www.asix.com.pl


# Asix.Evo - Operator Actions

*Doc. No ENP7E004*
*Version: 2012-07-30*

ASKOM

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

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

# Table of Contents

1 Use of Operator Actions ..... 3
2 Complex Actions ..... 5
2.1 Parameters of Complex Actions ..... 5
3 List of Operator Actions ..... 7
4 Description of operator actions ..... 10
4.1 AcceptAlarm ..... 10
4.2 Actions ..... 11
4.3 ActionSet ..... 12
4.4 AsbaseAdd ..... 13
4.5 AsbaseClose ..... 14
4.6 AsbaseDelete ..... 15
4.7 AsbaseLoad ..... 16
4.8 AsbaseNavigate ..... 17
4.9 AsbaseOpen ..... 19
4.10 AsbasePrint ..... 21
4.11 AsbaseShow ..... 22
4.12 AsbaseUpdate ..... 23
4.13 AstrendAsix6 ..... 24
4.14 AstrendDisplay ..... 26
4.15 AstrendPrint ..... 28
4.16 Break ..... 30
4.17 CancelControls ..... 31
4.18 ChangePassword ..... 32
4.19 CloseWindow ..... 33
4.20 ConfirmRole ..... 35
4.21 EditPatterns ..... 37
4.22 EndAlarm ..... 38
4.23 ExcludeAlarm ..... 39
4.24 Execute ..... 40
4.25 IncludeAlarm ..... 41
4.26 Login ..... 42
4.27 Logout ..... 43

4.28 Message ..... 44
4.29 MinimizeWindow ..... 45
4.30 Nothing ..... 46
4.31 OpenDiagram ..... 47
4.32 OpenWindow ..... 50
4.33 Perform ..... 54
4.34 PlaySound ..... 55
4.35 PlaySoundLooping ..... 56
4.36 PrintScreen ..... 57
4.37 RefreshVariable ..... 58
4.38 Run ..... 59
4.39 Script ..... 60
4.40 SecurityBrowser ..... 61
4.41 SecurityManager ..... 62
4.42 SendControls ..... 63
4.43 SetBounds ..... 64
4.44 SetPosition ..... 65
4.45 SetProperty ..... 67
4.46 SetSize ..... 68
4.47 SetVariable ..... 69
4.48 SetWindowSize ..... 71
4.49 ShowControlPanel ..... 72
4.50 ShowKeyboard ..... 73
4.51 ShowMenu ..... 74
4.52 ShowNumericKeyboard ..... 75
4.53 StartAlarm ..... 76
4.54 StopSound ..... 77
4.55 SwitchLanguage ..... 78
4.56 SwitchWorkMode ..... 79
4.57 Terminate ..... 80
4.58. ToggleBits ..... 81
4.59 VariableInfo ..... 82

# 1 Use of Operator Actions

Operator actions are basic means of the operator's control of the running application. The system designer shares the actions with the operator via the following mechanisms:

- Handling object events and synoptic diagrams
- User menu
- Schedule
- Global keys

**The general form of the actions is as follows:**

**action_name( *parameter1*, *parameter2*, … )**

The number and significance of parameters depend on the action name. In the simplest case, all parameters can be specified directly, e.g.:

OpenWindow(WindowP1,MainPanel,Room121,"")

The above action opens a window called *WindowP1*, and the *Room121* diagram will be inserted in the *MainPanel* panel.

Each of the action parameters may be also an optional expression. The same principles of building expressions as for synoptic object properties are valid for the actions. Upon execution of an action the values of all expressions used are calculated and only the actions formed that way may be executed, e.g.

OpenWindow(WindowP1,MainPanel,Room +Variable(RoomNo) ,"")

The diagram name parameter is created dynamically on the basis of the value of the *RoomNo* variable. If upon the action execution the variable value is equal to 101, the *Room101* diagram will be opened.

The action execution is also dependent on the context of its use. The action executed as part of the event handling has a context-specific access to the properties and parameters of objects or diagrams for which the event has been activated. The actions executed from the schedule or global key do not have such possibility. The actions executed via the user menu may have access to the properties and parameters of objects and diagrams, if the menu is opened by *ShowMenu* action used within the event handling process - in this case the context of use is inherited.

OpenWindow(WindowP1,MainPanel,Room +Variable() ,"")

In the above case the diagram name is created based on the main variable value of the object for which the event has been activated - the parameterless *Variable* function retrieves the value of the

contextual main variable. If upon the action execution the variable value is equal to 101, the *Room101* diagram will be opened.

OpenWindow(WindowP1,MainPanel,Room +Parameter(RoomNo) ,"")

In the above case the diagram name is created based on the value of a parameter named *RoomNo*. This parameter is the parameter of a diagram or template (also depending on the context of use). If upon the action execution the parameter value is equal to 101, the *Room101* diagram will be opened.

 **See also**

- AsixEvo_Expressions (PDF/CHM)
- AsixEvo_Visualization_Elements (PDF/CHM): *6.3. The Standard Object Events*

  *4.2. The Diagram Events*

  *5. Menu*
- Asix.Evo_Application_Parameterization (PDF/CHM): *3. Configuring the Scheduler*

# 2 Complex Actions

In the case where a need to perform a sequence of operations occurs, complex actions may be used. These are simply sets of regular operator actions which are executed sequentially.

The complex actions are defined in the *Action Sets* work panel via a tree of *Application Explorer* panel elements. A complex action is executed by the *ActionSet* operator action.

Generally, the complex action execution involves sequential execution of each of the component actions. The execution of a complex action may be conditionally aborted through the *Break* operator action, for example:

```
Break(Variable(RoomNo)==0)
```

If the *RoomNo* variable value is equal to 0, the component actions queued after the *Break* action will not be executed.

The operator actions constituting a complex action feature all the properties of regular operator actions. In particular, the values of expressions used to define the parameters are calculated not earlier than upon the execution. Standard inheritance of context also occurs; if the *ActionSet* action has been activated within the context of an event handling, all the component actions will be run in the same context.

**NOTE:**
In simple cases, instead of a complex action it is possible to alternatively use the simple *Actions* action which also allows to execute several component actions sequentially.

## 2.1 Parameters of Complex Actions

Complex actions are characterised by action parameters. They should not to be confused with the diagram and template parameters available through the *Parameter* function. It is possible to define the action parameters while defining the complex action. The parameters are identified by name. Then, it is possible to call the parameter anywhere within the component actions by entering into the action content the parameter name preceded by the "@" character.

The complex action execution procedure is explained by the following example.

Prerequisites:

- The *A1* complex action is declared to use the *Room* and *Setpoint* parameters
- One of the component actions is of the following form:

SetVariable(Setpoints+@Room,@Setpoint)

- The complex action is executed through the action running within the context of the event handling:

ActionSet(A1,RoomNo+Parameter(RoomNumber),”10+1”)

Execution:

- At the time of the *ActionSet* action execution the second parameter value of the action is calculated through a combination of the text *RoomNo* and the parameter value (of diagram or template) named *RoomNumber*. The value of the calculated parameter is handed over as *Room* parameter in a text format to the complex action. Let's assume that the calculated value is equal to "RoomNo99". The third parameter of the *ActionSet* action is handed over in an unchanged form of "10 +1" as the *Setpoint* parameter.
- Upon execution of the *SetVariable* component action, the variable name is calculated by combining the text "Setpoints" and the value of the *Room* component action (which yields the name *SetpointsRoomNo99*). The value is controlled directly with the value of the *Setpoint* parameter and will be equal to the text "10 +1".
- As a result, the *SetpointsRoomNo99* variable will be set to the text value "10 +1".

If the aim was setting the value of 11, the *ActionSet* action of the following form should be used:

ActionSet(A1,RoomNo+Parameter(RoomNumber),10+1)

In this case, the calculation of the expression 10 +1 would be preformed before executing the complex action.

# 3 List of Operator Actions

**Synoptic window management**

CloseWindow - closing synoptic windows

MinimizeWindow - minimizing synoptic windows

OpenDiagram - opening a synoptic diagram in a dynamically created window

OpenWindow - opening a predefined synoptic window or replacing a diagram in the panel of a window opened previously

SetWindowSize - resizing synoptic windows

**Access to process variables and monitoring of control operations**

CancelControls - cancelling of pending control operations

RefreshVariable - forced refreshing of a process variable value

SendControls - executing pending control operations

SetVariable - setting a process variable value

ToggleBits - changing bits of a process variable value

VariableInfo - displaying a window with information about the selected process variable

**Application performance control**

ShowControlPanel - opening the control panel window

SwitchLanguage - changing the current application interface language

SwitchWorkMode - switching the working mode of the application

Terminate - closing the application

**Operator action execution control**

Actions - executing a sequence of component actions

ActionSet - executing a complex action

Break - conditional breaking of a complex action running

Execute - executing an action the content of which is constructed dynamically

Nothing - blank action

Perform - conditional executing of an action

**Audiable signals**

PlaySound – playing an audio file

PlaySoundLooping - playing an audio file in loop mode

StopSound - stopping an audio file playing

**Integration with the alarm system**

AcceptAlarm - alarm acknowledgement

EndAlarm - alarm termination

ExcludeAlarm - excluding an alarm from handling

IncludeAlarm - resuming alarm handling

StartAlarm - starting an alarm

**Integration with authorisation system**

ChangePassword - changing the user password

ConfirmRole – verification of the assignment of a user to the role

Login – user login

Logout - user logout

SecurityBrowser - opening the security system logs browsing window

SecurityManager - opening a users' permissions management window

**Running the application components**

Run – running an external programme or file

Script – running a user script

Showmenu - opening a context menu

**Controlling the operation of AsTrend application**

AstrendAsix6 - controlling AsTrend application operation. Action used only for automatic conversion of the application from Asix6 system.

AstrendDisplay - displaying a trend with AsTrend application

AstrendPrint - printing a trend with AsTrend application

**Communication with AsBase module**

AsbaseAdd - adding a new record within the connection

AsbaseClose - closing a connection between the Asix.Evo application and AsBase module

AsbaseAdd - deleting a current record within the connection

AsbaseLoad - loading a value from the selected recipe to the set of process variables

AsbaseNavigate - controlling browsing and management of data within the connection.

AsbaseOpen – establishing a connection between the Asix.Evo application and AsBase module

AsbasePrint - printing data selected from the AsBase module archive

AsbaseShow - displaying an AsBase module window with an option of switching to the desired set of data

AsbaseUpdate - changing the field values of the current record within the connection

**Additional**

EditPatterns - opening a model trend editor window

Message - recording information into the event log

PrintScreen - saving an image of the current window, diagram or screen into the system clipboard or file

SetBounds – changing the position and size of object, template or object group

SetPosition – changing the position of object, template or object group

SetProperty - setting properties of an object or synoptic diagram

SetSize - changing the size of object or template

ShowKeyboard - displaying the window of on-screen keyboard

ShowNumericKeyboard – displaying the window of on-screen numeric keyboard

# 4 Description of operator actions

## 4.1 AcceptAlarm

**Intended use**

The action is used to acknowledge the alarm by the user. An alternative method for the alarm acknowledgement is to use the *Accept* function of *IAlarm* interface in the user script code. Alarms may also be acknowledged with the use of the *Active Alarms Viewer* object.

**Syntax**

**AcceptAlarm(*domain_name, alarm_ID*)**

**Parameters**

***domain_name***

Name of the alarm domain related to the action. If the name is not provided, the action relates to the default domain (the first one).

***alarm_ID***

Alarm ID to be acknowledged.

 **See also**

- *Alarm System* (PDF/CHM)

- *Active Alarms Viewer Object* (AsixEvo_Objects.PDF/CHM)

- *IAlarm* interface (AsixEvo_Scripts.PDF/CHM)

## 4.2 Actions

**Intended use**

The action is used for sequential execution of a component action set. Its functioning is similar to the complex action. It may be implemented in simpler cases where the number of the component actions and their complexity are not too great.

**Syntax**

**Actions(*action1, action2, ...*)**

**Parameters**

***action1, …, actionN***

The component actions which will be executed sequentially.

**Examples**

Actions( SetVariable(v1,1), SetVariable(v2,1))

The action runs consecutively two component actions that set the values of two process variables.

 **See also**

*2. Complex Actions*

## 4.3 ActionSet

**Intended use**

The action is used to run a complex action.

**Syntax**

**ActionSet(*name, parameter1, parameter2, ...*)**

**Parameters**

**name**

The name of a complex action.

***parameter1, …, parameterN***

The values of complex action parameters. The number of parameters used in the call must be in accordance with the number of parameters in the definition of a complex action. Successive call parameters will be substituted for the parameters of the complex action in the order of their occurrence in the definition of the complex action.

**Examples**

ActionSet(switchoff_aggregate, agr100, agr101)

This action runs the complex action called *switchoff_aggregate*. Assuming that the action *switchoff_aggregate* has two parameters named *variable1* and *variable2* defined in this order, upon the action execution the parameter *variable1* will be equal to *agr100*, and the parameter *variable2* will be equal to *agr101*.

 **See also**

*2. Complex Actions*

## 4.4 AsbaseAdd

**Intended use**

This action adds a new record within the connection established earlier with the *AsbaseOpen* action. The field values of the new record result from the values of the process variables of the variable set used in the connection.

**Syntax**

**AsbaseAdd (*connection_ID*)**

**Parameters**

***Connection_ID***

The identifier used earlier in the *AsbaseOpen* action should be specified in the parameter.

 **See also**

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Application*

- *AsbaseOpen* action

## 4.5 AsbaseClose

**Intended use**

This action closes the connection established earlier with the *AsbaseOpen* action.

*AsbaseClose* action should be used in the *Diagram Closed* event handling of the diagram for which the connection was established.

**Syntax**

**AsbaseClose (*connection_ID*)**

**Parameters**

***Connection_ID***

The identifier used earlier in the AsbaseOpen action should be specified in the parameter.

 **See also**

- *AsbaseOpen* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Module*

## 4.6 AsbaseDelete

**Intended use**

This action deletes the current record within the connection established earlier with the *AsbaseOpen* action.

**Syntax**

**AsbaseDelete (*connection_ID*)**

**Parameters**

***Connection_ID***

The identifier used earlier in the *AsbaseOpen* action should be specified in the parameter.

 **See also**

- *AsbaseOpen* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Module*

## 4.7 AsbaseLoad

**Intended use**

The action allows to load setpoints from a record in a selected group of recipes into process variables associated with this group in AsBase module. The record of setpoints may be selected by direct entering of the recipe name or by interactive selection of the name.

If the connection mode is employed, the *AsbaseNavigate* action with the $*Load* option should always be used as the action for recipe load.

**Syntax**

**AsbaseLoad(*recipe_group_ID, variable_set_ID, recipe_name*)**

**Parameters**

***recipe_group_ID***

The recipe group ID from which the recipe field values are loaded

***variable_set_ID***

The ID of the variable set which will be provided with values from the selected recipe.

***recipe_name***

The name of the recipe to be loaded. If the recipe name is not specified, the window with a list of all the recipes defined within the group of recipes will be displayed. From this list the user can choose the recipe to be loaded.

**Examples**

AsbaseLoad( Setpoints,Tank1 ,"" )

The action will display a message box with a list of names of all the recipes within the group Setpoints. The user will be able to choose the recipe whose field values are to be loaded into the process variables from the set called *Tank1*.

 **See also**

- *AsbaseNavigate* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Application*

## 4.8 AsbaseNavigate

**Intended use**

**- controlling browsing and management of data within the connection.**

This action is used to control browsing and management of data within the connection established previously with the *AsbaseOpen* action. The action enables changing the current record within the connection, executing the recipe load operation and changing the connection parameters.

**Syntax**

**AsbaseNavigate (*connection_ID, operation_type, operation_parameters*)**

**Parameters**

***connection_ID***

The identifier used earlier in the AsbaseOpen action should be specified in the parameter.

***operation_type***

This parameter determines the operation to be executed within the connection. The possible values include one of the below:

**$Count** - recalculates the number of records displayed within the connection according to the current filter and puts this number in the appropriate variable of the set of variables.

**$Filter** - defines a new filter and, after the filter application, forces the movement to the last record of the connection. Details of the filter syntax - see: *Operation with the AsBase Module* (Asix.Evo_Application_Parameterization.PDF/CHM).

**$First –** forces the movement to the first record of the connection.

**$Download** – sends the current record content into the variables from the set of variables specified explicitly in the parameter *operation_parameters*.

**$Last** – forces the movement to the last record of the connection.

**$Load** - forces the movement to the record specified by the name of the recipe or selected interactively by the user. The operation is only available for connections related to groups of recipes.

**$Next –** forces the movement to the next record of the connection.

**$Previous** – forces the movement to the previous record of the connection.

***operation_parameters***

For the *$Filter* operation, the parameter specifies new content of the filter. For the *$*Download operation, the parameter specifies the name of the set of variables into which the values of the current record should be sent. For the *$Load* operation, the parameter should be filled out with the name of the recipe - a blank value means a list of all the available names will be displayed from which the user can select the recipe.

**Examples**

AsbaseNavigate(conn1, $Next, null)

This action forces moving on to the next record within the data of the connection *conn1*. The variables of the variable set located within the connection will be filled out with the values loaded from the new current record.

AsbaseLoad(conn1, $Load, RC_123)

This action forces the movement to the recipe called *RC_123* within the recipe group resulting from the connection *conn1*. The variables of the variables set located within the connection will be filled out with the values loaded from the new current record.

 **See also**

- *AsbaseOpen* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Module*

## 4.9 AsbaseOpen

**Intended use**

This action establishes a connection used to exchange data between the Asix.Evo Application and the AsBase application. The action parameters define which data will be exchanged and through which process variables. When the action execution is completed, the current record data of the connection will be recorded in the interface variables. The action may be used only after the AsBase application parametrisation.

The *AsbaseOpen* action should be used in the *Diagram Opened* event handling process for the diagram which must remain opened during data exchange. To close the connection within the *Diagram Closed* event, the*AsbaseClose* action must be used.

**Syntax**

**AsbaseOpen (*connection_ID, data_type, source_ID, variable_set_ID, slave_source_ID, slave_variable_set_ID, filter*)**

**Parameters**

***Connection_ID***

The ID that will uniquely identify the connection with the specified data source and the set of variables should be specified in the parameter. The ID is used later in the other actions of the family *AsbaseXxx*.

***data_type***

This parameter specifies the type of data source. The possible value is one of the below constants:

**$Archive** - the connection relates to the registration set
**$Recipes** - the connection relates to the recipe group
**$Loads** - the connection relates to the recipe load history

***source_ID***

The data source ID, depending on the *data_type* parameter, is the identifier of the registration set or recipe group.

***variable_set_ID***

The variable set ID which will be used in communication between the Asix.Evo application and AsBase module.

***slave_source_ID***

The slave data source ID, i.e. source which in the AsBase parameterization has been declared as being in relation with the master source.

***slave_variable_set_ID***

The variable set ID which will be used in communication between the Asix.Evo application and AsBase module. The set for communication with the slave source.

***filter***

This filtering expression allows limiting the range of data available through the connection. Details of the filter syntax - see: *Operation with the AsBase Module* (Asix.Evo_Application_Parameterization.PDF/CHM).

**Examples**

AsbaseOpen(conn1, $Archive, Paints, PaintsDgr, null, null, ”V_Akt=1” )

This action creates the connection called *conn1*. This connection provides the access to the data records of the registration set of *Paints* ID satisfying the condition saying that the value of the set field of *Akt* ID will be 1. The data will be replaced by the variables set of ID *PaintsDgr*. The active record data of the connection will be stored in variables of this set.

 **See also**

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Module*

- *AsbaseClose*, *AsbaseAdd*, *AsbaseDelete*, *AsbaseNavigate*, *AsbaseUpdate* actions

## 4.10 AsbasePrint

**Intended use**

This action initiates printing the contents of selected tables by the AsBase module.

**Syntax**

**AsbasePrint (*data_type*, *source_ID*, *view_ID*)**

**Parameters**

***data_type***

This parameter specifies the type of data source. The possible values include one of the below:

**$Archive** - the connection relates to the registration set
**$Recipes** - the connection relates to the recipe group
**$Loads** - the connection relates to the recipe load history

***source_ID***

The data source ID, depending on the *data_type* parameter, is the identifier of the registration set or recipe group.

***view_ID***

The ID of the view associated in the Asbase application with the data source. This view specifies the records printing method and may also contain information used to filter the data.

**Examples**

AsbasePrint( $Recipes, Nastawy, ”” )

This action initiates printing all the records from the *Nastawy* recipe group in the default arrangement.

 **See also**

- Asix.Evo_Parametryzacja_aplikacji.PDF/CHM, *8. Operation with the AsBase Application*

## 4.11 AsbaseShow

**Intended use**

This action activates the AsBase module window (brings it to front). The data to be displayed may be specified in the action parameters.

**Syntax**

**AsbaseShow (*data_type, source_ID, variable_set_ID, view_ID*)**

**Parameters**

***data_type***

This parameter specifies the type of data source. The possible values is one of the below constants:

**$Archive** - the connection relates to the registration set
**$Recipes** - the connection relates to the recipe group
**$Loads** - the connection relates to the recipe load history

***source_ID***

The data source ID - depending on the *data_type* parameter, this is the identifier of the registration set or recipe group.

***variable_set_ID***

Variable set ID. Set only when the view window with the values of variables for the variable set is to be displayed.

***view_ID***

The ID of the view associated in the AsBase module with the data source. This view specifies the records displaying method and may also contain information used to filter the data.

**Examples**

AsbaseShow( $Recipes, Setpoints, ””,V1 )

This action will display the data of the Setpoints recipe group in the AsBase module window. Displaying method and range of data are determined by the view of *V1* ID defined in the AsBase module.

 **See also**

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Module*

## 4.12 AsbaseUpdate

**Intended use**

This action changes the current record within the connection established earlier with the *AsbaseOpen* action. The new values of the record fields result from the values of the process variables of the variable set used in the connection.

**Syntax**

**AsbaseUpdate (*connection_ID*)**

**Parameters**

***connection_ID***

The identifier used earlier in the *AsbaseOpen* action should be specified in the parameter.

 **See also**

- *AsbaseOpen* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *8. Operation with the AsBase Module*

## 4.13 AstrendAsix6

**Intended use**

This action is used to control the AsTrend module operation. This is the exact equivalent for the AsTrend action of the Asix6 system. This action is used only for automatic conversion of application from the Asix6 system. In applications developed from scratch, the *AstrendPrint* and *AstrendDisplay* actions should be used.

**Syntax**

**AstrendAsix6 (*trend_name, start_time, X_coordinate, Y_coordinate, monitor, additional_parameters*)**

**Parameters**

***trend_name***

The file name with a trend definition. Entering the "?" character opens the window in which the user may select a trend to be displayed.

Using the trend file name it is possible to specify the variables that will be added to the chart of variables declared in the definition of the trend. If the variable names are separated by the character #, the variables will be added to the trend. If the variable names are separated by the character <, the variables will replace the ones included in the definition of the trend.

***start_time***

Specifying the start time for the trend displayed. This parameter may be an absolute time value in the form of yyyymmddhhnnss, sometimes in the OPC format or may be the process variable name. In a situation where the parameter is the name of a variable, its value is interpreted as the number of seconds elapsed since 1/1/1970 for numeric variables, and as the time in the form of yyyymmddhhnnss or OPC format for text variables. It is also possible to specify the trend start and end time, separating them by a semicolon.

***X_coordinate***

This parameter specifies the X coordinate (top left corner) for the AsTrend module window. The value of -1 means that the relevant coordinate is loaded from the trend definition.

***Y_coordinate***

This parameter specifies the Y coordinate (top left corner) for the AsTrend module window. The value of -1 means that the relevant coordinate is loaded from the trend definition.

***monitor***

This parameter specifies the number of monitor on which the AsTrend module window should be opened. The value of 0 relates to the monitor on which the cursor is currently positioned, -1 relates to the monitor resulting from the x, y coordinates specified.

***additional_parameters***

Additional action parameters in compliance with the description of the Asix6 system *AsTrend* action.

 **See also**

- *AsTrendDisplay*, *AsTrendPrint* actions

- Asix.Evo_Application_Parameterization.PDF/CHM, *7. Operation with AsTrend Module*

## 4.14 AstrendDisplay

**Intended use**

This action opens the AsTrend application window displaying the trend selected. The action may include the trend template and start time. Also the names of the trend variables may be optionally included.

**Syntax**

**AstrendDisplay (*trend_name, start_time, X_coordinate, Y_coordinate, monitor*)**
**AstrendDisplay (*trend_name, start_time, X_coordinate, Y_coordinate, monitor*)**

**Parameters**

***trend_name***

The file name with a trend definition. Entering the "?" character opens the window in which the user may select a trend to be displayed.

After the trend file name it is possible to specify the variables that will be added to the chart of variables declared in the definition of the trend. If the variable names are separated by the character #, the variables will be added to the trend. If the variable names are separated by the character <, the variables will replace the ones included in the definition of the trend.

***start_time***

Specifying the start time for the trend displayed. This parameter may be the *DateTime* value, an absolute time value in the form of *yyyymmddhhnnss*, time value in the OPC format or may be the process variable name. In a situation where the parameter is the name of a variable, its value is interpreted as the number of seconds elapsed since 1/1/1970 for numeric variables, and as the time in the form of yyyymmddhhnnss or OPC format for text variables.

***end_time***

Specifying the end time for the trend displayed. Specified in the same way as for the start time. If the parameter is not used, the length of the trend period results from the period defined in the trend definition file.

***X_coordinate***

This parameter specifies the X coordinate (top left corner) for the AsTrend module window. The value of -1 means that the relevant coordinate is loaded from the trend definition.

***Y_coordinate***

This parameter specifies the Y coordinate (top left corner) for the AsTrend module window. The value of -1 means that the relevant coordinate is loaded from the trend definition.

***monitor***

This parameter specifies the number of monitor on which the AsTrend module window should be opened. The value of 0 relates to the monitor on which the cursor is currently positioned, -1 relates to the monitor resulting from the x, y coordinates specified.

**Examples**

AstrendDisplay (“tr1.trnx#z1#z2”, start_variable ,””,””,””)

This actions displays the *tr1.trnx* trend. The variables declared in the trend file will be complemented with variables *z1* and *z2*. The trend start time will be set based on the current value of the variable start_variable.

 **See also**

- *AsTrendPrint* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *7. Operation with AsTrend Module*

## 4.15 AstrendPrint

**Intended use**

This action prints the trend selected with the AsTrend module displaying this trend. The action may include the trend template and start time. Also the names of variables displayed in the trend may be optionally included.

**Syntax**

**AstrendPrint (*trend_name, start_time, title, header, footer*)**

**AstrendPrint (*trend_name, start_time, end_time, title, header, footer)***

**Parameters**

***trend_name***

The file name with a trend definition. Entering the "?" character opens the window in which the user may select a trend to be displayed.

Using the trend file name it is possible to specify the variables that will be added to the chart of variables declared in the definition of the trend. If the variable names are separated by the character #, the variables will be added to the trend. If the variable names are separated by the character <, the variables will replace the ones included in the definition of the trend.

***start_time***

Specifying the start time for the trend displayed. This parameter may be the *DateTime* value, an absolute time value in the form of *yyyymmddhhnnss*, time in the OPC format or may be the process variable name. In a situation where the parameter is the name of a variable, its value is interpreted as the number of seconds elapsed since 1/1/1970 for numeric variables, and as the time in the form of yyyymmddhhnnss or OPC format for text variables.

***end_time***

Specifying the end time for the trend displayed. Specified in the same way as for the start time. If the parameter is not used, the length of the trend period of time results from the period defined in the trend definition file.

***title***

This parameter specifies the new printout title, replacing the one included in the trend file.

***header***

This parameter specifies the new printout header, replacing the one included in the trend file.

***footer***

This parameter specifies the new printout footer, replacing the one included in the trend file.

**Examples**

AstrendPrint (tr1.trnx, ”NOW-1H” ,””,””,””)

This action prints the *tr1.trnx* trend with the default title, header and footer settings. The trend for the last hour is printed.

 **See also**

- *AsTrendDisplay* action

- Asix.Evo_Application_Parameterization.PDF/CHM, *7. Operation with AsTrend Module*

## 4.16 Break

**Intended use**

This action allows to conditionally break the currently running complex action. If the tested expression is of *true* value, the action will be interrupted. This action may also be used as one of the components of the action called *Actions*.

**Syntax**

**Break (*condition*)**

**Parameters**

***condition***

Logical condition, the value of which determines whether to stop execution of the complex action.

**Examples**

Break (Variable(v3)==0)

SetVariable ( v2, 0)

SetVariable ( v1, 0)

The control operation will be executed only if the current value of the variable *v3* is not equal to 0.

 **See also**

*2 Complex Actions*, *Perform*, *Actions*

## 4.17 CancelControls

**Intended use**

This action allows to cancel the selected pending control operations.

Some objects on the synoptic diagrams may be used in a delayed control mode. The *CancelControls* action restores normal operation mode for these objects and the control operations will not be executed.

**Syntax**

**CancelControls(*object_name*)**

**Parameters**

***object_name***

This parameter specifies the name of the objects for which the control operations are to be cancelled. The name may consists of special characters of the name templates, i.e. '*' and '?' .

**The parameter may also be one of the following constants:**

**$All** – the action applies to all objects in all windows and diagrams.
**$AllVisible** – the action applies to all objects in all visible windows. The window is visible when it is not hidden behind any other window.
**$Window** – the action applies to all objects in all diagrams of the current window (resulting from the context of the action use).
**$Diagram** – the action applies to all objects of the current diagram (resulting from the context of the action use).

**Examples**

CancelControls($Diagram)

All pending control operations on the active diagram (resulting from the context of use) will be aborted.

 **See also**

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *6. Organization of Control Operations*

- *SendControls* action

## 4.18 ChangePassword

**Intended use**

This action displays a window used to change the password of the currently logged user. The password may also be changed in the object of *Authorization field* type embedded on the synoptic diagram.

**Syntax**

**ChangePassword()**

 **See also**

- Asix.Evo_Application_Parametrization.PDF/CHM, *5. Permissions system*

- AsixEvo_Objects.PDF/CHM, *Authorization box* object

## 4.19 CloseWindow

**Intended use**

This action closes the opened synoptic windows. Closing of window groups based on the compliance with the name template is possible. Closing of windows may be limited to the selected monitor.

**Syntax**

**CloseWindow (*window_name, all*)**

**CloseWindow (*window_name, all, monitor*)**

**Parameters**

***window_name***

This parameter specifies the name of windows concerned by the action. You can specify the exact name or name template containing special characters '*' and '?'. The following rules apply to window searching:

- if the name of the window is blank, only the window related to the context of use is closed - the window from which the *CloseWindow* action was executed.
- if the name is specified, window searching is carried out in the following order:

1 - context window
2 - active window
3 - remaining open windows

- The application start-up windows can be closed only when the exact name of the window is specified.

***all***

If this parameter is *false*, only the first window meeting the conditions defined by parameters *window_name* and *monitor* is closed. If this parameter is *true*, all windows meeting the conditions are closed.

***monitor***

This parameter specifies the monitor the windows of which are to be closed. The following values of the parameter are available:

-1 - the action applies to all monitors.

0 - the windows of the monitor with the cursor on it are closed.

> 0 - the monitor number in accordance with the system numbering scheme - the action applies only to the specified monitor.

**Examples**

CloseWindow ( ”A*”, true, 0 )

This action closes all windows the names of which begin with an "A" and are located on the monitor pointed by the cursor.

 **See also**

- *OpenWindow*, *OpenDiagram*, *MinimizeWindow* actions

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM*, 5. Rules for Opening and Closing Synoptic Windows*

## 4.20 ConfirmRole

**Intended use**

This action is used to check whether the user is assigned the role being checked. The action is intended for use in complex actions. Depending on the mode the permissions of the currently logged user are checked or the login window is displayed, which forces the user identification upon the action execution.

The forced authorisation mode enables executing a complex action on the permissions of another user instead of those of the currently logged one. This action never causes a change of the currently logged user.
In the case of incorrect authentication or if no role is designated to the user, execution of the subsequent actions within the complex action is aborted.
This action may also be used as one of the components of the action called *Actions*.

**Syntax**

**ConfirmRole(*role_name, verification_mode*)**

**Parameters**

***role_name***
**The role name, the assignment to which is verified.**

***verification_mode***
**This parameter specifies user permissions verification method. The following variants are possible:**

**$Always** - confirmation of identity is required each time. Request for a user ID and password will be issued even if the currently logged user is assigned to the role being checked.
**$Never** - confirmation of identity is never executed. This action checks only whether the user is assigned to the role being checked.
**$IfRequired –** confirmation of identity will be executed only when the currently logged user is not assigned to the given role.

**Examples**

ConfirmRole (“SuperOperator”,$Never)

SetVariable ( v2, 0)

SetVariable ( v1, 0)

The control operations within the complex action will be executed only if the currently logged user is assigned to the *SuperOperator* role. The login window will not appear.

ConfirmRole (“SuperOperator”,$Always)

SetVariable ( v2, 0)

SetVariable ( v1, 0)

Regardless of the permissions of the currently logged user, the user verification proceeds. Proceeding to the next part of the action requires entering the ID and password of the user that is assigned to the *SuperOperator* role.

 **See also**

- Asix.Evo_Application_Parameterization.PDF/CHM, *5. Permissions System*

- AsixEvo_Techniques_of_Designing_Diagrams, *6.5. Controlling Permissions*

## 4.21 EditPatterns

**Intended use**

Activates the model trend editor. The model trend editor activated with the *EditPatterns* action enables editing local curves only.

**Syntax**

**EditPatterns()**

## 4.22 EndAlarm

**Intended use**

This action notifies the alarm termination event. The alarm that is the object of the action must be part of an external or undefined strategy. An alternative method for the alarm termination is to use the *Cancel* function of *IAlarm* interface in the user script code.

**Syntax**

**EndAlarm(*domain_name, alarm_ID*)**

**EndAlarm(*alarm_ID, alarm_ID, parameter_1, …*)**

**Parameters**

***domain_name***

Name of the alarm domain related to the action. If the name is not provided, the action relates to the default domain (the first one).

***alarm_ID***

Alarm ID to be terminated.

***parameter_1, …, parameter_n***

Additional parameters are logged in the alarm log.

**See also**

- AsixEvo_System_of_Alarms.PDF/CHM, *Strategies of Alarm Detection*
- AsixEvo_Scripts.PDF/CHM, *IAlarm interface*
- *StartAlarm* action

## 4.23 ExcludeAlarm

**Intended use**

This action excludes the alarm from handling. If the alarm is excluded, subsequent events of the alarm start and end are ignored. The alarms may also be excluded with the use of the *Active Alarms Viewer* object.

**Syntax**

**ExcludeAlarm(*domain_name, alarm_ID*)**

**Parameters**

***domain_name***

Name of the alarm domain related to the action. If the name is blank, the action relates to the default domain (the first one).

***alarm_ID***

Alarm ID to be excluded.

 **See also**

- AsixEvo_System_of_Alarms.PDF/CHM

- AsixEvo_Objects.PDF/CHM, *Active Alarms Viewer object*

- *IncludeAlarm* action

## 4.24 Execute

**Intended use**

This action executes another action specified in the call parameter. The action allows executing the actions defined in the variable definition database or created according to the application condition. In the latter case, the *Execute* action should be used only if the action type is not known in advance.

**Syntax**

**Execute(*action*)**

**Parameters**

***action***

Dynamically created content of the action to be executed.

**Examples**

Execute( Attribute(*Action*) )

In the above example, the action contents is loaded from the attribute *Action* of the contextual variable (the main variable of the object for which the *Execute* action was executed in response to an event).

Execute ( ”SetVariable (v1,” + Variable(v1)==0?1:0 )

In this case, the *SetVariable* action is created dynamically, the contents of which depends on the variable *v1*. Although the structure is correct, the simpler form acting the same way is recommended:

SetVariable (v1,Variable(v1)==0?1:0)

Execute (Variable(v1)==0? ”SetVariable (v1,1)”:”Nothing()“)

Depending on the value of variable v1, the *SetVariable* or *Nothing* action is executed. The same effect can be obtained with the *Perform* action:

Perform (Variable(v1)==0, SetVariable (v1,1), Nothing())

 **See also**

*Nothing*, *Perform* actions

## 4.25 IncludeAlarm

**Intended use**

This action resumes handling of a previously excluded alarm. The alarms may also be excluded from handling with the use of the *Active Alarms Viewer* object.

**Syntax**

**IncludeAlarm(*domain_name, alarm_ID*)**

**Parameters**

***domain_name***

Name of the alarm domain related to the action. If the name is blank, the action relates to the default domain (the first one).

***alarm_ID***

Alarm ID with handling to be resumed.

 **See also**

- AsixEvo_System_of_Alarms.PDF/CHM
- AsixEvo_Objects.PDF/CHM, *Active Alarm Viewer object*
- *ExcludeAlarm* action

## 4.26 Login

**Intended use**

This action displays a window used to log on the user. Other available login methods include:

- The window of the control panel that can be opened in any application with the use of *Ctrl-Shift-F1* keys
- In the object of *Authorization box* type embedded on the synoptic diagram.

**Syntax**

**Login()**

 **See also**

- AsixEvo_Objects.PDF/CHM, *Authorization box object*

- Asix.Evo_Application_Parameterization, *5. Permissions System*

## 4.27 Logout

**Intended use**

This acion logs current user out. Other available logout methods include:

- The window of the control panel that can be opened in any application with the use of *Ctrl-Shift-F1* keys
- In the object of *Authorization box* type embedded on the synoptic diagram.

**Syntax**

**Logout()**

 **See also**

**-** AsixEvo_Objects.PDF/CHM, *Authorization box* object

- Asix.Evo_Application_Parameterization, *5. Permissions system*

## 4.28 Message

**Intended use**

This action allows recording any message into the event log. The recorded messages are displayed in the message panel and in the objects of *Log Viewer* type.

**Syntax**

**Message(*message_typ, message_source, message_content*)**

**Parameters**

**message_type**

This parameter specifies the message priority. The following values are available:

**$Info – information message**

**$Error** – error message

**$Warning** – warning message

**message_source**

A short text identifying the source which has created the message.

**message_contents**

Any text included in the message.

**Examples**

**Message($Info, ”Test”, ”Action executed for the user ” + Variable(CurrentUser))**

The action records the information massage with information on the currently logged user into the *Log Viewer*.

 **See also**

- AsixEvo_Objects.PDF/CHM, *Log Viewer* object

## 4.29 MinimizeWindow

**Intended use**

This action minimizes the opened synoptic windows. Minimizing of window groups based on the compliance with the name template is possible.

**Syntax**

**MinimizeWindow(*window_name*)**

**Parameters**

***window_name***

This parameter specifies the name of windows concerned by the action. You can specify the exact name or name template containing special characters '*' and '?'. If the name of the window is blank, only the window related to the context of use is minimized - the window from which the *MinimizeWindow* action was executed.

**Examples**

**MinimizeWindow("")**

This action minimizes the window related to the context of the action use.

 **See also**

- *OpenWindow*, *CloseWindow*, *OpenDiagram* actions

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *Rules for Opening and Closing Synoptic Windows*

## 4.30 Nothing

**Intended use**

This action does not perform any activities. Designed as a blank alternative in structures using the *Perform* and *Execute* actions.

**Syntax**

**Nothing()**

**Examples**

Perform ( HasRole(”SuperOperator”), SetVariable ( v, 0), Nothing())

If the logged-in user is assigned to the *SuperOperator* role, the control operation is executed. If not, then nothing will be done.

Execute( Attribute(Action) )

In the above example, the contents of the action is taken from the *Action* attribute of the contextual variable. If no operation should be performed for some variables, the *Noting* action must be used in the variable definition database.

 **See also**

*Execute*, *Perform* actions

## 4.31 OpenDiagram

**Intended use**

This action allows to open the diagram in a dynamically, newly created window. If the diagram with the same name and identical set of parameters is already opened, the activation of a previously created window just occurs.

**Syntax**

**OpenDiagram (*diagram_name, parameters, title, opening_mode, X_coordinate, Y_coordinate, attributes, replacement_mode*)**

**Parameters**

***diagram_name***

This parameter specifies the name of the diagram to be displayed.

***parameters***

This parameter transfers the relevant parameter data to the diagram being opened. The set of parameters allowed depends on the definition of the diagram. The parameters are given as a sequence of pairs *parameter_name = parameter_value* separated by a semicolon, e.g.

*Colour*=Red;*Variable*=v1

***title***

This parameter specifies the text to be displayed on the window title bar.

***opening_mode***

This parameter specifies the behaviour of the window after opening. The possible values include one of the below:

**$Normal** – standard window
**$ Dialog** - dialogue box, switching to other open windows is disabled until the dialogue box is closed
**$ Temp** - temporary window, upon switching to any other window, the temporary windows will be closed

***X_coordinate***

This parameter specifies the window X coordinate (top left corner). The parameter may be a numerical value or one of the following constants:

**&CursorLocation** – X coordinate defines the current position of the cursor. The coordinate may be modified in order to prevent the window being opened from moving outside the monitor screen area.

**& ActiveLocation** - X coordinate of the window being opened is the same as the X coordinate of the window that was active upon the action execution.

***Y_coordinate***

This parameter specifies the X coordinate of the window being created (top left corner). The rules for providing the parameter are the same as for the *X_coordinate* parameter.

***attributes***

This parameter defines the layout of the window being created. The value of this parameter is a text string consisting of the following attribute names separated by a semicolon.

**Noframe** – the window will have no frame.
**NoTitleBar–** the window will have no title bar.
**NoButtons** – the window title bar will not display the system buttons.
**NoMinButton** – the title bar will not display the minimize button.
**NoMaxButton–** the title bar will not display the maximize button.
**FixedSize** – the window size will be fixed.
**ToolWindow** – the window layout will be similar to the system tool window (amongst others, the title bar height will be smaller).
**TopMost** – the window will be displayed as "always on top", i.e. the window will always be on the top of standard windows.
**NoTaskBar** – the window will not be displayed on the system task bar.
**Maximized** – the window will be displayed in full screen mode, according to the x and y coordinates.
**Center** - the window will be positioned in the centre of the monitor screen, according to the x and y coordinates.

***replacement_mode***

This parameter determines the additional actions which are undertaken when the window is opened. The possible values are one of the below constants:

**$ None** – no additional operations are to be performed.
**$ Closable** – all opened windows which have no closing interlock and are located on the screen of the newly opened window will be closed.
$ ExceptCurrent – all opened windows which have no closing interlock and are located on the screen of the newly opened window will be closed, except the window being active upon the action execution.

**Examples**

OpenDiagram ( diag1,”room=11”, null, $Dialog, $CursorLocation, $CursorLocation, “NoTitleBar,FixedSize”, $None )

This action opens the *diag1* diagram inside the new window and transfers the *room* parameter value into it. This window is of dialogue type and its position is determined by the current cursor position. The title bar is not shown, and the window cannot change its size. Every other window of the application stays opened.

 **See also**

- *OpenWindow*, *CloseWindow*, *MinimizeWindow* actions

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, 5. *Rules for Opening and Closing Synoptic Windows*

## 4.32 OpenWindow

**Intended use**

This actions is used to open predefined synoptic windows or to replace the diagrams displayed in the window panels. This is the basic action used to create a navigation scheme for the application.

**Syntax**

**OpenWindow (*window name, panel_name, diagram_name, parameters, new_window, X_coordinate, Y, replacement_mode*)**

**OpenWindow (*window name, panel_name, diagram_name, parameters, new_window*)**

**OpenWindow (*window name, panel_name, diagram_name, parameters*)**

**Parameters**

***window_name***

The name of the window associated with the action. It may be blank - then the window being the action object is determined by the context of use. For details, see Section *Algorithm*.

***panel_name***

The panel name, inside of which the diagram is to be opened. It may be blank - then the panel being the action object is determined by the context of use. For details, see Section *Algorithm*.

***diagram_name***

The name of the diagram associated with the action. It may be blank - then the action only opens the window leaving the diagram unchanged or opens the window with the default panel diagrams. For details, see Section *Algorithm*.

***parameters***

This parameter transfers the relevant parameter values to the diagram being opened. The set of parameters allowed depends on the definition of the diagram. The parameters are given as a sequence of pairs *parameter_name = parameter_value* separated by a semicolon, e.g.

*Colour*=Red;*Variable*=v1

***new_window***

The boolean *true/false* parameter which determines whether the new window should be created or not. For details, see Section *Algorithm*.

***X_coordinate***

This parameter specifies the X coordinate of the window being opened (top left corner). The parameter may be a numerical value or one of the following constants:

**$ Default** - the X coordinate results from the window definition or (for a window already opened) it is the current position. This is the default value when the action without *X_coordinate* parameter is used.
**&CursorLocation** – X coordinate defines the current position of the cursor. The coordinate may be modified in order to prevent the window being opened from moving outside the monitor screen area.
**& ActiveLocation** - X coordinate of the window being opened is the same as the X coordinate of the window that was active upon the action execution.

***Y_coordinate***

This parameter specifies the Y coordinate of the window being opened (top left corner). The rules for providing the parameter are the same as for the *X_coordinate* parameter.

***replacement_mode***

This parameter determines the additional actions which are undertaken when the window is opened. The possible values include one of the below:

**$ None** – no additional operations are to be performed (default value).
**$ Closable** – all opened windows which are not blocked against closing and are located on the screen of the newly opened window will be closed.
$ ExceptCurrent – all opened windows which are not blocked against closing and are located on the screen of the newly opened window will be closed, except the window being active upon the action execution.

**Algorithm**

1. If the diagram name is not specified:
   - If the window name is not specified either, the action object is the window associated with the context of use or the current window.
   - If the window name is specified and the *new_window* parameter is equal to *true*, the new window of the name specified including default diagrams is opened.
   - If the window name is specified and the *new_window* parameter is equal to *false*, the window of the name specified is searched between the currently opened windows (starting from the active one). If such a window is found, then it becomes the object of the action. If not, the new window of the name specified including default diagrams is opened.
2. If the diagram name is specified:

- If the window name is not specified, the specified diagram is opened inside the context window (according to the place of the *OpenWindow* action execution) and inside the specified panel (or context panel).
- If the window name is specified, it is checked whether the window with already opened diagram specified in the action call, with identical parameter set (in the order of search: context window, current window, other windows) is amongst the currently opened windows . If such a window is found, then it becomes the active window.
- If in the procedure described above the relevant window is not found, and the *new_window* parameter is equal to *false*, the window with the specified name is searched amongst the currently opened windows (in the order of search: context window, current window, other windows). If such a window is found, the specified diagram is opened inside the panel indicated by the action contents. If the window is not found or the *new_window* parameter is equal to *true*, the new window of the name specified is opened. The specified diagram is opened inside the panel indicated by the action contents, and the default diagrams are opened inside other panels.

**3. In any case, if the window being the action object is determined successfully, this windows becomes the active one, suitable correction of the window position is carried out, if necessary, and the windows indicated by the *replacement_mode* parameter are closed.**

**Examples**

OpenWindow (start, scheme, engine, variable=s1)

If the window named *start* is already opened, the diagram *engine* along with the *variable* = *s1* parameters will be opened inside the *scheme* panel. If such a diagram with the same parameters is already opened, only window activation occurs.

If the window called *start* is not already opened, it will be opened now. The diagram *engine* along with the *variable* = s1 parameters will be opened inside the *scheme* panel. The default diagrams will be opened inside the other panels.

OpenWindow ( null, scheme, engine, variable=s1, false, $Default, $Default, $Closable)

The diagram *engine* along with the *variable* = s1 parameters will be opened inside the *scheme* panel of the context window. The window position remains unchanged. In addition, all windows not blocked against closing will be closed.

**See also**

- *OpenDiagram*, *CloseWindow*, *MinimizeWindow* actions

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *5. Rules for Opening and Closing Synoptic Windows*

## 4.33 Perform

**Intended use**

This action enables checking the logical condition, and depending on its value, executing one of the two actions. The *Perform* action in some cases can replace the complex actions associated with the *Break* structure.

**Syntax**

**Perform(*condition, action_if_true, action_if_false*)**

**Parameters**

***condition***

A conditional expression, the value of which decides which of component actions is to be executed.

***action_if_true***

The action to be executed if the *condition* parameter is *true*.

***action_if_false***

The action to be executed if the *condition* parameter is *false*.

**Examples**

**Perform ( Variable(ActiveZone)==1, SetVariable(Zone1,0), PlaySound(MustNot) )**

If the *ActiveZone* variable is equal to 1, the *Zone1* variable will be set to 0. Otherwise the sound called *MustNot* will be played.

 **See also**

-*2 Complex Actions*, *Break* action

## 4.34 PlaySound

**Intended use**

The action plays the selected sound signal once. The sound signal must be previously added to the application sound set.

**Syntax**

**PlaySound(*sound_name*)**

**Parameters**

***sound_name***

The name of the sound signal to be played.

**Examples**

**PlaySound(alert)**

This action plays once a sound file declared in the sound set under the name *alert*.

 **See also**

- AsixEvo_System_of_Alarms.PDF/CHM, *Sound Signaling*

- *PlaySoundLooping*, *StopSound* actions

## 4.35 PlaySoundLooping

**Intended use**

The action plays the selected sound signal in loop mode. The sound signal is played in a loop until the *StopSound* action is executed or another sound signal starts to play.

The sound signal must be previously added to the application sound set.

**Syntax**

**PlaySoundLooping(*sound_name*)**

**Parameters**

***sound_name***

The name of the sound signal to be played.

**Examples**

**PlaySoundLooping(alert)**

This action plays in a loop mode a sound file declared in the sound set under the name *alert*.

 **See also**

- AsixEvo_System_of_Alarms.PDF/CHM, *Sound Signaling*

- *PlaySound*, *StopSound* actions

## 4.36 PrintScreen

**Intended use**

This action captures a picture of a selected area of the screen. It is possible to capture the active window or diagram. It is also possible to capture full screen. The captured picture is copied to the system clipboard or saved to a file. If the picture is saved to a file, the file format is *png* and it is saved in the *Screenshots* subdirectory of the application current working directory. The file name is generated automatically based on the current time and type of the picture captured.

**Syntax**

**PrintScreen (*area, save_to_file*)**

**Parameters**

***area***

This parameter defines the screen area to be captured. The following constants are available:

**$ Diagram** - an active diagram area will be captured.
**$ Screen** - a full area of the screen on which cursor is located will be captured.
**$ Window** - an active window area will be captured.

***save_to_file***

The boolean *true/false* parameter, which determines whether the picture should be saved to a file or not. The *false* value means that the picture will be copied only to the system clipboard.

**Examples**

```
PrintScreen($Window, true)
```

The captured picture of the active window area will be saved to a file in the *Screenshots* subdirectory of the application current working directory.

## 4.37 RefreshVariable

**Intended use**

This action enables forcing the variable value readout beyond the period of its planned cyclic refreshing according to the definition.

**Syntax**

**RefreshVariable(*variable_name*)**

**Parameters**

***variable_name***

The name of the process variable the value of which is to be refreshed. A blank name indicates the use of the main variable of the object related to the context of use. Suffix notation of the variable name in relation to the main variable of the context object is possible.

**Examples**

RefreshVariable (””)

This action forces refreshing the main variable value of the object for which the action is executed.

## 4.38 Run

**Intended use**

This action is used to run an external application. The program directly specified in the parameters may be run or system association of programs with file name extensions may be used.

**Syntax**

**Run(*file_name*)**

**Run(*file_name, arguments*)**

Parameters

***file_name***
**This parameter specifies the program to be run. This may be an exe file name or data file name. In the second case, the program associated within an operating system with the data file type will be run.**

**Searching procedure for an exe program is based on the system program path.**

***arguments***
**Text string of arguments transferred to the program being run. The format and meaning of the parameter is specific to the program.**

**Examples**

Run(”pdffiles\plan.pdf”)

The pdf file browser declared within the system will be launched and *plan.pdf* file from the *pdffiles* subdirectory of the application directory will be opened.

## 4.39 Script

**Intended use**

This action is used to run a user script.

**Syntax**

**Script(*name, parameter1, parameter2, ...*)**

**Parameters**

***name***

User script name.

***parameter1, …, parameterN***

Values of script parameters. The number and form of the parameters used in the call must be in accordance with the requirements of the script.

**Examples**

Script( CalculateConsumption, v1 )

The action runs a script named *CalculateConsumption*, transferring to it a single parameter of the value of *v1*.

 **See also**

- AsixEvo_Scripts.PDF/CHM

## 4.40 SecurityBrowser

**Intended use**

This action opens a dialogue box for viewing log files collected by the security system. The use of the action makes sense only if the security system is operating in the central mode. The access to the security system logs may be restricted according to the user permissions and the Asix system licence held.

**Syntax**

**SecuritBrowser(*log_type*)**

**Parameters**

***log_type***

This parameter specifies the log type, the contents of which must be shown at the window opening. The user can interactively switch to other types of logs.

**The parameter may also be one of the following constants:**

**$ Messages** - switch to the log of the security system messages
**$ Controls** - switch to the log of the completed control operations
**$ actions** - switch to the log of the operations carried out by the operator
**$Notes** - switch to the log of the operator's notes

 **See also**

- Asix.Evo_PApplication_Parameterization.PDF/CHM, *5. Permissions System*

## 4.41 SecurityManager

**Intended use**

This action opens a dialogue box for managing user permissions while the application is running.

**Syntax**

**SecurityManager()**

 **See also**

- Asix.Evo_Application_Parameterization.PDF/CHM, *5. Permissions System*

## 4.42 SendControls

**Intended use**

Some objects on the synoptic diagrams may be used in a delayed control mode. The *SendControls* action makes these objects perform pending control operations.

**Syntax**

**SendControls(*object_name*)**

**Parameters**

***object_name***

This parameter specifies the name of the objects for which control operations are to be performed. The name may consists of special characters of the name templates, i.e. '*' and '?' .

**The parameter may also be one of the following constants:**

**$All** – the action applies to all objects in all windows and diagrams.
**$AllVisible** – the action applies to all objects in all visible windows. The window is visible when it is not hidden behind any other window.
**$Window** – the action applies to all objects in all diagrams of the current window (resulting from the context of the action use).
**$Diagram** – the action applies to all objects of the current diagram (resulting from the context of the action use).

**Examples**

SendControls($Window)

All pending control operations on the active window (resulting from the context of use) will be performed.

 **See also**

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *6. Organization of Control Operations*

- *CancelControls* action

## 4.43 SetBounds

**Intended use**

This action allows to change the location and size of the selected element of the diagram. It is used for move and size change animation of the objects on the diagram. More information and examples of use may be found in the *SetPosition* action description.

**Syntax**

**SetBounds (*object_name, X_coordinate, Y_coordinate, width, height*)**

**SetBounds (*X_coordinate, Y_coordinate, width, height*)**

**Parameters**

***object_name***

The name of the element, the location and size of which is to be changed. The action variant containing no name relates to a change of the context element (object, template or group) location and size. The parameter can contain the characters '*' and '?'.

If the action has been used in the context of the object being part of the template, the searching the objects relating to the name specified applies only to objects of this template. Reference to the object which is not part of the template requires adding the prefix '!' to the object name.

***X_coordinate***

The new value of the controlled element X coordinate expressed in absolute units.

***Y_coordinate***

The new value of the controlled element Y coordinate expressed in absolute units.

***width***

The new width of the controlled element expressed in absolute units.

***height***

The new height of the controlled element expressed in absolute units.

 **See also**

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *Motion Animation and Resizing of Objects*

- *SetPosition*, *SetSize* actions

## 4.44 SetPosition

**Intended use**

This action allows to change the location and size of the selected element of the diagram. It is used for motion animation of the objects on the diagram. This action is typically used within the *Animation* event handling for the object, template or group. The position parameters transferred to the function must be expressed in absolute coordinates (0 to 1 000 000, absolute coordinate 1 000 000 corresponds to the right edge of the diagram). If relative coordinates (pixels) are used, the appropriate conversion has to be done with the use of the *RelToAbsX* and *RelToAbsY* functions.

**Syntax**

**SetPosition (*object_name, X_coordinate, Y_coordinate*)**

**SetPosition (*X_coordinate, Y_coordinate*)**

**Parameters**

***object_name***

The name of the element, the position of which is to be changed. The action variant containing no name relates to the change of the context element (object, template or group) position. The parameter can contain the characters '*' and '?'.

If the action has been used in the context of the object being part of the template, the searching of the objects relating to the name specified applies only to objects of this template. Reference to the object which is not part of the template requires adding the prefix '!' to the object name.

***X_coordinate***

The new value of the controlled element X coordinate expressed in absolute units.

***Y_coordinate***

The new value of the controlled element Y coordinate expressed in absolute units.

**Example**

```
SetPosition( RelToAbsX(100+Variable(pos)),"")
```

Only the X coordinate of the diagram element is changed. The variable *pos* contains the position data in form of pixel coordinates. After adding the offset of position (value 100 that determines change of the X coordinate of the diagram element position), the conversion to absolute coordinates is performed.

 **See also**

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *Motion Animation and Resizing of Objects*

- *SetBounds*, *SetSize* actions

- AsixEvo_Expressions_and_Functions.PDF/CHM, *RelToAbsX* function

- AsixEvo_Expressions_and_Functions.PDF/CHM, *RelToAbsY* function

## 4.45 SetProperty

**Intended use**

The action allows setting of any property of the selected object or diagram. This allows to control the appearance/ behaviour of the application.

**Syntax**

**SetProperty (*object_name, property_name, value*)**

**SetProperty (*property_name, value*)**

**Parameters**

***object_name***

The name of the object, the property of which is to be changed. If the name is not specified, the action applies to the context element (object, template, group or diagram) property change. The parameter can contain the characters '*' and '?'.

If the action has been used in the context of the object being part of the template, the searching the objects relating to the name specified applies only to objects of this template. Reference to the object which is not part of the template requires adding the prefix '!' to the object name.

***property_name***

The name of the property to be changed. The names of internal properties should be used. The simplest way to insert the correct name of the property into the action contents is to use the *Local Properties* command from the action editor menu.

***value***

The new value of the property being changed

**Examples**

SetProperty(Temp,MainVar,block1Temp)

This action changes the main variable of the object named *Temp* to *block1Temp*. This results in switching the object to display the status of the new process variable.

## 4.46 SetSize

**Intended use**

This action allows to change the size of the selected element of the diagram. It is used for size animation of the objects on the diagram. More information and examples of use may be found in the *SetPosition* action description.

**Syntax**

**SetSize (*object_name, width, height*)**

**SetSize (*width, height*)**

**Parameters**

***object_name***

The name of the element, the size of which is to be changed. The action variant containing no name relates to the change of the context element (object, template or group) size. The parameter can contain the characters '*' and '?'.

If the action has been used in the context of the object being part of the template, the searching of the objects relating to the name specified applies only to objects of this template. Reference to the object which is not part of the template requires adding the prefix '!' to the object name.

***width***

The new width of the controlled element expressed in absolute units.

***height***

The new height of the controlled element expressed in absolute units.

 **See also**

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *Motion Animation and Resizing of Objects*

- *SetPosition*, *SetBounds* actions

## 4.47 SetVariable

**Intended use**

This action enables direct sending the control value to the selected process variable.

The action execution is controlled by the authorisation system.

**Syntax**

**SetVariable(*variable_name, value*)**
**SetVariable(*variable_name, value, value_mask*)**

**Parameters**

***variable_name***

The name of the process variable which is to be provided with control. A blank name indicates the use of the controlled variable of the object related to the context of use. Suffix notation of the variable name according to the context object main variable is possible.

***value***

The control value sent to the variable. In case of the action variant containing no *value_mask* parameter, the value specified is sent without any change. The second variant means that the value is modified according to the algorithm specified in the description of the *value_mask* parameter.

***value_mask***

If this parameter is used, the following operations are performed during the action execution:

- the current value of the variable is read out;
- only those bits which correspond to the bits of value of 1 in the *value_mask* are changed in the readout value;
- a value modified this way is sent to the variable.

**Examples**

SetVariable (””,1)

Controlled variable of the object for which the action is executed will be set to the value of 1.

SetVariable(”#_S”,8,0xc)

A variable the name of which is a combination of the main name of the object (for which the action is executed) and the suffix "_S" will be set. The third bit of the variable will be set to 1 and the second one will be set to 0. The other bits remain unchanged. For example, if the

original value of the variable was equal to 0xff, upon the control execution it will be equal to 0xfb.

 **See also**

- *AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM,* 6. Organization of Control Operations

## 4.48 SetWindowSize

**Intended use**

This action changes the size of synoptic windows. Changing the size of window groups based on the compliance with the name template is possible.

This action is an alternative to manual resizing of the window carried out by the user. The user can resize the window which is not blocked and has a frame. It should be noted that window resizing changes the size of the panels inside the window. Diagrams may also be scaled to a new size.

**Syntax**

**SetWindowSize(*window_name, width, height*)**

**Parameters**

***window_name***

This parameter specifies the name of windows concerned by the action. You can specify the exact name or name template containing special characters '*' and '?'. If the name of the window is blank, only the window related to the context of use is resized - the window from which the *SetWindowSize* action was executed.

***Width***

The new window width.

***Height***

The new window height.

**Examples**

SetWindowsSize (null,400,200)

The action changes the context window size.

 **See also**

*- AsixEvo_Techniques_of_Designing_Diagrams*.PDF/CHM, *4.2. Controlling the Location and Size of Windows*

## 4.49 ShowControlPanel

**Intended use**

This action is used to open the control panel window. The control panel is a predefined window which enables performing the basic administrative tasks.

The window of the control panel may be opened in any application using *Ctrl-Shift-F1* keys.

**Syntax**

**ShowControlPanel()**

## 4.50 ShowKeyboard

**Intended use**

This action opens the window with an on-screen alphanumeric keyboard.

Syntax

**ShowKeyboard()**

 See also

- *ShowNumericKeyboard* action

## 4.51 ShowMenu

**Intended use**

This action opens the selected context menu. The menu is opened at the cursor position. The menus fixed to the synoptic windows are an alternative to the context menu.

Syntax

**ShowMenu(*menu_name, parameters*)**

**Parameters**

***menu_name***

The name of the menu defined in the application.

***parameters***

This parameter transfers the relevant parameter data to the menu being opened. The set of parameters allowed depends on the definition of the menu. The parameters are given as a sequence of pairs *parameter_name = parameter_value* separated by a semicolon, e.g.

*Colour*=Red;*Variable*=v1

 **See also**

- AsixEvo_Visualization_Elements.PDF/CHM, *5. Menu*

## 4.52 ShowNumericKeyboard

**Intended use**

This action opens the window with an on-screen numeric keyboard.

Syntax

**ShowNumericKeyboard()**

 See also

- *ShowKeyboard* action

## 4.53 StartAlarm

**Intended use**

This action notifies the alarm start event. The alarm that is the object of the action must be part of an external or undefined strategy. An alternative method for the alarm start notification is to use the R*aise* function of IAlarm interface in the user's script code.

**Syntax**

**StartAlarm(*domain_name, alarm_ID*)**

**StartAlarm(*domain_name, alarm_ID, parameter_1,…*)**

**Parameters**

***domain_name***

Name of the alarm domain related to the action. If the name is not provided, the action relates to the default domain (the first one).

***alarm_ID***

Alarm ID to be initiated.

***parameter_1, …, parameter_n***

Additional parameters logged in the alarm log.

 **See also**

- Asix.Evo_Alarm_System.PDF/CHM

- Asix.Evo_Alarm_System.PDF/CHM, *Strategies of Alarm Detection*

- AsixEvo_Scripts.PDF/CHM, *IAlarm interface*

- *EndAlarm* action

## 4.54 StopSound

**Intended use**

This action stops the currently played sound signal. With the alarm domain name specified, the action mutes alarm sound signal on the station where the action was executed and on the other stations involved in the domain operation.

**Syntax**

**StopSound()**

**StopSound(*domain_name*)**

**Parameters**

***domain_name***

Name of the alarm domain related to the action. The name in the form of '*' may be specified, which turns the alarm off for all alarm domains.

 **See also**

- Asix.Evo_Alarm_System.PDF/CHM, *Sound Signaling*

- *PlaySound*, *PlaySoundLooping* actions

## 4.55 SwitchLanguage

**Intended use**

This action enables switching the interface language. The current application language and the programme language associated with it are changed. The application must be configured for multilingual operation.

The application language may also be changed via the control panel window. The window of the control panel may be opened in any application using *Ctrl-Shift-F1* keys.

**Syntax**

**SwitchLanguage(*language_code*)**

**Parameters**

***language_code***

Language code containing two letters.

**Examples**

**SwitchLanguage(”en”)**

This action switches the application into English.

 **See also**

- AsixEvo_Application_Parameterization.PDF/CHM, *4. Parameterization of Multilingual Applications*

## 4.56 SwitchWorkMode

**Intended use**

The action allows switching the application run mode to the selected edit mode.

The action execution is controlled by the authorisation system. To switch to the edit mode, the user must be logged in and must have the access permissions for the relevant edit mode.

The operation mode may also be changed via the control panel window. The window of the control panel may be opened in any application using *Ctrl-Shift-F1* keys.

**Syntax**

**SwitchWorkMode(*edit_mode*)**

**Parameters**

***edit_mode***

This parameter determines the desired edit mode. The parameter must be one of the following constants:

**$Architekt –** full edit mode along with the Asix.Evo.exe Architect window.

**$ WindowEdit** - simplified edit mode for synoptic windows.

**$ DiagramEdit** - simplified edit mode for synoptic diagrams.

**Examples**

SwitchWorkMode($DiagramEdit)

The action switches the application to the simplified edit mode for synoptic diagrams.

 **See also**

- AsixEvo_How_to_Do_It.PDF/CHM, *Starting the Program and Switching the Operation Modes /* Switching from the Run Mode to Edit Mode

- Asix.Evo_Application_Parameterization.PDF/CHM, *5. Permissions System*

## 4.57 Terminate

**Intended use**

This action closes the application. It works only when the application is started in the run mode from the beginning.

The action execution is controlled by the authorisation system. To exit the application, the user must be logged in and must have the permissions to close the application.

There are alternative methods for closing the application:

- Closing all opened windows.
- Using the operation termination button located on the control panel window. The window of the control panel may be opened in any application using *Ctrl-Shift-F1* keys

**Syntax**

**Terminate()**

 **See also**

- Asix.Evo_Application_Parameterization.PDF/CHM, *5. Permissions System*

## 4.58. ToggleBits

**Intended use**

The action allows changing the values of selected bits of the process variable to the opposite.

The action execution is controlled by the authorisation system.

**Syntax**

**ToggleBits(*varable_name, bit_mask*)**

**Parameters**

***variable_name***

The name of the process variable which is to be provided with control. A blank name indicates the use of the controlled variable of the object related to the context of use. Suffix notation of the variable name according to the context object main variable is possible.

***bit_mask***

This parameter defines the variable bits to be changed. Only those bits which correspond to the bits of value of 1 in the *bit_mask* are changed.

**Examples**

ToggleBits (””,0x1)

The least significant bit value of the controlled variable of the object on account of which the action is executed, will be set to the opposite value.

 **See also**

- AsixEvo_Techniques_of_Designing_Diagrams.PDF/CHM, *6. Organization of Control Operations*

- Asix.Evo_Application_Parameterization.PDF/CHM, *5. Permissions System*

## 4.59 VariableInfo

**Intended use**

This action allows displaying the information window related to the variable selected. It is possible to specify the attribute names whose values are to be shown.

**Syntax**

**VariableInfo(*variable_name, attribute_list*)**

**Parameters**

***variable_name***

The process variable name for which the information window is to be displayed. A blank name indicates the use of the main variable of the object related to the context of use. Suffix notation of the variable name according to the context object main variable is possible.

***attribute_list***

The list of variable attribute names separated by commas. This parameter determines which of the variable attributes are to be displayed in the information window. A blank attribute list means that all the attributes will be displayed.

The preferred defining method to determine the attribute list is to use the global property called *SetOfAttributes* This allows defining the list once, and then applying it in all calls to the *VariableInfo* action by using the *Property* function.

**Examples**

VariableInfo (””, Property(InfoAttrs))

An information window for a main contextual variable of the object will be displayed. Names of the displayed attributes will be loaded from the definition of the global property called *InfoAttrs*.