AsixConnect4

Podręcznik użytkownika

Dok. Nr PLP4072
Wersja: 11-05-2006

1. Wstęp

Pakiet AsixConnect jest pakietem serwerów rozszerzającym możliwości zastosowań pakietu **asix** w dziedzinie wizualizacji i nadzoru procesów przemysłowych.

Komputer posiadający fizyczne połączenie ze sterownikiem (za pośrednictwem łącza szeregowego, magistrali przemysłowej lub sieci lokalnej) stanowi źródło informacji o zmiennych procesowych dla innych komputerów w sieci lokalnej; komputer taki jest serwerem danych bieżących i archiwalnych. W dostępie do tych danych pośredniczy pakiet AsixConnect.

Pakiet AsixConnect zawiera serwery *OPC*, *Automation*, *.NET* i *DDE* udostępniające wartości bieżące zmiennych procesowych z aplikacji pakietu **asix**, serwery *Automation*, *.NET* i *OLE DB* udostępniające wartości archiwalne zmiennych procesowych oraz serwer *.NET* udostępniający dane o alarmach. Ponadto pakiet zawiera serwer *Web Service* udostępniający wszystkie rodzaje danych aplikacji systemu **asix** w standardzie *XML Web Services.*

Każdy program środowiska Windows wyposażony w obsługę mechanizmu *Automation*, *OPC*, *.NET* lub *DDE* może współdziałać z aplikacją programu **asix** za pośrednictwem serwerów pakietu AsixConnect. Taki program może być zarówno konsumentem danych z procesu przemysłowego jak i źródłem danych dla celów sterowania nadrzędnego lub parametryzacji. Innymi słowy tą drogą w środowisku Windows dostępne są on-line wartości bieżące zmiennych procesowych jak i ich wartości archiwalne, czyli zarejestrowane przebiegi czasowe. Przykładami produktów wyposażonych w mechanizmy wymiany danych Automation i DDE są składowe pakietu Microsoft Office – Excel i Access. Aplikacje stworzone przy użyciu tych produktów i pakietu AsixConnect mogą efektywnie wzbogacać komputerowe systemy nadzoru. Aplikacje te mogą służyć do analizy i prezentacji danych, badań modelowych, specjalistycznego raportowania czy tworzenia baz danych procesowych.

AsixConnect jest integralnym elementem pakietu **asix**, ale jest także dostępny jako samodzielny produkt. Produkt ten może być stosowany na stacjach PC dołączonych do lokalnych sieci komputerowych i mających dostęp do serwerów danych wyposażonych w pakiety **asix**. W tym przypadku AsixConnect udostępnia w środowisku Windows dane importowane z oddalonych stanowisk komputerowych wyposażonych w łącza ze sterownikami procesów.

AsixConnect jest istotnym elementem strategii otwartości pakietu **asix** w jego zastosowaniach w środowisku systemów Windows 2003, XP, 2000 i NT4.

W kolejnych rozdziałach niniejszego podręcznika opisano instalację pakietu AsixConnect oraz sposób dostępu do zmiennych procesowych. Od czytelników podręcznika wymaga się znajomości podstaw mechanizmów *Automation*, *OLE DB*, *OPC*, *DDE, NET* i *XML Web Services*.

1.1. Elementy pakietu

Pakiet AsixConnect jest rozprowadzany jako część składowa systemu **asix** lub jako samodzielny pakiet. Pakiet AsixConnect zawiera następujące serwery:

- Serwery danych bieżących: *.NET, Automation*, *OPC*, *DDE* ,
- Serwery danych archiwalnych: *.NET, Automation*, *OLE DB*,
- Serwery baza zmiennych: *.NET, Automation*,

- Serwer alarmów: *.NET*.
- Serwer *Web Service*,
- Serwis DDE (instalowany opcjonalnie).

Do uruchomienia serwerów pakietu AsixConnect wymagany jest klucz *HASP asix* lub *HASP AsixConnect*. W obu przypadkach musi być w kluczu włączona flaga *Wersja 4*.

1.2. Licencjonowanie

Licencjonowanie pakietu AsixConnect omówione jest w dokumentacji handlowej.

1.3. Wymagania odnośnie systemu asix

Przy współpracy serwerów pakietu AsixConnect z systemem **asix** pracującym na tym samym komputerze, obsługiwane są wszystkie rodzaje licencji systemu **asix**. Przy współpracy z systemem **asix** pracującym na innym komputerze obsługiwane są wszystkie wersje systemu **asix** (w tym również starsze wersje pracujące pod systemem operacyjnym DOS), ale wymaganym rodzajem licencji systemu **asix** jest *serwer operatorski* (symbol *WAxS*).

1.4. Najważniejsze zmiany w pakiecie

Wersja 4.0

- Rozszerzenie pakietu o serwery *.NET* danych bieżących, archiwalnych, alarmów i bazy zmiennych.
- Rozszerzenie pakietu o serwer *Web Service*.
- Umożliwienie definiowania kanałów, czyli niezależnych zbiorów parametrów połączeń z serwerami systemu **asix**. Rozszerzenie pakietu o zewnętrzny program do konfiguracji parametrów kanałów.
- Zmiana nazw systemowych serwerów Automation, zachowanie kompatybilność działania z wersją 3.

2. Instalacja

2.1. Instalacja autonomiczna pakietu AsixConnect

Zestaw instalacyjny programu AsixConnect składa się z następujących elementów:

- Podręcznik użytkownika,
- Klucz ochrony HASP,
- Dysk CD zawierający oprogramowanie.

Z dysku CD należy wystartować program o nazwie *AsixConnect_4.EXE*. i postępować zgodnie z pojawiającymi się wskazówkami. Po zainstalowaniu programu w katalogu docelowym (domyślnie *c:\asix*) znajdą się następujące elementy:

- pliki serwerów wchodzących w skład pakietu AsixConnect;
- podkatalog *AC4Przyklady* zawierający przykłady wykorzystujące pakiet AsixConnect; przykłady zapisane są w arkuszach kalkulacyjnych w formacie programu Excel 2000;
- podkatalog *Aplikacje\Wytwornia_Kwasu* zawierający aplikację systemu **asix** wraz z bazą zmiennych; z danych tej aplikacji korzystają przykłady użycia pakietu AsixConnect;
- podkatalog *WebService* zawierający pliki serwera *WebService*.

Program instalacyjny pakietu AsixConnect nie obsługuje instalacji na komputerze, na którym jest zainstalowany pakiet **asix** w wersji 3. Instalacja pozwalająca na jednoczesną pracę pakietów AsixConnect 4 i **asix** 3 jest możliwa do wykonania wyłącznie przez pracownika firmy Askom.

2.2. Instalacja w ramach pakietu asix

Użytkownik w ramach pakietu **asix** otrzymuje pakiet AsixConnect i może używać oba te pakiety jednocześnie na tym samym komputerze.

Aby zainstalować pakiet AsixConnect razem z pakietem **asix,** należy z dysku CD pakietu **asix** wystartować program o nazwie AS*IX_PL_4.EXE* i postępować wg wyświetlanych wskazówek. Katalogiem domyślnym instalacji jest c*:\asix*.

3. Konfiguracja połączeń

3.1. Kanały

Kanałem nazywamy zbiór opcji konfiguracyjnych serwerów pakietu AsixConnect, opcji połączeń z serwerem systemu **asix** i opcji bazy zmiennych.

Kanał o nazwie '*' jest kanałem podstawowym serwerów pakietu AsixConnect. Kanał ten jest tworzony podczas instalacji pakietu i nie można go usunąć. Opcje z tego kanału są używane wtedy, gdy jako nazwa kanału zostanie użyty:

- * ;
- tekst pusty;
- tekst rozpoczynający się od *ANY.*

Jeżeli natomiast zostanie użyta nazwa kanału zdefiniowanego przy pomocy programu *Konfigurator* - to w takim przypadku używane są opcje z tego kanału.

Jeśli zostanie zastosowana nazwa kanału rozpoczynająca się od *NEW,* to do serwerów nie zostaną przesłane żadne opcje konfiguracyjne i klient powinien przesłać własny zestaw opcji za pomocą odpowiednich dla danego serwera mechanizmów.

Wykorzystanie niezdefiniowanej nazwy kanału innej niż wyszczególniono powyżej jest błędem.

3.2. Sposób specyfikowania nazwy kanału

Sposób specyfikowania nazwy kanału w poszczególnych serwerach pakietu AsixConnect prezentuje poniższa tabela.

Typ Serwer	**Sposób specyfikowania kanału**
Serwery *Automation*	Serwery udostępniają funkcję *LoadChannel*. Nazwa kanału jest parametrem funkcji.
Serwer DDE	Nazwą kanału jest *temat* (ang. *topic*) połączenia DDE.
Serwer OPC	Nazwą kanału jest ścieżka dostępu zmiennej (ang. *access path*).
Serwer OLE DB	Nazwę kanału należy przekazać w parametrze *Data Source (Źródło danych)*.
Serwery *.NET*	Nazwa kanału jest parametrem konstruktorów obiektów *.NET*.
Serwery *.NET* tworzone w środowisku ASP.NET przy użyciu funkcji *SessionServer*	Nazwa kanału pobierana jest z pliku *Web.Config* – patrz niżej.
Serwer *Web Service*	Nazwa kanału pobierana jest z pliku *Web.Config.*

Aplikacje ASP.NET używają pliku konfiguracyjnego o nazwie *Web.Config* do przechowywania domyślnej nazwy kanału. Plik ten znajduje się w katalogu aplikacji. Aby określić domyślną nazwę kanału, należy w elemencie nadrzędnym *configuration* utworzyć element *appSettings*; następnie w elemencie *appSettings* utworzyć jeden element *add* i

zdefiniować w nim dwa atrybuty. Pierwszy atrybut należy nazwać *key* i nadać mu wartość *"DefaultChannelName"*. Drugi atrybut powinien otrzymać nazwę *value* i jako wartość - nazwę kanału. Nazwę kanału ujmuje się w cudzysłowy.

PRZYKŁAD:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="DefaultChannelName" value="AsEmis" />
  </appSettings>
…
```

3.3. Plik konfiguracyjny

Informacje o zdefiniowanych kanałach są przechowywane w pliku *ASIXConnect.ini,* w katalogu, w którym jest zainstalowany pakiet AsixConnect.

Należy pamiętać, aby w razie instalacji nowej wersji pakietu **asix** lub AsixConnect, zrobić kopię tego pliku, gdyż w trakcie instalacji plik ten zostanie nadpisany.

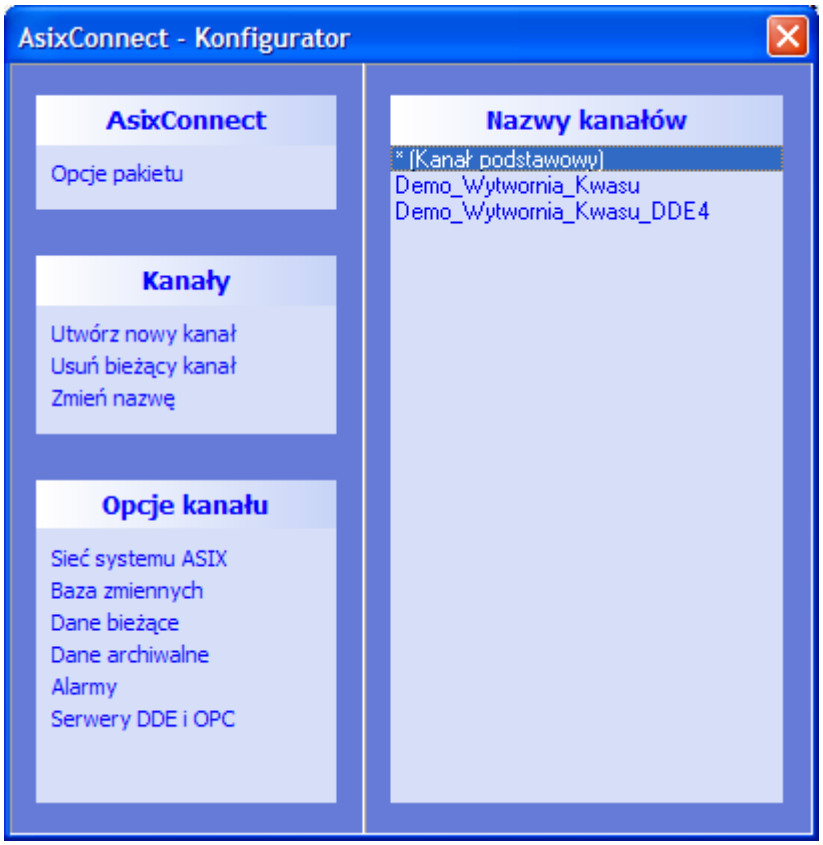
3.4. Konfiguracja interaktywna

3.4.1. Program Konfigurator

Do konfigurowania opcji kanału służy program *Konfigurator*. Program ten po uruchomieniu wyświetla okno prezentowane na poniższym rysunku.

W lewej części okna wyświetlana jest lista dostępnych operacji. W prawej części wyświetlana jest lista zdefiniowanych kanałów. Kanał o nazwie '*' istnieje zawsze.

3.4.2. Zarządzanie kanałami



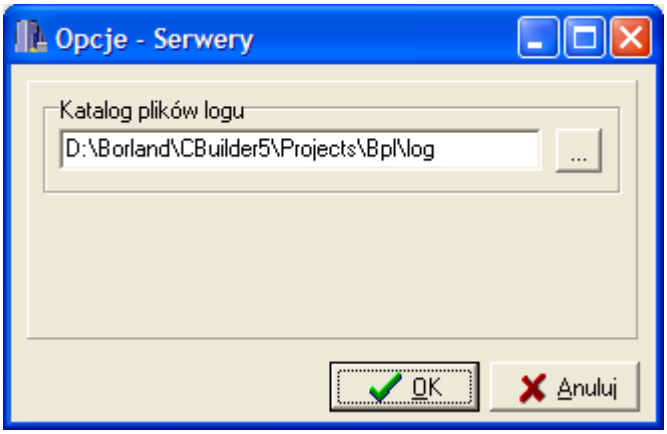
Polecenia odnoszące się do części Kanały mają następujące znaczenie:

Utwórz nowy kanał - utworzenie nowego kanału;
Usuń bieżący kanał - usunięcie aktualnie podświetlonego kanału;
Zmień nazwę - zmiana nazwy aktualnie podświetlonego kanału.

Nie można usunąć ani zmienić nazwy kanału podstawowego.

3.4.3. Opcje pakietu

Opcje z grupy *Opcje pakietu* dotyczą wszystkich serwerów pakietu AsixConnect.



Opcja *Katalog plików logu* określa, gdzie umieszczane są pliki logów serwerów. Domyślnie opcja nie jest zdefiniowana i pliki logów umieszczane są w katalogu, w którym pakiet AsixConnect jest zainstalowany.

Opcje z tej grupy mogą być modyfikowane tylko interaktywnie.

3.5. Konfiguracja programowa

Większość serwerów pakietu AsixConnect ma możliwość programowego ustawiania opcji. Ustawienie to polega na wywołaniu odpowiedniej funkcji i przekazaniu jako jej parametr tekstu o postaci:

```
Opcja1=Wartość1;Opcja2=Wartość2; ...
```

Liczba składników *Opcja=Wartość* jest dowolna. Składniki są od siebie oddzielone średnikami.

Sposoby ustalania opcji w poszczególnych serwerach pakietu AsixConnect ujęto w poniższej tabeli.

Typ serwer	**Sposób ustalania opcji**
Serwery *Automation*	Wywołanie funkcji *Init*.
Serwery *.NET*	
Serwer DDE	Wywołanie transakcji *XTYP_POKE* (zapis do serwera DDE). Jako parametr *item* należy podać nazwę *InitString*. Jako parametr *data* należy podać tekst inicjujący. Programowe modyfikowanie opcji w ramach danego połączenia ma wpływ tylko na opcje używane w tym połączeniu.
Serwer OPC	Serwery mogą korzystać tylko z opcji domyślnych kanału.
Serwer OLE DB	
Serwer *Web Service*	

Wymaganymi wartościami opcji może być tekst, liczba lub wartość logiczna. Dla opcji typu logicznego jako wartość logiczną *Prawda* można podać *tak*, *true* lub *1,* a jako wartość logiczną *Fałsz* można podać *nie*, *false* lub *0*.

Kolejność opcji w parametrze nie ma znaczenia dla kolejności ich interpretacji. Jeżeli napotkana zostanie opcja, dla której podano nieprawidłową wartość, to wykonywanie inicjalizacji jest przerywane i zwracana jest informacja o błędzie. Opcje nieznane są ignorowane.

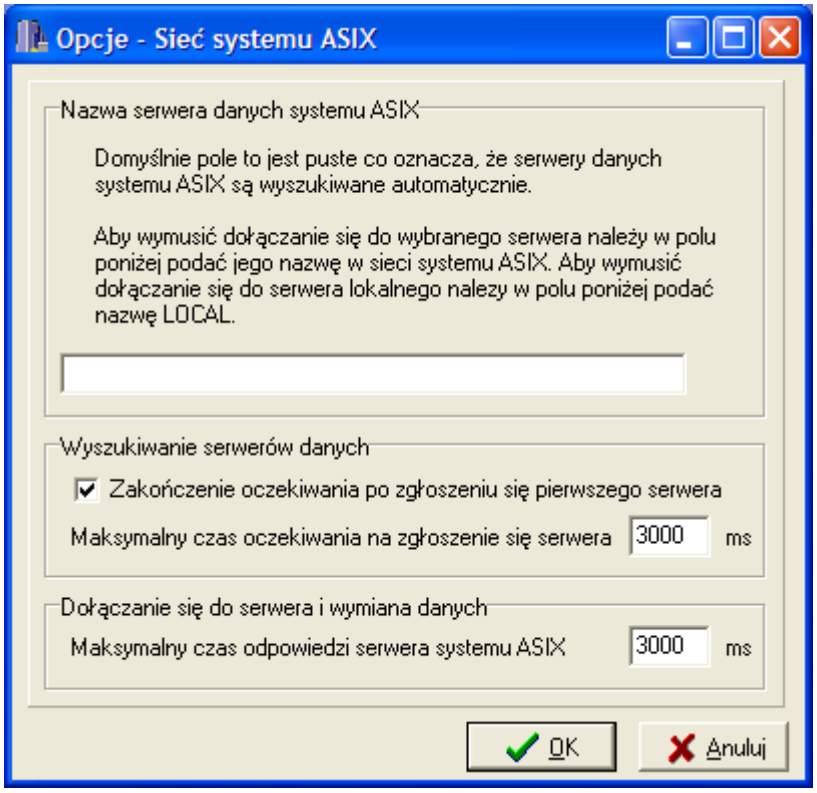
Opcje można modyfikować programowo do chwili wykonania pierwszej operacji na danych tj. odczytu, zapisu lub aktywowania zmiennej.

3.6. Opcje kanałów

3.6.1. Sieć systemu asix

3.6.1.1. Sieć systemu asix

Konfiguracja interaktywna



Opcja *Nazwa serwera danych systemu asix* umożliwia podanie nazwy lub nazw serwerów systemu **asix**. Dane będą pobierane tylko z serwerów umieszczonych na liście. Nazwy serwerów należy oddzielić od siebie przecinkami. Domyślnie opcja nie ma ustalonej wartości, co oznacza, że spośród znalezionych serwerów danych może być wybrany dowolny z nich. Aby wymusić dołączanie się tylko do serwera lokalnego, należy jako nazwę serwera podać *LOCAL*.

System **asix**, z którego mają być pobierane dane przez sieć lokalną, musi być systemem typu *serwer operatorski* (o symbolu *WAxS*).

Nazwę komputera klienta określa się w poniższy sposób.

Opis konfiguracji oprogramowania klienta	**Nazwa komputera klienta w systemie asix**
Używany jest tylko pakiet AsixConnect i nie utworzono pliku *aslink.ini* lub w sekcji *ASLINK*, nie zdefiniowano linii *Nazwa*.	Nazwa komputera klienta w systemie WINDOWS plus kropka dodana na końcu. Np. dla komputera o nazwie „CLI", nazwą komputera w systemie asix będzie „CLI.".
Używany jest tylko pakiet AsixConnect i utworzono pliku *aslink.ini*, w sekcji *ASLINK* zdefiniowano linię *Nazwa*.	Nazwa komputera klienta w systemie asix nadawana jest w pliku *aslink.ini*, w sekcji *ASLINK*, w linii zatytułowanej *Nazwa*.
Używane są na tym samym komputerze i pakiet AsixConnect i aplikacja systemu asix.	Nazwa komputera klienta w systemie asix nadawana jest w pliku *ini* aplikacji, w sekcji *ASLINK*, w linii zatytułowanej *Nazwa*.

Pozostałe opcje należą do zaawansowanych i zwykle nie ma potrzeby ich modyfikacji. Ich znaczenie opisano w punkcie *Wyszukiwanie serwerów danych systemu **asix** (patrz: punkt 3.6.1.2.)*.

Konfiguracja programowa

Opcja	**Nazwa opcji przy konfiguracji interaktywnej**	**Typ**	**Wartość domyślna**
AsixServerName	*Nazwa serwera danych systemu asix.*	Tekstowy	Brak (tekst pusty)
StopAfterFirstServerFound	*Zakończenie oczekiwania po zgłoszeniu się pierwszego serwera.*	Logiczny	Tak
FindServerTimeout	*Maksymalny czas oczekiwania na zgłoszenie się serwera.*	Liczbowy	3000 [ms]
NetTimeout	*Maksymalny czas oczekiwania na odpowiedź serwera systemu asix*	Liczbowy	3000 [ms]

3.6.1.2. Wyszukiwanie serwerów danych systemu asix

Wszystkie serwery danych bieżących, archiwalnych i alarmów pakietu AsixConnect stosują ten sam algorytm wyszukiwania serwerów danych aplikacji systemu **asix**.

W pierwszym kroku serwery pakietu AsixConnect wysyłają polecenie wzywające wszystkie serwery systemu **asix** dołączone do lokalnej sieci komputerowej i udostępniające dany zasób (kanał, archiwum lub serwer alarmów) do przedstawienia się. Zapytanie dotyczy również lokalnego serwera systemu **asix**. Zapytanie dotyczy tylko lokalnego serwera systemu **asix**, jeżeli komputer, na którym zainstalowano pakiet AsixConnect, nie jest podłączony do sieci komputerowej. Na czas oczekiwania na odpowiedź mają wpływ opcje *StopAfterFirstServerFound* i *FindServerTimeout.*

W drugim kroku wybierany jest jeden serwer spośród serwerów systemu **asix**, które odpowiedziały na zapytanie. Na wybór serwera ma wpływ opcja *AsixServerName*.

Szczegóły wariantów algorytmu wyszukiwania podane są w tabeli poniżej.

Lp	Wartość opcji *AsixServerName*	Algorytm wyszukiwania serwera
1.	brak (tekst pusty) *	Jeżeli użyto zewnętrzną listę serwerów to realizowany będzie wariant 3 algorytmu wyszukiwania serwerów.
		StopAfterFirstServerFound = tak Wybrany zostanie pierwszy serwer, który się zgłosi. Jeżeli połączenie z serwerem zostanie zerwane to, nowe połączenie może być nawiązane z innym serwerem udostępniającym te same dane procesowe.
		StopAfterFirstServerFound = nie Po upływie czasu równego *FindServerTimeout* z serwerów, które się zgłosiły, wybrany zostanie jeden serwer zgodnie z poniższymi preferencjami. Preferencje wymienione są od najważniejszej: 1. Czy źródło danych serwera nie jest uszkodzone? 2. Czy serwer jest serwerem lokalnym? 3. Czy serwer udostępnia własne dane (nie jest pomostem)? 4. Czy ma najmniejszą liczbę klientów? Jeżeli połączenie z serwerem zostanie zerwane, to nowe połączenie może być nawiązane z innym serwerem udostępniającym te same dane procesowe.
2.	*nazwa_serwer*	Przez czas równy maksymalnie *FindServerTimeou* serwer pakietu AsixConnect będzie czekał na zgłoszenie się pierwszego serwera o podanej nazwie. Jeżeli połączenie z serwerem zostanie zerwane to nowe połączenie może być nawiązane tylko z tym samym serwerem.
3.	*nazwa_serwera1, nazwa_serwera2...* (lista nazw serwerów oddzielonych przecinkami)	Przez czas równy maksymalnie *FindServerTimeou* serwer pakietu AsixConnect będzie czekał na zgłoszenie się serwera o nazwie z podanej listy nazw. Jeżeli połączenie z serwerem zostanie zerwane to nowe połączenie może być nawiązane tylko z serwerem o nazwie z podanej listy nazw.
4.	**LOCAL**	Przez czas równy maksymalnie *FindServerTimeou* serwer pakietu AsixConnect będzie czekał na zgłoszenie się serwera lokalnego. Jeżeli połączenie z serwerem zostanie zerwane to nowe połączenie może być nawiązane tylko z serwerem lokalnym.

Wartości domyślne opcji używanych podczas wyszukiwania serwerów danych systemu **asix** ujęto w kolejno prezentowanej tabeli.

Nazwa opcji	**Wartość domyślna**
StopAfterFirstServerFound	Tak
FindServerTimeout	3000 [ms] (3 sekundy)
AsixServerName	Brak (tekst pusty)

3.6.1.3. Zewnętrzna lista serwerów

Zewnętrzna lista serwerów umożliwia wyspecyfikowanie listy dopuszczalnych serwerów danych dla każdego zasobu aplikacji systemu **asix** niezależnie. Zewnętrzna lista serwerów używana jest tylko wtedy, gdy w konfiguracji kanału nie podano jawnie nazwy serwera, z którego mają być pobierane dane. Lista serwerów jest wspólna dla wszystkich kanałów.

Zewnętrzna lista serwerów znajduje się w pliku *ASIXConnect.ini,* w katalogu, w którym jest zainstalowany pakiet AsixConnect. Dla każdego rodzaju serwera należy zdefiniować sekcję; każda linia w sekcji powinna mieć następującą postać:

```
nazwa_zasobu = nazwa_serwera1, nazwa_serwera2...
```

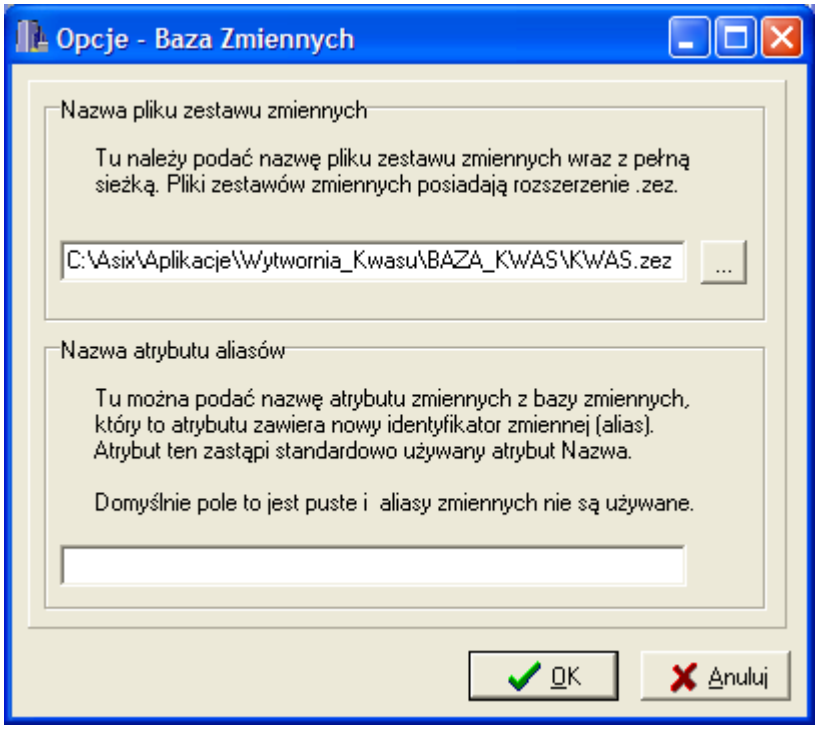
Nazwy sekcji pliku *ASIXConnect.ini* odpowiadające poszczególnym rodzajom serwerów oraz znaczenie elementu *nazwa_zasobu* podano w poniższej tabeli.

Typ serwera	**Nazwa sekcji**	**Znaczenie elementu *nazwa_zasobu***
Serwer danych bieżących	*AsixCTServers*	Nazwa kanału
Serwer danych archiwalnych	*AsixHTServers*	Nazwa archiwum
Serwer alarmów	*AsixALServers*	Nazwa sieciowa serwera alarmów

W liście nazw serwerów separatorem jest znak przecinka.

3.6.2. Baza zmiennych

Konfiguracja interaktywna



Opcja *Nazwa pliku zestawu zmiennych* służy do wprowadzenia informacji o położeniu bazy zmiennych, w której znajdują się definicje zmiennych aplikacji systemu **asix**.

Opcja *Nazwa atrybutu aliasów* umożliwia używanie w klientach serwerów pakietu AsixConnect alternatywnych nazw zmiennych. Alternatywne nazwy zmiennych muszą znajdować się w bazie zmiennych jako wartości pewnego atrybutu zmiennej; nazwa tego atrybutu musi być podana jako wartość opcji. Jeżeli opcja zostanie użyta, to zmienne, które nie mają wypełnionego atrybutu aliasów, będą niedostępne.

Alternatywne nazwy zmiennych są również używane przez serwera OPC przy przeglądaniu jego bazy zmiennych za pomocą mechanizmów OPC. Należy jednak pamiętać, że przy przeglądaniu bazy zmiennych używana jest tylko baza zdefiniowana w kanale podstawowym '*'.

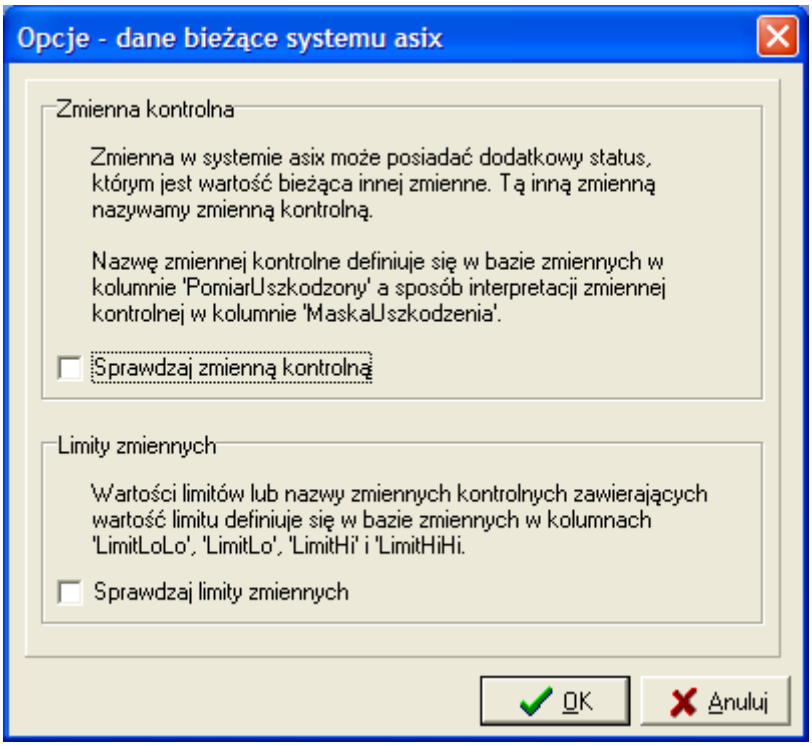
Alternatywne nazwy zmiennych nie są obsługiwane przez serwery .NET.

Konfiguracja programowa

Opcja	**Nazwa opcji przy konfiguracji interaktywnej**	**Typ**	**Wartość domyślna**
ItemsDatabase	*Nazwa pliku zestawu zmiennych*	Tekstowy	Brak
AliasAttributeName	*Nazwa atrybutu aliasów*	Tekstowy	Brak

3.6.3. Dane bieżące

Konfiguracja interaktywna



Opcje *Sprawdzaj zmienną kontrolną* i *Sprawdzaj limity zmiennych* umożliwiają włączenie sprawdzania zmiennych kontrolnych i sprawdzania limitów zmiennych analogicznie do sprawdzania wykonywanego przez obiekty *LICZBA* i *SŁUPEK* w systemie **asix**. Opcje te są domyślnie wyłączone. W przypadku, gdy kanał jest wykorzystywany przez dynamiczne strony HTML, opcje te muszą być włączone.

Aby włączyć sprawdzanie, należy zaznaczyć odpowiednie pole wyboru. Aby sprawdzanie mogło być przeprowadzone, w bazie zmiennych muszą znajdować się odpowiednie atrybuty, w których zdefiniowane są algorytmy sprawdzania. Chodzi o te same atrybuty, które są używane przy parametryzacji z bazy zmiennych obiektów LICZBA i SŁUPEK.

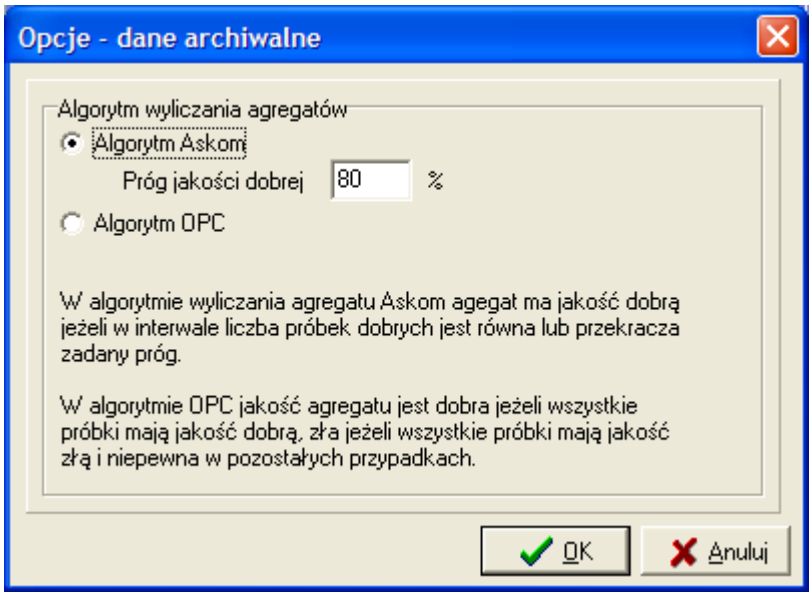
Informacje dotyczące atrybutów znajdują się w dokumentacji systemu **asix**.

Konfiguracja programowa

Opcja	**Znaczenie**	**Typ**	**Wartość domyślna**
CheckControlVariables	*Sprawdzaj zmienną kontrolną*	Logiczny	Nie
CheckLimits	*Sprawdzaj limity zmiennych*	Logiczny	Nie

3.6.4. Dane archiwalne

Konfiguracja interaktywna



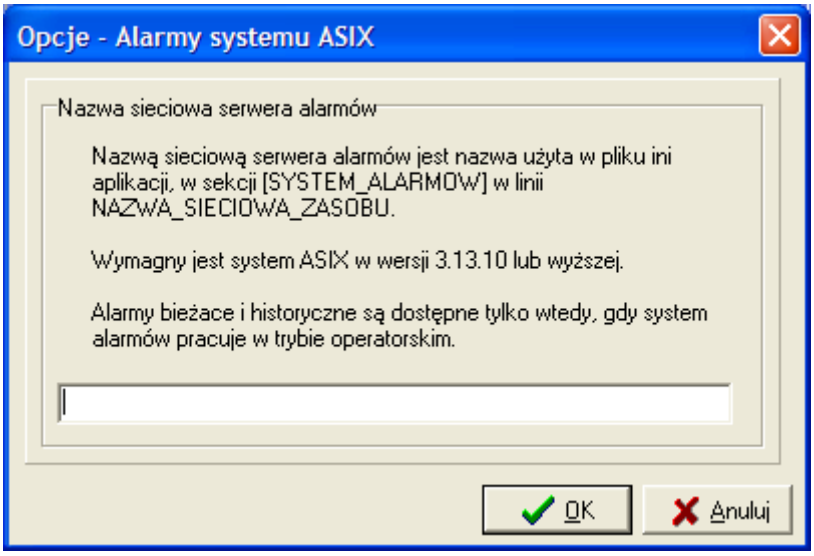
Opcja *Algorytm wyliczania agregatów* pozwala ustalić żądany rodzaj algorytmu. Algorytmy opisane są w punkcie Agregaty (*patrz: punkt 7.3.*).

Konfiguracja programowa

Opcja	**Znaczenie**	**Typ**	**Wartość domyślna**
AggregateMode	*Wybór algorytm wyliczania agregatów. Wartość 0 oznacza Algorytm Askom a wartość 1 oznacza Algortym OPC.*	Liczbowy	0
QualityGoodThreshold	*Próg jakości dobrej*	Liczbowy	80

3.6.5. Alarmy

Konfiguracja interaktywna



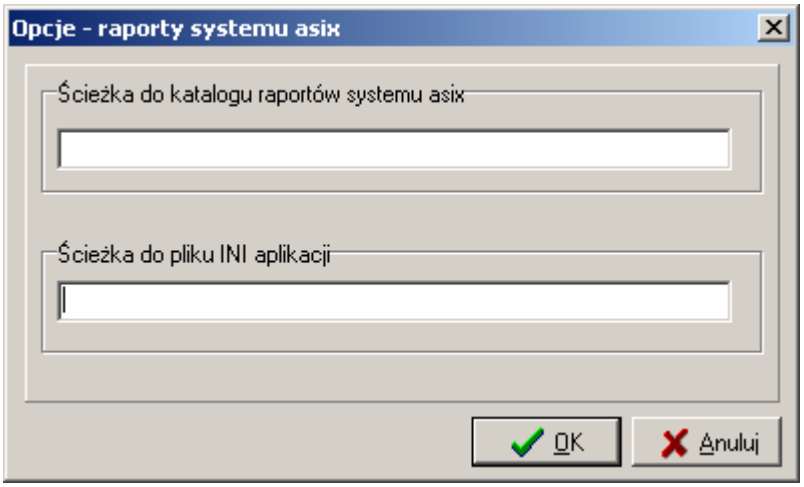
Podanie wartości opcji *Nazwa sieciowa serwera alarmów* jest konieczne, jeżeli klient będzie korzystał z serwera alarmów pakietu AsixConnect. Nazwa sieciowa serwera alarmów zdefiniowana jest w pliku *ini* aplikacji, w sekcji *SYSTEM_ALARMOW,* w linii *NAZWA_SIECIOWA_ZASOBU*. Alarmy są dostępne tylko wtedy, gdy w aplikacji jest zdefiniowana nazwa sieciowa i system alarmów pracuje w trybie operatorskim.

Konfiguracja programowa

Opcja	**Znaczenie**	**Typ**	**Wartość domyślna**
AlarmsSystemNetworkName	*Nazwa sieciowa serwera alarmów*	Tekstowy	Brak

3.6.6. Raporty

Konfiguracja interaktywna



Opcja *Ścieżka do katalogu raportów systemu asix* służy do wprowadzenia informacji o położeniu katalogu, w której znajdują się pliki definicji raportów oraz raporty aplikacji systemu **asix**.

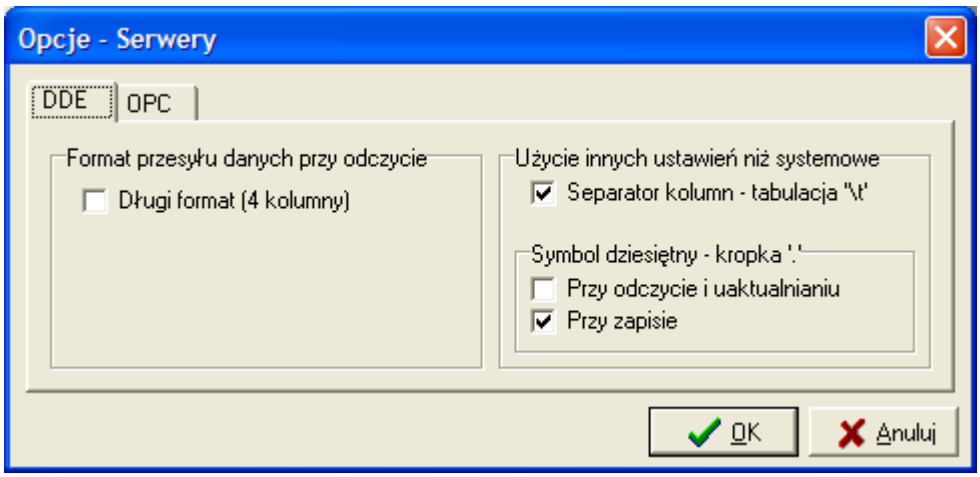
W polu *Ścieżka do pliki INI apllikacji* należy podać pełną ścieżkę do pliku ini aplikacji sytemu **asix**.

Konfiguracja programowa

Opcja	Znaczenie	Typ	Wartość domyślna
ReportsDirectory	*Ścieżka do katalogu raportów systemu asix*	*Tekstowy*	*Brak*
IniFilePath	*Ściezka do pliku INI aplikacji systemu asix*	*Tekstowy*	*Brak*

3.6.7. Serwery DDE i OPC

Konfiguracja interaktywna

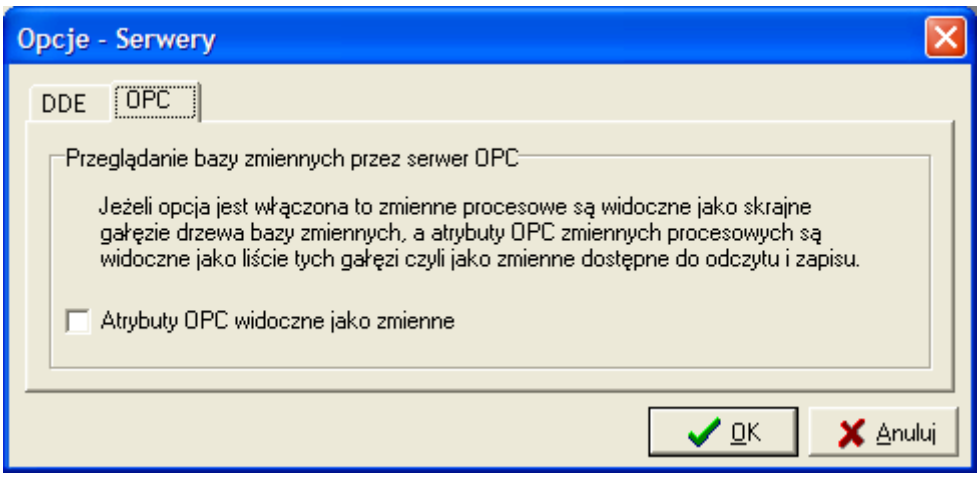


Domyślnie serwer DDE przesyła albo wartość zmiennej albo informację o błędzie w postaci tekstowego opisu błędu. Opcja *Długi format (4 kolumny)* pozwala włączyć przysyłanie przez serwer DDE szczegółowych informacji o zmiennych bieżących: osobno podawany jest status odczytu, wartość, jakość i stempel czasu.

Wyłącznie opcji *Separator kolumn – tabulacja '\t'* powoduje zmianę sposobu rozdzielania kolumn w tablicach danych. Zamiast znaku tabulacji stosowany jest separator listy z ustawień systemu operacyjnego. W języku polskim domyślnie jest to średnik ';'.

Opcje z grupy *Symbol dziesiętny – kropka '.'* sterują sposobem oddzielania części dziesiętnej w liczbach rzeczywistych przesyłanych z i do serwera DDE. Opcja *Przy odczycie i uaktualnianiu* jest domyślnie wyłączona, co oznacza, że przy odczycie danych bieżących i archiwalnych oraz uaktualnianiu danych bieżących stosowany jest symbol dziesiętny z ustawień systemu operacyjnego. W języku polskim jest to domyślnie przecinek ','. Opcja *Przy zapisie* jest domyślnie włączona, co oznacza, że przy zapisie danych bieżących do oddzielania części dziesiętnej stosowana jest kropka. Powodem takiego przyjęcia wartości

domyślnej jest to, że program Excel przy zapisie do serwera DDE zawsze stosuje jako symbol dziesiętny kropkę.



Opcja *Atrybuty OPC widoczne jako zmienne* zmienia sposób przeglądania bazy zmiennych przy pomocy mechanizmów OPC udostępnianych przez serwer OPC. Domyślnie, gdy opcja jest wyłączona, gałęziami drzewa reprezentującego bazę zmiennych są grupy zmiennych a liśćmi są poszczególne zmienne. Jeżeli opcja jest włączona, to wewnętrznymi gałęziami drzewa są grupy zmiennych, a zewnętrznymi gałęziami są poszczególne zmienne. Liśćmi są atrybuty zmiennych takie jak: wartość, opis, jednostka, zakres górny i zakres dolny.

Konfiguracja programowa

Opcja	**Znaczenie**	**Typ**	**Wartość domyślna**
DataFormatIsLong	*Długi format (4 kolumny)*	Logiczny	Nie
ColumnSeparatorIsTab	*Separator kolumn – tabulacja '\t'*	Logiczny	Tak
DecimalSymbolWhenReadIsDot	*Symbol dziesiętny – kropka '.' Przy odczycie i uaktualnianiu*	Logiczny	Nie
DecimalSymbolWhenWriteIsDot	*Symbol dziesiętny – kropka '.' Przy zapisie*	Logiczny	Tak

4. Baza zmiennych

4.1. Baza zmiennych

W systemie **asix** baza zmiennych jest miejscem przechowywanie wszelkich informacji o zmiennych procesowych. Na bazę zmiennych składają się baza atrybutów zmiennych, schemat bazy zmiennych i zestawy zmiennych.

Główna część bazy zmiennych, baza atrybutów zmiennych, zawiera informacje o samych zmiennych procesowych. Są to takie informacje jak nazwa, opis, jednostka czy limity pomiarowe. Sposób ułożenia tych informacji można sobie wyobrazić jako tabelę składającą się z wierszy i kolumn. Każdy wiersz tabeli zawiera wszystkie informacje o jednej zmiennej, każda kolumna tabeli zawiera wartość jednego atrybutu wszystkich zmiennych. Fizycznie baza atrybutów zmiennych jest bazą danych w formacie *Paradox* (dla serwerów .NET w formacie Microsoft Jet).

Struktura bazy atrybutów zmiennych, czyli spis wszystkich atrybutów, jakie może posiadać zmienna opisana jest w pliku schematu bazy zmiennych. Struktura bazy atrybutów jest rozszerzalna i może być dostosowywana do potrzeb danej aplikacji systemu **asix** podczas generacji bazy zmiennych. Plik ze schematem bazy ma nazwę *SCHEMAT.TXT*.

Programy wchodzące w skład systemu **asix** pobierają z bazy zmiennych atrybuty zmiennych potrzebne do wyświetlania masek technologicznych i sama baza atrybutów zmiennych jest dla nich całkowicie wystarczająca. Jednak dla operatora lub projektanta, który chce przeglądać bazę zmiennych, przeglądanie na raz całej bazy jest niewygodne. Potrzebny jest sposób na pokazywanie wycinków bazy zmiennych zawierających powiązane ze sobą zmienne. Dlatego w bazie zmiennych systemu **asix** wprowadzono możliwość podzielenia zmiennych na grupy, nazwania tych grup oraz ułożenie nazw tych grup w strukturę hierarchiczną. Struktura hierarchiczna jest typu drzewo i przedstawia strukturę procesu przemysłowego, strukturę działu firmy czy strukturę całego przedsiębiorstwa. Plik zawierający definicję ułożenia grup zmiennych nazywany jest zestawem zmiennych i stanowi on trzeci element bazy zmiennych. Jedna baza zmiennych może zawierać dowolną liczbę zestawów zmiennych. Pliki zawierające zestawy zmiennych mają rozszerzenie *.ZEZ*.

4.2. Serwer Automation

4.2.1. Serwer Automation

Serwer Automation bazy zmiennych umożliwia dostęp do bazy zmiennych aplikacji systemu **asix** przy użyciu mechanizmu Automation. Serwer Automation jest serwerem zaimplementowanym w postaci biblioteki dynamicznej DLL i uruchamianym wewnątrz przestrzeni adresowej klienta. Serwer ten zarejestrowany jest w systemie operacyjnym Windows jako obiekt o nazwie *XConnect.ServerVB*. Dokładny opis funkcji, własności i stałych udostępnianych przez obiekt znajduje się w dalszej części rozdziału. Serwer rejestruje w systemie operacyjnym również bibliotekę typów o nazwie *AsixConnect 4 Type Library*.

Serwer Automation danych bieżących jest serwerem w pełni zgodnym z mechanizmem Automation. Może być w pełni wykorzystany w językach programowania wyposażonych w obsługę mechanizmu Automation takich jak *Visual Basic*, *Visual Basic for Applications* (na przykład z pakietu *Microsoft Office*) czy *Visual Basic Script*.

Przy konwersji aplikacji Visual Basic korzystającej z pakietu AsixConnect w wersji 3 należ zamienić nazwę obiektu serwera *ServerBZ.App* na *XConnect.ServerVB* oraz zmienić nazwę używanej biblioteki typów na *AsixConnect 4 Type Library*.

4.2.2. Użycie serwera

Zamierzając wykonywać operacje na bazie zmiennych przy użyciu serwera Automation należy wykonać kolejno poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Napisać program operujący na serwerze *XConnect.ServerVB*. W programie tym należy:
  - utworzyć obiekt typu *XConnect.ServerVB*;
  - wywołać funkcję *LoadChannel* podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając procedur *SelectVar* i *ReadAttribute* zrealizować wybieranie zmiennych i czytanie atrybutów.

Wszystkie parametry funkcji serwera są typu VARIANT.

Zamiast korzystać z kanałów, można również załadować bazę zmiennych używając funkcji *Init*.

4.2.3. Funkcja LoadChannel

Składnia wywołania funkcji:

```
LoadChannel ChannelName
```

Funkcja służy do inicjalizacji obiektu *XConnect.ServerVB* przez załadowanie kanału. Jako parametr C*hannelName* należy podać nazwę kanału (*patrz: punkt 3.Konfiguracja*).

4.2.4. Funkcja Init

Składnia wywołania funkcji:

```
Init InitString
```

Funkcja służy do ustawiania parametrów serwera. Opis znajduje się w punkcie *Konfiguracja programowa (patrz: punkt 3.5.)* , a dostępne opcje w punkcie *Baza zmiennych (patrz: punkt 3.6.2.)*.

4.2.5. Funkcja ReadAttribute

Składnia wywołania funkcji:

```
ReadAttribute (VarNamej, AttributeName)
```

Funkcja *ReadAttribute* zwraca wartość atrybutu zmiennej.

Jako parametr *VarNamej* należy przekazać nazwę zmiennej.

Jako parametr *AttributeName* należy przekazać nazwę atrybutu lub jedną ze stałych zdefiniowanych przez serwer Automation (wylistowanych w poniższej tabeli).

Nazwa stałej	**Znaczenie**
atrDescription	Opis zmiennej
atrEU	Jednostka pomiarowa
atrRangeLo	Dolny zakres pomiarowy
atrRangeHi	Górny zakres pomiarowy
atrSampleRate	Okres próbkowania
atrArchiveRate	Okres archiwizacji

4.2.6. Funkcja SelectAttribute

Składnia wywołania funkcji:

```
SelectAttribute (SelectedAttributeName)
```

Funkcja *SelectAttribute* służy do interaktywnego wybierania przez użytkownika nazwy atrybutu. Jej wywołanie powoduje wyświetlenie okna dialogowego zawierającego listę nazw atrybutów zdefiniowanych w bazie zmiennych. Użytkownik może wybrać jedną z nich i zatwierdzić wybór naciskając przycisk *OK* lub zrezygnować z wyboru naciskając przycisk *Anuluj*.

Funkcja *SelectAttribute* zwraca wartość *true*, jeżeli użytkownik nacisnął przycisk *OK* i zatwierdził wybór. Wartość *false* jest zwracana, jeżeli użytkownik zrezygnował z wyboru naciskając przycisk *Anuluj*.

Parametr *SelectedAttributeName* jest parametrem wejściowym i wyjściowym. Jeżeli na wejściu parametr ten zawierał nazwę atrybutu, to po wyświetleniu okna wyboru atrybutu atrybut o tej nazwie jest podświetlony. Jeżeli funkcja *SelectAttribute* zwróciła wartość *true,* to parametr ten zawiera nazwę wybranego atrybutu.

4.2.7. Funkcja SelectVar

Składnia wywołania funkcji:

```
SelectVar (SelectedVarName)
```

Funkcja *SelectVar* służy do interaktywnego wybierania przez użytkownika nazwy zmiennej z bazy zmiennych. Jej wywołanie powoduje wyświetlenie okna dialogowego zawierającego listę nazw zmiennych zdefiniowanych w bazie zmiennych. Użytkownik może wybrać jedną z nich i zatwierdzić wybór naciskając przycisk *OK* lub zrezygnować z wyboru naciskając przycisk *Anuluj*.

Funkcja *SelectVar* zwraca wartość *true*, jeżeli użytkownik nacisnął przycisk *OK* i zatwierdził wybór. Wartość *false* jest zwracana, jeżeli użytkownik zrezygnował z wyboru naciskając przycisk *Anuluj*.

Parametr *SelectedVarName* jest parametrem wejściowym i wyjściowym. Jeżeli na wejściu parametr ten zawiera nazwę zmiennej, to po wyświetleniu okna wyboru zmiennej zmienna o tej nazwie jest podświetlona. Na wyjściu, jeżeli funkcja *SelectVarName* zwróciła wartość *true*, to parametr ten zawiera nazwę wybranej zmiennej.

4.2.8. Funkcja SelectVars

Składnia wywołania funkcji:

```
SelectVars (SelectedVarsNames)
```

Funkcja *SelectVars* służy do interaktywnego wybierania przez użytkownika nazw zmiennych z bazy zmiennych. Jej wywołanie powoduje wyświetlenie okna dialogowego zawierającego listę nazw zmiennych zdefiniowanych w bazie zmiennych. Użytkownik może wybrać jedną lub więcej z nich i zatwierdzić wybór naciskając przycisk *OK* lub zrezygnować z wyboru naciskając przycisk *Anuluj*.

Funkcja *SelectVars* zwraca wartość *true*, jeżeli użytkownik nacisnął przycisk *OK* i zatwierdził wybór. Wartość *false* jest zwracana, jeżeli użytkownik zrezygnował z wyboru naciskając przycisk *Anuluj*.

Parametr *SelectedVarNames* jest parametrem wyjściowym. Jeżeli funkcja *SelectVars* zwróciła wartość *true,* to parametr ten zawiera tablicę nazw wybranych zmiennych.

4.3. Serwer .NET

4.3.1. Klasa ServerVB

4.3.1.1. Użycie serwera

Klasa *ServerVB* umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie odczytu danych z bazy zmiennych. Zamierzając operować na bazie zmiennych przy użyciu serwera *ServerVB,* należy wykonać kolejno poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji systemu **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Wygenerować projekt w pakiecie Visual Studio i następnie:
  - podświetlić w drzewie projektu folder *References*, z menu *Project* wybrać polecenie *Add Reference*, nacisnąć przycisk *Browse* i z podkatalogu *c:\asix* wybrać plik *XConnectNet.dll*, nacisnąć przycisk *OK* i zamknąć okno *Add Reference*. W każdym pliku z kodem źródłowym C# należy w regionie deklaracji *using* dodać linię:

    ```
    using XConnectNet;
    ```

- Napisać program operujący na obiekcie klasy *ServerVB*. W programie tym należy:
  - utworzyć obiekt typu *ServerVB*, podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając funkcji składowych serwera *Read* za*programować czytanie danych;
  - podczas wymiany danych należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer;

- po zakończeniu używania obiektu *ServerVB* wywołać jego funkcję składową *Dispose*;
- podczas wymiany danych należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer.

4.3.1.2. Konstruktor ServerVB

```
[C#]
public ServerVB(
  string channelName);
```

Funkcja służy do utworzenia i inicjalizacji obiektu klasy *ServerVB*.

Jako parametr *channelName* należy podać nazwę kanału (*patrz: punkt 3.Konfiguracja*).

4.3.1.3. Funkcja Dipsose

```
[C#]
public void Dispose();
```

Funkcja służy do zwolnienia zasobów używanych przez obiekt *ServerVB*. Funkcja musi być wywołana po zakończeniu używania obiektu *ServerVB*. Wywołanie powinno nastąpić z tego samego wątku, w którym utworzono obiekt.

4.3.1.4. Funkcja Init

```
[C#]
public void Init(
    string initString);
```

Funkcja służy do ustawiania parametrów serwera. Opis funkcji znajduje się w punkcie *Konfiguracja programowa (patrz: punkt 3.5.)*, a dostępne opcje w punkcie *Baza zmiennych (patrz: punkt 3.6.2.)*.

4.3.1.5. Funkcja ReadAttributes

```
[C#]
public string[] ReadAttributes(
    string varName,
    string[] attributeNames);
```

Funkcja *ReadAttributes* służy do czytania wartości atrybutów zmiennej.

Jako parametr *varName* należy przekazać nazwę zmiennej.

Jako parametr *attributeNames* należy przekazać tablicę nazwę atrybutów.

W wyniku funkcja zwraca tablicę tekstów zawierających wartości atrybutów.

4.3.1.6. Funkcja ReadAttributesN

```
[C#]
public string[,] ReadAttributesN(
    string[] varNames,
    string[] attributeNames);
```

Funkcja *ReadAttributesN* służy do czytania wartości atrybutów kilku zmiennych na raz.

Jako parametr *varNames* należy przekazać tablicę nazw zmiennych.

Jako parametr *attributeNames* należy przekazać tablicę nazwę atrybutów.

Funkcja zwraca w wyniku dwuwymiarową tablicę tekstów zawierających wartości atrybutów. W pierwszym wierszu znajdują się atrybuty pierwszej zmiennej, w drugim atrybuty drugiej zmiennej itd.

4.3.1.7. Praca w środowisku ASP.NET

W przypadku pracy w środowisku ASP.NET do utworzenia obiektu należy użyć wyrażenie *ServerVB.ServerPool.Get()*. Obiekt *ServerPool* jest polem statycznym klasy *ServerVB* i implementuje pulę serwerów danych bieżących. Za pomocą funkcji *Get* należy każdorazowo na początku generowania strony pobrać obiekt *ServerVB*. Deklaracja funkcji *Get* jest następująca:

```
[C#]
public ServerVB Get ();
```

Po zakończeniu generowania strony serwer musi być zwrócony do puli przy pomocy wyrażenia *ServerVB.ServerPool.Release()*. Jako parametr funkcji *Release* należy przekazać zwracany do puli serwer. Deklaracja funkcji *Release* jest następująca:

```
[C#]
public Release (ServerVB server);
```

Serwer można pobierać z puli również na początku każdej funkcji i zwracać na końcu funkcji. Aby być pewnym, że każdy pobrany serwer wróci do puli, należy główny kod funkcji zawrzeć w bloku *try* i zwracać serwer do puli w bloku *finally*.

PRZYKŁAD:

```
private void Page_Load(object sender, System.EventArgs e)
{
    ServerVB server = null;

    try
    {
        server = ServerVB.ServerPool.Get();

        // kod funkcji
    }
    catch(Exception e)
    {
        // obługa wyjątków zgłoszonych przy pobierania serwera z puli i
        // podczas działania funkcji
    }
    finally
    {
        if (server != null)
            ServerVB.ServerPool.Release(server);
    }
}
```

Pula serwerów*:*

- tworzy dla aplikacji kilka obiektów klasy *ServerVB* (nazwa kanału pobierana jest z pliku *Web.Config - patrz: 3.2. Sposób specyfikowania nazwy kanału*);
- przechowuje obiekty w pamięci podręcznej aplikacji ASP.NET;
- udostępnia obiekty dla kolejnych wywołań w ramach aplikacji ASP.NET;
- zgłasza wyjątek *PoolApplicationExeption*, gdy pula serwerów osiągnęła maksymalny rozmiar i w puli nie ma żadnego wolnego serwer.

4.3.2. Klasa ServerVBUI

4.3.2.1. Użycie serwera

Klasa *ServerVBUI* umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie wyboru zmiennych z bazy zmiennych oraz wyboru atrybutów bazy zmiennych. Zamierzając operować na bazie zmiennych przy użyciu serwera *ServerVBUI,* należy wykonać kolejno wymienione kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji systemu **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Wygenerować projekt w pakiecie Visual Studio i następnie:
  - podświetlić w drzewie projektu folder *References*, z menu *Project* wybrać polecenie *Add Reference*, nacisnąć przycisk *Browse* i z podkatalogu *c:\asix* wybrać plik *XConnectNet.dll*, nacisnąć przycisk *OK*. i zamknąć okno *Add Reference*. W każdym pliku z kodem źródłowym C# należy w regionie deklaracji *using* dodać linię:

    ```
    using XConnectNet;
    ```

- Napisać program operujący na obiekcie klasy *ServerVBUI*. W programie tym należy:
  - utworzyć obiekt typu *ServerVBUI*, podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając funkcji składowych serwera *Select* za*programować wymianę danych;
  - podczas wymiany danych należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer;
  - po zakończeniu używania obiektu *ServerVBUI* wywołać jego funkcję składową *Dispose*.

4.3.2.2. Konstruktor ServerVBUI

```
[C#]
public ServerVBUI(
  string channelName);
```

Funkcja służy do utworzenia i inicjalizacji obiektu klasy *ServerVBUI*.

Jako parametr *channelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja*).

4.3.2.3. Funkcja Dipsose

```
[C#]
public void Dispose();
```

Funkcja służy do zwolnienia zasobów używanych przez obiekt *ServerVBUI*. Funkcja musi być wywołana po zakończeniu używania obiektu *ServerVBUI*. Wywołanie powinno nastąpić z tego samego wątku, w którym utworzono obiekt.

4.3.2.4. Funkcja Init

```
[C#]
public void Init(
    string initString);
```

Funkcja służy do ustawiania parametrów serwera. Opis znajduje się w punkcie Konfiguracja programowa (*patrz: punkt 3*.), a dostępne opcje w punkcie Baza zmiennych (*patrz: punkt 3.6.2*.).

4.3.2.5. Funkcja SelectVar

```
[C#]
public bool SelectVar (
    IntPtr parentWindowHandle,
    ref string selectedVarName);
```

Funkcja *SelectVar* służy do interaktywnego wybierania przez użytkownika nazwy zmiennej z bazy zmiennych. Jej wywołanie powoduje wyświetlenie okna dialogowego zawierającego listę zmiennych zdefiniowanych w bazie zmiennych. Użytkownik może wybrać jedną z nich i zatwierdzić wybór naciskając przycisk *OK* lub zrezygnować z wyboru naciskając przycisk *Anuluj*.

Funkcja *SelectVar* zwraca wartość *true*, jeżeli użytkownik nacisnął przycisk *OK* i zatwierdził wybór. Wartość *false* jest zwracana, jeżeli użytkownik zrezygnował z wyboru naciskając przycisk *Anuluj*.

Parametr *parentWindowHandle* jest parametrem wejściowym. Jako wartość tego parametru należy przekazać uchwyt okna, z którego wywołano funkcję. W obiektach klasy *Form* uchwyt okna zwraca własność *Handle*.

Parametr *selectedVarName* jest parametrem wejściowym i wyjściowym. Jeżeli na wejściu parametr ten zawiera nazwę zmiennej, to po wyświetleniu okna wyboru zmiennej zmienna o tej nazwie jest podświetlona. Na wyjściu, jeżeli funkcja *SelectedVarName* zwróciła wartość *true*, to parametr ten zawiera nazwę wybranej zmiennej.

4.3.2.6. Funkcja SelectVars

```
[C#]
public bool SelectVars(
    IntPtr parentWindowHandle,
    ref string[] selectedVarNames);
```

Funkcja *SelectVars* służy do interaktywnego wybierania przez użytkownika nazw zmiennych z bazy zmiennych. Jej wywołanie powoduje wyświetlenie okna dialogowego zawierającego listę nazw zmiennych zdefiniowanych w bazie zmiennych. Użytkownik może wybrać jedną lub więcej z nich i zatwierdzić wybór naciskając przycisk *OK* lub zrezygnować z wyboru naciskając przycisk *Anuluj*.

Funkcja *SelectVars* zwraca wartość *true*, jeżeli użytkownik nacisnął przycisk *OK* i zatwierdził wybór. Wartość *false* jest zwracana, jeżeli użytkownik zrezygnował z wyboru naciskając przycisk *Anuluj*.

Parametr *parentWindowHandle* jest parametrem wejściowym. Jako wartość tego parametru należy przekazać uchwyt okna, z którego wywołano funkcję. W obiektach klasy *Form* uchwyt okna przechowywany jest we własności *Handle*.

Parametr *selectedVarName* jest parametrem wyjściowym. Jeżeli funkcja *selectedVarName* zwróciła wartość *true,* to parametr ten zawiera tablicę nazw wybranych zmiennych.

4.3.2.7. Funkcja SelectAttribute

```
[C#]
public bool SelectAttribute(
```

```
    IntPtr parentWindowHandle,
    ref string selectedAttributeName);
```

Funkcja *SelectAttribute* służy do interaktywnego wybierania przez użytkownika nazwy atrybutu. Jej wywołanie powoduje wyświetlenie okna dialogowego zawierającego listę nazw atrybutów zdefiniowanych w bazie zmiennych. Użytkownik może wybrać jedną z nich i zatwierdzić wybór naciskając przycisk *OK* lub zrezygnować z wyboru naciskając przycisk *Anuluj*.

Funkcja *SelectAttribute* zwraca wartość *true*, jeżeli użytkownik nacisnął przycisk *OK* i zatwierdził wybór. Wartość *false* jest zwracana, jeżeli użytkownik zrezygnował z wyboru naciskając przycisk *Anuluj*.

Parametr *parentWindowHandle* jest parametrem wejściowym. Jako wartość tego parametru należy przekazać uchwyt okna, z którego wywołano funkcję. W obiektach klasy *Form* uchwyt okna przechowywany jest we własności *Handle*.

Parametr *selectedAttributeName* jest parametrem wejściowym i wyjściowym. Jeżeli na wejściu parametr ten zawierał nazwę atrybutu, to po wyświetleniu okna wyboru atrybutu atrybut o tej nazwie jest podświetlony. Jeżeli funkcja *SelectAttribute* zwróciła wartość *true,* to parametr ten zawiera nazwę wybranego atrybutu.

5. Opis stanu pomiaru

5.1. Opis stanu pomiaru

Do opisania stanu pomiaru używana jest trójka: stempel czasu, jakość i wartość. W zależności od serwera trójka może być przedstawiana jako struktura, trzy parametry lub tekst, w którym elementy oddzielone są znakiem separatora.

Stempel czasu podawany jest w formacie naturalnym dla danego serwera. Czas zawsze podawany jest jako czas lokalny (z wyjątkiem serwera OPC, gdzie stosowany jest czas UTC).

Jakość podawana jest jako liczba 16-bitowa (w przypadku serwerów danych bieżących) lub 32-bitowa (przypadku serwerów danych archiwalnych). W liczbie tej znaczenie bitów zgodne jest ze specyfikacją OPC 2.05. Opis bitów podano w rozdziałach poniżej.

Wartość pomiaru może przyjąć jeden z następujących typów:

- iczba 16-bitowa ze znakiem lub bez znaku;
- iczba 32-bitowa ze znakiem lub bez znaku;
- iczba rzeczywista pojedynczej lub podwójnej precyzji;
- tekst;
- tablica liczb.

5.2. Jakość pomiaru

Element *jakość* reprezentuje jakość stanu wartości pomiaru.

Dolne 8 bitów (bity 0-7) flagi jakości są zdefiniowane jako trzy pola bitowe: *Quality*, *Substatus* i *Limit;* bity te ułożone są w następujący sposób:

```
QQSSSSLL
```

Bity 8-15 flagi jakości są pozostawione do wykorzystania przez twórców oprogramowania.

Bity te określane są jako pole bitowe *Vendor*.

5.3. Pole bitowe Quality

QQ	Wartość bitów	Definicja	Opis
0	00SSSSLL (0x00)	*Bad*	Wartość nie jest dostępna; szczegóły w polu *Substatus*.
1	01SSSSLL (0x40)	*Uncertain*	Jakość wartości jest niepewna; szczegóły w polu *Substatus*. W aplikacjach systemu asix jakość niepewną traktuje się jak „prawie" złą. Sposób wyświetlania wartości o jakości niepewnej jest taki sam jak wartość o jakości złej i wartości pola Substatus równym Last Known Value.
2	10SSSSLL	*N/A*	Nieużywane przez OPC.
3	11SSSSLL (0xC0)	*Good*	Jakość wartości jest dobra

Zaleca się, aby klient sprawdzał co najmniej zawartość pola bitowego *Quality*. Sprawdzanie zwykle odbywa się przez wykonanie operacji iloczynu bitowego pola *Quality* i maski pola bitowego *Quality* o wartość *0xC0*. Wynik porównuje się ze stałymi *Bad*, *Uncertain* i *Good* jak w przykładzie (język C/C++/C#):

```
int q = Quality & 0xC0;
if (q == 0x00)
{
    // jakość zła
}
else
if (q == 0x40)
{
    // jakość niepewna
}
else
{
   // jakość dobra.
}
```

5.4. Pole bitowe Substatus dla jakości Bad (złej)

SSSS	Wartość bitów	Definicja	Opis
0	000000LL (0x00)	*Non-specific*	Wartość jest zła, ale nie jest znany właściwy powód tego stanu.
1	000001LL (0x04)	*Configuration Error*	Istnieje problem z konfiguracją serwera..
2	000010LL (0x08)	*Not Connected*	Wymaga się, aby wejście było logicznie dołączone a nie jest. Źródło danych nie dostarczyło wartości.
3	000011LL (0x0C)	*Device Failure*	Wykryto uszkodzenie urządzenia.
4	000100LL (0x10)	*Sensor Failure*	Wykryto uszkodzenie czujnika. Jakość jest sygnalizowana przez drajwer urządzenia lub przez zmienną kontrolną (w drugim przypadku tylko, gdy włączona jest opcja *Sprawdzaj zmienne kontrolne*).
5	000101LL (0x14)	*Last Known Value*	Błąd komunikacji. Dostępna jest ostatnia wartość. „Wiek" wartości określony jest jej przez stempel czasu.
6	000110LL (0x18)	*Comm Failure*	Błąd komunikacji. Ostatnia wartość nie jest dostępna.
7	000111LL (0x1C)	*Out of Service*	Blok nie jest skanowany lub jest zablokowany. Używane także, gdy czytana jest zmienna, która nie jest aktywna.
8-15		*N/A*	Nieużywane przez OPC.

5.5. Pole bitowe Substatus dla jakości UNCERTAIN (niepewnej*)*

SSSS	Wartość bitów	Definicja	Opis
0	010000LL (0x40)	*Non-specific*	Wartość jest niepewna, ale nie jest znany właściwy powód tego stanu.
1	010001LL (0x44)	*Last Usable Value*	Urządzenie, które wpisywało wartości zmiennej zaprzestało wpisywania. Wartość powinna być rozumiana jako nieświeża.
2-3		*N/A*	Nieużywane przez OPC.
4	010100LL (0x50)	*Sensor Not Accurate*	Albo wartość przekroczyła któryś z limitów czujnika lub czujnik jest rozkalibrowany.
5	010101LL (0x54)	*Engineering Units Exceeded*	Wartość przekracza limity zdefiniowane dla pomiaru.
6	010110LL (0x58)	*Sub-Normal*	Wartość jest określana na podstawie kilku źródeł i liczba dostępnych źródeł jest mniejsza od wymaganej.
7-15		*N/A*	Nieużywane przez OPC.

5.6. Pole bitowe dla Substatus dla jakość GOOD (dobrej)

SSSS	Wartość bitów	Definicja	Opis
0	110000LL(0xC0)	*Non-specific*	Wartość jest dobra.
1-5		*N/A*	Nieużywane przez OPC.
6	110110LL(0xD8)	*Local Override*	Wartość została nadpisana. Zwykle oznacza to, że wejście zostało odłączone i ręcznie wprowadzono nową wartość.
7-15		*N/A*	Nieużywane przez OPC.

5.7. Pole bitowe Limit

Pole *Limit* jest ważne niezależnie od zawartości pól *Quality* i *Substatus*.

LL	Wartość bitowa	Definicja	Opis
0	QQSSSS00 (0x00)	*Not Limited*	Wartość może rosnąć i maleć.
1	QQSSSS01 (0x01)	*Low Limited*	Wartość przekroczyła dolny limit.
2	QQSSSS10 (0x02)	*High Limited*	Wartość przekroczyła górny limit.
3	QQSSSS11 (0x03)	*Constant*	Wartość jest stała i nie może się zmieniać.

Przy odczycie wartości zmiennej bieżącej z systemu **asix** pole bitowe *Limit* przyjmuje zwykle wartość *Not Limited*. Jeżeli włączona jest opcja serwera danych bieżących *Sprawdzaj limity zmiennych,* to przy przekroczeniu limitów ostrzegawczych pojawią się wartości *Low Limited* lub *High Limited*. Przy przekroczeniu limitów alarmowych dodatkowo włączana jest flaga *Second Limit* w polu bitowym *Vendor*.

Przy odczycie wartości atrybutów zmiennej z bazy zmiennych pole bitowe *Limit* przyjmuje zawsze wartość *Constant*.

5.8. Pole bitowe Vendor

Serwery pakietu AsixConnect wykorzystują jedną flagę z tego pola o nazwie *Second Limit*.

Wartość szesnastkowa	Definicja	Opis
0x0800	*Second Limit*	Wartość przekroczyła drugi limit, czyli limit alarmowy. Pole bitowe *Limit* określa czy przekroczono limit górny czy dolny.

Flaga *Second Limit* może pojawić się tylko wtedy, gdy włączona jest opcja serwera danych bieżących o nazwie *Sprawdzaj limity zmiennych*.

5.9. Pole bitowe danych archiwalnych

Flagi specyficzne dla danych archiwalnych mieszczą się w zakresie numerów bitów 16-31. Flagi te są wykorzystywane przez serwer .NET danych archiwalnych i serwer *Web Service.*

Wartość szesnastkowa	**Definicja**	**Opis**
0x00100000	*Quality Bad – No Bound*	Nie można podać wartości brzegowej, ponieważ nie jest dostępne archiwum dla chwili czasowej odpowiadającej temu brzegowi okresu czasu.
0x00200000	*Quality Bad – No Data*	Wartość zmiennej procesowej nie jest dostępna, ponieważ nie jest dostępne archiwum z danymi z podanego okresu czasu.
0x01000000	*Asix Archive End*	Wartość zmiennej procesowej nie jest dostępna, ponieważ przy czytaniu napotkano na koniec archiwum. Powodem wystąpienia tego statusu jest próba czytanie danych z okresu, który jeszcze nie nastąpił. Problemem może też być brak synchronizacji zegarów komputerów pomiedzy komputerem klienta a serwerem systemu asix.

6. Dane bieżące

6.1. Identyfikatory

W serwerach danych bieżących do jednoznacznego zidentyfikowania zmiennej procesowej wystarczy sama nazwa zmiennej, tj. nazwa używana w aplikacji systemu **asix**. Wszystkie pozostałe informacje o zmiennej potrzebne do komunikacji z systemem **asix** pobierane są z bazy zmiennych. W celu otrzymania bazy zmiennych należy zwrócić się do administratora aplikacji systemu **asix**, z której mają być pobierane dane.

W opcjach kanału, interaktywnie lub programowo, należy określić pełną ścieżkę pliku zestawu zmiennych bazy zmiennych. Szczegóły znajdują się w rozdziale dotyczącym konfiguracji serwerów Baza zmiennych (*patrz: punkt 3.6.2*.).

W najprostszym przypadku identyfikatorem jest sama nazwa zmiennej, ale identyfikatory mogą być znacznie bardziej złożone. Identyfikatory dzielimy na proste i pośrednie.

Identyfikatory proste

Zestawienie identyfikatorów prostych podano w kolejno prezentowanej tabeli. Zapis <*Zmienna*> oznacza nazwę zmiennej w aplikacji systemu **asix**. Zapis <*Atrybut*> oznacza nazwę atrybutu w bazie zmiennych aplikacji systemu **asix**.

Identyfikator	**Opis**	**Możliwe typy wartości**
<*Zmienna*> <*Zmienna*>.*CV*	Wartość bieżąca zmiennej.	R4, I2, I4, UI2, UI4
<*Zmienna*>.*DATA_TYPE*	Typ kanoniczny wartości bieżącej zmiennej.	I2
<*Zmienna*>.*EU_UNIT*	Jednostka zmiennej.	STRING
<*Zmienna*>.*DESC*	Opis zmiennej.	STRING
<*Zmienna*>.*HI_EU*	Wartość maksymalna wartości bieżącej zmiennej.	R8
<*Zmienna*>.*LO_EU*	Wartość minimalna wartości bieżącej zmiennej.	R8
<*Zmienna*>.<*Atrybut*>	Wartość atrybutu zmiennej.	STRING, R8
<*Zmienna*>.*FormatedCV*	Wartość bieżąca zmiennej formatowana zgodnie z wartością atrybutu *Format* tej samej zmiennej.	STRING
<*Zmienna*>.*PercentageCV*	Procentowa wartość bieżąca zmiennej.	R8
<*Zmienna*>.*BarCVs*	Procentowa wartość bieżąca zmiennej z uwzględnieniem bazy słupka oraz procentowa wartość bieżąca bazy słupka (tablica dwuelementowa).	[R8, R8]
<*Zmienna*>.<*Atrybut*>.*CV*	Wartość atrybutu zmiennej w bazie zmiennych, jeżeli wartość ta jest liczbą.	R8
<*Zmienna*>.<*Atrybut*>.*CV*. *FormatedCV*	Wartość atrybutu zmiennej formatowana zgodnie z wartością atrybutu *Format* tej samej zmiennej. Surowa wartość atrybutu jest liczbą.	STRING
<*Zmienna*>.<*Atrybut*>. *CV.PercentageCV*	Procentowa wartość atrybutu zmiennej. Surowa wartość atrybutu jest liczbą.	R8

Identyfikatory pośrednie

Identyfikator pośredni zawsze zaczyna się od frazy <*Zmienna*>.<*Atrybut*>.*CV*. Atrybut <*Atrybut*> zmiennej <*Zmienna*> musi zawierać nazwę innej zmiennej – taka zmienna nazywana jest zmienną pośrednią.

Przykładem zmiennej pośredniej jest zmienna udostępniająca wartości limitu ostrzegawczego czy alarmowego innej zmiennej.

Zestawienie możliwości tworzenia identyfikatorów pośrednich podano w kolejnej tabeli.

Identyfikator	Opis	Typ
<Zmienna>.<Atrybut>.CV	Wartość bieżąca zmiennej pośredniej.	R4, I2, I4, UI2, UI4
<Zmienna>.<Atrybut>.CV. FormatedCV	Wartość bieżąca zmiennej pośredniej formatowana zgodnie z wartością atrybutu *Format* tejże zmiennej pośredniej.	STRING
<Zmienna>.<Atrybut>. CV.PercentageCV	Procentowa wartość bieżąca zmiennej pośredniej.	R8
<Zmienna>.<Atrybut>.CV.BarCV	Procentowa wartość bieżąca zmiennej pośredniej z uwzględnieniem bazy słupka oraz procentowa wartość bieżąca bazy słupka (tablica dwuelementowa).	[R8, R8]
<Zmienna>.<Atrybut>.CV .<Atrybut2>	Wartość atrybutu *<Atrybut2>* zmiennej pośredniej.	STRING, R8

6.2. Praca bez bazy zmiennych

Generalnie zaleca się używanie bazy zmiennych i korzystanie z informacji zawartych w tym rozdziale powinno mieć miejsce tylko w wyjątkowych przypadkach.

Serwery danych bieżących pakietu AsixConnect akceptują również identyfikatory opisowe zmiennych; jeżeli są używane tylko identyfikatory opisowe, to nie jest potrzebne ładowanie bazy zmiennych.

W czasie sprawdzania poprawności identyfikatora opisowego nie jest wykonywana żadna komunikacja z systemem **asix** mająca na celu jego weryfikację. Ewentualne późniejsze błędy związane z komunikacją z systemem **asix** sygnalizowane są w polu/parametrze *jakość* (ang. *quality*) przy odczycie wartości zmiennej i poprzez wynik operacji odczytu i zapisu wartości zmiennych.

Identyfikator opisowy ma postać:

```
asmen.<nazwa kanału>.<nazwa zmiennej>.<typ zmiennej>
```

gdzie:

- *nazwa kanału* – nazwa grupy zmiennych bieżących zarejestrowanych w module Asmen aplikacji systemu **asix**;
- *nazwa zmiennej* – nazwa, pod jaką zmienna procesowa jest znana w aplikacji systemu **asix**;
- *typ zmiennej* – dwa lub trzy znaki określające typ zmiennej zgodnie z kolejno prezentowaną tabelą.

Typ zmiennej	Opis typu zmiennej
R4	Zmienna rzeczywista
I2	Zmienna całkowita 16-bitowa ze znakiem
UI2	Zmienna całkowita 16-bitowa bez znaku (słowo)
I4	Zmienna całkowita 32-bitowa ze znakiem
UI4	Zmienna całkowita 32-bitowa bez znaku (podwójne słowo)
UI1	Zmienna całkowita 8-bitowa bez znaku (bajt)

Typ zmiennej w systemie **asix** można określić na podstawie jej funkcji przeliczającej.

Przykład identyfikatora zmiennej:

```
asmen.Camac.06C_A111AF00.R4
```

Identyfikator ten oznacza zmienną procesową o nazwie *06C_A111AF00* dostępną w kanale *Camac*. Zmienna ta przyjmuje wartości typu rzeczywistego.

UWAGA:
W systemie **asix** w identyfikatorach nazw kanałów i zmiennych rozróżniane są duże i małe litery. Może to być czasem powodem problemów, ponieważ niektóre klienty DDE automatycznie zmieniają wszystkie litery na małe lub na duże.

6.3. Określanie praw zapisu

6.3.1. Zapis prosty

Zapis prosty jest to operacja, w której zmiennej procesowej nadawana jest nowa wartość. Stempel czasu (czyli czas bieżący) jest nadawany przez system **asix**; przyjmuje się, że jakość jest dobra.

Operacja zapisu prostego powiedzie się, jeżeli:
- zmienna jest zmienną do zapisu; zależy to od drajwera sterownika i od samego sterownika; zmienne najczęściej są zmiennymi do zapisu;
- klient ma prawo do nadawania zmiennej w kanale, do którego zmienna należy, nowej wartości.

Klient ma zawsze prawo do zapisu zmiennych w dowolnym kanale w aplikacji systemu **asix** pracującej na tym samym komputerze co klient. Aby nadać klientowi pracującemu na innym komputerze prawo do zapisu zmiennych w kanale, należy w pliku aplikacji systemu **asix**, w sekcji ASMEN, umieścić następujący wpis:

```
ZEZWOLENIE_ZAPISU = <nazwa kanału>, <nazwa komputera klienta>
```

Zapis *<nazwa kanału>* oznacza nazwę kanału systemu **asix**, do którego chcemy nadać prawa do zapisu. Zapis *<nazwa komputera klienta>* oznacza nazwę komputera klienta w systemie **asix**, który ma mieć prawo zapisu do tego kanału. Nazwa komputera klienta określana jest w sposób opisany w poniższej tabeli.

Opis konfiguracji oprogramowania klienta	**Nazwa komputera klienta w systemie asix**
Używany jest tylko pakiet AsixConnect i nie utworzono pliku *aslink.ini* lub w sekcji *ASLINK*, nie zdefiniowano linii *Nazwa*.	Nazwa komputera klienta w systemie WINDOWS plus kropka dodana na końcu. Np. dla komputera o nazwie „CLI”, nazwą komputera w systemie asix będzie „CLI.”.
Używany jest tylko pakiet AsixConnect i utworzono pliku *aslink.ini*, w sekcji *ASLINK* zdefiniowano linię *Nazwa*.	Nazwa komputera klienta w systemie asix nadawana jest w pliku *aslink.ini*, w sekcji *ASLINK*, w linii zatytułowanej *Nazwa*.
Używane są na tym samym komputerze i pakiet AsixConnect i aplikacja systemu asix.	Nazwa komputera klienta w systemie asix nadawana jest w pliku *ini* aplikacji, w sekcji *ASLINK*, w linii zatytułowanej *Nazwa*.

PRZYKŁAD:

```
ZEZWOLENIE_ZAPISU = SINEC1, KE2
```

Powyższy wpis oznacza, że serwery z pakietu AsixConnect uruchomione na komputerze, którego nazwą w sieci systemu **asix** jest KE2, mają prawo do zapisu do zmiennych w kanale SINEC1.

6.3.2. Zapis rozszerzony

Zapis rozszerzony jest to operacja, w której zmiennej procesowej nadawana jest przez klienta nowa wartość, jakość i stempel czasu.

Operacja zapisu rozszerzonego powiedzie się, jeżeli:

- zmienna jest zmienną do zapisu; zależy to od drajwera sterownika i od samego sterownika; zmienne najczęściej są zmiennymi do zapisu;
- klient ma prawo do nadawania zmiennej w kanale, do którego zmienna należy, nowej wartości, nowego statusu i nowego stempla czasu.

Aby klient miał takie prawo:

- dla kanału musi być skonfigurowane zezwolenie zapisu, do którego należy zmienna, tak jak to opisano w punkcie *Zapis rozszerzony* (*patrz: punkt 6.3.2*.);
- kanał musi być obsługiwany przez drajwer o nazwie *NONE*;
- w pliku *ini* aplikacji systemu **asix** musi istnieć sekcja o nazwie takiej samej jak nazwa kanału; sekcja musi zawierać wpis *ZAPIS_CZASU_I_STATUSU=TAK*.

PRZYKŁAD fragmentu pliku *ini*:

```
[ASMEN]
KANAL1=NONE
ZEZWOLENIE_ZAPISU = KANAL1, KE2
[KANAL1]
ZAPIS_CZASU_I_STATUSU = TAK
```

Powyższy fragmentu pliku oznacza, że serwery z pakietu AsixConnect uruchomione na komputerze, którego nazwą w sieci systemu **asix** jest *KE2*, mają prawo do zapisu wartości, statusu i stempla czasu do zmiennych w kanale *KANAL1*.

UWAGA:
Funkcja *WriteEx* obsługiwana jest od wersji systemu **asix** sprzedawanej po 2000/04/20 (moduł *Netsrv* w wersji 2.2.0 lub późniejszej oraz moduł *Asmen* w wersji 3.2.8 lub późniejszej).

6.4. Serwer Automation

6.4.1. Serwer Automation

Mechanizm Automation, stworzony przez firmę Microsoft i dostępny w ramach systemów operacyjnych z rodziny Windows, umożliwia aplikacjom udostępnianie swojej funkcjonalności jako programowalnych obiektów „zawierających" funkcje, własności i zdarzenia. Serwer Automation danych bieżących umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie danych bieżących przy użyciu mechanizmu Automation. Serwer Automation jest serwerem zaimplementowanym w postaci biblioteki dynamicznej DLL i uruchamianym wewnątrz przestrzeni adresowej klienta. Serwer ten zarejestrowany jest w systemie operacyjnym Windows jako obiekt o nazwie *XConnect.ServerCT*. Dokładny opis funkcji, własności, zdarzeń i stałych obsługiwanych przez obiekt znajduje się w dalszej części rozdziału. Serwer rejestruje w systemie operacyjnym również bibliotekę typów pod nazwą *AsixConnect 4 Type Library*.

Serwer Automation danych bieżących jest serwerem w pełni zgodnym z mechanizmem Automation. Może być w pełni wykorzystany w językach programowania wyposażonych w obsługę mechanizmu Automation takich jak *Visual Basic*, *Visual Basic for Applications* (na przykład z pakietu *Microsoft Office*) czy *Visual Basic Script*.

Przy konwersji aplikacji Visual Basic korzystającej z pakietu AsixConnect w wersji 3 należy zamienić nazwę serwera obiektu serwera *ServerCT.App* na *XConnect.ServerCT* oraz zmienić nazwę używanej biblioteki typów na *AsixConnect 4 Type Library*. Ponadto należy zauważyć, że serwer zwraca pełną jakość zmiennej, a nie tylko flagi *Good*, *Bad* i *Uncertain*.

6.4.2. Użycie serwera

Zamierzając wykonywać operacje na zmiennych bieżących przy użyciu serwera Automation należy wykonać poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Napisać program operujący na serwerze *XConnect.ServerCT*. W programie tym należy:
  - utworzyć obiekt o typie *XConnect.ServerCT*;
  - wywołać funkcję *LoadChannel* podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając procedur *Read* i *Write* zrealizować wymianę danych
    lub/i
  - uaktywnić automatyczne przysyłanie danych przez serwer za pośrednictwem zdarzeń:
    - przekazać obiektowi handler zdarzenia *DataChange*, aby otrzymywać za jego pośrednictwem informacje o aktualnych wartościach zmiennych;
    - używając procedury *SetItemActive* uaktywnić możliwość przekazywania przez serwer aktualnych wartości zmiennej;

- nadać wartość *True* własności o nazwie *Active*, aby serwer zaczął generować zdarzenia *DataChange*.

Wszystkie parametry procedur, wszystkie własności i wszystkie parametry handlerów zdarzeń są typu VARIANT.

Zamiast korzystać z kanałów, można ustawić parametry serwery (w tym załadować bazę zmiennych) używają funkcji *Init*.

6.4.3. Funkcja LoadChannel

Składnia wywołania funkcji:

```
LoadChannel ChannelName
```

Funkcja służy do inicjalizacji obiektu *XConnect.ServerCT* poprzez załadowanie kanału. Jako parametr C*hannelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja*).

6.4.4. Funkcja Init

Składnia wywołania funkcji:

```
Init InitString
```

Funkcja służy do ustawiania parametrów serwera. Opis znajduje się w punkcie Konfiguracja programowa (*patrz: punkt 3.*), a dostępne opcje w następnych punktach.

6.4.5. Funkcja Read

Składnia wywołania funkcji:

```
Read DataSource, ItemID, Value, Quality, TimeStamp
```

Funkcja *Read* służy do czytania bieżących wartości zmiennych procesowych.

Parametr *DataSource* jest parametrem wejściowym i określa źródło danych. Może on przyjąć jedną z trzech wartości: *dsCache*, *dsDevice*, *dsDrive*. Ich znaczenie opisano w kolejno prezentowanej tabeli.

Źródło danych	Opis źródła danych	Czas odczytu	Opóźnienie zmiennej	Zastosowanie	Wartość liczbowa	Uwagi
dsCache	Pamięć podręczna serwera Automation	Najkrótszy	Największe: okres próbkowania systemu asix + czas od ostatniego uaktualnienia pamięci podręcznej serwera	Zmienne, które mają być czytane wielokrotnie oraz zmienne w danej chwili aktywne	1	Czytana zmienna musi być aktywna (*patrz: punkt Funkcja SetItemActive*)
dsDevice	Pamięć podręczna systemu asix	Średni	Średnie: okres próbkowania systemu asix + czas przesłania przez sieć	Zmienne, które mają być czytane niewielką liczbę razy	2	
dsDriver	Sterownik przemysłowy	Najdłuższy	Najmniejsze: czas odczytu z urządzenia + czas przesłania przez sieć	Stosować tylko, gdy potrzebna jest wartość zmiennej procesowej „z bieżącej chwili”	3	

UWAGA:
Odczyt z pamięci podręcznej systemu **asix** (*dsDevice*) obsługiwany jest przez system **asix** w wersjach sprzedawanych po 2000/06/01 (moduł Netsrv w wersji 3.1.0 lub późniejszej). W starszych wersjach odczyt z pamięci podręcznej systemu **asix** zamieniany jest automatycznie na odczyt ze sterownika.

Parametr *ItemID* jest parametrem wejściowy; powinien zawierać identyfikator zmiennej.

Po zakończeniu operacji czytania parametr *Value* zawiera wartość zmiennej, parametr *Quality* zawiera jakość zmiennej, a parametr *TimeStamp* zawiera stempel czasu zmiennej.

Parametr *Quality* (jakość zmiennej) zawiera flagę jakości, opisaną w rozdziale *Pole bitowe Quality* (*patrz: punkt 3.5.*).

Parametr *TimeStamp* zawiera wartość typu VARIANT/DATE reprezentującą stempel czasu bieżącej wartości zmiennej; używany jest czas lokalny. Sam parametr *TimeStamp* jest parametrem opcjonalnym.

6.4.6. Funkcja SetItemActive

Składnia wywołania funkcji:

```
SetItemActive ItemID, ActiveState
```

Funkcja *SetItemActive* służy do uaktywniania i dezaktywowania zmiennej procesowej, czyli dodaje lub usuwa zmienną z listy zmiennych odświeżanych w pamięci *cache*.

Parametr *ItemID* jest parametrem wejściowy; powinien zawierać identyfikator zmiennej.

Parametr *ActiveState* powinien zawierać wartość *True*, jeżeli zmienna ma się stać aktywna i wartość *False*, jeżeli ma się stać nieaktywna.

Jeżeli zmienna jest aktywna to:

- system **asix** przesyła do serwera Automation bieżącą wartość zmiennej; okres przesyłania jest równy okresowi próbkowania zmiennej w systemie **asix**;
- wartość zmiennej może być czytana z pamięci podręcznej serwera Automation przy użyciu funkcji *Read* podając jako parametr *DataSource* stałą *dsCache*;
- wartość zmiennej może być przesyłana automatycznie do klienta za pośrednictwem zdarzenia *DataChange*.

6.4.7. Funkcja Write

Składnia wywołania funkcji:

```
Write ItemID, Value
```

Funkcja *Write* służy do nadawania zmiennej procesowej nowej wartości.

Parametr *ItemID* jest parametrem wejściowy; powinien zawierać identyfikator zmiennej, której ma być nadana nowa wartość.

Jako parametr *Value* należy podać nową wartość zmiennej procesowej. Wartością tą może być liczba całkowita lub rzeczywista. Wraz z wartością, zmiennej nadawana jest jakość dobra (*qualityGood*) i stempel czasu równy czasowi bieżącemu systemu **asix**.

Sposób konfiguracji praw zapisu opisany jest w rozdziale *Zapis prosty (patrz: punkt 6.3.1.)*.

6.4.8. Funkcja WriteEx

Składnia wywołania funkcji:

```
WriteEx ItemID, Value, Quality, TimeStamp
```

Funkcja *WriteEx* służy do nadawania zmiennej procesowej nowej wartości, nowego statusu i nowego stempla czasu.

Parametr *ItemID* jest parametrem wejściowy; powinien zawierać identyfikator zmiennej, której ma być nadana nowa wartość.

Jako parametr *Value* należy podać nową wartość zmiennej procesowej. Wartością tą może być liczba całkowita lub rzeczywista.

Jako parametr *Quality* należy podać nowy status zmiennej procesowej. Wartością tą może być jedna z trzech stałych: *qualityGood*, *qualityUncertain* lub *qualityBad*.

Jako parametr *TimeStamp* należy podać nowy stempel czasu zmiennej procesowej. Wartość ta musi być typu *VARIANT / DATE (czyli* data i czas).

Sposób konfiguracji praw zapisu opisany jest w rozdziale *Zapis rozszerzony (patrz: punkt 6.3.2.)*.

6.4.9. Własność Active

Własność *Active* jest własnością i do odczytu i do zapisu. Służy ona do sterowania przesyłaniem przez serwer do klienta bieżących wartości aktywnych zmiennych. Zmienna

jest zmienną aktywną, jeżeli wywołano funkcję *SetItemActive* z parametrem *ItemID* zawierającym nazwę tej zmiennej i parametrem *ActiveState* równym *True*. Aby bieżące wartości były przesyłane, własność *Active* musi mieć wartość *True*. Domyślnie własność *Active* ma wartość *False*.

6.4.10. Własność ServerState

Własność *ServerState* jest własnością tylko do odczytu. Zwraca ona aktualny stan serwera. Własność może przyjmować jedną z wartości ujętych w poniższej tabeli.

Wartość	**Znaczenia**	**Wartość liczbowa**
ssRunning	Serwer pracuje poprawnie, zestaw zmiennych został załadowany.	1
ssFailed	Błąd przy uruchamianiu serwera.	2
ssNoConfig	Serwer pracuje poprawnie, lecz zestaw zmiennych nie jest załadowany.	3
ssSuspended	Praca serwera jest chwilowo zatrzymana (ta wartość nie jest obecnie używana przez serwer Automation pakietu AsixConnect).	4
ssTest	Serwer pracuje w trybie testowym. Ta wartość jest używana, gdy włączona jest opcja *Symulacja danych bieżących.*	5

6.4.11. Własność StartTime

Własność *StartTime* jest własnością tylko do odczytu. Zawiera ona czas uruchomienia serwera.

6.4.12. Zdarzenie DataChange

Składnia zdarzenia:

```
DataChange ItemID, Value, Quality, TimeStamp
```

Zdarzenie *DataChange* jest wywoływane po każdej zmianie wartości zmiennej procesowej. Aby zdarzenie było wywoływane zmienna procesowa musi być aktywna (*patrz: punkt 6.4.6. Funkcja SetItemActive*) oraz własność *Active* serwera musi mieć wartość *True*.

Parametry zdarzenia *DataChange* mają takie same znaczenia jak parametry funkcji *Read*.

6.4.13. Obsługa błędów

Jeżeli wykonanie przez serwer operacji nie powiedzie się, to klient otrzyma kod błędu. Kod ten opisany jest w dokumentacji serwera. Klient może użyć standardowego mechanizmu Automation tj. funkcji *GetErrorInfo* i interfejsu *IErrorInfo* do pobrania tekstowego opisu błędu.

Jeżeli klientem jest program napisany w dowolnej z odmian języka *Visual Basic,* to po wystąpieniu błędu program przechodzi do wykonywania linii zadeklarowanej przy użyciu

instrukcji *ON ERROR GOTO*. Informacja o błędzie dostępna jest wtedy za pośrednictwem standardowego obiektu języka *Visual Basic* o nazwie *Err*. Obiekt ten udostępnia kod błędu i jego tekstowy opis.

PRZYKŁAD:

```
Sub test ()
    On Error GoTo blad
    ...
    ' kod programu korzystający z serwera Automation
    ...
    Exit Sub
blad:
    MsgBox Err.Description
End Sub
```

6.5. Serwer DDE

6.5.1. Serwer DDE

Mechanizm DDE, stworzony przez firmę Microsoft i dostępny w ramach systemów operacyjnych z rodziny Windows, umożliwia aplikacjom udostępnianie swojej funkcjonalności poprzez obsługę kilku komunikatów zdefiniowanych w tym mechanizmie. Serwer DDE danych bieżących umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie danych bieżących przy użyciu mechanizmu DDE. Serwer DDE jest serwerem zaimplementowanym w postaci programu typu EXE. Po uruchomieniu serwer ten rejestruje się w systemie operacyjnym Windows jako serwer o nazwach *ServerCTDDE* i *CTDDE*.

Serwer DDE zmiennych procesowych musi być uruchomiony zanim program klienta podejmie próbę połączenia z nim. Serwer można uruchomić z menu *Start* systemu Windows lub Eksploratorem systemu Windows startując plik wykonywalny o nazwie *ServerCTDDE.exe*. Plik ten znajduje się w katalogu, w którym zainstalowano pakiet AsixConnect.

6.5.2. Użycie serwera

Zamierzając operować na danych bieżących przy użyciu serwera DDE należy wykonać poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji systemu **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Uruchomić serwer DDE.

Korzystając z języka programowania C++ należy w programie:

- utworzyć połączenie z serwerem DDE podając jako parametr *service* tekst *ServerCTDDE* lub *CTDDE*; jako parametr *topic* należy podać nazwę wcześniej skonfigurowanego kanału;
- używając transakcji *XTYP_REQUEST* i *XTYP_POKE* zrealizować wymianę danych.

Korzystając z języka programowania Visual Basic należy w programie:

- utworzyć połączenie z serwerem DDE podając jako pierwszy parametr funkcji *DDEInitiate* tekst *CTDDE*; jako drugi parametr funkcji należy podać nazwę wcześniej skonfigurowanego kanału;

- używając funkcji *DDERequest* i *DDEPoke* zrealizować wymianę danych.

6.5.3. Operacje DDE wspierane przez serwer

Funkcje opisane w tym punkcie są funkcjami systemu Windows służącymi do komunikacji z serwerami DDE. Funkcje te są dostępne z poziomu języka C/C++ , udostępniane przez standardową bibliotekę DDEML ułatwiającą korzystanie z mechanizmu DDE. Odpowiedniki tych funkcji dostępne w języku Visual Basic zostały omówione w punkcie *Wykorzystanie serwera DDE w arkuszu programu Excel (patrz: punkt 6.5.6.)*.

DDEConnect
Funkcja *DDEConnect* służy do nawiązania połączenia z serwerem DDE.

Chcąc dołączyć się do serwera DDE systemu **asix,** należy poddać jako parametr *service* tekst *ServerCTDDE* lub *CTDDE*. Jako parametr *topic* należy podać nazwę kanału. Jeżeli użytkownik skonfigurował kanał podstawowy, to wtedy jako wartość parametru *service* należy podać tekst pusty lub jeden znak * (gwiazdka).

DDEDisconnect
Funkcja *DDEDisconnect* służy klientowi do rozłączenia połączenia z serwerem DDE.

DDEClientTransaction
Funkcja *DdeClientTransaction* umożliwia wykonanie kilku operacji zwanych w terminologii DDE transakcjami. Rodzaj transakcji zależy od wartości parametru *wType* tej funkcji. Stałe, które można użyć jako wartość tego parametru, omówione są w poniżej. W języku Visual Basic do realizacji każdego rodzaju transakcji przewidziano osobną funkcję.

XTYP_REQUEST
Transakcja *XTYP_REQUEST* służy do czytania bieżącej wartości zmiennej. Parametrem transakcji *item* jest identyfikator zmiennej, której wartość chcemy przeczytać. Dane pobierane są z pamięci podręcznej systemu **asix**. W przypadku przesyłania liczby rzeczywistej domyślnie separatorem części dziesiętnej jest taki sam znak, jaki podano w ustawieniach systemu Windows.

XTYP_POKE
Transakcja *XTYP_POKE* służy do nadania zmiennej nowej wartości. Parametrem transakcji jest identyfikator zmiennej i jej nowa wartość. W przypadku przesyłania liczby rzeczywistej domyślnie separatorem części dziesiętnej jest kropka. W opcjach serwera DDE separator części dziesiętnej można zmienić na taki sam, jaki podano w ustawieniach systemu Windows.

Sposób konfiguracji praw zapisu opisany jest w rozdziale *Zapis prosty (patrz: punkt 6.3.1.)*.

Opis ustawiania parametrów serwera znajduje się w punkcie *Konfiguracja programowa* (*patrz: punkt 3.*), a dostępne opcje w następnych punktach.

XTYP_ADVSTART
Transakcja *XTYP_ADVSTART* uaktywnia przekazywanie przez serwer klientowi aktualnej wartości zmiennej. Parametrem transakcji jest identyfikator zmiennej, której wartość chcemy otrzymywać.

Możliwe jest równoczesne przekazywanie aktualnych wartości grupy zmiennych. Jako parametr *item* należy wtedy podać listę identyfikatorów zmiennych oddzielonych od siebie

średnikami. Maksymalny rozmiar listy identyfikatorów zmiennych wynosi 255 znaków. Ograniczenie to jest narzucone przez mechanizm DDE.

XTYP_ADVSTOP

Transakcja *XTYP_ADVSTOP* zatrzymuje przekazywanie przez serwer klientowi aktualnej wartości zmiennej. Parametrem transakcji jest identyfikator zmiennej dla pojedynczych zmiennych lub lista identyfikatorów zmiennych dla grup zmiennych.

XTYP_EXECUTE

Transakcja ta nie jest używana przez serwer DDE pakietu AsixConnect.

6.5.4. Format przesyłanych danych

Serwer DDE może przesyłać dane w formacie jednokolumnowym lub czterokolumnowym. Domyślnie używany jest format jednokolumnowy, lecz można to zmienić w konfiguracji pakietu.

W formacie jednokolumnowym przesyłana jest albo wartość zmiennej albo informacja o błędzie. Informacja o błędzie przesyłana jest w postaci tekstowego opisu błędu.

W formacie czterokolumnowym w pierwszej kolumnie znajduje się status odczytu zmiennej, w drugiej wartość zmiennej, w trzeciej jakość zmiennej, a w czwartej stempel czasu.

Status mniejszy od zera oznacza, że operacja odczytu zmiennej nie powiodła się. Status ten jest jednocześnie kodem błędu odczytu. Status równy 0 lub większy od zera oznacza, że operacja odczytu powiodła się i że pozostałe kolumny zawierają poprawne wartości.

Jako jakość może być przesłana jedna z trzech wartości: *qualityGood*, *qualityUncertain* lub *qualityBad*. Znaczenie tych wartości prezentuje poniższa tabela.

Quality	**Znaczenie**	**Wartość liczbowa**
qualityGood	Druga kolumna zawiera aktualną wartość zmiennej procesowej	0xC0(192)
qualityUncertain	Druga kolumna zawiera aktualną wartość zmiennej procesowej, ale wartość jest niepewna, bo na przykład przekracza ona wartość maksymalną przewidzianą dla tego pomiaru	0x40 (64)
qualityBad	Aktualna wartość zmiennej procesowej nie jest dostępna	0

Przy czytaniu lub uaktualnianiu zmiennej dane są przesyłane w postaci jednego wiersza zawierającego jedną lub cztery kolumny. Przy uaktualnianiu grupy zmiennych dane są przesyłane w postaci tablicy, w której każdy wiersz zawiera informacje o jednej zmiennej.

Serwer DDE używa formatu przesyłania danych *CF_TEXT* czyli formatu tekstowego; liczby przesyłane są w tym formacie w reprezentacji tekstowej. Jeśli wiersz zawiera cztery kolumny, to sposób rozdzielania kolumn zależy od ustawień programu. Kolumny mogą być oddzielone od siebie albo separatorem listy, takim jaki ustawiono w Panelu Sterowania systemu Windows, albo znakiem tabulacji. Jeśli przesyłana jest tablica, to wiersze tablicy oddzielane są od siebie znakiem nowego wiersza. Dane w formacie *CF_TEXT* zakończone są znakiem o kodzie 0.

6.5.5. Przesyłanie informacji o błędach

Jeśli operacja nie powiodła się, to chcąc otrzymać kod błędu należy odczytać wartość zmiennej o specjalnej nazwie *_LastError_*. Czytając zmienną o nazwie *_LastErrorMessage_* można otrzymać tekstowy opis błędu. Nie należy zapominać o znakach podkreślenia na początku i końcu nazw.

Bardzo ważne jest, aby w programach napisanych w języku Visual Basic po operacji zapisu zawsze pobierać informację o wystąpieniu błędu, ponieważ sam język takiego błędu nie sygnalizuje.

6.5.6. Wykorzystanie serwera DDE w arkuszu programu Excel

Zakładamy, że serwer DDE przesyła dane w domyślnym formacie jednokolumnowym.

Komórka arkusza kalkulacyjnego w programie Excel może zawierać formułę będącą odwołaniem do danych zdalnych dostępnych za pośrednictwem protokołu DDE. Postać formuły jest następująca:

```
= service | topic ! item
```

Jako *service* należy podać nazwę serwera DDE programu AsixConnect, czyli *ServerCTDDE* lub *CTDDE*. Jako *topic* należy podać nazwę kanału (znak * jeżeli ma być wykorzystany kanał podstawowy). Jako *item* należy podać identyfikator zmiennej. Jeśli któryś z członów formuły zawiera spacje lub znaki inne niż alfanumeryczne to należy człon ten ująć w apostrofy. Po wprowadzeniu formuły program Excel nawiązuje połączenie z serwerem DDE i próbuje rozpocząć odświeżanie wartości zmiennej. Wynik próby może być trojaki:

- OK – w komórce pojawia się aktualna wartość zmiennej i jest ona uaktualniana;
- błąd przy próbie dostępu do zmiennej napotkany przez serwer DDE – Excel otrzymuje tekstowy opis błędu;
- błąd rozpoczęcia odświeżania z powodu niemożności dołączenie do serwera DDE – Excel wyświetla błąd N/A (ang. *not available*). Należy sprawdzić czy serwer DDE jest uruchomiony i czy podano prawidłową nazwę serwera DDE.

Nazwy funkcji DDE dostępnych w języku Visual Basic ujęto w poniższej tabeli.

Nazwa funkcji/operacji DDE	**Nazwa funkcji języka Visual Basic**
DDEConnect	*DDEInitiate*
DDEDisconnect	*DDETerminate*
DdeClientTransaction, XTYP_REQUEST	*DDERequest*
DdeClientTransaction, XTYP_POKE	*DDEPoke*

Transakcje *XTYP_ADVSTART* i *XTYP_ADVSTOP* nie mają swych odpowiedników w zestawie funkcji języka Visual Basic.

Grupy zmiennych

Aby rozpocząć uaktualnianie grupy zmiennych, należy wpisać do komórek arkusza tzw. formułę tablicową. Formułę tablicową można wprowadzić do zakresu komórek w kształcie prostokąta. Należy wybrać dany zakres i przejść do edycji naciskając klawisz F2. W formule tej *item* zawiera nazwy wielu zmiennych oddzielonych średnikami. Maksymalny rozmiar parametru *item* wynosi 255 znaków. Ograniczenie to jest narzucone przez mechanizm DDE.

Używanie grup zmiennych zamiast pojedynczych zmiennych ma ogromny wpływ na wydajność uaktualniania wartości zmiennych. Mechanizm DDE umożliwia uaktualnianie ok. 200 zmiennych w ciągu 1s (na komputerze Pentium 166 MHz). Jeśli zamiast pojedynczych zmiennych użyte zostaną grupy zmiennych, to maksymalna wydajność rośnie prawie proporcjonalnie do rozmiaru grupy. Na przykład przy grupach składających się z 25 zmiennych można przesłać nawet 3000 zmiennych na sekundę.

Wybrany zakres komórek powinien mieć jedną lub cztery kolumny szerokości w zależności od trybu przesyłania zmiennych przy uaktualnianiu, oraz tyle wierszy ile nazw zmiennych zawiera uaktualniana grupa zmiennych. Po zakończeniu edycji formuły należy wcisnąć i przytrzymać klawisze *CTRL+SHIFT* i następnie nacisnąć klawisz *ENTER*. W ten sposób wprowadzona formuła umieszczona jest we wszystkich komórkach zakresu.

Próba skasowania którejkolwiek komórki kończy się komunikatem o błędzie. Formułę tablicową można skasować dopiero po wybraniu całego zakresu, którego ona dotyczy. Aby szybko wybrać cały obszar zajęty przez formułę tablicową, należy uaktywnić jedną z komórek z tego obszaru i nacisnąć kombinację klawiszy *CTRL-/* (kreska pochylona w prawo, ang. *slash*).

6.5.7. Usługa serwer DDE

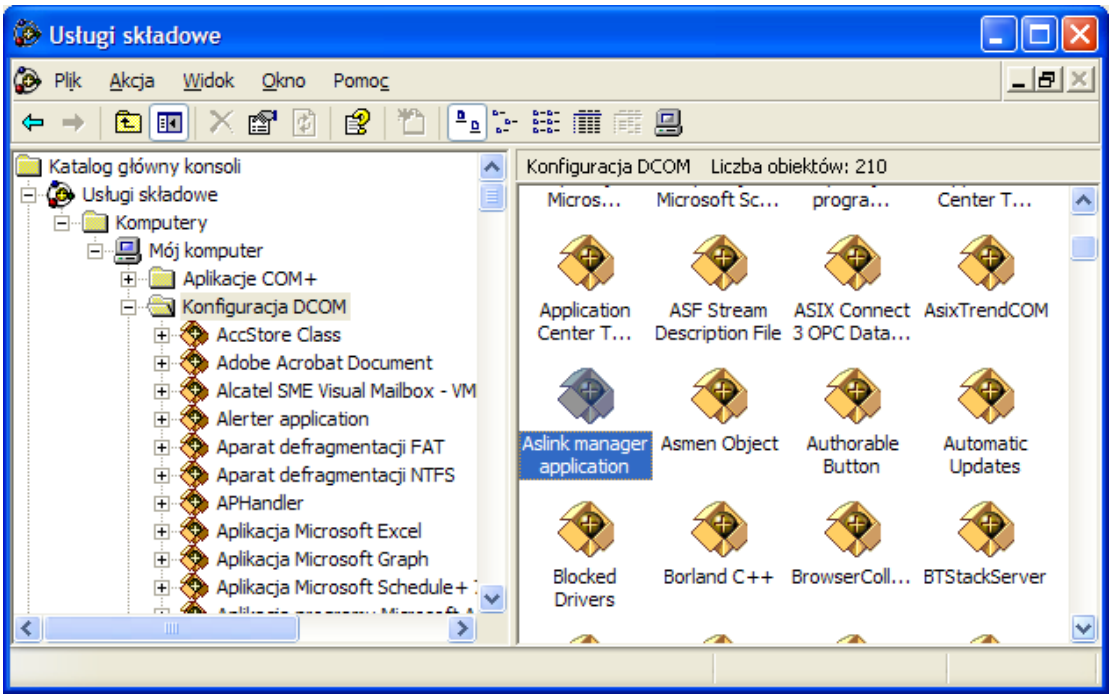
Pakiet AsixConnect zawiera moduł będący serwerem DDE, ale pracującym nie jako zwykła aplikacja, ale jako usługa systemu Windows. Moduł ten może pracować w systemach operacyjnych Windows 2003/XP/2000/NT4. Usługa DDE jest dodawana do listy usług zarejestrowanych w systemie operacyjnym podczas instalacji pakietu AsixConnect. Usługa nosi nazwę *DDE serwer danych bieżących systemu asix* i jest skonfigurowana jako „usługa uruchamiana ręcznie”.

Aby usługa była uruchamiana automatycznie podczas startu systemu operacyjnego, należy za pomocą apletu *Usługi* w *Panel Sterowania/Narzędzia Administracyjne* zmienić sposób uruchamiania usługi z *ręczny* na *automatyczny*. W systemie operacyjnym Windows NT4 aplet dostępny jest w *Panelu Sterowania.*

Usługa *Serwer DDE* korzysta z tego samego pliku konfiguracyjnego, co wszystkie serwery pakietu AsixConnect. Aby zmienić opcje usługi S*erwer DDE*, należy zmienić opcje za pomocą programu *Konfigurator* i ponownie uruchomić usługę.

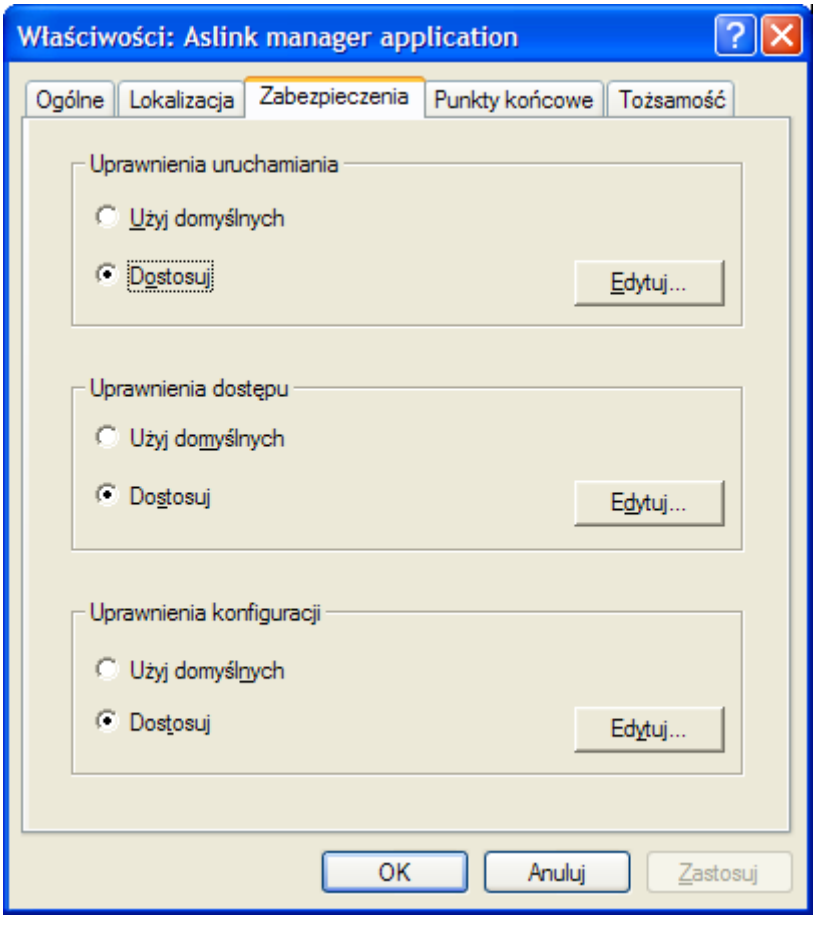
UWAGA:
Usługa *Serwer DDE* pakietu AsixConnect wymaga przeprowadzenia konfiguracji systemowych praw dostępu do komponentów pakietu. Procedura konfigurowania składa się z poniższych kroków.

KROK 1
Uruchomienie programu *dcomcnfg.exe*.

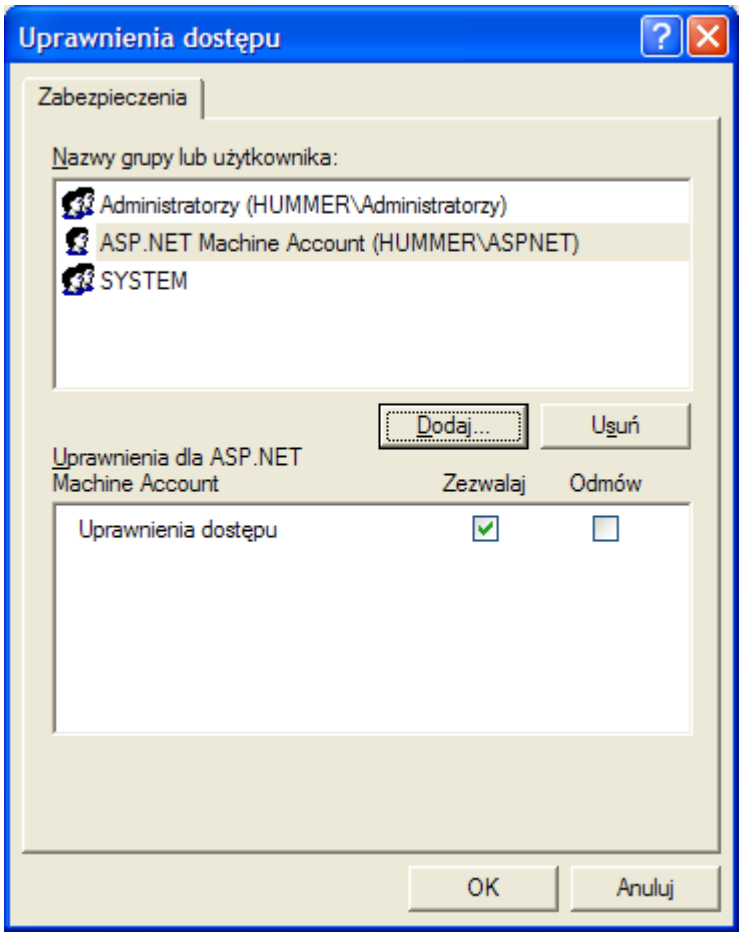


KROK 2

W oknie głównym programu należy wyszukać pozycję Konfiguracja *DCOM/Aslink manager application* i z menu *Akcja* wybrać polecenie *Właściwości (okno poniżej)*.



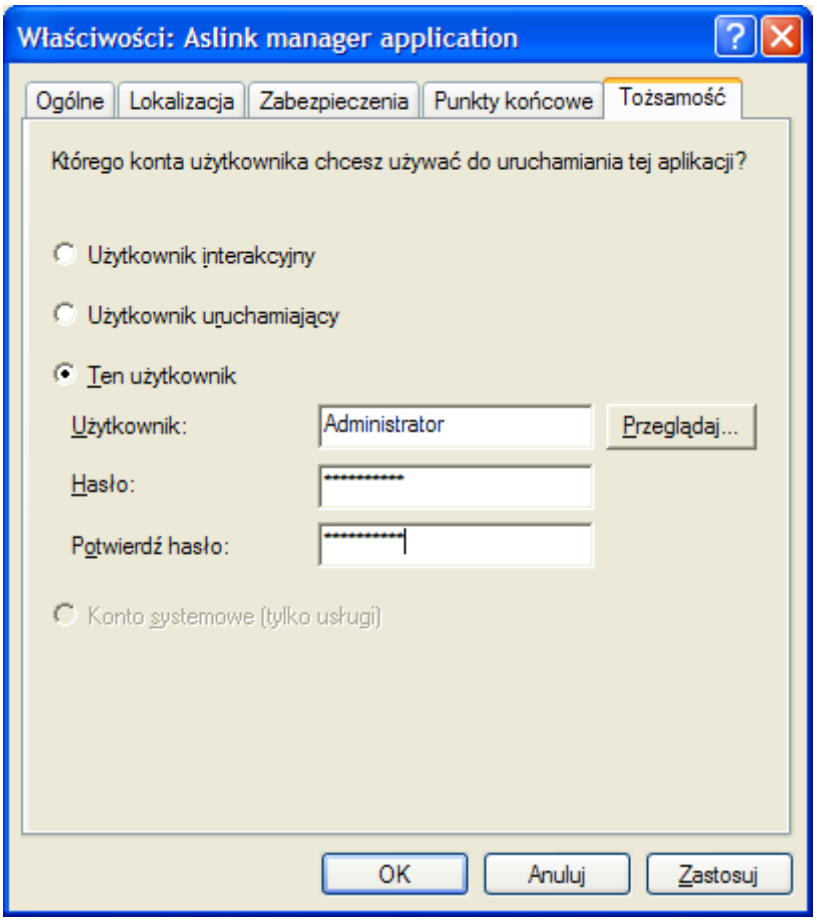
KROK 3
W zakładce *Zabezpieczenia*, po zaznaczeniu opcji *Dostosuj* i naciśnięciu przycisku *Edytuj* w polu *Uprawnienia dostępu* zostanie otworzone okno, w którym należy dodać użytkownika ASPNET do użytkowników uprawnionych i zatwierdzić operację przyciskiem *OK*.



UWAGA:
Aby zmiany dotyczące uprawnień zostały zapisane, należy zatwierdzić je przyciskiem *Zastosuj* (okno '*Właściwości: Aslink manager application*').

W oknie *Właściwości: Aslink manager object* nacisnąć (**!**) przycisk *Zastosuj*.

KROK 4
W kolejnym kroku należy określić tożsamość użytkownika, który będzie miał uprawnienia do uruchamiania modułu Aslink. Do tego celu służy zakładka *Tożsamość* w oknie '*Właściwości: Aslink manager application'*. Jako nazwę użytkownika należy podać *Administratora* i odpowiadające mu hasło. Jeżeli na danym komputerze nie będzie pracował system **asix**, można również wybrać opbcję *Użytkownik uruchamiający*. Wprowadzane zmiany zatwierdzane są przyciskiem *Zastosuj*.



6.6. Serwer OPC

6.6.1. Specyfikacja

Serwer OPC systemu **asix** jest zgodny ze specyfikacją *Data Access Custom Interface Standard Version 2.04* dostępną na serwerze internetowym *http://www.opcfoundation.org*. Specyfikacja ta jest również umieszczana w podkatalogu *Dokumentacja/OPC* podczas instalacji pakietu AsixConnect.

Specyfikacja *OPC* (*OLE for Process Control*) stworzona została przez konsorcjum OPC zawiązane przez kilkadziesiąt czołowych firm zajmujących się tworzeniem oprogramowania do kontroli procesów przemysłowych. Specyfikacja ta umożliwia tworzenie serwerów udostępniających dane bieżące z różnych systemów poprzez jeden, wspólny interfejs.

Serwer OPC udostępniający dane bieżące systemu **asix** rejestruje się w systemie Windows pod nazwą *ServerCTOPC.App* i pod tą nazwą dostępny jest we wszystkich narzędziach mogących korzystać z usług serwerów OPC.

6.6.2. Szczegóły implementacji

6.6.2.1. Wstęp

Serwer OPC jest całkowicie zgodny ze specyfikacją OPC. Specyfikacja ta zawiera jednak elementy opcjonalne lub możliwe do zaimplementowania na różne sposoby. Dlatego w rozdziale poniżej opisano sposób implementacji takich elementów.

6.6.2.2. Obiekt Serwer OPC

Serwer OPC (zwany dalej krótko serwerem) udostępniający dane bieżące systemu **asix** rejestruje się w systemie Windows pod nazwą *ServerCTOPC.App*. Serwer rejestruje się również w systemowym rejestrze kategorii komponentów jako komponent zgodny ze specyfikacją *OPC Data Access 2.0*. Ponadto dodawany jest klucz o nazwie: *HKEY_CLASSESS_ROOT\ServerCTOPC.App\OPC*, co umożliwia również starszym wersjom klientów OPC identyfikowanie serwera jako zgodnego ze specyfikacją OPC.

Serwer zaimplementowany jest w postaci lokalnego serwera COM (ang. *out-of-process*), czyli w postaci pliku wykonywalnego EXE. Wywołania OPC obsługiwane są w wątku głównym procesu serwera. Dodatkowy wątek roboczy pracujący w tle zajmuje się uaktualnianiem pamięci podręcznej wartości i statusów zmiennych; uaktualnianie wykonywane jest na podstawie danych przysyłanych za pośrednictwem modułu komunikacji sieciowej systemu **asix**.

Serwer jest serwerem typu *multiple-use* tzn. jeden proces serwera może obsługiwać wielu klientów, a każdy klient „otrzymuje" swój obiekt COM reprezentujący serwer. Serwer używa model obsługi wątków o nazwie *APARTMENT_THREADED*. W modelu tym do tworzenia i obsługi obiektów używany jest główny wątek procesu i wszystkie odwołania do obiektów są obsługiwane w ramach tego wątku.

Serwer obsługuje dwa języki: polski (identyfikator systemowy *langid* 0x0409) i angielski US (*langid* 0x0415). Jeżeli użytkownik próbuje ustalić jako obowiązujący język dialekt języka angielskiego inny niż US (na przykład UK, AUS), to taka operacja kończy się powodzeniem, lecz traktowana jest jako wybranie języka angielskiego US. Domyślnym językiem serwera jest język polski, jeżeli językiem systemu Windows jest polski, natomiast w przeciwnym wypadku - językiem domyślnym jest angielski US.

Jako status serwera zwracana jest stała *OPC_STATUS_NOCONFIG*, jeżeli w opcjach serwera nie podano żadnej bazy zmiennych lub zdefiniowanej bazy zmiennych nie udało się załadować. Obsługiwane są wtedy tylko opisowe identyfikatory zmiennych.

6.6.2.3. Przeglądanie bazy zmiennych

Serwer OPC umożliwia klientowi przeglądanie dostępnych w serwerze identyfikatorów zmiennych. Serwer posiada przestrzeń hierarchiczną zdefiniowaną przez wybrany zestaw zmiennych.

Opcja *Atrybuty OPC widoczne jako zmienne* zmienia sposób przeglądania bazy zmiennych. Domyślnie, gdy opcja jest wyłączona, gałęziami drzewa reprezentującego bazę zmiennych są grupy zmiennych a liśćmi są poszczególne zmienne. Jeżeli opcja jest włączona to wewnętrznymi gałęziami drzewa są grupy zmiennych, a zewnętrznymi gałęziami są poszczególne zmienne. Liśćmi są atrybuty zmiennych takie jak wartość, opis, jednostka, zakres górny i zakres dolny.

Zaimplementowane są wszystkie rodzaje filtrowania identyfikatorów, czyli:

- pobranie identyfikatorów, które mają identyfikatory podrzędne;
- pobranie identyfikatorów, które nie mają identyfikatorów podrzędnych;
- pobranie wszystkich identyfikatorów na bieżącym poziomie i poniżej.

Zaimplementowana jest także obsługa kryterium filtrowania, czyli wzorca, z którym muszą być zgodne zwracane identyfikatory. Do sprawdzania zgodności ze wzorcem używana jest zalecana przez specyfikację OPC funkcja *MatchPattern* działająca jak funkcja *LIKE* w

języku Visual Basic. W tabeli poniżej przedstawiono znaki specjalne dozwolone we wzorcu i ich możliwości dopasowywania.

Znak we wzorcu	**Może być dopasowany do**
?	Dowolna pojedyncza litera.
*	Zero lub więcej liter.
#	Dowolna pojedyncza cyfra (0–9).
[lista_znaków]	Dowolny pojedynczy znak z listy znaków.
[!lista_znaków]	Dowolny pojedynczy znak nie znajdujący się na liście znaków.

Nie zaimplementowano filtrowania po typie zmiennej, ponieważ serwer praktycznie każdą zmienną może przekazać w dowolnym formacie, o ile odpowiedniej konwersji może dokonać systemowa funkcja Windows *VariantChangeType*. Nie zaimplementowano również filtrowania po prawach dostępu, ponieważ nie zaimplementowano jeszcze w systemie **asix** udostępniania informacji o prawach zapisu do zmiennych.

Funkcja *GetItemID* wg specyfikacji OPC zwraca dla podanego identyfikatora „w pełni kwalifikowany" identyfikator w hierarchicznej przestrzeni identyfikatorów, czyli identyfikator wraz z całą ścieżką do niego. Serwer nie generuje jednak takich nazw, lecz zwraca jedynie sam identyfikator. Wybrana została taka implementacja, ponieważ w systemie **asix** identyfikatory zmiennych są unikalne i uzupełnianie ich ścieżką z hierarchii zmiennych jest niepotrzebne.

Funkcja *BrowseAccessPath* zawsze zwraca informację, że nie jest możliwe wyszukiwanie ścieżek dostępu.

6.6.2.4. Przeglądanie własności zmiennych

Serwer OPC umożliwia przeglądanie dostępnych własności dla danej zmiennej. Dla wszystkich zmiennych obsługiwane są następujące własności: *Typ kanoniczny*, *Wartość*, *Status*, *Stempel czasu*, *Prawa dostępu*, *Okres skanowanie serwera*, *Opis*. Dla niektórych zmiennych mogą też być dostępne następujące własności: *Jednostka*, *Górny zakres*, *Dolny zakres*.

Dla każdej własności dostępny jest jej kod, krótki opis tekstowy i typ własności. Serwer uwzględnia bieżący język przy przesyłaniu opisów własności tzn. opisy są w języku polskim, jeśli aktualnie używany jest język polski i w języku angielskim w przeciwnym wypadku.

6.6.2.5. Ścieżka dostępu do zmiennej

Parametr zmiennej *ścieżka dostępu do zmiennej* (ang. *access path*) interpretowany jest jako nazwa kanału pakietu **asix**, za pośrednictwem którego zmienna ma być pobierana. Opis definiowania kanałów znajduje się w punkcie Kanały (*patrz: punkt 3.1.*) i dalszych.

6.6.2.6. Zmienne procesowe

Jako minimalny okres odświeżania zmiennych przyjęto 1 sekunda. Jeśli klient zażąda krótszego okresu, to okres ten zostanie zwiększony do okresu minimalnego.

Akceptowany jest każdy żądany przez klienta typ zmiennej. Jeśli typ ten jest inny niż typ zmiennej w systemie **asix,** to wykonywana jest konwersja przy użyciu systemowej funkcji Windows *VariantChangeType*. Jeśli konwersja będzie niemożliwa to klient otrzyma kod błędu *OPC_E_BADTYPE*.

Jeśli dodawana zmienna ma być aktywna, to wykonywana jest próba inicjalizacji uaktualniania zmiennej z systemu **asix**. Wynik tej próby nie ma wpływu na wynik funkcji *AddItem*.

Jeśli wystąpił błąd przy inicjalizacji uaktualniania zmiennej, to cyklicznie będą podejmowane próby rozpoczęcia uaktualniania, aż do czasu, gdy operacja ta się powiedzie. Aktualny stan zmiennej jest zwracany przez funkcje czytające wartość zmiennej z pamięci podręcznej serwera (ang. *cache*).

6.6.2.7. Operacje synchroniczne

Funkcja *Read* służy do odczytu wartości własności zmiennych należących do danej grupy, ich jakości i stempla czasu. Serwer posiada wewnętrzny bufor wartości zmiennych i z niego pobierane są wartości przy odczycie zmiennych z pamięci podręcznej. Przy odczycie z urządzenia (ang. *device*) dane pobierane są za pośrednictwem sieci systemu **asix**. Na czas odczytu z urządzenia największy wpływ może mieć oczekiwanie na znalezienie serwera danych systemu **asix**. Czas ten ustawiany jest w oknie dialogowym konfigurującym serwer i standardowo wynosi 3 sekundy. Wyszukiwanie wykonywane jest tylko przy pierwszym odczycie z danego zasobu systemu **asix** i tylko, jeśli nie ma żadnych aktywnych zmiennych mających źródło danych w tym zasobie.

6.6.2.8. Operacje asynchroniczne

Operacje asynchroniczne są kolejkowane i funkcje asynchroniczne kończą swoje działanie natychmiast, dzięki czemu klient może kontynuować pracę.

Serwer może kolejkować jednocześnie tylko jedną operację tego samego typu, co jest zgodne ze specyfikacją OPC. Serwer zwraca błąd *CONNECT_E_ADVISELIMIT* przy próbie kolejkowania większej liczby transakcji.

Generalnie serwer używa głównego wątku do obsługi klientów OPC oraz wysyłania i odbierania danych z systemu **asix**. Wątek roboczy odbiera i obsługuje tylko komunikaty o zmianach wartości zmiennych. Oznacza to, że przy operacjach asynchronicznych główny wątek najpierw wstawia do kolejki informację o operacji do wykonania. Po poinformowaniu klienta o przyjęciu operacji do realizacji zacznie się realizacja operacji asynchronicznej. W trakcie realizacji operacji asynchronicznej wątek główny jest zajęty i ewentualne żądania wykonania innych operacji na rzecz klienta OPC czekają na zakończenie operacji asynchronicznej.

6.6.2.9. Zapis wartości, jakości i stempla czasu

Funkcja *Write* interfejsu *ISyncIO* została rozszerzona względem specyfikacji OPC o możliwość nadawania zmiennej jednocześnie nowej wartości, jakości i stempla czasu. Aby ją wykorzystać, należy przekazać jako odpowiedni element tablicy – parametru *pItemValues* – wartość typu *VARIANT* zawierającą trzyelementowa, jednowymiarową tablicę. Pierwszym elementem tej tablicy jest nowa wartość zmiennej, drugim elementem jakość zmiennej, a trzecim stempel czasu. Jakość musi być podana w formacie *VARIANT/VT_I4* lub do niego konwertowanym. Stempel czasu musi być podany w formacie *VARIANT/VT_DATE* lub do niego konwertowanym.

6.7. Serwer .NET

6.7.1. Użycie serwera

Klasa *ServerCT* umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie danych bieżących. Zamierzając operować na danych bieżących przy użyciu serwera .NET należy wykonać poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji systemu **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Wygenerować projekt w pakiecie Visual Studio i następnie:
  - podświetlić w drzewie projektu folder *References*, z menu *Project* wybrać polecenie *Add Reference*, nacisnąć przycisk *Browse* i z podkatalogu *c:\asix* wybrać plik *XConnectNet.dll*, nacisnąć przycisk *OK* i zamknąć okno *Add Reference*; w każdym pliku z kodem źródłowym C# należy w regionie deklaracji *using* dodać linię:

    ```
    using XConnectNet;
    ```

- napisać program operujący na serwerze *ServerCT*. W programie tym należy:
  - utworzyć obiekt typu *ServerCT*, podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając funkcji *Read* i *Write* zrealizować wymianę danych;
  - podczas wymiany danych należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer.

6.7.2. Konstruktor ServerCT

```
[C#]
public ServerCT(
  string channelName);
```

Funkcja służy do utworzenia i inicjalizacji obiektu klasy *ServerCT*.

Jako parametr *channelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja*).

6.7.3. Funkcja Dipsose

```
[C#]
public void Dispose();
```

Funkcja służy do zwolnienia zasobów używanych przez obiekt *ServerCT*. Funkcja musi być wywołana po zakończeniu używania obiektu *ServerCT*. Wywołanie powinno nastąpić z tego samego wątku, w którym utworzono obiekt.

6.7.4. Funkcja Init

```
[C#]
public void Init(
    string initString);
```

Funkcja służy do ustawiania parametrów serwera. Opis funkcji znajduje się w punkcie *Konfiguracja programowa* (*patrz: punkt 3.5.*), a dostępne opcje w następnych punktach.

6.7.5. Funkcja Read

```
[C#]
public ItemState Read(
    DataSource dataSource,
    string itemID);
public ItemState[] Read(
    DataSource dataSource,
    string[] itemIDs);
public DataSet Read(
    DataSource dataSource,
    string[] itemIDs,
    CTDataSetFlags dataSetFlags);
```

Funkcja o nazwie *Read* służy do czytania bieżących wartości zmiennych procesowych.

Parametr *dataSource* określa źródło danych. Może on przyjąć jedną z trzech wartości opisanych w tabeli poniżej.

Źródło danych	**Opis źródła danych**	**Czas odczytu**	**Opóźnienie zmiennej**	**Zastosowanie**	**Uwagi**
DataSource. Cache	Pamięć podręczna serwera .NET	Najkrótszy	Największe: okres próbkowania systemu asix + czas od ostatniego uaktualnienia pamięci podręcznej serwera	Zmienne, które mają być czytane wielokrotnie oraz zmienne w danej chwili aktywne	Czytana zmienna musi być aktywna (patrz punkt Funkcja *SetItemActive*)
DataSource. Device	Pamięć podręczna systemu asix	Średni	Średnie: okres próbkowania systemu asix + czas przesłania przez sieć	Zmienne, które mają być czytane niewielką liczbę razy	
DataSource .Driver	Sterownik przemysłowy	Najdłuższy	Najmniejsze: czas odczytu z urządzenia + czas przesłania przez sieć	Stosować tylko, gdy potrzebna jest wartość zmiennej procesowej „z bieżącej chwili”	

W pierwszej wersji funkcji *Read* drugim parametrem jest *itemId*. Jest to parametr wejściowy i powinien zawierać identyfikator zmiennej, której wartość ma być przeczytana. W wyniku funkcja zwraca strukturę typu *ItemState* zawierającą stan zmiennej.

W drugiej wersji funkcji *Read* drugim parametrem jest *itemIDs*. Jest to parametr wejściowy i powinien zawierać tablicę identyfikatorów zmiennej, których wartości mają być przeczytane. W wyniku funkcja zwraca tablicę struktur typu *ItemState* zawierających stan zmiennych.

W trzeciej wersji funkcji *Read* stan zmiennych zwracany jest jako obiekt klasy *DateSet*. Obiekt ten zawiera jedną tabelę o nazwie *CTData*. Trzecim parametrem funkcji *Read* jest wartość typu *CTDataSetFlags*.

Stała	Opis
CTDataSetFlags.Default	Tabela *CTData* zawiera kolumny o nazwach *ItemID*, *ReadResult*, *ErrorString*, *TimeStamp*, *Quality* i *ItemValue*. Każdy wiersz tabeli zawiera stan jednej zmiennej. Kolumna *ItemValue* zawiera wartości typu *Object*.
CTDataSetFlags.ItemsInColumns	Tabela *CTData* zawiera tyle kolumn ile nazw zmiennych znajduje się w parametrze *itemIDs*. Po odczycie tabela zawiera jeden wiersz z bieżącymi wartościami poszczególnych zmiennych. Wartość jest typu *Object*.
CTDataSetFlags.ItemValueAsString	Wartość zmiennej zwracana jest jako tekst. Stała może być użyta samodzielnie lub jednocześnie ze stałą *CTDataSetFlags.ItemsInColumns*.

UWAGA:
Odczyt z pamięci podręcznej systemu **asix** (parametr *dataSource* równy *DataSource.Device*) obsługiwany jest przez system **asix** w wersjach sprzedawanych po 2000/06/01 (moduł Netsrv w wersji 3.1.0 lub późniejszej). W starszych wersjach odczyt z pamięci podręcznej systemu **asix** zamieniany jest automatycznie na odczyt ze sterownika.

6.7.6. Funkcja Write

```
[C#]
public void Write (ItemState[] itemStates);
```

Funkcja *Write* służy do nadawania zmiennej procesowej nowej wartości.

Parametr *itemStates* jest parametrem wejściowy i wyjściowym. Każdy obiekt w tablicy *itemStates* powinien zawierać w polu *name* nazwę zmiennej oraz w polu *dataValue* wartość, która ma być w wyniku zapisu nadana zmiennej procesowej. Wartością tą może być liczba całkowita lub rzeczywista. Wraz z wartością zmiennej nadawana jest jakość dobra i stempel czasu równy czasowi bieżącemu w systemie operacyjnym (na serwerze aplikacji systemu **asix**).

Sposób konfiguracji praw zapisu opisany jest w rozdziale *Zapis prosty* (*patrz: punkt 6.3.1.*).

Wynik operacji znajduje się polu *result* struktury *ItemState*. Jeżeli funkcja *Succeeded()* struktury *ItemState* zwraca wartość *true,* to oznacza to, że operacja zapisu powiodła się.

6.7.7. Funkcja Write - zapis rozszerzony

Możliwe jest nadawanie zmiennej procesowej jednocześnie nowej wartości, nowego statusu i nowego stempla czasu. W tym celu w strukturze *ItemState* w polu *timeStamp* należy podać nowy stempel czasu zmiennej, a w polu *quality* nową wartość jakości zmiennej.

Sposób konfiguracji praw zapisu rozszerzonego opisany jest w rozdziale *Zapis rozszerzony (patrz: punkt 6.3.2.)*.

6.7.8. Struktura ItemState

Obiekt typu *ItemState* służy do przekazywania wartości zmiennej. Używają go funkcje *Read* i *Write* oraz zdarzenie *ItemsChange*.

Po zakończeniu działania funkcji *Read* lub przy wywołaniu zdarzenia *ItemsChange* zawartość struktury przyjmuje postać opisaną w poniższej tabeli.

Pole	**Zawartość**
ItemName	Nazwa zmiennej, której stan zawiera obiekt.
ReadResult	Wynik odczytu lub zapisu zmiennej. Wartość ujemna oznacza błąd i jest to jednocześnie kod błędu. Wartość 0 lub dodatnia oznacza, że odczyt zakończył się sukcesem i pola T*imeStamp*, Q*uality* i *ItemValue* są wypełnione.
ReadSucceeded()	Funkcja zwraca wartość *true*, jeżeli wartość pola *ReadResult* wskazuje, że odczyt lub zapis zakończył się sukcesem.
GetErrorString()	Zwraca tekstowy opis kodu błędu zawartego w polu *ReadResult*.
TimeStamp	Stempel czasu przeczytanego stanu zmiennej; używany jest czas lokalny.
Quality	Jakość przeczytanego stanu zmiennej wg specyfikacji OPC. Możliwe wartości opisane w tabeli niżej. Jakość dobra lub niepewna oznacza, że pole *ItemValue* jest wypełnione.
ItemValue	Wartości zmiennej. Pole jest typu *object* i zawiera wartość typu odpowiadającego przeczytanej zmiennej.
IsQualityGood()	Funkcja zwraca wartość *true*, jeżeli wartość pola *Quality* wskazuje, że jakość stanu zmiennej jest dobra i pole *ItemValue* jest wypełnione.

6.7.9. Funkcja SetItemActive

```
[C#]
public ItemState[] SetItemActive(
    string[] itemIDs,
    bool activeState);
```

Funkcja *SetItemActive* służy do uaktywniania i dezaktywowania zmiennych procesowych, czyli dodaje lub usuwa zmienne z listy zmiennych odświeżanych w pamięci podręcznej serwera.

Parametr *itemIDs* jest parametrem wejściowy; powinien zawierać tablicę identyfikatorów zmiennych.

Parametr *activeState* powinien zawierać wartość *true*, jeżeli zmienne mają się stać aktywne i wartość *false,* jeżeli mają się stać nieaktywne.

Jeżeli zmienna jest aktywna, to:

- system **asix** przesyła do obiektu klasy *ServerCT* bieżącą wartość zmiennej; okres przesyłania jest równy okresowi próbkowania zmiennej w systemie **asix**;
- wartość zmiennej może być czytana z pamięci podręcznej obiektu klasy *ServerCT* przy użyciu funkcji *Read* przy podaniu jako parametr d*ataSource* wartości *DataSource.Cache*;

- wartość zmiennej może być przesyłana automatycznie do klienta za pośrednictwem zdarzenia *ItemsChange,* o ile własność obiektu klasy *ServerCT* o nazwie *Active* ma wartość *true*.

6.7.10. Zdarzenie ItemsChange

Deklaracja źródła zdarzeń:

```
[C#]
public event ItemsChange OnItemsChange;
```

Deklaracja delegacji, która może być powiązana ze źródłem zdarzeń:

```
[C#]
public void ItemsChangeHandler(
    ItemState[] itemsStates);
```

Zdarzenie *OnItemsChange* jest wywoływane po każdej zmianie wartości zmiennej procesowej. Aby zdarzenie było wywoływane:

- zmienna procesowa musi być aktywna (*patrz: punkt 6.7.9. Funkcja SetItemActive*);
- własność *Active* serwera musi mieć wartość *true*.

6.7.11. Własność Active

```
[C#]
public bool Active {get; set;}
```

Własność *Active* jest własnością i do odczytu i do zapisu. Służy ona do sterowania przesyłaniem przez serwer do klienta bieżących wartości aktywnych zmiennych. Zmienna jest zmienną aktywną, jeżeli wywołano funkcję *SetItemActive* z parametrem itemIDs zawierającym nazwę tej zmiennej i parametrem a*ctiveState* równym t*rue*. Aby bieżące wartości były przesyłane, własność *Active* musi mieć wartość t*rue*. Domyślnie własność *Active* ma wartość f*alse*.

6.7.12. Praca w środowisku ASP.NET

W przypadku pracy w środowisku ASP.NET do utworzenia obiektu należy użyć wyrażenie *ServerCT.ServerPool.Get()*. Obiekt *ServerPool* jest polem statycznym klasy *ServerCT* i implementuje pulę serwerów danych bieżących. Za pomocą funkcji *Get* należy każdorazowo na początku generowania strony pobrać obiekt *ServerCT*. Deklaracja funkcji *Get* jest następująca:

```
[C#]
public ServerCT Get ();
```

Po zakończeniu generowania strony serwer musi być zwrócony do puli przy pomocy wyrażenia *ServerCT.ServerPool.Release()*. Jako parametr funkcji *Release* należy przekazać zwracany do puli serwer. Deklaracja funkcji *Release* jest następująca:

```
[C#]
public Release (ServerCT server);
```

Serwer można pobierać z puli również na początku każdej funkcji i zwracać na końcu funkcji. Aby być pewnym, że każdy pobrany serwer wróci do puli, należy główny kod funkcji zawrzeć w bloku *try* i zwracać serwer do puli w bloku *finally*.

```
private void Page_Load(object sender, System.EventArgs e)
{
    ServerCT server = null;

    try
```

```
    {
        server = ServerCT.ServerPool.Get();

        // kod funkcji
    }
    catch(Exception e)
    {
        // obługa wyjątków zgłoszonych przy pobierania serwera z puli i
        // podczas działania funkcji
    }
    finally
    {
        if (server != null)
            ServerCT.ServerPool.Release(server);
    }
}
```

Pula serwerów*:*

- tworzy dla aplikacji kilka obiektów klasy *ServerCT* (nazwa kanału pobierana jest z pliku *Web.Config* patrz Sposób specyfikowania nazwy *kanału*);
- przechowuje obiekty w pamięci podręcznej aplikacji ASP.NET;
- udostępnia obiekty dla kolejnych wywołań w ramach aplikacji ASP.NET;
- zgłasza wyjątek *PoolApplicationExeption*, gdy pula serwerów osiągnęła maksymalny rozmiar i w puli nie ma żadnego wolnego serwera.

7. Dane archiwalne

7.1. Identyfikatory

W serwerach danych archiwalnych do jednoznacznego zidentyfikowania zmiennej procesowej wystarczy sama nazwa zmiennej, tj. nazwa używana w aplikacji systemu **asix**. Wszystkie pozostałe informacje o zmiennej potrzebne do komunikacji z systemem **asix** pobierane są z bazy zmiennych. W celu otrzymania bazy zmiennych należy zwrócić się do administratora aplikacji systemu **asix**, z której mają być pobierane dane.

W opcjach kanału pakietu AsixConnect, interaktywnie lub programowo, należy określić pełną ścieżkę do pliku zestawu zmiennych bazy zmiennych. Szczegóły znajdują się w rozdziale dotyczącym konfiguracji Baza zmiennych (*patrz: punkt 3.6.2.*).

Identyfikatorem w najprostszym przypadku jest sama nazwa zmiennej, ale identyfikatory mogą być bardziej złożone. Zestawienie identyfikatorów podano w tabeli poniżej.

Identyfikator	**Opis**	**Możliwe typy wartości**
<*Zmienna*>	Wartość bieżąca zmiennej.	R8, STRING[1]
<*Zmienna*>.*FormatedCV*	Wartość bieżąca zmiennej formatowana zgodnie z wartością atrybutu *Format* tej samej zmiennej.	R8, STRING[1]
<*Zmienna*>.*PercentageCV*	Procentowa wartość bieżąca zmiennej.	R8, STRING[1]

Typ STRING dostępny jest tylko w serwerze .NET.

7.2. Praca bez bazy zmiennych

Generalnie zaleca się używanie bazy zmiennych i korzystanie z informacji zawartych w tym rozdziale powinno mieć miejsce tylko w wyjątkowych przypadkach.

Serwery danych archiwalnych pakietu AsixConnect akceptują również opisowe identyfikatory zmiennych i w przypadku używania tylko nich nie jest potrzebne ładowanie bazy zmiennych.

W czasie sprawdzania poprawności identyfikatora opisowego nie jest wykonywana żadna komunikacja z systemem **asix** mająca na celu jego weryfikację. Ewentualne późniejsze błędy związane z komunikacją z systemem **asix** sygnalizowane są poprzez wynik operacji odczytu i zapisu wartości próbek zmiennych.

Identyfikator opisowy ma postać:

```
aspad.<nazwa archiwum>.<typ archiwum>.<nazwa zmiennej>
```

gdzie:

- *nazwa archiwum* – nazwa grupy zmiennych archiwalnych zarejestrowanych w module Aspad aplikacji systemu **asix**;
- *typ archiwum* – jedna litera – typ archiwum, z którego mają być czytane wartości zmiennej;
- *nazwa zmiennej* – nazwa, pod jaką zmienna procesowa jest znana w aplikacji systemu **asix**.

System **asix** w wersji 2.68, 3.x i 4.x udostępnia następujące typy archiwów: D, M, Y, H i B.

Przykład identyfikatora zmiennej:

```
aspad.Camac.D.06C_A111AF00
```

Identyfikator ten oznacza zmienną procesową o nazwie *06C_A111AF00* dostępną w archiwum *Camac*. Archiwum jest typu *D*.

UWAGA:
W systemie **asix** w identyfikatorach nazw archiwów i zmiennych rozróżnianie są duże i małe litery.

7.3. Agregaty

7.3.1. Opis agregatów

Listę obsługiwanych agregatów opisuje kolejno prezentowana tabela.

Nazwa angielska	Nazwa polska	Sposób wyliczania agregatu
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 maksymalną a minimalną 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.
Quality_Good	*Jakość_Dobra*	Procent próbek o jakości dobrej w interwale
Quality_Uncertain	*Jakość_Niepewna*	Procent próbek o jakości niepewnej w interwale
Quality_Bad	*Jakość_Zła*	Procent próbek o jakości złej w interwale

W zależności od wartości opcji *Algorytm wyliczania agregatów* ustalana jest jakość agregatu i jego stempel czasu. W pakiecie AsixConnect 4 domyślnie używany jest algorytm Askom; w poprzedniej wersji jedynym dostępnym był algorytm OPC.

7.3.2. Algorytm Askom

W algorytmie *Askom,* na podstawie parametrów funkcji czytania danych archiwalnych czyli *periodStart*, *periodLen* i *resampleInterval,* ustalane są stemple czasu i liczba agregatów, które mają być przeczytane. Stempel czasu pierwszego agregatu równy jest *periodStart*, drugiego *periodStart* + *resampleInterval* itd.

Następnie dla każdego agregatu określa się początek i długość okresu danych, które należy przeczytać, aby wyliczyć agregat. W związku z tym, dla pierwszego agregatu dane czytane są z okresu, którego początek jest równy *periodStart* - *resampleInterval*, dla drugiego agregatu z okresu, którego początek jest równy *periodStart* itd. Wyjątkiem jest agregat *Start,* w którym dla pierwszego agregatu dane czytane są z okresu, którego początek jest równy *periodStart*, dla drugiego agregatu z okresu, którego początek jest równy *periodStart* + *resampleInterval* itd.

Jakość próbki ma wartość *qualityGood*, jeżeli procent wszystkich próbek o jakości *qualityGood* użytych do obliczenia agregatu jest równa lub przekracza wartość opcji *Próg jakości dobrej*. Domyślnie próg ten wynosi 80%.

7.3.3. Algorytm OPC

W algorytmie OPC, przy obliczaniu agregatów, okres czasu, z którego pobierane są dane, dzielony jest na interwały o stałej długości. Agregat jest obliczany dla każdego interwału przy wykorzystaniu danych zarchiwowanych w czasie trwania tego interwału. Przyjmuje się, że każdy interwał jest przedziałem zamkniętym lewostronnie i otwartym prawostronnie.

Stempel czasu agregatu jest równy czasowi początku interwału, z wyjątkiem agregatu *End*, dla którego jest on równy końcowi interwału.

Jakość próbki ma wartość *qualityGood*, jeżeli jakość wszystkich próbek użytych do obliczenia agregatu ma wartość *qualityGood*. Jeżeli jakość jednej lub więcej próbek użytych przy obliczaniu agregatu jest różna od *qualityGood*, to jakość agregatu ma wartość *qualityUncertainSubNormal*. Jeżeli status wszystkich próbek użytych przy obliczaniu agregatu ma wartość *qualityBad,* to status agregatu ma również wartość *qualityBad*.

7.4. Format czasu OPC

Składnia formatu czasu względnego OPC jest następująca:

```
keyword +/- offset +/- offset …
```

Możliwe wartości *keyword* i *offset* podane są w poniższych tabelach. Spacje i znaki tabulacji są ignorowane. Każdy parametr *offset* musi być poprzedzony liczbą całkowitą specyfikującą jego krotność i kierunek.

Keyword	**Opis**
NOW	Czas bieżący serwera danych archiwalnych.
SECOND	Początek bieżącej sekundy.
MINUTE	Początek bieżącej minuty.
HOUR	Początek bieżącej godziny.
DAY	Początek bieżącego dnia.
WEEK	Początek bieżącego tygodnia.
MONTH	Początek bieżącego miesiąca.
YEAR	Początek bieżącego roku.

Offset	Opis
S	Przesunięcie czasu w sekundach.
M	Przesunięcie czasu w minutach.
H	Przesunięcie czasu w godzinach.
D	Przesunięcie czasu w dniach.
W	Przesunięcie czasu w tygodniach.
MO	Przesunięcie czasu w miesiącach.
Y	Przesunięcie czasu w latach.

Na przykład, napis *DAY -1D+7H30M* mógłby reprezentować czas początkowy danych do raportu dziennego generowanego w dniu bieżącym (*DAY* = pierwszy stempel czasu dnia dzisiejszego). Zapis *-1D* daje pierwszy stempel czasu dnia wczorajszego, *+7H* daje godzinę 7:00 wczoraj, *+30M* daje godzinę 7:30 wczoraj; znak + w ostatnim offsecie jest przenoszony z poprzedniego offsetu.

Podobnie, *MONTH-1D+5h* oznacza godzinę 5:00 ostatniego dnia poprzedniego miesiąca, *NOW-1H15M* oznacza 1 godzinę i 15 minut temu, a *YEAR+3MO* oznacza datę 1 kwietnia bieżącego roku.

W formacie tym można wyrazić również długość okresu czasu. Należy wtedy w opisanym formacie pominąć pierwszy człon *keyword*.

7.5. Serwer Automation

7.5.1. Serwer Automation

Serwer Automation danych archiwalnych jest serwerem zaimplementowanym w postaci biblioteki dynamicznej DLL i uruchamianym wewnątrz przestrzeni adresowej klienta. Serwer ten rejestrowany jest w systemie operacyjnym Windows jako obiekt o nazwie *XConnect.ServerHT*. Serwer rejestruje w systemie operacyjnym również bibliotekę typów pod nazwą *AsixConnect 4 Type Library*.

Przy konwersji aplikacji Visual Basic korzystającej z pakietu AsixConnect w wersji 3 należ zamienić nazwę serwera obiektu serwera *ServerHT.App* na *XConnect.ServerHT* oraz zmienić nazwę używanej biblioteki typów na *AsixConnect 4 Type Library*.

7.5.2. Użycie serwera

Zamierzając wykonywać operacje na danych archiwalnych przy użyciu serwera Automation należy wykonać poniższe kroki

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Napisać program operujący na serwerze *XConnect.ServerHT*. W programie należy:
  - utworzyć obiekt o typie *XConnect.ServerHT*;

- wywołać funkcję *LoadChannel* podając jako parametr nazwę wcześniej skonfigurowanego kanału;
- używając procedur *ReadRaw* i *ReadProcessed* zrealizować wymianę danych.

Wszystkie parametry procedur są typu VARIANT.

Zamiast korzystać z kanałów, można ustawić parametry serwery (w tym załadować bazę zmiennych) używając funkcji *Init*.

7.5.3. Funkcja LoadChannel

Składnia wywołania funkcji:

```
LoadChannel ChannelName
```

Funkcja służy do inicjalizacji obiektu *XConnect.ServerHT* przez załadowanie kanału. Jako parametr C*hannelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja*).

7.5.4. Funkcja Init

Składnia wywołania funkcji:

```
Init InitString
```

Funkcja służy do ustawiania parametrów serwera. Opis znajduje się w punkcie *Konfiguracja programowa* (*patrz: punkt 3.5.*), a dostępne opcje w następnych punktach.

7.5.5. Funkcja ReadRaw

Składnia funkcji:

```
ReadRaw ItemId, DateTimeFrom, DateTimeTo, Data
```

Funkcja *ReadRaw* czyta wartości, statusy i stemple czasu zmiennej z archiwum z podanego okresu czasu. Funkcja jest przeznaczona dla klientów chcących przeczytać rzeczywiste wartości zapisane w archiwum. Dostarczane są również wartości graniczne, aby można było w razie potrzeby interpolować wartość zmiennej dla początku i końca okresu, w którym dane mają być wyświetlone.

Parametr *ItemID* jest parametrem wejściowym; powinien zawierać identyfikator zmiennej, której próbki mają być przeczytane.

Parametry *DateTimeFrom* i *DateTimeTo* są parametrami wejściowymi i powinny zawierać, odpowiednio, początek i koniec okresu, z którego mają być pobrane dane. Parametry te powinny zawierać wartość typu *DATE* lub *STRING*. Wartość typu *DATE* zawiera bezpośrednio stempel czasu; używany powinien być czas lokalny. Wartość typu *STRING* uważana jest za czas względny. Rozpoznawane są dwa formaty czasu względnego: *OPC* i *asix-raporter*. Składnia OPC opisana jest w punkcie *Format czasu OPC (patrz: punkt 7.4.)*. Format *asix-raporter* opisany jest w dokumentacji systemu **asix** w rozdziale dotyczącym modułu R*aporter*.

Po zakończeniu operacji czytania parametr *Data* zawiera tablicę przeczytanych próbek. Tablica zawiera tyle wierszy ile jest przeczytanych próbek. W każdym wierszu pierwszy element zawiera stempel czasu próbki, drugi zawiera jakość próbki, a trzeci wartość próbki.

W tablicy zwracane są również wartości graniczne dla podanego okresu czasu. Jeżeli w archiwum nie zarejestrowano próbki dokładnie w chwili czasowej *DateTimeFrom,* to zwracana jest ostania próbka sprzed tej chwili czasowej. Jeżeli w archiwum nie zarejestrowano próbki dokładnie w chwili czasowej *DateTimeTo,* to zwracana jest pierwsza próbka zarejestrowana za tą chwilą czasową. Jeżeli nie można pobrać z archiwum wartości granicznych, to zwracana jest próbka, której pole jakość ma wartość *qualityBadNoBound*.

7.5.6. Funkcja ReadProcessed

Składnia funkcji:

```
ReadProcessed ItemId, DateTimeFrom, DateTimeTo, AggregateName, ResampleInterval, Data
```

Funkcja *ReadProcessed* oblicza agregaty w podanym okresie czasu dla podanej zmiennej procesowej na podstawie danych w archiwum systemu **asix**.

Znaczenie parametrów *ItemId*, *DateTimeFrom*, *DateTimeTo* i *Data* jest takie same jak dla funkcji *ReadRaw* (Funkcja ReadRaw).

Sposób wyliczania agregatów opisano w punkcie *Agregaty*.

Parametr *AggregateName* powinien zawierać nazwę agregatu.

Parametr *ResampleInterval* powinien zawierać długość interwału. Długość można podać w formacie opisanym w punkcie *Format czasu OPC*. W formacie należy pominąć człon *keyword*.

7.5.7. Własność ShowProgresWindow

Własność *ShowProgresWindow* jest własnością i do odczytu i do zapisu. Jeżeli własność ma wartość *True,* to podczas transmisji danych z archiwum wyświetlane jest okno obrazujące przebieg tej operacji. Domyślnie własność *ShowProgresWindow* ma wartość *False*.

7.6. Serwer OLE DB

7.6.1. Serwer OLE DB

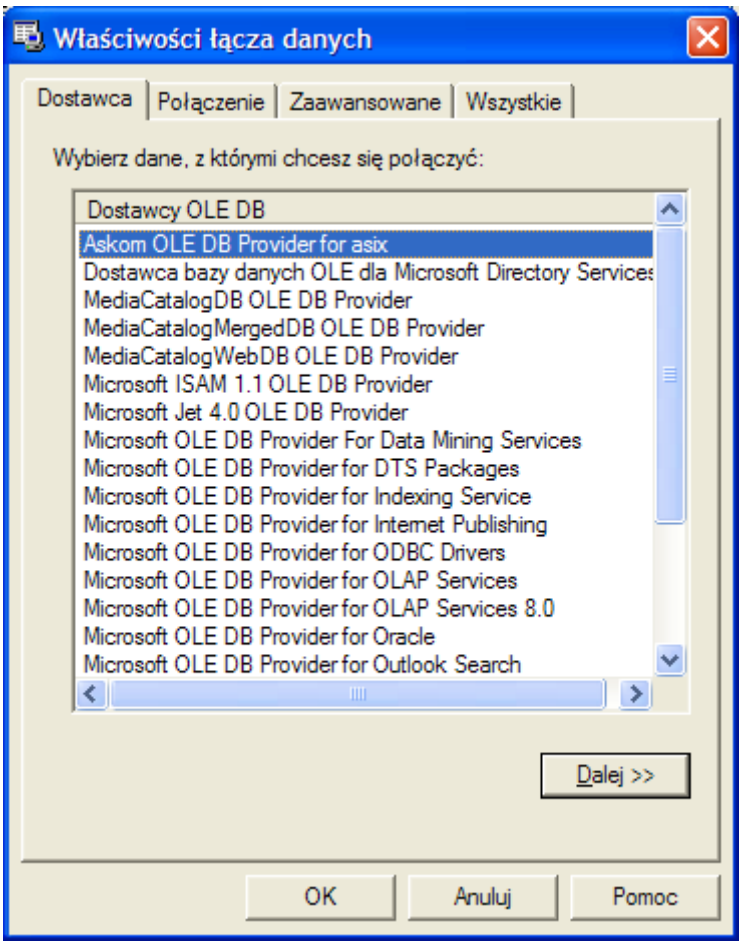
Serwer OLE DB udostępnia dane z aplikacji systemów **asix** klientom potrafiącym komunikować się bezpośrednio z serwerem używając protokołu OLE DB lub, częściej, pośrednio przy użyciu biblioteki ADO (ActiveX Data Objects). Serwer OLE DB systemu **asix** umożliwia pobieranie danych surowych lub zagregowanych jedną z dziewięciu funkcji agregujących. Możliwy jest wyłącznie odczyt danych, nie jest możliwe modyfikowanie tą drogą danych istniejących ani wstawianie nowych.

Serwer jest zgodny ze specyfikacją OLE DB firmy Microsoft, ale implementuje podstawowy zakres funkcjonalności tej specyfikacji. Dostępne są takie obiekty OLE DB jak źródło danych (*data source*), sesja (*session*), polecenie (*command*) i zbiór rekordów (*rowset*). Językiem zapytań serwera jest a*six.SQL*, którego składnia jest wzorowana na języku SQL.

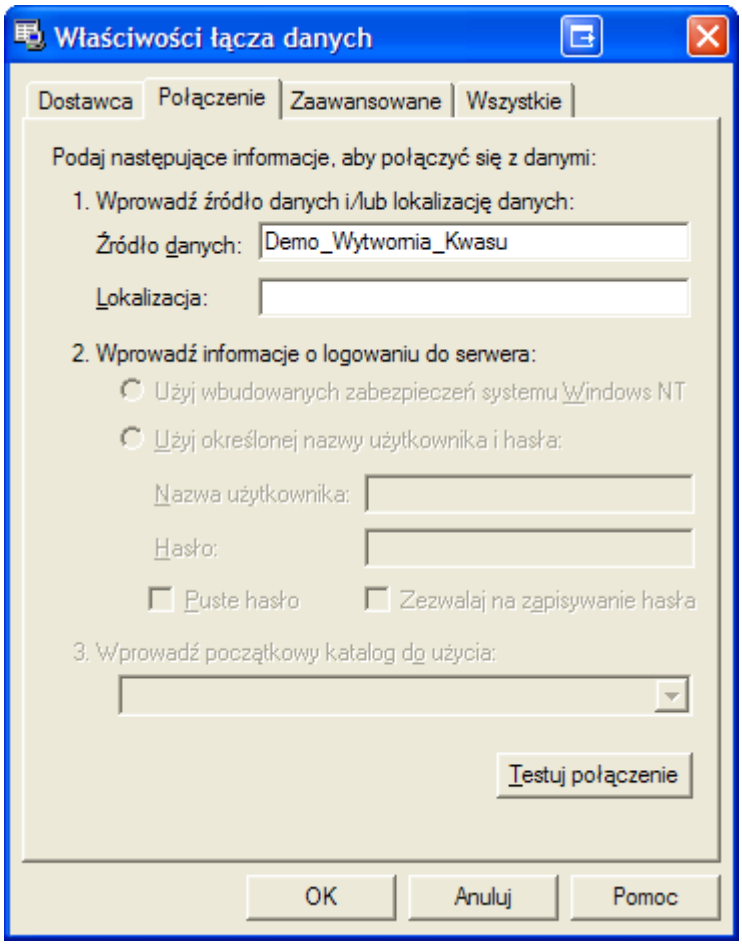
7.6.2. Identyfikacja i parametryzacja

Serwer OLE DB systemu **asix** rejestruje się w systemie operacyjnym po nazwą skróconą *ServerHTOleDB.App* i pod nazwą pełną *Askom OLE DB Provider for asix*.

Zamieszczone poniżej okna dialogowe wyświetlane są przez klientów OLE DB korzystających z systemowych mechanizmów wyboru i konfiguracji serwera OLE DB. Niektóre aplikacje mogą korzystać w tym zakresie z własnych mechanizmów.



Serwer obsługuje standardowy parametr o nazwie *Data Source* (*Źródło danych*). Jako Parametr *Data Source* należy podać nazwę kanału.



7.6.3. Tabele

Serwer udostępnia jedną tabelą o nazwie *ListaZmiennych* (w wersji angielskojęzycznej *VariablesList*). Tabela składa się z dwu kolumn o nazwach *Nazwa* i *Opis* (w wersji angielskojęzycznej *Name* i *Description*) zawierających nazwę i opis zmiennej. Tabela zawiera informacje o wszystkich zmiennych znajdujących się w bazie zmiennych wyspecyfikowanej w parametrze *Data Source*.

7.6.4. Zapytania - asix.SQL

Do opisu składni języka a*six.SQL* użyto metajęzyk stosowany przez firmę Microsoft w dokumentacji serwera Microsoft SQL 2000.

Znak metajęzyka	**Znaczenie**
DUŻE LITERY	Słowa kluczowe języka a*six.SQL*.
Kursywa	Parametry podawane przez użytkownika.
| (kreska pionowa)	Oddziela składniki wewnątrz nawiasów. Można wybrać tylko jeden ze składników.
[] (nawiasy kwadratowe)	Składnik opcjonalny.
{} (nawiasy klamrowe)	Składniki obowiązkowy.
[,...n]	Poprzedni składnik może być powtórzony wiele razy. Powtórzenia należy oddzielać przecinkiem.
[...n]	Poprzedni składnik może być powtórzony wiele razy. Powtórzenia należy oddzielić znakami pustymi.
wytłuszczenie	Tekst, który musi być wpisany tak jak podano.

Język dostępu do danych a*six.SQL* ma tylko jedno polecenie SELECT o następującej składni:

SELECT select_list
FROM var_name
WHERE search_condition
[AGGREGATE aggregate_name [, resample_interva]]
*select_list ::= { * | { column_name [AS column_name_alias]}[,...n] }*
column_name ::= { **TimeStamp** | **Status** | **Value** }
search_condition ::= { time_predicate [AND (value_status_search_condition)] }
time_predicate ::= { **TimeStamp** BETWEEN *date* AND *date* }
value_status_search_condition ::=
{ [NOT] *predicate* | (*value_status_search_condition*) }
[{ AND | OR }
[NOT] { *predicate* | (*value_status_search_condition*) }] [*...n*] }
predicate ::=
{ **Status** { = | < > | ! = } s*tatus_value*
| **Value** { = | < > | ! = | > | > = | ! > | < | < = | ! < } *number*
| **Value** IS [NOT] NULL }
s*tatus_value* ::= { **qualityGood | qualityUncertain | qualityUncertainSubNormal |**
qualityBad | qualityBadNoData | qualityBadNoBound }
aggregate_name ::= { **Start** | **End** | **Delta** | **Min** | **Max** |
Range | **Total** | **Average** | **Average0 }**

W klauzuli *SELECT* wartość * jest synonimem podania wszystkich trzech nazw kolumn w kolejności *TimeStamp*, *Quality*, *Value*. Opcjonalnie, jako *column_name_alias* można podać alternatywną nazwę zastępującą w zapytaniu nazwę kolumny. Nazwa alternatywna nie może być użyta w klauzuli WHERE. Jeżeli nazwa alternatywna zawiera spacje lub znaki *niealfanumeryczne,* to należy ją ująć w nawiasy kwadratowe.

W wyniku wykonania zapytania:

- w kolumnie *TimeStamp* przekazany będzie stempel czasu próbki;
- w kolumnie *Quality* przekazana będzie jakość próbki;
- w kolumnie *Value* przekazana będzie wartość próbki jeżeli jakość próbki będzie dobra. Jeżeli jakość próbki będzie miał wartość *qualityBad*, *qualityBadNoData* lub *qualityBadNoBound*. W kolumnie *Value* przekazana będzie wartość NULL.

W klauzuli FROM jako *var_name* należy podać nazwę jednej zmiennej aplikacji systemu **asix**. Możliwe jest użycie identyfikatora opisowego (*patrz: punkt 7.2. Praca bez bazy zmiennych*). Identyfikator opisowy należy ująć w cudzysłów.

W klauzuli WHERE jako *date* należy podać tekst ujęty w cudzysłów zawierający datę w jednym z formatów:

- data bezwzględną w formacie międzynarodowym zadeklarowanym w systemie operacyjnym;
- data bezwzględną w formacie ODBC *yyyymmddhhnnss*,
- data względna w formacie opisanym w punkcie *Format czasu OPC (patrz: punkt 7.4.)*.

Jako *number* należy podać liczbę całkowitą lub rzeczywistą w formacie dziesiętnym. Separatorem części dziesiętnej jest kropka.

W klauzuli AGGREGATE jako parametr *aggregate name* należy podać nazwę jednego z dostępnych sposobów agregowania opisanych w punkcie *Agregaty (patrz: punkt 7.3.)* . Jako *sample rate* należy podać, z jakim okresem ma być próbkowana zmienna. Okres czasu należy podać w formacie opisanym w punkcie *Format czasu OPC* (*patrz: punkt 7.4.*) przy czym należy pominąć człon *keyword*. Jeżeli parametr *sample rate* zostanie pominięty, to użyty zostanie okres próbkowania zmiennej zadeklarowany w systemie **asix**. Jeżeli cała klauzula AGGREGATE zostanie pominięta, to czytane będą wartości surowe.

Jeżeli użyta zostanie klauzula AGGREGATE, to postać przeczytanych danych jest taka sama, jak podano przy opisie funkcji *ReadProcessed (*Funkcja ReadProcessed*)* serwera Automation danych archiwalnych. Jeżeli klauzula AGGREGATE zostanie pominięta, to postać przeczytanych danych jest taka sama, jaką podano przy opisie funkcji *ReadRaw (*Funkcja ReadRaw*)*.

7.6.5. Przykłady zapytań

Odczyt surowych wartości zmiennej K8_11U14 z dnia 2002-06-16 od godziny 8:00 do 8:10. Czas podany jest w formacie polskim.

```
select Timestamp, Value from K8_11U14
where Timestamp between '2002-06-16 8:00:00'
                and '2002-06-16 8:10:00'
```

Odczyt surowych wartości zmiennej K8_11U14 z dnia 2002-06-16 od godziny 8:00 do 8:10. Czas podany jest w formacie ODBC. Zwracany jest stempel czasu, jakość i wartość próbek.

```
select * from K8_11U14
where Timestamp between '2002-06-16 8:00:00'
                and '2002-06-16 8:10:00'
```

Odczyt surowych wartości zmiennej K8_11U14 z poprzedniej godziny. Nazwa zmiennej, dla przykładu, ujęta jest w nawiasy. Nawiasy należy stosować, jeżeli nazwa zmiennej zawiera inne znaki niż litery, cyfry i znak podkreślenia.

```
select Timestamp, Value from [K8_11U14]
where Timestamp between 'hour-2h' and 'hour-1h'
```

Odczyt wartości średnich 5-minutowych zmiennej z bieżącej godziny.

```
select * from K8_11U14
where Timestamp between 'hour-1h' and 'hour'
aggregate Average, '5m'
```

Odczyt wartości minimalnych z okresów godzinowych zmiennej z poprzedniej doby.

```
select * from K8_1 U14
where Timestamp between 'day-1d' and 'day'
aggregate Min, '1h'
```

Odczyt wartości minimalnych z okresów godzinowych zmiennej z poprzedniej doby.

```
select * from K8_11U14
where Timestamp between 'day-1d' and 'day'
aggregate Min, '1h'
```

Odczyt wartości średnich 5-minutowych zmiennej z poprzedniej doby. Wybierane są tylko próbki, w których wartość jest mniejsze niż 66.

```
select * from K8_11U14
where Timestamp between 'day-1d' and 'day' and value < 66
aggregate Average, '5m'
```

Odczyt wartości średnich 5-minutowych zmiennej z bieżącej godziny. Wybierane są tylko próbki, których jakość jest dobra.

```
select * from K8_11U14
where Timestamp between 'hour' and 'hour+1h'
      and quality = qualityGood
aggregate Average, '5m'
```

Odczyt wartości średnich 5-minutowych zmiennej z poprzedniej i bieżącej godziny. Wybierane są tylko próbki, w których wartość jest dostępna (jakość nie jest zła).

```
select * from K8_11U14
where Timestamp between 'hour-1h' and 'hour+1h'
      and value is not null
aggregate Average, '5m'
```

Odczyt surowych wartości zmiennej K8_11U14 z dnia 2002-06-16 od godziny 8:00 do 16:00. Wybierane są tylko próbki, w których wartość jest mniejsza niż 63 lub większa niż 73.

```
select Timestamp, Value from K8_11U14
where Timestamp between '2002-06-16 8:00:00'
                and '2002-06-16 9:00:00'
      and (value < 63 or value > 73)
```

7.7. Serwer .NET

7.7.1. Użycie serwera

Klasa *ServerHT* umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie danych archiwalnych. Zamierzając operować na danych archiwalnych przy użyciu serwera *ServerHT,* należy wykonać poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Wejść w posiadanie bazy zmiennych aplikacji systemu **asix**, z której mają być pobierane dane bieżące lub wygenerować taką bazę.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Wygenerować projekt w pakiecie Visual Studio i następnie:
    - podświetlić w drzewie projektu folder *References*, z menu *Project* wybrać polecenie *Add Reference*, nacisnąć przycisk *Browse* i z podkatalogu *c:\asix* wybrać plik *XConnectNet.dll*, nacisnąć przycisk *OK* i zamknąć okno *Add Reference*. W każdym pliku z kodem źródłowym C# należy w regionie deklaracji *using* dodać linię:

```
using XConnectNet;
```

- napisać program operujący na serwerze *ServerHT*. W programie tym należy:
  - utworzyć obiekt typu *ServerHT*, podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając funkcji *ReadRaw* i *ReadProcessed* zrealizować wymianę danych;
  - podczas wymiany danych należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer.

7.7.2. Konstruktor ServerHT

```
[C#]
public ServerHT(
  string channelName);
```

Funkcja służy do utworzenia i inicjalizacji obiektu klasy *ServerHT*.

Jako parametr *channelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja*).

7.7.3. Funkcja Dipsose

```
[C#]
public void Dispose();
```

Funkcja służy do zwolnienia zasobów używanych przez obiekt *ServerHT*. Funkcja musi być wywołana po zakończeniu używania obiektu *ServerHT*. Wywołanie musi nastąpić z tego samego wątku, w którym utworzono obiekt.

7.7.4. Funkcja Init

```
[C#]
public void Init(
    string initString);
```

Funkcja służy do ustawiania parametrów serwera. Opis funkcji znajduje się w punkcie *Konfiguracja programowa (patrz: punkt 3.5.)*, a dostępne opcje w następnych punktach.

7.7.5. Funkcje ReadRaw

```
[C#]
public ReadRawResult ReadRaw (
    string itemID,
    DateTime periodStart,
    TimeSpan periodLen);
```

Funkcja *ReadRaw* służy do czytania archiwalnych, surowych wartości zmiennych procesowych z podanego okresu czasu. Funkcja jest przeznaczona dla klientów chcących przeczytać rzeczywiste wartości zapisane w archiwum. Funkcja dostarcza wartości graniczne, aby można było w razie potrzeby interpolować wartość zmiennej dla początku i końca okresu, w którym dane mają być wyświetlone.

Parametr *itemID* jest parametrem wejściowym i powinien zawierać identyfikator zmiennej, której próbki mają być przeczytane.

Parametry *periodStart* i *periodLen* są parametrami wejściowymi i powinny zawierać, odpowiednio, początek i długość okresu, z którego mają być pobrane dane.

Po zakończeniu operacji czytania wynik jest zwracany w postaci obiektu klasy *ReadRawResult*.

Funkcja zwraca również wartości graniczne dla podanego okresu czasu. Jeżeli w archiwum nie zarejestrowano próbki dokładnie w chwili czasowej *dateTimeFrom,* to zwracana jest ostania próbka sprzed tej chwili czasowej. Jeżeli w archiwum nie zarejestrowano próbki dokładnie w chwili czasowej *dateTimeTo,* to zwracana jest pierwsza próbka zarejestrowana za tą chwilą czasową. Jeżeli nie można pobrać z archiwum wartości granicznych, to zwracana jest próbka, której pole *jakość* ma wartość *Quality Bad – No Bound.*

7.7.6. Funkcje ReadProcessed

```
[C#]
public ReadProcessedResult ReadProcessed (
    string itemID,
    Aggregate itemAggregate,
    DateTime periodStart,
    TimeSpan periodLen,
    TimeSpan resampleInterval);

public ReadProcessedResult[] ReadProcessed (
    string[] itemIDs,
    Aggregate[]itemAggregates,
    DateTime periodStart,
    TimeSpan periodLen,
    TimeSpan resampleInterval);

public void ReadProcessed(
    string[] itemIDs,
    Aggregate[]itemAggregates,
    DateTime periodStart,
    TimeSpan periodLen,
    TimeSpan resampleInterval,
    HTDataSetFlags dataSetFlags,
    out bool allOk,
    out DataSet dataSet);
```

Funkcja *ReadProcessed* oblicza agregaty (takie jak wartość początkowa czy średnia) w podanym okresie czasu dla podanej zmiennej procesowej na podstawie danych w archiwum aplikacji systemu **asix**.

Parametr *itemID* jest parametrem wejściowym i powinien zawierać identyfikator zmiennej, której próbki mają być przeczytane. W drugiej i trzeciej wersji funkcji użyty jest parametr *itemIDs* i jako jego wartość należ podać tablicę identyfikatorów zmiennych.

Parametr *itemAggregate* powinien zawierać typ agregatu, jaki ma być użyty do przetworzenia surowych wartości archiwalnych. W drugiej i trzeciej wersji funkcji użyty jest parametr *itemAggregates* i jako jego wartość należ podać tablicę agregatów. Sposób wyliczania agregatów opisano w punkcie *Agregaty (patrz: punkt 7.3.)*. Lista obsługiwanych agregatów podana jest w tabeli poniżej.

Identyfikator	Sposób wyliczania agregatu
Aggregate.Start	Wartość na początku interwału.
Aggregate.End	Wartość na końcu interwału.
Aggregate.Delta	Różnica wartości na końcu i na początku interwału.
Aggregate.Min	Minimalna wartość w interwale.
Aggregate.Max	Maksymalna wartość w interwale
Aggregate.Range	Różnica między maksymalną a minimalną wartością w interwale.
Aggregate.Total	Suma wartości ważonych czasowo w interwale (całka po czasie).
Aggregate.Average	Średnia wartości ważonych czasowo w interwale.
Aggregate.Average0	Średnia wartości ważonych czasowo w interwale. W okresach, w których wartość jest niedostępna do obliczeń używana jest wartość 0.

Parametry *periodStart* i *periodLen* są parametrami wejściowymi i powinny zawierać, odpowiednio, początek i długość okresu, z którego mają być pobrane dane.

Parametr *resampleInterval* jest parametrem wejściowym i powinien zawierać długość interwału, dla którego wyliczane są agregaty.

Trzecia wersji funkcji *ReadProcessed* zawiera parametr d*ataSetFlags*. Jako jego wartość należy przekazać stałą typu *HTDataSetFlags*.

Stała	Opis
HTDataSetFlags.Default	Dla każdej wartości w parametrze *itemIDs* tworzona jest w tabeli *HTData* jedna kolumna o nazwie takiej jak nazwa zmiennej. Jeżeli nazwa zmiennej powtarza się, to kolejnych powtórzeń dodawane są przyrostki '-1', '-2' itd. W kolumnie tej będą umieszczane wartości próbek zmiennej. Jeżeli jakość próbki nie jest dobra to w kolumnie umieszczana jest wartość DBNull. Tabela *HTData* zawiera również jedną kolumnę o nazwie *TimeStamp* zawierającą stempel czasu próbek.
HTDataSetFlags. ShowQuality	W tabeli *HTDataSet* dla każdej zmiennej, oprócz kolumny wartości, umieszczona zostanie również kolumna z jakością próbki. Kolumna ta ma nazwę składająca się z nazwy zmiennej, do której dodano tekst '*-Qulity'*.
HTDataSetFlags. ShowErrorStringWhenReadError	W kolumnie wartości umieszczany jest opis błędu jeżeli wystąpił błąd odczytu. Jeżeli flaga nie jest użyta to umieszczana jest wartość pusta.

Po zakończeniu operacji czytania wynik jest zwracany w postaci obiektu klasy *ReadProcessedResult* dla pierwszej wersji funkcji lub w postaci tablicy obiektów klasy *ReadProcessedResult* dla drugiej wersji funkcji. Trzecia wersja funkcji zawiera dwa parametry wyjściowe *allOk* i *dataSet*. Parametr *allOk* określa czy operacje odczytu wszystkich zmiennych zakończyły się powodzeniem. Parametr *dataSet* zwraca obiekt klasy *DataSet* zawierający przeczytane dane archiwalne.

7.7.7. Funkcja ReadProcessedAsString

```
[C#]
public ReadProcessedAsStringResult ReadProcessedAsString (
    string itemID,
    Aggregate itemAggregate,
    DateTime periodStart,
    TimeSpan periodLen,
    TimeSpan resampleInterval);

public ReadProcessedAsStringResult[] ReadProcessedAsString (
    string[] itemIDs,
    Aggregate[]itemAggregates,
    DateTime periodStart,
    TimeSpan periodLen,
    TimeSpan resampleInterval);

public void ReadProcessedAsString (
    string[] itemIDs,
    Aggregate[]itemAggregates,
    DateTime periodStart,
    TimeSpan periodLen,
    TimeSpan resampleInterval,
    HTDataSetFlags dataSetFlags
    out bool allOk,
    out DataSet dataSet);
```

Funkcje *ReadProcessedAsString* służą głównie do czytania sformatowanych wartości próbek zmiennej.

Działanie funkcji *ReadProcessedAsString* jak i jej parametry są takie same jak funkcji *ReadProcessed*. Jedną różnicą jest to, że wynik zwracany jest w postaci struktur *ReadProcessedAsStringResult* dla pierwszej wersji funkcji oraz, w postaci tablicy struktur *ReadProcessedAsStringResult* dla drugiej wersji funkcji.

Trzecia wersja funkcji zawiera dwa parametry wyjściowe *allOk* i *dataSet*. Parametr *allOk* określa, czy operacje odczytu wszystkich zmiennych zakończyły się powodzeniem. Parametr *dataSet* zwraca obiekt klasy *DataSet* zawierający przeczytane dane archiwalne.

7.7.8. Funkcja RelativeDateTime

```
[C#]
public DateTime RelativeDateTime (
    string time,
    DateTime now);
```

Funkcja *RelativeDateTime* konwertuje czas podany w formacie względnym na czas bezwzględny.

7.7.9. Funkcja RelativeTimeSpan

```
[C#]
public TimeSpan RelativeTimeSpan (
    string time,
    DateTime now);
```

Funkcja *RelativeTimeSpan* konwertuje długość okresu czas podaną w formacie względnym na długość bezwzględną.

7.7.10. Klasa ReadRawResult

Obiekt klasy *ReadRawResult* służy do przekazywania wyniku działania funkcji *ReadRaw*.

Deklaracja klasy *ReadRawResult* jest następująca:

```
public class ReadRawResult
{
    public bool ReadSucceeded();
    public Int32 ReadResult;
    public string ErrorString;

    public string ItemID;
    public DateTime PeriodStart;
    public TimeSpan PeriodLen;

    public ItemSample []Samples;
};
```

Po zakończeniu funkcji czytającej dane archiwalne zawartość struktury jest następująca:

Pole	**Zawartość**
ReadSucceeded()	Funkcja zwraca wartość *true*, jeżeli wartość pola *ReadResulty* wskazuje, że operacja odczytu zakończyła się powodzeniem.
ReadResult	Kod zakończenia operacji odczytu. Wartość mniejsza od zera oznacza błąd.
ErrorString	Tekstowy opis kodu zakończenia operacji odczytu.
ItemID	Nazwa zmiennej, które dotyczy odczyt.
PeriodStart	Początek okresu czasu z którego czytano dane archiwalne.
PeriodLen	Długość okresu czasu z którego czytano dane archiwalne.
Samples	Odczytane dane archiwalne.

7.7.11. Klasa ReadProcessedResult

Obiekt klasy *ReadProcessedResult* służy do przekazywania wyniku działania funkcji *ReadProcessed*.

Deklaracja klasy *ReadProcessedResult* jest następująca:

```
public class ReadProcessedResult
{
    bool ReadSucceeded();
    Int32 ReadResult;
    string ErrorString;

    string ItemID;
    Aggregate ItemAggregate;

    DateTime PeriodStart;
    TimeSpan PeriodLen;
    TimeSpan ResampleInterval;

    ItemSample []Samples;
};
```

Po zakończeniu funkcji czytającej dane archiwalne zawartość struktury jest następująca:

Pole	**Zawartość**
ReadSucceeded()	Funkcja zwraca wartość *true*, jeżeli wartość pola *ReadResulty* wskazuje, że operacja odczytu zakończyła się powodzeniem.
ReadResult	Kod zakończenia operacji odczytu. Wartość mniejsza od zera oznacza błąd.
ErrorString	Tekstowy opis kodu zakończenia operacji odczytu.
ItemID	Nazwa zmiennej, które dotyczy odczyt.
ItemAggregate	Nazwa agregatu zmiennej.
PeriodStart	Początek okresu czasu z którego czytano dane archiwalne.
PeriodLen	Długość okresu czasu z którego czytano dane archiwalne.
ResampleInterval	Długość interwału, dla którego wyliczane są agregaty.
Samples	Odczytane dane archiwalne.

7.7.12. Klasa ReadProcessedAsStringResult

Obiekt klasy *ReadProcessedAsStringResult* służy do przekazywania wyniku działania funkcji *ReadProcessedAsString*.

Deklaracja klasy *ReadProcessedAsStringResult* jest następująca:

```
public class ReadProcessedAsStringResult
{
    public bool ReadSucceeded();
    public Int32 ReadResult;
    public string ErrorString;

    public string ItemID;
    Aggregate ItemAggregate;

    public DateTime PeriodStart;
    public TimeSpan PeriodLen;
    public TimeSpan ResampleInterval;

    public ItemStringSample []Samples;
};
```

Znaczenie pól jest takie same jak w klasie *ReadProcessedResult*. Jedyną różnicą jest typ pola *Samples*.

7.7.13. Struktura ItemSample

Obiekty struktury *ItemSample* służą do przekazywania wartości archiwalnych zmiennych. Używają ich klasy *ReadRawResult* i *ReadProcessedResult*.

Deklaracja struktury *ItemSample* jest następująca:

```
public struct ItemSample
{
    public DateTime TimeStamp;
    public Int32    Quality;
    public double   ItemValue;
```

```
    public bool     IsQualityGood();
};
```

Po zakończeniu funkcji czytającej dane archiwalne zawartość struktury jest następująca:

Pole	**Zawartość**
TimeStamp	Stempel czasu próbki zmiennej; używany jest czas lokalny.
Quality	Jakość próbki zmiennej. Jakość dobra lub niepewna oznacza, że pole *ItemValue* jest wypełnione.
ItemValue	Wartości zmiennej. Pole jest typu *double*.
IsQualityGood()	Funkcja zwraca wartość *true*, jeżeli wartość pola *Quality* wskazuje, że jakość stanu zmiennej jest dobra.

Jakość próbki może przyjąć jedną z wartości opisanych w rozdziale *Jakość pomiaru* (*patrz: punkt 5.2.*). Ponadto w przypadku jakości złej mogą być ustawione flagi specyficzne dla danych archiwalnych. Flagi te mieszczą się w zakresie numerów bitów 16-31 (*szczegóły – patrz poniższa tabela*).

Wartość bitów	**Definicja**	**Opis**
0x00100000	*Quality Bad – No Bound*	Nie można podać wartości brzegowej, ponieważ nie jest dostępne archiwum dla chwili czasowej odpowiadającej temu brzegowi okresu czasu.
0x00200000	*Quality Bad – No Data*	Wartość zmiennej procesowej nie jest dostępna, ponieważ nie jest dostępne archiwum z danymi z podanego okresu czasu.
0x01000000	*Asix Archive End*	Wartość zmiennej procesowej nie jest dostępna, ponieważ przy czytaniu napotkano na koniec archiwum. Powodem wystąpienia tego statusu jest próba czytanie danych z okresu, który jeszcze nie nastąpił. Problemem może też być brak synchronizacji zegarów komputerów miedzy komputerem klienta a serwerem systemu asix.

7.7.14. Struktura ItemStringSample

Obiekt typu *ItemStringSample* służy do przekazywania wartości archiwalnych zmiennych w postaci tekstowej. Struktura *ItemStringSample* używana jest przez klasę *ReadProcessedAsStringResult*.

Deklaracja struktury *ItemStringSample* jest następująca:

```
public struct ItemStringSample
{
    public DateTime TimeStamp;
    public Int32    Quality;
    public string   ItemValue;
```

```
    public bool     IsQualityGood();
};
```

Po zakończeniu funkcji czytającej dane archiwalne zawartość struktury jest następująca:

Pole	**Zawartość**
TimeStamp	Stempel czasu próbki zmiennej; używany jest czas lokalny.
Quality	Jakość próbki zmiennej. Możliwe wartości opisane w tabeli niżej. Jakość dobra lub niepewna oznacza, że pole *ItemValue* jest wypełnione.
ItemValue	Wartości zmiennej. Pole jest typu *string*.
IsQualityGood()	Funkcja zwraca wartość *true*, jeżeli wartość pola *Quality* wskazuje, że jakość stanu zmiennej jest dobra.

7.7.15. Obiekt DataSet

Obiekt klasy *DataSet* używany jest do zwracania danych archiwalnych przez funkcje *ReadProcessed* i *ReadProcessedAsString*. Obiekty *DataSet* zawierają dwie tabele: *HTData* i *ReadResult*.

Tabela *HTData* zawiera dane archiwalne. Pierwsza kolumna w tabeli nosi nazwę *TimeStamp* i zawiera stempel czasu próbek. Dla każdej wartości w parametrze *itemIDs* funkcji *ReadProcessed* tworzona jest w tabeli *HTData* jedna kolumna wartości o nazwie takiej jak nazwa zmiennej. Jeżeli nazwa zmiennej powtarza się, to do kolejnych powtórzeń dodawane są przyrostki ‘-1’, ‘-2’ itd. W kolumnie wartości umieszczane są wartości próbek zmiennej. Jeżeli jakość próbki nie jest dobra, to w kolumnie umieszczana jest wartość *DBNull*. W tabeli dla każdej zmiennej umieszczona może być również kolumna jakości próbek. Kolumna ta ma nazwę składająca się z nazwy zmiennej, do której dodano przyrostek ‘-*Quality*’.

W przypadku funkcji *ReadProcessed* tabela *HTData* zawiera dane w postaci zmiennoprzecinkowej; w przypadku funkcji *ReadProcessedAsString* tabela *HTData* zawiera dane w postaci tekstowej.

Tabela *ReadResult* zawiera informacje o operacji odczytu dla poszczególnych zmiennych.

Opis kolumn tabeli ujęto w kolejno prezentowanej tabeli.

Pole	Zawartość
ColumnID	Nazwa kolumny w tabeli *HTData* odpowiadającej danej zmiennej.
ItemID	Nazwa zmiennej, które dotyczy odczyt.
ItemAggregate	Nazwa agregatu.
PeriodStart	Początek okresu czasu z którego czytano dane archiwalne.
PeriodLen	Długość okresu czasu z którego czytano dane archiwalne.
ResampleInterval	Długość interwału, dla którego wyliczane są agregaty.
ReadResult	Kod zakończenia operacji odczytu. Wartość mniejsza od zera oznacza błąd.
ErrorString	Tekstowy opis kodu zakończenia operacji odczytu.

7.7.16. Praca w środowisku ASP.NET

W przypadku pracy w środowisku ASP.NET do utworzenia obiektu należy użyć wyrażenie *ServerHT.ServerPool.Get()*. Obiekt *ServerPool* jest polem statycznym klasy *ServerHT* i implementuje pulę serwerów danych bieżących. Za pomocą funkcji *Get* należy każdorazowo na początku generowania strony pobrać obiekt *ServerHT*. Deklaracja funkcji *Get* jest następująca:

```
[C#]
public ServerHT Get ();
```

Po zakończeniu generowania strony serwer musi być zwrócony do puli przy pomocy wyrażenia *ServerHT.ServerPool.Release()*. Jako parametr funkcji *Release* należy przekazać zwracany do puli serwer. Deklaracja funkcji *Release* jest następująca:

```
[C#]
public Release (ServerHT server);
```

Serwer można pobierać z puli również na początku każdej funkcji i zwracać na końcu funkcji. Aby być pewnym, że każdy pobrany serwer wróci do puli, należy główny kod funkcji zawrzeć w bloku *try* i zwracać serwer do puli w bloku *finally*.

```
private void Page_Load(object sender, System.EventArgs e)
{
    ServerHT server = null;

    try
    {
        server = ServerHT.ServerPool.Get();

        // kod funkcji np.:
        ChartXMLData.InsertChartHTML ("chart.tee", Chart1);
        ChartXMLData chartXMLData = new ChartXMLData();

        chartXMLData.Add (serverHT, "KW_A000", Aggregate.Start, "KW_A000",
                          HTChartFlags.Default, "1M");
        chartXMLData.Add (serverHT, "KW_A008", Aggregate.Average, "KW_A000",
                          HTChartFlags.Default, "1M");

        chartXMLData.SetChartPeriod (serverHT, "DAY-24H", "24H");
        chartXMLData.ReadData (serverHT, XMLIsland1);
    }
    catch(Exception e)
    {
        // obługa wyjątków zgłoszonych przy pobierania serwera z puli i
```

```
            // podczas działania funkcji
        }
        finally
        {
            if (server != null)
                ServerHT.ServerPool.Release(server);
        }
    }
```

Pula serwerów*:*

- tworzy dla aplikacji kilka obiektów klasy *ServerHT* (nazwa kanału pobierana jest z pliku *Web.Config* patrz Sposób specyfikowania nazwy *kanału*);
- przechowuje obiekty w pamięci podręcznej aplikacji ASP.NET;
- udostępnia obiekty dla kolejnych wywołań w ramach aplikacji ASP.NET;
- zgłasza wyjątek *PoolApplicationExeption*, gdy pula serwerów osiągnęła maksymalny rozmiar i w puli nie ma żadnego wolnego serwera.

8. Alarmy

8.1. Serwer .NET

8.1.1. Użycie serwera

Klasa *ServerAL* umożliwia dostęp do części funkcjonalności systemu **asix** w zakresie alarmów. Zamierzając operować na alarmach przy użyciu serwera *ServerAL* należy wykonać poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Wygenerować projekt w pakiecie Visual Studio i następnie:
  - podświetlić w drzewie projektu folder *References*, z menu *Project* wybrać polecenie *Add Reference*, nacisnąć przycisk *Browse* i z podkatalogu *c:\asix* wybrać plik *XConnectNet.dll*, nacisnąć przycisk *OK*. i zamknąć okno *Add Reference*. W każdym pliku z kodem źródłowym C# należy w regionie deklaracji *using* dodać linię:

    ```
    using XConnectNet;
    ```

- Napisać program operujący na serwerze *ServerAL*. W programie tym należy:
  - utworzyć obiekt typu *ServerAL*, podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając funkcji *ReadActive* i *ReadHistorical za*programować wymianę danych;
  - podczas wymiany danych należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer.

8.1.2. Konstruktor ServerAL

```
[C#]
public ServerAL(
  string channelName);
```

Funkcja służy do utworzenia i inicjalizacji obiektu klasy *ServerAL*.

Jako parametr *channelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja*).

8.1.3. Funkcja Dipsose

```
[C#]
public void Dispose();
```

Funkcja służy do zwolnienia zasobów używanych przez obiekt *ServerAL*. Funkcja musi być wywołana po zakończeniu używania obiektu *ServerAL*. Wywołanie musi nastąpić z tego samego wątku, w którym utworzono obiekt.

8.1.4. Funkcja Init

```
[C#]
public void Init(
    string initString);
```

Funkcja służy do ustawiania parametrów serwera. Opis funkcji znajduje się w punkcie *Konfiguracja programowa* (*patrz: punkt 3.5.*) , a dostępne opcje w następnych punktach.

8.1.5. Funkcje ReadActive

```
[C#]
public Alarm[] ReadActive();
public Alarm[] ReadActive(
    AlarmType alarmType,
    AlarmStatus alarmStatus,
    string textMask,
    int[] groups);
```

Funkcja *ReadActive* służy do pobierania informacji o aktywnych alarmach w aplikacji systemu **asix**.

Pierwsza wersja funkcji *ReadActive* zwraca wszystkie aktywne alarmy. Druga wersja funkcji *ReadActive* umożliwia filtrowanie pobieranych alarmów aktywnych przez podanie typu alarmów, statusu alarmów, maski tekstu alarmów i tablic numerów grup alarmów.

Parametr *alarmType* określa wymagany typ alarmu. Jako jego wartość należy podać jedną ze stałych opisanych w tabeli poniżej lub też sumę bitową tych stałych.

Stała	**Opis**
AlarmType.NoFiltering	Filtrowanie po typie alarmu wyłączone.
AlarmType.System	Alarmy systemowe
AlarmType.Message	Komunikaty
AlarmType.Warning	Ostrzeżenia
AlarmType.Alarm	Alarmy
AlarmType. ImportantAlarmr	Ważne alarmy
AlarmType.All	Suma bitowa wszystkich rodzajów alarmów

Parametr *alarmStatus* określa wymagany status alarmu. Jako jego wartość należy podać jedną ze stałych opisanych w tabeli poniżej lub też sumę bitową tych stałych.

Stała	**Opis**
AlarmStatus. NoFiltering	Filtrowanie po statusie alarmu wyłączone
AlarmStatus.Started	Alarm rozpoczęty
AlarmStatus.Finished	Alarm zakończony
AlarmStatus.Acknowledged	Alarm potwierdzony
AlarmStatus.NotAcknowledged	Alarm niepotwierdzony
AlarmStatus.All	Suma bitowa wszystkich rodzajów statusów alarmów

Jeżeli parametr *alarmsStatus* ma zawierać jedną z flag *Started*, *Finished*, *Acknowledged* lub *NotAcknowledged,* to należ pamiętać, aby zawsze **jednocześnie** użyć przynajmniej jedną z flag *Started* lub *Finished* **i** przynajmniej jedną z flag *Acknowledged* lub *NotAcknowledged*.

Jako parametr *textMask* należy podać maskę tekstu alarmu, czyli fragmentu tekstu, który musi pojawić się w opisie alarmu, aby został on przez funkcję *ReadActive* zwrócony. We wzorcu ma znaczenie wielkość liter. We wzorcu mogą pojawiać się specjalne znaki '*' i '?'.

Znak '*' oznacza, że w jego miejscu w opisie alarmu może pojawić się dowolna liczba znaków.

Znak '?' oznacza, że w jego miejscu w opisie alarmu może pojawić jeden dowolny znak.

Najbardziej typowe wzorce mają postać:
tekst – odpowiada alarmom, których opis zaczyna się od *tekst*,
**tekst* – odpowiada alarmom, których opis zawiera *tekst* w dowolnym miejscu.

Jako parametr *groups* należy podać tablicę numerów grup alarmów. Tablica może liczyć do 10 elementów. Jeżeli jako parametr *groups* podana zostanie tablica pusta lub wartość *null*, to filtrowanie po grupach nie będzie stosowane.

8.1.6. Funkcje ReadHistorical

```
[C#]
public Alarm[] ReadHistorical (
    DateTime periodStart,
    TimeSpan periodLen,
    int maxNumberOfAlarms);
public Alarm[] ReadHistorical (
    DateTime periodStart,
    TimeSpan periodLen,
    int maxNumberOfAlarms,
    AlarmType alarmType,
    AlarmStatus alarmStatus,
    string textMask,
    string idRange,
    int[] groups);
```

Funkcja o nazwie *ReadHistorical* służy do pobierania informacji o historycznych alarmach zarejestrowanych w archiwum alarmów aplikacji systemu **asix**. Część parametrów tych funkcji jest taka sama jak parametry funkcji *ReadActive* opisane w poprzednim punkcie.

Parametry *periodStart* i *periodLen* określające okres czasu, z którego mają być pobrane alarmy.

Parametr *maxNumberOfAlarms* pozwala ograniczyć maksymalną liczbę pobranych alarmów.

Parametr *idRange* pozwala ograniczyć zakres pobieranych alarmów do ściśle określonej numerami. Można zadeklarować listę numerów oddzielonych przecinkiem (np. 2,34,789), zakres numerów alarmów (np. 3-128,300-572) lub połączyć oba sposoby specyfikacji alarmów do wyświetlenia (np. 2,4,5-89).

8.1.7. Funkcja Alarms2DataSet

```
[C#]
public static DataSet Alarms2DataSet(
    Alarm[] alarms,
    bool ascending);
```

Funkcja *Alarms2DataSet* umożliwia utworzenie obiektu klasy *DataSet* z danych przechowywanych w tablicy struktur typu *Alarm*.

Tablica struktur typu *Alarm* zwykle jest wynikiem działania funkcji *ReadActive* lub *ReadHistorical*. Tablicę te należy przekazać jako pierwszy parametr funkcji *Alarms2DataSet*.

Drugi parametr funkcji *Alarms2DataSet* o nazwie *ascending* określa, czy dane o alarmach zostaną wstawione do obiektu *DataSet* posortowane rosnąco czy malejąco. Sortowanie odbywa się po czasie alarmu.

Wynikowy obiekt typu *DataSet* zawiera jedną tabelę o nazwie *ALData*.

8.1.8. Struktura Alarm

Obiekty struktury *Alarm* służą do przekazywania wartości alarmów. Używają ich funkcje *ReadActive* i *ReadHistorical*.

Deklaracja struktury *Alarm* jest następująca:

```
[C#]
public struct Alarm
{
  public int Id;
  public String Text;
  public DateTime TimeStamp;
  public AlarmType AlarmType;
  public AlarmStatus AlarmStatus;
};
```

Po zakończeniu działania funkcji *ReadActive* lub *ReadHistorical* zawartość struktury *Alarm* jest następująca:

Pole	**Zawartość**
Id	Numer alarmu.
Text	Tekst alarmu.
TimeStamp	Stempel czasu alarmu; używany jest czas lokalny.
AlarmType	Typ alarmu.
AlarmStatus	Status alarmu.

8.1.9. Praca w środowisku ASP.NET

W przypadku pracy w środowisku ASP.NET do utworzenia obiektu należy użyć wyrażenie *ServerAL.ServerPool.Get()*. Obiekt *ServerPool* jest polem statycznym klasy *ServerAL* i implementuje pulę serwerów danych bieżących. Za pomocą funkcji *Get* należy każdorazowo na początku generowania strony pobrać obiekt *ServerAL*. Deklaracja funkcji *Get* jest następująca:

```
[C#]
public ServerAL Get ();
```

Po zakończeniu generowania strony serwer musi być zwrócony do puli przy pomocy wyrażenia *ServerAL.ServerPool.Release()*. Jako parametr funkcji *Release* należy przekazać zwracany do puli serwer. Deklaracja funkcji *Release* jest następująca:

```
[C#]
public Release (ServerAL server);
```

Serwer można pobierać z puli również na początku każdej funkcji i zwracać na końcu funkcji. Aby być pewnym, że każdy pobrany serwer wróci do puli, należy główny kod funkcji zawrzeć w bloku *try* i zwracać serwer do puli w bloku *finally*.

```
private void Page_Load(object sender, System.EventArgs e)
{
    ServerAL server = null;

    try
    {
        server = ServerAL.ServerPool.Get();

        // kod funkcji
    }
    catch(Exception e)
    {
        // obługa wyjątków zgłoszonych przy pobierania serwera z puli i
        // podczas działania funkcji
    }
    finally
    {
        if (server != null)
            ServerAL.ServerPool.Release(server);
    }
}
```

Pula serwerów*:*

- tworzy dla aplikacji kilka obiektów klasy *ServerAL* (nazwa kanału pobierana jest z pliku *Web.Config* patrz Sposób specyfikowania nazwy *kanału*);
- przechowuje obiekty w pamięci podręcznej aplikacji ASP.NET;
- udostępnia obiekty dla kolejnych wywołań w ramach aplikacji ASP.NET;
- zgłasza wyjątek *PoolApplicationExeption*, gdy pula serwerów osiągnęła maksymalny rozmiar i w puli nie ma żadnego wolnego serwer.

9. Raporty

9.1. Serwer .NET

9.1.1. Użycie serwera

Klasa *ServerRP* umożliwia dostęp do raportów wygenerowanych w systemie **asix**. Przed użyciem serwera .NET należy wykonać poniższe kroki.

- Zainstalować pakiet AsixConnect.
- Za pomocą programu *Konfigurator* skonfigurować kanał podstawowy lub założyć i skonfigurować własny kanał.
- Wygenerować projekt w pakiecie Visual Studio i następnie:
  - podświetlić w drzewie projektu folder *References*, z menu *Project* wybrać polecenie *Add Reference*, nacisnąć przycisk *Browse* i z podkatalogu *c:\asix* wybrać plik *XConnectNet.dll*, nacisnąć przycisk *OK* i zamknąć okno *Add Reference*; w każdym pliku z kodem źródłowym C# należy w regionie deklaracji *using* dodać linię:

    ```
    using XConnectNet;
    ```

- napisać program operujący na serwerze *ServerRP*. W programie tym należy:
  - utworzyć obiekt typu *ServerRP*, podając jako parametr nazwę wcześniej skonfigurowanego kanału;
  - używając funkcji serwera pobrać informacje o raportach i same raporty;
  - podczas pobierania informacji należy pamiętać o obsłudze wyjątków, ponieważ mogą one być zgłaszane przez serwer.

9.1.2. Konfiguracja plików definicji raportów

Dla poprawnej pracy raportera należy skonfigurować pliki definicji raportów. Polega to na dodaniu do tych plików wiersza informacyjnego zawierającego nazwę grupy raportów oraz długość okresu czasu generowania raportu. Informacje te są wykorzystywane przez funkcje pobierające informacje o raportach i plikach definicji (opisane poniżej).

Składnia wiersza informacyjnego wygląda następująco:

comment_character REPORT_INFO=GROUP:Name; PERIOD:Length

gdzie:
comment_character – znak komentarza. W zależności od formatu pliku definicji należy użyć odpowiedniego znaku komentarza zgodnie z tabelą:

Typ pliku (rozszerzenie)	**Znak komentarza**
Plik definicji raportu (*.r)	/
Skrypt VBScript (*.vbs)	‘ lub //
Skrypt JavaScript (*.js)	//

Name – nazwa grupy raportów (odpowiada węzłom drzewa)
Length – długość okresu czasu generowania raportu (odpowiada „liściom” drzewa).
Dostępne wartości to:

- D - dzień
- W - tydzień
- M - miesiąc
- Q - kwartał
- Y - rok
- O – inny

PRZYKŁAD

```
/ REPORT_INFO = GROUP:Zduny; PERIOD:D
```

9.1.3. Konstruktor ServerRP

```
[C#]
public ServerRP(
  string channelName);
```

Funkcja służy do utworzenia i inicjalizacji obiektu klasy *ServerRP*.

Jako parametr *channelName* należy podać nazwę kanału (*patrz: punkt 3. Konfiguracja połączeń)*.

9.1.4. Funkcja Dipsose

```
[C#]
public void Dispose();
```

Funkcja służy do zwolnienia zasobów używanych przez obiekt *ServerRP*. Funkcja musi być wywołana po zakończeniu używania obiektu *ServerRP*. Wywołanie musi nastąpić z tego samego wątku, w którym utworzono obiekt.

9.1.5. Funkcja Init

```
[C#]
public void Init(
    string initString);
```

Funkcja służy do ustawiania parametrów serwera. Opis funkcji znajduje się w punkcie *Konfiguracja programowa* (*patrz: punkt 3.5.*) , a dostępne opcje w następnych punktach.

9.1.6. Funkcja GetReportsInfo

```
[C#]
public ReportInfoNet[] GetReportsInfo(
    string group,
    int period,
    bool lastReportsOnly);
```

Funkcja *GetReportsInfo* służy do pobierania informacji o raportach wygenerowanych w aplikacji systemu **asix**. Wynik działania funkcji zwracany jest w strukturze *ReportInfoNet*.

Parametr *group* określa grupę, z której pochodzić będą czytane raporty. Podanie w tym miejscu wartości „*” spowoduje odczytanie wszystkich raportów (niezależnie od wartości parametru *period*).

Parametr *period* określa okres raportu. Dozwolone są następujące wartości określające okres czasu generacji raportu:

Wartość	Opis
-1	Dowolne raporty
0	Raporty dobowe
1	Raporty tygodniowe
2	Raporty miesięczne
3	Raporty kwartalne
4	Raporty roczne
5	Raporty o niezdefiniowanym okresie

Parametr *lastReportsOnly* określa czy w przypadku gdy istnieje kilka raportów dotyczących tego samego okresu, mają zostać zwrócone wszystkie raporty czy tylko ostatni (najbardziej aktualny).

9.1.7. Funkcja ReadReportsInfo

```
[C#]
public DataSet ReadReportsInfo(
    string group,
    int period,
    bool lastReportsOnly);
```

Działanie funkcji *ReadReportsInfo* jest identyczne jak funkcji *GetReportsInfo*. Różnica polega na tym, że informacje o raportach zwracane są w obiekcie klasy *DataSet*.

Wynikowy obiekt typu *DataSet* zawiera jedną tabelę o nazwie *ReportsInfo*.

9.1.8. Funkcja GetDefFilesInfo

```
[C#]
public DefFileInfoNet[] GetDefFileInfo();
```

Funkcja *GetDefFileInfo* służy do pobierania informacji o wszystkich plikach definicji raportów. Wynik działania funkcji zwracany jest w strukturze *DefFileInfoNet*.

9.1.9. Funkcja ReadDefFilesInfo

```
[C#]
public DefFileInfoNet[] GetDefFileInfo();
```

Działanie funkcji *ReadDefFilesInfo* jest identyczne jak funkcji *GetDefFilesInfo*. Różnica polega na tym, że informacje o raportach zwracane są w obiekcie klasy *DataSet*.

Wynikowy obiekt typu *DataSet* zawiera jedną tabelę o nazwie *DefFilesInfo*.

9.1.10. Funkcja GetReportsDirectoryPath

```
[C#]
public string GetReportsDirectoryPath();
```

Funkcja zwraca ścieżkę do katalogu raportów aplikacji.

9.1.11. Struktura ReportInfoNet

Struktura *ReportInfoNet* służy do przechowywania informacji o raportach.

Deklaracja struktury jest następująca:

```
public struct ReportInfoNet
{
    String    FileName;
    String    ReportName;
    DateTime  ReportDate;
    DateTime  FileDate;
    long      FileSize;
    int       ReportFormat;
    int       ReportType;
    int       ReportPeriod;
    String    ReportGroup;
};
```

Po zakończeniu działania funkcji czytającej informacje o raportach, struktura ma następującą zawartość:

Pole	**Zawartość**
FileName	Nazwa pliku raportu
ReportName	Nazwa raportu
ReportDate	Data raportu
FileDate	Data utworzenia pliku raportu
FileSize	Rozmiar pliku
ReportFormat	Format pliku raportu (TXT/HTML)
ReportType	Typ raportu (skryptowy/dyskowy)
ReportPeriod	Okres raportu (dostępne wartości opisano w pkt. 9.1.6)
ReportGroup	Grupa raportu

9.1.12. Struktura DefFileInfoNet

Struktura *DefFileInfoNet* służą do przechowywania informacji o plikach definicji raportów. Deklaracja struktury jest następująca:

```
public struct DefFileInfoNet
{
    String    FileName;
    int       ReportPeriod;
    String    ReportGroup;
};
```

Struktura przechowuje informacje o nazwie, okresie i grupie pliku definicji raportów. Dostępne wartości okresu raportu (*ReportPeriod*) opisane są w punkcie 9.1.6.

9.1.13. Praca w środowisku ASP.NET

W przypadku pracy w środowisku ASP.NET do utworzenia obiektu należy użyć wyrażenie *ServerRP.ServerPool.Get()*. Obiekt *ServerPool* jest polem statycznym klasy *ServerRP* i implementuje pulę serwerów danych bieżących. Za pomocą funkcji *Get* należy każdorazowo

na początku generowania strony pobrać obiekt *ServerRP*. Deklaracja funkcji *Get* jest następująca:

```
[C#]
public ServerRP Get ();
```

Po zakończeniu generowania strony serwer musi być zwrócony do puli przy pomocy wyrażenia *ServerRP.ServerPool.Release()*. Jako parametr funkcji *Release* należy przekazać zwracany do puli serwer. Deklaracja funkcji *Release* jest następująca:

```
[C#]
public Release (ServerRP server);
```

Serwer można pobierać z puli również na początku każdej funkcji i zwracać na końcu funkcji. Aby być pewnym, że każdy pobrany serwer wróci do puli, należy główny kod funkcji zawrzeć w bloku *try* i zwracać serwer do puli w bloku *finally*.

```
private void Page_Load(object sender, System.EventArgs e)
{
    ServerRP server = null;

    try
    {
        server = ServerRP.ServerPool.Get();

        // kod funkcji
    }
    catch(Exception e)
    {
        // obługa wyjątków zgłoszonych przy pobierania serwera z puli i
        // podczas działania funkcji
    }
    finally
    {
        if (server != null)
            ServerRP.ServerPool.Release(server);
    }
}
```

Pula serwerów*:*

- tworzy dla aplikacji kilka obiektów klasy *ServerRP(*nazwa kanału pobierana jest z pliku *Web.Config* patrz 3.2. *Sposób specyfikowania nazwy kanału*);
- przechowuje obiekty w pamięci podręcznej aplikacji ASP.NET;
- udostępnia obiekty dla kolejnych wywołań w ramach aplikacji ASP.NET;
- zgłasza wyjątek *PoolApplicationExeption*, gdy pula serwerów osiągnęła maksymalny rozmiar i w puli nie ma żadnego wolnego serwer.

10. Serwer Web Service

10.1. Serwer Web Service

Serwer *Web Service* pakietu AsixConnect umożliwia dostęp do pełnej funkcjonalności aplikacji systemu **asix** za pośrednictwem protokołu *XML Web Services*.

10.2. Instalacja

Serwer *WebService* znajduje się w katalogu *c:\asix\WebService*. Aby udostępnić go w sieci, należy uruchomić program *Internetowe usługi informacyjne*. Program znajduje się w menu *Start/Panel sterowania/Narzędzia administracyjne*. W oknie programu należy podświetlić element *Domyślna witryna sieci Web*, z menu *Akcja/Nowy* wybrać polecenie *Katalog wirtualny*. Uruchamia się wizard, w którym jako *Alias* należy podać tekst *WebService*, a jako *Katalog c:\asix\WebService*. Pozostałe opcje należy pozostawić bez zmian. Domyślnie przy próbach dostępu do serwera używana jest autentyfikacja Windows. Aby włączyć dostęp anonimowy, należy podświetlić nowo utworzony katalog wirtualny, z menu *Akcja* wybrać polecenie *Właściwości*, wybrać zakładkę *Zabezpieczenia katalogów*, w polu *Dostęp anonimowy* nacisnąć przycisk *Edytuj* i włączyć opcję *Dostęp anonimowy*.

Od tego momentu serwer Web Service jest dostępny pod adresem http://localhost/WebService/XConnectWebService.asmx.

Pod adresem http:// localhost /WebService/XConnectWebService.asmx?WSDL dostępny jest opis usług serwera w języku WSDL.

Serwer Web Service udostępnia dane tylko lokalnie. Aby móc udostępniać dane w sieci lokalnej lub w sieci Internet, należy dokupić licencję *asix4Internet*.

10.3. Plik konfiguracyjny Web.Config

Serwer Web Service używa pliku konfiguracyjnego o nazwie *Web.Config* do przechowywania domyślnej nazwy kanału. Plik ten znajduje się w katalogu c*:\asix\WebService*. Sposób definiowania kanałów opisany jest w rozdziale *Kanały (patrz: punkt 3.1.)*.

Aby określić domyślną nazwę kanału, należy w pliku *Web.Config*, w elemencie nadrzędnym *configuration* utworzyć element *appSettings*. Następnie w elemencie *appSettings* należy utworzyć jeden element *add* i zdefiniować w nim dwa atrybuty. Pierwszy atrybut należy nazwać *key* i nadać mu wartość *"DefaultChannelName"*. Drugi atrybut należy nazwać *value* i nadać mu jako wartość nazwę kanału. Nazwę kanału należy ująć w cudzysłowy.

PRZYKŁAD:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="DefaultChannelName" value="AsEmis" />
  </appSettings>
...
```

Jeżeli serwer *WebService* będzie dostarczał dane dla dynamicznych stron HTML, to w kanale należy włączyć opcje *Zmienna kontrolna* i *Limity zmiennych.*

10.4.Klienci

10.4.1. Internet Explorer

Najprostszym sposobem przetestowania prawidłowości działania serwera Web Service jest użycie przeglądarki Internet Explorer jako klienta serwera. Po skonfigurowaniu serwera należy uruchomić przeglądarkę i otworzyć stronę:
http://localhost/WebService/XConnectWebService.asmx.

Załadowana strona zawiera listę wszystkich funkcji udostępnianych przez serwer. Po wybraniu funkcji pojawiają się informacje o tej funkcji. Ponadto dla części funkcji wyświetlane są pola, gdzie można podać parametry funkcji oraz przycisk *Wywołaj*.

Przy użyciu przeglądarki Internet Explorer można wywołać tylko te funkcje, które w parametrach i wynikach używają wyłącznie typów prostych. Po wybraniu funkcji, podaniu parametrów i naciśnięciu przycisku *Wywołaj* otwierana jest nowa strona zawierająca wynik działania funkcji. Wynik podawany jest w formacie XML. Funkcje, które można wywołać z przeglądarki to: *Read, ReadRaw*, *ReadProcessed*, *ReadActiveAlarms* i *ReadHistoricalAlarms*. Dla parametrów, które wymagają podania daty, należy użyć formatu *yyyy/mm/ddThh:mm:ss* (na przykład *2004/07/13T01:00:00)*.

10.4.2. Aplikacja WebForm

Aplikacji WebForm Visual Studio 2003, która ma korzystać z serwera *WebSerivice* musi zawierać przede wszystkim odwołanie do tego serwera. W tym celu po wygenerowaniu projektu należy:

- podświetlić w drzewie projektu folder *References;*
- z menu *Project* wybrać polecenie *Add Web Reference*;
- wpisać adres URL, pod którym serwer *WebService* jest dostępny, czyli *http://localhost/WebService/XConnectWebService.asmx* i nacisnąć przycisk *Go*;
- podać jako nazwę referencji *Asix* i nacisnąć przycisk *Add Reference*.

Serwer *WebService* wymaga, aby klient obsługiwał mechanizm *cookie*. W tym celu, po utworzeniu w aplikacji obiektu serwera typu *Asix.XConnectWebService*, należy przypisać do jego właściwości *CookieContainer* obiekt typu *System.Net.CookieContainer*.

```
Asix.XConnectWebService serwer;
serwer = new Asix.XConnectWebService();
serwer.CookieContainer = new System.Net.CookieContainer();
```

Obiekt klasy *Asix.XConnectWebService* zawiera wszystkie funkcje serwera *WebService*. Poniższy przykład przedstawia operację odczytu z serwera wartości bieżącej zmiennej KW_A110.

```
Asix.ItemStateWS itemStateWS = server.Read("KW_A110");
if (itemStateWS.ReadResult >= 0)
{
    // odczyt powiódł się
    // wartość
    textBox1.Text = itemStateWS.ItemValues[0].ToString ();
    // jakość
    textBox2.Text = "0x" + System.Convert.ToString (itemStateWS.Quality, 16).PadLeft
(4,'0');
    // stempel czasu
    textBox3.Text = itemStateWS.TimeStamp.ToString();
}
else
{
    // odczyt zakończył się błędem
```

```
    textBox1.Text = itemStateWS.ErrorString;
    textBox2.Text = textBox3.Text = "";
}
```

10.5.Baza zmiennych

```
[C#]
string[] ReadAttributes(
    string varName,
    string[] attributeNames);
string[][] ReadAttributesN(
    string[] varNames,
    string[] attributeNames);
```

Funkcja *ReadAttributes* służy do czytania wartości atrybutów zmiennej. Jej działanie jest takie same jak funkcji *ReadAttributes* klasy *ServerHT* (*patrz: punkt 4.3.1.5. Funkcja ReadAttributes*).

Funkcja *ReadAttributesN* służy do czytania wartości atrybutów wielu zmiennych. Jej działanie jest takie samo jak funkcji *ReadAttributesN* klasy *ServerHT* (*patrz: punkt 4.3.1.6. Funkcja ReadAttributesN*).

10.6.Dane bieżące

```
[C#]
ItemStateWS Read(
    string   itemID);
ItemStateWS[] ReadN(
    string[] itemIDs);
```

Funkcje *Read* i *ReadN* służą do czytania bieżących wartości zmiennych procesowych.

Funkcja *Read* jako parametr posiada nazwę zmiennej i zwraca wartość tej zmiennej w postaci struktury *ItemStateWS*.

Funkcja *ReadN* jako parametr posiada tablicę nazw zmiennej, a zwraca wartość tych zmiennych w postaci tablicy struktur *ItemStateWS*.

Struktura ItemStateWS

Obiekty typu *ItemStateWS* służą do przekazywania wartości zmiennych. Struktura *ItemStateWS* używana jest przez funkcje *Read* i *ReadN*.

Deklaracja struktury *ItemStateWS* jest następująca:

```
[C#]
public struct ItemStateWS
{
  public String ItemID;
  public Int32 ReadResult;
  public bool ReadSucceeded;
  public String ErrorString;

  public DateTime TimeStamp;
  public Int32 Quality;
  public Object ItemValues;
};
```

Po zakończeniu działania funkcji *Read* zawartość struktury przyjmuje postać opisaną w kolejno prezentowanej tabeli.

Pole	**Zawartość**
ItemID	Nazwa zmiennej.
ReadResult	Wynik czytania. Wartość ujemna oznacza błąd i jest to jednocześnie kod błędu. Wartość 0 lub dodatnia oznacza, że odczyt zakończył się sukcesem. Pola *TimeStamp*, *Quality* i *ItemValues* są wtedy wypełnione.
ReadSucceeded	Pole zawiera wartość *true*, jeżeli wartość pola *ReadResult* wskazuje, że odczyt zakończył się sukcesem.
ErrorString	Pole zawiera tekstowy opis kodu błędu zawartego w polu *ReadResult*.
TimeStamp	Stempel czasu pomiaru; używany jest czas lokalny.
Quality	Jakość pomiaru.
ItemValues	Wartości pomiaru. Pole jest typu *object* i zawiera tablicę wartości typu odpowiadającego przeczytanej zmiennej. Najczęściej jest to tablica jednoelementowa. Tylko w wypadku tablic lub czytania zmiennej zawierającej w identyfikatorze frazę *BarCVs* pole zawiera tablicę wieloelementową.

10.7. Dane archiwalne surowe

```
[C#]
ReadRawResult ReadRaw(
    string itemID,
    DateTime periodStart,
    int periodLenS);
```

Funkcja o nazwie *ReadRaw* służy do czytania surowych wartości archiwalnych zmiennych procesowych z podanego okresu czasu. Jej działania jest takie same jak funkcji *ReadRaw* obiektu *ServerHT* opisanej w rozdziale *Funkcje ReadRaw (patrz: punkt 7.7.5.)*.

10.8. Dane archiwalne agregowane

```
[C#]
ReadProcessedResult ReadProcessed(
    string itemID,
    Aggregate itemAggregate,
    DateTime periodStart,
    int periodLenS,
    int resampleIntervalS);
ReadProcessedResult [] ReadProcessedN(
    string[] itemIDs,
    Aggregate[] itemAggregates,
    DateTime periodStart,
    int periodLenS,
    int resampleIntervalS);
```

Funkcja *ReadProcessed* oblicza agregaty w podanym okresie czasu dla podanej zmiennej procesowej. Jej działanie jest takie samo jak funkcji *ReadProcessed* obiektu *ServerHT* (*patrz: punkt 7.7.6. Funkcje ReadProcessed*). Inny jest sposób podawania długości okresu i długości interwału – wartości te należy podawać w sekundach.

Funkcja *ReadProcessedN* działa tak samo jak funkcja *ReadProcessed*, ale umożliwia przy jednym wywołaniu pobranie danych dotyczących wielu zmiennych i ich agregatów. Przykładowo, aby jednym wywołaniem pobrać wartości minimalne, maksymalne i średnie

zmiennej, należy jako pierwszy parametr przesłać tablicę zawierającą trzy razy tą samą nazwę zmiennej, a jako drugi parametr tablicę zawierającą identyfikatory odpowiednich agregatów.

Aby uzyskać maksymalną wydajność przy odczycie dwu lub więcej agregatów tej samej zmiennej, należy grupować wartości parametru *itemIDs* wg nazwy zmiennych.

10.9. Alarmy aktywne

```
[C#]
Alarm[] ReadActiveAlarms();
Alarm[] ReadActiveAlarmsEx(
    Severity severity,
    Status status,
    string textMask,
    int[] groups);
```

Funkcje o nazwie *ReadActiveAlarms* i *ReadActiveAlarmsEx* służą do pobierania informacji o aktywnych alarmach w aplikacji systemu **asix**. Działania funkcje *ReadActiveAlarms* jest takie samo jak bezparametrowej funkcji *ReadActive* klasy *ServerAL*.

Działanie funkcji *ReadActiveAlarmsEx* jest takie samo jak 4-parametrowej funkcji *ReadActive* klasy *ServerAL.*

10.10. Alarmy historyczne

```
[C#]
Alarm[] ReadHistoricalAlarms(
    DateTime periodStart,
    int periodLenS,
    int maxNumberOfAlarms);
Alarm[] ReadHistoricalAlarmsEx(
    DateTime periodStart,
    int periodLenS,
    int maxNumberOfAlarms,
    Severity severity,
    Status status,
    string textMask,
    string idRange,
    int[] groups);
```

Funkcje o nazwie *ReadHistoricalAlarms* i *ReadHistoricalAlarmsEx* służą do pobierania informacji o historycznych alarmach w aplikacji systemu **asix**.

Działanie funkcji *ReadHistoricalAlarms* jest takie samo jak 3-parametrowej funkcji *ReadHistorical* klasy *ServerAL.* Z kolei działanie funkcji *ReadHistoricalAlarmsEx* jest takie samo jak 8-parametrowej funkcji *ReadHistorical* klasy *ServerAL* (*patrz: punkt 8.1.6. Funkcje ReadHistorical*). Inny jest sposób podawania długości okresu – wartość tę należy podawać w sekundach.

11. Diagnozowanie pracy serwerów

11.1. Logi

Podczas pracy każdy serwer wpisuje do pliku logu informacje o rozpoczęciu i zakończeniu pracy oraz o poważnych błędach napotkanych podczas pracy. Log ten ma nazwę:

```
<identyfikator>.<bieżąca data>.log
```

Jako *identyfikator* występuje:

Rodzaj serwera	Identyfikator
Wszystkie serwery Automation	*XConnect*
DDE danych bieżących	*ServerCTDDE*
DDE danych bieżących – usługa	*ServiceCTDDE*
OPC danych bieżących	*ServerCTOPC*
Wszystkie serwery .NET, serwer WebService	*XConnectNet*
OLEDB danych archiwalnych	*ServerHTOLEDB*

Jako <*bieżąca data*> używana jest data utworzenia logu w formacie *RRRRMMDD*. Jeżeli plik logu przekroczy danego dnia rozmiar 10MB, to tego dnia o północy plik logu zostanie zamknięty i zostanie utworzony nowy plik logu z nową datą. Jeżeli na dysku będzie mało miejsca, to stare pliki logów będą automatycznie usuwane.

PRZYKŁAD nazwy logu utworzonego 28 sierpnia 2001 przez serwer OPC danych bieżących:

```
ServerCTOPC.20010828.log
```

Logi domyślnie znajdują się w podkatalogu *Log* katalogu, w którym zainstalowano pakiet AsixConnect, czyli najczęściej c*:\asix\Log*. Katalog, w którym umieszczane są pliki logów, można zmienić używając programu *Konfigurator (patrz: punkt 3.4.3. Opcje pakietu*).

11.2. Kody błędów

Każda z funkcji udostępnianych przez serwer OPC pakietu AsixConnect zwraca 0, jeżeli jej wykonanie zakończyło się pomyślnie. Jeżeli wykonanie zakończyło się pomyślnie, ale istnieją dodatkowe informacje na temat wykonania funkcji, to zwracana jest wartość dodatnia. W przypadku wystąpienia błędu zwracana jest wartość ujemna.

Kody błędów, jakie może zwrócić serwer OPC podane są w poniższej tabeli.

Nazwa kodu błędu	Kod błędu	Opis
ASKOM_E_NetInit	0xC0048100L	Błąd przy inicjalizacji sieci systemu asix
ASKOM_E_NetInitTimeout	0xC0048101L	Przekroczony czas oczekiwania przy inicjalizacji sieci systemu

		asix
ASKOM_E_NetInstallClient	0xC0048102L	Błąd przy inicjalizacji klienta sieci systemu asix
ASKOM_E_NetDeinstallClient	0xC0048103L	Błąd przy deinicjalizacji klienta sieci systemu asix
ASKOM_E_NetFindServer	0xC0048104L	Błąd przy wywołaniu funkcji Wyszukiwanie serwera danych
ASKOM_E_NetFindServer_NotFound	0xC0048105L	Nie znaleziono serwera danych w sieci systemu asix
ASKOM_E_NetLinkToServer	0xC0048106L	Błąd przy dołączaniu do serwera danych systemu asix
ASKOM_E_NetLinkToServerTimeout	0xC0048107L	Przekroczony czas oczekiwania przy dołączaniu do serwera danych systemu asix
ASKOM_E_NetLinkToServerNegativeAnswer	0xC0048108L	Błąd przy dołączaniu do serwera danych systemu asix – odpowiedź negatywna
ASKOM_E_NetCloseLink	0xC0048109L	Błąd przy zamykaniu połączenia z serwerem danych systemu asix.
ASKOM_E_NetSendData	0xC004810aL	Błąd przy wysyłaniu danych w sieci systemu asix
ASKOM_E_NetAnswerTimeout	0xC004810bL	Przekroczony czas oczekiwania na odpowiedź w sieci systemu asix
ASKOM_E_ItemUnknown	0xC0048120L	Błąd - zmienna nieznana w systemie asix.
ASKOM_E_TheSameItemWrittenMoreThanOnce	0xC0048121L	Błędne parametry przy zapisie - ta sama zmienna systemu asix zapisywana więcej niż raz.
ASKOM_E_WriteError	0xC0048128L	System asix zwrócił błąd przy próbie zapisu.
ASKOM_E_WriteError_NetworkServerNotFound	0xC0048129L	System asix zwrócił błąd przy próbie zapisu - serwer obsługujący zmienną nie znaleziony.
ASKOM_E_WriteError_IllegalCommand	0xC004812aL	System asix zwrócił błąd przy próbie zapisu - urządzenie uznało komendę za niedozwoloną.
ASKOM_E_WriteError_DriverTimeout	0xC004812bL	System asix zwrócił błąd przy próbie zapisu - timeout przy zapisie do urządzenia.
ASKOM_E_WriteError_DriverTransmisionError	0xC004812cL	System asix zwrócił błąd przy próbie zapisu - błąd transmisji podczas komunikacji z urządzeniem.
ASKOM_E_WriteError_DeviceRequestIllegal	0xC004812dL	System asix zwrócił błąd przy próbie zapisu - urządzenie uznało polecenie za niedozwolone.
ASKOM_E_GroupItemsCantBeUsed	0xC0048139L	Funkcja nie obsługuje identyfikatorów grup zmiennych.
ASKOM_E_ErrorInitializingHaspKey	0xC0048047L	Nieudana inicjalizacja klucza HASP.
ASKOM_E_HaspNotFound	0xC0048048L	Nie znaleziono klucza HASP.
ASKOM_E_HaspConstructorExpired	0xC004804AL	Czas pracy w trybie konstruktora upłynął
ASKOM_E_VarBaseNotFound	0xC004a001L	W podanym katalogu nie znaleziono bazy zmiennych
ASKOM_E_ErrorDuringOpeningVarBase	0xC004a006L	Błąd podczas otwierania bazy zmiennych

ASKOM_E_ErrorDuringOpeningVarBase_SharedMemLocation	0xC004a008L	Błąd podczas otwierania bazy zmiennych - nieprawidłowy parametr BDE SharedMemLocation
ASKOM_S_DataTransmisionInterruptedByUser	0x00048140L	Transmisja danych przerwana przez użytkownika.
ASKOM_E_ErrorDuringReceivingArchiveVarInformation	0xC0048141L	Błąd przy pobieraniu informacji o zmiennej archiwalnej.
ASKOM_E_ErrorDuringOpeningArchiveVar	0xC0048142L	Błąd przy otwieraniu zmiennej archiwalnej.
ASKOM_E_ErrorDuringClosingArchiveVar	0xC0048143L	Błąd przy zamykaniu zmiennej archiwalnej.
ASKOM_E_ErrorDuringSeekingArchiveVarData	0xC0048144L	Błąd przy wyszukiwaniu danych zmiennej archiwalnej.
ASKOM_E_ErrorDuringReadingArchiveVarData	0xC004815L	Błąd przy czytaniu zmiennej archiwalnej.
ASKOM_E_ErrorDuringConfirmingReadingArchiveVarData	0xC0048146L	Błąd potwierdzenia przy czytaniu zmiennej archiwalnej

11.3. Serwer DDE

Jeżeli skrypt napisany w języku *VisualBasic*, wykonuje odczyt z serwera DDE, który został nagle zamknięty, to funkcja programu Excel zwraca błąd 2023.

12. Przykłady

12.1. Przykłady

Po zainstalowaniu programu w katalogu docelowym (domyślnie c:*asix*) w podkatalogu AC4Przyklady znajdują się przykłady wykorzystania pakietu AsixConnect.

Wszystkie przykłady wykorzystują aplikację demonstracyjną *Wytwornia_Kwasu* instalowaną w katalogu *c:\Asix\Aplikacje\Wytwornia_Kwasu*. Można uruchomić tę aplikację wybierając z menu *Start/Programy/asix/Demo* program *Wytwornia Kwasu*. Aplikację należy uruchomić przed załadowaniem przykładów.

Przykłady użycia serwerów Automation, DDE i OLE DB zapisane są w arkuszach kalkulacyjnych w formacie programu Excel 97/2002. Większość przykładów zawiera krótkie makra napisane w języku Visual Basic. Aby móc obejrzeć makra, należy po załadowaniu arkusza kalkulacyjnego wybrać z menu *Narzędzia/Makro* polecenie *Edytor Visual Basic* lub nacisnąć kombinację klawiszy *Alt-F11*. Przykłady korzystają z kanałów pakietu AsixConnect o nazwach *Demo_Wytwornia_Kwasu* i *Demo_Wytwornia_KwasuDDE4*. Kanały te są tworzone podczas instalacji pakietu **asix**.

12.2. Serwer Automation - dane bieżące i baza zmiennych

Przykład ten demonstruje sposób dostępu do danych bieżących i bazy zmiennych aplikacji systemu **asix** przy użyciu serwera Automation.

W arkuszu tym ustanowiono powiązanie z biblioteką serwera Automation, dzięki czemu makra mogą bezpośrednio korzystać z obiektów serwera Automation pakietu AsixConnect. Oprócz łatwości programowania uzyskuje się również maksymalną wydajność komunikacji. Aby sprawdzić, jakie są aktualnie ustanowione powiązania, należy z menu *Tools* edytora Visual Basic wybrać polecenie *References*. Biblioteka serwera Automation danych bieżących nosi nazwę *AsixConnect 4 Type Library*.

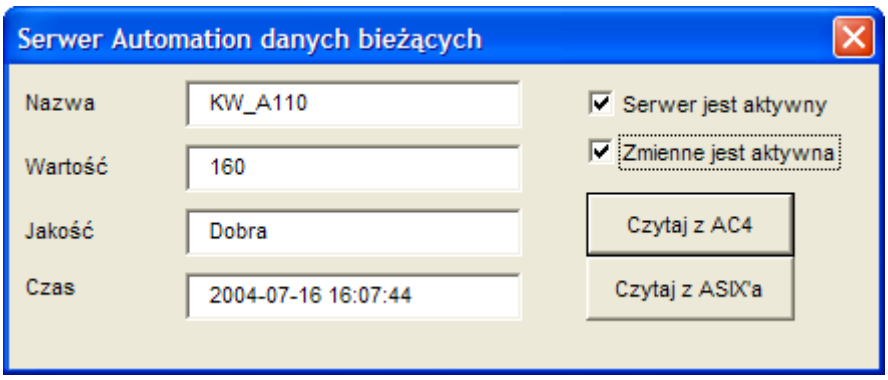
W zakładce *ReadWrite* arkusza znajduje się tabelka zawierająca nazwy czterech zmiennych, ich wartości, statusy i stemple czasu. Ponadto znajdują się tam przyciski *Czytaj*, *Zapisz*, *Wybierz nazwę* i *Pokaż opis*.

Po naciśnięciu przycisku *Czytaj* odpowiadającego danej zmiennej wywoływane jest makro. Makro to, za pośrednictwem serwera Automation danych bieżących, pobiera wartość zmiennej i wpisuje ją do tabeli. Jeżeli wpisze się do kolumny *Wartość* liczbę i naciśnie przycisk *Zapisz,* to liczba ta zostanie przesłana do systemu **asix** jako nowa wartość zmiennej.

UWAGA:
Przy zapisie należy sprawdzić czy spełnione są warunki umożliwiające wykonywanie zapisów do aplikacji systemu **asix** opisane w punkcie *Zapis prosty (patrz: punkt 6.3.1.)*.

Po naciśnięciu przycisku *Wybierz nazwę* wywoływane jest makro, które za pośrednictwem serwera Automation bazy zmiennych wyświetla okno wyboru zmiennej. Jeżeli użytkownik wybierze nazwę zmiennej i zamknie okno naciskając przycisk OK., to wybrana nazwa pojawi się w kolumnie *Nazwa zmiennej*. Po naciśnięciu przycisku *Pokaż opis* pojawia się okno wyświetlające opis zmiennej pobrany z bazy zmiennej.

W zakładce *DataChange* znajduje się przykład przesyłania przez serwer Automation danych bieżących do klienta. Naciśnięcie przycisku *Start* powoduje wyświetlenie poniższego okna dialogowego.



Jeżeli opcja *Zmienna* jest aktywna, to zmienna zaczyna być przesyłana z aplikacji systemu **asix** do serwera Automation. Można tę zmienną czytać naciskając przycisk *Czytaj z AC4*. Jeżeli opcja nie jest zaznaczona, to działa tylko czytanie bezpośrednio z systemu **asix** (przycisk *Czytaj z asix'a*). Jeżeli dodatkowo włączona jest opcja *Serwer jest aktywny,* to zmienna jest przesyłana również z serwera Automation do aplikacji Visual Basic. Wywoływane zdarzenie (procedura *asix_DataChange*) wypisuje przysłaną wartość zmiennej w oknie dialogowym.

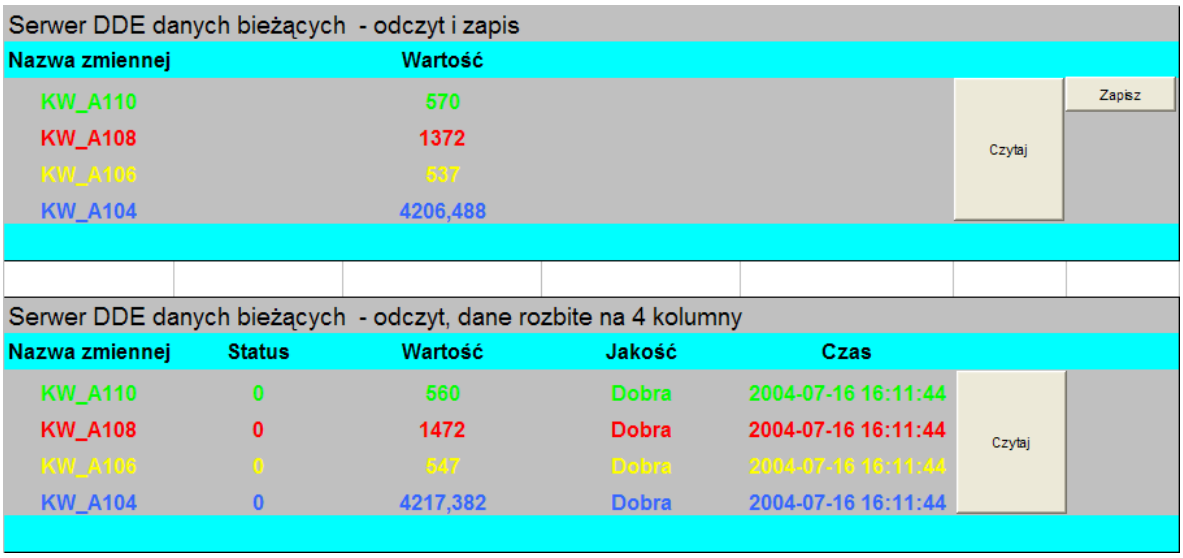
12.3. Serwer Automation - dane bieżące - późne wiązanie

Przykład ten demonstruje sposób dostępu do danych bieżących aplikacji systemu **asix** przy użyciu serwera Automation. Przykład ten jest funkcjonalnie analogiczny do przykładu *Serwer Automation - dane bieżące* jednak nie wykorzystano w nim wczesnego wiązania z serwerem Automation przez ustanowione powiązanie z biblioteką typów serwera Automation. W przykładzie do utworzenia obiektu Automation użyto funkcji *CreateObject* zamiast operatora *New*. Ten sposób tworzenia obiektu jest jedynym dostępnym w języku *VBScript*.

12.4. Serwer DDE - dane bieżące

Przykład ten demonstruje sposób dostępu do danych bieżących aplikacji systemu **asix** przy użyciu serwera DDE. Aby przykład działał, należy wcześniej uruchomić serwer DDE.

W zakładce *ReadWrite* arkusza znajdują się dwie tabelki. Pierwsza zawiera nazwy czterech zmiennych i ich wartości oraz przyciski *Czytaj* i *Zapisz*. Druga zawiera dodatkowe kolumny wyświetlające statusy i stemple czasu.



Po naciśnięciu przycisku *Czytaj* z pierwszej tabeli wywoływane jest makro. Makro to za pośrednictwem serwera Automation danych bieżących pobiera wartości wszystkich zmiennych i wpisuje je do tabeli. Dane pobierane są w formacie jednokolumnowym. Jeżeli wpisze się liczbę do kolumny *Wartość* w wierszu zmiennej *KW_A110* i naciśnie przycisk *Zapisz,* to liczba ta zostanie przesłana do systemu **asix** jako nowa wartość zmiennej.

UWAGA:
Przy zapisie należy sprawdzić, czy spełnione są warunki umożliwiające wykonywanie zapisów do aplikacji systemu **asix** opisane w punkcie *Zapis prosty (patrz: punkt 6.3.1.)*.

Po naciśnięciu przycisk *Czytaj* z drugiej tabeli wywoływane jest makro. Makro to za pośrednictwem serwera Automation danych bieżących pobiera wartości wszystkich zmiennych i wpisuje je do tabeli. Dane pobierane są w formacie czterokolumnowym z rozbiciem na status, wartość, jakość i stempel czasu.

12.5. Serwer DDE – dane bieżące – uaktualnianie

Przykład ten demonstruje sposób dostępu do danych bieżących aplikacji systemu **asix** przy użyciu serwera DDE poprzez przysyłanie do klienta DDE bieżącej wartości zmiennej. W arkuszu w kolumnie G znajdują się formuły będą odwołaniami do danych zdalnych udostępnianych przez serwer DDE. Aby przykład działał, należy wcześniej uruchomić serwer DDE.

Formuły w kolumnie G są odwołaniami do pojedynczych zmiennych. Wyjątkiem jest formuła znajdująca się wierszach 46-49, która jest formułą grupową i odwołuje się do czterech zmiennych.

12.6. Serwer OPC – dane bieżące

Do pakietu nie jest dołączany przykład użycia serwera OPC. Nie jest możliwe wykorzystanie serwera OPC w makrze napisanym w języku Visual Basic.

Możliwości serwera OPC można poznać pobierając z witryny *http://www.matrikon.com/drivers/* darmowy program *OPC Client*. Aby obejrzeć stan zmiennych, należy wykonać poniższe kroki.

- Skonfigurować kanał podstawowy przy pomocy programu pakietu AsixConnect o nazwie *Konfigurator*.

- Uruchomić program *Matrikon OPC Explorer.,*
- Zaznaczyć serwer *ServerCTOPC.App* i z menu *Server* wybrać polecenie *Connect*.
- Z menu *Server* wybrać polecenie *Add Group*, podać dowolną nazwę grupy i nacisnąć *OK*.
- Z menu *Group* wybrać polecenie *Add Items*.
- Umieścić w oknie *Tags To Be Added* dowolną liczbę zmiennych i z menu *File* wybrać polecenie *Update and Return to Explorer*.
- Po chwili w oknie głównym pojawią się informacje o stanie wybranych zmiennych.

12.7. Serwer Automation – dane archiwalne

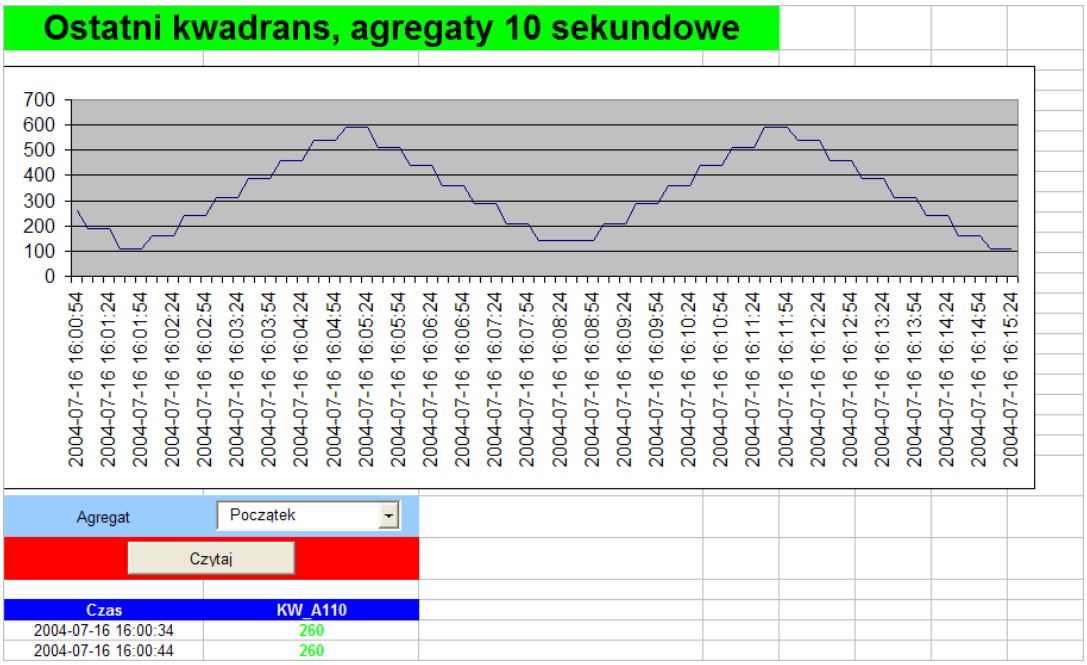
Przykład ten demonstruje sposób dostępu do danych archiwalnych aplikacji systemu **asix** przy użyciu serwera Automation. Przykład do poprawnej pracy wymaga programu Microsoft Excel 2000 lub XP.

W arkuszu tym ustanowiono powiązanie z biblioteką serwera Automation, dzięki czemu makra mogą bezpośrednio korzystać z obiektów serwera Automation pakietu AsixConnect. Oprócz łatwości programowania uzyskuje się również maksymalną wydajność komunikacji. Aby sprawdzić, jakie są aktualnie ustanowione powiązania, należy z menu *Tools* edytora Visual Basica wybrać polecenie *References*. Biblioteka serwera Automation danych bieżących nosi nazwę *AsixConnect 4 Type Library*.

W zakładce Agregaty godzinowe arkusza znajduje się przykład raportu zawierającego agregaty godzinowe.

W przykładzie tym w komórce C6 znajduje się data generowania raportu, a w komórce C7 godzina początku generowania raportu. W komórce C8 znajduje się lista rozwijana zawierająca dostępne agregaty. Domyślnie wybrana jest średnia. Po naciśnięciu przycisku Czytaj wywoływane jest makro, które uruchamia serwer Automation i wywołuje funkcję pobierania danych przetworzonych – agregatów. Wyniki wpisywane są w zakresie B12-C35.

W zakładce *Ostatni kwadrans* arkusza znajduje się przykład raportu zawierającego agregaty 10-sekundowe wyliczane z danych z ostatnich 15 minut. W przykładzie tym, przy wywołaniu funkcji pobierania danych przetworzonych, specyfikując okres czasu, korzysta się z czasu względnego.



12.8.Serwer OLE DB - Dane archiwalne – Makro

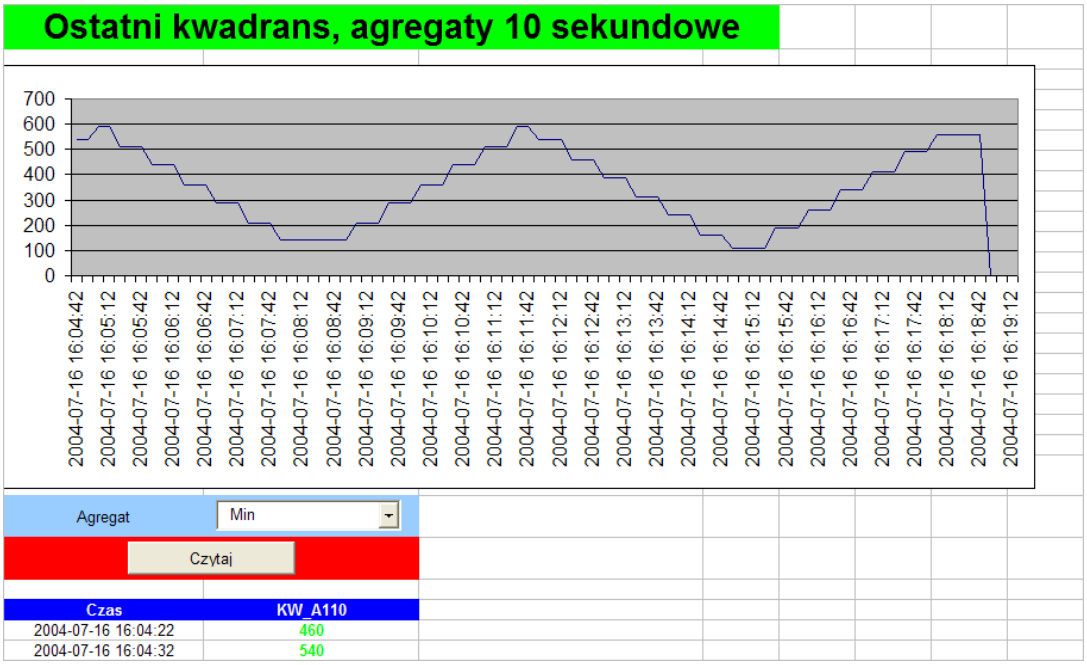
Przykład ten demonstruje sposób dostępu do danych archiwalnych aplikacji systemu **asix** przy użyciu serwera OLE DB. Do poprawnej pracy przykład wymaga programu Microsoft Excel 2000 lub 2002 (XP).

W arkuszu tym ustanowiono powiązanie z biblioteką ADO. Aby sprawdzić, jakie są aktualnie ustanowione powiązania, należy z menu *Tools* edytora *Visual Basic* wybrać polecenie *References*. Biblioteka serwera Automation danych bieżących nosi nazwę *Microsoft Active Data Objects 2.5*. Wersja biblioteki ADO może się różnić, ale powinna być wyższa niż 2.1.

W zakładce Agregaty godzinowe arkusza znajduje się przykład raportu zawierającego agregaty godzinowe.

W przykładzie tym w komórce C6 znajduje się data generowania raportu, a w komórce C7 godzina początku generowania raportu. W komórce C8 znajduje się lista rozwijana zawierająca dostępne agregaty. Domyślnie wybrana jest średnia. Po naciśnięciu przycisku *Czytaj* wywoływane jest makro, które uruchamia serwer OLE DB i pobiera dany. Wyniki wpisywane są w komórkach zakresie B12-C35.

W zakładce *Ostatni kwadrans* arkusza znajduje się przykład raportu zawierającego agregaty 10 sekundowe wyliczane z danych z ostatnich 15 minut. W przykładzie tym przy pobieraniu danych przetworzonych, specyfikując okres czasu, korzysta się z czasu względnego.



12.9.Serwer OLE DB - Borland C++Builder 6

Program przykładowy został stworzony przy użyciu pakietu *Borland C++Builder 6 Enterprise Edition*. Można go również skompilować przy użyciu wersji *Professional* uzupełnionej o bibliotekę *ADO Express*.

Program pobiera dane z przykładowej aplikacji systemu **asix** AC3Demo. Program zakłada, że **asix** został zainstalowany w katalogu *c:\asix*. W tabeli wyświetlane są średnie 5-minutowe zmiennej K811_U14 za ostatnią godzinę.

Głównym elementem programu jest obiekt *ADOConnection1*. Służy on do nawiązania połączenia z serwerem OLE DB. Własności *ConnectionString* zawiera opis tego połączenia. Pakiet *C++Builder 6* do skonfigurowania własności *ConnectionString* używa mechanizmu opisanego w punkcie Identyfikacja i parametryzacja. Zapytanie jest umieszczone w obiekcie *ADOQuery1*. Po zaznaczeniu pola *Czytaj dane* obiekty *ADOConnection1* i *ADOQuery1* są aktywowane i dane pojawiają się w obiekcie tabela *DBGrid1*.

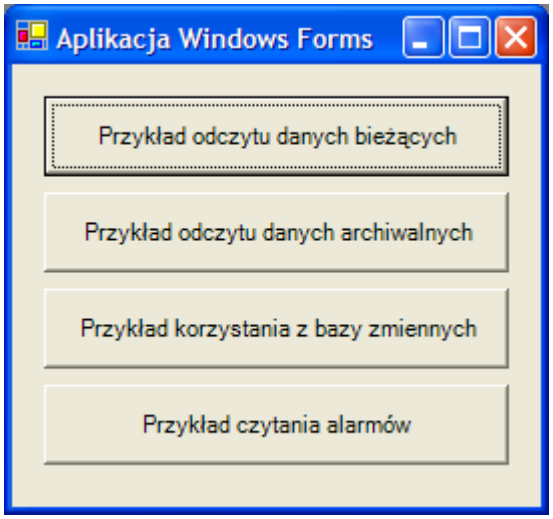
12.10. Serwery .NET

12.10.1. Serwery .NET

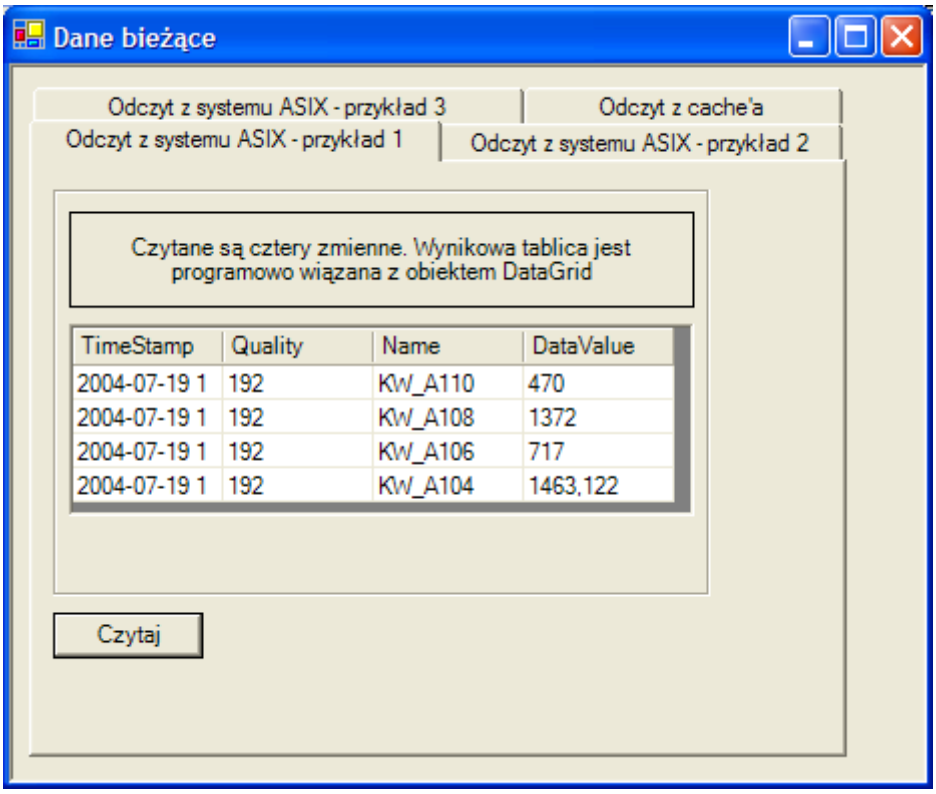
Przykłady wykorzystania serwerów .NET znajdują się w podkatalogu AC4Przyklady\Serwery .NET. W podkatalogu WindowsForm znajduje się aplikacja typu Windows Forms w wersji językowej C#. W podkatalogu WebForm znajduje się aplikacja typu WebForms.

12.10.2. Windows Forms

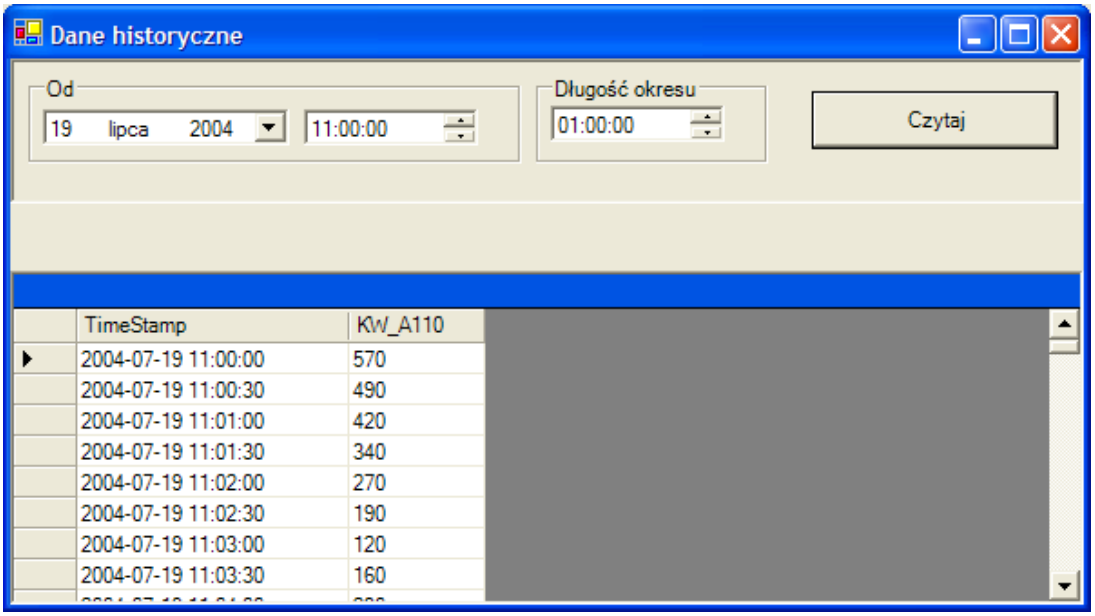
Po uruchomieniu programu pojawia się okno główne:



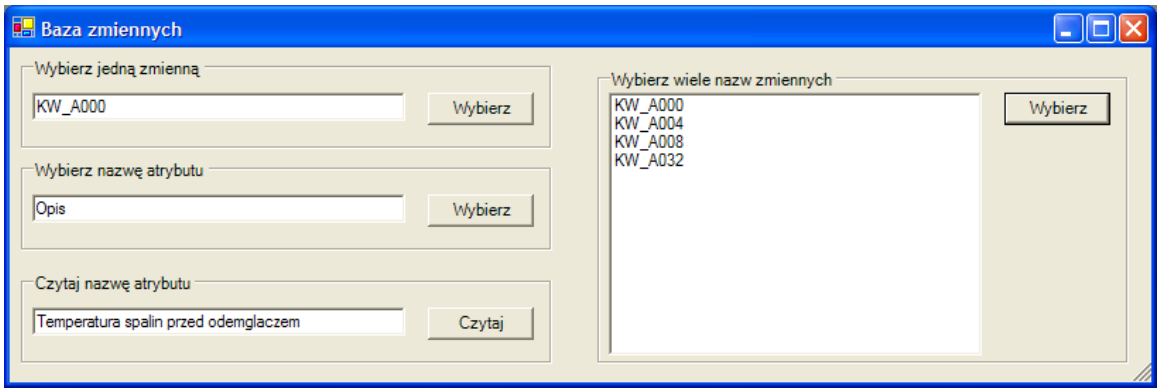
Naciśnięcie odpowiednich przycisków spowoduje uruchomienie przykładowych okien dla poszczególnych rodzajów danych.



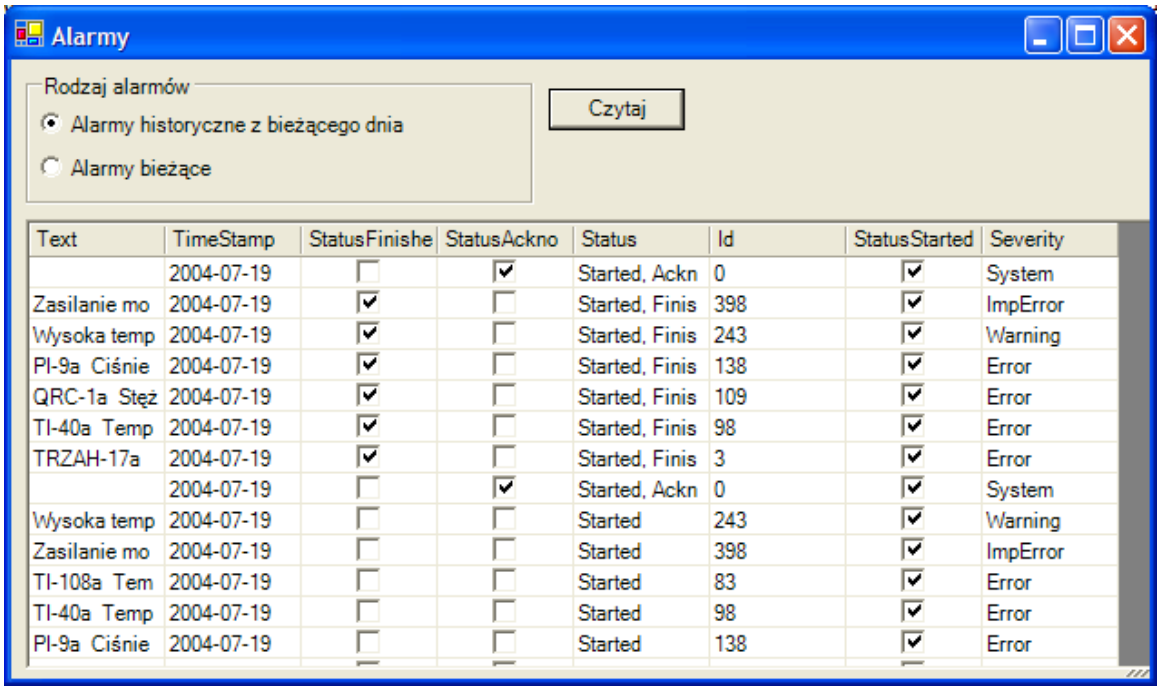
W zakładce *Odczyt z systemu ASIX – przykład 1* znajduje się wynik przykładu odczytu danych oraz wstawiania danych do obiektu typu DataGrid. Pozostałe zakładki demonstrują inne sposoby wstawiania danych do elementów wizualnych oraz demonstrują sposób odczytu z pamięci podręcznej serwera i uaktualniania danych.



W oknie znajduje się wynik przykładu odczyt danych historycznych z aplikacji systemu **asix**.



W oknie znajduje się wynik przykładu wybierania zmiennych przy pomocy okna wyboru zmiennej oraz czytania wartości atrybutów z bazy zmiennych.



Text	TimeStamp	StatusFinishe	StatusAckno	Status	Id	StatusStarted	Severity
	2004-07-19	☐	☑	Started, Ackn	0	☑	System
Zasilanie mo	2004-07-19	☑	☐	Started, Finis	398	☑	ImpError
Wysoka temp	2004-07-19	☑	☐	Started, Finis	243	☑	Warning
PI-9a Ciśnie	2004-07-19	☑	☐	Started, Finis	138	☑	Error
QRC-1a Stęż	2004-07-19	☑	☐	Started, Finis	109	☑	Error
TI-40a Temp	2004-07-19	☑	☐	Started, Finis	98	☑	Error
TRZAH-17a	2004-07-19	☑	☐	Started, Finis	3	☑	Error
	2004-07-19	☐	☑	Started, Ackn	0	☑	System
Wysoka temp	2004-07-19	☐	☐	Started	243	☑	Warning
Zasilanie mo	2004-07-19	☐	☐	Started	398	☑	ImpError
TI-108a Tem	2004-07-19	☐	☐	Started	83	☑	Error
TI-40a Temp	2004-07-19	☐	☐	Started	98	☑	Error
PI-9a Ciśnie	2004-07-19	☐	☐	Started	138	☑	Error

W oknie znajduje się wynik przykładu czytania alarmów historycznych. Możliwy jest również odczyt danych bieżących.

12.10.3. Web FormTest

Przykład wykorzystania serwera WebService w aplikacji napisanej w technologii *WebForm* znajduje się w podkatalogu *AC4Przyklady\Serwery .NET\WebFormTest*. Przykład zawiera cztery strony *aspx* wyświetlające dane bieżące, archiwalne surowe i przetworzone oraz alarmy.

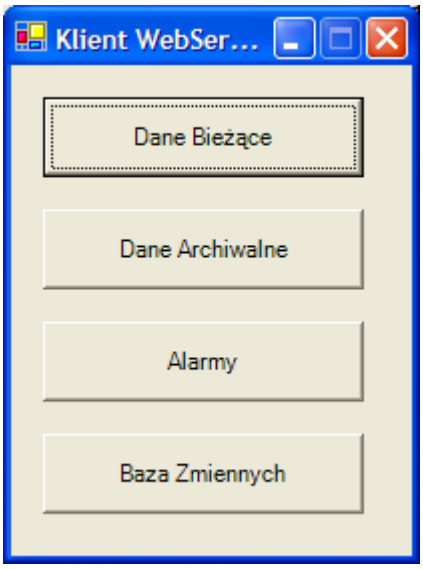
12.11. Serwer Web Service

12.11.1. Web Forms

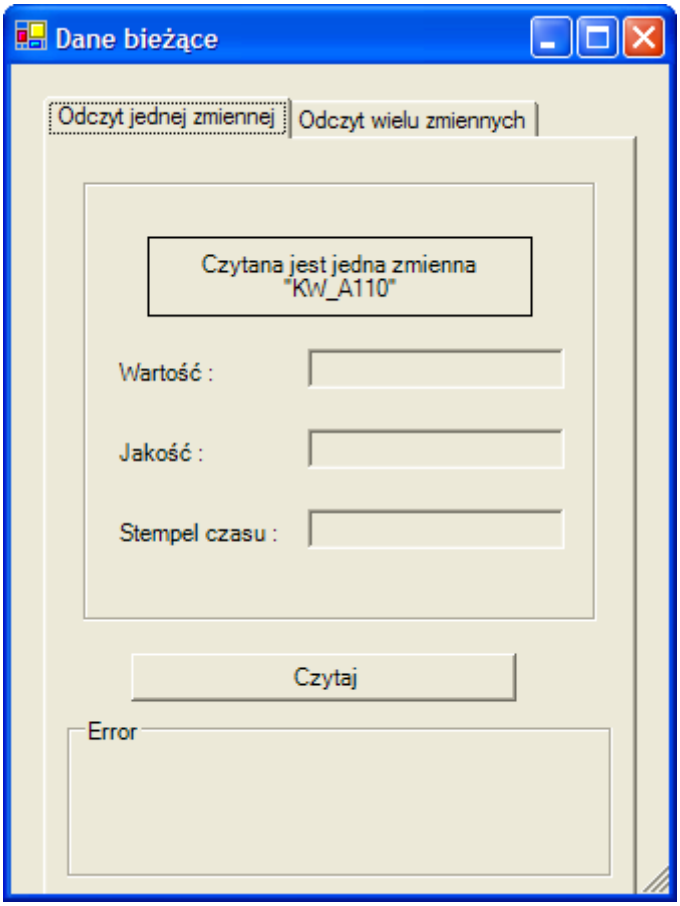
Przykład wykorzystania serwera WebService w aplikacji napisanej w technologii *WebForm* znajduje się w podkatalogu *AC4Przyklady\Serwer WebService\Klient_WebService*.

Program przykładowy nie zawiera kodu do autentyfikacji Windows i aby mógł się on dołączyć do serwera *WebService* serwer musi mieć włączony dostęp anonimowy.

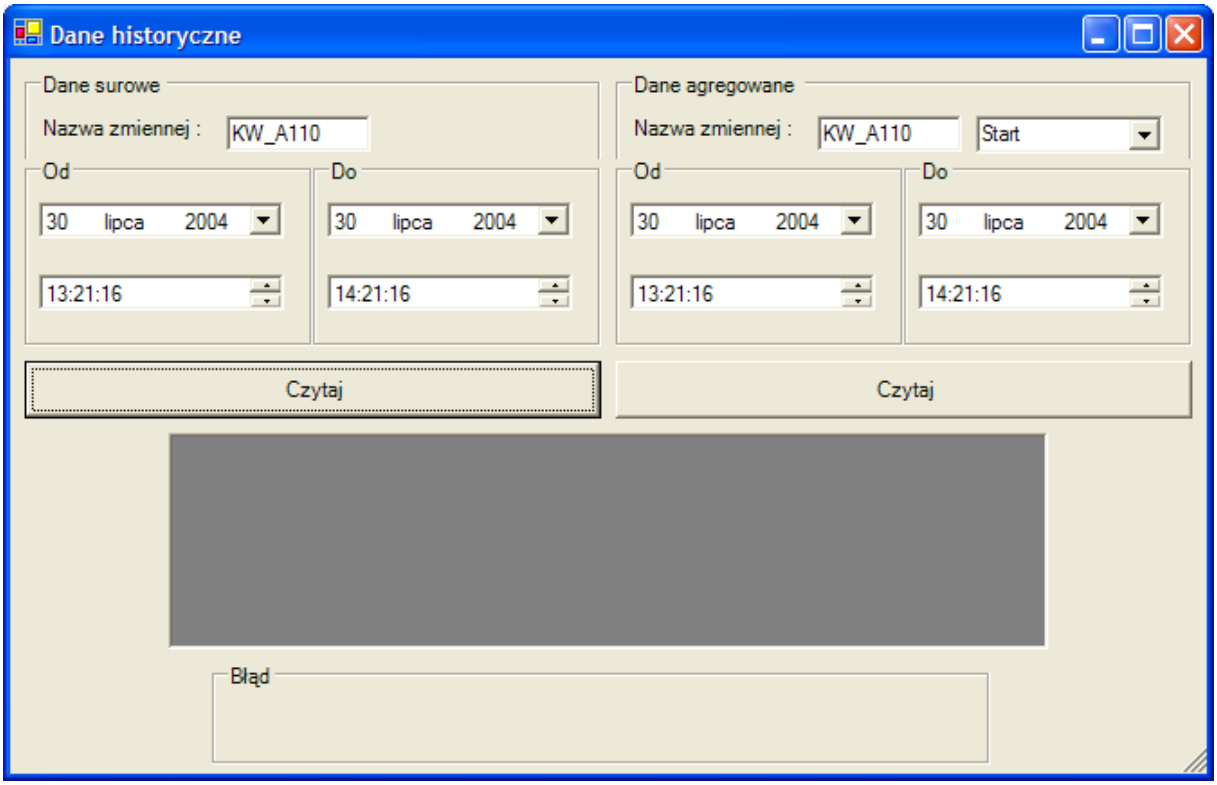
Po uruchomieniu programu pojawia się okno główne:



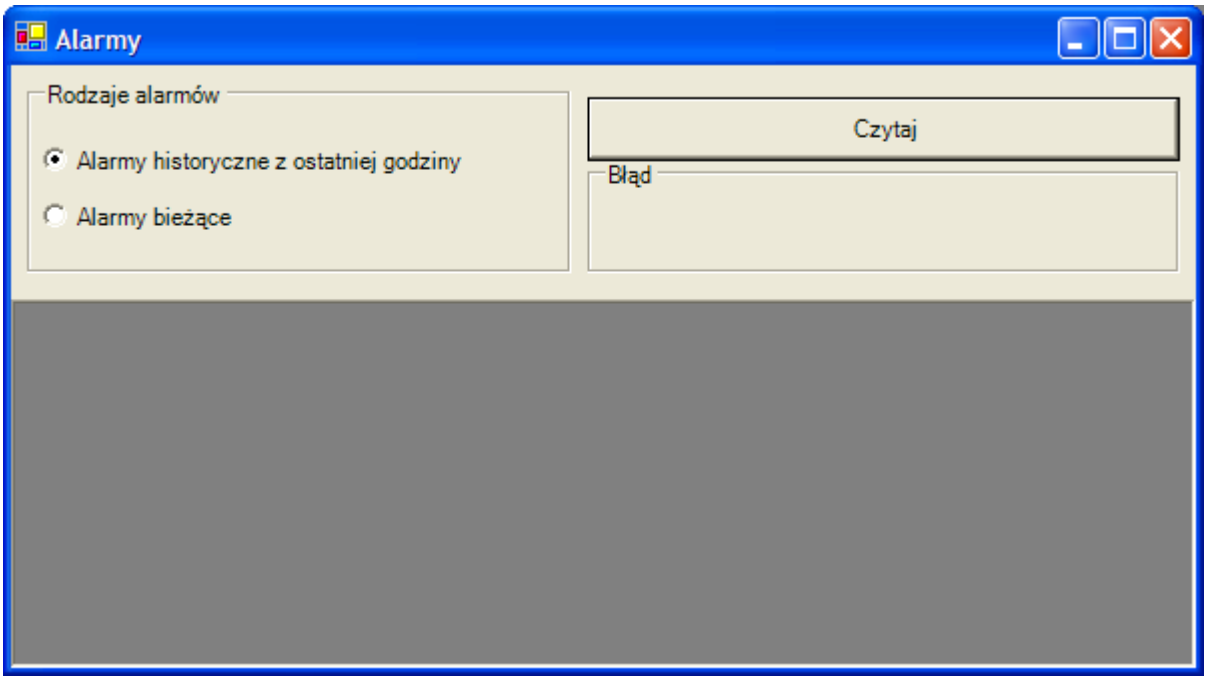
Naciskając odpowiednie przyciski pojawiają się kolejne okna przykładowe dla poszczególnych rodzajów danych.



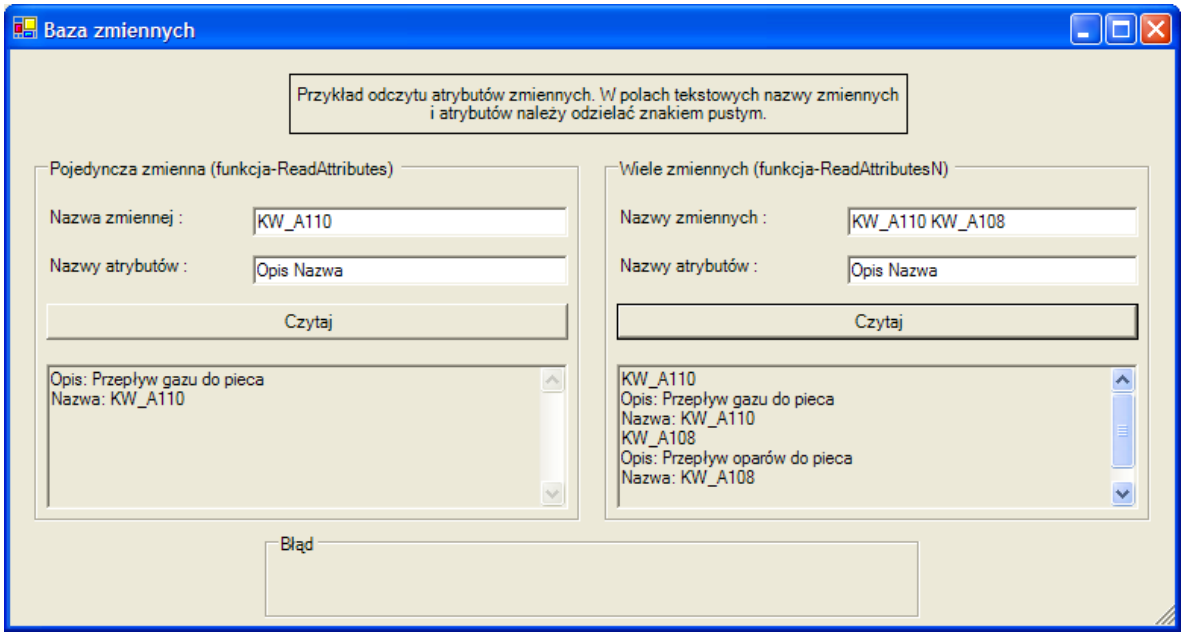
Naciśnięcie przycisku *Czytaj* powoduje odczyt wartość jednej zmiennej przy użyciu funkcji *Read*. W zakładce *Odczyt wielu zmiennych* znajduje się przykład odczytu wielu zmiennych na raz przy pomocy funkcji *ReadN*.



Naciśnięcie lewego przycisku *Czytaj* powoduje odczyt wartości historycznych surowych zmiennej przy pomocy funkcji serwera *ReadRaw*. Naciśnięcie prawego przycisku *Czytaj* powoduje odczyt agregowanych wartości historycznych zmiennej przy pomocy funkcji serwera *ReadProcessed*.



Naciśnięcie przycisku *Czytaj* powoduje odczyt wartości alarmów. W zależności od wybranej opcji w polu *Rodzaje alarmów* używana jest funkcja *ReadHistoricalAlarms* lub *ReadActiveAlarms*.



Naciśnięcie lewego przycisku *Czytaj* powoduje odczyt wartości atrybutów jednej zmiennej przy pomocy funkcji serwera *ReadAttributes*. Naciśnięcie prawego przycisku *Czytaj* powoduje odczyt agregowanych wartości historycznych zmiennej przy pomocy funkcji serwera *ReadAttributesN*.