonana po zakończeniu operacji. Wewnątrz funkcji zwrotnej *ArcQueryFun* wykonywany jest zapis do logu komunikatów wszystkich wartości archiwalnych, które zostały zwrócone poprzez właściwość *Values* obiektu klasy *IArcQueryResult* przekazanego w wywołaniu funkcji. Następnia praca skryptu zostaje zakończona.

UnregisterQueryResultEvent

```
void UnregisterQueryResultEvent();
```

```
Sub UnregisterQueryResultEvent()
```

Wyrejestrowanie funkcji oczekującej na zakończenie operacji odczytu, która została wcześniej zarejestrowana przy pomocy funkcji *RegisterQueryResultEvent*.

Wait

```
bool Wait();
```

Function **Wait**() As **Boolean**

Funkcja zawiesza wykonywanie skryptu do momentu zakończenia realizacji operacji odczytu danych archiwalnych. Funkcja ma wbudowane 5-minutowe ograniczenie czasu oczekiwania. Jeżeli dane nie zostaną odczytane w tym czasie, to funkcja zwraca wartość *false*.

Alternatywą dla funkcji *Wait* jest użycie funkcji *Sleep* interfejsu *IApplication* połączone z monitorowaniem wartości statusu *QueryResult*.

3.8 Klasa ArcVariableState

Obiekt klasy *ArcVariableState* przechowuje informacje dotyczące pojedynczej wartości archiwalnej zmiennej procesowej. Tablicę wartości archiwalnych, będącą wynikiem zapytania o przebieg archiwalny zmiennej, można otrzymać poprzez właściwość *Values* interfejsu *IArcQueryResult*.

3.8.1 Właściwości

AsDouble

```
double AsDouble { get; }
```

ReadOnly Property **AsDouble** As **Double**

Właściwość zwraca wartość archiwalną zmiennej procesowej z momentu czasu określonego właściwością *TimeStamp* po konwersji na liczbę zmiennoprzecinkową. Wymagane jest spełnienie warunku, że typ zmiennej procesowej dopuszcza konwersję na liczbę zmiennoprzecinkową. Wartość nieskonwertowana dostępna jest za pośrednictwem właściwości *Value*.

AsInteger

```
int AsInteger { get; }
```

ReadOnly Property **AsInteger** As **Integer**

Właściwość zwraca wartość archiwalną zmiennej procesowej z momentu czasu określonego właściwością *TimeStamp* po konwersji na liczbę 32-bitową liczbę całkowitą. Wymagane jest spełnienie warunku, że typ zmiennej procesowej dopuszcza konwersję na liczbę całkowitą. Wartość nieskonwertowana dostępna jest za pośrednictwem właściwości *Value*.

EndTimeStamp

```
DateTime EndTimeStamp { get; }
```

ReadOnly Property **EndTimeStamp** As **Date**

Właściwość zwraca stempel czasu lokalnego dla końca przedziału wartości próbki.

EndTimeStampUTC

```
DateTime EndTimeStampUTC { get; }
```

ReadOnly Property **EndTimeStampUTC** As **Date**

Właściwość zwraca stempel czasu UTC dla końca przedziału wartości próbki.

FormattedStatus

```
string FormattedStatus { get; }
```

ReadOnly Property **FormattedStatus** As **String**

Właściwość zwraca szczegółowy status archiwalnej wartości zmiennej procesowej w konwencji statusu interfejsu komunikacyjnego OPC po skonwertowaniu na postać tekstową, opisującą w przyjazny sposób wszystkie ustawione statusy częściowe.

IsBad

```
bool IsBad { get; }
```

ReadOnly Property **IsBad** As **Boolean**

Właściwość określa, czy status wartości zmiennej jest niepoprawny.

IsGood

```
bool IsGood { get; }
```

ReadOnly Property **IsGood** As **Boolean**

Właściwość określa, czy status wartości zmiennej jest poprawny.

IsUncertain

```
bool IsUncertain { get; }
```

ReadOnly Property **IsUncertain** As **Boolean**

Właściwość określa, czy status wartości zmiennej jest niepewny.

Status

```
Askom.Ascent.StatusOpc Status { get; }
```

ReadOnly Property **Status** As **Askom.Ascent.StatusOpc**

Właściwość zwraca szczegółowy status wartości archiwalnej zmiennej procesowej w konwencji statusu interfejsu komunikacyjnego OPC. Do prostych testów statusu zmiennej zalecane jest korzystanie z właściwości *IsGood*, *IsBad* i *IsUncertain*. Do konwersji na postać tekstową można użyć właściwości *FormattedStatus*.

StatusOpc jest typem wyliczeniowym (enum). Do bardziej zaawansowanych testów statusu można użyć elementów typu wyliczeniowego w połączeniu z operatorami bitowych operacji logicznych. Można też użyć obiekt klasy *Askom.Ascent.AsQuality*, który posiada szereg właściwości ułatwiających zawansowane testowanie statusu OPC. Obiekt *AsQuality* należy utworzyć na podstawie statusu OPC wartości zmiennej, zgodnie z poniższym przykładem.

Przykład:

C#

```
IVariable v = aApplication.GetVariable("var", true);

...

Askom.Ascent.AsQuality q = new Askom.Ascent.AsQuality(v.VariableState.Status);
if (q.IsWaiting)
{
...
}
```

VB

```
Dim v As IVariable = aApplication.GetVariable("var", True)

...

Dim q As Askom.Ascent.AsQuality = New Askom.Ascent.AsQuality(v.VariableState.Status)
If (q.IsWaiting)
...
End If
```

TimeStamp

```
DateTime TimeStamp { get; }
```

ReadOnly Property **TimeStamp** As **Date**

Właściwość zwraca stempel czasu wartości archiwalnej.

TimeStampUTC

```
DateTime TimeStampUTC { get; }
```

ReadOnly Property **TimeStampUTC** As **Date**

Właściwość zwraca stempel czasu UTC dla wartości próbki.

Value

```
object Value { get; }
```

ReadOnly Property **Value** As **Object**

Właściwość zwraca wartość archiwalną zmiennej procesowej z momentu czasu określonego właściwością *TimeStamp.*

3.9 Interfejs IVariableControlState

Interfejs *IVariableControlState* służy do kontroli wykonania operacji zapisu zmiennych procesowych zainicjowanej metodą *SetVariablesSync* interfejsu *IApplication*. Właściwości i metody interfejsu pozwalają na sprawdzenie, czy i w jaki sposób operacja się zakończyła.

Przykład zastosowania znajduje się w opisie metody *SetVariablesSync* interfejsu *IApplication*.

3.9.1 Właściwości

ErrorMsg

```
string ErrorMsg { get; }
```

ReadOnly Property **ErrorMsg** As **String**

Właściwość zwraca tekst opisujący ewentualny błąd operacji zapisu (gdy właściwość *ControlStatus* jest równa *VariableControlStatus.Error*).

ControlStatus

```
VariableControlStatus ControlStatus { get; }
```

ReadOnly Property **ControlStatus** As **VariableControlStatus**

Właściwość zwraca aktualny status wykonywania operacji zapisu. Typ wyliczeniowy *VariableControlStatus* może przyjmować następujące wartości:

- Pending – operacja zapisu jest w trakcie wykonywania
- Success – operacja zakończona powodzeniem
- Error – operacja zapisu zakończona błędem

VariableName

```
string VariableName { get; }
```

```
ReadOnly Property VariableName As String
```

Właściwość zwraca nazwę zmiennej, której dotyczy kontrolowana operacja zapisu.

3.10 Interfejs IVariableControlResult

Interfejs *IVariableControlResult* służy do kontroli wykonania operacji zapisu zmiennych procesowych zainicjowanej metodami *SetVariableBitsSync*, *SetVariableValueSync* i *ToggleBitsSync* interfejsu *IVariable*. Właściwości i metody interfejsu pozwalają na sprawdzenie, czy i w jaki sposób operacja się zakończyła.

Przykład zastosowania znajduje się w opisie metody *SetVariableValueSync* interfejsu *IVariable*.

3.10.1 Właściwości

ErrorMsg

```
string ErrorMsg { get; }

ReadOnly Property ErrorMsg As String
```

Właściwość zwraca tekst opisujący ewentualny błąd operacji zapisu (gdy właściwość *ControlStatus* jest równa *VariableControlStatus.Error*).

ControlStatus

```
VariableControlStatus ControlStatus { get; }

ReadOnly Property ControlStatus As VariableControlStatus
```

Właściwość zwraca aktualny status wykonywania operacji zapisu. Typ wyliczeniowy *VariableControlStatus* może przyjmować następujące wartości:

- Pending – operacja zapisu jest w trakcie wykonywania
- Success – operacja zakończona powodzeniem
- Error – operacja zapisu zakończona błędem

3.10.2 Metody

Wait

```
bool Wait();
```

```
Function Wait() As Boolean
```

Funkcja zawiesza wykonywanie skryptu do momentu zakończenia realizacji operacji zapisu zmiennej.

Alternatywą dla funkcji *Wait* jest użycie funkcji *Sleep* interfejsu *IApplication* połączone z monitorowaniem wartości statusu *ControlStatus*.

3.11 Interfejs IAlarm

Interfejs *IAlarm* pozwala na uzyskanie informacji dotyczących definicji alarmu i jego aktualnego stanu. Pozwala też na kontrolę stanu alarmu: zgłaszanie, kończenie i potwierdzanie alarmu.

Referencję do interfejsu *IAlarm* dla wybranego alarmu można uzyskać poprzez wywołanie funkcji *GetAlar* z interfejsu *IApplication*.

3.11 Interfejs IAlarm

Interfejs *IAlarm* pozwala na uzyskanie informacji dotyczących definicji alarmu i jego aktualnego stanu. Pozwala też na kontrolę stanu alarmu: zgłaszanie, kończenie i potwierdzanie alarmu.

Referencję do interfejsu *IAlarm* dla wybranego alarmu można uzyskać poprzez wywołanie funkcji *GetAlar* z interfejsu *IApplication*.

3.11.1 Właściwości

AlarmState

```
AlarmState AlarmState { get; }
```

ReadOnly Property **AlarmState** As **AlarmState**

Właściwość zwraca obiekt klasy *AlarmState*, z którego można odczytać informacje dotyczące aktualnego stanu alarmu kontrolowanego przez posiadaną referencję do interfejsu *IAlarm*. Jeżeli w logu alarmów aktywnych zarejestrowane jest więcej niż jedno zdarzenie, to zwracany jest stan zdarzenia najmłodszego.

Definition

```
AlarmProperties Definition { get; }
```

ReadOnly Property **Definition** As **AlarmProperties**

Właściwość zwraca obiekt klasy *AlarmProperties*, z którego można odczytać informacje dotyczące definicji alarmu kontrolowanego przez posiadaną referencję do interfejsu *IAlarm*.

Domain

```
string Domain { get; }
```

ReadOnly Property **Domain** As **String**

Właściwość zwraca nazwę domeny alarmowej, do której należy alarm kontrolowany przez posiadaną referencję do interfejsu *IAlarm*.

Id

```
string Id { get; }
```

ReadOnly Property **Id** As **String**

Właściwość zwraca identyfikator alarmu kontrolowanego przez posiadaną referencję do interfejsu *IAlarm*.

3.11.2 Metody

Accept

```
bool Accept();
```

Function **Accept**() As **Boolean**

Metoda służy do potwierdzenia alarmu. Jej efekt działania jest identyczny z potwierdzeniem alarmu przez operatora w tabeli alarmów. Funkcja zwraca wstępny status wykonania. Wartość *true* oznacza, że operacja została przyjęta do realizacji. Potwierdzenie może jednak nie dojść do skutku z powodu niespełnienia warunków sprawdzanych w późniejszych fazach realizacji operacji, np. brak uprawnienia stanowiska do potwierdzania alarmów w domenie.

Cancel

```
bool Cancel(params object[ ] aParameters);

bool Cancel(DateTime aEndTime, params object[ ] aParameters);

Function Cancel(ByVal ParamArray aParameters( ) As Object) As Boolean

Function Cancel(ByVal aEndTime As Date , ByVal ParamArray aParameters( )
As Object) As Boolean
```

Metoda pozwala na zgłoszenie końca zdarzenia alarmowego. Opcjonalna tabela parametrów *aParameters* służy do przekazania dodatkowych danych związanych ze zdarzeniem. Znaczenie tych danych zależy wyłącznie od projektanta aplikacji. Wersja wywołania z parametrem *aEndTime* pozwala na jawne podanie czasu zdarzenia. W przypadku wywołania bez parametru *aEndTime* czas zdarzenia jest ustawiany automatycznie na chwilę bieżącą. Metodę można wykonać tylko dla alarmów, które należą do zewnętrznej strategii rozpoznania alarmów.

Funkcja zwraca wstępny status wykonania. Wartość *true* oznacza, że operacja została przyjęta do realizacji. Zgłoszenie może jednak zostać zignorowane z powodu niespełnienia warunków sprawdzanych w późniejszych fazach realizacji operacji, np. stanowisko nie jest kontrolerem aktywnym domeny i użyto zwykłej strategii zewnętrznej (nie globalnej).

Raise

```
bool Raise(params object[ ] aParameters);

bool Raise(DateTime aStartTime, params object[ ] aParameters);

Function Raise(ByVal ParamArray aParameters( ) As Object) As Boolean

Function Raise(ByVal aStartTime As Date , ByVal ParamArray aParameters( )
As Object) As Boolean
```

Metoda pozwala na zgłoszenie początku zdarzenia alarmowego. Opcjonalna tabela parametrów *aParameters* służy do przekazania dodatkowych danych związanych ze zdarzeniem. Znaczenie tych danych zależy wyłącznie od projektanta aplikacji. Wersja wywołania z parametrem *aStartTime* pozwala na jawne podanie czasu zdarzenia. W przypadku wywołania bez parametru *aStartTime* czas zdarzenia jest ustawiany automatycznie na chwilę bieżącą. Metodę można wykonać tylko dla alarmów, które należą do zewnętrznej strategii rozpoznania alarmów.

Funkcja zwraca wstępny status wykonania. Wartość *true* oznacza, że operacja została przyjęta do realizacji. Zgłoszenie może jednak zostać zignorowane z powodu niespełnienia warunków sprawdzanych w późniejszych fazach realizacji operacji, np. stanowisko nie jest kontrolerem aktywnym domeny i użyto zwykłej strategii zewnętrznej (nie globalnej).

RegisterStateEvent

```
void RegisterStateEvent(AlarmEvent aFunction);
```

```
Sub RegisterStateEvent(ByVal aFunction As AlarmEvent)
```

Funkcja służy do zarejestrowania funkcji zwrotnej, która będzie wywoływana przy każdej zmianie stanu kontrolowanego alarmu. Dotyczy to zarówno zdarzeń początku i końca alarmu, jak i zmian stanu spowodowanych działaniami operatora, np. potwierdzenie alarmu.

Parametr *aFunction* określa funkcję monitorującą stan alarmu – musi ona być zgodna z delegatem *AlarmEvent.*

```
delegate void AlarmEvent(string aDomain, string aAlarm, AlarmState aAlarmState,
AlarmEventType aEventType);
```

Delegate Sub **AlarmEvent(**ByVal aDomain As **String**, ByVal aAlarm As **String**, ByVal aAlarmState As **AlarmState** ,ByVal aEventType As **AlarmEventType)**

W momencie wywołania funkcji monitorującej przekazywane są do niej cztery parametry. Parametry *aDomain* i *aAlarm* określają odpowiednio nazwę domeny i identyfikator alarmu, którego wywołanie funkcji zwrotnej dotyczy. Parametr *aAlarmState* przekazuje obiekt, z którego można odczytać aktualny stan alarmu. Parametr *aEventType* określa rodzaj zdarzenia. Typ wyliczeniowy *AlarmEventType* przybiera następujące wartości: Alarm*Begin* (wykrycie początku alarmu) *, AlarmEnd* (wykrycie końca alarmu), *AlarmManagement* (zmiana stanu wynikająca z działań operatora, np. potwierdzenia alarmu), *DomainStart* (uruchomienie pracy domeny), *DomainStop* (zatrzymanie pracy domeny).

Funkcję monitorującą można wyrejestrować przy pomocy metody *UnregisterStateEvent*.

Przykład:

C#

```
IApplication application;
void AlarmFun(string aDomain, string aAlarm, AlarmState aAlarmState,
              AlarmEventType aEventType)
{
```

```
// kod funkcji zwrotnej
         if (aEventType == AlarmEventType.AlarmBegin)
            application.ReportMessage(EvoMessageType.Info, aAlarm + ": alarm rozpoczęty");
      }

      public bool Initialize(string[] aParameters, IApplication aApplication)
      {
         application = aApplication;
         IAlarm ia = application.GetAlarm("Blok1","DemoB1");
         ia.RegisterStateEvent(AlarmFun);
         return false;
      }
```

VB

```
Dim application As IApplication
Sub AlarmFun(aDomain As String, aName As String, aAlarmState As AlarmState, aEventType As
AlarmEventType)

'  kod funkcji zwrotnej
   If aEventType = AlarmEventType.AlarmBegin Then
       application.ReportMessage(EvoMessageType.Info, aName & ": alarm rozpoczęty")
   End If

End Sub

Public Function Initialize(ByVal aParameters() As String, ByVal aApplication As _
                           IApplication) As Boolean Implements IScript.Initialize

   application = aApplication
   Dim ia As IAlarm
   ia = application.GetAlarm("Blok1", "DemoB1")
   ia.RegisterStateEvent(AddressOf AlarmFun)
   Return False

       End Function
```

Powyższy kod tworzy obiekt *IAlarm* dla alarmu *DemoB1* z domeny *Blok1*. Następnie rejestrowana jest funkcja monitorująca *AlarmFun*, która przy każdej zmianie stanu alarmu sprawdza rodzaj zdarzenia i zapisuje w logu komunikatów informację o zdarzeniach początku alarmu.

UnregisterStateEvent

```
void UnregisterStateEvent();
```

Sub **UnregisterStateEvent**()

Wyrejestrowanie funkcji monitorującej zmiany stanu alarmu, która została wcześniej zarejestrowana przy pomocy funkcji *RegisterStateEvent*.

3.12 Klasa AlarmState

Obiekt *AlarmState* przechowuje informacje dotyczące aktualnego stanu alarmu. Obiekty *AlarmState* można uzyskać za pośrednictwem właściwości *AlarmState* interfejsu *IAlarm.* Dostarczane są także jako parametr funkcji monitorującej zmiany stanu alarmów, rejestrowanej funkcją *RegisterStateEvent* interfejsu *IAlarm*.

Dostęp do stanu alarmów realizowany jest na bazie logu alarmów aktywnych – oznacza to, że szczegółowe dane dotyczące stanu alarmu są dostępne tylko wtedy, gdy zdarzenie dotyczące tego alarmu znajduje się w logu alarmów aktywnych. Gdy dane nie są dostępne, właściwość *IsActive* obiektu jest równa *false*.

3.12.1 Właściwości

AcceptanceOperatorId

```
string AcceptanceOperatorId { get; }
```

ReadOnly Property **AcceptanceOperatorId** As **String**

Właściwość zwraca identyfikator użytkownika, który potwierdził zdarzenie alarmowe.

AcceptanceStationName

```
string AcceptanceStationName { get; }
```

ReadOnly Property **AcceptanceStationName** As **String**

Właściwość zwraca nazwę stanowiska, na którym zostało potwierdzone zdarzenie alarmowe.

AcceptanceTimeStamp

```
DateTime AcceptanceTimeStamp { get; }
```

ReadOnly Property **AcceptanceTimeStamp** As **Date**

Właściwość zwraca czas potwierdzenia zdarzenia alarmowego przez użytkownika.

BeginDetectionTimeStamp

```
DateTime BeginDetectionTimeStamp { get; }
```

ReadOnly Property **BeginDetectionTimeStamp** As **Date**

Właściwość zwraca czas wykrycia początku zdarzenia alarmowego.

BeginExprValue

```
object BeginExprValue { get; }
```

ReadOnly Property **BeginExprValue** As **Object**

Właściwość zwraca wartość wyrażenia powiązanego z alarmem, wyliczoną w momencie wykrycia początku alarmu.

BeginTimeStamp

```
DateTime BeginTimeStamp { get; }
```

ReadOnly Property **BeginTimeStamp** As **Date**

Właściwość zwraca czas początku zdarzenia alarmowego.

EndDetectionTimeStamp

```
DateTime EndDetectionTimeStamp { get; }
```

ReadOnly Property **EndDetectionTimeStamp** As **Date**

Właściwość zwraca czas wykrycia końca zdarzenia alarmowego.

EndExprValue

```
object EndExprValue { get; }
```

ReadOnly Property **EndExprValue** As **Object**

Właściwość zwraca wartość wyrażenia powiązanego z alarmem, wyliczoną w momencie wykrycia końca alarmu.

EndTimeStamp

```
DateTime EndTimeStamp { get; }
```

ReadOnly Property **EndTimeStamp** As **Date**

Właściwość zwraca czas końca zdarzenia alarmowego.

FormattedBeginText

```
string FormattedBeginText { get; }
```

ReadOnly Property **FormattedBeginText** As **String**

Właściwość zwraca tekst opisujący początek zdarzenia alarmowego. W zwróconym tekście wygenerowane są wszystkie dynamicznie tworzone wstawki związane z parametrami zgłoszenia.

FormattedEndText

```
string FormattedEndText { get; }
```

ReadOnly Property **FormattedEndText** As **String**

Właściwość zwraca tekst opisujący koniec zdarzenia alarmowego. W zwróconym tekście wygenerowane są wszystkie dynamicznie tworzone wstawki związane z parametrami zgłoszenia.

IsAccepted

```
bool IsAccepted { get; }
```

ReadOnly Property **IsAccepted** As **Boolean**

Właściwość określa, czy zdarzenie alarmowe zostało potwierdzone przez użytkownika.

IsActive

```
bool IsActive { get; }
```

ReadOnly Property **IsActive** As **Boolean**

Właściwość określa, czy zdarzenie dotyczące kontrolowanego przez obiekt alarmu znajduje się w logu alarmów aktywnych. Wartość *false* oznacza, że brak jest informacji o alarmie i wszystkie pozostałe właściwości obiektu *AlarmState* mają nieokreśloną wartość.

IsExcluded

```
bool IsExcluded { get; }
```

ReadOnly Property **IsExcluded** As **Boolean**

Właściwość określa, czy alarm jest wykluczony z obsługi. Alarmy wykluczone nie są pokazywane na tabelach alarmów, ale dane o nich są przechowywane w logu alarmów aktywnych i są dostępne dla skryptów.

IsFinished

```
bool IsFinished { get; }
```

ReadOnly Property **IsFinished** As **Boolean**

Właściwość określa, czy alarm przechowywany w logu alarmów aktywnych został już zakończony. W zależności od konfiguracji, log alarmów aktywnych może przez pewien czas pamiętać dane dotyczące zdarzenia już zakończonego. Alarm należy traktować jako aktywny (niezakończony) wtedy, gdy właściwość *IsActive* jest równa *true* i właściwość *IsFinished* jest równa *false*.

WasFiltered

```
bool WasFiltered { get; }
```

ReadOnly Property **WasFiltered** As **Boolean**

Właściwość określa, czy zdarzenie alarmowe zarejestrowane w logu podlegało mechanizmowi filtrów czasowych.

3.13 Klasa AlarmProperties

Obiekt *AlarmProperties* przechowuje informacje dotyczące definicji alarmu. Obiekt można uzyskać za pośrednictwem właściwości *Definition* interfejsu *IAlarm*.

3.13.1 Właściwości

Action

```
string Action { get; }
```

ReadOnly Property **Action** As **String**

Właściwość zwraca treść akcji operatorskiej zdefiniowanej dla alarmu.

DetectionParameters

```
string DetectionParameters { get; }
```

ReadOnly Property **DetectionParameters** As **String**

Właściwość zwraca atrybut *Parametry wykrywania* definicji alarmu. W przypadku realizacji zewnętrznej strategii wykrywania alarmów przy pomocy mechanizmu skryptów, można wykorzystać właściwość *DetectionParameters* jako źródło informacji precyzującej sposób rozpoznawania stanu alarmu.

BeginText

```
string BeginText { get; }
```

ReadOnly Property **BeginText** As **String**

Właściwość zwraca tekst początku alarmu w oryginalnej formie użytej w definicji alarmu – wstawki formatujące {} nie są przekształcone.

EndText

```
string EndText { get; }
```

ReadOnly Property **EndText** As **String**

Właściwość zwraca tekst końca alarmu w oryginalnej formie użytej w definicji alarmu – wstawki formatujące {} nie są przekształcone.

Id

```
string Id { get; }
```

ReadOnly Property **Id** As **String**

Właściwość zwraca identyfikator alarmu, którego atrybuty są przechowywane w obiekcie.

IsStartOnly

```
bool IsStartOnly { get; }
```

ReadOnly Property **IsStartOnly** As **Boolean**

Właściwość zwraca informację, czy alarm należy do kategorii powiadomienie. Alarmy tej kategorii nie posiadają zdarzeń końca.

Priority

```
int Priority { get; }
```

ReadOnly Property **Priority** As **Integer**

Właściwość zwraca priorytet (ważność) alarmu. Możliwe są poniższe wartości:

- 0 – alarm krytyczny
- 1 – alarm ważny
- 2 – alarm zwykły
- 3 – ostrzeżenie
- 4 – komunikat

Strategy

```
string Strategy { get; }
```

ReadOnly Property **Strategy** As **String**

Właściwość zwraca nazwę strategii wykrywania alarmu.

3.13.2 Metody

GetAlarmAttribute

```
object GetAlarmAttribute(string aAttributeId);
```

```
Function GetAlarmAttribute (ByVal aAttributeId As  String) As Object
```

Funkcja zwraca definicję atrybutu, którego identyfikator określony jest parametrem *aAttributeId*. Metoda pozwala na pobranie wartości dowolnego atrybutu, także grupującego.

3.14 Klasa AlarmDomainStatistics

Obiekt *AlarmDomainStatistics* przechowuje informacje statystyczne dotyczące aktualnego stanu domeny alarmowej. Obiekty *AlarmDoamiStatistics* można uzyskać za pośrednictwem metody *GetAlarmDomainStatistics* interfejsu *IApplication.*

3.14.1 Właściwości

ActiveAlarmsCount

```
uint ActiveAlarmsCount{ get; }
```

```
ReadOnly Property ActiveAlarmsCount As UInteger
```

Właściwość zwraca aktualną liczbę alarmów aktywnych w domenie.

DefinedAlarmsCount

```
uint DefinedAlarmsCount{ get; }
```

```
ReadOnly Property DefinedAlarmsCount As UInteger
```

Właściwość zwraca liczbę alarmów zdefiniowanych w domenie.

ExcludedAlarmsCount

```
uint ExcludedAlarmsCount{ get; }
```

```
ReadOnly Property ExcludedAlarmsCount As UInteger
```

Właściwość zwraca aktualną liczbę alarmów wykluczonych w domenie.

FilteredAlarmsCount

```
uint FilteredAlarmsCount{ get; }
```

```
ReadOnly Property FilteredAlarmsCount As UInteger
```

Właściwość zwraca aktualną liczbę alarmów filtrowanych w domenie.

HistoricalAlarmsCount

```
long HistoricalAlarmsCount{ get; }
```

ReadOnly Property **HistoricalAlarmsCount** As **Long**

Właściwość zwraca aktualną liczbę alarmów w pliku logu alarmów historycznych domeny.

HistoricalLogFileSize

long HistoricalLogFileSize(get; }

ReadOnly Property **HistoricalLogFileSize** As **Long**

Właściwość zwraca aktualną rozmiar pliku logu alarmów historycznych domeny.

UnacceptedAlarmsCount

uint UnacceptedAlarmsCount{ get; }

ReadOnly Property **UnacceptedAlarmsCount** As **UInteger**

Właściwość zwraca aktualną liczbę alarmów niepotwierdzonych w domenie.

3.15 Klasa TrendPatternPoint

Obiekt *TrendPatternPoint* definiuje pojedynczy punkt krzywej wzorcowej. Tablicę punktów krzywej wzorcowej można pobrać przy pomocy metody *GetTrendPatternPoints* interfejsu *IApplication.*

3.15.1 Właściwości

TimeStamp

```
TimeSpan TimeStamp { get; }
```

ReadOnly Property **TimeStamp** As **System.TimeSpan**

Właściwość określa czas punktu krzywej wzorcowej – podawany jako odstęp czasu od początku krzywej.

Value

```
double Value { get; }
```

ReadOnly Property **Value** As **Double**

Właściwość określa wartość punktu krzywej wzorcowej.

3.16 Klasa AsalertAccess

Obiekt klasy *AsalertAccess* pozwala na wysłanie alertu(powiadomienia) za pośrednictwem serwera programu AsAlert. Dostęp do obiektu *AsalertAccess* uzyskuje się za pośrednictwem właściwości *Asalert* interfejsu *IApplication*.

3.16.1 Metody

Send

```
bool Send(string aTo, string aFrom, string aSubject, string aMessage);
```

```
Function Send (ByVal aTo As String, ByVal aFrom As String, ByVal Subject As String, ByVal aMessage As String) As Boolean
```

Metoda *Send* pozwala na wysłanie alertu za pośrednictwem programu AsAlert.

Parametr *aTo* określa nazwę adresata, do którego alert ma zostać wysłany. Adresat ten musi zostać wcześniej zdefiniowany w programie AsAlert.

Parametry *aFrom*, *aSubject* i aMessage określają treść alertu, kolejno: nazwę nadawcy, temat alertu i jego treść.

Metoda zwraca status poprawności wykonania operacji.

Przed wykonaniem metody *Send* należy użyć metodę *SetAccessData* w celu określenia parametrów komunikacji z programem AsAlert (o ile są różne od domyślnych).

SetAccessData

```
void SetAccessData(string aMachineName, string aUserName, string aPassword);
```

```
Sub SetAccessData(ByVal aMachineName As String, ByVal aUserName As String, ByVal aPassword As String)
```

Metoda służy do określenia parametrów komunikacyjnych z serwerem programu AsAlert. Parametr *aMachineName* określa systemową nazwę komputera, na którym program AsAlert został zainstalowany. Domyślna, pusta wartość oznacza komunikację z serwerem lokalnym, znajdującym się na tym samym komputerze co aplikacja.

Parametr *aUserName* pozwala na określenie nazwy użytkownika zdefiniowanego w programie AsAlert, który zostanie użyty w operacji wysłania alertu. Domyślna nazwa użytkownika to *AsixEvo*. Parametr *aPassword* służy do podania hasła użytkownika (domyślnie – brak hasła). W celu poprawnego działania, należy utworzyć odpowiednika użytkownika w programie AsAlert.

3.17 Klasa AsbaseAccess

Obiekt klasy *AsbaseAccess* pozwala na dostęp do danych zarchiwizowanych w bazie danych modułu AsBase. Dostęp do obiektu AsbaseAccess uzyskuje się za pośrednictwem właściwości Asbase interfejsu *IApplication*.

W celu poprawnego działania obiektu AsbaseAccess konieczne jest skonfigurowanie sposobu dostępu do bazy Asbase w zakładce *Źródła danych* panelu ustawień stanowisk aplikacji.

3.17.1 Metody

ExecuteQuery

```
System.Data.DataTable ExecuteQuery(string aQuery);
```

```
Function ExecuteQuery(ByVal aQuery As String) As System.Data.DataTable
```

Metoda pozwala na wykonanie dowolnego zapytania podanego w parametrze *aQuery* na bazie danych AsBase. Struktura bazy i nazewnictwo kolumn opisane jest w dokumentacji modułu Asbase.

Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy. Nazewnictwo kolumn jest opisane w dokumentacji modułu Asbase.

UWAGA

Chociaż, zasadniczo metoda przeznaczona jest do wykonywania zapytań typu *Select*, to możliwe jest również wykonanie komend zmieniających dane w bazie danych. Należy jednak zachować maksymalną ostrożność - nieprawidłowe użycie może doprowadzić do utraty całości zgromadzonych danych.

ReadArchive

```
System.Data.DataTable ReadArchive(string aArchiveId, string aDetailsDataSourceId, int aRecordCount, DateTime? aStartTimeUtc, DateTime? aEndTimeUtc, string aWherePhrase, string aOrderPhrase);
```

Function **ReadArchive**(ByVal aArchiveId As **String**, ByVal aDetailsDataSourceId As **String**, ByVal aRecordCount As **Integer**, ByVal aStartTimeUtc As **DateTime?**, ByVal aEndTimeUtc As **DateTime?**, ByVal aWherePhrase As **String**, ByVal aOrderPhrase As **String**) As **System.Data.DataTable**

Metoda służy od odczytu danych zgromadzonych w zestawie rejestracji o identyfikatorze podanym w parametrze *aArchiveId*. Opcjonalnie, możliwe jest dołączenie danych z połączonego relacją drugiego zestawu rejestracji lub grupy receptur, określonych identyfikatorem podanym w parametrze *aDetailsDataSourceId*. Relacja pomiędzy oboma źródłami danych musi być założona w programie Asbase.

Parametr *aRecordCount* pozwala ograniczyć liczbę zwróconych rekordów - wartość 0 oznacza brak ograniczenia.

Parametry *aStartTimeUtc* i *aEndTimeUtc* służą do określenia zakresu czasowego dla pobieranych rekordów. Czasy podawane są w standardzie UTC. Przekazanie wartości *null* oznacza brak ograniczenia czasowego.

Parametr*aWherePhrase* pozwala opcjonalnie dodać dodatkowe kryteria selekcji rekordów. Postać parametru musi być zgodna ze składnią klauzuli *where* języka SQL. Nazwy kolumn opisane są dokumentacji modułu Asbase.

Parametr *aOrderPhrase* pozwala opcjonalnie zmienić uporządkowanie rekordów. Standardowo rekordy są uporządkowane zgodnie z czasem rejestracji. Postać parametru musi być zgodna ze składnią klauzuli *order by* języka SQL.

Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy. Nazewnictwo kolumn jest opisane w dokumentacji modułu Asbase.

ReadLoads

System.Data.DataTable ReadLoads(string aRecipesId, string aDetailsDataSourceId, string aRecipeName, int aRecordCount, DateTime? aStartTimeUtc, DateTime? aEndTimeUtc, string aWherePhrase, string aOrderPhrase);

Function **ReadLoads**(ByVal aRecipeId As **String**, ByVal aDetailsDataSourceId As **String**, ByVal aRecipeName As **String**, ByVal aRecordCount As **Integer**, ByVal aStartTimeUtc As **DateTime?**, ByVal aEndTimeUtc As **DateTime?**, ByVal aWherePhrase As **String**, ByVal aOrderPhrase As **String**) As **System.Data.DataTable**

Metoda służy od odczytu danych zgromadzonych w tabeli rejestrującej zdarzenia załadowania receptur z grupy receptur określonej identyfikatorem podanym w parametrze *aRecipeId.* Opcjonalnie, możliwe jest dołączenie danych z połączonego relacją zestawu rejestracji określonego identyfikatorem podanym w parametrze *aDetailsDataSourceId*. Relacja pomiędzy oboma źródłami danych musi być założona w programie AsBase.

Parametr *aRecipeName* pozwala na ograniczenie odczytywanych rekordów tylko do tych, które dotyczą receptury o podanej nazwie. W parametrze można użyć znaków specjalnych w notacji SQL (% i _) w celu wybrania więcej niż jednej nazwy. Pusty parametr (lub %) oznaczają brak selekcji po nazwach receptur.

Parametr *aRecordCount* pozwala ograniczyć liczbę zwróconych rekordów - wartość 0 oznacza brak ograniczenia.

Parametry *aStartTimeUtc* i *aEndTimeUtc* służą do określenia zakresu czasowego dla pobieranych rekordów. Czasy podawane są w standardzie UTC. Przekazanie wartości *null* oznacza brak ograniczenia czasowego.

Parametr*aWherePhrase* pozwala opcjonalnie dodać dodatkowe kryteria selekcji rekordów. Postać parametru musi być zgodna ze składnią klauzuli *where* języka SQL. Nazwy kolumn opisane są dokumentacji modułu Asbase.

Parametr *aOrderPhrase* pozwala opcjonalnie zmienić uporządkowanie rekordów. Standardowo rekordy są uporządkowane zgodnie z czasem rejestracji. Postać parametru musi być zgodna ze składnią klauzuli *order by* języka SQL.

Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy. Nazewnictwo kolumn jest opisane w dokumentacji modułu Asbase.

ReadRecipes

```
System.Data.DataTable ReadRecipes(string aRecipesId, string aRecipeName, int
aRecordCount, string aWherePhrase, string aOrderPhrase);
```

```
Function ReadRecipes(ByVal aRecipeId As String, ByVal aRecipeName As String, ByVal
aRecordCount As Integer, ByVal aWherePhrase As String, ByVal aOrderPhrase As String) As
System.Data.DataTable
```

Metoda służy od odczytu danych receptur zgromadzonych w tabeli definicji receptur dla grupy receptur określonej identyfikatorem podanym w parametrze *aRecipeId.*

Parametr *aRecipeName* pozwala na ograniczenie odczytywanych rekordów tylko do tych, które dotyczą receptury o podanej nazwie. W parametrze można użyć znaków specjalnych w notacji SQL (% i _) w celu wybrania więcej niż jednej nazwy. Pusty parametr (lub %) oznaczają brak selekcji po nazwach receptur.

Parametr *aRecordCount* pozwala ograniczyć liczbę zwróconych rekordów - wartość 0 oznacza brak ograniczenia.

Parametr*aWherePhrase* pozwala opcjonalnie dodać dodatkowe kryteria selekcji rekordów. Postać parametru musi być zgodna ze składnią klauzuli *where* języka SQL. Nazwy kolumn opisane są dokumentacji modułu Asbase.

Parametr *aOrderPhrase* pozwala opcjonalnie zmienić uporządkowanie rekordów. Standardowo rekordy są uporządkowane zgodnie z nazwami receptur. Postać parametru musi być zgodna ze składnią klauzuli *order by* języka SQL.

Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy. Nazewnictwo kolumn jest opisane w dokumentacji modułu AsBase.

3.18 Klasa AsloggerAccess

Obiekt klasy *AsloggerAccess* pozwala na dostęp do danych zarchiwizowanych w bazie danych modułu AsLogger. Możliwy jest także zapis nowych serii. Dostęp do obiektu *AsloggerAccess* uzyskuje się za pośrednictwem właściwości *Aslogger* interfejsu *IApplication*.

Bezpośrednio po uzyskaniu obiektu *AsloggerAccess* należy wykonąc jego metodę *SetDataAccess* w celu określenie parametrów dostępu do bazy danych modułu AsLogger.

Klasa *AsloggerAccess* udostępnia szereg metod służących do odczytu zarejestrowanych serii pomiarowych. Wszystkie te metody zwracają informacje w obiektach standardowej klasy *DataTable*. Pozwala to m.in. na bezpośrednie użycie odczytanych danych w systemie generacji raportów skryptowych.

Klasa *AsloggerAccess* pozwala też na zapis nowych serii pomiarowych. Służy do tego metoda *CreateSeries* zwracająca obiekt klasy *AsloggerSeries*, za pośrednictwem którego seria danych jest zapisywana do bazy danych.

3.18.1 Metody

CreateSeries

```
AsloggerSeries CreateSeries(string aPlanName, string aDescription, DateTime
aSeriesTimeUtc, int aSamplingTime);
```

```
Function CreateSeries(ByVal aPlanName As String, ByVal aDescription As String, ByVal
aSeriesTimeUtc As DateTime, ByVal aSamplingTime As Integer) As AsloggerSeries
```

Metoda jest wykorzystywane w procesie zapisu nowej serii pomiarowej - tworzy obiekt klasy *AsloggerSeries* służacy następnie do konstrukcji zawartości serii i jej zapisu.

Parametr *aPlanName* określa nazwę planu, do którego seria będzie należała. Parametr *aDescription* służy do dodania do serii jej opisu.

Parametr *aSeriesTimeUtc* określa stempel czasu zapisywanej serii w standardzie UTC, a *aSamplingTime* to okres czasu pomiędzy kolejnymi wartościami serii wyrażony w mikrosekundach.

UWAGA

Dodanie serii do bazy danych wymaga wcześniejszego zdefiniowania, w bazie danych modułu AsLogger, planu rejestracji wraz ze wszystkimi punktami składowymi.

DeleteSeries

```
bool DeleteSeries(string aPlanName, int? aSeriesId);
```

```
Function DeleteSeries(ByVal aPlanName As String, ByVal aSeriesId As Integer?) As
Boolean
```

Metoda służy do usunięcia serii pomiarowej określonej nazwą planu (parametr *aPlanName*) i numerem identyfikacyjnym serii (parametr *aSeriesId*). Zestaw poprawnych identyfikatorów serii można uzyskać przy pomocy metody *ReadSeries*. Przekazenie wartości pustej w parametrze *aSeriesId* oznacza usunięcie ostatniej zarejestrownej serii.

ExecuteQuery

System.Data.DataTable ExecuteQuery(string aQuery);

Function **ExecuteQuery**(ByVal aQuery As **String**) As **System.Data.DataTable**

Metoda pozwala na wykonanie dowolnego zapytania podanego w parametrze *aQuery* na bazie danych AsLogger. Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy.

Użycie metody wymaga znajomości struktury bazy AsLogger. Metodę należy używac tylko wtedy, gdy żadna ze specjalizowanych metod klasy *AsLoggerAccess* nie jest wystarczająca.

UWAGA

Chociaż, zasadniczo metoda przeznaczona jest do wykonywania zapytań typu Select, to możliwe jest również wykonanie komend zmieniających dane w bazie danych. Należy jednak zachować maksymalną ostrożność - nieprawidłowe użycie może doprowadzić do utraty całości zgromadzonych danych.

ReadPlans

System.Data.DataTable ReadPlans(string aPlanName, string aWherePhrase);

Function **ReadPlans**(ByVal aPlanName As **String**, ByVal aWherePhrase As **String**) As **System.Data.DataTable**

Metoda służy do odczytu informacji dotyczących planów rejestracji zdefiniowanych w bazie danych mudułu AsLogger. Parametr *aPlanName* służy do określenia nazwy planu rejestracji. Przekazanie wartość pustej (*null* lub *Nothing)* powoduje zwrócenie danych dla wszystkich planów rejesteracji. W treści parametru można też użyć znaków specjalnych w konwencji SQL (% lub _) w celu wyboru podgrupy planów.

Jeżeli selekcja po nazwie planu jest niewystarczająca, to można użyć parametr *aWherePhrase*. Parametr powinien być zgodny ze składnią klauzuli *where* języka SQL. Użycie parametru wymaga znajomości struktury bazy AsLogger.

ReadPoints

System.Data.DataTable ReadPoints(string aPlanName, string aWherePhrase);

Function **ReadPoints**(ByVal aPlanName As **String**, ByVal aWherePhrase As **String**) As **System.Data.DataTable**

Metoda służy do odczytu informacji dotyczących punktów wchodzących w skład planu rejestracji o nazwie określonej parametrem *aPlanName*.

Standardowo zwracane są dane wszystkich punktów planu. Zestaw można ograniczyć przy pomocy parametru *aWherePhrase*. Parametr powinien być zgodny ze składnią klauzuli *where* języka SQL. Użycie parametru wymaga znajomości struktury bazy AsLogger. W szczególności wartość parametru *Name = ’P1'* , oznacza odpytanie o dane konfiguracyjne wyłącznie dla punktu o nazwie *P1.*

ReadSeries

System.Data.DataTable ReadSeries(string aPlanName, DateTime? aStartTimeUtc, DateTime? aEndTimeUtc, string aWherePhrase, bool aAddLocalTimeColumn);

Function **ReadSeries**(ByVal aPlanName As **String**, ByVal aStartTimeUtc As **DateTime?**, ByVal aEndTimeUtc As **DateTime?**, ByVal aWherePhrase As **String**, ByVal aAddLocalTimeColumn As **Boolean**) As **System.Data.DataTable**

Metoda służy do odczytu informacji dotyczących zarejestrowanych serii pomiarowych dla planu rejestracji o nazwie określonej parametrem *aPlanName*.

Parametry *aStartTimeUtc* i *aEndTimeUtc* służą do określenia zakresu czasowego dla pobieranych rekordów. Czasy podawane są w standardzie UTC. Przekazanie wartości pustej (*null* lub *Nothing*) oznacza brak ograniczenia czasowego.

Standardowo zwracane są dane wszystkich serii z okresu określonego parametrami *aStartTime* i *aEndTime*. Zestaw można ograniczyć przy pomocy parametru *aWherePhrase*. Parametr powinien być zgodny ze składnią klauzuli *where* języka SQL. Użycie parametru wymaga znajomości struktury bazy AsLogger.

Parametr *aAddLocalTimeColumn* pozwala na dodanie do wynikowego obiektu *DataTable* kolumny *BeginTimeLocal* przekazującej czasy początków serii jako czasy lokalne. Standardowo, dostępny jest

tylko czas UTC w kolumnie *BeginTimeUtc*. Dodanie czasu lokalnego może być szczególnie przydatne w przypadku tworzenia raportów za pośrednictwem klasy *Report*.

W zestawie kolumn, w zwróconym przez metodę obiekcie *DataTable*, szczególnie istotna jest kolumna *IdSerie*. Przechowuje ona identyfikatory serii, które mogą być później wykorzystane do odczytu danych serii metodą *ReadSeriesData* lub usunięcia serii metodą *DeleteSeries*.

ReadSeriesAggregates

```
System.Data.DataTable ReadSeriesAggregates(string aPlanName, SeriesAggregateId
aAggregate, DateTime? aStartTimeUtc, DateTime? aEndTimeUtc, string aWherePhrase, bool
aAddLocalTimeColumn, params string[] aPointsNames);
```

```
Function ReadSeriesAggregates(ByVal aPlanName As String, ByVal aAggregate
As SeriesAggregateId, ByVal aStartTimeUtc As DateTime?, ByVal aEndTimeUtc
As DateTime?, ByVal aWherePhrase As String, ByVal aAddLocalTimeColumn As Boolean,
ByVal ParamArray aPointsNames As String()) As System.Data.DataTable
```

Metoda służy do odczytu agregatów zarejestrowanych serii pomiarowych dla planu rejestracji o nazwie określonej parametrem *aPlanName*.

Parametr *aAggregate* określa typ agregatu:

SeriesAggregateId.SeriesMinimum - minimalna wartość w serii

SeriesAggregateId.SeriesMaximum - maksymalna wartość w serii

SeriesAggregateId.SeriesAverage - średnia wartość serii

Parametry *aStartTimeUtc* i *aEndTimeUtc* służą do określenia zakresu czasowego serii pomiarowych, dla których agregaty są pobierane. Czasy podawane są w standardzie UTC. Przekazanie wartości pustej (*null* lub *Nothing*) oznacza brak ograniczenia czasowego.

Standardowo zwracane są agregaty wszystkich serii z okresu określonego parametrami *aStartTime* i *aEndTime*. Zestaw serii można ograniczyć przy pomocy parametru *aWherePhrase*. Parametr powinien być zgodny ze składnią klauzuli *where* języka SQL. Użycie parametru wymaga znajomości struktury bazy AsLogger.

Parametr *aAddLocalTimeColumn* pozwala na dodanie do wynikowego obiektu *DataTable* kolumny *SampleTimeLocal* przekazującej czasy agregatów jako czasy lokalne. Standardowo, dostępny jest tylko czas UTC w kolumnie *SampleTimeUtc*. Dodanie czasu lokalnego może być szczególnie przydatne w przypadku tworzenia raportów za pośrednictwem klasy *Report*.

Parametr *aPointsNames* służy do przekazania nazw punktów serii, dla których agregaty mają zostać pobrane. Jeżeli parametr nie zostanie użyty, to pobrane będą agregaty wszystkich punktów.

ReadSeriesData

System.Data.DataTable ReadSeriesData(string aPlanName, AggregateId aAggregate, int? aSeriesId, string aWherePhrase, bool aAddLocalTimeColumn, params string[] aPointsNames);

Function **ReadSeriesData**(ByVal aPlanName As **String**, ByVal aAggregate As **AggregateId**, ByVal aSeriesId As **Integer?**, ByVal aWherePhrase As **String**, ByVal aAddLocalTimeColumn As **Boolean**, ByVal ParamArray aPointsNames As **String()**) As **System.Data.DataTable**

Metoda służy do odczytu zarejestrowanych wartości w serii pomiarowej określonej nazwą planu (parametr *aPlanName*) i numerem identyfikacyjnym serii (parametr *aSeriesId*). Zestaw poprawnych identyfikatorów serii można uzyskać przy pomocy metody *ReadSeries*. Przekazenie wartości pustej w parametrze *aSeriesId* oznacza odczyt wartości dla ostatniej zarejestrownej serii.

Parametr *aAggregate* określa typ odczytywanych wartości punktów:

AggregateId.RawData - dane surowe

AggregateId.Minimum - okresowa minimalna wartość (liczona z każdych kolejnych 100 wartości surowych)

AggregateId.Maximum - okresowa maksymalna wartość (liczona z każdych kolejnych 100 wartości surowych)

AggregateId.Average - okresowa średnia wartość (liczona z każdych kolejnych 100 wartości surowych)

AggregateId.SeriesMinimum - minimalna wartość w serii

AggregateId.SeriesMaximum - maksymalna wartość w serii

AggregateId.SeriesAverage - średnia wartość serii

Parametr *aWherePhrase* pozwala dodatkowo ograniczyć zestaw zwracanych wartości. Parametr powinien być zgodny ze składnią klauzuli *where* języka SQL. Użycie parametru wymaga znajomości struktury bazy AsLogger.

Parametr *aAddLocalTimeColumn* pozwala na dodanie do wynikowego obiektu *DataTable* kolumny *SampleTimeLocal* przekazującej czasy próbek jako czasy lokalne. Standardowo, dostępny jest tylko czas UTC w kolumnie *SampleTimeUtc*. Dodanie czasu lokalnego może być szczególnie przydatne w przypadku tworzenia raportów za pośrednictwem klasy *Report*.

Parametr *aPointsNames* służy do przekazania nazw punktów serii, dla których wartości mają zostać pobrane. Jeżeli parametr nie zostanie użyty, to pobrane będą wartości dla wszystkich punktów.

SetAccessData

```
void SetAccessData(string aServerName, string aDatabaseName);
```

```
Sub SetAccessData (ByVal aServerName As String, ByVal aDatabaseName As String)
```

Metoda *SetAccessData* powinna być zawsze wywoływana jako pierwsza po uzyskaniu dostępu do obiektu *AsloggerAccess*. Służy ona do określenia lokalizacji bazy danych modułu AsLogger. Parametr *aServerName* określa nazwę serwera SQL, a parametr *aDatabaseName* nazwę bazy danych.

3.19 Klasa AsloggerSeries

Obiekt klasy *AsloggerSeries* służy do tworzenia nowych serii pomiarowych. Obiekty tej klasy tworzone są za pomocą metody *CreateSeries* klasy *AsloggerAccess*.

W typowym schemacie postępowania, po utworzeniu obiektu serii, dodawane są do niego wartości serii dla poszczególnych punktów planu rejestracji przy pomocy metody *AddPointValues*. Po przekazaniu wszystkich wartości dla wszystkich punktów należy wykonać metodę *Save*, która zapisuje kompletną serię pomiarową w bazie danych.

3.19.1 Właściwości

SeriesLength

```
int SeriesLength{ get; }
```

```
ReadOnly Property SeriesLength As Integer
```

Właściwość zwraca aktualną liczbę rekordów wartości punktów, dodanych do serii przy pomocy metod *AddPointValues* i *AddPointsRecord*.

WriteCount

```
int WriteCount{ get; }
```

```
ReadOnly Property WriteCount As Integer
```

Właściwość zwraca aktualną liczbę zapisanych rekordów w trakcie wykonywania metody *Save.* Właściwość pozwala na monitorowanie postępu prac w trakcie zapisu długich serii pomiarowych.

3.19.2 Metody

AddPointsRecord

```
bool AddPointsRecord(params System.Collections.ICollection[] aValues);
```

```
Function AddPointsRecord(ByVal ParamArray aValues As System.Collections.ICollection()) As Boolean
```

Metoda służy dodawania do serii wartości punktów w trybie rekordów - czyli, w pojedynczym wywołaniu dodawana jest jedna wartość dla każdego z punktów serii. Przed wywołaniami metody *AddPointsRecord* należy zdefiniować punkty, dla których wartości będą dodawane. W tym celu należy odpowiednią ilość razy wykonać metodę *AddPointValues* z pustym parametrem *aValues*.

Parametr *aValues* metody*AddPointsRecord* służy do przekazania rekordu wartości punktów. Można przekazać dowolną kolekcję (w najprostszym przypadku tablicę) obiektów, które można skonwertować na wartość typu *double*. Kolejne elementy kolekcji określają wartości dla punktów serii zgodnie z kolejnością ich dodania wywołaniami *AddPointValues*.

AddPointValues

```
bool AddPointValues(string aPointName, System.Collections.ICollection aValues = null);
```

Function **AddPointValues**(ByVal aPointName As **String**, Optional ByVal aValues As **System.Collections.ICollection** = Nothing) As **Boolean**

Podstawowa metoda służącą do dodania do serii pomiarowej punktów wraz z danymi serii.

Parametr *aPointName* określa nazwę punktu - punkt o tej nazwie musi być zdefiniowany w planie rejestracji, do którego seria będzie należeć. Parametr *aValues* służy do przekazania wartości serii dla punktu. Można przekazać dowolną kolekcję (w najprostszym przypadku tablicę) obiektów, które można skonwertować na wartość typu *double*. Kolejne elementy kolekcji określają kolejne wartości punktu dla serii pomiarowej.

W procesie dodawania wartości punktów obowiązują poniższe zasady:

a) Można wielokrotnie stosować metodę dla tej samej nazwy punktu - nowe wartości będą dopisywane do danych wcześniej dodanych.
b) Jeżeli dodawane są dane do więcej niż jednego punktu, to końcowa liczba wartości dla wszystkich punktów musi być identyczna.
c) Nie jest wymagane dodanie wartości do wszystkich punktów zdefiniowanych w planie rejestracji. Punkty bez dodanych wartości będą mieć wszystkie dane serii wypełnione wartością NULL.

Save

```
bool Save();
```

Function **Save**() As **Boolean**

Metoda służy do zapisu zawartości serii pomiarowej w bazie danych. Przed wykonaniem metody należy do serii dodać jej zawartość przy pomocy metod *AddPointValues* i *AddPointsRecord*.

3.20 Klasa AsserviceAccess

Obiekt klasy *AsserviceAccess* pozwala na dostęp do danych zarchiwizowanych w bazie danych modułu AsService. Dostęp do obiektu *AsserviceAccess* uzyskuje się za pośrednictwem właściwości *Asservice* interfejsu *IApplication*.

W celu poprawnego działania obiektu *AsserviceAccess* konieczne jest skonfigurowanie sposobu dostępu do bazy AsService w zakładce *Źródła danych* panelu ustawień stanowisk aplikacji.

Klasa udostępnia dwie metody do odczytu stanu i definicji liczników: *ReadArcCounters* i *ReadCounters*. Zestaw zwracanych informacji jest podobny. Występuję pomiędzy nimi jednak poniższe różnice:

- Funkcja *ReadCounters* zwraca pojedynczy rekord dla każdego licznika. Dane resetu dotyczą ostatniego zdarzenia resetu lub są wypełnione pustymi wartościami.
- Funkcja ReadArcCounters zwraca rekordy tylko dla tych liczników, dla których zarejestrowano zdarzenie resetu. Liczba rekordów dla każdego licznika wynika z liczby zarejestrowanych zdarzeń resetu.

3.20.1 Metody

ReadArcCounters

```
System.Data.DataTable ReadArcCounters(string aCounterId, string[] aFieldNames,
DateTime? aStartTime, DateTime? aEndTime, string aWherePhrase, string aOrderPhrase);
```

```
Function ReadArcCounters(ByVal aCounterId As String, ByVal aFieldNames As String(),
ByVal aStartTime As DateTime?, ByVal aEndTime As DateTime?, ByVal aWherePhrase As
String, ByVal aOrderPhrase As String) As System.Data.DataTable
```

Metoda służy od odczytu historii resetów liczników o identyfilkatorach określonych w parametrze *CounterId.* W treści parametru należy użyć znaków specjalnych w notacji SQL (% i _) w celu wybrania więcej niż jednego licznika. Pusty parametr (lub %) oznacza odczyt danych dla wszystkich liczników.

Tablica nazw pól liczników przekazywana w parametrze *aFieldNames* służy do określenia jakie dane dotyczące liczników mają być odczytane z bazy danych. Najłatwiej posłużyć się nazwami pól zdefiniowanymi w statycznej klasie *AsServiceAcces.Names*.

Parametry *aStartTime* i *aEndTime* służą do określenia zakresu czasowego resetów liczników dla pobieranych rekordów. Przekazanie wartości *null* oznacza brak ograniczenia czasowego.

Parametr*aWherePhrase* pozwala opcjonalnie dodać dodatkowe kryteria selekcji rekordów. Postać parametru musi być zgodna ze składnią klauzuli *where* języka SQL.

Parametr *aOrderPhrase* pozwala opcjonalnie zmienić uporządkowanie rekordów. Standardowo rekordy są uporządkowane zgodnie z identyfikatorami liczników i czasem resetu. Postać parametru musi być zgodna ze składnią klauzuli *order by* języka SQL.

Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy.

ReadCounters

System.Data.DataTable ReadCounters(string aCounterId, string[] aFieldNames, LimitCheckMode aLimitMode, string aWherePhrase, string aOrderPhrase);

Function **ReadCounters**(ByVal aCounterId As **String**, ByVal aFieldNames As **String()**, ByVal aLimitMode As **LimitCheckMode**, ByVal aWherePhrase As **String**, ByVal aOrderPhrase As **String**) As **System.Data.DataTable**

Metoda służy od odczytu danych opisujących definicję, aktualny stan i ostatni reset liczników o identyfilkatorach określonych w parametrze *CounterId.* W treści parametru należy użyć znaków specjalnych w notacji SQL (% i _) w celu wybrania więcej niż jednego licznika. Pusty parametr (lub %) oznacza odczyt danych dla wszystkich liczników.

Tablica nazw pól liczników przekazywana w parametrze *aFieldNames* służy do określenia jakie dane dotyczące liczników mają być odczytane z bazy danych. Najłatwiej posłużyć się nazwami pól zdefiniowanymi w statycznej klasie *AsServiceAcces.Names*.

Parametr *aLimitMode* pozwala ograniczyć zestaw liczników, do tych które przekraczają aktualnie wybrany limit alarmowy.

Parametr*aWherePhrase* pozwala opcjonalnie dodać dodatkowe kryteria selekcji rekordów. Postać parametru musi być zgodna ze składnią klauzuli *where* języka SQL.

Parametr *aOrderPhrase* pozwala opcjonalnie zmienić uporządkowanie rekordów. Standardowo rekordy są uporządkowane zgodnie z identyfikatorami liczników. Postać parametru musi być zgodna ze składnią klauzuli *order by* języka SQL.

Metoda zwraca obiekt klasy *DataTable* zawierający odczytane rekordy.

Przykład:

C#

```
DataTable dt = aApplication.Asservice.ReadCounters("U%", new string[] {
            AsserviceAccess.Names.Identifier,
            AsserviceAccess.Names.Name,
            AsserviceAccess.Names.Value,
            AsserviceAccess.Names.OperatingLevel},
      AsserviceAccess.LimitCheckMode.Operating, null, null);
```

VB

```
Dim names As String() = { _
             AsserviceAccess.Names.Identifier, _
             AsserviceAccess.Names.Name, _
             AsserviceAccess.Names.Value, _
             AsserviceAccess.Names.OperatingLevel }

Dim dt As DataTable = aApplication.Asservice.ReadCounters("U%", names, _
                      AsserviceAccess.LimitCheckMode.Operating, Nothing, Nothing)
```

Powyższe wywołanie odczytuje dane dla wszystkich liczników o identyfikarze zaczynającym się od litery *U* i z przekroczonym limitem operacyjnym. Odczytywane pola to identyfikator i nazwa licznika, bieżąca wartość licznika i limit operacyjny.

3.21 Klasa AsserviceAccess.Names

Klasa *AsServiceAccessNames* jest statyczną klasą, która udostępnia zestaw stałych tekstowych określających nazwy wszystkich pól liczników, które można pobrać metodami *ReadCounters* i *ReadArcCounters*.

Tab. Zestaw pól liczników.

Pola związane z definicją i aktualnym stanem licznika	
Identifier	Identyfikator licznika
Name	Tekstowy opis licznika
CounterType	Typ licznika
CounterTypeDesc	Tekstowy opis typu licznika
VarName	Nazwa zmiennej procesowej na podstawie której przeliczana jest wartość licznika
AggrName	Nazwa agregatu używanego do przeliczeń
MainCounter	Flaga True/False określająca, czy licznik jest licznikiem głównym urządzenia
RefreshFreq	Okres przeliczania wartości licznika

StartValue	Początkowa wartość licznika
Value	Aktualna wartość licznika
StartTime	Czas początku zliczania
NoValueTime	Okres czasu braku odczytu poprawnych wartości zmiennej źródłowej
LastReadTime	Czas ostatniego przeliczenia licznika
NextReadTime	Czas planowanego następnego przeliczenia licznika
OperatingLevel	Wartość limitu eksploatacyjnego
CyclicOperatingLevel	Flaga True/False określająca, czy wartość limitu eksploatacyjnego jest odtwarzana po każdym resecie
WarningLevel	Wartość limitu ostrzegawczego
WarningLevel2	Wartość drugiego limitu ostrzegawczego
AutoResetReason	Przyczyna automatycznego resetu licznika
AutoResetReasonDesc	Tekstowy opis przyczyny automatycznego resetu licznika
OperatingTask	Czynność eksploatacyjna związana z przekroczeniem limitu licznika
OperatingTaskDesc	Tekstowy opis czynności eksploatacyjnej
Pola związane z definicją urządzenia, do którego licznik należy	
DeviceId	Identyfikator urządzenia
DeviceName	Nazwa urządzenia
DeviceDesc	Tekstowy opis urządzenia
DeviceSymbol	Alfanumeryczny symbol urządzenia
DeviceRegistryNum	Numer ewidencyjny urządzenia
DeviceFactoryNum	Numer fabryczny urządzenia
DeviceProdYear	Rok produkcji urządzenia
DevicePurchaseDate	Data zakupu urządzenia
DevicePowerInst	

DeviceDimensions	Rozmiary urządzenia
DeviceMass	Masa całkowita urządzenia
DeviceVirtual	Flaga True/False określająca, czy urządzenie jest wirtualne
DeviceMachine	Nazwa maszyny, do której urządzenie należy
DeviceMachineDesc	Opis tekstowy maszyny
DeviceDepartment	Nazwa wydziału, do którego urządzenie należy
DeviceDepartmentDesc	Opis tekstowy wydziału
DeviceProducer	Producent urządzenia
DeviceProducerDesc	Opis tekstowy producenta
DeviceSupplier	Dostawca urządzenia
DeviceSupplierDesc	Opis tekstowy dostawcy
DeviceLine	Nazwa linii, do której urządzenie należy
DeviceLineDesc	Opis tekstowy linii
DeviceGroup	Nazwa grupy, do której urządzenie należy
DeviceGroupDesc	Opis tekstowy grupy
DevicePowerInst	Moc zasilania urządzania
DevicePowerType	Typ zasilania urządzenia
DevicePowerTypeDesc	Tekstowy opis typu zasilania
DeviceStatus	Status urządzenia
DeviceStatusDesc	Tekstowy opis statusu
DeviceTechCond	Stan techniczny urządzenia
DeviceTechCondDesc	Tekstowy opis stanu technicznego
Pola opisujące ostatni reset licznika	
ArcEndValue	Wartość licznika w momencie resetu
ArcStartValue	Początkowa wartość licznika
ArcStartTime	Moment uruchomienia licznika

ArcEndTime	Moment resetu licznika
ArcNoValueTime	Okres czasu, przez który brakowało odczytu poprawnych danych źródłowych
ArcProtocolNo	Numer protokołu związanego z resetem licznika
ArcExecutedBy	Osoba wykonująca reset
ArcResetReason	Przyczyna resetu licznika
ArcResetReasonDesc	Opis tekstowy przyczyny resetu

3.22 Interfejs IPowerGuardMeter

3.22.1 Właściwości

Name

```
string Name { get; }
```

ReadOnly Property **Name** As **String**

Właściwość zwraca nazwę miernika strażnika mocy kontrolowanego przez posiadaną referencję do interfejsu IPowerGuardMeter.

MeterState

```
PowerGuardMeterState MeterState { get; }
```
ReadOnly Property **MeterState** As **PowerGuardMeterState**

Właściwosc złożona opisująca bieżący stan miernika strażnika mocy.

3.22.2 Metody

RegisterStateEvent

```
void RegisterStateEvent(PowerGuardMeterEvent aFunction);
```

```
Sub RegisterStateEvent(ByVal aFunction As  PowerGuardMeterEvent)
```

Funkcja służy do zarejestrowania funkcji zwrotnej, która będzie wywoływana przy każdej zmianie stanu kontrolowanego miernika strażnika mocy. Parametr *aFunction* określa funkcję monitorującą stan miernika – musi ona być zgodna z delegatem *PowerGuardMeterEvent.*

```
delegate void PowerGuardMeterEvent (string aMeterName, PowerGuardMeterState aMeterState);
```

```
Delegate Sub PowerGuardMeterEvent (ByVal aMeterName As  String, ByVal aMeterState As  PowerGuardMeterState)
```

W momencie wywołania funkcji monitorującej przekazywane są do niej dwa parametry. Parametr *aMeterName* określa nazwę miernika. Możliwe jest podłączenie tej samej funkcji monitorującej do wielu mierników, a poprzez parametr *aMeterName* można rozróżnić, którego miernika wywołanie dotyczy. Parametr *aMeterState* przekazuje obiekt, z którego można odczytać aktualny stan miernika.

UnregisterStateEvent

```
void UnregisterStateEvent();
```

```
Sub UnregisterStateEvent()
```

Wyrejestrowanie funkcji monitorującej wartości miernika, która została wcześniej zarejestrowana przy pomocy funkcji *RegisterStateEvent*.

3.23 Klasa PowerGuardMeterState

3.23.1 Właściwości

CurrentCycleStartTime

```
DateTime CurrentCycleStartTime { get; }
```

ReadOnly Property **CurrentCycleStartTime** As `DateTime`

Właściwość zwraca czas początku bieżącego cyklu miernika strażnika mocy.

CurrentCycleEndTime

```
DateTime CurrentCycleEndTime { get; }
```

ReadOnly Property **CurrentCycleEndTime** As `DateTime`

Właściwość zwraca czas końca bieżącego cyklu miernika strażnika mocy.

CurrentCycleLimit

```
double CurrentCycleLimit { get; }
```

ReadOnly Property **CurrentCycleLimit** As `Double`

Właściwość zwraca limit obowiązujący dla bieżącego cyklu miernika strażnika mocy.

CurrentCycleAverageValue

```
double CurrentCycleAverageValue { get; }
```

ReadOnly Property **CurrentCycleAverageValue** As `Double`

Właściwość zwraca estymowaną wartość średnią dla bieżącego cyklu miernika strażnika mocy.

CurrentCycleSafeRemainingValue

```
double CurrentCycle SafeRemainingValue { get; }
```

ReadOnly Property **CurrentCycle SafeRemainingValue** As `Double`

Właściwość zwraca wartość bezpieczną dla pozostałych interwałów dla bieżącego cyklu miernika strażnika mocy.

UnitText

```
string CurrentCycle UnitText { get; }
```

ReadOnly Property **UnitText SafeRemainingValue** As `String`

Właściwość zwraca jednostke pomiarową miernika strażnika mocy.

IsPredictionSafe

```
bool CurrentCycle IsPredictionSafe { get; }
```

ReadOnly Property **CurrentCycle IsPredictionSafe** As `Bool`

Właściwość informuje czy przewidywana wartość średnia na koniec bieżącego cyklu nie przekracza limitu miernika strażnika mocy.

Załącznik 1. Rodzaje agregatów danych archiwalnych

W każdym przypadku, w którym w aplikacji wymagany jest dostęp do zagregowanych danych historycznych konieczne jest podanie nazwy agregatu. Dotyczy to obiektu *Wykres,* funkcji *HorizonAggregate* i *RangeAggregate*, interfejsów modelu skryptowego.

Nazwa	Znaczenie
none	brak agregacji
start	Wartość na początku interwału.
end	Wartość na końcu interwału.
delta	Różnica wartości na końcu i na początku interwału.
min	Minimalna wartość w interwale.

max	Maksymalna wartość w interwale.
range	Różnica między maksymalna a minimalna wartością w interwale.
total	Suma wartości ważonych czasowo w interwale (całka po czasie).
average	Średnia wartości ważonych czasowo w interwale.
average0	Średnia wartości ważonych czasowo w interwale. W okresach, w których wartość jest niedostępna do obliczeń, używana jest wartość 0.
sumup	Suma wartości dodatnich zmian, od ostatniej znanej wartości przed rozpoczęciem, do ostatniej wartości przed końcem przedziału próbkowania.
sumdown	Suma wartości bezwzględnych ujemnych zmian, od ostatniej znanej wartości przed rozpoczęciem, do ostatniej wartości przed końcem przedziału próbkowania.
prevknown	Ostatnio znana (dobra lub niepewna) wartość przed rozpoczęciem przedziału próbkowania. Podaje ostatnią wartość, nawet jeżeli wystąpiła ona bardzo dawno.
last	Ostatnio znana (dobra lub niepewna) wartość w przedziale próbkowania wyłączając koniec przedziału - dla odróżnienia od agregatu *end*, który zawiera koniec przedziału.
stdev	Ważone czasowo odchylenie standardowe w przedziale próbkowania.
rms	Ważona czasowo średnia kwadratowa w przedziale próbkowania.
avglk	Średnia ważona czasowo w przedziale próbkowania. W miejsce złych lub brakujących wartości wstawia ostatnią znaną wartość.
totallk	Całka w przedziale próbkowania, przy czasie liczonym w sekundach. W miejsce złych lub brakujących wartości wstawia ostatnią znaną wartość.

Szczegółowy opis agregatów oraz ich parametryzacji znajduje się w części Serwer Danych > Obsługa danych procesowych Serwera Danych > 1.3. Aspad - program archiwizacji danych.

Asix.Evo - Uruchomienie aplikacji WWW

1. Wymagania

Asix.Evo uruchamiany w przeglądarce wymaga do działania systemów operacyjnych Windows (XP SP2 lub nowszych) lub systemów serwerowych (Windows Server 2003 lub nowszych). Dodatkowo wymagana jest instalacja platformy **Microsoft .NET 4 Framework 4.0** oraz przeglądarki Internet Explorer w wersji 6 lub nowszej.

2. Pierwsze uruchomienie

Na komputerze, na którym aplikacja Asix.Evo w wersji przeglądarkowej jest uruchamiana pierwszy raz, z danego serwera należy odpowiednio skonfigurować przeglądarkę oraz platformę .Net. Konfigurację tę można przeprowadzić automatycznie za pomocą programu konfiguracyjnego BrowserConfigurator.exe. Program ten jest dostępny na serwerze WWW, na którym została opublikowana aplikacja Asix.Evo dla przeglądarek.

a. Uruchamiamy stronę główną Asixa.Evo z danego serwera, wpisując w przeglądarce Internet Explorer adres **http://<ADRES_SERWERA>/EvoNet, gdzie <ADRES_SERWERA>** jest fizycznym adresem IP lub nazwą serwera, na którym opublikowano aplikację.

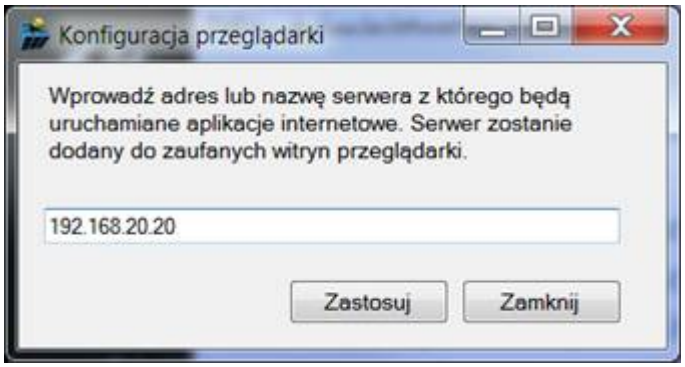


Rys. Okno strony głównej serwera aplikacji przeglądarkowych systemu Asix.Evo.

b. Pobieramy i zapisujemy na dysku (np. na pulpicie) ze strony głównej Asix.Evo program ***BrowserConfigurator.exe***, wybierając na jego hiperłączu z menu kontekstowego opcję *Zapisz element docelowy jako…*;.

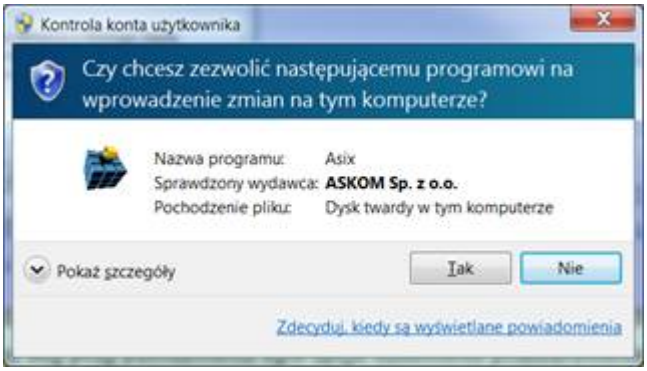
c. Uruchamiamy program ***BrowserConfigurator.exe*** z lokalizacji, w której został zapisany. Nie należy uruchamiać tego programu bezpośrednio z przeglądarki - nawet jeśli taka opcja jest dostępna. Do przeprowadzenia pełnej i poprawnej konfiguracji program wymaga uprawnień administracyjnych do lokalnego komputera.

d. W polu edycyjnym programu należy wprowadzić adres lub nazwę serwera, na którym jest opublikowana aplikacja Asix.Evo, a następnie wcisnąć przycisk *Zastosuj*.



Rys. Okno konfiguracji przeglądarki.

e. Gdy program wyświetli systemowe okienko kontroli konta użytkownika, to należy zezwolić na dokonanie zmian konfiguracji systemu, wciskając przycisk Tak. W przypadku, gdy bieżący użytkownik Windows nie posiada uprawnień administracyjnych, należy w tym okienku podać login i hasło użytkownika o uprawnieniach administratora.



Rys. Okno kontroli konta użytkownika.

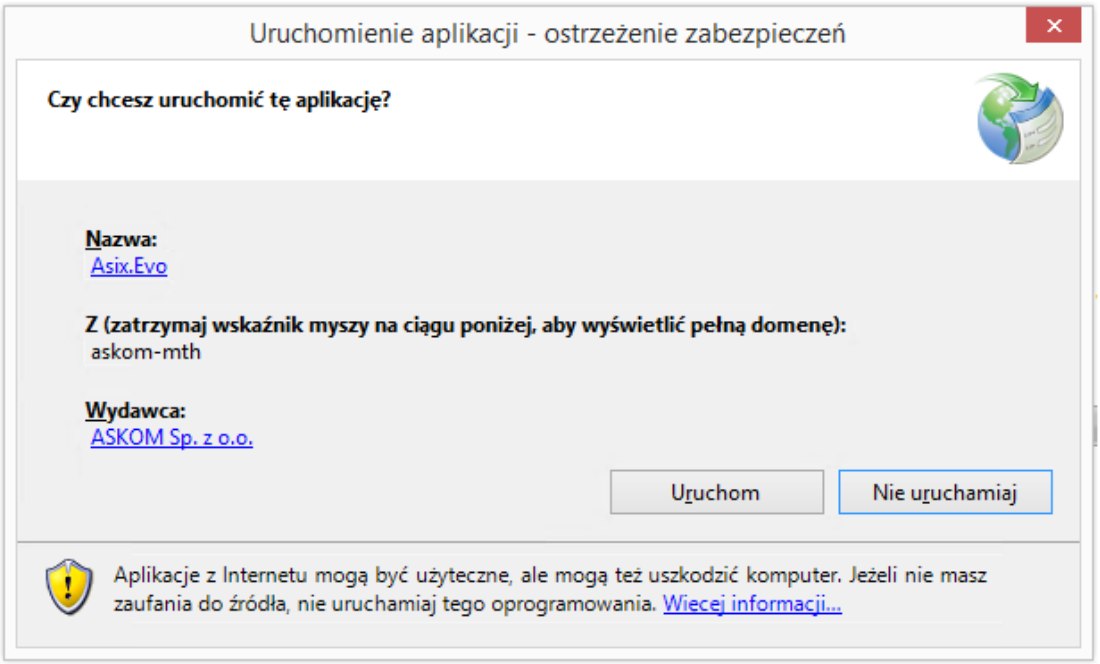
f. Gdy program dokona konfiguracji i wyświetli na końcu okienko informujące o pomyślnym przebiegu, to należy go zamknąć oraz należy też zamknąć okno przeglądarki IE. W przypadku, gdy było otwartych więcej okien przeglądarki IE lub jedno okno z wieloma zakładkami, to należy zamknąć je wszystkie. W przypadku gdy

program konfiguracyjny zgłosi błąd lub poinformuje, że nie może dokonać konfiguracji, należy skontaktować się z administratorem systemu lub serwera WWW.

g. Uruchamiamy przeglądarkę Internet Explorer, wpisujemy adres jak w pkt. a), a następnie uruchamiamy aplikację Asix.Evo, klikając w jedno z hiperłączy opublikowanych aplikacji. Każda aplikacja może być uruchomiona w trybie ***Lite*** albo ***Pro***. Tryb *Lite* umożliwia uruchomienie aplikacji w trybie tylko do odczytu, bez możliwości wykonywania sterowań zmiennych procesowych, potwierdzeń i zmiany wykluczenia alarmów oraz innych operacji wpływających na działanie systemu. Tryb Pro pozwala na wszystkie te czynności. Dostępność tych trybów zależy od liczby wykupionych licencji WWW typu *Pro* i *Lite*.

h. Pierwsze uruchomienie aplikacji na danym komputerze może trwać nawet do kilku minut, gdyż przeglądarka musi w tym czasie pobrać z serwera aktualną wersję oprogramowania oraz pliki definicji opublikowanej aplikacji.

i. Za pierwszym uruchomieniem aplikacji w przeglądarce, system Windows może wymagać zaakceptowania certyfikatu oprogramowania firmy Askom. Należy wcisnąć przycisk *Uruchom*.

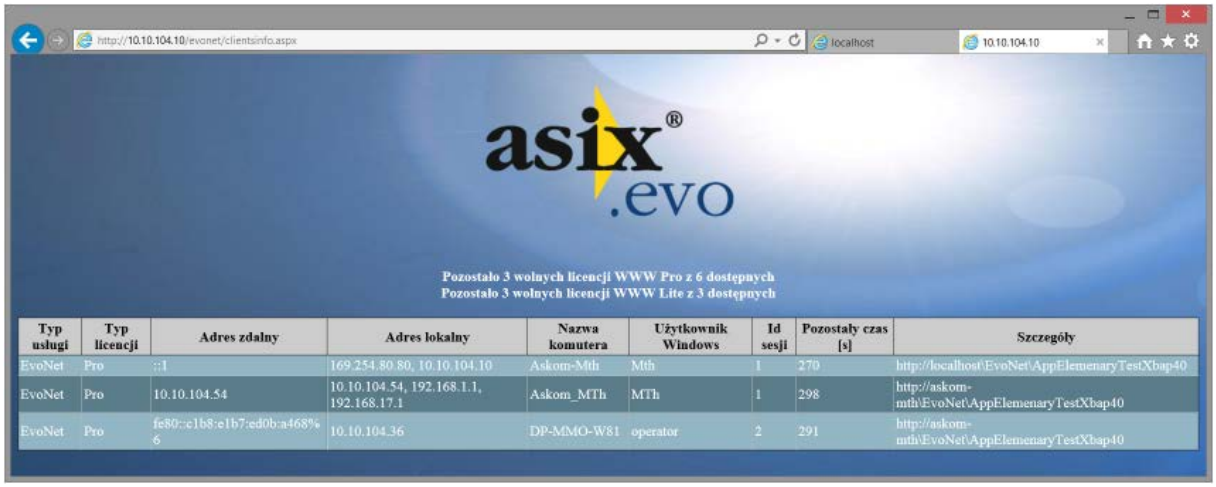


Rys. Okno akceptowania certyfikatu oprogramowania firmy Askom.

j. Jeżeli aplikacja nie uruchomi się lub zgłosi błąd podczas uruchamiania, to należy skontaktować się z administratorem systemu lub serwera WWW.

3 Sprawdzanie używanych i dostępnych licencji WWW

Wpisując w przeglądarce Internet Explorer adres **http://<ADRES_SERWERA>/EvoNet/clientsinfo.aspx** uzyskujemy dostęp do strony wyświetlającej informację o liczbie dostępnych i wolnych licencji WWW (typu Lite i Pro) oraz szczegółowe dane o aktualnie korzystających z tych licencji komputerach.



Typ usługi	Typ licencji	Adres zdalny	Adres lokalny	Nazwa komutera	Użytkownik Windows	Id sesji	Pozostały czas [s]	Szczegóły
EvoNet	Pro	::1	169.254.80.80, 10.10.104.10	Askom-Mth	Mth	1	270	http://localhost\EvoNet\AppElemenaryTestXbap40
EvoNet	Pro	10.10.104.54	10.10.104.54, 192.168.1.1, 192.168.17.1	Askom_MTh	MTh	1	298	http://askom-mth\EvoNet\AppElemenaryTestXbap40
EvoNet	Pro	fe80::e1b8:e1b7:ed0b:a468%6	10.10.104.36	DP-MMO-W81	operator	2	291	http://askom-mth\EvoNet\AppElemenaryTestXbap40

Rys. Okno z informacją o licencjach WWW.

Asix.Evo - Jak to zrobić

1 Uruchamianie programu i zmiana trybów pracy

1.1 Wystartowanie programu w trybie architekta

Uruchom program *AsixEvo.exe* bez parametrów. W otwartym oknie architekta aplikacji otwórz właściwą aplikację lub utwórz nową przy pomocy komend z menu *Plik*.

Ewentualnie podaj nazwę katalogu definicyjnego aplikacji w komendzie startowej programu *AsixEvo*.

1.2 Uruchomienie aplikacji w trybie wykonawczym

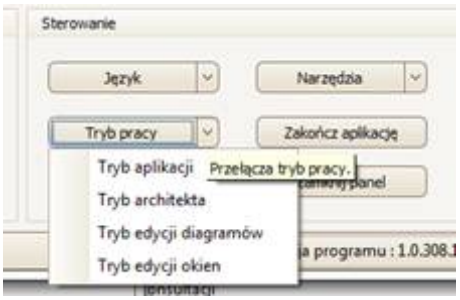
Uruchom program *AsixEvo.exe* przekazując w parametrach nazwę katalogu definicyjnego aplikacji oraz przełącznik */run*, np.

AsixEvo /run c:\asixapp\EC

Zalecane jest utworzenie skrótu uruchamiającego aplikację przy pomocy kreatora skrótów. Kreator uruchamiany jest komendą *Kreator skrótów* w menu *Narzędzia* okna architekta.

1.3 Przejście z trybu wykonawczego do trybu edycyjnego

Otwórz okno panelu kontrolnego przy pomocy kombinacji klawiszy *ctrl-shift-F1*.



Wybierz nowy tryb pracy komendami kontekstowymi przycisku *Tryb pracy*.

Komendy *Tryb edycji diagramów* i *Tryb edycji okien* powodują przejście do trybu uproszczonej edycji aktualnie otwartych diagramów lub okien. Inne zmiany w aplikacji nie będą możliwe. Powrót do trybu wykonawczego nastąpi w momencie zamknięcia okna przybornika edycyjnego.

Komenda *Tryb architekta* powoduje przejście do standardowego okna architekta z pełnymi możliwościami edycyjnymi. Powrót do trybu wykonawczego nastąpi w momencie wykonania komendy *Powrót do trybu działania* z menu głównego okna architekta.

Funkcje otwarcia panelu kontrolnego oraz zmiany trybu działania mogą zostać wkomponowane w diagramy aplikacji poprzez użycie akcji operatorskich *ShowControlPanel* i *SwitchWorkMode*.

1.4 Edycja dużych okien i diagramów

W przypadku posiadania dwóch monitorów można zdecydować się na oddokowanie przy pomocy myszki paneli roboczych typu *Właściwości* i *Eksplorator aplikacji* i przesunięcie ich na drugi monitor. Zwolni to miejsce na monitorze podstawowym.

Alternatywna metoda to użycie komendy *Tryb pełnoekranowy* z menu głównego. Zostaną ukryte wszystkie zbędne elementy – pozostanie jedynie edytowane okno lub diagram i uproszczony panel uniwersalnego przybornika. Jeżeli panel będzie przesłaniał interesujący nas fragment ekranu, to wystarczy w momencie najechania kursorem myszki przytrzymać wciśnięty klawisz *shift* – okno przybornika automatyczne się odsunie.

Powrót do zwykłego trybu edycyjnego nastąpi w momencie zamknięcia przybornika.

1.5 Testowanie działania diagramów w trakcie edycji

Pierwszy test działania diagramu odbywa się bezpośrednio w trakcie jego edycji. Wygląd obiektów odzwierciedla aktualny stan zmiennych procesowych. Jeżeli konfiguracja kanałów, archiwów i parametrów źródła danych została wykonana, to pobierane są rzeczywiste dane. Można też posłużyć się panelem *Podgląd zmiennych* w celu weryfikacji wartości zmiennych lub ich symulowania.

W trakcie edycji nie wszystko można sprawdzić. Nie są wykonywane animacje ruchu oraz obsługa właściwości widoczności obiektu. Nie są też dostępne funkcje interaktywne obiektów. Przejść do pełnego testu działania można poprzez wykonanie komendy *Podgląd* menu kontekstowego lub użycie przycisku  paska narzędziowego.

Panel *Podgląd zmiennych* jest też dostępny w trybie testowania diagramu. Należy otworzyć panel kontrolny (*ctrl-shift-F1)* i otworzyć panel podglądu komendą kontekstową przycisku *Narzędzia*.

Powrót do trybu edycji nastąpi w momencie zamknięcia diagramu lub użycia przycisku *Powrót do Architekta* w panelu kontrolnym.

1.6 Testowanie działania aplikacji w trakcie edycji

Wykonaj komendę *Uruchom aplikację* z menu głównego okna architekta. Zostaną otwarte okna startowe aplikacji z załadowanymi domyślnymi diagramami. Możliwe będzie przetestowanie działania diagramów łącznie z działaniem funkcji przełączania diagramów w panelach okien.

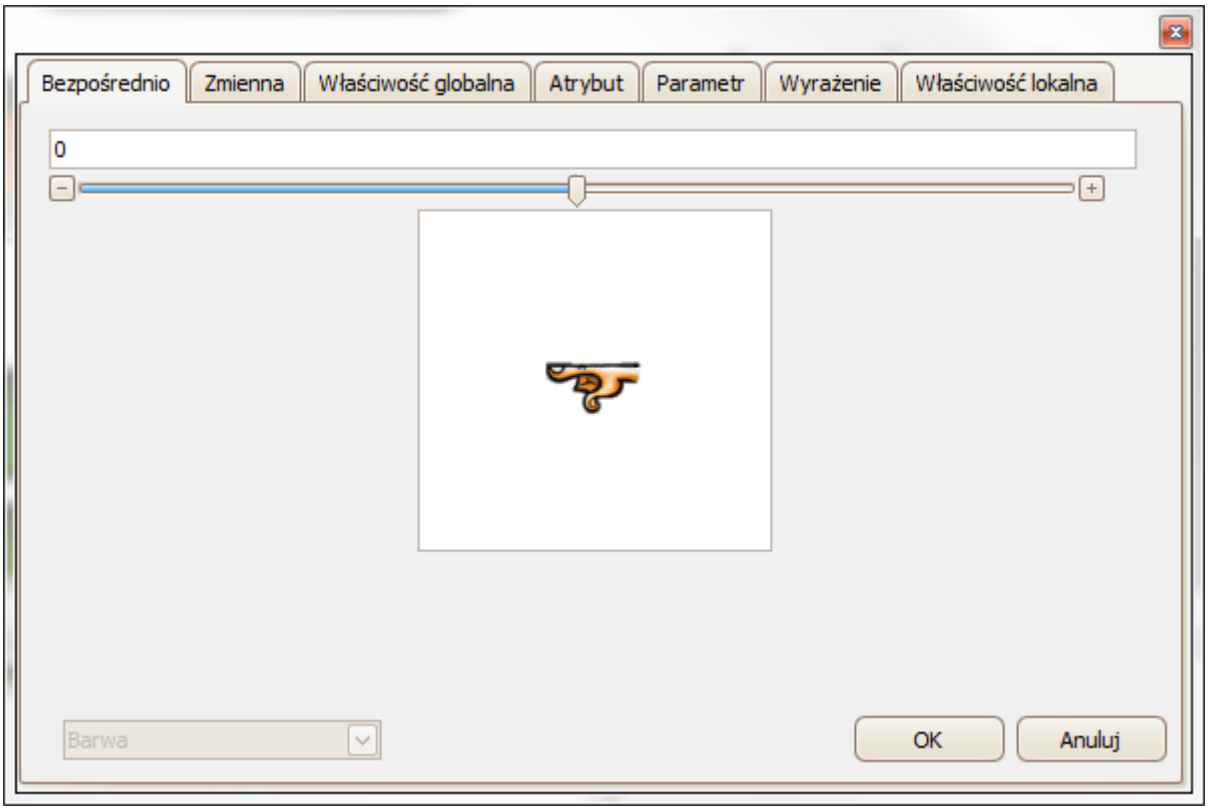
Alternatywnie, można otworzyć okno do edycji i wykonać komendę *Uruchom normalnie* z menu kontekstowego. Bieżące okno zostanie otwarte z w trybie testowym z załadowanymi domyślnymi diagramami.

Powrót do trybu architekta nastąpi w momencie zamknięcia wszystkich okien testowych.

2 Edycja właściwości

2.1 Edycja interaktywna

Użyj przycisk  umieszczony w polu właściwości. Otwarte zostanie okno dialogowe z zestawem zakładek służących do wyboru trybu edycji właściwości.



Wybierz właściwy wariant i wprowadź wartość lub wybierz z listy dostępnych wartości.

2.2 Edycja ręczna

Zacznij wpisywać nową wartość. W zależności od rodzaju właściwości zostanie wyświetlona lista możliwych wartości zgodnych z kontekstem użycia i dotychczas wprowadzoną wartością. Dokończ edycję lub wybierz wartość z listy.

2.3 Użycie wartości wcześniej wprowadzonych

Użyj przycisk  umieszczony w polu właściwości. Zostanie wyświetlona lista wcześniej użytych wartości zgodnych z typem edytowanej właściwości. Wybierz odpowiednią wartość.

2.4 Zmiana właściwości typu wyliczeniowego

Kliknij podwójnie na polu właściwości. Wartość będzie się zmieniała sekwencyjnie.

2.5 Wprowadzenie odwołania do zmiennej procesowej

Użyj przycisk  umieszczony w polu właściwości. Wybierz zmienną w oknie wyboru zmiennych procesowych. Alternatywna metoda to rozpoczęcie wprowadzenia od znaku &. Zostanie wyświetlona prosta lista nazw zmiennych zgodnych z dotychczas wprowadzoną wartością. Wybierz zmienną z listy. Dla właściwości zmiennej głównej i sterowanej, znak & trzeba pominąć – należy od razu rozpocząć wprowadzanie nazwy.

2.6 Wprowadzenie odwołania do atrybutu zmiennej

Rozpocznij od wprowadzenia znaku @. Zostanie wyświetlona lista nazw atrybutów zgodnych z dotychczas wprowadzoną wartością. Wybierz nazwę atrybutu z listy. Odwołanie do atrybutu przez notację *@nazwa* oznacza odczyt atrybutu zmiennej głównej obiektu.

2.7 Wprowadzenie odwołania do parametru diagramu lub wzorca

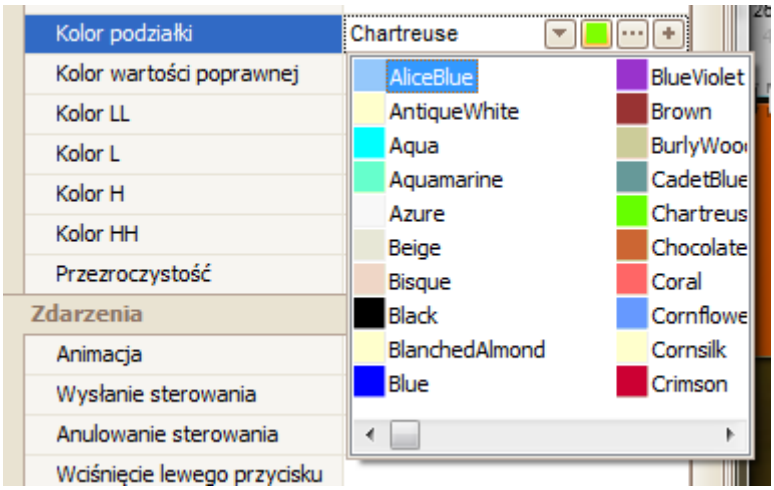
Rozpocznij od wprowadzenia znaku %. Zostanie wyświetlona lista nazw parametrów zgodnych z dotychczas wprowadzoną wartością. Wybierz parametr z listy.

2.8 Wprowadzenie odwołania do właściwości globalnej

Rozpocznij od wprowadzenia znaku !. Zostanie wyświetlona lista nazw właściwości globalnych zgodnych z dotychczas wprowadzoną wartością. Wybierz nazwę właściwości z listy.

2.9 Wybór z okna wyboru elementu

Dla niektóre rodzajów właściwości możliwy jest wybór z wartości w oknie wyboru elementu. Dotyczy to właściwości kolorów, nazwy obrazka, rodzaju kursora. Dla tych właściwości w polu edycji pojawi się dodatkowy przycisk – jego wygląd zależy od typu wartości.



3 Edycja okien i diagramów

3.1 Utworzenie okna lub diagramu

W panelu eksploratora aplikacji kliknij dwukrotnie na węźle *Okna* lub *Diagramy*, lub na węźle grupy okien lub diagramów.

Alternatywnie, w trakcie edycji okna, użyj komendę *Utwórz dopasowany diagram* z menu

kontekstowego panelu okna. Spowoduje to utworzenie diagramu o rozmiarze zgodnym z

aktualnym rozmiarem panelu.

3.2 Utworzenie grupy okien lub diagramów

W panelu eksploratora aplikacji użyj komendę *Utwórz okno* menu kontekstowego węzła *Okna* lub *Diagramy*.

3.3 Zmiana lokalizacji okna, diagramu lub grupy okien lub diagramów

W panelu eksploratora aplikacji przeciągnij wybrany element i upuść w nowym miejscu.

3.4 Zmiana nazwy okna, diagramu lub grupy

Użyj komendę *Zmień nazwę* z menu kontekstowego elementu. Alternatywnie, otwórz element do edycji i zmień właściwość *Nazwa okna* lub *Nazwa diagramu.*

3.5 Utworzenie kopii okna lub diagramu

Użyj komendę *Klonuj okno* lub *Klonuj diagram* z menu kontekstowego elementu.

3.6 Otwarcie okna lub diagramu do edycji

W panelu eksploratora aplikacji kliknij dwukrotnie na węźle wybranego okna lub diagramu.

3.7 Zmiana wyglądu okna

Ustaw właściwości okna odpowiadające za wygląd okna.

3.8 Dodanie panelu do okna

Otwórz okno do edycji. Z menu kontekstowego okna użyj komendę *Dodaj panel*.

3.9 Zmiana położenia panelu

Zaznacz panel myszką i przeciągnij w nowe miejsce. Ewentualnie zaznacz panel i zmień właściwości *X* i *Y*.

3.10 Przyklejenie panelu do wybranej krawędzi okna

Zaznacz panel i zmień właściwość *Dokowanie*.

3.11 Zmiana rozmiaru panelu

Zaznacz panel myszką i przeciągnij któryś z kwadratowych markerów na ramce selekcji. Ewentualnie zaznacz panel i zmień właściwości *Szerokość* i *Wysokość*.

3.12 Dołączenie menu do okna

Wpisz nazwę utworzonego wcześniej menu do właściwości *Menu* okna.

3.13 Utworzenie nowego obiektu

W panelu *Przybornik* w sekcjach *Obiekty*, *Ulubione* lub *Ostatnio używane* zlokalizuj obiekt właściwego typu. Następnie przeciągnij obiekt w odpowiednie miejsce ekranu. Zostanie założony obiekt o standardowej wielkości. Zmień następnie rozmiar obiektu. Alternatywnie kliknij w przyborniku na nazwie obiektu i zaznacz myszką obszar na diagramie, w którym należy utworzyć obiekt.

W przypadku tworzenia kilku obiektów tego samego typu, kolejne obiekty można tworzyć przez wybranie nazwy klasy obiektu z menu kontekstowego diagramu i zaznaczenie obszaru obiektu.

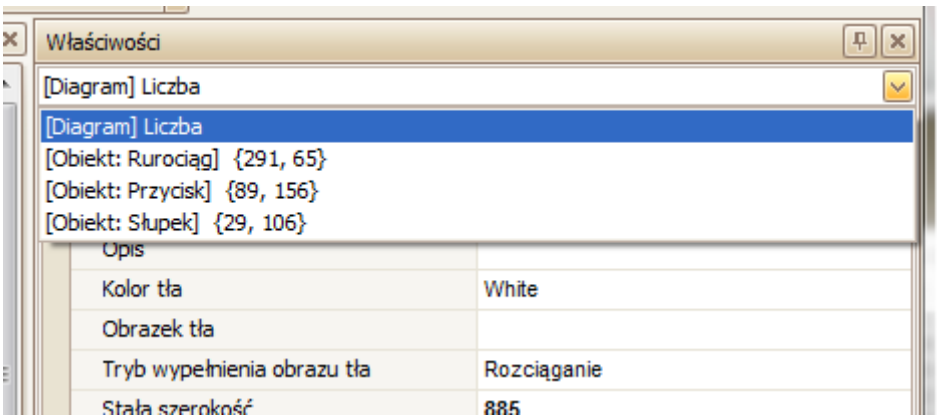
3.14 Utworzenie nowego obiektu wierzchołkowego

Niektóre obiekty (np. *Rurociąg*) w momencie zakładania definiowane są zestawem wierzchołków. W tym przypadków, zamiast zaznaczania ramki, należy kliknąć kolejno w miejscach wierzchołków. Sekwencję oznaczania wierzchołków kończy kliknięcie prawym przyciskiem myszki.

3.15 Wybranie elementu do edycji

Kliknij myszką w obszarze elementu. W panelu *Właściwości* zostaną pokazane definicje właściwości wybranego elementu.

Jeżeli element jest niewidoczny, to można wybrać element z listy elementów dostępnej w panelu *Właściwości.*



Jeżeli kilka obiektów znajduje się w tym samym obszarze ekranu, to kolejne kliknięcia wybierają sekwencyjnie poszczególne obiekty.

3.16 Wybranie wielu obiektów

Zaznacz myszką ramkę otaczającą obiekty, które należy zaznaczyć. Zaznaczanie należy zacząć w obszarze niezajętym przez żaden obiekt. Jeżeli ramka będzie zaznaczana od lewej do prawej strony, to wybrane zostaną wszystkie obiekty chociaż częściowo objęte ramką. Jeżeli ramka będzie zaznaczana od prawej do lewej strony, to wybrane zostaną tylko obiekty całkowicie objęte ramką.

Alternatywna metoda selekcji to kolejne kliknięcie na wybieranych obiektach z przytrzymaniem wciśniętego klawisza *Ctrl*.

3.17 Wybranie wszystkich obiektów tego samego typu

Wybierz jeden z obiektów, a następnie wykonaj komendę *Zaznacz wszystkie tego samego typu* z menu kontekstowego lub użyj przycisk  paska narzędziowego.

3.18 Wybranie wszystkich obiektów

Wykonaj komendę *Zaznacz wszystko* menu kontekstowego lub użyj przycisk  paska narzędziowego (skrót *ctrl-a*).

3.19 Utworzenie grupy obiektów

Czasami korzystne jest połączenie obiektów w grupę. Grupy ułatwiają edycję dużych diagramów. Pozwalają też na wspólne sterowanie animacją ruchu obiektów. W przypadku wybrania obiektu należącego do grupy, wybierane są też automatycznie pozostałe obiekty grupy.

Po wybraniu obiektów grupowanie wykonaj komendą *Grupuj* z menu kontekstowego *Grupowanie* lub przycisku  paska narzędziowego (skrót klawiszowy *ctrl-g*). Grupa może być usunięta korzystając z komendy *Rozgrupuj* (skrót *ctrl-u*).

Jeżeli zachodzi potrzeba pojedynczej edycji obiektów grupy, można je tymczasowo rozgrupować korzystając z komendy *Rozgrupuj tymczasowo* (skrót *ctrl-t*). Obiekty zostaną automatycznie ponownie zgrupowane przy kolejnym otwarciu diagramu. Można też wykonać ponowne zgrupowanie poprzez zaznaczenie dowolnego z obiektów grupy i użycie komendy *Grupuj ponownie*.

3.20 Kopiowanie obiektów

Zaznacz obiekty, które mają zostać skopiowane i wykonaj komendę *Klonuj* menu kontekstowego lub użyj przycisk  (skrót klawiszowy *ctrl-d*).

Alternatywny sposób, pozwalający na skopiowanie obiektów także na inny diagram, to użycie metody kopiuj/wklej. Zaznacz obiekty do skopiowania. Wykonaj komendę *Kopiuj* menu kontekstowego lub użyj przycisk  (skrót *ctrl-c*). Wybierz diagram, na który należy skopiować obiekty i wykonaj komendę *Wklej* lub użyj przycisk  (skrót *ctrl-v*). W przypadku korzystania ze skrótu klawiszowego, można wskazać myszką dokładne miejsce skopiowania obiektów.

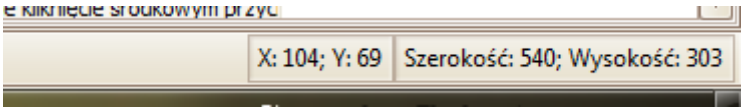
Zamiast komendy *Wklej* można też użyć komendę *Wklej ze skalowaniem*. W przypadku kopiowania obiektów pomiędzy diagramami o różnych rozmiarach, obiekty zostaną odpowiednio przeskalowane.

3.21 Zmiana pozycji obiektów

Zaznacz obiekt lub obiekty, które należy przesunąć. Wciśnij lewy klawisz myszki i przeciągnij obiekty do nowej pozycji.

Można też użyć przycisków strzałek na klawiaturze. Przyciśnięcie takiego przycisku powoduje przesunięcie obiektów o pojedynczy piksel w wybranym kierunku. Jeżeli dodatkowo zostanie wciśnięty klawisz *Shift*, to przesunięcie będzie wykonywane co 10 pikseli.

Aktualne koordynaty obiektów można śledzić w polu wyświetlanym w pasku statusowym.



Pozycję obiektu można zmienić też przez bezpośrednie ustawienie właściwości obiektu *X* i *Y.*

3.22 Zmiana rozmiaru pojedynczego obiektu

Zaznacz obiekt, którego rozmiar ma być zmieniony. Następnie najedź kursorem myszki na jeden z markerów zmiany rozmiaru na ramce selekcji. Wciśnij lewy klawisz myszki i przeciągnij marker w nowe miejsce. Po wciśnięciu przycisku myszki można też kontynuować zmianę rozmiaru przy pomocy klawiszy strzałek, ewentualnie z dodatkowym wciśnięciem klawisza *Shift.*

Kombinacja klawiszy C*trl-strzałka* pozwala na zmianę aktywnego markera zmiany pozycji.

Rozmiar obiektu można zmienić też przez bezpośrednie ustawienie właściwości obiektu *Szerokość* i *Wysokość.*

3.23 Zmiana rozmiaru grupy obiektów

Zaznacz grupę obiektów, których rozmiar ma być zmieniony. Następnie postępuj jak dla przypadku pojedynczego obiektu.

Różnica w działaniu, w stosunku do pojedynczego obiektu, jest taka, że zmiana rozmiaru bezpośrednio dotyczy tylko ramki selekcji. Obiekty zmieniają swój rozmiar proporcjonalnie do zajmowanego obszaru.

Jeżeli chodzi o ustawienie jednakowej szerokości lub wysokości kilku obiektów, to po ich zaznaczeniu należy bezpośrednio zmienić wartość właściwości *Szerokość* lub *Wysokość*.

3.24 Edycja kształtu obiektów wierzchołkowych

Zaznacz obiekt przez podwójne kliknięcie. Oprócz standardowej ramki selekcji wyświetlone zostaną również markery wierzchołków. Zaznacz odpowiedni marker przy pomocy myszki i przeciągnij go w nowe położenie. Do przesunięcia markera można też używać klawiszy strzałek lub kombinacji *shift-strzałka*. Zmienić aktywny markera można przy pomocy klawiszy *ctrl-strzałka*.

Jeżeli zachodzi potrzeba dodania lub usunięcia wierzchołków, to należy użyć komendy *Wstaw wierzchołek* i *Usuń wierzchołek* z menu kontekstowego *Wierzchołki* lub z menu przycisku .

3.25 Transformacja kształtu obiektów wierzchołkowych

W przypadku obiektów wierzchołkowych możliwe są transformacje ich kształtu polegające na przekształceniu symetrycznym wierzchołków lub ich obrocie.

Po wybraniu obiektu należy użyć odpowiednie komendy z menu kontekstowego *Wierzchołki/Transformacje wierzchołków* lub z menu *Transformacje wierzchołków* przycisku .

3.26 Dopasowanie rozmiaru obiektu do rozmiaru obrazka

Wybierz obiekt klasy *Obraz* i wykonaj komendę *Dopasuj do rozmiaru obrazu* lub użyj przycisk . Rozmiar obiektu zostanie tak ustawiony żeby wyświetlany obraz nie podlegał skalowaniu lub obcinaniu.

3.27 Dopasowanie rozmiaru obiektu do rozmiaru diagramu

Wybierz obiekt i wykonaj komendę *Dopasuj do diagramu* z menu kontekstowego *Wyrównaj do* lub przycisku . Obiekt wypełni cały obszar diagramu.

3.28 Wycofanie zmian położenia i rozmiaru obiektów oraz odtworzenie obiektów usuniętych

Edytor diagramów przechowuje wielopoziomową historię zmian rozmiaru i położenia obiektów oraz obiektów kasowanych. Można powrócić do wcześniejszych wersji diagramu przez wielokrotne wykonanie komendy *Cofnij* menu kontekstowego lub użycie przycisku  paska narzędziowego (skrót klawiszowy *ctrl-z*).

3.29 Tymczasowe ukrywanie obiektów w trybie edycji

W przypadku diagramów z dużą ilością obiektów, szczególnie nałożonych jeden na drugim, pomocne może być tymczasowe ukrycie niektórych obiektów.

Zaznacz obiekty, które należy ukryć. Wykonaj komendę *Wybrane obiekty* z menu kontekstowego *Ukryj* lub przycisku .

Alternatywnie, wybierz obiekt, który ma zostać odsłonięty i Wykonaj komendę *Obiekty przykrywające* z menu kontekstowego *Ukryj* lub przycisku . Zostaną ukryty wszystkie obiektu, których położenie pokrywa się w jakikolwiek sposób z wybranym obiektem.

Aby ponownie pokazać wszystkie obiekty, użyj komendę *Pokaż wszystkie* z menu kontekstowego lub kliknij na przycisku .

3.30 Blokowanie zmian rozmiaru i położenia obiektów

Możliwe jest zabezpieczenie się przed przypadkową zmianą rozmiaru lub położenia obiektów w trakcie edycji diagramu. W tym celu zaznacz obiekty i wykonaj odpowiednią komendę z menu kontekstowego *Zablokuj* lub przycisku .

Żeby usunąć blokady, zaznacz obiekty i wykonaj komendę *Odblokuj* menu kontekstowego lub użyj przycisk .

3.31 Ustalenie kolejności przesłaniania się obiektów

Sposób wzajemnego przesłaniania się obiektów położonych w tym samym obszarze diagramu sterowany jest wartością właściwości *Warstwa*. Obiekty o wyższym numerze warstwy przesłaniają

inne obiekty. Można sterować przesłanianiem przez jawne ustawienie właściwości *Warstwa*. Można też ustawić sposób przesłania interakcyjnie.

Wybierz obiekt lub obiekty i wykonaj którąś z komend menu kontekstowego *Kolejność* lub przycisku . Zmieni się sposób przesłaniania, a jednocześnie ulegnie zmianie właściwość *Warstwa* zaznaczonych obiektów. Można też stosować skróty klawiszowe: *ctrl-home* przesuwa obiekty na szczyt, a *crtl-end* na sam spód.

3.32 Wyrównywanie położenia obiektów

Wybierz obiekty i wykonaj odpowiednią komendę z menu kontekstowego *Wyrównaj do* lub przycisku . Można też użyć któryś ze skrótów klawiszowych *alt-strzałka*. W zależności od użytej komendy lub skrótu, odpowiednia krawędź wybranych obiektów zostanie wyrównana.

3.33 Dosuwanie obiektów

Jeżeli zachodzi potrzeba takiego rozmieszczenia obiektów, aby przylegały one jeden do drugiego, to zaznacz te obiekty i wykonaj odpowiednią komendę z menu kontekstowego *Dosuń do* lub przycisku . W zależności od użytej komendy lub skrótu, obiekt skrajny pozostanie bez zmiany, a pozostałe zostaną tak spozycjonowane żeby do siebie przylegały w odpowiedniej osi.

3.34 Rozsuwanie obiektów

Czasami zachodzi konieczność równomiernego rozmieszczenia obiektów na diagramie. W tym celu zaznacz obiekty (co najmniej 3) i wykonaj komendę *Poziomo* lub *Pionowo* z menu kontekstowego *Rozsuń* lub przycisku .

Obiekty skrajne pozostaną niezmienione, a pozostałe zostaną tak rozmieszczone, żeby odstęp pomiędzy ich lewymi (górnymi) krawędziami pozostawał stały.

3.35 Nakładanie obiektów

Wybierz obiekty, które chcesz nałożyć centralnie jeden na drugim. W zależności od wymaganego sposobu centrowania obiektów wykonaj odpowiednią komendę z menu kontekstowego *Wyśrodkuj* lub z menu przycisku .

3.36 Utworzenie wzorca z obiektów diagramu

Wybierz obiekty diagramy, z których należy utworzyć wzorzec. Wykonaj komendę *Połącz we wzorzec* z menu kontekstowego *Grupowanie* lub z menu przycisku  (skrót klawiszowy *ctrl-p)*. W otwartym panelu edycyjnym diagramu zmień właściwość *Nazwa diagramu*, a następnie wykonaj operacje zapisu (skrót *ctrl-s*).

3.37 Osadzenie wzorca na diagramie

W panelu *Przybornik* w sekcjach *Wzorzec*, *Ulubione* lub *Ostatnio używane* zlokalizuj nazwę właściwego wzorca. Następnie przeciągnij wzorzec w odpowiednie miejsce ekranu. Wzorzec zostanie osadzony na diagramie, a jego rzomiar będzie zgodny z rozmiarem oryginalnym. Alternatywnie kliknij w przyborniku na nazwie wzorca i zaznacz myszką obszar na diagramie, w którym należy osadzić wzorzec. Rozmiar wzorca wraz z jego obiektami składowymi zostanie odpowiednio przeskalowany.

Oryginalny rozmiar wzorca można ustawić przy pomocy komendy *Oryginalny rozmiar wzorca* menu kontekstowego lub poprzez użycie przycisku  paska narzędziowego.

W przypadku wzorców parametryzowanych, po jego osadzeniu należy jeszcze w panelu właściwości określić parametry osadzenia.

3.38 Odłączenie osadzonego wzorca od źródła

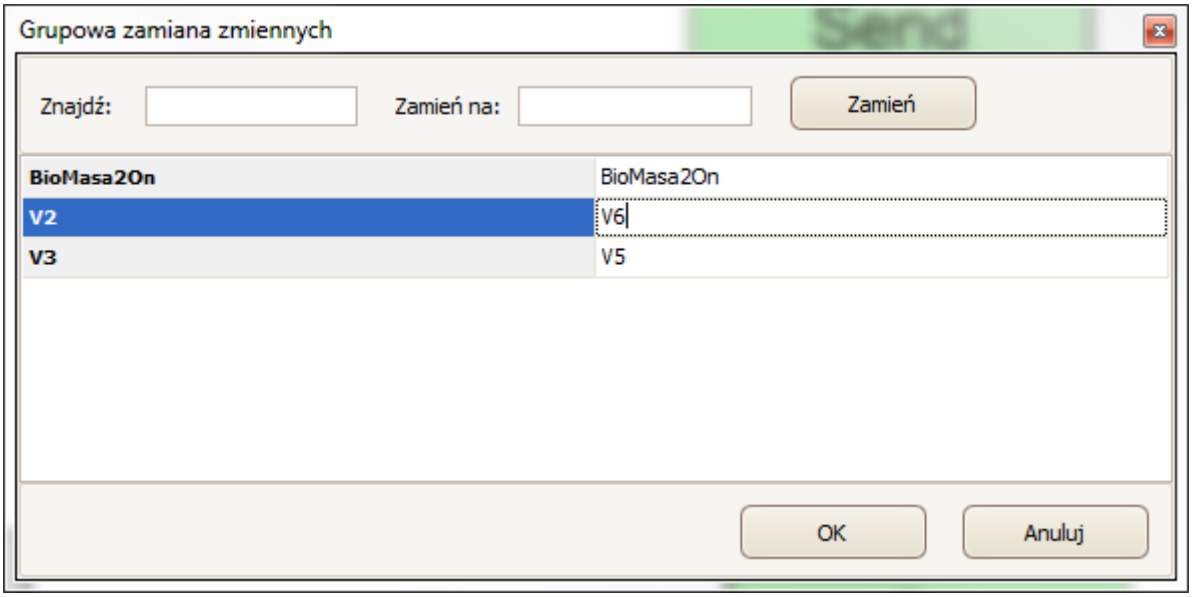
Wybierz osadzony diagram i wykonaj komendę *Rozłącz wzorzec* z menu kontekstowego *Grupowanie* lub z menu przycisku .

Po rozłączeniu, wszystkie obiekty składowe wzorca będą dostępne do normalnej edycji. Późniejsze zmiany definicji wzorca nie będą miały żadnego znaczenia dla rozłączonych obiektów.

3.39 Grupowa zamiana używanych zmiennych

W przypadku pojedynczych obiektów prostą metodą zamiany nazw używanych zmiennych jest bezpośrednia modyfikacja odpowiednich właściwości obiektu. Jeżeli zmiana polega na podmianie zmiennej głównej w wielu obiektach na tą samą nazwę, to można wybrać wszystkie obiekty, w których zmiana ma być wykonana i wprowadzić nową nazwę w polu właściwości *Główna zmienna*.

W przypadku ogólnym można posłużyć się funkcją globalnej podmiany. Wybierz obiekty, w których zmiana ma być wykonana (w szczególności użyj *ctrl-a* aby zaznaczyć wszystkie obiekty) i wykonaj komendę *Grupowa zamiana zmiennych* menu kontekstowego lub użyj przycisk .



W wyświetlonym oknie dialogowym, w lewej kolumnie pokazane są wszystkie oryginalnie użyte zmienne, a w prawej nowe nazwy.

W okienku pokazane są wszystkie zmienne użyte we właściwościach określających bezpośrednio nazwę zmiennej (np. *Główna zmienna*) oraz zmienne użyte w notacji skrótowej &.

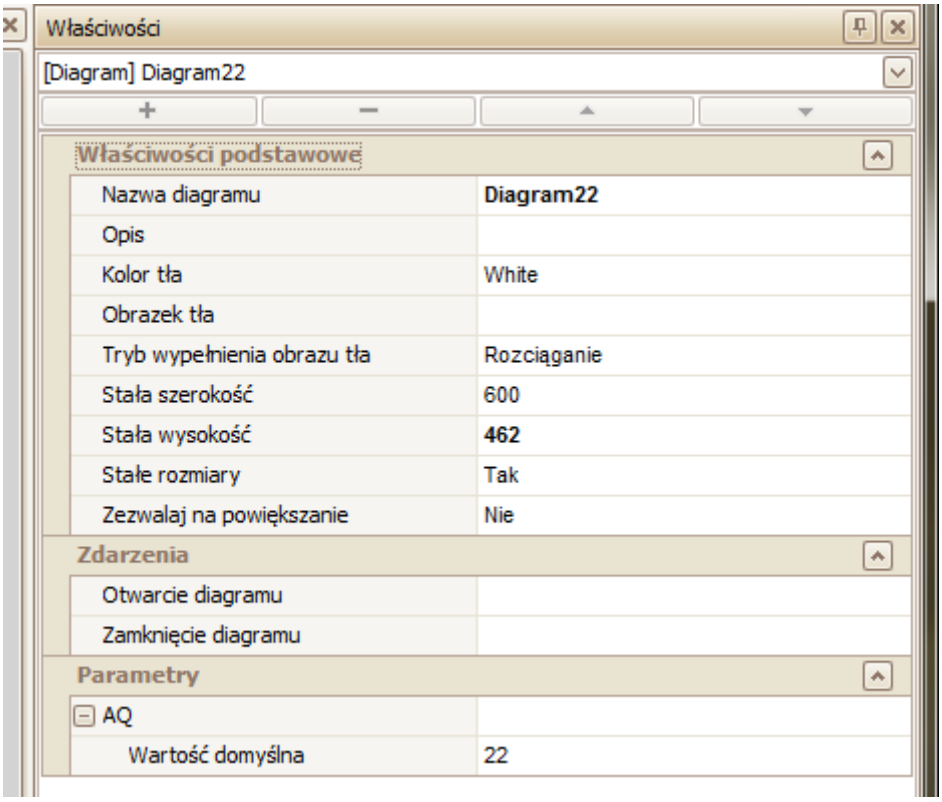
3.40 Podgląd wartości i symulacja zmiennych

W trakcie edycji diagramu pomocny może być podgląd na bieżące wartości zmiennych, ewentualnie ich symulacja lub ręczne ustawianie. Wszystkie te funkcje spełnia panel *Podgląd zmiennych* otwierany komendą *Podgląd zmiennych* z menu *Widok*. Zmienne do panelu podglądu można dodać przy pomocy przycisków umieszczonych wewnątrz panelu. W przypadku podglądu zmiennych użytych w konkretnych obiektach można posłużyć się funkcją automatycznego dodawania zmiennych.

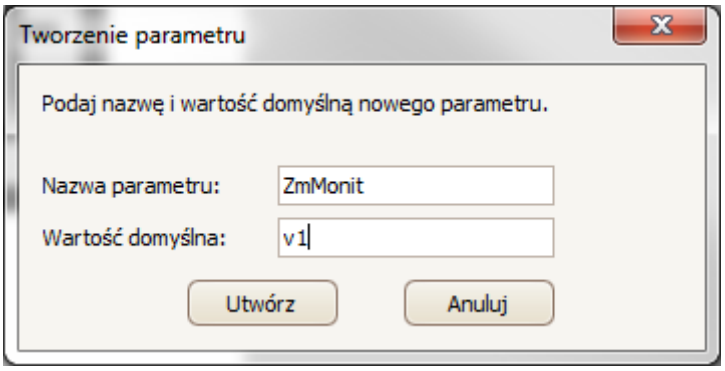
Wybierz obiekty, których zmienne mają być pokazane w panelu podglądu (w szczególności użyj *ctrl-a* aby zaznaczyć wszystkie obiekty) i wykonaj komendę *Dodaj zmienne do symulacji* menu kontekstowego lub użyj przycisk .

3.41 Dodanie parametrów do diagramu lub wzorca

Kliknij myszką w obszarze diagramu lub wzorca nieprzykrytym obiektami lub użyj przycisk  aby usunąć selekcję obiektów. W panelu *Właściwości* wyświetlone zostaną właściwości diagramu (wzorca). Kliknij na wierszu *Parametry*, a następnie na przycisku *+*.



Zostanie wyświetlone okno definicji nowego parametru.



Wprowadź nazwę parametru oraz jego wartość domyślną. Wartości domyślne są używane wtedy, gdy diagram został otwarty bez podania parametrów. W szczególności przy otwarciu diagramu w trybie edycyjnym.

Przyciskiem – w panelu właściwości można usunąć wskazany parametr.

Można też utworzyć nowy parametr przez odwołanie się do nieistniejącego parametru. W dowolnej właściwości obiektu wprowadź nazwę parametru w notacji skrótowej %. Jeżeli taki parametr nie będzie istniał, to program zaproponuje jego utworzenie.

4 Inne operacje edycyjne

4.1 Dodanie obrazka do aplikacji

Każdy obrazek graficzny używany w aplikacji musi być najpierw do tej aplikacji dodany. W tym celu, za pośrednictwem panelu eksploratora aplikacji, należy otworzyć panel roboczy *Obrazki*, a następnie korzystając z menu kontekstowego panelu użyć komendę *Importuj z pliku* lub *Importuj ze standardowych*. W pierwszym przypadku będzie można dodać dowolny obrazek znajdujący się na dysku komputera, a w drugim przypadku obrazek z katalogu instalacyjnego pakietu. W obu przypadkach plik zostanie skopiowany do podkatalogu *Images* w katalogu definicyjnym aplikacji. Od tego momentu będzie można go używać w aplikacji.

4.2 Edycja obrazka

Otwórz panel roboczy *Obrazki*, zaznacz właściwy obrazek i z menu kontekstowego wykonaj komendę *Edytuj.*

Alternatywnie, zaznacz na diagramie obiekt typu *Obraz* i użyj przycisk  paska narzędziowego. Aktualnie pokazywany obrazek zostanie otwarty w edytorze graficznym.

Obrazki są edytowane przy pomocy zewnętrznego edytora. Można wybrać dowolny edytor korzystając z okna parametrów otwieranego komendą *Ustawienia edycji diagramów* z menu *Widok*.

Znane problemy

Problem	Opis i rozwiązanie
Bardzo długi czas uruchamiania AsixEvo i innych plików wykonywalnych pakietu	Wszystkie pliki wykonywalne pakietu AsixEvo są podpisane cyfrowo certyfikatem firmy Askom. Przy niektórych konfiguracjach komputerów, które nie posiadają dostępu do Internetu a tym samym do serwera weryfikacji certyfikatów, system Windows długo próbuje zweryfikować certyfikat programu przez co znacznie wydłuża jego start. Rozwiązaniem problemu jest w takim wypadku utworzenie dla wszystkich plików wykonywalnych (exe) w katalogu instalacyjnym AsixEvo plików konfiguracyjnych o tej samej nazwie co plik wykonywalny z rozszerzeniem „.config" np.: dla pliku *„AsixEvo.exe"* będzie to plik o nazwie *„AsixEvo.exe.config"*. Każdy z tych plików powinien zawierać poniższy wpis: <?xml version="1.0" encoding="utf-8"?> <configuration> <runtime> <generatePublisherEvidence enabled="false" /> </runtime> </configuration> W podkatalogu *„ExeConfigs"* katalogu instalacyjnego AsixEvo znajdują się gotowe pliki konfiguracyjne dla plików wykonywalnych. W przypadku wystąpienia opisanego problemu wystarczy je skopiować do głównego katalogu instalacyjnego.
Błąd ładowania w przeglądarce aplikacji AsixEvo opublikowanych na systemie Windows Server 2003	W menadżerze IIS dla katalogu wirtualnego *EvoNet* należy dodać dwa typy MIME: • Dla .dll typ application/x-msdownload • Dla .mdb typ application/x-msdownload • Dla .* typ text/plain Ustawienia te są dostępne we właściwościach katalogu wirtualnego w zakładce *Nagłówki HTTP* w sekcji *Typy MIME*.

Wyświetlany w przeglądarce błąd dostępu do zasobów serwera IIS, „Failed to access IIS metabase”	W linii poleceń systemu Windows na serwerze na którym jest opublikowana aplikacja Asixa Evo należy wykonać polecenia: • %systemroot%\Microsoft.NET\Framework\v2.0.50727\ aspnet_regiis –r • %systemroot%\system32\iisreset.exe
Błąd typu ***“Method not found: 'Boolean System.Threading.WaitHandle.*** ***WaitOne(System.TimeSpan)'”*** **lub inny wskazujący na nieprawidłowe odwołanie do metod systemowych .NetFramework.**	AsixEvo wymaga Microsoft .Net Framework w wersji 3.51 (z Service Pack 1). Jeżeli w systemie jest zainstalowany tylko .Net Framework 3.5 to należy doinstalować do niego SP1.
Błąd typu: ***„Nie można uzyskać dostępu do metabazy usług IIS”*** **występujący na systemie Windows XP spowodowany może być niezalecaną kolejnością instalacji IIS, po zainstalowaniu pakietów .Net Framework.**	W celu usunięcia błędu należy wykonać polecenie: • %systemroot%\Microsoft.NET\Framework\v2.0.50727\ aspnet_regiis –r oraz jeżeli posiadamy zainstalowana wersje .NET Framework4 dodatkowo polecenie: • %systemroot%\Microsoft.NET\Framework\v4.0.30319\ aspnet_regiis.exe. Po wykonaniu poleceń aspnet_regiis konieczne jest wykonanie restartu IIS np. za pomocą polecenia: • iisreset Jeżeli powyższa metoda nie naprawi błędu usługi IIS, należy odinstalować pakiet IIS oraz wszystkie wersje .NET Framework. Po wykonaniu tych czynności zainstalować IIS, a następnie wymagany .NET Framework 3.51. Instalacja innych wersji .NET Framework jest opcjonalna.

Rodzaje agregatów danych archiwalnych

W każdym przypadku, w którym w aplikacji wymagany jest dostęp do zagregowanych danych historycznych konieczne jest podanie nazwy agregatu. Dotyczy to obiektu *Wykres,* funkcji Evo, interfejsów modelu skryptowego.

Szczegółowy opis agregatów oraz ich parametryzacji znajduje się w dokumentacji modułu archiwizacji Aspad (patrz: Serwer Danych > Obsługa danych procesowych Serwera Danych > Aspad - program archiwizacji danych).
UWAGA: Nazewnictwo agregatów w wyżej wymienionych zastosowaniach różni się od nazw pojawiających się w parametryzacji danych archiwalnych Aspada (w programie Architekt.exe przy parametryzacji archiwów modułu Aspad - patrz: Serwer Danych > Obsługa danych procesowych Serwera Danych > Aspad - program archiwizacji danych > Agregacja danych archiwalnych > Zestaw agregatów).

Nazwa w Evo	Nazwa w Architekt.exe	Znaczenie
none		brak agregacji
start	Początek	Wartość na początku interwału.
end	Koniec	Wartość na końcu interwału.
delta	Przyrost	Różnica wartości na końcu i na początku interwału.
min	Min	Minimalna wartość w interwale.
max	Max	Maksymalna wartość w interwale.
range	Zakres	Różnica między maksymalna a minimalna wartością w interwale.
total	Suma	Suma wartości ważonych czasowo w interwale (całka po czasie).
average	Średnia	Średnia wartości ważonych czasowo w interwale.
average0	Średnia0	Średnia wartości ważonych czasowo w interwale. W okresach, w których wartość jest niedostępna do obliczeń, używana jest wartość 0.
sumup	Suma_Wzrostów	Suma wartości dodatnich zmian, od ostatniej znanej wartości przed rozpoczęciem, do ostatniej wartości przed końcem przedziału próbkowania.
sumdown	Suma_Spadków	Suma wartości bezwzględnych ujemnych zmian, od ostatniej znanej wartości przed rozpoczęciem, do ostatniej wartości przed końcem przedziału próbkowania.
prevknown	Poprzedni_Znany	Ostatnio znana (dobra lub niepewna) wartość przed rozpoczęciem przedziału próbkowania. Podaje ostatnią wartość, nawet jeżeli wystąpiła ona bardzo dawno.
last	Ostatni	Ostatnio znana (dobra lub niepewna) wartość w przedziale próbkowania wyłączając koniec przedziału - dla odróżnienia od agregatu *end*, który zawiera koniec przedziału.
stdev	Odchylenie_Standardowe	Ważone czasowo odchylenie standardowe w przedziale próbkowania.
rms	Średnia_Kwadratowa	Ważona czasowo średnia kwadratowa w przedziale próbkowania.

avglk	Średnia_Ostatnia_Znana	Średnia ważona czasowo w przedziale próbkowania. W miejsce złych lub brakujących wartości wstawia ostatnią znaną wartość.
totallk	Suma_Ostatnia_Znana	Całka w przedziale próbkowania, przy czasie liczonym w sekundach. W miejsce złych lub brakujących wartości wstawia ostatnią znaną wartość.
Total0	Suma0	Suma wartości ważonych czasowo (całka po czasie); w okresach, w których próbka ma jakość zła, do obliczeń używana jest wartość 0.
QGood	Jakość_Dobra	Procent próbek o jakości dobrej w interwale.
QBad	Jakość_Zła	Procent próbek o jakości złej w interwale.
QPoor	Jakość_Niepewna	Procent próbek o jakości niepewnej w interwale.
DGood	Jakość_Dobra_Czas	Czas trwania (w sekundach) czasu w przedziale, w którym dane mają jakość dobrą.
DBad	Jakość_Zła_Czas	Czas trwania (w sekundach) czasu w przedziale, w którym dane mają jakość złą.
DPoor	Jakość_Niepewna_Czas	Czas trwania (w sekundach) czasu w przedziale, w którym dane mają jakość niepewną.
TofMin	Czas_Min	Stempel czasu wartości minimalnej.
TofMax	Czas_Max	Stempel czasu wartości maksymalnej.

Serwer Danych

1. Obsługa danych procesowych Serwera Danych

1.1. Serwer Danych

Aplikacje typu Evo do odczytu wartości zmiennych ze sterowników oraz ich archiwizacji wykorzystują moduły Asmen i Aspad. Moduły te uruchamiane są w ramach tzw. **serwera danych**.

Praktycznie w każdej aplikacji, przynajmniej jedno stanowisko pracuje w tym trybie. Serwer Danych odczytuje dane ze sterowników kontrolujących proces oraz może je archiwizować.

Parametryzacja Serwera Danych wymaga podania nazwy pliku konfiguracyjnego XML oraz wariantu pracy:

Asix.Evo.exe > eksplorator aplikacji > *Ustawienia stanowisk* > *Źródła danych* > *Ustawienia źródła danych procesowych*

Plik konfiguracyjny Serwera Danych może zostać utworzony w trakcie budowy nowej aplikacji przy pomocy kreatora. Do jego edycji służy program Architekt.

Przy pomocy przycisku  **można uruchomić program Architekt z załadowaniem właściwego pliku konfiguracyjnego.**

1.2. Asmen - menedżer komunikacyjny

Wymiana informacji pomiędzy systemem wizualizacji i sterownikiem obiektu technologicznego może odbywać się przy użyciu łączy szeregowych lub sieci lokalnych. Wybór rodzaju połączenia zależy od konfiguracji sprzętowej oraz wymagań stawianych systemowi wizualizacji odnośnie liczby wymienianych danych oraz częstości przeprowadzania transmisji.

Wymiana informacji między komputerami systemu Asix może odbywać się za pośrednictwem sieci lokalnej. Taki rodzaj połączenia nazywany będzie kanałem sieciowym. Umożliwia on udostępnianie wartości zmiennych bieżących oraz archiwalnych Asixa komputerom nie posiadającym fizycznego połączenia ze sterownikiem. Oczywiście co najmniej jeden komputer w sieci lokalnej musi mieć fizyczne połączenie ze sterownikiem lub też ich siecią (np. SINECL2), aby móc pracować jako serwer danych dla pozostałych komputerów.

Systemy cechujące się dużą liczbą wymienianych informacji wymagają z reguły wykorzystania sieci lokalnych, natomiast w przypadku pozostałych systemów można ograniczyć się do zastosowania sprzęgów na łączach szeregowych.

Projektowanie systemu Asix w zakresie dostępu do danych obiektowych polega na przygotowaniu deklaracji:

kanałów transmisji - kanałem transmisji jest fizycznie zrealizowane połączenie pomiędzy komputerem systemu Asix i sterownikiem na obiekcie, określone przez nazwę

drajwera działającego według określonego protokołu wraz z listą określonych parametrów. Kanał transmisji może stanowić również połączenie między komputerami systemu Asix przy użyciu sieci lokalnej. Ten typ połączenia stosuje się głównie w celu przekazania danych do komputerów pracujących jako tzw. terminale sieciowe, które nie posiadają połączenia ze sterownikami i nie tworzą zazwyczaj własnego archiwum. Istnieje również możliwość specyfikowania symulowanego kanału transmisji, który nie wymaga połączenia z obiektem. Opcja ta jest szczególnie przydatna na etapie projektowania i testowania aplikacji.

zmiennych procesowych - zmienną procesową jest przetworzona informacja opisująca stan określonego parametru technologicznego, powstała na bazie informacji odczytanych z obiektu;

Aktualnie zaleca się przechowywanie definicji zmiennych w bazie typy SQL lub Jet (Access) lub bezpośrednio w arkuszach kalkulacyjnych Excel - szczegóły **patrz:** *Serwer Danych > 2. Parametryzacja Serwera Danych > 2.2.1.2.1. Typ i format bazy danych*.

Istnieje również możliwość specyfikowania zmiennych procesowych, które służą wyłącznie dla wewnętrznych potrzeb aplikacji i nie posiadają powiązań z obiektem.

W oparciu o specyfikacje w plikach konfiguracyjnych tworzone jest repozytorium bieżących wartości zmiennych procesowych (przechowywane przez moduł Asmen w pamięci operacyjnej komputera), do którego mają dostęp wszystkie komponenty systemu Asix. Wartości przechowywanych zmiennych procesowych są na bieżąco aktualizowane poprzez cykliczny odczyt z obiektu. Informacje z obiektu przesyłane są najczęściej w postaci surowej i z tego powodu wymagają wstępnego przetworzenia. Przetworzenie to realizowane jest przy wykorzystaniu *funkcji przeliczających*, deklarowanych przez projektanta dla każdej zmiennej procesowej z osobna.

W systemie Asix wszystkie operacje związane z dostępem do zmiennych procesowych są realizowane przez moduł Asmen. Moduł Asmen nadzoruje proces cyklicznej aktualizacji wartości zmiennych procesowych w bazie danych oraz pośredniczy we wszystkich operacjach związanych z przesyłem sterowań do obiektu. Bezpośrednia wymiana danych pomiędzy komputerem i sterownikiem obiektu jest wykonywana poprzez drajwery pracujące pod kontrolą Asmena. Zestaw drajwerów wykorzystywanych w systemie wizualizacji zależy każdorazowo od sposobu połączenia danego systemu z obiektem.

Szczególnym trybem pracy modułu Asmen jest działanie w trybie symulacji. Polega ono na tym, że wartości zmiennych procesowych nie są określane w oparciu o dane pochodzące z fizycznego kanału transmisji, lecz są generowane przy użyciu specyficznych funkcji przeliczających, zaprojektowanych pod kątem symulowania pracy z obiektem. Tryb symulacji jest szczególnie przydatny na etapie projektowania aplikacji, gdy połączenie z obiektem jest najczęściej niemożliwe.

Parametry pracy modułu Asmen deklaruje się w programie Architekt:

program Architekt > *Obszary i komputery* > moduł *Dane bieżące*

Patrz: *Serwer Danych > 2. Parametryzacja Serwera Danych > 2.8. Konfiguracja parametrów danych bieżących*

1.2.1. Deklarowanie kanałów transmisji

Deklaracje kanałów transmisji pozwalają na przekazanie do menedżera Asmen listy drajwerów używanych przez aplikację oraz parametrów pracy tych drajwerów. Deklaracje kanałów transmisji wymagają określenia następujących parametrów:

nazwa_logiczna - nazwa logiczna kanału transmisji; długość nazwy logicznej nie może przekroczyć 9 znaków; znakami zastrzeżonymi są spacja, przecinek oraz znak tabulacji;

nazwa_drajwera - nazwa drajwera łącza fizycznego używanego w danym kanale transmisji; można użyć nazwę NONE, która oznacza protokół symulujący łącze fizyczne, oraz nazwę NETWORK, która oznacza kanał sieciowy;

parametry - lista parametrów, specyficzna dla każdego z realizowanych przez określone drajwery protokołów łącza fizycznego; protokół symulujący łącze fizyczne (NONE) oraz protokół sieciowy (NETWORK) nie wymagają podania parametrów.

Kanały transmisji deklarowane są w pliku konfiguracyjnym XML za pośrednictwem programu Architekt:

Architekt > moduł *Dane Bieżące*

Każdy kanał posiada swoją definicję, która jest dodawana do modułu *Dane bieżące* poprzez kliknięcie prawym klawiszem myszy na wspomnianym module i wybranie komendy *Dodaj kanał*.

Patrz: *Serwer Danych > 2. Parametryzacja Serwera Danych > Konfiguracja parametrów danych bieżących > 2.8.4. Definicja kanałów komunikacyjnych*.

Kluczową rolę w definicji kanału odgrywa wybór protokołu (realizowanego przez drajwer), według którego będzie realizowana transmisja danych. Protokół ten wynika oczywiście z tego, jak jest zrealizowane sprzętowo połączenie komputera systemu Asix ze sterownikiem obiektu.

Specyficznym rodzajem protokołu jest protokół o nazwie **NETWORK**. Nie realizuje on fizycznego połączenia ze sterownikiem, a jedynie połączenie z innym komputerem, który takie połączenie ma. Drajwer kanału sieciowego po nawiązaniu połączenia wysyła zapytanie do wszystkich serwerów, podając nazwę logiczną kanału. Odpowiadają na to pytanie jedynie serwery, które posiadają zadeklarowany taki kanał. Jeżeli odpowie więcej niż jeden serwer, to wtedy do nawiązania połączenia wybiera się serwer najmniej obciążony. Jeżeli w czasie pracy nastąpi wyłączenie serwera, wówczas drajwer kanału sieciowego przełączy się na inny serwer.

Kanał sieciowy może być zastosowany w celu realizacji terminali sieciowych wykorzystujących bieżące zmienne systemu Asix.

Innym specyficznym rodzajem protokołu jest obsługiwany wewnętrznie przez Asmena protokół o nazwie **NONE**. Protokół ten nie realizuje fizycznego połączenia ze sterownikiem. Może być zastosowany w celu:

- testowania aplikacji w trybie symulacji,

- umożliwienia wymiany informacji pomiędzy programami systemu Asix za pomocą zmiennych procesowych.

Symulacja zmian wartości zmiennych procesowych korzystających z protokołu NONE może być realizowana w oparciu o specyficzne funkcje przeliczające lub też poprzez odrębny program symulacyjny modyfikujący wartości zmiennych procesowych według żądanego przez projektanta algorytmu. W celu wykorzystania protokołu NONE dla przetestowania aplikacji należy:

- usunąć deklaracje fizycznych kanałów transmisji,
- utworzyć nowe deklaracje kanałów transmisji o nazwach logicznych identycznych z tymi, jakie były użyte w oryginalnych deklaracjach kanałów transmisji. Jako protokół należy zadeklarować protokół o nazwie NONE,
- po zakończeniu testów należy powrócić do oryginalnych deklaracji kanałów transmisji.

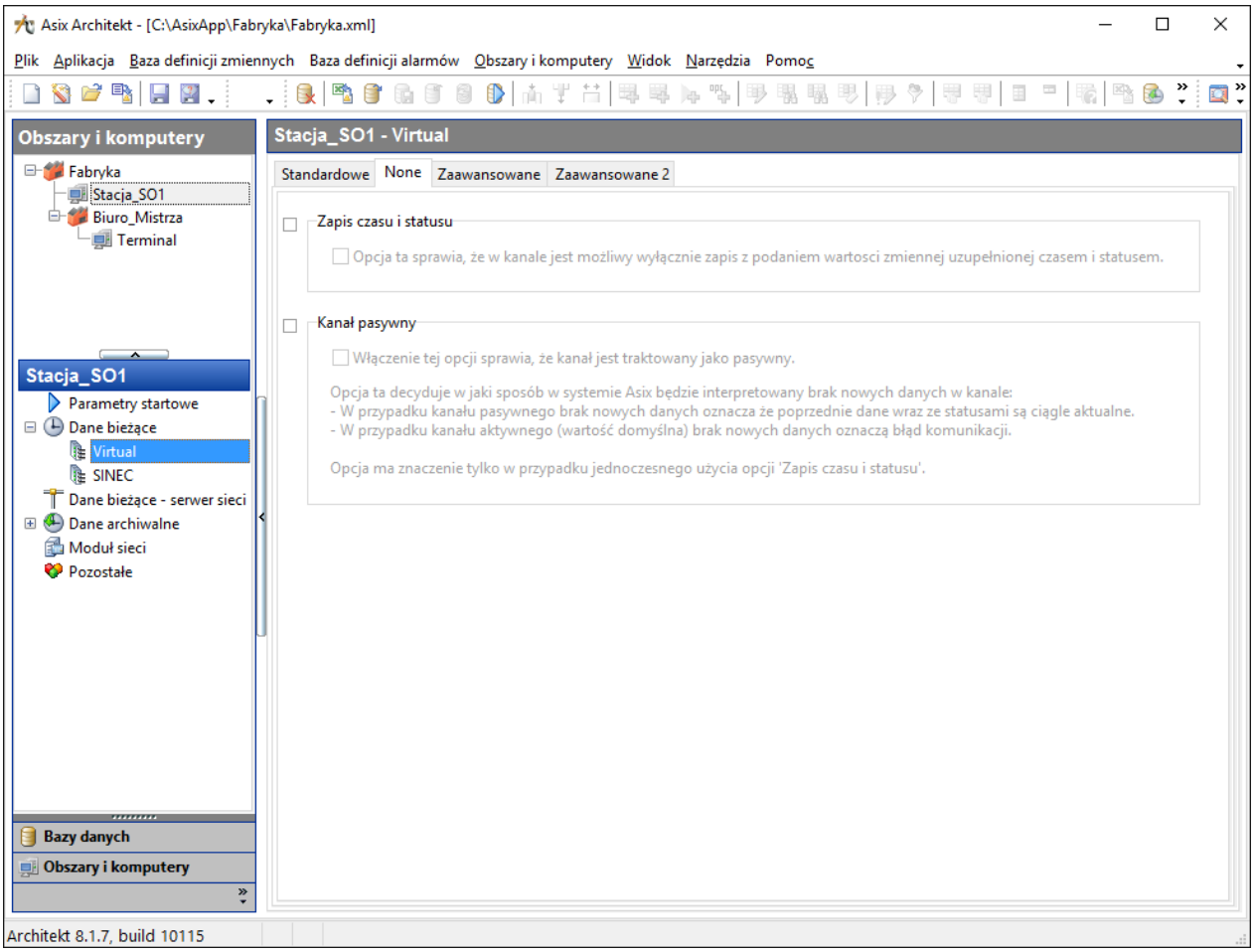
Powyższy schemat postępowania można uzupełnić o dodanie do definicji zmiennych procesowych funkcji przeliczających przeznaczonych do symulacji przebiegów wartości zmiennych. Funkcje te zapewniają samoczynną zmianę wartości zmiennych zgodnie z algorytmem specyficznym dla konkretnej funkcji. W pakiecie systemu Asix znajdują się aktualnie dwie symulacyjne funkcje przeliczające: ON/OFF i PILA.

Wymiana danych poprzez zmienne procesowe kanału typu NONE polega na tym, że jeden z programów systemu Asix wypracowuje wartości zmiennych i wpisuje je do bazy danych Asmena. Pozostałe elementy systemu (program archiwizacji, obiekty wizualizacyjne) mogą pobierać te dane w normalny sposób, nie zdając sobie sprawy z tego, że dane te nie są transmitowane ze sterownika obiektu.

W pozycji opisującej protokół typu NONE nie podaje się żadnych dodatkowych parametrów.

Zmienne tablicowe z kanału NONE są traktowane jako jedna zmienna.

Istnieje możliwość zadeklarowania dla kanałów typu NONE sposobu dostępu do zmiennych polegającego na tym, że zmienne otrzymują wartości w efekcie wykonania operacji zapisu - wymagającej podania nie tylko wartości ale również czasu i statusu. W tym celu należy jako drajwer komunikacyjny zadeklarować dla danego kanału drajwer o nazwie NONE i włączyć opcję *Zapis czasu i statusu.* Rozwiązanie to zostało stworzone w celu umożliwienia importu (np. z wykorzystaniem mechanizmu skryptu) wartości zmiennych z aplikacji zewnętrznych, np. arkusza kalkulacyjnego i przekazania ich do archiwum modułu Aspad. Domyślnie wartości zmiennych kanału NONE pozbawione są stempla czasu i statusu, przez co nie jest możliwa ich archiwizacja - dopiero użycie opcji *Zapis czasu i statusu* pozwala zapisać wartości tych zmiennych w archiwum Aspada.



Rysunek: Deklaracja zapisu czasu i statusu dla kanału None.

W chwili wystartowania program Asmen wymaga, aby drajwery wszystkich protokołów zadeklarowanych w pliku konfiguracyjnym Serwera Danych znajdowały się w kartotece, z której startuje system Asix. Wyjątkiem jest drajwer protokołu symulującego łącze fizyczne (NONE), który jest wbudowany w moduł Asmen. Sposób ładowania poszczególnych drajwerów jest podany przy opisie drajwerów dostępnych w systemie Asix.

1.2.2. Wyłączanie kanałów z obsługi

Aby wyłączyć kanał komunikacyjny z obsługi:

1. dla każdego kanału, który chcemy wyłączać/włączać z obsługi należy utworzyć zmienną kontrolną, której nazwa jest zbudowana wg zasady: 2 podkreślenia + nazwa kanału + 2 podkreślenia, np. dla kanału TEST zmienna kontrolna ma nazwę __TEST__

2. zmienna kontrolna musi być zadeklarowana w kanale NONE i używać funkcji przeliczającej NIC

3. wyłączenie kanału z obsługi wymaga ustawienia wartości zmiennej kontrolnej na 0

4. włączenie kanału do obsługi wymaga ustawienia wartości zmiennej kontrolnej na != 0

5. jeśli kanał jest wyłączony z obsługi, to wszystkie operacje na zmiennych z tego kanału (odświeżanie, odczyt, zapis) są kończone z błędem na poziomie Asmena (bez wykonywania operacji przez drajwer)

6. domyślnie kanał jest włączony do obsługi, a wartość zmiennej kontrolnej jest ustawiana przez Asmena na 1

7. wyłączenie/włączenie kanału jest kwitowane wpisem do panelu operatorskiego

Jeśli kanał ma być wyłączony z obsługi już na starcie aplikacji, to należy skorzystać z opcji wpisanej w Architekcie, w module *Pozostałe,* na zakładce *Opcje wprowadzane bezpośrednio*:

Nazwa sekcji: ASMEN
Nazwa opcji: WYLACZENIE_KANALU
Wartość opcji: *nazwa_ kanału_Asmena*

1.2.3. Deklarowanie zmiennych procesowych

Zaleca się deklarowanie zmiennych procesowych w bazie definicji zmiennych.

Zaleca się przechowywanie definicji zmiennych w bazie typy SQL lub Jet (Access) lub bezpośrednio w arkuszach kalkulacyjnych Excel - szczegóły na ten temat **patrz:** *Serwer Danych > 2. Parametryzacja Serwera Danych > 2.2.1.2.1. Typ i format bazy danych*.

Parametry definiujące zmienne w bazie definicji zmiennych zawarte są w polach o następujących nazwach:

Parametr	**Pole**
Nazwa zmiennej	Nazwa
Opis zmiennej	Opis
Adres zmiennej	Adres
Deklaracja kanału transmisji	Kanal
Rozmiar zmiennej	LiczbaElementow
Częstość odświeżania	OkresProbkowania
Funkcja przeliczająca	FunkcjaPrzeliczajaca
Parametry funkcji przeliczającej:	
Zakres wejściowy od..	ZakresFunkcjiAsmenLo
Zakres wejściowy ..do	ZakresFunkcjiAsmenHi
Zakres wyjściowy od..	ZakresPomiarowyLo
Zakres wyjściowy ..do	ZakresPomiarowyHi

Patrz: Pełna lista atrybutów systemowych bazy definicji zmiennych znajduje się w *Serwer Danych > 2. Parametryzacja Serwera Danych > Załącznik 1.*

Przy deklarowaniu zmiennych procesowych nie określa się ich typu. Typ zmiennej procesowej w postaci surowej (odczytywanej lub zapisywanej do sterownika) wynika z protokołu transmisji oraz adresu symbolicznego. Typ zmiennej procesowej w postaci przetworzonej (przekazywanej przez Asmena do aplikacji lub z aplikacji do Asmena) wynika z zastosowanej funkcji przeliczającej. Zmienne procesowe mogą przyjmować następującą postać:

- liczba 16-bitowa bez znaku (WORD),
- liczba 16-bitowa ze znakiem (INTEGER),
- liczba 32-bitowa bez znaku (DWORD),
- liczba 32-bitowa ze znakiem (LONG),
- liczba zmiennoprzecinkowa (FLOAT).

Moduł Asmen korzysta z następujących parametrów zmiennych:

nazwa_zmiennej - nazwa symboliczna zmiennej procesowej jednoznacznie identyfikująca zmienną dla wszystkich komponentów systemu Asix (długość nazwy nie może przekroczyć 50 znaków, a nazwa nie może zawierać spacji, dwukropków, kropki, wszelkiego typu nawiasów oraz przecinka);

opis_zmiennej - dowolny tekst zawierający np. technologiczny opis zmiennej procesowej (w aktualnej wersji nie może zawierać przecinków);

adres - adres symboliczny, którego forma jest specyficzna dla każdego z drajwerów systemu Asix; jest on używany przez drajwer w celu lokalizacji zmiennej w sterowniku;

kanał - nazwa logiczna kanału transmisji (jedna z nazw podanych w deklaracji kanałów transmisji);

ilość - liczba elementów wchodzących w skład zmiennej procesowej (zmienna może być tablicą); np w przypadku bloków danych db specyfikuje się liczbę słów danych dw, których wartości zostaną w formie tablicy przypisane zmiennej procesowej (dla protokołów sterowników SIMATIC); w typowym przypadku liczba elementów wynosi 1;

częstość - częstość aktualizacji wartości zmiennej procesowej (w sekundach);

funkcja - nazwa funkcji przeliczającej służącej do przeliczenia wartości otrzymanej ze sterownika na wartość przekazywaną do komponentów systemu Asix; w trakcie inicjalizacji zmiennych procesowych sprawdzana jest zgodność typu zmiennej pobieranej ze sterownika (określana przez drajwer) z typem zmiennej wejściowej funkcji przeliczającej; w przypadku rozbieżności, zmienna procesowa nie jest inicjowana; dopuszczalne jest zamienne używanie typów integer, word, dword oraz long;

parametry - lista parametrów funkcji przeliczającej przypisanych danej zmiennej; w przypadku, gdy nie jest podana pełna lista parametrów, to pominiętym parametrom zostaną nadane wartości domyślne, specyficzne dla każdej z funkcji przeliczających;

Adres symboliczny jest używany dla jednoznacznego określenia zmiennej w sterowniku, której wartość zostanie przypisana zmiennej procesowej w systemie Asix. Na podstawie adresu symbolicznego określany jest typ zmiennej oraz jej lokalizacja w pamięci

sterownika. Sposób tworzenia adresu jest specyficzny dla typu kanału, w którym zmienna się znajduje.

1.2.4. Uruchamianie modułu Asmen

Asmen jest ładowany automatycznie w trakcie ładowania Asixa, a parametry jego działania pobierane są z pliku konfiguracyjnego Serwera Danych. Parametryzacja modułu Asmen musi uwzględniać wszystkie zmienne używane w systemie.

1.2.5. Parametry konfiguracyjne modułu Asmen

Parametry Asmena są zadawane w pliku konfiguracyjnym Serwera Danych XML za pośrednictwem programu Architekt.

Patrz: *Serwer Danych > 2. Parametryzacja Serwera Danych > 2.8. Konfiguracja parametrów danych bieżących*.

1.2.6. Funkcje przeliczające

Funkcje przeliczające są używane przez moduł Asmen przy przekształcaniu wartości zmiennych procesowych z postaci surowej na postać rzeczywistą. Każda zmienna posiada indywidualnie deklarowaną funkcję przeliczającą. Niektóre z udostępnionych funkcji przeliczających pozwalają na symulowanie zmian wartości zmiennych według zadanego algorytmu.

System Asix zawiera następujące funkcje przeliczające (nazwy funkcji należy pisać wielkimi literami):

- **ANALOG**, **ANALOG_FP** - przekształcenie liniowe
- **ANALOG12STATUS** - przekształcenie liniowe z uwzględnieniem bitów statusu pomiaru
- **ANALOGWORD_FP** - specjalne przekształcenie liniowe
- **BIT0_Z_N** - sklejanie słowa z zerowych bitów n zmiennych
- **BITN_Z_N** - sklejanie słowa z dowolnych bitów n zmiennych
- **CAMAC6_1**, **CAMAC6_2** - specjalna obsługa interfejsu CAMAC
- **CZAS** - dokonuje konwersji wartości zmiennej procesowej
- **CZAS_SEK** - zamienia przy odczycie wartość zmiennej procesowej
- **CZAS_MSEK** - zamienia przy odczycie wartość zmiennej procesowej
- **CYRK1** - konwersja bit - liczba
- **DATACZAS** - konwersja liczby sekund na łańcuch
- **DATACZAS_MEC** - konwersja czasu licznika MEC na łańcuch
- **DW2FP** - funkcja pozwala na zmianę zmiennych typu DWORD na zmienne typu FLOAT
- **GRADIENT** - obliczanie gradientu
- **ILORAZ** - skalowanie

- **ILORAZ_DW** - skalowanie
- **LICZ** - konwersja z formatu BCD
- **MASKA** - iloczyn logiczny wartości zmiennej i parametru funkcji
- **MAXIMUM_Z_N**, **MINIMUM_Z_N** - ekstremum wartości n zmiennych
- **MNOZNIK_DOUBLE** - przekształcenie liniowe
- **MNOZNIK_KG** - konwersja typów zmiennopozycyjnych
- **MNOZNIK_FP** - konwersja typów zmiennopozycyjnych
- **MNOZNIK_DW** - konwersja typów zmiennopozycyjnych
- **MNOZNIK**, **MNOZNIK_INT**, **MNOZNIK_INT64**, **MNOZNIK_LONG**, **MNOZNIK_UINT64** - przekształcenie liniowe
- **NEGBIT_BYTE**, **NEGBIT**, **NEGBIT_DW** - negacja wybranych bitów odpowiednio w bajcie, słowie i podwójnym słowie
- **NIC**, **NIC_DW**, **NIC_DD**, **NIC_DOUBLE**, **NIC_FP** - przekazanie wartości bez zmiany
- **NIC_BYTE**, **NIC_INT**, **NIC_INT64**, **NIC_LONG** - przekazanie wartości bez zmiany
- **NIC_KG**, **NIC_TEXT** - przekazanie wartości bez zmiany
- **NIC_UINT64** - przekazanie wartości bez zmiany
- **ON/OFF** - symulacja zmian dwustanowych
- **PILA**, **PILA_FP**, **PILA_INT64**, **PILA_DOUBLE** - symulacja przebiegu piłokształtnego
- **PROCENT**, **PROCENT_FP** - przeliczanie na wartość procentową
- **PRZESUN_L** - przesunięcie wartości zmiennej w lewo
- **PRZESUN_P** - przesunięcie wartości zmiennej w prawo
- **SREDNIA** - średnia typu "ruchome okienko"
- **SREDNIA_Z_N** - aktualna wartość średniej arytmetycznej listy zmiennych
- **STALA**, **STALA_FP** - nadanie wartości stałej zmiennej Asmena
- **STATUS** - aktualny status zmiennej
- **SUMA_Z_N** - suma wartości n zmiennych
- **SUWAK**, **SUWAK1**, **SUWAK1_FP** - funkcja pomocnicza dla obiektów wizualizacyjnych klasy SUWAK
- **TAB2DOUBLE** - zwraca double zbudowany na bazie wartości zmiennej tablicowej
- **TABELA** - przekształcenie nieliniowe
- **WEZ_CZAS_DW** - znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest pierwszym parametrem funkcji
- **WEZ_CZAS_FP** - znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest pierwszym parametrem funkcji
- **WEZ_CZAS_LONG** - znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest pierwszym parametrem funkcji
- **WEZ_STCZAS_DW** - status zmiennej jest równy wartości zmiennej, której nazwa jest pierwszym parametrem funkcji; znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji
- **WEZ_STCZAS_FP** - status zmiennej jest równy wartości zmiennej, której nazwa jest pierwszym parametrem funkcji; znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji
- **WEZ_STCZAS_LONG** - status zmiennej jest równy wartości zmiennej, której nazwa jest pierwszym parametrem funkcji; znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji
- **WSREDNIA_Z_N** - aktualna wartość średniej ważonej listy zmiennych.
- **ZAKRES** - konwersja zmiennej procesowej odczytanej ze sterownika.
- **ZCZAS** - wartość i status są kopią wartości i statusu zmiennej, której nazwa jest pierwszym parametrem funkcji, znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji.
- **ZCZAS_DW** - wartość i status są kopią wartości i statusu zmiennej, której nazwa jest pierwszym parametrem funkcji, znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji.
- **ZSTATUS** - wartość zmiennej jest kopią statusu zmiennej, będącej parametrem funkcji'

- **ZSTATUS_CZAS** - wartość jest kopią wartości zmiennej będącej pierwszym parametrem, status jest kopią statusu zmiennej będącej drugim parametrem, znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest trzecim parametrem funkcji

Poniżej opisane zostały wszystkie funkcje przeliczające systemu Asix. Dla każdej funkcji przeliczającej podano:

- typ zmiennej czytanej/pisanej z/do ze sterownika (typ zmiennej PLC);
- typ zmiennej przekazywanej do/z programu użytkownika (typ zmiennej PC);
- listę parametrów (wraz z ich typami oraz wartościami domyślnymi) według kolejności występowania w linii deklaracji zmiennej.

- ***Funkcja ANALOG***

Funkcja ANALOG dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

oraz konwersji wartości zmiennej procesowej przesyłanej do sterownika według równania:

*PLCVal = MinPLCVal + (PCVal - MinPCVal) * k*

Współczynnik *k* jest wyliczany następująco:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

typ zmiennej PLC - INTEGER
typ zmiennej PC - INTEGER

Parametry:

MinPLCVal	- INTEGER	(wartość domyślna - 0)
MaxPLCVal	- INTEGER	(wartość domyślna - 2048)
MinPCVal	- INTEGER	(wartość domyślna - 0)
MaxPCVal	- INTEGER	(wartość domyślna - 2048)

- ***Funkcja ANALOG_FP***

Funkcja ANALOG_FP dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

*PLCVal = MinPLCVal + (PCVal - MinPCVal) * k*

Współczynnik *k* jest wyliczany następująco:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:
MinPLCVal - INTEGER (wartość domyślna 0)
MaxPLCVal - INTEGER (wartość domyślna 2048)
MinPCVal - FLOAT (wartość domyślna 0.0)
MaxPCVal - FLOAT (wartość domyślna 2048.0)

---

1. *Funkcja ANALOG_FP działa jak funkcja NIC_FP w przypadku użycia ich dla zmiennych z kanału NONE.*
2. *Funkcja nie wykonuje operacji zapisu, jeśli została użyta dla zmiennej procesowej z kanału NONE, dla którego NIE zadeklarowano zapisu z czasem i statusem.*

- ***Funkcja ANALOG12STATUS***

Funkcja ANALOG12STATUS dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

*PLCVal = MinPLCVal + (PCVal - MinPCVal) * k*

Współczynnik *k* jest wyliczany następująco:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:
MinPLCVal - INTEGER (wartość domyślna 0)
MaxPLCVal - INTEGER (wartość domyślna 2048)
MinPCVal - FLOAT (wartość domyślna 0.0)
MaxPCVal - FLOAT (wartość domyślna 2048.0)

W parametrach funkcji nie trzeba uwzględniać faktu, że odczytana wartość analogu jest przesunięta o 3 bity w lewo.

Funkcja znajduje zastosowanie do wszystkich urządzeń obsługujących 12-bitowe wartości pomiarów, w których na trzech najmłodszych bitach umieszczane są bity statusu pomiaru
(dane w formacie SIEMENS z diagnostyką wewnątrz słowa statusowego), np. do modułów SIMATIC S5 , BECHOFF, WAGO.

Funkcja działa jak funkcja ANALOG_FP, ale dodatkowo ustawiany jest status stosownie do stanu bitów statusowych czytanych z modułu AI:
a / *bit1 = 0, bit0 = 1 - przekroczenie zakresu górnego;zwracany jest status*

OPC_LIMIT_HIGH (lub AVD_BAD, jeśli Asmen nie jest w trybie OPC);
b/ *bit1 = 1, bit0 = 1 - przekroczenie zakresu dolnego; zwracany jest status OPC_LIMIT_LOW (lub AVD_BAD, jeśli Asmen nie jest w trybie OPC);*
c/ *dla pozostałych kombinacji bitów statusowych zwracany jest status OPC_QUALITY_GOOD (lub AVD_GOOD, jeśli Asmen nie jest w trybie OPC).*

- ***Funkcja ANALOGWORD_FP***

Funkcja ANALOGWORD_FP dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według następujących zasad.
Funkcja ANALOG_FP dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal = MinPCVal + (PLCVal - MinPLCVal) * k + offset*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

*PLCVal = MinPLCVal + (PCVal – MinPCVal - offset) * k*

Współczynnik *k* jest wyliczany następująco:

k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:

MinPLCVal	- INTEGER	(wartość domyślna 0)
MaxPLCVal	- INTEGER	(wartość domyślna 2048)
MinPCVal	- FLOAT	(wartość domyślna 0.0)
MaxPCVal	- FLOAT	(wartość domyślna 2048.0)
offset	- FLOAT	(wartość domyślna 0.0)

Funkcja ANALOGWORD_FP działa jak funkcja NIC_FP w przypadku użycia ich dla zmiennych z kanału NONE.

Funkcja nie wykonuje operacji zapisu, jeśli została użyta dla zmiennej procesowej z kanału NONE, dla którego NIE zadeklarowano zapisu z czasem i statusem (włączony parametr 'Zapis czasu i statusu').

- ***BIT0_Z_N***

BIT0_Z_N konstruuje liczbę 16-bitową bez znaku (WORD), w której wartości poszczególnych bitów są równe wartości najmłodszych bitów (bitu 0) aktualnych wartości zmiennych, wymienionych w liście. Z wartości zmiennej będącej na pierwszym miejscu w liście buduje się bit 0, drugim miejscu listy - bit 1 itd. Liczba parametrów funkcji jest ograniczona do 16.

typ zmiennej PLC - BYTE, WORD, DWORD
typ zmiennej PC - WORD

parametry:
k, to minimalna liczba zmiennych poprawnie odczytanych ze sterownika, przy której uznaje się poprawność wartościowania funkcji; wartość 0 oznacza, że wszystkie zmienne muszą być odczytane poprawnie. Gdy liczba poprawnie odczytanych zmiennych jest mniejsza od zadeklarowanej, to funkcja zwraca wartość przypadkową ze statusem błędu.
Val_1 .. Val_*n* nazwy zmiennych Asmena oddzielone spacją , n<=16

W przypadku zadeklarowania jako jednej ze zmiennych Val_i zmiennej typu ***FLOAT*** *przed wykonaniem funkcji wartość zmiennej jest konwertowana na WORD (funkcja ma sens dla zmiennych binarnych, których wartości drajwer zwraca jako FLOAT).*

- ***BITN_Z_N***

BITN_Z_N stanowi rozszerzenie funkcjonalności funkcji *Bit0_Z_N* poprzez:

1. możliwość wyboru dowolnego bitu wartości zmiennej
2. możliwość forsowania wartości bitu poprzez użycie parametru 0 lub 1 zamiast nazwy zmiennej
3. możliwość zanegowania wartości bitu zmiennej poprzez użycie suffixa NOT

Przykład:

ZM_MASKA, , XX, TEST, 1, 1, BITN_Z_N, 4, ZM_01:3 ZM_02:8:NOT ZM_03:4 0 1 ZM_04:1:NOT 1

Poszczególne bity zmiennej ZM_MASKA mają następujące wartości:

bit 0 - bit 3 zmiennej ZM_01
bit 1 - zanegowana wartość bitu 8 zmiennej ZM_02
bit 2 - bit 4 zmiennej ZM_03
bit 3 - 0
bit 4 - 1
bit 5 - zanegowana wartość bitu 1 zmiennej ZM_04
bit 6 - 1

- ***Funkcja CAMAC6_1***

Funkcja CAMAC6_1 dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według następujących zasad:

dla PLCVall >= 2048

PLCVal = (PLCVal - 2048)/2048

dla PLCVall < 2048

PLCVal = (2048 - PLCVal)/2048

*PCVal = PLCVal * (MaxPCVal - MinPCVal) + MinPCVal*

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:

MinPLCVal	- INTEGER	(wartość domyślna 0)
MaxPLCVal	- INTEGER	(wartość domyślna 4096)
MinPCVal	- FLOAT	(wartość domyślna -75.0)

MaxPCVal - FLOAT (wartość domyślna 300.0)

- ***Funkcja CAMAC6_2***

Funkcja CAMAC6_2 liczy średnią z próbek wartości zmiennej procesowej odczytywanych ze sterownika według następujących zasad:

dla PLCVall >= 2048

PLCVal = (PLCVal - 2048)/2048

dla PLCVall < 2048

PLCVal = (2048 - PLCVal)/2048

*PCVal = PLCVal * (MaxPCVal - MinPCVal) + MinPCVal*

Średnia liczona jest za okres *Okres* (podawany w minutach), a wynik uznawany jest za poprawny, jeśli stopa procentowa poprawnie odczytanych próbek wynosi *Stopa*.

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:

MinPLCVal	- INTEGER	(wartość domyślna 0)
MaxPLCVal	- INTEGER	(wartość domyślna 4096)
MinPCVal	- FLOAT	(wartość domyślna -75.0)
MaxPCVal	- FLOAT	(wartość domyślna 300.0)
Okres	- INTEGER	(wartość domyślna 5)
Stopa	- INTEGER	(wartość domyślna 80)

- ***Funkcja CZAS***

Funkcja CZAS dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika (traktowanej jako licznik czasu w formacie STEP5) na liczbę obrazującą czas w jednostkach określonych przez parametr funkcji CZAS.

Przy zapisie wartości zmiennej procesowej do sterownika dokonywana jest konwersja liczby obrazującej czas w jednostkach użytkownika na licznik czasu w formacie STEP5.
Typy formatu czasu użytkownika podane są poniżej:
typ 0 - czas w 0.01 sek
typ 1 - czas w 0.1 sek
typ 2 - czas w sekundach
typ 3 - czas w minutach
typ 4 - czas w godzinach

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametry:
format czasu - WORD (wartość domyślna 2)

- ***Funkcja CZAS_SEK***

Funkcja CZAS_SEK zamienia przy odczycie wartość zmiennej procesowej typu WORD (traktowanej jako liczba sekund) na łańcuch znaków ASCII o formacie:

mm:ss

gdzie:
mm - minuty
ss - sekundy

Funkcja CZAS_SEK zamienia przy zapisie łańcuch znaków ASCII o formacie:

mm:ss

gdzie:
mm - minuty
ss - sekundy
na liczbę sekund w formacie WORD.

Zmienna procesowa wykorzystująca przedmiotowe funkcje nie może należeć do kanału NONE. Funkcja nie może być stosowana do zmiennych tablicowych.

- ***Funkcja CZAS_MSEK***

Funkcja CZAS_MSEK zamienia przy odczycie wartość zmiennej procesowej typu DWORD (traktowanej jako liczba milisekund) na łańcuch znaków ASCII o formacie:

hh:mm:ss

gdzie:
hh - godziny
mm - minuty
ss - sekundy

Funkcja CZAS_MSEK zamienia przy zapisie łańcuch znaków ASCII o formacie:

hh:mm:ss

gdzie:
hh - godziny
mm - minuty
ss - sekundy
na liczbę milisekund w formacie DWORD.

Zmienna procesowa wykorzystująca przedmiotowe funkcje nie może należeć do kanału NONE. Funkcja nie może być stosowana do zmiennych tablicowych.

- ***Funkcja CYRK1***

Funkcja CYRK1 dokonuje przekształcenia wartości zmiennej procesowej odczytanej ze sterownika na jedną z liczb 16-bitowych bez znaku znajdujących się w tablicy przypisanej

danej zmiennej procesowej. Zawartość oraz rozmiar tablicy określany jest na podstawie deklaracji zmiennej procesowej. Przekształcenie polega na odnalezieniu pozycji pierwszego niezerowego bitu w 16 bitowej liczbie odczytanej ze sterownika i użycia tej pozycji jako indeksu przy odczycie z tablicy. Poszukiwanie niezerowego bitu rozpoczyna się od bitu podanego w deklaracji zmiennej i kontynuuje w stronę malejących rangą bitów. Liczba testowanych bitów podana jest w deklaracji zmiennej. W przypadku, gdy odczytane słowo jest równe 0, to jako wartość zwracany jest ostatni parametr, podany przy deklaracji zmiennej procesowej.

W przypadku pisania do sterownika funkcja CYRK1 przeszukuje tablicę przypisaną zmiennej procesowej w celu odnalezienia w niej wzorca identycznego z wartością przekazaną do funkcji. W przypadku odnalezienia wzorca do sterownika przekazywana jest liczba 16-bitowa składająca się z 1 na pozycji odpowiadającej indeksowi odnalezionego wzorca i zer na pozostałych pozycjach. Wyjątek stanowi wzorzec odnaleziony na najstarszej pozycji tablicy - w takim przypadku do sterownika przekazywane jest 0. W przypadku, gdy wzorzec nie zostanie odnaleziony, funkcja zwraca błąd.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametry:
numer bitu, od którego rozpoczyna się poszukiwania - WORD
liczba przeszukiwanych bitów - WORD
kolejne elementy tablicy wzorców
maks. 16 + 1 (ostatni dla 0) elementów - WORD

- ***Funkcja DATACZAS***

Funkcja DATACZAS zamienia liczbę sekund przekazaną w postaci liczby typu DWORD na łańcuch ASCII o postaci:

DD-MM-YYYYbHH:NN:SS

gdzie:
DD - dwie cyfry dnia
MM - dwie cyfry miesiąca
YYYY - cztery cyfry roku
HH - dwie cyfry godziny
NN - dwie cyfry minut
SS - dwie cyfry sekund
b - spacja
Może być wykorzystywana wyłącznie w przypadku drajwera NONE.
Funkcja nie ma parametrów.

- ***Funkcja DATACZAS_MEC***

Funkcja DATACZAS_MEC zamienia czas licznika MEC podany jako liczbę sekund przekazaną w postaci łańcucha 17 bajtów na łańcuch ASCII o postaci:

YYYY-MM-DD HH:MM

Czas MEC`a przekazywany jest na indeksie 0
Funkcja nie ma parametrów.

- ***Funkcja DATACZAS_MUZ***

Funkcja DATACZAS_MUZ konwertuje znacznik czasu MultiMuza (double) na łańcuch ASCII w formacie:

yyyy-mm-dd hh:nn:ss.zzz

- ***Funkcja DW2FP***

Funkcja pozwala na zamianę zmiennych typu DWORD na zmienną typu FLOAT.

- ***GRADIENT***

GRADIENT zwraca wartość zmiennej wyliczoną wg. wzoru:

$y = \{(t_2 - t_1) / T\} * m$

gdzie:
t_2 - wartość bieżąca wyliczona jako średnia ważona
t_1 - wartość z chwili t_0-T wyliczona jako średnia ważona
T - okres, za który liczony jest gradient (w sekundach)
średnia ważona $t = t_{-1}*(1-k) + t_b*(k)$
k - waga w zakresie (0 – 1)
t_{-1} - wartość t wyliczona w poprzednim cyklu
t_b - wartość w danej chwili
m - mnożnik

typ zmiennej PC - FLOAT

Parametry:
T - WORD
k - FLOAT
m - FLOAT

Wartością źródłową dla funkcji GRADIENT może być zmienna surowa w sterowniku, jak i zmienna systemu Asix zadeklarowana w module Asmen.

PRZYKŁAD

Przykładowa deklaracja zmiennej adresowanej w sterowniku i przeliczanej funkcją gradient

ZMIENNA1,gradient zmiennej w sterowniku,ED100.0,KANAL,1,1,GRADIENT,15,0.1,60

Znaczenie parametrów:
Adres zmiennej - DB 100, DW0
Okres T - 15 sekund
Współczynnik k - 0.1
Przeliczenie m - 60 sekund

ZMIENNA1 - jest wyliczoną szybkością zmian zmiennej o adresie ED100.0 w jednostkach sterownika na minutę.

Zmienna w sterowniku musi być podawana w jednostkach rzeczywistych.

PRZYKŁAD

Przykładowa deklaracja zmiennej systemu Asix przeliczanej funkcją gradient:

ZMIENNA2,gradient zmiennej ,,NONE,1,1,GRADIENT,15,0.1,60,ZMIENNA3

Znaczenie parametrów:
Okres T - 15 sekund
Współczynnik k - 0.1
Przeliczenie m - 60 sekund
Zmienna źródłowa - ZMIENNA3

ZMIENNA2 - jest wyliczoną szybkością zmian zmiennej ZMIENNA3 w jednostkach ZMIENNEJ3 na minutę.

- ***Funkcja ILORAZ***

Funkcja ILORAZ dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

PCVal = PLCVal / k

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

*PLCVal = PCVal * k*

gdzie ***k*** (DZIELNIK) jest parametrem funkcji.

typ zmiennej PLC - WORD
typ zmiennej PC - FLOAT

Parametry:
dzielnik - FLOAT (wartość domyślna 10.0)

- ***Funkcja ILORAZ_DW***

Funkcja ILORAZ_DW dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

PCVal = PLCVal / k

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

*PLCVal = PCVal * k*

gdzie ***k*** (DZIELNIK) jest parametrem funkcji.

typ zmiennej PLC - DWORD

typ zmiennej PC - FLOAT

Parametry:
dzielnik - FLOAT (wartość domyślna 10.0)

- ***Funkcja LICZ***

Funkcja LICZ dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika (traktowanej jako stan licznika w formacie BCD z 3-ma dekadami) na 16-bitową liczbę bez znaku.
Przy zapisie wartości zmiennej procesowej do sterownika dokonywana jest konwersja 16-bitowej liczby bez znaku na liczbę w formacie BCD z 3-ma dekadami.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

nie posiada parametrów

- ***Funkcja MASKA***

Funkcja MASKA przy odczycie wykonuje operację iloczynu logicznego wartości zmiennej odczytanej ze sterownika i parametru funkcji MASKA, podanego w deklaracji zmiennej.
Przy zapisie do sterownika wartość zmiennej nie jest modyfikowana.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametr:
liczba w formacie heksadecymalnym (wartość domyślna FFFF).

- ***MAXIMUM_Z_N***

MAXIMUM_Z_N zwraca największą wartość ze zmiennych, wymienionych w liście jako parametry i przeznaczona jest do użycia ze zmienną zadeklarowaną w kanale typu NONE.
Zmienne wymienione w liście parametrów mogą być różnego typu, wartość funkcji jest konwertowana na typ FLOAT. Nie ogranicza się liczby zmiennych ani ich kolejności w deklaracji zmiennej, do której przypisano funkcję MAXIMUM_Z_N. W przypadku, gdy deklaracja nie mieści się w jednej linii z powodu długiej listy nazw zmiennych, to należy ją rozdzielić na kilka linii, które (poza ostatnią) mają na końcu znak kontynuacji "\".

typ zmiennej PLC - dowolny
typ zmiennej PC - FLOAT

Parametry:
k to minimalna liczba zmiennych poprawnie odczytanych ze sterownika, przy której uznaje się poprawność wartościowania funkcji, wartość 0 oznacza, że wszystkie zmienne muszą być odczytane poprawnie. Gdy liczba poprawnie odczytanych zmiennych jest mniejsza od zadeklarowanej, to funkcja zwraca wartość przypadkową ze statusem błędu.
Val_1,..Val_*n* dowolnego typu, nazwy zmiennych Asmena.

Wartość funkcji jest obliczana następująco:

PCVal=max(Val_1,Val_2,..,Val_n)

- ***MINIMUM_Z_N***

MINIMUM_Z_N zwraca najmniejszą wartość ze zmiennych, wymienionych w liście jako parametry i przeznaczona jest do użycia ze zmienną zadeklarowaną w kanale typu NONE. Zmienne wymienione w liście parametrów mogą być różnego typu, wartość funkcji jest konwertowana na typ FLOAT. Nie ogranicza się liczby zmiennych ani ich kolejności w deklaracji zmiennej, do której przypisano funkcję MINIMUM_Z_N. W przypadku, gdy deklaracja nie mieści się w jednej linii z powodu długiej listy nazw zmiennych, to należy ją rozdzielić na kilka linii, które (poza ostatnią) mają na końcu znak kontynuacji "\".

typ zmiennej PLC - dowolny
typ zmiennej PC - FLOAT

Parametry:
k to minimalna liczba zmiennych poprawnie odczytanych ze sterownika, przy której uznaje się poprawność wartościowania funkcji, wartość 0 oznacza, że wszystkie zmienne muszą być odczytane poprawnie. Gdy liczba poprawnie odczytanych zmiennych jest mniejsza od zadeklarowanej, to funkcja zwraca wartość przypadkową ze statusem błędu.
Val_1,..Val_n dowolnego typu, nazwy zmiennych Asmena.

Wartość funkcji jest obliczana następująco:

PCVal=min(Val_1,Val_2,..,Val_n)

- ***Funkcja MNOZNIK_DOUBLE***

Funkcja MNOZNIK_DOUBLE dokonuje konwersji wartości zmiennej procesowej typu DOUBLE według równania:

*PCVal= A * PLCVal + B*

typ zmiennej PLC - DOUBLE
typ zmiennej PC - DOUBLE

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

- ***Funkcja MNOZNIK_FP***

Funkcja MNOZNIK_FP dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal= A * PLCVal + B*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

PLCVal = (PCVal - B) / A

typ zmiennej PLC - FLOAT
typ zmiennej PC - FLOAT

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)
#ABS - użytkownik dostaje wartość absolutną

Funkcja nie wykonuje operacji zapisu, jeśli została użyta dla zmiennej procesowej z kanału NONE, dla którego NIE zadeklarowano zapisu z czasem i statusem.

- ***Funkcja MNOZNIK_KG***

Funkcja MNOZNIK_KG dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal(IEEE 754) = A * PLCVal(KG) + B*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

PLCVal(KG) = (PCVal(IEEE 754) - B) / A

Przy odczycie zmiennej ze sterownika funkcja przeliczająca **MNOZNIK_KG** dokonuje konwersji z formatu zmiennoprzecinkowego KG na format zmiennoprzecinkowy zgodny z IEEE 754. Przy zapisie dokonywana jest konwersja z formatu zmiennoprzecinkowego zgodnego z IEEE 754 na format zmiennoprzecinkowy KG.

typ zmiennej PLC - FLOAT (KG)
typ zmiennej PC - FLOAT

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

Funkcja nie wykonuje operacji zapisu, jeśli została użyta dla zmiennej procesowej z kanału NONE, dla którego NIE zadeklarowano zapisu z czasem i statusem.

- ***Funkcja MNOZNIK_DW***

Funkcja MNOZNIK_DW dokonuje konwersji wartości zmiennej procesowej typu *dword* (liczba 32-bitowa bez znaku) odczytanej ze sterownika według równania:

*PCVal = A * PLCVal + B*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

PLCVal = (PCVal - B) / A

typ zmiennej PLC - DWORD
typ zmiennej PC - FLOAT

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

- ***Funkcja MNOZNIK***

Funkcja MNOZNIK dokonuje konwersji wartości zmiennej procesowej typu *word* (liczba 16-bitowa bez znaku) odczytanej ze sterownika według równania:

*PCVal = A * PLCVal + B*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

PLCVal = (PCVal - B) / A

typ zmiennej PLC - WORD
typ zmiennej PC - FLOAT

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

- ***Funkcja MNOZNIK_INT***

Funkcja MNOZNIK_INT dokonuje konwersji wartości zmiennej procesowej (liczba 16-bitowa ze znakiem) odczytanej ze sterownika według równania:

*PCVal = A * PLCVal + B*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

PLCVal = (PCVal - B) / A

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

- ***Funkcja MNOZNIK_INT64***

Funkcja MNOZNIK_INT64 dokonuje konwersji wartości zmiennej procesowej typu INT64 według równania:

*PCVal = A * PLCVal + B*

typ zmiennej PLC - INT64
typ zmiennej PC - INT64

Parametry:
A -FLOAT (wartość domyślna 1.0)
B -FLOAT (wartość domyślna 0.0)

- ***Funkcja MNOZNIK_LONG***

Funkcja MNOZNIK_LONG dokonuje konwersji wartości zmiennej procesowej (liczba 32-bitowa ze znakiem) odczytanej ze sterownika według równania:

*PCVal = A * PLCVal + B*

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

PLCVal = (PCVal - B) / A

typ zmiennej PLC - LONG
typ zmiennej PC - FLOAT

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

- ***Funkcja MNOZNIK_UINT64***

Funkcja MNOZNIK_UINT64 dokonuje konwersji wartości zmiennej procesowej typu INT64 bez znaku według równania:

*PCVal = A * PLCVal + B*

typ zmiennej PLC - UINT64
typ zmiennej PC - UINT64

Parametry:
A - FLOAT (wartość domyślna 1.0)
B - FLOAT (wartość domyślna 0.0)

- ***Funkcja NEGBIT_BYTE***

Funkcja NEGBIT_BYTE przekazuje użytkownikowi 8-bitową wartość odczytanej zmiennej procesowej po zanegowaniu bitów zgodnie z maską.

typ zmiennej PLC - BYTE
typ zmiennej PC - BYTE

Parametr:
8-bitowa maska negowanych bitów w postaci heksdecymalnej

- ***Funkcja NEGBIT***

Funkcja NEGBIT przekazuje użytkownikowi 16-bitową wartość odczytanej zmiennej po zanegowaniu bitów zgodnie z maską.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametr:
16-bitowa maska negowanych bitów w postaci heksdecymalnej

- ***Funkcja NEGBIT_DW***

Funkcja NEGBIT_DW przekazuje użytkownikowi 32-bitową wartość odczytanej zmiennej po zanegowaniu bitów zgodnie z maską.

typ zmiennej PLC - DWORD
typ zmiennej PC - DWORD

Parametr:
32-bitowa maska negowanych bitów w postaci heksdecymalnej

- ***Funkcja NIC***

Funkcja NIC przekazuje użytkownikowi 16-bitową wartość zmiennej procesowej w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_BYTE***

Funkcja NIC_BYTE przekazuje użytkownikowi wartość zmiennej procesowej w postaci liczby 8-bitowej w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - BYTE
typ zmiennej PC - BYTE

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_DD***

Funkcja NIC_DD przekazuje użytkownikowi wartość zmiennej procesowej, reprezentowaną w sterowniku przez podwójne słowo, dokonując w nim zamiany miejscami starszego i młodszego słowa.

typ zmiennej PLC - DWORD
typ zmiennej PC - DWORD

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_DOUBLE***

Funkcja NIC_DOUBLE przekazuje użytkownikowi wartość zmiennej procesowej tylu *double* w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - DOUBLE
typ zmiennej PC - DOUBLE

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.
przetwarzanej.

- ***Funkcja NIC_DW***

Funkcja NIC_DW przekazuje użytkownikowi 32-bitową wartość zmiennej procesowej w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - DWORD
typ zmiennej PC - DWORD

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_FP***

Funkcja NIC_FP przekazuje użytkownikowi wartość zmiennej procesowej typu *float* w postaci przekazanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - FLOAT
typ zmiennej PC - FLOAT

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;

- ***Funkcja NIC_INT***

Funkcja NIC_INT przekazuje użytkownikowi wartość zmiennej procesowej w formie liczby 16-bitowej ze znakiem w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - INTEGER
typ zmiennej PC - INTEGER

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_INT64***

Funkcja NIC_INT64 przekazuje użytkownikowi wartość zmiennej procesowej w formie liczby 64-bitowej ze znakiem w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - INTEGER 64
typ zmiennej PC - INTEGER 64

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_LONG***

Funkcja NIC_LONG przekazuje użytkownikowi wartość zmiennej procesowej w formie liczby 32-bitowej ze znakiem w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - LONG
typ zmiennej PC - LONG

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_KG***

Funkcja NIC_KG przekazuje użytkownikowi wartość zmiennej procesowej odczytanej ze sterownika w formacie zmiennoprzecinkowym KG (specyficzny format sterowników firmy SIEMENS) i zamienionej na format zmiennoprzecinkowy zgodny z IEEE 754. Przy zapisie dokonywana jest konwersja z formatu IEEE 754 na format KG.

typ zmiennej PLC - FLOAT (KG)
typ zmiennej PC - FLOAT

Parametry:
MIN - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej;
MAX - liczba typu jak zmienna PC, nie wpływa na przeliczenie wartości, służy do określenia zakresu zmian zmiennej przetwarzanej.

- ***Funkcja NIC_TEXT***

Funkcja NIC_TEXT przekazuje użytkownikowi wartość zmiennej procesowej w postaci łańcucha znaków ASCII zakończonego zerem.

typ zmiennej PLC - BYTE
typ zmiennej PC - BYTE

nie posiada parametrów

- ***Funkcja NIC_UINT64***

Funkcja NIC_UINT64 przekazuje użytkownikowi wartość zmiennej procesowej w formie liczby 64-bitowej bez znaku w postaci odczytanej przez drajwer, nie dokonując żadnych dodatkowych przekształceń.

typ zmiennej PLC - UINT64
typ zmiennej PC - UINT64

- ***Funkcja ON/OFF***

Funkcja ON/OFF została opracowana dla celów testów prowadzonych bez udziału sterownika (przy użyciu kanału typu NONE). Pozwala ona generować przebiegi dwustanowe o zadanych wartościach.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametry:
wartość ON - WORD (wartość domyślna 1)
wartość OFF - WORD (wartość domyślna 0)
wartość początkowa - WORD (wartość domyślna 0)

- ***Funkcja PILA***

Funkcja PILA została opracowana dla celów testów prowadzonych bez udziału sterownika i przy użyciu kanału typu NONE. Pozwala ona generować przebiegi piłokształtne o zadanych wartościach granicznych oraz zadanym kroku zmiany wartości. Zmienna procesowa przyjmuje wartości typu INTEGER.

typ zmiennej PLC - INTEGER
typ zmiennej PC - INTEGER

Parametry:
początkowy kierunek narastania (1 - w górę, 0 - w dół) - INTEGER (wartość domyślna 1)
krok - INTEGER (wartość domyślna 1)
maksimum - INTEGER (wartość domyślna 100)
minimum - INTEGER (wartość domyślna -100)
wartość początkowa - INTEGER (wartość domyślna 0)

- ***Funkcja PILA_DOUBLE***

Funkcja PILA_DOUBLE została opracowana dla celów testów prowadzonych bez udziału sterownika i przy użyciu kanału typu NONE. Pozwala ona generować przebiegi piłokształtne o zadanych wartościach granicznych oraz zadanym kroku zmiany wartości. Zmienna procesowa przyjmuje wartości typu DOUBLE.

typ zmiennej PLC - DOUBLE
typ zmiennej PC - DOUBLE

Parametry:
początkowy kierunek narastania (1 - w górę, 0 - w dół) - DOUBLE (wartość domyślna 1)
krok - DOUBLE (wartość domyślna 1.0)
maksimum - DOUBLE (wartość domyślna 100.0)
minimum - DOUBLE (wartość domyślna -100.0)
wartość początkowa - DOUBLE (wartość domyślna 0.0)

- ***Funkcja PILA_FP***

Funkcja PILA_FP została opracowana dla celów testów prowadzonych bez udziału sterownika i przy użyciu kanału typu NONE. Pozwala ona generować przebiegi piłokształtne o zadanych wartościach granicznych oraz zadanym kroku zmiany wartości. Zmienna procesowa przyjmuje wartości typu FLOAT.

typ zmiennej PLC - FLOAT
typ zmiennej PC - FLOAT

Parametry:
początkowy kierunek narastania (1 - w górę, 0 - w dół) - INTEGER (wartość domyślna 1)
krok - FLOAT (wartość domyślna 1.0)
maksimum - FLOAT (wartość domyślna 100.0)
minimum - FLOAT (wartość domyślna -100.0)
wartość początkowa - FLOAT (wartość domyślna 0.0)

- ***Funkcja PILA_INT64***

Funkcja PILA_INT64 została opracowana dla celów testów prowadzonych bez udziału sterownika i przy użyciu kanału typu NONE. Pozwala ona generować przebiegi piłokształtne o zadanych wartościach granicznych oraz zadanym kroku zmiany wartości. Zmienna procesowa przyjmuje wartości typu INTEGER 64.

typ zmiennej PLC - INTEGER 64
typ zmiennej PC - INTEGER 64

Parametry:
początkowy kierunek narastania (1 - w górę, 0 - w dół) - INTEGER 64 (wartość domyślna 1)
krok - INTEGER 64 (wartość domyślna 1)
maksimum - INTEGER 64 (wartość domyślna 100)
minimum - INTEGER 64 (wartość domyślna -100)
wartość początkowa - INTEGER 64 (wartość domyślna 0)

- ***Funkcja PROCENT***

Funkcja PROCENT dokonuje konwersji wartości zmiennej procesowej typu *INTEGER* odczytanej ze sterownika do postaci typu *FLOAT* według równania:

*PCVal = (PLCVal / k) * 100.0*

oraz konwersji wartości zmiennej procesowej pisanej do sterownika z postaci *FLOAT* na wartość typu *INTEGER* według równania:

*PLCVal = (PCVal / 100.0) * k*

gdzie ***k*** jest parametrem funkcji.

typ zmiennej PLC - INTEGER
typ zmiennej PC - FLOAT

Parametry:
wartość odniesienia - FLOAT (wartość domyślna 2048.0)

- ***Funkcja PROCENT_FP***

Funkcja PROCENT_FP operuje na zmiennej procesowej typu *FLOAT*. Funkcja dokonuje konwersji zmiennoprzecinkowej wartości zmiennej procesowej odczytanej ze sterownika według równania:

*PCVal = (PLCVal / k) * 100.0*

oraz konwersji wartości zmiennoprzecinkowej pisanej do sterownika według równania:

*PLCVal = (PCVal / 100.0) * k*

gdzie ***k*** jest parametrem funkcji.

typ zmiennej PLC - FLOAT
typ zmiennej PC - FLOAT

Parametry:
wartość odniesienia - FLOAT (wartość domyślna 2048.0)

- ***Funkcja PRZESUN_L***

Funkcja PRZESUN_L przy odczycie przesuwa w lewo wartość odczytaną ze sterownika o liczbę bitów określoną przez pierwszy parametr funkcji, a następnie wykonuje operację iloczynu logicznego przesuniętej wartości i drugiego parametru funkcji.
Przy zapisie do sterownika wartość zmiennej nie jest modyfikowana.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametr:
przesunięcie (liczba z zakresu 0 - 16) (wartość domyślna 0)
liczba w formacie heksadecymalnym (wartość domyślna FFFF)

- ***Funkcja PRZESUN_P***

Funkcja PRZESUN_P przy odczycie przesuwa w prawo wartość odczytaną ze sterownika o liczbę bitów określoną przez pierwszy parametr funkcji, a następnie wykonuje operację iloczynu logicznego przesuniętej wartości i drugiego parametru funkcji.
Przy zapisie do sterownika wartość zmiennej nie jest modyfikowana.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametr:
przesunięcie (liczba z zakresu 0 - 16) (wartość domyślna 0)
liczba w formacie heksadecymalnym (wartość domyślna FFFF)

- ***Funkcja SREDNIA***

Funkcja SREDNIA liczy średnią z próbek wartości zmiennej procesowej odczytywanych ze sterownika według następujących zasad:

$$PCVal = MinPCVal + (PLCVal - MinPLCVal) * k$$

oraz konwersji zmiennej procesowej przesyłanej do sterownika według równania:

$$PLCVal = MinPLCVal + (PCVal - MinPCVal) * k$$

Współczynnik *k* jest wyliczany następująco:

$$k = (MaxPCVal - MinPCVal)/(MaxPLCVal - MinPLCVal)$$

Średnia liczona jest za okres *Okres* (podawany w minutach), a wynik uznawany jest za poprawny, jeśli stopa procentowa poprawnie odczytanych próbek wynosi *Stopa*.

Parametrem funkcji może być WYŁĄCZNIE liczba (wyrażona w minutach), która spełnia jeden z poniższych warunków:
dzieli godzinę na równe odcinki czasu,

dzieli dobę na równe odcinki czasu będące wielokrotnością godziny.

Wywołania Aspada muszą być zsynchronizowane z zegarem systemowym, a nie z chwilą uruchomienia **asix**a.

typ zmiennej PLC - dowolny
typ zmiennej PC - FLOAT

Parametry:
MinPLCVal - nie dotyczy
MaxPLCVal - nie dotyczy
MinPCVal - FLOAT (wartość domyślna 0.0)
MaxPCVal - FLOAT (wartość domyślna 2048.0)
okres - INTEGER (wartość domyślna 5)
stopa - INTEGER (wartość domyślna 80)

- ***SREDNIA_Z_N***

SREDNIA_Z_N zwraca aktualną wartość średniej arytmetycznej zmiennych, wymienionych w liście jako parametry i przeznaczona jest do użycia ze zmienną zadeklarowaną w kanale typu NONE. Zmienne wymienione w liście parametrów mogą być różnego typu, wartość funkcji jest konwertowana na typ FLOAT. Nie ogranicza się liczby zmiennych ani ich kolejności w deklaracji zmiennej, do której przypisano funkcję SREDNIA_Z_N. W przypadku, gdy deklaracja nie mieści się w jednej linii z powodu długiej listy nazw zmiennych, to należy ją rozdzielić na kilka linii, które (poza ostatnią) mają na końcu znak kontynuacji "\".

typ zmiennej PLC - dowolny
typ zmiennej PC - FLOAT

Parametry:
k - to minimalna liczba zmiennych poprawnie odczytanych ze sterownika, przy której uznaje się poprawność wartościowania funkcji, wartość 0 oznacza, że wszystkie zmienne muszą być odczytane poprawnie. Gdy liczba poprawnie odczytanych zmiennych jest mniejsza od zadeklarowanej, to funkcja zwraca wartość przypadkową ze statusem błędu.
Val_1,..Val_*n* dowolnego typu, nazwy zmiennych Asmena.

Wartość funkcji jest obliczana następująco:

$PCVal=(Val_1+Val_2+..+Val_n)/n$

- ***Funkcja STALA***

Jeśli parametrem funkcji jest liczba typu INTEGER, to funkcja STALA przypisuje zmiennej procesowej wartość liczby.
Jeśli parametrem funkcji jest nazwa zmiennej procesowej, to funkcja STALA przypisuje zmiennej procesowej aktualną wartość zmiennej-parametru. Typem rzeczywistym zmiennej-parametru musi być INTEGER.
Funkcja służy tylko od odczytu.

typ zmiennej PC - INTEGER

Parametry:
IntegerVal - INTEGER (wartość domyślna 0)
lub
nazwa - STRING

- ***Funkcja STALA_FP***

Jeśli parametrem funkcji jest liczba typu FLOAT, to funkcja STALA_FP przypisuje zmiennej procesowej wartość liczby.
Jeśli parametrem funkcji jest nazwa zmiennej procesowej, to funkcja STALA_FP przypisuje zmiennej procesowej aktualną wartość zmiennej-parametru. Typem rzeczywistym zmiennej-parametru musi być FLOAT.
Funkcja służy tylko od odczytu.

typ zmiennej PC - FLOAT

Parametry:
FloatVal - FLOAT (wartość domyślna 0)
lub
nazwa - STRING

- ***Funkcja STATUS***

STATUS zwraca aktualny status zmiennej będącej parametrem funkcji.
Zmienna, która używa funkcji STATUS, musi być zdefiniowana w kanale NONE lub NETSRV.
Funkcja zezwala wyłącznie na operacje odczytu.

PRZYKŁAD

VAL, zmienna ze sterownika, KANAL_PLC,1,1,NIC
STAT, zmienna pokazująca status zmiennej VAL1,KANAL_NONE,1,1,STATUS,VAL

- ***SUMA_Z_N***

SUMA_Z_N zwraca aktualną wartość sumy zmiennych, wymienionych w liście jako parametry i przeznaczona jest do użycia ze zmienną zadeklarowaną w kanale typu NONE. Zmienne wymienione w liście parametrów mogą być różnego typu, wartość funkcji jest konwertowana na typ FLOAT. Nie ogranicza się liczby zmiennych ani ich kolejności w deklaracji zmiennej, do której przypisano funkcję SUMA_Z_N. W przypadku, gdy deklaracja nie mieści się w jednej linii z powodu długiej listy nazw zmiennych, to należy ją rozdzielić na kilka linii, które (poza ostatnią) mają na końcu znak kontynuacji "\".

typ zmiennej PLC - dowolny
typ zmiennej PC - FLOAT

Parametry:
k - to minimalna liczba zmiennych poprawnie odczytanych ze sterownika, przy której uznaje się poprawność wartościowania funkcji, wartość 0 oznacza, że wszystkie zmienne muszą być odczytane poprawnie. Gdy liczba poprawnie odczytanych zmiennych

jest mniejsza od zadeklarowanej, to funkcja zwraca wartość przypadkową ze statusem błędu.
Val_1,..Val_*n* dowolnego typu, nazwy zmiennych Asmena.

Wartość funkcji jest obliczana następująco:

PCVal=Val_1+Val_2+..+Val_n

- ***Funkcja SUWAK***

Funkcja SUWAK zeruje najstarszy bit przy odczycie wartości zmiennej procesowej ze sterownika, natomiast ustawia 1 na najstarszym bicie przy zapisie wartości zmiennej do sterownika.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

nie posiada parametrów

Funkcja nie wykonuje operacji zapisu, jeśli została użyta dla zmiennej procesowej z kanału NONE, dla którego NIE zadeklarowano zapisu z czasem i statusem.

- ***Funkcja SUWAK1***

Funkcja SUWAK1 działa podobnie do funkcji ANALOG z następującymi różnicami:

- przy odczycie ze sterownika najstarszy bit wartości zmiennej procesowej zostaje wstępnie wyzerowany,
- przed zapisem do sterownika najstarszy bit wartości zmiennej procesowej zostaje ustawiony na 1.

typ zmiennej PLC - WORD
typ zmiennej PC - WORD

Parametry (jak dla funkcji ANALOG):
MinPLCVal - INTEGER (wartość domyślna - 0)
MaxPLCVal - INTEGER (wartość domyślna - 2048)
MinPCVal - INTEGER (wartość domyślna - 0)
MaxPCVal - INTEGER (wartość domyślna - 2048)

Funkcja nie wykonuje operacji zapisu, jeśli została użyta dla zmiennej procesowej z kanału NONE, dla którego NIE zadeklarowano zapisu z czasem i statusem.

- ***Funkcja SUWAK1_FP***

Funkcja SUWAK1_FP działa podobnie do funkcji ANALOG_FP z następującymi różnicami:

- przy odczycie ze sterownika najstarszy bit wartości zmiennej procesowej zostaje wstępnie wyzerowany,

- przed zapisem do sterownika najstarszy bit wartości zmiennej procesowej zostaje ustawiony na 1.

typ zmiennej PLC - WORD
typ zmiennej PC - FLOAT

Parametry (jak dla funkcji ANALOG_FP):
MinPLCVal - INTEGER (wartość domyślna 0)
MaxPLCVal - INTEGER (wartość domyślna 2048)
MinPCVal - FLOAT (wartość domyślna 0.0)
MaxPCVal - FLOAT (wartość domyślna 2048.0)

- ***Funkcja TAB2DOUBLE***

Funkcja TAB2DOUBLE - zwraca double zbudowany na bazie wartości zmiennej tablicowej.

Parametry funkcji (oddzielone przecinkami):
a/ nazwa zmiennej tablicowej (rozmiar zmiennej tablicowej musi wynosić 8 bajtów)
b/ identyfikator typu konwersji (na razie tylko typ 1 – SENTRON)
c/ dzielnik (domyślnie 1)

UWAGA:
- zmienna korzystająca z funkcji TAB2DOUBLE musi być w kanale NONE lub NETWORK,
- funkcja może być wykorzystana wyłącznie do odczytu.

PRZYKŁAD

Przykłady użycia funkcji TAB2DOUBLE:

```
#  tablica 4-ech rejestrów HR jest rzutowana na DOUBLE
#
XXTAB, ,HR802, SENTRON, 4, 1, NIC
XXDBL, ,xxxx,  NONE,   1, 1, TAB2DOUBLE, XXTAB, 1, 1000
#  tablica 2-ch podwójnych rejestrów HRLM jest rzutowana na DOUBLE
#
XXTAB1, ,HRLM834, SENTRON, 2, 1, NIC_DW
XXDBL1, ,xxxx,  NONE,   1, 1, TAB2DOUBLE, XXTAB1, 1, 1000
```

- ***TABELA***

TABELA przekształca wartość zmiennej źródłowej wg funkcji nieliniowej opisanej w tabeli przekazanej w pliku tekstowym, którego każda linia składa się z wartości zmiennej oraz przyporządkowanej jej wartości rzeczywistej. Linie rozpoczynające się od znaku **#** lub **;** są ignorowane (mogą zawierać komentarz). Wartości pośrednie podlegają linearyzacji, w przypadku, gdy wartość źródłowa jest poza zakresem zadeklarowanym w pliku definicji funkcja zwraca błąd pomiaru. Zmienna, której przypisano funkcję TABELA musi być w kanale typu NONE.

Możliwe jest zadeklarowanie w sekcji Asmena ścieżki do kartoteki, w której będą przechowywane wszystkie tabele, używane przez funkcje przeliczające danej aplikacji - służy do tego celu pozycja:

SCIEZKA_TABEL=<*ścieżka*>

Funkcja zwraca wartość:

PCVal=f(PCVal_1)
gdzie:
PCVAL_1 - to nazwa wcześniej zadeklarowanej zmiennej Asmena;
f(x) - funkcja zdefiniowana tabelarycznie w pliku tekstowym.

typ zmiennej PC - FLOAT

Parametry:
PCVAL_1 - FLOAT|LONG|DWORD|WORD|INT16
plik_definicji - tekst

PRZYKŁAD

Przykład deklaracji zmiennej:

AAA, Opis - zmienna AAA zależna od BBB, , NONE, 1, 1, TABELA,BBB,tablica1.txt

Przykład zawartości pliku tablica1.txt definiującego tabelę:

```
10 15.7
20 21
30 26.4
40 32
50 37.1
```

Jako separatora należy użyć spacji lub tabulatora. Dla liczb ułamkowych jako separator dziesiętny musi być użyta kropka.

- ***WEZ_CZAS_DW***

Znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest pierwszym parametrem funkcji. Zakłada się, że:

- wartość surowa zmiennej jest typu DWORD,
- wartość zmiennej będącej pierwszym parametrem funkcji jest typu DWORD i zawiera liczbę sekund od 1.1.1970 lub 1.1.1980,
- typ znacznika czasu określa drugi parametr funkcji (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

LiczbaParametrow = 2

Parametr1 = Nazwa zmiennej, której wartość (musi być typu DWORD) zostanie skonwertowana na znacznik czasu zmiennej
Parametr2 = Typ czasu przekazanego w Parametr1 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***WEZ_CZAS_FP***

Znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest pierwszym parametrem funkcji. Zakłada się, że:

- wartość surowa zmiennej jest typu FLOAT,
- wartość zmiennej będącej pierwszym parametrem funkcji jest typu DWORD i zawiera liczbę sekund od 1.1.1970 lub 1.1.1980,
- typ znacznika czasu określa drugi parametr funkcji (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

LiczbaParametrow = 2

Parametr1 = Nazwa zmiennej, której wartość (musi być typu DWORD) zostanie skonwertowana na znacznik czasu zmiennej
Parametr2 = Typ czasu przekazanego w Parametr1 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***WEZ_CZAS_LONG***

Znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest pierwszym parametrem funkcji. Zakłada się, że:

- wartość surowa zmiennej jest typu LONG,
- wartość zmiennej będącej pierwszym parametrem funkcji jest typu DWORD i zawiera liczbę sekund od 1.1.1970 lub 1.1.1980,
- typ znacznika czasu określa drugi parametr funkcji (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

LiczbaParametrow = 2

Parametr1 = Nazwa zmiennej, której wartość (musi być typu DWORD) zostanie skonwertowana na znacznik czasu zmiennej
Parametr2 = Typ czasu przekazanego w Parametr1 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***WEZ_STCZAS_DW***

Status zmiennej jest równy wartości zmiennej, której nazwa jest pierwszym parametrem funkcji. Znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji.

Zakłada się, że surowa zmienna jest typu DWORD.

LiczbaParametrow = 3
Parametr1 = Nazwa zmiennej, której wartość będzie zwrócona jako status OPC. Zmienna musi być typu BYTE
Parametr2 = Nazwa zmiennej, której wartość zawiera liczbę sekund od 1.1.1970 lub 1.1.1980 (w UTC). Zmienna musi byc typu DWORD
Parametr3 = Typ czasu przekazanego w Parametr2 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***WEZ_STCZAS_FP***

Status zmiennej jest równy wartości zmiennej, której nazwa jest pierwszym parametrem funkcji. Znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji.

Zakłada się, że surowa zmienna jest typu FLOAT.

LiczbaParametrow = 3
Parametr1 = Nazwa zmiennej, której wartość będzie zwrócona jako status OPC (zmienna musi być typu BYTE)
Parametr2 = Nazwa zmiennej, której wartość (musi byc typu DWORD) zawiera liczbę sekund od 1.1.1970 lub 1.1.1980 (w UTC)
Parametr3 = Typ czasu przekazanego w Parametr2 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***WEZ_STCZAS_LONG***

Status zmiennej jest równy wartości zmiennej, której nazwa jest pierwszym parametrem funkcji. Znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji.

Zakłada się, że wartość surowa zmiennej jest typu LONG.

LiczbaParametrow = 3
Parametr1 = Nazwa zmiennej, której wartość będzie zwrócona jako status OPC. Zmienna musi być typu BYTE
Parametr2 = Nazwa zmiennej, której wartość zawiera liczbę sekund od 1.1.1970 lub 1.1.1980 (w UTC). Zmienna musi byc typu DWORD
Parametr3 = Typ czasu przekazanego w Parametr2 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***WSREDNIA_Z_N***

WSREDNIA_Z_N zwraca aktualną wartość średniej ważonej zmiennych, wymienionych w liście jako parametry wraz z ich wagami. Parametry używa się parami: *waga_zmiennej/wartość_zmiennej*. Dodatkowym parametrem poprzedzającym listę wag i zmiennych, jest liczba określająca minimalną ilość par *waga/zmienna*, które muszą być prawidłowo odczytane, aby wynik funkcji miał status poprawny. Zmienne wymienione w liście parametrów mogą być różnego typu. Wartość funkcji jest konwertowana na typ FLOAT. Nie ogranicza się liczby zmiennych ani ich kolejności w deklaracji zmiennej, do której przypisano funkcję WSREDNIA_Z_N. Kolejne zmienne oddzielane są spacjami. W przypadku, gdy deklaracja nie mieści się w jednej linii z powodu długiej listy nazw zmiennych, to należy ją rozdzielić na kilka linii, które (poza ostatnią) mają na końcu znak kontynuacji "\".

typ zmiennej PLC - dowolny
typ zmiennej PC - FLOAT

Parametry:
k - to minimalna liczba par waga/zmienna poprawnie odczytanych ze sterownika, przy której uznaje się poprawność wartościowania funkcji, wartość 0 oznacza, że wszystkie zmienne muszą być odczytane poprawnie. Gdy liczba poprawnie odczytanych zmiennych jest mniejsza od zadeklarowanej, to funkcja zwraca wartość przypadkową ze statusem błędu.
waga_1,wart_1,..wart_*n*,waga_*n* dowolnego typu, nazwy zmiennych Asmena.

Wartość funkcji jest obliczana następująco:

PCVal=(waga_1*wart_1+waga_2*wart_2+..+waga_n*wart_n)/(waga_1+waga_2+..+waga_n)

PRZYKŁAD

Przykład deklaracji zmiennej Asmena:

SREDNIA_WAZONA, I0.2, KANAL1, 1, 1, WSREDNIA_Z_N, 0, WAGA_1 WART_1 WAGA_2 WART_2 WAGA_3 WART_3

- ***ZAKRES***

Funkcja ZAKRES dokonuje konwersji wartości zmiennej procesowej odczytanej ze sterownika wg równania:

PCVal = MinPLCVal + (PLCVal - MinPCVal) * k

oraz konwersji wartości zmiennej procesowej przesyłanej do sterownika wg równania:

PLCVal = MinPCVal + (PCVal - MinPLCVal) / k

gdzie:
k = (MaxPLCVal - MinPLCVal)/(MaxPCVal - MinPCVal)
PLCVal - zmienna w PLC
PCVal - zmienna w PC

typ zmiennej PLC - FLOAT
typ zmiennej PC - FLOAT

Parametry wg kolejności:
wartości min zakresu PC - FLOAT
wartości max zakresu PC - FLOAT
wartości min zakresu PLC - FLOAT
wartości max zakresu PLC - FLOAT

Przy użyciu kanału NONE, typ zmiennej jest ustawiany na FLOAT.

- ***ZCZAS***

Wartość i status są kopią wartości i statusu zmiennej, której nazwa jest pierwszym parametrem funkcji, znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji. Zakłada się, że wartość drugiej zmiennej jest typu DWORD i zawiera liczbę sekund od 1.1.1970 lub 1.1.1980, typ znacznika czasu określa trzeci parametr funkcji (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

LiczbaParametrow = 3

Parametr1 = Nazwa zmiennej, której status i wartość będą zwrócone jako status i wartość
Parametr2 = Nazwa zmiennej (musi byc typu DWORD), której wartość zostanie skonwertowana na znacznik czasu zmiennej
Parametr3 = Typ czasu przekazanego w Parametr2 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***ZCZAS_DW***

Wartość i status są kopią wartości i statusu zmiennej, której nazwa jest pierwszym parametrem funkcji, znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest drugim parametrem funkcji. Zakłada się, że:
- wartość zmiennej określonej przez pierwszy parametr jest typu DWORD
- wartość drugiej zmiennej jest typu DWORD i zawiera liczbę sekund od 1.1.1970 lub 1.1.1980,
- typ znacznika czasu określa trzeci parametr funkcji (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

LiczbaParametrow = 3

Parametr1 = Nazwa zmiennej, której status i wartość będą zwrócone jako status i wartość (zmienna musi być typu DWORD)
Parametr2 = Nazwa zmiennej, której wartość (musi być typu DWORD) zostanie skonwertowana na znacznik czasu zmiennej
Parametr3 = Typ czasu przekazanego w Parametr2 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

- ***ZSTATUS***

Wartość zmiennej jest kopią statusu zmiennej, będącej parametrem funkcji.

LiczbaParametrow = 1

Parametr1 = Nazwa zmiennej, której status będzie zwrócony jako wartość funkcji

- ***ZSTATUS_CZAS***

Wartość jest kopią wartości zmiennej będącej pierwszym parametrem, status jest kopią statusu zmiennej będącej drugim parametrem, znacznik czasu jest tworzony z wartości zmiennej, której nazwa jest trzecim parametrem funkcji. Zakłada się, że wartość trzeciej zmiennej jest typu DWORD i zawiera liczbę sekund od 1.1.1970 lub 1.1.1980, typ znacznika czasu określa czwarty parametr funkcji (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

LiczbaParametrow = 4

Parametr1 = Nazwa zmiennej, której wartość będzie zwrócona jako wartość zmiennej
Parametr2 = Nazwa zmiennej, której status będzie zwrócony jako status zmiennej
Parametr3 = Nazwa zmiennej (musi byc typu DWORD), której wartość zostanie skonwertowana na znacznik czasu zmiennej
Parametr4 = Typ czasu przekazanego w Parametr3 (0 - sekundy od 1.1.1970, 1 - sekundy od 1.1.1980)

1.2.7. Diagnostyka poprawnego formatu liczb odczytywanych ze sterowników

Moduł Asmen sprawdza, czy odczytywane ze sterowników liczby FLOAT są zgodne z formatem IEEE 754.

Jeśli zostanie znaleziona liczba o wartości NAN lub INF, to zmiennej (przypisanej do tej liczby) zostaje nadana wartość 0 oraz status błędu AVD_BAD. Równolegle wyprowadzany jest komunikat w oknie panelu kontrolnego zawierający typ błędu (NAN lub INF) oraz nazwę zmiennej o złym formacie. O ile został zadeklarowany plik logu dla danej aplikacji, wówczas powyższy komunikat zostanie tam zapisany.

1.2.8. Drajwery komunikacyjne

W skład systemu Asix wchodzą drajwery obsługujące następujące typy połączeń ze sterownikami obiektu przemysłowego.

Zestaw dostępnych protokołów jest systematycznie powiększany. Firma ASKOM podejmuje się, na życzenie klienta, opracowania dowolnego protokołu transmisji na zasadach określonych w cenniku systemu Asix.

Szczegółowe omówienie drajwerów komunikacyjnych znajduje się w odrębnym podręczniku *"Drajwery komunikacyjne"*.

Drajwer		Protokół
ADAM	-	protokół dla sterowników ADAM firmy ADVANTECH
AGGREGATE	-	drajwer Aggregate pozwala definiować zmienne, których wartości powstają w wyniku przeliczania wartości innych zmiennych systemu Asix
AirPointer	-	komunikacja odbywa się przy pomocy protokołu HTTP przy wykorzystaniu stron info.php i download.php serwera webowego stacji monitoringu
AK	-	protokół AK dla analizatorów MLT2 f-my Emerson
AM_SA85	-	protokół do łączności z siecią Modbus Plus firmy Schneider Automation
Areva	-	protokół do komunikacji z urządzeniami MiCOM
AS511	-	protokół wykorzystujący interfejs programatora sterowników serii SIMATIC firmy SIEMENS
AS512	-	protokół procesorów komunikacyjnych CP524/525 sterowników serii SIMATIC firmy SIEMENS
AS512S7	-	protokół procesora komunikacyjnego CP340 sterowników SIMATIC S7 firmy SIEMENS
BACnetIP		protokół BACnet/ IP
BASKI	-	drajwer jest przeznaczony do importu danych z systemu imisji BASKI zainstalowanego w El. Bełchatów do systemu Asix
BAZA	-	drajwer pozwala na import danych z baz danych do systemu Asix
BUFOR	-	uniwersalny protokół wymieniający informacje z programem użytkownika poprzez wspólną pamięć
CALEC	-	drajwer Calec służy do pobierania bieżących wartości zmiennych z urządzenia CALEC MCP firmy Aquametro zgodnie

z protokołem opisanym w dokumencie "MCP Datenauslesung mit dem lowlevel Protokoll"

CAN_AC_PCI - protokół wymiany danych pomiędzy sterownikami SELECONTROL MAS firmy Selectron Lyss AG oraz komputerem systemu Asix

CANOPEN - protokół sieci CANOPEN sterowników SELECTRON MAS firmy Selectron Lyss AG

CipAB - protokół sterownikówserii Logix5000 firmy Allen-Bradley

COMLI - protokół (COMunication Link) do komunikacji ze sterownikami typu SattCon, AC 800C, AC 800M, AC 250 firmy ABB. Wymiana danych poprzez łącze szeregowe RS-232 lub RS-485

CPIII - protokół do komunikacji z panelami kontrolnymi CP-III/E, wykorzystywanymi do sterowania sprężarkami firmy MYCOM (MAYEKAWA)

CZAZ - protokół do komunikacji z urządzeniami CZAZ-U oraz CZAZ-UM

DATAPAF - protokół do łączności z licznikiem energii DataPAF

DDE - drajwer DDE służy do definiowania kanału modułu Asmen, odwołującego się do zmiennych udostępnianych przez drajwer sterownika przemysłowego zaimplementowanego w postaci serwera DDE (zwanego dalej krótko serwerem DDE)

Diva - protokół kamer przemysłowych DIVA firmy VDG Security B.V.

Dlms - protokół liczników energii elektrycznej rodziny ZxD/ZxG/ZxQ firmy Landys&Gyr

DlmsTcpip - protokół DLMS; do komunikacji jest wykorzystywany Ethernet

DMS285 - protokół komputerów emisji D-MS285

DMS500 - protokół komputerów emisji D-MS500 (poprzednia nazwa - DURAG)

DNP3 - protokół stosowany w elektroenergetycznych systemach sterowania i nadzoru

DP - protokół urządzeń zgodnych z PROFIBUS DP przy wykorzystaniu karty PROFIboard

DP5412 - protokół urządzeń zgodnych z PROFIBUS DP przy wykorzystaniu kart firmy Siemens

DSC - protokół do wymiany danych pomiędzy komputerem systemu Asix i sterownikiem DSC 2000

DXF351 - protokół do komunikacji z urządzeniami Compart DXF351 firmy Endress+Hauser

Ecl - protokół do komunikacji z regulatorami ECL Comfort 210/310 produkowanymi przez firmę Danfoss

E2tangoTcpip - protokół sterowników polowych e^{2}TANGO produkowanych przez firmę Elektrometal Energetyka

EcoMUZ - protokół urządzeń zabezpieczających ecoMUZ firmy JM Tronik

EcoMUZ2 - protokół urządzeń zabezpieczających ecoMUZ-2 firmy JM Tronik

EQABP - protokół liczników energii elektrycznej EQABP firmy POZYTON

EQM - protokół liczników energii elektrycznej typu EQM firmy POZYTON w Częstochowie

Esser - protokół centrali p-poż Esser 8008 firmy Honeywell

FESTO - protokół wykorzystujący interfejs diagnostyczny dla sterowników firmy FESTO

FILE2ASIX - drajwer FILE2ASIX jest przeznaczony do importu danych do systemu Asix z plików tekstowych mających określoną strukturę

FP1001 - protokół monitorów przepływu wody i pary firmy METRONIC Kraków

GFCAN - protokół sieci CAN przy użyciu karty komunikacyjnej firmy Garz & Fricke Industrieautomation GmbH

Global - drajwer CtGlobal służy do wymiany danych pomiędzy aplikacją systemu Asix a tzw. plikiem wymiany, będącym kontenerem na aktualne parametry zmiennych drajwera (nazwa, status, wartość, znacznik czasu). W aplikacji składającej się z wielu komputerów zawartość pliku wymiany może być synchronizowana pomiędzy poszczególnymi komputerami. W ten sposób można propagować na wszystkie komputery zmiany wartości zmiennych dokonywane na jednym z nich

IEC61850 - protokół IEC61850

K3N - protokół wymiany danych pomiędzy sterownikami serii K3N firmy OMRON oraz komputerem systemu Asix

K-BUS - protokół K-BUS jest wykorzystywany do wymiany danych pomiędzy regulatorami kotłów serii Dekamatic f-my VIESSMANN dołączonymi do koncentratora Dekatel-G (lub Vitocom 200) oraz komputerem systemu Asix

Lb480 - protokół koncentratorów LB-480 firmy LAB_EL Elektronika Laboratoryjna

LG - protokół sterowników LG Master-K i Glofa GM

Logo - protokół do wymiany danych pomiędzy systemem Asix i sterownikiem Logo OBA5 firmy SIEMENS za pomocą łącza programatora przedmiotowego sterownika

LUMBUS - protokół wymiany danych pomiędzy regulatorami RG72 produkowanymi przez Lubuskie Zakłady Aparatów Elektrycznych "LUMEL" w Zielonej Górze oraz komputerem systemu Asix

LZQM - wymiana danych z licznikami energii elektrycznej typu LZQM, produkowanymi przez Zakład Elektronicznych Urządzeń Pomiarowych POZYTON sp. z o.o. w Częstochowie. Komunikacja odbywa się przy pomocy łączy szeregowych w standardzie CLO

M200 - drajwer komputera przepływu Spirax Sarco M200

MACMAT - protokół GAZ-MODEM korektorów przepływu gazu MACMAT

Max1000 - komunikacja w oparciu o tzw. 'protokół sieciowy' z systemem MAX 1000 firmy ULTRAK zarządzającym pracą kamer. Drajwer realizuje wyłącznie funkcje przełączania kamer oraz synchronizacji czasu (drajwer jest źródłem czasu)

M-BUS - podzbiór standardowego protokołu do odczytu danych z urządzeń pomiarowych, wykorzystywany między innymi przez mierniki ciepła MULTICAL firmy KAMSTRUP A/S

MEC - protokół wymiany danych pomiędzy systemem Asix i licznikami ciepła MEC07 i MEC08 wyprodukowanymi przez Instytut Techniki Cieplnej w Łodzi. Wymiana danych odbywa się przy pomocy łączy szeregowych w standardzie RS-232.

MegaMuz - protokół do wymiany danych pomiędzy systemem Asix i mikroprocesorowymi urządzeniami zabezpieczającymi typu megaMuz, produkowanymi przez firmę JM-Tronik w Warszawie. Do komunikacji jest wykorzystywany interfejs RS-485 zabezpieczeń.

MegaMuz2 - protokół do wymiany danych pomiędzy systemem Asix i mikroprocesorowymi urządzeniami zabezpieczającymi typu megaMuz, produkowanymi przez firmę JM-Tronik w Warszawie. Komunikacja jest realizowana w trybie MODBUS RTU przy pomocy łącza szeregowego.

MegaMuz_TCPIP - protokół do wymiany danych pomiędzy systemem Asix i mikroprocesorowymi urządzeniami zabezpieczającymi typu megaMuz, produkowanymi przez firmę JM-Tronik w Warszawie. Do komunikacji jest wykorzystywany Ethernet.

MegaMuz2_TCPIP - protokół do wymiany danych pomiędzy systemem Asix i mikroprocesorowymi urządzeniami zabezpieczającymi typu megaMuz2, produkowanymi przez firmę JM-Tronik w Warszawie. Do komunikacji jest wykorzystywany Ethernet.

MELSECA - protokół procesora komunikacyjnego A1SJ71C24-R2 sterowników serii MELSEC-A

MEVAS - protokół wymiany danych pomiędzy komputerem emisyjnym MEVAS produkowanymi przez Lubuskie Zakłady Aparatów Elektrycznych "LUMEL" w Zielonej Górze oraz komputerem systemu Asix

MicroSmart - protokół wykorzystywany do wymiany danych ze sterownikami MicroSmart firmy IDEC

MODBUS - podzbiór standardowego protokołu komunikacyjnego wykorzystywany między innymi przez sterowniki Modicon firmy AEG oraz sterowniki firmy GE Fanuc

MODBUS_TCPIP - protokół wymiany danych pomiędzy systemem Asix i innymi komputerami/urządzeniami za pomocą protokołu MODBUS, zrealizowanego w oparciu o sieć Ethernet z protokołem TCP/IP

MODBUSSLV - protokół MODBUS, w którym Asix działa jako SLAVE

MPI - protokół interfejsu MPI sterowników SIMATIC S7 firmy SIEMENS; łącze szeregowe

MPS - protokół łącza szeregowego mierników parametrów sieci energetycznej MPS firmy OBR Metrologii Elektrycznej w Zielonej Górze

MSP1X - protokół MSP1X jest wykorzystywany do wymiany danych pomiędzy sterownikami MSP1X f-my ELMONTEX oraz komputerem systemu Asix.

Muel - protokół do wymiany danych pomiędzy systemem Asix i komputerem technicznym systemu MUEL

MultiMuz - protokół służy do wymiany danych pomiędzy systemem Asix i mikroprocesorowymi urządzeniami zabezpieczającymi typu MultiMUZ, produkowanymi przez firmę JM-Tronik w Warszawie

MultiMuz_tcpip - protokół do komunikacji z urządzeniami zabezpieczającymi typu MultiMUZ, produkowanymi przez firmę JM-Tronik w Warszawie

MultiMuz3 - protokół do komunikacji z mikroprocesorowymi urządzeniami zabezpieczającymi typu MultiMUZ3, produkowanymi przez firmę JM-Tronik w Warszawie

MultiMuz3_TCPIP - protokół do komunikacji z mikroprocesorowymi urządzeniami zabezpieczającymi typu MultiMUZ3, produkowanymi przez firmę JM-Tronik w Warszawie

MUPASZ - protokół kanału pustego (wirtualnego)

Mupasz710_RS - protokół do komunikacji z mikroprocesorowymi urządzeniami zabezpieczającymi typu Mupasz 710, produkowanymi przez Instytut Tele- i Radiotechniczny w Warszawie

MupaszRtu - protokół do wymiany danych pomiędzy systemem Asix i urządzeniami Mupasz2001G, Mupasz07 oraz Mupasz Compact G01 - produkowanymi przez ITR Warszawa.

MupaszRtu_TCPIP - protokół do wymiany danych pomiędzy systemem ASIX i mikroprocesorowymi urządzeniami zabezpieczającymi typu Mupasz 710, produkowanymi przez Instytut Tele- i

Radiotechniczny w Warszawie. Komunikacja jest realizowana w trybie MODBUS RTU na łączu Ethernet poprzez port 502.

Mus04 - protokół do wymiany danych z mikroprocesorowymi urządzeniami sterującymi MUS-04 firmy ELEKTROMETAL S.A. w Cieszynie

MUZ - protokół do wymiany danych pomiędzy Mikroprocesorowymi Urządzeniami Zabezpieczającymi MUZ typu MUZ-RO

NCP - protokół do wymiany danych pomiędzy systemem Asix i sterownikami serii MN firmy Invensys (dawniej Satchwell)

NetLink - protokół MPI/Profibus do komunikacji z urządzeniami SIMATIC S7 z wykorzystaniem modułu NetLink Lite SYSTEME HELMHOLZ

NetLinkPro - protokół do komunikacji ze sterownikami S7 przy pomocy gateway'a NetLinkPRO (gateway Ethernet <-> MPI/Profibus). Producentem NetLinkPro jest SYSTEME HELMHOLZ

NONE - protokół NONE; nie realizuje fizycznego połączenia ze sterownikiem. Może być zastosowany w celu:

- testowania aplikacji w trybie symulacji,
- umożliwienia wymiany informacji pomiędzy programami systemu Asix za pomocą zmiennych procesowych

NordicRF - wymiana danych z czytnikiem kodów kreskowych Nordic ID RF 601. Komunikacja odbywa się przy wykorzystaniu łącza szeregowego RS-232 stacji bazowej

OMRON - do wymiany danych ze sterownikami OMRON

OmronTcpip - protokół FINS

OPC - protokół do wymiany danych między systemem Asix a dowolnym sterownikiem przemysłowym lub programem klasy SCADA, dla którego dostępny jest serwer danych zgodny ze specyfikacją OPC 1.0 lub OPC 2.05

PA5 - drajwer protokołu liczników przepływu PA5

PD21 - protokół do komunikacji z koncentratorem PD21 produkowanym przez LUMEL S.A. Zielona Góra

Pmc4000 - służy do wymiany danych pomiędzy systemem Asix i centralą ppoż POLON 4800 wg protokołu PMC-4000

PPI - protokół dla sterowników SIEMENS S7 serii 200

Protherm300 - driver do wymiany danych pomiędzy systemem Asix i sterownikiem Protherm 300 DIFF firmy Process-Electronic GmbH

PROTRONICPS - PROTRONIC PS firmy Hartmann & Braun

S700 - protokół S700 jest wykorzystywany do wymiany danych pomiędzy analizatorami gazu serii S700 firmy Maihak oraz komputerem systemu Asix.

S7_TCPIP - protokół wykorzystywany do wymiany danych ze sterownikami SIMATIC serii S7 poprzez łącze Ethernet przy wykorzystaniu standardowej karty sieciowej komputera

SAPIS7 - protokół sterowników SIMATIC S7 poprzez łącze MPI lub poprzez procesor komunikacyjny magistrali PROFIBUS (implementacja funkcji S7)

S-BUS - protokół S-BUS jest wykorzystywany do wymiany danych pomiędzy sterownikami PCD firmy SAIA Burgess Electronics oraz komputerem systemu Asix

SbusTcpip - do wymiany danych pomiędzy systemem Asix i rodziną sterowników PCD SAIA-Burgess przy wykorzystaniu protokołu Ethernet S-Bus

SINECH1	-	protokół procesora komunikacyjnego CP1430 sterowników SIMATIC S5 (Ethernet)
SINECL2	-	protokół procesorów komunikacyjnych CP5430 sterowników serii SIMATIC firmy SIEMENS
Si400	-	protokół przeznaczony do wymiany danych pomiędzy systemem Asix i centralą alarmową Sintony SI 400 firmy SIEMENS
SNG	-	protokół do komunikacji z urządzeniami instalacji SNG firmy Synergia Tech z Warszawy
SNMP	-	drajwer umożliwia odczyt i zapis wartości obiektów protokołu SNMPv1 i SNMPv2c - zarządzania różnymi elementami sieci telekomunikacyjnych, takimi jak routery, przełączniki, komputery czy centrale telefoniczne. Drajwer realizuje swoje funkcje za pomocą SNMP Management API
SNPX	-	służy do wymiany danych pomiędzy systemem Asix i jednostkami centralnymi sterowników serii 90-30 oraz modułami CMM i PCM serii 90 firmy GE Fanuc
Sp100H	-	protokół sterowników SP-100H centrali deszczowej Aqua Center Industrial 100 H 14-60 firmy MARLEy Polska Sp. z o.o.
SPA	-	protokół wykorzystywany do komunikacji z urządzeniami podłączonymi do magistrali SPA firmy ABB
Srio	-	wymiana danych z koncentratorem SRIO 500M (prod. ABB) komunikującym system nadrzędny z urządzeniami podłączonymi do magistrali SPA, np. zabezpieczeniami przekaźnikowymi i urządzeniami sygnalizacyjnymi. Komunikacja z koncentratorem jest realizowana przy pomocy łączy szeregowych w standardzie RS232, warstwa transportu danych - w oparciu o protokół ANSI X3.28 w trybie full-duplex z sumą kontrolną BCC
SRTP	-	drajwer jest przeznaczony do wymiany danych pomiędzy systemem Asix i sterownikami firmy GE Fanuc Automation serii VersaMax Nano/Micro przy wykorzystaniu konwertera IC200SET001 oraz sterownikami serii WersaMax i Series 90, przy wykorzystaniu modułu komunikacyjnego IC693CMM321, po sieci Ethernet z protokołem TCP/IP.
TALAS	-	protokół komputerów emisji TALAS
TwinCAT	-	przeznaczony do wymiany danych pomiędzy systemem Asix i systemem TwinCAT firmy Beckhoff Industrie Elektronik
TwinCATTcpip		przeznaczony do wymiany danych pomiędzy systemem Asix i systemem TwinCAT firmy Beckhoff Industrie Elektronik (korzysta z interfejsu ADS/AMS over TCPIP (bez użycia bibliotek Beckhoff'a)
Wago	-	protokół do wymiany danych ze sterownikami Wago
Vantage	-	protokół stacji pogodowych rodziny Vantage Pro produkowanych przez Davis Instruments Corp. USA
ZDARZENIE ZMIENNA	-	drajwer do generowania zmiennych procesowych typu WORD (słowo 16 bitowe) na podstawie bieżących wartości zdarzeń alarmowych w systemie Asix
ZxD400	-	drajwer protokołu liczników energii serii ZxD400 f-my Landys & Gyr

1.2.9. Wymiana danych poprzez lokalną sieć komputerową

Lokalna sieć komputerowa może być wykorzystana do stworzenia takiej konfiguracji systemu Asix, w której tylko część komputerów (*serwery danych*) ma bezpośrednie połączenie ze sterownikami, natomiast pozostałe komputery (*stacje robocze*) korzystają z informacji udostępnianych przez serwery danych za pośrednictwem lokalnej sieci komputerowej.

Kanały transmisji systemu Asix służące do pozyskiwania danych poprzez lokalną sieć komputerową noszą nazwę *kanału sieciowego*.

Wymiana danych w lokalnej sieci komputerowej pomiędzy stacjami systemu Asix jest realizowana przez moduł NETSRV. Moduł ten jest ładowany automatycznie przy instalacji Asixa.

Konfiguracji sieci dokonuje się za pomocą programu Architekt w pliku konfiguracyjnym Serwera Danych.

Patrz: *Serwer Danych > Parametryzacja Serwera Danych > 2.9. Konfiguracja parametrów danych bieżących dotyczących serwera sieci*.

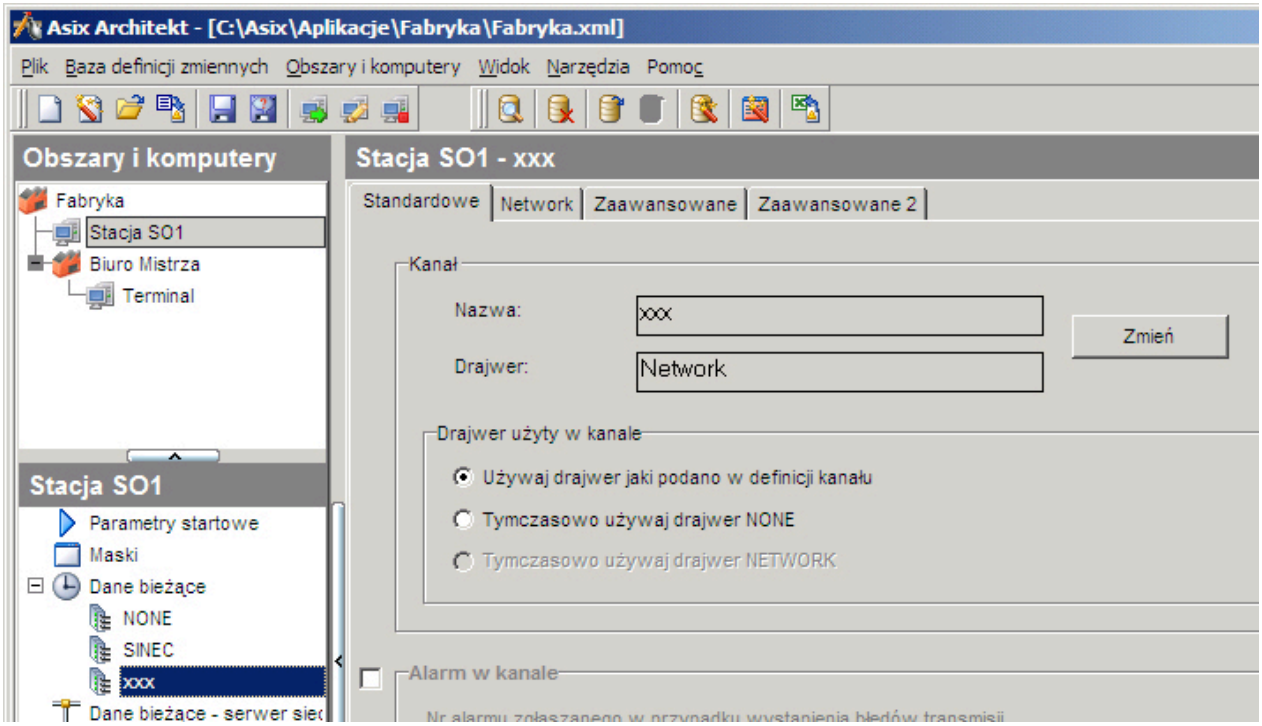
1.2.9.1. Deklaracja kanału sieciowego

Parametry służące do deklaracji kanałów sieciowych obejmują:

Nazwa - nazwa logiczna kanału sieciowego,
NETWORK - nazwa drajwera,
Sieciowa nazwa serwera - nazwa komputera, który może pełnić rolę serwera danych (opcja).

program Architekt > *Obszary i komputery* > *Dane bieżące* > dodana pozycja kanału dla drajwera NETWORK > zakładka *Standardowe* > Parametry:

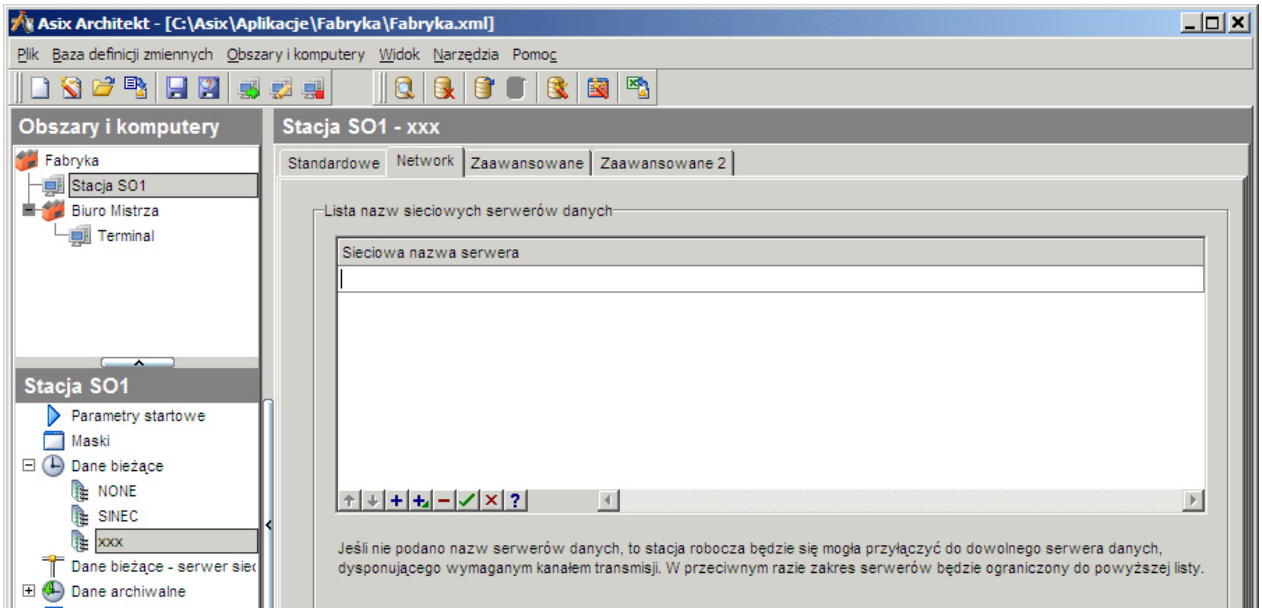
Nazwa
Drajwer: NETWORK



Rysunek: Deklaracja kanału NETWORK.

Architekt > *Obszary i komputery* > *Dane bieżące* > dodana pozycja kanału dla drajwera NETWORK > zakładka *Network*> Parametry:

Sieciowa nazwa serwera



Rysunek: Deklaracja komputerów pełniących rolę serwerów danych.

Deklaracja oznacza, że wartości zmiennych procesowych zadeklarowanych w kanale sieciowym *nazwa_logiczna*, będą pochodziły z serwera danych, na którym zadeklarowano kanał transmisji mający identyczną nazwę logiczną, jak przedmiotowy kanał sieciowy. Przedmiotowy kanał transmisji serwera danych musi spełniać jeden z poniższych warunków:
- obsługuje bezpośrednie połączenie ze sterownikiem,
- jest kanałem typu NONE,
- jest kanałem sieciowym komputera działającego w trybie POMOST *(***Patrz:** *Serwer Danych > Obsługa danych procesowych Serwera Danych > Wymiana danych poprzez lokalną sieć komputerową > 1.2.9.3. Praca stacji roboczej w trybie POMOST)*.

Jeśli w deklaracji kanału sieciowego nie podano nazw serwerów danych, to stacja robocza będzie się mogła przyłączyć do dowolnego serwera danych, dysponującego wymaganym kanałem transmisji. W przeciwnym razie zakres serwerów będzie ograniczony do tych komputerów, których nazwy umieszczono w deklaracji kanału sieciowego.

W nazwach komputerów sieciowych **nie są rozróżniane** duże i małe litery.

1.2.9.2 Deklaracja zmiennych procesowych w kanałach sieciowych

Aby stacja robocza mogła uzyskać zmienną procesową z serwera danych, muszą być spełnione następujące warunki:

- zmienna procesowa używana na stacji roboczej musi być zadeklarowana na serwerze danych,
- deklaracja zmiennej procesowej na serwerze danych i deklaracja zmiennej procesowej na stacji roboczej muszą być identyczne.

Jeśli jeden z tych warunków nie będzie spełniony, to na etapie logowania się do serwera danych na panelu operatorskim stacji roboczej zostanie wyprowadzony komunikat o błędnie zadeklarowanej zmiennej procesowej.

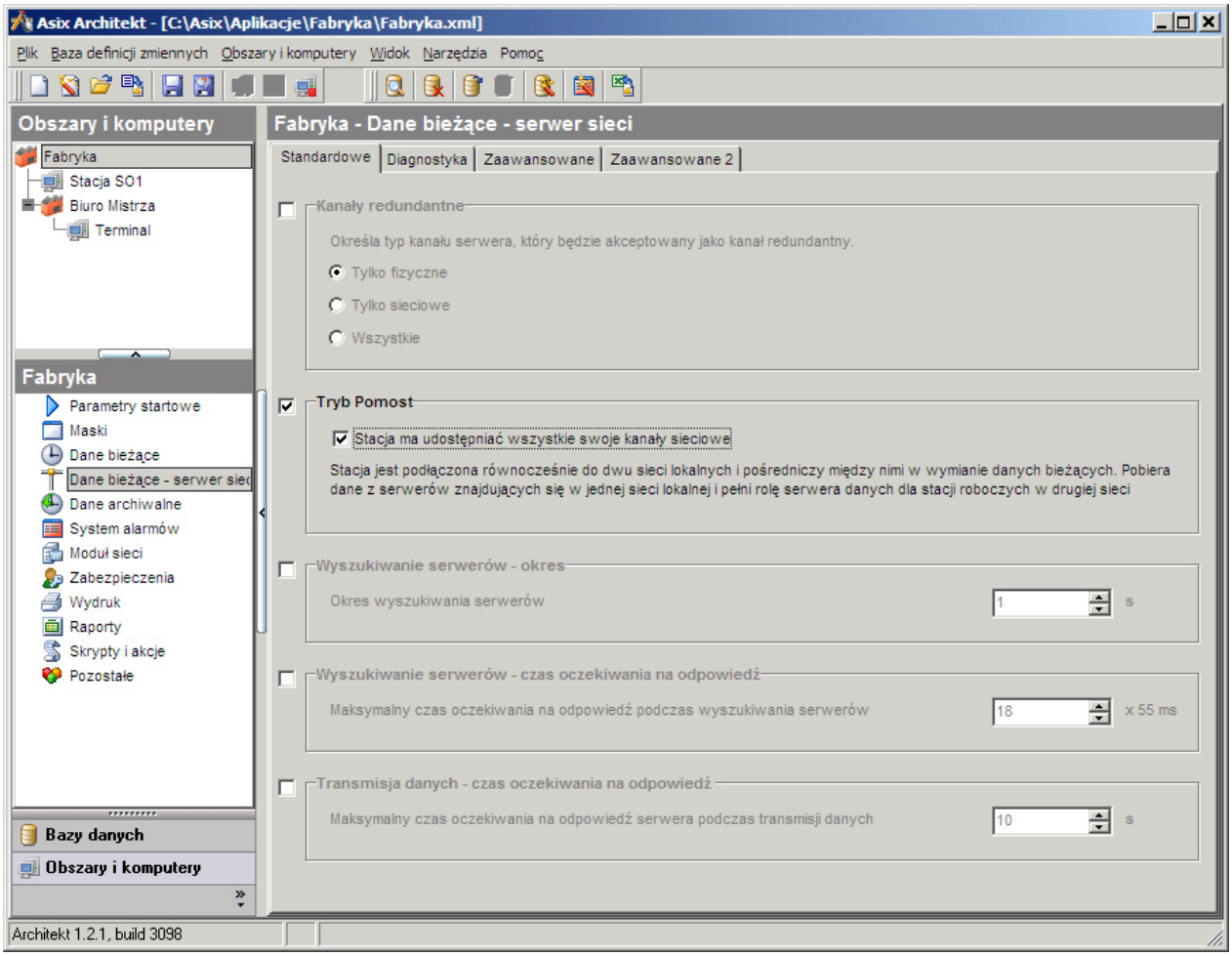
1.2.9.3. Praca stacji roboczej w trybie POMOST

W typowych zastosowaniach systemu Asix serwery danych udostępniają jedynie te dane, które pochodzą z kanałów fizycznych (bezpośrednie połączenie ze sterownikiem). Istnieją jednak specyficzne rozwiązania, w których stacja robocza musi pełnić rolę serwera danych dla innych stacji roboczych. Sytuacja taka jest typowa w konfiguracjach, w których stacja robocza systemu Asix jest pomostem (*gateway*) pomiędzy dwiema lokalnymi sieciami komputerowymi. W takim przypadku kanały sieciowe stacji roboczej muszą być traktowane na równi z kanałami fizycznymi w zakresie udostępniania danych w lokalnej sieci komputerowej. Ten tryb pracy stacji roboczej jest określany mianem pracy w trybie POMOST.

Praca stacji roboczej w trybie POMOST deklarowana jest przy użyciu parametru *Tryb Pomost*:

Architekt > *Obszary i komputery* > moduł *Dane bieżące - serwer sieci* > zakładka *Standardowe*.

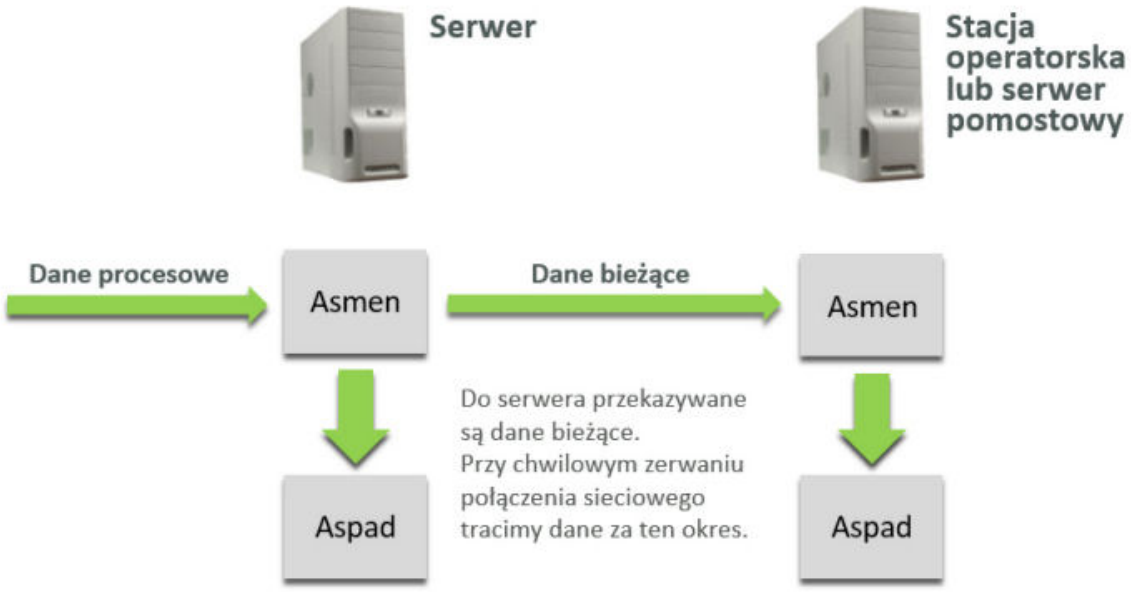
Na tej zakładce deklaruje się wszystkie opcje pracy programu ***netsrv.dll***, który realizuje pracę Asixa w trybie pomost (gateway).



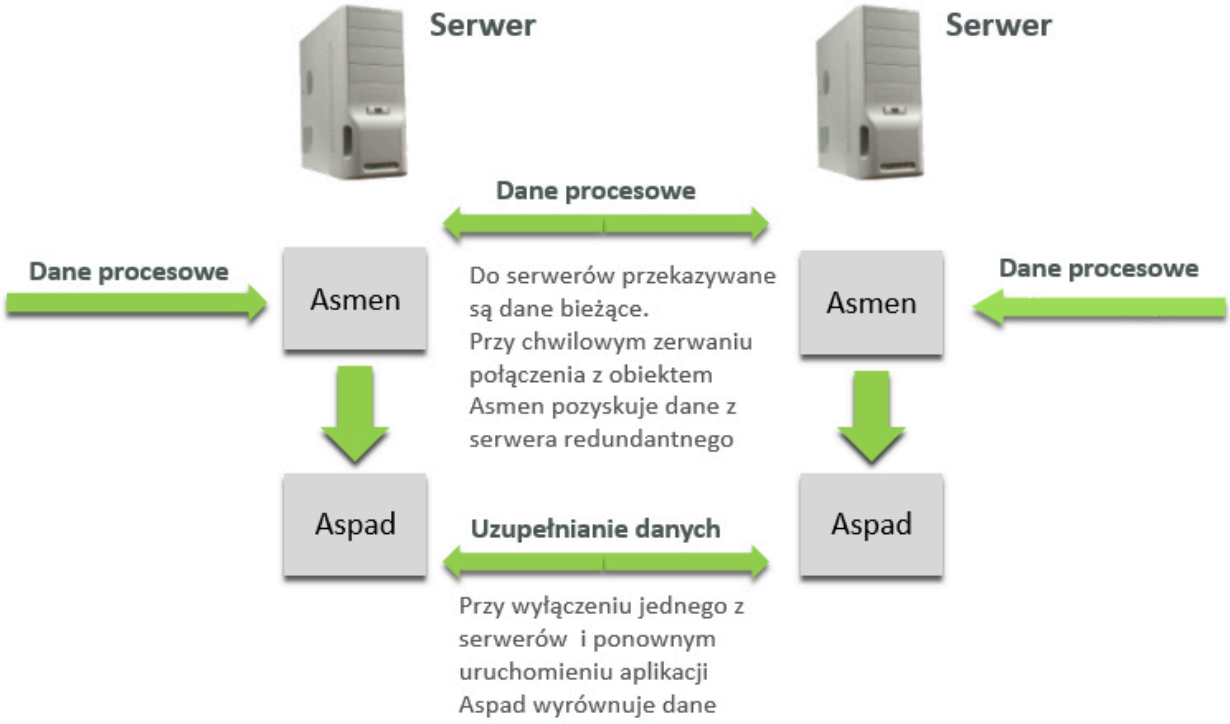
Rysunek: Deklaracja pracy stacji w trybie POMOST.

1.2.9.4. Redundancja

W celu podniesienia niezawodności komunikacji ze sterownikami w systemie Asix zaimplementowano redundantne kanały transmisji. Pozwalają one na dostęp do źródła danych zastępczym kanałem sieciowym w sytuacji, kiedy podstawowy kanał transmisji nie działa poprawnie.



Rys. Klasyczne archiwum - archiwizacja danych z modułu Asmen.



Rys. Klasyczne archiwum - redundancja (dwa lub więcej serwerów).

Redundancja może być deklarowana jedynie dla kanałów fizycznych, tzn. takich, które wykorzystują bezpośrednie połączenie ze sterownikiem.

Parametry, służące do deklaracji kanałów redundantnych, deklarowane są za pomocą opcji *Redundancja* dla każdego kanału indywidualnie. Opcja dostępna jest w:

Architekt > *Obszary i komputery* > moduł *Dane bieżące* > pozycja zadeklarowanego kanału > zakładka *Zaawansowane*:

opcja *Redundancja*:

serwer_1, serwer_2, ..., serwer_n

gdzie:

serwer_i - nazwa komputera systemu Asix, na którym należy szukać kanału zastępczego

Deklaracja opcji *Redundancja* oznacza, że w przypadku błędów transmisji w kanale transmisji *nazwa_logiczna*, dane pomiarowe będą pochodziły z serwera danych, na którym zadeklarowano kanał transmisji mający identyczną nazwę *nazwa_logiczna*.

Jeśli w deklaracji kanału redundantnego nie podano nazw serwerów danych, to kanał zastępczy będzie mógł być udostępniony przez dowolny serwer danych, dysponujący wymaganym kanałem transmisji. W przeciwnym razie kanał zastępczy będzie poszukiwany jedynie na tych serwerach danych, których nazwy umieszczono w deklaracji kanału redundantnego.

Na stacji Asixa st1, poprzez którą są realizowane sterowania z innej stacji Asixa st2, w przypadku redundancji, powinno być zadeklarowanie zezwolenie wykonania tych sterowań:

Architekt > *Obszary i komputery* > moduł *Dane bieżące* > pozycja zadeklarowanego kanału > zakładka *Zaawansowane*:

opcja *Zezwolenie zapisu*:

stacja_1, stacja_2, ..., stacja_n

gdzie:

stacja_i - nazwa komputera systemu Asix, która posiada uprawnienia do zdalnego sterowania w tym kanale

1.3. Aspad - program archiwizacji danych

Aspad jest modułem systemu Asix przeznaczonym do rejestracji i późniejszego odtwarzania przebiegów czasowych zmiennych. W systemie Asix przebiegi te są wykorzystywane przy tworzeniu wykresów i raportów, a także mogą być użyte przez przeliczniki do obliczania syntetycznych zmiennych oraz eksportowane za pośrednictwem programu **AsixConnect** w formacie DDE lub OLE do innych programów, np. EXCEL.

1.3.1. Ogólna charakterystyka programu Aspad

1.3.1.1. Podstawowe cechy programu Aspad

Program Aspad dla wersji pakietu Asix charakteryzuje się następującymi cechami:

- Pozwala na archiwizację zmiennych każdego typu dozwolonego w module Asmen, pod warunkiem, że są to zmienne skalarne. Mogą to być zarówno zmienne procesowe, jak i uzyskiwane w wyniku przeliczeń lub symulacji. W obecnej wersji nie można archiwizować tablic.
- Oprócz wartości i czasu przechowuje także status wartości zgodny ze standardem OPC.
- Gromadzi dane zarówno w plikach o specjalnie zaprojektowanej strukturze (archiwum D, M, Y z podziałem na pliki odpowiednio dobowe, miesięczne lub roczne), jak i w typowych bazach danych (archiwum B).
- Dla typu B umożliwia pozyskiwanie danych historycznych z urządzeń z pamięcią.
- Dla typu B pozwala na archiwizację danych napływających asynchronicznie i niesekwencyjnie, np. z systemów telemetrycznych.
- Umożliwia budowanie mniej lub bardziej złożonych warunków archiwizacji poszczególnych zmiennych.
- Dla typów D, M i Y oszczędza pamięć dyskową, zapamiętując jedynie istotne zmiany archiwizowanych wartości oraz stosując proste metody kompresji.
- Dzięki odpowiedniej strukturze wewnętrznej plików umożliwia szybki dostęp do danych pochodzących z dowolnego okresu.
- Pozwala na podział archiwum na pliki dobowe, miesięczne lub roczne.
- Umożliwia deklarowanie czasu przechowywania plików typu D, M i Y, po którym stare pliki będą automatycznie usuwane (przy czym przez 10 % zadanego czasu są jeszcze zachowywane ich tymczasowe kopie, dla uniknięcia problemów przy przypadkowych zmianach czasu komputera).
- Używa konwencji nazw plików, z dołączoną nazwą zasobu i czytelnym kodowaniem czasu aż do 9999 roku.
- Może aktywnie archiwizować lub tylko odtwarzać stare dane.
- Udostępnia dane w sieci.
- Pozwala na podział zmiennych archiwizowanych na grupy przypisane do oddzielnych zasobów archiwizowanych w oddzielnych plikach, w oddzielnych lokalizacjach, oddzielnie parametryzowanych i zarządzanych.
- Umożliwia określenie, z której stacji w sieci należy pobierać dane dla określonych zasobów,
- Zezwala na automatyczne tworzenie kopii zapasowych plików archiwalnych oddzielnie dla każdego zasobu i rodzaju pliku. Można określać, ile ostatnich plików ma się znaleźć na nośniku zapasowym.
- Za pośrednictwem modułu AsixConnect udostępnia dane niezależnym programom działającym w systemie Windows.

- Pozwala na przygotowanie i późniejsze odtwarzanie przebiegów wzorcowych.
- Umożliwia pracę w trybie gorącej rezerwy, uzupełniając dane z innych stacji o identycznym archiwum.
- Pozwala na określenie, którym stacjom w sieci mogą być udostępnione zgromadzone dane.
- Udostępnia interfejs programowy dla dynamicznej parametryzacji w trakcie działania oprogramowania Asix.
- Umożliwia zadawanie parametrów archiwizacji przebiegów bezpośrednio z bazy zmiennych systemu Asix.

- W pakiecie udostępniony jest także program **AspadTools**, który pozwala na:
  - zmianę konwencji nazw ze starej (Aspad 5) na nową (Aspad 6) i odwrotnie;
  - konwersję archiwum na inny typ (np. M na D);
  - eksport do pliku tekstowego;
  - sprawdzenie i naprawę archiwum;
  - zbieranie statystyk dotyczących wielkości danych dla poszczególnych zarchiwizowanych przebiegów zmiennych;
  - zmianę nazwy lub usuwanie zmiennej z pliku archiwalnego.

Opis programu AspadTools można znaleźć w odrębnym podręczniku *"AspadTools"*.

1.3.1.2. Parametry konfiguracyjne modułu Aspad

Parametry Aspada są deklarowane w pliku konfiguracyjnym Serwera Danych przy użyciu programu Architekt:

Architekt > *Obszary i komputery* > moduł *Dane Archiwalne*

Patrz: część Serwer Danych > Parametryzacja Serwera Danych > rozdz. *2.10. Konfiguracja danych archiwalnych*

1.3.1.3. Podstawowe rodzaje i typy archiwum Aspada

Podstawowym rodzajem archiwum jest **STANDARD**. Archiwum to jest zapisywane w binarnych plikach na dysku.

Drugim rodzajem archiwum jest **MSSQL**. Jest to archiwum zapisywane w bazie danych MS SQL.

Aspad gromadzi dane w plikach zawierających przebiegi z jednego okresu dobowego, miesięcznego lub rocznego odpowiednio do typu archiwum:

Archiwum typu **D** (*ang. day*)	- dane gromadzone w plikach dobowych;
Archiwum typu **M** (*ang. month*)	- dane gromadzone w plikach miesięcznych;
Archiwum typu **Y** (*ang. year*)	- dane gromadzone w plikach rocznych;
Archiwum typu **H** (*ang. horizon*)	- dane gromadzone w jednym pliku;
Archiwum typu **B** (Baza Danych)	- dane gromadzone w typowych bazach danych;
Archiwum typu **X**	- dane gromadzone w bazie MS SQL;

Aspad zapewnia dostęp do danych ze wszystkich okresów, których pliki znajdują się w kartotekach wskazanych przy parametryzacji.

Archiwa typu **H** (z horyzontem - *ang.* ***h****orizon*) są przechowywane w jednym pliku dla określonej grupy zmiennych. Dane starsze niż zadany horyzont są automatycznie zapominane, a miejsce w pliku jest odzyskiwane do dalszej archiwizacji. Ten typ archiwum można stosować do tymczasowej archiwizacji dodatkowych przebiegów, np. do szybkiej rejestracji niektórych przebiegów w celach diagnostycznych. W obecnej wersji zaleca się w tym celu stosowanie archiwum **D** z zadanym horyzontem przechowywania plików, a archiwum **H** zachowano jedynie dla zachowania zgodności z poprzednimi wersjami programu Aspad.

Archiwizacja dla typu **B** to gromadzenie wartości w bazie danych. Aktualnie są to bazy danych w standardzie MS SQL. Nadaje się on do rejestracji wartości z małą częstotliwością dając w zamian dużą elastyczność przy korzystaniu z takiej bazy danych, a także możliwość uzupełniania danych w archiwum.

Archiwizacja dla typu **X** to gromadzenie wartości w bazie danych. To archiwum, zapisywane w bazie danych SQL, jest poddawane kompresji danych i umożliwia przechowywanie czasu z zadaną precyzją, maksymalnie do 100 nanosekund. Dane mogą być archiwizowane zgodnie z tempem odświeżania wartości przez moduł danych bieżących Asmen, czyli z minimalnym okresem 100 milisekund.

Oprócz wymienionych sposobów archiwizacji można także generować przebiegi wzorcowe, traktując je później podobnie jak archiwum. Przebiegi tego typu oznaczone są literą **P** (*ang.* ***p****attern*).

Można jeszcze wyróżnić podział archiwum ze względu na sposób udostępniania danych:

Archiwum do zapisu - podstawowe archiwum pozwalające na zapis i odczyt danych z plików archiwalnych;

Archiwum do odczytu - to archiwum pozwalające tylko na odczyt danych z plików archiwalnych;

Archiwum sieciowe - to archiwum pozwalające na odczyt danych przez sieć z innego komputera;

Archiwum pomocnicze - to archiwum pozwalające na zapis i odczyt danych, służące jako redundancja archiwum podstawowego na innym komputerze;

Archiwum replikowane - archiwum pozwalające na zapis i odczyt danych z archiwum; dane do archiwum pobierane są z archiwum innego komputera.

Patrz: sposób parametryzacji poszczególnych archiwów w *Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych* > 2.10.9. *Parametryzacja archiwum*.

Archiwum pomocnicze jest to archiwum do zapisu umieszczane na ***serwerze pomocniczym*** danego zasobu. Archiwum pomocnicze powinno stanowić redundancję archiwum podstawowego na serwerze podstawowym. Różnica między archiwum pomocniczym a archiwum podstawowym polega na preferencjach dla archiwum podstawowego podczas przyłączania klientów sieciowych. Klient przyłączy się do archiwum pomocniczego tylko wtedy, gdy w sieci nie znajdzie archiwum podstawowego.

Archiwum pomocnicze jest stosowane w celu zapewnienia tymczasowej redundancji na czas konserwacji lub awarii serwera podstawowego. Można np. zadeklarować archiwum podstawowe ze 100-dniowym horyzontem i archiwum pomocnicze z horyzontem 10-dniowym. Pozwoli to na uzupełnienie archiwum podstawowego za okres przerwy nie przekraczającej 10 dni, przy czym w obecności serwera podstawowego klienty przyłączą się do niego, zapewniając obserwację danych z horyzontem 100-dniowym.

Archiwum replikowane ma być stosowane na serwerach, na których Asmen nie ma kanałów fizycznych dla danych archiwizowanych w danym zasobie, a równocześnie w sieci dostępne jest inne archiwum, z którego można pozyskać dane historyczne. Dla archiwum replikowanego Aspad nie pobiera danych bieżących z Asmena. W zamian stale pozyskuje dane historyczne z innego serwera, na którym archiwizowane są dane z kanałów fizycznych, zapisując je we własnym archiwum. Typowym zastosowaniem będą serwery pośredniczące.

Sposób działania nieco się różni dla archiwum standardowego i SQL.

Archiwum standardowe
Dla archiwum standardowego Aspad po uzupełnieniu danych przy starcie systemu, kontynuuje to uzupełnianie, cyklicznie sprawdzając nowe dane pojawiające się na drugim serwerze. Nawet dłuższe przerwy w komunikacji są na bieżąco nadrabiane po uzyskaniu łączności, aż do ograniczenia podanego w deklaracji: program Architekt > *Obszary i komputery* > *Dane archiwalne* > <*zadeklarowane archiwum*> > zakładka *Uzupełnianie* > opcja *Minimalny okres uzupełniania*.
Także w zakładce *Uzupełnianie* można określić nazwy serwerów, z których mają być pobierane dane do replikacji.
Opcja *Bez uzupełniania* na zakładce *Uzupełnianie* dla takiego archiwum nie ma sensu i jest ignorowana.

Archiwum SQL
Jeżeli dla danego zasobu nie podano deklaracji synchronizacji archiwum SQL, to archiwum replikowane działa tak samo, jak standardowe.
Jeżeli zadeklarowano opcję *Synchronizacja* w Architekt > *Obszary i komputery* > *Dane archiwalne* > <*archiwum*> > zakładka *Synchronizacja*, to dane są tylko synchronizowane na poziomie baz danych SQL, a wyłączone jest zarówno pobieranie danych bieżących z Asmena, jak i uzupełnianie przez Aslinka. Synchronizacja ma tą przewagę nad zwykłym uzupełnianiem, że uwzględnia wszystkie zmiany w archiwum, także dane historyczne dopisywane lub zmieniane z opóźnieniem.

1.3.1.3.1. Archiwum replikowane

Jak działa archiwum replikowane?

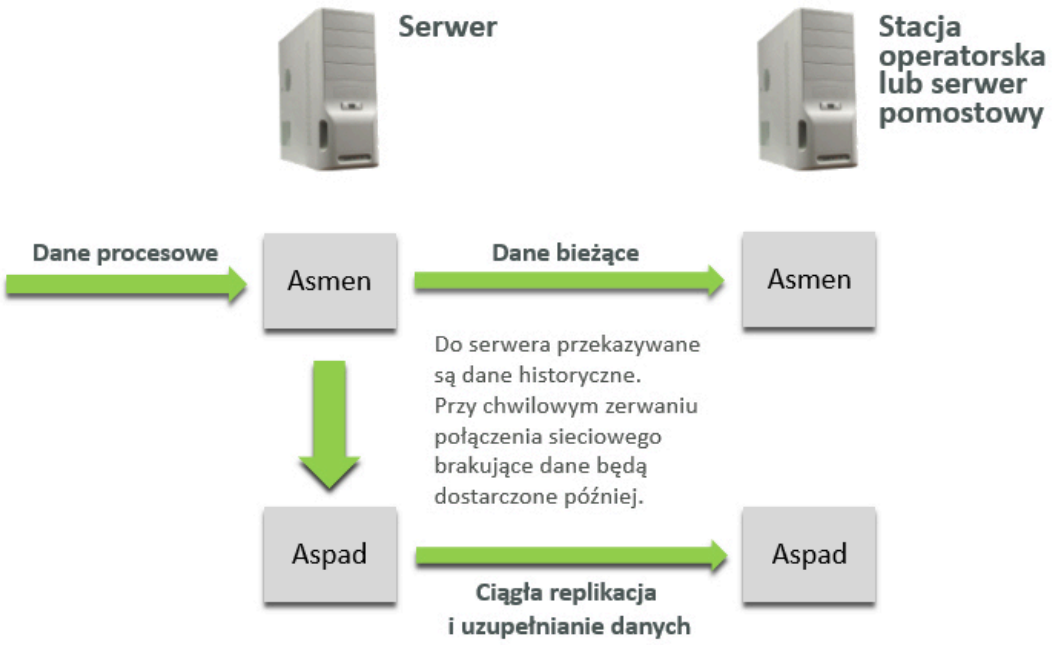
Archiwum replikowane ma zastosowanie na serwerach, na których Asmen nie ma kanałów fizycznych dla danych archiwizowanych w danym zasobie, a równocześnie w sieci dostępne jest inne archiwum, z którego można pozyskać dane historyczne. Dla archiwum replikowanego Aspad nie pobiera danych bieżących z Asmena. W zamian stale pozyskuje dane historyczne z innego serwera, na którym archiwizowane są dane z kanałów fizycznych, zapisując je we własnym archiwum. Typowym zastosowaniem są serwery pośredniczące.
Sposób działania nieco się różni dla archiwum standardowego i SQL.

Archiwum standardowe

Dla archiwum standardowego Aspad po uzupełnieniu danych przy starcie systemu, kontynuuje to uzupełnianie (replikuje dane), cyklicznie sprawdzając nowe dane pojawiające się na drugim serwerze. Nawet dłuższe przerwy w komunikacji są na bieżąco nadrabiane po uzyskaniu łączności, aż do ograniczenia podanego w deklaracji *Minimalny okres uzupełniania* na zakładce *Uzupełnianie* w parametrach danego archiwum.
Także na wspomnianej zakładce *Uzupełnianie* można określić nazwy serwerów, z których mają być pobierane dane do replikacji.



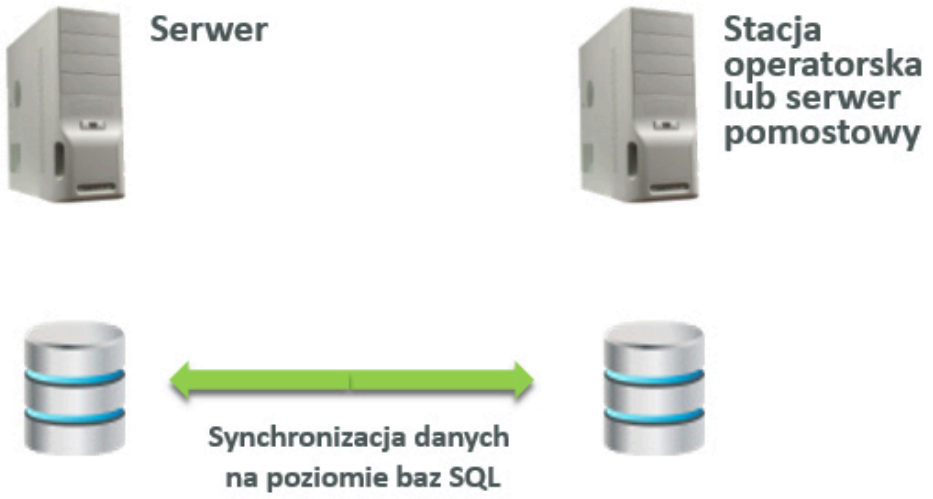
Rys. Klasyczne archiwum - archiwizacja danych z modułu Asmen.



Rys. Archiwum replikowane - realizacja architektury kolektor-serwer.

Archiwum SQL

Jeżeli dla danego zasobu nie podano deklaracji synchronizacji archiwum SQL, to archiwum replikowane działa tak samo, jak standardowe.
Jeżeli zadeklarowano opcję *Synchronizacja* w Architekt > *Obszary i komputery* > *Dane archiwalne* > <*archiwum*> > zakładka *Synchronizacja*, to **dane są tylko synchronizowane na poziomie baz danych SQL, a wyłączone jest zarówno pobieranie danych bieżących z Asmena, jak i uzupełnianie przez Aslinka.**



Rys. Synchronizacja na poziomie baz danych SQL.

Synchronizacja ma tę przewagę nad zwykłym uzupełnianiem, że uwzględnia wszystkie zmiany w archiwum, także dane historyczne dopisywane lub zmieniane z opóźnieniem.

Deklaracja

Deklaracja wygląda dokładnie tak samo, jak archiwum do zapisu:

Architekt > *Obszary i komputery* > *Dane archiwalne* > <*dodana pozycja archiwum*> > zakładka *Standardowe* > opcja ***Archiwum replikowane***

Jakie problemy pozwala wyeliminować archiwum replikowane?

Przy zastosowaniu pozostałych dostępnych typów archiwum moduł Aspad pozyskuje dane do archiwizacji na 3 sposoby:

1. Dane bieżące lub historyczne pobiera z modułu Asmen.

2. Podczas startu uzupełnia brakujące dane z innego serwera za czas wyłączenia Asixa.

3. Synchronizuje dane archiwum SQL z innym serwerem, jeżeli podano dla tego archiwum deklarację synchronizacji.

Na serwerach, gdzie ASMEN nie ma swoich kanałów fizycznych takie rozwiązanie ma kilka wad:

1. Archiwizowane dane muszą być dostarczane kanałami sieciowymi Asmena nie rzadziej niż okres archiwizacji. W konsekwencji:

    a. jeżeli Asmen nie zdąży przekazać danych z takim cyklem, to ma tylko jeden dodatkowy cykl na nadrobienie zaległości; później pojawia się dziura w archiwum;

    b. jeżeli okres odświeżania Asmena jest krótszy niż Aspada, to dane są dostarczane częściej niż jest to potrzebne.

2. Przy chwilowym braku komunikacji z źródłem danych powstaje dziura w archiwum, która nie jest uzupełniana. Wyjątkiem jest archiwum SQL z deklaracją synchronizacji.

3. Jeżeli zadeklarowano ***Archiwum do zapisu*** oraz zadeklarowano synchronizację, to dane do archiwum SQL dostarczane są podwójnie: pierwszy raz jako dane bieżące Asmena, a drugi jako dane z synchronizacji z drugim archiwum.

Archiwum replikowane powstało w celu wyeliminowania powyższych niedogodności.

Gdzie stosować archiwum replikowane?

Wybór pomiędzy archiwum replikowanym a zwykłym archiwum do zapisu zależy głównie od 2 czynników: czy na danej stacji Asmen dostarcza dane z kanałów fizycznych, czy też sieciowych oraz jak szybko dane historyczne mają się pojawiać na wykresach w przelicznikach, skryptach, itd.

Dane z kanałów fizycznych, czy sieciowych

Jeżeli zastosujemy archiwum replikowane, to dane do archiwizacji będą zawsze pobierane z archiwum innego komputera. Będzie tak nawet wtedy, gdy chwilowo ten inny komputer jest wyłączony, co oznacza, że w naszym archiwum znajdą się tylko te dane, które były wcześniej zarchiwizowane w innym i będzie brakowało danych za czas jego wyłączenia. Utracilibyśmy w ten sposób zdolność korzystania z redundantnych kanałów Asmena na naszym komputerze.

Wnioski:
1. Archiwum replikowane zwykle nadaje się do zastosowania na serwerach pośredniczących (z uwagą o opóźnieniu dostarczanych danych).
2. Archiwum replikowanego nie należy stosować na redundantnych stacjach operatorskich.

3. Archiwum replikowane może być zastosowane na stacji operatorskiej, na której wszystkie dane tego archiwum pochodzą z kanałów sieciowych i mogą być pobrane z bliźniaczego archiwum komputera, który posiada kanały fizyczne dla tych danych.

Opóźnienie danych

Dane archiwizowane z dostarczanych danych bieżących (w pozostałych typach archiwum) są dostępne praktycznie natychmiast, czyli mogą być na bieżąco obserwowane na wykresach, czy też użyte w skryptach lub przelicznikach.

W przypadku danych replikowanych serwer wysyła dane nie częściej niż raz na dziesięć sekund, a w przypadku archiwum SQL nie częściej niż podano w deklaracji synchronizacji (o ile taką deklarację zastosowano). Dopiero po ich dostarczeniu dane będą dostępne na stacji z archiwum replikowanym.
Dane będą przy tym opatrzone prawidłowymi znacznikami czasu, a jedynie opóźni się ich pojawienie w archiwum.

Wnioski:
1. Archiwum replikowane można stosować tam, gdzie nie jest istotne opóźnienie pojawienia się danych historycznych w stosunku do danych bieżących.
2. Nie wolno stosować archiwum replikowanego na stacjach, na których wykres musi nadążać z dokładnością do sekund za danymi bieżącymi.
3. Nie wolno stosować archiwum replikowanego na stacjach, na których skrypty lub przeliczniki nie są odporne na opóźnione pojawienie się danych historycznych.

1.3.1.4. Cechy archiwizacji prowadzonej w plikach o dedykowanym formacie

Cechy archiwizacji prowadzonej w plikach o dedykowanym formacie:

• nielimitowana liczba zmiennych rejestrowanych;
• maksymalna częstotliwość próbkowania = 1 s;
• kompresja danych – surowa próbka (10 bajtów) podlega kompresji, co wielokrotnie zmniejsza jej pojemność; stosowane są dwa rodzaje kompresji:

  • w pierwszej kolejności pomijane są próbki z nieistotną zmianą wartości (tzw. strefa nieczułości);
  • w drugim etapie zapisana próbka poddawana jest kompresji, zmniejszającej jej objętość typowo do 1-3 bajtów;

• typy danych archiwalnych: bajt, pojedyncze/podwójne słowo, zmienny przecinek;
• automatyczne zarządzanie biblioteką archiwum;
• wyliczanie i archiwizowanie szeregu predefiniowanych agregatów radykalnie przyśpieszających zaawansowaną analizę danych; efekt przyśpieszenia jest szczególnie zauważalny w instalacjach sieciowych, gdzie wiele stanowisk jednocześnie korzysta z zasobów jednego serwera;
• uzupełnianie „w locie” luk w archiwum, jeśli tylko w sieci zidentyfikowany został (samoczynnie) inny komputer posiadający w swoim archiwum brakujące dane.
Mechanizm ten umożliwiaa realizację systemów, w którym centralna baza danych procesowych przechowywana w dedykowanym formacie plików Asix, zostanie wcześniej czy później automatycznie uzupełniona o brakujące informacje.

1.3.1.5. Cechy archiwizacji prowadzonej w relacyjnej bazie danych SQL

Archiwizacja danych prowadzona w relacyjnej bazie danych SQL:

- nielimitowana liczba zmiennych rejestrowanych;
- maksymalna częstotliwość próbkowania = 10 próbek na sekundę;
- wsteczne uzupełnianie danych historycznych wymagane np.:
  - w systemach dyspozytorskich ze zbieraniem danych na łączach modemowych, w paczkach za określony przedział czasu;
  - w komunikacji z urządzeniami umożliwiającymi buforowanie danych pomiarowych, po przerwie spowodowanej brakiem komunikacji.
- typy danych archiwalnych: bajt, pojedyncze/podwójne słowo, zmienny przecinek.

1.3.1.6. Agregacja danych w systemie Asix

Od wersji 5 systemu Asix dostępny jest mechanizm wyliczania i archiwizowania predefiniowanych agregatów radykalnie przyśpieszających zaawansowana analizę danych archiwalnych. Efekt przyśpieszenia jest szczególnie zauważalny w instalacjach sieciowych, gdzie wiele stanowisk jednocześnie korzysta z zasobów jednego serwera (**patrz:** Serwer Danych > 1. Obsługa danych procesowych Serwera Danych > Aspad - program archiwizacji danych > Ogólna charakterystyka programu Aspad > *Agregacja danych archiwalnych*).

1.3.2. Moduły związane z Aspadem

W skład modułów Aspada wchodzą:

Aspad.dll	- podstawowy moduł programu Aspad,
AsSQL.dll	- moduł dostępu do bazy danych typu SQL,
BBrowserSQL.exe	- program przeglądarki bazy danych,
AspadTools.exe	- narzędzia do konserwacji plików archiwalnych,
PEdit.exe	- program edycji przebiegów wzorcowych w wersji klasycznej,
PSelect.exe	- interfejs przebiegów wzorcowych dla skryptów w wersji klasycznej.

1.3.3. Wymagania systemowe

Aspad jest przeznaczony do współpracy z innymi modułami systemu Asix. Jest on dostarczany jako biblioteka DLL. Ładowanie biblioteki odbywa się automatycznie podczas ładowania Asixa.

Dla zapewnienia poprawnej pracy Aspada konieczne jest, by Asmen został sparametryzowany tak, aby dostarczał wszystkie archiwizowane dane.

Dla prawidłowej pracy w sieci wymagany jest także moduł Aslink wraz z oprogramowaniem sieciowym.

Dodatkowo w przypadku pracy z archiwum B, X i przebiegami wzorcowymi P wymagany jest dostęp do serwera SQL. Do pakietu Asix dołączany jest instalator SQL Server Express.

Konfiguracja modułu Aspad pod kątem realizacji archiwizacji typu B opisana została w części Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Archiwizacja typu B > *Dodatkowe wymagania archiwum typu B*.

Wymagania dla archiwum X: Serwer Microsoft SQL od wersji 2008 R2.

W części Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Wymagania systemowe > *Pamięć operacyjna* podano metody oszacowania wielkości pamięci operacyjnej i dyskowej wymaganej dla pracy modułu archiwizacji Aspad w zależności od liczby zmiennych archiwizowanych i typów archiwum.

1.3.3.1. Pamięć operacyjna

Obszar pamięci potrzebny modułowi Aspad jest sumą wielkości kodu i danych oraz przydzielonych obszarów roboczych.

Obszar pamięci operacyjnej przydzielanej dynamicznie zależy głównie od liczby archiwizowanych przebiegów zmiennych. Struktury danych związane z każdym przebiegiem archiwizowanym w sposób D, M lub Y zajmują od około 0.8 KB do 1.3 KB, a dla typu B około 0.3 KB. Jeżeli zmienna jest archiwizowana na kilka sposobów, to należy ją uwzględnić dla każdego sposobu oddzielnie.

Aspad przydziela pamięć w miarę potrzeby w trakcie działania systemu. Początkowa zajętość pamięci dla archiwum D, M lub Y wynosi zwykle 0.8 KB na zmienną i nie może być traktowana jako całkowite zapotrzebowanie pamięci. W przypadku, gdy bufor o wielkości 0.5 KB przypisany do każdej zmiennej zostanie zapełniony danymi, jest inicjowana operacja jego zapisu na dysk, a dla nowych danych przydzielany jest nowy bufor. W efekcie, do momentu zakończenia zapisu, zmienna zajmuje 1.3 KB. Jeżeli taka operacja zostanie zainicjowana dla wielu zmiennych równocześnie, to może chwilowo wzrosnąć zajętość pamięci. Zjawisko takie zachodzi w chwili przechodzenia do nowego pliku, np. na początku nowej doby, gdy opisana operacja jest wykonywana dla wszystkich przebiegów typu D.

PRZYKŁAD

Dla struktur danych 5000 zmiennych w archiwum D i 5000 zmiennych w archiwum M należy w sumie przewidzieć około 13 MB niestronicowanej pamięci RAM.

1.3.3.2. Pamięć dyskowa dla archiwum D, M, Y i H

Wielkość potrzebnej pamięci dyskowej zależy w głównej mierze od liczby i wielkości przechowywanych rekordów zawierających kolejne punkty wszystkich archiwizowanych przebiegów zmiennych.

Dla archiwum typu H należy przewidzieć miejsce na dane pochodzące z okresu dwukrotnie dłuższego niż zadany horyzont czasowy.

Dla archiwum typu D, M i Y należy zapewnić obszar wystarczający dla danych z tylu plików danego typu, ile będzie przechowywanych równocześnie na dysku.

Dodatkowo w plikach zapamiętywane są informacje organizujące wewnętrzną strukturę plików. W archiwum D, M, Y i H te dodatkowe informacje zajmują ponad 10 % pliku dla przebiegów zawierających bardzo dużą liczbę danych (rzędu kilkudziesięciu KB dla jednej zmiennej). Dla mniejszej liczby danych udział ten rośnie.

Oszacowanie wielkości pamięci dyskowej zajmowanej przez rekordy danych nie może być jednoznaczne, gdyż dane te poddawane są kompresji, której efektywność zależy od charakteru przebiegu i sposobu parametryzacji archiwum. Każdy rekord przed kompresją składa się z 1 bajta statusu opisującego rekord (np. odróżniającego dziurę od danych), 4 bajtów znacznika czasu, 2 bajtów jakości zmiennej (zgodnej ze standardem OPC) i od 1 do 4 bajtów wartości zmiennej, w zależności od jej typu. Do pliku zapisywane są tylko te rekordy, dla których wartość zmieniła się w istotny sposób, tzn. o więcej niż zadaną przy parametryzacji dokładność rejestracji. Dla przebiegów wolnozmiennych (w odniesieniu do zadanej częstotliwości próbkowania) rejestracji podlegają tylko niektóre rekordy. W trakcie kompresji większość znaczników czasu zostaje zredukowana do 1 bajta lub nawet całkowicie pomijana, zostawiając jedynie ślad w bajcie statusu. Kompresja jest tym lepsza, im mniejsza jest wymagana dokładność rejestracji czasu. Kompresja pozwala także na redukcję większości wartości zmiennej do 1 lub 0 bajtów. Jedynie dla wartości zmieniających się skokowo o wartości dużo większe niż zadana dokładność rejestracji, kompresja może nie być tak skuteczna. Słowo jakości jest zapisywane tylko w razie jego zmiany. Także bajt statusu może zostać pominięty, jeżeli w kolejnych rekordach rozmiary zredukowanego znacznika czasu i wartości zmiennej są takie same.

W efekcie przy archiwizacji liczb stałoprzecinkowych lub zmiennoprzecinkowych z zadaną dokładnością rejestracji typowa wielkość rekordu po kompresji wynosi 1-2 bajty. Ta sama wielkość rekordu jest osiągana także dla liczb zmiennoprzecinkowych bez zadanej dokładności rejestracji przy niektórych funkcjach przeliczających Asmena, dla których wartość surowa jest liczbą stałoprzecinkową. Dla innych liczb zmiennoprzecinkowych typowa wielkość rekordu wynosi 3-5 bajtów.

Także dziury w archiwum włączone są do kompresji i zajmują 1-2 bajty.

1.3.3.3. Pamięć dyskowa dla archiwum B

W przypadku stosowania baz danych typu SQL. Baza oprócz samych danych, zawiera też indeksy, pola ograniczające rekordy oraz puste obszary, wynikające ze sposobu rozmieszczania danych w plikach. Dodatkowo tworzony jest też plik logu transakcji. Szacunkowo wielkość samej bazy danych, bez logu transakcji, wynosi:

- 142 bajty na próbkę dla pełnego rekordu z wszystkimi polami opcjonalnymi;
- 113 bajtów na próbkę dla rekordu bez pól opcjonalnych.

Archiwum B nie nadaje się do gromadzenia dużej ilości danych. Jego zaletą jest jednak stosowanie standardowych plików bazy danych. Jest też jedynym archiwum, w którym można uzupełniać braki danych.

1.3.4. Pliki tworzone przez programy modułu Aspad

Programy modułu Aspad samodzielnie tworzą szereg plików o różnym przeznaczeniu, umieszczając je w kartotekach zadanych w pliku konfiguracyjnym. Zawierają one dane, informacje o konfiguracji, raporty z przetwarzania plików archiwalnych, kopie oryginalnych plików archiwalnych po ich ewentualnym przetwarzaniu oraz komunikaty diagnostyczne.

1.3.4.1. Pliki archiwalne D, M, Y i H

Podział archiwum na pliki pod kątem czasu
Podział plików na zasoby
Konwencja nazw plików
Przykłady

Podział archiwum na pliki pod kątem czasu

Przebiegi archiwalne typu D, M, Y i H gromadzone są w czterech rodzajach plików różniących się czasem gromadzonych danych. Każda zmienna może być archiwizowana równocześnie w kilku typach plików. Dla przebiegów D, M i Y tworzone są pliki zawierające dane z określonego okresu: doby (D), miesiąca (M) lub roku (Y).

Na początku każdego okresu tworzony jest nowy plik o unikatowej nazwie. Pliki z poprzednich okresów przechowywane są do momentu, gdy minie okres zadany w parametrach archiwum. Niektóre pliki mogą zostać programowo zabezpieczone przed automatycznym usuwaniem, co jest określane w parametrach systemu Asix. Takie pliki mogą być usunięte podczas konserwacji systemu. Najstarsze pliki są też usuwane automatycznie w przypadku braku miejsca na dysku. Wcześniej cyklicznie wysyłane są ostrzeżenia, co pozwala na świadomą reakcję, np. kopiowanie archiwum na nośnik zapasowy.

Takie podejście pozwala na elastyczny podział archiwum na części. Pliki dobowe są najmniejsze, co ułatwia ich konserwację, składowanie, czy selektywne usuwanie. Zwykle jednak korzystanie z plików dobowych jest uzasadnione tylko dla często próbkowanych i zmieniających się danych. Zastosowanie ich np. do danych godzinnych wiąże się ze stratą pamięci dyskowej, bowiem deklaracja zmiennej w nowym pliku zajmuje około 40 bajtów, a pamięć dyskowa dla danych przebiegu zmiennej jest przydzielana porcjami po 512. Zapamiętanie 24 rekordów, które zwykle po kompresji zajmują 1-2 bajty na rekord, spowoduje konieczność zajęcia 552 bajtów na zapamiętanie kilkudziesięciu punktów.

Przykładem efektywnego wykorzystania dysku jest archiwizacja danych chwilowych w plikach dobowych, średnich 5-minutowych w miesięcznych, a średnich godzinnych w rocznych.

Pliki archiwum H nie są dzielone na okresy, a miejsce zwalniane przez stare dane jest ponownie wykorzystywane przez nowe.

Podział plików na zasoby

W pliku inicjującym poszczególne grupy zmiennych przypisuje się do nazwanych zasobów parametryzowanych w deklaracjach **ARCHIWUM**, a każdy zasób jest archiwizowany w oddzielnych plikach o deklarowanej dla każdego zasobu oddzielnie lokalizacji. Nazwa zasobu jest częścią nazwy pliku.

Konwencja nazw plików

Aktualna konwencja nazw plików wykorzystuje długie nazwy plików, pozwala na kodowanie dowolnej daty i stosuje, zgodnie z konwencją Windows, jednolite rozszerzenie nazwy pliku. Została ona wprowadzona w wersji 6 programu Aspad, zapewniając dostosowanie nazw plików do nowych cech Aspada i wymogów systemu Windows.

Nazwa pliku jest zbudowana z kilku członów wg wzorca:

*Trrrrmmdd-zasób***.ahf**

gdzie:
T - jednoliterowa nazwa rodzaju archiwum (D,M,Y,H);
rrrr - 4-cyfrowy rok, dla archiwum H zastępowany tekstem **xxxx;**
mm - 2-cyfrowy miesiąc, dla archiwum H i Y zastępowany tekstem **xx**;
dd - 2-cyfrowy dzień miesiąca, dla archiwum H, M i Y zastępowany tekstem **xx**;
zasób - zadeklarowana w pliku inicjującym nazwa zasobu;
.ahf - stały tekst rozszerzenia (*ang*. **A**SPAD **H**istorical **F**ile).

PRZYKŁADY

M201605xx-BLOK.ahf - oznacza plik miesięczny z maja 2016 dla zasobu BLOK,
D20160416-ZAKL1.ahf - oznacza plik dobowy z 16.04.2016 dla zasobu ZAKL1.

1.3.4.2. Pliki archiwum B i P

W trakcie działania archiwum B i P moduł *ASSQL.dll* tworzy bazy danych w domyślnym katalogu serwera SQL. Zawartość baz danych może być w tym przypadku przeglądana przy wykorzystaniu standardowych narzędzi przeznaczonych do przeglądania baz danych typu SQL.

1.3.5. Konfigurowanie systemu przy instalacji modułu Aspad

Przed uruchomieniem modułu Aspad należy:

- odpowiednio sformatować dysk i zainstalować system Windows - ze względu na niezawodność i wydajność **zaleca się archiwizację tylko na partycji NTFS**,
- tworzyć pliki konfiguracyjne Aspada (pliki konfiguracyjne zawierające opisy warunków przy archiwizacji warunkowej),
- odpowiednio sparametryzować programy Aslink i Asmen, aby mogły współpracować z Aspadem, dostarczając mu dane do archiwizacji.

Nie trzeba tworzyć ręcznie kartotek zawierających archiwum, gdyż Aspad może je utworzyć sam w momencie uruchomienia.

1.3.6. Konfigurowanie i uruchomienie modułu Aspad

Aspad jest ładowany automatycznie w trakcie ładowania Asixa, a parametry jego działania pobierane są z pliku konfiguracyjnego XML Serwera Danych. Ładowanie Aspada może zostać pominięte, jeżeli zostanie załączona opcja *Symulacja*:

Architekt > *Obszary i komputery* > moduł *Dane archiwalne* > zakładka *Standardowe*

1.3.6.1. Specyfikacja odcinków czasu w plikach konfiguracyjnych

W plikach konfiguracyjnych często zachodzi potrzeba określenia horyzontu czasowego, częstotliwości lub okresu wykonywania określonych akcji. W parametryzacji Aspada zastosowano jednolity sposób ich specyfikacji. Opis czasu podaje się w następującej formie:

<liczba><jednostka> [<liczba><jednostka> [...]]

gdzie:

<liczba> - jest liczbą podanych po niej jednostek czasu, mieszczącą się w granicach od 1 do 32767,
<jednostka> - określa jednostkę czasu, którą może być:
s - sekunda,
m - minuta,
g lub h - godzina,
d - doba.

PRZYKŁAD

3600s	lub 1g	= 1 godzina
1g30m	lub 1g 30m lub 90m	= 1 godzina 30 minut
48h	lub 48g lub 2d	= 2 doby

1.3.6.2. Zadawanie podstawowych parametrów archiwizacji

Przy startowaniu archiwizacji konieczne jest określenie kilku jej globalnych parametrów. Do ich specyfikacji służy moduł *Dane archiwalne* w programie Architekt. Dokładne opisy poszczególnych opcji podano w:

Serwer Danych > Parametryzacja Serwera Danych > *2.10. Konfiguracja danych archiwalnych*

Archiwum Aspada dzieli się na zasoby:

Archiwum do odczytu,
Archiwum do zapisu,
Archiwum pomocnicze,
Archiwum sieciowe.

Przebiegi zmiennych danego zasobu są zapisywane lub czytane do/z plików archiwalnych umieszczonych w kartotekach określonych dla powyższych rodzajów archiwum.

Archiwum definiowane jest przez następujące parametry:

zasób - logiczna nazwa archiwum,
rodzaj - rodzaj parametryzowanego zasobu:
STANDARD - archiwizacja w plikach,
MSSQL - archiwizacja w bazie danych,
dodatkowe parametry - parametry zależne od rodzaju zasobu.

Patrz: Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych > *2.10.9. Parametryzacja archiwum*

1.3.6.3. Automatyczne tworzenie kopii zapasowych

Aspad może cyklicznie tworzyć kopie zapasowe plików archiwalnych typu D, M i Y na dodatkowym dysku. Do tego celu służy opcja ***Kopia*** określająca, które pliki i kiedy należy kopiować.

Patrz: Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych > *4.8.9.5. Parametryzacja archiwum > Parametry określające sposób tworzenia kopii zapasowej archiwum*

1.3.6.4. Automatyczne kasowanie najstarszych plików

Możliwe jest zadawanie czasu, po jakim określone pliki danego zasobu mają być automatycznie usunięte. Stwarza to możliwość przechowywania przebiegów, które mają być zapominane po określonym czasie, wyrażonym w dniach.

Przypadkowa, nawet chwilowa, duża zmiana czasu komputera mogłaby doprowadzić do automatycznego usunięcia ważnych plików. W celu uniknięcia takiej sytuacji, po upływie terminu ważności pliku, jego nazwa *.ahf zamieniana jest na *.ahr, a ten jest jeszcze przechowywany przez kolejne 10% okresu ważności pliku.

Patrz: Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych > *Parametryzacja archiwum > 2.10.9.1. Parametry standardowe*

1.3.6.5. Sieciowy dostęp do danych Aspada

Aspad, w obecności programu Aslink, udostępnia w sieci komputerowej dane zgromadzone w swoich plikach archiwalnych (warunkiem jest jeszcze posiadanie klucza z uprawnieniami serwera). Aspad jest wtedy serwerem sieciowym tych zasobów. Udostępnione dane można czytać na innych komputerach w sieci za pośrednictwem programów Aspad, dla których te same zasoby zostały wyspecyfikowane w parametrach archiwum sieciowego. Dalej będą one nazywane klientami sieciowymi tych zasobów.

Dla klienta parametr ***Powtarzanie szukania serwerów*** określa, jak często ponawiać próbę szukania serwera, jeżeli zasób nie jest aktualnie dostępny w sieci:

Architekt > *Obszary i komputery* > moduł *Dane archiwalne* > zakładka *Standardowe*

Parametr ***Wyszukiwanie serwerów*** mówi jak długo, w trakcie wyszukiwania zasobów w sieci, należy czekać na zgłoszenia serwerów:

Architekt > *Obszary i komputery* > moduł *Dane archiwalne* > zakładka *Wyszukiwanie serwerów*

1.3.6.6. Parametry archiwizacji zmiennych

Parametry archiwizacji są pobierane z bazy definicji zmiennych.

Parametry archiwizacji w bazie definicji zmiennych zawarte są w polach o następujących nazwach:

Parametr	**Pole**
Nazwa zmiennej	Nazwa
Nazwa zasobu	Archiwum
Parametry archiwizacji	ParametryArchiwizacji

Opisane parametry charakteryzują sposób archiwizacji zmiennej i stanowią uzupełnienie pozostałych parametrów zadanych w bazie zmiennych.

Należy tutaj zwrócić szczególną uwagę na typ zmiennej, określony przez funkcję przeliczającą modułu Asmen. Sposób pakowania zmiennej w archiwum jest zależny od jej

typu. Dlatego **Aspad nie może archiwizować pod tą samą nazwą zmiennej danych innego typu niż już zgromadzone w archiwum**. Załóżmy, że dla zmiennej była zadeklarowana funkcja przeliczająca NIC (typu WORD). Powstało archiwum zmiennej typu WORD. Następnie funkcję przeliczającą zmieniono na NIC_INT (typu INTEGER). Aspad nie może już poprawnie archiwizować tej zmiennej, a przy starcie ostrzega o różnicach typu danych zadeklarowanych w module Asmen i typu danych zarchiwizowanych.

Deklaracja parametrów archiwizacji zmiennych ma następującą postać:

NZm, RArc, Okres, Hor, Dt, Dx, Atr, War, PrzedWar, PoWar

gdzie:

NZm - nazwa zmiennej zgodna z deklaracją w konfiguracji modułu Asmen;
RArc - jednoliterowe oznaczenie rodzaju archiwizacji;
Okres - okres próbkowania wartości zmiennej; powinien on odpowiadać rzeczywistym potrzebom, gdyż zbyt krótki okres spowoduje zwiększenie zajętości dysku oraz większe obciążenie czasowe systemu, a przy zbyt długim okresie nie zostanie oddany w pełni charakter przebiegu wartości;
Hor - minimalny horyzont czasowy przechowywania danych dla zmiennej typu H i B; wartość 0 oznacza nieskończoność, co jest wartością domyślną; zaleca się jednak określanie skończonego horyzontu, bo pozostawienie wartości 0 może doprowadzić do przekroczenia limitu 2GB dla plików archiwum H i B;
Dt - dokładność rejestracji czasu, np. 30s oznacza, że dane pobrane zarówno o 14:27:45 jak i o 14:28:14 zostaną zarejestrowane z czasem 14:28:00; podanie możliwie dużej wartości *dt* pozwoli na lepsze upakowanie danych w plikach archiwalnych; minimalne *dt* wynosi 1;
Dx - dokładność rejestracji danych, która określa, przy jakich zmianach wartości zmiennej należy zapamiętać nową wartość; przy mniejszych zmianach kolejne wartości nie będą rejestrowane jako nieistotne, a przy odczycie będzie w tym miejscu obowiązywała poprzednia wartość zapisana w archiwum, np. 5 oznacza, że należy rejestrować nowe wartości tylko wtedy, gdy zmiana w stosunku do poprzednio zarejestrowanej jest większa lub równa 5. 0 będzie oznaczało rejestrację przy każdej zmianie, z pominięciem rejestracji wartości stałej; podanie możliwie dużej wartości *dx* pozwoli na lepsze upakowanie danych w plikach archiwalnych;
Atr - **patrz:** *Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Konfiguracja i uruchamianie modułu Aspad > 1.3.6.8. Atrybuty archiwizacji*; można podać kilka atrybutów oddzielonych pionową kreską | ;
War - nazwa opcjonalnego warunku archiwizacji (**patrz:** *Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > 1.3.7. Archiwizacja warunkowa*);Archiwizacja_warunkowa;. definicje warunków archiwizacji znajdują się w plikach lub mogą być deklarowane bezpośrednio w parametrze *War*;
PrzedWar - okres przed wystąpieniem warunku *War*, z którego dane mają być zapisane w archiwum; ten parametr nie powinien być zbyt duży, gdyż wymaga to tworzenia odpowiednich buforów w pamięci operacyjnej, obejmujących liczbę próbek danych przypadających na zadany okres;
PoWar - okres po zaniku warunku *War*, z którego dane mają być zapisane w archiwum.

Poszczególne pola deklaracji oddzielone są przecinkami. Dwa przecinki nie rozdzielone wartością parametru oznaczają przyjęcie wartości domyślnej. Pominięcie kilku ostatnich parametrów przypisze im wartości domyślne.

UWAGA: Poza **archiwizacją podstawową** istnieje również możliwość zadeklarowania dla danej zmiennej **dodatkowej archiwizacji** (w ramach tego samego archiwum) - patrz: *Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Konfiguracja i uruchamianie modułu Aspad > 1.3.6.7. Archiwizacja podstawowa i dodatkowa*. Deklaracja parametrów dodatkowej archiwizacji umieszczana jest w bazie definicji zmiennych w polu *ParametryArchiwizacji* zaraz za deklaracją podstawowej archiwizacji (deklaracje obu archiwizacji należy oddzielić średnikiem). Składnia archiwizacji dodatkowej jest taka sama jak dla archiwizacji podstawowej.

PRZYKŁAD

Deklaracja parametrów archiwizacji zmiennych.

ZKZ_kwas_BAZA.xls

	A	F	S	T
1	Nazwa	Opis	Archiwum	ParametryArchiwizacji
2	KW_A000	Temperatura spalin przed odemglaczem	KWAS	D, 1s, , 1s, 0, nie_odtwarzaj
3	KW_A004	Temperatura kwasu siarkowego	KWAS	D, 1s, , 1s, 5
4	KW_A008	Temperatura wody ciepłej	KWAS	M, 1s, , 1s, 0
5	KW_A032	Temperatura H2S przed piecem	KWAS	H, 1s, 1g, 1s, 0

BAZA_KW_pomiary / BAZA_KW_napedy / BAZA_KW_D

Rysunek: Przykładowa deklaracja zmiennych w bazie danych.

```
# Zmienną KW_A000 zapisuj w archiwum typu D,
# archiwum typu D (pliki dobowe),
# dane pobieraj co 1 sekundę,
# czas rejestruj z dokładnością do 1 sekundy,
# rejestruj każdą zmianę wartości zmiennej,
# nie inicjuj wartości zmiennej Asmena.
KW_A000, D, 1s, , 1s, 0, nie_odtwarzaj

# Zmienną KW_A004 zapisuj w archiwum typu D,
# dane pobieraj co 1 sekundę,
# czas rejestruj z dokładnością do 1 sekundy,
# rejestruj tylko gdy wartość zmieni się o więcej niż 5.
KW_A004, D, 1s, , 1s, 5

# Zmienną KW_A008 zapisuj w archiwum typu M,
# dane pobieraj co 1 sekundę,
# przechowuj przez 1 godzinę,
# czas rejestruj z dokładnością do 1 sekundy,
# rejestruj każdą zmianę wartości zmiennej.
```

```
KW_A008, M, 1s, , 1s, 0

# Zmienną KW_A008 zapisuj w archiwum typu H,
# archiwum typu H,
# dane pobieraj co 1 sekundę,
# przechowuj przez 1 godzinę,
# czas rejestruj z dokładnością do 1 sekundy,
# rejestruj każdą zmianę wartości zmiennej,
# nie inicjuj wartości zmiennej Asmena.
KW_A032, H, 1s, 1g, 1s, 0, nie_odtwarzaj
```

1.3.6.7 Archiwizacja podstawowa i dodatkowa

W ramach danego archiwum (pole ***Archiwum*** w bazie definicji zmiennych) zmienna może podlegać dodatkowej archiwizacji w sytuacji, gdy oprócz archiwów przechowywanych w plikach np. dobowych, wymagana jest archiwizacja danych w bazie SQL. W innym przypadku może zachodzić potrzeba przeanalizowania dla danego wycinka czasu danych archiwizowanych z większą częstotliwością niż przewiduje to zasadnicza archiwizacja - wówczas również należy skorzystać z archiwizacji dodatkowej.

Sposób deklarowania archiwizacji dodatkowej dla danej zmiennej wymaga następujących czynności:

1. Dla danego zasobu (archiwum), w którym ma być prowadzona dodatkowa archiwizacja - zadeklarowania stosownego rodzaju archiwum (STANDARD, MS SQL), adekwatnego do planowanego typu dodatkowej archiwizacji (D, M, Y, H, B, P):

   program Architekt > *Dane archiwalne* > <*nazwa archiwum*> > zakładka *Standardowe*

   W sytuacji, gdy oprócz archiwów przechowywanych w plikach np. dobowych (w ramach podstawowej archiwizacji), wymagana jest archiwizacja danych w bazie SQL (jako archiwizacja dodatkowa) - należy zaznaczyć 2 rodzaje archiwum: STANDARD i MS SQL.

2. W definicji archiwizowanej zmiennej (w bazie definicji zmiennych) należy w polu *Parametry archiwizacji* zadeklarować parametry archiwizacji dodatkowej. Deklaracja parametrów dodatkowej archiwizacji umieszczana jest za deklaracją podstawowej archiwizacji (deklaracje obu archiwizacji należy oddzielić średnikiem).

   Składnia archiwizacji dodatkowej jest taka sama jak dla archiwizacji podstawowej (**patrz:** Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Konfiguracja i uruchomienie modułu Aspad > *1.3.6.6. Parametry archiwizacji zmiennych*).

   W przypadku deklarowania parametrów archiwizacji dodatkowej przy użyciu edytora parametrów archiwizacji, w oknie głównym edytora należy wybrać jako rodzaj archiwizacji opcję: 'Archiwizacja podstawowa i dodatkowa' i zdefiniować parametry archiwizacji dodatkowej na osobnej zakładce edytora (**patrz:**

Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Konfiguracja i uruchomienie modułu Aspad > *1.3.6.6. Parametry archiwizacji zmiennych*;
Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Konfiguracja i uruchomienie modułu Aspad > *1.3.6.9. Edytor parametrów archiwizacji zmiennych*).

1.3.6.8. Atrybuty archiwizacji

ODTWARZAJ

Na starcie Aspad będzie wpisywał do bazy danych modułu Asmen ostatnią wartość zarejestrowaną w archiwum. Pozwoli to na przechowywanie i odtwarzanie wartości zadawanych przez operatora systemu Asix także w okresie wyłączenia komputera. Atrybut przeznaczony generalnie dla zmiennych należących do kanału transmisji Asmena typu NONE (brak odświeżania zmiennej poprzez odczyt ze sterownika). Od momentu startu modułu Aspad wartości zadane przed wyłączeniem będą dostępne dla wszystkich użytkowników modułu Asmen w danym systemie.

NIE_ODTWARZAJ odwołuje deklarację ODTWARZAJ.

NIE_PAKUJ

Atrybut NIE_PAKUJ zmienia dwie cechy archiwizacji.

- W archiwum w każdym cyklu archiwizacji nastąpi zapis wartości, pod warunkiem, że uda się pozyskać poprawną wartość. Nie będą więc pomijane w kolejnych cyklach zapisy wartości stałej.
- Przy odczycie Aspad będzie zorientowany na odczyt punktów, a nie odcinków. W zwykłym trybie Aspad udostępnia odcinki, co oznacza, że jeżeli wartość została zapisana z czasem 1:00 i jest ważna przez godzinę (cykl próbkowania wynosi 1 godz.), to przy odczycie o godzinie 2:00 pojawi się nowa wartość, a jeżeli jej nie ma (dziura w archiwum), to zostanie tam powtórzona wartość z 1:00. Pozwala to na narysowanie odcinka od godziny 1:00 do 2:00 i pokazanie dziury w archiwum dopiero od 2:00. Atrybut NIE_PAKUJ spowoduje odczyt tylko wartości z 1:00.

PAKUJ odwołuje deklarację NIE_PAKUJ.

KAŻDY

Atrybut KAŻDY może być stosowany tylko z archiwum typu B. Zmienna musi być przy tym pozyskiwana z kanału, którego drajwer udostępnia dodatkowe usługi pozwalające na pobieranie każdej przychodzącej wartości. Aktualnie takie możliwości ma drajwer SINAUT.

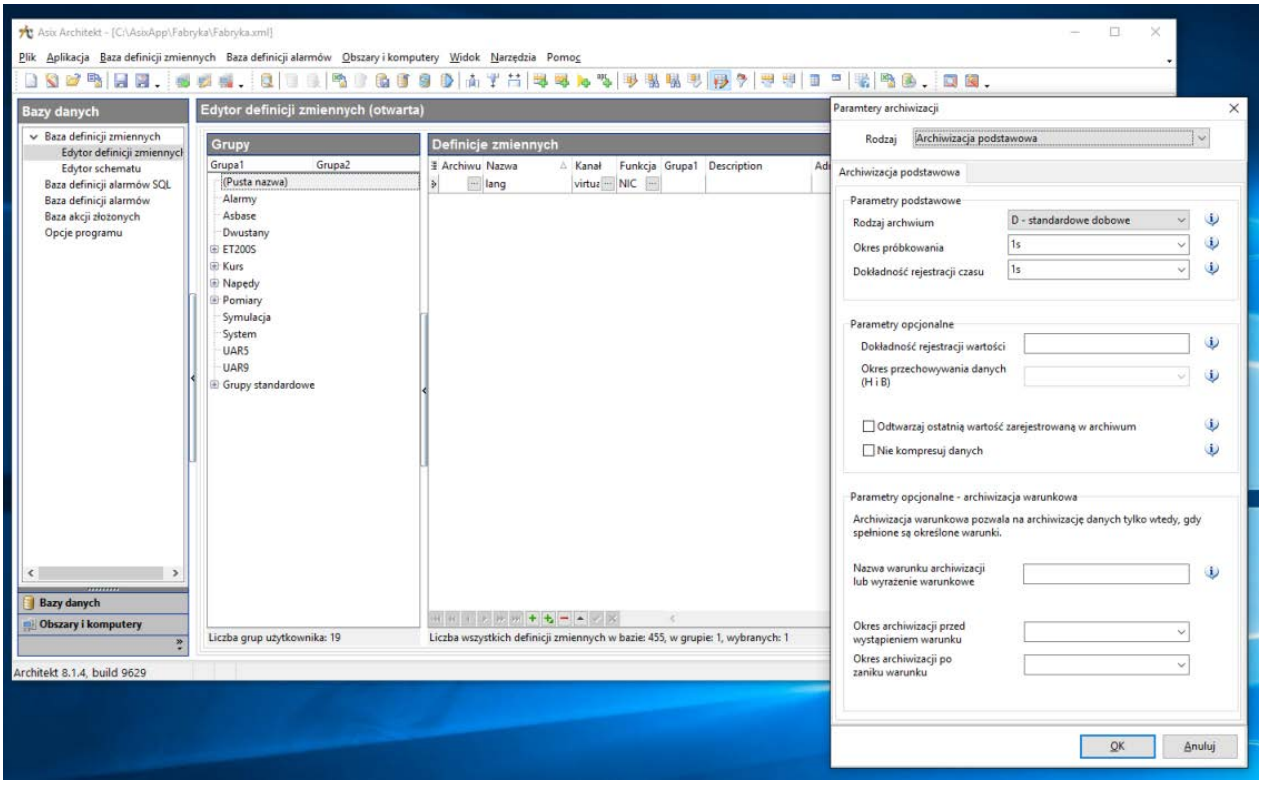
Atrybut KAŻDY spowoduje zapis do archiwum B wszystkich wartości pobranych przez drajwer, niezależnie od cyklu próbkowania i kolejności ich udostępniania. Cykl próbkowania oznacza wtedy jedynie czas ważności pobranych danych.

NIE_KAŻDY odwołuje deklarację KAŻDY.

1.3.6.9. Deklaracja archiwizacji zmiennych - edytor parametrów

W przypadku tworzenia / definiowania bazy definicji zmiennych przy użyciu programu Architekt definiowanie parametrów archiwizacji umożliwia '*Edytor parametrów archiwizacji zmiennych*', dostępny w:

program Architekt > *Bazy danych* > *Edytor definicji zmiennych* (przy załączonym trybie Edytor tworzenia bazy i otwartej bazie definicji zmiennych) > pole *Parametry archiwizacji* > edytor parametrów archiwizacji



Rys. Edytor parametrów archiwizacji.

Wybór rodzaju archiwizacji odbywa się za pośrednictwem pola ***Rodzaj*** - dostępne opcje to:

- Bez archiwizacji;
- Archiwizacja podstawowa;
- Archiwizacja podstawowa i dodatkowa

Podstawowe parametry:

Pole ***Rodzaj archiwum*** określa typ archiwum:

Archiwum typu **D** (ang. day) - dane gromadzone w plikach dobowych;
Archiwum typu **M** (ang. month) - dane gromadzone w plikach miesięcznych;
Archiwum typu **Y** (ang. year) - dane gromadzone w plikach rocznych;
Archiwum typu **H** (ang. horizon) - dane gromadzone w jednym pliku;

Archiwum typu **B** (Baza Danych) SQL;	- dane gromadzone w bazie danych MS
Archiwum typu **P** (ang. pattern)	- dane wzorcowe;
Archiwum typu **X** SQL;	- dane gromadzone w bazie danych MS

Archiwa typu **D**, **M**, **Y**, **H** wymagają archiwum STANDARD. Stąd zmienna, która w parametrach archiwizacji ma zadeklarowany jeden z wymienionych typów, musi mieć w parametrach archiwum zadeklarowany jako rodzaj archiwum: **STANDARD** (Architekt > *Dane archiwalne* > <*nazwa archiwum*> > zakładka *Standardowe*).

Pozostałe typy archiwum **B, X** i **P** deklarowane są dla archiwum **MS SQL**.

Pozostałe pola edytora umożliwiają zadeklarowanie następujących parametrów archiwizacji:

- ***Okres próbkowania***;
- ***Dokładność rejestracji czasu***;

Parametry opcjonalne:

- ***Dokładność rejestracji wartości***;
- ***Okres przechowywania danych (H i B)***;
- ***Odtwarzaj ostatnią wartość zarejestrowaną w archiwum***;
- ***Nie kompresuj danych***;

Opcjonalne - archiwizacja warunkowa:

- ***Nazwa warunku archiwizacji***;
- ***Okres archiwizacji przed wystąpieniem warunku***;
- ***Okres archiwizacji po zaniku warunku***;

1.3.7. Archiwizacja warunkowa

Archiwizacja warunkowa pozwala na archiwizację tylko wtedy, gdy spełnione są określone warunki. Można zadawać, z jakiego okresu przed wystąpieniem warunku i po jego zaniku dane mają zostać zapamiętane w archiwum.

Warunki mogą być definiowane bezpośrednio w definicji zmiennej w bazie definicji zmiennej w polu *Warunki archiwizacji* lub znajdować się w osobnym pliku warunków.

1.3.7.1. Deklaracje warunków archiwizacji

Deklaracja w plikach

Warunki archiwizacji są definiowane przy pomocy wyrażeń umieszczanych w pliku. Każde wyrażenie musi być zdefiniowane w odrębnej linii. Nazwa warunku podana w pliku jest następnie umieszczana w definicji zmiennej w polu *Parametry archiwizacji*. Nazwę pliku z warunkami archiwizacji należy zadeklarować w: programie Architekt > *Obszary i komputery* > *Dane archiwalne* > zakładka *Standardowe* > opcja ***Warunki***

Wartości i argumenty wyrażeń są stałymi lub zmiennymi wartościami zmiennoprzecinkowymi, maskami 32-bitowymi, lub wartościami logicznymi. Argumentami wyrażeń mogą być inne wyrażenia, czyli można definiować wyrażenia złożone. Nazwy wyrażeń nie mogą się powtarzać.

Jeżeli wartość wyrażenia nie jest wartością logiczną, a zostanie użyta bezpośrednio jako warunek, to jest dokonywana konwersja:

- maski 32-bitowe, w których wszystkie bity są zerowe są konwertowane na fałsz, inne na prawdę,
- liczby z zakresu (-0.5;0.5) są konwertowane na fałsz, inne na prawdę.

Linia wyrażenia ma następującą strukturę:

Nazwa_wyrażenia, Operator, Arg1, Arg2, ...

gdzie:

Nazwa_wyrazenia - identyfikator wyrażenia zawierający do 15 znaków; nazwa wyrażenia może się powtarzać; nazwy wyrażeń stanowią odrębną grupę, dzięki czemu nie dochodzi do konfliktu np. z identyczną nazwą zmiennej;
Operator - nazwa jednego z dopuszczalnych operatorów;
Arg1, Arg2, ... - argumenty wyrażenia.

Argumenty mają postać:

[*u*][*$*]*źródło*

gdzie:

u i **$** - są opcjonalne i oznaczają:
u - jest jednym z operatorów jednoargumentowych:
- - oznacza zmianę znaku z wcześniejszą konwersją do postaci liczby zmiennoprzecinkowej,
~ - oznacza negację bitów maski z wcześniejszą konwersją do postaci maski 32-bitowej,
! - oznacza negację z wcześniejszą konwersją do postaci wartości logicznej.
$ - oznacza pobranie wartości wyliczonej przy poprzednim wartościowaniu warunku dla danej zmiennej i zachowanie aktualnej wartości do przyszłych wyliczeń.

Źródło danych *źródło* ma jedną z następujących postaci:

liczba_dziesiętna	- wartość liczbowa zmiennoprzecinkowa (np. 9999);
0x*liczba_szesnastkowa*	- maska 32-bitowa zakodowana szesnastkowo (np. 0xFFFF);
>>	- plus nieskończoność;
<<	- **m**inus nieskończoność;
@nazwa_zmiennej	- wartość zmiennej procesowej;
@	- wartość zmiennej procesowej, dla której aktualnie wyznaczany jest warunek; taki warunek ma różne wartości w zależności od tego, dla której zmiennej jest wyznaczany;
?*nazwa_wyrażenia*	- *w*artość innego, wcześniej zdefiniowanego wyrażenia. Można w ten sposób definiować wyrażenia złożone.

Deklaracja in-line warunku archiwizacji

W skład definicji zmiennej może wchodzić warunek archiwizacji tej zmiennej. Warunek umieszczany jest w polu *Archiwizacja zmiennej*, w parametrze *War*.

Ponieważ większość warunków ma prostą jednowierszową postać, to w wielu aplikacjach pozwoli to uniknąć tworzenia pliku warunków.
Takie warunki można wprowadzać w edytorze bazy definicji zmiennych Architekta. Nie można ich stosować z wcześniejszymi wersjami Architekta, gdyż nieprawidłowo interpretuje on dodatkowe przecinki w definicji zmiennej.

Deklarację in-line można zdefiniować na 2 niżej opisane sposoby.

Warunek jako wyrażenie w deklaracji zmiennej

Zamiast nazwy warunku można wpisać bezpośrednio wyrażenie, zgodne ze składnią definicji warunku w pliku warunków, ujęte w nawiasy (). W takim wyrażeniu pomija się pierwszy parametr określający nazwę warunku.

Przykłady

Archiwizuj, gdy bit 4 zmiennej AAA jest równy 1:
(BAND, @AAA, 0x10)

Archiwizuj, gdy bity 4 lub 8 zmiennej AAA zmieniły stan:
(BNE, @AAA, $@AAA, 0x110)

Archiwizuj, gdy bit 0 zmiennej archiwizowanej zmienia stan z 0 na 1:
(BGT, @, $@, 0x1)

Archiwizuj, gdy wartość zmiennej archiwizowanej jest poza przedziałem (20,80):
(IN, @, <<, 20, 80, >>)

Warunek jako argument w deklaracji zmiennej

Zamiast nazwy warunku można wpisać bezpośrednio wyrażenie, zgodne ze składnią argumentu warunku w pliku warunków. Takie wyrażenie musi się zaczynać od !, -, ~, $, @ lub ?.

Przykłady

Archiwizuj, gdy zmienna AAA jest poza przedziałem (-0.5; 0.5):
@AAA

Archiwizuj, gdy zmienna AAA jest w przedziale (-0.5; 0.5):
!@AAA

Archiwizuj, gdy poprzednia wartość zmiennej AAA była poza przedziałem (-0.5; 0.5):
$@AAA

Archiwizuj, gdy warunek QQQ nie jest spełniony:
!?QQQ

1.3.7.2. Operatory warunków archiwizacji

Operatory logiczne
Operatory relacyjne
Operatory logiczne bitowe
Operatory relacyjne bitowe
Operatory arytmetyczne
Operator testu na wartość mieszczącą się w zadanych przedziałach

Operatory warunków mogą mieć różny typ wyniku oraz różną liczbę i typ argumentów, w zależności od rodzaju operatora. W razie niezgodności typu argumentu z typem akceptowanym przez operator, dokonywana jest automatycznie konwersja. Niżej opisano zdefiniowane do tej pory operatory.

Operatory logiczne

OR	suma logiczna
AND	iloczyn logiczny
XOR	logiczny iloczyn wykluczający
Typ argumentów:	wartość logiczna
Typ wyniku:	wartość logiczna
Liczba argumentów:	dowolna

Operatory relacyjne

LT	mniejszy
LE	mniejszy lub równy
EQ	równy
NE	różny

GE większy lub równy
GT większy
Typ argumentów: wartość liczbowa
Typ wyniku: wartość logiczna
Liczba argumentów: 2

Operatory logiczne bitowe

BAND iloczyn logiczny wszystkich bitów argumentów
BOR suma logiczna wszystkich bitów argumentów
BXOR iloczyn wykluczający wszystkich bitów argumentów
Typ argumentów: maska 32-bitowa
Typ wyniku: maska 32-bitowa
Liczba argumentów: dowolna

Operatory relacyjne bitowe

BLT ustawia bity zadane w *Arg3*, które w *Arg1* są < od bitów w *Arg2*
BLE ustawia bity zadane w *Arg3*, które w *Arg1* są < lub = bitom w *Arg2*
BEQ ustawia bity zadane w *Arg3*, które w *Arg1* są = bitom w *Arg2*
BNE ustawia bity zadane w *Arg3*, które w *Arg1* nie są = bitom w *Arg2*
BGE ustawia bity zadane w *Arg3*, które w *Arg1* są > lub = bitom w *Arg2*
BGT ustawia bity zadane w *Arg3*, które w *Arg1* są > lub = bitom w *Arg2*
Typ argumentów: maska 32-bitowa
Typ wyniku: maska 32-bitowa
Liczba argumentów: 2 lub 3. Jeżeli podano tylko 2 argumenty, to zakłada się, że *Arg3*=0xFFFFFFFF.

Operatory arytmetyczne

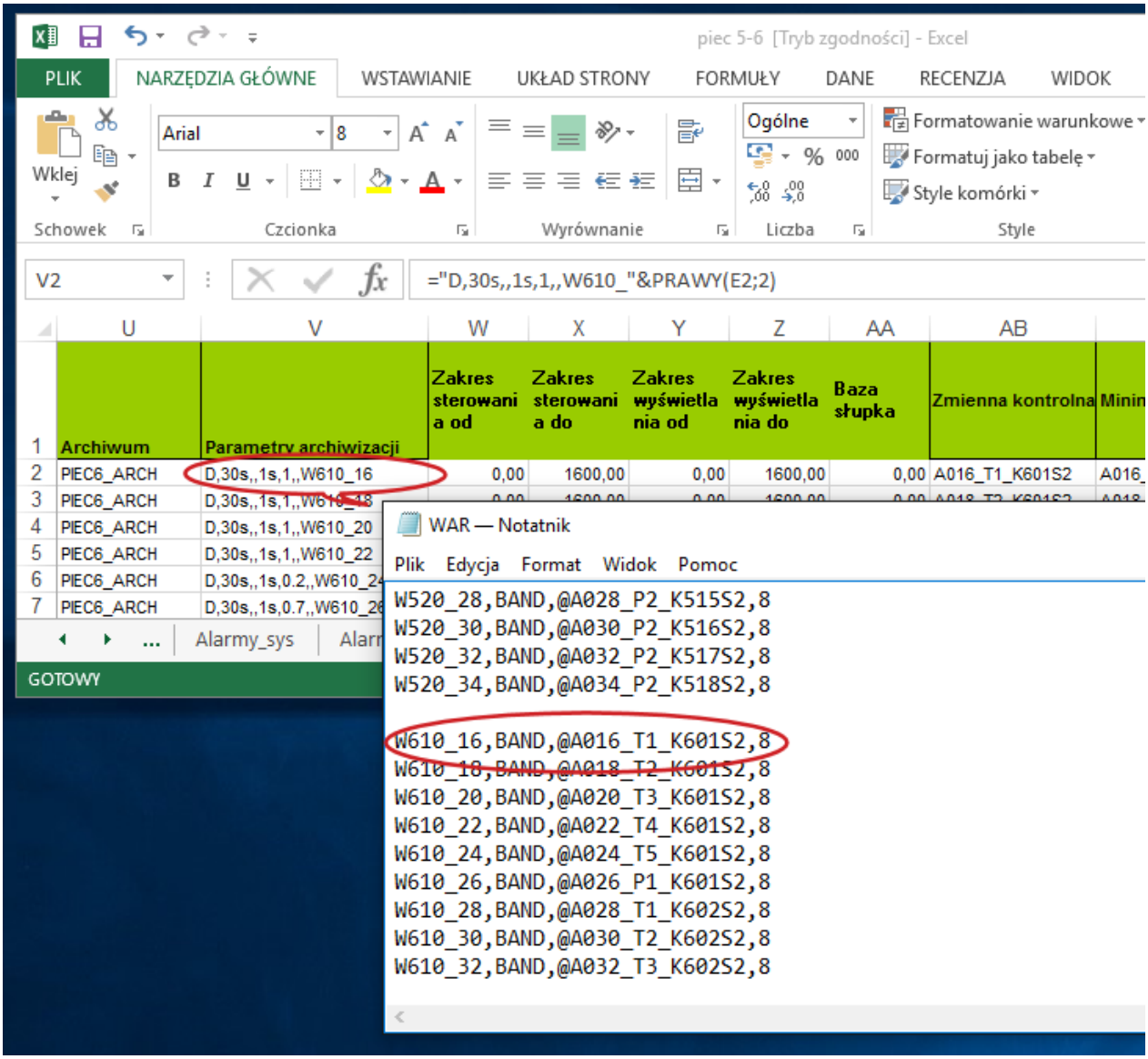
ADD suma arytmetyczna
MUL iloczyn arytmetyczny
DIV iloraz arytmetyczny *Arg1* / *Arg2*.
Typ argumentów: wartość liczbowa
Typ wyniku: wartość liczbowa
Liczba argumentów: ADD i MUL - dowolna, DIV - 2.

Operator testu na wartość mieszczącą się w zadanych przedziałach

IN spełniony, gdy wartość Arg1 mieści się w przedziale [*Arg2*, *Arg3*], lub w przedziale [*Arg4*, *Arg5*] itd.
Typ argumentów: wartość liczbowa
Typ wyniku: wartość logiczna
Liczba argumentów: nieparzysta nie mniejsza niż 3.

1.3.7.3. Przykłady warunków

Archiwizuj, gdy bity 4. lub 8. zmiennej AAA zmieniły stan:
WARUNEK, BNE, @AAA, $@AAA, 0x110
Archiwizuj, gdy bit 0. zmiennej archiwizowanej zmienia stan z 0 na 1:
WARUNEK, BGT, @, $@, 0x1
Archiwizuj, gdy wartość zmiennej archiwizowanej jest poza przedziałem (20,80) i ustawiony jest jeden z bitów 0-3 zmiennej WWW:
W1, IN, @, <<, 20, 80, >>
W2, BAND, @WWW, 0xF
WARUNEK, AND, ?W1, ?W2
Archiwizuj w chwili, gdy wartość zmiennej archiwizowanej przekracza granicę 100:
W3, GT, @, 100
WARUNEK, AND, ?W3, !$?W3



Rys. Przykład pliku z arunkami archiwizacji.

1.3.7.4. Deklaracja in-line warunku archiwizacji

W skład deklaracji zmiennej może wchodzić warunek archiwizacji tej zmiennej. Do tej pory wpisywało się tam nazwę warunku, którego definicja znajdowała się w osobnym pliku warunków. Aspad 7.12 dostępny od wersji Asix 6.0.2 umożliwia zdefiniowanie warunku bezpośrednio w deklaracji zmiennej (deklaracja in-line).
Ponieważ większość warunków ma prostą jednowierszową postać, to w wielu aplikacjach pozwoli to uniknąć tworzenia pliku warunków.

Takie warunki można wprowadzać w edytorze bazy definicji zmiennych Architekta. Nie można ich stosować z wcześniejszymi wersjami Architekta, gdyż nieprawidłowo interpretuje on dodatkowe przecinki w definicji zmiennej.

Deklarację in-line można zdefiniować na 2 niżej opisane sposoby.

Warunek jako wyrażenie w deklaracji zmiennej

Zamiast nazwy warunku można wpisać bezpośrednio wyrażenie, zgodne ze składnią definicji warunku w pliku warunków, ujęte w nawiasy (). W takim wyrażeniu pomija się pierwszy parametr określający nazwę warunku.

Przykłady

Archiwizuj, gdy bit 4 zmiennej AAA jest równy 1:
(BAND, @AAA, 0x10)

Archiwizuj, gdy bity 4 lub 8 zmiennej AAA zmieniły stan:
(BNE, @AAA, $@AAA, 0x110)

Archiwizuj, gdy bit 0 zmiennej archiwizowanej zmienia stan z 0 na 1:
(BGT, @, $@, 0x1)

Archiwizuj, gdy wartość zmiennej archiwizowanej jest poza przedziałem (20,80):
(IN, @, <<, 20, 80, >>)

Warunek jako argument w deklaracji zmiennej

Zamiast nazwy warunku można wpisać bezpośrednio wyrażenie, zgodne ze składnią argumentu warunku w pliku warunków. Takie wyrażenie musi się zaczynać od !, -, ~, $, @ lub ?.

Przykłady

Archiwizuj, gdy zmienna AAA jest poza przedziałem (-0.5; 0.5):
@AAA

Archiwizuj, gdy zmienna AAA jest w przedziale (-0.5; 0.5):
!@AAA

Archiwizuj, gdy poprzednia wartość zmiennej AAA była poza przedziałem (-0.5; 0.5):

$@AAA

Archiwizuj, gdy warunek QQQ nie jest spełniony:
!?QQQ

1.3.8. Archiwizacja typu B

1.3.8.1. Specyficzne cechy archiwizacji typu B

Archiwum typu B pozwala na wykorzystanie typowych **baz danych** do przechowywania archiwum Aspada. Z jednej strony pozwala to na bardziej elastyczne wykorzystanie takiego archiwum i stwarza możliwość korzystania z niego także w programach nie należących do pakietu Asix. Jest także jedynym rodzajem archiwizacji, w którym możliwe jest **wypełnienie dziur** powstałych w archiwum. Z drugiej strony nie jest w nim stosowana żadna forma kompresji danych lub pomijania nieistotnych zmian wartości, a każdy rekord zajmuje ponad **100 bajtów** (uwzględniając pliki indeksowe).

Asix umożliwia przechowywanie archiwów w bazach danych typu SQL.

Ponieważ zapis i odczyt archiwum B jest wolniejszy, a wielkość plików większa niż archiwum D czy M, rozsądnym rozwiązaniem jest stosowanie np. archiwum D do rejestracji przebiegów dla wykresów, a archiwum B do przechowywania danych zagregowanych (np. średnie godzinne do raportów).

Archiwum B umożliwia ponadto:

- archiwizację dodatkowych informacji o archiwizowanych punktach,
- archiwizację poszczególnych grup zmiennych w wielu bazach danych,
- archiwizację niesekwencyjną (wartości mogą przychodzić w dowolnej kolejności),
- automatyczne tworzenie kartoteki archiwum i nazwy bazy danych w środowisku SQL,
- przeglądanie zawartości baz danych archiwum B za pomocą programu BBrowserSQL lub standardowych narzędzi przeznaczonych do przeglądania baz danych typu SQL.

1.3.8.2. Dodatkowe wymagania archiwum typu B

Realizacja archiwizacji danych z wykorzystaniem bazy typu SQL

Oprogramowanie systemowe:

- Realizacja połączeń z bazami SQL wymaga:
- MDAC w wersji 2.8 lub nowszego po stronie klienta bazy danych (stacja z systemem Asix) oraz (standardowo jest składnikiem systemów Windows 7 i Windows 2003 i nowszych),
- Microsoft SQL Server 2008 lub nowszy po stronie serwera bazy danych,
- użytkownik bazy danych musi mieć konto na stacji serwera SQL, zapewniające mu dostęp do zapisu do bazy danych,
- dysk z bazą danych nie może mieć włączonej kompresji.

Oprogramowanie Asix:
Dostęp do bazy SQL realizują Aspad.dll wraz z AsSQL.dll.

Aby możliwa była archiwizacja typu B, należy załadować dodatkową bibliotekę AsSQL.dll poprzez zadeklarowanie archiwum rodzaju MS SQL przy użyciu programu Architekt.

Architekt > *Obszary i komputery* > moduł *Dane archiwalne* > dodana definicja archiwum z następującymi parametrami:
- *Nazwa* zasobu,
- *Rodzaj archiwum*: MS SQL,
- dodatkowe parametry.

Uprawnienia użytkownika:
Uprawnienia użytkownika są weryfikowane na poziomie Windows. Dlatego bieżący użytkownik musi mieć na komputerze serwera SQL nadane uprawnienia dostępu do bazy danych.

Przy pierwszym uruchomieniu baza danych jest tworzona - a użytkownik w tym momencie musi mieć także uprawnienia do tworzenia baz danych oraz użytkowników.

Uwagi te odnoszą się do wszystkich baz danych deklarowanych w pliku konfiguracyjnym Serwera Danych, także do wykorzystywanych tylko w deklaracji synchronizacji (***Patrz:*** *Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych > Parametryzacja archiwum* > rozdz. *2.10.9.6. Deklaracja synchronizacji zawartości archiwum typu B z danymi w bazie MS SQL*).

Deklaracja udziału na serwerze:

UWAGA:
W wersjach Asix 9.0.1 lub nowszych z MS SQL Server 2008 lub nowszym, udział ASSQL jest zbędny.

Zapisy do archiwum oraz synchronizacja wymagają utworzenia na komputerze serwera, do którego zapis jest wykonywany, katalogu tymczasowego, udostępnianego jako udział o nazwie **ASSQL**. Musi on być udostępniony do zapisu dla wszystkich użytkowników, na konto których zapisy będą dokonywane. Katalog ten jest wykorzystywany w celu wymiany plików, pozwalających na wielokrotne zwiększenie szybkości zapisu do bazy.

1.3.8.3. Uzupełnianie danych w archiwum B ze sterownika

Archiwum B pozwala na uzupełnianie danych w dziurach powstałych na skutek braku łączności ze źródłem danych lub wyłączenia komputera. Aby takie uzupełnienie mogło nastąpić, konieczne jest, aby urządzenie będące źródłem danych, udostępniało oprócz danych bieżących także dane historyczne. Także Asmen z drajwerem tego urządzenia muszą udostępniać możliwość sięgnięcia do tych danych. Dane historyczne udostępniane są przez następujące drajwery: BAZA, DMS285, DMS500, Talas, Macmat, DataPAF, Mevas.

Parametry ***Czas pierwszego pobierania*** i ***Okres pobierania*** pozwalają określić, kiedy należy dokonać takiego uzupełnienia:

Architekt > *Obszary i komputery* > moduł *Dane archiwalne* > zakładka *Pobieranie danych historycznych*

1.3.8.4. Synchronizacja archiwum z bliźniaczą bazą danych

Funkcja synchronizacji archiwum z bliźniaczą bazą danych pozwala na realizację uzupełniania danych w trakcie działania systemu, uwzględniającą uzupełnianie danych historycznych i ręczną edycję danych.

Synchronizacja powoduje kopiowanie przyrostowe. Kopiowane są tylko dane, które zostały dopisane lub zmienione w archiwum źródłowym od ostatniego kopiowania. W przypadku konfliktu z już istniejącymi danymi w archiwum docelowym znajdą się dane o lepszej jakości. W przypadku danych o tej samej jakości preferowane są dane wprowadzane ręcznie.

Można skonfigurować wiele deklaracji synchronizacji, także dla jednego zasobu, co pozwala np. na uzupełnianie danych z kilku bliźniaczych serwerów.

Szczegóły dotyczące deklaracji synchronizacji znajdują się w:

Patrz: Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych > Parametryzacja archiwum > rozdz. *2.10.9.6. Deklaracja synchronizacji zawartości archiwum typu B z danymi w bazie MS SQL*

Porady dotyczące wykorzystania deklaracji synchronizacji

Deklaracja synchronizacji może znaleźć kilka zastosowań. Niżej zostaną pokazane przykłady zastosowania w celu:

- wzajemnego uzupełniania danych stacji operatorskich,
- budowy archiwum nadążnego na serwerze pośredniczącym,
- wykonania kopii archiwum.

Wzajemne uzupełnianie danych serwerów

Deklaracja synchronizacji pozwala na wzajemne uzupełnianie danych między serwerami, które ma kilka zalet w stosunku do typowego uzupełniania danych w module Aspad:

- pozwala na uzupełnianie danych w trakcie pracy systemu, a nie tylko na starcie,
- uzupełnia wszystkie zmiany w archiwum, włączając w to wprowadzone w trakcie pracy systemu dane historyczne i wpisywane ręcznie.

Budowa archiwum nadążnego na serwerze pośredniczącym

Na serwerze pośredniczącym można zadeklarować zasób tylko do odczytu, a następnie jego synchronizację ze stacji operatorskich. Taki serwer nie będzie pobierał danych bieżących z modułu ASMEN. Dane będą cyklicznie pozyskiwane ze stacji operatorskich przy użyciu synchronizacji baz danych. Synchronizacja obejmie wszystkie dopisane dane, także historyczne.

Wykonanie kopii archiwum

Deklarację synchronizacji można stosować do wykonania kopii zapasowej. Kopia powinna być umieszczona na innej stacji niż oryginał. Kopia będzie w pełni funkcjonalną bazą danych.

Na serwerze SQL, na którym tworzona jest kopia, nie musi w tym wypadku działać żaden Asix, a jedynym celem operacji synchronizacji jest uzyskanie zapasowej bazy danych.

Takiego rodzaju kopii można używać tylko w ograniczonym zakresie, gdyż w stosunku do systemowej kopii zapasowej SQL ma ona kilka wad:

- ma duże rozmiary, gdyż zawiera pliki logu transakcji, pliki indeksowe i wolne obszary, wynikające ze sposobu rozmieszczania informacji w serwerze SQL,
- nie można jej tworzyć na dysku z kompresją.

Wymagania

Deklaracja udziału na komputerze serwera:
Zapisy do archiwum oraz synchronizacja wymagają utworzenia na komputerze serwera, do którego zapis jest wykonywany, katalogu tymczasowego, udostępnianego jako udział o nazwie **ASSQL**. Musi on być udostępniony do zapisu dla wszystkich użytkowników, na konto których zapisy będą dokonywane. Katalog ten jest wykorzystywany w celu wymiany plików, pozwalających na wielokrotne zwiększenie szybkości zapisu do bazy.

Udział ASSQL musi zostać utworzony także wtedy, gdy baza danych znajduje się na tym samym komputerze, z którego dokonuje się zapisu.

1.3.8.5. Deklaracja bazy danych

Przy archiwizacji w bazie danych typu SQL konieczne jest zadeklarowanie odpowiedniego zasobu archiwalnego, bazy danych i nazwy serwera.

Przy pierwszym uruchomieniu baza jest automatycznie tworzona, a użytkownik w tym momencie musi mieć uprawnienia do tworzenia baz danych i użytkowników na serwerze.

Szczegóły dotyczące deklaracji bazy danych zostały omówione w:

Serwer Danych > Paraemtryzacja Serwera Danych > rozdz. *2.10.9. Konfiguracja danych archiwalnych > Parametryzacja archiwum*

1.3.8.6. Deklaracja zamiennych do archiwizacji w bazie danych

Deklaracja zmiennych do archiwizacji w bazie danych dokonywana jest przy użyciu odpowiednich parametrów archiwizacji zmiennych ( **Patrz:** *1.3.6.6. Parametry archiwizacji zmiennych*).

1.3.8.7. Dodatkowe informacje o archiwizowanych punktach

Typowo archiwizowane są wartość i czas punktu, przy czym można tak sparametryzować archiwum, aby rozszerzyć rekord o dodatkowe pola. Archiwum B pozwala na zapamiętanie następującego zestawu pól.

Archiwizacja w bazie danych SQL

Baza danych archiwum B w SQL może zawierać następujące pola:

1. Pola obowiązkowe:
   - czas podstawowy (**T**),
   - jakość danej,
   - wartość (**X**).
2. Pola opcjonalne:
   - czas dokładny danej (**XT**),
   - czas zapisu rekordu (**WT**),
   - okres ważności próbki (**VT**).

X wartość
Liczba określająca wartość zmiennej.

T czas podstawowy
Przy archiwizacji z zadanym cyklem próbkowania jest to czas początku kolejnego cyklu. Jest on wyznaczany w następujący sposób:
czas danej zaokrąglany jest do najbliższej wielokrotności zadanej dokładności rejestracji, np. przy dokładności rejestracji równej 5 minut 1:17:30 zostanie zaokrąglony do 1:20:00, 1:22:00 do 1:20:00, a 1:22:30 do 1:25:00;
tak zaokrąglony czas jest ponownie zaokrąglany w dół do wielokrotności cyklu próbkowania, np. przy cyklu 15-minutowym czas 1:20:00 i 1:25:00 będą zaokrąglone do 1:15:00.
Przy archiwizacji bez zadanego cyklu (atrybut *KAŻDY*) jest to czas danej zaokrąglany tylko do zadanej dokładności rejestracji.

XT czas danej - pole opcjonalne
Dokładny czas przypisany do zmiennej, zaokrąglony z dokładnością zadaną przy parametryzacji. Jeżeli archiwum nadano atrybut **KAŻDY**, to jest on równy czasowi podstawowemu.

WT czas zapisu - pole opcjonalne
Czas zapisu danej w bazie. Pole jest istotne w systemach z opóźnionym pozyskiwaniem danych.

VT okres ważności próbki - pole opcjonalne
Określa, czy w bazie danych będzie archiwizowany czas ważności próbki (+VT), czy

też na być pomijany, a domyślnym czasem ważności ma być okres archiwizacji (-VT); w efekcie, dla -VT, gdy zostanie w przyszłości skrócony okres archiwizacji, to dane historyczne będą oddzielone dziurami; domyślnie czas ważności jest archiwizowany. Włączenie opcji powoduje dodanie 4-bajtowego pola do każdego rekordu danych;

SV 4-bajtowe identyfikatory zmiennych - pole opcjonalne
Po zaznaczeniu identyfikatory zmiennych będą 4-bajtowe, zamiast 2-bajtowych. Opcja ma znaczenie tylko w aplikacjach, w których przewidujemy deklaracje ponad 32 767 zmiennych w pojedynczej bazie danych, wliczając w to zmienne, które zostały dodane i później usunięte. Koszt, to ponad 4 bajty na próbkę (2 bajty w tablicy i ponad 2 bajty w indeksie).

SD 8-bajtowe identyfikatory rekordów archiwizowanych danych - pole opcjonalne.
Po zaznaczeniu identyfikatory rekordów archiwizowanych danych będą 8-bajtowe, zamiast 4-bajtowych. Opcja ma znaczenie tylko w aplikacjach, w których przewidujemy możliwość zapisu ponad 2 147 483 647 próbek w pojedynczej bazie danych, wliczając w to próbki, które zostały zapisane i później usunięte oraz próbki, które zostały poprawione, np. w wyniku synchronizacji z innym archiwum (próbka o lepszej jakości zastępuje próbkę o gorszej, a licznik próbek zwiększa się o 1). Koszt, to ponad 8 bajtów na próbkę (4 bajty w tablicy i ponad 4 bajty w indeksie).

Szczegóły dotyczące parametryzacji dodatkowych informacji archiwizowanych punktów znajdują się w:

Serwer Danych > Parametryzacja Serwera Danych > Konfiguracja danych archiwalnych > Parametryzacja archiwum > 2.10.9.1. Parametry standardowe

1.3.8.8. Archiwizacja w wielu bazach danych

Możliwe jest deklarowanie dla jednej aplikacji wielu archiwów/baz danych, w których prowadzona będzie archiwizacja zmiennych procesowych. Oznacza to utworzenie wielu archiwów, z których każdy przypisany jest do innej bazy MS SQL.

1.3.8.9. Archiwizacja niesekwencyjna

Po zadaniu atrybutu KAŻDY do archiwum B będą zapisywane wszystkie pobrane przez drajwer Asmena dane, niezależnie od kolejności ich pobierania.

Patrz: Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Konfigurowanie i uruchomienie modułu Aspad > *1.3.6.8. Atrybuty archiwizacji*

1.3.8.10. Przeglądanie zawartości bazy danych archiwum B

W przypadku baz danych typu SQL - istnieje możliwość przeglądania ich zawartości przy użyciu standardowych narzędzi przeznaczonych do tego celu.

1.3.9. Przebieg wzorcowy typu P

Uwaga:
Przebiegi wzorcowe tego typu stosowane były w Asixie klasycznym. W Asix.Evo zaleca się stosowanie innych rozwiązań.

Deklaracja takiego przebiegu nie może być umieszczona w archiwum do zapisu – musi to być deklaracja archiwum do odczytu lub sieciowego.

Dla archiwum do odczytu, zawierającego przebiegi typu P, konieczna jest deklaracja bazy danych (Archiwum do odczytu zadeklarowane jako MS SQL), identycznie jak dla przebiegów typu B.

Deklaracja przebiegu wzorcowego wymaga:

1. utworzenia archiwum;
2. zadeklarowania w definicji zmiennej, która ma być archiwizowana, nazwy przebiegu w polu Nazwa, typu archiwum P w parametrach archiwizacji (pole Parametry archiwizacji) oraz nazwy utworzonego archiwum (pole Archiwum).

Nazwa przebiegu nie powinna być nazwą żadnej zmiennej Asixa.

Czas ważności danej jest dla przebiegu P zawsze ustawiany na 1 dobę, a pozostałe parametry są nieistotne.

* * *

Do generacji przebiegów wzorcowych służy program **PEdit**.

Szczegółowy opis użycia programu PEdit można znaleźć w dokumentacji Przebiegi Wzorcowe.

1.3.9.1. Deklaracja przebiegu wzorcowego

Przebieg wzorcowy jest oznaczany jako typ **P**. Deklaracja takiego przebiegu nie może być umieszczona w archiwum do zapisu - musi to być deklaracja archiwum do odczytu lub sieciowego.

Dla archiwum do odczytu, zawierającego przebiegi typu P, konieczna jest deklaracja bazy danych (Archiwum do odczytu zadeklarowane jako MSSQL), identycznie jak dla przebiegów typu B.

Deklaracja przebiegu wzorcowego wymaga:

1. utworzenia archiwum;

2. zadeklarowania w definicji zmiennej, która ma być archiwizowana, nazwy przebiegu w polu *Nazwa*, typu archiwum P w parametrach archiwizacji (pole *Parametry archiwizacji*) oraz nazwy utworzonego archiwum (pole *Archiwum*).

Nazwa przebiegu nie powinna być nazwą żadnej zmiennej Asixa.

Czas ważności danej jest dla przebiegu P zawsze ustawiany na 1 dobę, a pozostałe parametry są nieistotne.

1.3.9.2. Generacja przebiegów wzorcowych

Do generacji przebiegów wzorcowych służy program **PEdit**.

Szczegółowy opis użycia programu PEdit można znaleźć w odrębnej dokumentacji "*Przebiegi Wzorcowe".*

1.3.10. Agregacja danych archiwalnych

Od wersji 5 system Asix posiada funkcję agregacji danych archiwalnych. Funkcję tę realizuje moduł **Agregator**. Obliczone agregaty udostępniane są programom pobierającym dane za pośrednictwem serwerów **AsixConnect** (programy: AsTrend, AsAlarm i moduł skryptów).

Agregator odczytuje dane surowe i na ich podstawie wylicza zadane agregaty. Może je także przechowywać w archiwum danych zagregowanych.

Agregator pozwala również na wykonanie agregacji przy odczycie danych. Na podstawie danych surowych oraz zapisanych danych zagracowanych Agregator wylicza agregaty także w okresach innych niż zarchiwizowane.

Archiwum wartości zagregowanych przechowywane jest w plikach binarnych o wewnętrznym formacie.

1.3.10.1. Wymagania agregacji

Agregator korzysta z podstawowego rodzaju archiwum danych, co wymaga określenia podstawowego typu archiwizacji dla każdej zmiennej.

1.3.10.2 Parametryzacja

Parametry agregacji i archiwizacji agregatów przechowywane są w pliku konfiguracyjnym XML Serwera Danych systemu Asix.

Parametryzacji należy dokonać przy użyciu programu Architekt:

Architekt > *Obszary i komputery* > *Dane archiwalne* > <*nazwa zadeklarowanego archiwum*> > zakładka *Agregaty*:

Parametry:

Lista okresów - jest listą okresów agregacji w konwencji modułu danych archiwalnych oddzielonych pionowymi liniami, np. 5m | 1h - oznacza wyliczanie agregatów 5-minutowych i 1-godzinnych.

Lista agregatów - jest listą nazw agregatów, oddzielonych pionowymi liniami, np.: Jakość_Dobra | Jakość_Zła | Średnia

Horyzont agregacji - to ograniczenie na najstarsze dane do agregacji, podane w konwencji modułu danych archiwalnych, np.: 366d - oznacza, że nie będą archiwizowane agregaty starsze niż rok. Horyzont agregacji nie jest wymagany - brak wartości oznacza agregację wszystkich danych.

1.3.10.3 Zestaw agregatów

Lista dostępnych agregatów:

- ***Początek*** - wartość na początku interwału

- ***Koniec*** - wartość na końcu interwału

- ***Przyrost*** - różnica wartości na końcu i na początku interwału

- ***Min*** - minimalna wartość w interwale

- ***Max*** - maksymalna wartość w interwale

- ***Zakres*** - różnica między maksymalna a minimalna wartością w interwale

- ***Suma*** - suma wartości ważonych czasowo w interwale (całka po czasie)

- ***Suma0*** - suma wartości ważonych czasowo (całka po czasie); w okresach, w których próbka ma jakość zła, do obliczeń używana jest wartość 0

- ***Średnia*** - średnia wartości ważonych czasowo w interwale

- ***Średnia0*** - średnia wartości ważonych czasowo w interwale; w okresach, w których wartość jest niedostępna do obliczeń, używana jest wartość 0.

- ***Jakość_Dobra*** - procent próbek o jakości dobrej w interwale

- ***Jakość_Niepewna*** - procent próbek o jakości niepewnej w interwale

- ***Jakość_Zła*** - procent próbek o jakości złej w interwale

- ***Jakość_Dobra_Czas*** - czas trwania (w sekundach) czasu w przedziale, w którym dane mają jakość dobrą

- ***Jakość_Niepwewna_Czas*** - czas trwania (w sekundach) czasu w przedziale, w którym dane mają jakość niepewną

- ***Jakość_Zła_Czas*** - czas trwania (w sekundach) czasu w przedziale, w którym dane mają jakość złą

- ***Suma_Wzrostów*** - suma wartości dodatnich zmian, od ostatniej znanej wartości przed rozpoczęciem się do ostatniej wartości przed końcem przedziału próbkowania.

Jednym ze sposobów wykorzystania agregatu jest obliczanie liczby załączeń. Wystarczy w tym celu zagregować zmienną, która przyjmuje wartość 0 przy wyłączonym urządzeniu, a wartość 1 przy włączonym.

- ***Suma_Spadków*** - suma wartości bezwzględnych ujemnych zmian, od ostatniej znanej wartości przed rozpoczęciem się do ostatniej wartości przed końcem przedziału próbkowania.

Jednym ze sposobów wykorzystania agregatu jest obliczanie liczby wyłączeń. Wystarczy w tym celu zagregować zmienną, która przyjmuje wartość 0 przy wyłączonym urządzeniu, a wartość 1 przy włączonym.

- ***Poprzedni_Znany*** - ostatnio znana (dobra lub niepewna) wartość przed rozpoczęciem przedziału próbkowania. Podaje ostatnią wartość, nawet jeżeli wystąpiła ona bardzo dawno.

- ***Ostatni*** - ostatnio znana (dobra lub niepewna) wartość w przedziale próbkowania wyłączając koniec przedziału - dla odróżnienia od agregatu End, który zawiera koniec przedziału.

- ***Odchylenie_Standardowe*** - ważone czasowo odchylenie standardowe w przedziale próbkowania. Wynik jest zawsze typu Double.

Odchylenie standardowe jest liczone jako $\sqrt{\frac{\sum_{i=1}^{n}(t_i x_i - Avg(x))^2}{\sum_{i=1}^{n} t_i}}$

- ***Średnia_Kwadratowa*** - ważona czasowo średnia kwadratowa w przedziale próbkowania. Wynik jest zawsze typu FLOAT.

Średnia kwadratowa jest liczona jako $\sqrt{\frac{\sum_{i=1}^{n}(t_i x_i)^2}{\sum_{i=1}^{n} t_i}}$

- ***Średnia_Ostatnia_Znana*** - średnia ważona czasowo w przedziale próbkowania; w miejsce złych wartości wstawia ostatnią znaną wartość. Wynik jest zawsze typu Double.

Średnia_Ostatnia_Znana jest lepsza od *Średnia* dla zmiennych, które z natury mają w dłuższych okresach wartość stałą, np. sygnał pracy urządzenia.

- ***Suma_Ostatnia_Znana*** - całka w przedziale próbkowania; w miejsce złych lub brakujących wartości wstawia ostatnią znaną wartość. Wynik jest zawsze typu Double.

Jednym ze sposobów wykorzystania agregatu jest obliczanie czasu pracy urządzeń. Wystarczy w tym celu zagregować zmienną, która przyjmuje wartość 0 przy wyłączonym urządzeniu, a wartość 1 przy włączonym. Wynik będzie czasem pracy w sekundach.

1.3.11 Archiwum X - obsługa danych ze znacznikami czasu o dokładności poniżej sekundy

Dane bieżące

We wcześniejszych wersjach pakietu moduł danych bieżących Asmen mógł pozyskiwać dane bieżące i historyczne z częstotliwością oraz precyzją danych do jednej sekundy (nie licząc zdarzeń i alarmów).

Od pakietu Asix 9.0.5 dostępna jest nowa wersja, która może odświeżać dane bieżące do 10 razy na sekundę. Oczywiście warunkiem jest odpowiednio szybkie udostępnianie danych przez źródło danych (np. sterownik) i wystarczająco szybkie łącze.

Dane historyczne

Moduł danych historycznych Aspad mógł do tej pory archiwizować dane nie częściej niż co sekundę, z dokładnością czasu do jednej sekundy.

Aspad 9, który znalazł się w wersji Asix 9.1, został zmieniony tak, aby mógł przetworzyć wszystkie dane modułu danych bieżących z pełną dokładnością czasu, nawet do 100 nanosekund. Może także rejestrować dane z taką rozdzielczością, czyli teoretycznie co 100 nanosekund. W praktyce ograniczeniem jest dopuszczalna częstotliwość odświeżania źródła danych, przepustowość łącza i możliwości modułu danych bieżących, czyli maksymalnie do 10 próbek na sekundę.

Nie każdy rodzaj archiwum może rejestrować dane z taką częstotliwością. W archiwach standardowych pozostaje ograniczenie do jednej próbki na sekundę. Archiwum B umożliwia teraz zapis z dokładnością do 1/300 sekundy, przy czym ze względu na duży rozmiar rekordów, nie zaleca się stosowania częstego próbkowania dla tego typu archiwum dla zmiennych przechowywanych w dłuższych okresach czasu.

W związku z powyższymi ograniczeniami został opracowany zupełnie nowy rodzaj archiwizacji – Archiwum X. To archiwum, zapisywane w bazie danych SQL, jest poddawane kompresji danych i umożliwia przechowywanie czasu z zadaną precyzją, maksymalnie do 100 nanosekund. Dane mogą być archiwizowane zgodnie z tempem odświeżania wartości przez moduł danych bieżących Asmen, czyli z minimalnym okresem 100 milisekund.

Parametryzacja archiwizacji zmienia się nieznacznie, przy czym:

- Architekt daje dodatkowo możliwość zadeklarowania Archiwum X i umożliwia od razu
- utworzenie bazy danych na serwerze SQL, jeżeli użyjemy konta o wystarczających uprawnieniach;
- w bazie zmiennych można teraz zadeklarować typ archiwizacji X;
- w parametrach archiwizacji można zadać częstotliwość i precyzję rejestracji poniżej
- sekundy, podając po kropce część ułamkową sekund;
- nie dopuszcza się parametryzacji zmiennych w archiwum X, bez określenia większej
- od zera precyzji rejestracji wartości zmiennej;
- domyślna wartość precyzji znacznika czasu, gdy pominięto ten parametr zmiennej, dla
- archiwum D, M, Y, H i B wynosi 1 sekundę, a dla archiwum X jest równa okresowi rejestracji zmiennej;
- wartość precyzji znacznika czasu, gdy podano zero w parametrach zmiennej, dla archiwum D, M, Y i H wynosi 1 sekundę, dla B – 100 milisekund, a dla archiwum X 100 nanosekund.

Są pewne ograniczenia w użyciu danych Archiwum X:

- dane nie są dostępne w wersji klasycznej Asixa.

Wymagania:

Serwer Microsoft SQL od wersji 2008 R2.

1.4. Aslink - moduł sieciowy

Aslink jest modułem, który umożliwia innym komponentom systemu Asix wzajemną komunikację poprzez lokalną sieć komputerową. Aslink definiuje zestaw usług sieciowych, z których korzystają wszystkie inne programy systemu Asix, sparametryzowane do pracy w sieci.

W swoim działaniu Aslink wykorzystuje usługi dostarczane przez emulator NETBIOS, tzn. program implementujący jeden z najpowszechniej stosowanych interfejsów sieciowych w warstwie transportu. Specyfikacja NETBIOS została opracowana przez IBM; Aslink opiera się na specyfikacji NETBIOS wersja 3.0 zawartej w IBM Local Area Network Technical Reference (IBM Part Number SC30-3587-00). Podstawą działania Aslinka są NETBIOS'owe usługi połączeniowego (inaczej: sesyjnego) przesyłu danych.

Wersja Aslinka dla systemów WINDOWS NT, WINDOWS 2000, WINDOWS XP wykorzystuje standardowy interfejs NETBIOS udostępniany przez wymienione systemy operacyjne.

Zwracamy uwagę, że instalowanie usług sieciowych jest zbędne w tych konfiguracjach Asixa, gdzie nie jest wymagany dostęp do sieci komputerowej.

1.4.1. Ogólna charakterystyka modułu Aslink

1.4.1.1. Usługi wiarygodnego przesyłu danych

Podstawowa usługa świadczona między innymi na rzecz takich komponentów pakietu Asix jak Asmen, Aspad, czy podsystem obsługi alarmów - to wiarygodny przesył danych pomiędzy każdą parą procesów użytkowych, działających na dowolnych stanowiskach przyłączonych do sieci komputerowej i realizujących wspólny protokół użytkowy. W tym zakresie usługi modułu Aslink ukierunkowane są na wsparcie komunikacji opartej na istnieniu procesu "*serwera*" i korzystającego z jego usług procesu "*klienta*". W szczególności Aslink zapewnia:

- wyszukanie w sieci procesów pełniących rolę serwerów w odniesieniu do zasobów potrzebnych procesom klientów,
- nawiązanie połączenia (tzw. *kanału*) pomiędzy klientem i wskazanym przez niego serwerem zasobu,
- dwukierunkowy, wiarygodny przesył danych w każdym z *kanałów,*
- powiadomienie procesów, że *kanał* został zamknięty wskutek awarii sieci lub na jawne polecenie któregoś z partnerów.

1.4.1.2. Usługa synchronizacji czasu

W przypadku instalacji sieciowych Asixa bardzo istotne jest zapewnienie zgodności czasu na poszczególnych stanowiskach. W tym celu przynajmniej jedno stanowisko powinno dysponować dobrym źródłem aktualnego czasu i spełniać funkcje serwera czasu, czyli systemu udostępniającego czas innym stanowiskom sieciowym. Funkcja synchronizacji czasu została zaimplementowana w module sieciowym Aslink. W zależności od parametryzacji Aslink może pełnić funkcję serwera czasu lub jedynie synchronizować swój czas z aktywnym serwerem czasu. Każdy serwer posiada swój priorytet. W danej chwili aktywny jest tylko serwer o najwyższym priorytecie. Pojawienie się w sieci nowego serwera o wyższym priorytecie powoduje przejście aktualnego serwera w stan zwykłej synchronizacji tj. odbierania czasu od nowego serwera. Wyłączenie (lub awaria) aktualnego serwera powoduje aktywizację jednego z pozostałych serwerów. Innym parametrem charakteryzującym pracę serwera czasu jest częstotliwość, z jaką serwer rozgłasza w sieci swój aktualny czas. Wszystkie stacje, odbierające aktualny czas rozgłaszany przez aktywny serwer, dokonują synchronizacji swojego czasu, gdy różnica pomiędzy czasem własnym i serwera przekroczy pewną dopuszczalną, określoną parametrem wartość.

1.4.1.3. Usługi zdalnego zapisu plików

Moduł Aslink świadczy na rzecz modułów wbudowanych w pakiet Asix także usługi związane z transferem plików pomiędzy stanowiskami sieciowymi. Na tej bazie możliwe jest w sieci stanowisk Asix zdalne przeglądanie katalogów, skopiowanie pliku na dysk lokalny, zapis pliku na dysk zdalny, czy skasowanie pliku. Opcje związane z zapisem/usuwaniem plików można udostępniać wybiórczo na poszczególnych stanowiskach, odpowiednio w tym celu parametryzując moduł Aslink.

1.4.1.4. Konfiguracje wielokartowe

ASLINK może współpracować z kilkoma kartami sieciowymi jednocześnie. Nie powoduje to jednak bezpośredniego połączenia komputerów dwóch różnych sieci. Przepływ danych pomiędzy sieciami wymaga jeszcze działania na komputerach pomostu odpowiednio sparametryzowanych modułów Asmena i Aspada.

1.4.2. Typy obsługiwanych rodzajów komunikacji

Aslink może pracować w każdej sieci akceptowanej przez systemy operacyjne, na których pracuje system Asix.

Istnieją 2 typy pracy:

1. **oparty na NETBIOS** - w konfiguracji karty sieciowej wymaga włączenia obsługi protokołu NetBT;

działa we wszystkich wersjach systemu Asix;

działa w dwóch wariantach:

- tylko w jednej sieci lokalnej - tryb zwykły; aktywne poszukiwanie serwerów;
- w przypadku użycia w kilku segmentach - połączenia sesyjne (należy użyć opcji ***Połączenia sesyjne***); wymaga zdefiniowania wszystkich serwerów, z którymi będą nawiązywane połączenia;

2. **wykorzystanie własnego protokołu bazującego na TCPIP**;

działa w sieci rozległej i wielu segmentach sieci lokalnej;

wymaga zdefiniowania wszystkich serwerów, z którymi będą nawiązywane połączenia;

działa w systemie Asix od wersji 6.0

Zaleca się wykorzystanie własnego protokołu bazującego na TCPIP.

Komunikacja pomiędzy modułami systemu Asix na jednym komputerze odbywa się po protokole wewnętrznym.

Jeśli program Architekt został uruchomiony bez uruchamiania aplikacji systemu Asix, a serwer Asix6 używa tylko protokołu TCP/IP lub znajduje się w innej podsieci, to należy skonfigurować połączenie TCP/IP z tym serwerem używając polecenia: Architekt > menu *Narzędzia* > *Moduł sieci - opcje dla serwerów*.

1.4.3. Parametryzacja modułu ASLINK

Parametryzacja modułu Aslinkj w całości odbywa się przy użyciu programu Architekt.

Patrz: Serwer Danych > Parametryzacja Serwera Danych, cały rozdział *2.11. Konfiguracja modułu sieci*

1.4.3.1. Identyfikacja stacji

Stacje identyfikowane są za pomocą nazw. Każda stacja powinna mieć unikalną w sieci nazwę. Oprócz nazwy indywidualnej, każda stacja należy do grupy stacji posiadających wspólną nazwę grupową. Wymiana danych pomiędzy stacjami należącymi do różnych grup nie jest możliwa. Dotyczy to również synchronizacji czasu - czas stacji jest synchronizowany tylko wewnątrz jednej grupy. Wyjątkiem od powyższej zasady są stacje pracujące pod kontrolą modułu sieciowego w wersji niższej niż 5.00.000. Moduł sieciowy w wersji 5.00.000 lub wyższej, sparametryzowany do współpracy ze stacjami o niższych wersjach będzie mógł się z nimi łączyć oraz synchronizować czas. Stacje pracujące pod kontrolą starszych wersji modułu sieciowego należą do wszystkich grup.

Parametr nazwy służy do nadawania stacji unikalnej nazwy sieciowej:

program Architekt > *Obszary i komputery* > *Moduł sieci* > zakładka *Nazwa komputera* > opcja ***Automatyczne nadawanie sieciowych nazw komputerom*** / ***Sieciowa nazwa komputera***

Parametr grupy przyporządkowuje stacji nazwę grupową (dla komunikacji poprzez NetBIOS):

program Architekt > *Obszary i komputery* > *Moduł sieci* > zakładka *Komunikacja poprzez NetBIOS* > *Zaawansowane* > ***Nazwa grupy***

1.4.3.2. Dostęp do sieci przy komunikacji poprzez NetBIOS

Moduł sieciowy Aslink realizuje dostęp do sieci za pomocą adapterów logicznych. Adapter logiczny to fizyczny adapter sieciowy oraz zestaw związanych z nim protokołów sieciowych. Adaptery logiczne identyfikowane są za pomocą numerów. W systemie Windows powiązania numerów logicznych z protokołami i rzeczywistymi adapterami sieciowymi można ustalić za pomocą ustawień sieciowych systemu operacyjnego. Do odczytu tych powiązań można również wykorzystać program AslView.

Zestaw adapterów logicznych podaje się za pomocą parametru ***Używane karty sieciowe i protokoły*** (Architekt > *Obszary i komputery* > *Moduł sieci* > zakładka *Komunikacja poprzez NetBIOS* > *Protokoły sieciowe*). Wartością tego parametru są nazwy protokołów lub numery adapterów logicznych oddzielone przecinkiem. Jeśli jako wartość podano znak "*", to zostaną wykorzystane wszystkie dostępne w systemie operacyjnym adaptery logiczne.

W niektórych zastosowaniach konieczne jest ograniczenie zbioru stacji, z którymi można nawiązywać połączenie. Do tego celu służy parametr ***Karty sieciowe i protokoły***

używane przez moduł sieciowy do szukania serwerów (Architekt > *Obszary i komputery* > *Moduł sieci* > zakładka *Komunikacja poprzez NetBIOS* > *Protokoły sieciowe*). Jako wartość podaje się numery adapterów, na których będą szukane serwery. Parametr ma wpływ jedynie na działanie klientów na stacji lokalnej. Nie ogranicza on połączeń dokonywanych z inicjatywy zdalnych klientów. Brak tego parametru lub podanie znaku '*' jako jego wartości spowoduje, że do szukania będą wykorzystane wszystkie adaptery.

Innym sposobem ograniczenia zbioru stacji, z którymi mogą być nawiązywane połączenia jest użycie parametru ***Filtrowanie sieciowych nazw komputerów*** (Architekt > *Obszary i komputery* > *Moduł sieci* > zakładka *Zaawansowane*). Wartością tego parametru są oddzielone przecinkiem nazwy stacji, z którymi mogą być nawiązywane połączenia. Parametr dotyczy tylko nawiązywania połączeń ze zdalnymi serwerami. Klienty na zdalnych stacjach mogą łączyć się z serwerami na lokalnej stacji bez ograniczeń. Jeśli nazwa filtru zawiera znaki "?", to porównywanie nazwy zdalnej stacji z filtrem nie będzie przeprowadzone w miejscu tych znaków. Przykładowo, jeśli w sieci znajdują się stacje o nazwach A1X, A2X, i B1X, a filtr ma postać A?X, to będą możliwe połączenia z serwerami na stacjach A1X i A2X, ale nie będą możliwe ze stacją B1X. Nazwa filtru może zawierać znak "*". W takim przypadku nazwy filtru i stacji są porównywane tylko w części poprzedzającej znak "*".

PRZYKŁADY

W systemie zdefiniowane są adaptery ujęte w tabeli poniżej.

Tabela: Adaptery logiczne modułu sieciowego ASLINK.

Numer logiczny adaptera	„Ścieżka sieciowa" zdefiniowana w systemie
0	NetBT->Tcpip->Karta Realtek RTL8029(AS) PCI Ethernet Adapter
1	NetBT->Tcpip->Karta Realtek RTL8029(AS) PCI Ethernet Adapter #2
2	NetBT->Tcpip-> Karta 3Com EtherLink III ISA (3C509/3C509b) w trybie Legacy"
3	Nbf->Karta Realtek RTL8029(AS) PCI Ethernet Adapter
4	Nbf->Karta Realtek RTL8029(AS) PCI Ethernet Adapter #2
5	Nbf->Tcpip-> Karta 3Com EtherLink III ISA (3C509/3C509b) w trybie Legacy"
6	Nbf->WAN Miniport (NetBEUI, Dial Out)
7	Nbf->WAN Miniport (NetBEUI, Dial Out)
8	Nbf->WAN Miniport (NetBEUI, Dial In)
9	NwlnkNb

Wybrane protokoły: NBF
wszystkie adaptery logiczne z protokołem Nbf, ale bez łączy typu WAN

Wybrane protokoły: Realtek&#2
wszystkie adaptery logiczne związane z kartą " Karta Realtek RTL8029(AS) PCI Ethernet Adapter #2"

Wybrane protokoły: Nbf+Realtek, IPX
wszystkie adaptery logiczne związane z protokołem Nbf i kartami Realtek oraz z protokołem IPX (NwlnkNb)

Wybrane protokoły: tcpip
wszystkie adaptery logiczne związane z protokołem TCP/IP, ale bez łączy typu WAN

Wybrane protokoły: tcpip&3Com
adapter logiczny 2 (protokół TCP/IP na karcie 3Com)

Wybrane protokoły: NETBEUI&In
adapter logiczny 8 (połączenie WAN przychodzące)

Wybrane protokoły: **-** TCPIP&3Com
wszystkie adaptery z wyjątkiem adaptera logicznego 2 (protokół TCP/IP i karta 3Com) oraz połączeń typu WAN

Wybrane protokoły: **-** IPX
wszystkie adaptery logiczne za wyjątkiem adapterów związanych z protokołem IPX i łączami typu WAN

Wybrane protokoły: **-** /2
wszystkie adaptery logiczne za wyjątkiem adapterów związanych z kartą o numerze 2 (numer slotu) oraz łączami typu WAN

Wybrane protokoły: 1,3,5
adaptery logiczne 1, 3 i 5

Wybrane protokoły: *, wan
wszystkie adaptery logiczne

Domyślna wartość specyfikacji adapterów ma postać:
Wybrane protokoły: *
tzn. wszystkie adaptery z wyjątkiem tych związanych z łączami typu WAN

1.4.3.3. Synchronizacja czasu

Moduł sieciowy Aslink może synchronizować swój czas z innymi stacjami (Architekt > *Obszary i komputery* > *Moduł sieci* > Komunikacja przez NetBIOS > zakładka *Synchronizacja czasu*).

Synchronizację czasu z innymi stacjami można wyłączyć opcją ***Synchronizacja czasu***. Jeśli różnica czasu pomiędzy stacją a zdalnym serwerem czasu przekroczy wartość określoną parametrem ***Maksymalna różnica czasu***, w co najmniej tylu kolejno otrzymanych pakietach, ile zdefiniowano parametrem ***Liczba pakietów czasu***, to nastąpi ustawienie czasu lokalnej stacji. Maksymalną różnicę czasu podaje się w sekundach. Do ograniczenia zbioru serwerów czasu, z którymi czas może być synchronizowany, służy parametr ***Ograniczenia synchronizacji czasu***. Jego wartością jest lista serwerów czasu. Podobnie jak w przypadku filtrów, nazwy serwerów mogą zawierać znaki "?" i "*". Aby stacja zmieniła swój czas, musi otrzymać od jednego serwera czasu przynajmniej taką ilość kolejnych pakietów czasu, która jest określona parametrem ***Liczba pakietów czasu***.

Aby stacja pełniła w sieci funkcję serwera czasu, należy podać parametry ***Priorytet serwera czasu*** oraz ***Interwał serwera czasu***. Jeśli jedna z tych wartości jest mniejsza

niż zero, to stacja nie będzie pełnić funkcji serwera czasu. Jeśli wartość parametru jest równa 0, to jako wartość przyjmuje się 5 dla priorytetu i 10 dla interwału. Priorytet serwera to liczba określająca ważność serwera czasu. W danej chwili jest aktywny tylko jeden serwer o najwyższym priorytecie. Aktywny serwer czasu rozsyła pakiety sieciowe zawierające bieżący czas serwera czasu z interwałem określonym parametrem ***Interwał serwera czasu***. Interwał podaje się w sekundach.

Jeżeli serwer przestaje być aktywny, to jego funkcje może przejąć inny serwer, który był nieaktywny z powodu niskiego priorytetu. Serwer może podjąć funkcje serwera czasu, jeżeli dotychczasowy serwer nie nada żadnego pakietu przez okres czasu równy iloczynowi wartości parametru ***Warunek aktywacji serwera czasu*** (zdefiniowany na lokalnej stacji) i interwału dotychczasowego serwera czasu (zdefiniowanej na zdalnej stacji serwera czasu). Aby uniknąć sytuacji, w której błędna parametryzacja zdalnego serwera czasu (np. interwał równy 1 godzina), powoduje, że sieć nie jest w długim okresie czasu synchronizowana, wprowadzono parametr ***Interwał serwera czasu***. Jeżeli interwał serwera czasu, z którego nowy serwer nie otrzymuje pakietów czasu, jest większy niż wartość określona tym parametrem, to do obliczenia czasu, po jakim nowy serwer podejmie działanie, przyjmowana jest wartość określona parametrem.

Jeśli różnica czasu pomiędzy czasem serwera a czasem lokalnym jest większa niż wartość określona parametrem ***Duża różnica czasu***, to zmiana czasu stacji nastąpi dopiero po otrzymaniu przynajmniej tylu pakietów, ile jest określone parametrem ***Duża różnica czasu - licznik***. Parametry te dają czas na wprowadzenie korekty błędnego ustawienia czasu przez operatora zdalnej stacji.

Serwer nadaje pakiety czasu do wszystkich wykorzystywanych adapterów logicznych. Jeśli klient serwera czasu obsługuje kilka adapterów logicznych, to otrzymane z jednego adaptera pakiety czasu może przesyłać do pozostałych adapterów. Funkcja ta może być wykorzystywana do synchronizacji czasu w podsieciach innych niż podsieć, w której pracuje serwer czasu. Aby uruchomić rozsyłanie czasu do innych sieci, należy parametrowi ***Replikacja pakietów czasu*** nadać wartość *Tak* na stacji podłączonej do wielu podsieci. Pakiety czasu mogą podlegać dalszej replikacji na kolejnych stacjach aż do chwili, gdy ilość replikacji przekroczy wartość parametru ***Replikacja pakietów czasu*** (parametr ustawiany dla Serwera). Parametr definiuje się na serwerze czasu. Aby zmienić maksymalną ilość replikacji na stacjach, które dokonują replikacji, należy podać parametr ***Replikacja pakietów czasu*** (parametr ustawiany dla Klienta).

1.4.3.4. Zabezpieczenie hasłem

Z pomocą programu AslView jest możliwe wykonanie funkcji, które mogą mieć istotny wpływ na działanie modułu sieciowego. Aby uniknąć przypadkowego lub nieuprawnionego dostępu do tych funkcji, wprowadzono możliwość zabezpieczenia modułu sieciowego hasłem. Hasło można podać za pomocą parametru ***Blokada zmian*** (Architekt > *Obszary i komputery* > *Moduł sieci* > Komunikacja przez NetBIOS > zakładka *Diagnostyka - AslView*). Wartością tego parametru mogą być słowa Tak, Nie lub tekst hasła. Jeśli podano Tak, to wykonanie funkcji chronionych hasłem będzie niemożliwe. Jeśli wartością parametru jest słowo Nie, to dostęp do modułu sieciowego jest nieograniczony. Inne wartości są interpretowane jako hasło, które musi zostać podane przez użytkownika przed wykonaniem funkcji chronionej hasłem. Raz wprowadzone hasło jest ważne przez 10 minut. Każde kolejne wykonanie funkcji chronionej hasłem powoduje rozpoczęcie odliczania czasu od początku.

1.4.3.5. Połączenia ze starszymi wersjami modułu sieciowego

Stacja pracująca pod kontrolą modułu sieciowego w wersji 5.00.000 lub późniejszej może łączyć się ze starszymi wersjami modułu sieciowego tylko wtedy, gdy włączony jest parametr ***Aslink 4*** (Architekt > *Obszary i komputery* > *Moduł sieci* > Komunikacja poprzez NetBIOS > zakładka *Zaawansowane*). Możliwość ta wiąże się ze zwiększonym obciążeniem sieci w fazie nawiązywania połączenia. Domyślnie parametr *Aslink4* jest wyłączony.

Uruchomienie modułu sieciowego w wersji mniejszej niż 5.00.000 na stacji, na której uprzednio pracował moduł nowszej wersji, musi wiązać się z ponownym uruchomieniem wszystkich zdalnych stacji na których pracują nowsze wersje modułu sieciowego (należy ponownie uruchomić aplikacje korzystające z modułu sieciowego Aslink). Konieczność ponownego uruchomienia nie dotyczy stacji pracujących z modułem w starszej wersji.

1.4.3.6. Diagnostyka

Parametry sterują informacjami diagnostycznymi wyprowadzanymi przez moduł sieciowy Aslink. Niniejszy dokument nie zawiera danych pozwalających na pełną interpretację informacji diagnostycznych. Celem diagnostyki jest pomoc w ustaleniu przyczyn nieprawidłowości funkcjonowania połączeń sieciowych systemu Asix i powinna być ona uruchamiana przez pracowników firmy ASKOM lub przy ich wsparciu.

Informacje diagnostyczne wygenerowane przez moduł sieciowy mogą być zapisywane w pliku log i/lub przekazywane do programów AslView przyłączonych do modułu sieciowego.

Parametr ***Plik logu*** (Architekt > *Obszary i komputery* > *Moduł sieci* > zakładka *Diagnostyka*) definiuje nazwę oraz położenie pliku, do którego będą wyprowadzane informacje diagnostyczne. W polu *Maksymalny rozmiar [MB]* można podać maksymalną wielkość pliku log wyrażoną w megabajtach.

1.5. Obsługa alarmów dla modułów AsTrend, AsAlarm i AsRaport

Wykorzystanie danych alarmów przez programy AsTrend, AsAlarm i AsRaport wymaga przechowywania archiwum zdarzeń alarmów w bazie SQL oraz dołączenia definicji alarmów do bazy definicji zmiennych.

Generowanie bazy definicji alarmów SQL (czyli takich, których archiwum zdarzeń przechowywane jest w bazie SQL) dołączonej do bazy definicji zmiennych SQL szczegółowo opisuje rozdział Serwer Danych > Parametryzacja Serwera Danych > *2.3. Dodanie definicji alarmów SQL do bazy definicji zmiennych*.

2. Parametryzacja Serwera Danych

2.1. Praca z programem Architekt

2.1.1. Wstęp

Architekt umożliwia w pełni wizualne zaprojektowanie, skonfigurowanie i bieżącą edycję parametrów Serwera Danych. W szczególności dotyczy to edycji definicji kanałów i definicji adresów zmiennych.

Parametryzując aplikację przy wykorzystaniu programu Architekt, użytkownik wykorzystuje do tego celu okna wyposażone w system wygodnych w obsłudze zakładek, które w uporządkowany sposób grupują opcje odpowiedzialne za poszczególne obszary funkcjonalne projektowanej aplikacji. Moduł Architekt stanowi potężne narzędzie integrujące konfigurację wszystkich kluczowych modułów odpowiedzialnych za sposób działania systemu Asix.

Program Architekt zawiera w sobie moduł VarDef, odpowiedzialny za zarządzanie bazą definicji zmiennych. Umożliwia w pełni interaktywną i wizualną obsługę bazy definicji, tworzenie struktury bazy i jej edycję oraz edycję zawartości bazy. Architekt zapewnia obsługę baz Microsoft SQL Server 2008R2, 2012, 2014, 2016, baz formatu MDB (Jet / Microsoft Access), a także baz definicji zmiennych bezpośrednio pobieranych ze skoroszytów programu Excel. Dla zapewnienia kompatybilności z wcześniejszymi wersjami pakietu Asix, definicje zmiennych mogą być również pobierane z bazy formatu Paradox i plików tekstowych.

2.1.2. Środowisko pracy modułu Architekt

2.1.2.1. Uruchomienie modułu Architekt

Moduł Architekt można uruchomić wybierając kolejno z menu *Start* systemu operacyjnego:

Programy > ***Asix*** > ***Architekt***

2.1.2.2. Zamykanie modułu Architekt

Zamknięcie modułu Architekt następuje w wyniku wybrania polecenia ***Zakończ***  z menu *Plik* okna programu Architekt.

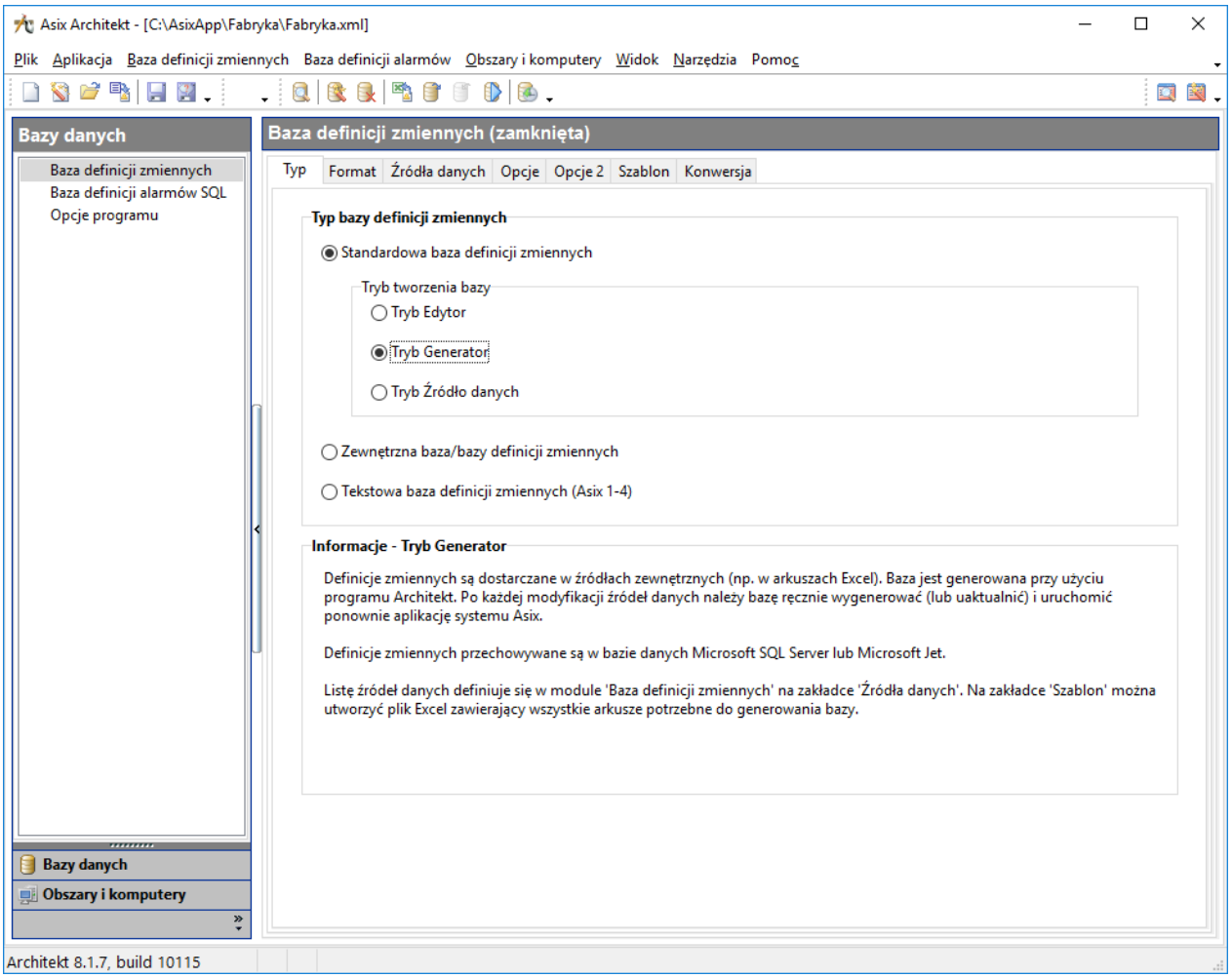
2.1.2.3. Okno główne modułu Architekt

Program Architekt posiada przyjazny interfejs wizualny. Środowisko pracy projektanta stanowi okno programu podzielone na dwa główne bloki:
- lewy blok kategorii parametrów dla baz danych i danych aplikacji;
- prawy blok parametrów konfiguracyjnych.

Parametry konfigurujące pracę aplikacji w zakresie dostępu do danych procesowych zapisywane są do **jednego wspólnego pliku XML** (ang. *Extensible Markup Language*), który powinien znaleźć się w podkatalogu DataServer katalogu głównego aplikacji.

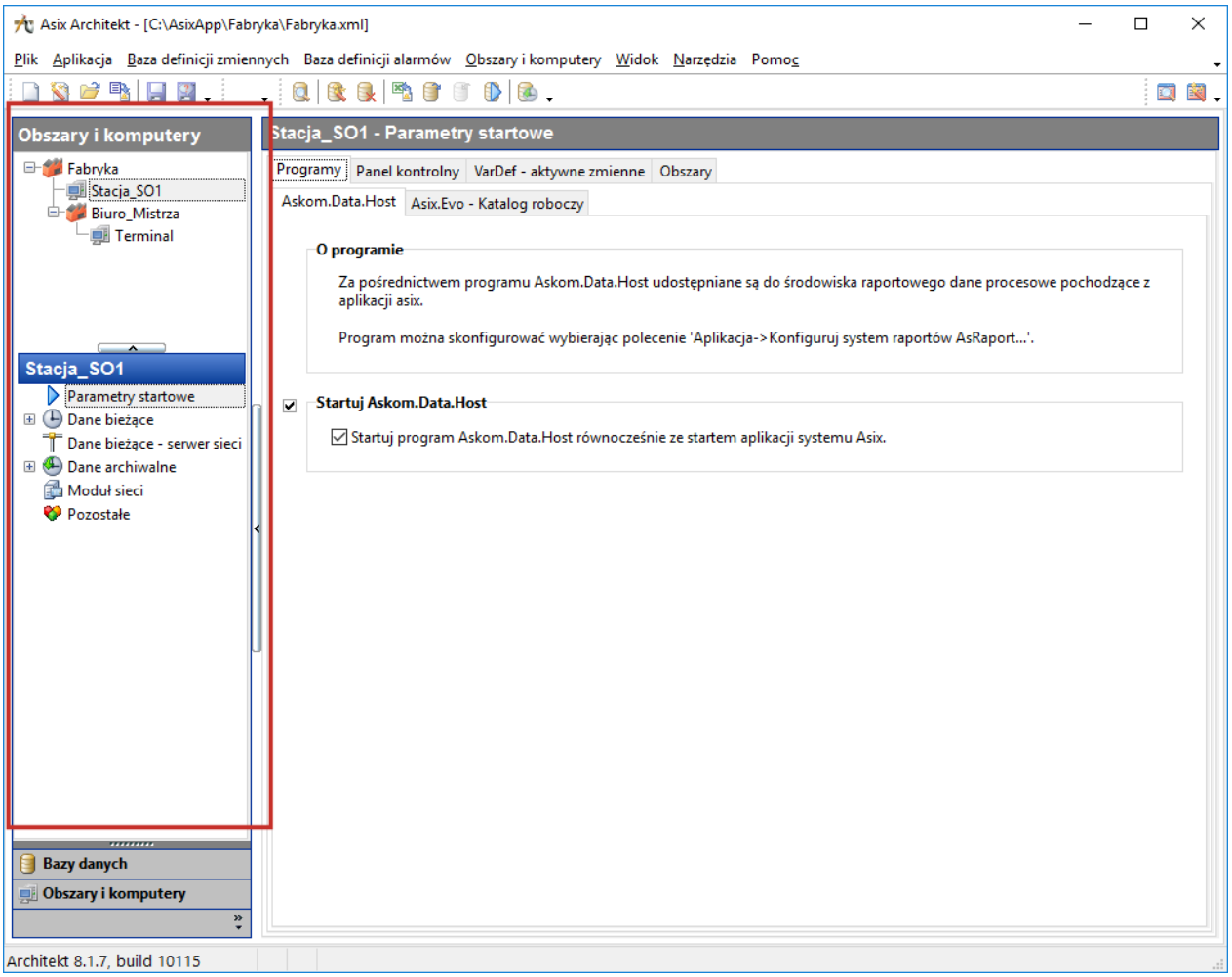
Zawartość lewego bloku okna Architekta przełączana jest przyciskami *Bazy danych* i *Obszary i komputery* (znajdującymi się w dolnej części bloku).

Podłużny przycisk ***Bazy danych*** wyświetla drzewo parametrów do konfiguracji bazy definicji zmiennych i generowania bazy definicji alarmów SQL.



Rysunek: Okno Architekta - widok grup parametrów do konfiguracji baz danych definicji zmiennych i alarmów.

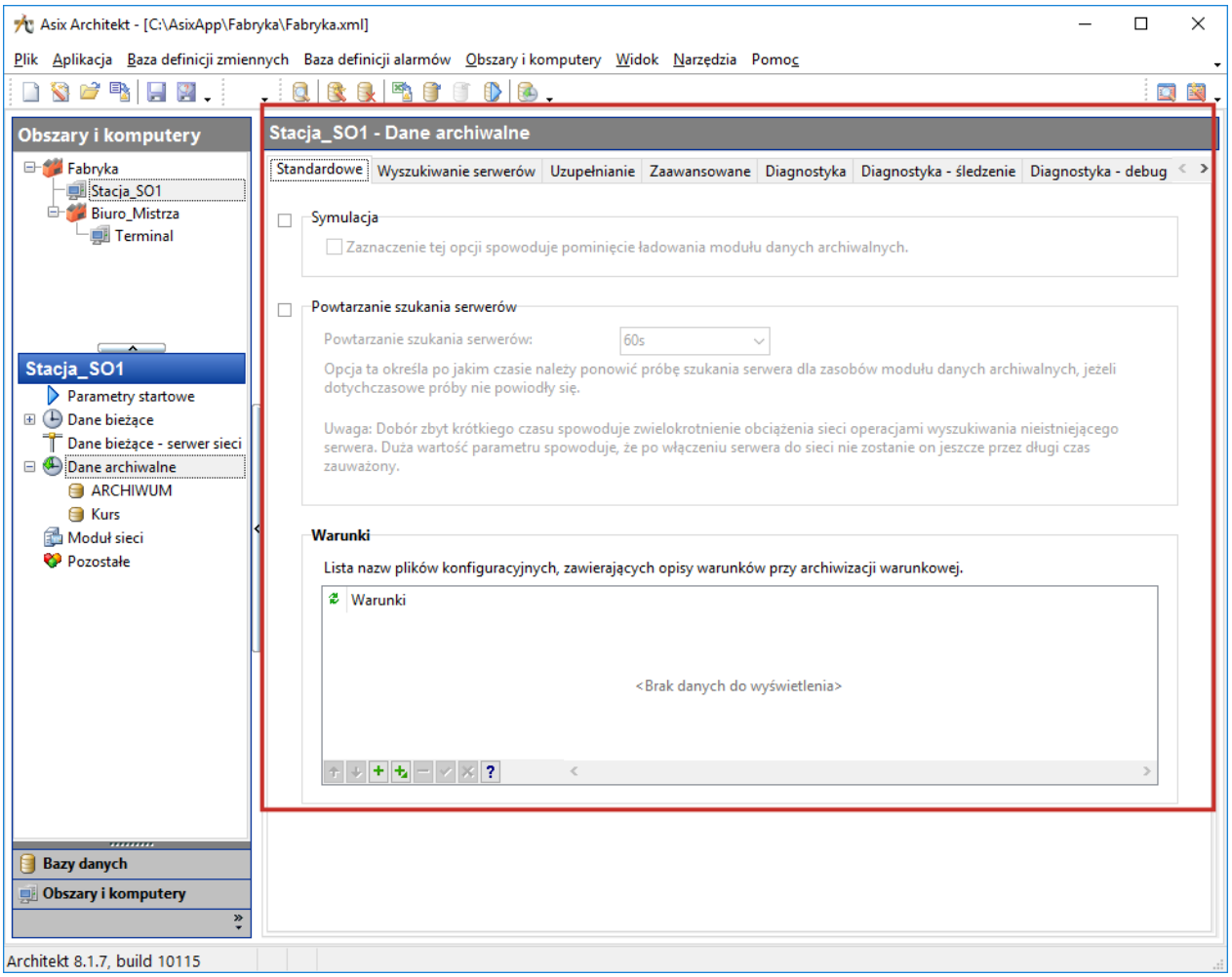
Podłużny przycisk ***Obszary i komputery*** aktywuje blok zawierający listę komputerów, dla których konfigurowana jest aplikacja systemu Asix oraz drzewo grup parametrów konfigurujących Serwer Danych tej aplikacji.



Rysunek: Okno Architekta - widok listy komputerów oraz drzewa grup parametrów do konfiguracji Serwera Danych aplikacji.

Parametry konfiguracyjne dostępne w prawym bloku okna Architekta - umieszczone zostały na tematycznych **zakładkach**.

W bloku tym projektant nadaje pożądane wartości poszczególnym parametrom aplikacji oraz edytuje bazę danych, w oparciu o którą aplikacja pracuje.

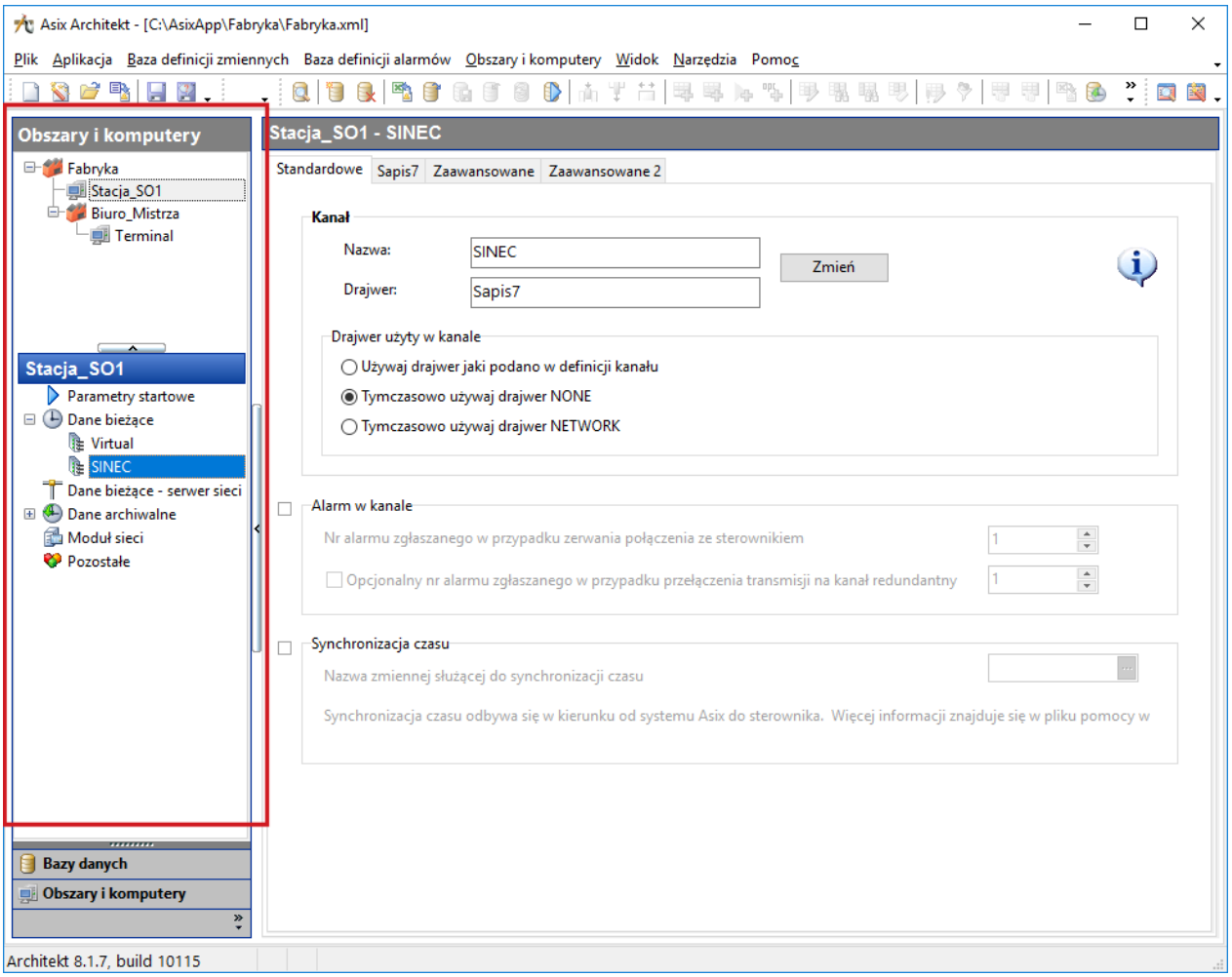


Rysunek: Okno Architekta - przykładowa zakładka do parametryzacji danych archiwalnych.

Użytkownik używając środowiska konfiguracyjnego Architekta nie zakłóca pracy aplikacji, która działa w tle. Niektóre zmiany w ustawieniach parametrów aktualizowane są automatycznie w chwili zapisu, choć dla większości parametrów wymagany jest restart aplikacji.

2.1.2.3.1 Drzewo modułów

Głównym narzędziem nawigacyjnym służącym do lokalizacji poszczególnych obszarów funkcjonalnych programu Architekt jest drzewo modułów aktywowane podłużnym przyciskiem *Obszary i komputery*.



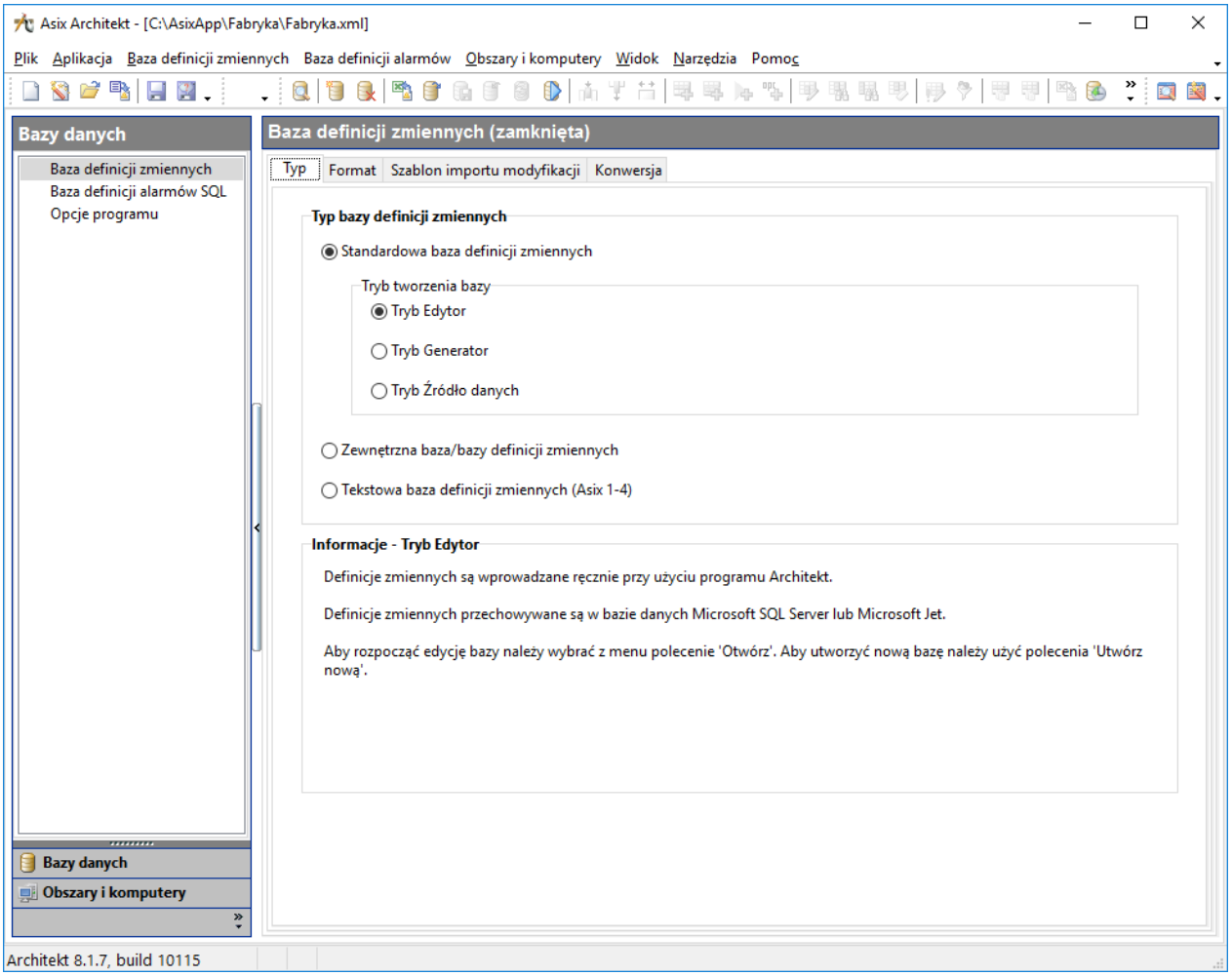
Rysunek: Okno Architekta - widok drzewa modułów aplikacji.

Ponieważ rzeczywista aplikacja działająca na obiekcie często pracuje w różny sposób na różnych stanowiskach komputerowych, wymaga to zdefiniowania w jednym pliku XML kilku różnych zestawów parametrów dla poszczególnych stanowisk. Zestaw parametrów dla wybranego stanowiska komputerowego dostępny jest w bloku parametrów konfiguracyjnych po kliknięciu na wybranym ***Komputerze*** z listy zadeklarowanych komputerów w polu *Obszary i komputery*.

Konfiguracja na poziomie ***Obszaru*** powoduje globalne ustawienia dla wszystkich komputerów należących do tego obszaru, natomiast konfiguracja na poziomie *Komputera* pozwala zindywidualizować ustawienia w zależności od typu komputera (inne dla stacji operatorskiej a inne dla dedykowanego serwera danych w sieci zakładowej). Ustawienia na poziomie *Komputera* są ważniejsze od ustawień na poziomie *Obszaru*.

Parametry konfigurujące pracę aplikacji podzielone zostały na następujące moduły parametrów:
- **Parametry startowe,**
- **Dane bieżące,**
- **Dane bieżące - serwer sieci,**
- **Dane archiwalne,**
- **Moduł sieci,**
- **Pozostałe.**

Widok drzewa modułów dla baz danych (podłużny przycisk *Bazy danych*), w zależności od ustawionego typu bazy definicji zmiennych, obejmuje pozycje omówione poniżej.



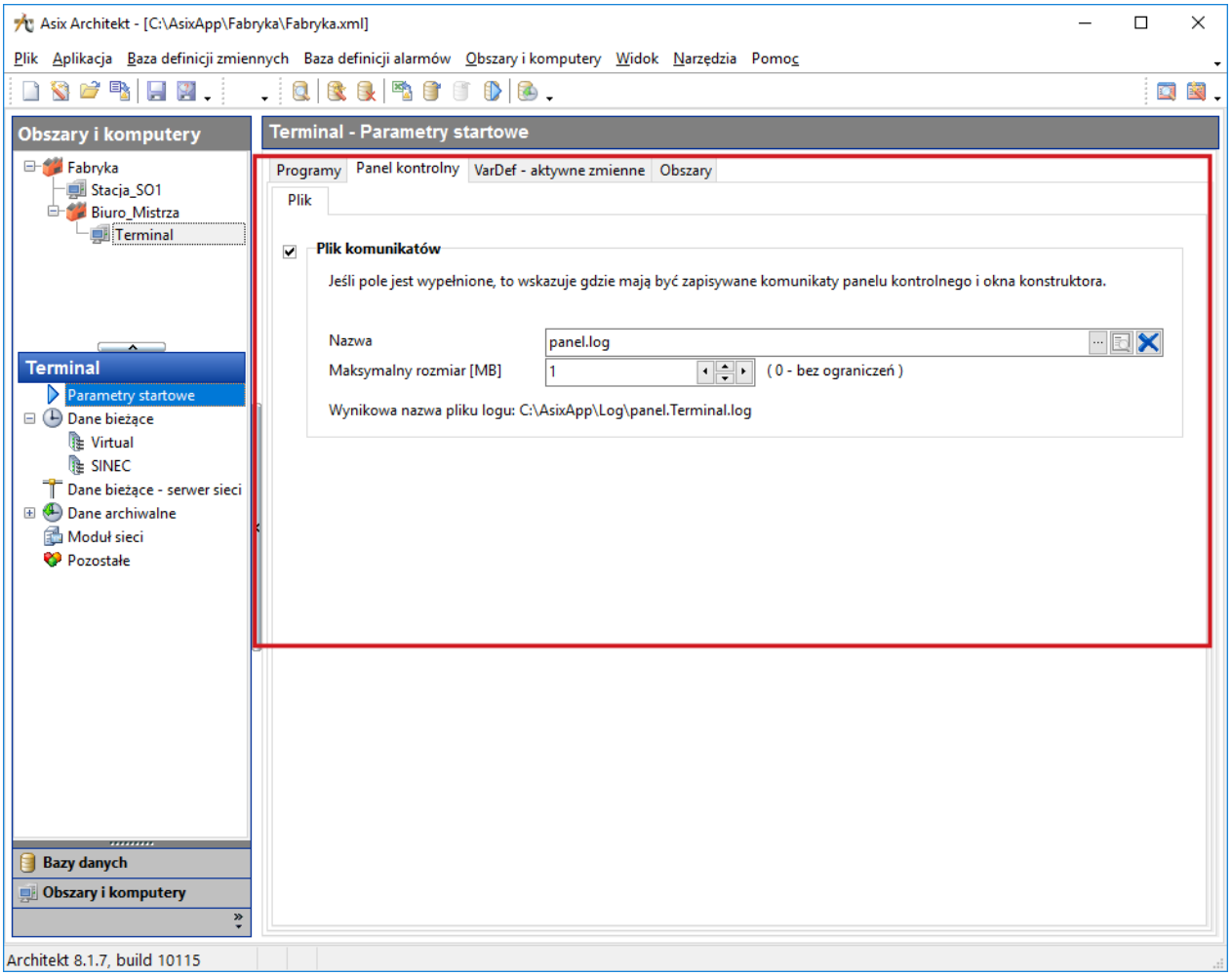
Rysunek: Okno Architekta - widok drzewa modułów dla baz danych.

- **Baza definicji zmiennych** - parametry określające ***Typ bazy***, ***Format bazy, Źródła danych*** oraz ***Opcje***;
  - **Edytor definicji zmiennych** (dostępny dla typu bazy definicji zmiennych - *Standardowa baza definicji zmiennych,* po otwarciu bazy);
  - **Edytor schematu** (dostępny dla typu bazy definicji zmiennych - *Standardowa baza definicji zmiennych*, po otwarciu bazy);
- **Baza definicji alarmów SQL** - generator bazy definicji alarmów SQL;
- **Opcje programu** - pozwalające ustawić indywidualnie dla każdej aplikacji:
  - widok okna głównego na starcie Architekta (zakładka *Opcje* > ***Start***),
  - silnik odczytu plików XLSXLSX i wybór edytora plików XLS/XLSX (zakładka *Opcje* > ***Źródła danych - silnik odczytu***),
  - parametry dla okna przeglądania bazy definicji zmiennych (zakładka *Opcje* > ***Baza definicji zmiennych***).

  (szczegóły patrz: Serwer Danych > Parametryzacja Serwera Danych > Środowisko pracy modułu Architekt *2.1.2.4. Indywidualne opcje pracy programu Architekt*).

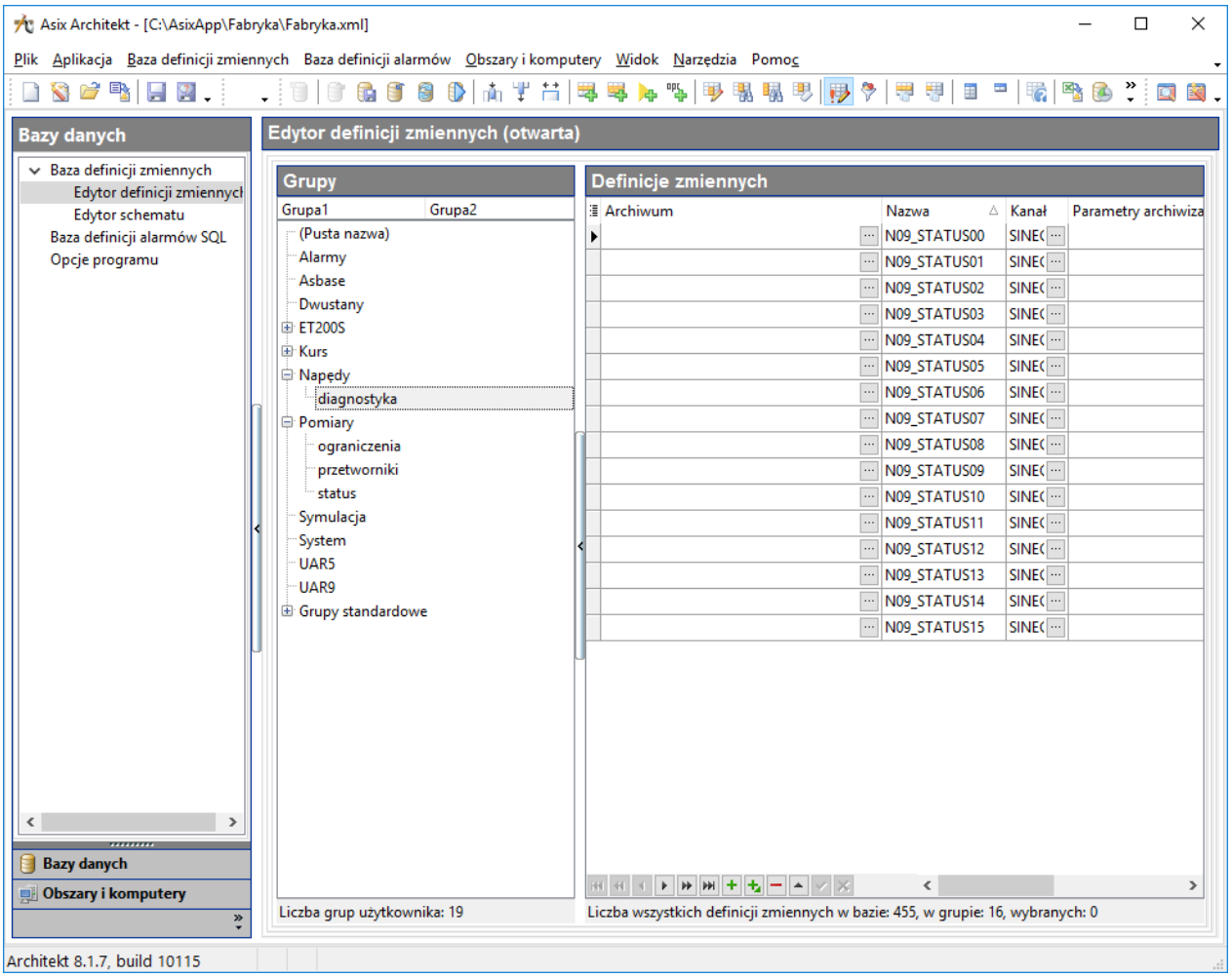
2.1.2.3.2. Blok parametrów konfiguracyjnych

Blok parametrów konfiguracyjnych umożliwia dokonywanie ustawień parametrów.



Rysunek: Okno Architekta – przykładowy widok bloku parametrów konfiguracyjnych.

Blok ten zamienia się w okno edytora atrybutów zmiennych, gdy otwarta jest baza definicji zmiennych w formacie SQL lub Jet (w grupie parametrów *Bazy danych* > *Baza definicji zmiennych* > *Typ bazy* zaznaczona jest opcja *Standardowa baza definicji zmiennych* > *Tryb edytor*)



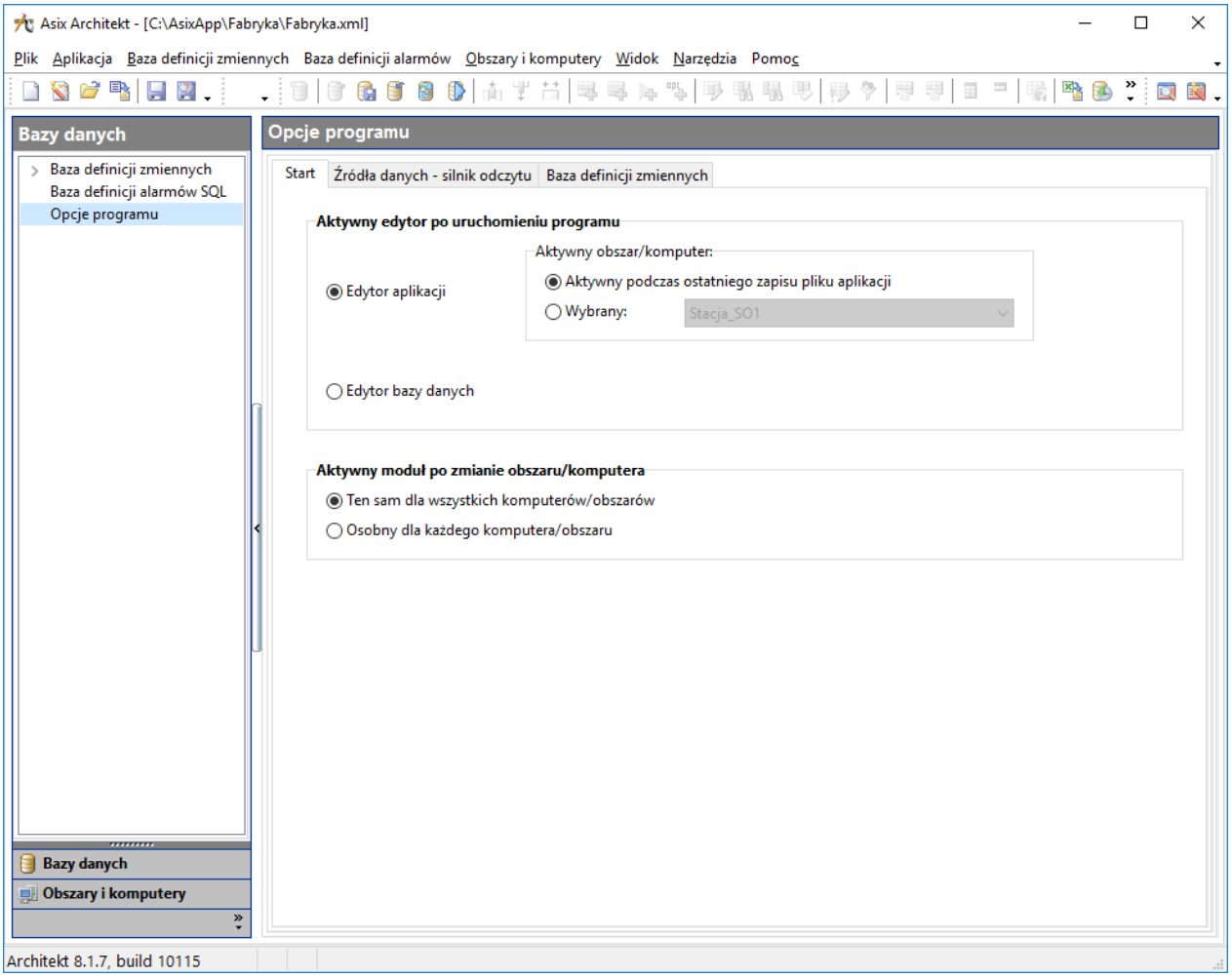
Rysunek: Okno edycji bazy definicji zmiennych.

2.1.2.4. Indywidualne opcje pracy programu Architekt

Istnieje możliwość określenia przez użytkownika pewnych domyślnych opcji pracy programu Architekt, obowiązujących podczas pracy nad danym plikiem XML.
W widoku drzewa modułów dla baz danych (podłużny przycisk *Bazy danych*) znajduje się moduł *Opcje programu*, który daje możliwość dokonania poniższych ustawień.

Zakładka ***Start***:

- ***Aktywny edytor po uruchomieniu programu*** - pozwala zadeklarować, który edytor uruchamiany będzie na starcie programu:
  - ***Edytor aplikacji*** (ze wskazaniem aktywnego obszaru/komputera),
  - ***Edytor bazy danych;***
- ***Aktywny moduł po zmianie obszaru/komputera*** **-** aktywuje mechanizm, który zapamiętuje moduł i zakładkę ostatnio otwieraną dla danego obszaru lub komputera podczas bieżącej sesji:
  - ***Ten sam dla wszystkich komputerów/obszarów*** - ostatnio otwierany moduł i zakładka będą domyślnie aktywowane po przejściu do dowolnego obszaru/komputera;
  - ***Osobny dla każdego komputera/obszaru*** **-** zostanie zapamiętany ostatnio otwierany moduł i zakładka indywidualnie dla każdego komputera/obszaru.



Rysunek: Moduł opcji pracy programu Architekt - zakładka 'Start'.

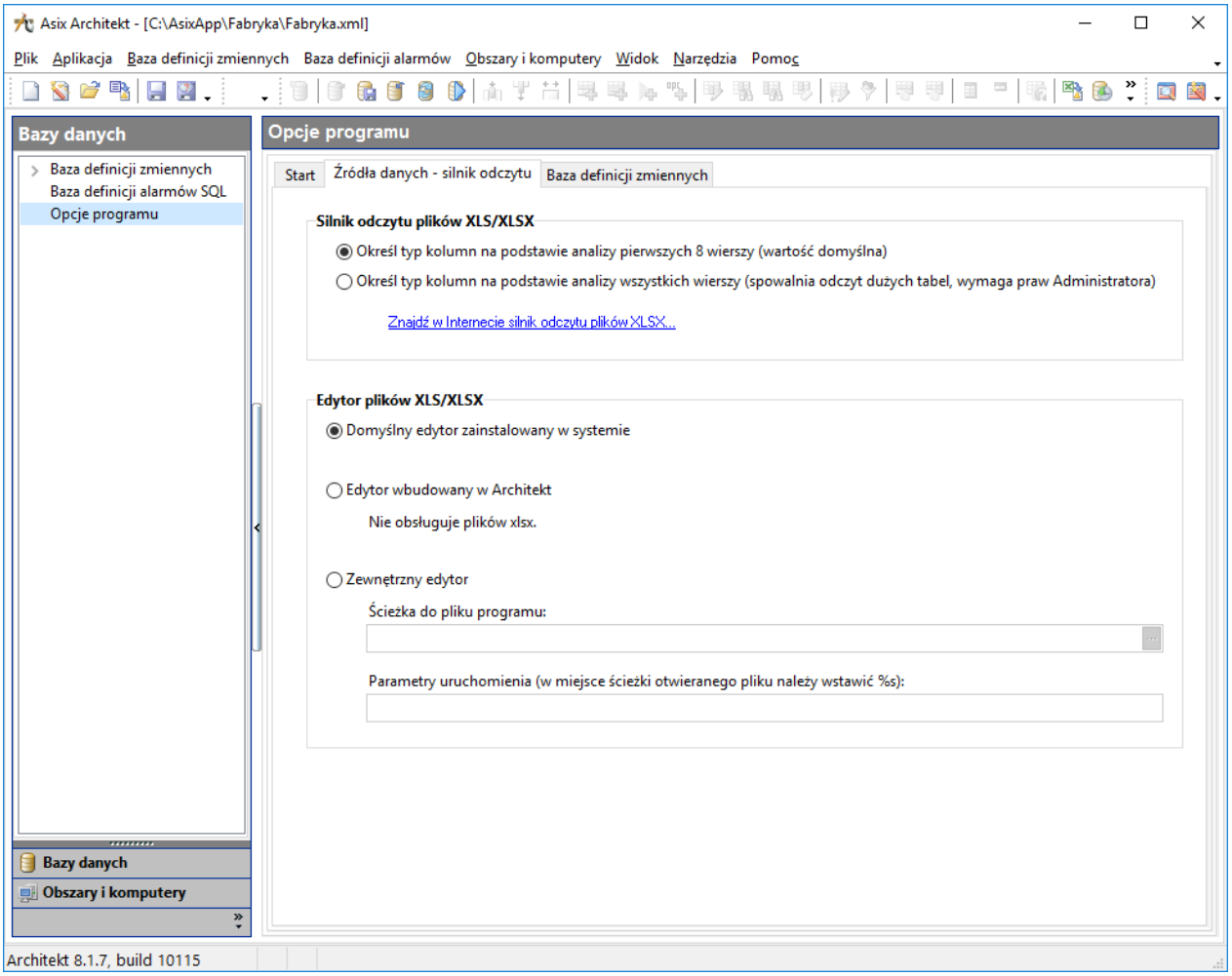
Zakładka ***Źródła danych*** - ***silnik odczytu*** pozwala zadeklarować silnik odczytu plików XLS oraz edytor plików XLS.

Silnik odczytu plików XLS/XLSX:

- ***Określ typ kolumn na podstawie analizy pierwszych 8 wierszy (wartość domyślna),***
- ***Określ typ kolumn na podstawie analizy wszystkich wierszy (spowalnia odczyt dużych tabel, wymaga praw Administratora)***

Edytor plików XLS/XLSX:

- ***Domyślny edytor zainstalowany w systemie,***
- ***Edytor wbudowany w Architekt,***
- ***Zewnętrzny edytor.***

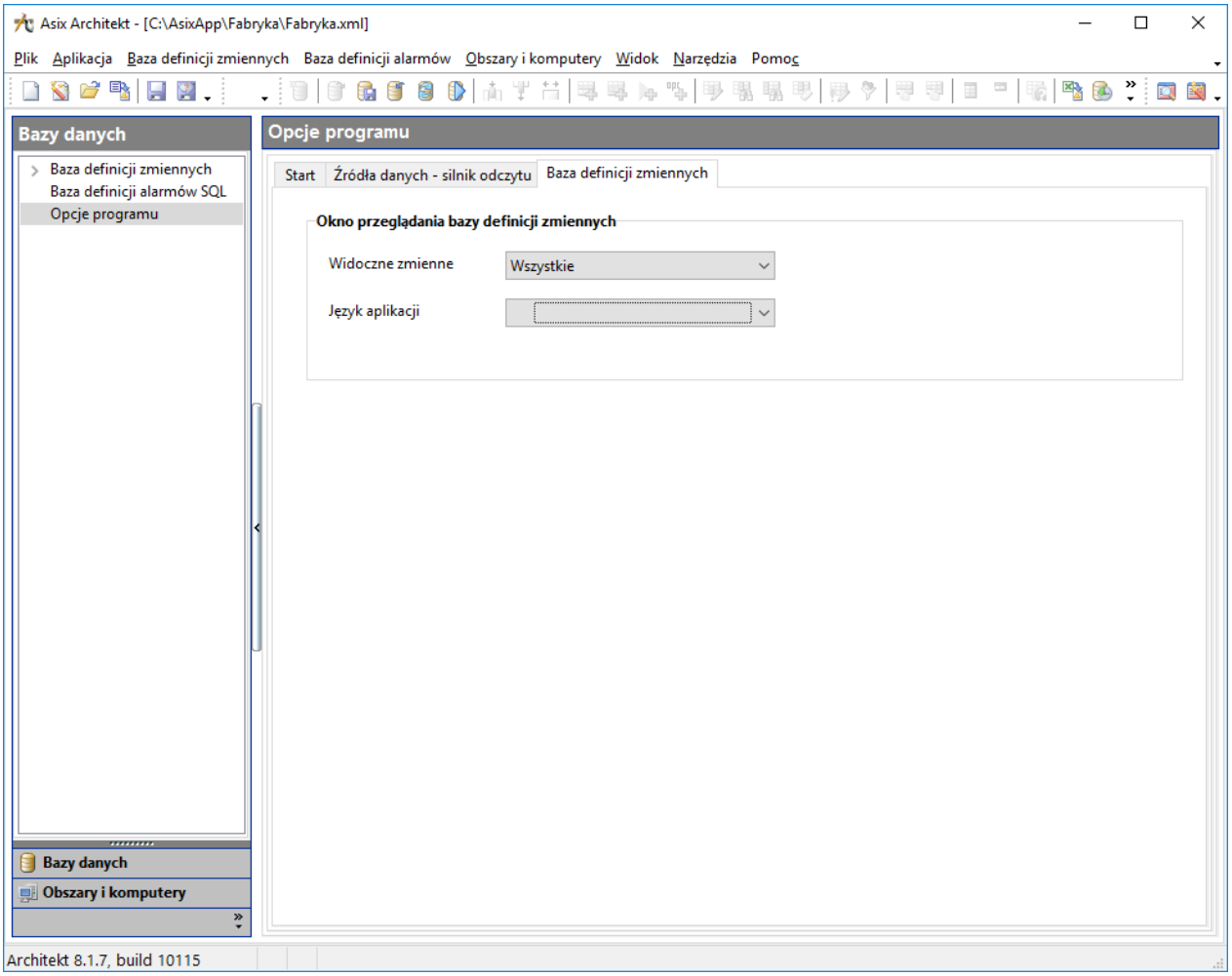


Rysunek: Moduł opcji pracy programu Architekt - zakładka 'Źródła danych - silnik odczytu'.

Zakładka ***Baza definicji zmiennych*** - pozwala określić parametry pomocne w sprawdzaniu działania okna wyboru zmiennej; efekty wyboru języka widoczne są w oknie *'Przeglądanie zmiennych'* aktywowanym przyciskiem :

- ***Okno przeglądania bazy definicji zmiennych***:
  - ***Widoczne zmienne -*** pozwala zawęzić zestaw wyświetlanych definicji zmiennych do zmiennych aktywnych lub archiwizowanych;
  - ***Język aplikacji -*** umożliwia wybór języka, w jakim wyświetlane są nagłówki i wartości atrybutów zmiennych (nagłówki atrybutów dla wymaganych języków deklarowane są w schemacie bazy definicji

zmiennych, a tłumaczenia wartości atrybutów (np. atrybutu *Opis*) w bazie definicji zmiennych).



Rysunek: Moduł opcji pracy programu Architekt - zakładka 'Baza definicji zmiennych'.

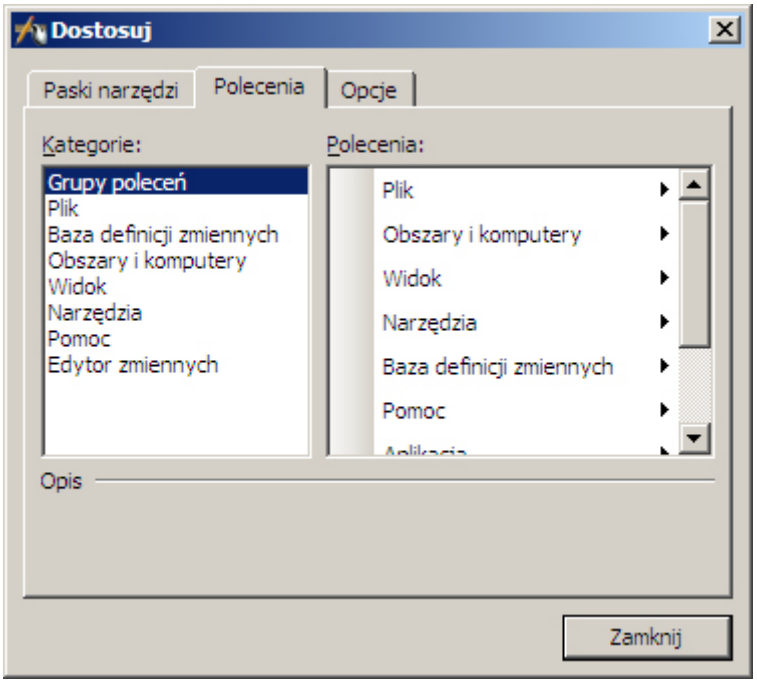
2.1.2.5. Menu główne, dostępne polecenia

Komendy wykonywane przez moduł Architekt zostały zgrupowane w 9 kategoriach:

1. *Plik*,
2. *Aplikacja*,
3. *Baza definicji zmiennych*,
4. *Baza definicji alarmów.*
5. *Obszary i komputery*,
6. *Widok*,
7. *Narzędzia*,
8. *Pomoc*,
9. *Edytor.*

Użytkownik ma możliwość samodzielnego określenia, które polecenia chce mieć wyświetlone w oknie Architekta. Służy to tego polecenie *Dostosuj...*, uruchamiane po

kliknięciu prawym klawiszem myszy na dowolnym pasku narzędzi bądź menu głównym. Wywołuje ono okno *'Dostosuj'*, które umożliwia konfigurację zawartości menu głównego oraz pasków narzędzi i dostępnych poleceń w zależności od preferencji użytkownika.



Rysunek: Okno umożliwiające konfigurację zawartości pasków narzędzi.

W oknie modułu Architekt dostępne są następujące kategorie poleceń:

Plik

 - *Nowa aplikacja* (*Ctrl* + *N*);

 - *Nowa aplikacja - kreator*;

Nowa aplikacja - kreator OPC... - uruchamia kreator aplikacji w trybie importu bazy z serwera OPC;
Nowa aplikacja - kreator OPC UA... - uruchamia kreator aplikacji w trybie importu bazy z serwera OPC UA;

 - *Nowa aplikacja - importer*;

 - *Otwórz* (*Ctrl* + O); otwiera plik XML;

 - *Zapisz* (*Ctrl* + *S*);

 - *Zapisz jako*;

 - *Zakończ*; zakończenie pracy modułu Architekt;

Wyświetlana jest też lista ostatnio otwieranych plików konfiguracyjnych XML

Aplikacja

 - *Konfiguruj system raportów AsRaport ...*

 - *Konfiguruj aplikacje internetowe...*;

 - *Konfiguruj wygląd aplikacji AsPortal*;

Baza Definicji Zmiennych - zestaw poleceń dostępny w zależności od typu bazy definicji zmiennych !!!

 - *Pokaż* (*F7*) - otwiera okno przeglądania bazy definicji zmiennych;

 - *Generuj (Ctrl + G)* - generuje bazę definicji zmiennych;

 - *Otwórz* (*Ctrl + B*) - otwiera bazę definicji zmiennych do edycji;

 - *Zapisz* - zapisuje zmiany w bazie definicji zmiennych;

 - *Zamknij* - zamyka bazę definicji zmiennych (kończy edycję);

 - *Przeładuj* - zamyka i ponownie otwiera bazę definicji zmiennych;

 - *Aktualizuj on-line* - w przypadku, gdy niemożliwe jest zatrzymanie aplikacji i jej restart w celu zmodyfikowania bazy definicji zmiennych, możliwe jest przeprowadzenie aktualizacji bazy w trybie on-line;

 - *Pokaż modyfikacje* - pozwala prześledzić zmiany w definicjach zmiennych (w tym dodawanie i usuwanie zmiennych);

 - *Importuj modyfikacje* - pozwala zaimportować do istniejącej bazy definicji zmiennych MS SQL lub Jet zmiany w definicji zmiennych wykorzystując arkusz Excel jako źródło;

 - *Exportuj*... - pozwala wyeksportować definicje zmiennych do arkusza kalkulacyjnego;
Importuj z serwera OPC... - otwiera okno importu bazy definicji zmiennych z serwera OPC do pliku xls;
Importuj z serwera OPC UA... - otwiera okno importu bazy definicji zmiennych z serwera OPC UA do pliku xls;

 - *Utwórz nową;*

 - *Usuń*;

 - *Utwórz kopię zapasową*;

 - *Przywróć z kopii*;

 - *Dodaj atrybut* - dodawanie nowego atrybutu do definicji zmiennej;

 - *Usuń atrybut* - usuwanie nowego atrybutu do definicji zmiennej;

 - *Zmień rozmiar atrybutu;*
Edytor
Mirror

 - *Pokaż atrybuty - wszystkie*

 - *Pokaż atrybuty - bieżące*
Pokaż atrybuty - systemowe

***Edytor* - polecenia edytora bazy definicji zmiennych:**

 - *Wstaw definicję zmiennej* - dodanie rekordu definicji nowej zmiennej;
 - *Wstaw kopię definicji zmiennej* - dodanie rekordu skopiowanej definicji nowej zmiennej;
 - *Wstaw definicję zmiennej systemowej* - dodanie rekordu definicji zmiennej systemowej - **uruchamiany jest edytor z listą obsługiwanych zmiennych systemowych**;
 - *Wstaw definicję zmiennej z kanału OPC*
 - *Zmień wartości* - po zaznaczeniu wybranych wartości danego atrybutu w tabeli z definicjami zmiennych umożliwia zmianę tych wartości;
 - *Zamień zaznaczone* - po zaznaczeniu wybranych wartości danego atrybutu w tabeli z definicjami zmiennych umożliwia zamianę pewnej określonej wartości atrybutu w obszarze zaznaczonych pól na nową wartość;
 - *Zamień w kolumnie* - po kliknięciu prawym klawiszem myszy w dowolnym polu danego atrybutu w tabeli z definicjami zmiennych pozwala użytkownikowi zmienić określoną wartość atrybutu w obszarze całej kolumny;
 - *Wyczyść zaznaczone;*

Natychmiastowa edycja - załączenie opcji umożliwia natychmiastową edycję bieżącej komórki po wpisaniu dowolnego znaku;
Filtrowanie
 - *Kopiuj wiersz do schowka;*
 - *Dodaj nazwy zmiennych do schowka;*

***Mirror*:**
Konfigurowanie - polecenie uruchamia okno z instrukcją uruchomienia mirroringu;
Konfigurowanie użytkownika_asix_internal - kopiowanie użytkownika _asix_internal z serwera głównego na mirror;
 - *Pokaż atrybuty - wszystkie (Shift + Ctrl + 0);*
 - *Pokaż atrybuty – bieżące (Shift + Ctrl + 9);*
Pokaż atrybuty – systemowe;

Baza definicji alarmów

 - *Pokaż* (*F8*)- uruchamia okno wyboru alarmów
 - *Generuj* (*Ctrl + H*) – generuje bazę definicji alarmów;
 - *Pokaż SQL...* - uruchamia okno wyboru alarmów SQL;
 - *Generuj SQL* (*Ctrl + J*) – generuje bazę definicji alarmów SQL;

Obszary i komputery

 - *Dodaj obszar*;

 - *Usuń obszar*;

 - *Zmień nazwę obszaru*;

Utwórz kopię obszaru...

 *- Konwertuj obszar na komputer;*

 - *Dodaj komputer w bieżącym obszarze...*;

 - *Dodaj komputer...;*

 - *Usuń komputer*;

 - *Zmień nazwę komputera*;

Utwórz kopię komputera... - tworzy kopię komputera zaznaczonego w chwili wykonywania polecenia w bloku *Obszary i komputery.*

Utwórz kopię komputera - terminal... - tworzy kopię terminala zaznaczonego w chwili wykonywania polecenia w bloku *Obszary i komputery.*

 *- Konwertuj komputer na obszar;*

Importuj komputer/obszar z pliku... - powoduje import komputera lub obszaru z pliku XML (do którego uprzednio wyeksportowano wskazany komputer/obszar np. z pliku XML parametryzującego inną aplikację systemu Asix;

Eksportuj komputer/obszar z pliku... - powoduje wyeksportowanie definicji komputera/obszaru do pliku XML;

 - *Dodaj kanał*;

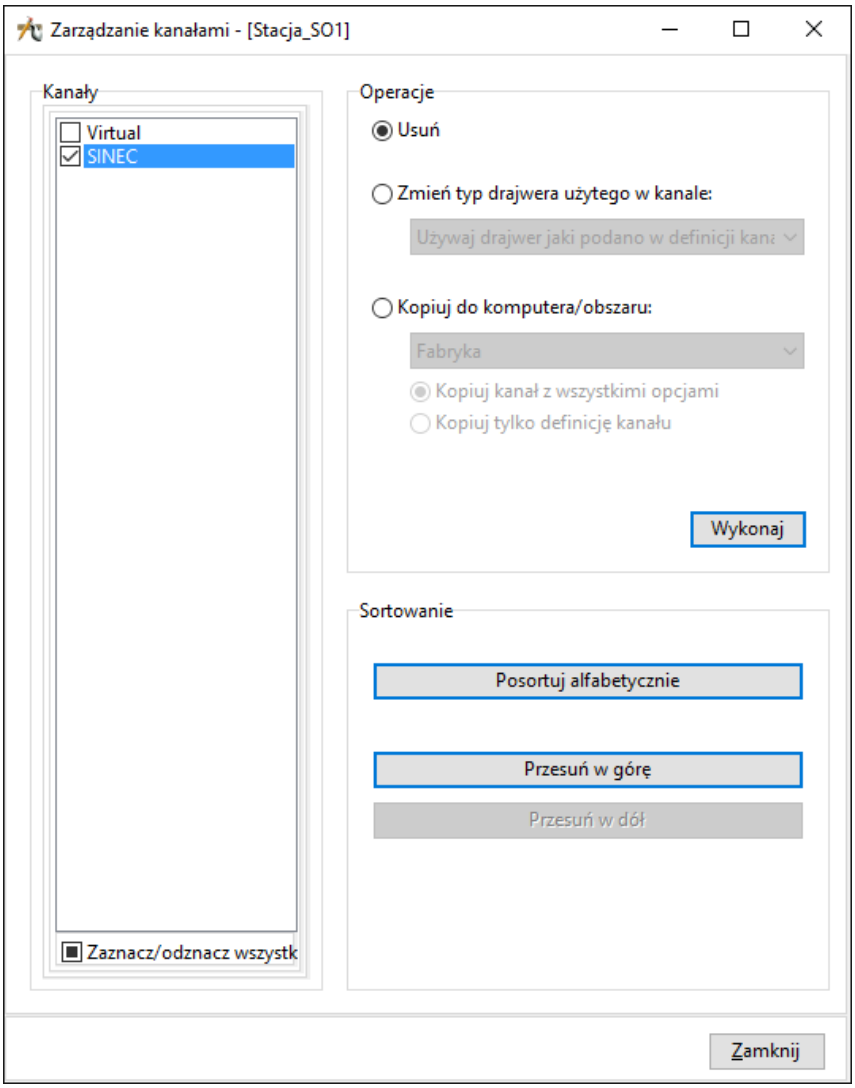
Utwórz kopię kanału...

 - *Zmień definicję kanału* – umożliwia zmianę nazwy kanału i typu drajwera;

 - *Usuń kanał*;

Zarządzaj kanałami... - pozwala wykonać nast. operacje na kanałach dla danego komputera/obszaru:

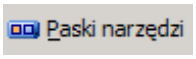
Usuń;
Zmień typ na: fizyczny, NONE lub sieciowy;
Kopiuj do komputera/obszaru;
Sortowanie



Rysunek: Okno 'Zarządzanie kanałami'.

 - *Dodaj archiwum*;
Utwórz kopię archiwum...
 - *Zmień nazwę archiwum*;
 - *Usuń archiwum*;
Zarządzaj archiwami...- pozwala wykonać nast. operacje na archiwach dla danego komputera/obszaru:
Usuń;
Zmień typ archiwum: do zapisu, do odczytu, pomocnicze lub sieciowe;
Kopiuj do komputera/obszaru;
Sortowanie
Zarządzaj opcjami - uruchamia menadżera opcji, który pozwala przeglądać wszystkie parametry ustawione dla poszczególnych obszarów funkcjonalnych aplikacji;
Importuj opcje - uruchamia menadżera opcji, który pozwala zaimportować wybrane wartości opcji z zewnętrznych plików *.xml;
 - *Informacje...*; wyświetla zależności pomiędzy komputerami i obszarami;

Widok



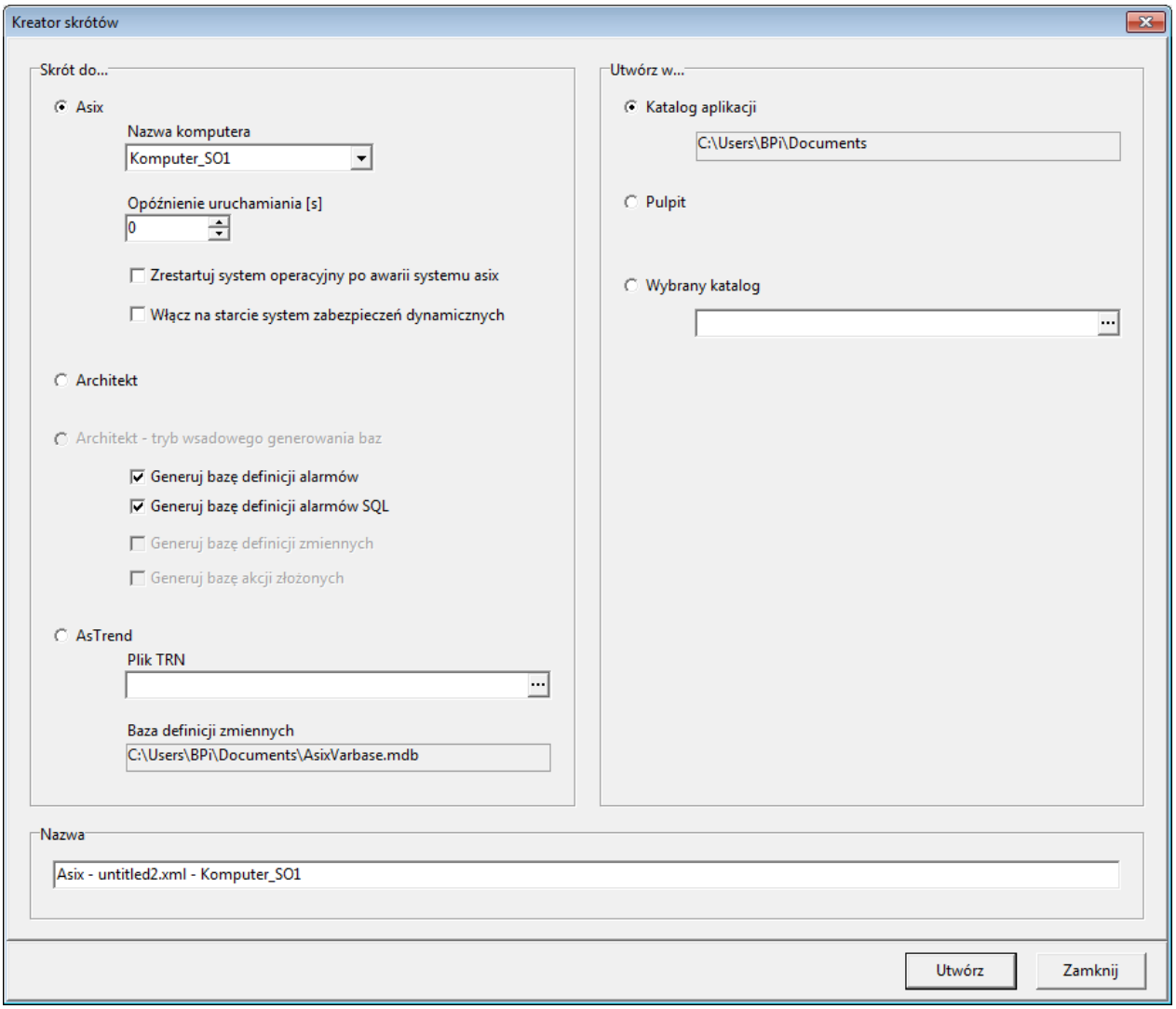
- obejmuje listę dostępnych pasków narzędzi i opcję *Dostosuj...*; *Obszary/komputery*, *Moduły opcji aplikacji*, *Moduły opcji baz danych*, *Opcje programu* - przyciski pozwalają przełączać się do bloku obszarów i komputerów oraz do bloków opcji;

 - *Pokazuj opcje zaawansowane*; aktywuje ukryte zakładki zaawansowanych parametrów w modułach opcji aplikacji;

 - ***Pokaż tylko opcje Asix.Evo - opcja redukuje zestaw parametrów dostępnych dla klasycznej wersji Asixa do tych parametrów, które są wymagane dla aplikacji projektowanej przy użyciu Asixa.Evo;***

Narzędzia

 - *Kreator skrótów*; umożliwia tworzenie skrótu do uruchamiania różnych aplikacji systemu Asix;



Rysunek: Okno do tworzenia skrótu uruchamiającego aplikację systemu Asix.

 - *Arkusz kalkulacyjny*; uruchamia okno wbudowanego arkusza kalkulacyjnego; *Edytor tabeli* - uruchamia edytor obiektu Tabela do przeglądania wartości zmiennych

 - *Wybór języka*; uruchamia program wyboru języka interfejsu użytkownika systemu Asix;

 - *AsTrend*; uruchamia program do graficznej prezentacji danych archiwalnych gromadzonych przez program archiwizacji danych ASPAD;

*Konwerter trendów 1* - uruchamia konwerter trendów służący do przekonwertowania pliku dat zawierającego definicje trendów (trendy definiowane w plikach dat dostępne były w systemie Asix do wersji 3) do postaci definicji trendów obsługiwanych przez program AsTrend.

 - *AsBase*; uruchamia program AsBase;

- *AsRaport* - uruchamia program AsRaport do edycji zapytać SQL dla raportów tworzonych w środowisku Reporting Services firmy Microsoft;

- *Askom.Data.Host* - uruchamia program do konfiguracji środowiska umożliwiającego udostępnianie do środowiska raportowego MS Reporting Services danych procesowych pochodzących z aplikacji systemu *Asix*;

 - *AsAlarm* - uruchamia program AsAlarm z załadowana baza VarDef aktualnej aplikacji;

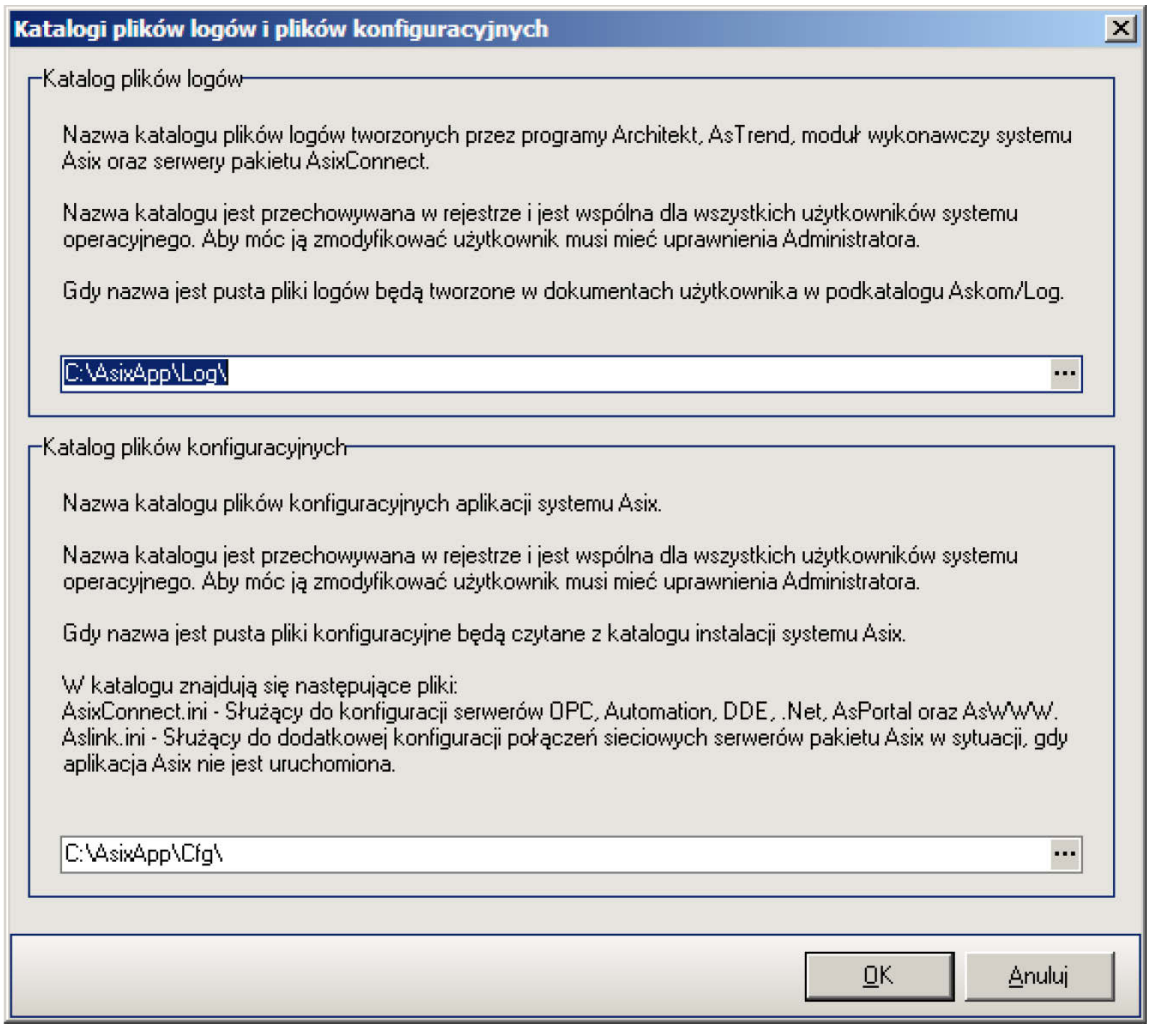
 - *Konfiguracja AsixConnect*; uruchamia konfiguratora pakietów serwerów AsixConnect;

- *Moduł sieci – opcje* – opcja uruchamia edytor ustawień sieciowych, które zapisywane są w pliku aslink.ini (stosowane gdy na komputerzen nie ma aplikacji Asixa, za to są inne programy, które korzystają z danych aplikacji Asixa (np. AsTrend, AsixConnect);

 - *Konfiguracja Alertów*; uruchamia program konfiguratora alertów;

 - *Konfiguracja AsAudit*; uruchamia program konfiguratora AsAudit;

 - *Katalogi logów i ustawień* - uruchamia program konfiguratora umożliwiającego deklarowanie katalogów dla plików logów tworzonych przez program AsTrend, serwery pakietu AsixConnect, program Architekt i moduł wykonawczy systemu Asix oraz deklarowanie katalogów plików konfiguracyjnych aplikacji systemu Asix.



Rysunek: Konfigurator katalogów logów.

 - *Menadżer pakietów Asix* - uruchamia program konfiguratora umożliwiającego przygotowanie środowiska MS Windows do uruchomienia dowolnej zainstalowanej na komputerze wersji pakietu Asix (na komputerze może być zainstalowanych kilka wersji pakietu, zaczynając od wersji 2).

 - *Wersja pakietu systemu Asix* - wyświetla okno z metryką pakietu.

Pomoc

 - poszczególne ikony pomocy uruchamiają podręczniki pomocy dla wybranych zagadnień;

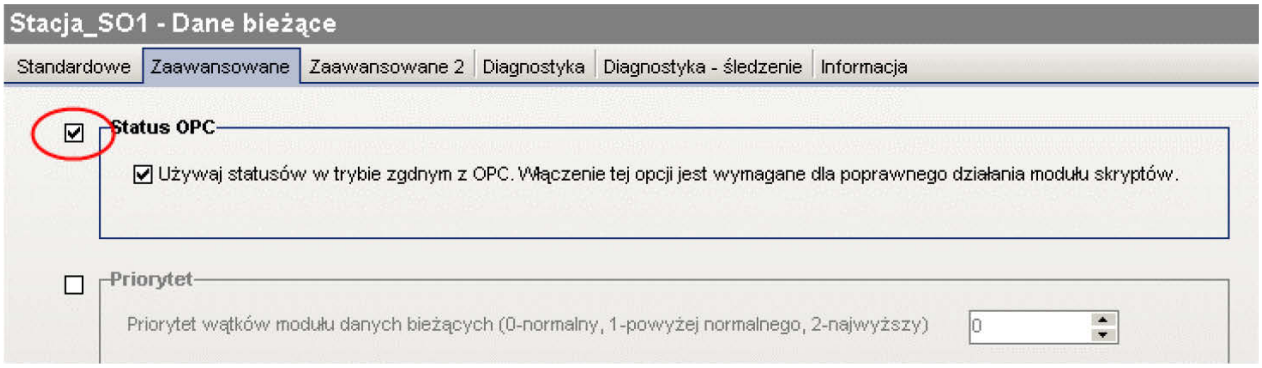
 - *Zgłoszenie usterki* lub *Zgłoszenie propozycji modyfikacji*;

 - *O programie*; wyświetla podstawowe informacje o programie.

2.1.3. Wskazówki dotyczące obsługi opcji domyślnych w modułach konfiguracyjnych

Każda opcja parametryzująca pracę aplikacji posiada zewnętrzny checkbox, znajdujący się z lewej strony opcji, który w odróżnieniu od pól deklarujących wartości parametrów, służy wyłącznie do załączania lub wyłączania wartości parametrów.

Parametry posiadające zadeklarowaną wartość domyślną, np. czas, nazwę pliku lub zaznaczone pole checkbox, należy uaktywnić załączając je zewnętrznym checkboxem.



Rysunek: Widok załączonej opcji domyślnej.

W trakcie parametryzowania aplikacji należy pamiętać o poniższych zasadach:

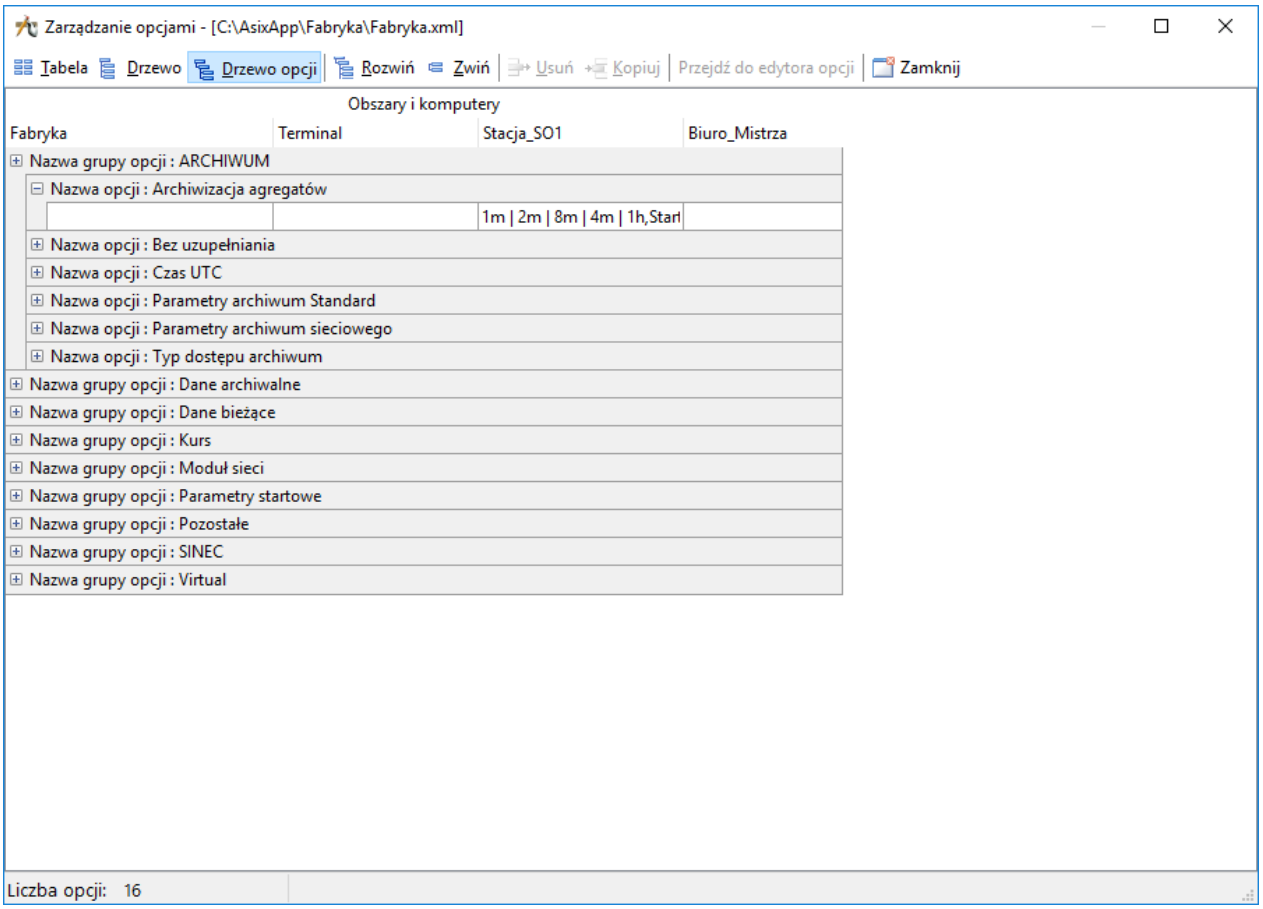
- opcje wyłączone są nieedytowalne; dopiero zaznaczenie zewnętrznego pola checkbox pozwala na zmianę wartości parametru;
- w momencie wyłączenia opcji dla *Obszaru* automatycznie przywracana jest domyślna wartość parametru;
- w momencie kiedy opcja dla *Komputera* zostanie wyłączona, parametr automatycznie przyjmuje wartość zadeklarowaną dla *Obszaru* lub (w przypadku braku takiej deklaracji w *Obszarze*) przywracana jest domyślna wartość parametru.

2.1.4. Zarządzanie opcjami za pomocą okna z zestawieniem wszystkich zadeklarowanych opcji

Wykorzystując okno *'Zarządzanie opcjami'* projektant może w prosty sposób uzyskać dostęp do wszystkich parametrów, jakie zostały ustawione w pliku konfiguracyjnym XML Serwera Danych w ramach konfigurowania poszczególnych 'obszarów' i 'komputerów'. Wspomniane okno daje wgląd do wartości zdefiniowanych opcji z jednoczesną możliwością wykonania prostych operacji edycyjnych na wybranych opcjach - dostępne są operacje usuwania i kopiowania opcji z/do wybranych obszarów/komputerów (w tym również techniką drag-and-drop).

Dalsza edycja zaznaczonej opcji możliwa jest po naciśnięciu przycisku ***Przejdź do edytora opcji***. Okno zarządzania opcjami zostaje wtedy zamknięte, a Architekt otwiera tę zakładkę modułu konfigurowania parametrów, która umożliwia zmianę wybranej opcji.

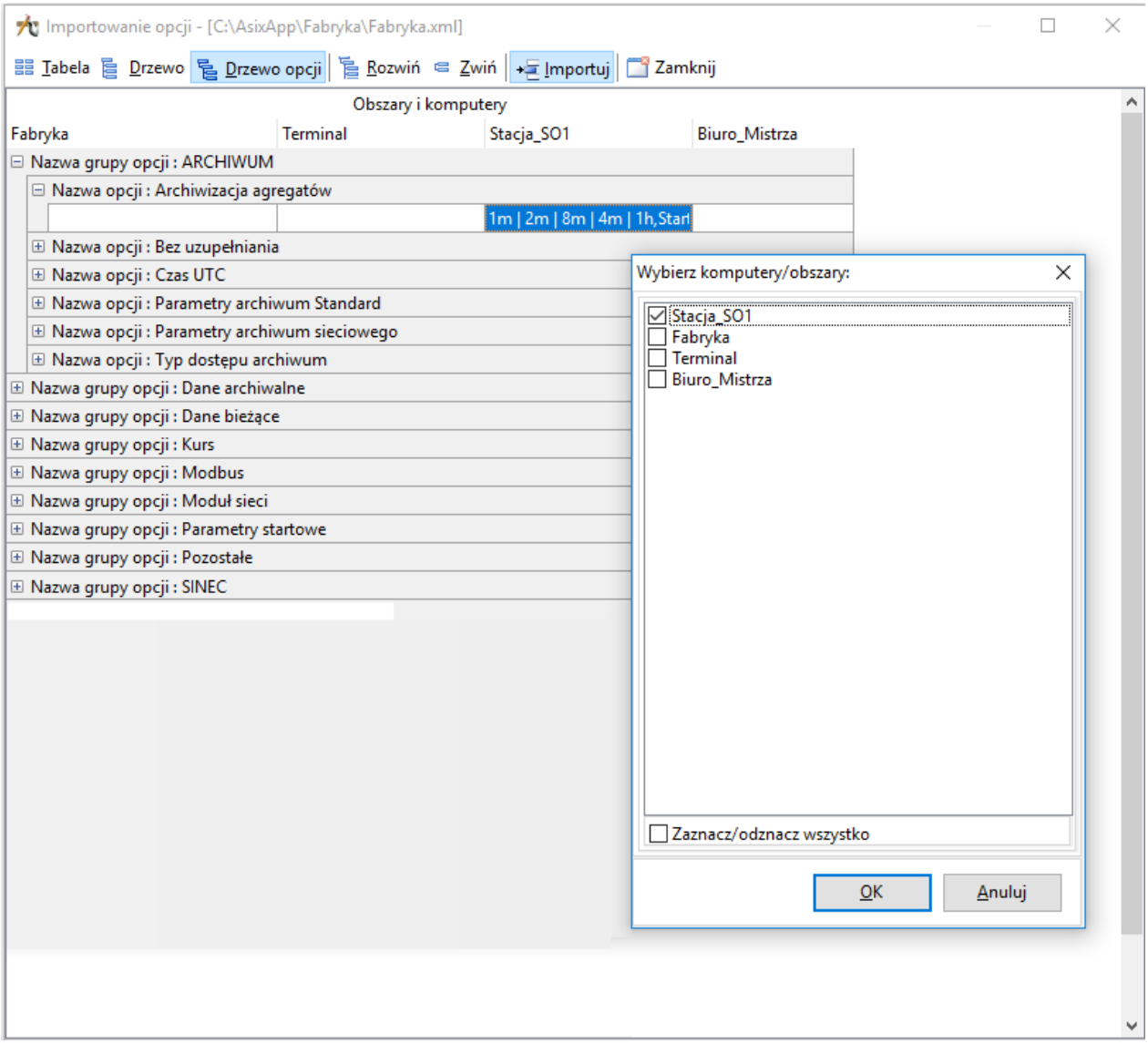
Wszystkie opcje wykorzystane do parametryzowania aplikacji wyświetlane są w oknie *'Zarządzanie opcjami'* w postaci zestawienia tabelarycznego w jednym z trzech możliwych do wyboru widoków: Tabela, Drzewo i Drzewo opcji. Okno uruchamiane jest za pomocą komendy *Zarządzaj opcjami…* z menu *Obszary i komputery*.



Rysunek: Okno 'Zarządzanie opcjami'.

2.1.5. Importowanie wybranych opcji z zewnętrznego pliku *.xml

Przygotowując aplikację, projektant ma możliwość importowania ustawień wybranych parametrów z plików *.xml Serwera Danych innych aplikacji. Komenda *Importuj opcje…* z menu *Obszary i komputery* uruchamia okno wyboru pliku XML Serwera Danych innej aplikacji, a następnie okno *'Importowanie opcji'*, które wyglądem przypomina narzędzie zarządzania opcjami pliku konfiguracyjnego XML Serwera Danych, z tą różnicą, że oglądane parametry pochodzą z pliku *.xml, z którego parametry będą importowane. Aby zaimportować do swojej aplikacji wartość dowolnego parametru, należy ten parametr zaznaczyć, przycisnąć przycisk *Importuj*, a następnie wskazać docelowy obszar/komputer.



Rysunek: Okno 'Importowanie opcji'.

2.2. Obsługa baz definicji zmiennych aplikacji systemu Asix

2.2.1 Wstęp

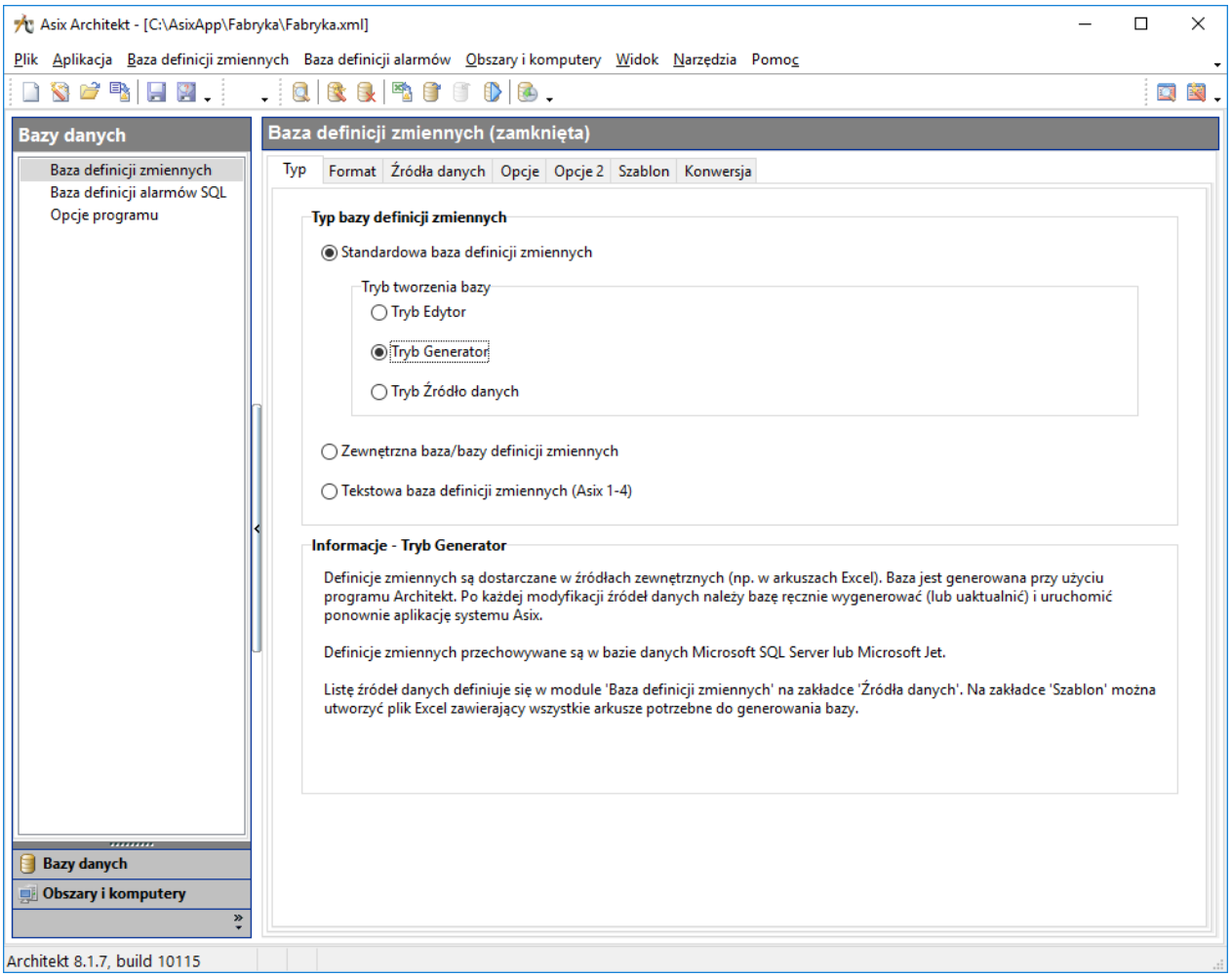
Baza definicji zmiennych VarDef stanowi podstawowe repozytorium całości informacji o wszystkich zmiennych projektowanego systemu wizualizacji. Jest fundamentem, na którym opiera się konstruowanie aplikacji asixowej.

Każdej zmiennej identyfikowanej poprzez unikalną **Nazwę** odpowiada jeden rekord w bazie definicji zmiennych. Pola tego rekordu zawierają wartości tzw. **atrybutów zmiennej**. Lista atrybutów zawiera pozycje obligatoryjne, opcjonalne oraz definiowane przez użytkownika. Atrybutami obligatoryjnymi są: opis zmiennej, jednostka, nazwa kanału komunikacji z obiektem, deklaracja sposobu przetwarzania przy pobieraniu ze sterownika, parametry archiwizacji. Atrybuty opcjonalne to np.: limity, format, zakresy. Poza powyższymi, użytkownik może również utworzyć swoje atrybuty tekstowe, indywidualnie dla każdego projektu, np.: KKS'y, czy dane montażowe.

W bazie definicji zmiennych VarDef poza definicjami zmiennych przechowywane są również informacje na temat: atrybutów grupujących, zestawów atrybutów, funkcji przeliczających, nagłówków atrybutów, wykorzystywanych w zaprojektowanej aplikacji języków narodowych atrybutów zmiennych, informacje na temat atrybutów zastępczych (dla aplikacji wielojęzycznych). Te dodatkowe dane dostępne są z poziomu modułu *Baza definicji zmiennych* > *Edytor schematu*.

Możliwe jest użycie skoroszytu programu Excel bezpośrednio jako bazy definicji zmiennych z pominięciem kroku generowania bazy definicji zmiennych. Po załadowaniu bazy skoroszyt nie jest już używany.

Dane konfiguracyjne dotyczące bazy definicji zmiennych konfigurowane są w Architekcie przy użyciu modułu *Baza definicji zmiennych* (otwieranego podłużnym przyciskiem *Bazy danych*, znajdującym się w lewym bloku okna Architekt) i zapisywane w pliku konfiguracyjnym XML Serwera Danych.



Rysunek: Okno modułu konfigurowania bazy definicji zmiennych.

2.2.1.1. Tryb tworzenia bazy danych

Program Architekt zapewnia tworzenie nowych baz definicji zmiennych systemu Asix od wersji 5 w następujących trybach:

1. baza jest tworzona i edytowana ręcznie - ***Tryb Edytor*;**
2. baza danych jest generowana na podstawie danych dostarczonych w źródłach zewnętrznych (np. arkuszach kalkulacyjnych Excel) - ***Tryb Generator***;
3. definicje zmiennych pobierane są bezpośrednio z arkusza Excel z pominięciem kroku generowania bazy zmiennych - ***Tryb Źródło danych***.

2.2.1.2. Format bazy danych

W systemie wizualizacji od wersji Asix 7 dane bazy definicji zmiennych mogą być przechowywane:

- w bazie **MSQL**,
- w bazie **Jet**,
- bezpośrednio w plikach **Excel**,

- w formacie plików tekstowych wykorzystywanym w starszych wersjach Asixa [*] (Tekstowa baza definicji zmiennych dla **Asix 1-4**)
lub
- w bazie zewnętrznej (**MSQL** lub **Jet**) utworzonej programem Architekt, ale parametryzowanej innym plikiem XML niż projektowanej aplikacji. Istnieje możliwość jednoczesnego odczytu definicji zmiennych z kilku zewnętrznych baz definicji zmiennych (patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Wstęp > Format bazy danych > *2.2.1.2.1. Łączone bazy definicji zmiennych*).

Baza Jet
Zalecanym formatem bazy definicji zmiennych jest baza Microsoft Jet. Nie wymaga ona instalowania serwera bazy danych, a sama baza zawiera się w jednym pliku .mdb, który z łatwością można przenosić pomiędzy komputerami.

Definicje zmiennych przechowywane w źródłach zewnętrznych - plikach XLS
Przechowywanie definicji zmiennych w arkuszu kalkulacyjnym jest wygodne na etapie projektowania aplikacji z uwagi na czytelność przechowywanych informacji oraz łatwość, z jaką użytkownik może wprowadzać zmiany do struktury i wartości danych.

Baza MS SQL
Baza MS SQL zalecana jest dla dużych aplikacji pracujących na ok. 20 000 i więcej zmiennych.

Bazę MS SQL należy również zastosować w sytuacji, kiedy projektowana aplikacja ma być wielojęzyczna.

_____

[*] W systemie Asix definicje zmiennych mogą być pobierane z plików tekstowych ze względu na kompatybilność z wcześniejszymi wersjami systemu. Stosowanie wspomnianych formatów przechowywania definicji zmiennych nie jest jednak aktualnie zalecane. Bazy w formacie Paradox - pliki MDB należy przekonwertować do aktualnego formatu baz danych systemu Asix.

* * *

Typ i format bazy definicji zmiennych wybierany jest na zakładkach ***Typ*** i ***Format*** modułu *Bazy danych > Baza definicji zmiennych*.

Zakładka ***Typ*** pozwala zdecydować, czy baza ma być tworzona w standardowym formacie: MBD lub MS SQL (ze wskazaniem trybu tworzenia: Edytor, Generator lub Źródło danych), czy tez w starym formacie (pliki tekstowe).

W przypadku wyboru typu bazy w standardowym formacie zostanie wyświetlona zakładka ***Format*** z wyborem formatu bazy danych: *Baza danych serwera Microsoft SQL Serwer* lub *Plik w formacie Mdb (Jet/Microsoft Access)*.

* * *

W przypadku wyboru bazy w formacie serwera MS SQL Server należy przestrzegać następujących zasad:

Serwer SQL musi mieć włączone oba rodzaje autentyfikacji: SQL Server i Windows.

W przypadku użycia lokalnego serwera SQL - konto, pod jakim pracuje serwer, musi mieć dostęp do katalogu TEMP. Najłatwiej to osiągnąć konfigurując proces serwera SQL do pracy pod kontem o nazwie LocalSystem.

W przypadku użycia zdalnego serwera SQL z włączoną zaporą, na komputerze serwera:
- dla głównej instancji musi być otwarty port tcp/1433;
- dla instancji nazwanej musi być otwarty port używany przez tę instancję;
- dla wszystkich w sieci nazw serwerów SQL musi być otwarty port udp/1434.

2.2.1.2.1. Łączone bazy definicji zmiennych

Dopuszczalnym typem bazy definicji zmiennych są łączone bazy definicji zmiennych, parametryzowane odrębnymi plikami XML.

Wymaga to:

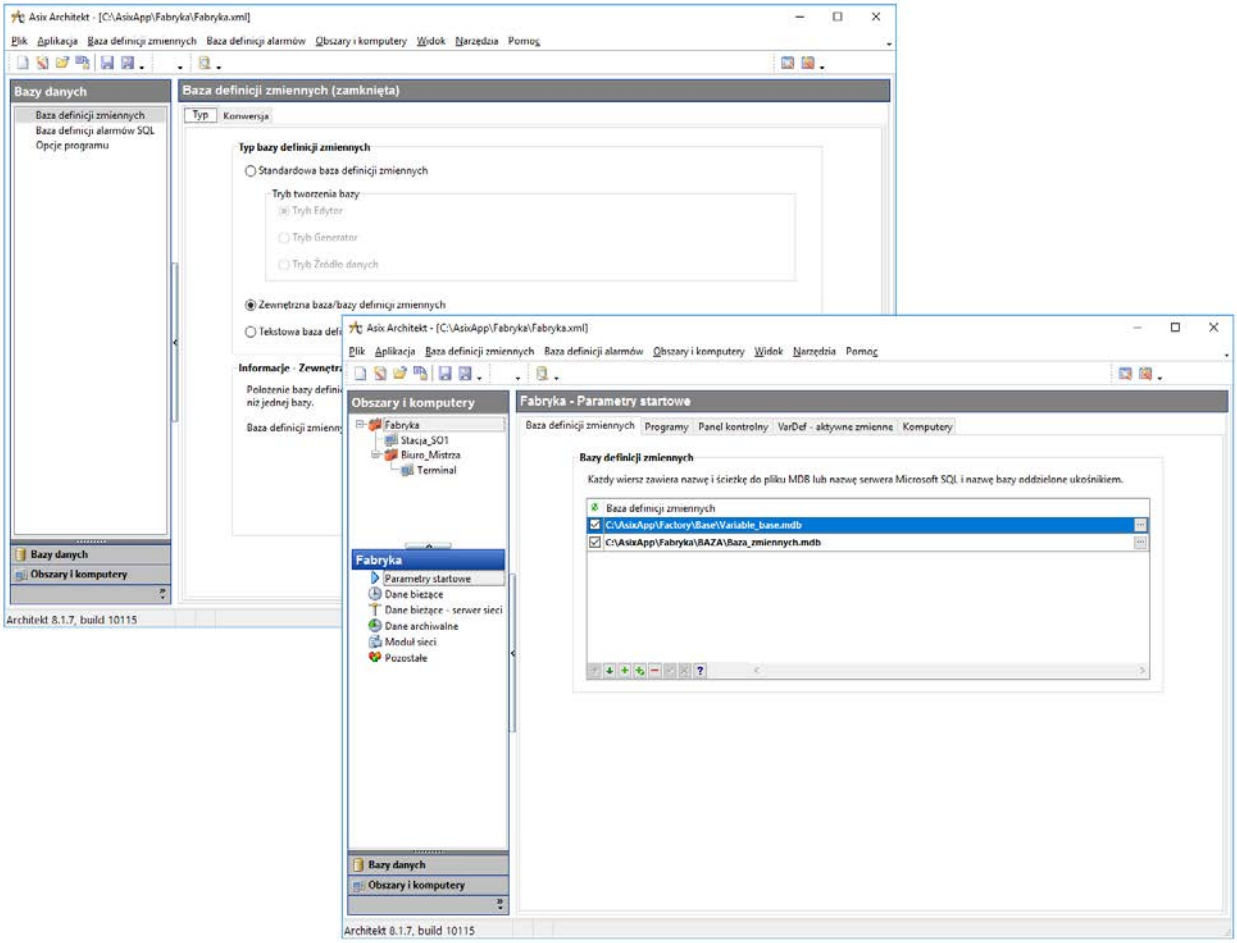
1. zadeklarowania w Architekcie odpowiedniego typu bazy definicji zmiennych:

Architekt > ***Bazy danych*** > zakładka ***Typ*** > opcja ***Zewnętrzna baza/bazy definicji zmiennych***

2. oraz wskazania zewnętrznych baz definicji zmiennych:

Architekt > ***Obszary i komputery*** > moduł ***Parametry startowe*** > zakładka ***Baza definicji zmiennych***

Deklaracja bazy obejmuje podanie w przypadku bazy MDB: nazwy i ścieżki do pliku MDB, a w przypadku bazy MS SQL: nazwy serwera Microsoft SQL i nazwy bazy.



Rys. Deklaracja łączonych baz definicji zmiennych.

2.2.1.3. Atrybuty zmiennych

Atrybuty zmiennych dzielone są na 3 rodzaje:

1. **atrybuty obligatoryjne**,
2. **atrybuty opcjonalne**,
3. **atrybuty użytkownika**.

Atrybuty obligatoryjne są to wszystkie atrybuty używane przez system Asix do uruchomienia aplikacji, np.: *Nazwa*, *Adres*, *Funkcja przeliczająca*.

Pełne zestawienie obligatoryjnych i opcjonalnych atrybutów zmiennych znajduje się w ***Załącznik 1***.

Nazwy i typy atrybutów obligatoryjnych i opcjonalnych są ustalone i niezmienne. Wartość atrybutu *Nazwa* musi być unikalna w całej bazie definicji zmiennych.

Wśród atrybutów opcjonalnych jest *ZmiennaNieaktywana*. Jeżeli wartość atrybutu *ZmiennaNieaktywana* wynosi 1, to zmienna nie jest widoczna w systemie Asix dla modułów Asmen i Aspad.

Atrybuty użytkownika są to wszystkie pozostałe atrybuty. Użytkownik sam ustala ich nazwy (atrybuty mogą być wyłącznie typu tekstowego).

Ponadto w bazie zmiennych istnieją **atrybuty wewnętrzne** tworzone na potrzeby systemu bazy definicji zmiennych i nie ujawniane użytkownikowi. Atrybuty te to: *DataUtworzenia*, *DataUsunięcia*.

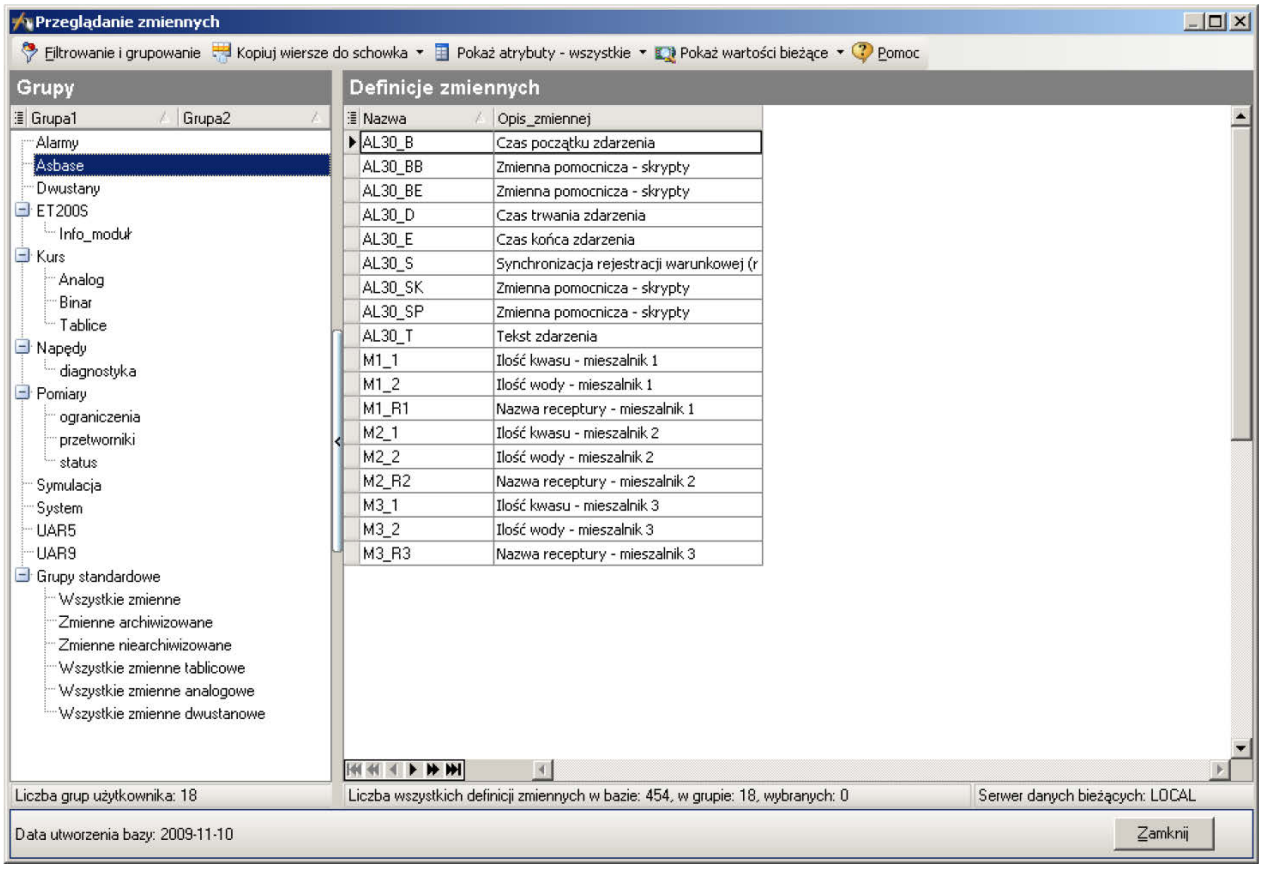
2.2.1.4. Nazewnictwo atrybutów i tabel

Nazwa atrybutu może zawierać wszelkie znaki z alfabetu łacińskiego, cyfry, spacje, itd. (Można używać znaków alfabetu lokalnego - ale tylko dla aplikacji jednojęzycznych). Nazwa ta jest używana i w bazie i w interfejsie użytkownika.

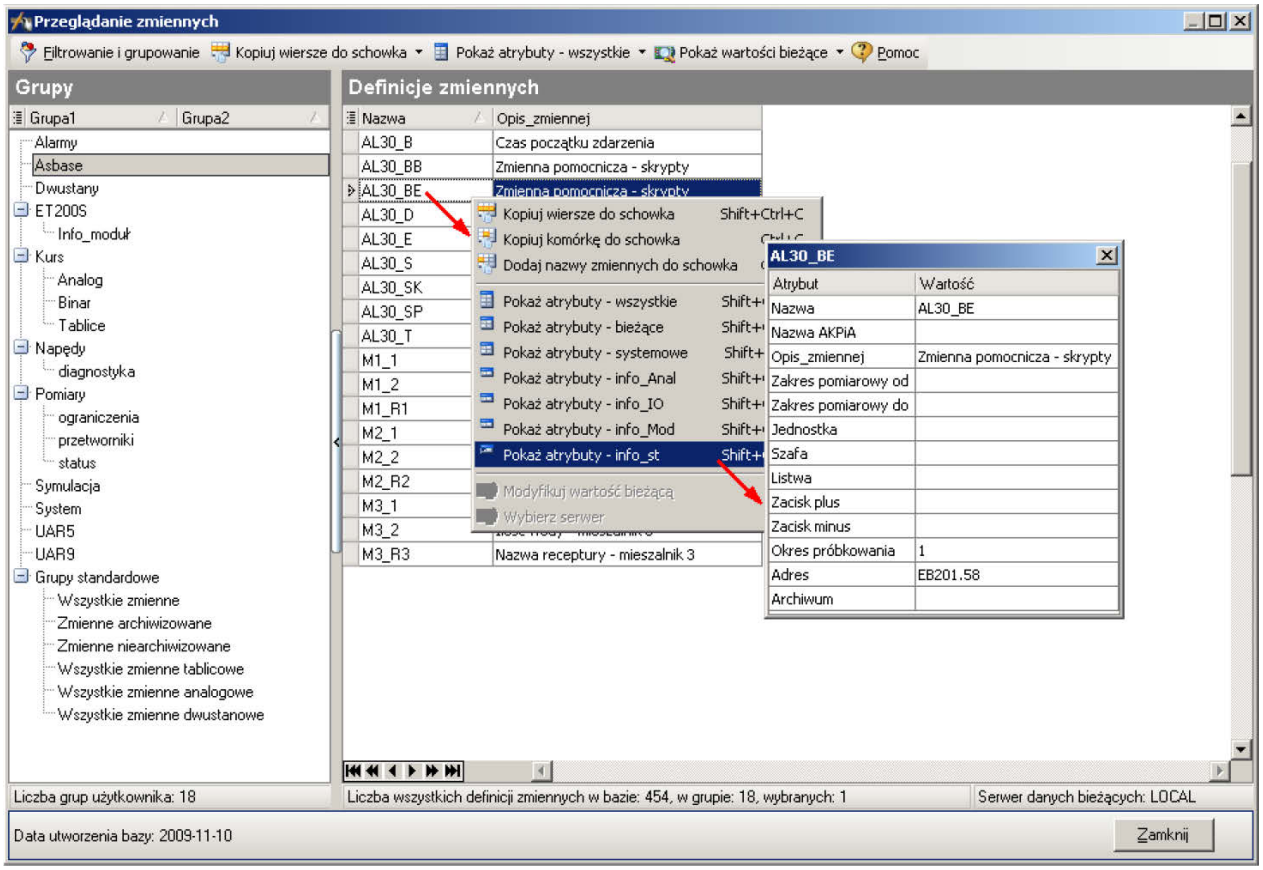
Wszystkie nazwy atrybutów systemowych oraz nazwy tabel bazy definicji zmiennych wewnętrznie są w języku angielskim. Nazwy te w interfejsie użytkownika pojawiają się w oknie edycji definicji zmiennych / oknie przeglądania zmiennych / oknie opisu zmiennych w zależności od wersji językowej Architekta - w języku polskim lub angielskim.

Przy imporcie danych nazwy mogą być podawane zarówno w języku polskim jak i angielskim.

Wyświetlanie lokalnych nazw nagłówków w dowolnym zadeklarowanym przez projektanta języku wymaga samodzielnego zdefiniowania tych nazw - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Tworzenie bazy w trybie Edytora > Tworzenie schematu bazy definicji zmiennych – Edytor > *2.2.2.2.4. Tłumaczenia nazw atrybutów zmiennych*.



Rys. Okno podglądu definicji zmiennych.



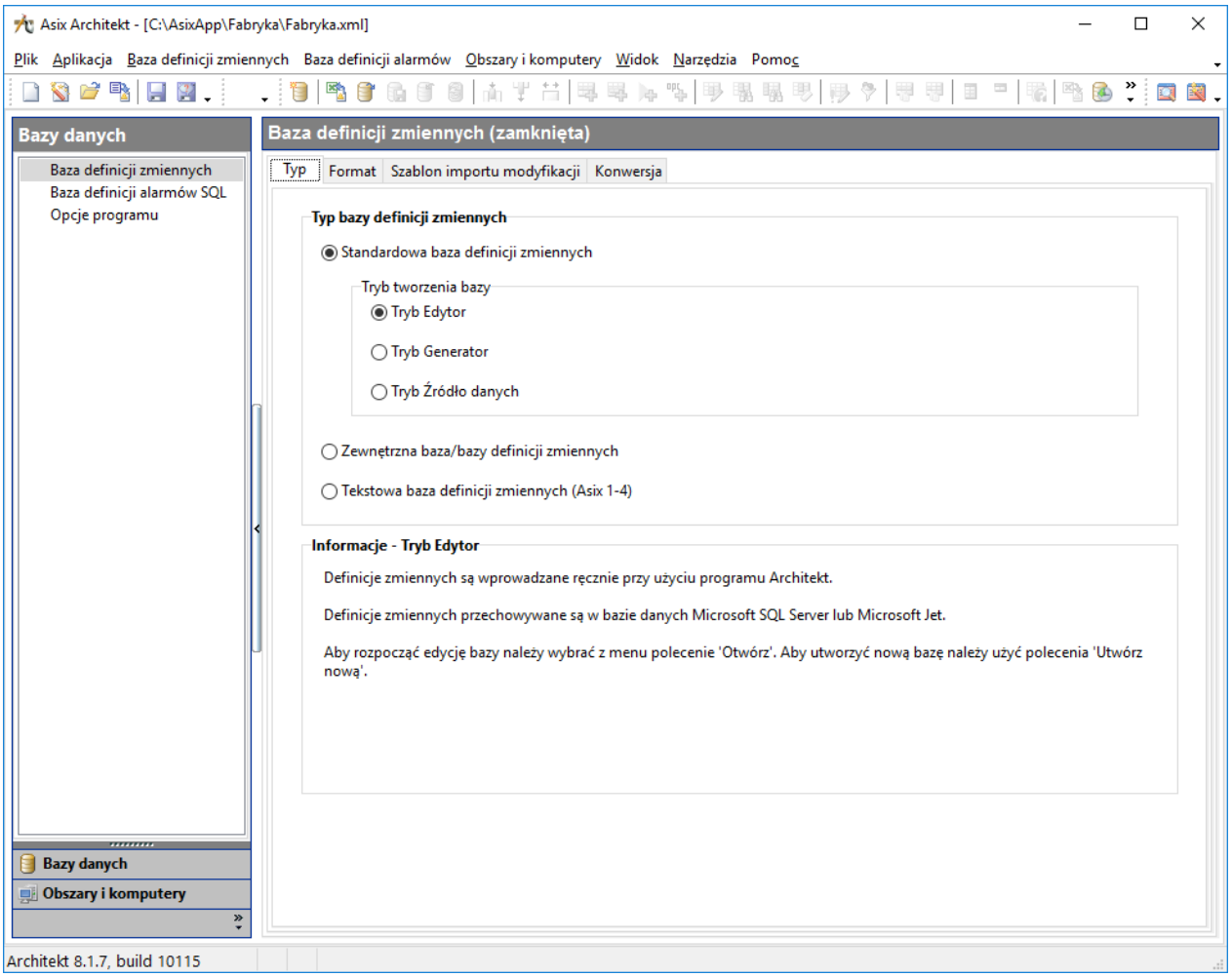
Rys. Okno opisu wybranej zmiennej - widok atrybutów z zadeklarowanego zestawu atrybutów.

2.2.2. Tworzenie bazy w trybie Edytora

Tworzenie bazy definicji zmiennych w trybie Edytora realizowane jest w następujących krokach:

1. Określ typ bazy definicji zmiennych:

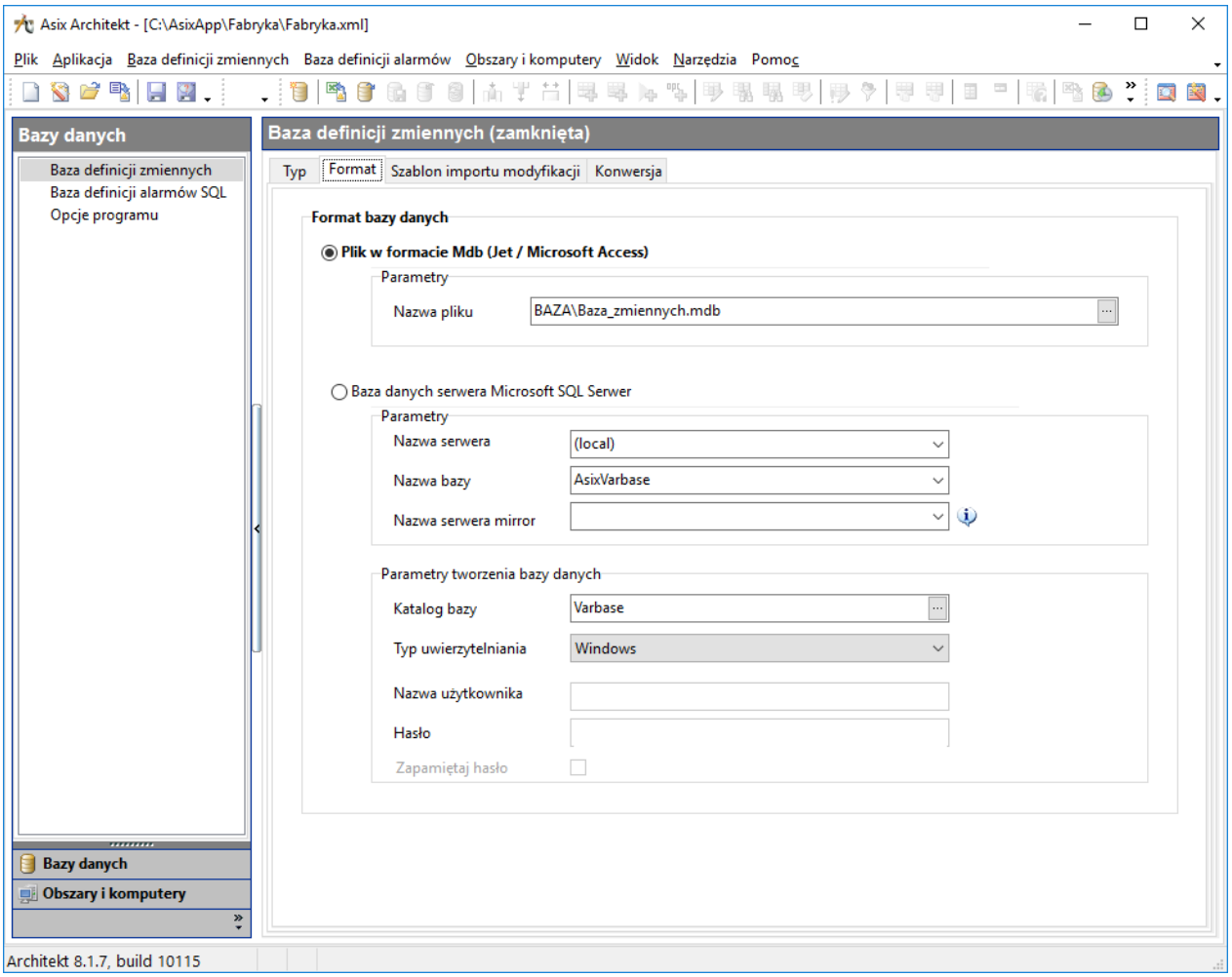
   Baza definicji zmiennych > zakładka *Typ* > tryb tworzenia bazy ustawiony na *Tryb Edytor*



Rys. Baza definicji zmiennych/zakładka Typ - Tryb Edytor.

2. Wybierz format bazy definicji zmiennych:

Baza definicji zmiennych > zakładka *Format* > format bazy danych ustawiony na: *Baza danych serwera Microsoft SQL Serwer* lub *Plik w formacie Mdb (Jet/Microsoft Access)*



Rys. Baza definicji zmiennych - zakładka 'Format'.

3. Podaj nazwę tworzonej bazy.

W przypadku baz danych MS SQL wymagane jest określenie parametrów:
- *Nazwa serwera*;
- *Nazwa bazy*;
- *Katalog bazy*;
- *Tyb uwierzytelniania*:
  - *Windows*,
  - *Serwer MSQL* - wymaga określenia *nazwy użytkownika* oraz *Hasła*.
- *Nazwa serwera mirror* - w przypadku obsługi mirroringu pole wymaga zadeklarowania nazwy serwera mirror (patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > *2.2.15. Mirroring*).

4. Wybierz polecenie *Utwórz nową* ;

5. Otwórz bazę i wprowadź dane atrybutów zmiennych przy użyciu *Edytora definicji zmiennych*

(patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Tworzenie bazy w trybie Edytora > Edycja bazy definicji zmiennych – Edytor definicji > *2.2.2.1.3. Dodawanie i usuwanie atrybutów zmiennych*);

6. Zdefiniuj schemat bazy definicji zmiennych za pomocą *Edytora schematu* (patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Tworzenie bazy w trybie Edytora > *2.2.2.2. Tworzenie schematu bazy definicji zmiennych - Edytor schematu*).

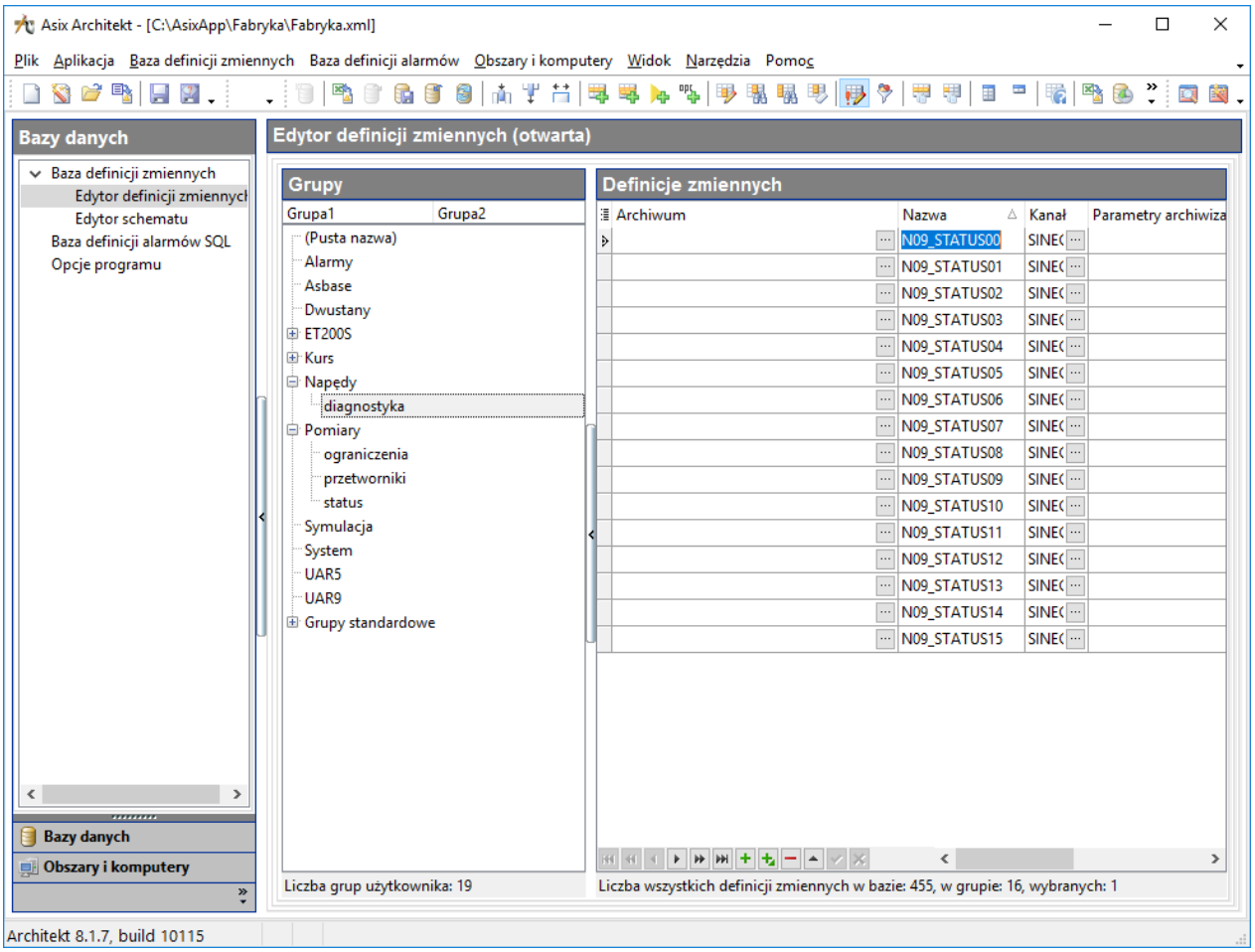
2.2.2.1. Edycja bazy definicji zmiennych – Edytor definicji

2.2.2.1.1. Dodawanie, usuwanie i modyfikowanie zmiennych

Polecenia:

- dodania rekordu nowej zmiennej powyżej wskazanego rekordu (przycisk );
- dopisania nowego rekordu zmiennej jako ostatniego w kolejności (przycisk ),
- usunięcia zaznaczonego rekordu (przycisk );
- edycji zaznaczonego pola (przycisk  lub naciśnięcie klawisza *F2* powoduje przejście do trybu edycji danego pola);

dostępne są z poziomu paska poleceń okna *Definicje zmiennych* w *Edytorze definicji zmiennych*.



Rys. Pasek poleceń do edycji rekordu bazy definicji zmiennych.

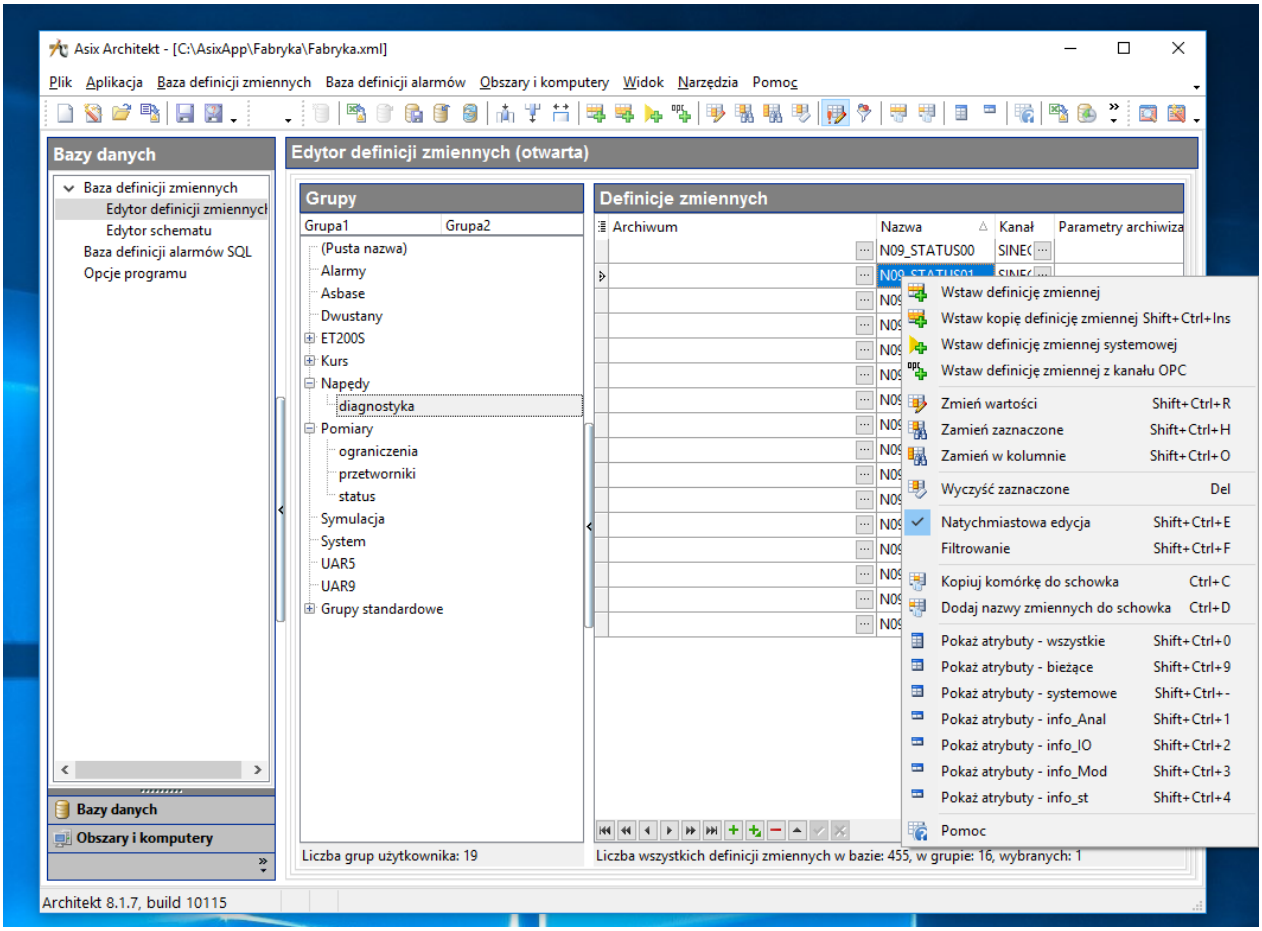
Usunięcie zmiennej nie spowoduje fizycznego usunięcia zmiennej z bazy, ale zaznaczenie zmiennej jako usuniętej – dzięki temu w tworzonym raporcie zmian będzie można ująć również informacje na temat zmiennych usuniętych (patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > *2.2.11. Eksport bazy do arkusza Excel*).

2.2.2.1.2. Dodawanie, usuwanie i natychmiastowa edycja zmiennych przy użyciu poleceń menu lokalnego / dodawanie zmiennych systemowych / wstawianie definicji zmiennej z kanału OPC

Klikając w edytorze definicji zmiennych prawym przyciskiem myszy na rekordzie (na dowolnym atrybucie) wybranej zmiennej uruchamiane jest menu lokalne umożliwiające:

- dodanie nowej zmiennej w wyniku skopiowania rekordu wybranej zmiennej - polecenie *Wstaw kopię definicji zmiennej* (*Shift + Ctr + Ins*) - rekord nowej zmiennej dodawany jest powyżej zmiennej wybranej; wartości atrybutów nowej zmiennej są identyczne z wartościami atrybutów zmiennej wskazanej, za wyjątkiem atrybutu *Nazwa*, któremu należy nadać indywidualną nazwę;

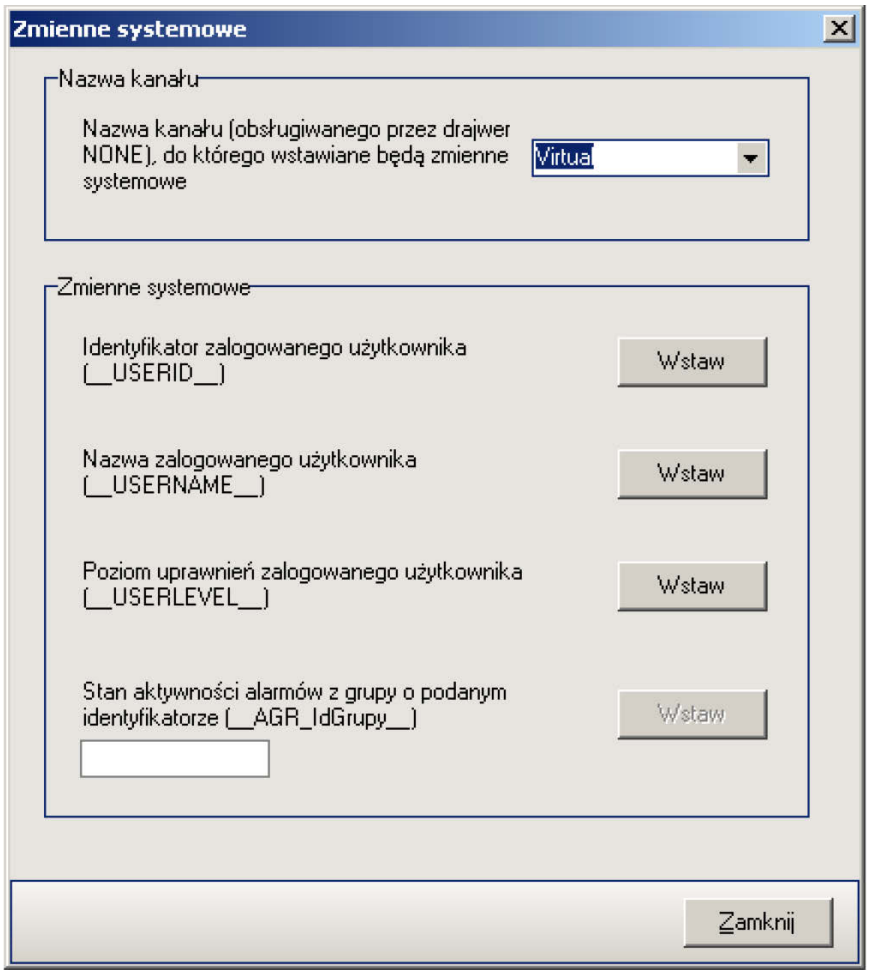
- usuwanie zmiennej - polecenie *Wyczyść zaznaczone* (*Del*) - usuwany jest rekord wskazanej zmiennej;
- natychmiastową edycję bieżącej komórki po wpisaniu dowolnego znaku - opcja *Natychmiastowa edycja* (*Shift + Ctrl +E*);



Rys. Menu lokalne rekordu zmiennej.

Menu lokalne umożliwia również dodawanie zmiennych systemowych - polecenie *Wstaw definicję zmiennej systemowej* - uruchamiane jest okno, za pomocą którego można dodać do bazy definicji zmiennych następujące zmienne systemowe:

__USERID__ - zapewnia dostęp do identyfikatora aktualnie zalogowanego użytkownika;
__USERNAME__ - zapewnia dostęp do nazwy aktualnie zalogowanego użytkownika;
__USERLEVEL__ - zapewnia dostęp do aktualnie obowiązującego poziomu uprawnień;
__AGR_IdGrupy__ - pozwala na wyświetlenie stanu aktywności alarmów należących do wspólnej grupy.

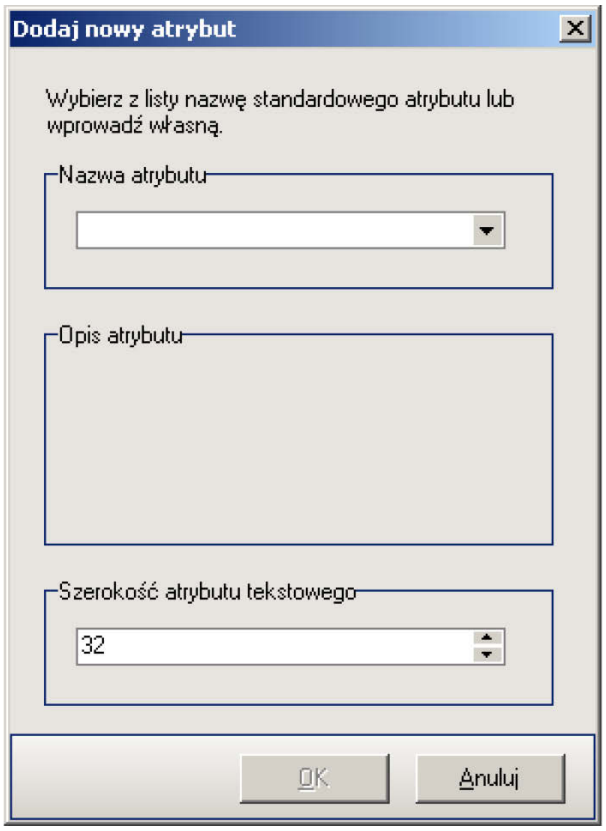


Rys. Menu lokalne rekordu zmiennej - dodawanie zmiennej systemowej.

2.2.2.1.3. Dodawanie i usuwanie atrybutów zmiennych

Nie można usunąć atrybutów obligatoryjnych.

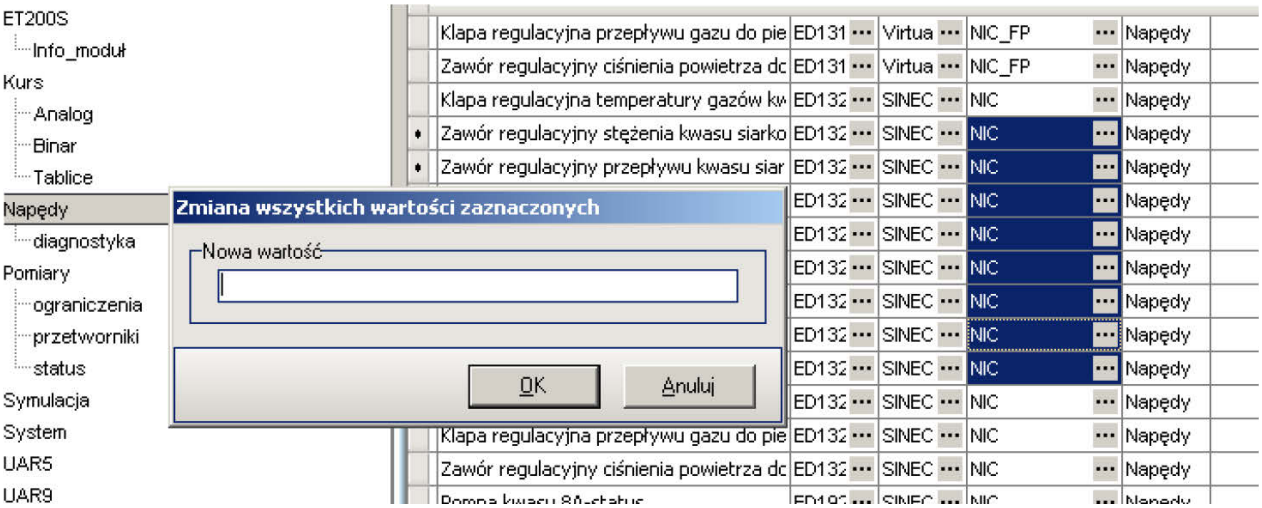
Przyciski *Dodaj atrybut* , *Usuń atrybut*  znajdują się w menu *Baza Definicji Zmiennych.*



Rys. Okno dodawania nowego atrybutu.

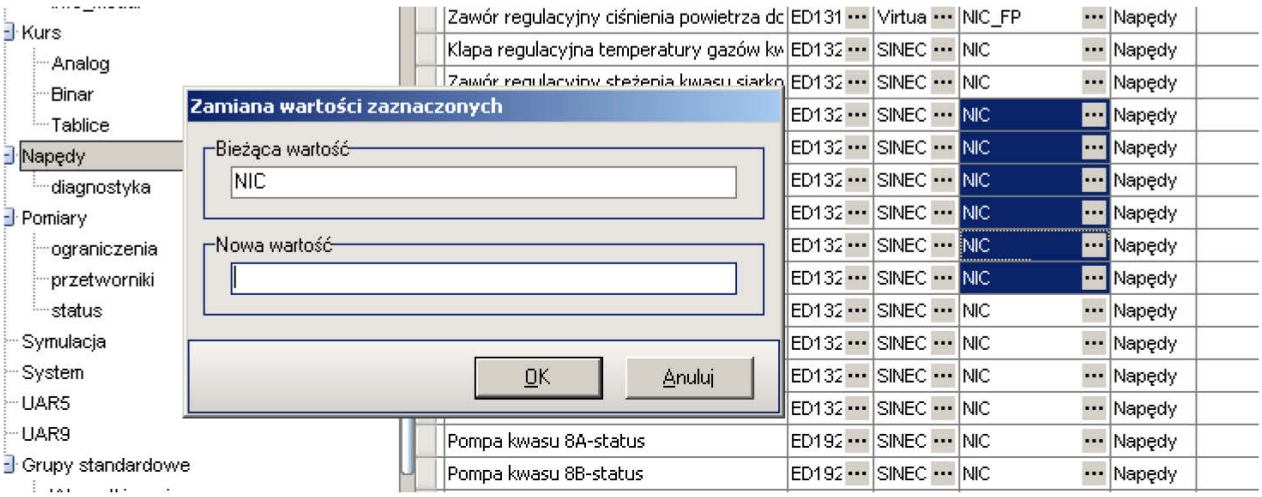
2.2.2.1.4. Edycja wartości atrybutu dla kilku zmiennych jednocześnie

Polecenie *Zmień wartości* dostępne w menu lokalnym wywoływanym po zaznaczeniu wybranych wartości danego atrybutu umożliwia zmianę tych wartości – patrz: rys. poniżej.



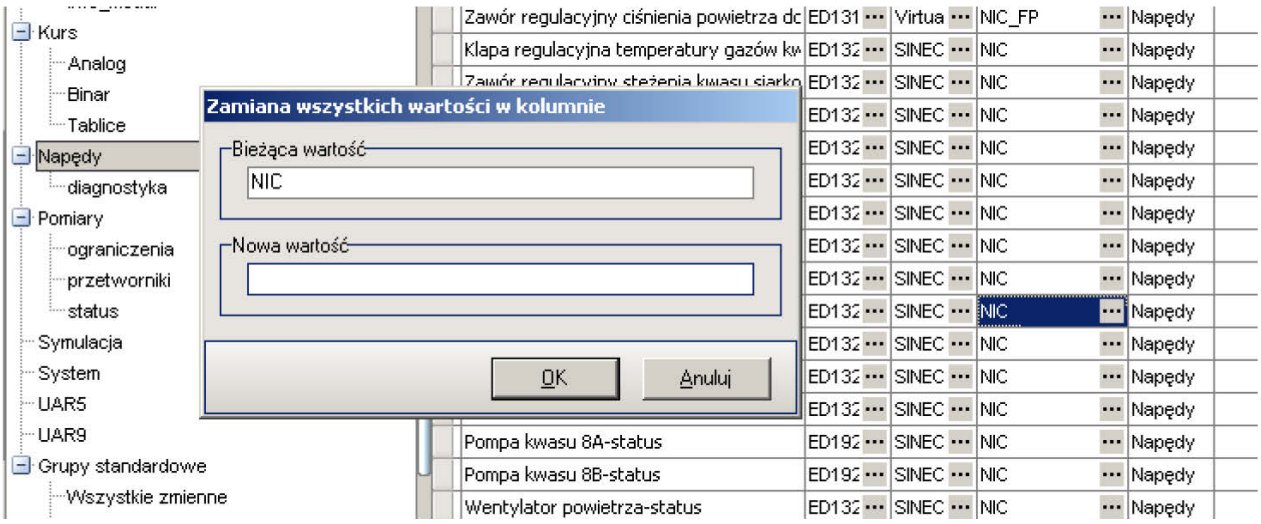
Rys. Zamiana wszystkich zaznaczonych wartości danego atrybutu.

Polecenie *Zamień zaznaczone* dostępne w menu lokalnym wywoływanym po zaznaczeniu wybranych wartości danego atrybutu umożliwia zamianę pewnej określonej wartości atrybutu w obszarze zaznaczonych pól na nową wartość - patrz: rys. poniżej.



Rys. Zamiana określonej wartości atrybutu w obszarze zaznaczonych pól.

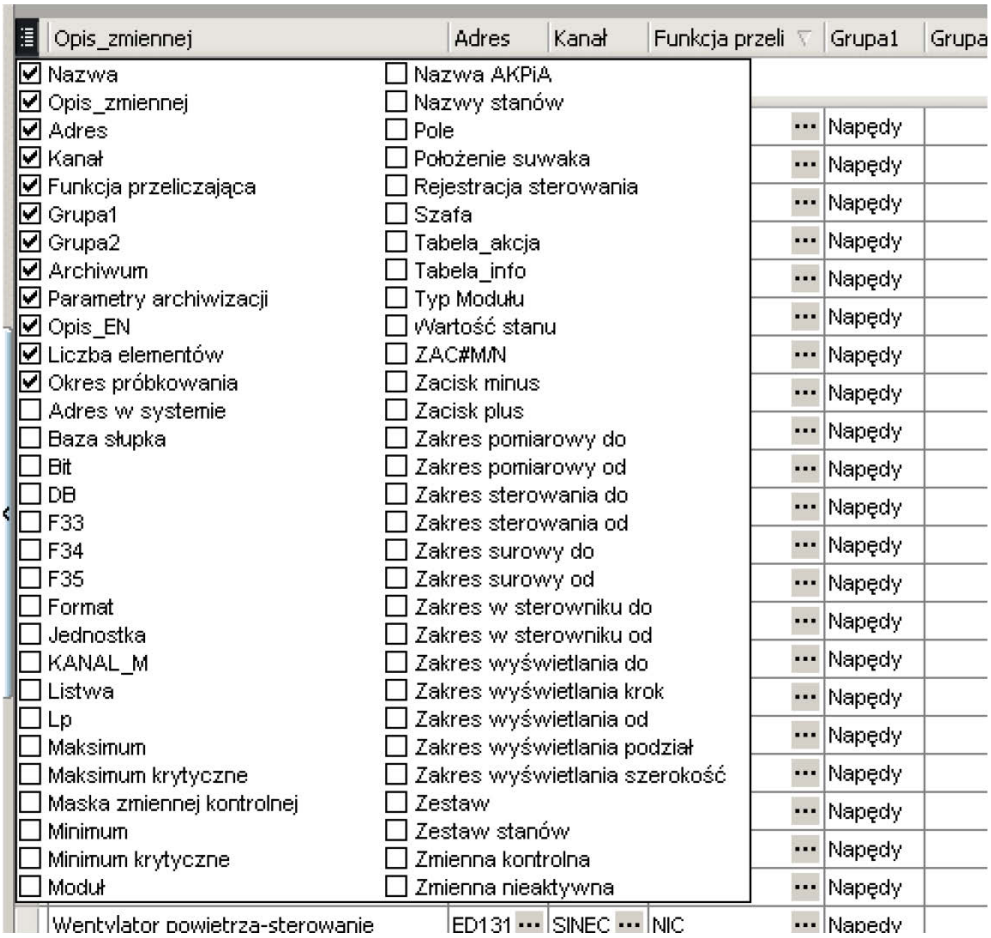
Polecenie *Zamień w kolumnie* dostępne w menu lokalnym wywoływanym po kliknięciu prawym klawiszem myszy w dowolnym polu danego atrybutu pozwala użytkownikowi zmienić określoną wartość atrybutu w obszarze całej kolumny, w której chociaż jedno pole jest zaznaczone - patrz: rys. poniżej.



Rys. Zamiana określonej wartości atrybutu w obszarze zaznaczonej kolumny.

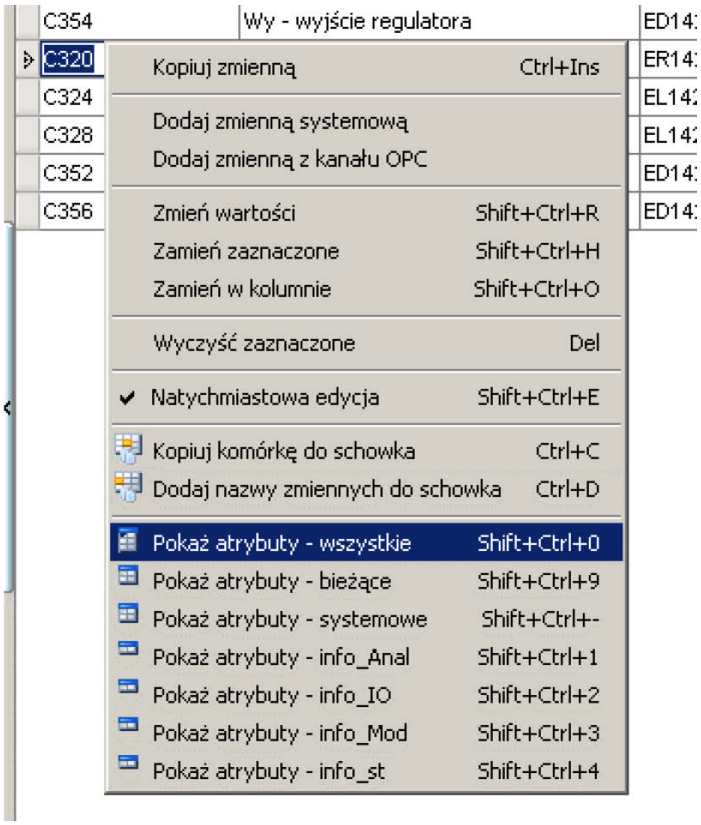
2.2.2.1.5. Wybór widocznych atrybutów zmiennych

Lista atrybutów wszystkich zmiennych wyświetlana w oknie edytora atrybutów zmiennych oraz oknie podglądu zmiennych może być określana przez użytkownika przy użyciu przycisku (pierwsza pozycja wiersza nagłówków listy zmiennych). Naciśnięcie go otwiera listę wszystkich atrybutów danej bazy definicji zmiennych.



Rys. Przycisk wyboru widocznych atrybutów zmiennych.

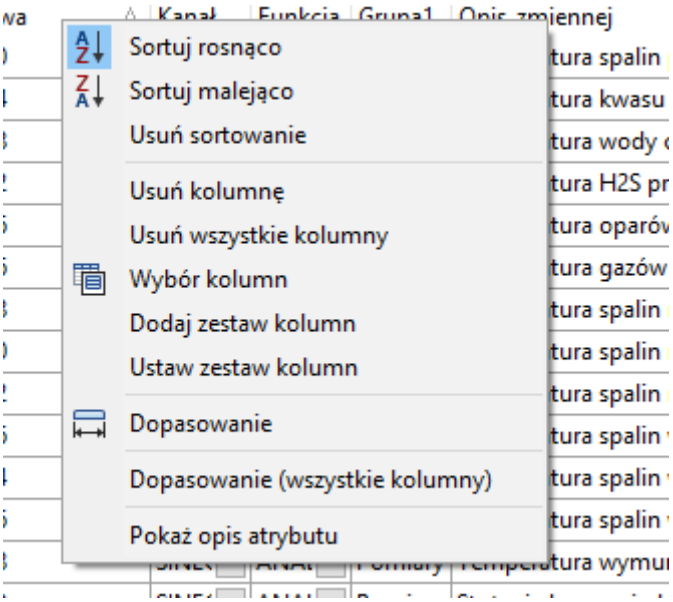
Można również podejrzeć wartości (wszystkich / bieżących lub zadeklarowanych w zestawie) atrybutów wybranej zmiennej, klikając w edytorze definicji zmiennych lub oknie przeglądania zmiennych prawym przyciskiem myszy na rekordzie wybranej zmiennej (patrz: *rysunek poniżej*).



Rys. Menu lokalne dostępne dla rekordów zmiennej.

Po kliknięciu prawym przyciskiem myszy na dowolne pole wiersza nagłówków pojawia się menu lokalne nagłówków umożliwiające:

- sortowanie wg wartości rosnącej wskazanego atrybutu - polecenie ,
- sortowanie wg wartości malejącej wskazanego atrybutu - polecenie ,
- usunięcie efektu sortowania,
- usunięcie kolumny,
- usunięcie wszystkich kolumn (poza kolumną Nazwa),
- wybór kolumny - uruchamia okno wyboru nagłówków kolumn wszystkich zadeklarowanych w bazie definicji zmiennych atrybutów, które metodą drag-and-drop można przeciągnąć do pola grupowania lub do wiersza wyświetlanych atrybutów zmiennych - polecenie ;
- wybór zestawu kolumn - uruchamia okno z zestawami atrybutów, jakie wykorzystywane są do parametryzacji określonych modułów systemu **Asix** oraz poszczególnych obiektów, umieszczanych na maskach wizualizacyjnych aplikacji; wskazanie danego zestawu atrybutów powoduje, że w edytorze bazy wyświetlone zostaną kolumny parametrów parametryzujących pracę określonego modułu/obiektu;
- dopasowanie szerokości kolumny do wartości atrybutów - polecenie (dotyczy wybranej kolumny) lub polecenie dopasowanie (wszystkie kolumny), które dotyczy wszystkich kolumn jednocześnie;
- wyświetlenie okna z opisem atrybutu aktualnie zaznaczonej kolumny.



Rys. Menu lokalne dla wiersza nagłówków atrybutów.

2.2.2.1.6. Edytory atrybutów zmiennych

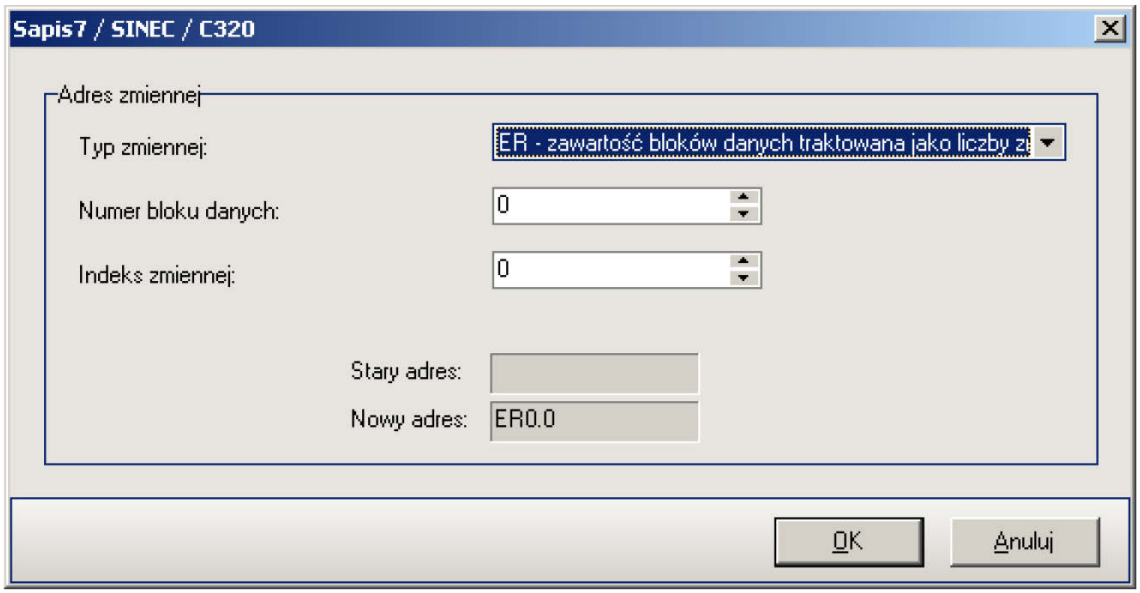
Edycję wybranych atrybutów zmiennych ułatwiają edytory uruchamiane przyciskiem ..., znajdującym się w polu atrybutu. Aktualnie dostępne są następujące edytory:

1. Adresów zmiennych
2. Parametrów archiwizacji
3. Maski zmiennej
4. Achiwum
5. Formatu zmiennej
6. Funkcji przeliczającej
7. Zmiennej kontrolnej
8. Edytor wyboru kanału
9. Edytor limitów

* * *

Edytor adresu zmiennej (uruchamiany z pola *Adres*) uruchamia okno zawierające zestaw pól umożliwiających zdefiniowanie adresu zmiennej. Zestaw pól zależy od drajwera zadeklarowanego dla kanału, do którego została przyporządkowana dana zmienna.

Edytor adresu zmiennej dostępny jest dla wybranych drajwerów. W pozostałych przypadkach należy adres zmiennej wprowadzić ręcznie na podstawie sposobu adresowania zmiennych dla danego drajwera.



Rysunek: Przykładowe okno edytora adresu zmiennej dla kanału z drajwerem Sapis7.

* * *

Edytor parametrów archiwizacji zmiennych (uruchamiany z pola *Parametry archiwizacji*) umożliwia zadeklarowanie parametrów archiwizacji podstawowej i archiwizacji dodatkowej.

W ramach danego archiwum (pole *Archiwum* w bazie definicji zmiennych) zmienna może podlegać dodatkowej archiwizacji w sytuacji, gdy oprócz archiwów przechowywanych w plikach np. dobowych, wymagana jest archiwizacja danych w bazie SQL. W innym przypadku może zachodzić potrzeba przeanalizowania dla danego wycinka czasu danych archiwizowanych z większą częstotliwością niż przewiduje to zasadnicza archiwizacja - wówczas również należy skorzystać z archiwizacji dodatkowej.

Wybór rodzaju archiwizacji odbywa się za pośrednictwem pola ***Rodzaj*** - dostępne opcje to:
- Bez archiwizacji;
- Archiwizacja podstawowa;
- Archiwizacja podstawowa i dodatkowa.

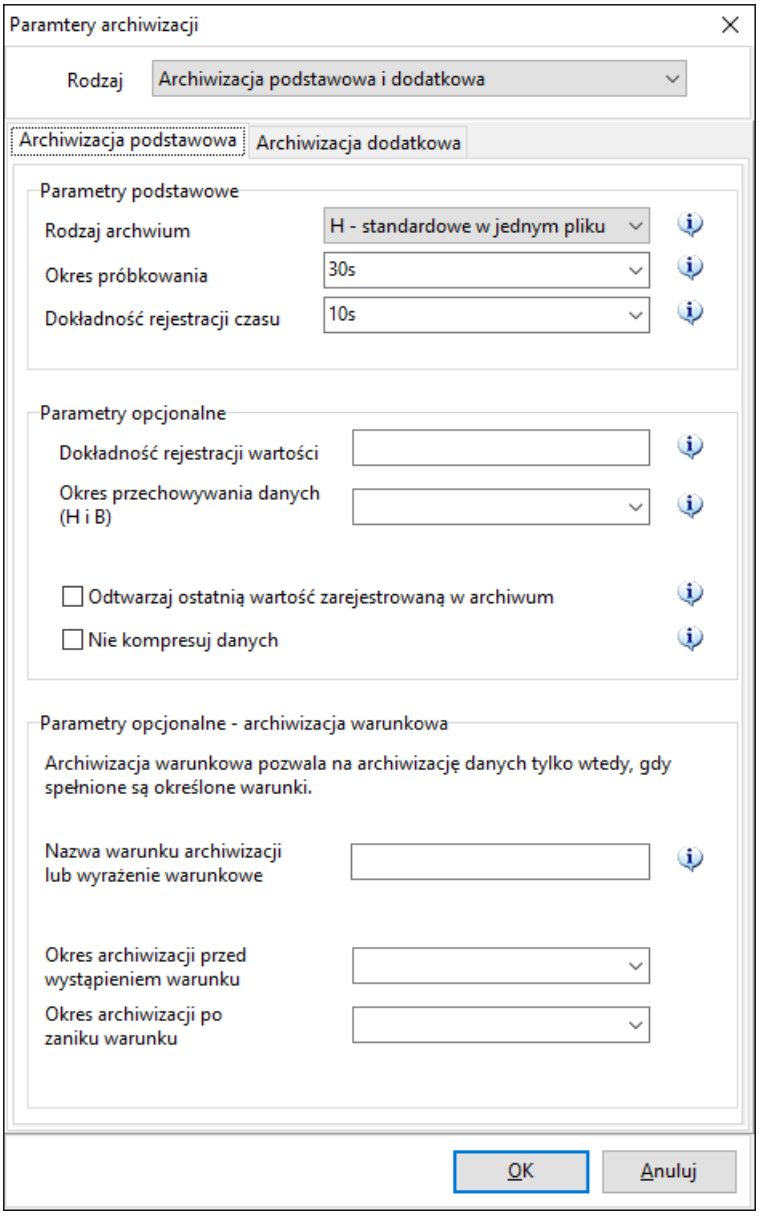
Pole ***Rodzaj archiwum*** określa typ archiwum:
Archiwum typu **D** (ang. day) - dane gromadzone w plikach dobowych;
Archiwum typu **M** (ang. month) - dane gromadzone w plikach miesięcznych;
Archiwum typu **Y** (ang. year) - dane gromadzone w plikach rocznych;
Archiwum typu **H** (ang. horizon) - dane gromadzone w jednym pliku;
Archiwum typu **B** (Baza Danych) - dane gromadzone w typowych bazach danych;
Archiwum typu **P** (ang. pattern) - dane wzorcowe;

Archiwa typu **D**, **M**, **Y**, **H** wymagają archiwum STANDARD. Stąd zmienna, która w parametrach archiwizacji ma zadeklarowany jeden z wymienionych typów, musi mieć w parametrach archiwum zadeklarowany jako rodzaj archiwum: **STANDARD** (Architekt > *Dane archiwalne* > <*nazwa archiwum*> > zakładka *Standardowe*).

Pozostałe typy archiwum **B** i **P** deklarowane są dla archiwum **MS SQL**.

Pozostałe pola edytora umożliwiają zadeklarowanie pozostałych parametrów archiwizacji:

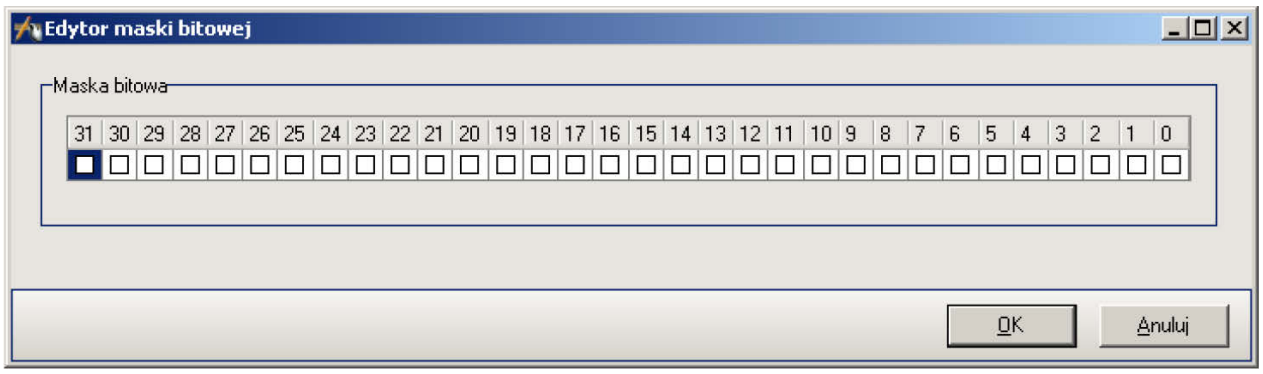
- *Okres próbkowania*;
- *Dokładność rejestracji czasu*;
- *Dokładność rejestracji wartości*;
- atrybuty rejestracji: *Odtwarzaj ostatnią wartość zarejestrowaną w archiwum*, *Nie kompresuj danych*;
- *Okres przechowywania danych* (tylko dla archiwów H i B).



Rysunek: Okno edytora parametrów archiwizacji (archiwizacja podstawowa).

* * *

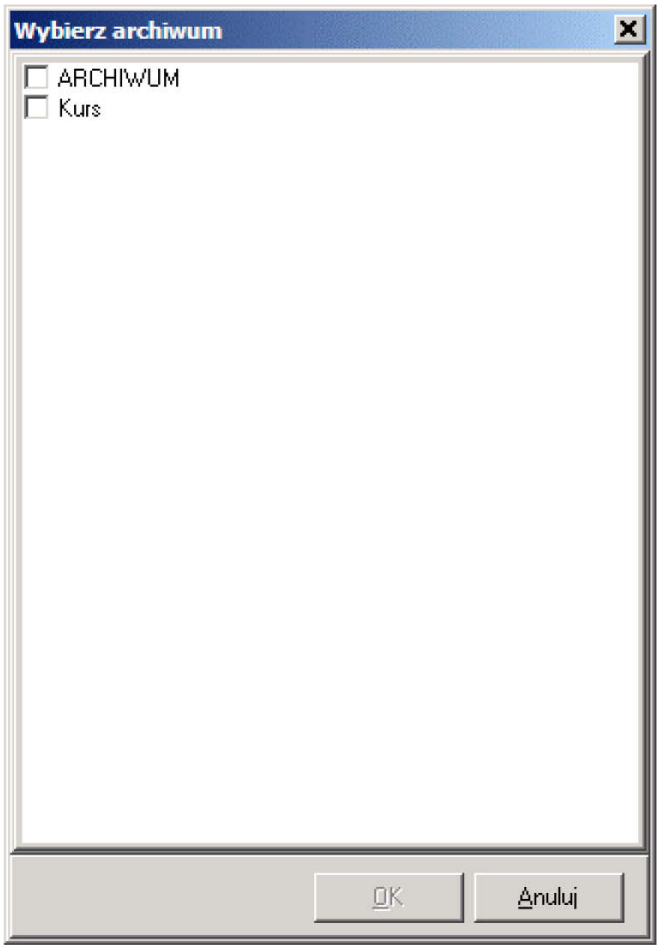
Edytor maski zmiennej (uruchamiany z pola *Maska zmiennej*):



Rysunek: Okno edytora maski bitowej zmiennej.

* * *

Edytor ***wyboru archiwum*** (uruchamiany z pola *Archiwum*):



Rysunek: Okno wyboru archiwum.

* * *

Edytor formatu (uruchamiany z pola *Format*) ułatwia definiowanie formatu zmiennej. Dostępne formaty:

- Liczba całkowita określana przez:

  Typ;
  Szerokość;
  Typ danej wejściowej;

- Liczba zmiennoprzecinkowa określana przez:

  Typ;
  Szerokość;
  Precyzja

- Data - dane wejściowa jest traktowana jako ilość sekund od 1970-01-01 00:00, 32 bity określany przez:

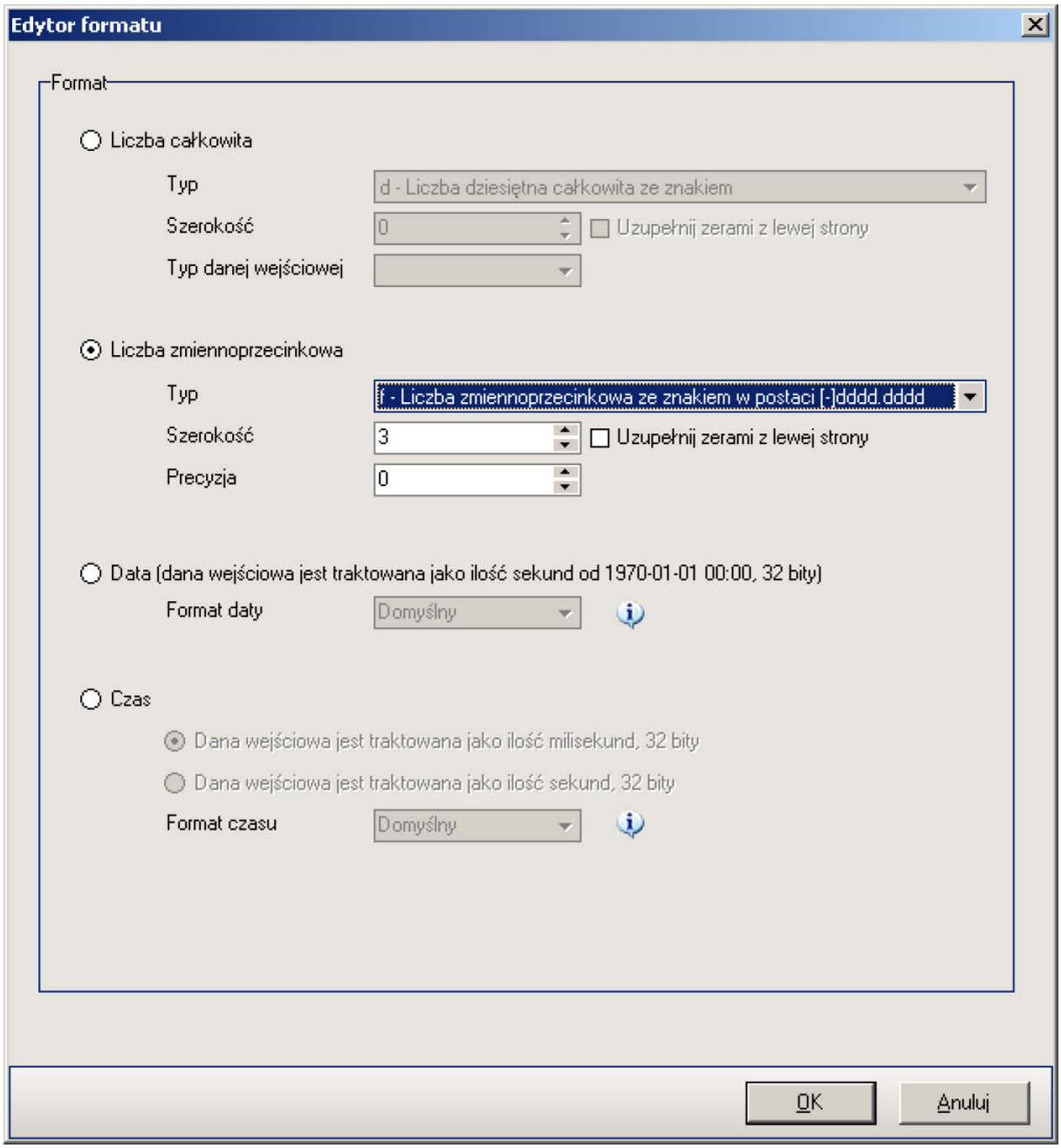
  Format daty - treścią formatu jest ciąg znaków określających separatory i elementy składowe daty w dowolnej kolejności. Elementy składowe to:
  y - rok
  m - miesiąc
  d - dzień
  h - godzina
  n - minuta
  s - sekunda

  Brak określenia formatu daty oznacza formatowanie według domyślnego wzorca: y/m/d h:n:s

- Czas określany przez opcje:

  Dana wejściowa jest traktowana jako ilość milisekund, 32 bity,
  Dana wejściowa jest traktowana jao ilość sekund, 32 bity,
  Format czasu - treścią formatu jest ciąg znaków określających separatory i elementy składowe czasu w dowolnej kolejności. Elementy składowe to:
  d - ilość dni
  h - ilość godzin
  n - ilość minut
  s - ilość sekund
  z - ilość milisekund

  Brak określenia formatu czasu oznacza formatowanie według domyślnego wzorca: h:n:s



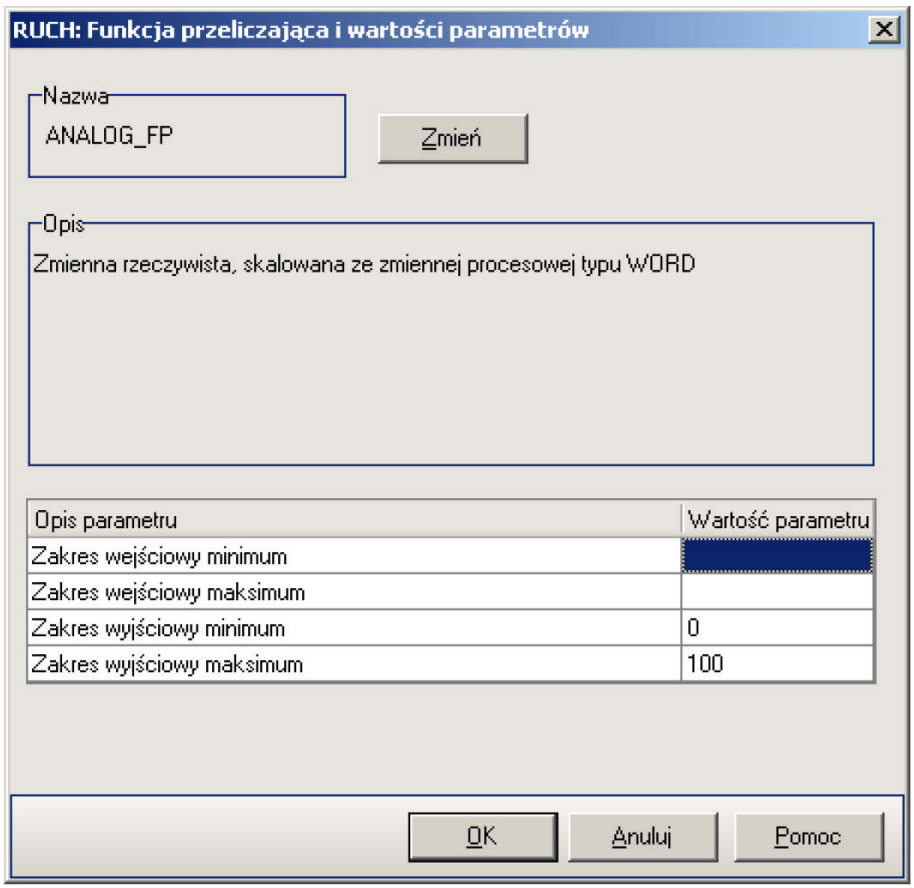
Rys. Edytor atrybutu Format.

***

Edytor funkcji przeliczających (uruchamiany z pola *Funkcja przeliczająca*), który podpowiada projektantowi składnię dla każdej z wbudowanych w system Asix funkcji przeliczających.

Pełny zestaw funkcji przeliczających - patrz: część Serwer Danych > Obsługa danych procesowych Serwera Danych > Asmen - menedżer komunikacyjny > *1.2.6. Funkcje przeliczające*.

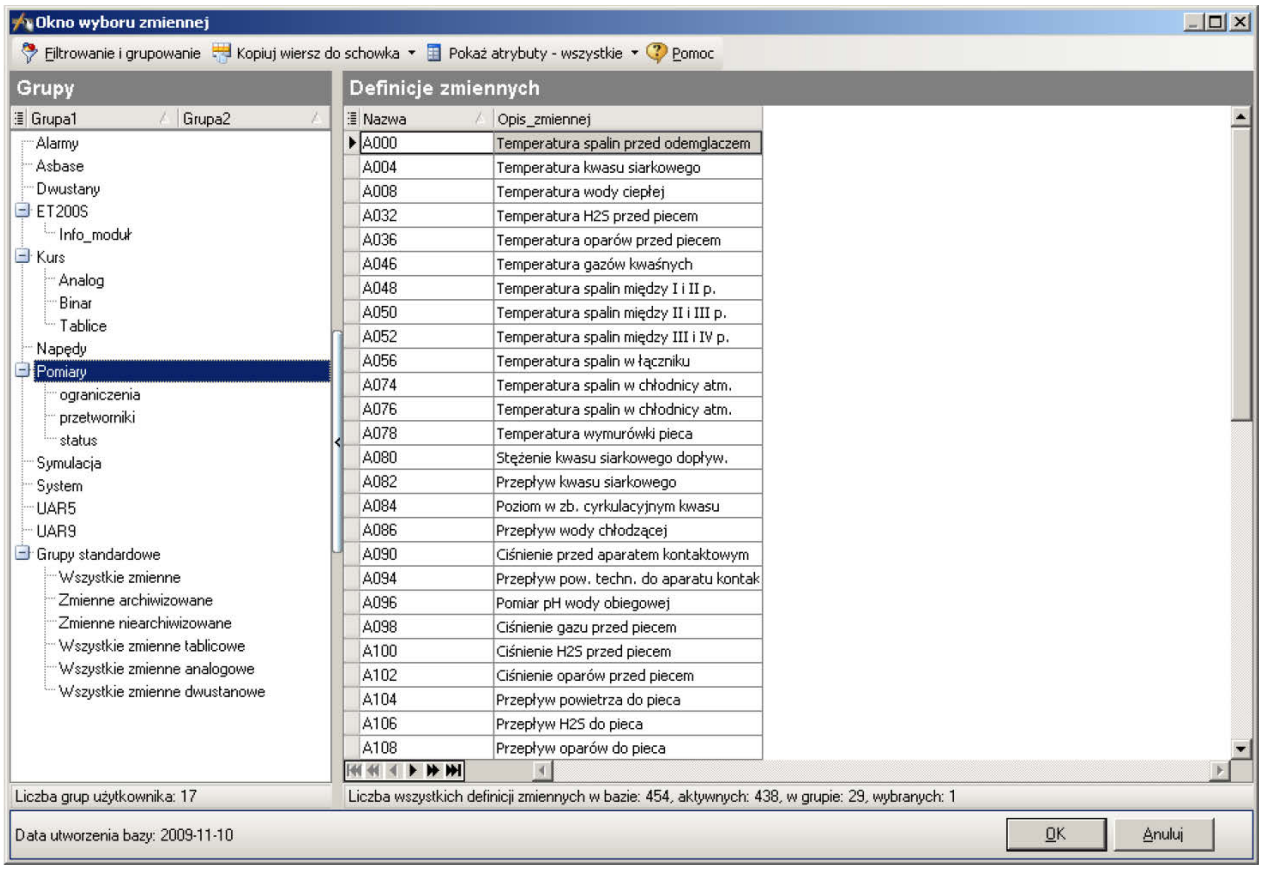
Funkcje przeliczające używane są przez moduł ASMEN przy przekształcaniu wartości zmiennych procesowych z postaci surowej na postać rzeczywistą. Każda zmienna posiada indywidualnie deklarowaną funkcję przeliczającą.



Rys. Okno edytora funkcji przeliczających.

* * *

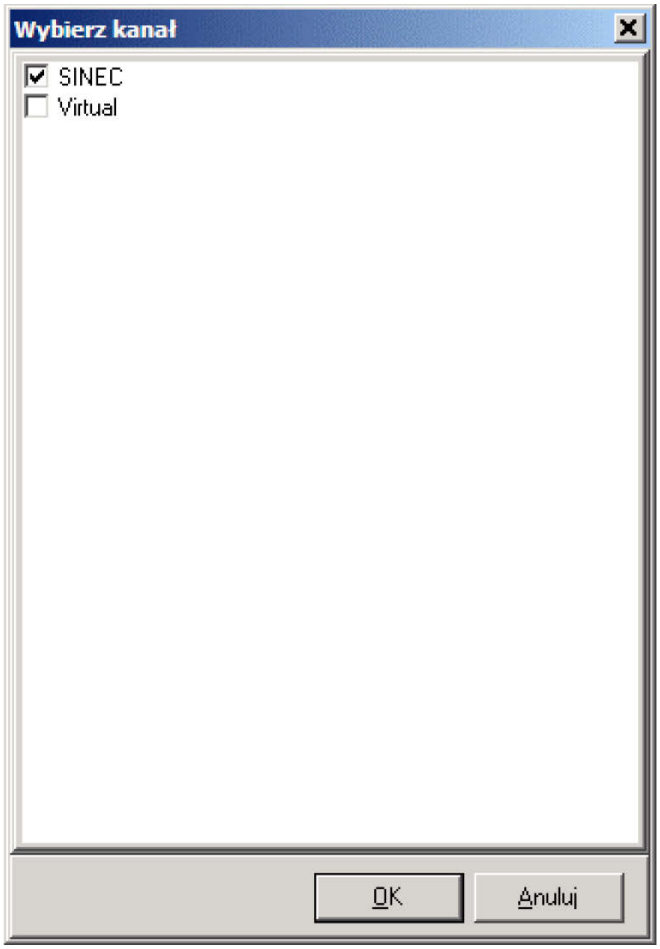
Edytor zmiennej kontrolnej (uruchamiany z pola *Zmienna kontrolna*) uruchamia okno wyboru zmiennej kontrolnej:



Rys. Okno wyboru zmiennej.

* * *

Edytor wyboru kanału (uruchamiany z pola *Kanał*):

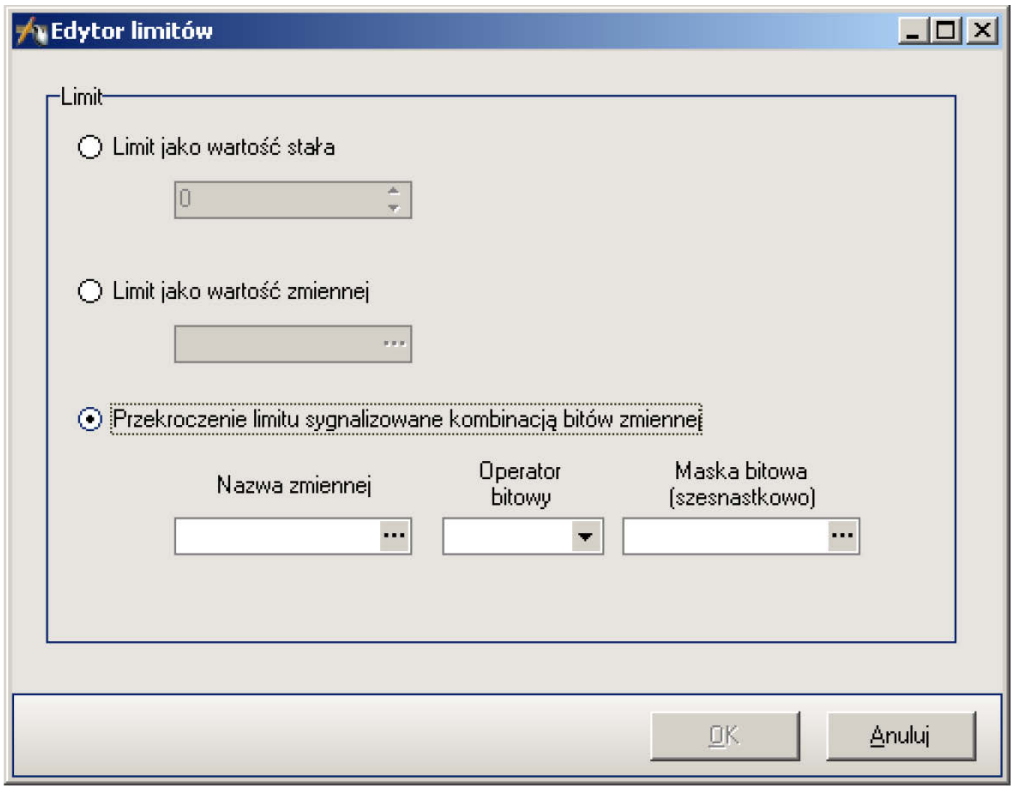


Rys. Edytor wyboru kanału.

* * *

Edytor limitów (uruchamiany z pola: *Maksimum*, *Maksimum krytyczne*, *Minimum*, *Minimum krytyczne*). Umożliwia zadeklarowanie limitu w jednej z trzech następujących postaci:

- Limit jako wartość stała
- Limit jako wartość zmiennej
- Przekroczenie limitu sygnalizowane kombinacja bitów zmiennej (*Nazwa zmiennej*, *Operator bitowy*, *Maska bitowa (szesnastkowo)*)

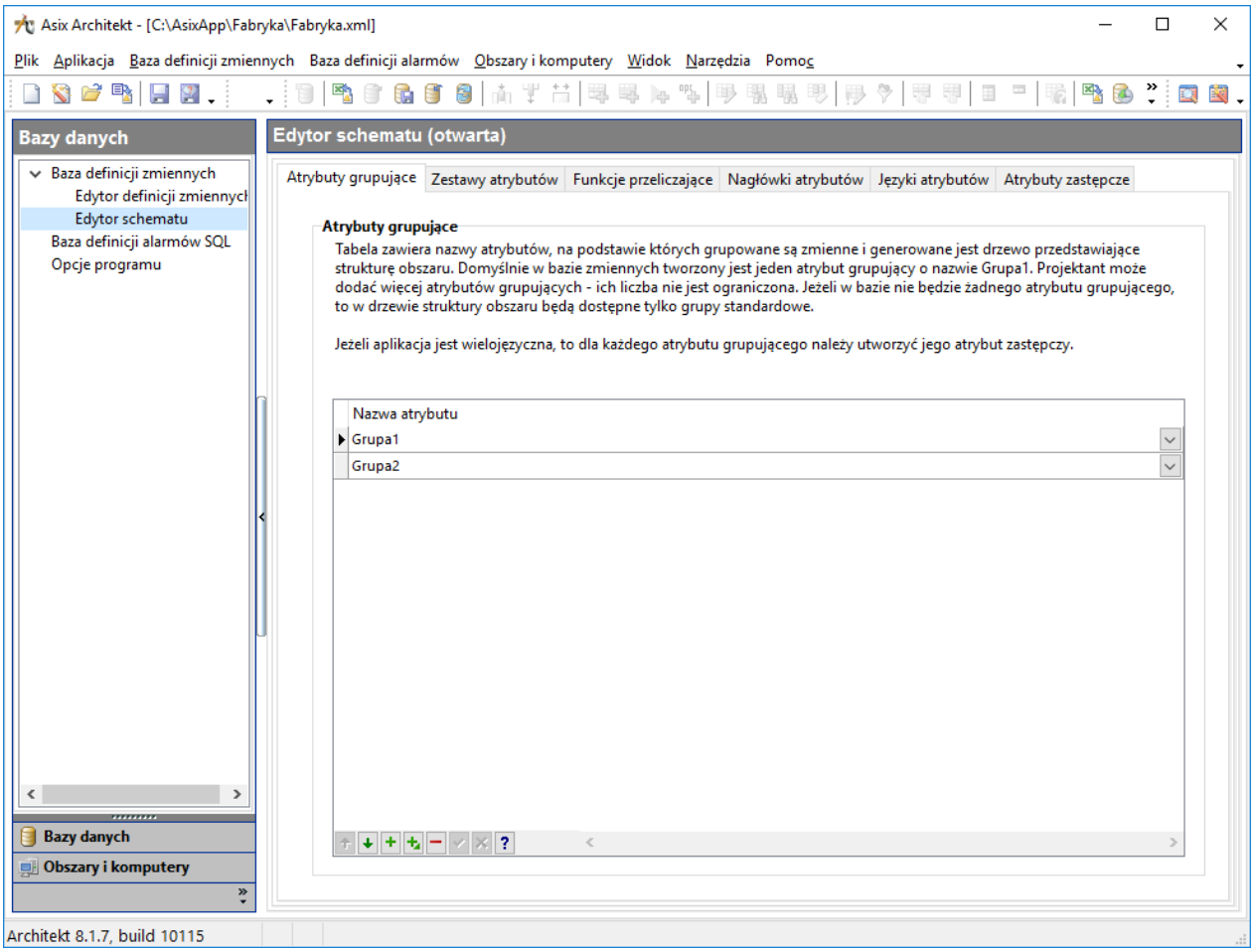


Rys. Okno edytora limitów.

2.2.2.2. Tworzenie schematu bazy definicji zmiennych – Edytor schematu

Schemat bazy definicji zmiennych obejmuje informacje dotyczące atrybutów grupujących, definicji zestawów atrybutów, parametrów funkcji przeliczających, nagłówków atrybutów, języków narodowych atrybutów zmiennych, deklaracji atrybutów zastępczych (dla aplikacji wielojęzycznych).

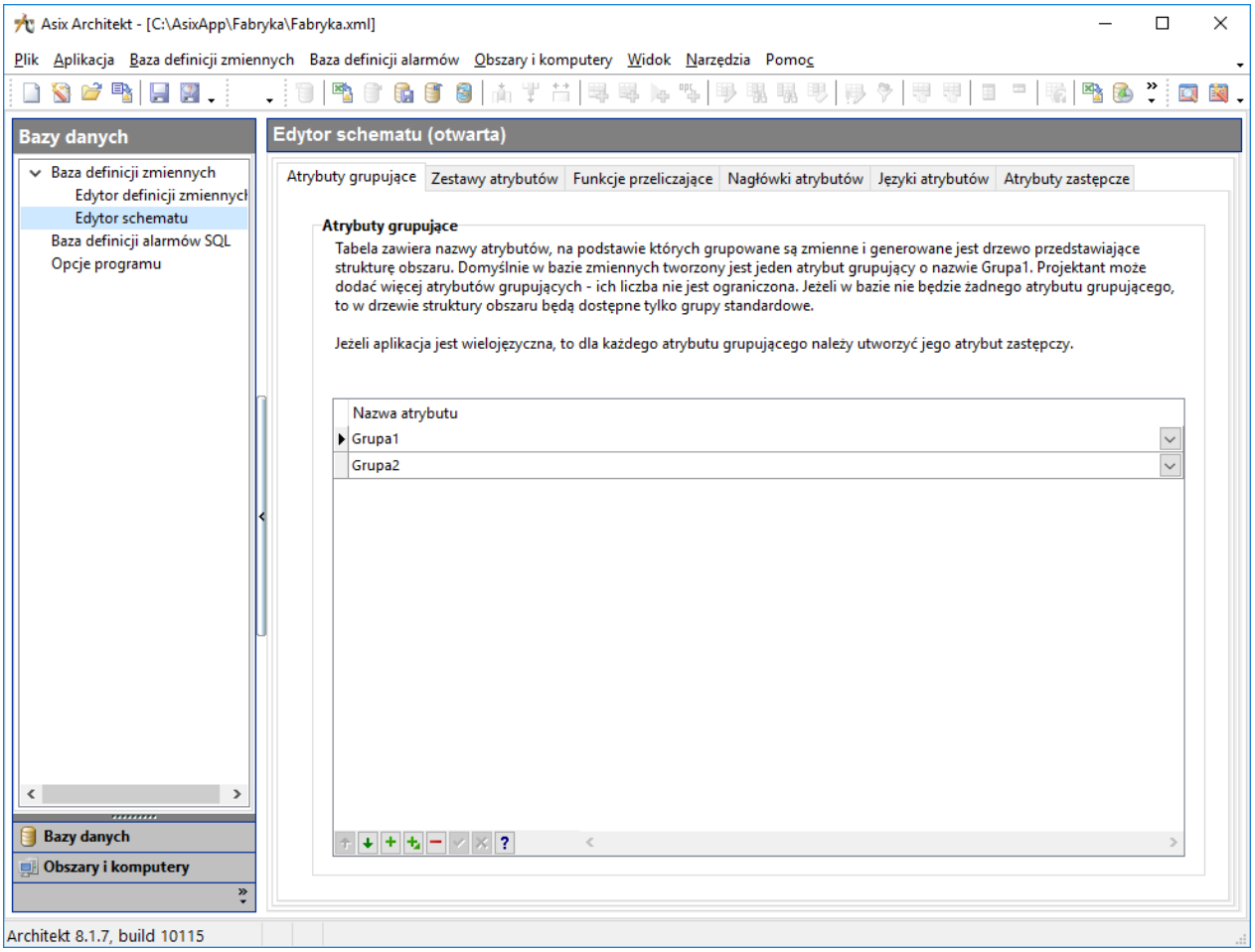
Edycja tych informacji możliwa jest za pomocą *Edytora schematu*, dostępnego w module *Baza definicji zmiennych* po otwarciu bazy.



Rys. Widok edytora schematu.

2.2.2.2.1. Grupowanie zmiennych

Grupowanie zmiennych opiera się na przypisaniu zmiennych do dowolnej liczby grup identyfikowanych przez atrybut grupujący, dodany do *Edytora schematu*, na zakładce *Atrybuty grupujące*.



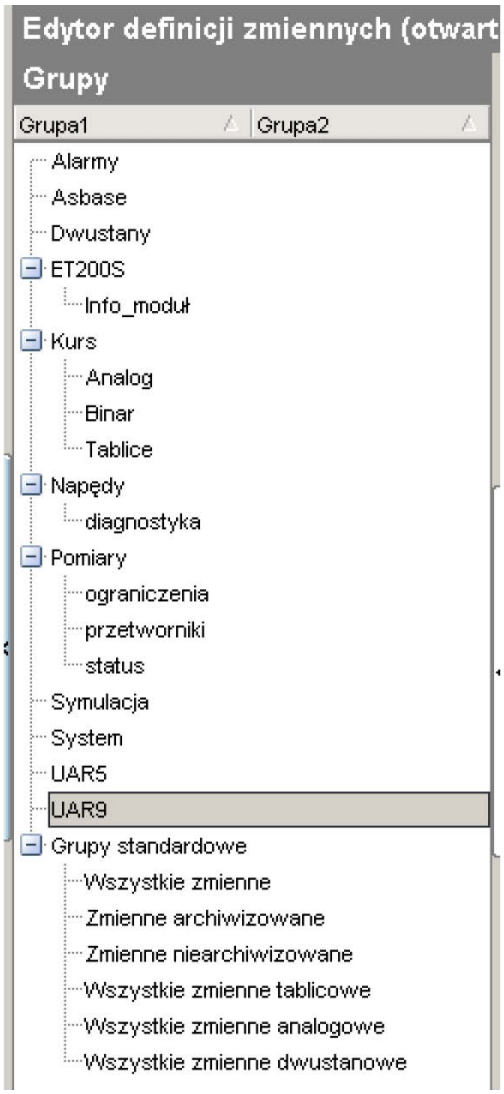
Rys. Okno definiowania atrybutów grupujących.

Aby można było zadeklarować atrybut grupujący w zakładce *Atrybuty grupujące* - najpierw atrybut ten musi zostać dodany do bazy definicji zmiennych w *Edytorze definicji zmiennych*.

Domyślnie w bazie definicji zmiennych tworzony jest jeden atrybut grupujący o nazwie *Grupa1*. Projektant może dodać więcej atrybutów grupujących - ich liczba nie jest ograniczona.

Także nazwy atrybutów grupujących mogą być dowolne.

Zaleca się, aby baza miała zadeklarowany przynajmniej jeden atrybut grupujący. Zmienna, dla której pominięto wartość pola atrybutu grupującego, będzie widoczna w systemie w grupie o nazwie *(Pusta nazwa)* widocznej pośród grup użytkownika.



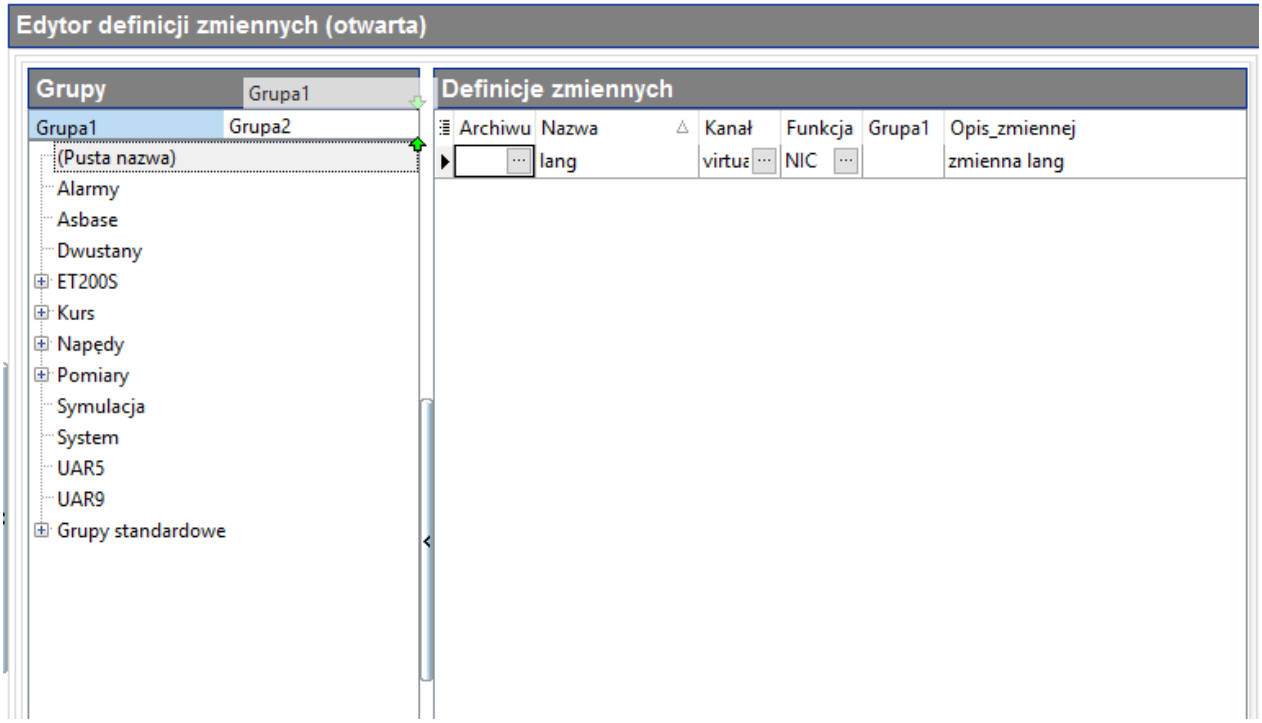
Rys. Drzewo grup zmiennych.

Przykładowo, zmienna może mieć atrybuty grupujące: **Wydział** (wartości np.: Bloki, Pomocniczy), **Jednostka** (Blok1, Blok2, Demineralizacja, Odpopielanie, itd.), **Urządzenie** (Camac, S7_1, GeFanuc11, itp.) i **Dziedzina Wartości** (Analog, Dwustan, Średnia_5m, itd.).

Po otwarciu okna edytora definicji zmiennych / okna przeglądania zmiennych drzewo obrazujące strukturę bazy jest generowane automatycznie na podstawie atrybutów grupujących.

Drzewo ma tyle poziomów, ile jest atrybutów grupujących. Na pierwszym poziomie wyświetlane są wszystkie unikalne wartości atrybutu, który został zadeklarowany jako pierwszy w *Edytorze schematu*, zakładce *Atrybuty grupujące*. Następnie dla każdej wartości atrybutu pierwszego tworzonych jest tyle węzłów, ile jest różnych wartości atrybutu zadeklarowanego jako drugi. Tak samo dzieje się dla pozostałych poziomów.

Aby zmienić kolejność wartości atrybutów - wystarczy metodą drag-and-drop uchwycić nagłówek atrybutu grupującego i przenieść go na inną pozycję w bloku *Grupy* okna edytora definicji zmiennych lub okna podglądu zmiennych.



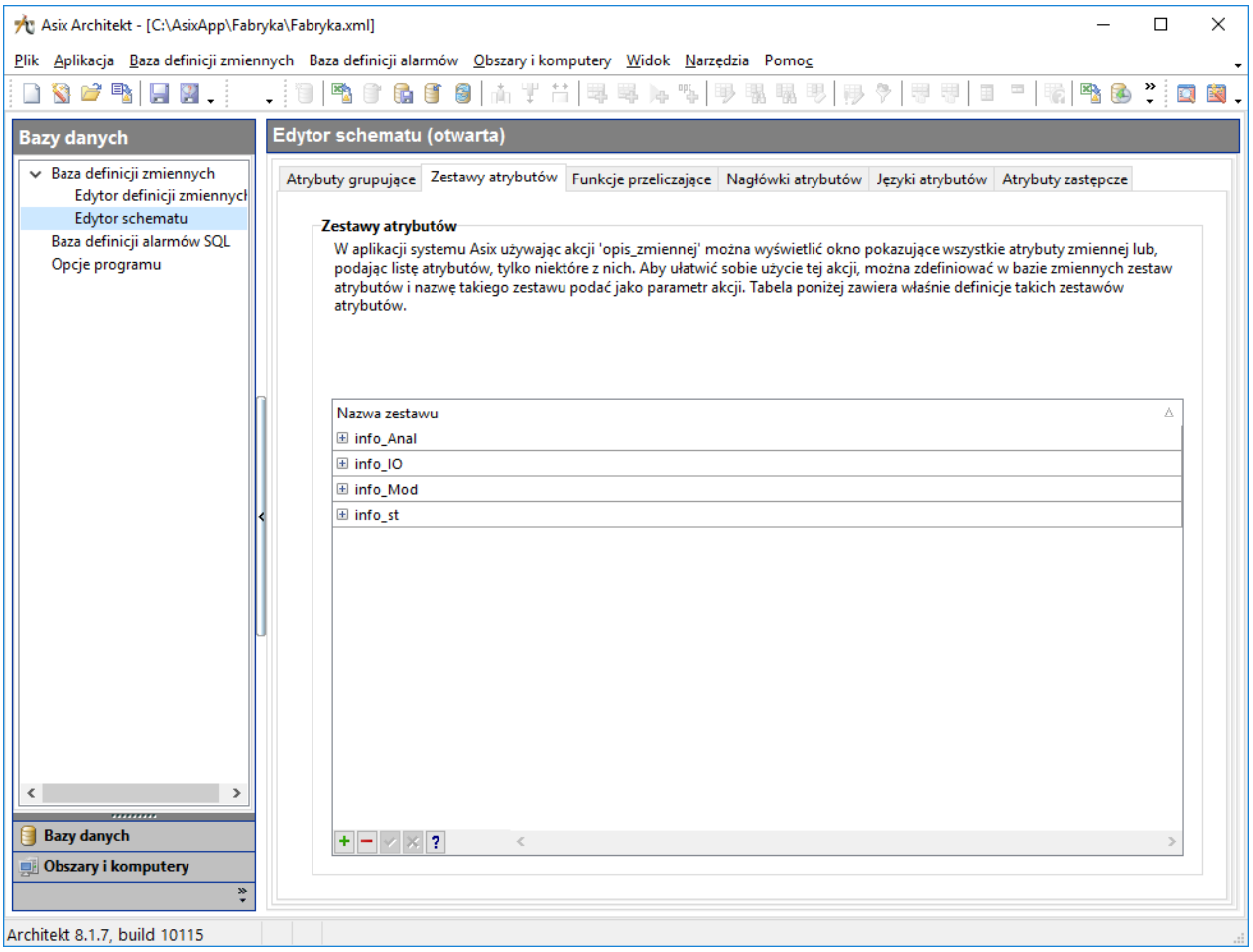
Rys. Zmiana kolejności grupowania zmiennych wg atrybutów.

2.2.2.2.2. Zestawy atrybutów

Zestawy atrybutów definiowane są na potrzeby aplikacji systemu Asix. Używane są w akcji OPIS_ZMIENNEJ oraz w oknie wyboru zmiennej (oknie edytora definicji zmiennych / oknie przeglądania zmiennych) dla zawężenia ilości wyświetlanych atrybutów zgodnie z potrzebami projektanta.

Miejscem, gdzie użytkownik może utworzyć swoje zestawy atrybutów jest zakładka *Zestawy atrybutów* w edytorze schematu (patrz: *Rysunek: Okno definiowania zestawu atrybutów.*).

Obecnie zestaw składa się tylko z listy nazw atrybutów. Szerokość okna atrybutów musi być zawsze podawana przy wywołaniu akcji OPIS_ZMIENNEJ.

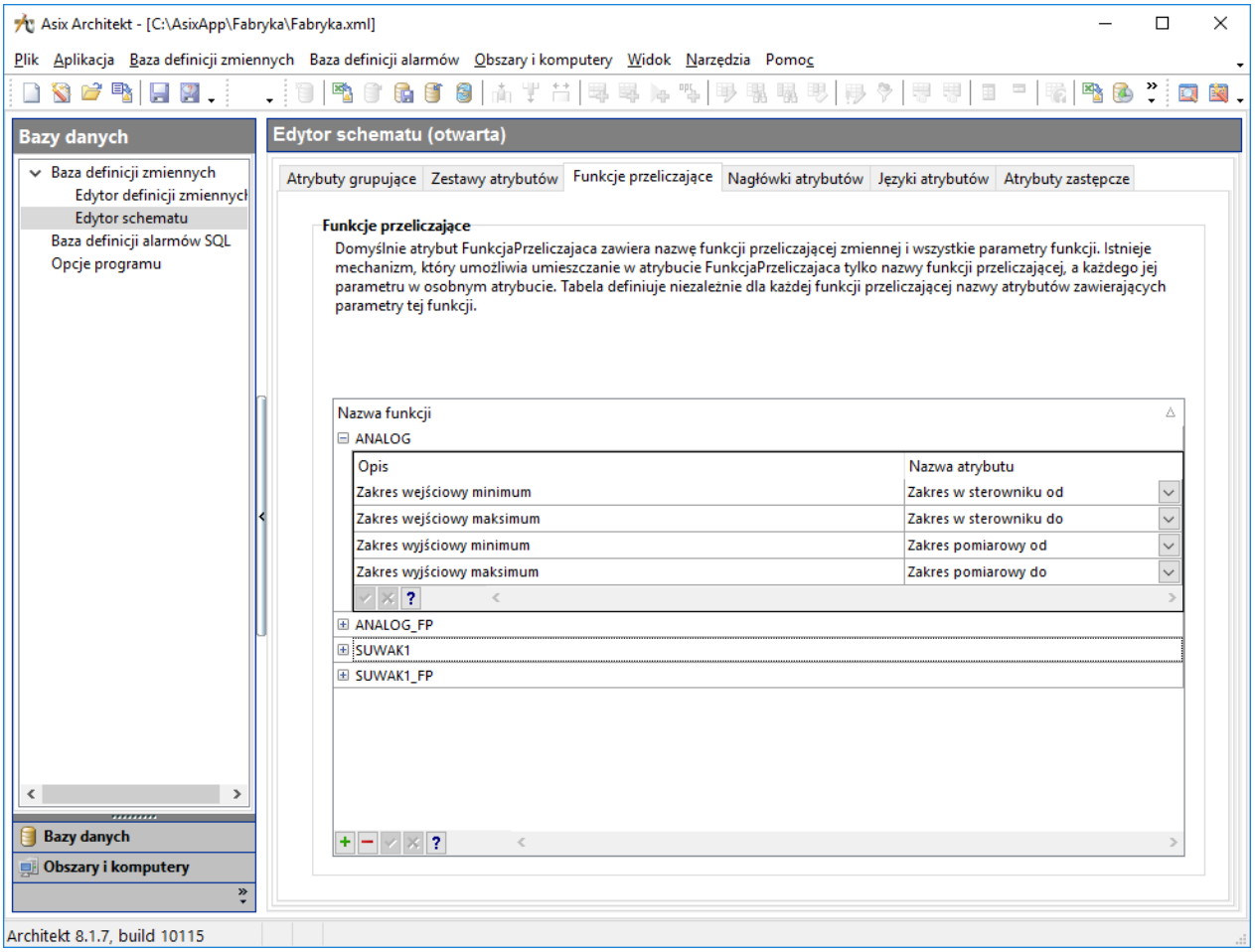


Rys. Okno definiowania zestawu atrybutów.

2.2.2.2.3. Parametry funkcji przeliczających

Parametry funkcji przeliczających definiowane są na potrzeby aplikacji systemu Asix.

W bazie definicji zmiennych atrybut FunkcjaPrzeliczajaca domyślnie zawiera nazwę funkcji przeliczającej zmiennej i wszystkie parametry funkcji. Istnieje jednakże mechanizm, który umożliwia umieszczanie w atrybucie FunkcjaPrzeliczajaca tylko nazwy funkcji przeliczającej, a każdego jej parametru w osobnym atrybucie. Tabela FunkcjePrzeliczające definiuje niezależnie dla każdej funkcji nazwy atrybutów zawierających parametry tej funkcji.

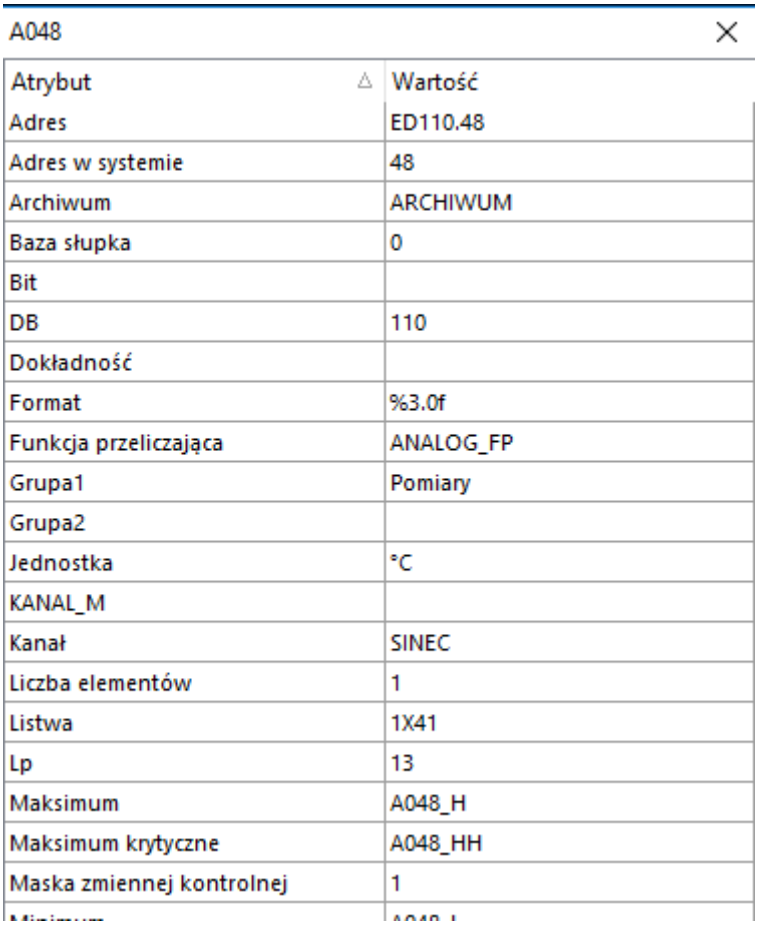


Rys. Okno definiowania parametrów funkcji przeliczających.

2.2.2.2.4. Tłumaczenia nazw atrybutów zmiennych

Baza posiada możliwość definiowania lokalnych, przetłumaczonych na dowolny zadeklarowany przez użytkownika język, nazw atrybutów systemowych i atrybutów użytkownika.

Lokalne nazwy atrybutów pojawiają się jako tytuły kolumn w oknie opisu zmiennej (dostępnym w Edytorze definicji zmiennych lub w oknie podglądu zmiennych).

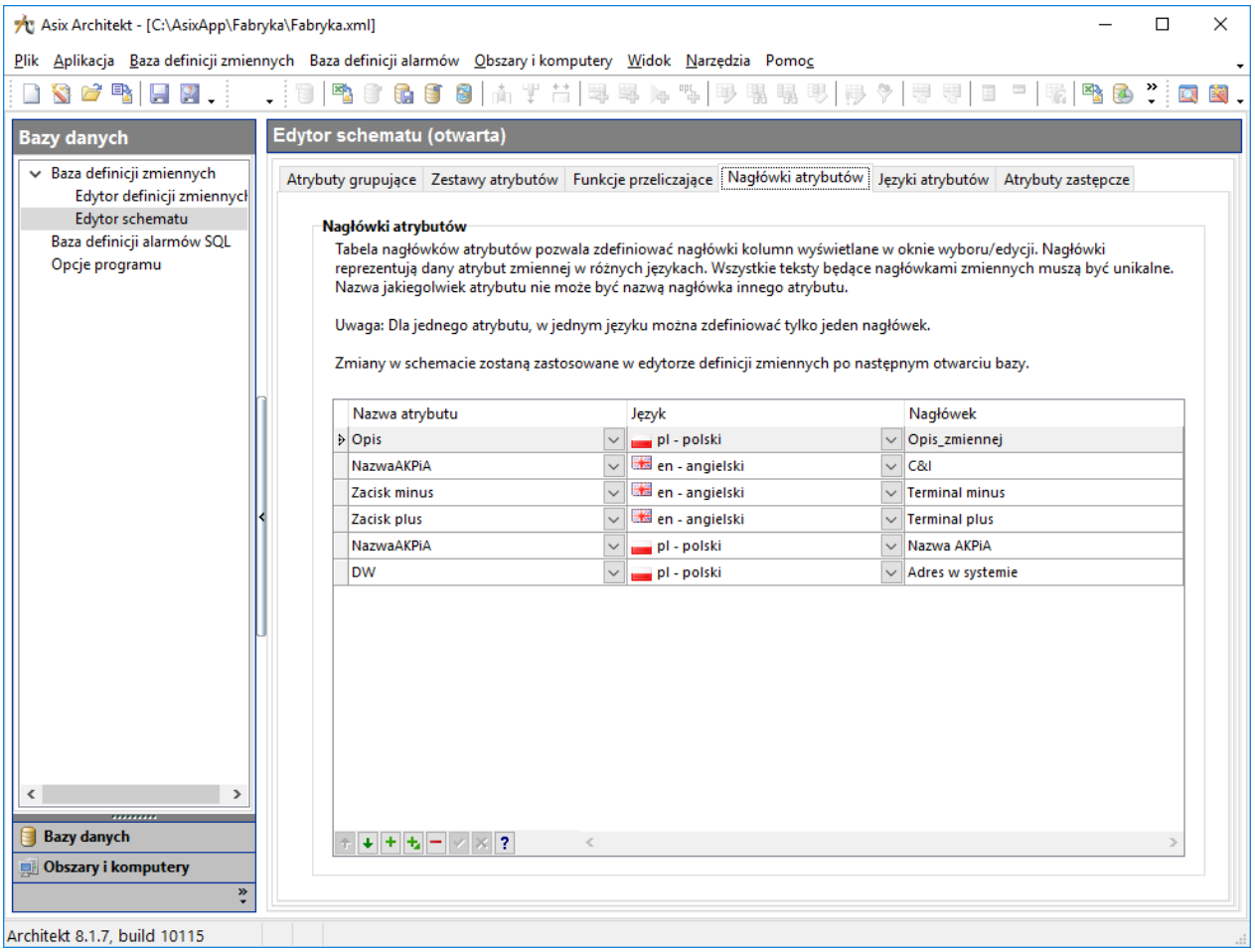

A048

Atrybut	Wartość
Adres	ED110.48
Adres w systemie	48
Archiwum	ARCHIWUM
Baza słupka	0
Bit	
DB	110
Dokładność	
Format	%3.0f
Funkcja przeliczająca	ANALOG_FP
Grupa1	Pomiary
Grupa2	
Jednostka	°C
KANAL_M	
Kanał	SINEC
Liczba elementów	1
Listwa	1X41
Lp	13
Maksimum	A048_H
Maksimum krytyczne	A048_HH
Maska zmiennej kontrolnej	1

Rys. Okno opisu zmiennej - polskie nazwy atrybutów.

Każdy język (poza PL i EN) wymaga zadeklarowania przez użytkownika nazw atrybutów zmiennych w *Edytorze schematu* bazy definicji zmiennych:

Bazy danych > Baza definicji zmiennych > Edytor schematu > Nagłówki atrybutów

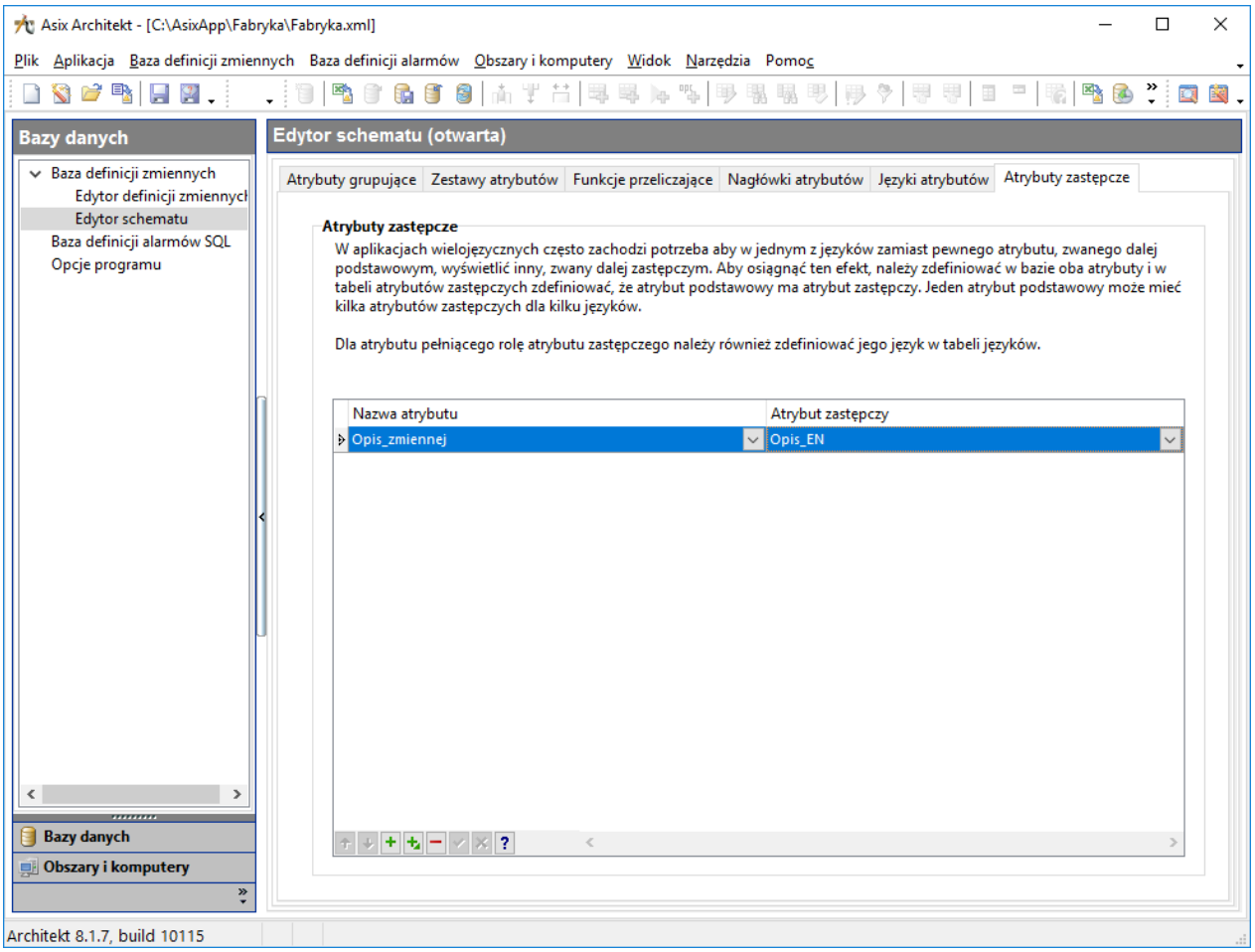


Rys. Okno deklarowania lokalnych nazw atrybutów zmiennych.

2.2.2.2.5. Atrybuty zastępcze

Atrybuty zastępcze stosowane są w aplikacjach wielojęzycznych. Dla każdego atrybutu użytkownika oraz dla atrybutu *Opis* można zdefiniować atrybut zastępczy - tzw. alias.

Dany atrybut (atrybut podstawowy) może posiadać jeden alias dla każdego języka obsługiwanego przez aplikację. Zdefiniowanie aliasu atrybutu sprawia, że dla wybranego języka w miejsce atrybutu podstawowego używany jest alias.



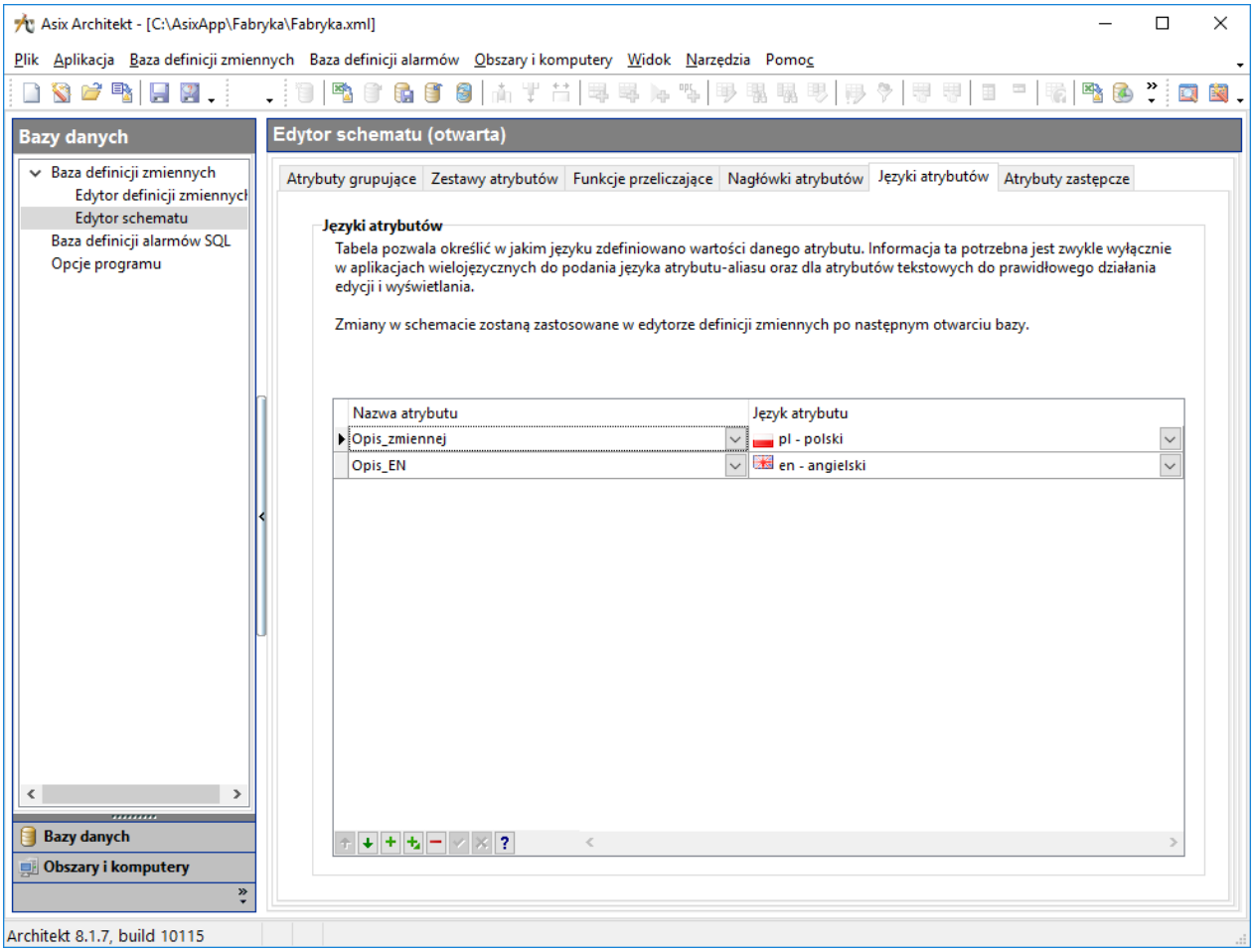
Rys. Okno definiowania atrybutów zastępczych.

W przypadku deklarowania atrybutu zastępczego, zarówno dla atrybutu podstawowego jak i zastępczego należy zdefiniować język w tabeli języków – zakładka *Języki atrybutów* w *Edytorze schematu*. Patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Tworzenie bazy w trybie Edytora > Tworzenie schematu bazy definicji zmiennych – Edytor > *2.2.2.2.6. Deklarowanie języków atrybutów.*

2.2.2.2.6. Deklarowanie języków atrybutów

Tabela języków atrybutów znajduje się w:

Bazy danych > *Baza definicji zmiennych* > *Edytor schematu* > zakładka *Języki atrybutów*



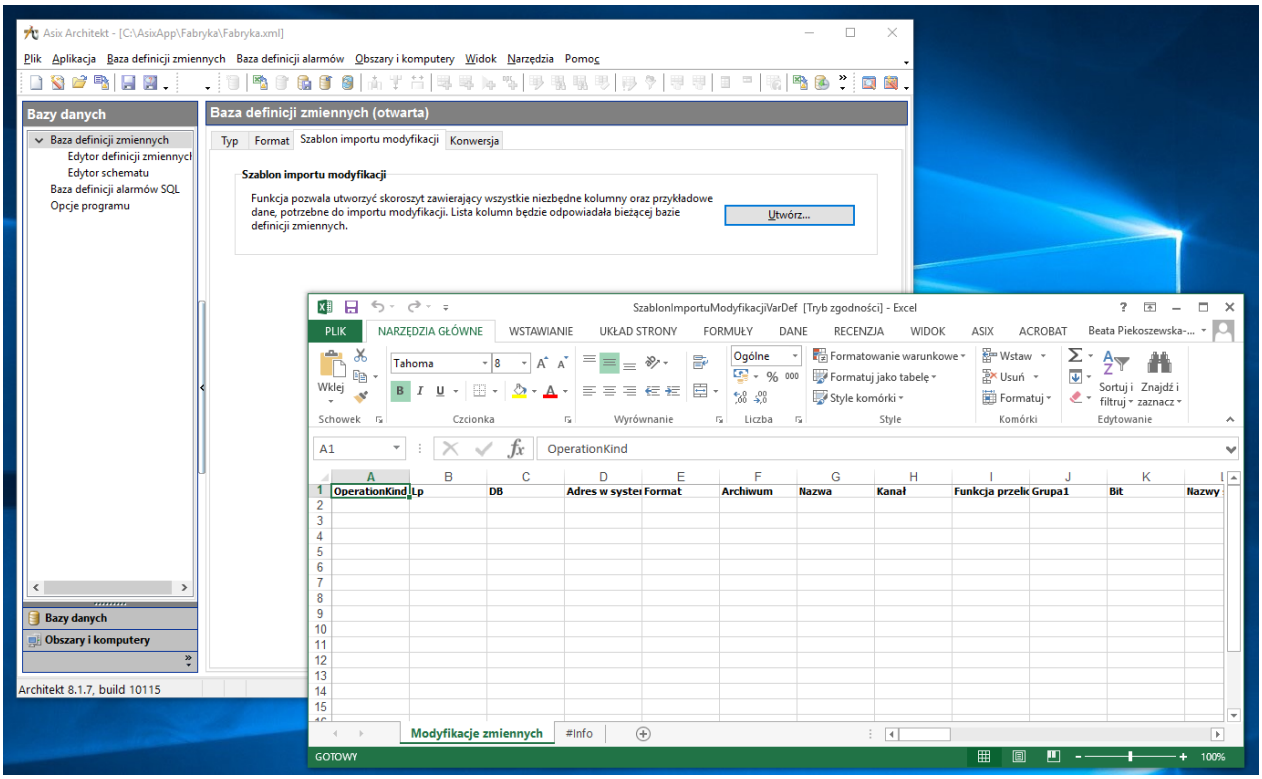
Rys. Okno definiowania języków atrybutów.

Tabela języków pozwala określić, w jakim języku zdefiniowano wartości danego atrybutu. Informacja ta potrzebna jest zwykle wyłączne w aplikacjach wielojęzycznych do zadeklarowania języka dla atrybutów tekstowych. Język deklarowany jest zarówno dla atrybutu podstawowego jak i zastępczego.

2.2.2.3. Import modyfikacji z arkusza Excel

Zmiany w definicji zmiennych polegające na dodaniu, usunięciu lub modyfikacji rekordu/rekordów mogą być w wygodny sposób importowane do bazy definicji zmiennych z arkusza Excel. Dla wygody użytkownika w module *Baza definicji zmiennych* > zakładce *Szablon importu modyfikacji* znajduje się funkcja umożliwiająca utworzenie skoroszytu (o domyślnej nazwie SzablonImportuModyfikacjiVarDef.xls) zawierającego wszystkie niezbędne kolumny oraz przykładowe dane potrzebne do importu modyfikacji.

Zakładka *Szablon importu modyfikacji* dostępna jest dla *Typu bazy definicji zmiennych*: *Standardowa baza definicji zmiennych / Tryb Edytor*. Lista kolumn w utworzonym szablonie importu odpowiada bieżącej (zadeklarowanej w zakładce *Format*) bazie definicji zmiennych.



Rys. Szablon importu modyfikacji VarDef.

Jak zaimportować do istniejącej bazy definicji zmiennych MS SQL lub Jet zmiany w definicji zmiennych wykorzystując arkusz Excel jako źródło?:

1. W pierwszym kroku wygeneruj szablon importu modyfikacji:

(wcześniej upewnij się, czy aktualnie włączonym trybem tworzenia bazy jest *Tryb Edytor*, a w zakładce *Baza definicji zmiennych* > *Format* zadeklarowana została właściwa baza)

Funkcja generująca szablon importu modyfikacji znajduje się w:

Baza definicji zmiennych > zakładka *Szablon importu modyfikacji* > przycisk *Utwórz*

2. W wygenerowanym skoroszycie, w zakładce *Modyfikacje zmiennych* wprowadź stosowne rekordy, pamiętając że:

- w przypadku dodania rekordu nowej zmiennej, należy w kolumnie *Rodzaj operacji* (ang. *OperationKind*) wstawić znak plus (+);

- w przypadku usunięcia rekordu zmiennej istniejącej w bazie, należy w kolumnie *Rodzaj operacji* (ang. *OperationKind*) wstawić znak minus (-);

- w przypadku modyfikacji rekordu zmiennej istniejącej w bazie, należy w kolumnie *Rodzaj operacji* (ang. *OperationKind*) wstawić znak gwiazdka (*);

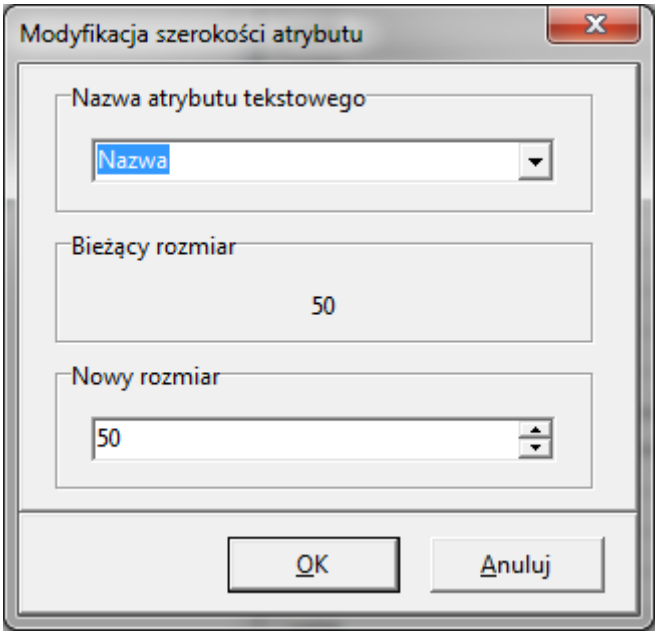
3. Otwórz istniejącą bazę definicji zmiennych (przycisk *Otwórz*).

4. Wybierz komendę *Importuj modyfikacje* i wskaż stosowny plik *. XLS ze zmodyfikowanymi rekordami zmiennych.

5. W oknie *'Wybierz arkusz'* zaznacz opcję *Modyfikacje zmiennych* i zatwierdź wybór przyciskiem *OK*.

2.2.2.4. Deklarowanie długości nazwy zmiennej

Sprawdzanie maksymalnego rozmiaru zmiennej w edytorze bazy definicji zmiennych jest wykonywane w odniesieniu do aktualnego rozmiaru kolumny ***Nazwa***. Chcąc zmienić ograniczenie dotyczące długości zmiennej, należy użyć opcji ***Zmień rozmiar atrybutu*** dla pola *Nazwa*. Opcja dostępna w menu edytora bazy definicji zmiennych.



Rys. Okno 'Modyfikacja szerokości atrybutu'.

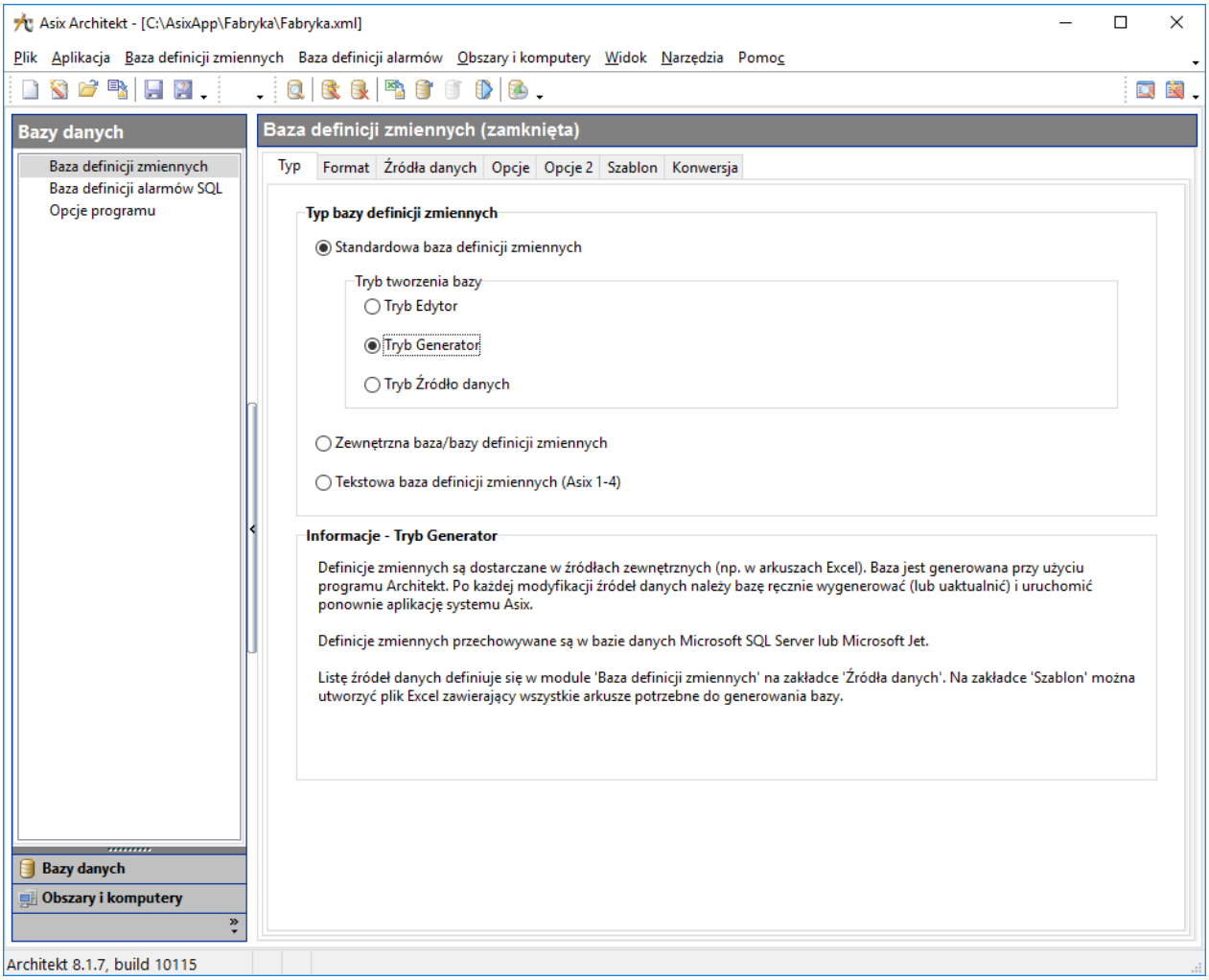
2.2.3. Tworzenie bazy w trybie Generatora

W przypadku tworzenia bazy definicji zmiennych w trybie Generatora, definicje zmiennych dostarczane są w zewnętrznych źródłach (np. w arkuszach Excel). Baza generowana jest przy użyciu programu Architekt. Po każdej modyfikacji źródeł danych należy bazę ręcznie wygenerować (lub uaktualnić) i uruchomić ponownie aplikację systemu Asix.

Szczególnym przypadkiem jest tworzenie zbiorczej bazy, generowanej z kilku istniejących baz definicji zmiennych - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Tworzenie bazy w trybie Generatora > 2.2.3.7. Zbiorcza baza definicji zmiennych.

Tworzenie bazy definicji zmiennych w trybie Generatora realizowane jest w następujących krokach:

1. Zdefiniuj typ bazy definicji zmiennych jako *Tryb Generator*



Rys. Baza definicji zmiennych > zakładka Typ - Tryb Generator.

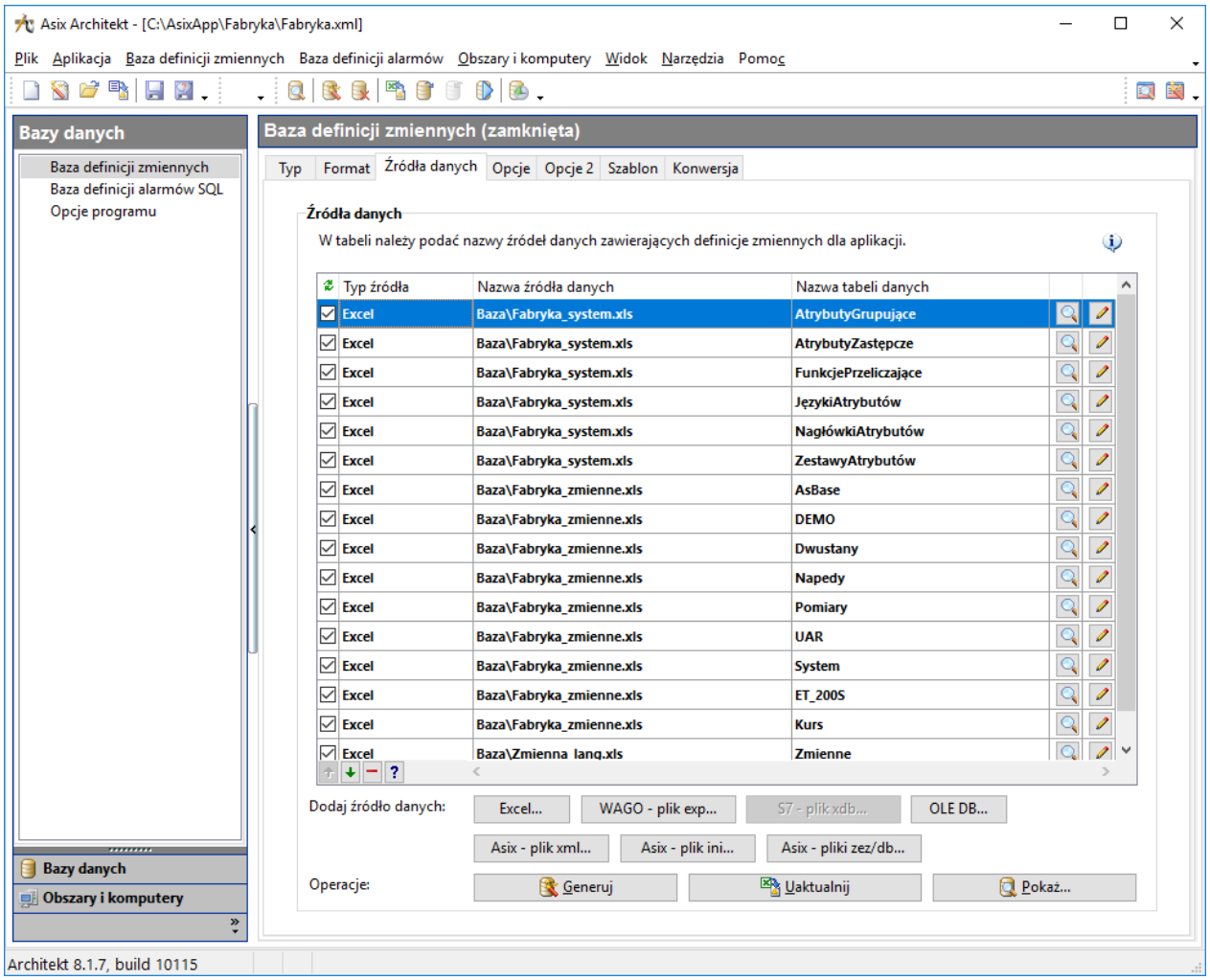
2. Określ format oraz nazwę bazy definicji zmiennych, która ma zostać wygenerowana:

Baza definicji zmiennych > zakładka *Format* > format bazy danych ustawiony na: *Baza danych serwera Microsoft SQL Serwer* lub *Plik w formacie Mdb (Jet/Microsoft Access)*

W przypadku baz danych MS SQL wymagane jest określenie parametrów:
- *Nazwa serwera*;
- *Nazwa bazy*;
- *Katalog bazy*;
- *Tryb uwierzytelniania*:
  - *Windows*,
  - *Serwer MSQL* - wymaga określenia *Nazwy użytkownika* oraz *Hasła*.

3. Określ źródła danych:

Baza definicji zmiennych > zakładka *Źródła danych* > pliki źródła danych (pliki Excel, obiekt OLE DB, pliki ZEZ/DB starszych wersji Asixa, pliki tekstowe starszych wersji Asixa lub pliki konfiguracyjne aplikacji formatu *.XML)



Rys. Baza definicji zmiennych - zakładka 'Źródło danych'.

4. Wybierz polecenie *Generuj* .

5. Ewentualnie otwórz bazę, aby zweryfikować przy użyciu *Edytora definicji zmiennych* poprawność generacji.

2.2.3.1. Opcje generowania bazy definicji zmiennych z wielu źródeł danych

Blok *Bazy danych* > *Baza definicji zmiennych* wyposażony został w zakładki *Opcje/Opcje2*, umożliwiające ustawienie parametrów dotyczących generowania bazy definicji zmiennych pochodzących z wielu źródeł - włączając w to przypadki, kiedy różne źródła zawierają definicje powtarzających się zmiennych.

Zakładka *Opcje* obejmuje następujące parametry:

- ***Generowanie informacji o położeniu definicji zmiennych***
  - ***Dodaj atrybut 'Źródło danych'*** - powoduje dodanie do bazy definicji zmiennych atrybutu przechowującego nazwę skoroszytu zawierającego definicję zmiennej;
  - ***Dodaj atrybut 'Tabela danych'*** - powoduje dodanie do bazy definicji zmiennych atrybutu przechowującego nazwę arkusza zawierającego definicję zmiennej;
- ***Sprawdzanie różnic podczas generowania bazy definicji zmiennych***
  - ***Włącz sprawdzanie różnic podczas generowania bazy definicji zmiennych*** - opcja umożliwia porównanie generowanej bazy z istniejącą bazą w trakcie generowania bazy definicji zmiennych; w przypadku wykrycia zmian pojawia się pytanie o kontynuowanie; porównywane są definicje zmiennych i tabele schematu.
  - ***Włącz zapis tylko różnic do bazy definicji zmiennych*** - opcja umożliwia zapisywanie tylko zmian wykrytych w trakcie generacji bazy definicji zmiennych zamiast czyszczenia bazy i zapisywania wszystkich danych; zapis samych zmian jest możliwy gdy nie występują zmiany schematu bazy.
- ***Tworzenie kopii zapasowej podczas generowania bazy definicji zmiennych***
  - ***Włącz tworzenie kopii zapasowej podczas generowania bazy definicji zmiennych*** – przy załączonej opcji generator bazy definicji zmiennych tworzy kopię dotychczasowej bazy w podkatalogu 'backup' zadeklarowanego katalogu bazy (lub w przypadku braku zadeklarowanego katalogu bazy danych serwera MS SQL podkatalog z kopią tworzony jest w domyślnym katalogu bazy serwera MS SQL).

Zakładka *Opcje 2* obejmuje następujące parametry:

- ***Dzielenie definicji zmiennej pomiędzy źródłami***
  - ***Pozwalaj na dzielenie definicji zmiennej pomiędzy źródłami*** - opcja przydatna w przypadku scalania definicji tych samych zmiennych pochodzących z wielu różnych źródeł;
- ***Wielokrotne definicje zmiennych - preferowanie zmiennych aktywnych***

- *W przypadku powtórzenia definicji zmiennej użyj pierwszej definicji zmiennej aktywnej* – w przypadku użycia opcji generator przyjmuje jako obowiązującą definicję pochodzącą z pierwszego dokumentu zawierającego tę definicję wg kolejności zadeklarowanych źródeł danych na zakładce *Źródła danych*.

- ***Wielokrotne definicje zmiennych - preferowanie kanałów fizycznych***

  Opcja ma zastosowanie dla źródeł danych w postaci plików *.ini i *.xml, ponieważ tylko dla tych źródeł możliwe jest określenie typu kanału w definicji zmiennej:
  Definicje zmiennych nie mogą się powtarzać;
  Definicje zmiennych mogą się powtarzać: *zezwalaj na jedną definicję w kanale fizycznym, ignoruj powtórzone definicje w kanałach typu NETWORK* - w praktyce oznacza to, że w trakcie generowania bazy zaakceptowana zostanie pierwsza napotkana definicja, a dla wszystkich definicji z pozostałych kanałów fizycznych wypisane zostanie ostrzeżenie w panelu logu *'Generacja bazy definicji zmiennych'*;
  Definicje zmiennych mogę się powtarzać: *akceptuj pierwszą definicję w kanale fizycznym, ignoruj pozostałe powtórzone definicje* - oznacza, że w trakcie generowania bazy zaakceptowana zostanie pierwsza definicja z kanału fizycznego, dla drugiej definicji z kanały fizycznego pojawi się ostrzeżenie w panelu logu *'Generacja bazy definicji zmiennych'*, a pozostałe powtarzające się definicje zostaną zignorowane.

2.2.3.2. Generowanie bazy z plików ZEZ/DB

W przypadku baz definicji zmiennych w formacie Paradox wykorzystywanych w starszych wersjach Asixa (wersja 4 i wcześniejsze) należy przekonwertować je do formatu standardowych baz systemu Asix. Patrz rozdział: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > 2.2.9. Konwertowanie bazy definicji zmiennych z formatu Paradox na format Jet lub MS SQL.

2.2.3.3. Generowanie bazy definicji zmiennych z arkusza Excel

W przypadku generowania bazy definicji zmiennych na podstawie arkuszy Excel, należy te arkusze przygotować w sposób szczegółowo opisany w rozdziale Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > *2.2.6. Przygotowanie zewnętrznego źródła danych - arkusz Excel*.

Architekt pozwala przygotować szablonowy plik Excel definicji zmiennych (Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > *2.2.6.10. Szablon bazy definicji zmiennych*).

2.2.3.4. Generowanie bazy definicji zmiennych na podstawie bazy OLE DB

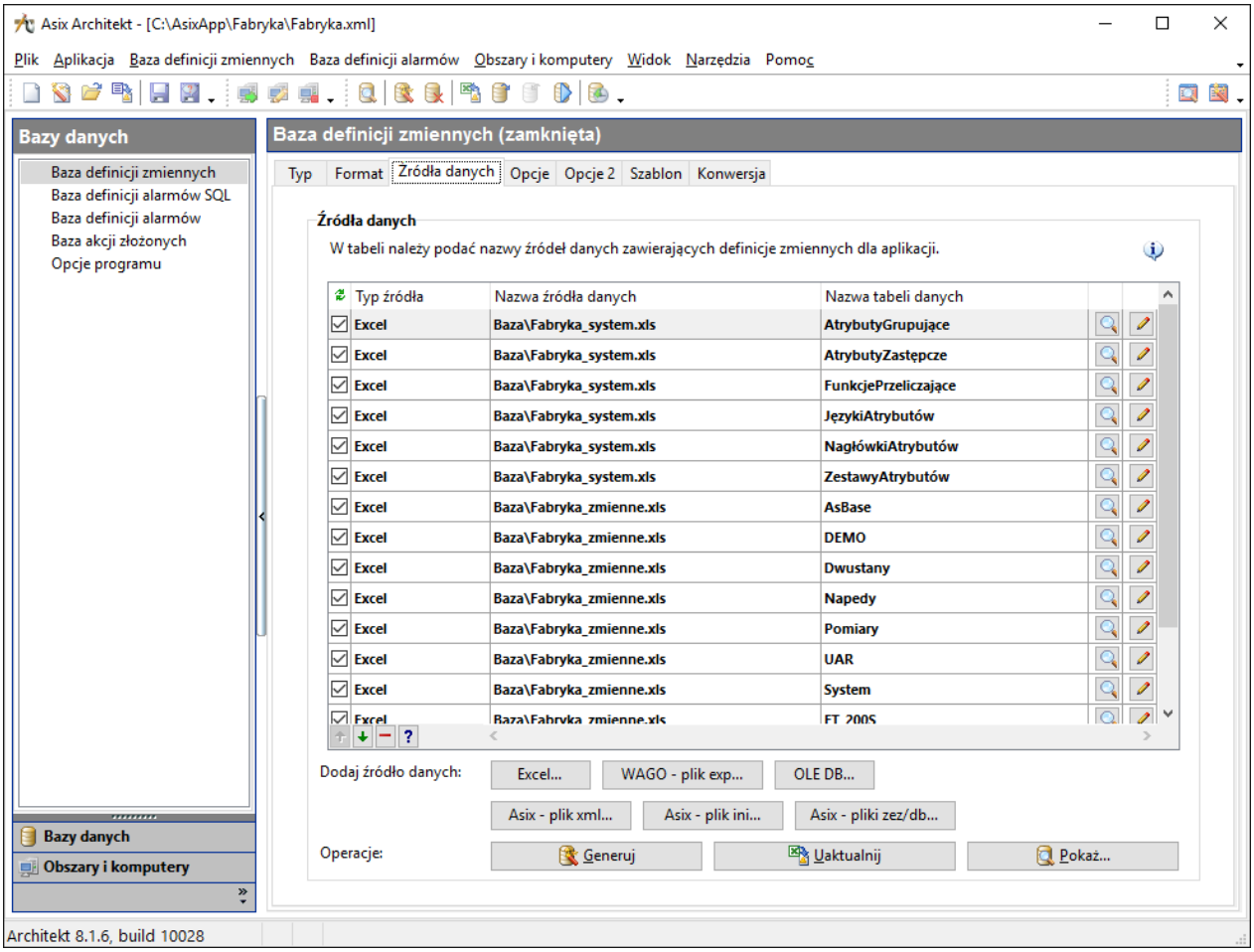
Istnieje możliwość wygenerowania bazy definicji zmiennych z bazy OLE DB po odpowiednim skonfigurowaniu właściwości łącza danych (dostawca, właściwości połączenia oraz opcje zaawansowane).

Struktura źródłowej bazy OLE DB powinna być przygotowana w sposób analogiczny do arkuszy Excel, wykorzystywanych jako źródło generowania bazy definicji zmiennych. Wymagane jest zatem zdefiniowanie następujących rodzajów danych:
- atrybuty zmiennych,
- grupowanie,
- zestawy atrybutów,
- funkcje przeliczające,
- nagłówki,
- języki atrybutów,
- nazwy zstępcze.

Powyższe dane należy przygotować według wskazówek omówionych w rozdziale Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > *2.2.6. Przygotowanie zewnętrznego źródła danych - arkusz Excel*.

2.2.3.5. Generowanie bazy definicji zmiennych na podstawie plików XML

W przypadku generowania zbiorczej bazy definicji zmiennych z baz kilku plików xml - zaleca się wskazanie źródeł danych przy użyciu przycisku ***Asix - plik xml...***.

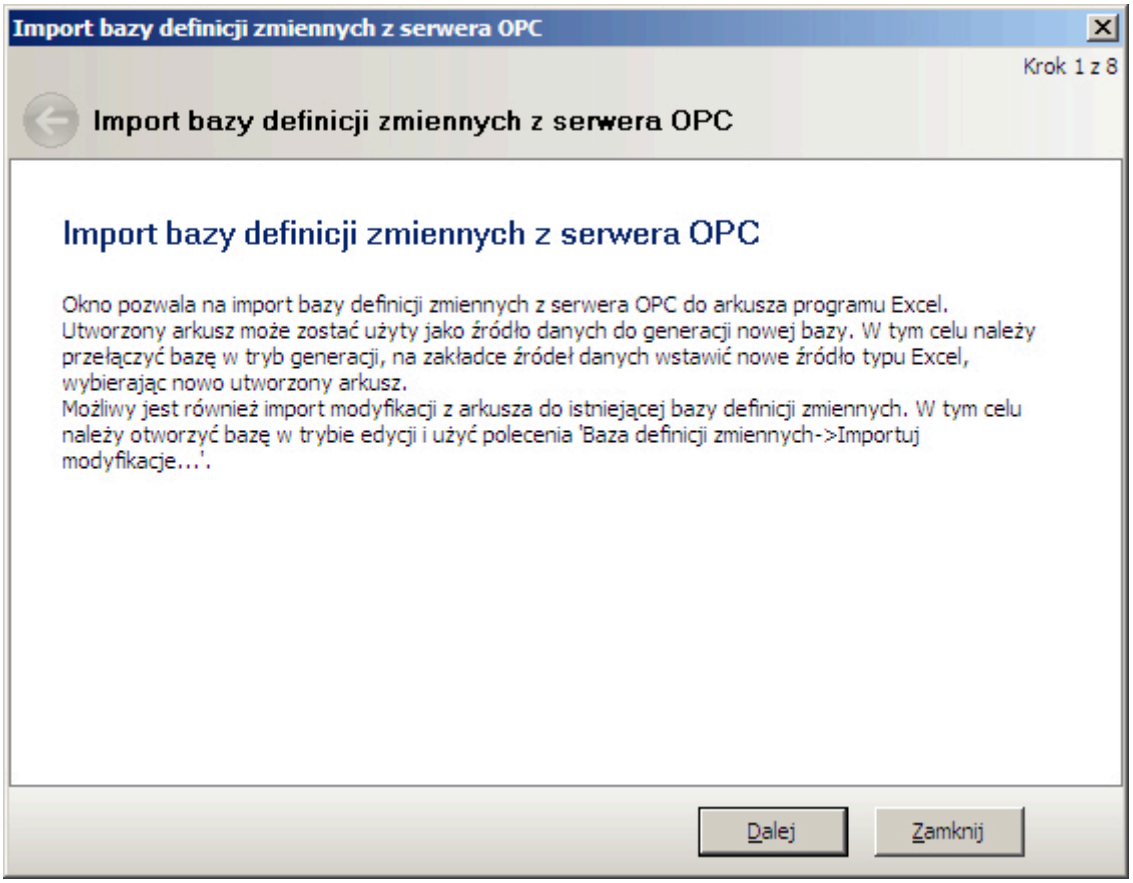


Rysunek: Okno deklarowania źródeł danych do wygenerowania bazy definicji zmiennych.

2.2.3.6. Import definicji zmiennych z serwera OPC / OPC UA

Definicje zmiennych z serwera OPC / OPC UA importowane są do arkusza programu Excel poleceniem ***Importuj z serwera OPC... /OPC UA...*** z menu ***Baza definicji zmiennych***. Polecenie to uruchamia kreator, który pozwala zdefiniować połączenie z serwerem OPC i sposób importu definicji zmiennych. Kreator automatycznie doda utworzony arkusz do listy źródeł danych projektowanej aplikacji (Architekt > ***Bazy danych*** > ***Baza definicji zmiennych*** > ***Źródła danych***), gdy aktywnym typem tworzenia aplikacji jest ***Tryb Generator***.
Jeśli baza jest generowana po raz pierwszy - należy w module ***Bazy danych*** > ***Baza definicji zmiennych*** na zakładce ***Typ*** zadeklarować jako typ tworzenia bazy - ***Tryb Generator***.

Jeżeli definicje zmiennych z serwera OPC zostały juz zaimportowane do arkusza Ecxel a następnie baza definicji zmiennych wygenerowana - istnieje możliwość wprowadzania zmian do arkusza zawierającego definicje zmiennych pochodzących z OPC i importowania tych zmian do bazy przy użyciu komendy ***Importuj modyfikacje...*** z menu ***Baza definicji zmiennych***. Opcja ta jest dostępna po ustawieniu w module ***Bazy danych*** > ***Baza definicji zmiennych*** na zakładce ***Typ*** typu tworzenia bazy ***Tryb Edytor*** i otwarciu bazy.



Rysunek: Kreator importu definicji zmiennych z serwera OPC.

2.2.3.7. Zbiorcza baza definicji zmiennych

Zbiorcza baza definicji zmiennych jest to baza definicji zmiennych generowanych z kilku istniejących baz definicji zmiennych. Przy generowaniu takiej bazy można napotkać dwa rodzaje problemów.

Informacje tego samego typu mogą znajdować się w wejściowych bazach w atrybutach o innych nazwach. W tym przypadku należy użyć transformacji nazw atrybutów, aby ujednolicić nazwy atrybutów.

Drugi problem to różne nazwy i liczby atrybutów grupujących. W takim przypadku również należy użyć transformacji nazw atrybutów, aby ujednolić nazwy atrybutów lub dodać brakujące atrybuty grupujące. Należy również użyć transformacji wartości atrybutów, aby podać wartości dodawanych atrybutów grupujących i, ewentualnie, dopasować wartości atrybutów grupujących do nazewnictwa stosowanego w zbiorczej bazie definicji zmiennych.

Uwaga: istnieje też możliwość czytania definicji zmiennych z łączonych baz definicji zmiennych, parametryzowanych przez zewnętrzne pliki XML. Patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Wstęp > Format bazy danych > 2.2.1.2.1. Łączone bazy definicji zmiennych.

2.2.3.8. Uaktualnianie źródeł danych

Jeśli baza definicji zmiennych została wygenerowana, dalsze uaktualnianie definicji zmiennych można przeprowadzać przy użyciu komendy ***Uaktualnij*** (moduł *Bazy danych > Baza definicji zmiennych* (dla typu: *Standardowa baza definicji zmiennych*; tryb *Generator*) > zakładka *Źródła danych*.

Funkcja uaktualniania źródeł danych zostaje zablokowana, gdy źródła zawierają tabele transformacji lub tabele zmiennych używanych.

2.2.3.9. Transformacja nazw i wartości atrybutów

W przypadku generowania baz definicji zmiennych na podstawie definicji zmiennych znajdujących się w arkuszu kalkulacyjnym *.xls dostępny jest mechanizm transformacji nazw atrybutów i wartości atrybutów. Transformacji takiej można dokonać definiując w arkuszu Excela nazwę atrybutu wejściowego i wyjściowego (dla transformacji nazw atrybutów) lub wejściowe i wyjściowe wartości atrybutów (dla transformacji wartości atrybutów). Zarówno nazwa jak i struktura takiego arkusza (tabeli) musi być ściśle określona:

- dla transformacji nazw atrybutów:

    - nazwa arkusza: *PrzekształceniaNazwAtr*
    - kolumny arkusza: *NazwaŹródłaDanych*, *AtrybutWejściowy*, *AtrybutWyjściowy*

    (szczegółowy opis - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > 2.2.6.8. Parametry transformacji nazw atrybutów i wartości)

Podanie jako *AtrybutWejściowy* nazwy pustej powoduje **dodanie** do źródła danych nowej kolumny.

Podanie jako *AtrybutWyjściowy* nazwy pustej powoduje **usunięcie** z źródła danych wskazanej kolumny.

- dla transformacji wartości atrybutów:

  - nazwa arkusza: *PrzekształceniaWartościAtr*
  - kolumny arkusza: *NazwaŹródłaDanych*, *Atrybut*, *WejściowaWartośćAtrybutu* i *WyjściowaWartośćAtrybutu*

  (szczegółowy opis - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > 2.2.6.8. Parametry transformacji nazw atrybutów i wartości)

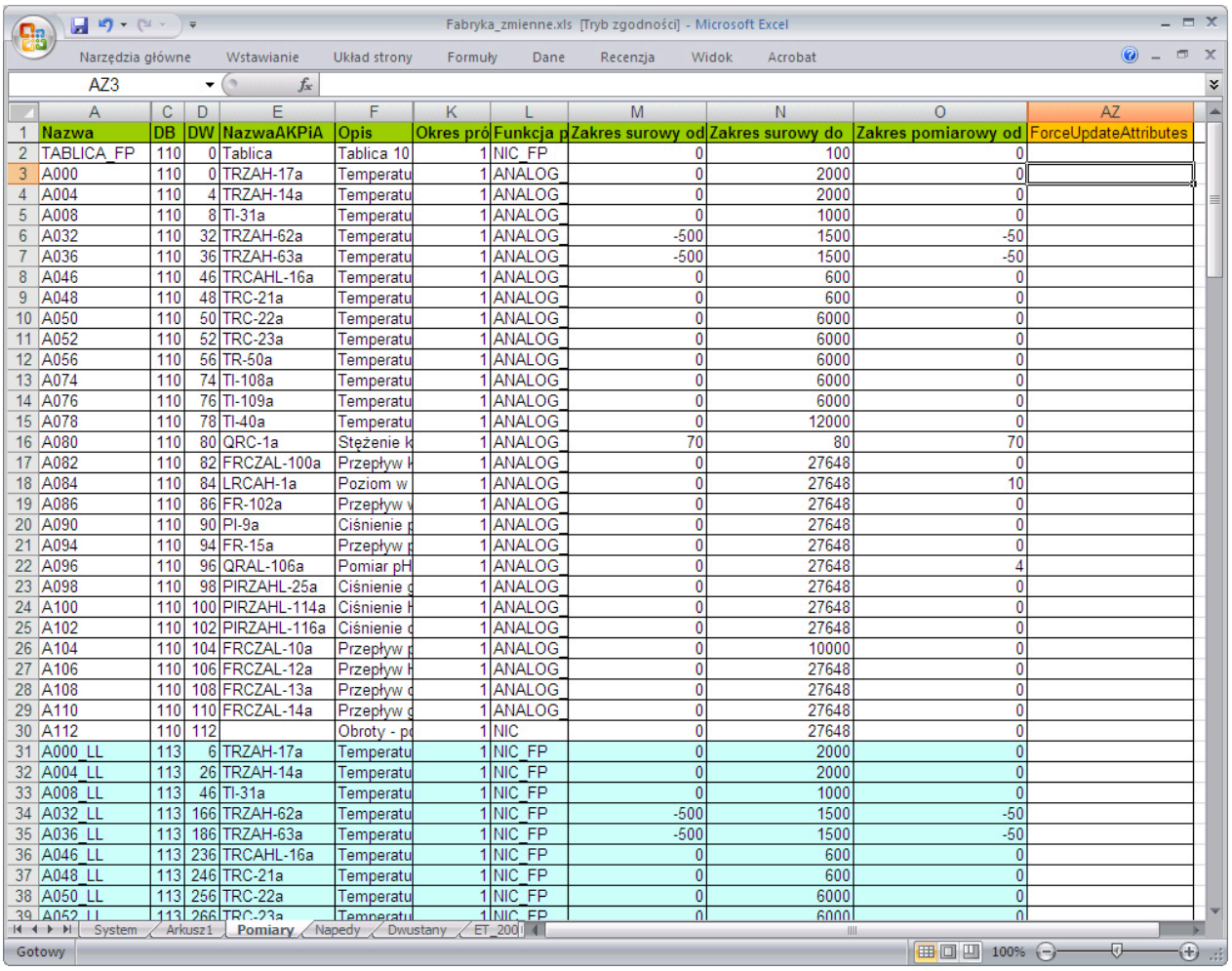
  Podanie jako *WejściowaWartośćAtrybutu* nazwy pustej powoduje **dodanie** do źródła danych nowej wartości atrybutu.

  Podanie jako *WyjściowaWartośćAtrybutu* nazwy pustej powoduje **usunięcie** z źródła danych wskazanej wartości atrybutu.

2.2.3.10. Nadpisywanie atrybutów przez wybraną tabelę źródła danych

Nadpisywanie wartości atrybutów przez wybraną tabelę źródła danych wymaga wstawienia dodatkowej kolumny o nazwie *ForceUpdateAttributes* do tabeli (arkusza) z docelowymi wartościami atrybutów, które chcemy nadpisać.

Nadpisywanie wartości atrybutów wykonywane jest dla konkretnych zmiennych - tabela z docelowymi wartościami atrybutów powinna zatem zawierać kolumnę z nazwami zmiennych, kolumny atrybutów z wartościami docelowymi oraz kolumnę *ForceUpdateAttributes (*kolumna ta może być pusta, wartości w niej nie są brane pod uwagę).



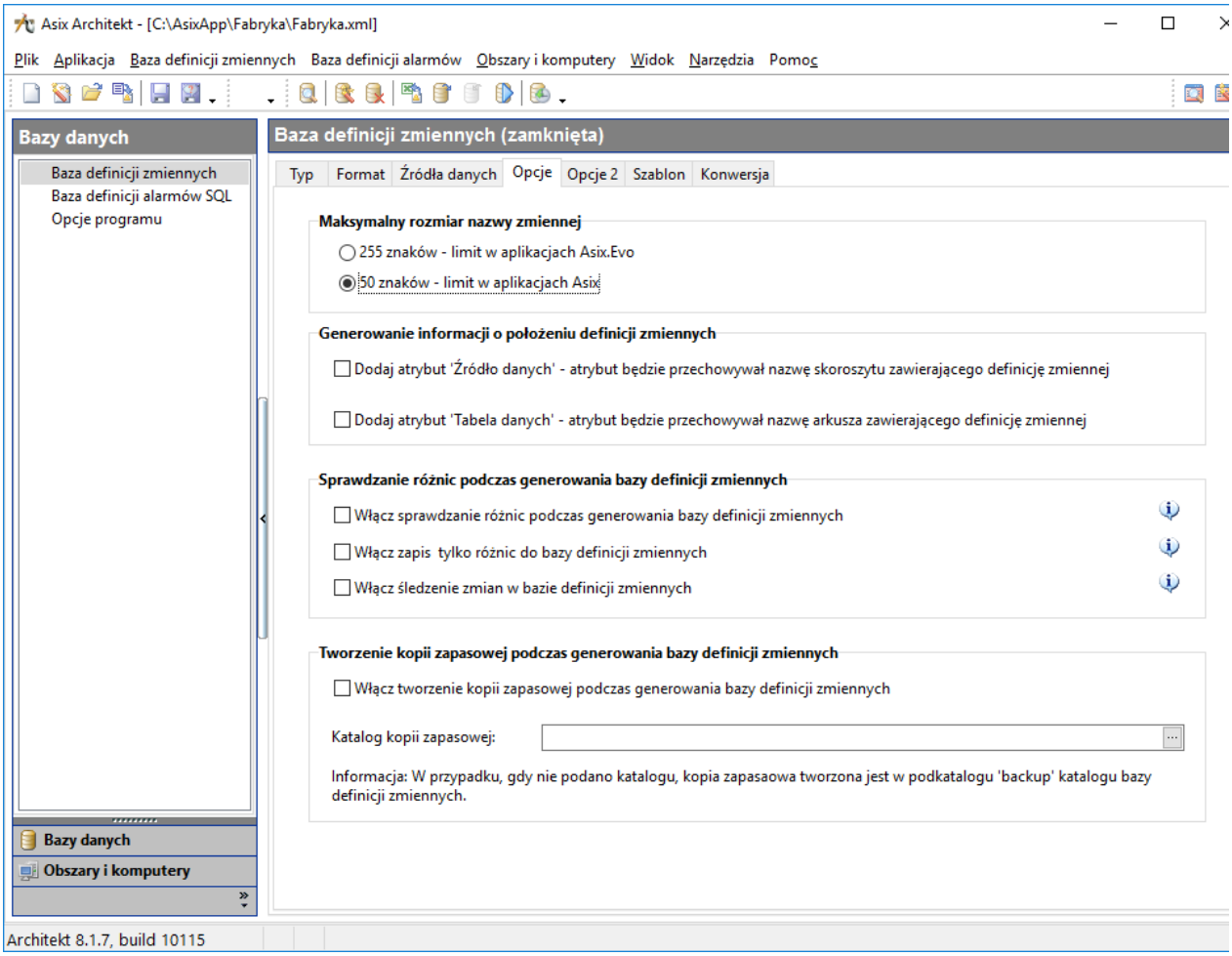
	A	C	D	E	F	K	L	M	N	O	AZ
1	Nazwa	DB	DW	NazwaAKPiA	Opis	Okres pró	Funkcja p	Zakres surowy od	Zakres surowy do	Zakres pomiarowy od	ForceUpdateAttributes
2	TABLICA_FP	110	0	Tablica	Tablica 10	1	NIC_FP	0	100	0	
3	A000	110	0	TRZAH-17a	Temperatu	1	ANALOG	0	2000	0	
4	A004	110	4	TRZAH-14a	Temperatu	1	ANALOG	0	2000	0	
5	A008	110	8	TI-31a	Temperatu	1	ANALOG	0	1000	0	
6	A032	110	32	TRZAH-62a	Temperatu	1	ANALOG	-500	1500	-50	
7	A036	110	36	TRZAH-63a	Temperatu	1	ANALOG	-500	1500	-50	
8	A046	110	46	TRCAHL-16a	Temperatu	1	ANALOG	0	600	0	
9	A048	110	48	TRC-21a	Temperatu	1	ANALOG	0	600	0	
10	A050	110	50	TRC-22a	Temperatu	1	ANALOG	0	6000	0	
11	A052	110	52	TRC-23a	Temperatu	1	ANALOG	0	6000	0	
12	A056	110	56	TR-50a	Temperatu	1	ANALOG	0	6000	0	
13	A074	110	74	TI-108a	Temperatu	1	ANALOG	0	6000	0	
14	A076	110	76	TI-109a	Temperatu	1	ANALOG	0	6000	0	
15	A078	110	78	TI-40a	Temperatu	1	ANALOG	0	12000	0	
16	A080	110	80	QRC-1a	Stężenie k	1	ANALOG	70	80	70	
17	A082	110	82	FRCZAL-100a	Przepływ k	1	ANALOG	0	27648	0	
18	A084	110	84	LRCAH-1a	Poziom w	1	ANALOG	0	27648	10	
19	A086	110	86	FR-102a	Przepływ v	1	ANALOG	0	27648	0	
20	A090	110	90	PI-9a	Ciśnienie p	1	ANALOG	0	27648	0	
21	A094	110	94	FR-15a	Przepływ p	1	ANALOG	0	27648	0	
22	A096	110	96	QRAL-106a	Pomiar pH	1	ANALOG	0	27648	4	
23	A098	110	98	PIRZAHL-25a	Ciśnienie g	1	ANALOG	0	27648	0	
24	A100	110	100	PIRZAHL-114a	Ciśnienie H	1	ANALOG	0	27648	0	
25	A102	110	102	PIRZAHL-116a	Ciśnienie g	1	ANALOG	0	27648	0	
26	A104	110	104	FRCZAL-10a	Przepływ p	1	ANALOG	0	10000	0	
27	A106	110	106	FRCZAL-12a	Przepływ H	1	ANALOG	0	27648	0	
28	A108	110	108	FRCZAL-13a	Przepływ d	1	ANALOG	0	27648	0	
29	A110	110	110	FRCZAL-14a	Przepływ g	1	ANALOG	0	27648	0	
30	A112	110	112		Obroty - pc	1	NIC	0	27648	0	
31	A000_LL	113	6	TRZAH-17a	Temperatu	1	NIC_FP	0	2000	0	
32	A004_LL	113	26	TRZAH-14a	Temperatu	1	NIC_FP	0	2000	0	
33	A008_LL	113	46	TI-31a	Temperatu	1	NIC_FP	0	1000	0	
34	A032_LL	113	166	TRZAH-62a	Temperatu	1	NIC_FP	-500	1500	-50	
35	A036_LL	113	186	TRZAH-63a	Temperatu	1	NIC_FP	-500	1500	-50	
36	A046_LL	113	236	TRCAHL-16a	Temperatu	1	NIC_FP	0	600	0	
37	A048_LL	113	246	TRC-21a	Temperatu	1	NIC_FP	0	600	0	
38	A050_LL	113	256	TRC-22a	Temperatu	1	NIC_FP	0	6000	0	
39	A052_LL	113	266	TRC-23a	Temperatu	1	NIC_FP	0	6000	0	

Rysunek: Nadpisywanie atrybutów - tabela docelowych wartości atrybutów.

2.2.3.11. Deklarowanie maksymalnej długości nazwy zmiennej

Dla baz definicji zmiennych tworzonych w trybie generatora istnieje możliwość użycia opcji definiującej maksymalną długość nazw zmiennych: 50 znaków dla aplikacji klasycznych i 255 dla aplikacji w wersji Evo. Opcja ***Maksymalny rozmiar nazwy zmiennej*** dostępna jest przy trybie tworzenia bazy definicji zmiennych przy użyciu generatora, na zakładce *Opcje:*

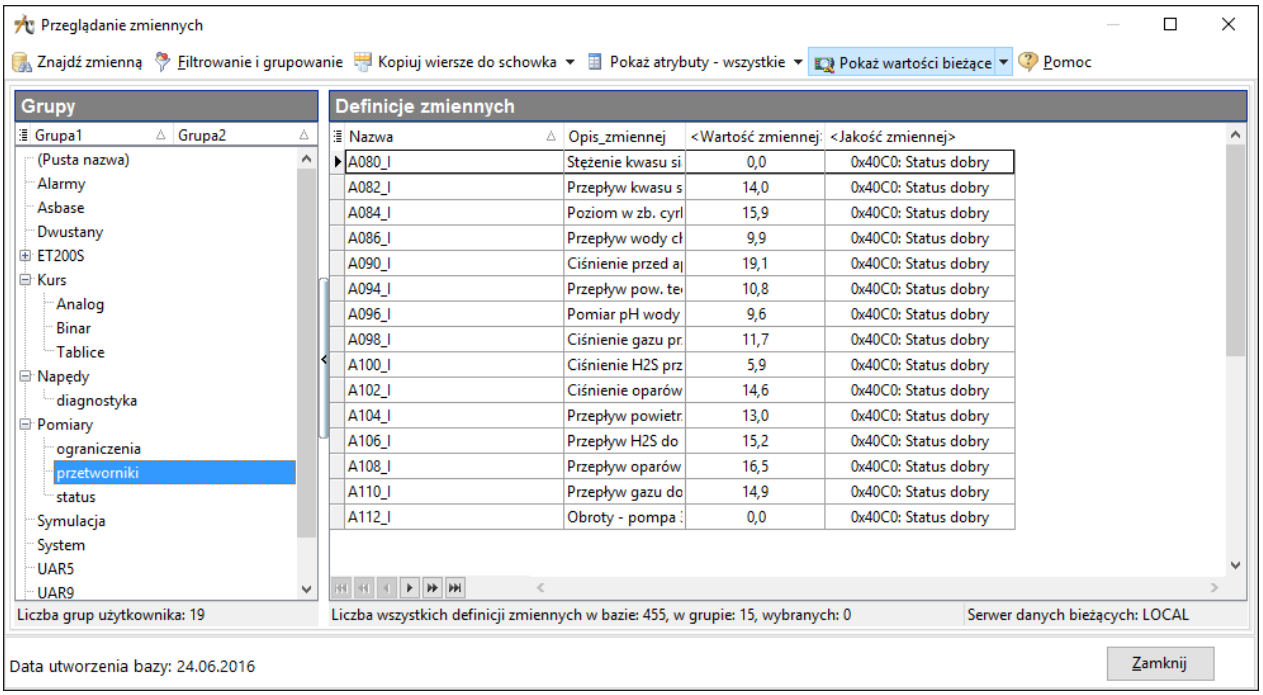
Architekt > *Bazy danych* > *Baza definicji zmiennych* > *Opcje*



Rys. Baza definicji zmiennych > Opcje > Maksymalny rozmiar nazwy zmiennej.

2.2.4. Przeglądanie zmiennych

Poza dostępem do definicji zmiennych, jaki zapewnia edytor bazy definicji zmiennych, definicje zmiennych mogą być przeglądanie za pomocą okna *'Przeglądanie zmiennych'*, uruchamianego przyciskiem *Pokaż.*



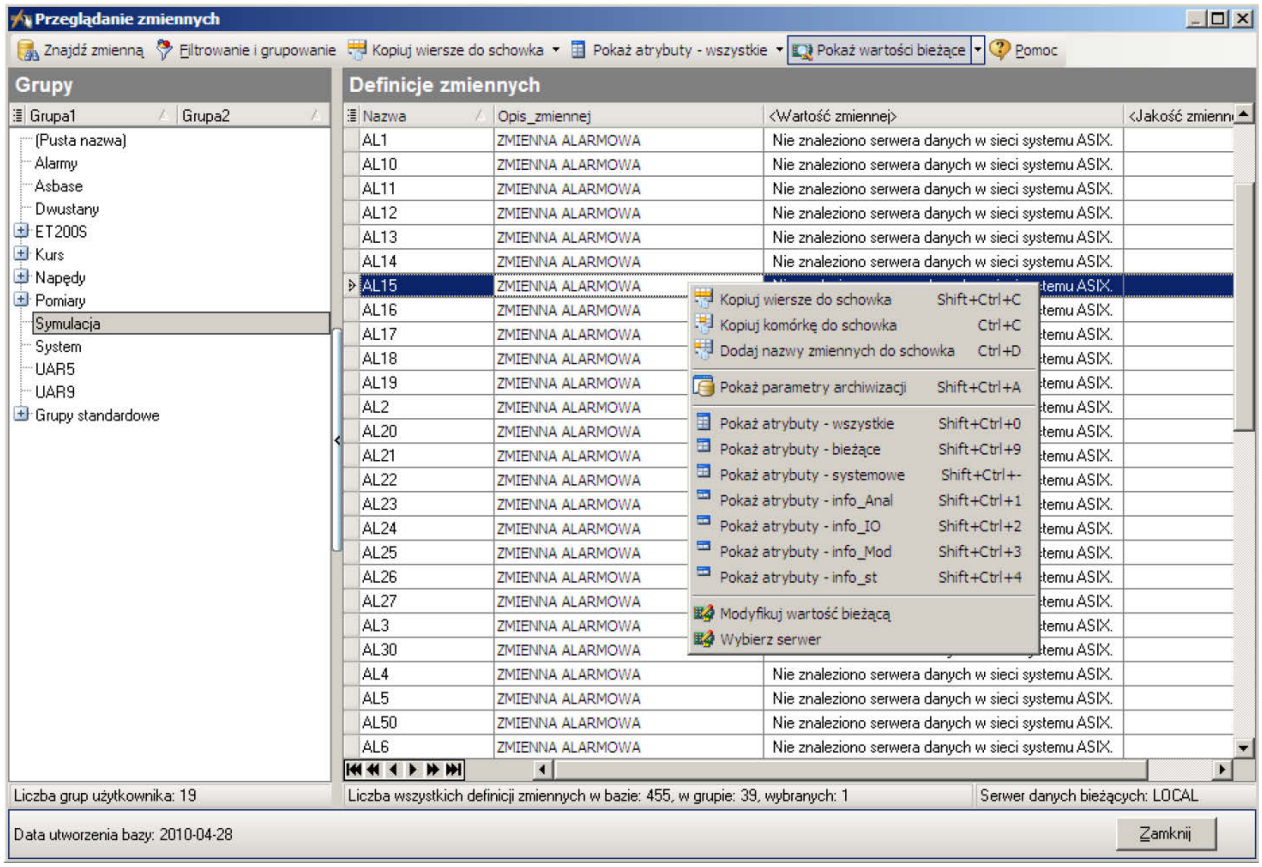
Rysunek. Okno 'Przeglądanie zmiennych'.

Dodatkowo okno daje możliwość przeglądania wartości bieżących zmiennych oraz modyfikowania tych wartości.

Do wyświetlenia wartości bieżących zmiennych służy przycisk ***Pokaż wartości bieżące***, po naciśnięciu którego zostają wyświetlone kolumny ***Wartość zmiennej*** i ***Jakość zmiennej***.

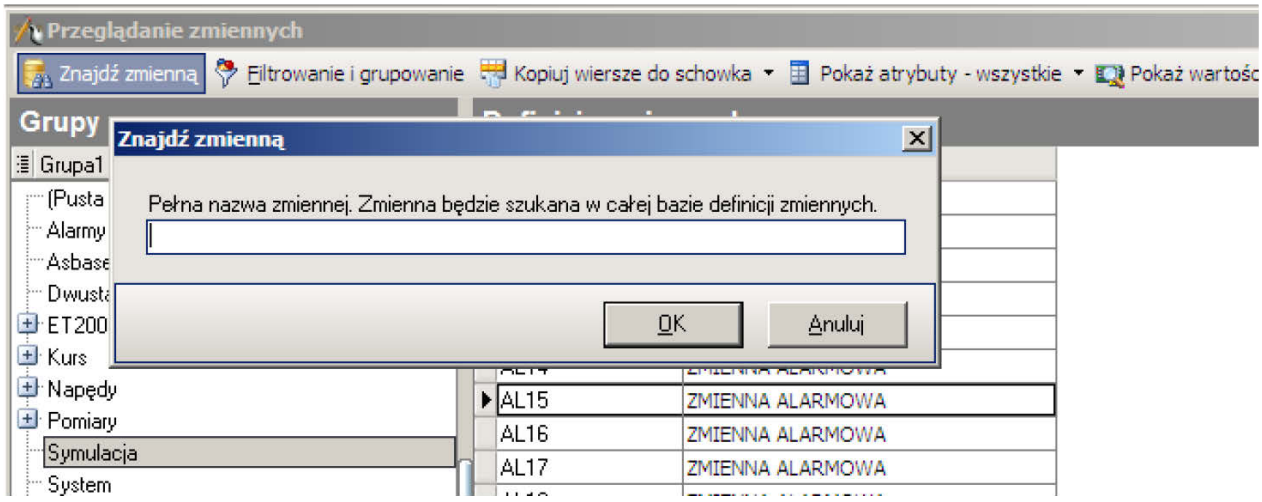
Funkcja jest dostępna tylko w oknie wyboru zmiennych wywoływanym bezpośrednio z programu Architekt. W trybie pokazywanie wartości bieżących ukrywany jest wiersz filtrów, ponieważ odświeżanie wartości bieżących i tak praktycznie uniemożliwia korzystanie z niego.

Aby dokonać zmiany wartości zmiennej, należy w wierszu wybranej zmiennej wybrać z rozwijanego menu ***Pokaż wartości bieżące*** polecenie ***Modyfikuj wartość bieżącą***.



Rysunek: Okno 'Przeglądanie zmiennych' - modyfikacja wartości bieżącej zmiennej.

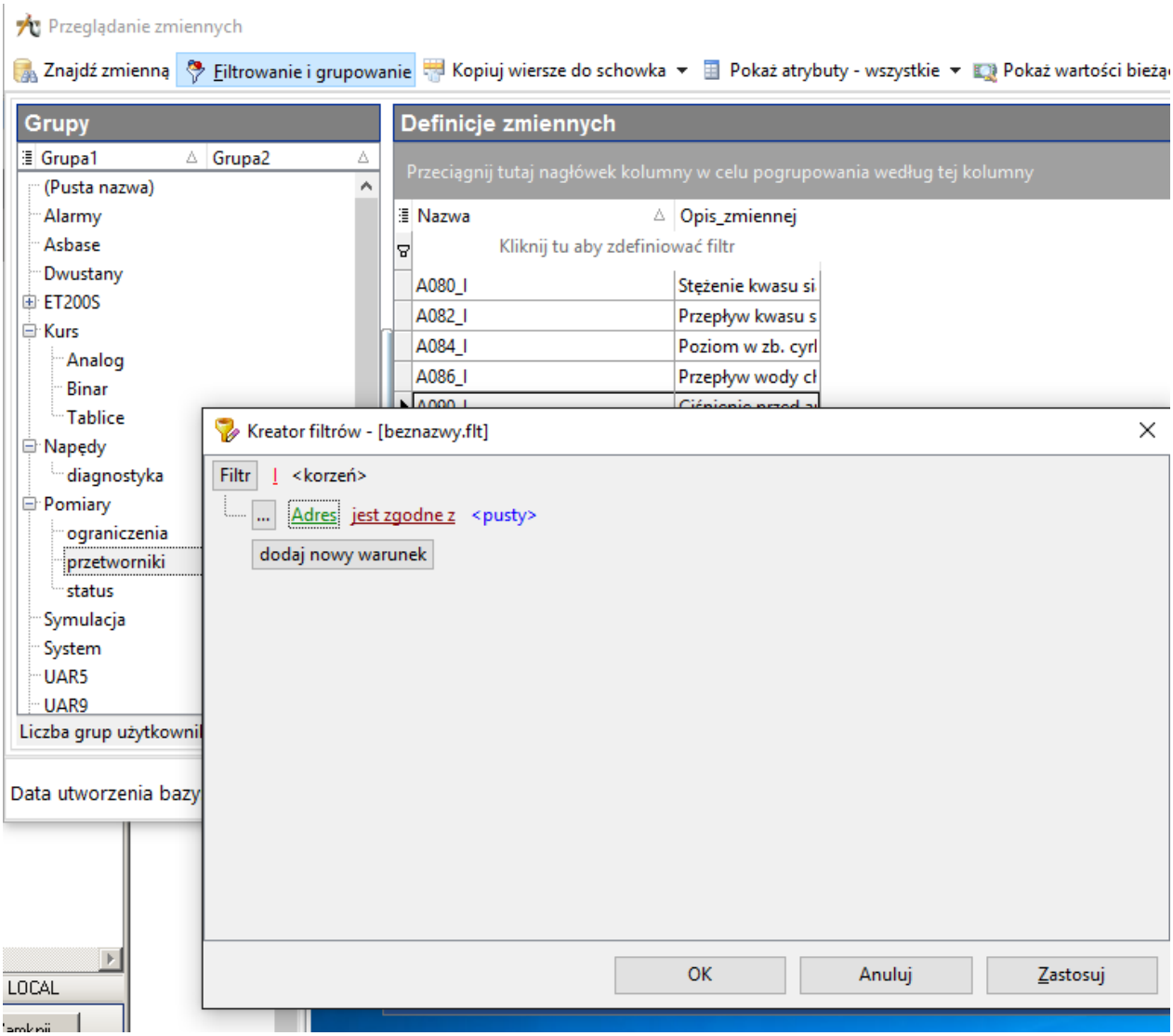
Okno *'Przeglądanie zmiennych'* umożliwia również wyszukiwanie zmiennych po nazwie.



Rysunek: *Okno 'Przeglądanie zmiennych' - wyszukiwanie zmiennych.*

2.2.4.1. Wyszukiwanie wartości atrybutu i filtrowanie rekordów po wartości atrybutu

Wyszukiwanie i filtrowanie rekordów wg wymaganej wartości atrybutu umożliwia *Kreator filtrów* wywoływany poleceniem ***Filtrowanie i grupowanie*** z paska poleceń okna '*Przeglądanie zmiennych*':



Rys. Okno kreatora filtrów dla rekordów bazy definicji zmiennych.

W edytorze definicji zmiennych istnieje mechanizm wyszukiwania inkrementalnego wartości dowolnego atrybutu. Po wpisaniu początkowych liter szukanej wartości w dowolną komórkę kolumny tego atrybutu mechanizm powoduje automatyczne przeniesienie do pierwszego rekordu z daną początkową wartością atrybutu. Mechanizm działa dla kolumn wszystkich atrybutów.

2.2.5 Praca ze skoroszytem Excel jako źródłem danych o atrybutach zmiennych bez generowania bazy definicji zmiennych

Użytkownik pracujący z najnowszą wersją pakietu Asix ma do dyspozycji możliwość takiego skonfigurowania pracy aplikacji, aby korzystała ona z definicji zmiennych pobieranych bezpośrednio z arkuszy kalkulacyjnych Excel . Arkusze te należy przygotować w sposób szczegółowo opisany w rozdziale Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > *2.2.6. Przygotowanie zewnętrznego źródła danych - arkusz Excel*.

Jak zadeklarować arkusz kalkulacyjny jako źródło definicji zmiennych?:

1. Zdefiniuj typ bazy definicji zmiennych jako *Tryb Źródło danych*

2. Załaduj źródła danych:

Baza definicji zmiennych > zakładka *Źródła danych* > wskaż plik *.XLS naciskając przycisk *Excel...*.

W oknie *'Wybierz tabele'* wskaż wszystkie pozycje i zatwierdź przyciskiem *OK*.

Krok powtórz tyle razy, aż zostaną zadeklarowane wszystkie pliki źródłowe.

3. Dane mogą być przeglądane w oknie *'Przeglądanie zmiennych'* po wybraniu komendy *Pokaż* .

Ten typ bazy definicji zmiennych można stosować tylko wtedy, gdy na komputerze pracuje system Asix i ZAWSZE jest on uruchamiany jako pierwszy. Wyklucza to sytuację, w której na komputerze pracuje serwer internetowy AsPortal lub AsWWW lub generowane są raporty przy pomocy programu AsixConnect.

---

[*] Baza jest generowana 'w locie' – w sposób niewidoczny dla użytkownika i tylko na czas działania aplikacji.

2.2.6. Przygotowanie zewnętrznego źródła danych – arkusz Excel

W przypadku, gdy źródłem danych niezbędnych do wygenerowania bazy definicji zmiennych (jak i w trybie tworzenia bazy *Tryb Źródło danych*, kiedy definicje zmiennych pobierane są bezpośrednio z arkusza Excel) jest arkusz Excel, dane muszą znajdować się w arkuszach w formacie XLS lub XLSX. Arkusze mogą być tworzone programem Microsoft Excel w wersji 97 lub późniejszej.

Architekt wyposażony został w prosty, ale w pełni funkcjonalny arkusz kalkulacyjny, który umożliwia edycję arkusza w formacie XLS - pomocny w sytuacji, gdy na komputerze nie jest zainstalowany pakiet Microsoft Excel.

Wyróżnionych zostało kilka rodzajów danych, które są umieszczone w odrębnych arkuszach w formie tabel [*] - rodzaje te zostaną omówione w kolejnych podrozdziałach:

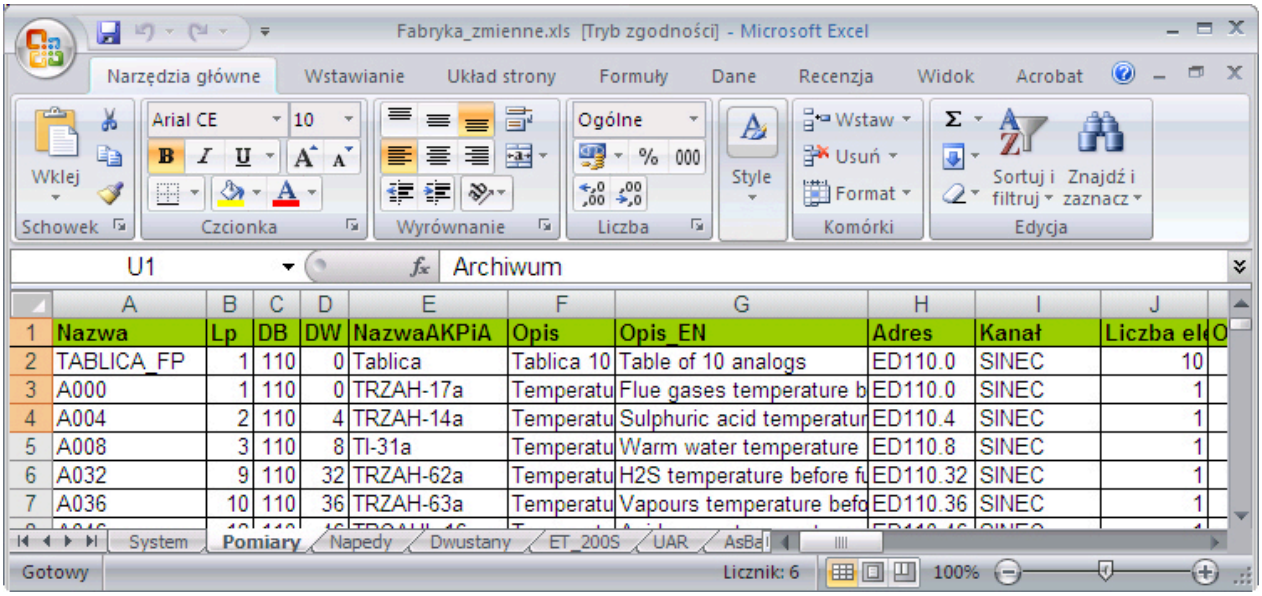
- **atrybuty zmiennych**,
- **grupowanie**,
- **zestawy atrybutów**,
- **funkcje przeliczające**,
- **nagłówki**,
- **języki atrybutów**,
- **nazwy zastępcze,**
- **parametry transformacji nazw atrybutów i wartości atrybutów.**

Zakładki, których nazwy rozpoczynają się od znaku '#', są to zakładki informacyjne, nie wchodzące w skład schematu bazy. Takie zakładki nie są widziane przez Architekta podczas wyboru źródeł danych.

2.2.6.1. Atrybuty zmiennych

Definicje zmiennych mogą znajdować się w jednym lub kilku arkuszach o dowolnej nazwie, w jednym lub kilku skoroszytach. Wszystkie atrybuty jednej zmiennej muszą znajdować się w jednym arkuszu.

W danym arkuszu nie muszą znajdować się wszystkie zdefiniowane w bazie atrybuty. Wszystkie atrybuty liczbowe, które nie są dla zmiennej zdefiniowane mają wartość NULL. Puste atrybuty tekstowe mają jako wartość tekst o długości 0.



Rysunek: Definicje zmiennych w arkuszu Excel w rozbiciu na kilka zakładek.

---

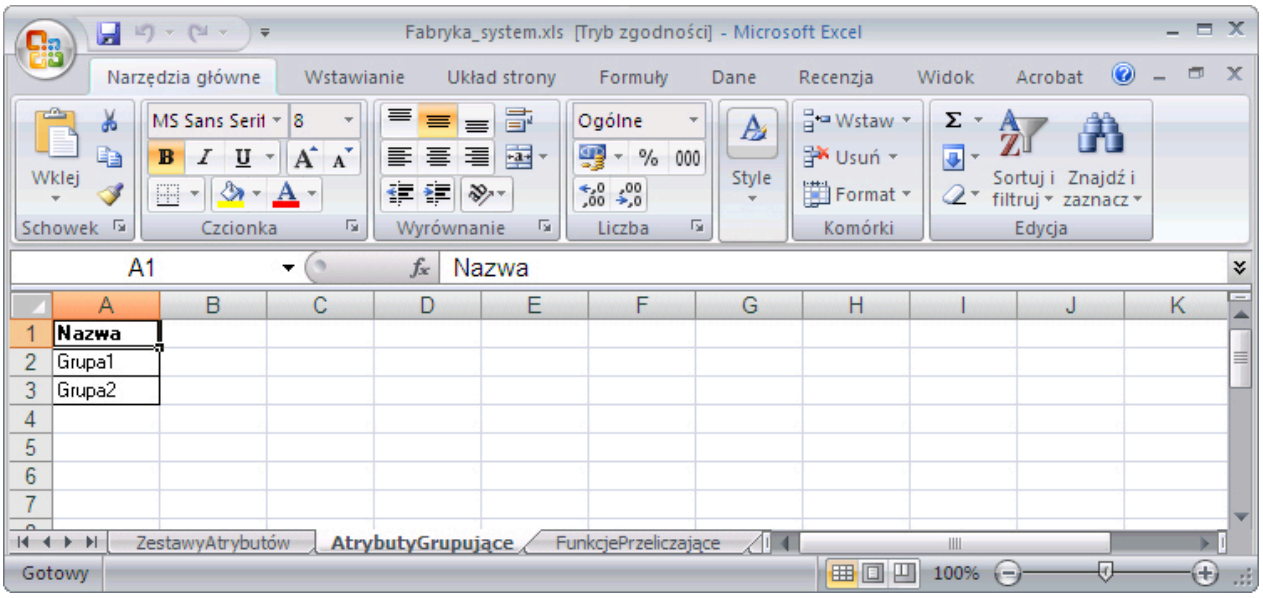
[*] Nazwa arkusza w skoroszycie Excel odpowiada nazwie tabeli w docelowej bazie, generowanej na podstawie arkusza.

Wszystkie nazwy atrybutów systemowych wewnętrznie są zadeklarowane w języku angielskim.

Przy imporcie danych nazwy atrybutów systemowych mogą być podawane zarówno w języku polskim jak i angielskim - zgodnie z ustalonym szablonem nazw atrybutów - patrz: ***Załącznik 1***.

2.2.6.2. Grupowanie

Atrybuty grupujące muszą znaleźć się w tabeli o ustalonej nazwie (w języku polskim: *AtrybutyGrupujące,* angielskim: *GroupingAttributes*).
Tabela składa się z jednej kolumny (pol.: *Nazwa*, ang.: *Name*). Każdy wiersz zawiera nazwę jednego atrybutu grupującego. Przykład - patrz: rysunek poniżej.
Nazwy atrybutów grupujących mogą być dowolne.

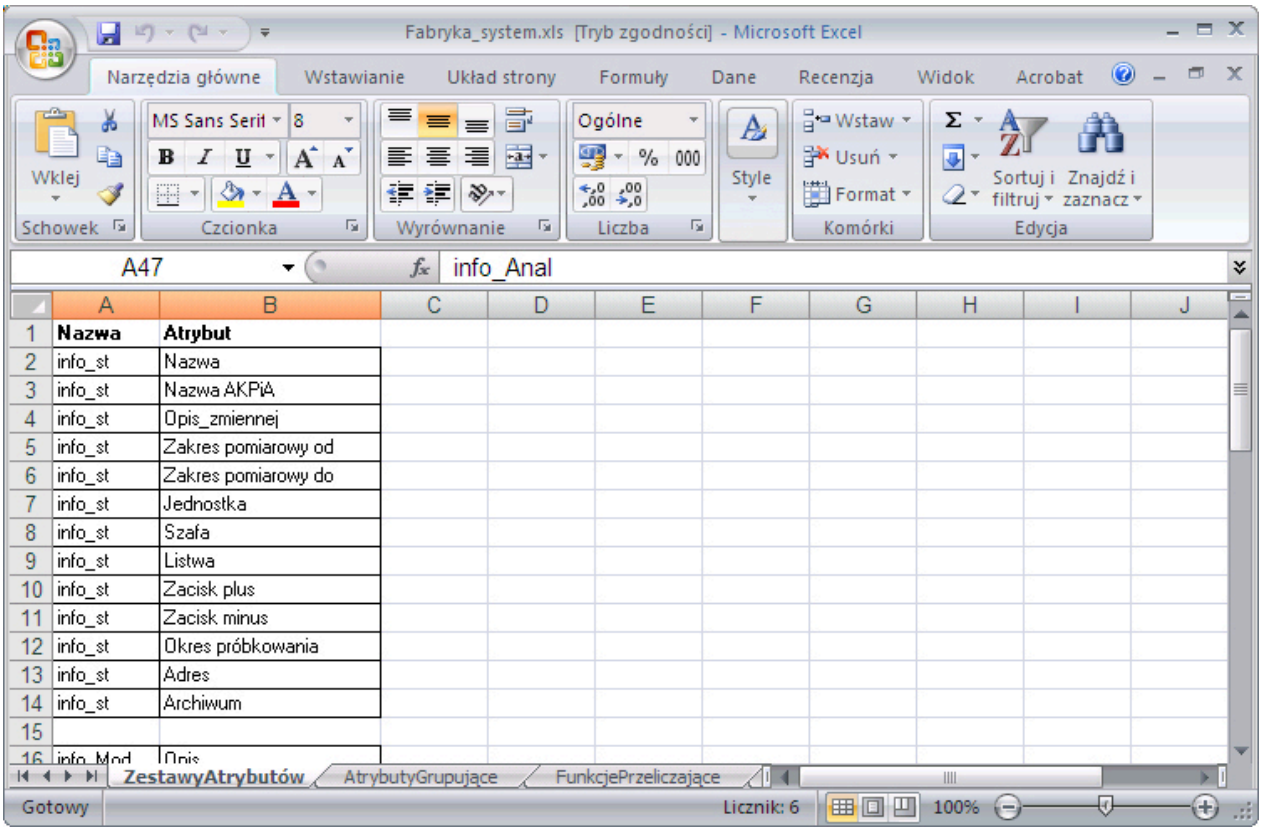


Rysunek: Arkusz Excel - atrybuty grupujące.

2.2.6.3. Zestawy atrybutów

Tabela zestawów atrybutów (o ustalonej nazwie - pol.: *ZestawyAtrybutów*, ang.: *AttributeSets*) składa się z dwóch kolumn. W pierwszej kolumnie (o nazwie - pol: *Nazwa*, ang.: *Name*) znajduje się nazwa zestawu, a w drugiej kolumnie (pol.: *Atrybut*, ang.: *Attribute*) - nazwa atrybutu.

Dla atrybutów systemowych należy użyć nazw atrybutów ustalonych dla języka polskiego lub angielskiego (nazw wewnętrznych lub nazw zadeklarowanych jako te, które mają się pojawiać w nagłówkach kolumn okna edytora definicji zmiennych, oknie podglądu definicji zmiennych lub oknie wyboru atrybutów - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > *2.2.6.5. Nagłówki*).



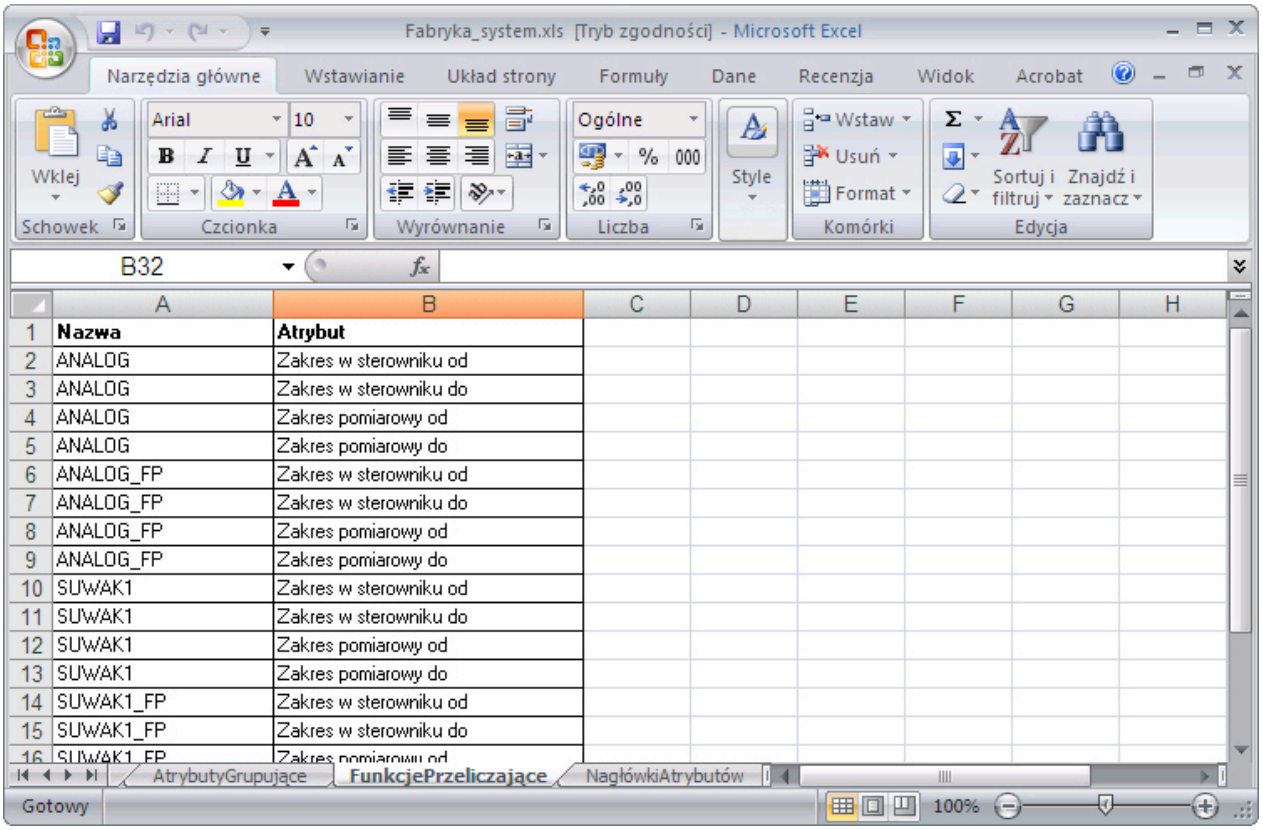
	A	B
1	**Nazwa**	**Atrybut**
2	info_st	Nazwa
3	info_st	Nazwa AKPiA
4	info_st	Opis_zmiennej
5	info_st	Zakres pomiarowy od
6	info_st	Zakres pomiarowy do
7	info_st	Jednostka
8	info_st	Szafa
9	info_st	Listwa
10	info_st	Zacisk plus
11	info_st	Zacisk minus
12	info_st	Okres próbkowania
13	info_st	Adres
14	info_st	Archiwum
15		
16	info_Mod	Opis

Rysunek: Arkusz Excel - zestawy atrybutów.

2.2.6.4. Parametry funkcji przeliczających

Tabela parametrów funkcji (o ustalonej nazwie - pol.: *FunkcjePrzeliczające*, ang.: *ConversionFunctions*) składa się z dwóch kolumn. W pierwszej kolumnie (pol.: *Nazwa*, ang.: *Name*) umieszczana jest nazwa funkcji, a w drugiej kolumnie (pol.: *Atrybut*, ang.: *Attribute*) - nazwa atrybutu.

Dla atrybutów systemowych należy użyć nazw atrybutów ustalonych dla języka polskiego lub angielskiego (nazw wewnętrznych lub nazw zadeklarowanych jako te, które mają się pojawiać w nagłówkach kolumn okna edytora definicji zmiennych, oknie podglądu definicji zmiennych lub oknie wyboru atrybutów - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > *2.2.6.5. Nagłówki*).



Rys. Arkusz Excel - parametry funkcji przeliczających.

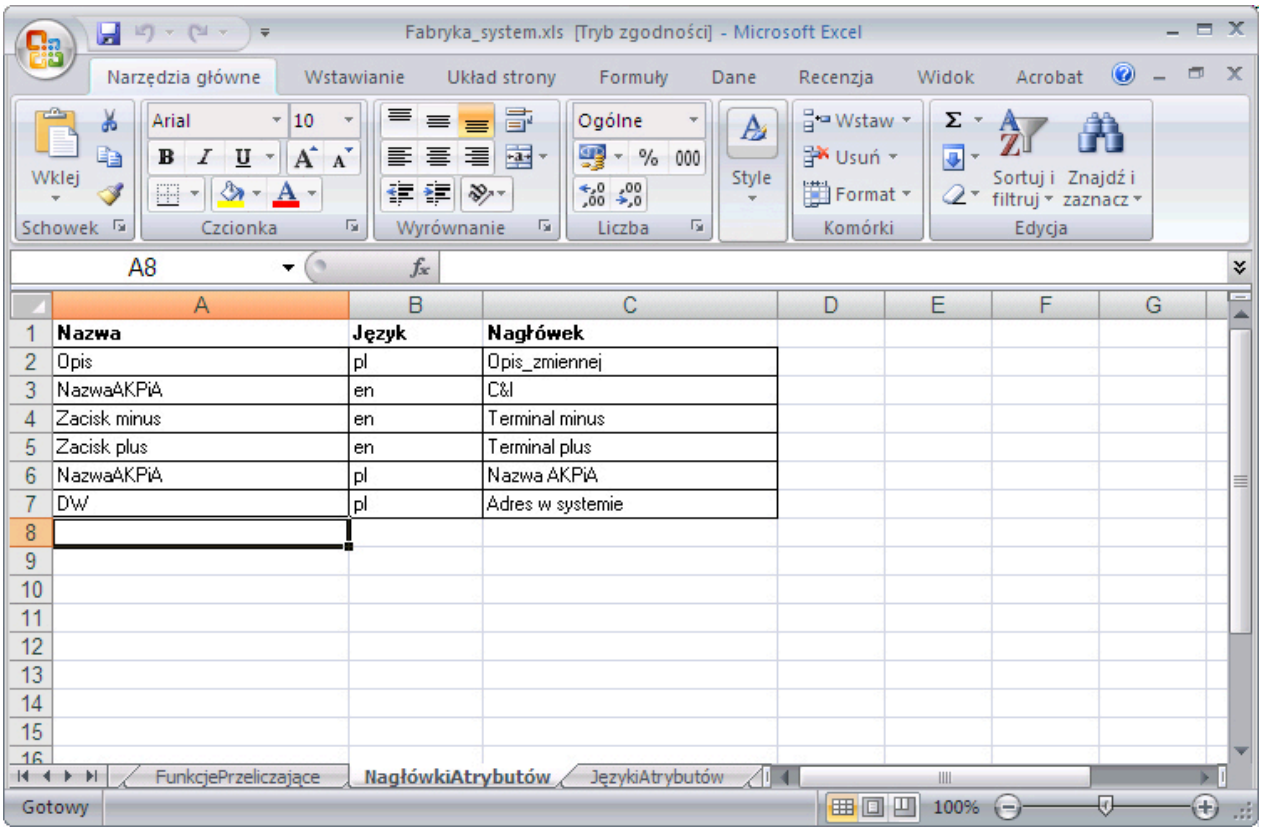
2.2.6.5. Nagłówki

Tabela nagłówków (polska nazwa ustalona tabeli: *NagłówkiAtrybutów*, angielska: ***AttributeCaptions***) pozwala zdefiniować nagłówki kolumn w oknach wyboru/edycji atrybutów zmiennych. Wszystkie teksty będące nagłówkami zmiennych muszą być unikalne. Nazwa jakiegokolwiek atrybutu nie może być nazwą nagłówka innego atrybutu.

W pierwszej kolumnie tabeli nagłówków (pol.: *Nazwa*, ang.: *Name*) należy umieścić nazwy atrybutów - dla atrybutów systemowych należy użyć nazw systemowych atrybutów w języku polskim lub angielskim; dla atrybutów użytkownika należy je podać w takiej formie, w jakiej zostały zdefiniowane w arkuszach zawierających definicje zmiennych.

W drugiej kolumnie (pol.: *Język*, ang.: *Language*) umieszczane są kody języków - zgodne z oznaczeniami ISO obsługiwanych języków. Zestawienie wszystkich kodów języków obsługiwanych przez Architekta zawiera ***Załącznik 3***.

Kolumna trzecia (pol.: *Nagłówek*, ang.: *Caption*) przeznaczona jest do określania wyświetlanych nazw nagłówków.



Rys. Arkusz Excel - nagłówki.

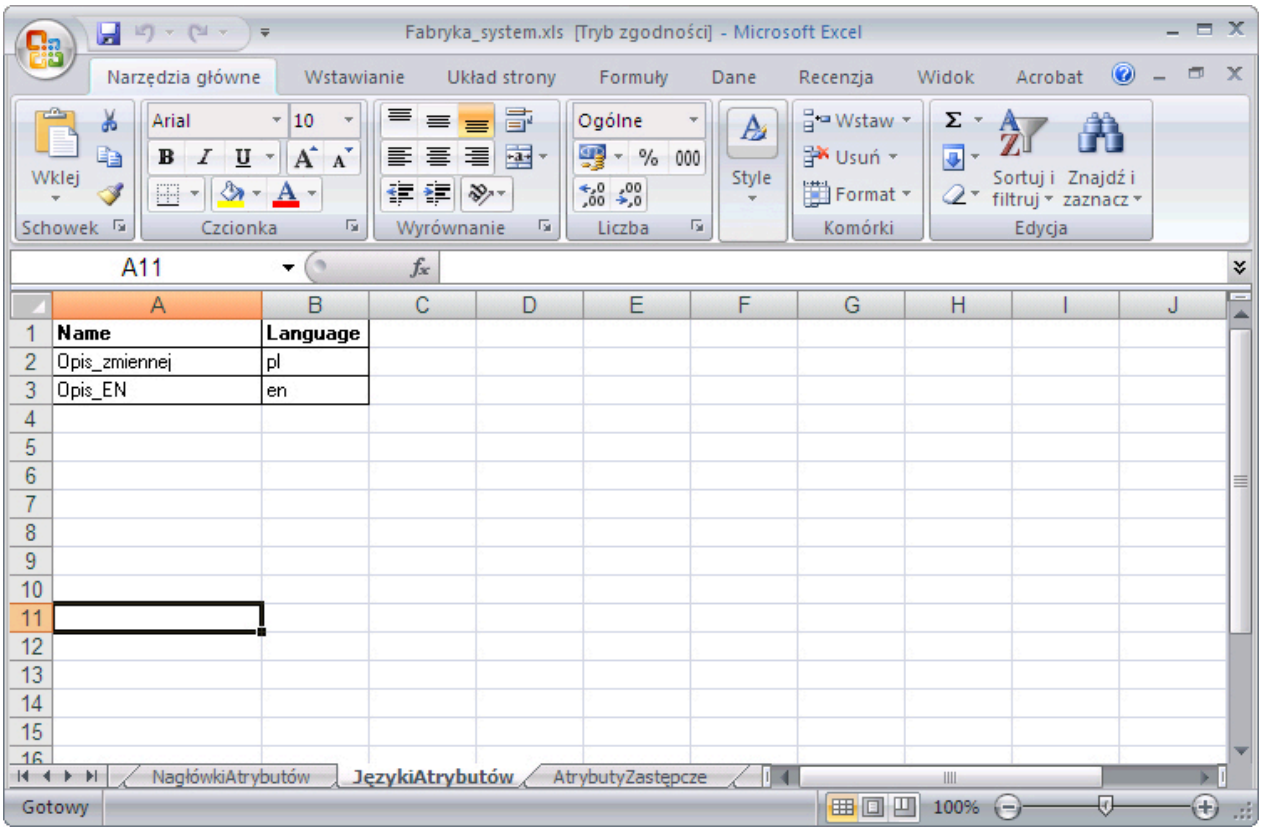
2.2.6.6. Języki atrybutów

Tabela o ustalonej nazwie (pol.: *JęzykiAtrybutów*, ang.: *AttributeLanguages*) pozwala określić, w jakim języku zdefiniowano wartości danego atrybutu. Informacja ta potrzebna jest zwykle wyłącznie w aplikacjach wielojęzycznych do podania języka atrybutu-aliasu oraz atrybutów tekstowych do prawidłowego działania edycji i wyświetlania.

W pierwszej kolumnie (pol.: *Nazwa*, ang.: *Name*) należy podać nazwy atrybutów (nazwy oryginalne lub nazwy zadeklarowane jako te, które mają się pojawiać w nagłówkach kolumn okna edytora definicji zmiennych, oknie podglądu definicji zmiennych lub oknie wyboru atrybutów - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > *2.2.6.5. Nagłówki*.).

W drugiej kolumnie (pol.: *Język*, ang.: *Language*) umieszczane są kody języków - zgodne z oznaczeniami ISO obsługiwanych języków.

Na poniższym ekranie zadeklarowano język angielski dla atrybutu *Opis_EN*, który zawiera angielskie opisy zmiennych. Po zadeklarowaniu atrybutu *Opis_EN* jako nazwy zastępczej dla atrybutu *Opis zmiennej* (patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > *2.2.6.7. Atrybuty zastępcze*) opisy zmiennych będą pojawiać się, stosowanie do załączonego języka aplikacji, albo w języku polskim albo w języku angielskim.



Rys. Arkusz Excel - języki atrybutów.

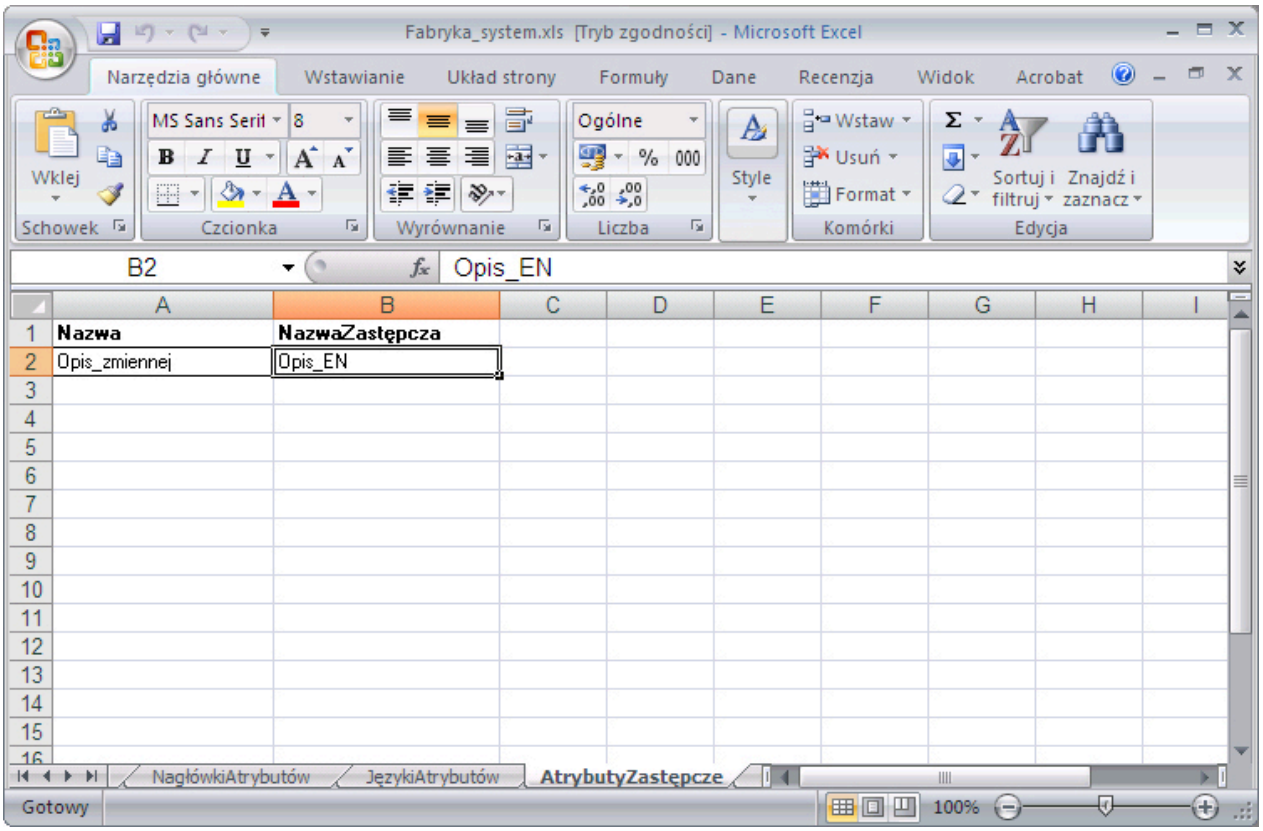
2.2.6.7. Atrybuty zastępcze

Definiowanie aliasów stosowane jest dla aplikacji wielojęzycznych, gdzie często zachodzi potrzeba, aby w jednym z języków zamiast pewnego atrybutu, zwanego dalej podstawowym, wyświetlić atrybut zastępczy (alias). Aby osiągnąć ten efekt, należy zdefiniować w bazie oba atrybuty i w tabeli aliasów zadeklarować, który jest atrybutem podstawowym a który zastępczym.

Dla atrybutu podstawowego i zastępczego należy zdefiniować język w tabeli języków - patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > *Języki atrybutów*.

Aliasy atrybutów muszą zostać określone w tabeli o ustalonej nazwie - pol.: *AtrybutyZastępcze*, ang.: *SubstituteAttributes*.

W pierwszej kolumnie (pol.: *Nazwa*, ang.: *Name)* powinny znaleźć się atrybuty podstawowe, a w drugiej kolumnie (pol.: *NazwaZastępcza*, ang.: *SubstituteName*) odpowiadające im aliasy.



Rys. Arkusz Excel - atrybuty zastępcze.

2.2.6.8. Parametry transformacji nazw atrybutów i wartości atrybutów

W przypadku generowania baz definicji zmiennych na podstawie definicji zmiennych znajdujących się w arkuszu kalkulacyjnym *.xls dostępny jest mechanizm transformacji nazw atrybutów i wartości atrybutów.

Parametry transformacji, tzn.:

- *NazwaŹródłaDanych*, *AtrybutWejściowy*, *AtrybutWyjściowy* dla transformacji nazw atrybutów
oraz
- *NazwaŹródłaDanych*, *Atrybut*, *WejściowaWartośćAtrybutu* i *WyjściowaWartośćAtrybutu* dla transformacji wartości atrybutów

należy zadeklarować odpowiednio na zakładkach: *PrzekształceniaNazwAtr* lub *PrzekształceniaWartościAtr* źródłowego pliku *.XLS.

Jako *NazwaŹródłaDanych* należy podać nazwę źródła danych taką jaka pojawia się w Architekcie w tabeli *Bazy danych* > *Baza definicji zmiennych* > *Źródła danych*, kolumnie *Nazwa źródła danych*. Nazwą tą może być np. nazwa arkusza Excel. Jako *AtrybutWejściowy* należy podać nazwę istniejącej kolumny. Jako *AtrybutWyjściowy* należy podać nową nazwę kolumny - nie może już istnieć kolumna o takiej nazwie.

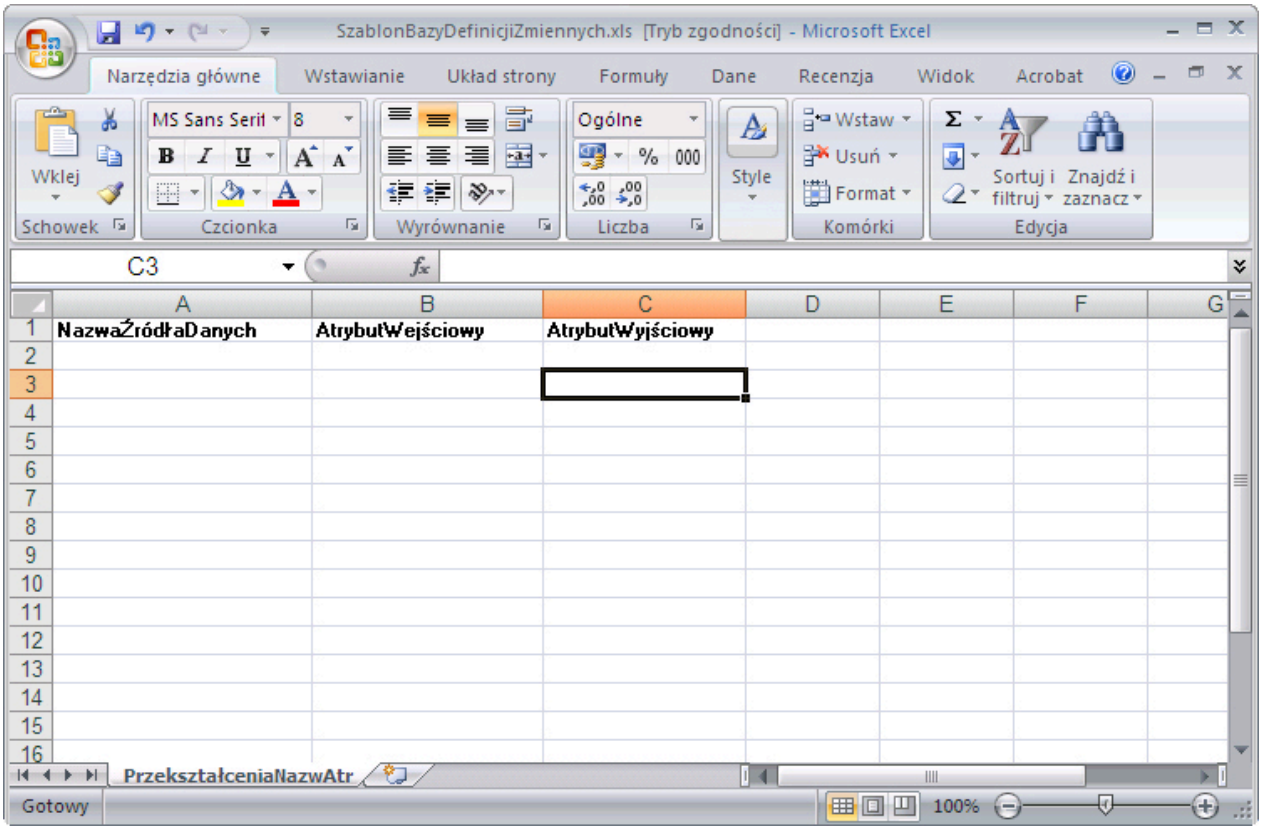
Podanie jako *AtrybutWejściowy* nazwy pustej powoduje **dodanie** do źródła danych nowej kolumny.

Podanie jako *AtrybutWyjściowy* nazwy pustej powoduje **usunięcie** z źródła danych wskazanej kolumny.

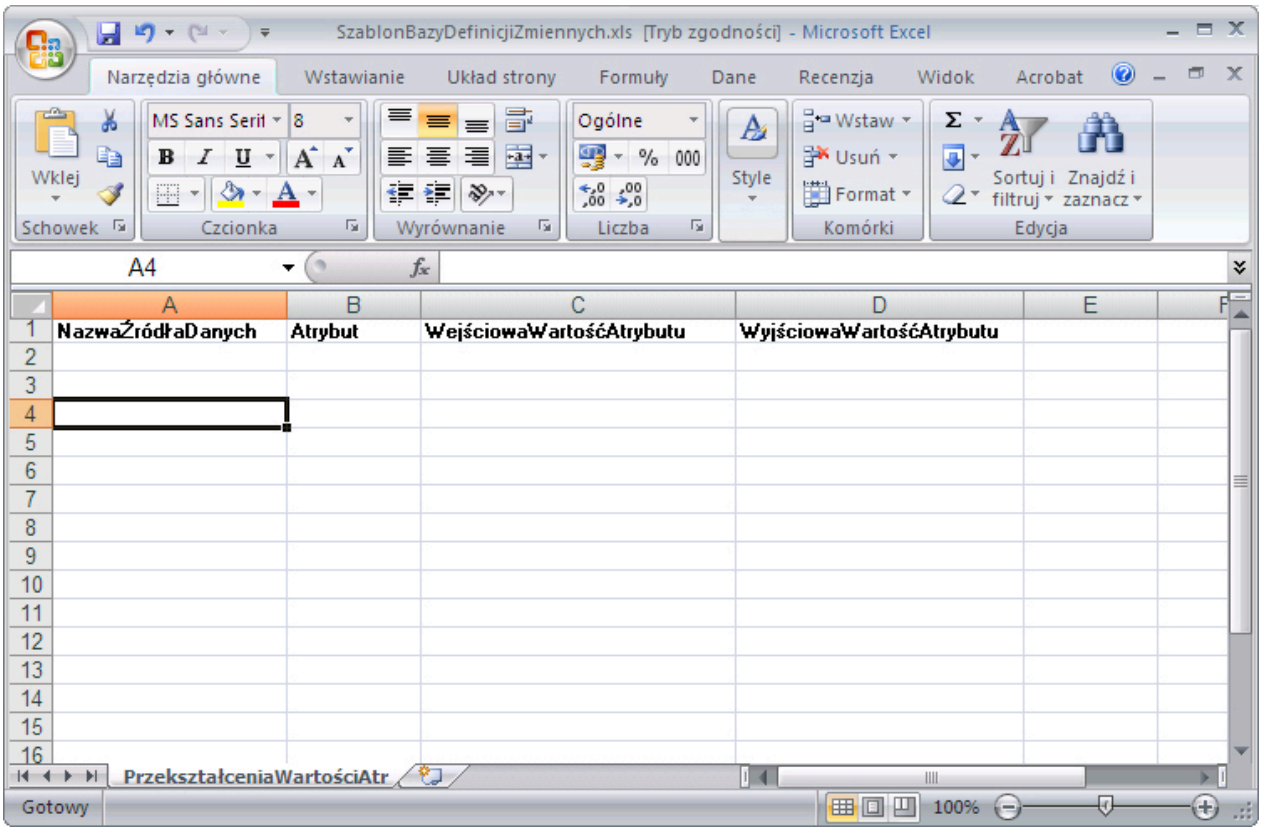
Jako *NazwaŹródłaDanych* należy podać wartość jak wyżej. Jako *Atrybut* należy podać nazwę atrybutu, którego wartości mają być przekształcane. Jako *WejściowaWartośćAtrybutu* należy podać wartość atrybut która ma zostać zmodyfikowana. Jako *WyjściowaWartośćAtrybutu* należy podać nową wartość atrybutu.

Podanie jako *WejściowaWartośćAtrybutu* nazwy pustej powoduje **dodanie** do źródła danych nowej wartości atrybutu.

Podanie jako *WyjściowaWartośćAtrybutu* nazwy pustej powoduje **usunięcie** z źródła danych wskazanej wartości atrybutu.



Rys. Arkusz Excel - PrzekształceniaNazwAtr.



Rysunek: Arkusz Excel - PrzekształceniaWartościAtr.

2.2.6.9. Zmienne używane

Arkusz z zestawem zmiennych używanych pozwala zawęzić liczbę zmiennych, których definicje mają być pobrane ze źródeł w trakcie generowania bazy definicji zmiennych.

Ten mechanizm ma zastosowanie głównie podczas generowania zbiorczych baz definicji zmiennych.

Zmienne używane muszą znaleźć się w tabeli (arkuszu) o ustalonej nazwie *ZmienneUżywane*. Tabela zawierać powinna jedną kolumnę *Nazwa*.

Zadeklarowanie arkusza (tabeli) *ZmienneUżywane* jako źródła danych, z którego wygenerowana zostanie baza definicji zmiennych (obok źródeł, z których pobrane zostaną definicje zmiennych) - pozwala kontrolować ilość zmiennych, które mają znaleźć się w bazie.

2.2.6.10. Szablon bazy definicji zmiennych

Architekt ułatwia przygotowanie pliku Excel z definicjami zmiennych, dając możliwość wygenerowania szablonowego skoroszytu zawierającego wybrane arkusze używane do wygenerowania bazy definicji zmiennych.



Wygenerowanie szablonu definicji zmiennych w pliku Excel:

1. Przejdź do:

   Baza definicji zmiennych > zakładka *Szablon* (przy załączonym typie standardowej bazy definicji zmiennych systemu *A*six tworzonej w trybie Generatora lub trybie *Źródło danych*)

2. Zaznacz arkusze, które mają się pojawić w szablonie

3. Kliknij przycisk ***Utwórz...***

2.2.6.11. Szablony wartości atrybutów

Przy generowaniu bazy dla pojedynczej bazy definicji zmiennych można wykorzystać transformację wartości atrybutów do zebrania w jednym miejscu często powtarzających się wartości atrybutów. Na przykład można podać dla każdej zmiennej jako parametry archiwizacji identyfikatory 'ArchiwizacjaPLC' lub 'ArchiwizacjaScript'. Następnie w tabeli transformacji wartości atrybutów podać, że 'ArchiwizacjaPLC' ma być zamieniana np. na D, 1s,,1s a 'ArchiwizacjaScript' na D, 10s,,10s. Patrz: Serwer Danych > Parametryzacja Serwera Danych > Obsługa baz definicji zmiennych aplikacji systemu Asix > Przygotowanie zewnętrznego źródła danych – arkusz Excel > 2.2.6.8. Parametry transformacji nazw atrybutów i wartości atrybutów.

2.2.6.12. Asix VarDef Excel Addin - edycja atrybutów bazy definicji zmiennych

Dodatek Asix VarDef Excel dedykowany arkuszowi kalkulacyjnemu od wersji MS Excel 2007 daje możliwość edytowania atrybutów definicji zmiennych przy użyciu tych samych edytorów, które dostępne są w przypadku edycji definicji zmiennych w bazie tworzonej w trybie Edytora - czyli bezpośrednio w programie Architekt (Typ bazy definicji zmiennych > *Standardowa baza definicji zmiennych* > *Tryb Edytor*).

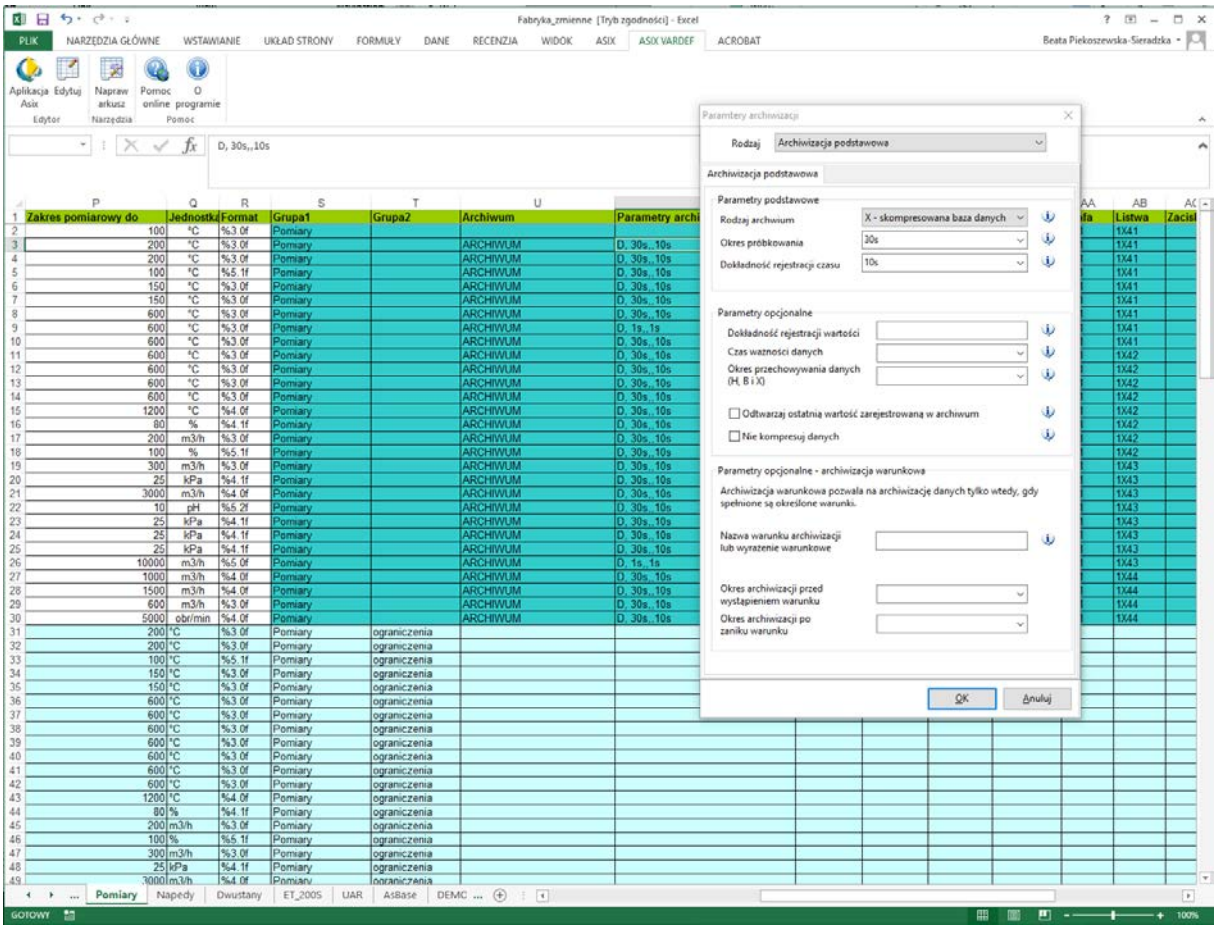
Lista atrybutów, które mogą być definiowane w arkuszu kalkulacyjnym MS Excel przy pomocy wspomnianych edytorów, obejmuje: Adres, Kanał, Funkcja przeliczająca, Format, Format_Evo, Archiwum, Parametry archiwizacji, limity, MaskaZmiennejKontrolnej, MeterId, MeterCycle, MeterInterval.

Opcja ***Aplikacja Asix*** pozwala załadować plik XML aplikacji ze wskazaniem stacji; zaleca się w pierwszej kolejności otworzyć plik Excel z definicjami zamiennych, a następnie załadować pliku XML aplikacji.

Opcja ***Edytuj*** pozwala na wywołanie edytora po zaznaczeniu komórki do edycji.

Opcja ***Napraw arkusz*** usuwa z arkusza formatowanie komórek znajdujących się poza obszarem danych.

Dodatek Asix VarDef Excel dostępny jest od wersji 9.1.0. pakietu Asix. Jego instalacja może odbyć się w trakcie instalacji pakietu Asix. Po zainstalowaniu pakietu Asix instalację dodatku Asix VarDef Excel można przeprowadzić (np. po aktualizacji pakietu Office) przy użyciu Konfiguratora Usług Pakietu Asix (asix extensions installer.exe).



Rys. Arkusz kalkulacyjny MS Excel - widok zakładki zainstalowanego dodatku Asix VarDef - edytor parametrów archiwizacji.

2.2.7. Przechowywanie opisów stanów zmiennych w bazie definicji zmiennych

W programie AsTrend od wersji 5 istnieje możliwość wyświetlania opisów stanów jako etykiet osi wartości dla wykresów fizycznych.

Mechanizm wyświetlania wartości stanów wymaga zadeklarowania zmiennych, które będą odpowiadać stanom zmiennych monitorowanych. W tym celu należy w pierwszej kolejności odpowiednio przygotować bazę definicji zmiennych - tak aby znalazły się w niej atrybuty obsługujące definicje stanów. Są to następujące atrybuty:

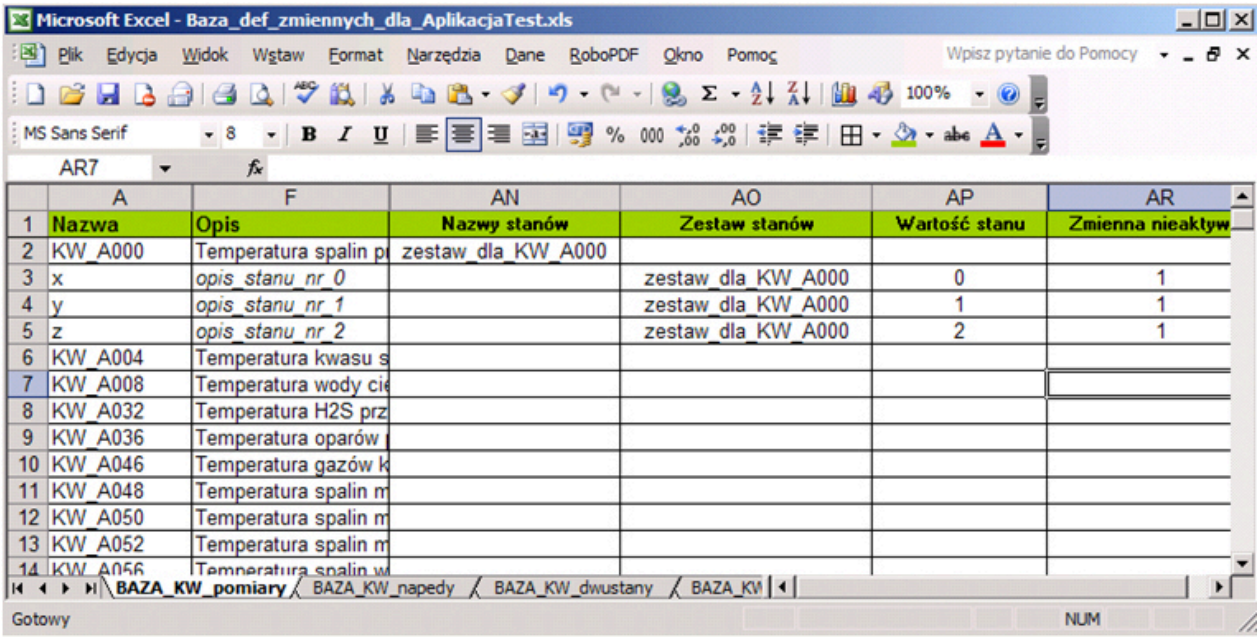
Nazwy stanów **-** nazwa zbioru stanów lub bezpośrednio zbiór opisów stanów zmiennej wg formatu:

<wartość_stanu_0>=<opis_stanu_0>;<wartość_stanu_1>=<opis_stanu_1>;..itd.

Zestaw stanów **-** nazwa zbioru stanów, do którego należy bieżący wiersz; atrybut używany tylko dla wierszy zawierających element zbioru opisów stanów;
Wartość stanu **-** wartość wyrażona w systemie dziesiętnym dla stanu, którego opis zawiera bieżący wiersz; atrybut używany tylko dla wierszy zawierających element zbioru opisów stanów zmiennej.

Należy zdefiniować tyle zmiennych, ile jest stanów danej zmiennej monitorowanej. Każda taka zmienna musi mieć zadeklarowany tekst, który będzie wyświetlany dla danego stanu, wartość stanu oraz nazwę zestawu stanów. Zmienne te powinny mieć włączony atrybut *Zmienna nieaktywna*.

Wszystkie zmienne z tekstami stanów należy powiązać ze zmienną monitorowaną za pomocą wspólnej nazwy zestawu stanów - którą dla zmiennej monitorowanej należy zadeklarować jako wartość atrybutu *Nazwy stanów*, a dla zmiennych z tekstami stanów jako wartość atrybutu *Zestaw stanów*. Teksty, które mają być wyświetlane na osi wartości wykresu należy zapisać jako wartości atrybutu *Opis* dla poszczególnych zmiennych odpowiadających wartościom stanów zmiennej monitorowanej.



	A	F	AN	AO	AP	AR
1	Nazwa	Opis	Nazwy stanów	Zestaw stanów	Wartość stanu	Zmienna nieaktyw
2	KW_A000	Temperatura spalin p	zestaw_dla_KW_A000			
3	x	opis_stanu_nr_0		zestaw_dla_KW_A000	0	1
4	y	opis_stanu_nr_1		zestaw_dla_KW_A000	1	1
5	z	opis_stanu_nr_2		zestaw_dla_KW_A000	2	1
6	KW_A004	Temperatura kwasu s				
7	KW_A008	Temperatura wody cie				
8	KW_A032	Temperatura H2S prz				
9	KW_A036	Temperatura oparów				
10	KW_A046	Temperatura gazów k				
11	KW_A048	Temperatura spalin m				
12	KW_A050	Temperatura spalin m				
13	KW_A052	Temperatura spalin m				
14	KW_A056	Temperatura spalin w				

Rysunek: Definicje atrybutów wykorzystywanych do wyświetlania opisów stanów.

Dodatkowo opis stanów dla zmiennych monitorowanych wyświetlany jest na etykietach i w polu pojawiającym się po kliknięciu wybranego punktu linii wykresu.

Przy dodawaniu zmiennych z bazy definicji zmiennych aplikacji Asixa oraz przy dodawaniu zmiennych zewnętrznych sprawdzane jest, czy zmienna ma zdefiniowane opisy stanów - jeśli tak, załączana jest aproksymacja schodkowa dla danej zmiennej.

2.2.8. Przechowywanie opisów stanów bitów w bazie definicji zmiennych

I PRZYPADEK - OPISY STANÓW BITÓW POCHODZĄ Z JEDNEJ ZMIENNEJ

Mechanizm wyświetlania wartości stanów bitów wymaga zadeklarowania zmiennych, które będą odpowiadać poszczególnym bitom zmiennej monitorowanej. W tym celu należy w pierwszej kolejności odpowiednio przygotować bazę definicji zmiennych - tak aby znalazły się w niej atrybuty obsługujące definicje stanów bitów. Są to następujące atrybuty:

Nazwy stanów **-** nazwa zbioru stanów (dla wiersza definicji zmiennej, której bity będą wyświetlane) lub opisy stanów bitów zmiennej (dla wierszy poszczególnych bitów);
Zestaw stanów **-** nazwa zbioru bitów, do którego należy bieżący wiersz; atrybut używany tylko dla wierszy poszczególnych bitów;
Wartość stanu - numer bitu, którego opis zawiera bieżący wiersz; atrybut używany tylko dla wierszy poszczególnych bitów.

Należy zdefiniować tyle zmiennych, ile jest bitów danej zmiennej monitorowanej. Każda taka zmienna musi mieć zadeklarowany tekst, który będzie wyświetlany dla danego stanu, wartość stanu oraz nazwę zestawu stanów. Zmienne te powinny mieć włączony atrybut ***Zmienna nieaktywna***.

Wszystkie bity z tekstami stanów należy powiązać ze zmienną monitorowaną za pomocą wspólnej nazwy zestawu stanów - którą to nazwę dla zmiennej monitorowanej należy zadeklarować jako wartość atrybutu ***Nazwy stanów***, a dla zmiennych z tekstami stanów bitów jako wartość atrybutu ***Zestaw stanów***. Teksty, które mają być wyświetlane na osi wartości wykresu należy zapisać jako wartości atrybutu ***Nazwy stanów*** dla zmiennych odpowiadających poszczególnym bitom zmiennej monitorowanej, wg. formatu:

0=<*opis stanu*>;1=<*opis stanu*>

Natomiast atrybut ***Wartość stanu*** określa numer bitu zmiennej monitorowanej.

	A	B	C	D	E	J
1	Zmienna nieaktywna	Nazwa	Nazwy stanów	Zestaw stanów	Wartość stanu	Nazw
2	0	N01				TY_16
3	0	N02				QY_1
4	0	N03				FY-10
5	0	N04				LY-1a
6	0	N05				TY_21
7	0	N06				TY_22
8	0	N07				TY_23
9	0	N08				FY_10
10	0	N09	N09_STATUS			FY_12
11	0	N10				FY_13
19	0	N13				
20	1	N09_STATUS00	0= - ;1=WYŁ	N09_STATUS	0	
21	1	N09_STATUS01	0= - ;1=ZAŁ	N09_STATUS	1	
22	1	N09_STATUS02	0= - ;1=ZAM	N09_STATUS	2	
23	1	N09_STATUS03	- rezerwa	N09_STATUS	3	
24	1	N09_STATUS04	0= - ;1=AWE	N09_STATUS	4	
25	1	N09_STATUS05	0= - ;1=REM	N09_STATUS	5	
26	1	N09_STATUS06	0= - ;1=MAN	N09_STATUS	6	
27	1	N09_STATUS07	0= - ;1=AUT	N09_STATUS	7	
28	1	N09_STATUS08	0= - ;1=kr.OTW	N09_STATUS	8	
29	1	N09_STATUS09	0= - ;1=kr.ZAM	N09_STATUS	9	
30	1	N09_STATUS10	- rezerwa	N09_STATUS	10	
31	1	N09_STATUS11	0= - ;1=Got.EL	N09_STATUS	11	
32	1	N09_STATUS12	0= - ;1=Got.TECHN	N09_STATUS	12	
33	1	N09_STATUS13	0=ZDALNY;1=LOKALNY	N09_STATUS	13	
34	1	N09_STATUS14	- rezerwa	N09_STATUS	14	
35	1	N09_STATUS15	- rezerwa	N09_STATUS	15	

Rysunek: Definicje atrybutów wykorzystywanych do wyświetlania opisów stanów bitów.

II PRZYPADEK - OPISY STANÓW BITÓW POCHODZĄ Z RÓŻNYCH ZMIENNYCH DWUSTANOWYCH

Mechanizm wyświetlania wartości stanów bitów zmiennych dwustanowych nie wymaga deklarowania dodatkowych zmiennych. Atrybutem obsługującym definicje stanów bitów jest w tym przypadku atrybut:

Nazwy stanów **-** zawierający opisy stanów zmiennej wg. formatu:

0=<*opis stanu*>;1=<*opis stanu*>

	A	B	C	D	E	F	G	H	I	
	Zmienna nieak	Nazwa	Nazwy stanów	Zestaw stanów	Wartość stanu	DB	DW	Bit	Opis	O
2	0	B000				45	0	0	Kontrola zasilania szafy	C
3	0	B001				45	1	0	Kontrola zasilania modułu 6ER1	M
4	0	B002				45	2	0	Gaz koksowniczy	C
5	0	B003				45	3	0	Zasilanie alarm	P
6	0	B004				45	4	0	Zasilanie alarm	P
7	0	B005				45	5	0	Pompa cyrkulacyjna 304	C
8	0	B006				45	6	0	Zasilanie alarm	P
9	0	B007				45	7	0	Zasilanie alarm	P
10	0	B008				45	8	0	Zasilanie alarm	P
11	0	B009				45	9	0	Zasilanie alarm	P
12	0	B10_0	0= - ;1=ZAM			45	10	0	Zawór - zamknięty	B
13	0	B10_1	0= - ;1=OTW			45	10	1	Zawór - otwarty	B
14	0	B10_2	0= - ;1=kr.ZAM			45	10	2	Zawór - krańcówka zamknięty	B
15	0	B10_3	0= - ;1=kr.OTW			45	10	3	Zawór - krańcówka otwarty	B
16	0	B10_4	0= - ;1=AWE			45	10	4	Zawór - Awaria elektryczna	B
17	0	B10_5	0= - ;1=REM			45	10	5	Zawór - Praca w trybie REMONTOWY	B
18	0	B10_6	0= - ;1=MAN			45	10	6	Zawór - Praca w trybie MANUALNYM	B
19	0	B10_7	0= - ;1=AUT			45	10	7	Zawór - Praca w trybie AUTOMATYCZ	B
20	0	B172_36	B172_36_STATUS			172	36		Zawór	B
21										

Rysunek: Definicje atrybutów wykorzystywanych do wyświetlania opisów stanów bitów zmiennych dwustanowych.

2.2.9. Konwertowanie bazy definicji zmiennych z formatu Paradox na Jet lub MS SQL

Architekt stwarza możliwość przekonwertowania wszystkich danych z bazy definicji zmiennych BDE utworzonej w starszej wersji pakietu Asix (baza w formacie Paradox) do bazy Jet lub SQL.

Dla starszych wersji systemu Asix baza definicji zmiennych w formacie Paradox składa się z:
- bazy atrybutów zmiennych - fizycznie pliki bazy Paradox;
- schematu bazy definicji zmiennych - plik *.TXT opisujący wszystkie atrybuty jakie może posiadać zmienna;
- zestawu zmiennych - plik tekstowy *.ZEZ zawierający definicję ułożenia grup zmiennych (jedna baza definicji zmiennych może zawierać dowolną ilość zestawów zmiennych).

Zasadniczo wskazane było przy tego typu bazach przygotowywanie definicji zmiennych w arkuszu kalkulacyjnym Excel, w oparciu o który następnie generowana była baza definicji zmiennych przy użyciu programu *Menedżer Bazy Zmiennych*. W przypadku mniejszych aplikacji - bazy definicji zmiennych bezpośrednio tworzone były i edytowane za pomocą programu *Edytor Bazy Zmiennych*.

Jak przekonwertować bazę definicji zmiennych formatu Paradox utworzoną w starszych wersjach Asixa na bazę Jet lub MS SQL?:

I przypadek, kiedy baza definicji zmiennych tworzona była bezpośrednio za pomocą Edytora Bazy Zmiennych (z pominięciem pliku *.XLS)

1. Zdefiniuj typ bazy definicji zmiennych jako *Tryb Generator*
2. Określ format oraz nazwę bazy definicji zmiennych, która ma zostać wygenerowana:

Baza definicji zmiennych > zakładka *Format* > format bazy danych ustawiony na: *Baza danych serwera Microsoft SQL Serwer* lub *Plik w formacie Mdb (Jet/Microsoft Access)*

W przypadku baz danych MS SQL wymagane jest określenie parametrów:
- *Nazwa serwera*;
- *Nazwa bazy*;
- *Katalog bazy*;
- *Tryb uwierzytelniania*:
  - *Windows*,
  - *Serwer MSQL* - wymaga określenia *Nazwy użytkownika* oraz *Hasła*.

3. Określ źródło danych:

Baza definicji zmiennych > zakładka *Źródła danych* > wskaż pliki ZEZ/DB (przycisk Asix - pliki zez/db).

4. W oknie *'Wybierz tabele'* wskaż wszystkie pozycje i zatwierdź przyciskiem *OK*.



Rys. Okno wyboru tabel do konwersji bazy na format Jet lub MS SQL.

5. Wybierz polecenie *Generuj* .

6. Otwórz bazę i w razie potrzeby wprowadź zmiany do atrybutów zmiennych przy użyciu *Edytora definicji zmiennych*; zmodyfikuj w miarę potrzeby schemat bazy definicji zmiennych za pomocą *Edytora schematu* (dostępny po otwarciu bazy).

Edycja zawartości bazy możliwa jest po przejściu na *tryb Edytor:*
Baza definicji zmiennych > zakładka *Typ* > opcja *Tryb Edytor*.

II przypadek, kiedy baza definicji zmiennych generowana była za pomocą Menedżera Bazy Zmiennych w oparciu o definicje zmiennych przygotowane w arkuszu kalkulacyjnym Excel

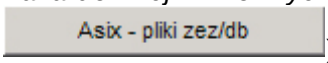
Założenie: *po konwersji bazy definicje zmiennych i schemat bazy będą mogły być aktualizowane z arkuszy Excel zgodnych z obowiązującym dla Asix 5-6 szablonem.*

1. Zdefiniuj typ bazy definicji zmiennych jako *Tryb Generator*
2. Określ format oraz nazwę bazy definicji zmiennych, która ma zostać wygenerowana:

Baza definicji zmiennych > zakładka *Format* > format bazy danych ustawiony na: *Baza danych serwera Microsoft SQL Serwer* lub *Plik w formacie Mdb (Jet/Microsoft Access)*

W przypadku baz danych MS SQL wymagane jest określenie parametrów:
- *Nazwa serwera*;
- *Nazwa bazy*;
- *Katalog bazy*;
- *Tryb uwierzytelniania*:
  - *Windows*,
  - *Serwer MSQL* - wymaga określenia *Nazwy użytkownika* oraz *Hasła*.

3. Określ źródło danych:

Baza definicji zmiennych > zakładka *Źródła danych* > wskaż pliki ZEZ/DB (przycisk 

)

4. W oknie *'Wybierz tabele'* wskaż wszystkie pozycje i zatwierdź przyciskiem *OK*.

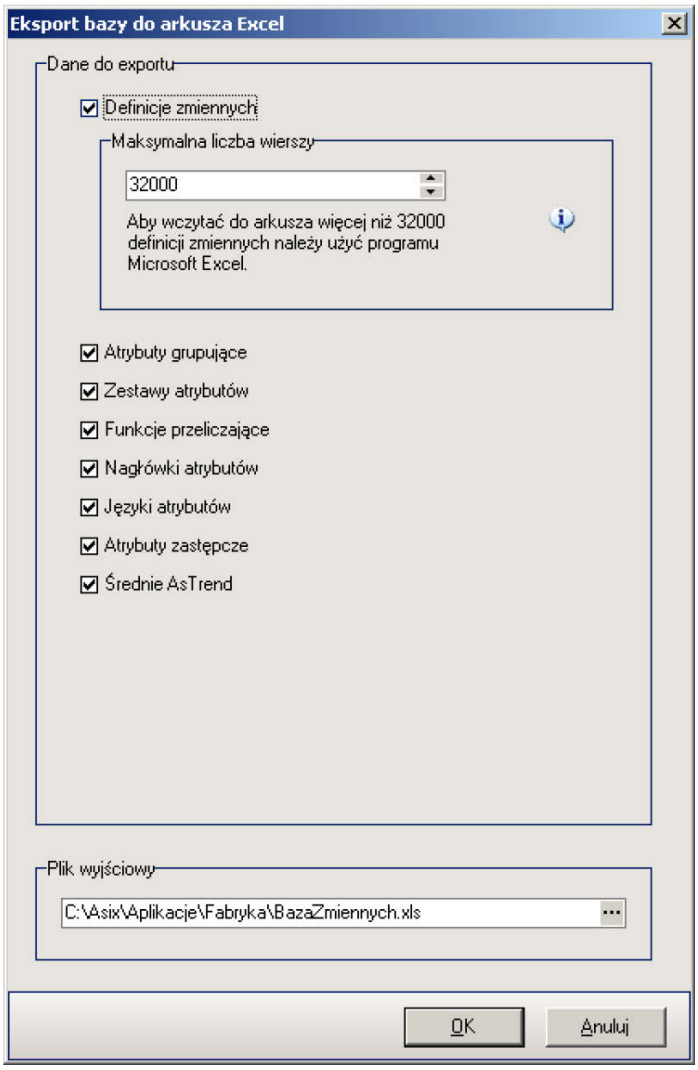
5. Wybierz polecenie *Generuj* , a następnie otwórz bazę i sprawdź, czy baza została prawidłowo wygenerowana; w razie potrzeby wprowadź zmiany do atrybutów zmiennych przy użyciu *Edytora definicji zmiennych*; wprowadź również zmiany (jeśli istnieje taka potrzeba) do schematu bazy definicji zmiennych za pomocą *Edytora schematu* (dostępny po otwarciu bazy).

Edycja bazy możliwa jest po przejściu na *tryb Edytor:*
Baza definicji zmiennych > zakładka *Typ* > opcja *Tryb Edytor* (baza musi być zamknięta w trakcie zmiany trybów)

7. Zamknij bazę i przejdź na tryb edytora:

Baza definicji zmiennych > zakładka *Typ* > opcja *Tryb Edytor*

8. Ponownie otwórz wygenerowaną bazę definicji zmiennych i wyeksportuj ją do arkusza Excel za pomocą komendy *Eksportuj* . W oknie *'Eksport bazy do Arkusza Excel'* wybierz wszystkie dostępne rodzaje danych do eksportu i zmień nazwę pliku wyjściowego na Schemat.xls (nazwa dowolna); zatwierdź zmiany przyciskiem *OK*.



Rys. Okno 'Eksport bazy do arkusza Excel'.

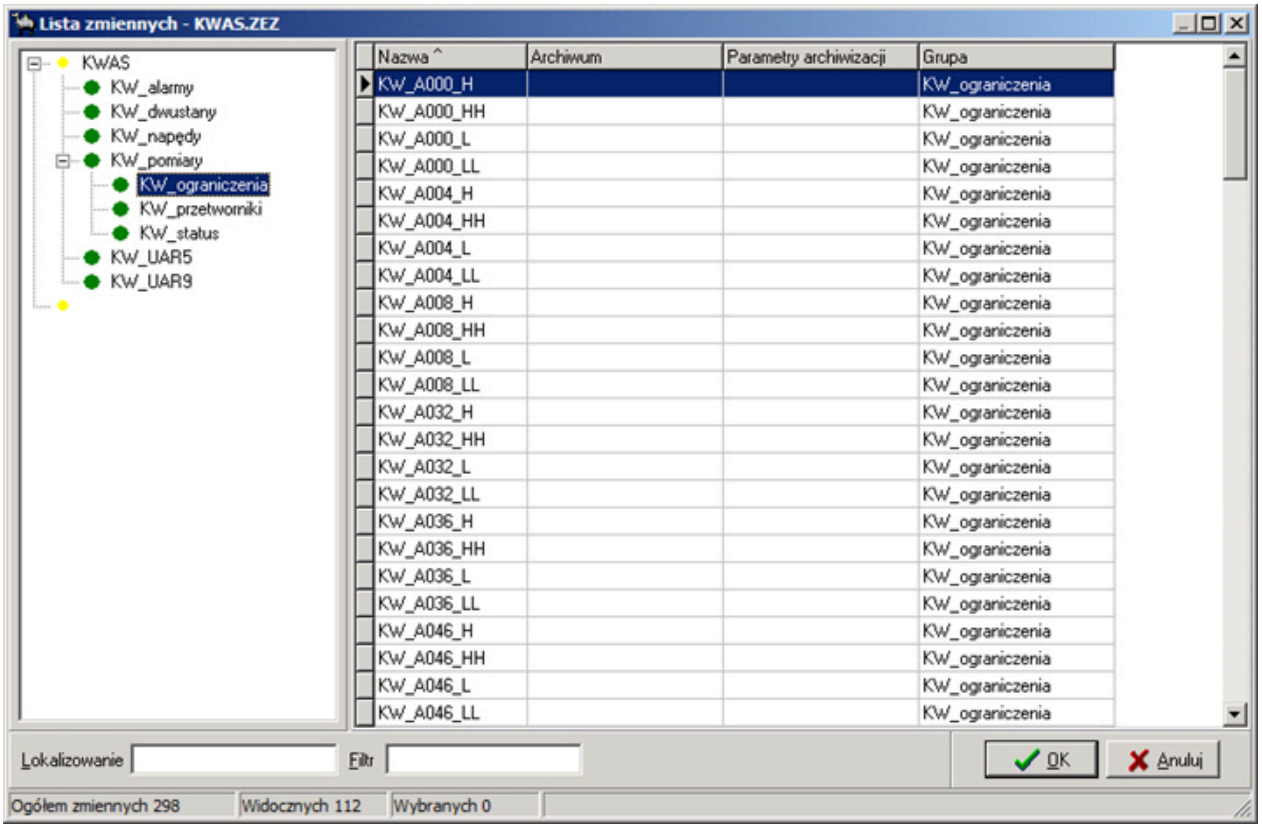
9. Zamknij bazę.

10. Otwórz skoroszyt Schemat.xls;

W arkuszu ***AtrybutyGrupujące*** znajdują się atrybuty grupujące, które należy 'ręcznie' dodać do definicji zmiennych przechowywanych w skoroszycie Excel służącym do tej pory jako źródło generowania i aktualizowania bazy definicji zmiennych formatu Paradox.

W starszych wersjach Asixa w bazie definicji zmiennych istniała możliwość zadeklarowania dla każdej zmiennej wyłącznie jednego atrybutu grupującego o ustalonej nazwie *Grupa*. Możliwe było jednak ułożenie grup zadeklarowanych w atrybucie *Grupa* w strukturę hierarchiczną typu drzewo wyświetlaną w oknie wyboru zmiennych - co pozwalało 'widzieć' daną zmienną jako element przynależny do

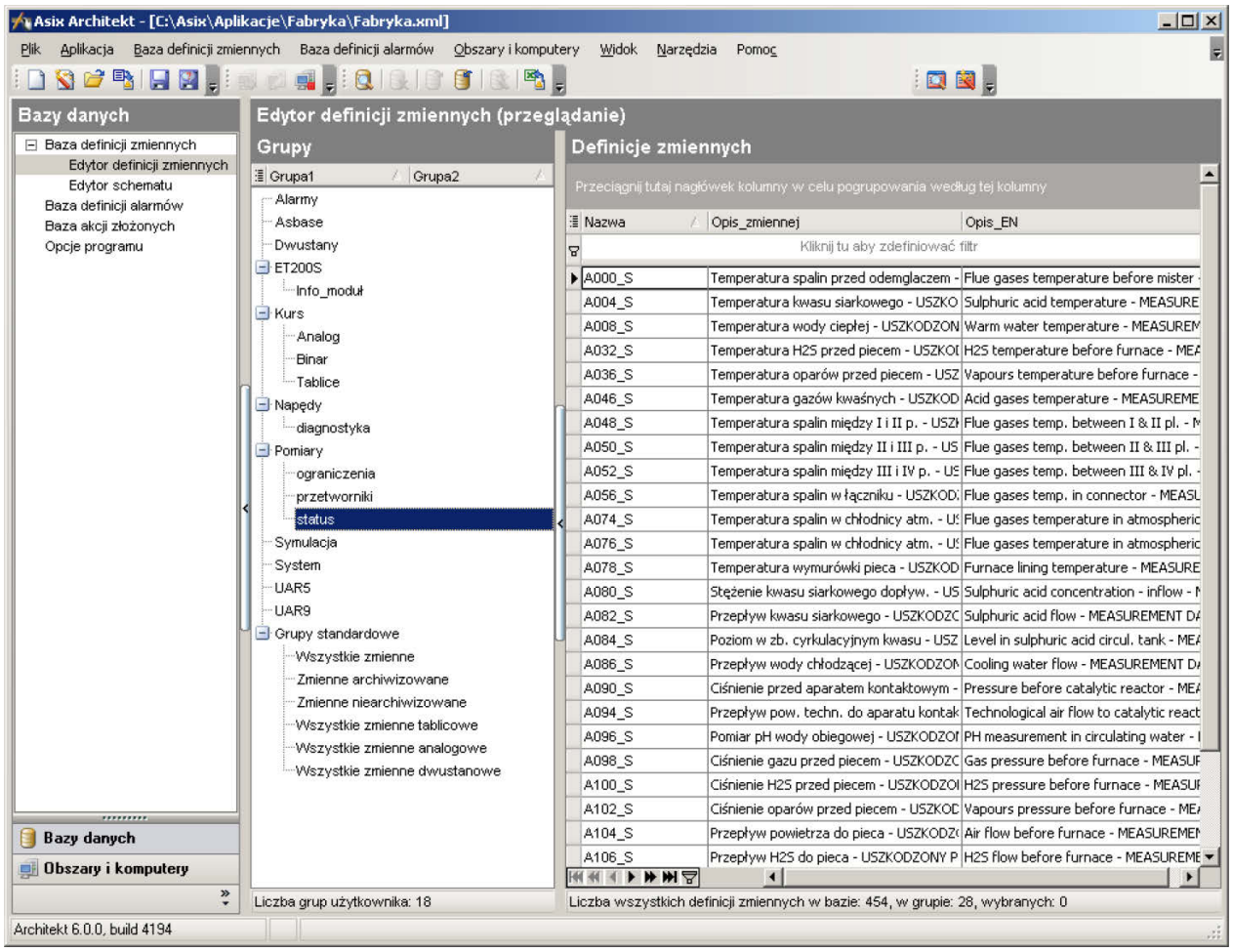
określonej grupy lub podgrupy. Patrz: *Rysunek: Okno listy zmiennych w pakiecie Asix4.*



Rys. Okno listy zmiennych w pakiecie Asix4.

Definicję ułożenia grup zmiennych zawierał plik *.ZEZ.

Aktualnie w trakcie konwersji bazy definicji zmiennych Paradox na format Jet lub MS SQL generowanych jest tyle atrybutów grupujących, ile wynika to ze struktury hierarchicznej zdefiniowanej w pliku *.ZEZ. Patrz: *Rysunek: Edytor definicji zmiennych - okno grupowania zmiennych.*



Rys. Edytor definicji zmiennych - okno grupowania zmiennych.

11. Otwórz skoroszyt, który do tej pory stanowił plik źródłowy do generowania/aktualizowania bazy definicji zmiennych Paradox i do wszystkich arkuszy z definicjami zmiennych dodaj kolumny atrybutów grupujących wygenerowanych w skoroszycie Schemat.xls

Dla każdej definicji zmiennych dodaj wartości atrybutów grupujących kopiując je z arkusza *Zmienne* pliku Schemat.xls, pamiętając o usunięciu starego atrybutu *Grupa*.

Tak zmodyfikowane arkusze z definicjami zmiennych mogą służyć jako źródło generowania/aktualizowania bazy definicji zmiennych Jet lub MS SQL.

13. Usuń ze skoroszytu Schemat.xls arkusz *Zmienne*.

Pozostałe zakładki zawierają przetłumaczone na język narodowy nazwy atrybutów zmiennych, deklaracje atrybutów zastępczych (dla aplikacji wielojęzycznych), definicje zestawów atrybutów, atrybutów grupujących - czyli wszystkie te informacje, które do tej pory w większości przechowywane były w pliku tekstowym Schemat.txt.

Plik Schemat.xls może odtąd stanowić źródło aktualizacji danych dostępnych w Architekcie z poziomu *Edytora schematu* modułu *Baza definicji zmiennych*.

14. Przejdź na tryb generatora:

Baza definicji zmiennych > zakładka *Typ* > opcja *Tryb Generator*

15. Na zakładce *Źródła danych* zaznacz i usuń wszystkie źródła danych.

16. Naciśnij przycisk Excel i wskaż plik *.xls zawierający definicje zmiennych.

17. W oknie *'Wybierz tabele'* zaznacz wszystkie arkusze z definicjami zmiennych i zatwierdź wybór przyciskiem *OK*.

18. Na zakładce *Źródła danych* ponownie naciśnij przycisk Excel i wskaż plik *.xls z danymi schematu bazy definicji zmiennych, czyli plik Schemat.xls.

19. W oknie *'Wybierz tabele'* zaznacz wszystkie arkusze i zatwierdź wybór przyciskiem *OK*.

20. Wybierz polecenie *Generuj* , a następnie otwórz bazę i sprawdź, czy baza została prawidłowo wygenerowana; wszelkie dane do aktualizacji bazy mogą być wprowadzane do plików *.xls, które przygotowane zostały w poprzednich krokach - każdorazowa zmiana w tych plikach wymaga ponownego wygenerowania bazy.

2.2.10. Konwertowanie bazy definicji zmiennych z plików tekstowych

Dla zachowania kompatybilności ze starszymi wersjami systemu Asix, Architekt pozwala również zaimportować dane definicji zmiennych znajdujące się w plikach tekstowych starszych aplikacji Asixa.

Jak przekonwertować bazę definicji zmiennych z plików tekstowych na bazę Jet lub MS SQL?:

1. Zdefiniuj typ bazy definicji zmiennych jako *Tryb Generator*.
2. Określ format oraz nazwę bazy definicji zmiennych, która ma zostać wygenerowana:

Baza definicji zmiennych > zakładka *Format* > format bazy danych ustawiony na: *Baza danych serwera Microsoft SQL Serwer* lub *Plik w formacie Mdb (Jet/Microsoft Access)*

W przypadku baz danych MS SQL wymagane jest określenie parametrów:
- *Nazwa serwera*;
- *Nazwa bazy*;
- *Katalog bazy*;
- *Tryb uwierzytelniania*:
  - *Windows*,
  - *Serwer MSQL* - wymaga określenia *Nazwy użytkownika* oraz *Hasła*.

3. Określ źródło danych:

Baza definicji zmiennych > zakładka *Źródła danych* > wskaż pliki tekstowe definicji zmiennych (przycisk *Asix - plik ini...*)

4. W oknie '*Wybierz tabele*' wskaż wszystkie pozycje i zatwierdź przyciskiem *OK*.

5. Wybierz polecenie *Generuj* .

6. Otwórz bazę i w razie potrzeby wprowadź zmiany do atrybutów zmiennych przy użyciu *Edytora definicji zmiennych*; zmodyfikuj w miarę potrzeby schemat bazy definicji zmiennych za pomocą *Edytora schematu.*

Edycja bazy możliwa jest po przejściu na *tryb Edytor:*
Baza definicji zmiennych > zakładka *Typ* > opcja *Tryb Edytor*.

2.2.10.1. Import definicji zmiennych z plików tekstowych - atrybuty grupujące

Wartość pierwszego atrybutu grupującego zmiennej, jest domyślnie taka sama jak nazwa pliku zmiennych bieżących, w którym zdefiniowana jest zmienna (nazwa pliku bez rozszerzenia i bez ścieżki). Aby jawnie określić wartości atrybutów grupujących zmiennej, należy umieścić w pliku definicji zmiennych bieżących, w postaci komentarza, linie deklarujące wartości atrybutów grupujących. Linia deklarująca wartości atrybutów grupujących jest linią o następującym formacie:

#Atrybuty : Atrybut1 = Wartość1 [,Atrybut2 = Wartość2...]

lub

#Attributes : Atrybut1 = Wartość1 [,Atrybut2 = Wartość2...]

Deklaracja rozpoczyna się od miejsca deklaracji i rozciąga się do następnej deklaracji lub do końca pliku.

2.2.11. Eksport bazy do arkusza Excel

Program Architekt umożliwia zapisanie wszystkich danych lub wybranych danych z bazy definicji zmiennych w arkuszu Excel. Maksymalna liczba eksportowanych wierszy to 32000.

Jak wyeksportować definicje zmiennych do arkusza Excel:

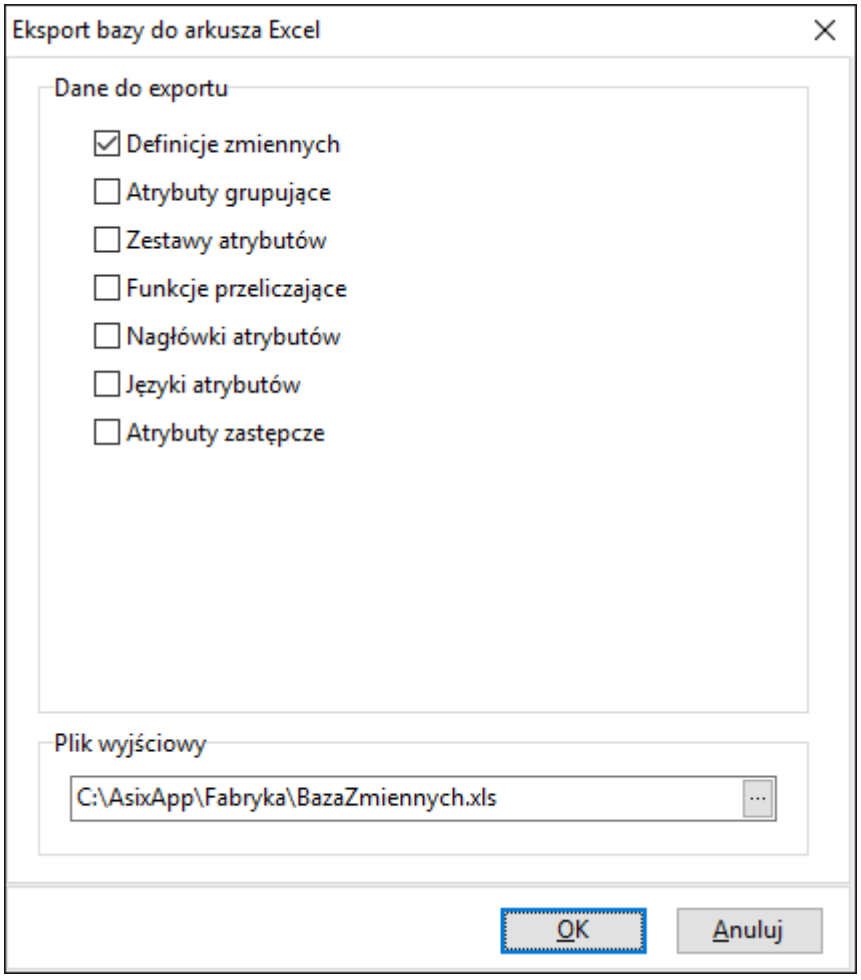
1. Otwórz bazę definicji zmiennych MSQL (lub Jet), podając jej lokalizację w:

Baza definicji zmiennych > zakładka *Format*

oraz klikając na komendę *Otwórz* z kategorii poleceń *Baza definicji zmiennych*.

2. Wybierz komendę *Eksportuj* .

3. W oknie *'Eksport bazy do arkusza Excel'* zaznacz dane, które mają być wyeksportowane oraz ścieżkę i nazwę docelowego pliku XLS (Patrz: *Rysunek: Okno 'Eksport bazy do arkusza Excel'.*).



Rys. Okno 'Eksport bazy do arkusza Excel'.

2.2.12. Dostęp z poziomu innych programów

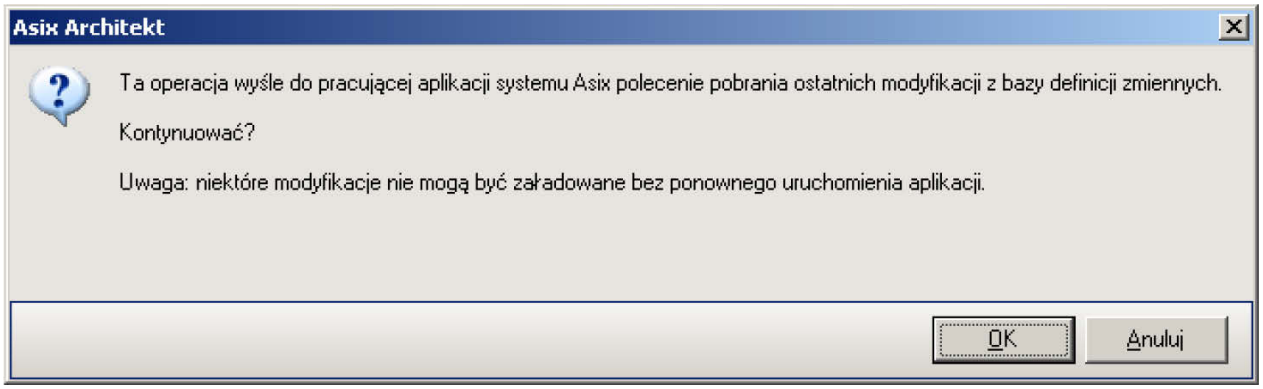
Baza definicji zmiennych jest bazą zarządzaną przez Microsoft SQL Server lub bazą Jet i dostęp do niej jest możliwy z każdego programu, który obsługuje ten system.

W dostępie do danych przechowywanych w bazie definicji zmiennych VarDef pośredniczy moduł AsixConnect za pomocą mechanizmu DDE, Automation lub .NET.

2.2.13. Uaktualnianie bazy definicji zmiennych on-line

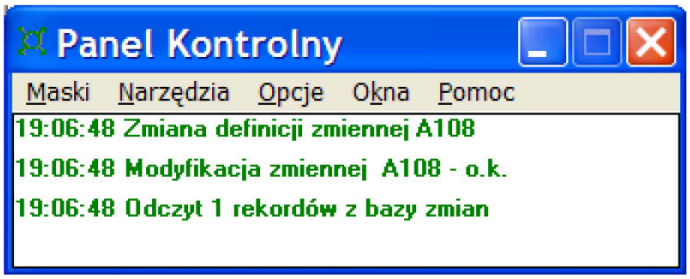
W przypadku, gdy niemożliwe jest zatrzymanie aplikacji i jej restart w celu zmodyfikowania bazy definicji zmiennych, możliwe jest przeprowadzenie aktualizacji bazy w trybie on-line. **Operacja taka jest możliwa jedynie, gdy zadeklarowany typ bazy definicji zmiennych to *Standardowa baza definicji zmiennych*, tryb "Edytor"**.

Zakładając, że używając edytora bazy definicji zmiennych dokonano jej modyfikacji (zmieniono definicje zmiennych, dodano nowe zmienne itp.), należy użyć przycisku z niebieskim trójkątem - *Aktualizuj on-line.*



Rysunek: Okno dialogowe akceptacji operacji pobrania przez Asix ostatnich modyfikacji z bazy definicji zmiennych.

Po potwierdzeniu chęci aktualizacji bazy on-line system **Asix** przeprowadzi ją, a w Panelu Kontrolnym pojawi się odpowiedni komunikat, podający liczbę zmodyfikowanych rekordów bazy oraz nazwy zmiennych:



Rysunek: Panel Kontrolny z informacja o odczycie przez Asixa aktualizacji z bazy definicji zmiennych.

UWAGA: W sytuacji, gdy definicje zmiennych przechowywane są w arkuszach kalkulacyjnych (w nich wprowadzane były aktualizacje tych definicji), na podstawie których generowana jest baza definicji zmiennych – przejście na tryb aktualizacji bazy za pomocą edytora bazy definicji zmiennych spowoduje, że źródłowe arkusze kalkulacyjne nie będą uwzględniały zmian wprowadzonych za pomocą edytora.

2.2.14. Zmienne systemowe

Szczególny rodzaj zmiennych stanowią **zmienne systemowe**:
__USERID__ - zapewnia dostęp do identyfikatora aktualnie zalogowanego użytkownika;
__USERNAME__ - zapewnia dostęp do nazwy aktualnie zalogowanego użytkownika;
__USERLEVEL__ - zapewnia dostęp do aktualnie obowiązującego poziomu uprawnień;
__AGR_IdGrupy__ - pozwala na wyświetlenie stanu aktywności alarmów należących do wspólnej grupy.

2.2.15. Mirroring

Opcjonalnie baza definicji zmiennych może obsługiwać mirroring. Aby użyć tę funkcje, należy podać nazwę serwera MS SQL, zawierającego mirror bazy definicji zmiennych.
Mirroring bazy definicji zmiennych znacząco zwiększa dostępność bazy.
Mirroring definiowany jest na etapie deklaracji formatu bazy definicji zmiennych:
Baza definicji zmiennych > zakładka *Format*



Konfiguracja mechanizmu mirroring dla bazy definicji zmiennych:

1. Skonfiguruj usługi serwera Microsoft SQL na komputerze głównym tak, żeby pracowały pod tym samym użytkownikiem (lokalnym lub domenowym). Dodaj użytkownika do listy użytkowników serwera SQL i nadaj mu rolę 'sysadmin'.
2. Na obu serwerach otwórz porty 1433 i 5022.
3. W programie Architekt w opcjach formatu bazy wprowadź nazwę serwera mirror i wygeneruj bazę definicji zmiennych na serwerze głównym.
4. Wykonaj backup bazy definicji zmiennych.
5. W programie Architekt za pomocą polecenia *Baza definicji zmiennych* > *Mirror* > *Kopiowanie użytkownika '_asix_internal'* skopiuj użytkownika '_asix_internal' z serwera głównego na mrror.
6. Odtwórz backup bazy definicji zmiennych na serwerze mirror w trybie RESTORE WITH NORECOVERY.
7. Za pomocą Microsoft SQL Server Management Studio włacz mirroring bazy (menu lokalne bazy > *Tasks* > *Mirror*).

2.3. Dodanie definicji alarmów SQL do bazy definicji zmiennych

Generowanie bazy definicji alarmów na podstawie alarmów SQL dotyczy przypadku, kiedy informacje o wszystkich zdarzeniach alarmowych przechowywane są w bazie SQL. Generator bazy definicji alarmów SQL w programie Architekt na podstawie archiwum zdarzeń przechowywanych w bazie SQL generuje bazę definicji alarmów dołączaną do bazy definicji zmiennych.



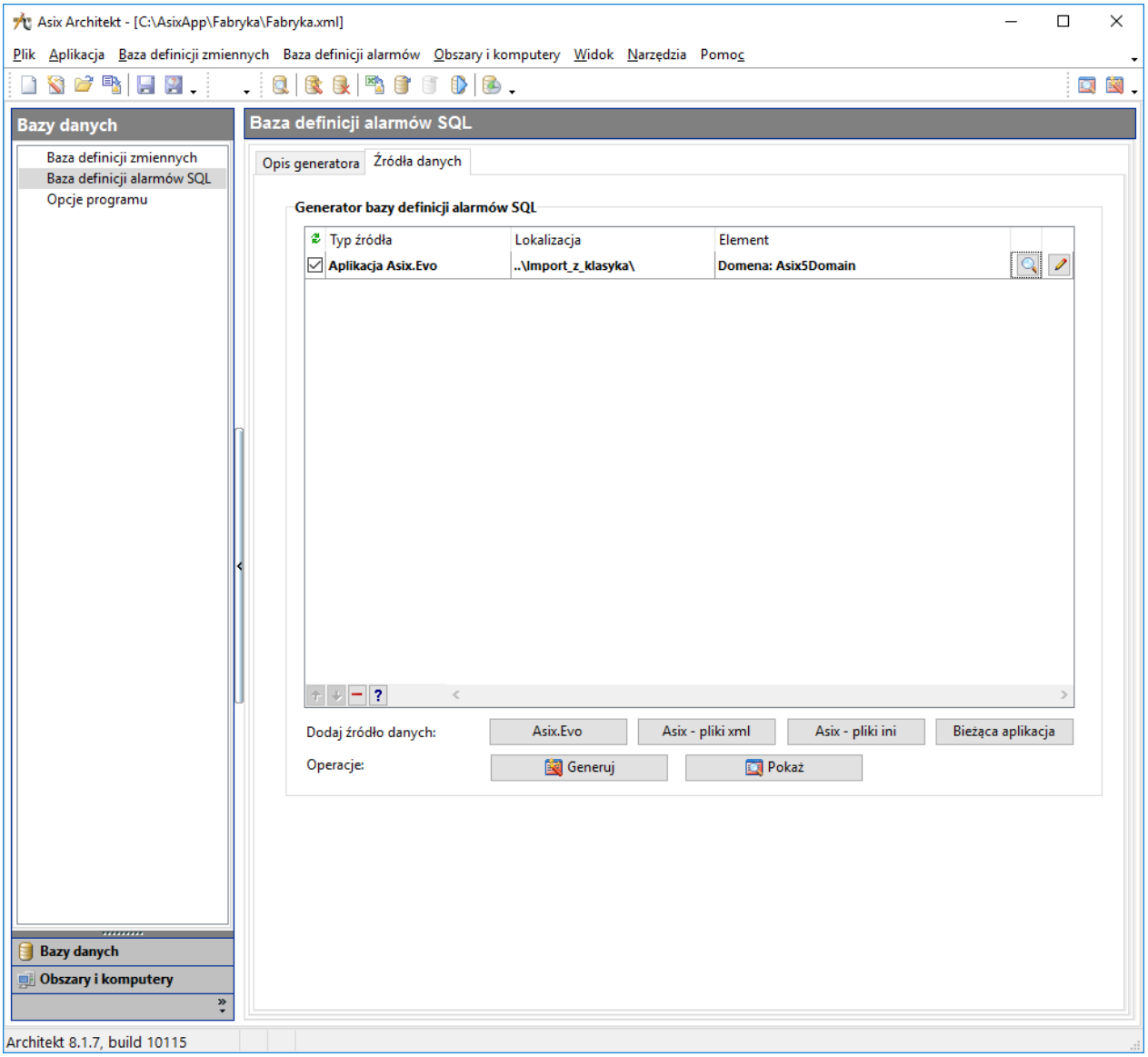
Aby dołączyć definicje alarmów SQL do bazy definicji zmiennych:

1. Określ źródło definicji alarmów: (źródłem jest aplikacja Asix.Evo, możliwy jest też odczyt plików XML ze starszych klasycznych wersji aplikacji Asix):

Architekt > *Bazy danych* > Baza definicji alarmów SQL > zakładka *Źródła danych* > opcja: ***Dodaj źródło danych***

5. Wygeneruj bazę definicji alarmów (operacja ta dołącza definicje alarmów i grup alarmów do istniejącej bazy definicji zmiennych):

Program Architekt > *Bazy danych* > *Baza definicji alarmów* > *zakładka Źródła danych* > opcja ***Operacje***: ***Generuj***.



Rys. Architekt - Generator bazy definicji alarmów SQL.

2.4. Utworzenie pliku konfiguracyjnego Serwera Danych

Utworzenie pliku konfiguracyjnego XML Serwera Danych stanowi pierwszy krok poprzedzający etap konfigurowania bazy definicji zmiennych.

Parametry konfigurujące pracę aplikacji oraz deklarujące bazę danych, w oparciu o którą aplikacja pracuje zapisywane są do **jednego wspólnego pliku XML** (ang. *Extensible Markup Language*), umieszczanego w kartotece głównej projektowanej aplikacji.

Tworzenie pliku konfiguracyjnego:

Wariant 1 - bez kreatora

1. Z menu głównego z pozycji ***Plik*** wybierz komendę ***Nowa aplikacja.***
2. Przejdź na widok pokazywania tylko opcji Asix.Evo: menu ***Widok*** > polecenie ***Pokaz tylko opcje Asix.Evo***
3. W lewym bloku okna Architekta kliknij na przycisk *Obszary i komputery*.

4. Na liście komputerów wprowadź następujące zmiany:

   - dodaj tyle pozycji komputerów i obszarów, ile wynika z projektu aplikacji - pamiętając, że obszar wspólny zawiera parametry, które w wielostanowiskowej konfiguracji systemu Asix są wspólne dla wszystkich stanowisk, na których uruchamiana jest aplikacja; natomiast konfiguracja na poziomie *Komputera* pozwala zindywidualizować ustawienia w zależności od typu komputera (inne dla stacji operatorskiej, a inne dla dedykowanego serwera danych w sieci zakładowej). Ustawienia na poziomie *Komputera* są ważniejsze od ustawień na poziomie *Obszaru*.

5. Zapisz plik w katalogu projektowanej aplikacji.

Wariant 2 - użycie kreatora

1. Z menu głównego z pozycji ***Plik*** wybierz komendę ***Nowa aplikacja - kreator*** .

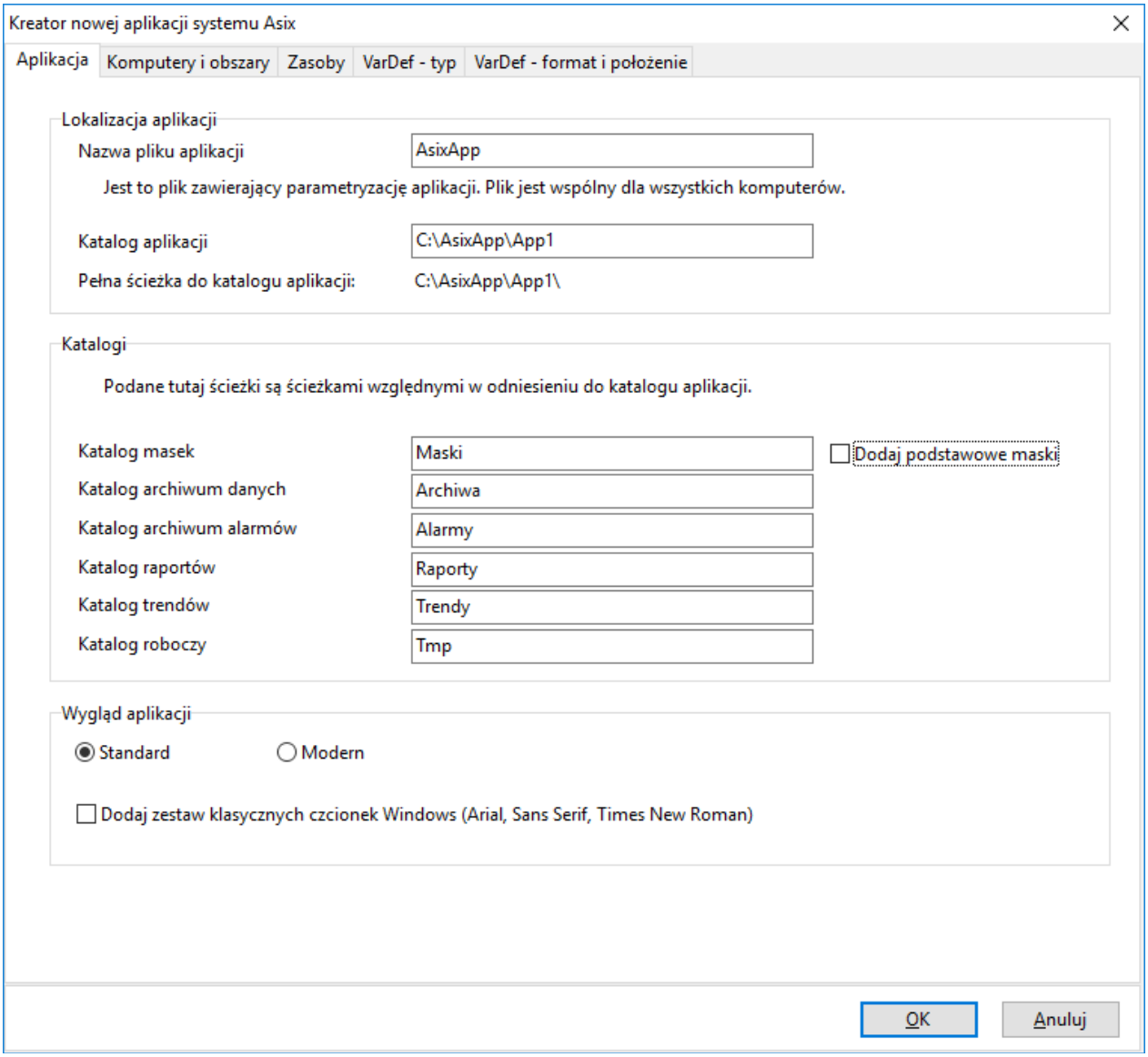
2. W oknie kreatora nowej aplikacji - zakładce ***Aplikacja*** ogranicz sie do określenia tylko tych opcji, które są istotne z punktu widzenia Serwera Danych:
- *Nazwa pliku aplikacji*,
- *Katalog aplikacji*;

oraz katalogi, których ścieżki są ścieżkami względnymi w odniesieniu do katalogu aplikacji:

- *Katalog archiwum danych,*
- *Katalog roboczy;*

(dla pozostałych katalogów pozostawić nazwy domyślne, ponieważ pola nie mogą pozostać puste)

3. Wprowadzone zmiany zatwierdź przyciskiem *OK*.



Rys. Kreator pliku konfiguracyjnego XML Serwera Danych - zakładka 'Aplikacja'.

3. W oknie kreatora nowej aplikacji - zakładce ***Komputery i obszary*** określ następujące dane:

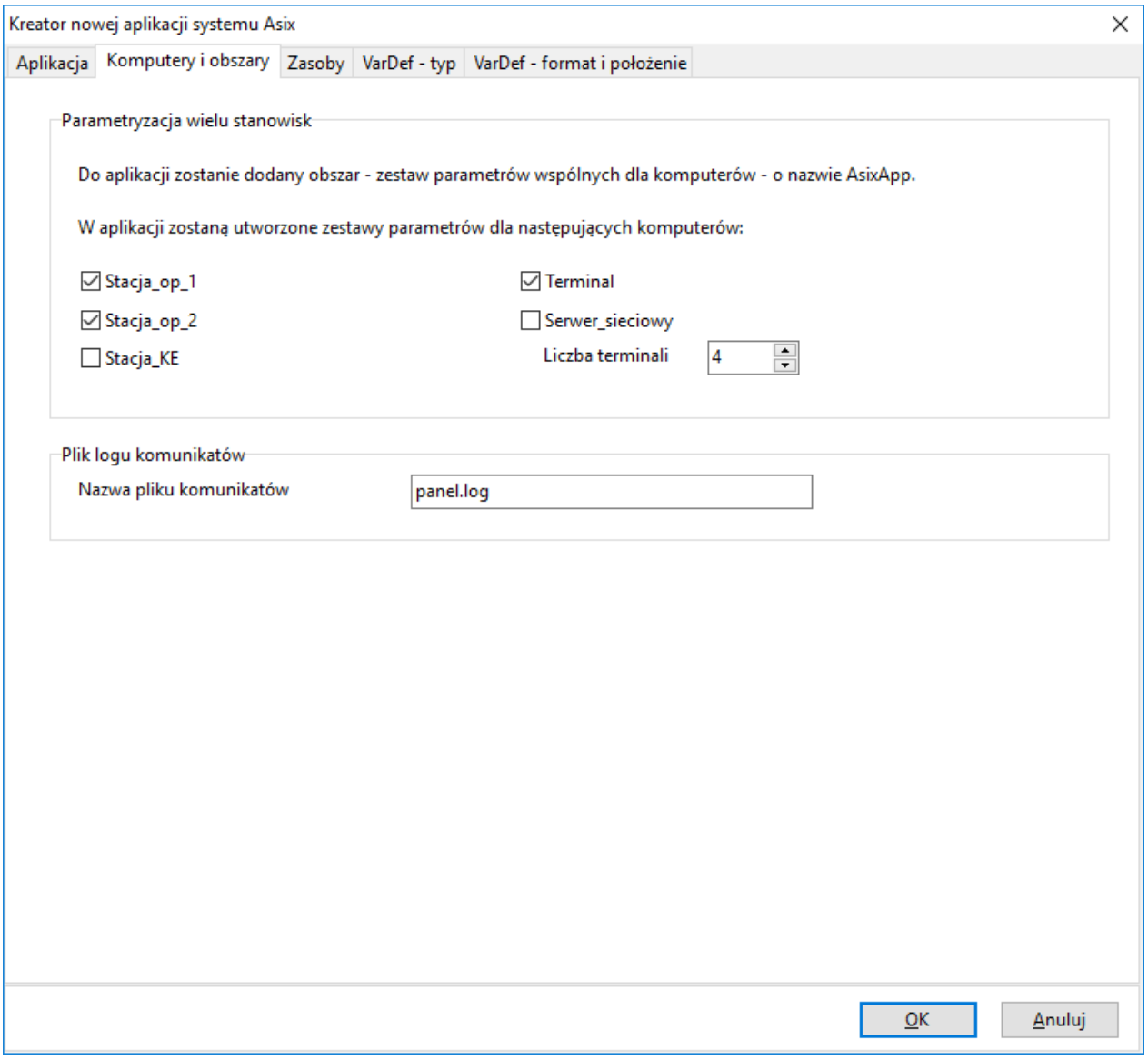
- w bloku ***Parametryzacja wielu stanowisk*** należy podać rodzaje komputerów, dla których zostaną utworzone zestawy parametrów:
  - *Stacja operatorska*,
  - *Stacja KE*,
  - *Terminal*,
  - *Serwer sieciowy*.

Nazwy komputerów tworzonych przez kreator dostosowano do konwencji nazw sieciowych – nie zawierają spacji i mają maksymalnie 15 znaków.

Określenie w tym miejscu wszystkich rodzajów komputerów, dla których kreator automatycznie utworzy zestawy parametrów aplikacji nie jest wymagane - projektant w każdej chwili może zmodyfikować ilość pozycji komputerów/obszarów wyświetlanych w bloku *Obszary i komputery*.
Poza tym wybór określonego rodzaju komputera nie ma wpływu na ilość i rodzaj parametrów, które znajdą się w zestawie. Dla każdego *Obszaru* i *Komputera* udostępniane są wszystkie możliwe do zadeklarowania parametry.

a także:
- *Nazwa pliku komunikatów* - plik ten zostanie utworzony w katalogu roboczym.

Wprowadzone zmiany zatwierdź przyciskiem *OK*.

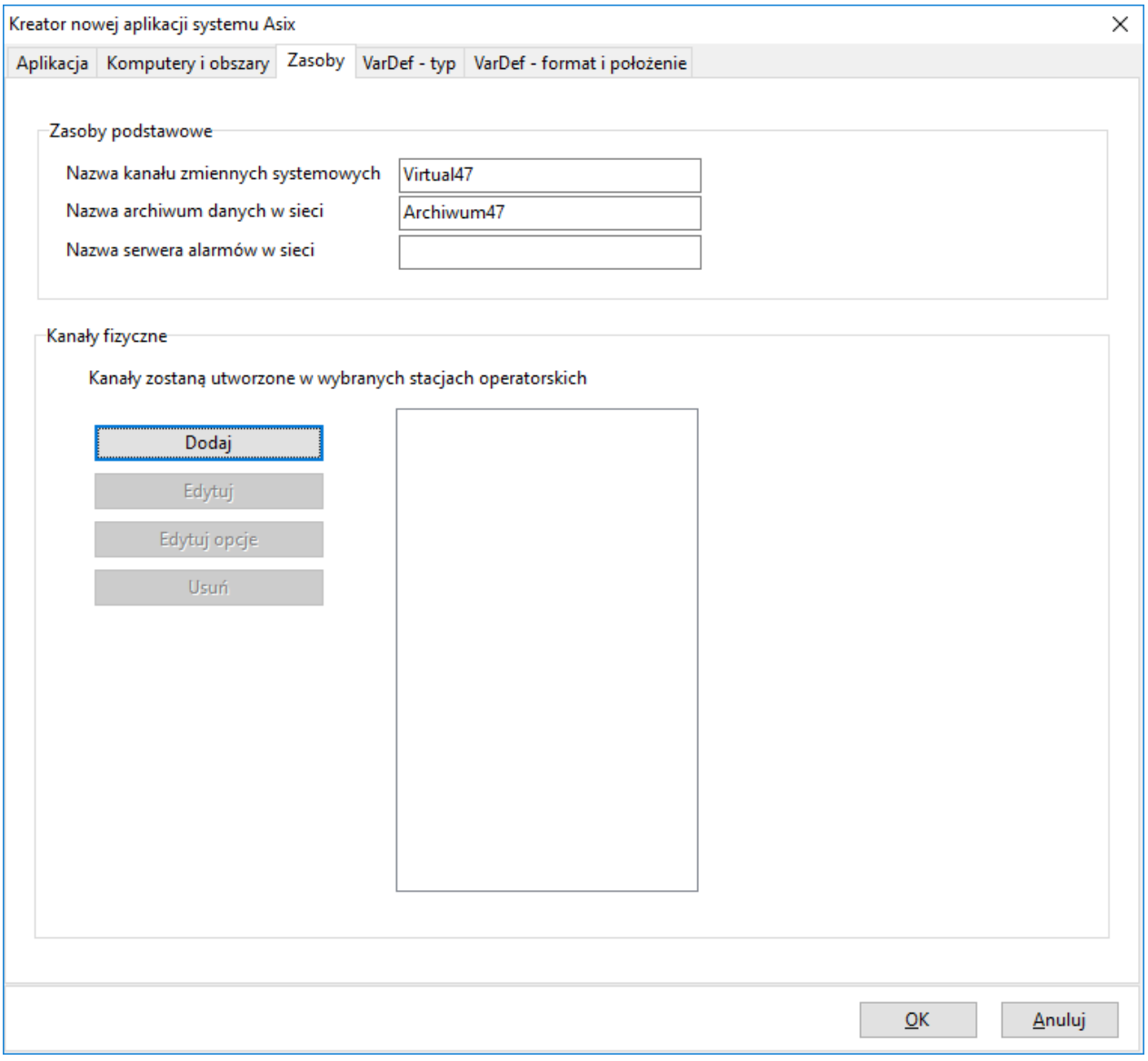


Rys. Kreator pliku konfiguracyjnego XML Serwera Danych - zakładka 'Komputery i obszary'.

4. W oknie kreatora nowej aplikacji - zakładce ***Zasoby*** określ następujące dane:

- w bloku ***Zasoby podstawowe*** należy zadeklarować następujące elementy:
- *Nazwa kanału zmiennych systemowych,*
- *Nazwa archiwum danych w sieci,*

- w bloku ***Kanały fizyczne*** należy zadeklarować kanały fizyczne, które zostaną utworzone w zadeklarowanych stacjach operatorskich.

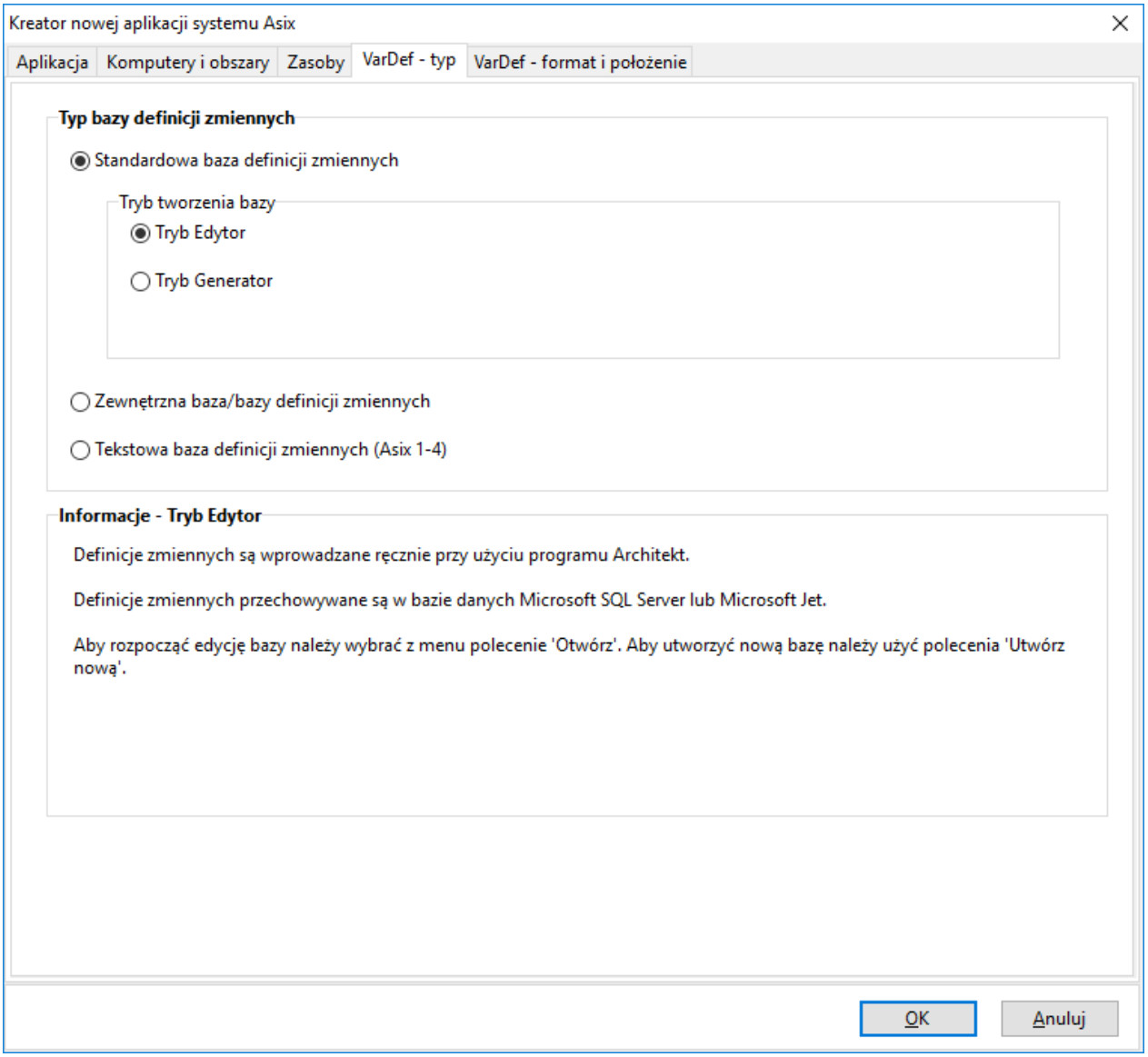


Rys. Kreator pliku konfiguracyjnego XML Serwera Danych - zakładka 'Komputery i obszary'.

5. W oknie kreatora nowej aplikacji - zakładce ***VarDef - typ*** określ typ bazy definicji zmiennych.

W przypadku zadeklarowania opcji *Tryb Generator* dopiero po utworzeniu pliku konfiguracyjnego Serwera Danych będzie można zadeklarować pliki źródłowe z definicjami zmiennych i wygenerować na ich podstawie bazę.

Wprowadzone zmiany zatwierdź przyciskiem *OK*.

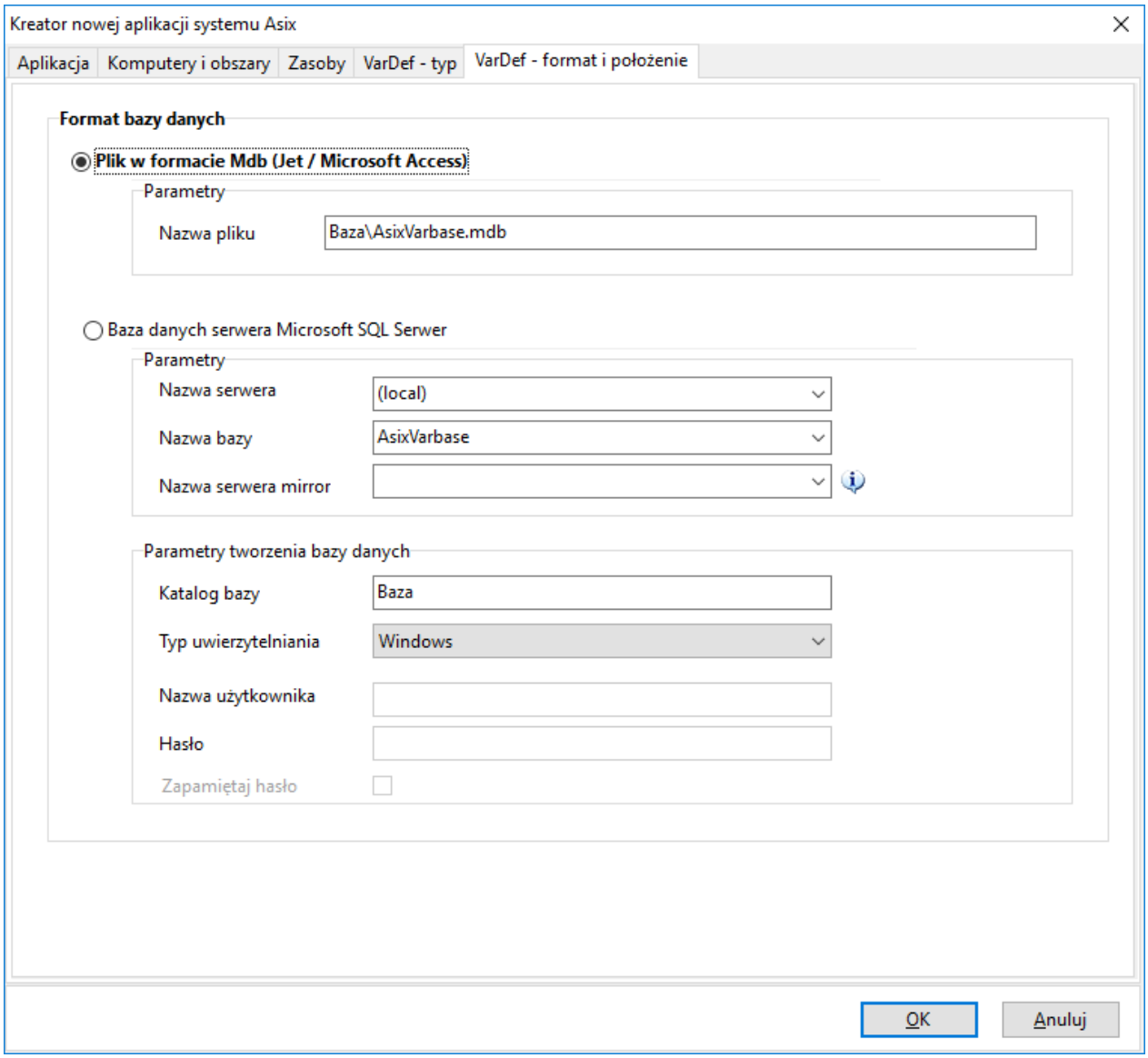


Rys. Kreator pliku konfiguracyjnego XML Serwera Danych - zakładka 'VarDef - typ'.

W przypadku wybrania opcji Tryb Generator i formatu bazy - *Baza danych serwera Microsoft SQL Serwer 2000/2005/2008* -
przed przystąpieniem do generowania pliku aplikacji za pomocą kreatora, łącznie z generowaniem bazy definicji zmiennych w formacie SQL, na komputerze powinien zostać zainstalowany serwer SQL.

6. W oknie kreatora nowej aplikacji - zakładce ***VarDef - format i położenie*** określ format oraz nazwę bazy definicji zmiennych.

W przypadku baz danych MS SQL wymagane jest określenie parametrów:
- *Nazwa serwera*;
- *Nazwa bazy*;
- *Nazwa serwera mirror*;
- *Katalog bazy*;
- *Tryb uwierzytelniania*:
  - *Windows*,
  - *Serwer MSQL* - wymaga określenia *Nazwy użytkownika* oraz *Hasła*.



Rys. Kreator pliku konfiguracyjnego XML Serwera Danych - zakładka 'VarDef - format i położenie'.

Kreator aplikacji tworzy domyślnych użytkowników: Administratora oraz Operatora z uprawnieniami 1 poziomu. Automatycznie tworzony jest również kanał NONE oraz 2 zmienne w bazie definicji zmiennych: __USERNAME__ i __USERLEVEL__. Kreator aplikacji zakłada również katalog plików trendów i ustawia ten katalog jako roboczy dla programu AsTrend.

2.4.1. Kreator w trybie importu bazy z serwera OPC / OPC UA

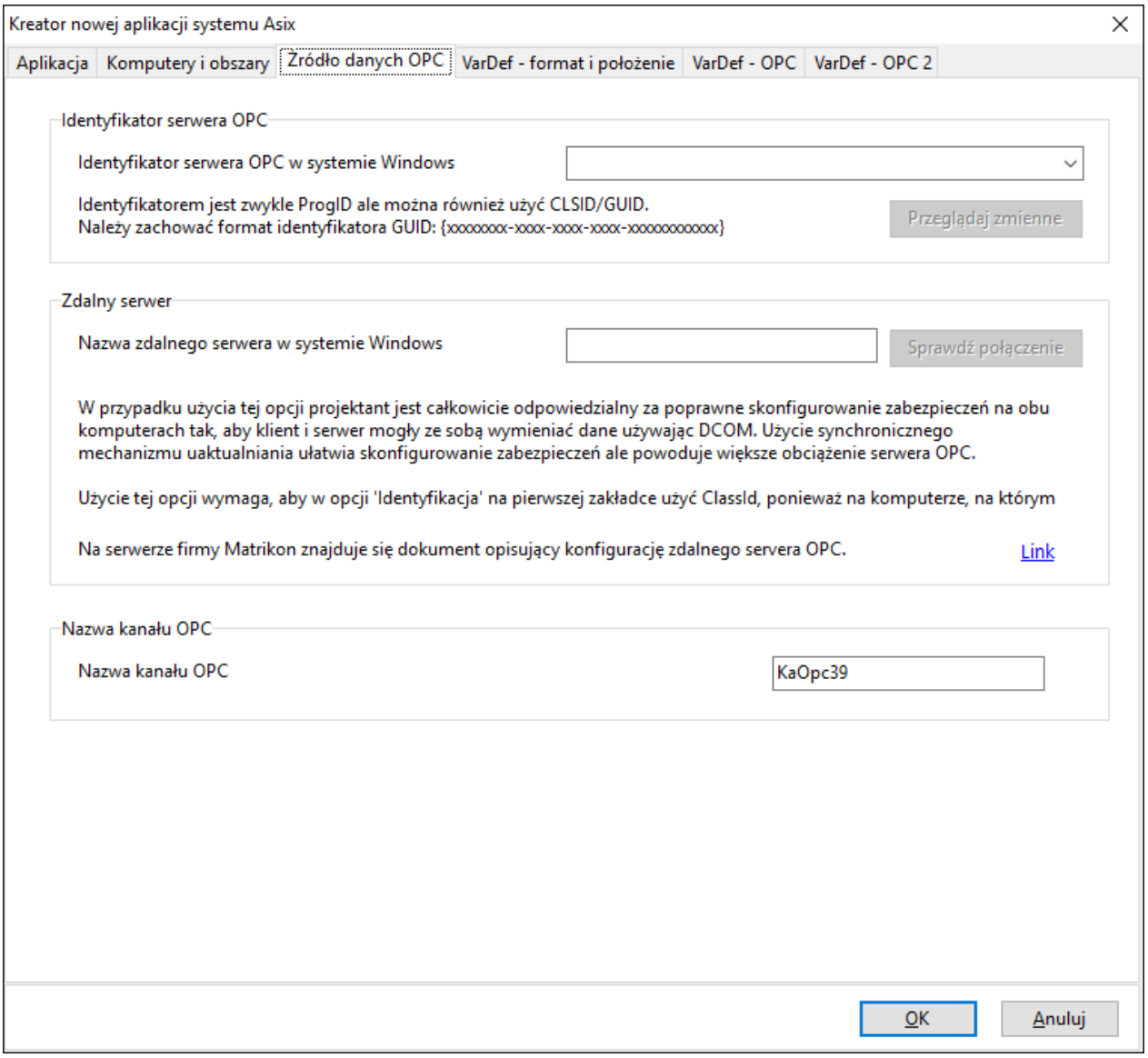
Utworzenie pliku konfiguracyjnego XML w trybie importu bazy z serwera OPC zakłada zaimportowanie definicji zmiennych z bazy serwera OPC do formatu bazy systemu Asix. W tym celu należy użyć kreatora aplikacji wywoływanego poleceniem ***Nowa aplikacja - kreator OPC*** / ***Nowa aplikacja - kreator OPC UA*** z menu ***Plik***.

Okno kreatora obejmuje ustawienia podstawowych parametrów aplikacji takich jak: lokalizacja aplikacji, struktura katalogów, plik logu, struktura komputerów/obszarów, format i położenie bazy definicji zmiennych. Dodatkowe informacje dotyczą źródła danych OPC oraz ustawień związanych z importem zmiennych z serwera OPC.

Zakładki ***Aplikacja***, ***Komputery i obszary***, ***VarDef - format i położenie*** zawierają parametry identyczne jak w przypadku standardowego kreatora aplikacji - patrz: Serwer Danych > 2. Parametryzacja Serwera Danych > 2.4. Utworzenie pliku konfiguracyjnego Serwera Danych.

Zakładka ***Źródło danych OPC*** wymaga podania następujących parametrów:

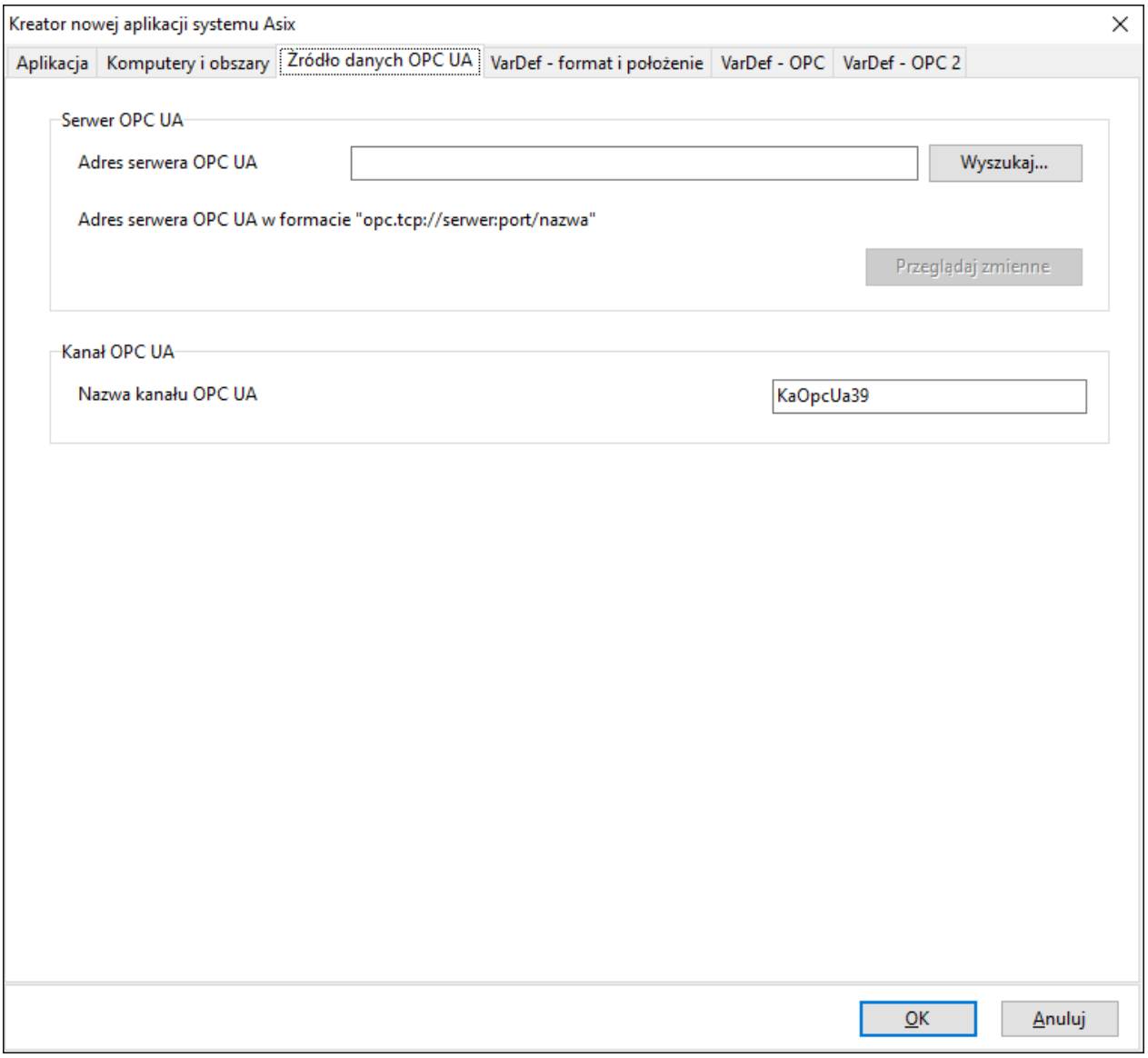
- ***Identyfikator serwera OPC w systemie Windows***;
- ***Nazwa zdalnego serwera w systemie Windows***;
- ***Nazwa kanału OPC***.



Rysunek: Kreator aplikacji - Źródło danych OPC.

Zakładka ***Źródło danych OPC UA*** wymaga podania następujących parametrów:

- ***Adres serwera OPC UA***;
- ***Nazwa kanału OPC UA***.

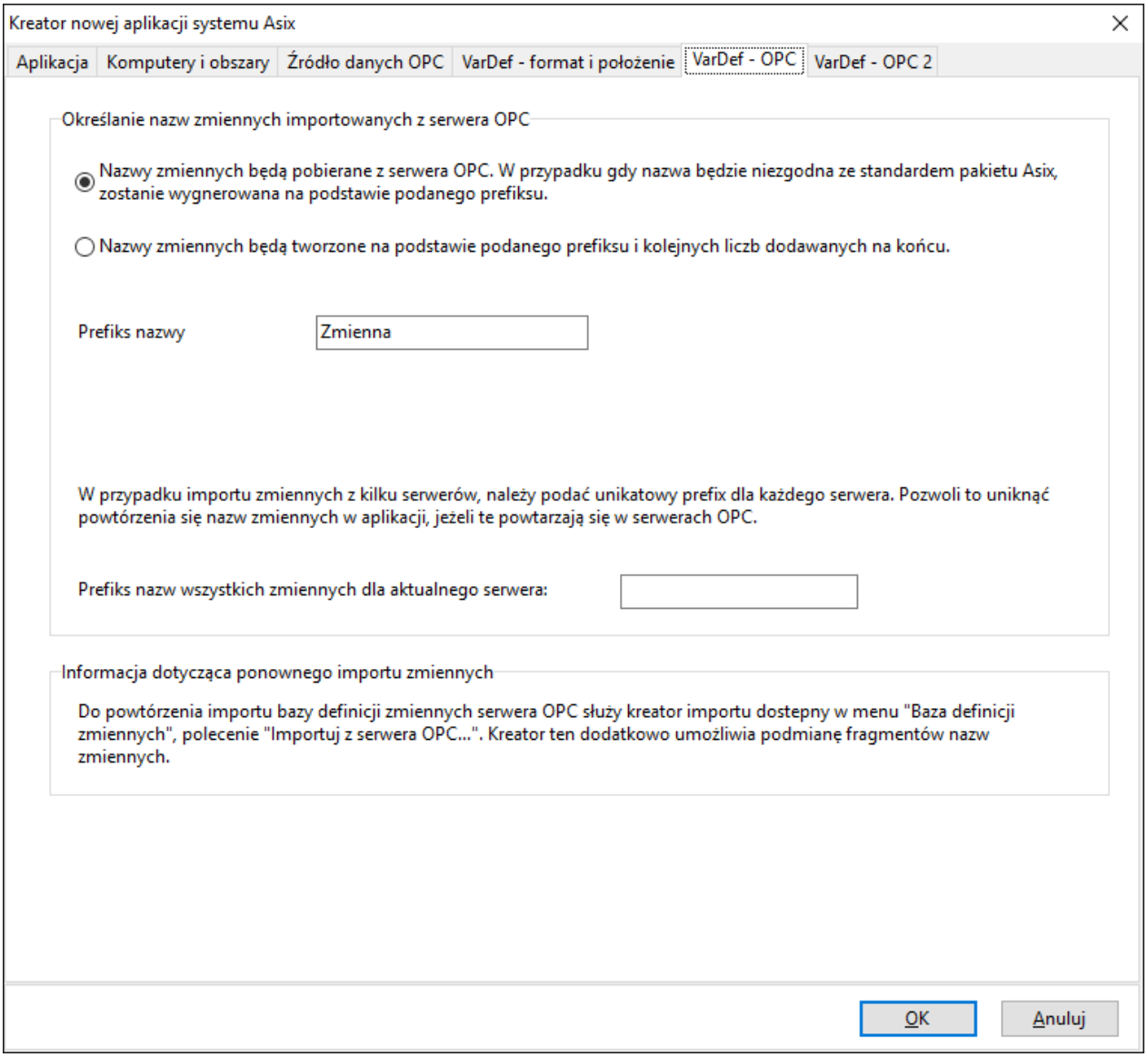


Rysunek: Kreator aplikacji - Źródło danych OPC UA.

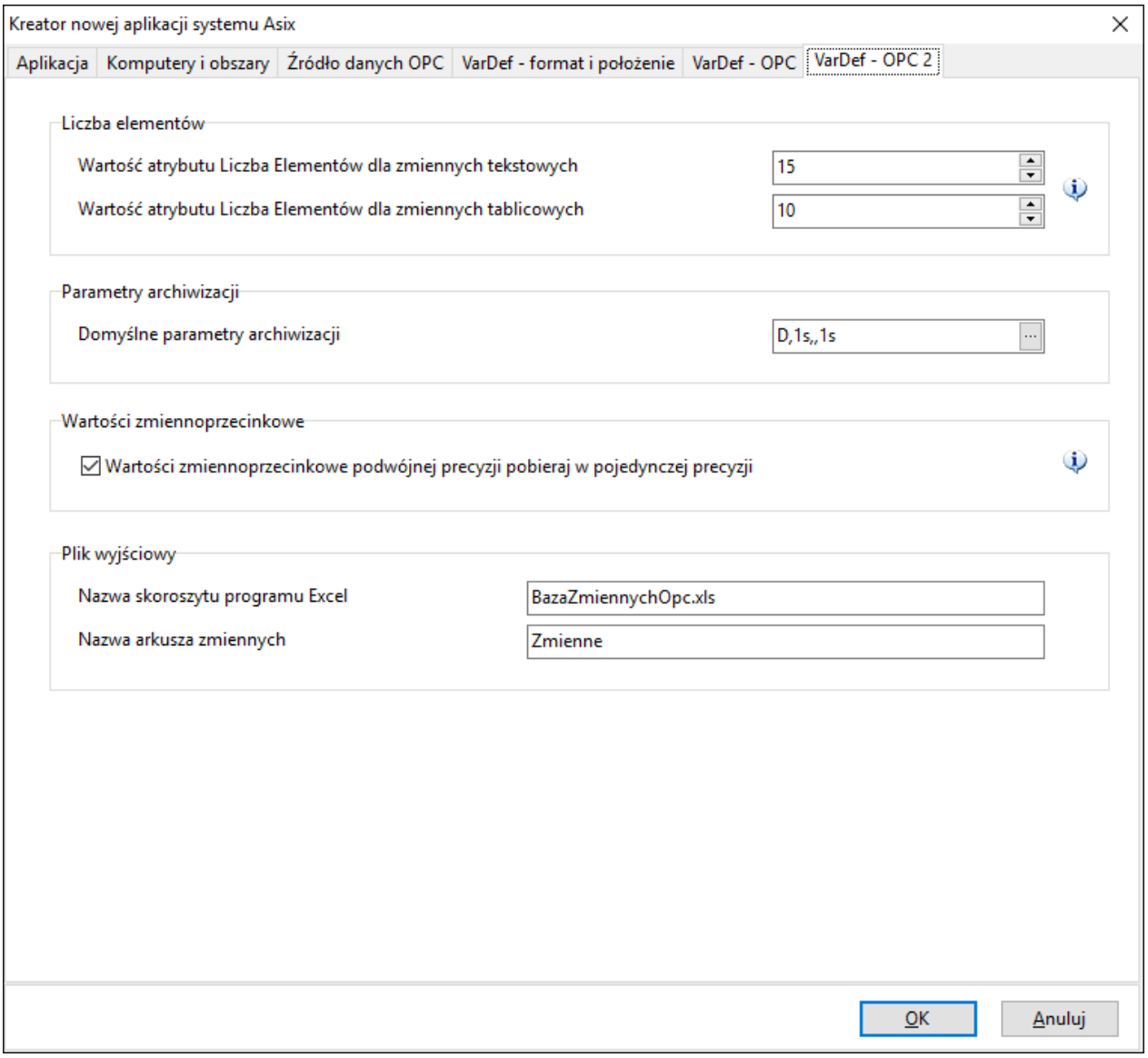
Zakładki ***VarDef - OPC*** / ***VarDef - OPC 2*** obejmuje deklarację następujących parametrów:

- ***Określenie nazw zmiennych importowanych z serwera OPC*** – nazwy pobierane z Sewera OPC; w przypadku niezgodności ze standardem **Asix**a nazwy generowane są na podstawie zadeklarowanego prefiksu i kolejnych liczb dodawanych na końcu.
- ***Liczba Elementów*** – wartość atrybutu dla zmiennych skalarnych ustawiana na 1. Dla zmiennych tekstowych / tablicowych należy wprowadzić ręcznie długość tekstu/rozmiar tablicy, ponieważ serwer OPC nie udostępnia tych informacji.
- ***Plik wyjściowy*** – zostanie wygenerowany skoroszyt o zadeklarowanej nazwie z arkuszem definicji zmiennych pobranych z serwera OPC; plik zostanie automatycznie zadeklarowany w źródle danych zawierającym definicje zmiennych dla aplikacji (Architekt > ***Bazy danych*** > ***Baza definicji zmiennych*** > ***Źródła danych***).
- ***Wartości zmiennoprzecinkowe*** – wartości zmiennoprzecinkowe podwójnej precyzji mogą być archiwizowane z pełną dokładnością tylko w archiwum typu B. Pozostałe archiwa zapisują liczby zmiennoprzecinkowe zawsze z pojedynczą precyzją.

- Parametry archiwizacji – domyślne wartości parametrów archiwizacji zmiennych procesowych.



Rysunek: Kreator aplikacji - VarDef - OPC (zakładka 1).

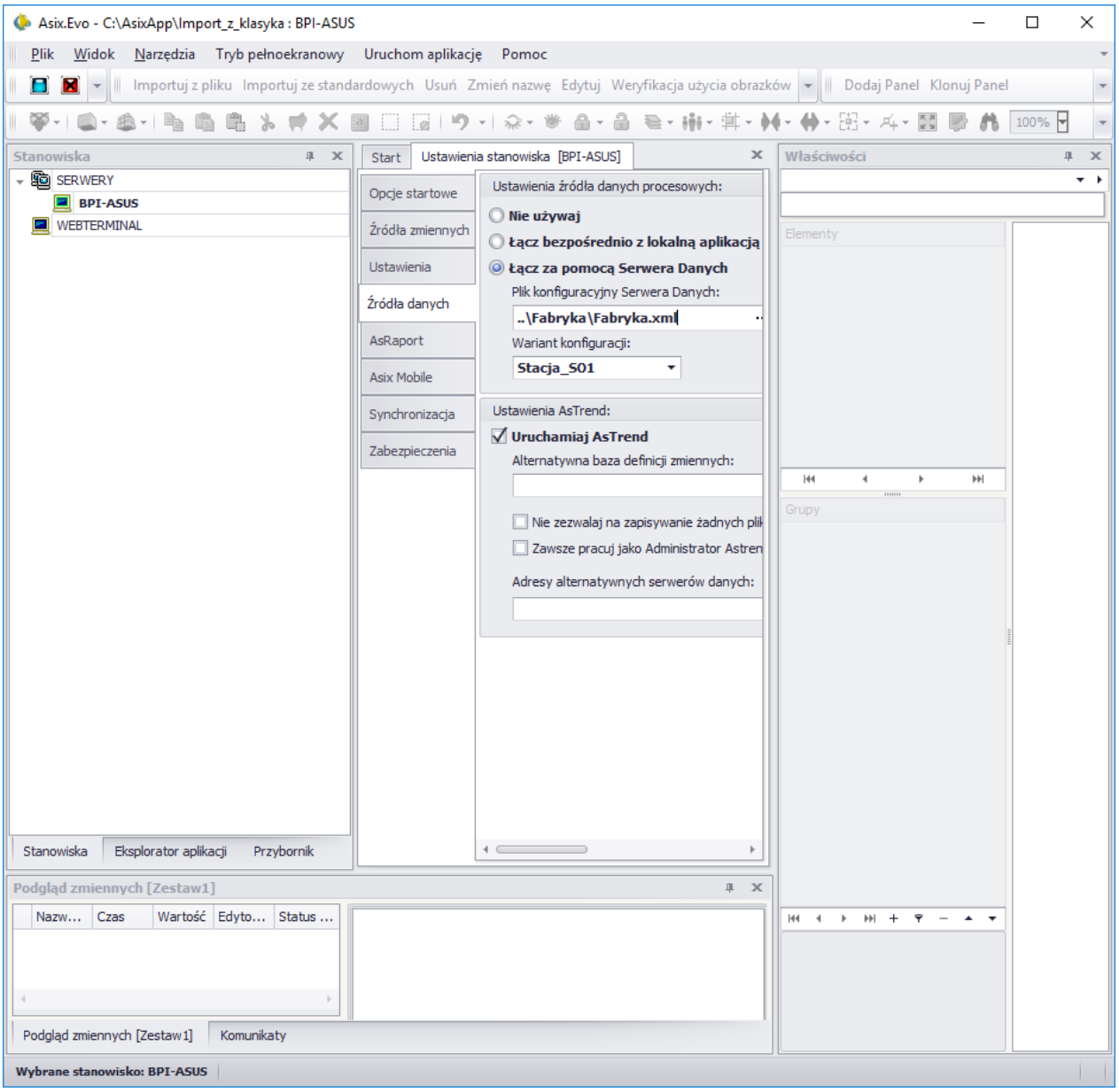


Rysunek: Kreator aplikacji - VarDef - OPC (zakładka 2).

2.5. Zadeklarowanie pliku konfiguracyjnego Serwera Danych

Plik konfiguracyjny serwera danych należy zadeklarować w ustawieniach źródła danych procesowych wybranego stanowiska:

program Asix.Evo.exe > eksplorator aplikacji dla danego stanowiska > ***Ustawienia stanowiska*** > ***Źródła danych*** > opcja ***Łącz za pomocą Serwera Danych***/***Plik konfiguracyjny Serwera Danych***



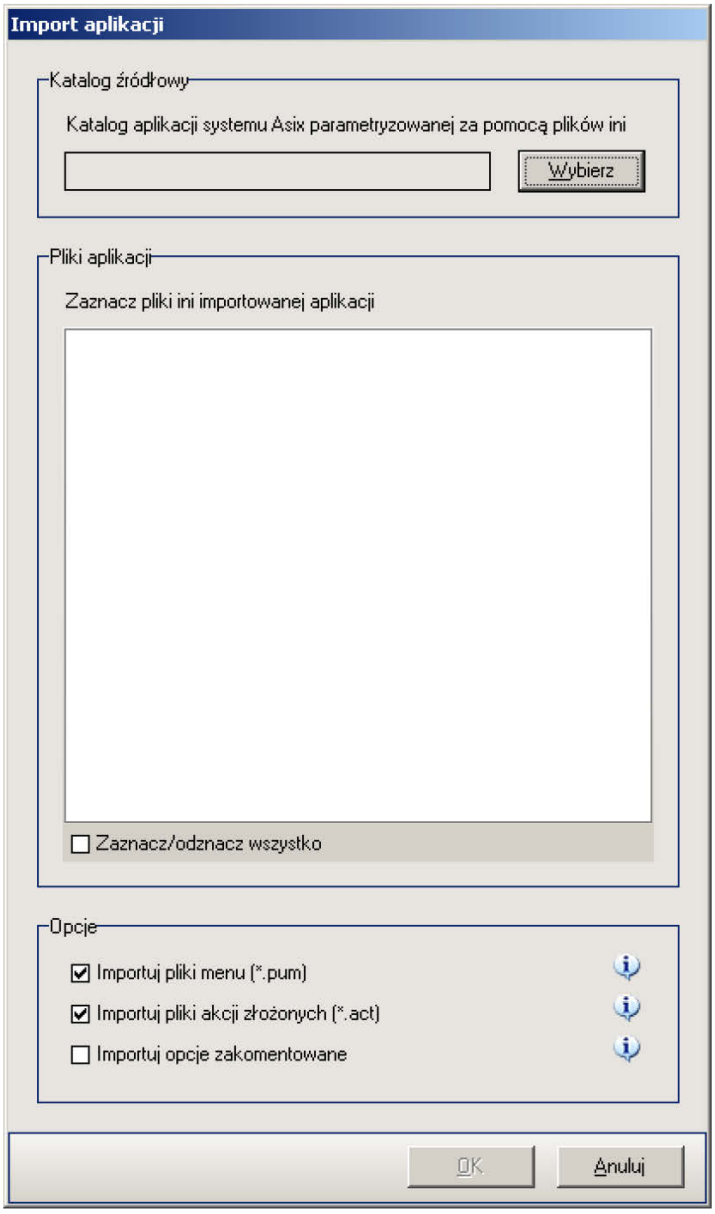
Rys. Zadeklarowanie pliku Serwera Danych w parametrach aplikacji Asix wersji Evo.

2.6. Import plików konfiguracyjnych *.INI

Dla aplikacji tworzonych w starszych wersjach pakietu Asix istnieje możliwość przekonwertowania plików aplikacji *.INI na format pliku konfiguracyjnego *.XML.

Służy do tego polecenie:

Plik > Nowa aplikacja - importer , które powoduje uruchomienie okna z parametrami importu:



*Rys. Okno importu aplikacji opartej na plikach *.INI.*

Okno ***'Import aplikacji'*** pozwala określić:

- ***Katalog źródłowy*** - parametr wskazuje katalog aplikacji systemu Asix sparametryzowanej za pomocą plików *.ini;
- ***Pliki aplikacji -*** pole to wskazuje pliki *.ini importowanej aplikacji;

Pozostałe opcje są nieistotne z punktu widzenia parametryzacji Serwera Danych aplikacji Asix w wersji Evo.

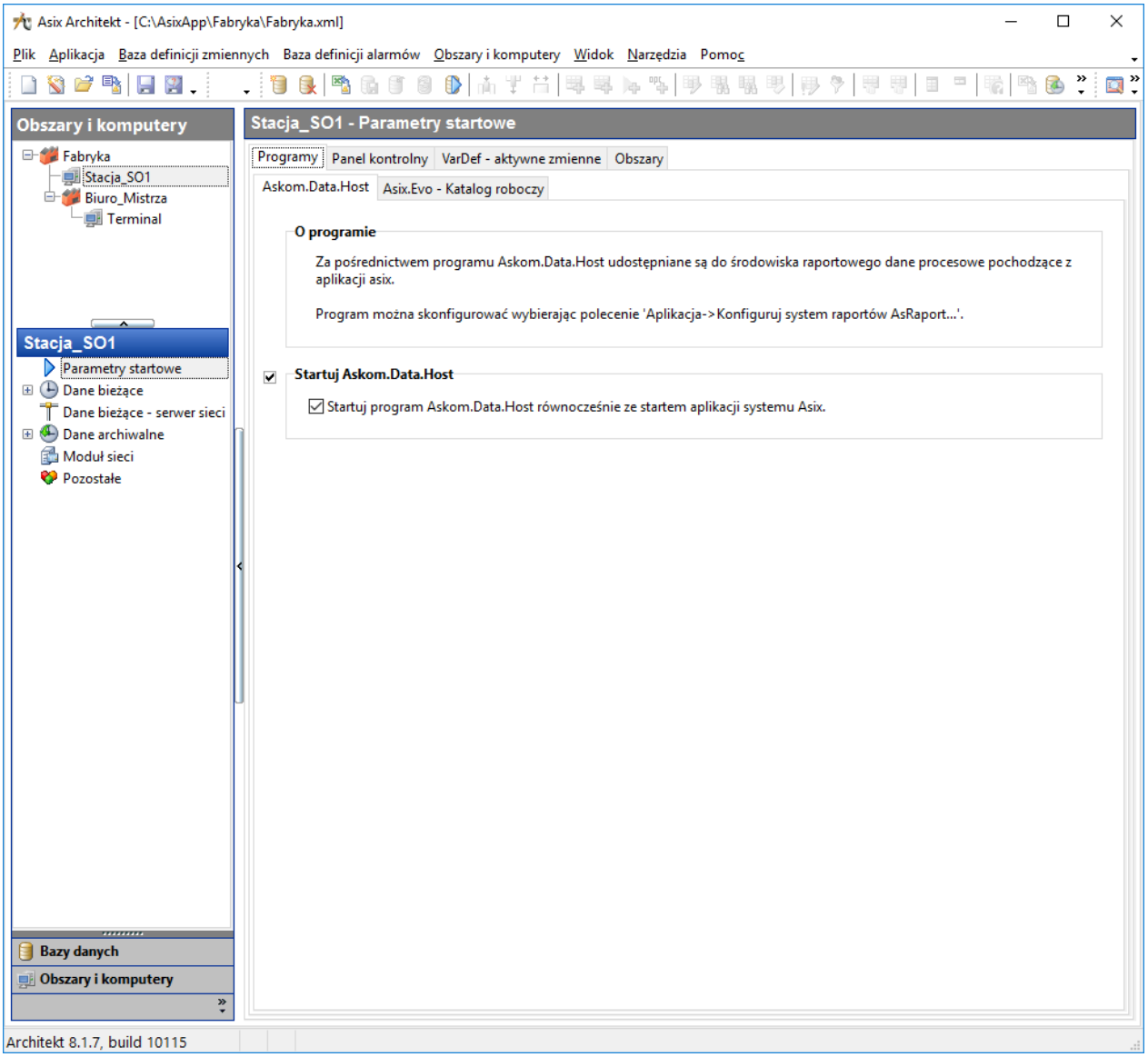
2.7. Konfiguracja parametrów startowych

Parametry startowe zawierają podstawowe parametry istotne na etapie uruchamiania aplikacji.

2.7.1. Deklaracja programów startowanych jednocześnie z aplikacją

Zakładka ***Programy***:

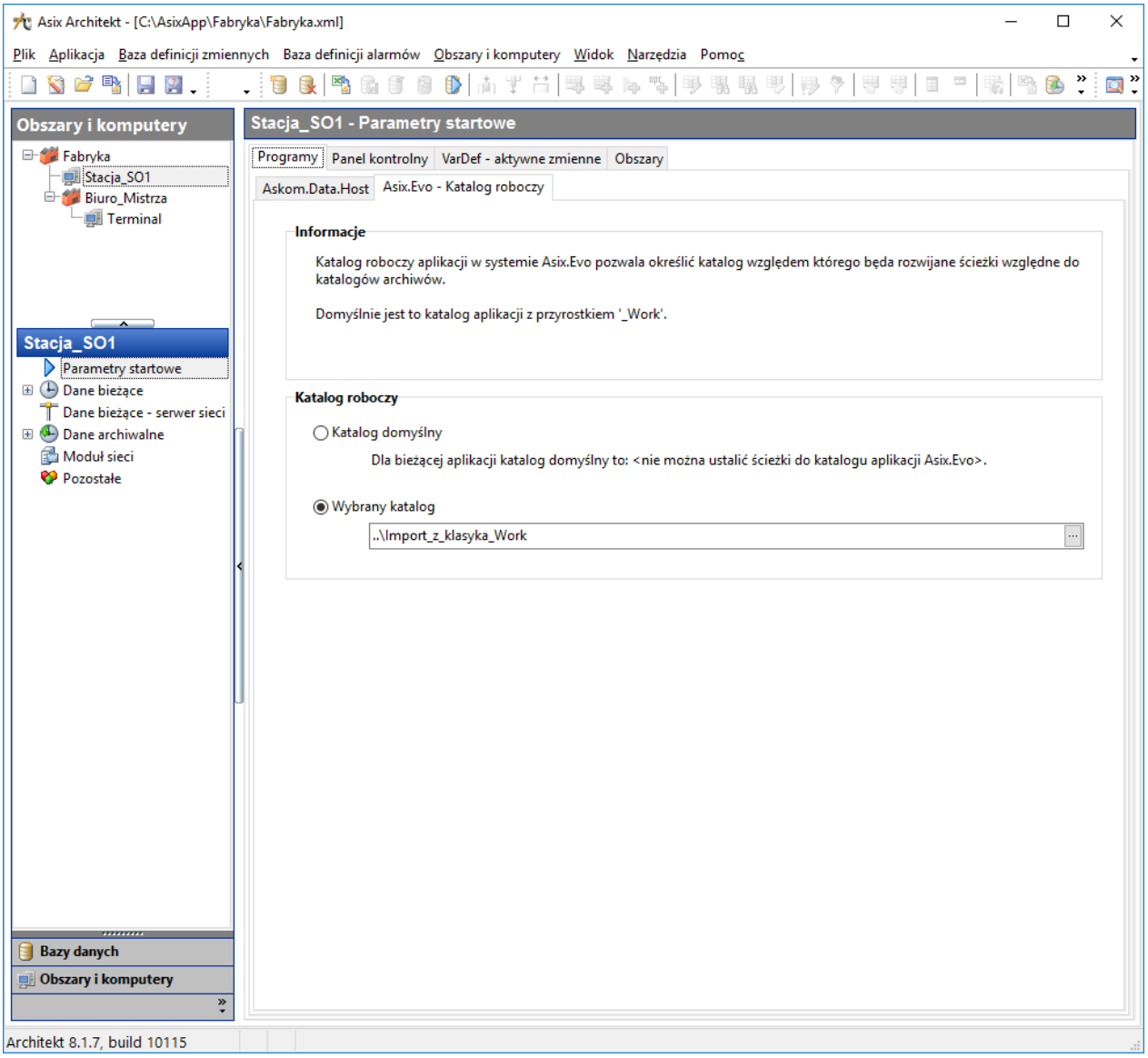
Askom.Data.Host:



Rysunek: Okno parametrów startowych - zakładka 'Askom.Data.Host'.

- ***Startuj Askom.Data.Host*** - pozwala zadeklarować startowanie programu Askom.Data.Host równocześnie ze startem aplikacji systemu Asix.

Asix.Evo - Katalog roboczy:



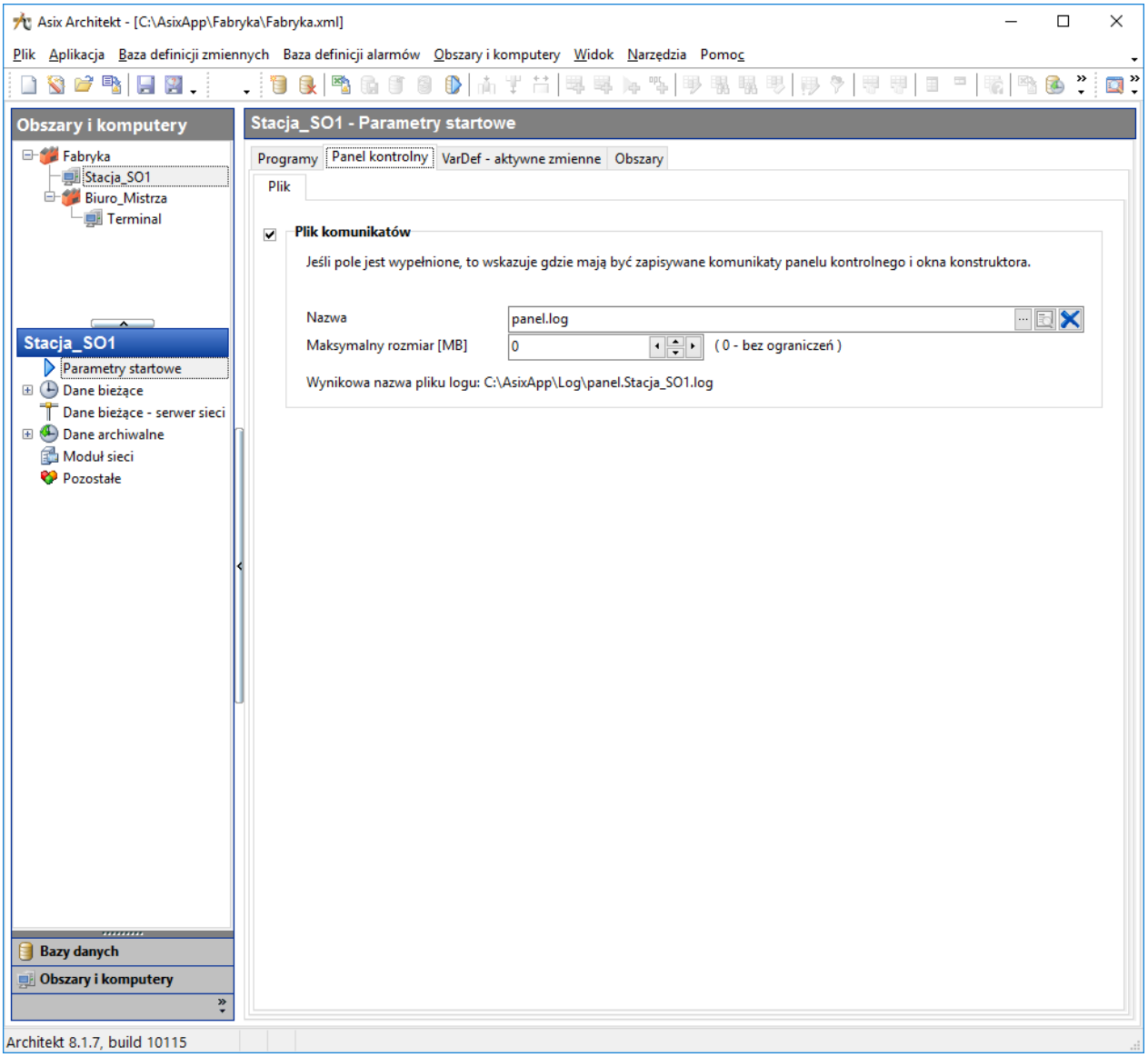
Rysunek: Okno parametrów startowych - zakładka 'Askom.Data.Host'.

- ***Katalog roboczy*** - pozwala zadeklarować katalog roboczy aplikacji w systemie Asix.Evo, względem którego rozwijane będą ścieżki względne do katalogu archiwów. Domyślnie jest to katalog aplikacji z przyrostkiem '_Work'.

2.7.2. Podstawowe parametry Panelu Kontrolnego i Konstruktora

Zakładka ***Panel kontrolny*** pozwala zdefiniować podstawowe parametry Panelu Kontrolnego i Konstruktora:

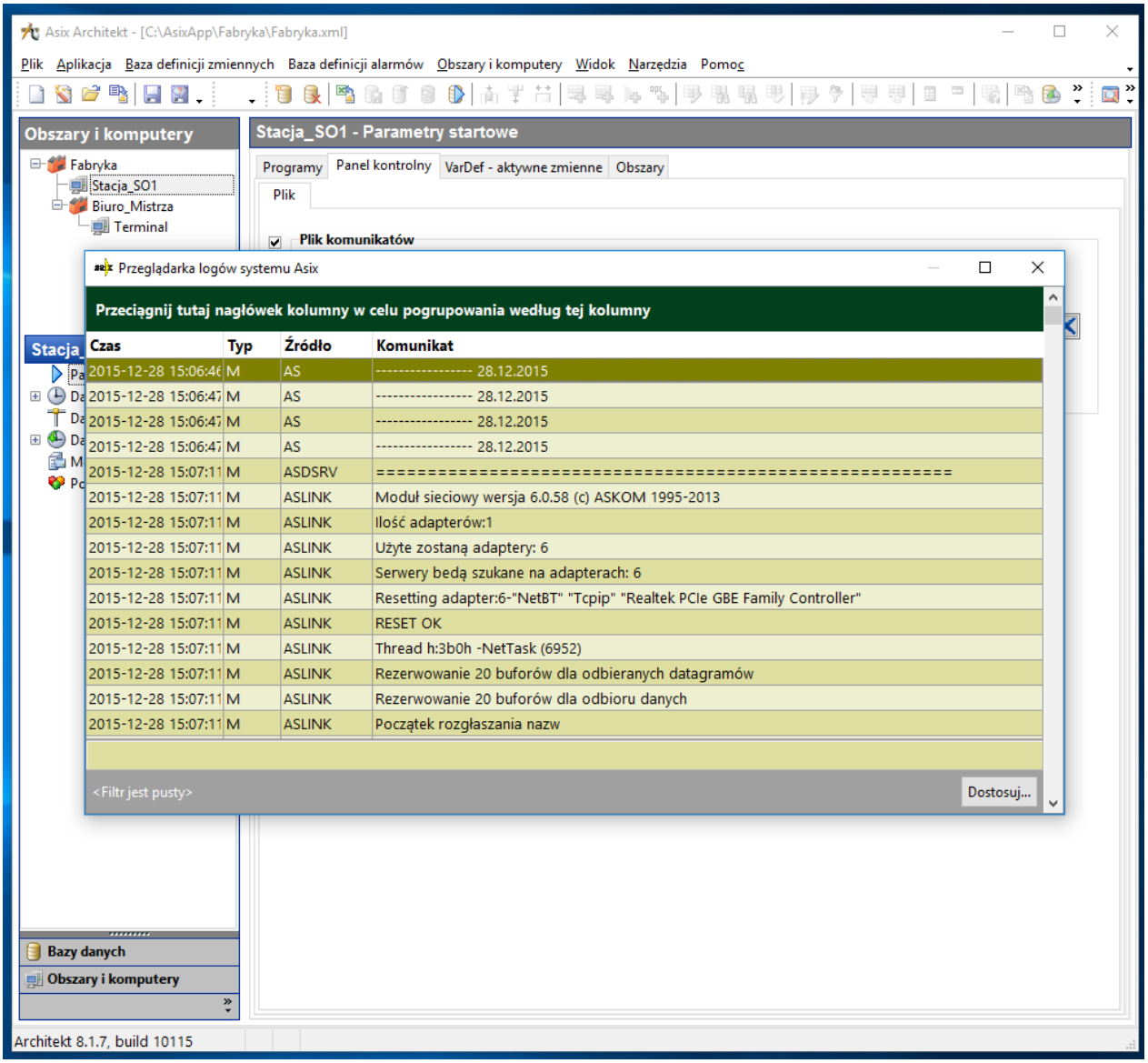
Plik:



Rysunek: Okno parametrów startowych - zakładka 'Panel Kontrolny/Plik'.

- ***Plik komunikatów*** - wskazuje miejsce, gdzie mają być zapisywane komunikaty Serwera Danych oraz maksymalny rozmiar pliku komunikatów.

  Wartość domyślna - komunikaty systemowe nie są zapisywane do pliku.

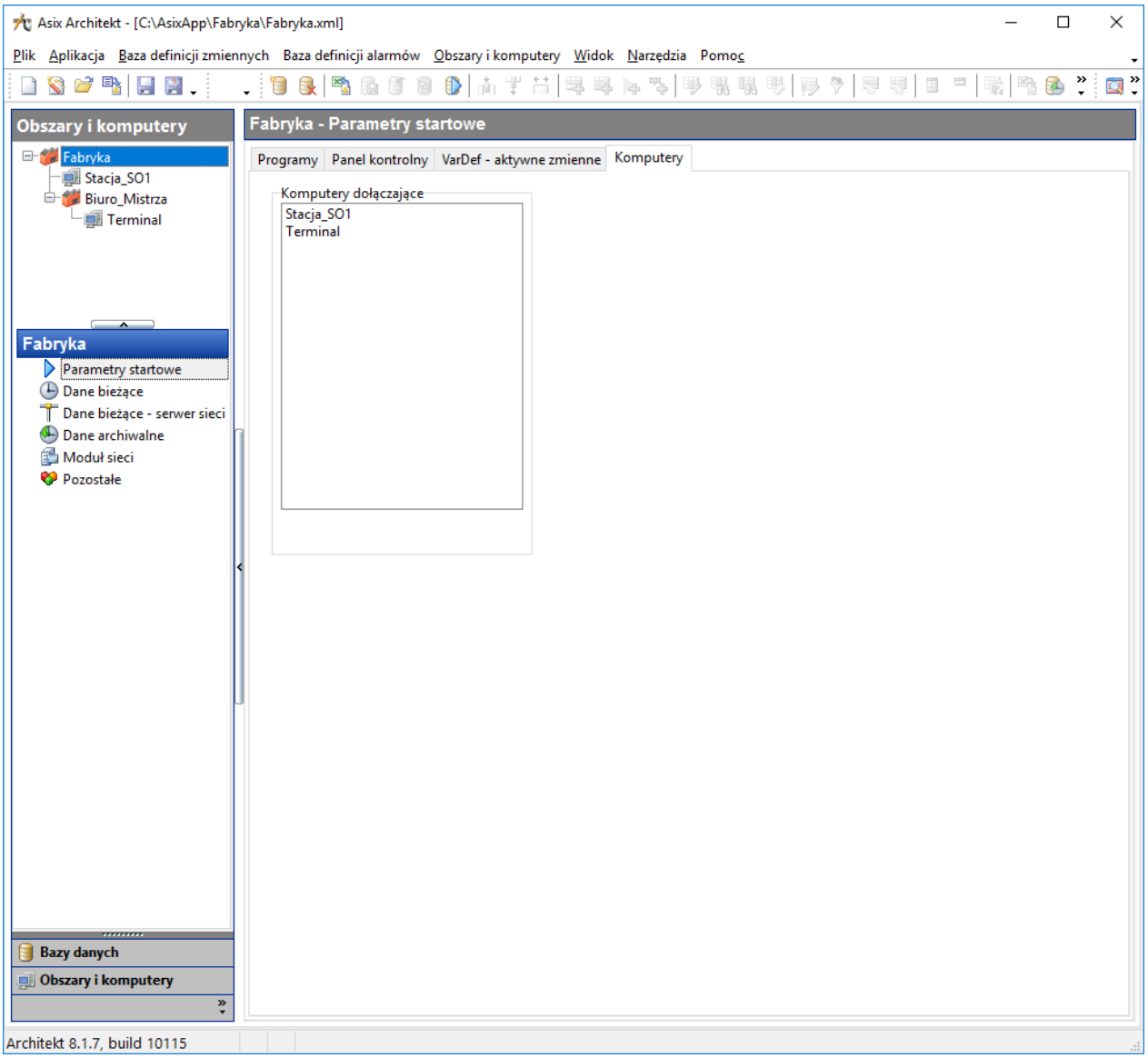


Ry. Przeglądarka logów Asixa.

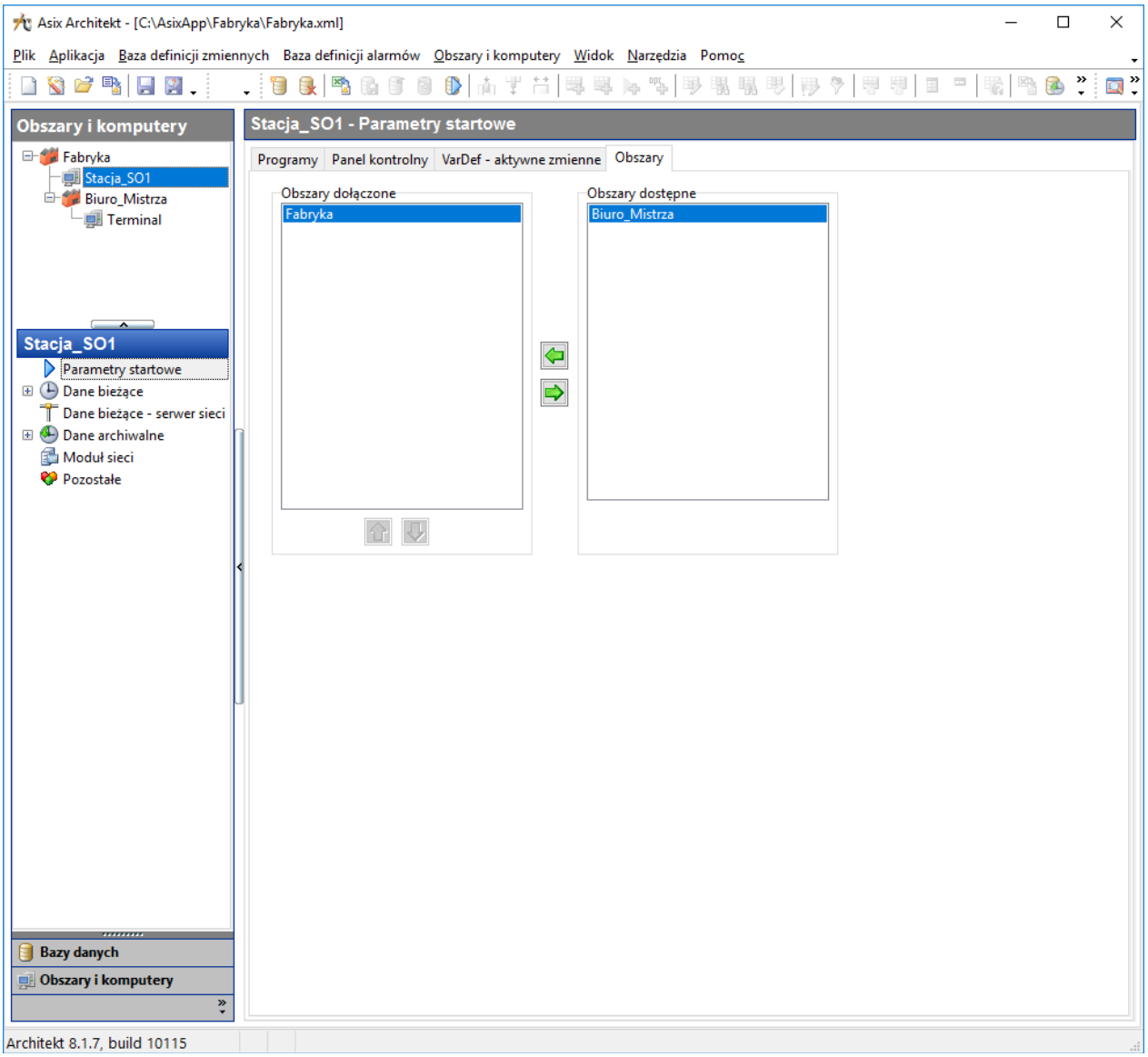
Przycisk uruchamia okno przeglądarki logów Asixa.
Przycisk powoduje skasowanie pliku logów.

2.7.3. Definiowanie zależności pomiędzy komputerami i obszarami

Widok zakładki ***Komputery*** zależny jest od tego, który element zaznaczony jest w drzewie komputerów i obszarów. Jeśli jest to *Obszar*, okno wyświetla listę komputerów podpiętych do danego obszaru (*patrz: Rysunek: Widok komputerów podłączonych do wskazanego obszaru.*). Jeśli jest to *Komputer*, pojawia się zakładka z zestawieniem obszarów dostępnych oraz obszarów, do których dołączony jest dany komputer (patrz: *Rysunek: Widok obszarów, do których podpięty jest wskazany komputer.*).



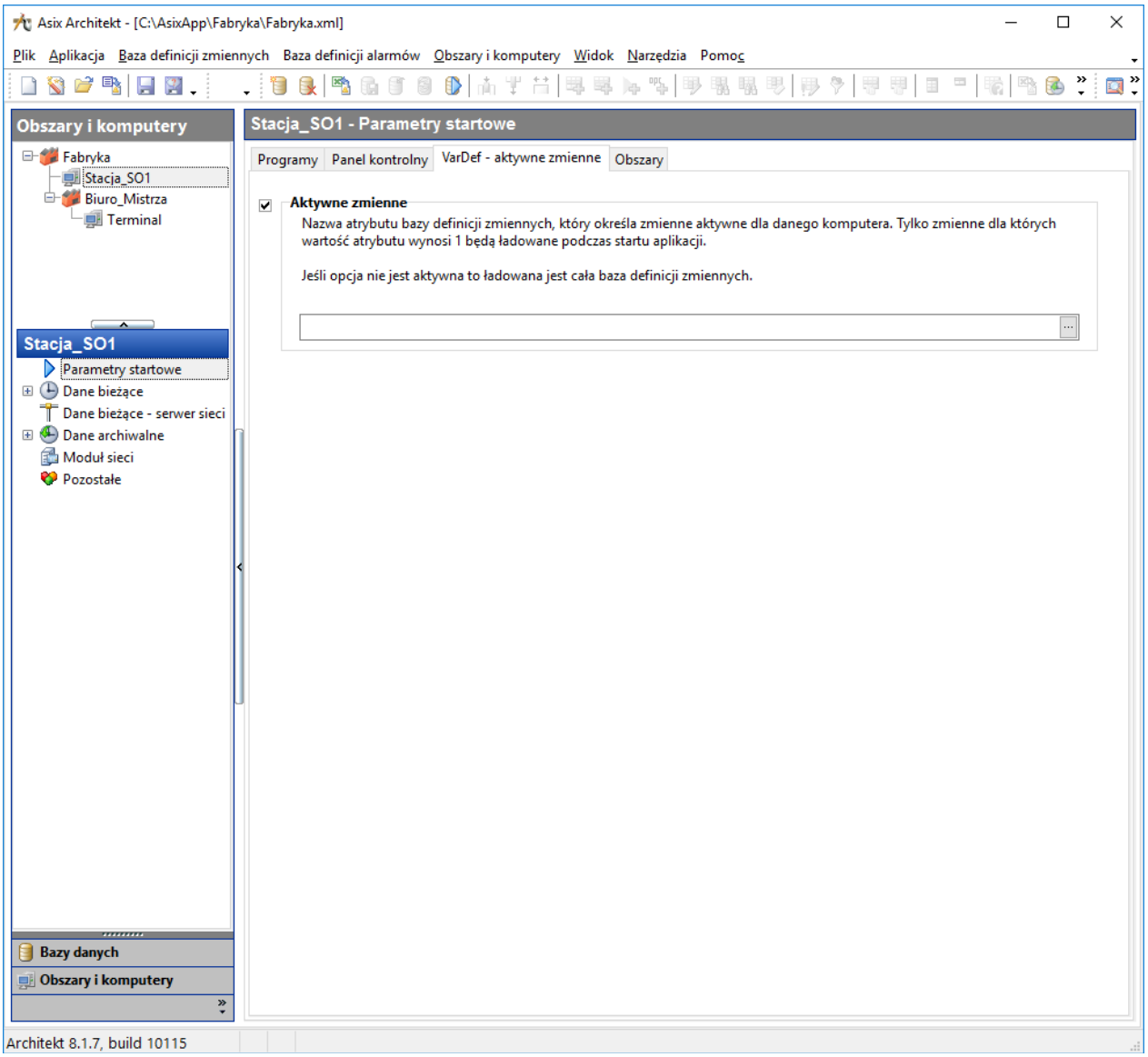
Rys. Widok komputerów podłączonych do wskazanego obszaru.



Rys. Widok obszarów, do których podpięty jest wskazany komputer.

2.7.4. Deklaracja aktywnych zmiennych

Zakładka ***VarDef - aktywne zmienne*** pozwala załączyć selekcję zmiennych wg atrybutu bazy definicji zmiennych, który pozwala określić zmienne aktywne dla danego komputera - jest to dowolny atrybut utworzony przez projektanta. Załączenie tej opcji i zadeklarowanie nazwy tego atrybutu spowoduje, ze podczas startu aplikacji ładowane będą tylko te zmienne, dla których wartość wskazanego atrybutu wynosi 1.

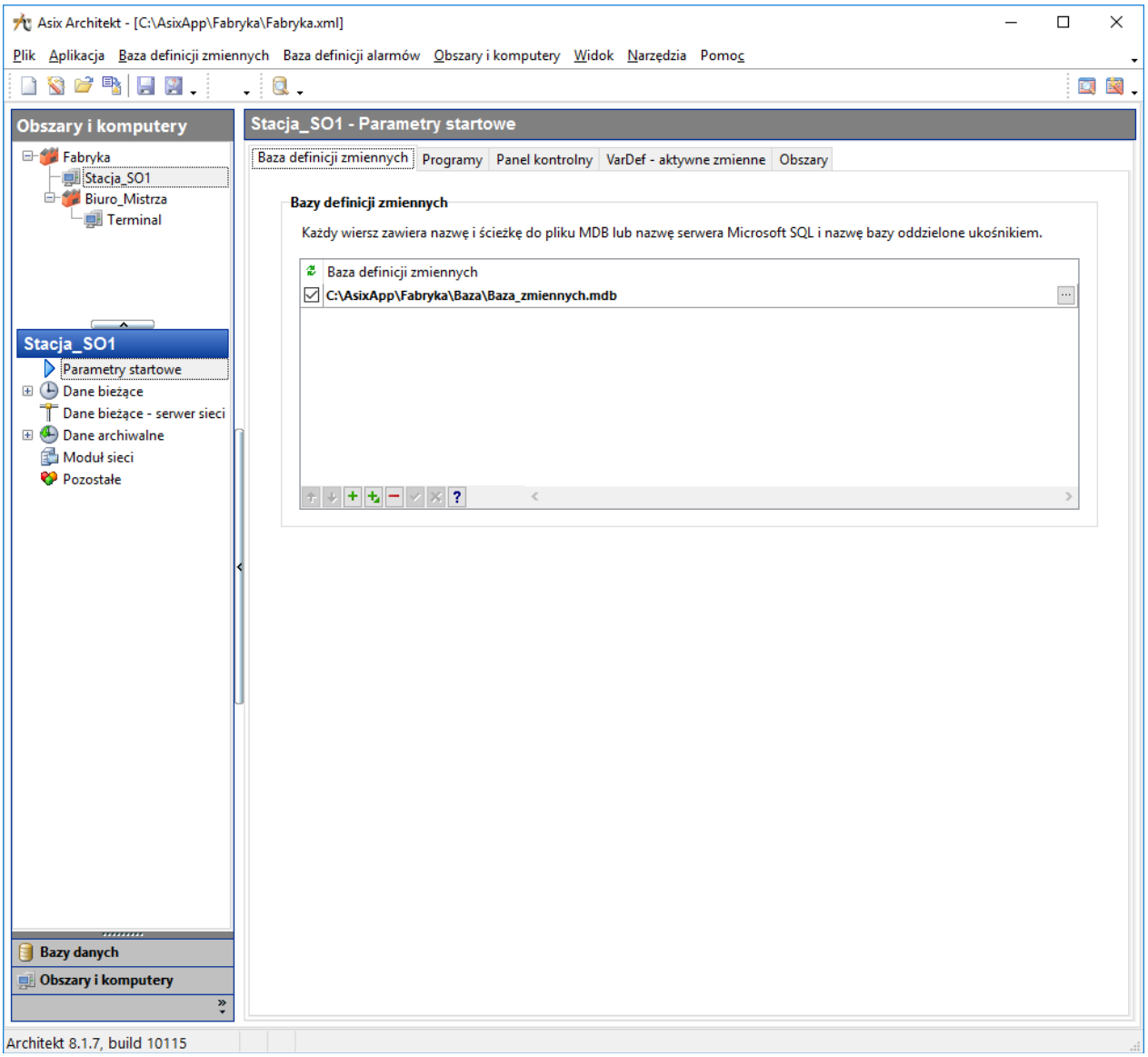


Rys. Okno parametrów startowych - zakładka 'VarDef - aktywne zmienne'.

2.7.5. Deklaracja zewnętrznej bazy definicji zmiennych

Zakładka *Baza definicji zmiennych* pojawia się w sytuacji, gdy:
1. dla aplikacji zadeklarowana jest *Zewnętrzna baza/bazy definicji zmiennych*:
Bazy danych > *Baza definicji zmiennych* > zakładka *Typ*

Zakładka ***Baza definicji zmiennych*** pozwala określić następujące opcje:



Rys. Okno parametrów startowych - zakładka 'Baza definicji zmiennych'.

- ***Bazy definicji zmiennych*** - pole to wskazuje lokalizację pliku zewnętrznej bazy definicji zmiennych - utworzonej za pomocą Architekta, parametryzowanej plikiem XML innym niż ten, który parametryzuje Serwer Danych aktualnej aplikacji.

2.8. Konfiguracja parametrów danych bieżących

Grupa parametrów  ***Dane bieżące*** obejmuje ustawienia pracy modułu Asmen, w tym szczegółową parametryzację kanałów transmisyjnych - fizycznie realizowanych połączeń pomiędzy komputerem systemu Asix i sterownikami na obiekcie oraz parametryzację drajwerów komunikacyjnych.

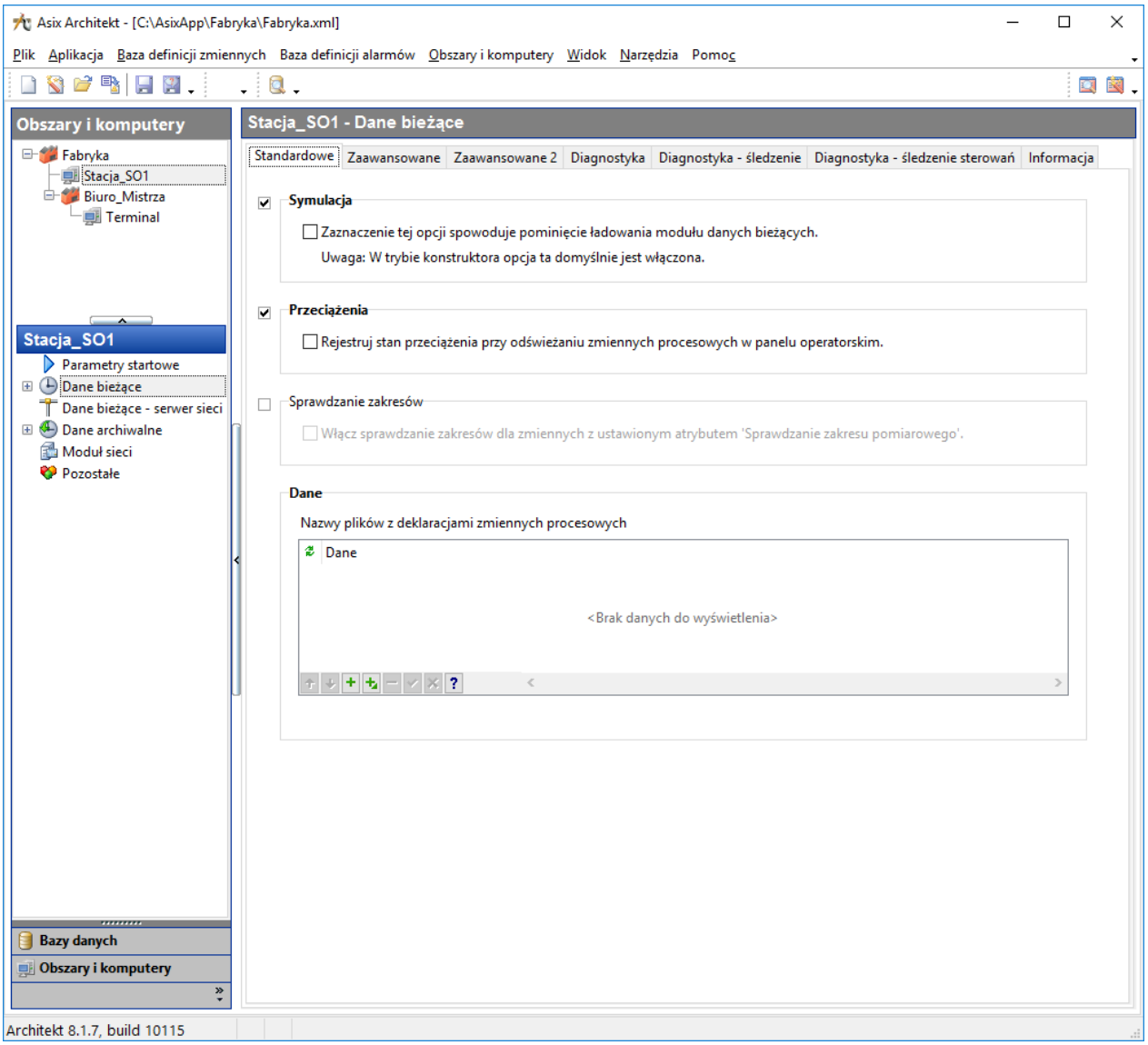
W systemie Asix wszystkie operacje związane z dostępem do zmiennych procesowych są realizowane przez moduł Asmen. Moduł Asmen nadzoruje proces cyklicznej aktualizacji wartości zmiennych procesowych w bazie danych oraz pośredniczy we wszystkich operacjach związanych z przesyłem sterowań do obiektu. Bezpośrednia wymiana danych pomiędzy komputerem i sterownikiem obiektu jest wykonywana poprzez drajwery pracujące pod kontrolą Asmena.

W oparciu o dokonaną parametryzację tworzone jest repozytorium bieżących wartości zmiennych procesowych (przechowywane przez moduł Asmen w pamięci operacyjnej komputera), do którego mają dostęp wszystkie komponenty systemu Asix. Wartości przechowywanych zmiennych procesowych są na bieżąco aktualizowane poprzez cykliczny odczyt z obiektu.


Szczegółowy opis funkcjonalności modułu Asmen znajduje się w części Serwer Danych > Obsługa danych procesowych Serwera Danych > rozdziale *1.2. Asmen - menedżer komunikacyjny*.

2.8.1. Parametry standardowe

Parametry dotyczące danych bieżących znajdujące się na zakładce ***Standardowe*** obejmują:



Rys. Okno parametrów danych bieżących - zakładka 'Standardowe'.

- ***Symulacja*** **-** Asmen działa w trybie symulacji danych po zaznaczeniu opcji *Zaznaczenie tej opcji spowoduje pominięcie ładowania modułu danych bieżących Asmen*.

  Wartość domyślna: domyślnie opcja jest wyłączona.

- ***Przeciążenia*** **-** wykorzystanie parametru pozwala na rejestrację stanu przeciążenia przy odświeżaniu zmiennych procesowych w panelu operatorskim;

  Wartość domyślna: domyślnie opcja jest włączona;

  Przeciążenia te dotyczą zmiennych - w sytuacji gdy zadeklarowany czas odświeżania zmiennych przekracza możliwości kanału transmisyjnego. (Częstość odświeżania deklarowana jest w definicji zmiennej).

- ***Sprawdzanie zakresów*** **-** włącza sprawdzanie zakresów dla zmiennych z ustawionym atrybutem *Sprawdzanie zakresu pomiarowego*;

  Wartość domyślna: domyślnie opcja jest wyłączona;

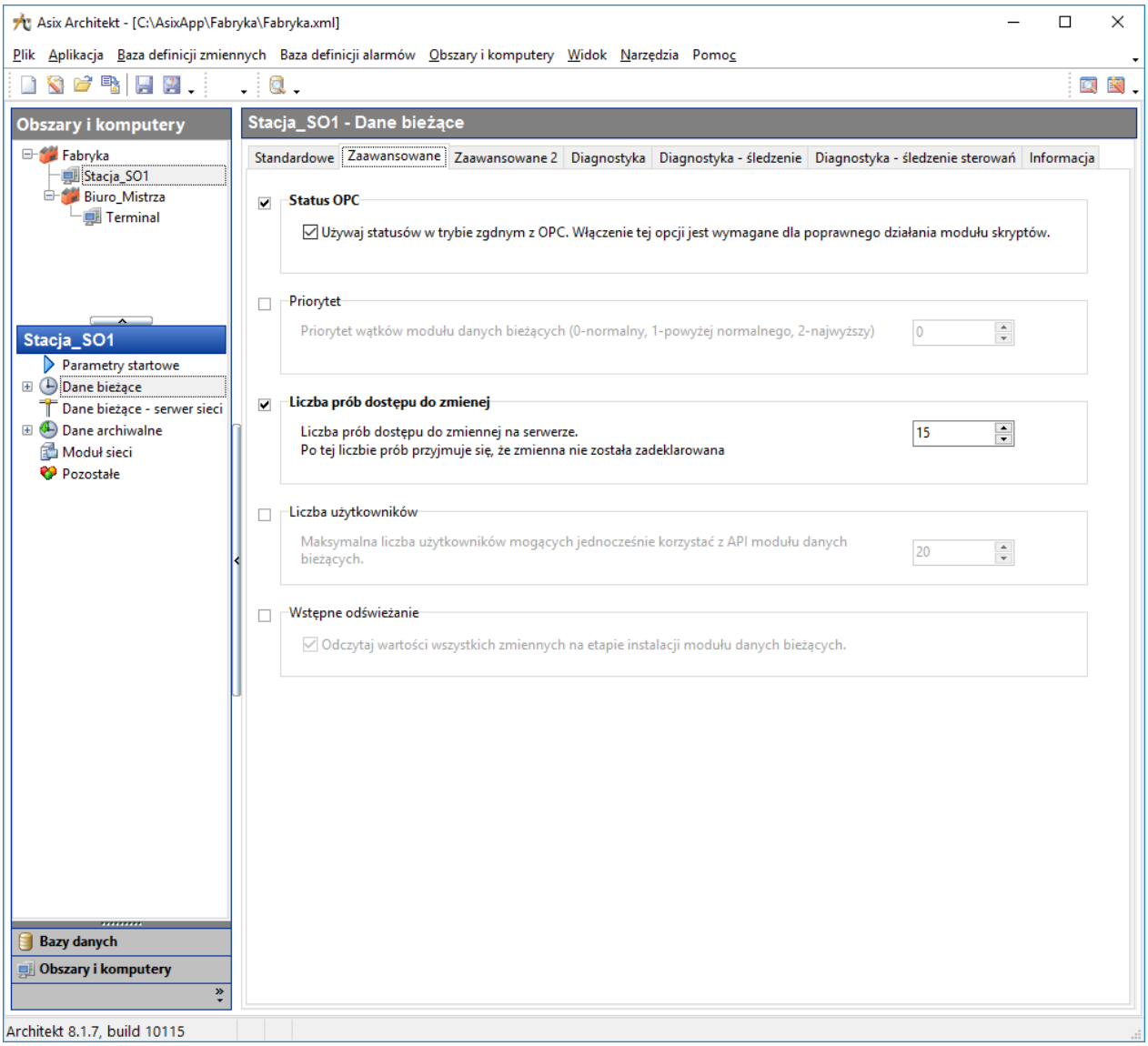
- ***Dane / Nazwy plików z deklaracjami zmiennych procesowych***

Pole dostępne jest wyłącznie wówczas, gdy definicje zmiennych procesowych pobierane są z plików tekstowych, co zostało uprzednio zadeklarowane w:
Moduł *Bazy danych* > zakładka *Typ* > *Typ bazy definicji zmiennych: Tekstowa baza definicji zmiennych (Asix 1-4).*

2.8.2. Parametry zaawansowane

Zakładki *Zaawansowane* i *Zaawansowane2* pojawiają się po wybraniu polecenia: *Obszary i komputery* > *Pokaż opcje zaawansowane*

Zakładki ***Zaawansowane*** i ***Zaawansowane 2*** pozwala na parametryzację następujących pozycji:



Rys. Okno parametrów danych bieżących - zakładka 'Zaawansowane'.

- ***Status OPC*** - deklaracja używania statusów w trybie zgodnym z OPC; włączenie tej opcji jest wymagane dla poprawnego działania modułu skryptów.
  Wartość domyślna: opcja włączona

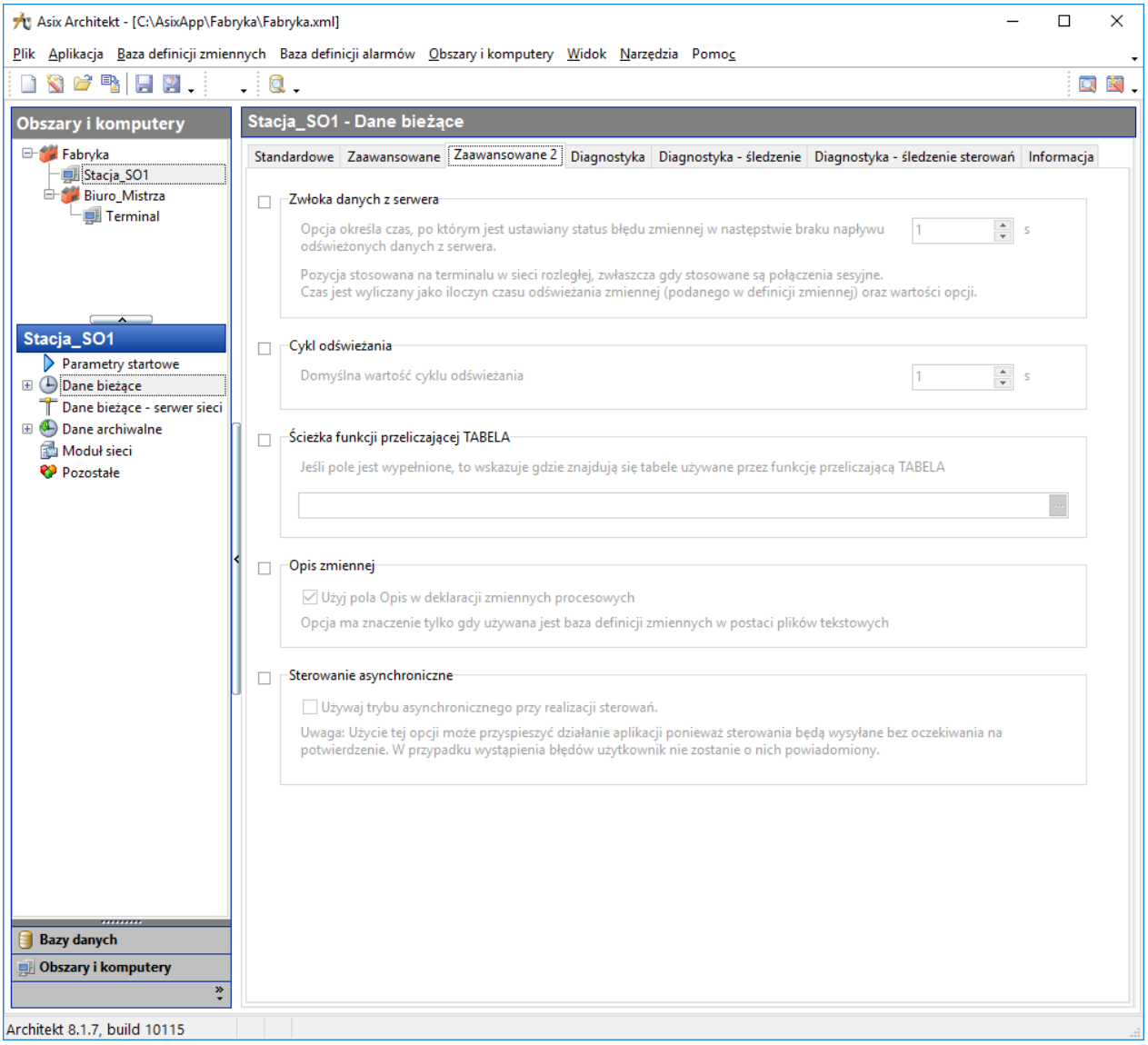
- ***Priorytet*** - deklaracja priorytetów wątków Asmena:
  - 0 - THREAD_PRIORITY_NORMAL,
  - 1 - THREAD_PRIORITY_ABOVE_NORMAL,
  - 2 - THREAD_PRIORITY_HIGHEST.

  Wartość domyślna: 0

- ***Liczba prób dostępu do zmiennej*** - deklaracja liczby prób zdalnego dostępu do zmiennej na serwerze, po której przyjmuje się, że zmienna na przedmiotowym serwerze nie została zadeklarowana;
  Wartość domyślna: 1

- ***Liczba użytkowników*** - pozwala zdefiniować maksymalną liczbę użytkowników korzystających z API Asmena. Pozycja ma zastosowanie w wieloportowych aplikacjach z drajwerem MUZ;
  Wartość domyślna: 128

- ***Wstępne odświeżanie*** - jeśli opcja zostanie zadeklarowana, na etapie instalacji Asmena następuje odczyt aktualnych wartości wszystkich zmiennych procesowych;
  Wartość domyślna: opcja załączona



Rys. Okno parametrów danych bieżących - zakładka 'Zaawansowane2'.

- ***Zwłoka danych z serwera*** - pozycja stosowana na terminalu; określa czas, po którym następuje ustawienie statusu błędu zmiennej w następstwie braku napływu odświeżonych danych z serwera; czas jest wyliczany jako iloczyn czasu odświeżania zmiennej (podanego w definicji zmiennej) oraz wartości pozycji; pozycja ma zastosowanie do wszystkich zmiennych jednocześnie; jednostka: sekunda.
  Wartość domyślna: 1 sekunda

- ***Cykl odświeżania*** - deklaracja wartości domyślnego cyklu odświeżania zmiennych procesowych (w sekundach), dla których nie podano wartości cyklu odświeżania w deklaracji zmiennej;
  Wartość domyślna: 1 sekunda

- ***Ścieżka funkcji przeliczającej TABELA***- deklaracja nazwy kartoteki, w której znajdują się tabele używane przez funkcje przeliczające TABELA;
  Wartość domyślna: brak zadeklarowanej ścieżki

- ***Opis zmiennej*** - określa użycie pola *Opis* w deklaracji zmiennych procesowych Asmena;

Użycie opcji jest zasadne wyłącznie wtedy - gdy deklaracje zmiennych procesowych pobierane są z plików - tekstowych.

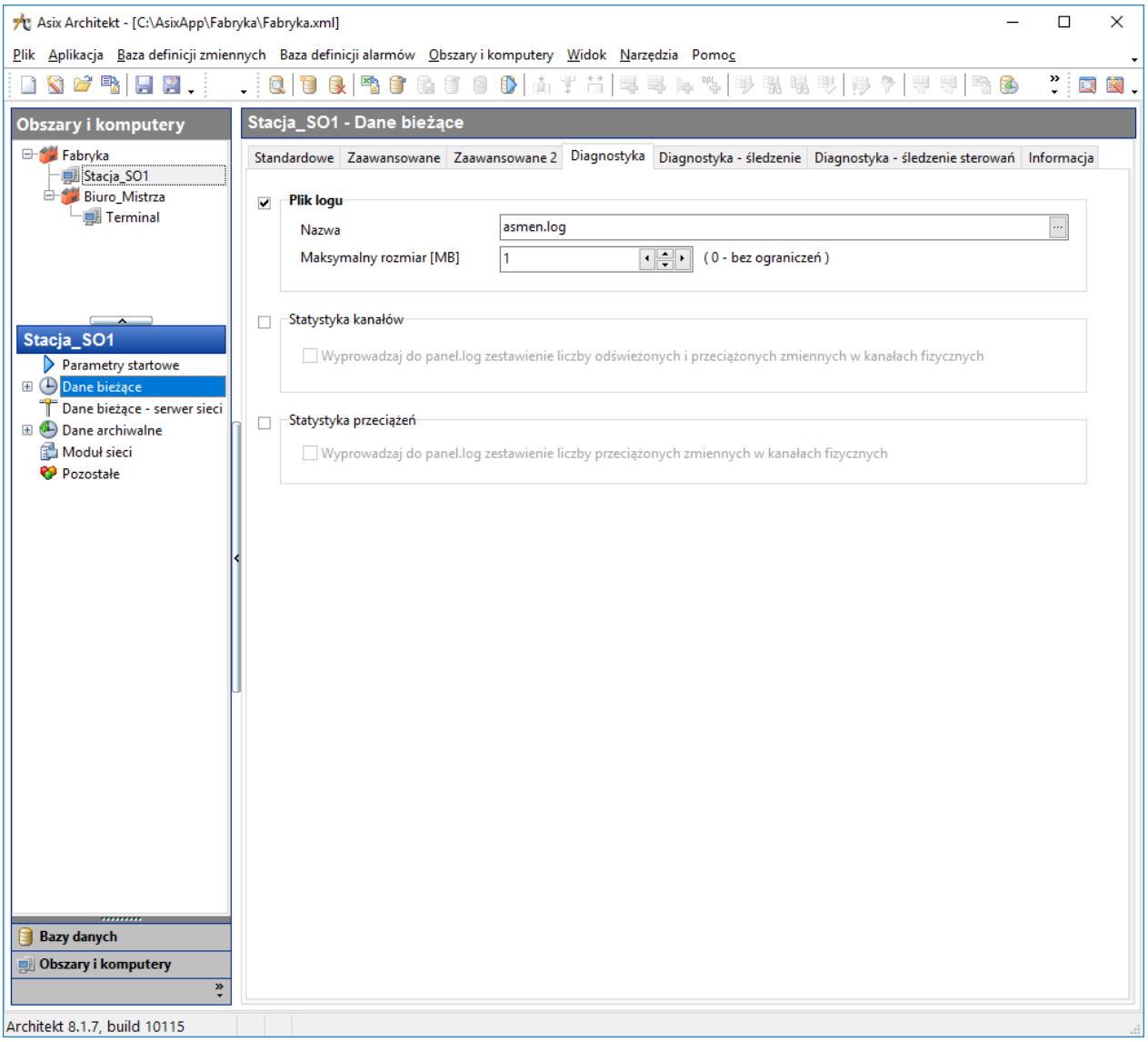
Wartość domyślna: opcja jest włączona

- ***Sterowanie asynchroniczne*** - deklaracja użycia trybu asynchronicznego przy realizacji sterowań w szybkich kanałach transmisji;
  Wartość domyślna: tryb asynchroniczny nie jest używany

2.8.3. Parametry diagnostyczne

Zakładka *Diagnostyka* pojawia się po wybraniu polecenia:
Widok > Pokaż opcje zaawansowane

Zakładka ***Diagnostyka*** obejmuje:



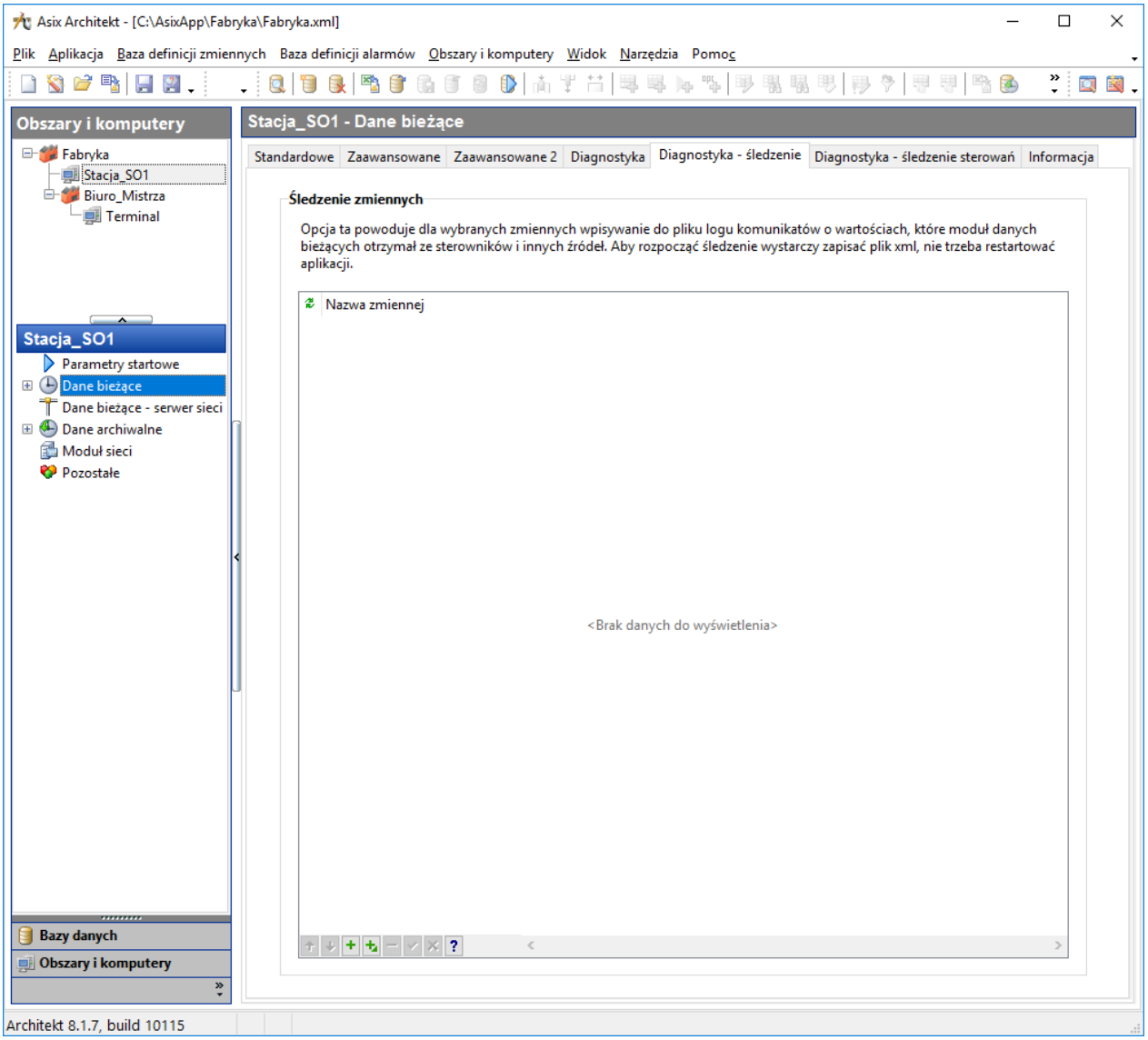
Rys. Okno parametrów danych bieżących - zakładka 'Diagnostyka'.

- ***Plik logu*** - domyślnie opcja załączana dla nowych aplikacji

- ***Statystyka kanałów*** - zestawienie liczby odświeżanych oraz przeciążonych zmiennych w poszczególnych kanałach fizycznych; zestawienie wyprowadzane do Panelu Kontrolnego co sekundę;
  Wartość domyślna: opcja jest wyłączona

- ***Statystyka przeciążeń*** - zestawienie liczby przeciążonych zmiennych w kanałach fizycznych; zestawienie jest wyprowadzane do Panelu Kontrolnego wtedy, gdy wystąpi przeciążenie;
  Wartość domyślna: opcja jest wyłączona

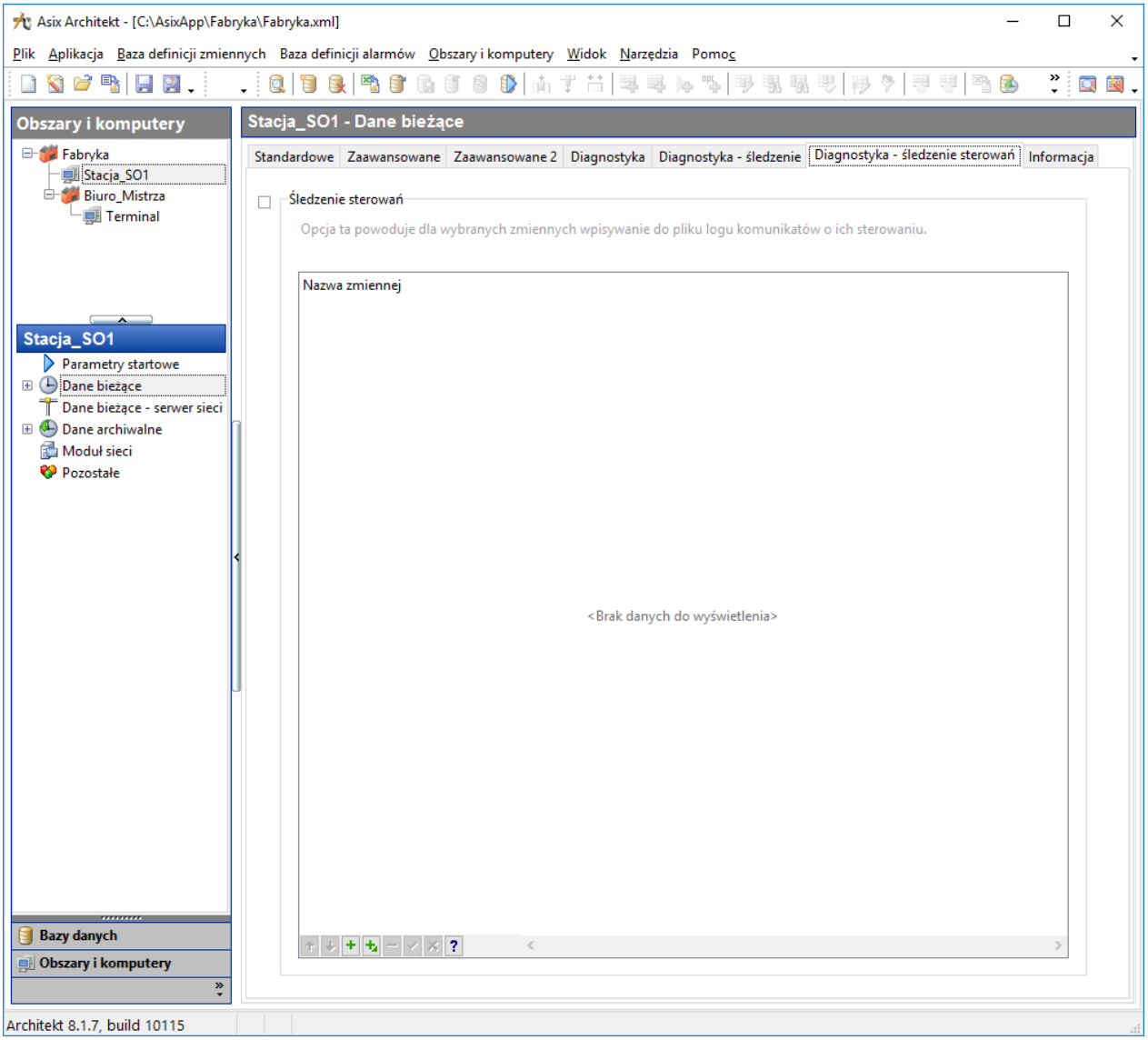
Zakładka ***Diagnostyka - śledzenie*** **obejmuje:**



Rys. Okno parametrów danych bieżących - zakładka 'Diagnostyka - śledzenie'.

- ***Śledzenie zmiennych*** - opcja ta powoduje dla wybranych zmiennych wpisywanie do pliku logu komunikatów o wartościach, które moduł danych bieżących otrzymał ze sterowników i innych źródeł.

Zakładka *Diagnostyka - śledzenie sterowań* obejmuje:

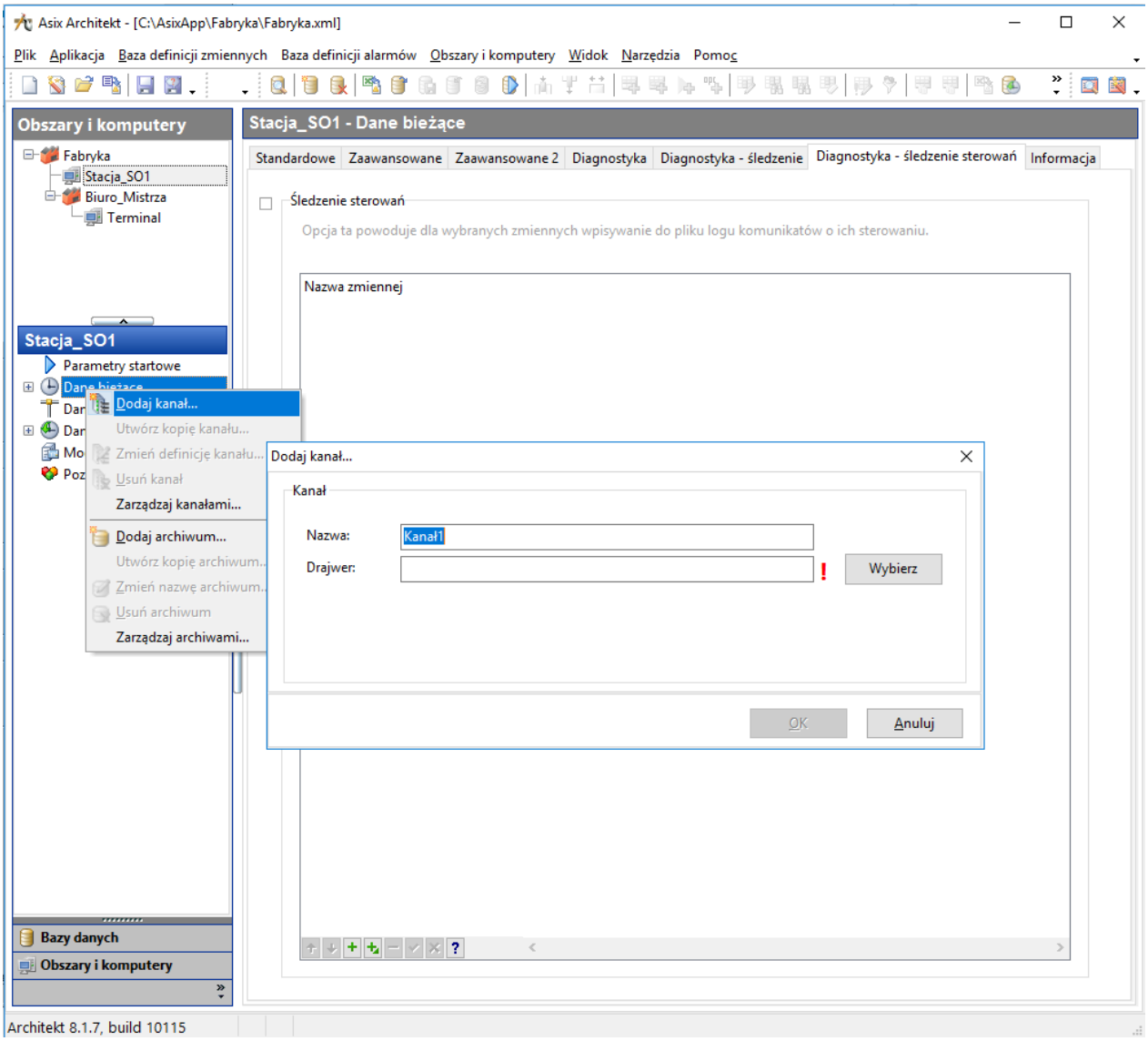


Rys. Okno parametrów danych bieżących - zakładka 'Diagnostyka - śledzenie sterowań'.

- ***Śledzenie sterowań***- opcja ta powoduje dla wybranych zmiennych wpisywanie do pliku logu komunikatów o ich sterowaniu.

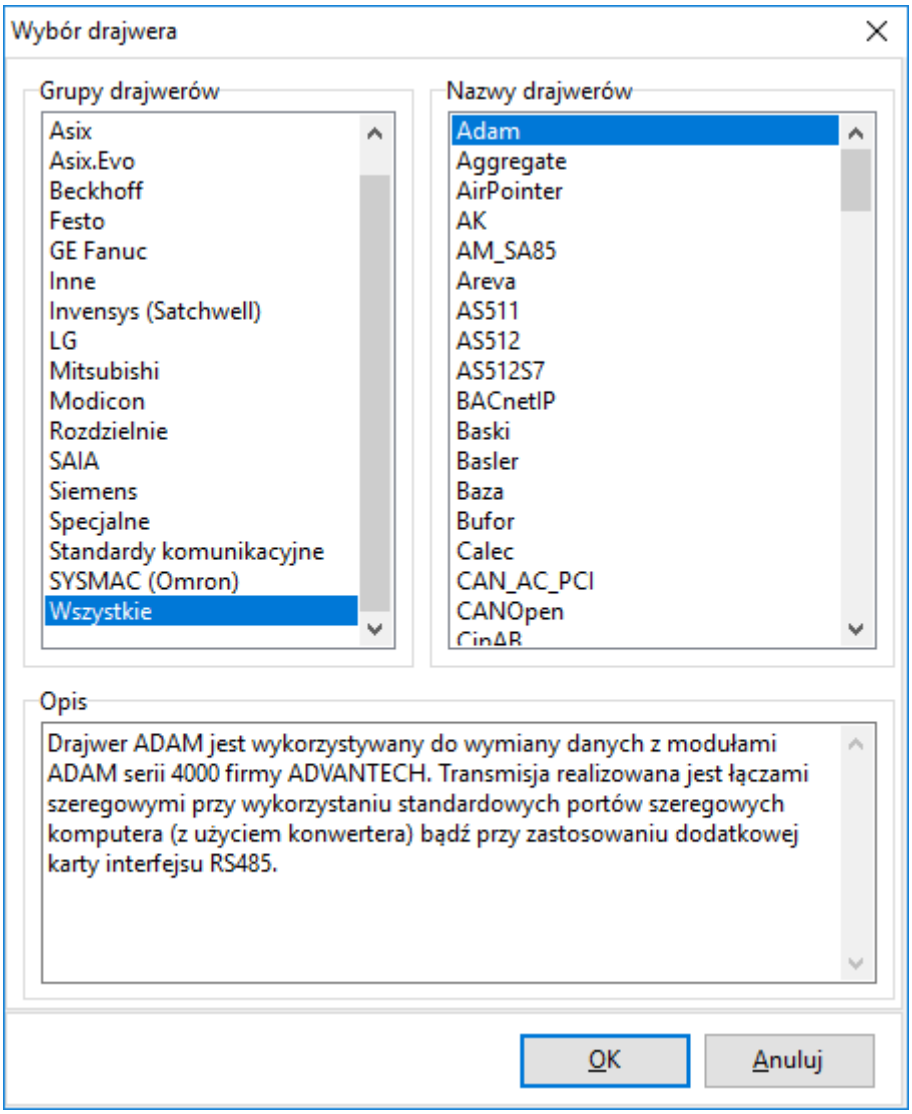
2.8.4. Definicja kanałów komunikacyjnych

Definicje kanałów komunikacyjnych dodawane są za pomocą polecenia ***Dodaj kanał*** po kliknięciu prawym klawiszem myszki na pozycji ***Dane bieżące***. Pojawia się wówczas okno *'Dodaj kanał'*, gdzie należy określić nazwę kanału i nazwę drajwera komunikacyjnego.



Rys. Okno deklarowania definicji kanałów.

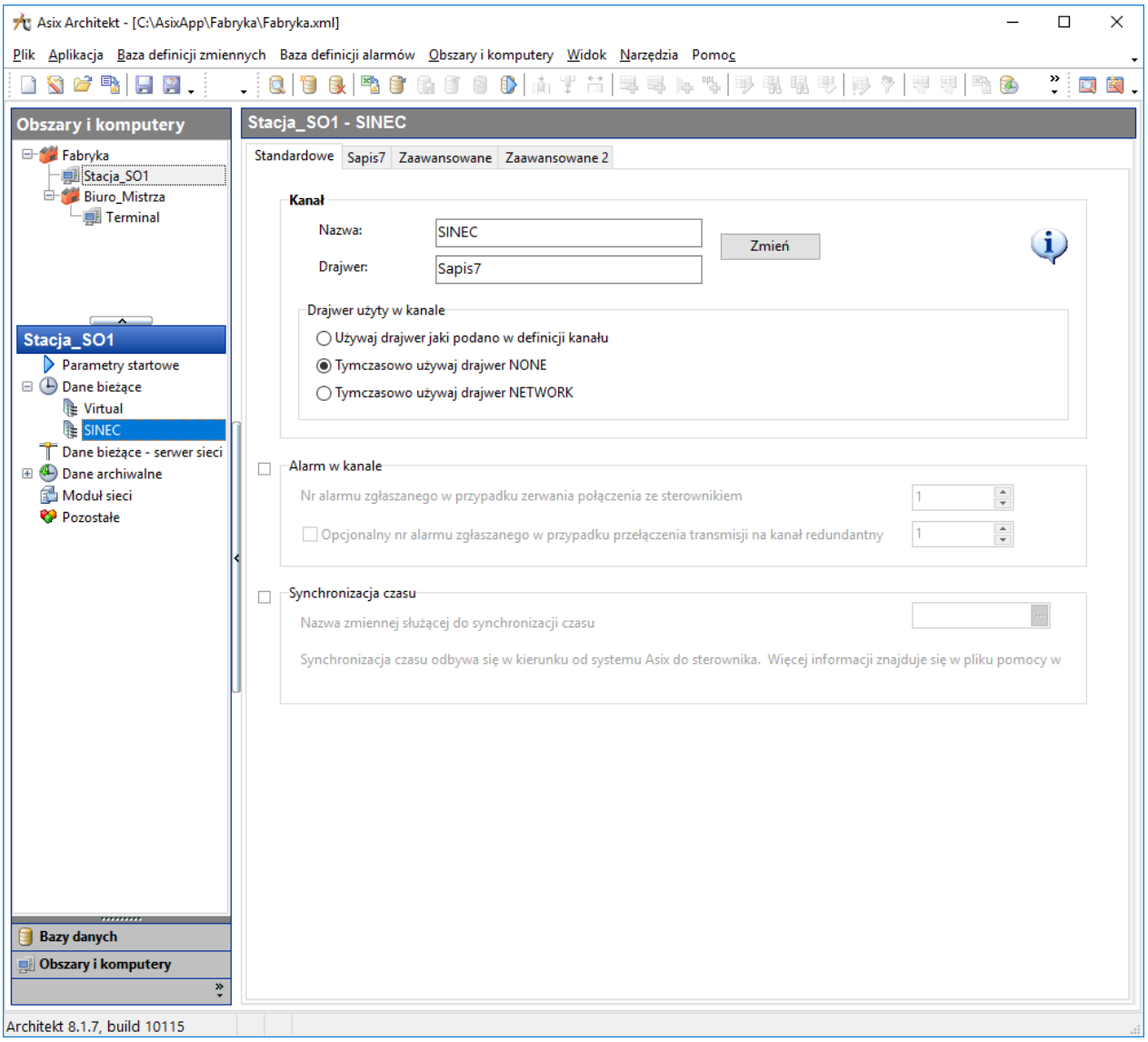
Wyboru nazwy drajwera należy dokonać z listy drajwerów dostępnych w pakiecie Asix, która uruchamiana jest przyciskiem *Wybierz* w oknie *'Dodaj kanał'*.



Rys. Okno wyboru drajwerów komunikacyjnych.

Po wybraniu konkretnego drajwera w grupie parametrów *Dane bieżące* pojawia się pozycja z nazwą wyselekcjonowanego drajwera. Kliknięcie na tę pozycję otwiera w oknie parametrów zakładki, które służą do parametryzacji tego drajwera - zakładki te zostały opisane poniżej. Ponadto, kliknięcie prawym klawiszem myszki na wybranym kanale uruchamia podręczne menu z komendami umożliwiającymi zmianę definicji kanału oraz usunięcie zaznaczonego kanału.

Zakładka ***Standardowe*** obejmuje standardowe parametry deklaracji każdego kanału:



Rys. Widok okna parametryzującego pracę drajwera komunikacyjnego - zakładka 'Standardowe'.

- ***Kanał/Nazwa*** **-** określa nazwę kanału;
- ***Kanał/Drajwer*** **-** określa nazwę drajwera;
- ***Drajwer użyty w kanale:***
  - ***Używaj drajwer jaki podano w definicji kanału***;
  - ***Tymczasowo używaj drajwer NONE*** **-** opcja stosowana w przypadku braku fizycznego podłączenia do sterownika; drajwer NONE może być stosowany w celu testowania aplikacji;
  - ***Tymczasowo używaj drajwer NETWORK*** **-** drajwer NETWORK jest specyficznym rodzajem drajwera; nie realizuje on fizycznego połączenia ze sterownikiem, a jedynie połączenie z innym komputerem, który takie połączenie ma; drajwer kanału sieciowego po nawiązaniu połączenia wysyła zapytanie do wszystkich serwerów podając nazwę logiczną

kanału; odpowiadają na to pytanie jedynie serwery, które posiadają zadeklarowany taki kanał; jeżeli odpowie więcej niż jeden serwer, to wtedy do nawiązania połączenia wybiera się serwer najmniej obciążony; jeżeli w czasie pracy nastąpi wyłączenie serwera, wówczas drajwer kanału sieciowego przełączy się na inny serwer;

- ***Alarm w kanale*** **(parametr dostępny dla wybranych drajwerów)** - deklaracja zgłoszenia alarmu w przypadku braku transmisji w kanale logicznym; opcjonalnie możliwe jest zgłoszenie nr alarmu w przypadku przełączenia transmisji na kanał redundantny (o ile kanał redundantny został zadeklarowany dla kanału logicznego);
  Wartość domyślna: brak deklaracji kanału podstawowego i redundantnego

- ***Synchronizacja czasu*** **(parametr dostępny dla wybranych drajwerów)** - synchronizacja czasu odbywa się w kierunku od systemu Asix do sterownika. Więcej informacji znajduje się w odrębnym podręczniku 'Drajwery komunikacyjne';
  Wartość domyślna: brak deklaracji

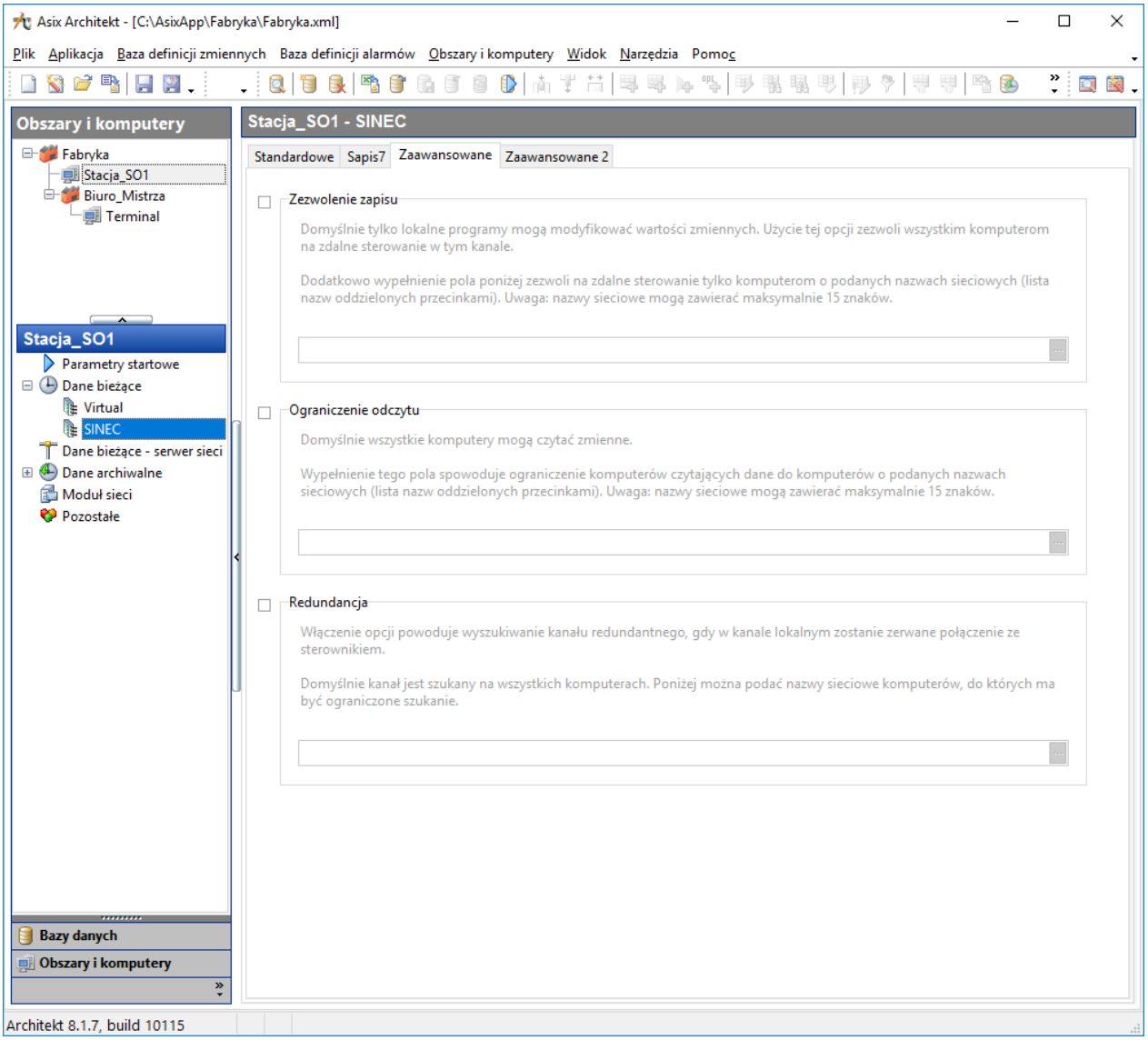
Pozostałe parametry dostępne na zakładce *Standardowe* zależą od konkretnego zadeklarowanego drajwera.

Zakładka z nazwą zadeklarowanego drajwera obejmuje parametry właściwe do skonfigurowania kanału komunikacyjnego dla danego drajwera.

Opisy parametryzacji poszczególnych drajwerów znajdują się w odrębnym podręczniku *"Drajwery komunikacyjne".*

Zakładka *Zaawansowane* i *Zaawansowane 2* pojawia się po wybraniu polecenia *Obszary i komputera* > *Pokaż opcje zaawansowane* z menu *Widok* modułu Architekt.

Zakładka ***Zaawansowane*** pozwala na parametryzację następujących pozycji:



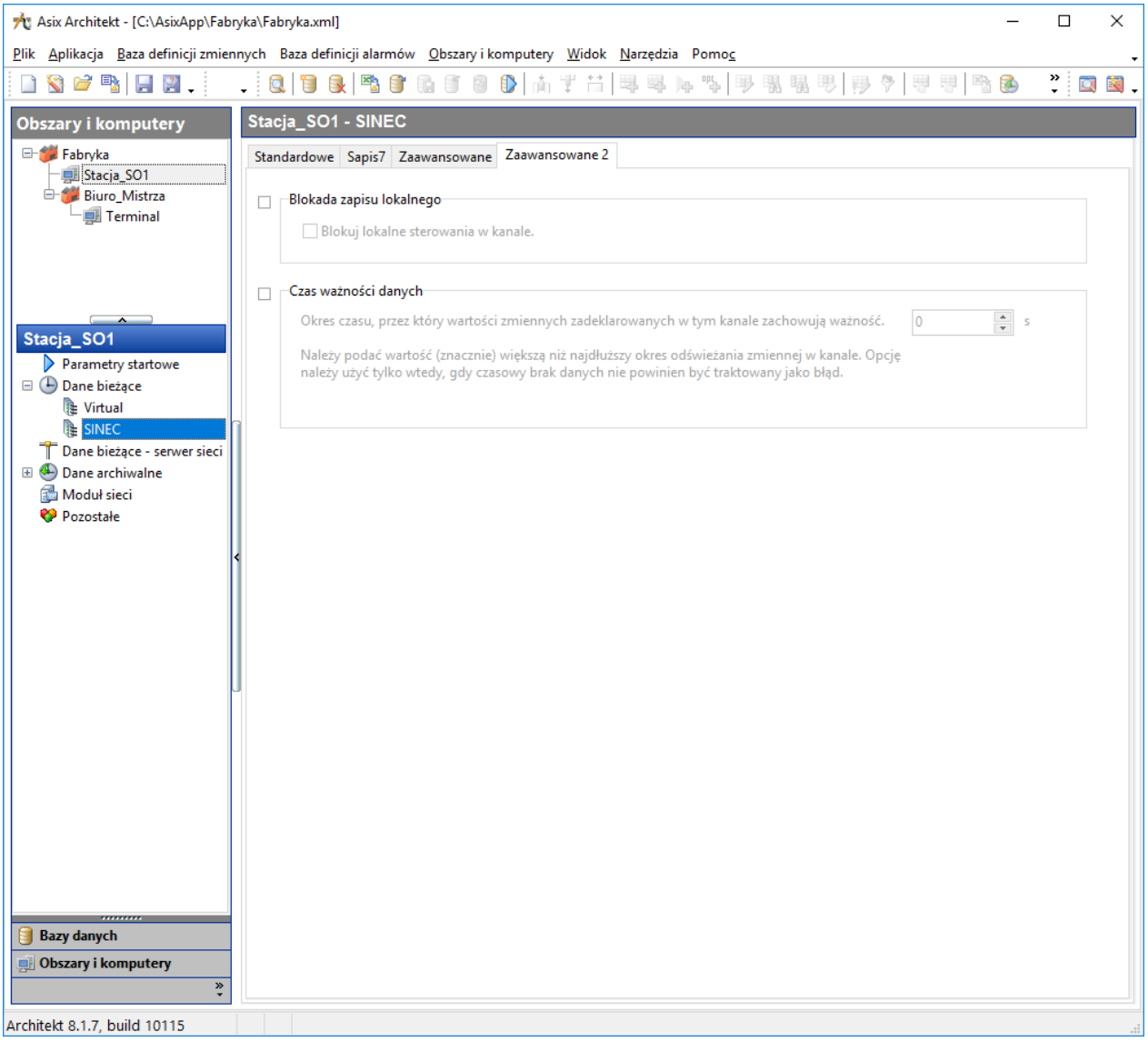
Rys. Widok okna parametryzujących pracę drajwera komunikacyjnego - zakładka 'Zaawansowane'.

- ***Redundancja*** - **(parametr dostępny dla wybranych drajwerów)** deklaracja kanału redundantnego dla kanału logicznego. Opcjonalnie można ograniczyć zakres poszukiwań poprzez zadeklarowanie listy komputerów. W polu należy wpisać nazwę kanału redundantnego, a po przecinku można podać nazwy komputerów, do których ma być ograniczone szukanie.
  Wartość domyślna: domyślnie kanał redundantny jest szukany na wszystkich komputerach sieci

- ***Zezwolenie zapisu*** - deklaracja nazw sieciowych komputerów, które mają uprawnienia do sterowania w danym kanale. Nazwy oddzielić należy przecinkiem.
  Wartość domyślna: domyślnie żaden z komputerów nie ma uprawnień do zapisu.

- ***Ograniczenie odczytu*** - deklaracja nazw sieciowych komputerów, które mają uprawnienia do czytania wartości zmiennych z danego kanału.
Wartość domyślna: domyślnie wszystkie komputery mogą czytać zmienne z tego kanału.

Zakładka ***Zaawansowane 2*** pozwala na parametryzację następujących pozycji:



Rys. Widok okna parametryzującego pracę drajwera komunikacyjnego - zakładka 'Zaawansowane 2'.

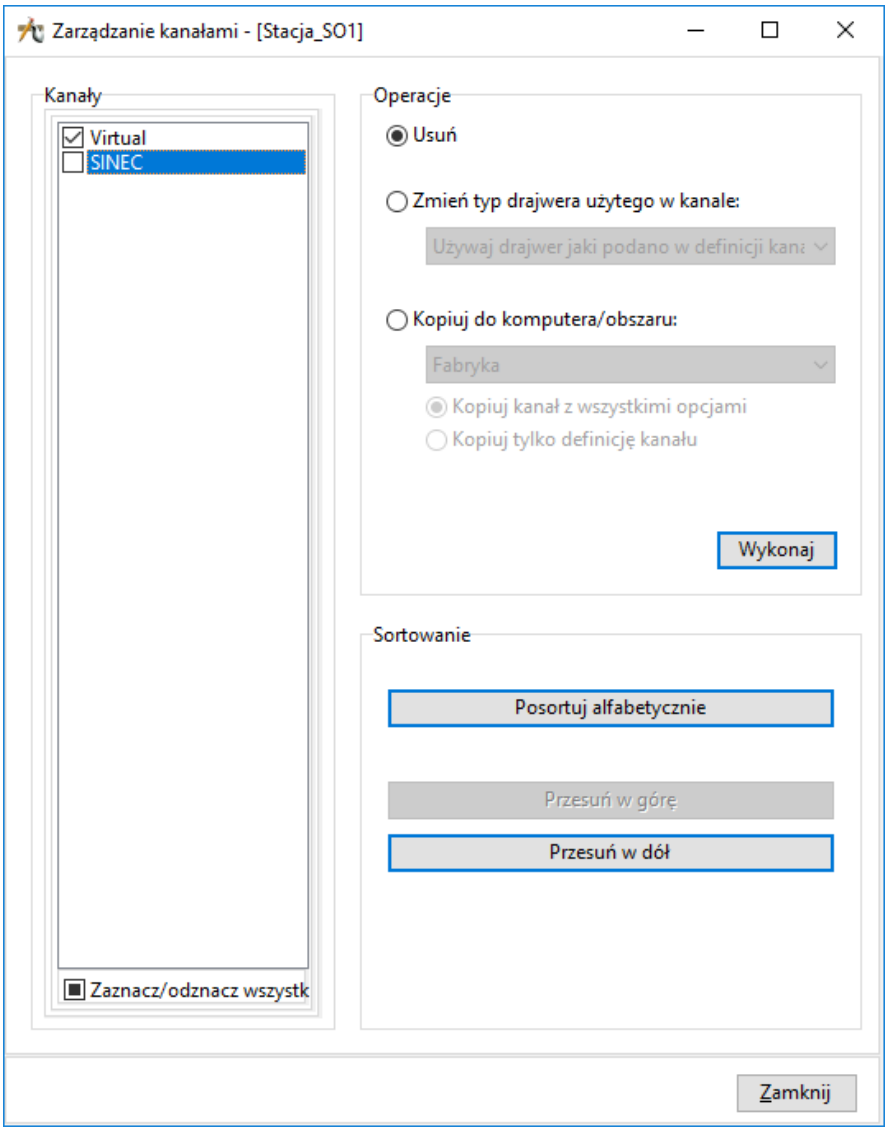
- ***Blokada zapisu lokalnego*** - blokada lokalnych sterowań w kanale logicznym;
Wartość domyślna: nie ma blokady lokalnych sterowań

- ***Czas ważności danych*** - okres czasu (w sekundach), przez który wartości zmiennych procesowych w kanale logicznym zachowują ważność bez względu na zadeklarowaną częstość odświeżania; ważność wartości zmiennych procesowych określa się na podstawie cyklu odświeżania;

  Wartość domyślna: 0

2.8.5. Zarządzanie kanałami

W menu uruchamianym po kliknięciu prawym klawiszem myszki na pozycji ***Dane bieżące*** dostępne jest również polecenie ***Zarządzaj kanałami***, umożliwiające masowe kopiowanie i usuwanie kanałów, zmianę typu drajwera użytego w kanale oraz sortowanie kolejności wyświetlania kanałów w drzewie modułu ***Dane bieżące***.



Rys. Okno 'Zarządzaj kanałami'.

2.9. Konfiguracja parametrów danych bieżących dotyczących serwera sieci

Konfiguracja parametrów z grupy  ***Dane bieżące - serwer sieci*** wiąże się z ustawieniem wartości parametrów konfigurujących sposób wymiany danych poprzez lokalną sieć komputerową.

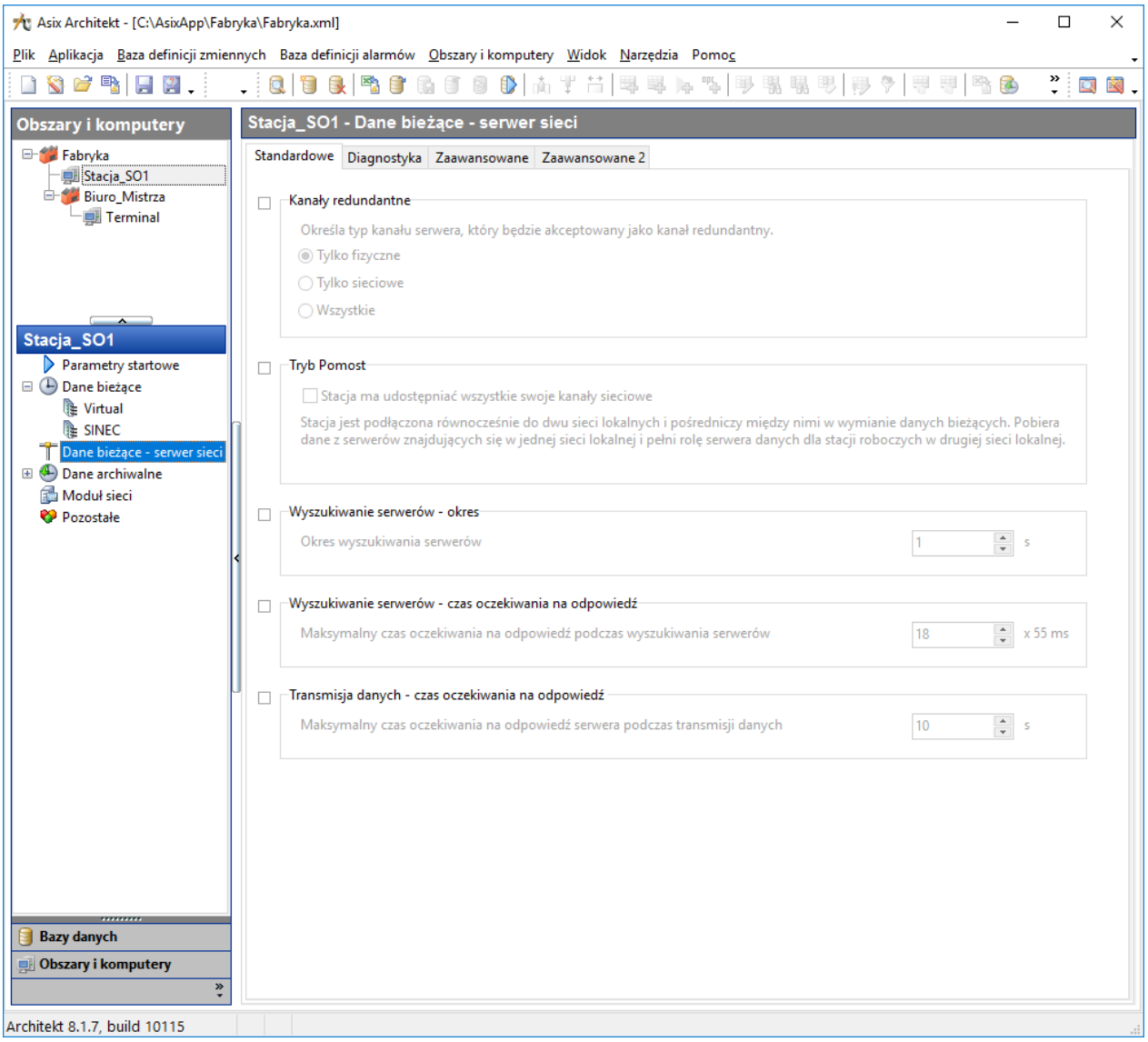
Lokalna sieć komputerowa może być wykorzystana do stworzenia takiej konfiguracji systemu Asix, w której tylko część komputerów (serwery danych) ma bezpośrednie połączenie ze sterownikami, natomiast pozostałe komputery (stacje robocze) korzystają z informacji udostępnianych przez serwery danych za pośrednictwem lokalnej sieci komputerowej.

Kanały transmisji systemu Asix służące do pozyskiwania danych poprzez lokalną sieć komputerową noszą nazwę kanału sieciowego.

Wymiana danych w lokalnej sieci komputerowej pomiędzy stacjami systemu Asix jest realizowana przez moduł NETSRV. Moduł ten jest ładowany automatycznie przy uruchomieniu Asixa.

2.9.1. Dane standardowe

Zakładka ***Standardowe*** obejmuje konfigurację następujących opcji:



Rys. Okno parametrów danych bieżących - serwera sieci - zakładka 'Standardowe'.

- ***Kanały redundantne*** - określa typ kanału serwera, który będzie akceptowany przez dany komputer zdalny jako kanał redundantny; dopuszczalne typy to:
  - FIZYCZNE - tylko kanał fizyczny,
  - SIECIOWE - tylko kanał sieciowy,
  - WSZYSTKIE - dowolny typ kanału;

  Wartość domyślna: FIZYCZNE

- ***Tryb pomost*** - deklaruje pracę stacji roboczej w trybie POMOST. W konfiguracjach, w których stacja robocza systemu Asix jest pomostem pomiędzy dwiema lokalnymi sieciami komputerowymi, kanały sieciowe stacji roboczej traktowane są na równi z kanałami fizycznymi w zakresie udostępniania danych w lokalnej sieci komputerowej. Terminale w sieci zakładowej mogą dołączyć się do kanałów sieciowych pomostu.
  Wartość domyślna: tryb POMOST nie jest zadeklarowany

- ***Wyszukiwanie serwerów - okres*** - pozwala określać odstęp czasu (w sekundach) pomiędzy operacjami poszukiwania serwerów;

Wartość domyślna: 1 sekunda

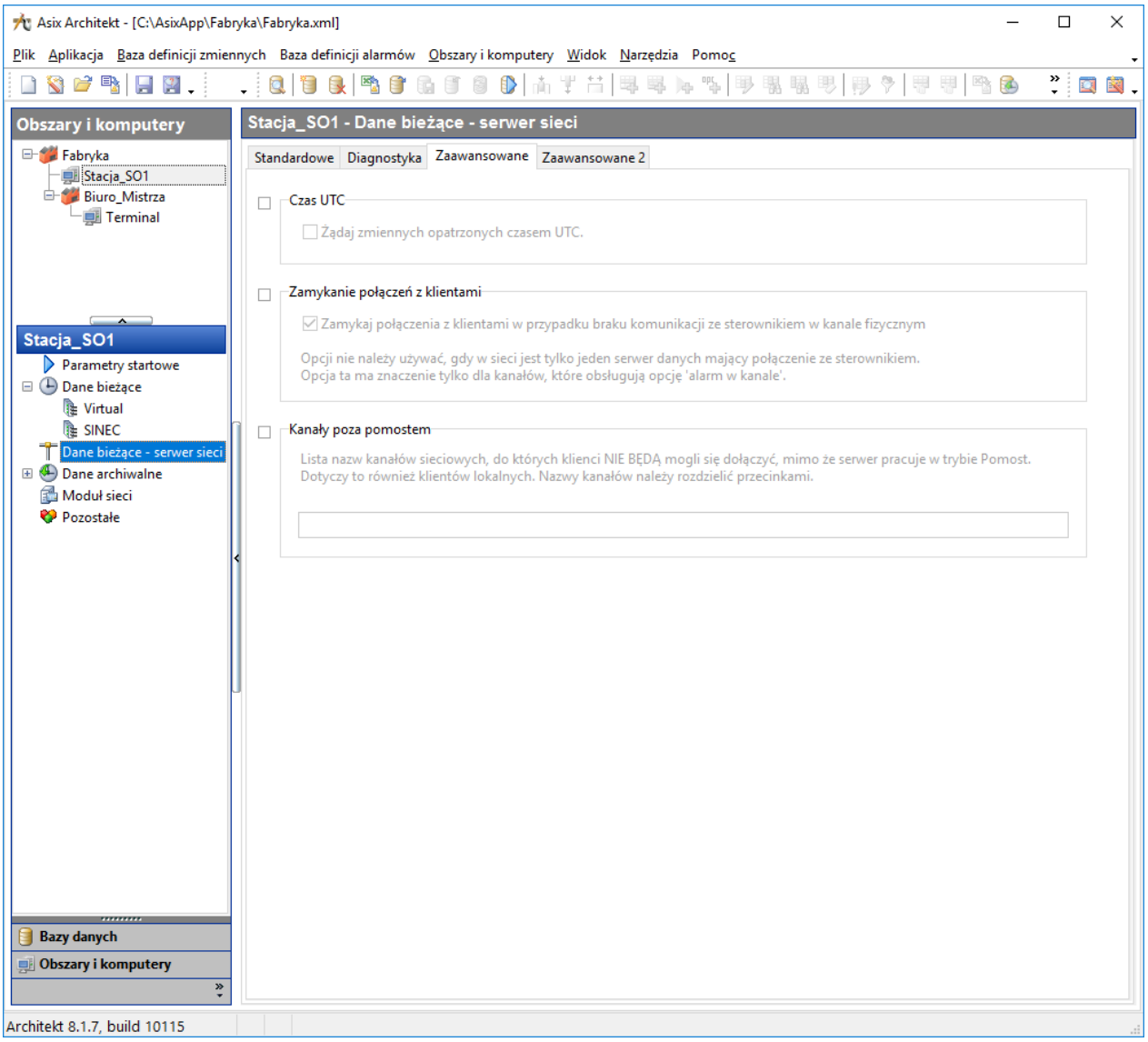
- ***Wyszukiwanie serwerów - czas oczekiwania na odpowiedź -*** deklaracja maksymalnego okresu oczekiwania na odpowiedź podczas wyszukiwania serwerów; okres podany jest jako liczba taktów o czasie trwania 55 ms każdy.
Wartość domyślna: 18 (odpowiada jednej sekundzie)

- ***Transmisja danych - czas oczekiwania na odpowiedź -*** deklaruje maksymalny czas oczekiwania na odpowiedź serwera podczas transmisji danych (w sekundach);
Wartość domyślna: 10 sekund

2.9.2. Dane zaawansowane

 Zakładka *Zaawansowane* i *Zaawansowane2* pojawia się po wybraniu polecenia *Widok > Pokaż opcje zaawansowane* z menu okna modułu Architekt.

Zakładki ***Zaawansowane*** i ***Zaawansowane2:***



Rys. Okno parametrów danych bieżących - serwera sieci - zakładka 'Zaawansowane'.

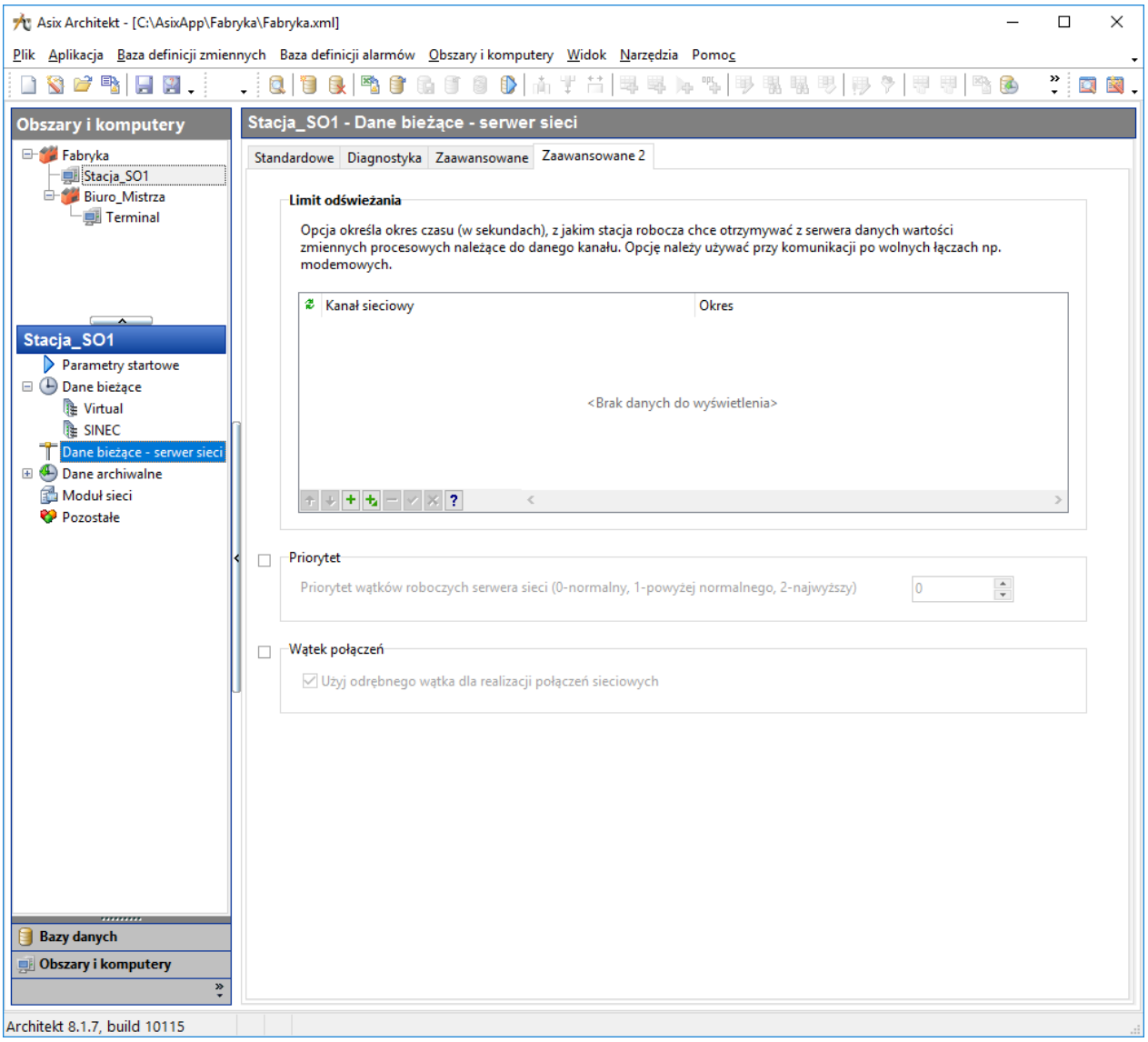
- ***Czas UTC*** - żądanie przekazywania wartości zmiennych opatrzonych czasem UTC.

Wartość domyślna: opcja wyłączona

- ***Zamykanie połączeń z klientami*** - deklaruje, czy stacja serwera będzie automatycznie zrywała połączenia ze stacjami zdalnymi w przypadku błędów komunikacji stwierdzonych w fizycznych kanałach komunikacyjnych stacji serwera; opcji nie należy używać, gdy w sieci jest tylko jeden serwer danych mający połączenie ze sterownikiem; opcja ta ma znaczenie tylko dla kanałów, które obsługują opcję *Alarm w kanale*.

Wartość domyślna: opcja włączona

- ***Kanały poza pomostem*** - lista zawiera spis kanałów sieciowych, rozdzielonych przecinkami, do których klienci NIE BĘDĄ mogli się dołączyć; dotyczy to również klientów lokalnych;



Rys. Okno parametrów danych bieżących - serwera sieci - zakładka 'Zaawansowane2'.

- ***Limit odświeżania*** **-** określa odstęp czasu (w sekundach), z jakim stacja robocza chce otrzymywać z serwera danych wartości zmiennych procesowych należących do kanału o nazwie zdefiniowanej w polu *Kanał*. Zaleca się używać tę opcję przy komunikacji po wolnych łączach.

  Wartość domyślna: wartości zmiennych procesowych są rozsyłane z okresem zgodnym z deklaracją zmiennej procesowej na serwerze danych

- ***Priorytet*** *- deklaracja priorytetów wątków NETSRVa:*
  - *0 - THREAD_PRIORITY_NORMAL,*
  - *1 - THREAD_PRIORITY_ABOVE_NORMAL,*
  - *2 - THREAD_PRIORITY_HIGHEST.*

  Wartość domyślna: 0

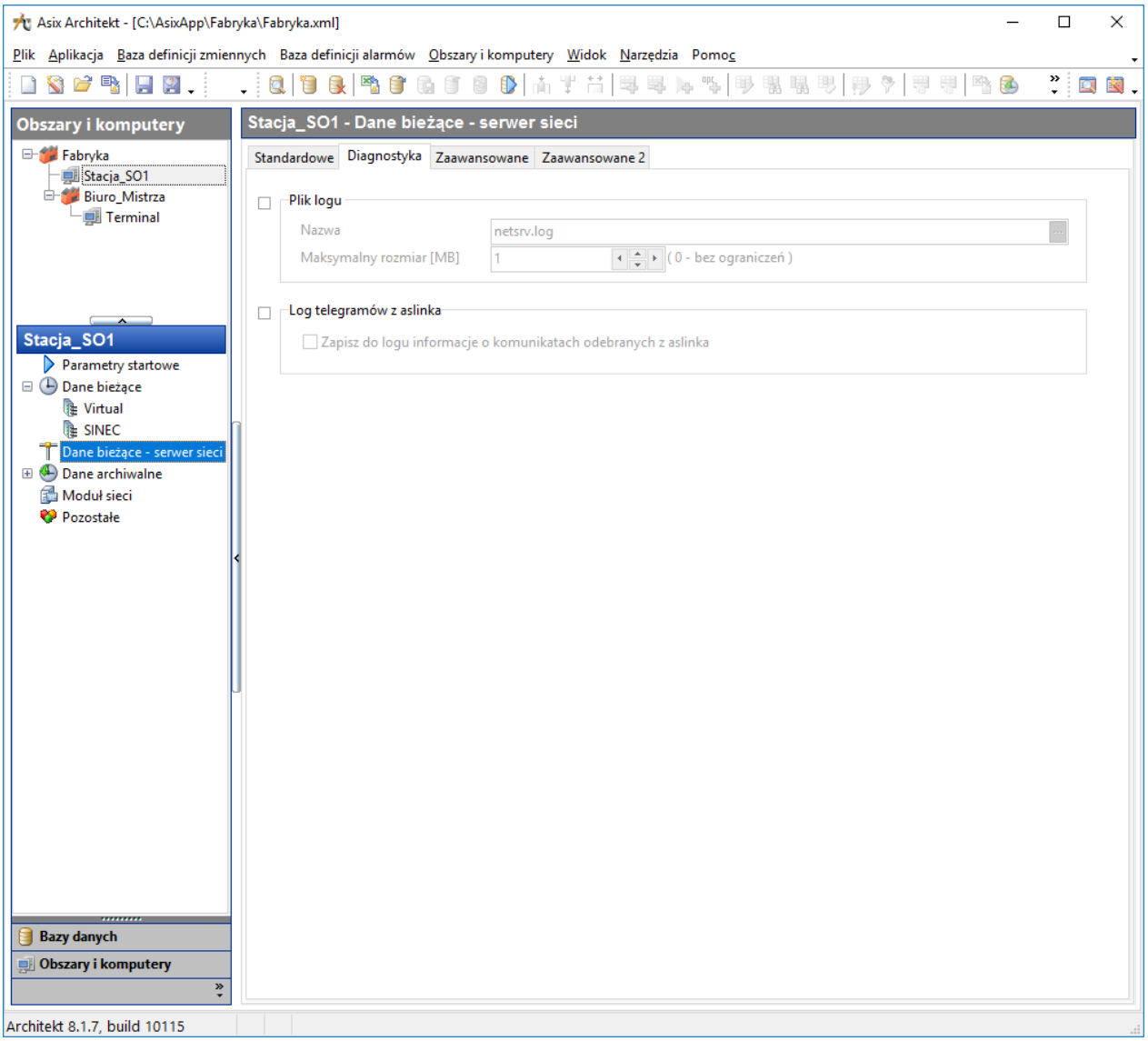
- ***Wątek połączeń*** **-** deklaracja użycia odrębnego wątku dla realizacji połączeń sieciowych;

  Wartość domyślna: opcja włączona

2.9.3. Parametry diagnostyczne

Zakładka *Diagnostyka* pojawia się po wybraniu polecenia *Widok* > *Pokaż opcje zaawansowane* z menu okna modułu Architekt.

Zakładka ***Diagnostyka:***



Rys. Okno parametrów danych bieżących - serwera sieci - zakładka Diagnostyka.

- ***Plik logu*** - deklaracja nazwy pliku logu z komunikatami diagnostycznymi NETSRV;

  - ***Nazwa*** - Wartość domyślna: plik logu nie jest tworzony

- ***Maksymalny rozmiar*** - wartość domyślna: 1 MB; po przekroczeniu rozmiaru zadeklarowanego w opcji dotychczasowa zawartość pliku jest kasowana

Szczegóły na temat wymiana danych poprzez lokalną sieć komputerową można znaleźć w podręczniku *"Asix - podręcznik użytkownika"*, rozdziale *Menedżer komunikacyjny ASMEN, Wymiana danych poprzez lokalną sieć komputerową.*

- ***Log telegramów z Aslinka -*** deklaracja zapisu komunikatów odbieranych z Aslinka do pliku logu, zadeklarowanego w powyższej pozycji ***Plik logu***; Wartość domyślna: opcja jest wyłączona

2.10. Konfiguracja danych archiwalnych

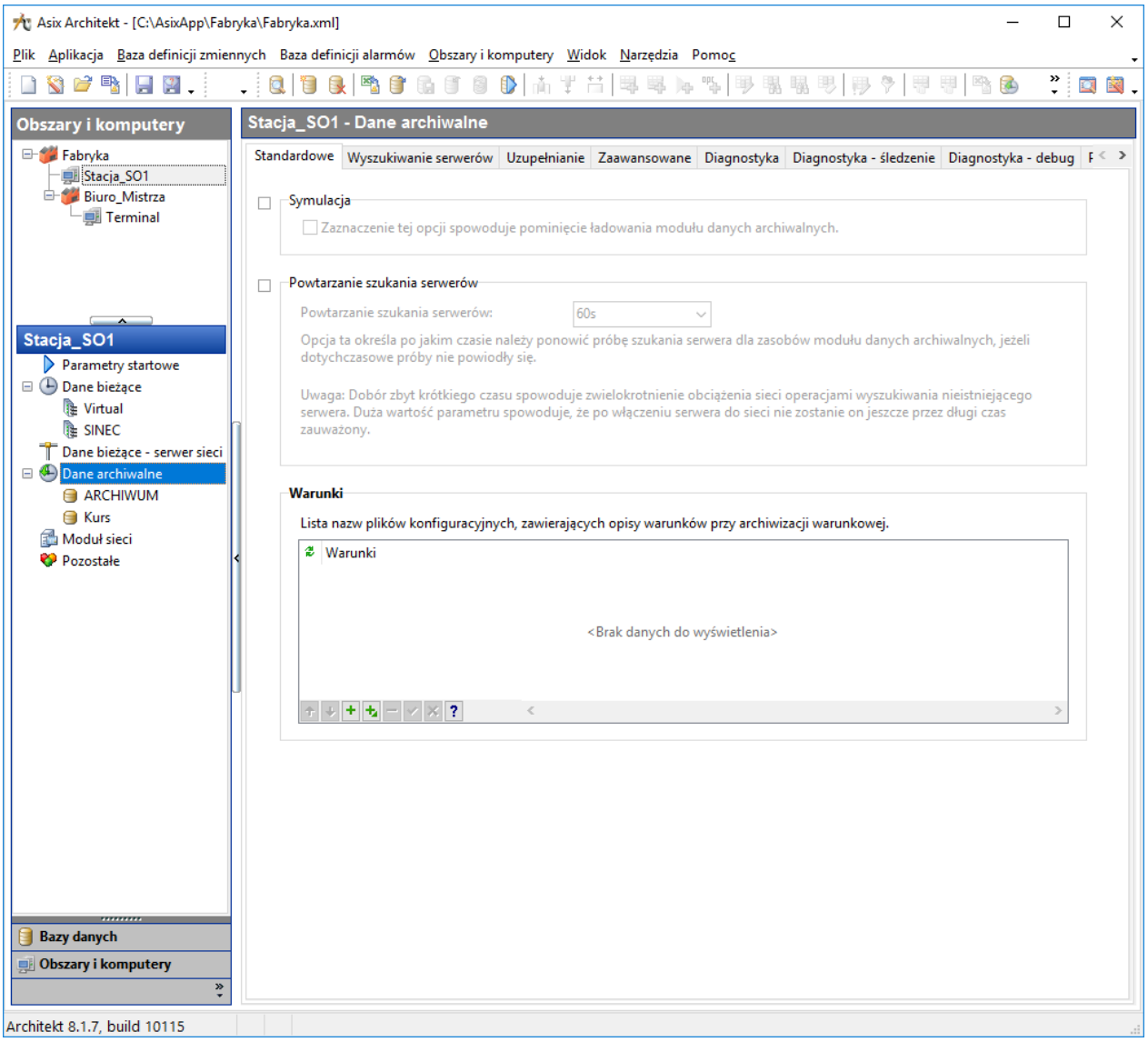
Grupa parametrów ***Dane archiwalne*** obejmuje ustawienia pracy modułu Aspad, w tym parametry rejestracji i późniejszego odtwarzania przebiegów czasowych zmiennych.

Moduł Aspad pozwala na archiwizację zmiennych każdego typu dozwolonego w module Asmen, pod warunkiem, że są to zmienne skalarne. Mogą to być zarówno zmienne procesowe, jak i uzyskiwane w wyniku przeliczeń lub symulacji. W obecnej wersji nie można archiwizować tablic. Oprócz wartości i czasu przechowywany jest także status wartości zgodny ze standardem OPC. Dane gromadzone są zarówno w plikach o specjalnie zaprojektowanej strukturze (archiwum D, M, Y - podział na pliki odpowiednio dobowe, miesięczne lub roczne), jak i w bazach danych (archiwum B).

Szczegółowy opis możliwości funkcjonalnych modułu Aspad znajduje się w części Serwer Danych > 1. Obsługa danych procesowych Serwera Danych > rozdziale *1.3. Aspad - program archiwizacji danych*.

2.10.1. Dane standardowe

Zakładka ***Standardowe*** pozwala określić następujące podstawowe parametry archiwizacji:



Rys. Okno parametrów danych archiwalnych - zakładka 'Standardowe'.

- ***Symulacja*** - zaznaczenie tej opcji spowoduje pominięcie ładowania Aspada;
  Wartość domyślna: Aspad jest ładowany

- ***Powtarzanie szukania serwerów*** - określa po jakim czasie należy ponowić próbę szukania serwera dla zasobów Aspada, jeżeli dotychczasowe próby nie powiodły się;
  Wartość domyślna: 60 sekund.

Dobór zbyt krótkiego czasu spowoduje zwielokrotnienie obciążenia sieci operacjami wyszukiwania nieistniejącego serwera. Duża wartość parametru spowoduje, że po włączeniu serwera do sieci nie zostanie on jeszcze przez długi czas zauważony.

- ***Warunki*** - pozwala zdefiniować pliki zawierające warunki w sytuacji stosowania archiwizacji warunkowej.

Szczegółowy opis archiwizacji warunkowej znajduje się w części Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Archiwizacja warunkowa > rozdziale Deklaracje warunków archiwizacji.

Przykłady warunków archiwizacji:

Archiwizuj, gdy bity 4. lub 8. zmiennej AAA zmieniły stan:

```
WARUNEK, BNE, @AAA, $@AAA, 0x110
```

Archiwizuj, gdy bit 0. zmiennej archiwizowanej zmienia stan z 0 na 1:

```
WARUNEK, BGT, @, $@, 0x1
```

Archiwizuj, gdy wartość zmiennej archiwizowanej jest poza przedziałem (20,80) i ustawiony jest jeden z bitów 0-3 zmiennej WWW:

```
W1, IN, @, <<, 20, 80, >>
  W2, BAND, @WWW, 0xF
  WARUNEK, AND, ?W1, ?W2
```

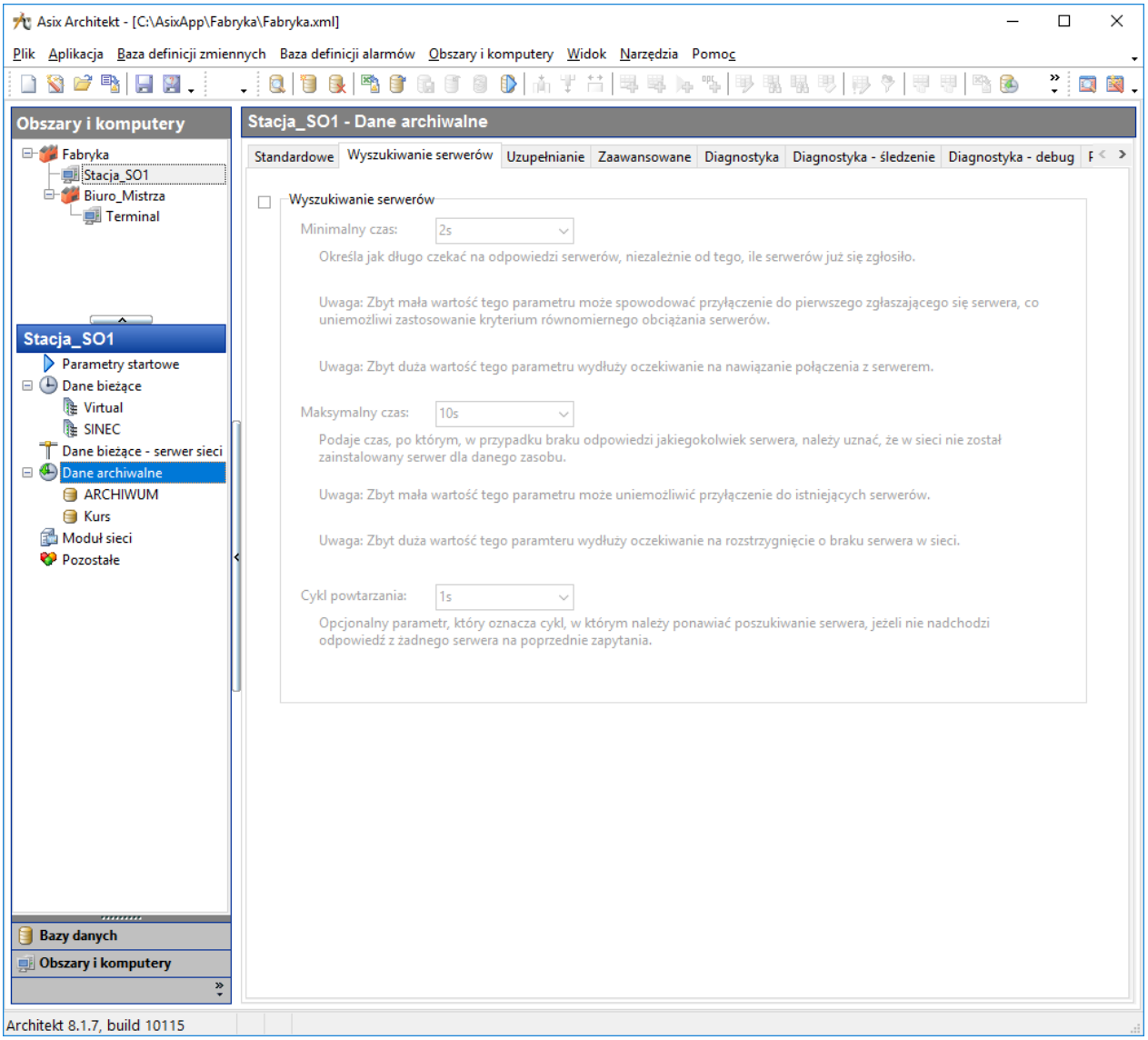
Archiwizuj w chwili, gdy wartość zmiennej archiwizowanej przekracza granicę 100:

```
W3, GT, @, 100
  WARUNEK, AND, ?W3, !$?W3
```

2.10.2. Deklaracja czasu oczekiwania na zgłoszenie serwerów

Zakładka *Serwer profile* pojawia się po wybraniu polecenia *Widok* > *Pokaż opcje zaawansowane* z menu okna modułu Architekt.

Zakładka ***Wyszukiwanie serwerów*** obejmuje następujące parametry:



Rys. Okno parametrów danych archiwalnych - zakładka 'Wyszukiwanie serwerów'.

- ***Wyszukiwanie serwerów*** **-** określa jak długo należy czekać na zgłoszenia serwerów po zainicjowaniu poszukiwania:
  - ***Minimalny czas*** - określa jak długo czekać na odpowiedzi serwerów, niezależnie od tego, ile serwerów już się zgłosiło.
  - ***Maksymalny czas*** - podaje czas, po którym, w przypadku braku odpowiedzi jakiegokolwiek serwera, należy uznać, że w sieci nie został zainstalowany serwer dla danego zasobu.
  - ***Cykl powtarzania*** - opcjonalny parametr, który oznacza cykl, w którym należy ponawiać poszukiwanie serwera, jeżeli nie nadchodzi odpowiedź z żadnego serwera na poprzednie zapytania.
    Wartość domyślna: odpowiednio 2 s, 10 s, 1 s

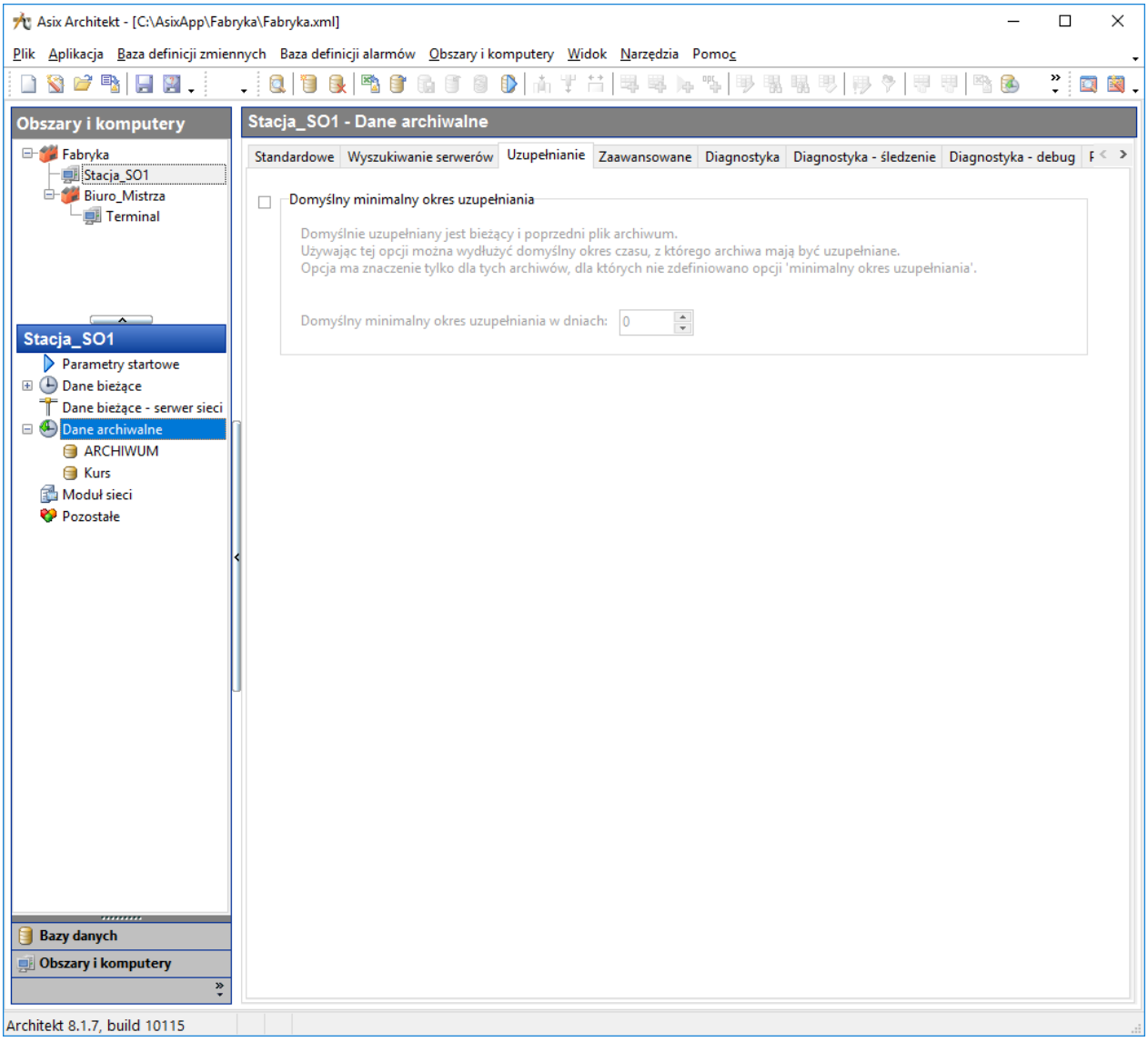
Zbyt mały parametr *Minimalny czas* może spowodować przyłączenie do pierwszego zgłaszającego się serwera, co uniemożliwi zastosowanie kryterium równomiernego obciążania serwerów.

Zbyt duży parametr *Minimalny czas* wydłuży oczekiwanie na nawiązanie połączenia z serwerem.

Zbyt mały parametr *Maksymalny czas* może uniemożliwić przyłączenie do istniejących serwerów.
Zbyt duży parametr *Maksymalny czas* wydłuży oczekiwanie na rozstrzygnięcie o braku serwera w sieci.

2.10.3. Parametryzacja okresu uzupełniania danych

Zakładka ***Uzupełnianie*** obejmuje następujące parametry:

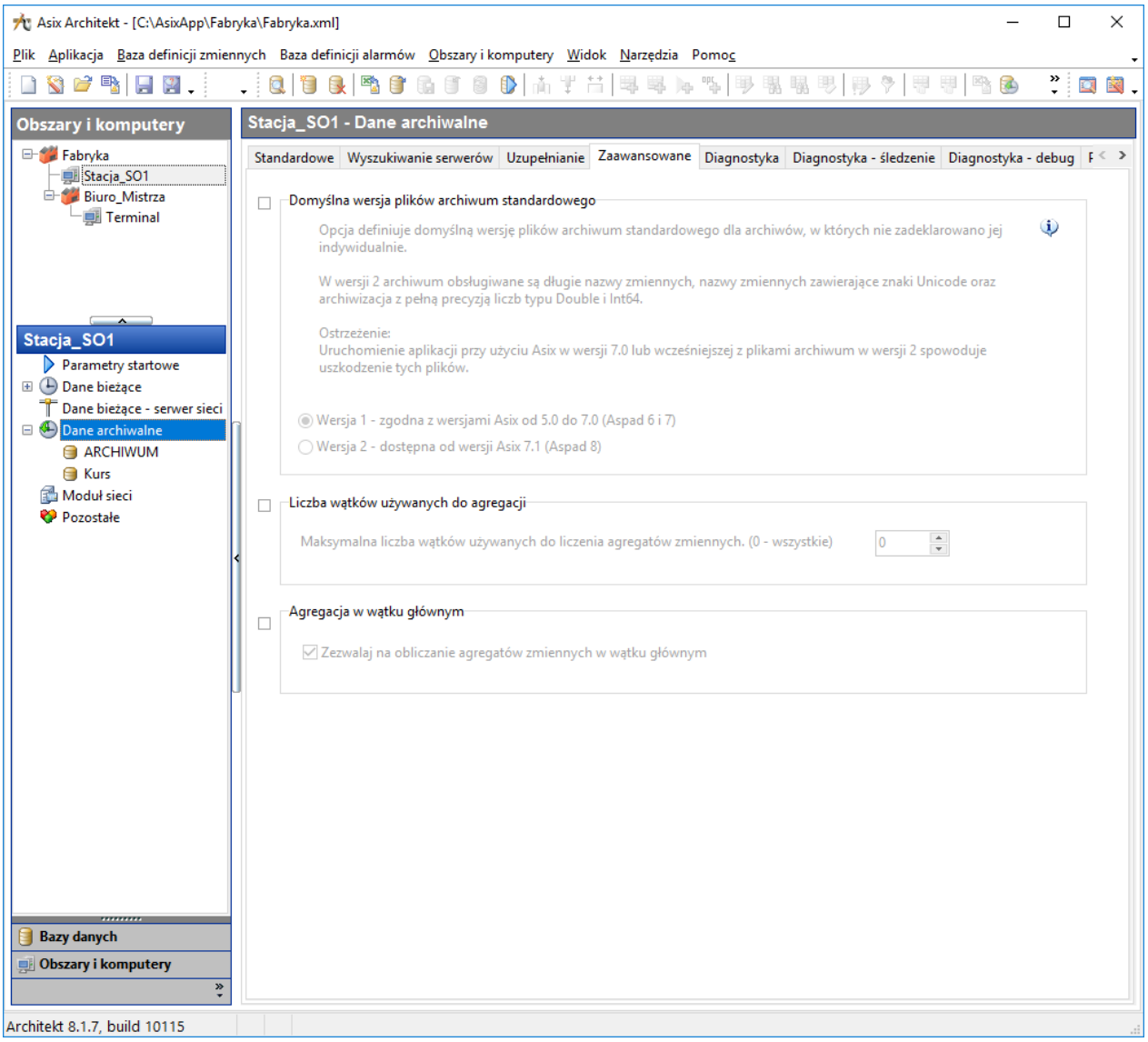


Rys. Widok okna parametryzującego okresu uzupełniania danych.

- ***Domyślny minimalny okres uzupełniania*** - używając tej opcji można wydłużyć domyślny okres czasu (w dniach), z którego archiwa mogą być uzupełniane. Opcja ma znaczenie tylko dla tych archiwów, dla których nie zdefiniowano opcji ***Minimalny okres uzupełniania***.

2.10.4. Deklaracja wersji plików archiwum standardowego

Zakładka ***Zaawansowane*** obejmuje następujące parametry:



Rys. Widok okna deklarującego wersje plików archiwum standardowego.

- ***Domyślna wersja plików archiwum standardowego*** - opcja definiuje domyślną wersję plików archiwum standardowego dla archiwów, w których nie zadeklarowano jej indywidualnie:

  Wersja 1 - zgodna z wersjami Asix od 5.0 do 7.0 (Aspad 6 i 7).

  Wersja 2 - dostępna od wersji Asix 7.1 (Aspad 8).

  W wersji 2 archiwum obsługiwane są długie nazwy zmiennych, nazwy zmiennych zawierające znaki UNICODE oraz archiwizacja z pełną precyzją liczb typu Double i Int64.

  OSTRZEŻENIE: Uruchomienie aplikacji przy użyciu Asixa w wersji 7.0 lub wcześniejszej z plikami archiwum w wersji 2 spowoduje uszkodzenie tych plików.

  INFORMACJE DODATKOWE:

  Nie wolno kopiować plików archiwum w wersji 2 do pakietów Asix wcześniejszych ni 7.1.

  Odczyt sieciowy archiwum w wersji 2 przez wcześniejsze wersje pakietu Asix podlega pewnym ograniczeniom:

  1. Nie są dostępne zmienne o długich nazwach.
  2. Funkcje uzupełniania danych, replikacji archiwum, obiekt WYKRES oraz skrypty VBSScript i JScript w Asixie klasycznym:

  - nie działają w pakiecie Asix 5.2 lub wcześniejszych,
  - działają w pakietach Asix 6 i Asix 7.0 tylko pod warunkiem aktualizacji modułu Aspad, przy czym wydajność odczytu jest wtedy zmniejszona w stosunku do archiwum w wersji 1.

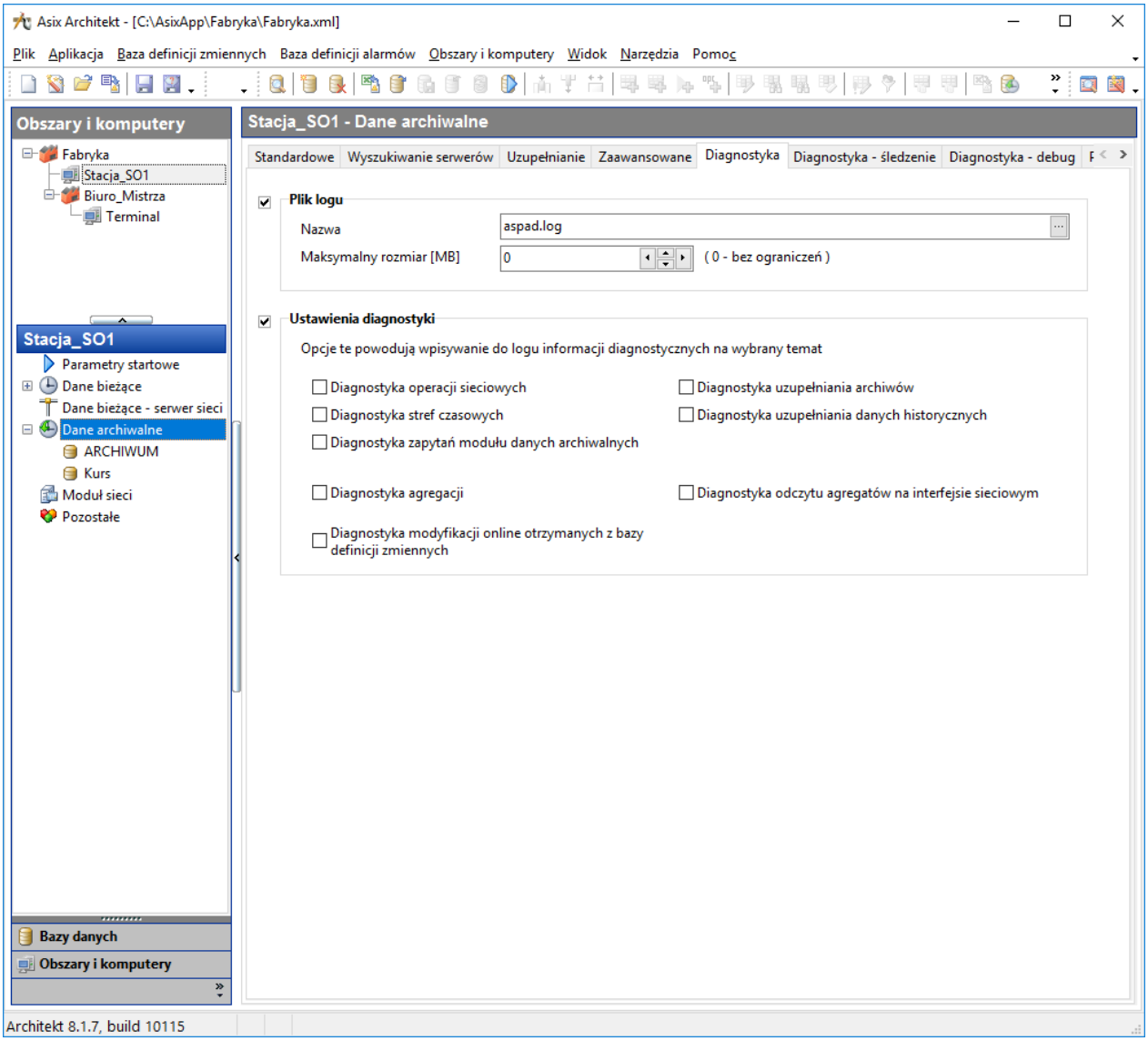
- ***Liczba wątków używanych do agregacji*** - opcja definiuje maksymalna liczbę wątków procesowa do liczenia agregatów zmiennych (0-wszystkie).

- ***Agregacja w wątku głównym*** - opcja deklaruje obliczanie agregatów zmiennych w głównym wątku procesora.

2.10.5. Parametry diagnostyczne

Zakładka *Diagnostyka* pojawia się po wybraniu polecenia *Widok > Pokaż opcje zaawansowane* z menu okna modułu Architekt.

Zakładka ***Diagnostyka*** określa:



Rys. Okno parametrów danych archiwalnych - zakładka 'Diagnostyka'.

- ***Nazwa pliku logu*** - określenie pliku, do którego ASPAD będzie wpisywał informacje diagnostyczne;
  Wartość domyślna: domyślnie tworzony jest plik logu aspad.log

- ***Ustawienia diagnostyki*** - pozwala określić, jakiego rodzaju informacje diagnostyczne będą zapisywane w pliku logu:

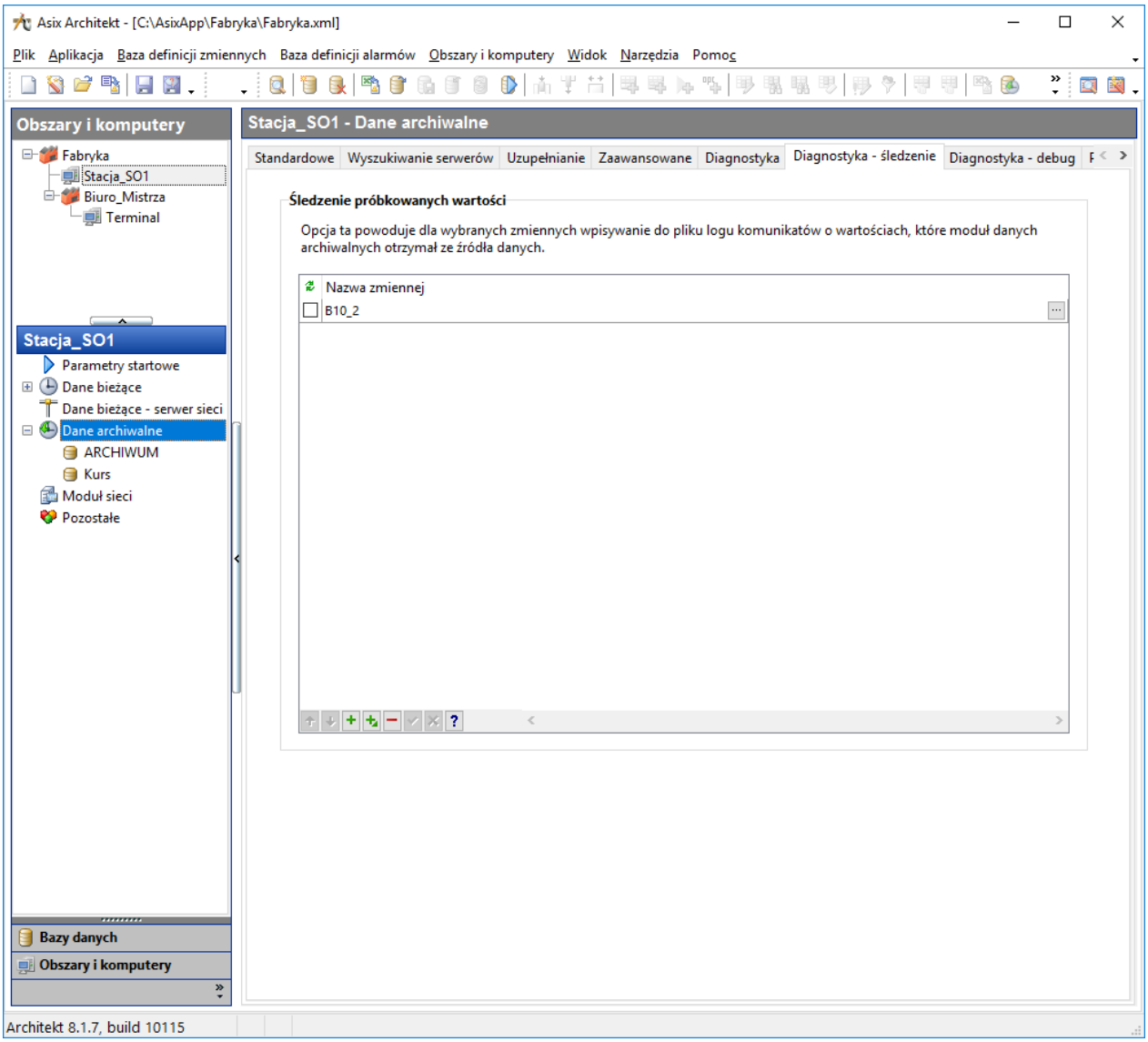
  - ***Diagnostyka operacji sieciowych***,
  - ***Diagnostyka stref czasowych,***
  - ***Diagnostyka zapytań modułu danych archiwalnych***,
  - ***Diagnostyka agregacji*** - pozwala załączyć diagnostykę agregacji;
  - ***Diagnostyka modyfikacji online otrzymanych z bazy definicji zmiennych***;
  - ***Diagnostyka uzupełniania archiwów***;
  - ***Diagnostyka uzupełniania danych historycznych***;

- ***Diagnostyka odczytu agregatów na interfejsie sieciowym*** - określenie pliku, do którego ASPAD będzie wpisywał informacje diagnostyczne;
- ***Diagnostyka modyfikacji on-line otrzymanych z bazy definicji zmiennych***;

Wartość domyślna: opcja wyłączona

2.10.6. Śledzenie próbkowanych wartości

Zakładka ***Diagnostyka - śledzenie*** określa:



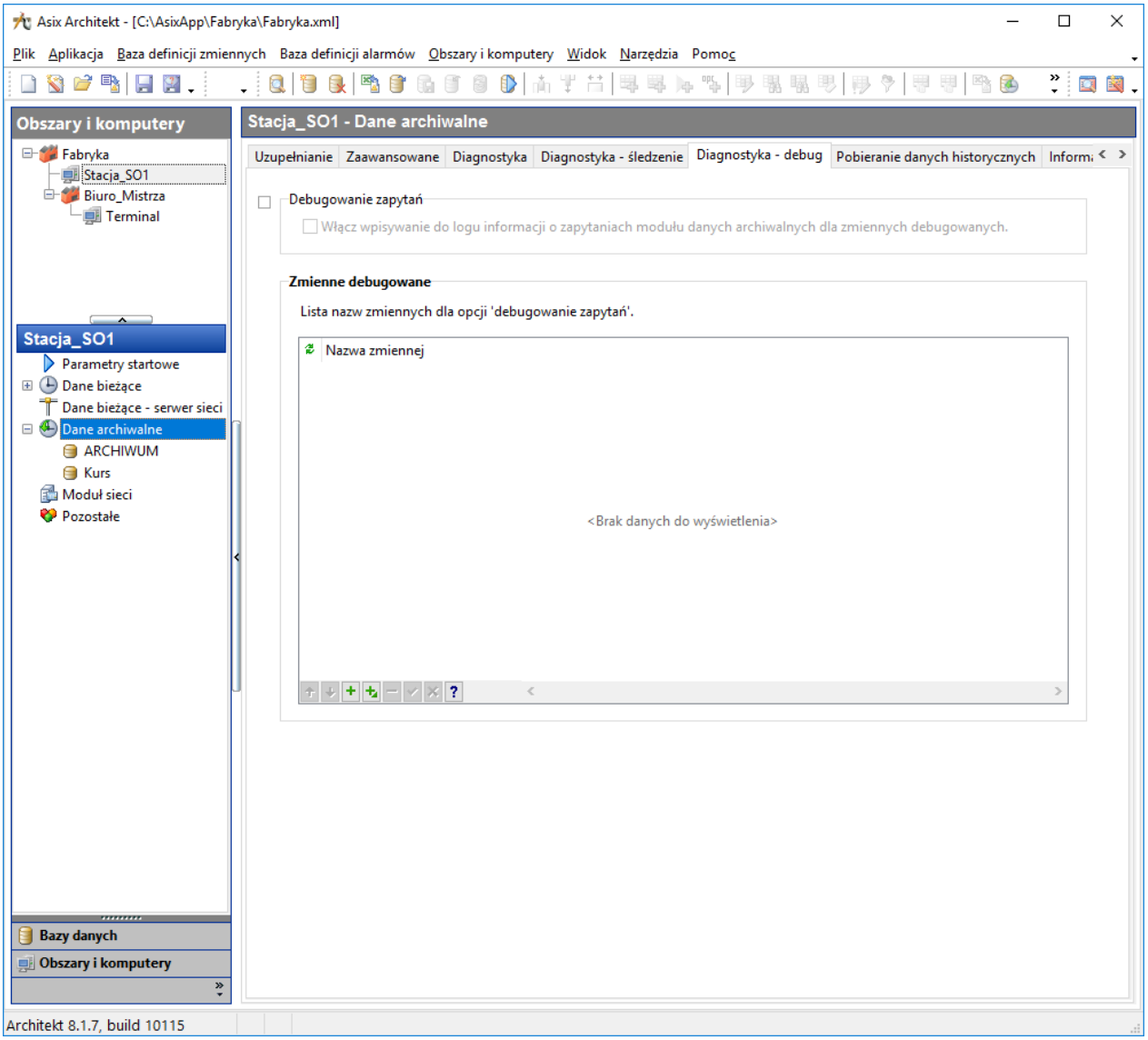
Rys. Okno parametrów danych archiwalnych - zakładka 'Diagnostyka - śledzenie'.

- ***Śledzenie próbkowanych wartości*** - opcja powoduje, że dla zadanej zmiennej wyprowadzane są do pliku logu (zadanego w deklaracji *Nazwa pliku logu*) komunikaty o wartościach, które moduł danych archiwalnych ASPAD otrzymał ze źródła danych;
  - Podanie samej nazwy zmiennej powoduje śledzenie wszystkich zdarzeń dla danej zmiennej i jej agregatów;
  - Podanie nazwy i typu archiwum oddzielonych przecinkiem (np. "A000,D") powoduje śledzenie tylko podanego typu archiwum zmiennej, bez śledzenia innych archiwów i agregatów;
  - Podanie nazwy z pełną specyfikacją agregatu (tak jak pojawia się w logach, np. " A000,M-1m-Avg) powoduje śledzenie tylko tego agregatu.
  Wartość domyślna: brak zadeklarowanych pozycji

 W obecnej wersji można w ten sposób śledzić tylko zmienne typu zmiennoprzecinkowego.

2.10.7. Debugowanie zapytań

Zakładka ***Diagnostyka - debug*** określa:



Rys. Okno parametrów danych archiwalnych - zakładka 'Diagnostyka - debug'.

- ***Debugowanie zapytań*** - opcja załącza wpisywanie do logu informacji o zapytaniach modułu sieciowego dla zmiennych debugowanych.
  Wartość domyślna: opcja wyłączona

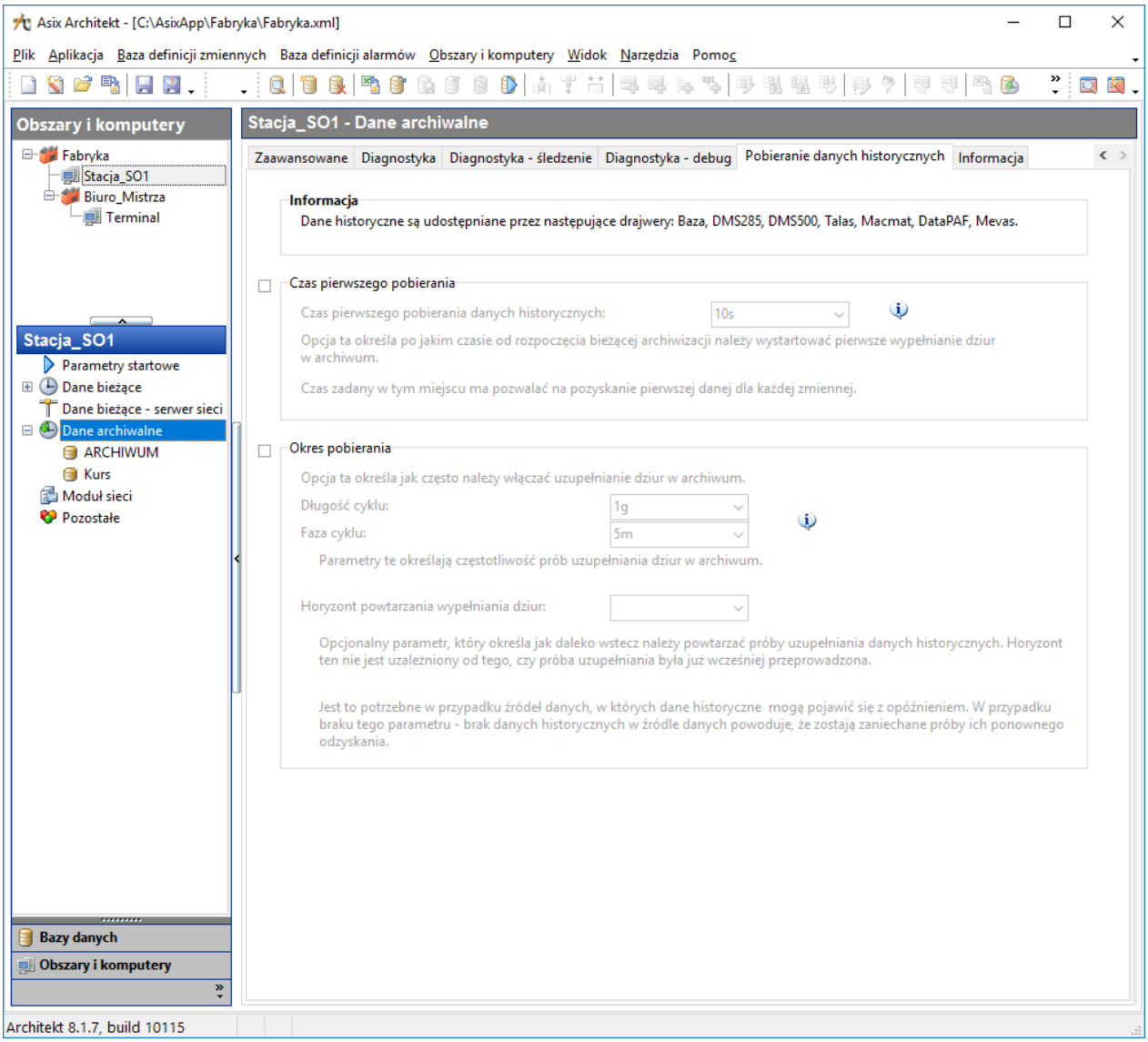
- ***Zmienne debugowane*** - opcja umożliwia deklarowanie zmiennych debugowanych.
  - Podanie samej nazwy zmiennej powoduje śledzenie wszystkich zdarzeń dla danej zmiennej i jej agregatów.
  - Podanie nazwy i typu archiwum oddzielonych przecinkiem (np. "A000,D") powoduje śledzenie tylko podanego typu archiwum zmiennej, bez śledzenia innych archiwów i agregatów.

- Podanie nazwy z pełną specyfikacją agregatu (tak jak pojawia się w logach, np. " A000,M-1m-Avg) powoduje śledzenie tylko tego agregatu.
Wartość domyślna: brak zadeklarowanych zmiennych

2.10.8. Parametryzacja cyklicznego pobierania danych historycznych dla wybranych drajwerów komunikacyjnych

Niektóre drajwery systemu Asix umożliwiają pobieranie przez aplikację brakujących danych historycznych z urządzeń (które same w sobie archiwizują wartości pomiarów) lub z zewnętrznej bazy danych. Dane historyczne udostępniane są przez następujące drajwery: Baza, DMS285, DMS500, Talas, Macmat, DataPAF, Mevas.

Parametryzacja sposobu pobierania danych historycznych odbywa się przy użyciu opcji znajdujących się na zakładce ***Pobieranie danych historycznych***:



Rys. Okno parametrów danych archiwalnych - zakładka 'Pobieranie danych historycznych'.

- ***Czas pierwszego pobierania -*** opcja ta określa, po jakim czasie od rozpoczęcia bieżącej archiwizacji należy wystartować pierwsze wypełnianie dziur;
  Wartość domyślna: 10 sekund

W procesie uzupełniania danych potrzebna jest informacja o tym, jaka jest długość dziury w archiwum wynikłej z ostatniego wyłączenia archiwizacji; dla uzyskania tej informacji niezbędne jest pozyskanie pierwszego punktu po tej dziurze; czas zadany w tym miejscu ma pozwalać na pozyskanie pierwszej danej dla każdej zmiennej.

- ***Okres pobierania -*** pozwala określić, jak często należy włączać uzupełnianie dziur w archiwum:

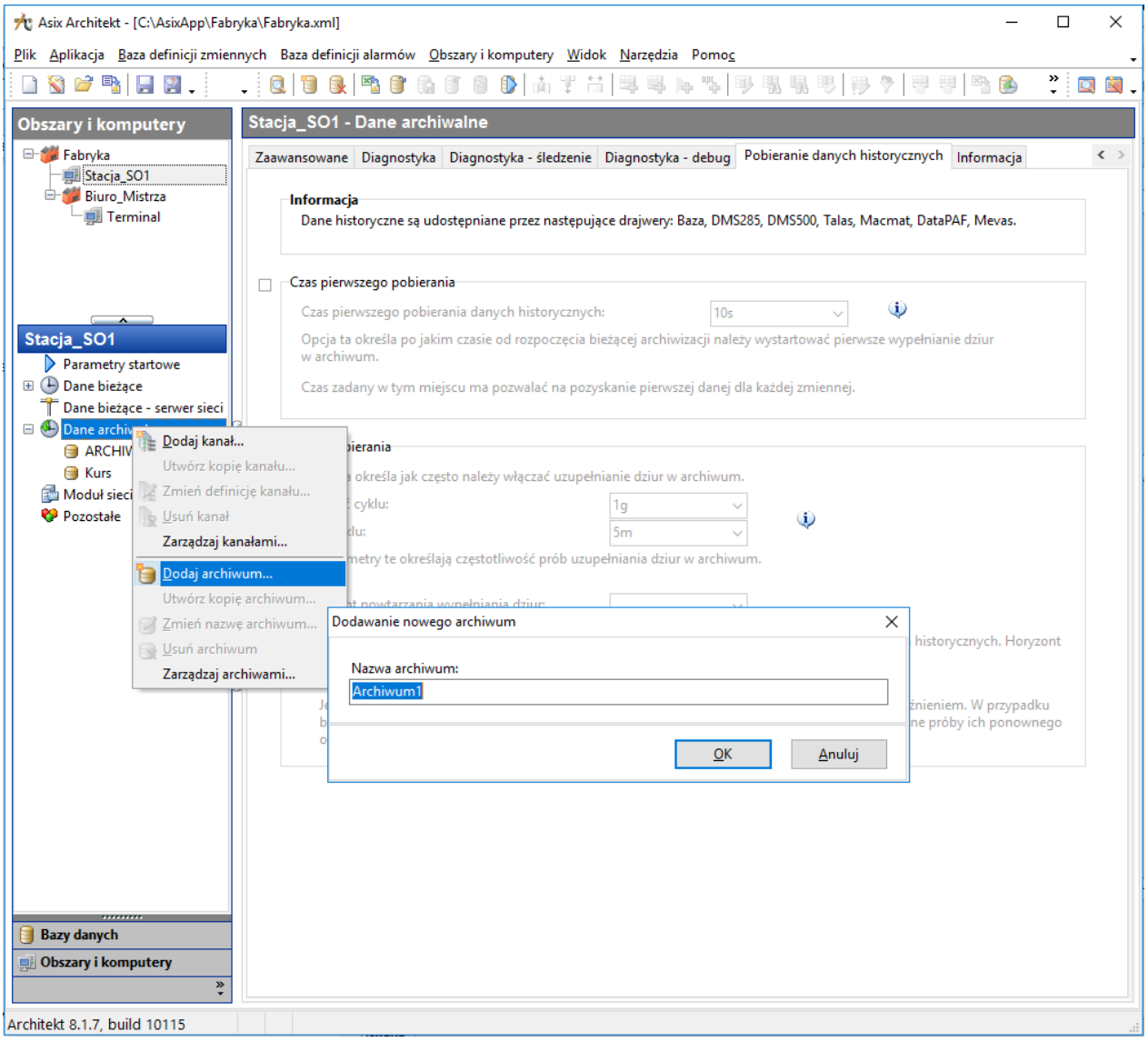
  - ***Długość cyklu*** i ***Faza cyklu*** - są to parametry, które określają częstotliwość prób uzupełniania dziur w archiwum – np.: 5m, 30s oznacza, że 30 sekund po każdych pełnych pięciu minutach nastąpi próba wypełniania dziur w archiwum;
    Wartość domyślna: dla długości cyklu równej 1 godzina i fazy cyklu 5 minut uzupełnianie archiwum będzie miało miejsce 5 minut po każdej pełnej godzinie; długość cyklu liczona jest według czasu UTC.

  - ***Horyzont wypełniania dziur*** - to dodatkowy, opcjonalny parametr, który określa jak daleko wstecz należy próbować uzupełnić dane historyczne; horyzont ten nie jest uzależniony od tego, czy próba uzupełniania była już wcześniej przeprowadzona; jest to potrzebne w przypadku źródeł danych, w których dane historyczne mogą pojawić się z opóźnieniem; w przypadku braku tego parametru - brak danych historycznych w źródle danych powoduje, że zostają zaniechane próby ich ponownego odzyskania;
    Wartość domyślna: brak

2.10.9. Parametryzacja archiwum

W przypadku stosowania archiwizacji wartości zmiennych niezbędne jest zadeklarowanie archiwum.

Archiwum deklarowane jest za pomocą polecenia ***Dodaj archiwum...*** po kliknięciu prawym klawiszem myszki na pozycji ***Dane archiwalne***. Pojawia się wówczas okno *'Dodawanie nowego archiwum'*, gdzie należy określić nazwę archiwum.



Rys. Wywołanie okna 'Dodawanie nowego archiwum'.

Aspad gromadzi dane w plikach zawierających przebiegi z jednego okresu dobowego, miesięcznego lub rocznego odpowiednio do typu archiwum:

Archiwum typu D (ang. day) - dane gromadzone w plikach dobowych;
Archiwum typu M (ang. month) - dane gromadzone w plikach miesięcznych;
Archiwum typu Y (ang. year) - dane gromadzone w plikach rocznych;
Archiwum typu H (ang. horizon) - dane gromadzone w jednym pliku;
Archiwum typu B (Baza Danych) - dane gromadzone w typowych bazach danych;
Archiwum typu P (ang. pattern) - dane wzorcowe;
Archiwum typu X - dane gromadzone w bazie MS SQL;

Patrz:
część *Serwer Danych* > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Ogólna charakterystyka programu Aspad > rozdział *1.3.1.3. Podstawowe rodzaje i typy archiwum Aspada*,

część Serwer Danych > Obsługa danych procesowych Serwera Danych > Program archiwizacji danych Aspad > Archiwizacja typu B.

Podstawowym rodzajem archiwum jest **STANDARD**. Archiwum to jest zapisywane w binarnych plikach na dysku.

Drugim rodzajem archiwum jest **MS SQL**. Jest to archiwum zapisywane w bazie danych SQL.

Można jeszcze wyróżnić podział archiwum ze względu na sposób udostępniania danych:

Archiwum do zapisu	- podstawowe archiwum pozwalające na zapis i odczyt danych z plików archiwalnych;
Archiwum do odczytu	- to archiwum pozwalające tylko na odczyt danych z plików archiwalnych;
Archiwum sieciowe	- to archiwum pozwalające na odczyt danych przez sieć z innego komputera;
Archiwum pomocnicze	- to archiwum pozwalające na zapis i odczyt danych, służące jako redundancja archiwum podstawowego na innym komputerze;
Archiwum replikowane	- to archiwum pozwalające na zapis i odczyt danych z archiwum; dane do archiwum pobierane są z archiwum innego komputera.

Aby skonfigurować archiwum przy użyciu modułu Architekt, należy zaznaczyć w drzewie parametrów aplikacji pozycję ***Dane archiwalne*** i kliknąć prawym klawiszem myszki - a następnie wybrać polecenie ***Dodaj archiwum***.

Po kliknięciu na utworzonym archiwum w oknie konfiguracji parametrów pojawią się zakładki, umożliwiające skonfigurowanie danego archiwum.

Archiwum pomocnicze jest to archiwum do zapisu umieszczane na serwerze pomocniczym danego zasobu. Archiwum pomocnicze powinno stanowić redundancję archiwum podstawowego na serwerze podstawowym. Różnica między archiwum pomocniczym a archiwum podstawowym polega na preferencjach dla archiwum podstawowego podczas przyłączania klientów sieciowych. Klient przyłączy się do archiwum pomocniczego tylko wtedy, gdy w sieci nie znajdzie archiwum podstawowego.

Archiwum pomocnicze jest stosowane w celu zapewnienia tymczasowej redundancji na czas konserwacji lub awarii serwera podstawowego. Można np. zadeklarować archiwum podstawowe ze 100-dniowym horyzontem i archiwum pomocnicze z horyzontem 10-dniowym. Pozwoli to na uzupełnienie archiwum podstawowego za okres przerwy nie przekraczającej 10 dni, przy czym w obecności serwera podstawowego klienty przyłączą się do niego, zapewniając obserwację danych z horyzontem 100-dniowym.

Archiwum replikowane ma zastosowanie na serwerach, na których Asmen nie posiada kanałów fizycznych dla danych archiwizowanych w danym zasobie, a równocześnie w sieci dostępne jest inne archiwum, z którego można pozyskać dane historyczne. Dla archiwum replikowanego Aspad nie pobiera danych bieżących z Asmena. W zamian stale pozyskuje dane historyczne z innego serwera, na którym archiwizowane są dane z kanałów

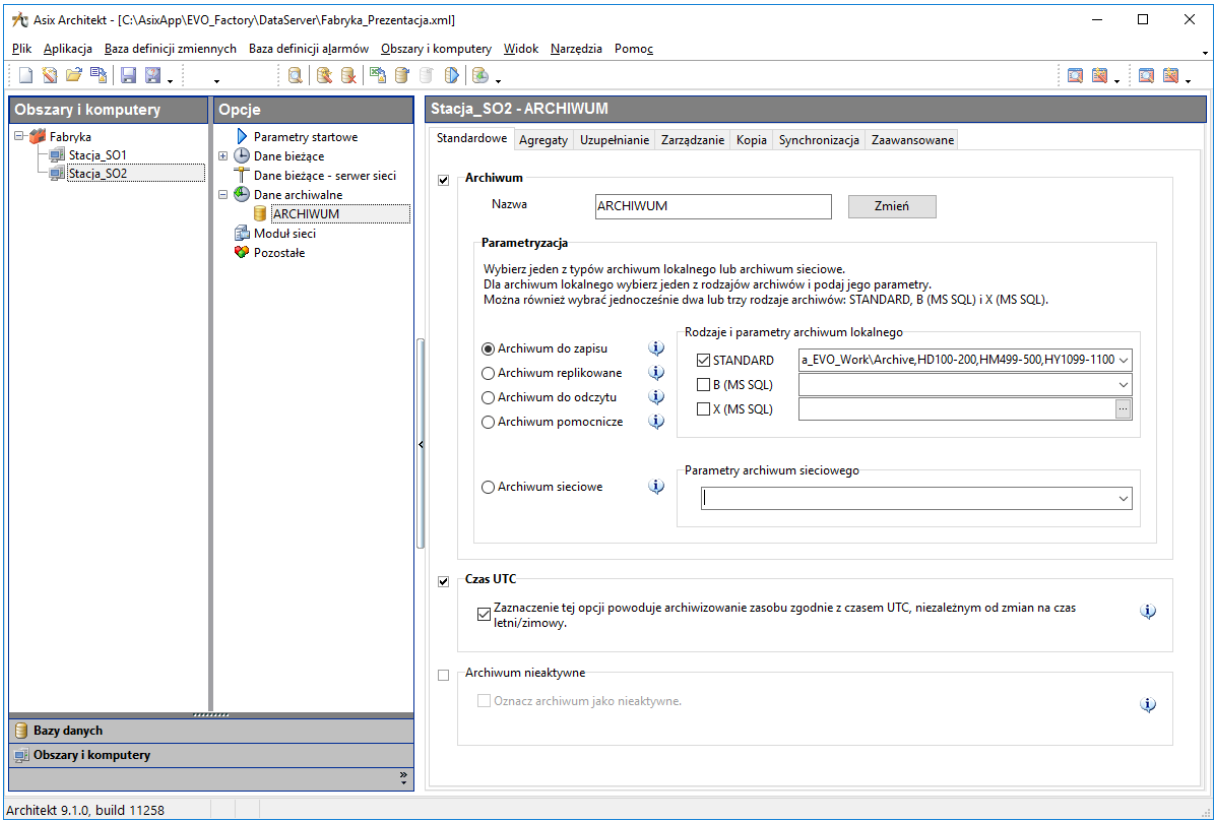
fizycznych, zapisując je we własnym archiwum. Typowym zastosowaniem będą serwery pośredniczące.
Sposób działania nieco się różni dla archiwum standardowego i SQL.

Archiwum standardowe
Dla archiwum standardowego Aspad po uzupełnieniu danych przy starcie systemu, kontynuuje to uzupełnianie, cyklicznie sprawdzając nowe dane pojawiające się na drugim serwerze. Nawet dłuższe przerwy w komunikacji są na bieżąco nadrabiane po uzyskaniu łączności, aż do ograniczenia podanego w deklaracji *Minimalny okres uzupełniania* na zakładce *Uzupełnianie* w parametrach danego archiwum.
Także na wspomnianej zakładce *Uzupełnianie* można określić nazwy serwerów, z których mają być pobierane dane do replikacji.
Deklaracja *Bez uzupełniania* na zakładce *Uzupełnianie* dla takiego archiwum nie ma sensu i jest ignorowana.

Archiwum SQL
Jeżeli dla danego zasobu nie podano deklaracji synchronizacji archiwum SQL, to archiwum replikowane działa tak samo, jak standardowe.
Jeżeli podano deklarację SYNCHRO, to dane są tylko synchronizowane na poziomie baz danych SQL, a wyłączone jest zarówno pobieranie danych bieżących z Asmena, jak i uzupełnianie przez Aslinka, tak jakby zadeklarowano opcję *Bez uzupełniania* na zakładce *Uzupełnianie* w parametrach danego archiwum.
Synchronizacja ma tę przewagę nad zwykłym uzupełnianiem, że uwzględnia wszystkie zmiany w archiwum, także dane historyczne dopisywane lub zmieniane z opóźnieniem.

2.10.9.1. Parametry standardowe

Zakładka ***Standardowe*** obejmuje następujące parametry:



Rys. Okno parametrów archiwum - zakładka 'Standardowe'.

- ***Archiwum***:
  - ***Nazwa*** - nazwa logiczna archiwum (zasobu);
    Wartość domyślna: Archiwum1
  - ***Typ archiwum*** ze względu na sposób udostępniania danych:

    (archiwum lokalne):
    'Archiwum do zapisu'
    'Archiwum do odczytu'
    'Archiwum pomocnicze'
    'Archiwum replikowane'

    lub
    'Archiwum sieciowe'

    Wartość domyślna: 'Archiwum do zapisu'
  - ***Rodzaj archiwum***:

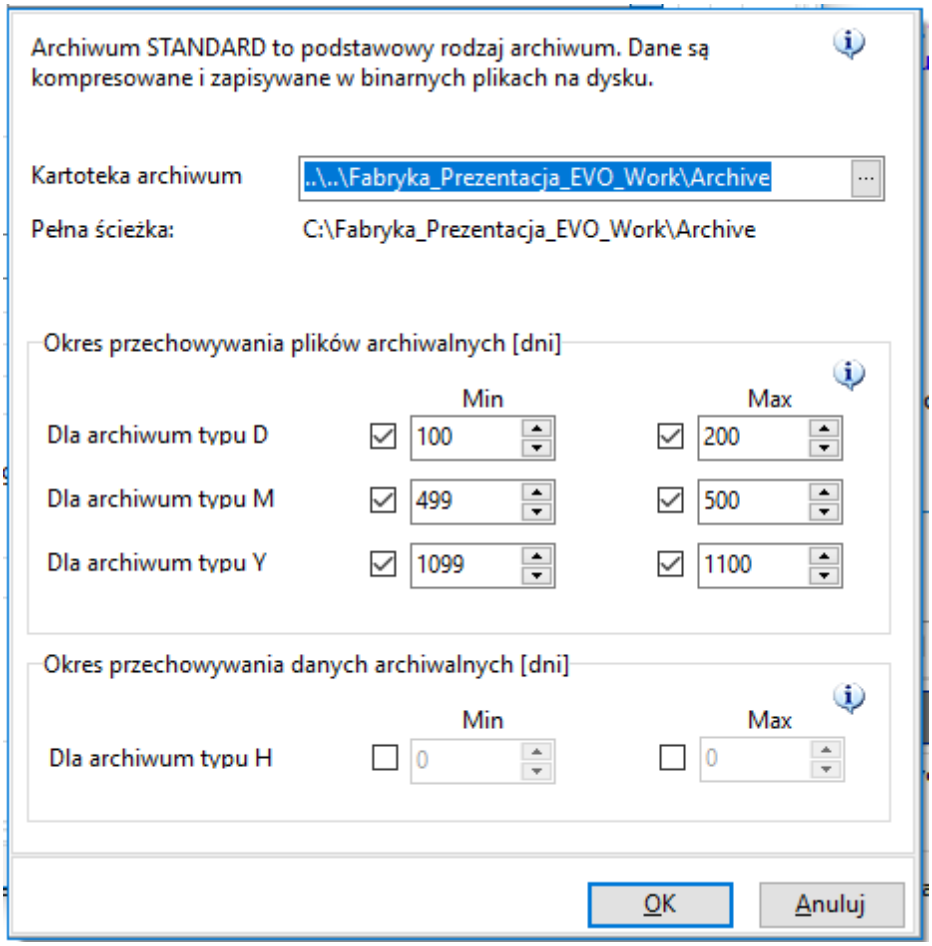
    'STANDARD'
    'B (MS SQL)'

'X (MS SQL)'

Wartość domyślna: 'STANDARD'

Archiwa STANDARD

Archiwa STANDARD wymagają podania dodatkowych parametrów, których prawidłowe określenie ułatwia okno uruchamiane przyciskiem :



Rys. Okno parametryzacji archiwów STANDARD.

- ***Kartoteka archiwum*** - względna lub bezwzględna ścieżka do kartoteki, w której będą gromadzone pliki archiwalne;
  Wartość domyślna: brak

- ***Okres przechowywania plików archiwalnych*** - okres przechowywania plików archiwalnych odpowiednio typu D, M, Y, przy czym:
  min określa liczbę dni, po zakończeniu okresu danego pliku, przez które nie wolno automatycznie kasować pliku, nawet gdy brak miejsca na dysku (domyślnie 1),

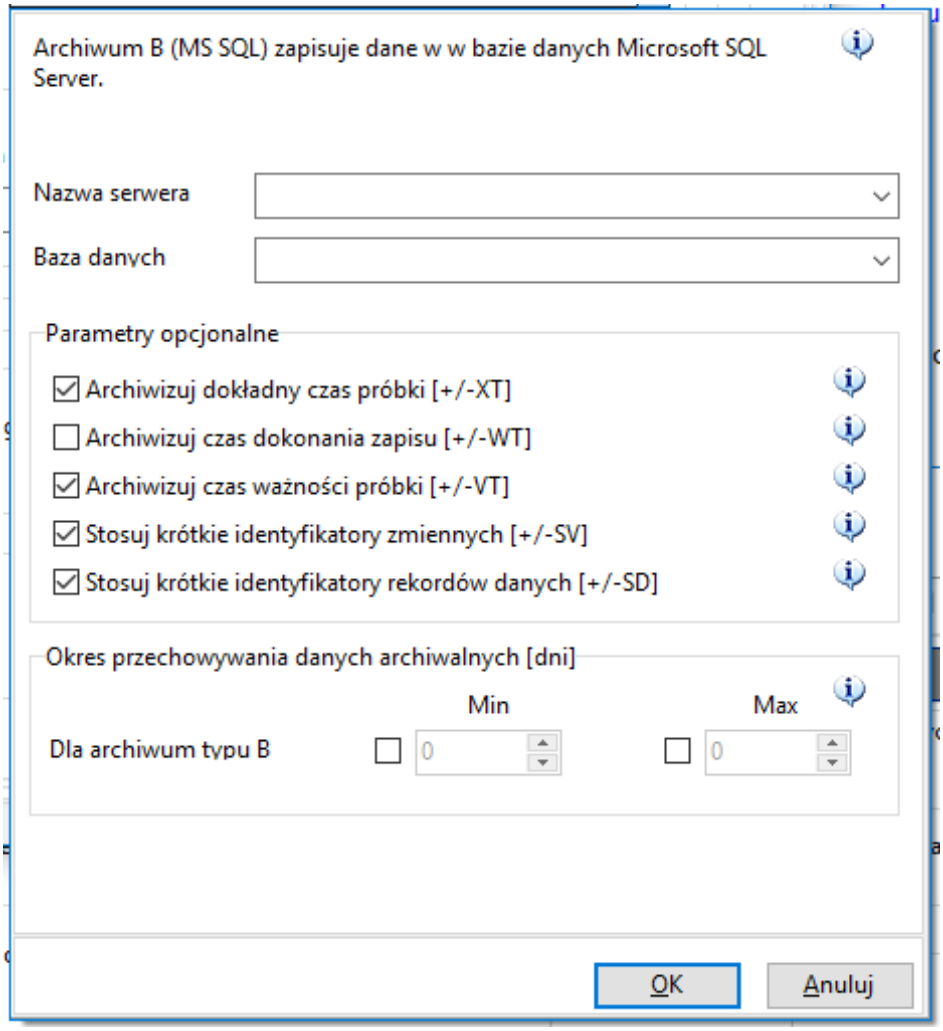
max określa liczbę dni, po których plik należy usunąć (domyślnie 0, co jest interpretowane jako nieskończoność).

- ***Okres przechowywania danych archiwalnych [dni]*** - okres przechowywania danych archiwalnych archiwum typu H, przy czym:
  min określa minimalną liczbę dni przechowywania danych,
  max określa liczbę dni przechowywania danych (0 interpretowane jest jako nieskończoność).

UWAGA: Deklaracja ma pierwszeństwo przed deklaracjami poszczególnych zmiennych w bazie definicji zmiennych. Jeśli wartość minimalna jest większa lub maksymalna mniejsza niż wartość podana w bazie, to użyta będzie wartość podana w edytorze.

Archiwa MS SQL

Archiwa MS SQL wymagają podania dodatkowych parametrów, których prawidłowe określenie ułatwia okno uruchamiane przyciskiem :



Rys. Okno parametryzacji archiwów MS SQL.

- ***Nazwa_serwera*** - względna lub bezwzględna ścieżka do kartoteki, w której będą gromadzone pliki archiwalne;
nazwa instancji serwera SQL, na którym znajduje się lub zostanie utworzona baza danych; jeżeli ten parametr jest pusty, to zostanie wykorzystany lokalny serwer (istnieje też możliwość deklaracji serwera LOCAL, co jest równoważne pustej nazwie serwera w deklaracji archiwum);

nazwa instancji ma postać:

nazwa_systemowa_komputera - dla podstawowej instancji serwera lub
*nazwa_systemowa_komputera**nazwa_instancji* - dla nazwanej instancji;
Wartość domyślna: brak

- ***Baza danych*** - nazwa bazy danych po stronie serwera; jeżeli ten parametr jest pusty, to nazwa bazy danych = **ASSQL**_*nazwa_zasobu*;
ta nazwa nie może się powtarzać dla różnych archiwów na tym samym serwerze SQL, ale może się powtarzać w różnych deklaracjach, jeżeli dotyczą one wielu serwerów
Wartość domyślna: brak

- ***Parametry opcjonalne***:

-XT, **+XT** - ***archiwizacja czasu danej*** - określa, czy w bazie danych będzie archiwizowany czas dokładny danej (+XT), czy też jedynie czas zaokrąglony do pełnego cyklu próbkowania (-XT) (wyjątkiem są dane, dla których zadeklarowano atrybut KAŻDY - dla takich danych czas nie jest zaokrąglany do cyklu próbkowania);
włączenie opcji powoduje dodanie 8-bajtowego pola do każdego rekordu danych;
Wartość domyślna: +XT - czas jest archiwizowany;

-WT, +WT - ***archiwizacja czasu dokonania zapisu*** - określa, czy w bazie danych będzie archiwizowany czas dokonania zapisu;
włączenie opcji powoduje dodanie 8-bajtowego pola do każdego rekordu danych;
Wartość domyślna: -WT - czas zapisu jest pomijany;

-VT, **+VT** - ***archiwizacja czasu ważności próbki*** - określa, czy w bazie danych będzie archiwizowany czas ważności próbki (+VT), czy też na być pomijany, a domyślnym czasem ważności ma być okres archiwizacji (-VT); w efekcie, dla -VT, gdy zostanie w przyszłości skrócony okres archiwizacji, to dane historyczne będą oddzielone dziurami; domyślnie czas ważności jest archiwizowany;
włączenie opcji powoduje dodanie 4-bajtowego pola do każdego rekordu danych;
Wartość domyślna: +VT - czas ważności jest archiwizowany;

-SV, **+SV** - ***liczba bajtów identyfikatora przebiegu*** - określa, czy identyfikator przebiegu zmiennej ma mieć 2 (+SV), czy 4 (-SV) bajty; dla +SV baza danych nie może zawierać więcej niż 32767 zmiennych na zasób (wlicza się w to zmienne kiedyś zadeklarowane, później usunięte);
wyłączenie opcji powoduje, dla każdego rekordu danych, zwiększenie pola identyfikatora przebiegu o 2 bajty w danych i o 2 bajty w indeksie bazy;
Wartość domyślna: +SV - używane są małe identyfikatory;

-SD, **+SD** - ***liczba bajtów identyfikatora rekordów*** - określa, czy identyfikator rekordu danych ma mieć 4 (+SD), czy 8 (-SD) bajtów; dla +SD do bazy nie można zapisać więcej niż 2 miliardów (2^{31}) rekordów danych na zasób (wlicza się w to każdy zapis i zmianę rekordu nawet, gdy zostanie on usunięty);
wyłączenie opcji powoduje, dla każdego rekordu danych, zwiększenie pola identyfikatora rekordu o 4 bajty w danych i o 4 bajty w indeksie bazy;
Wartość domyślna: +SD - używane są małe identyfikatory.

Dla baz danych zadeklarowanych jako MSSQL rekord zawiera dla każdej próbki domyślnie wartość, jakość, czas cyklu, czas dokładny, czas zapisu rekordu i okres ważności próbki. Tylko czas cyklu należy do klucza bazy danych. Trzy pierwsze pola są obowiązkowe.

Dla raz utworzonej bazy danych nie wolno zmieniać pól opcjonalnych w deklaracji, gdyż nie będą one wtedy odpowiadały strukturze tablic bazy danych.

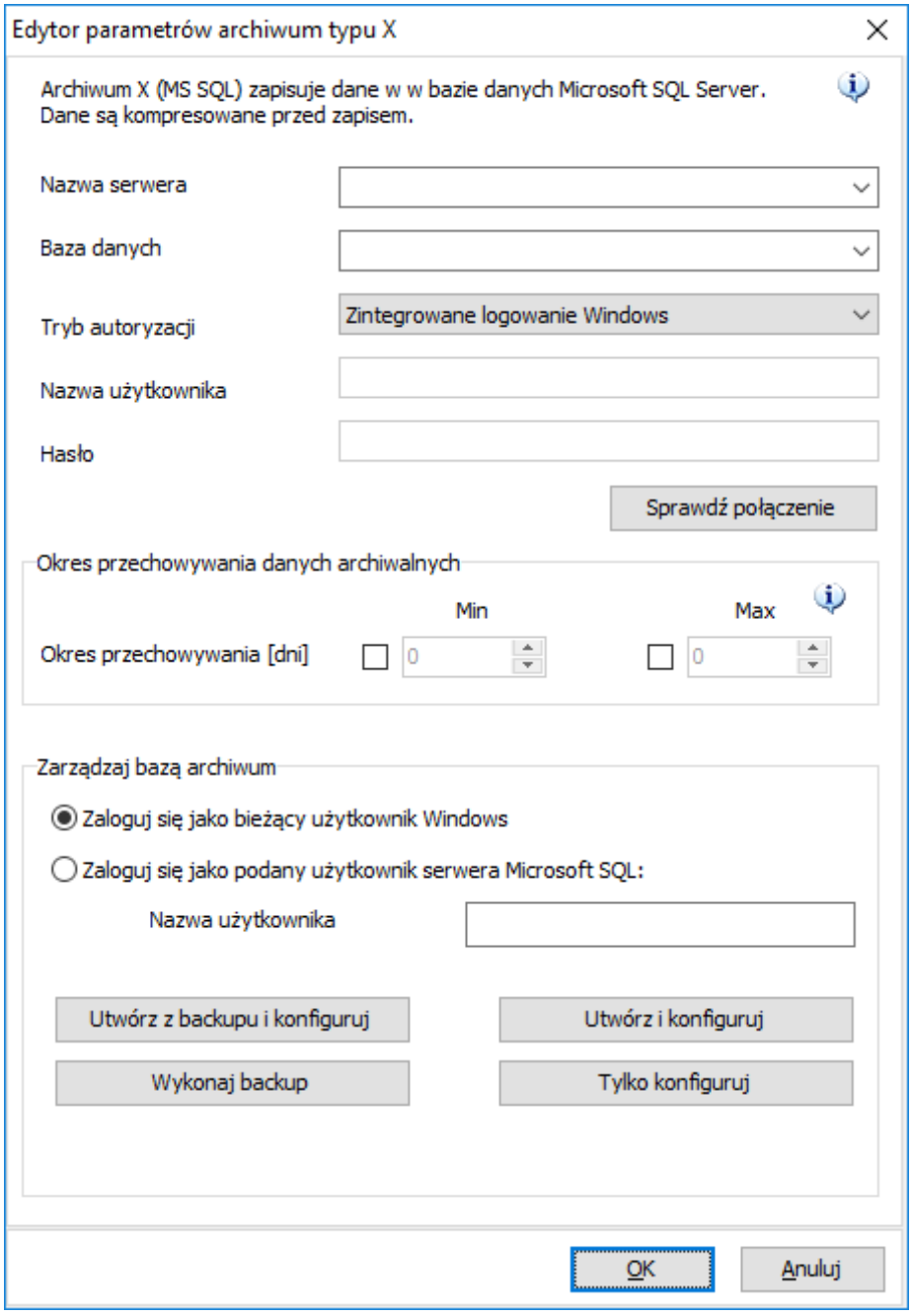
- ***Okres przechowywania danych archiwalnych [dni] -*** okres przechowywania danych archiwalnych archiwum typu H, przy czym:

min określa minimalną liczbę dni przechowywania danych,
max określa liczbę dni przechowywania danych (0 interpretowane jest jako nieskończoność).

UWAGA: Deklaracja ma pierwszeństwo przed deklaracjami poszczególnych zmiennych w bazie definicji zmiennych. Jeśli wartość minimalna jest większa lub maksymalna mniejsza niż wartość podana w bazie, to użyta będzie wartość podana w edytorze.

Archiwum X (MS SQL)

Archiwum X (MS SQL) wymaga podania dodatkowych parametrów, których prawidłowe określenie ułatwia okno uruchamiane przyciskiem ...:



Rysunek: Okno parametryzacji archiwum X (MS SQL).

- ***Nazwa_serwera*** - względna lub bezwzględna ścieżka do kartoteki, w której będą gromadzone pliki archiwalne;

  nazwa instancji serwera SQL, na którym znajduje się lub zostanie utworzona baza danych; jeżeli ten parametr jest pusty, to zostanie wykorzystany lokalny serwer (istnieje też możliwość deklaracji serwera LOCAL, co jest równoważne pustej nazwie serwera w deklaracji archiwum);

  nazwa instancji ma postać:

  nazwa_systemowa_komputera - dla podstawowej instancji serwera lub *nazwa_systemowa_komputera**nazwa_instancji* - dla nazwanej instancji;
  Wartość domyślna: brak

  Należy zwrócić uwagę, że Microsoft SQL Server Express Edition tandardowo instaluje się jako nazwana instancja instancja 'SQLEXPRESS'. Jako nazwę serwera należy wówczas podać <nazwa_komputera>\SQLEXPRESS.

- ***Baza danych*** - nazwa bazy danych po stronie serwera; jeżeli ten parametr jest pusty, to nazwa bazy danych = ***ARCX***_*nazwa_zasobu*;

  ta nazwa nie może się powtarzać dla różnych archiwów na tym samym serwerze SQL, ale może się powtarzać w różnych deklaracjach, jeżeli dotyczą one wielu serwerów
  Wartość domyślna: brak

- ***Tryb autoryzacji*** - domyślne ustawienie to 'Zintegrowane logowanie Windows' (ten tryb nie wymaga podania nazwy użytkownika i hasła, przyjmuje się, ze autoryzacja jest na bieżącego użytkownika zalogowanego w systemie Windows). Pozostałe warianty '_asix_internal' oraz 'Podany użytkownik serwera SQL' wymagają uprzedniego zdefiniowania na serwerze SQL w/w użytkowników. Administrator uprawniony do zarządzania bazą danych (identyfikowany w ramce *Zarządzaj bazą archiwum*) nadaje uprawnienia do Archiwum X powyższym użytkownikom, wykonując jedną z poniższych komend: *Utwórz z backupu i konfiguruj*, *Utwórz i konfiguruj*, *Tylko konfiguruj*.

- ***Nazwa użytkownika*** - wymaga podania użytkownika w żelazności od wybranego trybu autoryzacji.

- ***Hasło*** - wymaga podania hasła użytkownika w żelazności od wybranego trybu autoryzacji.

- ***Okres przechowywania danych archiwalnych [dni]*** - okres przechowywania danych archiwalnych archiwum typu X, przy czym:

  min określa minimalną liczbę dni przechowywania danych,
  max określa maksymalną liczbę dni przechowywania danych (0 interpretowane jest jako nieskończoność).

  UWAGA: Deklaracja ma pierwszeństwo przed deklaracjami poszczególnych zmiennych w bazie definicji zmiennych. Jeśli wartość minimalna jest większa lub maksymalna mniejsza niż wartość podana w bazie, to użyta będzie wartość podana w tym edytorze.

- ***Zarządzaj bazą archiwum:***

  Zaloguj się jako bieżący użytkownik Windows - użytkownik z uprawnieniami administratora na serwerze SQL, na którym założona będzie/jest baza archiwum X.

  Zaloguj się jako podany użytkownik serwera Microsoft SQL - użytkownik z uprawnieniami administratora na serwerze SQL, na którym założona będzie/jest baza archiwum X.

- ***Utwórz z backupu i konfiguruj*** - odtworzenie bazy z backupu i nadanie użytkownikowi podanemu w Trybie autoryzacji uprawnień do bazy archiwum X.

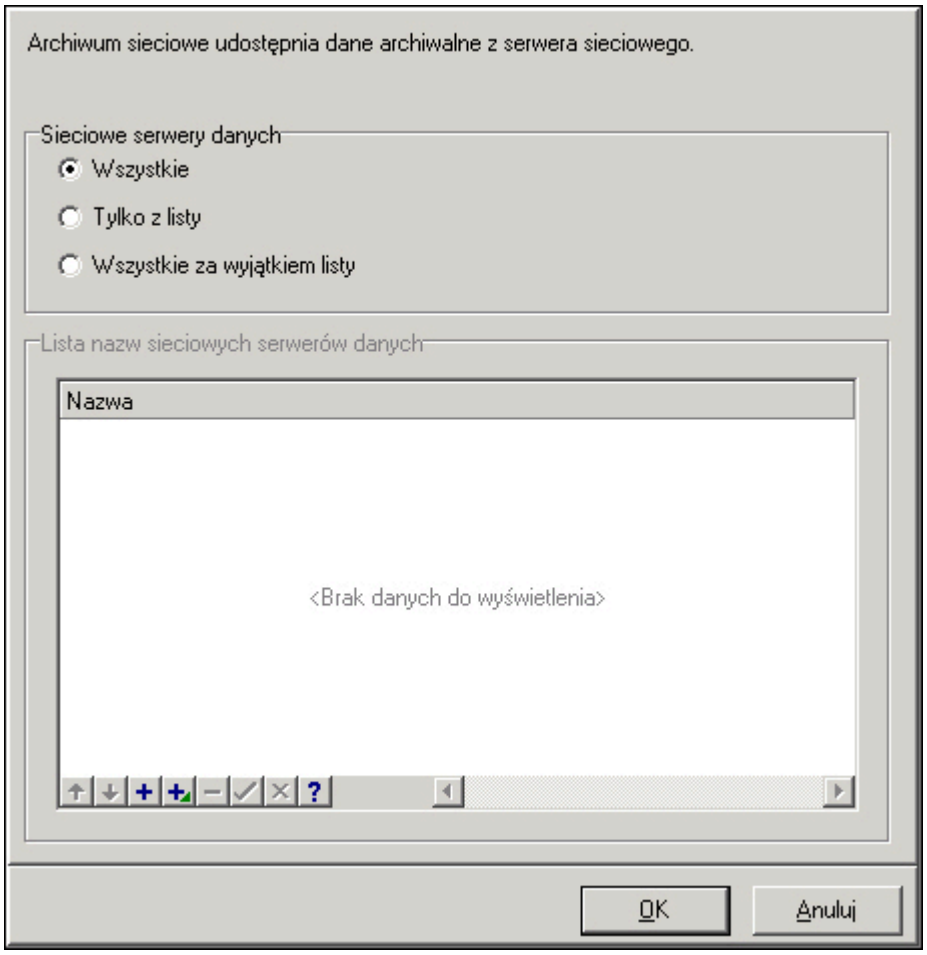
- ***Wykonaj backup*** -utworzenie backupu bazy.

- ***Utwórz i konfiguruj*** - utworzenie bazy i nadanie użytkownikowi podanemu w Trybie autoryzacji uprawnień do bazy archiwum X.

- ***Tylko konfiguruj*** - nadanie użytkownikowi podanemu w Trybie autoryzacji uprawnień do bazy archiwum X.

Archiwa sieciowe

Archiwa sieciowe wymagają podania stacji, do których wolno sięgać po dane dla zasobu przy uzupełnianiu archiwum lokalnego lub odczycie archiwum zdalnego. Deklaracja stacji odbywa się za pomocą okno uruchamianego przyciskiem :



Rys. Okno deklaracji stacji przy odczycie archiwum zdalnego.

- ***Komputery***:

  'Wszystkie'
  'Tylko z listy'
  'Wszystkie za wyjątkiem listy'

- ***Lista nazw sieciowych serwerów danych*** - pozwala zadeklarować stacje, z których nie można sięgać po dane dla zasobu przy uzupełnianiu archiwum lokalnego lub odczycie archiwum zdalnego.
  Wartość domyślna: 'Wszystkie'

* * *

- ***Czas UTC*** - deklaracja determinuje archiwizowanie zasobu zgodnie z czasem UTC;

Jeżeli deklarujemy takie zasoby, to ustawienia systemowe strefy czasowej MUSZĄ zawierać poprawną strefę z zaznaczoną automatyczną zmianą czasu letniego/zimowego. W przeciwnym razie czas będzie zapisywany niepoprawnie.

Jeżeli w już archiwizowanym zasobie deklarujemy *Czas UTC*, to trzeba skonwertować dotychczasowe archiwum. Dla archiwum standardowego można to wykonać przy pomocy specjalnej funkcji AspadTools. Trzeba pamiętać o sekwencyjnym przetwarzaniu plików archiwalnych - najlepiej konwertując wszystkie jednym poleceniem.

Gdy nie zostanie zadeklarowany czas UTC, to komunikacja pomiędzy ASPADami odbywa się w czasie lokalnym i nie jest wrażliwa na niewłaściwe ustawienia stref czasowych.

Jeżeli na serwerze zadeklarowano czas UTC, to na kliencie też musi zostać zadeklarowany czas UTC.

- ***Archiwum nieaktywne*** - zmienne należące do archiwum nieaktywnego: nie są archiwizowane, nie są liczone do limitu licencji, jeśli jednocześnie nie należą do kanałów fizycznych.

* * *

Przykłady deklaracji archiwum MS SQL:

Deklaracja archiwum do zapisu dla zasobu MojeArchiwum, na lokalnym serwerze, w bazie danych ASSQL_MojeArchiwum, bez archiwizacji czasu zapisu rekordu:

Archiwum do zapisu
Nazwa: 'MojeArchiwum'
Rodzaj archiwum: 'MS SQL'
Nazwa serwera: -
Baza danych: -

Deklaracja archiwum do zapisu dla zasobu MojeArchiwum, na lokalnym serwerze, w bazie danych MojeArchiwum, bez archiwizacji czasu zapisu rekordu:

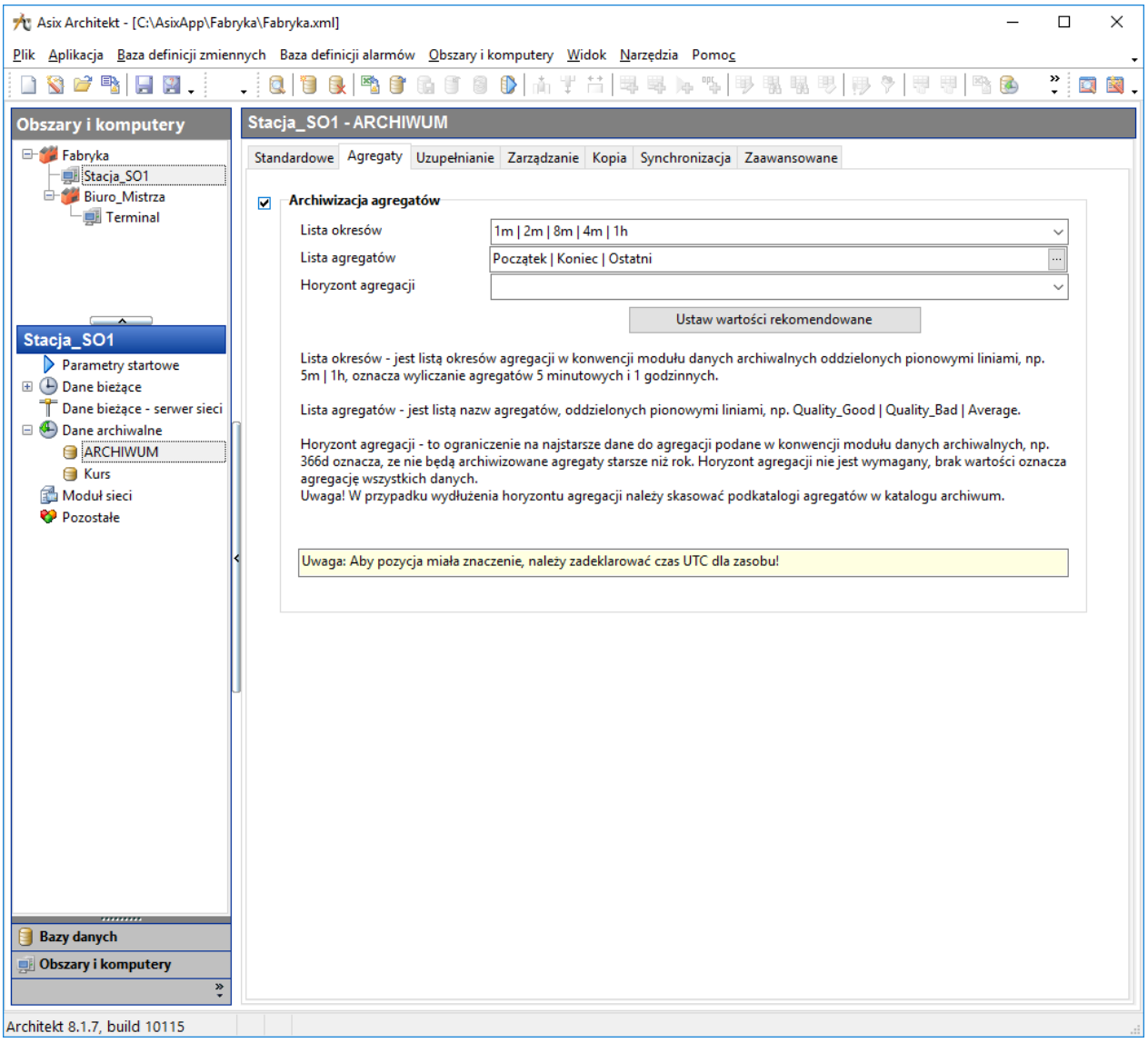
Archiwum do zapisu
Nazwa: 'MojeArchiwum'
Rodzaj archiwum: 'MS SQL'
Nazwa serwera: -
Baza danych: 'MojeArchiwum'

Deklaracja archiwum do odczytu dla zasobu 'ArcSrv1', na serwerze 'Srv1', w bazie 'ASSQL_Arc' z pełnym rekordem:

Archiwum do odczytu
Nazwa: 'ArcSrv1'
Rodzaj archiwum: 'MS SQL'
Nazwa serwera: 'Srv1'
Baza danych: 'ASSQL_Arc'
Archiwizacja czasu dokonania zapisu: R

2.10.9.2. Parametry archiwizacji agregatów

Zakładka ***Agregaty*** umożliwia parametryzację Agregatora - modułu systemu Asix przeznaczonego do wyliczania i archiwizacji danych statystycznych. Domyślnie funkcja jest załączona z rekomendowanymi ustawieniami.



Rys. Okno parametrów archiwum - zakładka 'Agregaty'.

- ***Lista okresów*** - pole umożliwia zadeklarowanie listy okresów agregacji, będących podstawą wyliczania wartości zagregowanych; wartości okresów należy deklarować w konwencji modułu Aspad, oddzielając je pionowymi liniami, np.: 5m|1h stanowi podstawę wyliczania agregatów 5-minutowych i 1-godzinnych;

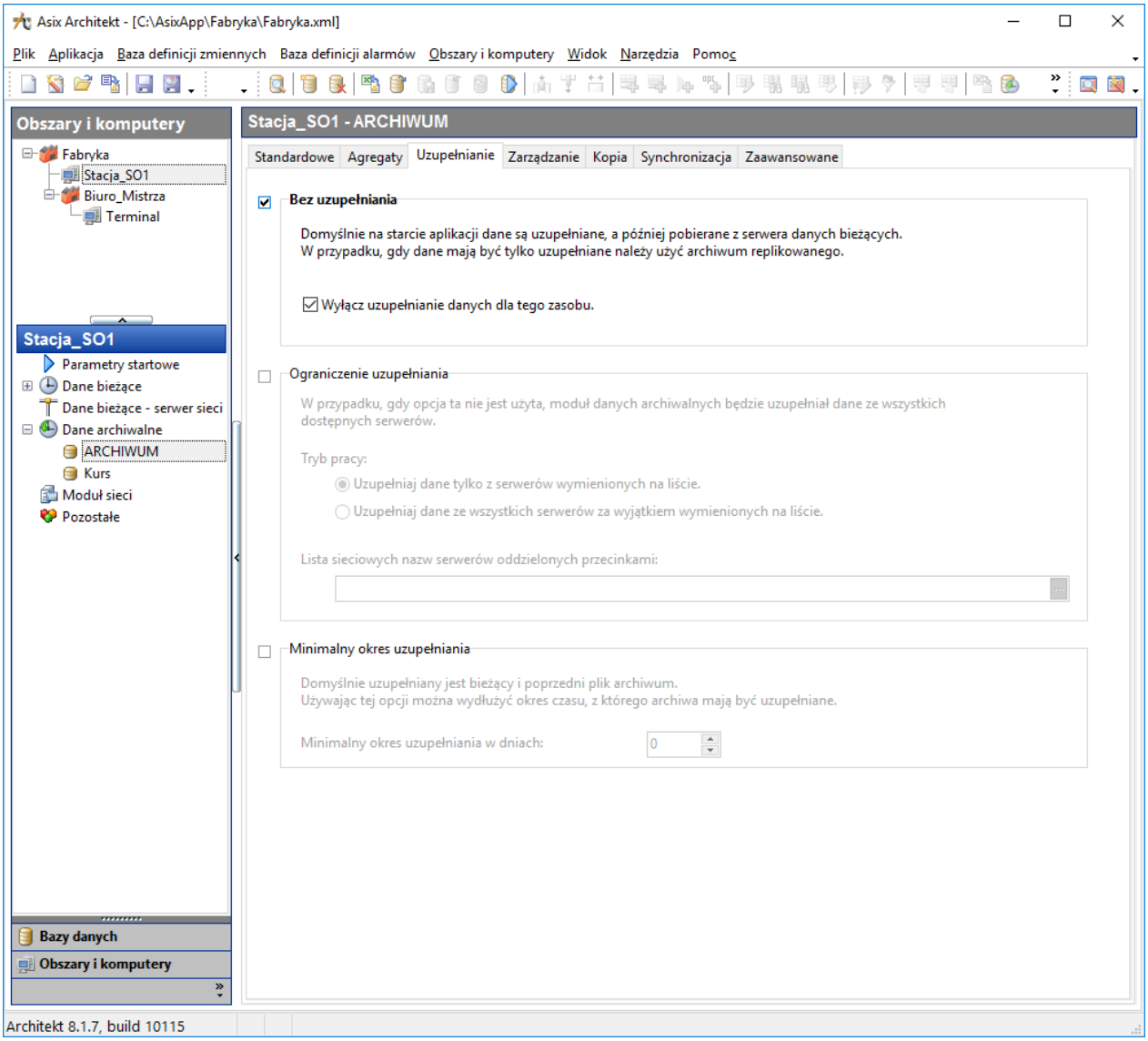
- ***Lista agregatów*** - pole, w którym należy określić agregaty, które będą wyliczane i archiwizowane; po rozwinięciu pola pojawia się okno umożliwiające selekcję agregatów z listy agregatów dostępnych w systemie Asix;

- ***Horyzont agregacji*** **-** oznacza okres, z którego dane będą agregowane; wartość horyzontu należy deklarować w konwencji modułu Aspad, np.: 366d - oznacza, że nie będą archiwizowane dane starsze niż rok.

Przycisk ***Ustaw wartości rekomendowane*** pozwala ustawić okresy danych 1m,1h oraz wszystkie agregaty. Wymienione opcje ustawiane są domyślnie przy tworzeniu nowego archiwum.

2.10.9.3. Zarządzanie uzupełnianiem danych

Zakładka ***Uzupełnianie*** pozwala określić następujące parametry:



Rys. Okno parametrów archiwum - zakładka 'Uzupełnianie'.

- ***Bez uzupełniania*** - włączenie opcji spowoduje zakaz uzupełniania danych dla danego archiwum (zasobu);
  Wartość domyślna: opcja wyłączona

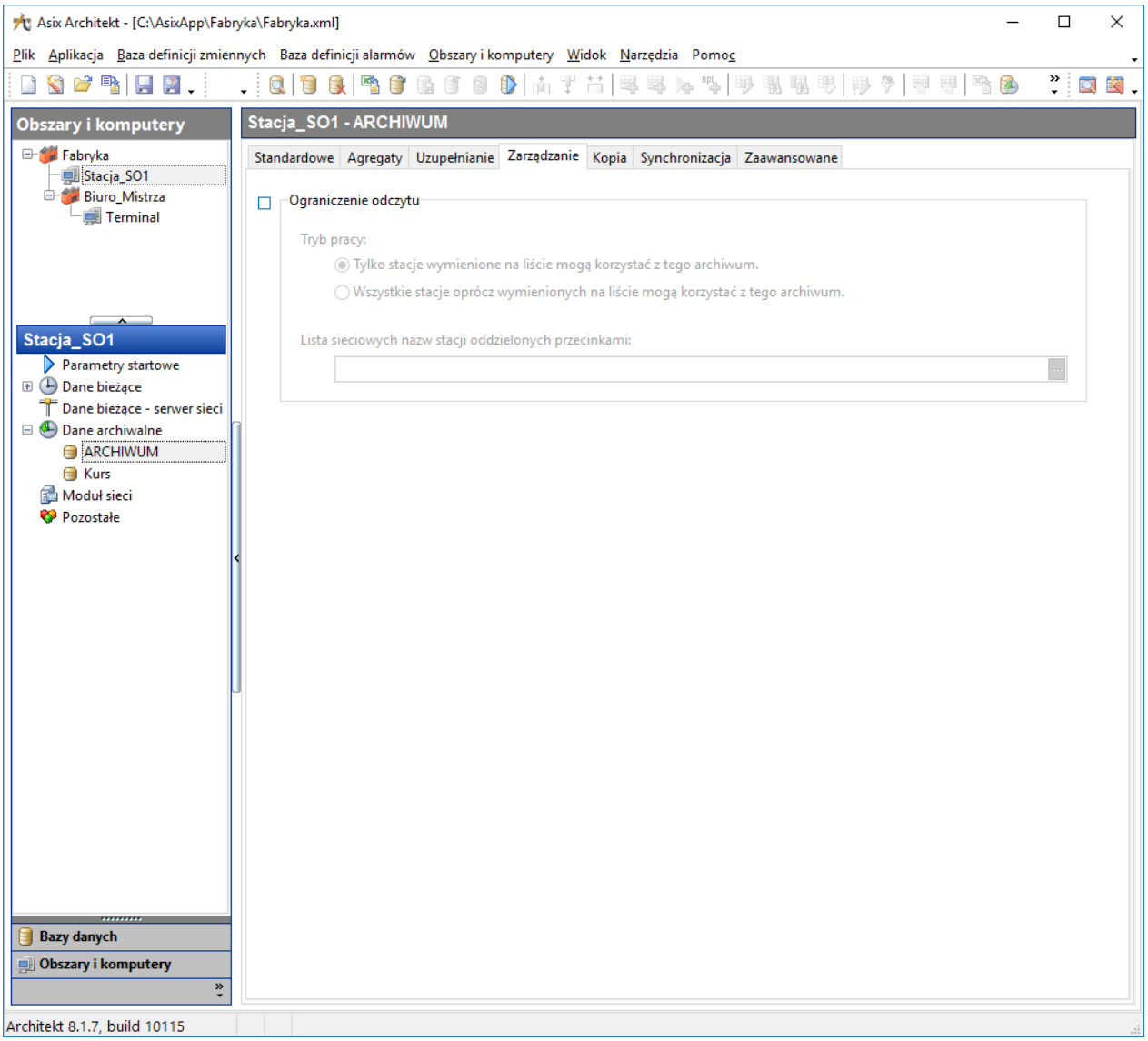
- ***Ograniczenia uzupełniania*** - pozwala zawęzić listę stacji, z których Aspad będzie uzupełniał dane; pod warunkiem, ze opcja *Bez uzupełniani* jest wyłączona:

  - ***Uzupełniaj dane tylko z serwerów wymienionych na liście*** - lista nazw stacji, do których wolno sięgać po dane dla zasobu przy uzupełnianiu archiwum lokalnego; pominięcie tej deklaracji oznacza akceptację dowolnego serwera dysponującego określonym zasobem.
  - ***Uzupełniaj dane ze wszystkich stacji za wyjątkiem wymienionych na liście*** - lista nazw stacji, do których nie wolno sięgać po dane dla zasobu przy uzupełnianiu archiwum lokalnego.
    Wartość domyślna: dane uzupełniane są z dowolnego serwera dysponującego danym zasobem

- ***Minimalny okres uzupełniania*** - używając opcji można wydłużyć okres czasu, z którego archiwa mają być uzupełniane.
  Wartość domyślna: uzupełniany jest bieżący i poprzedni plik archiwum.

2.10.9.4. Zarządzanie odczytem danych

Zakładka ***Zarządzanie*** pozwala określić następujące parametry:



Rys. Okno parametrów archiwum - zakładka 'Zarządzanie'.

- ***Ograniczenie odczytu*** - opcja pozwala ustalić tryb pracy, w którym określone stacje mogą korzystać z danego archiwum:

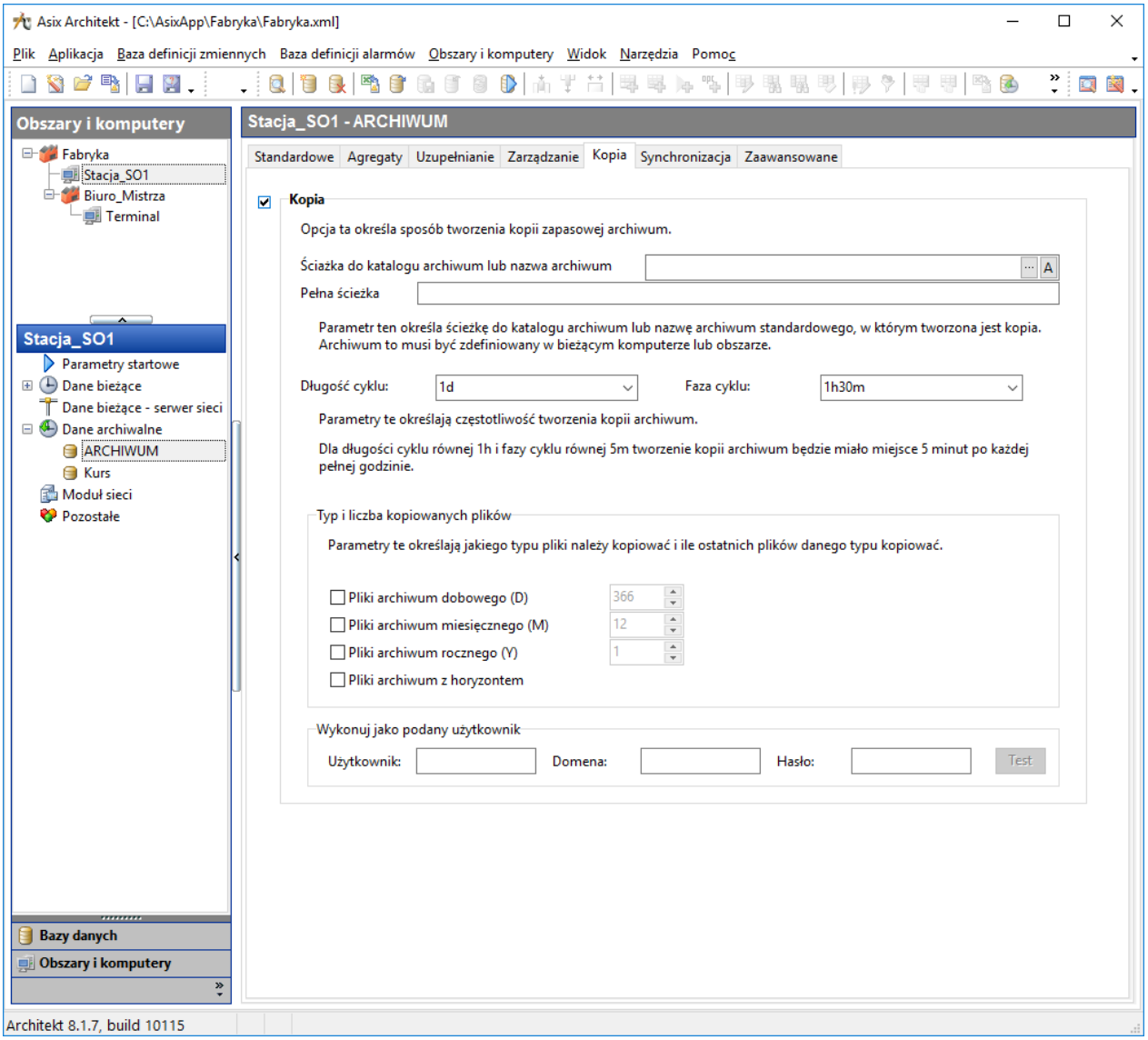
  Tryb pracy:

  - ***Tylko stacje wymienione na liście mogą korzystać z tego archiwum***;
  - ***Wszystkie stacje oprócz wymienionych na liście mogą korzystać z danego archiwum***;

  Nazwy sieciowe stacji należy wpisać, oddzielając je przecinkami.
  Wartość domyślna: nie ma ograniczeń dotyczących odczytu.

2.10.9.5. Parametry określające sposób tworzenia kopii zapasowej archiwum

Zakładka ***Kopia*** dotyczy opcji:



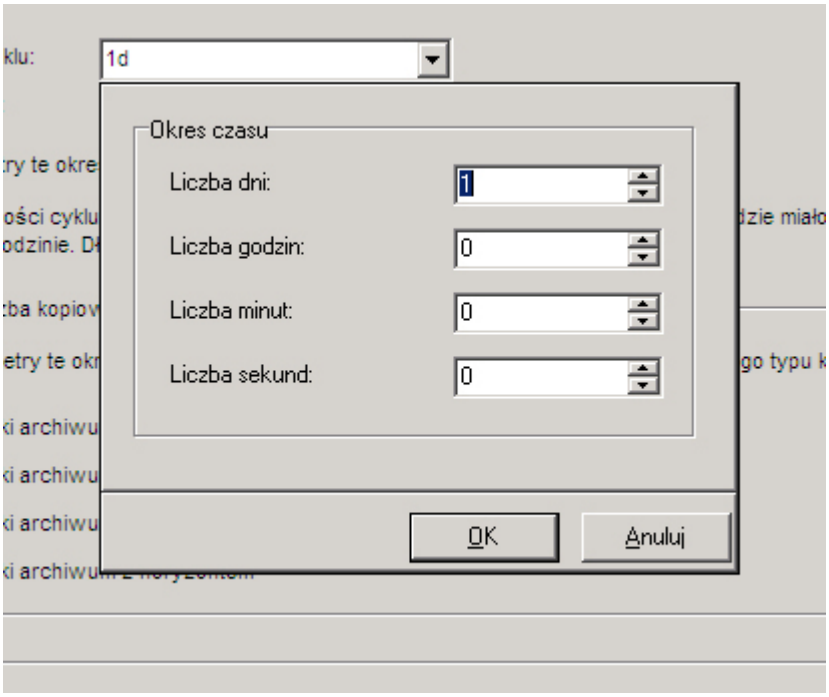
Rys. Okno parametrów archiwum - zakładka 'Kopia'.

- ***Kopia*** - parametryzuje sposób tworzenia kopii zapasowych:
  Wartość domyślna: kopia zapasowa nie jest tworzona

  - ***Ścieżka do katalogu archiwum lub nazwa archiwum*** - pozwala zadeklarować bezpośrednio katalog docelowy, w którym będzie tworzona kopia archiwum, bądź też dla zachowania kompatybilności ze starszą formą deklarowania kopii zapasowych - nazwę archiwum, w którym tworzona będzie kopia zapasowa (wcześniejsza forma deklarowania kopii

zapasowej zakładała bowiem utworzenie dodatkowego archiwum pomocniczego, które następnie deklarowane było jako miejsce docelowe tworzenia kopii zapasowej innego archiwum); aktualnie w przypadku wyboru archiwum po nazwie możliwy jest wybór wszystkich archiwów standardowych, a nie tylko pomocniczych.
Wartość domyślna: brak

- ***Długość cyklu*** - zadaje jak często inicjować tworzenie kopii zapasowych;
  Wartość domyślna: 1 dzień



Rys. Pole deklaracji długości cyklu inicjowania tworzenia kopii zapasowych.

- ***Faza cyklu*** - określa, z jakim opóźnieniem w stosunku do pełnego cyklu inicjować tworzenie kopii;
  Wartość domyślna: 1 godzina i 30 minut

- ***Typ i liczba kopiowanych plików*** - pozwala określić, jakiego typu pliki należy kopiować i ile ostatnich plików danego typu kopiować:
  - ***Pliki archiwum dobowego (D),***
  - ***Pliki archiwum miesięcznego (M),***
  - ***Pliki archiwum rocznego (Y),***
  - ***Pliki archiwum z horyzontem.***

  Wartość domyślna: brak deklaracji typu

- ***Wykonaj jako podany użytkownik***

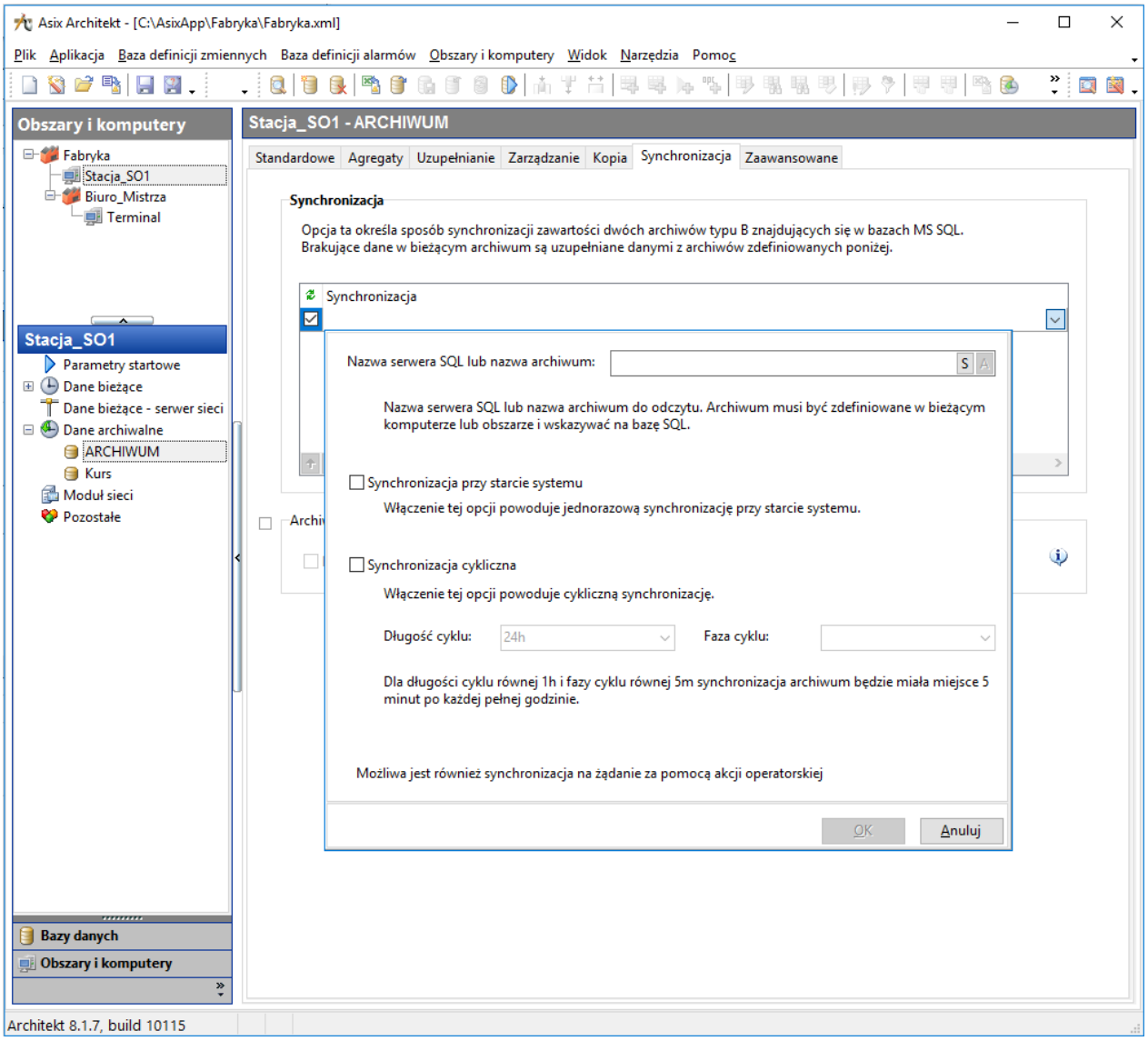
Dla jednego zasobu można zadeklarować kilka archiwów, do których tworzone będą kopie - co pozwala np. na kopiowanie plików dobowych w inne miejsce niż miesięcznych.

2.10.9.6. Deklaracja synchronizacji zawartości archiwum typu B z danymi w bazie MS SQL

Synchronizacja powoduje kopiowanie przyrostowe. Kopiowane są tylko dane, które zostały dopisane lub zmienione w archiwum źródłowym od ostatniego kopiowania. W przypadku konfliktu z już istniejącymi danymi w archiwum docelowym znajdą się dane o lepszej jakości. W przypadku danych o tej samej jakości preferowane są dane wprowadzane ręcznie.
Można podać wiele deklaracji synchronizacji, także dla jednego zasobu, co pozwala np. na uzupełnianie danych z kilku bliźniaczych serwerów.

Deklaracja synchronizacji archiwum MS SQL z danymi innego archiwum wymaga podania w parametrach synchronizacji (w parametrach archiwum, którego zawartość będzie synchronizowana) nazwy serwera SQL, z którego będą pobierane dane do synchronizacji. Wymagane jest przy tym, aby nazwy obu baz danych były identyczne.

Deklaracji takiej dokonuje się na zakładce ***Synchronizacja***:



Rys. Okno parametrów archiwum - zakładka 'Synchronizacja'.

- ***Nazwa archiwum lub serwera SQL*** - nazwa serwera SQL, z którego będą pobierane dane do synchronizacji lub nazwa archiwum, z którego pobierane będą dane do synchronizacji w przypadku deklaracji dodatkowego 'wirtualnego' archiwum, gdy nazwy baz danych (archiwum synchronizowanego i archiwum, z którym realizowana jest synchronizacja) różnią się - poniżej patrz przypadek z nazwą archiwum;
  Wartość domyślna: brak deklaracji synchronizacji.

- ***Synchronizacja przy starcie systemu*** - włączenie tej opcji powoduje jednorazową synchronizację przy starcie systemu.

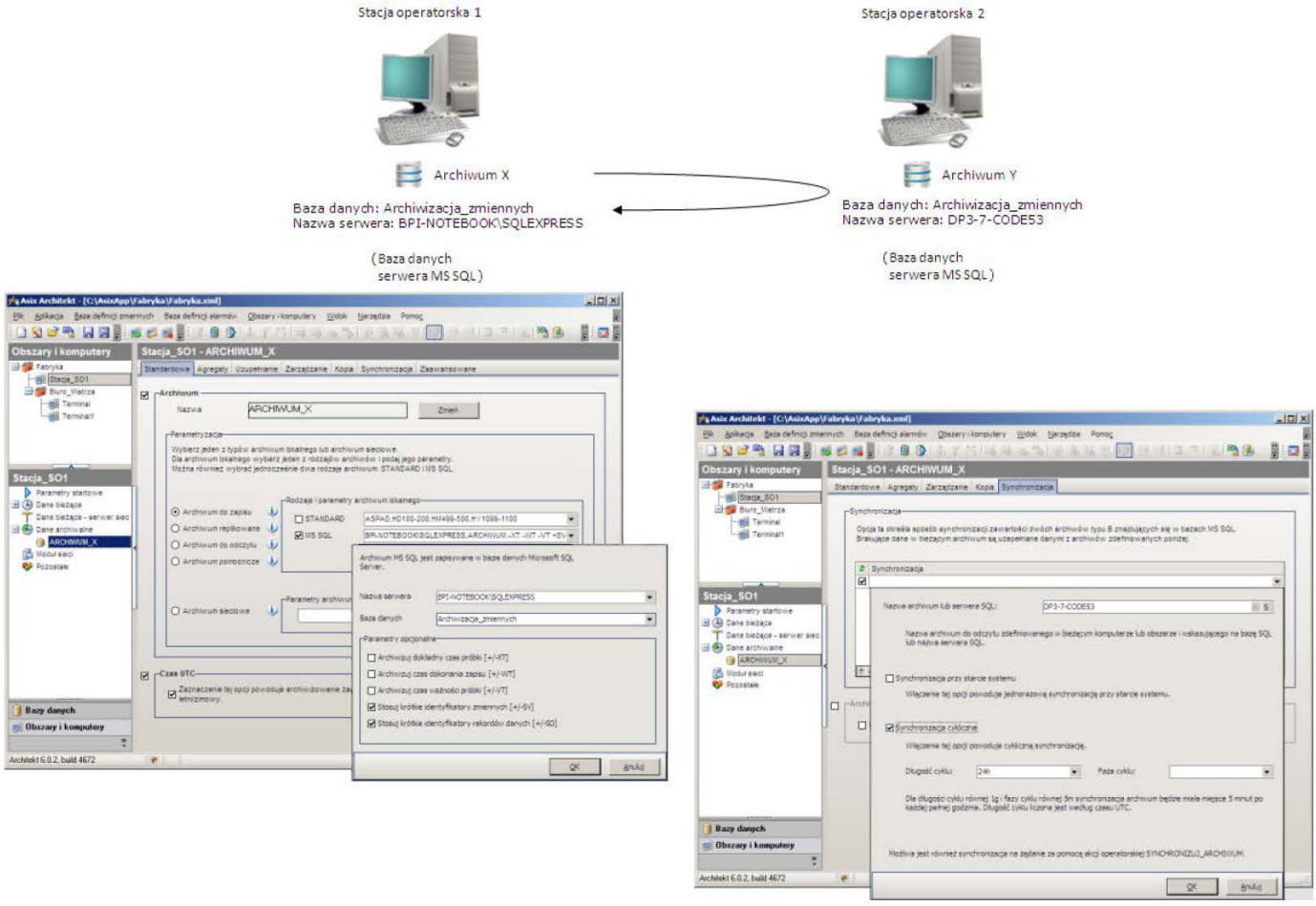
- ***Synchronizacja cykliczna*** - włączenie tej opcji powoduje synchronizację cykliczną.

    - ***długość cyklu*** - określa, jak często inicjować dopisywanie danych; w przypadku, gdy jako parametr cykl zostanie podana wartość 0, synchronizacja nie będzie wykonywana cyklicznie, a wówczas

przedmiotowa deklaracja będzie wykorzystywana tylko jako deklaracja dla akcji operatorskiej SYNCHRONIZUJ_ARCHIWUM; wartość domyślna parametru cykl: 24 godz.;

- ***faza cyklu*** - określa, z jakim opóźnieniem w stosunku do pełnego cyklu inicjować kopiowanie danych; wartość domyślna: 0.

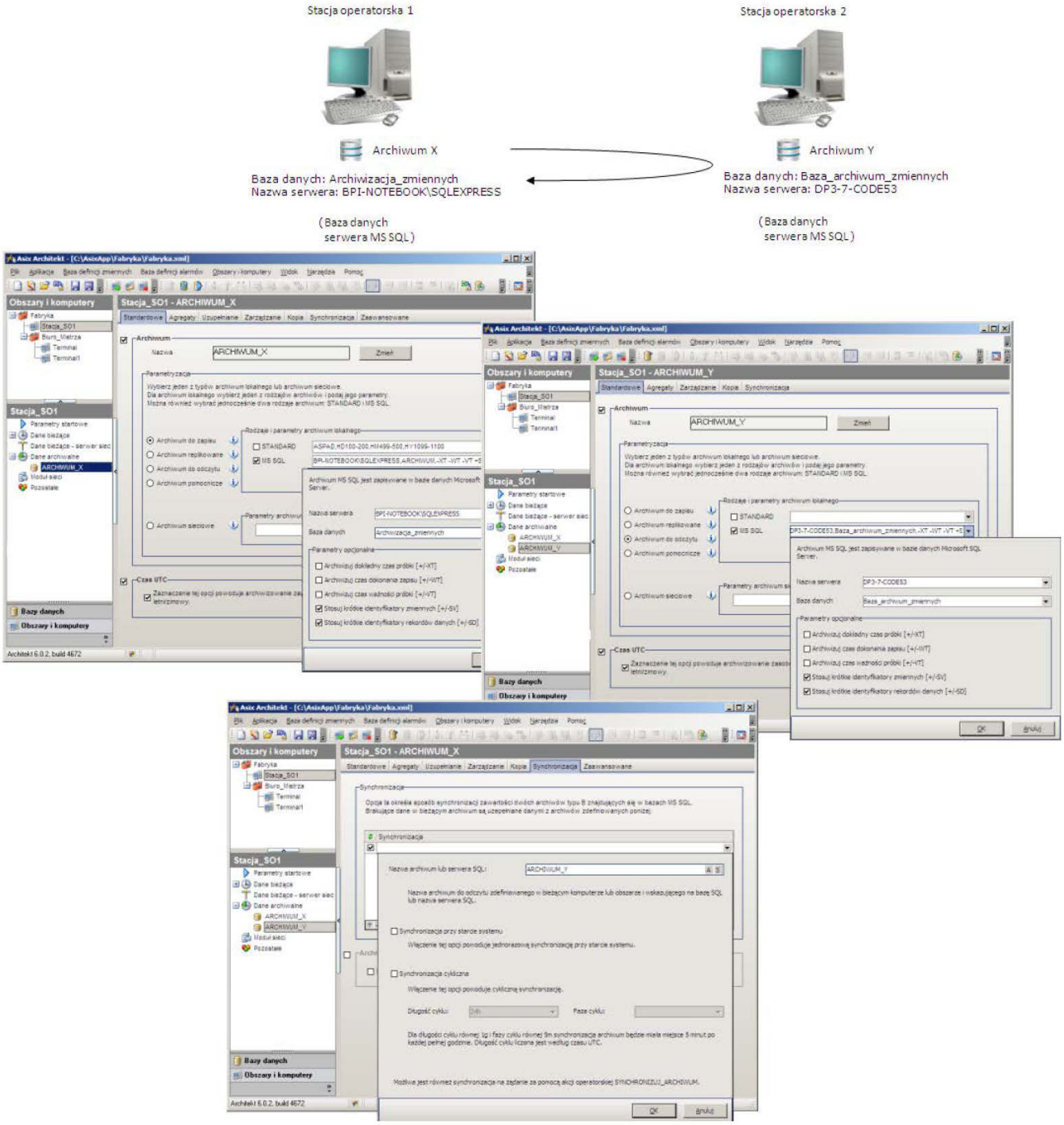
- ***Archiwum master*** - opcja powoduje, że przy zapisie do takiego archiwum na komputerze, na którym podano tę opcję, dane opatrywane są dodatkowym statusem AS_STAT_MASTER. Dla każdego archiwum na co najwyżej jednym z komputerów w sieci można podać taka opcję. Przy synchronizacji dane opatrzone statusem AS_STAT_MASTER są traktowane jako ważniejsze niż dane o tej samej jakości, ale bez tego statusu. Dzięki temu synchronizacja nie tylko wypełnia dziury w archiwum oraz zastępuje dane o gorszym statusie tymi o lepszym, ale także uzgadnia rekordy o różnych wartościach, a tej samej jakości, preferując dane z archiwum MASTER.



Rys. Przykład deklaracji synchronizacji archiwum X z archiwum Y przy założeniu identycznych nazw baz danych obu archiwów.

W sytuacji gdy nazwy baz danych obu archiwów różnią się, dla stacji operatorskiej, której archiwum będzie synchronizowane należy skonfigurować dodatkowe 'wirtualne' archiwum do odczytu, ze wskazaniem nazwy serwera i nazwy bazy danych MS SQL archiwum, z

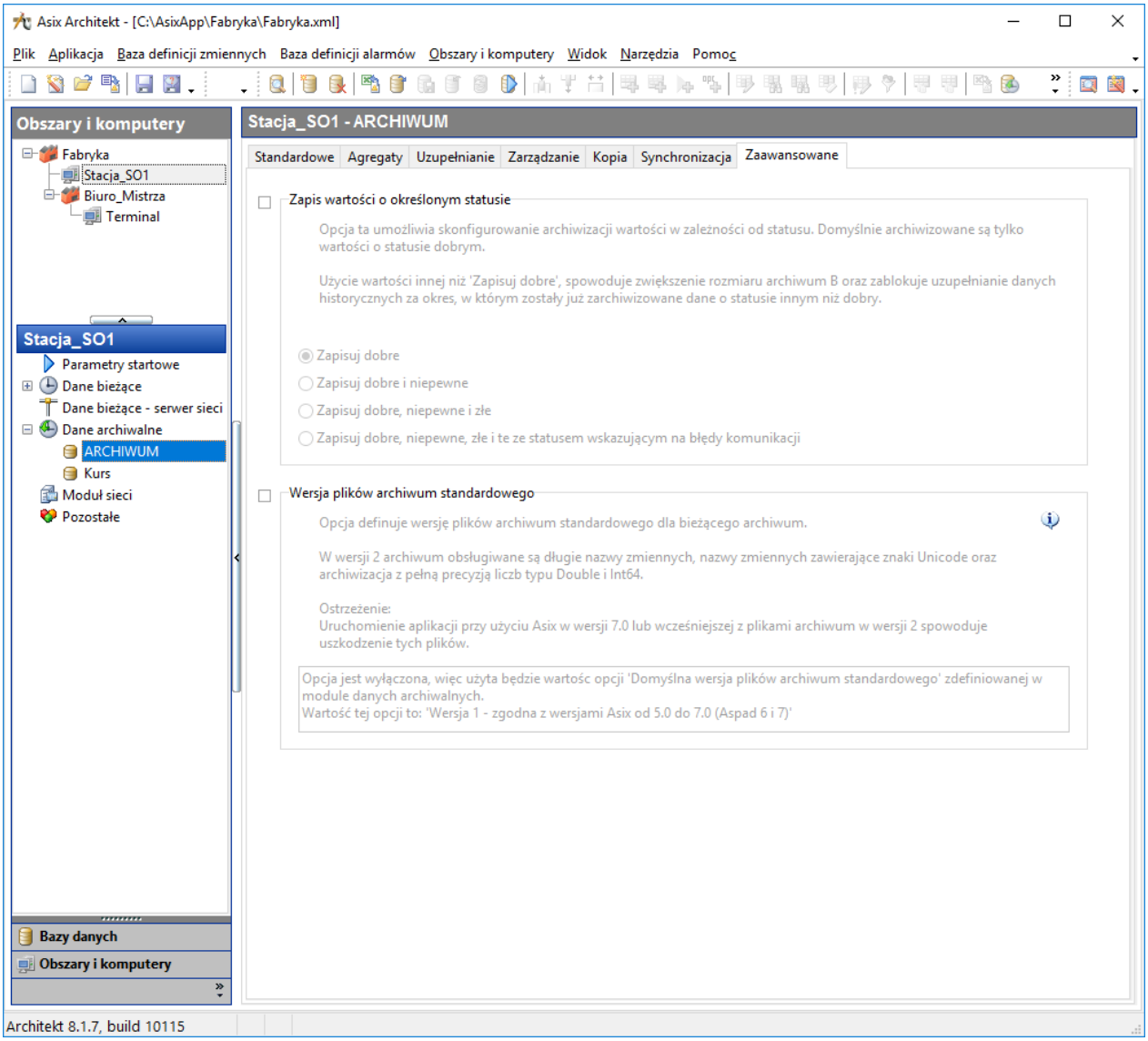
którym dane mają być synchronizowane (pozostałe parametry w konfiguracji 'wirtualnego' archiwum nie mają znaczenia).



Rys. Przykład deklaracji synchronizacji archiwum X z archiwum Y przy założeniu rożnych nazw baz danych obu archiwów.

2.10.9.7. Deklaracja wartości o określonym statusie

Zakładka ***Zaawansowane*** pozwala skonfigurować archiwizację wartości w zależności od statusu:



Rysunek: Okno parametrów archiwum - zakładka 'Zaawansowane'.

- ***Zapis wartości o określonym statusie*** - określa status wartości, które będą archiwizowane

Wartość domyślna: domyślnie archiwizowane są tylko wartości o statusie dobrym.

- ***Domyślna wersja plików archiwum standardowego*** - opcja definiuje wersję plików archiwum standardowego dla bieżącego archiwum:

Wersja 1 - zgodna z wersjami Asix od 5.0 do 7.0 (Aspad 6 i 7).

Wersja 2 - dostępna od wersji Asix 7.1 (Aspad 8).

W wersji 2 archiwum obsługiwane są długie nazwy zmiennych, nazwy zmiennych zawierające znaki UNICODE oraz archiwizacja z pełną precyzją liczb typu Double i Int64.

OSTRZEŻENIE: Uruchomienie aplikacji przy użyciu Asixa w wersji 7.0 lub wcześniejszej z plikami archiwum w wersji 2 spowoduje uszkodzenie tych plików.

INFORMACJE DODATKOWE:
Sprawny odczyt sieciowy archiwum w wersji 2 w obiektach Wykres, Skryptach oraz uzupełnianiu danych lub replikacji jest możliwy tylko przy użyciu Aspada 8 (Asix 7.1) lub nowszego.

W Asixie 5.2 lub starszym odczyt sieciowy archiwum w wersji 2 nie jest możliwy.

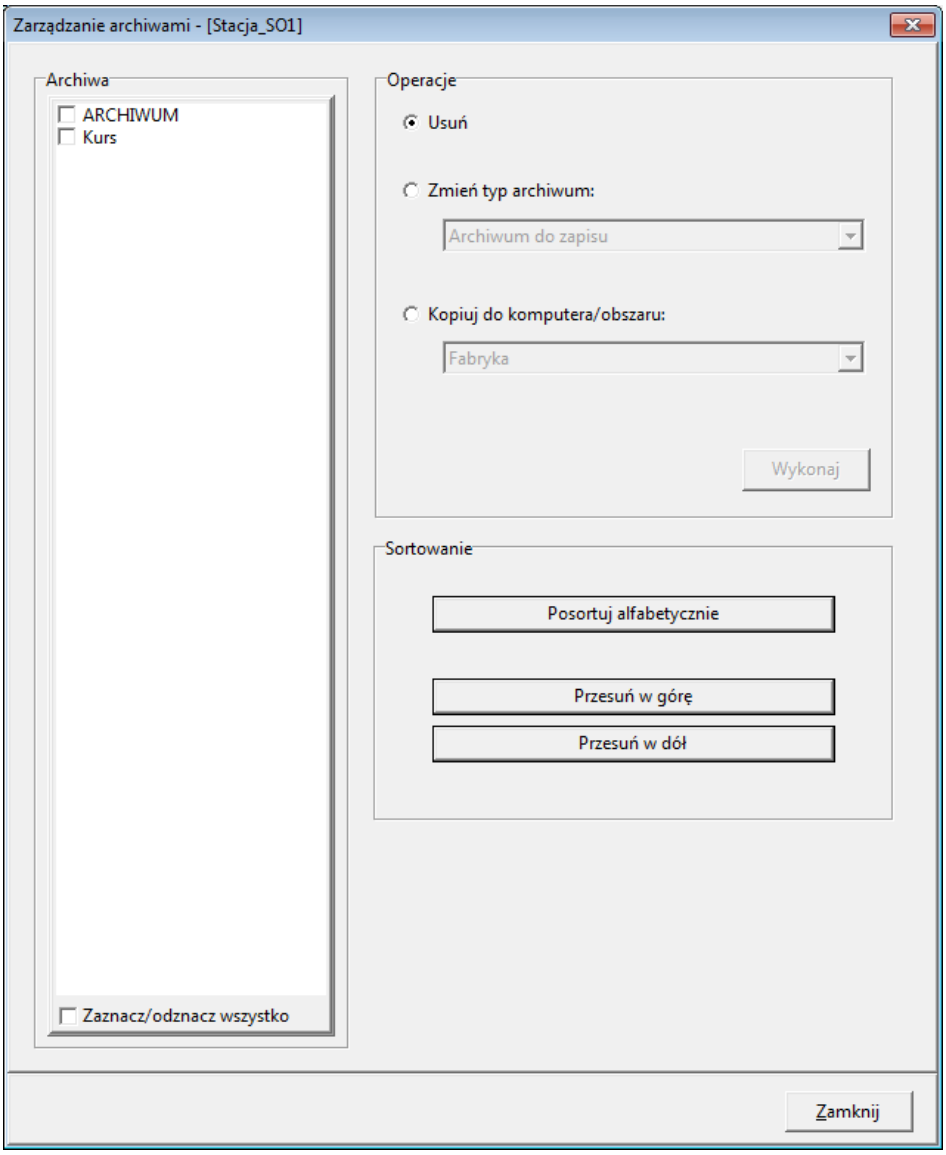
W pozostałych wersjach Asixa konieczna jest aktualizacja Aspada dedykowana dla danego pakietu, a wydajność odczytu będzie zmniejszona.

Obniżenie wydajności przy łączeniu się starszych Aspadów z nowym np. przy replikacji lub uzupełnianiu, wynika z tego, że nie mogą wtedy przesyłać danych w postaci skompresowanej, a tylko po dekompresji, co powoduje zmniejszoną wydajność operacji sieciowych. Jeszcze starsze Aspady (z Asixa 5.2) nie potrafią w tej sytuacji przesyłać ani danych skompresowanych, ani zdekompresowanych.

W przypadku programów: AsixConnectowi i Trendom jest wszystko jedno, czy łączą się z Aspadem 7 czy 8 i czy archiwum jest w wersji 1 czy 2.

2.10.10. Zarządzanie archiwami

W menu uruchamianym po kliknięciu prawym klawiszem myszki na pozycji ***Dane archiwalne*** dostępne jest również polecenie ***Zarządzaj archiwami***, umożliwiające masowe kopiowanie, usuwanie archiwów pomiędzy komputerami/obszarami oraz zmiana typu archiwum (do zapisu, do odczytu, pomocnicze, sieciowe).



Rysunek: Okno 'Zarządzanie archiwami'.

2.11. Konfiguracja modułu sieci

Grupa parametrów ***Moduł sieci*** obejmuje ustawienia pracy modułu Aslink.

ASLINK jest modułem, który umożliwia innym komponentom systemu Asix wzajemną komunikację poprzez lokalną sieć komputerową. Aslink definiuje zestaw usług sieciowych, z których korzystają wszystkie inne programy systemu Asix, sparametryzowane do pracy w sieci.

Podstawowa usługa świadczona między innymi na rzecz takich komponentów pakietu Asix, jak Asmen, Aspad czy podsystem obsługi alarmów - to wiarygodny przesył danych pomiędzy każdą parą procesów użytkowych, działających na dowolnych stanowiskach przyłączonych do sieci komputerowej i realizujących wspólny protokół użytkowy. W tym zakresie usługi modułu Aslink ukierunkowane są na wsparcie komunikacji opartej na istnieniu procesu "serwera" i korzystającego z jego usług procesu ”klienta".

W szczególności Aslink zapewnia:

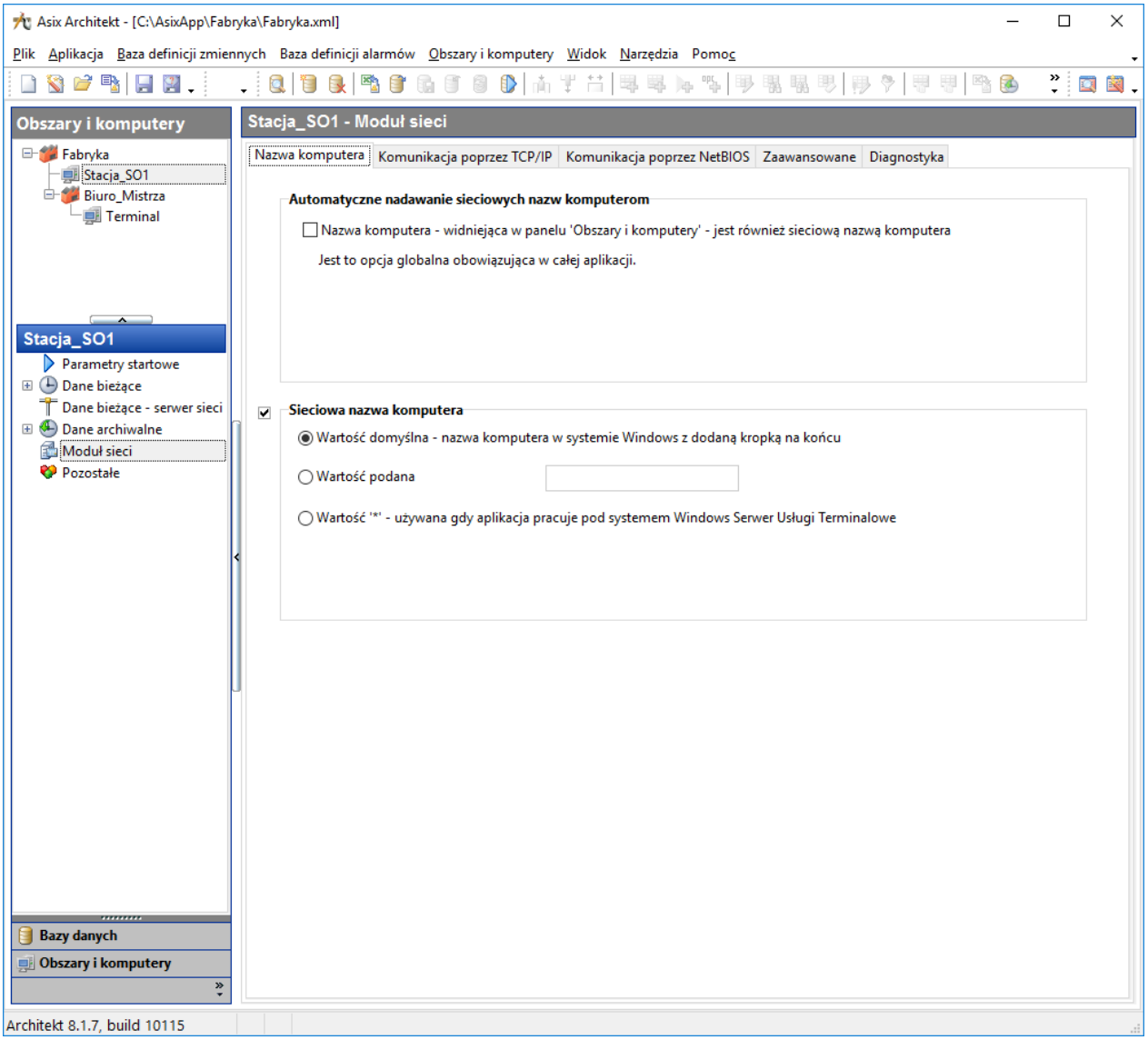
- wyszukanie w sieci procesów pełniących rolę serwerów w odniesieniu do zasobów potrzebnych procesom klientów,
- nawiązanie połączenia (tzw. kanału transmisji) pomiędzy klientem i wskazanym przez niego serwerem zasobu,
- dwukierunkowy, wiarygodny przesył danych w każdym z kanałów,
- powiadomienie procesów, że kanał został zamknięty wskutek awarii sieci lub na jawne polecenie któregoś z partnerów.

Szczegółowy opis funkcjonalności modułu Aslink znajduje się w części Serwer Danych > Obsługa danych procesowych Serwera Danych > rozdziale "*Aslink - moduł sieciowy*".

Ustawienia modułu sieciowego przechowywane są w pliku XML Serwera Danych. Gdy na danym komputerze nie aplikacji (tym samym pliku xml Serwera Danych), za to znajdują się: AsixConnect, AsTrend lub inna aplikacja korzystająca z danych aplikacji Asixa - w takiej sytuacji ustawienia sieciowe dla danego komputera przechowywane są w pliku aslink.ini. Edytor pliku aslink.ini dostępny jest z poziomu Architekta z menu *Narzędzia* i obejmuje konfigurację dokładnie tych samych opcji co *Moduł sieci*.

2.11.1. Nazwa komputera

Zakładka ***Nazwa komputera*** obejmuje parametryzację następujących elementów:



Rys. Okno parametrów modułu sieci - zakładka 'Nazwa komputera'.

- ***Automatyczne nadawanie sieciowych nazw komputerom*** - zaznaczenie opcji spowoduje, że nazwa komputera widniejąca w panelu 'Obszary i komputery' zostanie sieciową nazwą komputera;

  Wartość domyślna: automatycznie przydzielana jest nazwa komputera w systemie Windows, zdefiniowana w czasie parametryzacji sieci z dodaną na końcu kropką.

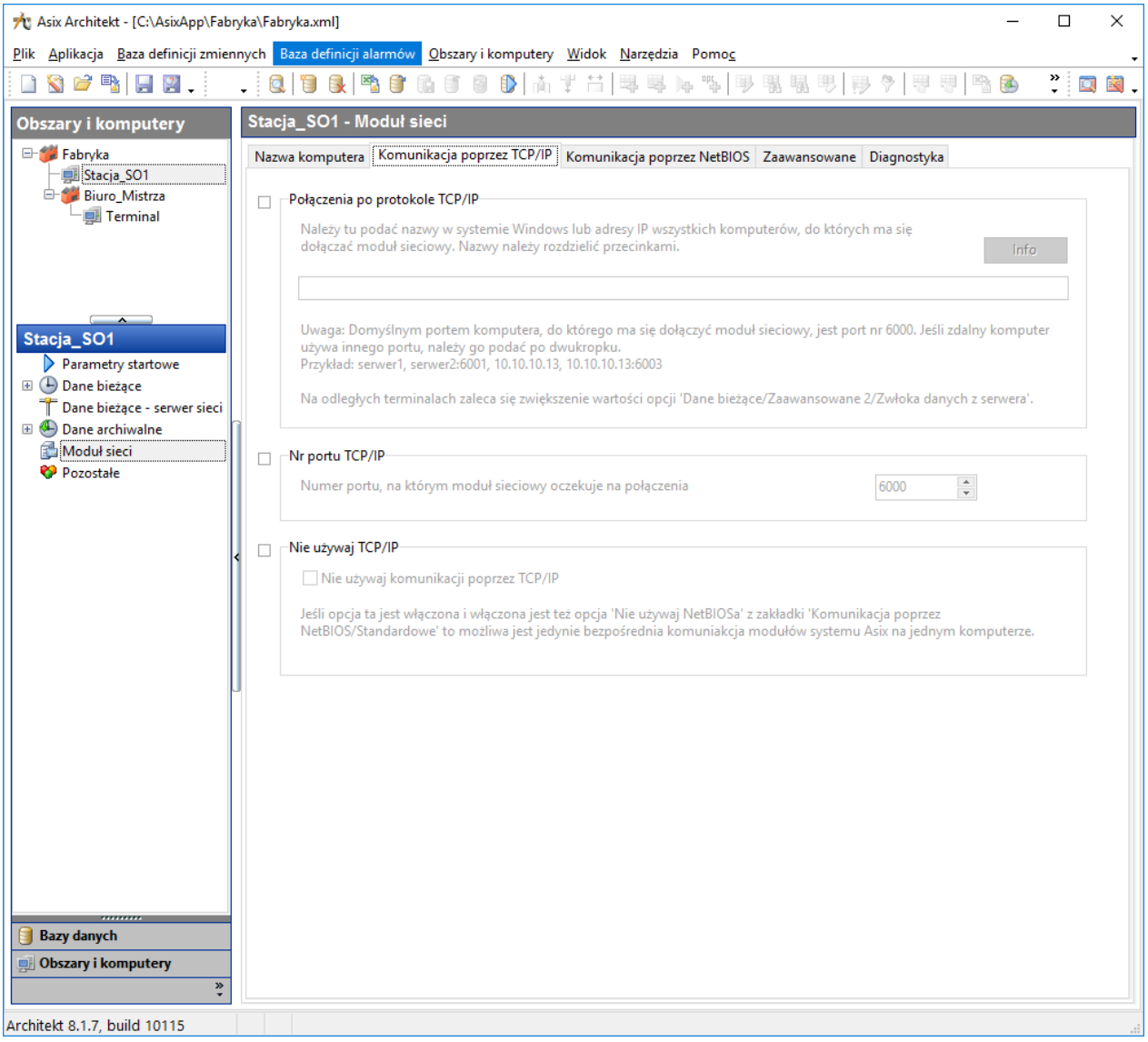
- ***Sieciowa nazwa komputera*** - nazwa sieciowa komputera; jeśli nie podano w tym polu nazwy, to zostanie wygenerowana nazwa na podstawie nazwy

komputera w systemie Windows, zdefiniowanej w czasie parametryzacji sieci z dodaną na końcu kropką;
Wartość domyślna: nazwa przydzielona automatycznie

Dopuszczalne wartości: tekst o maks. 15 znakach. Wielkość znaków ma znaczenie. Nazwa nie może zwierać znaku ”*”. Nie powinna również zawierać znaku ”,”

2.11.2. Komunikacja poprzez TCP/IP

Zakładka ***Komunikacja poprzez TCP/IP*** obejmuje parametryzację następujących elementów:



Rys. Okno parametrów modułu sieci - zakładka 'Komunikacja poprzez TCP/IP'.

- ***Połączenia po protokole TCP/IP*** - opcja wymaga podania nazwy w systemie Windows lub adresu IP wszystkich komputerów, do których ma się dołączać moduł sieciowy; nazwy należy rozdzielić przecinkami.

  Domyślnym portem komputera, do którego ma się dołączać moduł sieciowy, jest port numer 6000. Jeśli zdalny komputer używa innego portu, należy go podać po dwukropku, np.: serwer2:6001, 10.10.10.13:6003.

  Na odległych serwerach zaleca się zwiększenie wartości opcji *Dane bieżące > Zaawansowane 2 > Zwłoka danych z serwera*

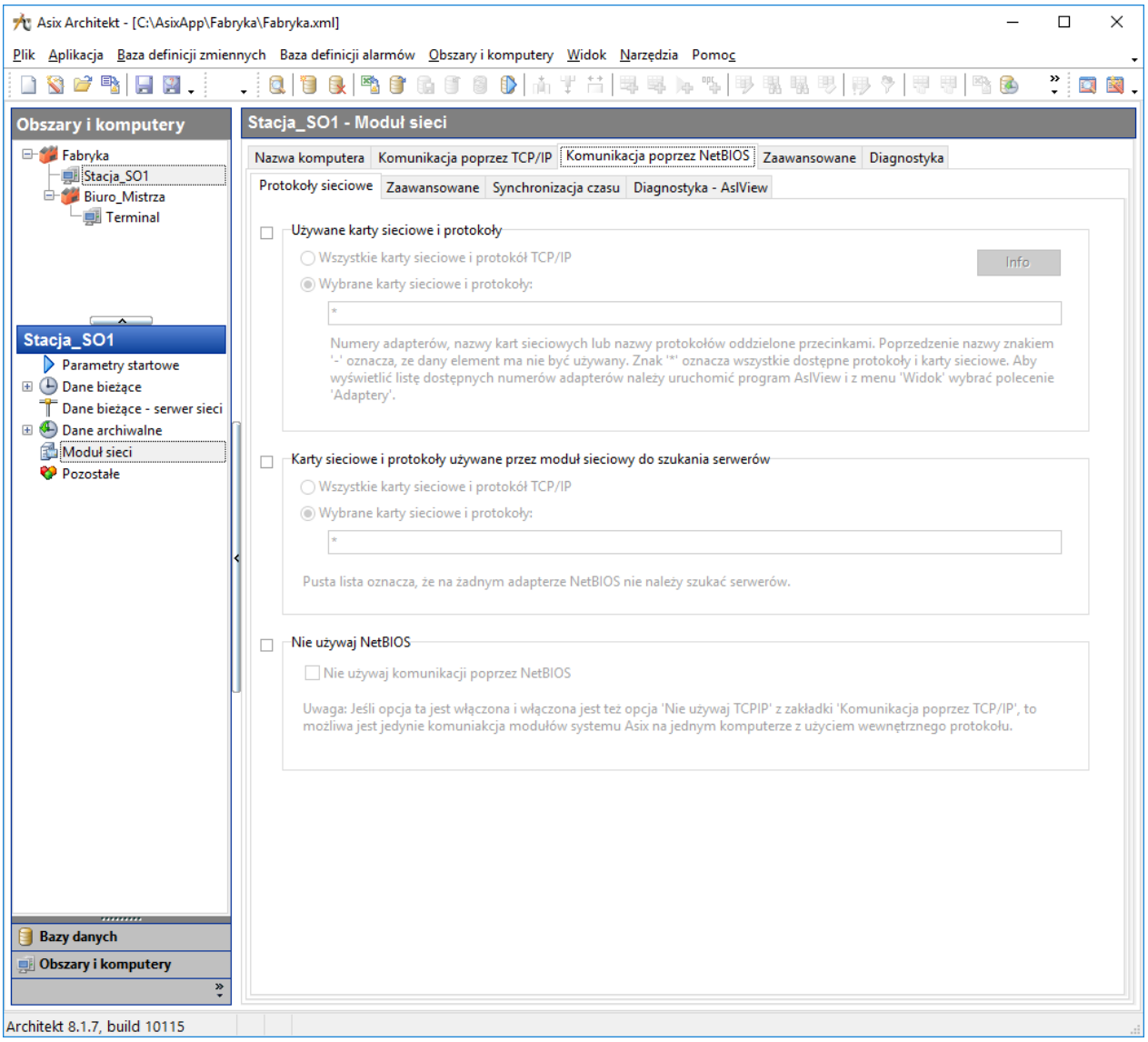
  Wartość domyślna: brak połączenia

- ***Nr portu TCP/IP*** - numer portu, na którym moduł sieciowy oczekuje na połączenie.
  Wartość domyślna: 6000

- ***Nie używaj TCP/IP*** - załączenie opcji oraz jednocześnie opcji ***Nie używaj NetBIOS*** z zakładki *Komunikacja przez NetBIOS/Protokoły sieciowe* powoduje, że możliwa jest jedynie komunikacja modułów systemu Asix na jednym komputerze z użyciem wewnętrznego protokołu.
  Wartość domyślna: opcja wyłączona

2.11.3. Komunikacja poprzez NetBIOS

Zakładka ***Komunikacja poprzez NetBIOS/Protokoły sieciowe*** obejmuje deklarację kart i protokołów sieciowych.



Rys. Okno parametrów modułu sieci - zakładka 'Komunikacja przez NetBIOS/Protokoły sieciowe'.

- ***Używane karty sieciowe i protokoły*** - opcja określa wykorzystywane do komunikacji protokoły sieciowe:
  - ***Wszystkie karty sieciowe i protokół TCP/IP***
  - ***Wybrane karty sieciowe i protokoły*** - pozwala zadeklarować nazwy kart sieciowych, protokołów lub numery adapterów oddzielone przecinkami; poprzedzenie nazwy znakiem '-' oznacza, że dany adapter nie może być używany; znak '*' oznacza wszystkie dostępne protokoły i karty sieciowe. Aby wyświetlić listę dostępnych adapterów, należy uruchomić program AslView i z menu *Widok* wybrać polecenie *Adaptery*.

Wartość domyślna: 'TCPIP'

- ***Karty sieciowe i protokoły używane przez moduł sieciowy do szukania serwerów*** - określa adaptery logiczne wykorzystywane przez moduł sieciowy do szukania serwerów;
  - ***Wszystkie karty sieciowe i protokół TCP/IP***
  - ***Wybrane karty sieciowe i protokoły***

  Wartość domyślna: wszystkie adaptery wykorzystywane przez moduł sieciowy.

Pozycje używane na komputerach typu 'POMOST'.

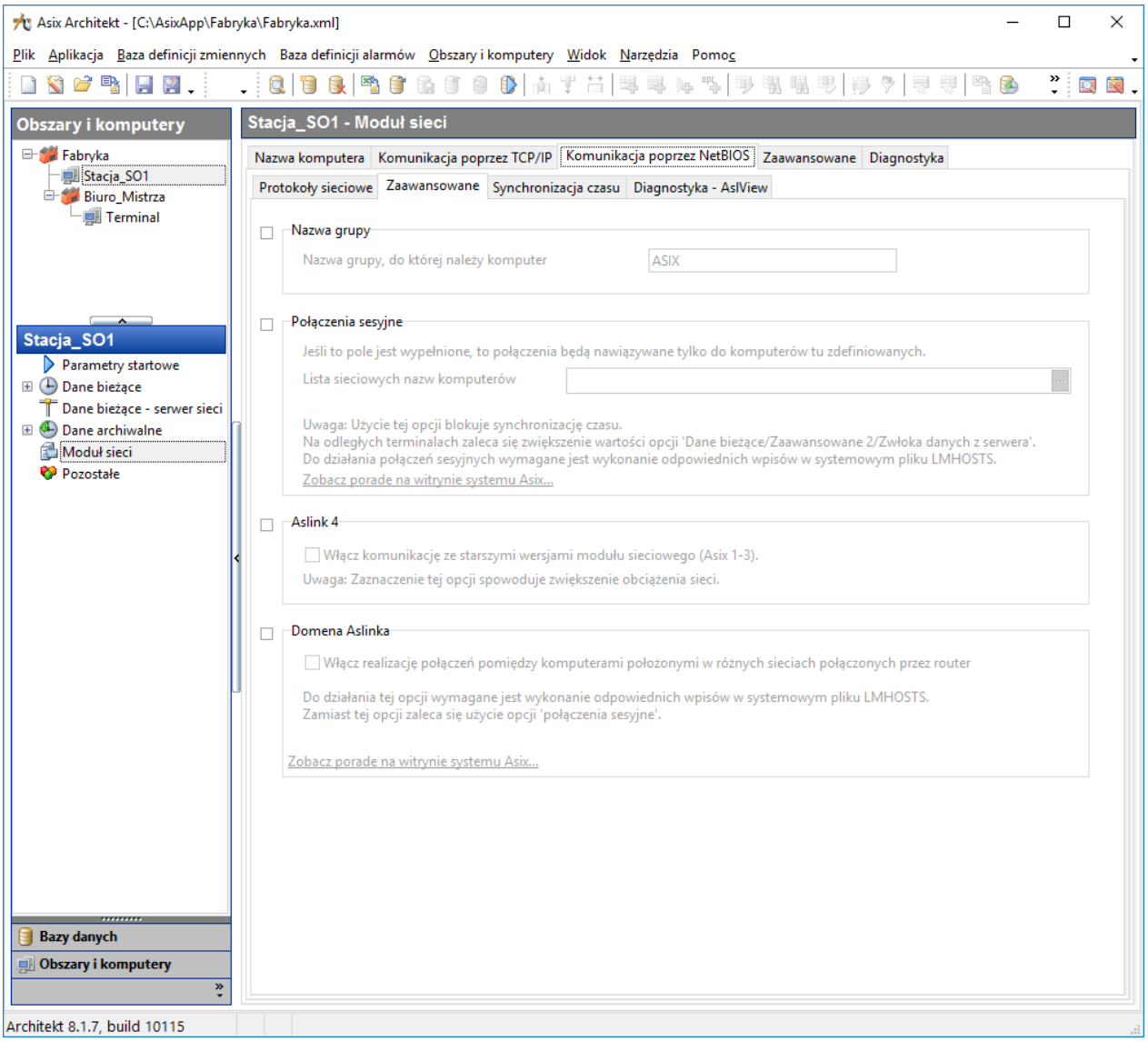
Dopuszczalne wartości: numery adapterów oddzielone przecinkami; jeżeli numer adaptera poprzedzony jest znakiem '-', to adapter nie zostanie użyty; jeśli podano '*', to zostaną wykorzystane wszystkie dostępne adaptery.

- ***Nie używaj NetBIOS*** - załączenie opcji oraz jednocześnie opcji ***Nie używaj TCP/IP*** z zakładki *Komunikacja poprzez TCP/IP* powoduje, że możliwa jest jedynie komunikacja modułów systemu Asix na jednym komputerze z użyciem wewnętrznego protokołu.

  *W*artość domyślna: opcja wyłączona

Zakładka *Zaawansowane* pojawia się po wybraniu polecenia *Widok* > *Pokaż opcje zaawansowane* z menu okna programu Architekt.

Zakładka ***Komunikacja poprzez NetBIOS/Zaawansowane*** obejmuje parametryzację następujących opcji:



Rys. Okno parametrów modułu sieci - zakładka 'Komunikacja przez NetBIOS/Zaawansowane'.

- ***Nazwa grupy*** - nazwa grupy, do której należy komputer;
  Wartość domyślna: ASIX

- ***Połączenia sesyjne*** - jeśli pole to jest wypełnione, to połączenia będą nawiązywane tylko do zdefiniowanych komputerów;
  Wartość domyślna: brak połączenia

  Uwaga:

  Użycie tej opcji blokuje synchronizację czasu.

  Na odległych terminalach zaleca się zwiększenie wartości opcji *Dane bieżące > Zaawansowane 2 > Zwłoka danych z serwera*

  Do działania połączeń sesyjnych wymagane jest wykonanie odpowiednich wpisów w systemowych plikach LMHOSTS.

- ***Aslink 4*** - opcja załącza komunikację ze starszymi wersjami modułu sieciowego (Asix 1-3);
  Wartość domyślna: komunikacja ze starszymi wersjami modułu sieciowego załączona

  Uwaga:

  Zaznaczenie tej opcji spowoduje zwiększenie obciążenia sieci.

- ***Domena Aslinka*** - włącza realizację połączeń pomiędzy komputerami położonymi w różnych sieciach połączonych przez router;
  Wartość domyślna: opcja wyłączona

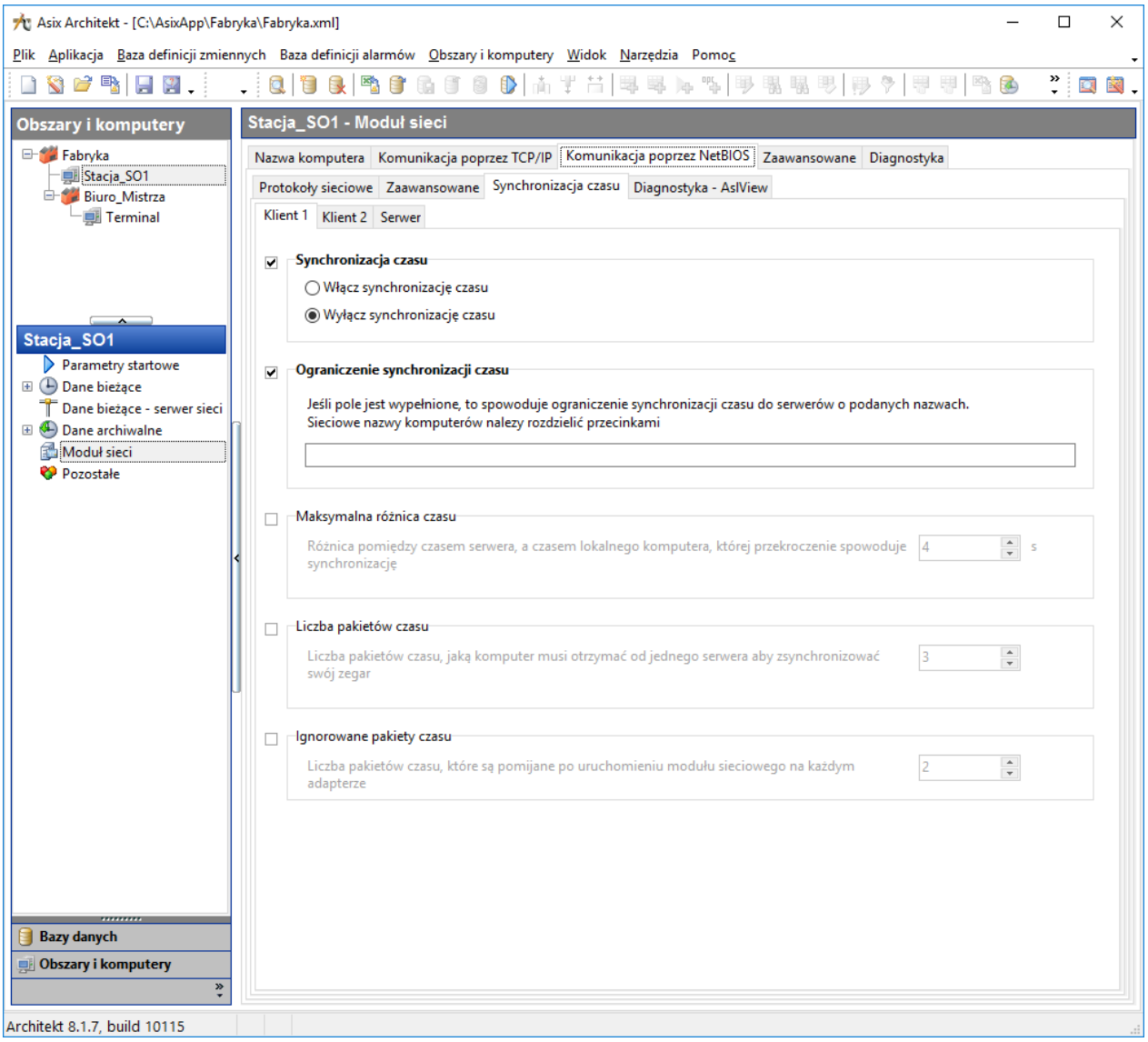
  Uwaga:

  Do działania tej opcji wymagane jest wykonanie odpowiednich wpisów w systemowych plikach LMHOSTS.

  Zamiast tej opcji zaleca się użycie opcji Połączenia sesyjne.

Zakładka *Synchronizacja czasu* pojawia się po wybraniu polecenia *Widok* > *Pokaż opcje zaawansowane* z menu okna programu Architekt.

Zakładka ***Komunikacja poprzez NetBIOS/Synchronizacja czasu/Klient 1*** umożliwia parametryzację synchronizacji czasu od strony klienta.



Rys. Okno parametrów modułu sieci - zakładka 'Komunikacja przez NetBIOS/Synchronizacja Czasu/Klient 1'.

- ***Synchronizacja czasu*** - opcja pozwala załączyć synchronizację czasu;
  Wartość domyślna: czas synchronizowany jest z wszystkimi dostępnymi serwerami czasu

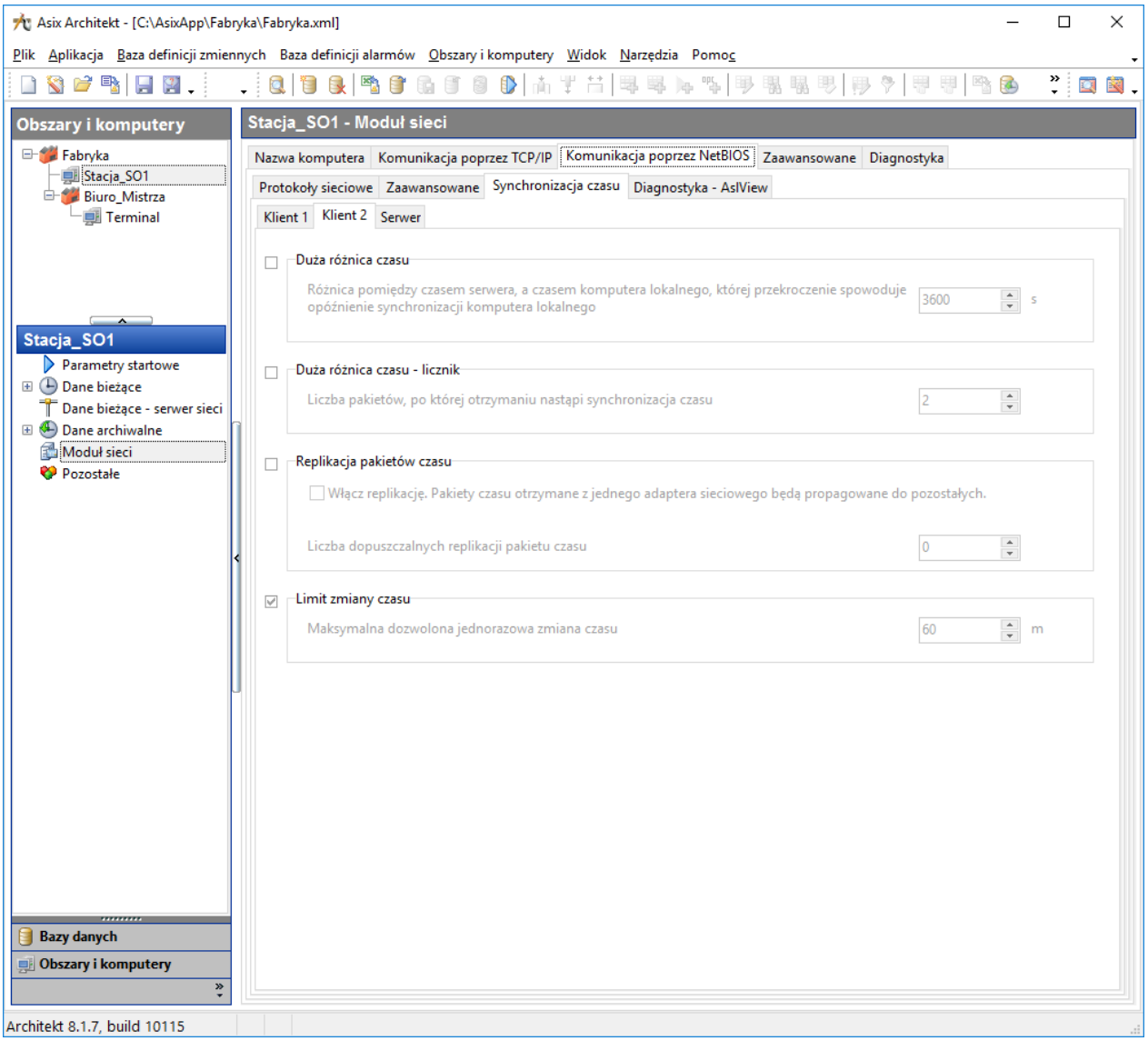
- ***Ograniczenia synchronizacji czasu*** - jeśli pole jest wypełnione, to synchronizacja czasu zostanie ograniczona do serwerów o podanych nazwach; sieciowe nazwy komputerów należy podać, oddzielając je przecinkami;
  Wartość domyślna: czas synchronizowany jest z wszystkimi dostępnymi serwerami czasu

- ***Maksymalna różnica czasu*** - różnica pomiędzy czasem serwera a czasem stacji lokalnej, której przekroczenie spowoduje synchronizację czasu lokalnego (CMOSu); wartość w sekundach;
  Wartość domyślna: 4 sekundy

- ***Liczba pakietów czasu*** - liczba pakietów czasu, jaką stacja musi otrzymać od jednego serwera, aby zsynchronizować swój zegar;
  Wartość domyślna: 3 pakiety

- ***Ignorowane pakiety czasu*** - liczba pakietów czasu, które są pomijane po uruchomieniu modułu sieciowego na każdym adapterze;
  Wartość domyślna: 2

Zakładka ***Komunikacja poprzez NetBIOS/Synchronizacja czasu/Klient 2*** umożliwia parametryzację synchronizacji czasu od strony klienta.



Rys. Okno parametrów modułu sieci - zakładka 'Komunikacja przez NetBIOS/Synchronizacja Czasu/Klient 2'.

- ***Duża różnica czasu/Duża różnica czasu - licznik*** określa:

  - różnicę (czas w milisekundach) pomiędzy czasem serwera czasu a czasem stacji lokalnej, której przekroczenie powoduje specjalne traktowanie odbieranych pakietów czasu, polegające na opóźnionej synchronizacji czasu stacji lokalnej;
    Wartość domyślna: 3600 milisekund

  - liczbę pakietów, po której otrzymaniu nastąpi synchronizacja czasu;
    Wartość domyślna: 2

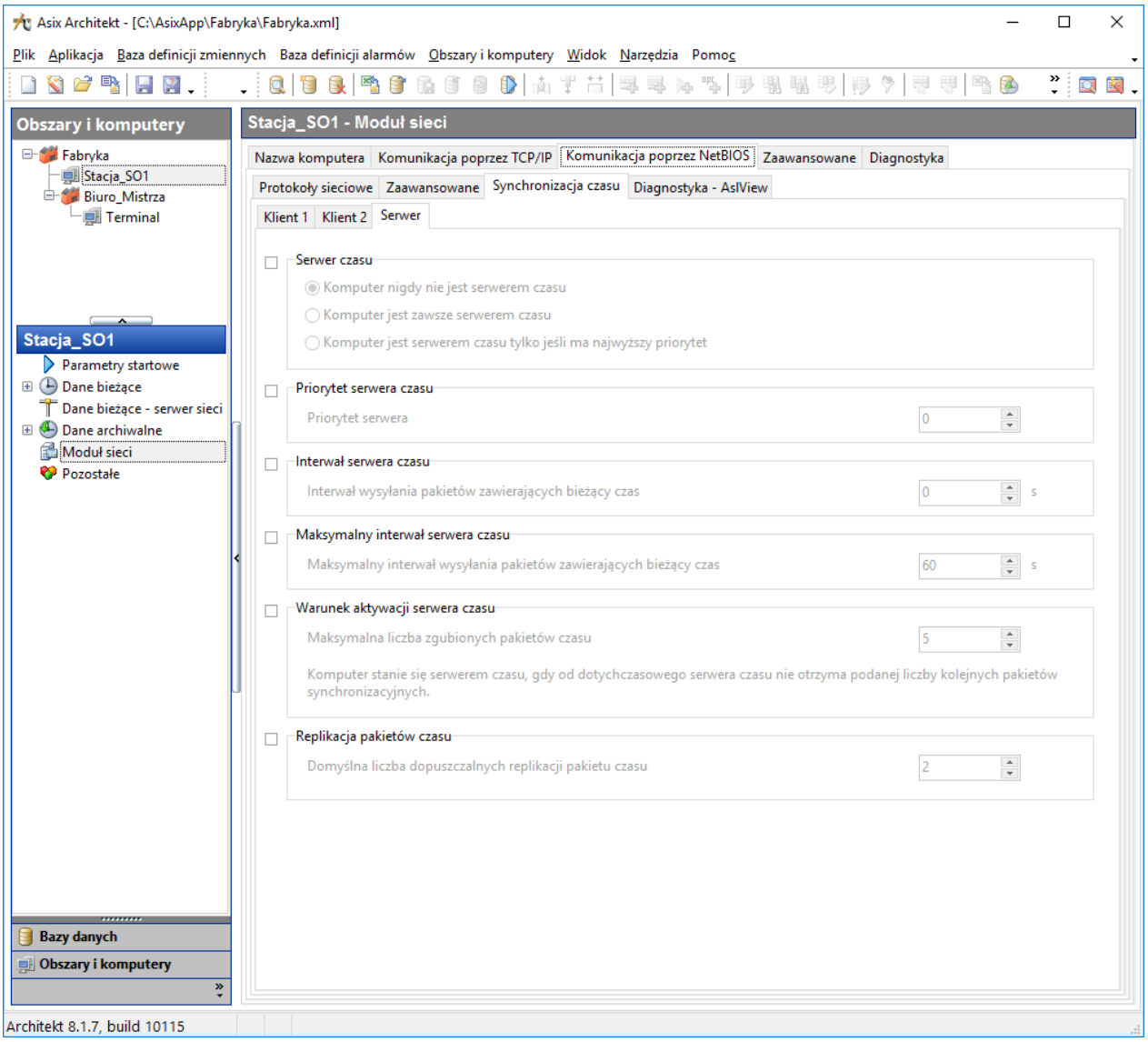
- ***Replikacja pakietów czasu*** - włączenie opcji spowoduje, że wszystkie odbierane z danego adaptera pakiety czasu będą rozsyłane do pozostałych adapterów; dodatkowo można określić liczbę dopuszczalnych replikacji pakietu czasu.
  Wartość domyślna: replikacja czasu wyłączona

- ***Limit zmiany czasu*** - pozycja pozwala zdefiniować jaka jest maksymalna dozwolona jednorazowa zmiana czasu;
  Wartość domyślna: domyślnie zmiany czasu nie są ograniczone

---

Podanie ujemnego limitu zmiany całkowicie blokuje możliwość modyfikacji czasu.

Zakładka ***Komunikacja poprzez NetBIOS/Synchronizacja czasu/Serwer*** umożliwia parametryzację synchronizacji czasu od strony serwera.



Rys. Okno parametrów modułu sieci - zakładka 'Komunikacja przez NetBIOS/Synchronizacja Czasu/Serwer'.

- ***Serwer czasu*** - określa, czy stacja będzie pracować jako serwer czasu (tj. rozgłaszać w sieci informacje o bieżącym czasie):
  - ***Komputer nigdy nie jest serwerem czasu***,
  - ***Komputer jest zawsze serwerem czasu -*** nawet jeśli w sieci pracuje serwer czasu o wyższym priorytecie,
  - ***Komputer jest serwerem czasu tylko jeśli ma najwyższy priorytet***;

  Wartość domyślna: komputer nigdy nie jest serwerem czasu

- ***Priorytet serwera czasu*** - określa priorytet stacji jako serwera czasu; im większa wartość tym większy priorytet (0 - 32768). Wartość priorytetu przesyłana jest razem z aktualnym czasem w pakietach synchronizacyjnych (BROADCAST). Jeżeli aktywny w danej chwili serwer czasu odbierze pakiet zawierający wyższy priorytet, to przestaje być aktywnym serwerem czasu.

Wartość domyślna: 0

- ***Interwał serwera czasu -*** określa interwał czasu (w sekundach), z jakim wysyłane są do sieci pakiety zawierające bieżący czas stacji.
  Wartość domyślna: 0

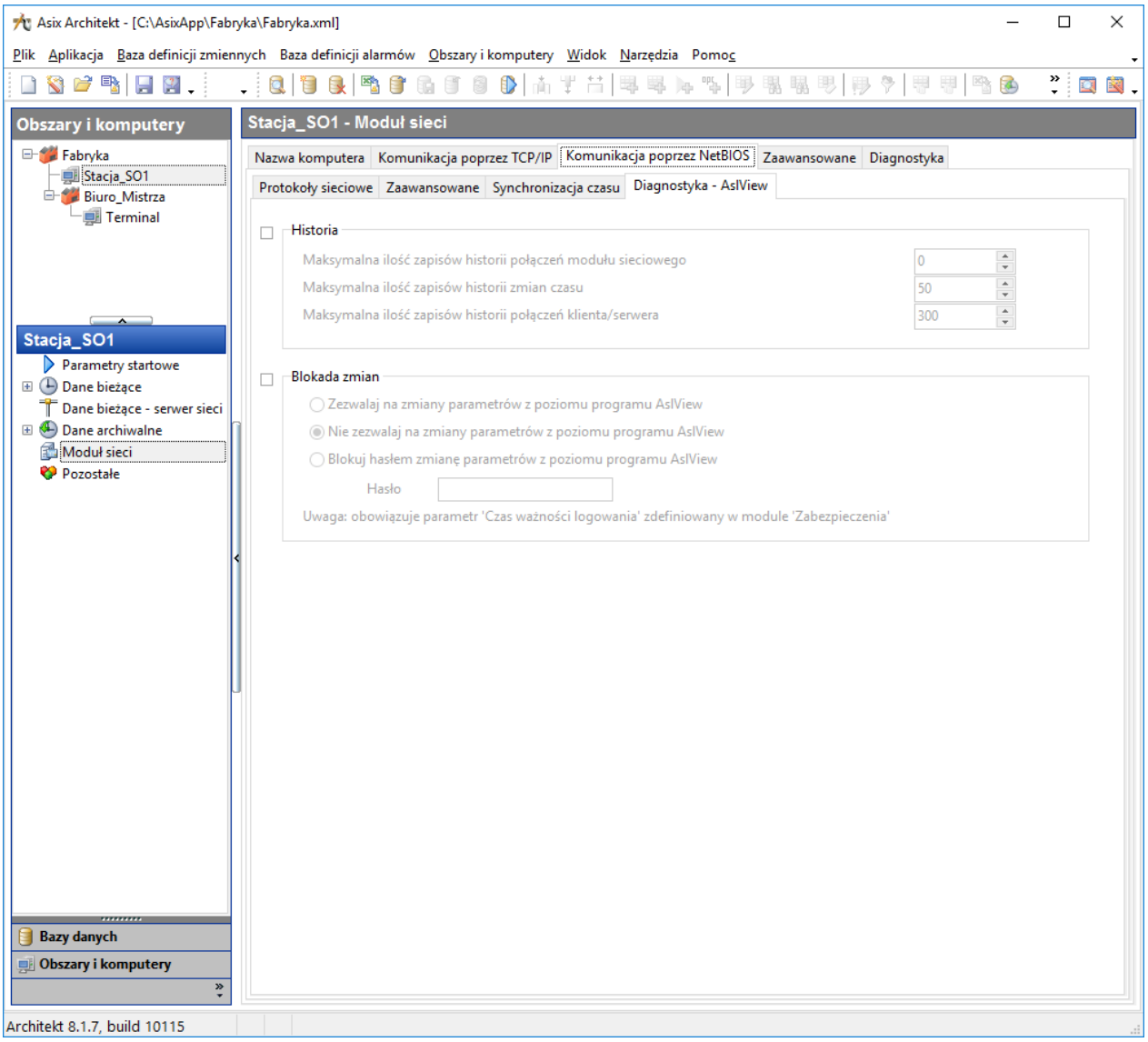
- ***Maksymalny interwał serwera czasu*** - określa maksymalny interwał wysyłania pakietów zawierających bieżący czas
  Wartość domyślna: 60 sekund

- ***Warunek aktywacji serwera czasu -*** stacja stanie się serwerem czasu, gdy od dotychczasowego serwera czasu nie dostanie określonej przez użytkownika liczby kolejnych pakietów synchronizacyjnych;
  Wartość domyślna: 5

- ***Replikacja pakietów czasu*** - określa liczbę dopuszczalnych replikacji pakietu czasu; maksymalna ilość kolejnych stacji, z jakich może się składać trasa pakietu czasu. Parametr ma znaczenie tylko na stacji będącej serwerem czasu.
  Wartość domyślna: 2

Zakładka *Diagnostyka - AslView* pojawia się po wybraniu polecenia Widok > *Pokaż opcje zaawansowane* z menu okna programu Architekt.

Zakładka ***Komunikacja poprzez NetBIOS/Diagnostyka - AslView*** dotyczy konfiguracji informacji diagnostycznych przekazywanych do programów AslView przyłączonych do modułu sieciowego.



Rys. Okno parametrów modułu sieci - zakładka 'Diagnostyka' / 'AslView'.

- ***Historia*** - określa:
  - ***Maksymalna ilość zapisów w historii połączeń modułu sieciowego***;
    Wartość domyślna: 0;

  - ***Maksymalna ilość zapisów w historii zmian czasu***;
    Wartość domyślna: 50;

  - ***Maksymalna ilość zapisów w historii połączeń klienta/serwera***;
    Wartość domyślna: 300;

- ***Blokada zmian*** - parametr określa uprawnienia dotyczące zmiany parametrów z poziomu programu AslView:
  - ***Zezwalaj na zmiany parametrów z poziomu programu AslView***;
  - ***Nie zezwalaj na zmiany parametrów z poziomu programu AslView***;
  - ***Blokuj hasłem zmianę parametrów z poziomu programu AslView***;

Hasło - parametr określa hasło, które musi wprowadzić użytkownik programu AslView przed dokonaniem czynności powodujących istotne zmiany w funkcjonowaniu modułu sieciowego.

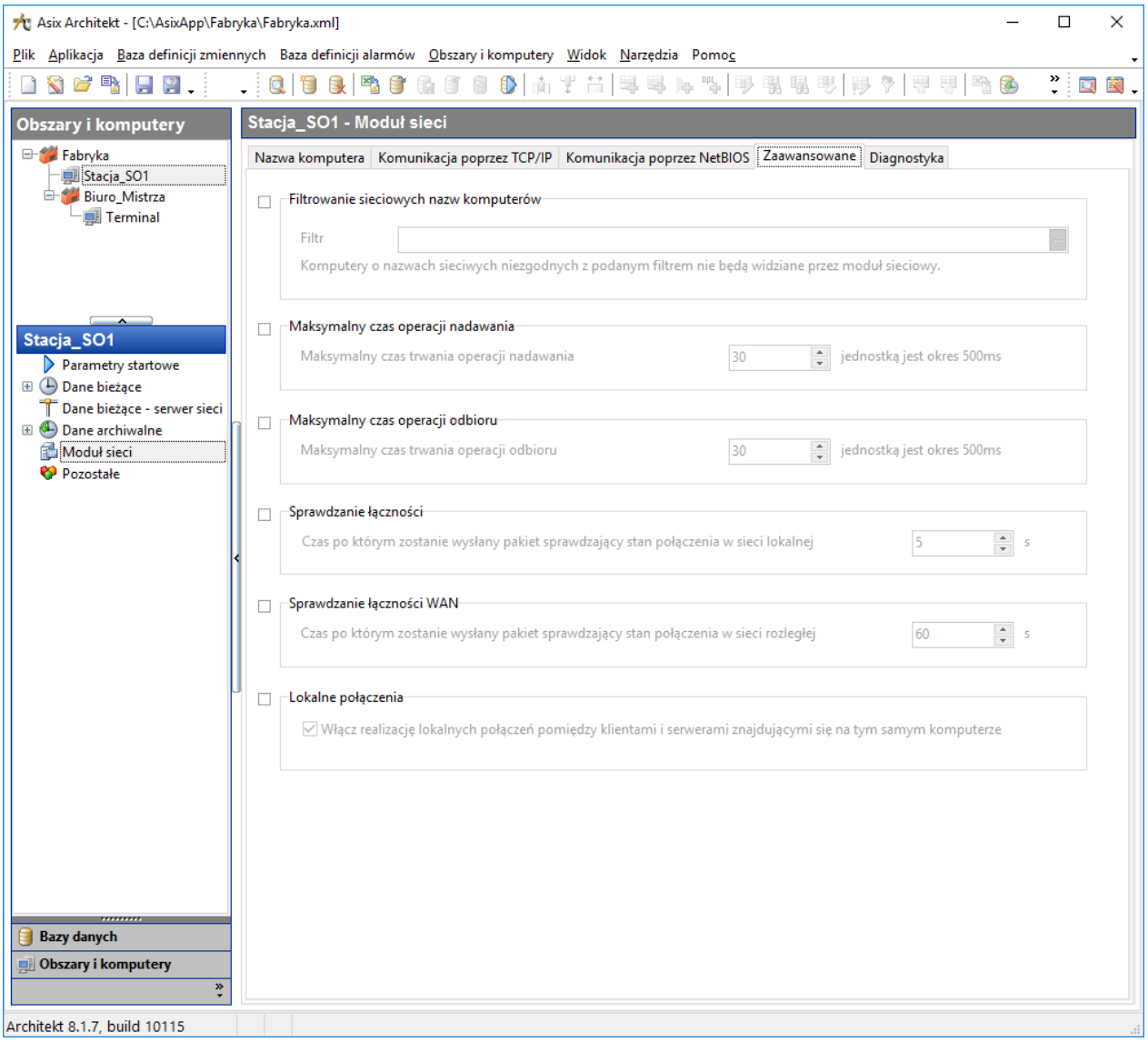
Wartość domyślna: nie ma zezwolenia na zmiany parametrów z poziomu programu AslView.

Obowiązuje parametr *Czas ważności logowania* zdefiniowany w module *Zabezpieczenia*.

2.11.4. Parametry zaawansowane

 Zakładka *Zaawansowane* pojawia się po wybraniu polecenia Widok > *Pokaż opcje zaawansowane* z menu okna programu Architekt.

Zakładka ***Zaawansowane*** pozwala określić następujące opcje.



Rys. Okno parametrów modułu sieci - zakładka 'Zaawansowane'.

- ***Filtrowanie sieciowych nazw komputerów*** **-** określenie tekstowego filtru, jakiemu podlegać będą nazwy stacji zdalnych; stacje o nazwach niezgodnych z filtrem nie będą widoczne; nazwa może zawierać znaki ? i * (tylko na końcu); nazwy kilku stacji podaje się oddzielając je przecinkami;
  Wartość domyślna: brak (wszystkie stacje są widoczne)

- ***Maksymalny czas operacji nadawania*** - maksymalny czas trwania operacji nadawania; jednostką jest okres czasu równy 500ms;
  Wartość domyślna: 30 (15 sekund)

- ***Maksymalny czas operacji odbioru*** **-** maksymalny czas trwania operacji odbioru; jednostką jest okres czasu równy 500ms;
  Wartość domyślna: 30 (15 sekund)

- ***Sprawdzanie łączności*** **-** czas, po którym zostanie wysłany pakiet sprawdzający stan połączenia w sieci lokalnej;

Wartość domyślna: 5 sekund

- ***Sprawdzanie łączności WAN -*** czas, po którym zostanie wysłany pakiet sprawdzający stan połączenia w sieci rozległej;
  Wartość domyślna: 60 sekund

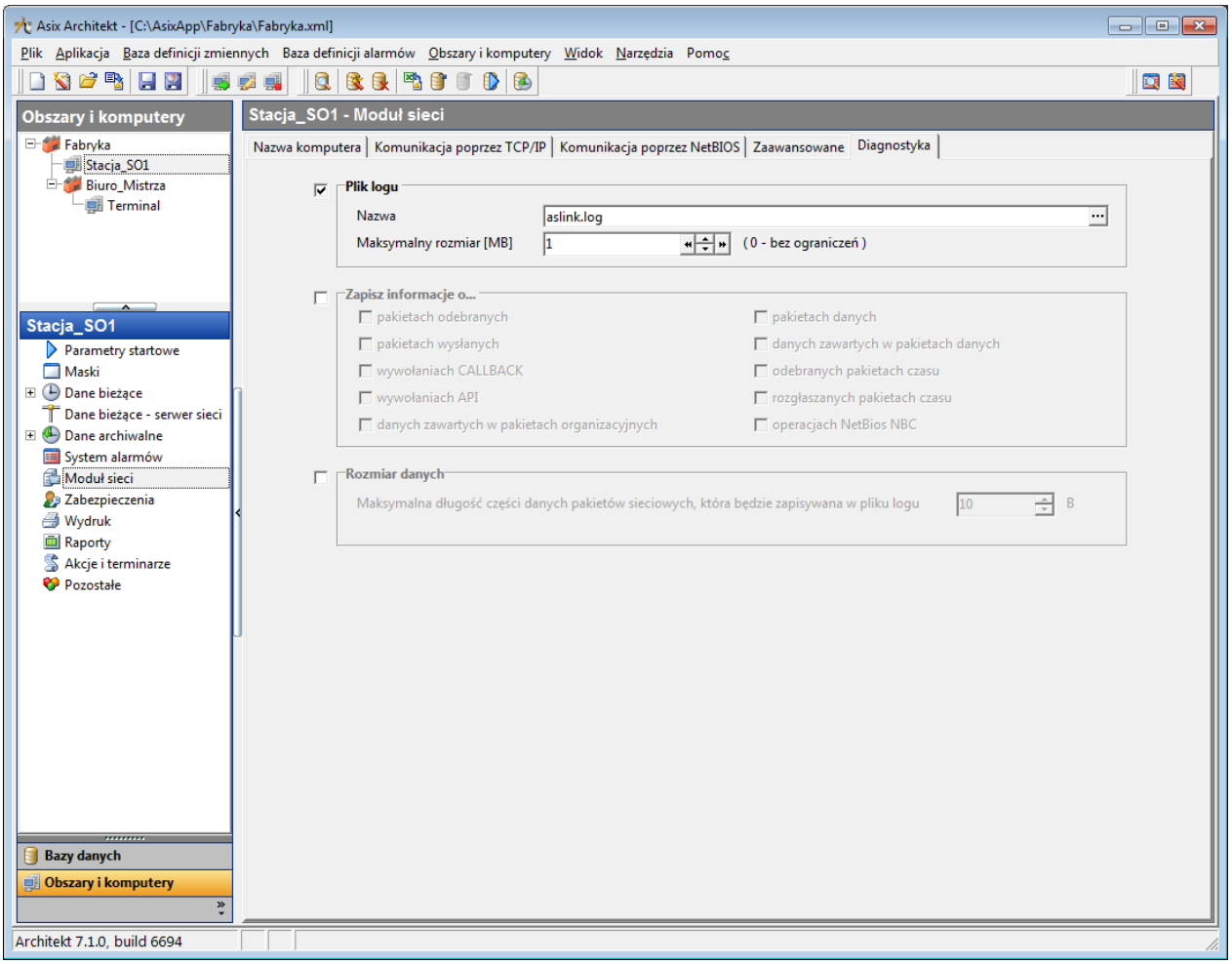
- ***Lokalne połączenia*** - możliwe połączenia pomiędzy klientami i serwerami znajdującymi się na tej samej stacji bez udziału oprogramowania sieciowego.
  Wartość domyślna: opcja jest włączona

Włączenie opcji jest koniczne w przypadku korzystania z programu AsixConnect pobierającego dane z lokalnego Asmena i Aspada.

2.11.5. Parametry diagnostyczne

Zakładka *Diagnostyka* pojawia się po wybraniu polecenia Widok > *Pokaż opcje zaawansowane* z menu okna programu Architekt.

Zakładka ***Diagnostyka*** obejmuje parametry sterujące informacjami diagnostycznymi wyprowadzanymi przez moduł Aslink:



Rys. Okno parametrów modułu sieci - zakładka 'Diagnostyka'.

- ***Plik logu*** - definiuje nazwę pliku, w którym będą zapisywane informacje diagnostyczne oraz maksymalny rozmiar pliku (w MB)
  Wartość domyślna: plik o nazwie 'aslink.log' o maksymalnym rozmiarze: 500 000 KB

- ***Zapisz informacje o ...*** - pozwala określić rodzaj informacji, która zostanie zapisana do pliku logu; istnieje możliwość zadeklarowania zapisu informacji o:
  - ***Pakietach odebranych,***
  - ***Pakietach wysłanych,***
  - ***Wywołania CALLBACK,***
  - ***Wywołania API,***
  - ***Danych zawartych w pakietach organizacyjnych,***
  - ***Pakietach danych,***
  - ***Danych \zawartych w pakietach danych,***
  - ***Rozgłaszanych pakietach czasu,***
  - ***Operacjach NetBios NBC***

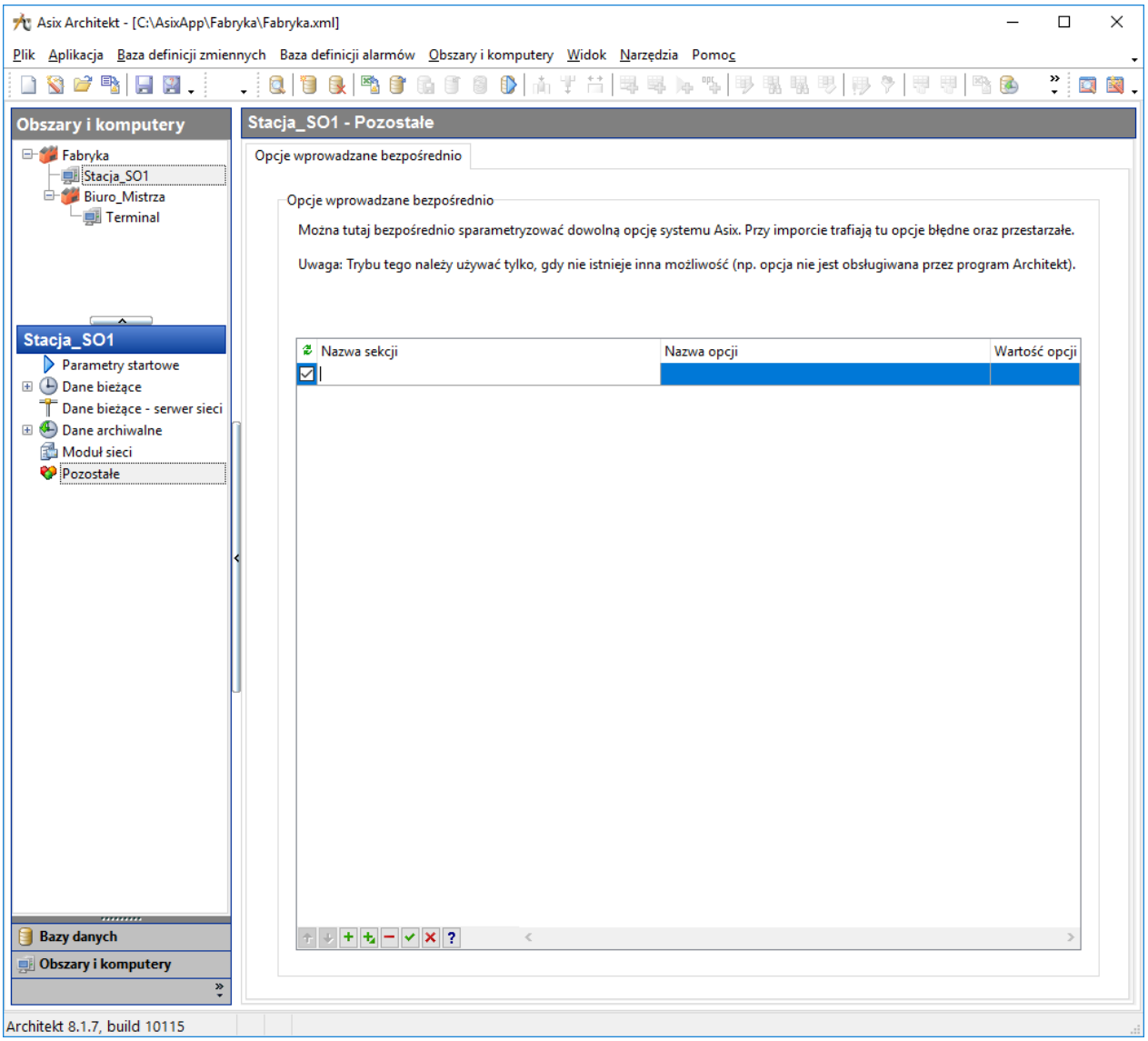
  Wartość domyślna: żaden rodzaj informacji nie jest zadeklarowany

- ***Rozmiar danych -*** określa maksymalną długość części danych pakietów sieciowych, która będzie zapisywana w pliku log;
  Wartość domyślna: 10 B.

2.12. Grupa Pozostałe

Grupa *Pozostałe* zawiera parametry nie zaklasyfikowane do pozostałych grup. Umożliwia deklarowanie opcji, które nie są obsługiwane przez edytory programu Architekt i należy je dopisać ręcznie bezpośrednio do pliku XML, podając nazwę sekcji, opcji i wartość opcji.

2.12.1. Deklaracja parametrów wprowadzanych bezpośrednio



Rys. Okno parametrów pozostałych - zakładka 'Opcje wprowadzane bezpośrednio'.

Zakładka ***Opcje wprowadzane bezpośrednio*** pozwala zadeklarować dla danej aplikacji pozycje, których edycja nie jest wspierana przez moduł Architekt - tzn. są to pozycje używane w starszych wersjach pakietu Asix lub zupełnie nowe pozycje.

Załącznik 1

Tabela 1. Lista atrybutów systemowych Bazy Definicji Zmiennych definiujących zmienne procesowe.

Nazwa wewnętrzna atrybutu zmiennej (PL)	Rodzaj atrybutu	Nawa domyślnie wyświetlana w nagłówku (PL)	Nazwa wewnętrzna atrybutu zmiennej (EN)	Nawa domyślnie wyświetlana w nagłówku (EN)	Typ	Opis
Nazwa	obligatoryjny	Nazwa	Name	Name	T	Unikalna nazwa symboliczna zmiennej procesowej jednoznacznie identyfikująca tę zmienną dla wszystkich komponentów systemu Asix
Opis	obligatoryjny	Opis	Description	Description	T	Dowolny tekst np. zawierający technologiczny opis zmiennej procesowej
Kanał	obligatoryjny	Kanał	Channel	Channel	T	Nazwa logiczna kanału transmisji (jedna z nazw podanych w deklaracji kanałów transmisji)
Adres	obligatoryjny	Adres	Address	Address	T	Adres symboliczny, którego forma jest specyficzna dla każdego z drajwerów systemu Asix
LiczbaElementow	obligatoryjny	Liczba elementów	ElementsCount	Elements coun	L	Liczba elementów wchodzących w skład zmiennej procesowej (zmienna może być tablicą)
OkresProbkowania	obligatoryjny	Okres próbkowania	SampleRate	Sample rate	L	okres aktualizacji wartości zmiennej procesowej (w sekundach)
FunkcjaPrzeliczajaca	obligatoryjny	Funkcja przeliczająca	ConversionFunction	Conversion function	T	Nazwa funkcji przeliczającej służącej do przeliczenia wartości otrzymanej ze sterownika na wartość przekazywaną do komponentów systemu Asix
Archiwum	obligatoryjny	Archiwum	Archive	Archive	T	Nazwa zasobu ASPAD (logiczna nazwa zasobu archiwum)
ParametryArchiwizacji	obligatoryjny	Parametry archiwizacji	ArchivingParameters	Archiving parameters	T	Parametry archiwizacji danej zmiennej
Grupa1	opcjonalny	Grupa1	Group1	Group1	T	Domyślnie generowany atrybut grupujący
Format	opcjonalny	Format	Format	Format	T	Format liczby w jakim będzie ona wyświetlana w systemie asix; format podany jest w konwencji języka C (np.: %4.2f oznacza 4 znaki, w tym 2 miejsca po przecinku)
FormatEvo	opcjonalny	FormatEvo	FormatEvo	FormatEvo	T	Format liczby w jakim będzie ona wyświetlana w aplikacji Asix.Evo
Jednostka	opcjonalny	Jednostka	Unit	Unit	T	Fizyczna jednostka zmiennej
ZakresPomiarowyOd	opcjonalny	Zakres pomiarowy od	MeasurementRangeFrom	Measurement range from	L	Maksymalny zakres możliwy do uzyskania - minimum
ZakresPomiarowyHi	opcjonalny	Zakres pomiarowy do	MeasurementRangeTo	Measurement range to	L	Maksymalny zakres możliwy do uzyskania - maksimum
ZakresSurowyOd	opcjonalny	Zakres surowy od	RawRangeFrom	Raw range fror	L	Parametr funkcji przeliczającej – zakres wejściowy od …
ZakresSurowyDo	opcjonalny	Zakres surowy do	RawRangeTo	Raw range to	L	Parametr funkcji przeliczającej – zakres wejściowy do …
ZakresPrzetwornikaOd	opcjonalny	Zakres przetwornika o	TransducerRangeFrom	Transducer range from	L	Zakres przetwornika - minimum
ZakresPrzetwornikaDo	opcjonalny	Zakres przetwornika d	TransducerRangeTo	Transducer range to	L	Zakres przetwornika - maksimum
ZakresWyswietlaniaOd	opcjonalny	Zakres wyświetlania o	DisplayRangeFrom	Display range from	L	Dolny zakres wyświetlania danej zmiennej – dla obiektów asixa
ZakresWyswietlaniaDo	opcjonalny	Zakres wyświetlania d	DisplayRangeTo	Display range t	L	Górny zakres wyświetlania danej zmiennej – dla obiektów asixa
ZakresSterowaniaOd	opcjonalny	Zakres sterowania od	SteeringRangeFrom	Steering range from	L	Dolny zakres wartości danej zmiennej ustawiany z asixa
ZakresSterowaniaDo	opcjonalny	Zakres sterowania do	SteeringRangeTo	Steering range to	L	Górny zakres wartości danej zmiennej ustawiany z asixa
BazaSlupka	opcjonalny	Baza słupka	BarBase	Bar base	T	Wielkość wczytywana z bazy przez obiekt SŁUPEK
NazwyStanow	opcjonalny	Nazwy stanów	StateNames	State names	T	Nazwa zbioru tekstów lub bezpośrednio zbiór opisów stanów (poszczególnych wartości całkowitych) zmiennej

ZestawStanow	opcjonalny	Zestaw stanów	StateSet	State set	T	Nazwa zbioru opisów stanów do którego należy bieżący wiersz. Atrybut używany tylko dla wierszy zawierających element zbioru opisów stanów
WartoscStanu	opcjonalny	Wartość stanu	StateValue	State value	T	Numer stanu, którego opis zawiera bieżący wiersz. Atrybut używany tylko dla wierszy zawierających element zbioru opisów stanów
MaskaZmiennejKontrolnej	opcjonalny	Maska zmiennej kontrolnej	ControlVariableMask	Control Variabl mask	T	Wskazuje na bit statusu sygnalizowanego przez zmienną kontrolną
ZmiennaKontrolna	opcjonalny	Zmienna kontrolna	ControlVariable	Control variabl	T	Nazwa zmiennej reprezentującej status zmiennej pomiarowej
LimitLoLo	opcjonalny	Minimum krytyczne	LimitLoLo	Limit LoLo	T	Nazwa zmiennej systemu Asix, która zawiera wartość minimum krytycznego dla zmiennej pomiarowej
LimitLo	opcjonalny	Minimum	LimitLo	Limit Lo	T	Nazwa zmiennej systemu Asix, która zawiera wartość minimum dla zmiennej pomiarowej
LimitHi	opcjonalny	Maksimum	LimitHi	Limit Hi	T	Nazwa zmiennej systemu Asix, która zawiera wartość maksimum dla zmiennej pomiarowej
LimitHiHi	opcjonalny	Maksimum krytyczne	LimitHiHi	Limit HiHi	T	Nazwa zmiennej systemu Asix, która zawiera wartość maksimum krytycznego dla zmiennej pomiarowej
ZakresWyswietlaniaKrok	opcjonalny	Zakres wyświetlania krok	DisplayRangeStep	Display range step	L	Atrybut wykorzystywany przez obiekt WYKRES. Oznacza szerokość podziałki głównej na osi OY (co ile jednostek będzie wyświetlana etykieta wartości na osi OY)
ZakresWyswietlaniaSzer	opcjonalny	Zakres wyświetlania szerokość	DisplayRangeWidth	Display range width	L	Atrybut wykorzystywany przez obiekt WYKRES. Oznacza szerokość etykiety wartości wyświetlanych na osi OY
ZakresWyswietlaniaPodzial	opcjonalny	Zakres wyświetlania podział	DisplayRangeDivision	Display range division	L	Atrybut wykorzystywany przez obiekt WYKRES. Wyznacza podział na mniejsze jednostki podziałki głównej (wyznaczonej przez atrybut *Zakres wyświetlania krok*) na osi OY
Dokładnosc	opcjonalny	Dokładność	Precision	Precision	T	Dokładność zaokrąglania w obiekcie LICZBA. Przykładowo podanie w tym polu liczby 10 spowoduje, że liczba będzie zaokrąglana do dziesiątek. Domyślnie nie ma zaokrąglania.
PolozenieSuwaka	opcjonalny	Położenie suwaka	SliderPosition	Slider position	T	Nazwa zmiennej, do której wpisywana będzie wartości reprezentująca położenie uchwytu obiektu SUWAK w trakcie przesuwania tegoż uchwytu
RejestracjaSterowania	opcjonalny	Rejestracja sterowania	ControlLogging	Control logging	Wartość logiczna	Wartość 1 oznacza, że sterowania zmiennej podlegają rejestracji w module AsAudit
ZmiennaNieaktywna	opcjonalny	Zmienna nieaktywna	ItemNotActive	Item not active	Wartość logiczna	Jeżeli wartość atrybutu wynosi 1, to zmienna nie jest widoczna w systemie Asix dla modułów Asmen i Aspad
WykresTytul	opcjonalny	Tytuł wykresu	ChartTitle	Chart title	T	Atrybut wykorzystywany przez obiekt WYKRES. Tytuł wykresu pobierany jest z atrybutu *Tytuł wykresu* pierwszej zmiennej zadeklarowanej na wykresie
ZakresSlupkaOd	opcjonalny	Zakres słupka od	BarRangeFrom	Bar range from	T	Atrybut wykorzystywany przez obiekt SŁUPEK. Zawiera dolny zakres wyświetlania danej zmiennej.
ZakresSlupkaDo	opcjonalny	Zakres słupka do	BarRangeTo	Bar range to	T	Atrybut wykorzystywany przez obiekt SŁUPEK. Zawiera górny zakres wyświetlania danej zmiennej.
SzybkieOdswiezanie	opcjonalny	Szybkie odświeżanie	FastRefresh	Fast refresh	Wartość logiczna	Atrybut wykorzystywany przez drajwer CtTwinCATTcpip od

						wersji 1.3.5. Zmienne z zadeklarowanym atrybutem *Fast Refresh* mają udostępnioną funkcję szybkiego odświeżania, która umożliwia odświeżanie zmiennych z maksymalną osiągalną szybkością łącza (nie wpływa to na archiwizację - zmienne dalej archiwizowane są z częstotliwością wynoszącą minimum 1 sekundę)
SprawdzanieZakresuPomiarowego	opcjonalny	Sprawdzanie zakresu pomiarowego	MeasurementRangeCheck	Measurement range check	Wartość logiczna	Wartość 1 oznacza, że dla podanej zmiennej monitorowane będą przekroczenia zakresu pomiarowego i sygnalizowane odpowiednim statusem OPC. Status zmiennej jest określany po porównaniu wartości pomiarowej z wartościami zakresów pomiarowych: a/ przy przekroczeniu zakresu o wartość <= od pasma zmienności wartości zmiennej (zakresTo - zakresLo) ustawiany jest status OPC_QUALITY_EGU_EXCEEDED b/ przy przekroczeniu zakresu o wartość > od pasma zmienności wartości zmiennej (zakresTo - zakresLo) ustawiany jest status OPC_QUALITY_SENSOR_FAILURE
GrupaUprawnienOdczytu	opcjonalny	Grupa uprawnień odczytu	ReadAuthorizationGroup	Read authorization group	T	Zadeklarowanie atrybutu dla bazy definicji zmiennych, z których korzystają aplikacje: AsPortal i AsTrend, powoduje zablokowanie możliwości odczytu wartości do momentu sprawdzenia uprawnień użytkownika (aby umożliwić dostęp do przeglądania wartości zmiennych, należy dla każdej zmiennej podać ,jako wartość atrybutu *Grupa uprawnień odczytu*, nazwę grupy użytkowników systemu Windows uprawnionych do przeglądania wartości zmiennej - działa przy zastosowaniu uwierzytelniania użytkowników przez system Windows).
GrupaUprawnienZapisu	opcjonalny	Grupa uprawnień zapisu	WriteAuthorizationGroup	Write authorization group	T	Zadeklarowanie atrybutu dla bazy definicji zmiennych, z których korzysta aplikacja AsPortal, powoduje zablokowanie możliwości odczytu wartości do momentu sprawdzenia uprawnień użytkownika (aby umożliwić dostęp do modyfikowania wartości zmiennych, należy dla każdej zmiennej podać ,jako wartość atrybutu *Grupa uprawnień zapisu*, nazwę grupy użytkowników systemu Windows uprawnionych do modyfikowania wartości zmiennej - działa przy zastosowaniu uwierzytelniania użytkowników przez system Windows).

T - tekst, L - liczba zmiennoprzecinkowa 64 bitowa