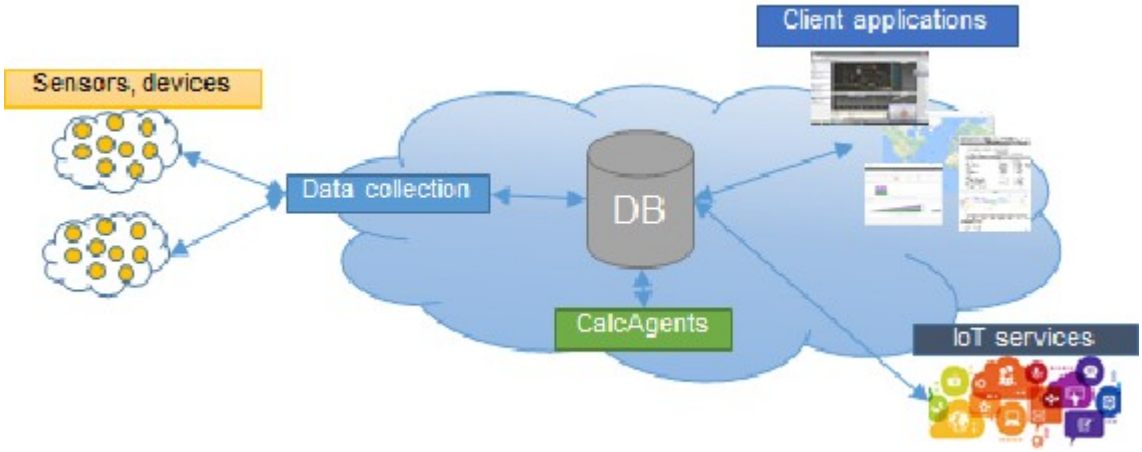
AutArch Manual

Table of Contents

1 Introduction

1.1 Overview

The AutArch (AA) system is a system for gathering of real time data from industrial process systems, other sensors and other general information into historical data to be used for analyze and optimization of processes, business improvements, KPI's and other evaluations.
The database is implemented in Microsoft SQL Server. The database consists of database tables, views, and stored procedures for handling storage and presentation of historical data and events.
The system consists of the database and several connecting applications communicating with the database.
The system has advanced functionality for compressed storage of historical data. It has built in functionality for efficient and fast presentation of data for analysis. The structure of data and availability of data from external applications are standardized and easy to access.
The data is stored in MS SQL Server standard time format with a time resolution of 3 ms.
In addition to standard built in functions in AutArch system including GUI framework there is a Business Intelligence tool packaged where enterprise data can be presented and analyzed in a even more flexible way. This platform is fully web based and adapted to mobile platforms.



1.2 Revision history

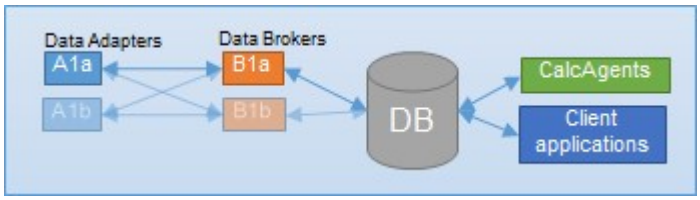
Date	Version	System Version	Description
2018-04-23		3.2.8	Made the manual printable, although incomplete. Added description of the SignalProp table.
2018-04-24		3.2.8	Made corrections to all existing database table descriptions. Added description of the MVSignalLog table.
2018-04-25		3.2.8	Made corrections to all existing database procedure descriptions. The database section is now up to date.
2018-04-26		3.2.8	Adjusted structure of entire manual. Added release notes and test documentation.
2018-04-27		3.2.8	Fixed documentation for XMZ Import/Export and all Workspace presentations.

2 Components

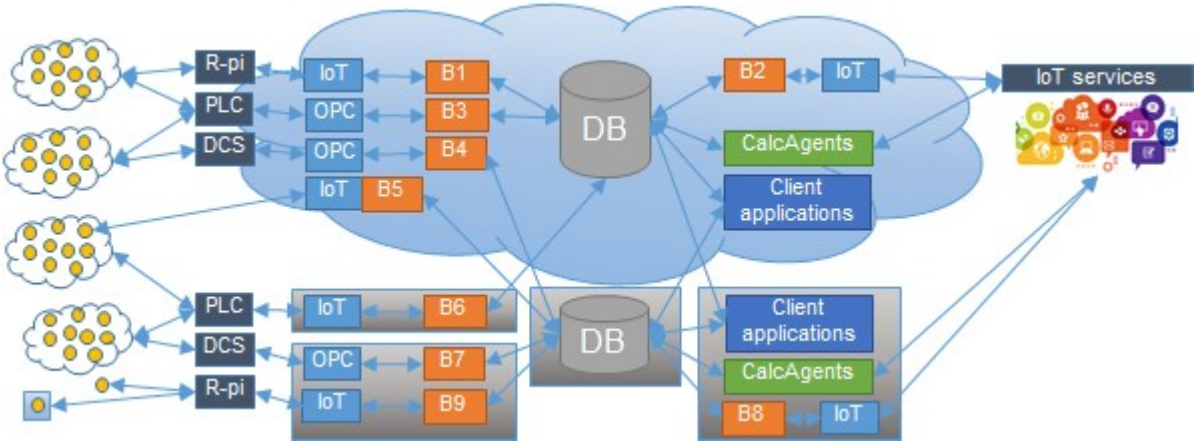
Following main structure components are used within an AutArch system setup

- ***Data adapters*** as external interfaces.
- ***Data brokers*** as distributed structure components to handle data transfer/processing in controlled manner.
- ***Database*** to store data
- ***Calc agents*** to process/calculate/evaluate data continuosly or batchwise.
- ***Client applications*** for data presentations, analyzing, reporting as well as for system supervision.

All components can be parallellised and distributed, locally stored, centralised. Cloud hosted or locally hosted. The configuration is selected completely dependent on customer system setup, where plant configurations, infrastructure, and system structure are determining parameters for system setup.



Picture shows different variants where components can be used either in local/distributed hosting or cloud hosted in any combinations.



2.1 Database

2.1.1 Overview

To be documented..

2.1.2 Date & time format

The dates and times are stored in the database in one of two ways:
- Using the local time zone of the database server's hosting computer
- Using UTC

Which of the ways the database uses can be checked by running the following query:

```
select CfgValue from GUICfgParam where CfgParam='UTC'
```

If the result from the above query is `true`, then UTC is used.
If the result is `false` or empty, local time zone is used.

While using local time zone has the advantage that it is more human readable, it also has the disadvantage of not handling daylight saving time switches very well. The switch from normal time to daylight saving time will interpolate the data during that hour, and the switch from daylight saving time to normal time will lose one hour of data!

2.1.3 Database tables

Tables	Description
SignalDef	Common signal definition table for signal types including events.
SignalProp	Extra properties on signals, in a key/value-style table.
ASignalDef	Signal definition table for analog signals with last added value.
ASignalLog	Compressed and archived analog values.
DSignalDef	Signal definition table for digital signals with last added value.
DSignalLog	Compressed and archived digital values.
MVSignalLog	Archived multi values.
CalcSignals	Relational table that holds information which signals are used in which calculated signals.
EventDef	Event definition table.
EventLog	Archived events.
Change	Top level table of changes made to the SignalLog tables.
ChangeSignal	Describes affected signals and periods.
ChangeASignalLog	Contains data removed from ASignalLog.
ChangeDSignalLog	Contains data removed from DSignalLog.
OPCServer	OPC Server definition and configuration table.
OPCTagGroup	OPC tag groups definition and configuration table.
DBPurgeStat	Configuration parameters for the scheduled DBPurge job.

UnitConvert	Definitions of conversion data between different engineering units.
TuneEvaluationLog	Result data from Compression tuning analyze.

2.1.3.1 SignalDef

Common signal definition table for signal types including events.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
SignalName	Allowed	A unique, user input, name for the signal. The SignalName can be changed. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
SignalDesc	Allowed	The description of the signal Valid for: Analog, Digital, Event Data type: nvarchar(255)
SignalType	Allowed	Bitmasked number to set the functionality of the tag. Bit: 1 – Digital signal 2 – Analog Signal 4 – Signal has events 8 – Text (for future use) 16 – Calculated 32 – Out 64 – Extend last value 128 – Signal has Multi values. Can be combined with Analog or Digital. 256 – Signal is periodic. The periodicty is set in the CalcFormula field. 512 – Disable recalc 1024 – Disable import 2048 – Disable correction 4096 – AutArch system diagnostic tag Valid for: Analog, Digital, Events, MultiValue Data type: smallint
CreationTime	Allowed	The CreationTime property is a timestamp when the signal was first created. The timestamp is generated by system and cannot be changed. Valid for: Analog, Digital, Events Data type: datetime
LogBlocking	Not allowed	The LogBlocking property blocks a signal from being logged

		to the database. When LogBlocking is activated or deactivated, the OPCDA application needs to be restarted to effectuate the changes. The OPCDA application excludes a blocked tag from scheduling. 0 – Signal is not blocked 1 – Signal is blocked, and no data is archived. Valid for: Analog, Digital Data type: bit
ValueName	Allowed	ValueName gives the possibility to use a description for the valuetype, e.g. “Power”. (For future use by connecting applications.) Valid for: Digital Data type: nvarchar(50)
ValueName0	Allowed	The ValueName0 property is used to provide a description for when signal is 0 (zero), such as “CLOSED” or “OFF”. Valid for: Digital Data type: nvarchar(50)
ValueName1	Allowed	The property is used to provide a description for when signal is 1 (one), such as “OPEN” or “ON”. Valid for: Digital Data type: nvarchar(50)
ValueUnit	Allowed	The unit for the signal, e.g m/s, kPa etc. Valid for: Analog Data type: nvarchar(50)
RangeMin	Allowed	RangeMin is used to set the default min range for the signal when opened in viewing applicatios, e.g. Trendviewer. Valid for: Analog Data type: float
RangeMax	Allowed	RangeMax is used to set the default max range for the signal when opened in viewing applications, e.g. Trendviewer. Valid for: Analog Data type: float
Interpol	Not allowed	If the Interpol property is set, the values are interpolated by straight line between each value. If the property is not set the values are plotted as steps. The Interpol property is only used with the return values from the Sig_Select_Calc and the Sig_Select_Serie procedures. 0 – No interpolation 1 – Interpolation Valid for: Analog, Digital Data type: bit

		More...
DisplayFormat	Allowed	The DisplayFormat property sets the number of decimals for viewing applications, such as Trendview application. The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted. For example, when the value 14059,42450 is used in different display formats, see following table. Valid for: Analog, Digital, Events Data type: nvarchar(20) More...
OPCGroupID	Allowed	A unique ID identifying the OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
OPCItemID	Allowed	The signals item id on the OPC server. Valid for: Analog, Digital Data type: nvarchar(1000)
Category1	Allowed	The category1 property is used for categorizing tags and filtering functions in viewing applications. Valid for: Analog, Digital Data type: nvarchar(20) More...
EventBlocking	Allowed	The EventBlocking property activates/inactivates events from being logged to the database. 0 = Event is not blocked 1 = Event is blocked and no data is archived. This attribute can be set anytime and changes take effect directly on archiving. Valid for: Events Data type: bit
CalcEnabled	Allowed	CalcEnabled property determines if calculation is enabled or not. 0 – Calculation not enabled. 1 – Calculation enabled. Valid for: Analog, Digital, MultiValue Data type: bit
CalcType	Allowed	Determines the type of calculation to execute for calculation signals.

		0 – a user written formula (CalcFormula property) 2 – time calculated by average value 4 – time calculated by maximum value 5 – time calculated by minimum value 10 – Python calculated 11 – Average periodic 12 – Sum periodic Valid for: Analog, Digital, MultiValue Data type: tinyint
CalcFormula	Allowed	The user written formula used for calculations. The formula calculation needs one or several input signals in the formula of which at least one input signal need to have the CalcTrigger set. The CalcTrigger triggers the calculation when the value of the signal changes. The signals that are used in the formula are stored in an relational table but they are edited by writing the signal names surrounded by "{" and "}" in the following way: `{SignalName1} + {SignalName2}` Valid for: Analog, Digital Data type: varchar(2000)
Exportflag	Allowed	The ExportFlag property is set to mark the signal for export. It is used by the AAExport application for exporting data to encrypted exportfiles. By setting combinations for this value, different export scope can be performed. 0 – no export 1 – signal export The flag is used in a bit coded way that makes it possible to flag a signal to be exported by different export jobs, se the AAExport documentation for examples. Valid for: Analog, Digital Data type: smallint
Custom01	Allowed	The Custom01 - 05 are free signal properties to be used by customer. Valid for: Analog, Digital Data type: nvarchar(100)
Custom02	Allowed	
Custom03	Allowed	
Custom04	Allowed	
Custom05	Allowed	
SignalName2	Allowed	The name of the signal in a alternative language. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
SignalDesc2	Allowed	The description of the signal in a alternative language. Valid for: Analog, Digital, Event Data type: nvarchar(255)
ValueUnit2	Allowed	The unit for the signal, e.g m/s, kPa etc. in a alternative

		language Valid for: Analog Data type: nvarchar(50)
RangeMin2	Allowed	RangeMin2 is used to set the default min range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit. Valid for: Analog Data type: float
RangeMax2	Alowed	RangeMax2 is used to set the default max range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit. Valid for: Analog Data type: float
DisplayFormat2	Allowed	The DisplayFormat2 property sets the number of decimals for viewing applications, such as Trendview application when using an alternative engineering unit. The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted. For example, when the value 14059,42450 is used in different display formats, see following table. Valid for: Analog, Digital, Events Data type: nvarchar(20) More...
TdrOPCGroupID	Allowed	A unique ID identifying the Trigged Data Recorder OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
TdrOPCItemID	Allowed	The signals item id on the OPC server for a Trigged Data Recorder signal. Valid for: Analog, Digital Data type: nvarchar(1000)
TdrPreBuffer	Allowed	The amount of data that will be saved to the database before the signal has been trigged by the Trigged Data Recorder. The time is specified in fractions of seconds. Valid for. Analog and Digital signals Data type: real
TdrPostBuffer	Allowed	The amount of data that will be saved to the database after the signal has been trigged by the Trigged Data Recorder. The time is specified in fractions of seconds.

		Valid for. Analog and Digital signals Data type: real
TdrTrigger	Allowed	Defines if the Signal is a trigger for the Trigged Data Recorder. 1 = The Signal is a trigger. Valid for. Analog and Digital signals Data type: bit
TdrLevel	Allowed	Defines on which flank the trigger Signal should trig the Trigged Data Recorder 1 = The Signal triggers when going from 0 to 1 0 = The Signal triggers when going from 1 to 0 Valid for. Analog and Digital signals Data type: bit
LiveRequest	Allowed	The last time a client requested the last value of this signal. Valid for: Analog Data type: datetime
ChangedBy	Allowed	The user name that made the last configuration change to this signal. All changes made to the signal are persisted in the SignalRev table. Data type: nvarchar(200)
Origin	Allowed	Describes the origin of the data for this signal. Data type: nvarchar(200)

2.1.3.2 SignalProp

Extra properties on signals, in a key/value-style table.

Columns	Data type	Allow NULLs	Description
SignalID	smallint	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
PropKey	nvarchar(50)	Not allowed	The property name (a.k.a the key)
ChangeTime	datetime	Not allowed	The date and time this property was set
ChangedBy	nvarchar (100)	Allowed	The user name that made the last configuration change to this signal.

			All changes made to the signal are persisted in the SignalRev table. Data type: nvarchar(200)
Active	bit	Allowed	Property is only used when active = 1
PropValue	sql_variant	Allowed	The property value

A typical entry for each signal is the key "Responsible".

2.1.3.3 ASignalDef

Signal definition table for analog signals with last added value.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
MaxDivergence	Allowed	MaxDivergence property is used for setting the range on compression function. If the derivate value between the last archived value and the last received value are within MaxDivergence the value will not be archived. The MaxDivergence can be set manually by an administrator or automatically by the CompTune application. The CompTune application performs an advanced signal analyze of each signal to find the accuracy level of the signal. When finding that with good accuracy, tuning of the compression can be made. Valid for: Analog Data type: float More...
LastChangeTime	Allowed	Timestamp for when the Last value was logged. Valid for: Analog, Digital Data type: datetime
LastValue	Allowed	The value of the last logged value. Valid for: Analog, Digital Data type: float (for Analog values), tinyint (for Digital values)
LastQuality	Allowed	Quality value for Last logged value logged. Valid for: Analog, Digital

		Data type: smallint
ForceSaveTime	Allowed	The ForceSaveTime property is a time, in seconds, for how long new values can be rejected due to the compression function. If the ForceSaveTime has passed since last archived value, the previous value will be archived (by force). Valid for: Analog, Digital Data type: bigint More...
RejectSaveTime	Allowed	RejectSaveTime overrides the compression function and rejects any values to be archived to the database, during this time (in seconds). Valid for: Analog, Digital Data type: bigint More...
InCalc	Allowed	The InCalc property sets whether the signal is in a calculation. It is also referred to as CalcTrigger. 1 – in calculation. 0 – not in calculation. Valid for: Analog, Digital Data type: bit
CalculationInterval	Allowed	The interval for a time calculated TSQL-formula to be executed, in seconds. The CalculationInterval property is ignored if the calculation is a formula calculated signal. Valid for: Analog Data type: bigint
GroupingPoint	Allowed	GroupingPoint is used to decide where to put the timestamp for a time calculated TSQL-formula. In procedures for storing formulas and internally in the database the following codes are used: 1 – The calculated value and timestamp is placed in the beginning of the calculated period. 2 – The calculated value and timestamp is placed in the middle of the calculated period. 4 – The calculated value and timestamp is placed at the end of the calculated period. Valid for: Analog Data type: tinyint
CalcReferenceTime	Allowed	A reference time for when the calculation will be triggered. Used as reference time for time calculations. When using many average calculation tags, the calculations can be spread in time. This is to avoid accumulation of concurrent calculations. To spread the calculations, the CalcReferenceTime property can be used. The default is that calculation occurs at the beginning of next period using

		`@@CalcReferenceTime = '2001-01-01 00:00:00'` but if following setting is used instead `@@CalcReferenceTime = '2001-01-01 00:00:10'` the calculation is moved to take place 10 seconds later. Valid for: Analog Data type: datetime
BackLog	Allowed	BackLog is the time in milliseconds, used to retrieve a previous value when the signal has an updated value that the compression will allow to be stored (outside MaxDivergence). The previous value is then stored at "number of BackLog" milliseconds before the new value, if the BackLog time is less than the last logged value. Without BackLog the trend for the signal might be unrepresented. If BackLog is set to 0 (zero) no BackLogging will occur. The BackLog value point is flagged with quality bit 512. Valid for: Analog Data type: bigint More...
TuneStatus	Allowed	The TuneStatus property is used by the CompTune application to determine the status of the compression for the signal. CompTune will change this property as different stages of the tuning are performed. The value is bit coded as: 1- Represents that the signal should collect data for tuning analyze 2 - Represents that the database has collected enough RAW data for this signal to start the tuning analyze. 4 - Represents that the signal has finished its tuning analyze. 8 - Represents that the tuning analyze has Checked the signals tuning. 16 - Represents that the tuning is Enabled for this signal. 32 - Represents that the value of the reference tag is to low and the database has stopped collecting RAW data. 16384 - Represents that the signal is imported by XMZ-import. Valid for: Analog Data type: smallint
TuneCount	Allowed	The number of processed compression auto tunings. For internal use only. Valid for: Analog Data type: smallint
SourceScale	Allowed	SourceScale and SourceOffset is used to rescale an input value if the input has a wrong measuring scale.

		If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula: `The Stored Value = The Input Value * SourceScale +` Valid for: Analog Data type: float
SourceOffset	Allowed	SourceOffset and SourceScale is used to rescale an input value if the input has a wrong measuring scale. If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula: `The Stored Value = The Input Value * SourceScale +` Valid for: Analog Data type: float

2.1.3.4 ASignalLog

Compressed and archived analog values.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
ChangeTime	Not allowed	The timestamp for when the value of the signal changed. This applies to both digital and analog signals. Valid for: Analog, Digital Data type. Datetime
ChangeValue	Not allowed	Value of the archived signal. Valid for: Analog, Digital Data type: real or float (for Analog, depending on database installation setting), tinyint (for Digital)
Quality	Not allowed	The Quality attribute represents the quality state for a signals value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...

2.1.3.5 DSignalDef

Signal definition table for digital signals with last added value.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
LastValue	Allowed	The value of the last logged value. Valid for: Analog, Digital Data type: float (for Analog values), tinyint (for Digital values)
LastChangeTime	Allowed	Timestamp for when the Last value was logged. Valid for: Analog, Digital Data type: datetime
TotalRunTime0	Allowed	The total number of seconds the signal has been 0 (zero), since the first value was logged. Valid for: Digital Data type: float (in def table) or int (in log table)
TotalRunTime1	Allowed	The total number of seconds the signal has been 1 (one), since the first value was logged. Valid for: Digital Data type: float (in def table) or int (in log table)
TotalChanges	Allowed	The total number of times the signal has changed from 0 (zero) to 1 (one). Valid for: Digital Data type: bigint (in def table) or int (in log table)
ServiceTime	Allowed	Timestamp when last service was registered by user. Valid for: Digital Data type: datetime
ServiceRunTime0	Allowed	The total time, in seconds, that the signal has been 0 (zero), since the last service. Valid for: Digital Data type: float (in def table) or int (in log table)
ServiceRunTime1	Allowed	The total time, in seconds, that the signal has been 1 (one), since the last service. Valid for: Digital Data type: float (in def table) or int (in log table)
ServiceChanges	Allowed	The number of times the signal has changed from 0 (zero) to 1 (one), since last service. Valid for: Digital Data type: bigint (in def table) or int (in log table)

ForceSaveTime	Allowed	The ForceSaveTime property is a time, in seconds, for how long new values can be rejected due to the compression function. If the ForceSaveTime has passed since last archived value, the previous value will be archived (by force). Valid for: Analog, Digital Data type: bigint More...
RejectSaveTime	Allowed	RejectSaveTime overrides the compression function and rejects any values to be archived to the database, during this time (in seconds). Valid for: Analog, Digital Data type: bigint More...
LastQuality	Allowed	Quality value for Last logged value logged. Valid for: Analog, Digital Data type: smallint
ServiceLimitChanges	Allowed	The limit of number of times the signal can change from 0 (zero) to 1 (one), since last service. Valid for: Digital Data type: bigint
ServiceLimitRunTime	Allowed	The limit of the total operating time, in seconds, that the signal is allowed to be active (0 or 1 depending on configuration of ActiveRunTime), since last service. Valid for: Digital Data type: bigint
ActiveRunTime	Allowed	This parameter is to display for the user which state that is the active and that is interesting as “operating state”. It determines the active runtime for the digital signal, 0 (zero) means that the signal is active if the value is 0 (zero) and vice versa. 0 – If 0 is the active runtime. 1 – If 1 is the active runtime. Valid for: Digital Data type: tinyint
InCalc	Allowed	The InCalc property sets whether the signal is in a calculation. It is also referred to as CalcTrigger. 1 – in calculation. 0 – not in calculation. Valid for: Analog, Digital Data type: bit

2.1.3.6 DSignalLog

Compressed and archived digital values.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
ChangeTime	Not allowed	The timestamp for when the value of the signal changed. This applies to both digital and analog signals. Valid for: Analog, Digital Data type. Datetime
ChangeValue	Not allowed	Value of the archived signal. Valid for: Analog, Digital Data type: real or float (for Analog, depending on database installation setting), tinyint (for Digital)
Quality	Not allowed	The Quality attribute represents the quality state for a signals value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...
TotalRunTime0	Not allowed	The total number of seconds the signal has been 0 (zero), since the first value was logged. Valid for: Digital Data type: float (in def table) or int (in log table)
TotalRunTime1	Not allowed	The total number of seconds the signal has been 1 (one), since the first value was logged. Valid for: Digital Data type: float (in def table) or int (in log table)
TotalChanges	Not allowed	The total number of times the signal has changed from 0 (zero) to 1 (one). Valid for: Digital Data type: bigint (in def table) or int (in log table)
ServiceRunTime0	Not allowed	The total time, in seconds, that the signal has been 0 (zero), since the last service. Valid for: Digital Data type: float (in def table) or int (in log table)

ServiceRunTime1	Not allowed	The total time, in seconds, that the signal has been 1 (one), since the last service. Valid for: Digital Data type: float (in def table) or int (in log table)
ServiceChanges	Not allowed	The number of times the signal has changed from 0 (zero) to 1 (one), since last service. Valid for: Digital Data type: bigint (in def table) or int (in log table)

2.1.3.7 MVSignalLog

Table to store Multi values.

Columns	Allow NULLs	Data type	Description
SignalID	Not allowed	smallint	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
MVId	Not allowed	bigint	Unique ID identifying this multi value
MVTime	Allowed	datetime2(7)	The user defined timestamp of this multi value
String01	Allowed	nvarchar(4000)	User defined string values. Could, for example, store a serialized object using JSON or XML.
String02			
String03			
String04			
String05			
String06			
String07			
String08			
String09			
String10			
DateTime01	Allowed	datetime2(7)	User defined datetimes.
DateTime02			
Decimal01	Allowed	float	User defined 64-bit floating point values.
Decimal02			
Decimal03			
Decimal04			

Blob01	Allowed	varbinary (MAX)	User defined binary data.

2.1.3.8 CalcSignals

Relational table that holds information which signals are used in whish calculated signals.

Columns	Data type	Allow NULLs	Description
SignalIDCalc	smallint	Not allowed	The SignalID of the calculated signal.
SignalIDParam	Smallint	Not allowed	The SignalID of the parameter signals that are used in the calculation formula.
ParamID	tinyint	Not allowed	The sequence number of the parameters in the calculation formula.

2.1.3.9 EventDef

The Eventdefinition in AA follow the OPCAE standard and uses the same attributes.

For a more detailed description on each attribute, see the OPC Alarms & Events Specification (Version 1.10).

The OPCAE standard is not as clear as the OPCDA standard. There are often differences between systems, but all the standard attributes are included in the AA database to store the values.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
EventID	Not allowed	The EventID is a unique ID that identifies the event and is generated by the system. Valid for: Events Data type: int
ConditionName	Allowed	OPCAE Standard field. The name of the condition related to this event notification. (The QueryConditionNames method gives clients a means of finding out the specific condition names which the event server supports for the specified event category. This method would typically be invoked prior to specifying an event filter. Condition names are server specific. The number of condition names returned will vary depending on the sophistication of the server, but is expected to be less than 30 for most servers, making this interface more appropriate than a custom enumerator. It is expected that the results of the Query will be fairly

		'stable' in most situations. However, the Server is in fact allowed to change the available selection at any time. Therefore, a Client should do (or at least allow as an option) a fresh Query every time a selection is to be presented to the end user.) Valid for: Events Data type: nVarChar(200)
Message	Allowed	Message text which describes the event. For condition-related events, this will generally include the description property of the active sub-condition. Valid for: Events Data type: nvarchar(1000)
EventCategory	Allowed	OPCAE Standard field. EventCategories define groupings of events supported by an OPC Event server. Examples of event categories might include “Process Events”, “System Events”, or “Batch Events”. Event categories may be defined for all event types, i.e. Simple, Tracking, and Condition-Related. However, a particular event category can include events of only one type. A given Source (e.g. “System” or “FIC101”) may generate events for multiple event categories. Names of event categories must be unique within the event server. The definition of event categories is server specific and is outside the scope of this specification. The name of the event category is included in every event notification. Event subscriptions may be filtered based on event category. Valid for: Events Data type: nvarchar(200)
Severity	Allowed	OPCAE Standard field. The severity value is an indication of the urgency of the sub-condition. This is also commonly called ‘priority’, especially in relation to process alarms. Values will range from 1 to 1000, with 1 being the lowest severity and 1000 being the highest. Typically, a severity of 1 would indicate an event which is informational in nature, while a value of 1000 would indicate an event of catastrophic nature which could potentially result in severe financial loss or loss of life. Valid for: Event Data type: smallint More...
SubConditionName	Allowed	OPCAE Standard field. Name of current sub-condition, for multi-state conditions. For a single-state condition, this contains the condition name. Valid for: Events Data type: nvarchar(200)

AckReqd	Allowed	Abbreviation for OPCAE standard field, "AckRequired". See separate description for this field.
EventDesc	Allowed	Additional field not specified in OPCAE standard. This field is used by Siemens PCS7 system that uses this field as own defined customer attributes. Valid for: Events Data type: nvarchar(1000)
EventServerNode	Allowed	Additional field not specified in the OPCAE standard. This field can be used to specify the collecting eventservernode when there are several eventservers connected to one AA system. The name of the eventserver is specified in the OPCAEClient applications config file. The parametername in the config file is "CollEventServerNode". Valid for: Events Data type: nvarchar(200)
EventCategoryID		EventCategoryID as a numeric value describing the Category of an Event. Valid for: Events Data type: int
CustomEvDef01		The CustomEvDef01 - 03 are free Event definition properties to be used by customer. Valid for: Event Data type: nvarchar(50)
CustomEvDef02		
CustomEvDef03		

2.1.3.10 EventLog

Archived events.

Columns	Allow NULLs	Description
SignalID	Not allowed	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
EventID	Not allowed	The EventID is a unique ID that identifies the event and is generated by the system. Valid for: Events Data type: int
EventTime	Not allowed	The timestamp for when the event was logged. Valid for: Events Data type: datetime
EventActiveTime	Allowed	OPCAE Standard field.

		ActiveTime is the time of the transition into the condition or sub-condition which is associated with this event notification. This time corresponds to SubCondLastActive property of the associated OPCCondition object and is used to correlate condition acknowledgements with a particular transition into the condition/sub-condition. Valid for: Events Data type: DateTime
Quality	Allowed	The Quality attribute represents the quality state for a signals value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...
ActorID	Allowed	OPCAE Standard field. ActorID is the identifier of the OPC Client which acknowledged the condition, which is maintained as the AcknowledgerID property of the condition. This is included in event notifications generated by condition acknowledgments. Valid for: Events Data type: nvarchar(50)
Cookie	Allowed	OPCAE Standard field. Server defined cookie associated with the event notification. This value can be used by a client when acknowledging the condition. This value is opaque to the client. Valid for: Events Data type: int
EventType	Allowed	OPCAE Standard field. EventType specifies what type of event it is. Predefined according to the standard is: OPC_SIMPLE_EVENT OPC_CONDITION_EVENT OPC_TRACKING_EVENT OPC_ALL_EVENTS Valid for: Events Data type: tinyint
NewState	Allowed	OPCAE Standard field NewState is A WORD bit mask of three bits specifying the new state of the condition: OPC_CONDITION_ACTIVE (= 2), OPC_CONDITION_ENABLED (= 1) , OPC_CONDITION_ACKED (= 4)

		Valid for: Event Data type: tinyint
ChangeMask	Allowed	OPCAE Standard field. ChangeMask indicates to the client which properties of the condition have changed, to cause the server to send the event notification. Indicates to the client which properties of the condition have changed, to have caused the server to send the event notification. It may have one or more of the following values: OPC_CHANGE_ACTIVE_STATE = 1 (The condition's active state has changed.) OPC_CHANGE_ACK_STATE = 2 (The condition's acknowledgment state has changed.) OPC_CHANGE_ENABLE_STATE = 4 (The condition's enabled state has changed.) OPC_CHANGE_QUALITY = 8 (The ConditionQuality has changed.) OPC_CHANGE_SEVERITY = 16 (The severity level has changed.) OPC_CHANGE_SUBCONDITION = 32 (The condition has transitioned into a new sub-condition.) OPC_CHANGE_MESSAGE = 64 (The event message has changed (compared to prior event notifications related to this condition).) OPC_CHANGE_ATTRIBUTE = 128 (One or more event attributes have changed (compared to prior event notifications related to this condition).) If the event notification is the result of a Refresh, these bits are to be ignored. For a "new event",OPC_CHANGE_ACTIVE_STATE is the only bit which will always be set. Other values are server specific. (A "new event" is any event resulting from the related condition leaving the Inactive and Acknowledged state.) Valid for: Events Data type: tinyint
EventStatus	Allowed	Additional field not specified in OPCAE standard. Used for Siemens PCS7 system that uses this as own defined customer attributes. Valid for: Events Data type: varchar(10)
BackColor		The BackColor is an Event log property that can be used to store the events background color. The color should be represented as a HTML color specification. Valid for: Event Data type: varchar(10)

ForeColor		The ForeColor is an Event log property that can be used to store the events foreground color. The color should be represented as a HTML color specification. Valid for: Event Data type: varchar(10)
CustomEvLog01		The CustomEvLog01 - 03 are free Event log properties to be used by customer. Valid for: Event Data type: nvarchar(50)
CustomEvLog02		
CustomEvLog03		

2.1.3.11 OPCServer

OPC Server definition and configuration table.

Columns	Data type	Allow NULLs	Description
OPCServerID	Smallint	Not allowed	Unique ID identifying the OPC server and is generated by the system. Valid for: Data type: smallint
OPCServerName	nvarchar (200)	Allowed	Name of the OPC server that are connected. This name is case sensitive. Make sure that the name is entered correct. Valid for. Analog, Digital Data type: nvarchar(200)
OPCServerNode	nvarchar(50)	Allowed	The computer name or IP address where the OPC Server is located that are connected. Valid for: Analog, Digital Data type:nvarchar(50)
OPCDAClientName	nvarchar(50)	Allowed	The name of the OPCDA client instance connected to collect data from this OPCServer. The OPCDAClient name must be equal with the parameter specifying this in the configuration file of the OPCDA client application. Valid for: Analog, Digital Data type: nvarchar(50)
ForceUpdRate	bigint	Allowed	ForceUpdRate is the update rate in milliseconds for signals related to this server to be checked if MaxForceSaveTime has been passed for any tags. It also serves the purpose of moving last update time from this server to be used when presenting realtime data.

			This time is “kept alive” by an active pulse from the OPCClient. The Frequency of this pulse must not be longer than the ForceUpdRate. The Frequency of the pulse is set in the OPCDA applications configuration file. The parameter in the configuration file is “ChkSigForceUpdates_ms” and is entered in ms. Valid for: Analog, Digital Data type: bigint
TriggerUpdRate	bigint	Allowed	The update rate in milliseconds for time calculated signals related to this server, to be triggered for new calculation. This time must be shorter than the minimum average time configured for any tags. This time is “kept alive” by an active pulse from the OPCClient. The Frequency of this pulse must not be longer than the TriggerUpdRate. If no OPCClient is connected. The OPCServer “Internal” can be used instead. The Internal server is triggered by an internal scheduler, but the update rate can not be less than 60s for this Internal server. Valid for: Analog, Digital Data type: bigint

2.1.3.12 OPCTagGroup

OPC tag groups definition and configuration table.

Columns	Allow NULLs	Description
OPCGroupID	Not allowed	A unique ID identifying the OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
OPCGroupName	Allowed	Name of the OPC group. Valid for: Analog, Digital Data type: nvarchar(50)
DeadBand	Allowed	Deadband (in % of signals measure range) that signals in this group will have as exception deadband parameter. Changes within this deadband will not be sent as updated values from the OPCServer to the OPCClient. Valid for: Events Data type: smallint
UpdateRate	Allowed	Rate in milliseconds to update the OPC tag group with from

		the OPCServer. The rate is used when setting up the scheduling towards the OPCServer. Valid for: Analog, Digital Data type: int
AsyncRead	Allowed	This parameter defines that reading from OPCServer as Asynchronous, which means that only changes are sent, and not cyclical updates. Async read is the only supported method. Valid for: Analog, Digital Data type: bit
OPCServerID	Allowed	Unique ID identifying the OPC server and is generated by the system. Valid for: Data type: smallint

2.1.3.13 DBPurgeStat

Configuration parameters for the scheduled DBPurge job.

Columns	Data type	Allow NULLs	Description
SpaceLimitPercent	float	Not allowed	Configuration parameter that tells the DBPurge job how mush space in percents that should be left free. Normally 2.5%
FreeSpace	float	Allowed	The current free space in the database.
CheckTime	datetime	Allowed	The time when the DBPurge job last checked the free space.
DeleteStartTime	datetime	Allowed	The point from where data was last deleted.
DeleteRange	Int	Not allowed	The number of hours that was last deleted.

2.1.3.14 UnitConvert

Definitions of conversion data between different engineering units.

Columns	Allow NULLs	Description
SourceUnit	Not allowed	The SourceUnit is mapped to the Signals ValueUnit, e.g °C Valid for: Analog Data type: nvarchar(50)
TargetUnit	Not allowed	TargetScale and TargetOffset is used to convert an Signals value to an alternative Unit If TargetScale is not equal 0, the value is converted according to the following formula: `The Converted Value = The Stored Value * TargetSc`

		Valid for: Analog Data type: nvarchar(50)
Description	Allowed	Is used as a Description of the Conversion data type: nvarchar(200)
TargetScale	Allowed	TargetScale and TargetOffset is used to convert an Signals value to an alternative Unit If TargetScale is not equal 0, the value is converted according to the following formula: `The Converted Value = The Stored Value * TargetSc` Valid for: Analog Data type: float more...
TargetOffset	Allowed	TargetOffset and TargetScale is used to convert an Signals value to an alternative Unit If TargetScale is not equal 0, the value is converted according to the following formula: `The Converted Value = The Stored Value * TargetSc` Valid for: Analog Data type: float more...
TargetFormula	Allowed	This field is not used. Valid for: Analog Data type: varchar(2000)

2.1.3.15 TuneEvaluationLog

Result data from Compression tuning analyze.

Columns	Data type	Allow NULLs	Description
SignalID	Smallint	Not allowed	A unique ID that identifies the signal, generated by system.
TuneTime	Datetime	Not allowed	The time when the tuning/analyze took place.
RangeMin	Float	Allowed	The signals RangeMin value when the tuning/analyze took place.
RangeMax	Float	Allowed	The signals RangeMax value when the tuning/analyze took place.
PreviousMaxDivergence	Float	Allowed	The signals MaxDivergence value when the tuning/analyze took place
ProposedMaxDivergence	Float	Allowed	The proposed MaxDivergence from the tuning/analyze
NewMaxDivergence	Float	Allowed	The MaxDivergence set for the signal after the tuning took place
UpdateMaxDivergence	Bit	Allowed	A flag describing if the MaxDivergence was updated after this tuning.
TuneCount	Smallint	Allowed	The number of processed compression auto tunings. For internal use only.

			Valid for: Analog Data type: smallint
StartPeriod	Datetime	Allowed	The start time of the value points used for tuning/analyze.
EndPeriod	Datetime	Allowed	The end time of the value points used for tuning/analyze.
NumberOfSamples	Smallint	Allowed	The number of value points used for tuning/analyze.
ProposedCompressionRate	Float	Allowed	The compressionrate which resulted from the ProposedMaxDivergence
CompressionAdjusted	Int	Allowed	An indicator showing whether the compressionproposal from the tuning analyzer was adjusted. A value of 0 indicates that no adjusting took place. Negative value indicates that the value was adjusted down and a positive value indicates it was adjusted up.
AnalyzerMaxDivergence	Float	Allowed	The MaxDivergence the tuning analyzer proposed (deadband elimination)
AnalyzerCompressionRate	Float	Allowed	The compressionrate of the proposed MaxDivergence from the analyzer.
NewMaxCompressionRate	Float	Allowed	The compressionrate of the MaxDivergence set after the tuning
TuneStatus	Smallint	Allowed	The TuneStatus property is used by the CompTune application to determine the status of the compression for the signal. CompTune will change this property as different stages of the tuning are performed. The value is bit coded as: 1- Represents that the signal should collect data for tuning analyze 2 - Represents that the database has collected enough RAW data for this signal to start the tuning analyze. 4 - Represents that the signal has finished its tuning analyze. 8 - Represents that the tuning analyze has Checked the signals tuning. 16 - Represents that the tuning is Enabled for this signal. 32 - Represents that the value of the reference tag is to low and the database has stopped collecting RAW data. 16384 - Represents that the signal is imported by XMZ-import. Valid for: Analog Data type: smallint
MinValue	Float	Allowed	The min value of the signal during compression analyze.
MaxValue	Float	Allowed	The max value of the signal during compression analyze.
AvgValue	Float	Allowed	The average value of the signal during compression analyze.

2.1.4 Database views

Views	Description
vSignalAdmin	View for displaying signal configuration data in AutArch Admin application.
vEvents	View for displaying events in AutArch Events application.
vPOS	View for displaying maintenance data in AutArch POS application.

2.1.4.1 vSignalAdmin

View for displaying signal configuration data in AutArch Admin application.

Columns	Description
SignalID	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
SignalName	A unique, user input, name for the signal. The SignalName can be changed. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
SignalDesc	The description of the signal Valid for: Analog, Digital, Event Data type: nvarchar(255)
SignalType	"Analog", "Digital" or "Event"
CreationTime	The CreationTime property is a timestamp when the signal was first created. The timestamp is generated by system and cannot be changed. Valid for: Analog, Digital, Events Data type: datetime
LogBlocking	The LogBlocking property blocks a signal from being logged to the database. When LogBlocking is activated or deactivated, the OPCDA application needs to be restarted to effectuate the changes. The OPCDA application excludes a blocked tag from scheduling. 0 – Signal is not blocked 1 – Signal is blocked, and no data is archived. Valid for: Analog, Digital Data type: bit
ValueName	ValueName gives the possibility to use a description for the valuetype, e.g. "Power". (For future use by connecting applications.) Valid for: Digital

	Data type: nvarchar(50)
ValueName0	The ValueName0 property is used to provide a description for when signal is 0 (zero), such as “CLOSED” or “OFF”. Valid for: Digital Data type: nvarchar(50)
ValueName1	The property is used to provide a description for when signal is 1 (one), such as “OPEN” or “ON”. Valid for: Digital Data type: nvarchar(50)
ValueUnit	The unit for the signal, e.g m/s, kPa etc. Valid for: Analog Data type: nvarchar(50)
RangeMin	RangeMax is used to set the default max range for the signal when opened in viewing applications, e.g. Trendviewer. Valid for: Analog Data type: float
RangeMax	RangeMax is used to set the default max range for the signal when opened in viewing applications, e.g. Trendviewer. Valid for: Analog Data type: float
Interpol	If the Interpol property is set, the values are interpolated by straight line between each value. If the property is not set the values are plotted as steps. The Interpol property is only used with the return values from the Sig_Select_Calc and the Sig_Select_Serie procedures. 0 – No interpolation 1 – Interpolation Valid for: Analog, Digital Data type: bit More...
DisplayFormat	The DisplayFormat property sets the number of decimals for viewing applications, such as Trendview application. The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted. For example, when the value 14059,42450 is used in different display formats, see following table. Valid for: Analog, Digital, Events Data type: nvarchar(20) More...

OPCGroupID	A unique ID identifying the OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
Category1	The category1 property is used for categorizing tags and filtering functions in viewing applications. Valid for: Analog, Digital Data type: nvarchar(20) More...
OPCItemID	The signals item id on the OPC server. Valid for: Analog, Digital Data type: nvarchar(1000)
MaxDivergence	MaxDivergence property is used for setting the range on compression function. If the derivate value between the last archived value and the last received value are within MaxDivergence the value will not be archived. The MaxDivergence can be set manually by an administrator or automatically by the CompTune application. The CompTune application performs an advanced signal analyze of each signal to find the accuracy level of the signal. When finding that with good accuracy, tuning of the compression can be made. Valid for: Analog Data type: float More...
ActiveRunTime	This parameter is to display for the user which state that is the active and that is interesting as “operating state”. It determines the active runtime for the digital signal, 0 (zero) means that the signal is active if the value is 0 (zero) and vice versa. 0 – If 0 is the active runtime. 1 – If 1 is the active runtime. Valid for: Digital Data type: tinyint
ForceSaveTime	The ForceSaveTime property is a time, in seconds, for how long new values can be rejected due to the compression function. If the ForceSaveTime has passed since last archived value, the previous value will be archived (by force). Valid for: Analog, Digital Data type: bigint More...

RejectSaveTime	RejectSaveTime overrides the compression function and rejects any values to be archived to the database, during this time (in seconds). Valid for: Analog, Digital Data type: bigint More...
SignalType_Digital	1 = Digital Signal
SignalType_Analog	1 = Analog Signal
SignalType_Event	1 = The Signal has Events
SignalType_Text	1 = The Signal has Notes (for future use)
SignalType_Calculated	1 = The Signal is formula calculated
SignalType_Out	1 = The Signal is an Output signal (for future use)
InCalc	The InCalc property sets whether the signal is in a calculation. It is also referred to as CalcTrigger. 1 – in calculation. 0 – not in calculation. Valid for: Analog, Digital Data type: bit
CalcEnabled	CalcEnabled property determines if calculation is enabled or not. 0 – Calculation not enabled. 1 – Calculation enabled. Valid for: Analog, Digital, MultiValue Data type: bit
CalcType	Determines the type of calculation to execute for calculation signals. 0 – a user written formula (CalcFormula property) 2 – time calculated by average value 4 – time calculated by maximum value 5 – time calculated by minimum value 10 – Python calculated 11 – Average periodic 12 – Sum periodic Valid for: Analog, Digital, MultiValue Data type: tinyint
CalcFormula	The user written formula used for calculations. The formula calculation needs one or several input signals in the formula of which at least one input signal need to have the CalcTrigger set. The CalcTrigger triggers the calculation when the value of the signal changes. The signals that are used in the formula are stored in an relational table but they are edited by writing the signal names surrounded by "{" and "}" in the following way:

	`{SignalName1} + {SignalName2}` Valid for: Analog, Digital Data type: varchar(2000)
CalculationInterval	The interval for a time calculated TSQL-formula to be executed, in seconds. The CalculationInterval property is ignored if the calculation is a formula calculated signal. Valid for: Analog Data type: bigint
GroupingPoint	GroupingPoint is used to decide where to put the timestamp for a time calculated TSQL-formula. In procedures for storing formulas and internally in the database the following codes are used: 1 – The calculated value and timestamp is placed in the beginning of the calculated period. 2 – The calculated value and timestamp is placed in the middle of the calculated period. 4 – The calculated value and timestamp is placed at the end of the calculated period. Valid for: Analog Data type: tinyint
CalcReferenceTime	A reference time for when the calculation will be triggered. Used as reference time for time calculations. When using many average calculation tags, the calculations can be spread in time. This is to avoid accumulation of concurrent calculations. To spread the calculations, the CalcReferenceTime property can be used. The default is that calculation occurs at the beginning of next period using `@@CalcReferenceTime = '2001-01-01 00:00:00'` but if following setting is used instead `@@CalcReferenceTime = '2001-01-01 00:00:10'` the calculation is moved to take place 10 seconds later. Valid for: Analog Data type: datetime
BackLog	BackLog is the time in milliseconds, used to retrieve a previous value when the signal has an updated value that the compression will allow to be stored (outside MaxDivergence). The previous value is then stored at “number of BackLog” milliseconds before the new value, if the BackLog time is less than the last logged value. Without BackLog the trend for the signal might be unrepresented. If BackLog is set to 0 (zero) no BackLogging will occur. The BackLog value point is flagged with quality bit 512. Valid for: Analog Data type: bigint

	More...
ExportFlag	The ExportFlag property is set to mark the signal for export. It is used by the AAExport application for exporting data to encrypted exportfiles. By setting combinations for this value, different export scope can be performed. 0 – no export 1 – signal export The flag is used in a bit coded way that makes it possible to flag a signal to be exported by different export jobs, se the AAExport documentation for examples. Valid for: Analog, Digital Data type: smallint
TuneStatus	The TuneStatus property is used by the CompTune application to determine the status of the compression for the signal. CompTune will change this property as different stages of the tuning are performed. The value is bit coded as: 1- Represents that the signal should collect data for tuning analyze 2 - Represents that the database has collected enough RAW data for this signal to start the tuning analyze. 4 - Represents that the signal has finished its tuning analyze. 8 - Represents that the tuning analyze has Checked the signals tuning. 16 - Represents that the tuning is Enabled for this signal. 32 - Represents that the value of the reference tag is to low and the database has stopped collecting RAW data. 16384 - Represents that the signal is imported by XMZ-import. Valid for: Analog Data type: smallint
TuneCount	The number of processed compression auto tunings. For internal use only. Valid for: Analog Data type: smallint
SignalType_ManualEditable	1 = The Signal is manual editable.
EventBlocking	The EventBlocking property activates/inactivates events from being logged to the database. 0 = Event is not blocked 1 = Event is blocked and no data is archived. This attribute can be set anytime and changes take effect directly on archiving. Valid for: Events Data type: bit
Custom01	The Custom01 - 05 are free signal properties to be used by customer. Valid for: Analog, Digital Data type: nvarchar(100)
Custom02	
Custom03	

Custom04	
Custom05	
SignalName2	The name of the signal in a alternative language. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
SignalDesc2	The description of the signal in a alternative language. Valid for: Analog, Digital, Event Data type: nvarchar(255)
ValueUnit2	The unit for the signal, e.g m/s, kPa etc. in a alternative language Valid for: Analog Data type: nvarchar(50)
RangeMin2	RangeMin2 is used to set the default min range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit. Valid for: Analog Data type: float
RangeMax2	RangeMax2 is used to set the default max range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit. Valid for: Analog Data type: float
DisplayFormat2	The DisplayFormat2 property sets the number of decimals for viewing applications, such as Trendview application when using an alternative engineering unit. The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted. For example, when the value 14059,42450 is used in different display formats, see following table. Valid for: Analog, Digital, Events Data type: nvarchar(20) More...
SourceScale	SourceScale and SourceOffset is used to rescale an input value if the input has a wrong measuring scale. If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula: `The Stored Value = The Input Value * SourceScale + Sou` Valid for: Analog Data type: float

SourceOffset	SourceOffset and SourceScale is used to rescale an input value if the input has a wrong measuring scale. If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula: `The Stored Value = The Input Value * SourceScale + Sou` Valid for: Analog Data type: float
TuneStatus_Collect	1 = The signal should collect data for tuning analyze.
TuneStatus_Collected	1 = The database has collected enough RAW data for this signal to start the tuning analyze.
TuneStatus_Finished	1 = The signal has finished its tuning analyze.
TuneStatus_Checked	1 = The tuning analyze has Checked the signals tuning.
TuneStatus_Enabled	1 = The tuning is Enabled for this signal.
TuneStatus_OverRefLimit	1 = The value of the reference tag is to low and the database has stopped collecting RAW data.
OPCGroupName	Name of the OPC group. Valid for: Analog, Digital Data type: nvarchar(50)
SourceRangeMin	SourceRangeMin is a calculated value based on SourceScale and SourceOffset. If SoureScale is not equal 0 then The signals RangeMin is rescaled and returned as SourceRangeMin. Valid for: Analog Data type: float
SourceRangeMax	SourceRangeMax is a calculated value based on SourceScale and SourceOffset. If SoureScale is not equal 0 then The signals RangeMax is rescaled and returned as SourceRangeMax. Valid for: Analog Data type: float
TdrOPCGroupID	A unique ID identifying the Trigged Data Recorder OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
TdrOPCGroupName	A unique Name identifying the Trigged Data Recorder OPC group . Valid for: Analog, Digital Data type: nvarchar(50)
TdrOPCItemID	The signals item id on the OPC server for a Trigged Data Recorder signal. Valid for: Analog, Digital

	Data type: nvarchar(1000)
TdrPreBuffer	The amount of data that will be saved to the database before the signal has been trigged by the Trigged Data Recorder. The time is specified in fractions of seconds. Valid for. Analog and Digital signals Data type: real
TdrPostBuffer	The amount of data that will be saved to the database after the signal has been trigged by the Trigged Data Recorder. The time is specified in fractions of seconds. Valid for. Analog and Digital signals Data type: real
TdrTrigger	Defines if the Signal is a trigger for the Trigged Data Recorder. 1 = The Signal is a trigger. Valid for. Analog and Digital signals Data type: bit
TdrLevel	Defines on which flank the trigger Signal should trig the Trigged Data Recorder 1 = The Signal triggers when going from 0 to 1 0 = The Signal triggers when going from 1 to 0 Valid for. Analog and Digital signals Data type: bit
TdrLogSignals	A list of signals that should be logged by the Trigged Data Recorder when trigged by this signal. The signals are stored in an relational table but they are edited by writing an list of signal names in the following way: `{SignalName1}{SignalName2}{SignalName3}...` Valid for: Analog, Digital Data type: nvarchar(2000)

2.1.4.2 vEvents

View for displaying events in AutArch Events application.

Columns	Description
SignalID	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
SignalName	A unique, user input, name for the signal. The SignalName can be changed. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)

SignalDesc	The description of the signal Valid for: Analog, Digital, Event Data type: nvarchar(255)
EventBlocking	The EventBlocking property activates/inactivates events from being logged to the database. 0 = Event is not blocked 1 = Event is blocked and no data is archived. This attribute can be set anytime and changes take effect directly on archiving. Valid for: Events Data type: bit
EventID	The EventID is a unique ID that identifies the event and is generated by the system. Valid for: Events Data type: int
ConditionName	OPCAE Standard field. The name of the condition related to this event notification. (The QueryConditionNames method gives clients a means of finding out the specific condition names which the event server supports for the specified event category. This method would typically be invoked prior to specifying an event filter. Condition names are server specific. The number of condition names returned will vary depending on the sophistication of the server, but is expected to be less than 30 for most servers, making this interface more appropriate than a custom enumerator. It is expected that the results of the Query will be fairly 'stable' in most situations. However, the Server is in fact allowed to change the available selection at any time. Therefore, a Client should do (or at least allow as an option) a fresh Query every time a selection is to be presented to the end user.) Valid for: Events Data type: nVarChar(200)
Message	Message text which describes the event. For condition-related events, this will generally include the description property of the active sub-condition. Valid for: Events Data type: nvarchar(1000)
EventCategory	OPCAE Standard field. EventCategories define groupings of events supported by an OPC Event server. Examples of event categories might include “Process Events”, “System Events”, or “Batch Events”. Event categories may be defined for all event types, i.e. Simple, Tracking, and Condition-Related. However, a particular event category can include events of only one type. A given Source (e.g. “System” or “FIC101”) may generate events for multiple event categories. Names of event categories must be unique within the event server. The definition of event categories is server specific and is outside the scope of this specification. The name of the event category is included in every event notification. Event subscriptions may be filtered based on event category.

	Valid for: Events Data type: nvarchar(200)
Severity	OPCAE Standard field. The severity value is an indication of the urgency of the sub-condition. This is also commonly called 'priority', especially in relation to process alarms. Values will range from 1 to 1000, with 1 being the lowest severity and 1000 being the highest. Typically, a severity of 1 would indicate an event which is informational in nature, while a value of 1000 would indicate an event of catastrophic nature which could potentially result in severe financial loss or loss of life. Valid for: Event Data type: smallint More...
SubConditionName	OPCAE Standard field. Name of current sub-condition, for multi-state conditions. For a single-state condition, this contains the condition name. Valid for: Events Data type: nvarchar(200)
AckReqd	Abbreviation for OPCAE standard field, "AckRequired". See separate description for this field.
EventTime	The timestamp for when the event was logged. Valid for: Events Data type: datetime
EventActiveTime	OPCAE Standard field. ActiveTime is the time of the transition into the condition or sub-condition which is associated with this event notification. This time corresponds to SubCondLastActive property of the associated OPCCondition object and is used to correlate condition acknowledgements with a particular transition into the condition/sub-condition. Valid for: Events Data type: DateTime
Quality	The Quality attribute represents the quality state for a signals value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...
EventStatus	Additional field not specified in OPCAE standard. Used for Siemens PCS7 system that uses this as own defined customer attributes. Valid for: Events Data type: varchar(10)

ActorID	OPCAE Standard field. ActorID is the identifier of the OPC Client which acknowledged the condition, which is maintained as the AcknowledgerID property of the condition. This is included in event notifications generated by condition acknowledgments. Valid for: Events Data type: nvarchar(50)
Cookie	OPCAE Standard field. Server defined cookie associated with the event notification. This value can be used by a client when acknowledging the condition. This value is opaque to the client. Valid for: Events Data type: int
EventType	OPCAE Standard field. EventType specifies what type of event it is. Predefined according to the standard is: OPC_SIMPLE_EVENT OPC_CONDITION_EVENT OPC_TRACKING_EVENT OPC_ALL_EVENTS Valid for: Events Data type: tinyint
ENABLED	OPCAE Standard field. This field is defined in the OPCAE standard as “enabled” state. If “Active”, then the condition or area has been enabled. The condition is currently being checked by the OPC Event Server. The ENABLED condition is a state of the “NewState” values. 2 = OPC_CONDITION_ENABLED. Valid for: Events Data type: bit
ACTIVE	OPCAE Standard field. The ACTIVE condition is a state of the “NewState” values. 1 = OPC_CONDITION_ACTIVE. The state indicates that the associated object is in ACTIVE condition. Valid for: Events Data type: bit
ACKED	This field is defined in the OPCAE standard as “Acknowledged” indication. If “Active”, the condition has been acknowledged The ACKED condition is a state of the “NewState” values. 4 = OPC_CONDITION_ACKED. Valid for: Events Data type: bit
ChangeMask	OPCAE Standard field. ChangeMask indicates to the client which properties of the condition have changed, to cause the server to send the event notification. Indicates to the client which properties of the condition have changed, to have caused the server to send the event notification. It may have one or

	more of the following values: OPC_CHANGE_ACTIVE_STATE = 1 (The condition's active state has changed.) OPC_CHANGE_ACK_STATE = 2 (The condition's acknowledgment state has changed.) OPC_CHANGE_ENABLE_STATE = 4 (The condition's enabled state has changed.) OPC_CHANGE_QUALITY = 8 (The ConditionQuality has changed.) OPC_CHANGE_SEVERITY = 16 (The severity level has changed.) OPC_CHANGE_SUBCONDITION = 32 (The condition has transitioned into a new sub-condition.) OPC_CHANGE_MESSAGE = 64 (The event message has changed (compared to prior event notifications related to this condition).) OPC_CHANGE_ATTRIBUTE = 128 (One or more event attributes have changed (compared to prior event notifications related to this condition).) If the event notification is the result of a Refresh, these bits are to be ignored. For a "new event",OPC_CHANGE_ACTIVE_STATE is the only bit which will always be set. Other values are server specific. (A "new event" is any event resulting from the related condition leaving the Inactive and Acknowledged state.) Valid for: Events Data type: tinyint
EventServerNode	Additional field not specified in the OPCAE standard. This field can be used to specify the collecting eventservernode when there are several eventservers connected to one AA system. The name of the eventserver is specified in the OPCAEClient applications config file. The parametername in the config file is "CollEventServerNode". Valid for: Events Data type: nvarchar(200)
NewState	OPCAE Standard field NewState is A WORD bit mask of three bits specifying the new state of the condition: OPC_CONDITION_ACTIVE (= 2), OPC_CONDITION_ENABLED (= 1) , OPC_CONDITION_ACKED (= 4) Valid for: Event Data type: tinyint
EventDesc	Additional field not specified in OPCAE standard. This field is used by Siemens PCS7 system that uses this field as own defined customer attributes. Valid for: Events Data type: nvarchar(1000)
EventCategoryID	EventCategoryID as a numeric value describing the Category of an Event. Valid for: Events Data type: int

CustomEvDef01	The CustomEvDef01 - 03 are free Event definition properties to be used by customer. Valid for: Event Data type: nvarchar(50)
CustomEvDef02	
CustomEvDef03	
BackColor	The BackColor is an Event log property that can be used to store the events background color. The color should be represented as a HTML color specification. Valid for: Event Data type: varchar(10)
ForeColor	The ForeColor is an Event log property that can be used to store the events foreground color. The color should be represented as a HTML color specification. Valid for: Event Data type: varchar(10)
CustomEvLog01	The CustomEvLog01 - 03 are free Event log properties to be used by customer. Valid for: Event Data type: nvarchar(50)
CustomEvLog02	
CustomEvLog03	

2.1.4.3 vPOS

View for displaying maintenance data in AutArch POS application.

Columns	Description
SignalName	A unique, user input, name for the signal. The SignalName can be changed. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
SignalName2	The name of the signal in a alternative language. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
SignalID	Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
SignalDesc	The description of the signal Valid for: Analog, Digital, Event Data type: nvarchar(255)
SignalDesc2	The description of the signal in a alternative language. Valid for: Analog, Digital, Event Data type: nvarchar(255)

TotalRunTimeActive	The total number of seconds the signal has been active (0 or 1 depending on configuration, TotalRunTime0 or TotalRunTime1), since the first value was logged. Valid for: Digital Data type: int
TotalChanges	The total number of times the signal has changed from 0 (zero) to 1 (one). Valid for: Digital Data type: bigint (in def table) or int (in log table)
ServiceTime	Timestamp when last service was registered by user. Valid for: Digital Data type: datetime
ServiceRunTimeActive	The total time, in seconds, that the signal has been active (0 or 1 depending on configuration of ActiveRunTime), since last service. Valid for: Digital Data type: int
ServiceChanges	The number of times the signal has changed from 0 (zero) to 1 (one), since last service. Valid for: Digital Data type: bigint (in def table) or int (in log table)
ServiceLimitRunTime	The limit of the total operating time, in seconds, that the signal is allowed to be active (0 or 1 depending on configuration of ActiveRunTime), since last service. Valid for: Digital Data type: bigint
ServiceLimitChanges	The limit of number of times the signal can change from 0 (zero) to 1 (one), since last service. Valid for: Digital Data type: bigint
TimeDiff	The difference, in seconds between ServiceLimitRuntime – ServiceRunTimeActive. Valid for. Digital Data type: bigint
ChangeDiff	The ChangeDiff property is the difference between ServieLimitChanges and ServiceChanges. Which means it is the number of times the signal can change before it reaches service limit (ServiceLimitChanges). Valid for: Digital Data type: bigint

2.1.5 Database procedures

Views	Description
Tag configuration store and delete procedures.	

Sig_Def_Store	Procedure to create or change signal definition data for Digital, Analog or Event signals.
Sig_Def_Drop	Procedure to delete signal definition with related data for Digital, Analog or Event signals.
Data store and delete procedures.	
ASig_Store	Procedure to store a value to an analog signal.
DSig_Store	Procedure to store a value to a digital signal.
Sig_Store	Wrapper procedure for ASig_Store and DSig_Store.
Sig_Delete	Procedure to delete data within a time range related to a specific signal.
EV_Store	Procedure to store an event to an event signal.
EV_Delete	Procedure to delete events within a time range related to a specific event signal.
Data access procedures.	
Sig_Select_Log	Procedure to retrieve archived data within a time range for a specific Analog or Digital signal.
Sig_Select_Calc	Procedure to retrieve interpolated data from archive with specified time interval within a time range for Analog and Digital signals.
Sig_Select_Serie	Procedure to retrieve normal and “extreme values” from archive with specified number of periods within a time range for Analog and Digital signals.

2.1.5.1 Configuring signals

2.1.5.1.1 Sig_Def_Store

This procedure creates or updates Signal definition data, for Digital, Analog or Event signals.

Parameters	Default value	Description
@@SignalID		Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID’s will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
@@SignalName	NULL	A unique, user input, name for the signal. The SignalName can be changed. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
@@SignalDesc	NULL	The description of the signal Valid for: Analog, Digital, Event Data type: nvarchar(255)
@@SignalType	NULL	Bitmasked number to set the functionality of the tag.

		Bit: 1 – Digital signal 2 – Analog Signal 4 – Signal has events 8 – Text (for future use) 16 – Calculated 32 – Out 64 – Extend last value 128 – Signal has Multi values. Can be combined with Analog or Digital. 256 – Signal is periodic. The periodicty is set in the CalcFormula field. 512 – Disable recalc 1024 – Disable import 2048 – Disable correction 4096 – AutArch system diagnostic tag Valid for: Analog, Digital, Events, MultiValue Data type: smallint
@@LogBlocking	0	The LogBlocking property blocks a signal from being logged to the database. When LogBlocking is activated or deactivated, the OPCDA application needs to be restarted to effectuate the changes. The OPCDA application excludes a blocked tag from scheduling. 0 – Signal is not blocked 1 – Signal is blocked, and no data is archived. Valid for: Analog, Digital Data type: bit
@@ValueName	NULL	ValueName gives the possibility to use a description for the valuetype, e.g. “Power”. (For future use by connecting applications.) Valid for: Digital Data type: nvarchar(50)
@@ValueName0	NULL	The ValueName0 property is used to provide a description for when signal is 0 (zero), such as “CLOSED” or “OFF”. Valid for: Digital Data type: nvarchar(50)
@@ValueName1	NULL	The property is used to provide a description for when signal is 1 (one), such as “OPEN” or “ON”. Valid for: Digital Data type: nvarchar(50)
@@ValueUnit	NULL	The unit for the signal, e.g m/s, kPa etc. Valid for: Analog Data type: nvarchar(50)

@@RangeMin	NULL	RangeMin is used to set the default min range for the signal when opened in viewing applicatios, e.g. Trendviewer. Valid for: Analog Data type: float
@@RangeMax	NULL	RangeMax is used to set the default max range for the signal when opened in viewing applications, e.g. Trendviewer. Valid for: Analog Data type: float
@@Interpol	1 for analog signals, 0 for digital signals	If the Interpol property is set, the values are interpolated by straight line between each value. If the property is not set the values are plotted as steps. The Interpol property is only used with the return values from the Sig_Select_Calc and the Sig_Select_Serie procedures. 0 – No interpolation 1 – Interpolation Valid for: Analog, Digital Data type: bit More...
@@DisplayFormat	NULL	The DisplayFormat property sets the number of decimals for viewing applications, such as Trendview application. The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted. For example, when the value 14059,42450 is used in different display formats, see following table. Valid for: Analog, Digital, Events Data type: nvarchar(20) More...
@@OPCGroupID	-1	A unique ID identifying the OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
@@Category1	NULL	The category1 property is used for categorizing tags and filtering functions in viewing applications. Valid for: Analog, Digital Data type: nvarchar(20)

		More...
@@OPCItemID	NULL	The signals item id on the OPC server. Valid for: Analog, Digital Data type: nvarchar(1000)
@@MaxDivergence	0.5 for analog signals.	MaxDivergence property is used for setting the range on compression function. If the derivate value between the last archived value and the last received value are within MaxDivergence the value will not be archived. The MaxDivergence can be set manually by an administrator or automatically by the CompTune application. The CompTune application performs an advanced signal analyze of each signal to find the accuracy level of the signal. When finding that with good accuracy, tuning of the compression can be made. Valid for: Analog Data type: float More...
@@ActiveRunTime	1 for digital signals	This parameter is to display for the user which state that is the active and that is interesting as “operating state”. It determines the active runtime for the digital signal, 0 (zero) means that the signal is active if the value is 0 (zero) and vice versa. 0 – If 0 is the active runtime. 1 – If 1 is the active runtime. Valid for: Digital Data type: tinyint
@@ForceSaveTime	28800	The ForceSaveTime property is a time, in seconds, for how long new values can be rejected due to the compression function. If the ForceSaveTime has passed since last archived value, the previous value will be archived (by force). Valid for: Analog, Digital Data type: bigint More...
@@RejectSaveTime	NULL	RejectSaveTime overrides the compression function and rejects any values to be archived to the database, during this time (in seconds). Valid for: Analog, Digital Data type: bigint

		More...
@@InCalc	NULL	The InCalc property sets whether the signal is in a calculation. It is also referred to as CalcTrigger. 1 – in calculation. 0 – not in calculation. Valid for: Analog, Digital Data type: bit
@@BackLog	NULL	BackLog is the time in milliseconds, used to retrieve a previous value when the signal has an updated value that the compression will allow to be stored (outside MaxDivergence). The previous value is then stored at “number of BackLog” milliseconds before the new value, if the BackLog time is less than the last logged value. Without BackLog the trend for the signal might be unrepresented. If BackLog is set to 0 (zero) no BackLogging will occur. The BackLog value point is flagged with quality bit 512. Valid for: Analog Data type: bigint More...
@@CalcEnabled	NULL	CalcEnabled property determines if calculation is enabled or not. 0 – Calculation not enabled. 1 – Calculation enabled. Valid for: Analog, Digital, MultiValue Data type: bit
@@CalcType	NULL	Determines the type of calculation to execute for calculation signals. 0 – a user written formula (CalcFormula property) 2 – time calculated by average value 4 – time calculated by maximum value 5 – time calculated by minimum value 10 – Python calculated 11 – Average periodic 12 – Sum periodic Valid for: Analog, Digital, MultiValue Data type: tinyint
@@CalcFormula	NULL	The user written formula used for calculations. The formula calculation needs one or several input signals in the formula of which at least one input

		signal need to have the CalcTrigger set. The CalcTrigger triggers the calculation when the value of the signal changes. The signals that are used in the formula are stored in an relational table but they are edited by writing the signal names surrounded by "{" and "}" in the following way: `{SignalName1} + {SignalName2}` Valid for: Analog, Digital Data type: varchar(2000)
@@CalculationInterval	NULL	The interval for a time calculated TSQL-formula to be executed, in seconds. The CalculationInterval property is ignored if the calculation is a formula calculated signal. Valid for: Analog Data type: bigint
@@GroupingPoint	NULL	GroupingPoint is used to decide where to put the timestamp for a time calculated TSQL-formula. In procedures for storing formulas and internally in the database the following codes are used: 1 – The calculated value and timestamp is placed in the beginning of the calculated period. 2 – The calculated value and timestamp is placed in the middle of the calculated period. 4 – The calculated value and timestamp is placed at the end of the calculated period. Valid for: Analog Data type: tinyint

@@CalcReferenceTime	NULL	A reference time for when the calculation will be triggered. Used as reference time for time calculations.
		When using many average calculation tags, the calculations can be spread in time. This is to avoid accumulation of concurrent calculations. To spread the calculations, the CalcReferenceTime property can be used. The default is that calculation occurs at the beginning of next period using
		`@@CalcReferenceTime = '2001-01-01 00:00:00'`
		but if following setting is used instead
		`@@CalcReferenceTime = '2001-01-01 00:00:10'`
		the calculation is moved to take place 10 seconds later.
		
		Valid for: Analog
		Data type: datetime

@@TuneStatus	NULL	The TuneStatus property is used by the CompTune application to determine the status of the compression for the signal. CompTune will change this property as different stages of the tuning are

		performed. The value is bit coded as: 1- Represents that the signal should collect data for tuning analyze 2 - Represents that the database has collected enough RAW data for this signal to start the tuning analyze. 4 - Represents that the signal has finished its tuning analyze. 8 - Represents that the tuning analyze has Checked the signals tuning. 16 - Represents that the tuning is Enabled for this signal. 32 - Represents that the value of the reference tag is to low and the database has stopped collecting RAW data. 16384 - Represents that the signal is imported by XMZ-import. Valid for: Analog Data type: smallint
@@ExportFlag	NULL	The ExportFlag property is set to mark the signal for export. It is used by the AAExport application for exporting data to encrypted exportfiles. By setting combinations for this value, different export scope can be performed. 0 – no export 1 – signal export The flag is used in a bit coded way that makes it possible to flag a signal to be exported by different export jobs, se the AAExport documentation for examples. Valid for: Analog, Digital Data type: smallint
@@EventBlocking	0	The EventBlocking property activates/inactivates events from being logged to the database. 0 = Event is not blocked 1 = Event is blocked and no data is archived. This attribute can be set anytime and changes take effect directly on archiving. Valid for: Events Data type: bit
@@TuneCount	NULL	The number of processed compression auto tunings. For internal use only. Valid for: Analog Data type: smallint
@@Custom01	NULL	The Custom01 - 05 are free signal properties to be used by customer. Valid for: Analog, Digital
@@Custom02	NULL	

@@Custom03	NULL	Data type: nvarchar(100)
@@Custom04	NULL	
@@Custom05	NULL	
@@SignalName2	NULL	The name of the signal in a alternative language. Valid for: Analog, Digital, Events, MultiValue Data type: nvarchar(1000)
@@SignalDesc2	NULL	The description of the signal in a alternative language. Valid for: Analog, Digital, Event Data type: nvarchar(255)
@@ValueUnit2	NULL	The unit for the signal, e.g m/s, kPa etc. in a alternative language Valid for: Analog Data type: nvarchar(50)
@@RangeMin2	NULL	RangeMin2 is used to set the default min range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit. Valid for: Analog Data type: float
@@RangeMax2	NULL	RangeMax2 is used to set the default max range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit. Valid for: Analog Data type: float
@@DisplayFormat2	NULL	The DisplayFormat2 property sets the number of decimals for viewing applications, such as Trendview application when using an alternative engineering unit. The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted. For example, when the value 14059,42450 is used in different display formats, see following table. Valid for: Analog, Digital, Events Data type: nvarchar(20) More...
@@SourceScale	NULL	SourceScale and SourceOffset is used to rescale an input value if the input has a wrong measuring

		scale. If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula: `The Stored Value = The Input Value * SourceS` Valid for: Analog Data type: float
@@SourceOffset	NULL	SourceOffset and SourceScale is used to rescale an input value if the input has a wrong measuring scale. If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula: `The Stored Value = The Input Value * SourceS` Valid for: Analog Data type: float
@@OPCGroupName	NULL	Name of the OPC group. Valid for: Analog, Digital Data type: nvarchar(50)
@@SourceRangeMin	NULL	If SourceRangeMin and SourceRangeMax is used when creating or updating a Signal this value can be used to calculate and set the correct values of: SourceScale and SourceOffset based on RangeMin and RangeMax. The SourceScale is calculated according to: SourceScale = (RangeMax - RangeMin) / (SourceRangeMax - SourceRangeMin) And the SourceOffset is calculated according to: SourceOffset = RangeMin - (SourceRangeMin * SourceScale) Valid for: Analog Data type: float
@@SourceRangeMax	NULL	If SourceRangeMin and SourceRangeMax is used when creating or updating a Signal this value can be used to calculate and set the correct values of: SourceScale and SourceOffset based on RangeMin and RangeMax. The SourceScale is calculated according to: SourceScale = (RangeMax - RangeMin) / (SourceRangeMax - SourceRangeMin) And the SourceOffset is calculated according to: SourceOffset = RangeMin - (SourceRangeMin * SourceScale)

		Valid for: Analog Data type: float
@@TdrOPCGroupID	NULL	A unique ID identifying the Trigged Data Recorder OPC group and is generated by the system. Valid for: Analog, Digital Data type: smallint
@@TdrOPCGroupName	NULL	A unique Name identifying the Trigged Data Recorder OPC group. Valid for: Analog, Digital Data type: nvarchar(50)
@@TdrOPCItemID	NULL	The signals item id on the OPC server for a Trigged Data Recorder signal. Valid for: Analog, Digital Data type: nvarchar(1000)
@@TdrPreBuffer	NULL	The amount of data that will be saved to the database before the signal has been trigged by the Trigged Data Recorder. The time is specified in fractions of seconds. Valid for. Analog and Digital signals Data type: real
@@TdrPostBuffer	NULL	The amount of data that will be saved to the database after the signal has been trigged by the Trigged Data Recorder. The time is specified in fractions of seconds. Valid for. Analog and Digital signals Data type: real
@@TdrTrigger	NULL	Defines if the Signal is a trigger for the Trigged Data Recorder. 1 = The Signal is a trigger. Valid for. Analog and Digital signals Data type: bit
@@Tdrlevel	NULL	Defines on which flank the trigger Signal should trig the Trigged Data Recorder 1 = The Signal triggers when going from 0 to 1 0 = The Signal triggers when going from 1 to 0 Valid for. Analog and Digital signals Data type: bit
@@TdrLogSignals	NULL	A list of signals that should be logged by the Trigged Data Recorder when trigged by this signal. The signals are stored in an relational table but they are edited by writing an list of signal names in the following way:

		{SignalName1}{SignalName2}{SignalName3}... Valid for: Analog, Digital Data type: nvarchar(2000)
@@ChangedBy	NULL	The user name that made the last configuration change to this signal. All changes made to the signal are persisted in the SignalRev table. Data type: nvarchar(200)
@@Origin	NULL	Describes the origin of the data for this signal. Data type: nvarchar(200)

Return values	Data type	Description
SignalID	Smallint	Store Succeeded = SignalID.is returned. Failure = 0 or negative number is returned.

2.1.5.1.1.1 Examples

The following samples are written in SQL code to legible the syntax in a “straight” way.
The operations can also be made by the AutArchAdmin application in the graphical user interface.

The minimum of parameters required to create an analog signal is according to this example.

```
Sig_Def_Store @@SignalName = 'INDOORTEMP', @@SignalType = 2
```

The SignalID for the created signal is returned and the following properties are default set to:
Logblocking = 0
Interpol = 1
MaxDivergence = 0.5
ForceSaveTime = 28800
RejectSaveTime = 0
Backlog = 0

To get data from an OPCServer the signal must be related to an OPCTagGroup and a correct entered OPCItemID must exist for the signal.

Make sure that an OPCServer is created in the OPCServer table and an OPCGroup is created in the OPCGroup table first to be able to connect the tag to a Group and Server.

```
Sig_Def_Store @@SignalName = 'INDOORTEMP', @@SignalType = 2, @@OPCGroupID = 16, @@OPCItemID =
```

When modifying signals you can either use SignalName or SignalID.

```
Sig_Def_Store @@SignalID = 4949, @@SignalType = 2, @@OPCGroupID = 16, @@OPCItemID = 'Triangle
```

These are the minimum parameters that are required to create a digital signal.

```
Sig_Def_Store @@SignalName = 'COOLER', @@SignalType = 1
```

The SignalID for the created signal is returned and the following properties are default set to:
Logblocking = 0
Interpol = 0

ActiveRunTime = 1
ForceSaveTime = 28800
RejectSaveTime = 0

Normally when creating a calculated signal, the graphic user interface in the Admin application should be used where validation of the calculation etc. can be done in an easy way.

This example of a simple formula uses the INDOORTEMP in Cº and stores the calculated value as a new value in Fº in the new tag INDOORTEMP_F.

```
Sig_Def_Store @@SignalName = 'INDOORTEMP_F', @@SignalType = 18, @@CalcEnabled = 1, @@CalcType =
```

The calculated signal is now created and enabled, but there is nothing triggering the calculation. To trigger the calculation, at least one of the input signals to the formula has to be set to triggering.

E.g. this will update the configuration of INDOORTEMP to trigger calculations.

```
Sig_Def_Store @@SignalName = 'INDOORTEMP', @@SignalType = 2, @@Incalc = 1
```

Sum of two signals:

```
{GENERATOR1} + {GENERATOR2}
```

In this case, when more than one signal is used in the formula, one of the signals or both signals can be used as triggers.
If only GENERATOR1 is used as trigger, the calculations are only executed when value changes for GENERATOR1. When calculation is triggered, it uses the updated changed value of GENERATOR1 and the last known value of GENERATOR2. If the formula is triggered by both signals, the result will be a more frequent calculation that will result in more CPU load.

Using constants:

```
{GENERATOR1} + (10 / 3)
```

When constants are used, they should be converted into the data type float to avoid loosing precision in the calculation.

The previous example should be expressed like this:

```
{GENERATOR1} + (convert(float, 10) / convert(float, 3))
```

By doing it this way, all constants are treated as float and a correct calculation with full accuracy is guaranteed.

Conditional calculation:

```
CASE
  WHEN {GENERATOR1} > 0.5 THEN 1
  WHEN {GENERATOR2} > 0.5 THEN 1
  ELSE 0
END
```

The formula above returns 1 when at least one of the generators are running.

The formulas are using the same syntax as SQL Server Transact SQL, see Formulas used for calculation signals

This example shows how to create a one minute average calculated signal for the indoor temperature.

```
Sig_Def_Store @@SignalName = 'INDOORTEMP_AVG', @@SignalType = 18, @@CalcEnabled = 1, @@CalcTyp
```

The result is time stamped in the middle of each period. (Grouping point =2)

When using many average calculation tags, the calculations can be spread in time.
This is to avoid accumulation of concurrent calculations. To spread the calculations, the CalcReferenceTime property can be used.

The default is that calculation occurs at the beginning of next period using `@@CalcReferenceTime = '2001-01-01 00:00:00'`
but if following setting is used instead `@@CalcReferenceTime = '2001-01-01 00:00:10'` the calculation is moved to take place 10 seconds later.

2.1.5.1.2 Sig_Def_Drop

This procedure deletes both the configuration and all data related to the Signal.

Parameters	Default value	Description
@@SignalID		Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint

2.1.5.1.2.1 Example, drop a signal with all stored data

```
Sig_Def_Drop 4949
```

Warning! This will erase all data and all configurations for the signal. If there is a lot of data, this operation can take a while to process.
(The procedure will delete the signal without returning any status value.)

2.1.5.2 Store data

Database procedures for storing data to AutArch.

2.1.5.2.1 ASig_Store

This procedure stores an analog signal.

If the signal value is within the compression range or status is not changed, and the ForceSaveTime is not passed, the procedure is only updating the ASignalDef table. Else it is also adding a new record to the ASignalLog table.

If SignalTime is earlier than previously stored values for this signal the value is not stored to the database. Exception when "Override" is used. With "Override" activated, archiving will be forced.

Parameters	Default value	Description
@@SignalID		Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
@@SignalTime		The timestamp of the stored value. Data type: datetime
@@SignalValue		The value to be stored. Data type: float
@@Quality	192	The Quality attribute represents the quality state for a signals

		value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...
@@Override	0	Without override, normal compression evaluation will take place when storing the value. With override activated, the value is stored directly to the archive without compression evaluation. 0 – Override inactivated. 1 – Override activated. Data type: tinyint
@@CreateBacklog		When set to 1, creation of backlog values is disabled. Data type: bit
@@NoScale		When set to 1, the procedure should not scale the value according to the SourceScale and SourceOffset setting. Data type: bit

Return values	Data type	Description
ReturnStatus	int	0 – value is not stored 1 – value is stored

2.1.5.2.1.1 Examples

The following samples are written in SQL code to legible the syntax in a “straight” way.

This procedure call will store the value to the database if there is no data already in the database with a more recent time.

```
ASig_Store 1234, '2005-01-24 00:00:00', 123.456, 192
```

The override function is mainly used for importing old data in an existing database.
The override parameter forces the database to store the value even if there already is data with a more recent time in the database.
The procedure can not replace data if there already exists data with the exact same timestamp, normally a period of data must be cleared to make room for the new data that should replace the old data, see Sig_Delete.

```
ASig_Store 4949, '2005-01-24 00:00:00', 123.456, 192, 1
```

2.1.5.2.1.2 Compression principle

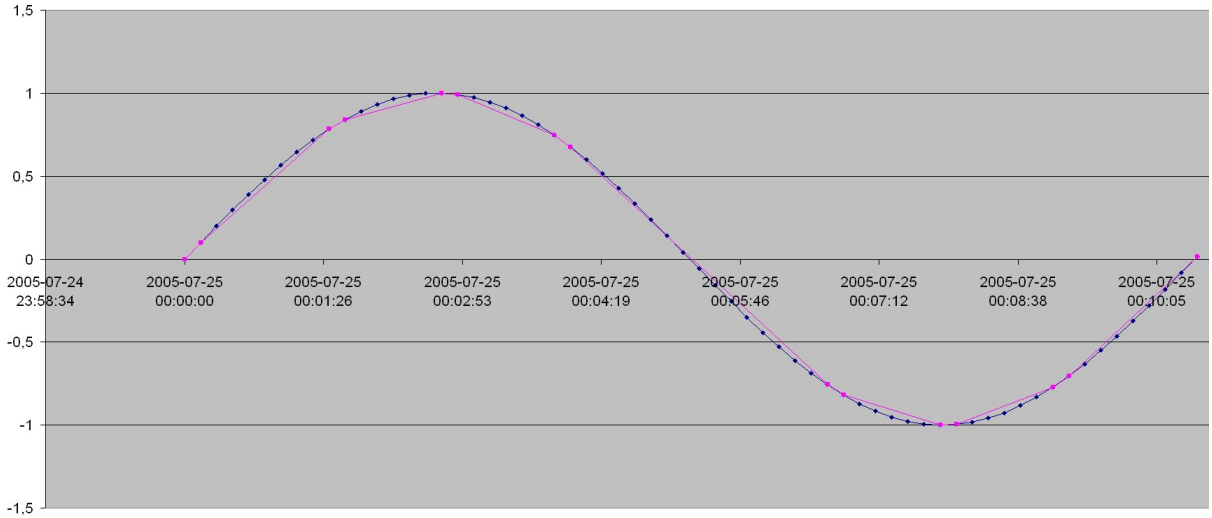
The compression level is tuned with a parameter “MaxDivergence”.
This parameter sets the max allowed divergence of a value compared to previous value and the change rate of the value.

In following example, this is illustrated with a signal that follows a sinus curve.

The maximal divergence is set to +/- 0,05. This will result in that only 15 values of the 64 sampled values are stored. This gives a compression rate of 77%.
A lower “Max Divergence” value will result in lower compression rate in the database, i.e. more

detailed data.

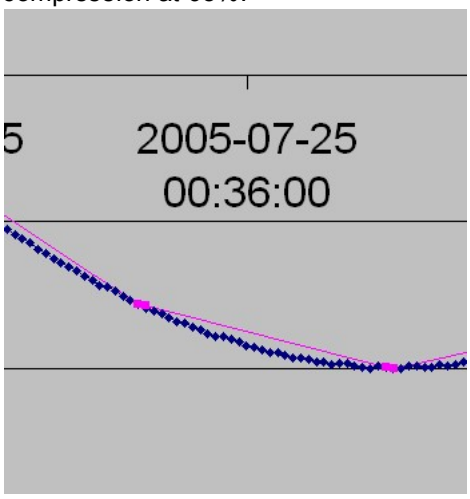
In the diagram below, the measured data is displayed in blue color. The values that after compression test have been stored in the database are displayed in purple color.



The example in the diagram above is a theoretic example that is based on a “clean” signal. In reality a signal often contain signal noise.

In the example below we illustrate the same sinus curve, but with noise.
In this example, there are 327 measured values, but still only 15 values are stored in the database.

This example shows that the same conditions but with more noise on the signal will result in a compression at 95%.



If there is no interest in storing values more often than e.g. once a minute, a parameter “Reject Save Time” can be set to 60 for each signal.
This will result in changes that appear more often than within this period will not be stored.

If there are signals that don’t get updated for a long period, or has a very slow or stable course or even are complete fixed then the parameter “Force Save Time” can be set to assure that the value will be stored within this time span.
If “Force Save Time” is set to 3600, the signal will store values in the database every hour regardless of whether the signal has been updated or not.

By default the “Reject Save Time” is set to 0 sec, i.e. all values are treated.

The “Force Save Time” is by default set to 28800 sec. (8 hours) i.e. all values are forced to be saved every 8 hour even if there has not been any updated values.

Quality changes also affect the storage principle to the database.
If a signal value normally has the quality “Good”, but changes to “Bad” with the same value, an archive of that quality change will be saved to the database.

The last/current value of a signal is always updated in the database together with its timestamp, status and change course (derivate).

2.1.5.2.2 DSig_Store

This procedure stores a digital signal and recalculates its operation timer.

If the signal value is not changed or status is not changed, and the ForceSaveTime is not passed, the procedure is only updating the DSignalDef table. Else it is also adding a new record to the DSignalLog table.

If SignalTime is earlier than previously stored values for this signal the value is not stored to the database.
Exception if “Override” is used. With “Override” activated, archiving will be forced.

Parameters	Default value	Description
@@SignalID		Unique ID identifying the signal and is generated by the system. The ID cannot be changed. If signals are imported to the database, new SignalID’s will be created if necessary. Valid for: Analog, Digital, Event, MultiValue Data type: smallint
@@SignalTime		The timestamp of the stored value. Data type: datetime
@@SignalValue		The value to be stored. Only 0 and 1 are legal values. Data type: tinyint
@@Quality	192	The Quality attribute represents the quality state for a signals value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...
@@Override	0	0 – Override inactivated. 1 – Override activated. Data type: tinyint
@@TotalChanges	NULL	For internal use. Only used if Override is activated. Data type: bigint
@@ServiceChanges	NULL	
@@TotalRunTime0	NULL	For internal use. Only used if Override is activated. Data type: float
@@TotalRunTime1	NULL	

@@ServiceRunTime0	NULL	
@@ServiceRunTime1	NULL	

Return values	Data type	Description
ReturnStatus	Int	0 – value is not stored 1 – value is stored

2.1.5.2.2.1 Examples

The following samples are written in SQL code to legible the syntax in a “straight” way.

This example will store the value ”1” to the database if there is no data with a more recent time in the database for that signal.

```
DSig_Store 1234, '2005-01-24 00:00:00', 1, 192
```

When storing a digital value, the operating time is recalculated in the following way:
- If the previous value was “0” and the new value is “1” then TotalChanges and ServiceChanges are incremented with 1.
- If the new value is “1” then TotalRunTime1 and ServiceRunTime1 is incremented with the time difference between the previous value and the timestamp of the new value.
- If the new value is “0” then TotalRunTime0 and ServiceRunTime0 is incremented with the time difference between the previous value and the timestamp of the new value.

The operation time is only recalculated if both the previous value and the new value have good quality .

This mode is only for internal use to make it possible to import data that already has calculated operation time. If this procedure is used, all data to the last value has to be replaced. The procedure does not recalculate the operation time for subsequent values.

This example will store the digital value and its operation time to the database.
The RunTime parameters are in seconds.

```
DSig_Store 1234, '2005-01-24 00:00:00', 1, 192, 1, 2000, 1000, 31536000, 31536000, 8640000, 86
```

If this procedure is used, a range of data must first be cleared to make room for the new data with the procedure Sig_Delete.

2.1.5.2.3 Sig_Store

This procedure is a wrapper procedure for ASig_Store and DSig_Stor.
The procedure can be used to store both analog and digital values. The procedure can take ether a SignalID or a SignalName as the identity for the signal.

Parameters	Default value	Description
@@SignalID		Can be either SignalID as a smallint or SignalName as a varchar.
@@SignalTime		The timestamp of the stored value. Data type: datetime
@@SignalValue		Use a float for analog signals and a tinyint for digital signals.

@@Quality	192	The Quality attribute represents the quality state for a signals value. The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status. The 8 Quality bits are arranged as follows: QQSSSSLL. Valid for: Analog, Digital, Events Data type: smallint More...
@@Override	0	0 – Override inactivated. 1 – Override activated. Data type: tinyint
@@TotalChanges	NULL	For internal use. Only used if Override is activated.
@@ServiceChanges	NULL	For internal use. Only used if Override is activated.
@@TotalRunTime0	NULL	For internal use. Only used if Override is activated.
@@TotalRunTime1	NULL	For internal use. Only used if Override is activated.
@@ServiceRunTime0	NULL	For internal use. Only used if Override is activated.
@@ServiceRunTime1	NULL	For internal use. Only used if Override is activated.

Return values	Data type	Description
ReturnStatus	Int	0 – value is not stored 1 – value is stored

2.1.5.2.4 EV_Store

This procedure stores an Event.

When configuring the database for Analog and digital signals from an OPCDAServer, the tag configuration must be done correct to get any data into the system.

Events can come from an OPCAEServer without being a created Event signal in the database. The signal will then be created automatically by the procedure.

If the incoming event has the same SignalName as an existing Analog or Digital signal in SignalDef, this base tag will be used as Analog and Event signal or Digital and Event signal. In this case, the event definition will be related to the existing signal.

The Event is stored in SignalDef, EventDef and EventLog. If EventBlocking in SignalDef is set to 1 the Event is not stored.

Parameters	Default value	Description
@@Source		The name of the Event signal or it's related "source" signal if the event is e.g. a limit on an Analog signal. The Source parameter is defined in the OPCAE standard according following text "An OPCSource may be a process tag (e.g. FIC101) or possibly a device or subsystem. An OPCSource may be an OPCItem if the OPC Event Server is (or is

		associated with) an OPC Data Access Server." In AA database, we have then used this parameter as the "SignalName" for a signal. Datatype: nvarchar(1000)
@@Time		The timestamp for when the event was logged. Valid for: Events Data type: datetime
@@Type	NULL	OPCAE Standard field. EventType specifies what type of event it is. Predefined according to the standard is: OPC_SIMPLE_EVENT OPC_CONDITION_EVENT OPC_TRACKING_EVENT OPC_ALL_EVENTS Valid for: Events Data type: tinyint
@@EventCategory	NULL	OPCAE Standard field. EventCategories define groupings of events supported by an OPC Event server. Examples of event categories might include "Process Events", "System Events", or "Batch Events". Event categories may be defined for all event types, i.e. Simple, Tracking, and Condition-Related. However, a particular event category can include events of only one type. A given Source (e.g. "System" or "FIC101") may generate events for multiple event categories. Names of event categories must be unique within the event server. The definition of event categories is server specific and is outside the scope of this specification. The name of the event category is included in every event notification. Event subscriptions may be filtered based on event category. Valid for: Events Data type: nvarchar(200)
@@Severity	NULL	OPCAE Standard field. The severity value is an indication of the urgency of the sub-condition. This is also commonly called 'priority', especially in relation to process alarms. Values will range from 1 to 1000, with 1 being the lowest severity and 1000 being the highest. Typically, a severity of 1 would indicate an event which is informational in nature, while a value of 1000 would indicate an event of catastrophic nature which could potentially result in severe financial loss or loss of life. Valid for: Event Data type: smallint More...

@@Message	NULL	Message text which describes the event. For condition-related events, this will generally include the description property of the active sub-condition. Valid for: Events Data type: nvarchar(1000)
@@ActorID	NULL	OPCAE Standard field. ActorID is the identifier of the OPC Client which acknowledged the condition, which is maintained as the AcknowledgerID property of the condition. This is included in event notifications generated by condition acknowledgments. Valid for: Events Data type: nvarchar(50)
@@ConditionName	NULL	OPCAE Standard field. The name of the condition related to this event notification. (The QueryConditionNames method gives clients a means of finding out the specific condition names which the event server supports for the specified event category. This method would typically be invoked prior to specifying an event filter. Condition names are server specific. The number of condition names returned will vary depending on the sophistication of the server, but is expected to be less than 30 for most servers, making this interface more appropriate than a custom enumerator. It is expected that the results of the Query will be fairly 'stable' in most situations. However, the Server is in fact allowed to change the available selection at any time. Therefore, a Client should do (or at least allow as an option) a fresh Query every time a selection is to be presented to the end user.) Valid for: Events Data type: nVarChar(200)
@@SubConditionName	NULL	OPCAE Standard field. Name of current sub-condition, for multi-state conditions. For a single-state condition, this contains the condition name. Valid for: Events Data type: nvarchar(200)
@@ChangeMask	NULL	OPCAE Standard field. ChangeMask indicates to the client which properties of the condition have changed, to cause the server to send the event notification. Indicates to the client which properties of the condition have changed, to have caused the server to send the event notification. It may have one or

		more of the following values: OPC_CHANGE_ACTIVE_STATE = 1 (The condition's active state has changed.) OPC_CHANGE_ACK_STATE = 2 (The condition's acknowledgment state has changed.) OPC_CHANGE_ENABLE_STATE = 4 (The condition's enabled state has changed.) OPC_CHANGE_QUALITY = 8 (The ConditionQuality has changed.) OPC_CHANGE_SEVERITY = 16 (The severity level has changed.) OPC_CHANGE_SUBCONDITION = 32 (The condition has transitioned into a new sub-condition.) OPC_CHANGE_MESSAGE = 64 (The event message has changed (compared to prior event notifications related to this condition).) OPC_CHANGE_ATTRIBUTE = 128 (One or more event attributes have changed (compared to prior event notifications related to this condition).) If the event notification is the result of a Refresh, these bits are to be ignored. For a "new event",OPC_CHANGE_ACTIVE_STATE is the only bit which will always be set. Other values are server specific. (A "new event" is any event resulting from the related condition leaving the Inactive and Acknowledged state.) Valid for: Events Data type: tinyint
@@NewState	NULL	OPCAE Standard field NewState is A WORD bit mask of three bits specifying the new state of the condition: OPC_CONDITION_ACTIVE (= 2), OPC_CONDITION_ENABLED (= 1) , OPC_CONDITION_ACKED (= 4) Valid for: Event Data type: tinyint
@@ConditionQuality	NULL	OPCAE Standard field. ConditionQuality Indicates the quality of the underlying data items upon which this condition is based. Condition Quality is stored as "Quality" in AA Eventlog table. Valid for: Events Data type: smallint
@@AckRequired	0	OPCAE Standard field. An indicator as to whether or not an acknowledgement is required. Many event notifications related to conditions do not normally

		require an acknowledgment, e.g. the receipt of an acknowledgment or the transition to the inactive state. Furthermore, some conditions may be configured (using facilities outside the scope of this specification) to not require acknowledgment even for transitions into the condition, or for transitions among sub-conditions (e.g. transition into LevelAlarm or transition from HighAlarm to HighHighAlarm). In this case, it is the responsibility of the server to automatically place the condition into the Acknowledged state, since an acknowledgment will never be received. Valid for: Events Data type: bit
@@ActiveTime	NULL	Used in Ev_Store procedure as ActiveTime. This is the same field as EventActiveTime. See description for this attribute in this appendix. Valid for: Events Data type: DateTime
@@Cookie	NULL	OPCAE Standard field. Server defined cookie associated with the event notification. This value can be used by a client when acknowledging the condition. This value is opaque to the client. Valid for: Events Data type: int
@@SignalDesc	NULL	Description of the signal. This is not a standard property from OPCAE standard. But when using RS232 interface or if OPCAEServer vendor has this as a “CustomerAttribute”, this can be used as description for the eventtag. If the event is related to an Analog or Digital signal this information comes from the Analog or Digital signals description field. Data type: nvarchar(255)
@@EventDesc	NULL	Additional field not specified in OPCAE standard. This field is used by Siemens PCS7 system that uses this field as own defined customer attributes. Valid for: Events Data type: nvarchar(1000)
@@EventStatus	NULL	Additional field not specified in OPCAE standard. Used for Siemens PCS7 system that uses this as own defined customer attributes. Valid for: Events Data type: varchar(10)
@@EventServerNode	NULL	Additional field not specified in the OPCAE standard. This field can be used to specify the collecting eventservernode when there are several eventservers connected to one AA system. The

		name of the eventserver is specified in the OPCAEClient applications config file. The parametername in the config file is “CollEventServerNode”. Valid for: Events Data type: nvarchar(200)
@@EventCategoryID	NULL	EventCategoryID as a numeric value describing the Category of an Event. Valid for: Events Data type: int
@@CustomEvDef01	NULL	The CustomEvDef01 - 03 are free Event definition properties to be used by customer. Valid for: Event Data type: nvarchar(50)
@@CustomEvDef02	NULL	
@@CustomEvDef03	NULL	
@@BackColor	NULL	The BackColor is an Event log property that can be used to store the events background color. The color should be represented as a HTML color specification. Valid for: Event Data type: varchar(10)
@@ForeColor	NULL	The ForeColor is an Event log property that can be used to store the events foreground color. The color should be represented as a HTML color specification. Valid for: Event Data type: varchar(10)
@@CustomEvLog01	NULL	The CustomEvLog01 - 03 are free Event log properties to be used by customer. Valid for: Event Data type: nvarchar(50)
@@CustomEvLog02	NULL	
@@CustomEvLog03	NULL	

2.1.5.2.4.1 Store a simple event

This example will store an event related to existing “INDOORTEMP” and change the signaltype of the signal from 2 to 6 (2= Analog + 4= Has events = 6).
If INDOORTEMP didn’t already exist, the tag would have created a new tag of signaltype 4.

```
EV_Store @@Source = 'INDOORTEMP', @@Time = '2005-07-25 14:00:01', @@Message = 'Too hot', @@Con
```

(The procedure will store the event without returning any status value.)

2.1.5.3 Delete data

Database procedures for deleting data from AutArch.

2.1.5.3.1 Sig_Delete

This procedure deletes signal records between StartTime and EndTime.

Parameters	Default value	Description
@@SignalID		Can be either SignalID as a smallint or SignalName as a varchar.
@@StartTime	NULL	Data type: datetime
@@EndTime	NULL	Data type: datetime

2.1.5.3.1.1 Delete a range of data for a signal

Warning! This will erase all data between the selected dates for the signal, if there is a lot of data, this operation can take a while to process.

```
Sig_Delete 4949, '2005-01-01', '2006-01-01'
```

(The procedure will delete the signal values without returning any status value.)

2.1.5.3.2 EV_Delete

The procedure deletes Event records between StartTime and EndTime.

Parameters	Default value	Description
@@SignalID		Can be either SignalID as a smallint or SignalName as a varchar.
@@StartTime	NULL	Data type: datetime
@@EndTime	NULL	Data type: datetime

2.1.5.3.2.1 Delete a range of events for a signal

Warning! This will erase all events between the selected dates for the signal, if there is a lot of data, this operation can take a while to process.

```
EV_Delete 1234, '2005-01-01', '2006-01-01'
```

(The procedure will delete the event records without returning any status value.)

2.1.5.4 Retrieve data

Database procedures for retrieving data from AutArch.

2.1.5.4.1 Example prerequisites

For the following examples following sample data is used:

2.1.5.4.1.1 Archived analog data

An analog signal with the SignalID: 4949 and SignalName 'INDOORTEMP', the signal is connected to an OPCServer.
Data in ASignalLog table

ChangeTime	ChangeValue	Quality
2005-01-25 00:00:00	0	192
2005-01-25 00:00:10	0,099833417	192
2005-01-25 00:01:30	0,78332691	192

2005-01-25 00:01:40	0,841470985	192
2005-01-25 00:02:40	0,999573603	192
2005-01-25 00:02:50	0,99166481	192
2005-01-25 00:03:50	0,745705212	192
2005-01-25 00:04:00	0,675463181	192
2005-01-25 00:06:40	-0,756802495	192
2005-01-25 00:06:50	-0,818277111	192
2005-01-25 00:07:50	-0,999923258	192
2005-01-25 00:08:00	-0,996164609	192
2005-01-25 00:09:00	-0,772764488	192
2005-01-25 00:09:10	-0,705540326	192
2005-01-25 00:10:30	0,0168139	192

Above data is stored in the Log table after compression.

2.1.5.4.1.2 Last recieved analog data

Data in ASignalDef table.

LastChangeTime	LastValue	LastQuality
2005-01-25 00:11:40	0,035675	192

This is the latest data in the database.
When a new value is stored in the database that is outside the compression range, this value will be archived into the log table.

2.1.5.4.1.3 Archived digital data

A digital signal with the SignalID: 4950 and SignalName 'COOLER', the signal is connected to an OPCServer and configured to interpret value "1" as Active.
Data in DSignalLog table.

ChangeTime	ChangeValue	Quality
2005-01-25 00:00:00	1	192
2005-01-25 00:00:30	0	192
2005-01-25 00:01:00	1	192
2005-01-25 00:01:30	0	192
2005-01-25 00:02:00	1	192
2005-01-25 00:02:30	0	192
2005-01-25 00:03:00	1	192
2005-01-25 00:03:30	0	192
2005-01-25 00:04:00	1	192
2005-01-25 00:04:30	0	192
2005-01-25 00:05:00	1	192
2005-01-25 00:05:30	0	192
2005-01-25 00:06:00	1	192

2005-01-25 00:06:30	0	192

2.1.5.4.1.4 Last recieved digital data

Data in DSignalDef table.

LastChangeTime	LastValue	LastQuality
2005-01-25 00:07:00	1	192

This is the latest data in the database.
When a new value is stored in the database that is outside the compression range, this value will be archived into the log table.

2.1.5.4.1.5 Data in OPCServer table

Data in OPCServer table.

LastForceUpdTime
2005-01-25 00:12:20

A watchdog in the OPCDA client updates this time to indicate when the system last had a good communication with the OPCServer.
This time is used to compensate for signal data that is suppressed in the OPCDA client's dead band filter. (This is only valid for signals that are connected to an OPCServer). The updated watchdog signal is only activated for each server as long as status for that server is "good".

2.1.5.4.2 Sig_Select_Log

This procedure returns a record set with all logged analog or digital signal values between StartDate and EndDate.

If StartDate and EndDate is omitted the procedure will return the latest record of the signal.

If StartDate is given and EndDate is omitted, the procedure will return the latest record prior to StartDate.
If StartDate is omitted and EndDate is given, the procedure will return the first record after EndDate.

Parameters	Default value	Description
@@SignalID		Can be either SignalID as a smallint or SignalName as a varchar.
@@StartDate	NULL	Data type: datetime
@@EndDate	NULL	Data type: datetime
@@Property	0	Defines what should be returned as ChangeValue, only used for digital signals 0 – Signal value 1 – Active TotalRunTime 2 – Inactive TotalRunTime 3 – TotalChanges 4 – Active ServiceRunTime 5 – Inactive ServiceRunTime 6 - ServiceChanges
@@Now	The SQL server's internal system clock	In a case where the signal is of type "ExtendLastValue", this is the time the value should be extended to. Data type: datetime

Return values	Data type	Description
ChangeTime	datetime	
ChangeValue	tinyint / bigint / float	Depending on signaltype and property. Tinyint for digitals signal values, float for analog values and bigint for all other properties.
Quality	smallint	

2.1.5.4.2.1 Examples

The following samples are written in SQL code to legible the syntax in a “straight” way.

The procedure can be called with either SignalName or SignalID eg.

```
Sig_Select_Log 'INDOORTEMP', '2005-01-24 00:00:00', '2005-01-26 00:00:00'
```

Or

```
Sig_Select_Log 4949, '2005-01-24 00:00:00', '2005-01-26 00:00:00'
```

In this example, the selected time range starts before first stored value and ends after latest stored value.

The result is according to table below.

ChangeTime	ChangeValue	Quality	
2005-01-25 00:00:00	0	192	These values are compressed and retrieved from ASignalLog table.
2005-01-25 00:00:10	0,099833417	192	
2005-01-25 00:01:30	0,78332691	192	
2005-01-25 00:01:40	0,841470985	192	
2005-01-25 00:02:40	0,999573603	192	
2005-01-25 00:02:50	0,99166481	192	
2005-01-25 00:03:50	0,745705212	192	
2005-01-25 00:04:00	0,675463181	192	
2005-01-25 00:06:40	-0,756802495	192	
2005-01-25 00:06:50	-0,818277111	192	
2005-01-25 00:07:50	-0,999923258	192	
2005-01-25 00:08:00	-0,996164609	192	
2005-01-25 00:09:00	-0,772764488	192	
2005-01-25 00:09:10	-0,705540326	192	
2005-01-25 00:10:30	0,0168139	192	
2005-01-25 00:11:40	0,035675	192	This value is the last stored and comes from ASignalDef table.
2005-01-25 00:12:20	0,035675	192	This value is a copy of the ASignalDef value but timestamped with last updated watchdog time from OPCServer table. This is used as compensation when a value is filtered out by the dead band filter in OPCDA client or in the DCS system.

If the signal had been a “Manual Editable” signal, the last value will be “current time” according to following:

ChangeTime	ChangeValue	Quality	

2005-01-26 00:00:00	0,035675	192	This value is a copy of the ASignalDef value but time stamped with current time. This is because a manual value shall keep its last value forever or until it gets a new value.

If instead the signal would be a “Calculated” signal, there wouldn’t be an extra last value. The last value returned is the value from the ASignalDef table.

```
Sig_Select_Log 'INDOORTEMP', '2005-01-25 00:03:00'
```

The result will be one or zero records.

ChangeTime	ChangeValue	Quality
2005-01-25 00:02:50	0,99166481	192

This is used when accessing real-time values from database.

```
Sig_Select_Log 'INDOORTEMP'
```

The result will be one or zero (if no data has been collected) records according to table below.

ChangeTime	ChangeValue	Quality	
2005-01-25 00:12:20	0,035675	192	This value is a copy of the ASignalDef value but timestamped with last updated watchdog time from OPCServer table. This is used as compensation when a value is filtered out by the dead band filter in OPCDA client or in the DCS system.

If the signal had been a “Manual Editable” signal, the last value will be “current time” according to following: (if the time when the procedure was called was 2005-01-27 12:00):

ChangeTime	ChangeValue	Quality	
2005-01-27 12:00:00	0,035675	192	This value is a copy of the ASignalDef value but time stamped with current time. This is because a manual value shall keep its last value forever or until it gets a new value.

If instead the signal would be a “Calculated” signal, there wouldn’t be an extra last value. The last value returned is the value from the ASignalDef table.

ChangeTime	ChangeValue	Quality	
2005-01-25 00:11:40	0,035675	192	This value is the last stored and comes from ASignalDef table.

In this example, the total time the cooler has been switched on is requested.

```
Sig_Select_Log 'COOLER', '2005-01-24 00:00:00', '2005-01-26 00:00:00', 1
```

The result will be according to table below. Active runtime is calculated when the signal is “1” dependent on configuration of ActiveRunTime in DSignalDef for the signal.

ChangeTime	ChangeValue (Active run time)	Quality	
2005-01-25 00:00:00	0	192	These values are compressed and retrieved from ASignalLog table.
2005-01-25 00:00:30	30	192	
2005-01-25 00:01:00	30	192	
2005-01-25 00:01:30	60	192	
2005-01-25 00:02:00	60	192	
2005-01-25 00:02:30	90	192	
2005-01-25 00:03:00	90	192	

2005-01-25 00:03:30	120	192	
2005-01-25 00:04:00	120	192	
2005-01-25 00:04:30	150	192	
2005-01-25 00:05:00	150	192	
2005-01-25 00:05:30	180	192	
2005-01-25 00:06:00	180	192	
2005-01-25 00:06:30	210	192	
2005-01-25 00:07:00	210	192	This value is the last stored and comes from DSignalDef table.
2005-01-25 00:12:20	530	192	This value is a copy of the DSignalDef value but timestamped with last updated watchdog time from OPCServer table. This is used as compensation when a value is filtered out by the dead band filter in OPCDA client or in the DCS system. The active runtime is recalculated based on the time difference between LastChangeTime in DSignalDef and LastForceSaveTime in OPCServer tables.

2.1.5.4.3 Sig_Select_Calc

This procedure returns a record set with calculated analog or digital signal values between start and end time. Each value is calculated based on previous archived value, the derivate of previous value and next archived value.

If an archived value included in the calculation has a quality that differs from “Good”, the calculated value will also get the same status e.g. “unknown”.

The number of records returned depends on the increment parameter. If increment input is a positive value, the increment parameter is in seconds between each record.
If increment input is a negative value, the increment parameter is in milliseconds.

If the values will be interpolated depends on the Interpol attribute for the signal.

The procedure can return the result in two modes the normal time mode and the duration mode. Duration mode is only valid for analog values and is activated by using parameters 9 – 11.

Parameters	Default value	Description
@@SignalID		Can be either SignalID as a smallint or SignalName as a varchar.
@@StartDate		Data type: datetime
@@EndDate	StartDate	Data type: datetime
@@Increment	1	Positive values = increment in seconds. Negative values = milliseconds. Data type: int
@@Property	0	For analog values: Time mode: 0 – Normal interpolated value for period 1 – Min value for period 2 – Max value for period 3 – Average value for period Duration mode:

		9 – Min value for period 10 – Max value for period 11 – Average value for period For digital values: 0 – Signal value 1 – Active TotalRunTime 2 – Inactive TotalRunTime 3 – TotalChanges 4 – Active ServiceRunTime 5 – Inactive ServiceRunTime 6 – ServiceChanges Data type: tinyint
@@GroupingPoint	0	GroupingPoint is used to decide where to put the timestamp.. In procedures for retrieving data the following codes are used: 0 – The calculated value and timestamp is placed in the beginning of the calculated period. 1 – The calculated value and timestamp is placed in the middle of the calculated period. 2 – The calculated value and timestamp is placed at the end of the calculated period. Valid for: Analog Data type: tinyint
@@Now	The SQL server's internal system clock	In a case where the signal is of type "ExtendLastValue", this is the time the value should be extended to. Data type: datetime

Return values	Data type	Description
TimeSerieDate	datetime or int	In Time mode a datetime is returned. The returned timestamp depends on GroupingPoint In Duration mode an int is retuned returning the duration in seconds.
CalcValue	bigint / float	The returned value depends on SignalType and Property.
Quality	smallint	

The following samples are based on same sample data as the samples for Sig_Select_Log.

2.1.5.4.3.1 Examples

The following samples are written in SQL code to legible the syntax in a “straight” way.

The procedure can be called with SignalName or SignalID eg.

```
Sig_Select_Calc 'INDOORTEMP', '2005-01-24 00:00:00', '2005-01-26 00:00:00', 30
```

Or

```
Sig_Select_Calc 4949, '2005-01-24 00:00:00', '2005-01-26 00:00:00', 30
```

In this example interpolated values are requested with an interval of 30s and a time range starting before first stored value and ends after last stored value. The result is according to table below.

TimeSerieDate	CalcValue	Quality	

2005-07-25 00:00:00	0	192	These interpolated values are based on compressed values retrieved from the ASignalLog table.
2005-07-25 00:00:30	0,27070679	192	
2005-07-25 00:01:00	0,52701685	192	
2005-07-25 00:01:30	0,78332691	192	
2005-07-25 00:02:00	0,894171858	192	
2005-07-25 00:02:30	0,973223167	192	
2005-07-25 00:03:00	0,950671544	192	
2005-07-25 00:03:30	0,827691745	192	
2005-07-25 00:04:00	0,675463181	192	
2005-07-25 00:04:30	0,406913367	192	
2005-07-25 00:05:00	0,138363553	192	
2005-07-25 00:05:30	-0,130186262	192	
2005-07-25 00:06:00	-0,398736076	192	
2005-07-25 00:06:30	-0,66728589	192	
2005-07-25 00:07:00	-0,848551469	192	
2005-07-25 00:07:30	-0,939374542	192	
2005-07-25 00:08:00	-0,996164609	192	
2005-07-25 00:08:30	-0,884464549	192	
2005-07-25 00:09:00	-0,772764488	192	
2005-07-25 00:09:30	-0,52495177	192	
2005-07-25 00:10:00	-0,254068935	192	
2005-07-25 00:10:30	0,0168139	192	
2005-07-25 00:11:00	0,0248972	192	
2005-07-25 00:11:30	0,0329805	192	This interpolated value is based on the value in the ASignalDeftable.
2005-07-25 00:12:00	0,035675	192	This interpolated value is based on a copy of the ASignalDef value but timestamped with last updated watchdog time from OPCServer table. This is used as compensation when a value is filtered out by the dead band filter in OPCDA client or in the DCS system.

If the signal had been a “Manual Editable” signal, the last values will be like following values in table below:

ChangeTime	ChangeValue	Quality	
2005-01-25 00:12:00	0,035675	192	This value is a copy of the ASignalDef value but time stamped with current time. This is because a manual value shall keep its last value forever or until it gets a new value. These values are based on a copy of the ASignalDef value but continued until current time. This is because a manual value shall keep its last value forever or until it gets a new value.
2005-01-25 00:12:30	0,035675	192	
2005-01-25 00:13:00	0,035675	192	
2005-01-25 00:13:30	0,035675	192	
…	0,035675	192	
2005-01-25 23:59:30	0,035675	192	

If instead the signal would be a “Calculated” signal, there wouldn’t be an extra last value. The last returned value is at 2005-07-25 00:11:00.

This example will return an interpolated value at a specific time.

```
Sig_Select_Calc 'INDOORTEMP', '2005-01-25 00:05:00'
```

The result is.

TimeSerieDate	CalcValue	Quality	
2005-07-25 00:05:00	0,138363553	192	This interpolated value is based on compressed values retrieved from ASignalLog table.

This example will generate average minute values for an analog signal and put time stamp on the values in middle of each period.

```
Sig_Select_Calc 'INDOORTEMP', '2005-01-25 00:00:00', '2005-01-25 00:05:00', 60, 3, 1
```

The result is.

ChangeTime	ChangeValue	Quality	
2005-01-25 00:00:30	0.269507062	192	These calculated values are based on compressed values retrieved from ASignalLog table.
2005-01-25 00:01:30	0.752259571	192	
2005-01-25 00:02:30	0.959046384	192	
2005-01-25 00:03:30	0.825254347	192	
2005-01-25 00:04:30	0.406913366	192	

In this example interpolated values are requested for total time the heater has been switched on.

```
Sig_Select_Calc ‘HEATER’, ‘2005-01-25 00:06:00’, ‘2005-01-25 00:08:00’, 15, 1
```

The result will be according to following table, the active runtime is interpolated when the signal was “1” dependent on configuration of ActiveRunTime in DSignalDef for the signal.

ChangeTime	ChangeValue (Active run time)	Quality	
2005-01-25 00:06:00	180	192	These interpolated values are based on compressed values retrieved from the DSignalLog table.
2005-01-25 00:06:15	195	192	
2005-01-25 00:06:30	210	192	
2005-01-25 00:06:45	210	192	
2005-01-25 00:07:00	210	192	This value is the latest stored value and it comes from the DSignalDef table.
2005-01-25 00:07:15	225	192	These values are based on a copy of the DSignalDef value but timestamped with the time from the OPCServer table. This is used as

			compensation when a value is filtered out by the dead band filter in the OPCDA client. The active runtime is recalculated based on the time difference between the LastChangeTime in DSignalDef and LastForceSaveTime in OPCServer tables.

The procedure can be called with SignalName or SignalID eg.

```
Sig_Select_Calc 'INDOORTEMP', '2005-01-24 00:00:00', '2005-01-26 00:00:00', 30, 11
```

Or

```
Sig_Select_Calc 4949, '2005-01-24 00:00:00', '2005-01-26 00:00:00', 30, 11
```

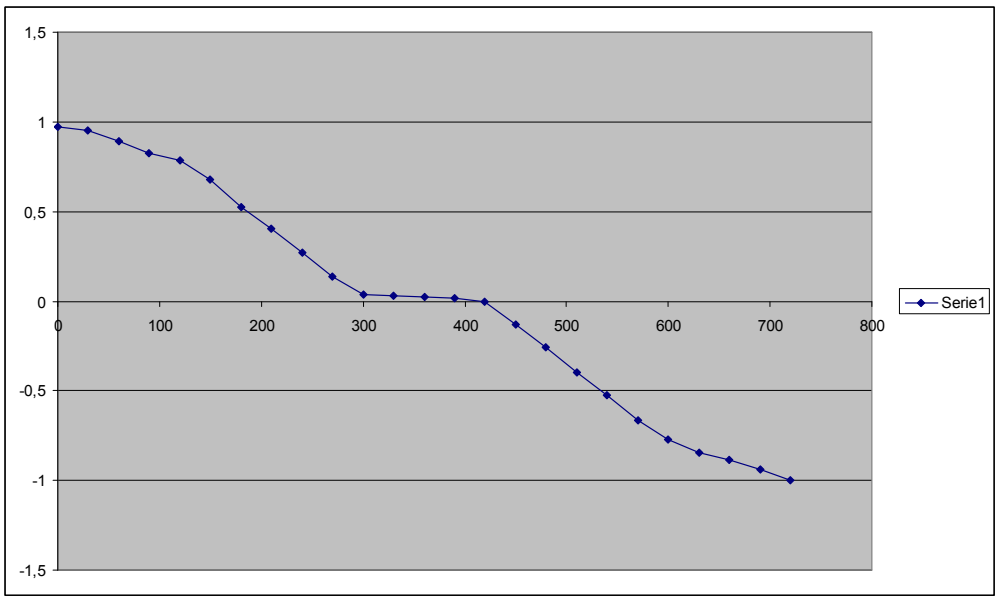
When the property parameter is 9, 10 or 11 duration mode is activated, in this case we use 11 e.g. duration based on average values for 30 seconds periods.

In this example average values are requested with an interval of 30s and a time range starting before first stored value and ends after last stored value.
We also want the result sorted as a duration result
The result is according to table below.

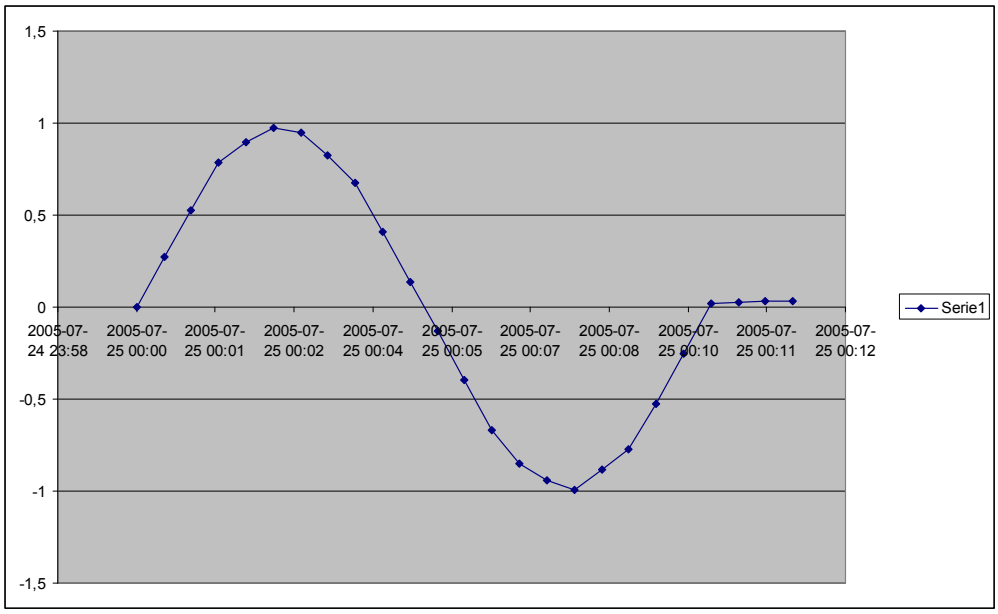
TimeSerieDate	CalcValue	Quality
0	0,973223	192
30	0,950672	192
60	0,894172	192
90	0,827692	192
120	0,783327	192
150	0,675463	192
180	0,527017	192
210	0,406913	192
240	0,270707	192
270	0,138364	192
300	0,035675	192
330	0,032981	192
360	0,024897	192
390	0,016814	192
420	0	192
450	-0,13019	192
480	-0,25407	192
510	-0,39874	192
540	-0,52495	192
570	-0,66729	192
600	-0,77276	192
630	-0,84855	192
660	-0,88446	192
690	-0,93937	192
720	-0,99616	192

The TimeSerieDate now returns the number of seconds as an integer and the values are sorted with the largest value first.

When we look at the result as a graph it looks like this:



As the opposite of the normal time data that looks like this:



2.1.5.4.4 Sig_Select_Serie

This function gets archived values with specified start time, end time and requested number of samples for the specified tag. When using the function, a start time, end time, numbers of samples or periods that will be returned are the inputs to the procedure.

The purpose of this function is getting data to a plot. To fill a plot on a screen, there is no need for more values than there are pixels on the screen.
But it is also very important not to miss a deviation or transient at a specific time. This procedure will always detect the max deviations and normal values. E.g. if values for a tag for a period of 6 months is requested and the signal has been stable all the time except for 1 minute during one day.
This minute may be just the minute you are interested in, but it would be lost in the resolution of that time range if no extra handling were implemented to take care of this.

The “sig_select_serie” will handle this in following way.
If values are requested for a long time range, the function must be provided with information about how many samples that is requested (the number of pixels available on the plot).
If 1000 pixels are available to use, 250 samples should be used as input for the number of samples requested.

The function will return between 0 and 4 values for each sample. The values returned are according to following principle.
For each sample period the first value, the highest value, the lowest value for the period will be presented and also the first value that represent a different status than previous value.
If the first value and the highest or/and the lowest value is the same, the function will only present one of them. This makes it possible to discover very fast transients over a long time range. When finding a transient, the time range can be changed to a shorter period to get more detailed data.

If the values will be interpolated depends on the Interpol attribute for the signal.

Parameters	Default value	Description
@@SignalID		Can be either SignalID as a smallint or SignalName as a varchar.
@@StartDate		Data type: datetime

@@EndDate		Data type: datetime
@@Samples		The number of samples to retrieve. Data type: int
@@Property	0	For analog values: 0 – Normal interpolated value for period 1 – Min value for period 2 – Max value for period 3 – Average value for period For digital values: 0 – Signal value 1 – Active TotalRunTime 2 – Inactive TotalRunTime 3 – TotalChanges 4 – Active ServiceRunTime 5 – Inactive ServiceRunTime 6 – ServiceChanges Data type: tinyint
@@Increment	3600	Only used for analog values when using min, max or average calculation, then you have to specify the length in seconds that the min, max or average function should use for its calculation. Data type: int
@@GroupingPoint	0	GroupingPoint is used to decide where to put the timestamp.. In procedures for retrieving data the following codes are used: 0 – The calculated value and timestamp is placed in the beginning of the calculated period. 1 – The calculated value and timestamp is placed in the middle of the calculated period. 2 – The calculated value and timestamp is placed at the end of the calculated period. Valid for: Analog Data type: tinyint
@@Debug		Not documented
@@Now	The SQL server's internal system clock	In a case where the signal is of type "ExtendLastValue", this is the time the value should be extended to. Data type: datetime

Return values	Data type	Description
TimeSerieDate	Datetime	
CalcValue	Bigint / float	Depending of signaltype and Properties
Quality	smallint	

2.1.5.4.4.1 Examples

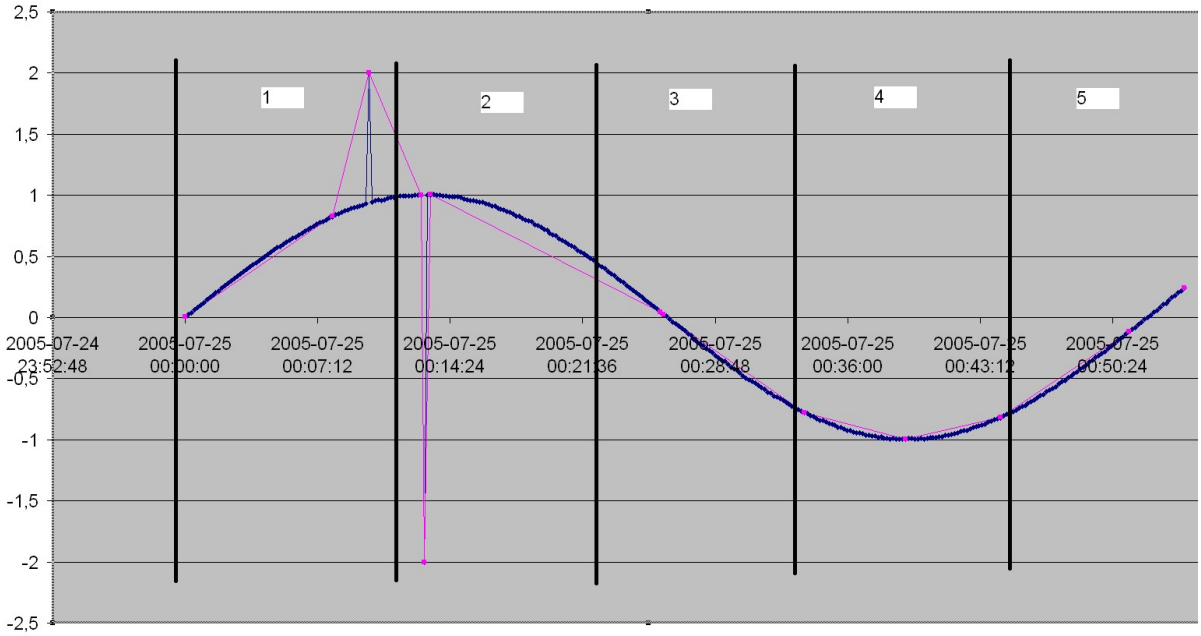
The following samples are written in SQL code to legible the syntax in a “straight” way.

```
Sig_Select_Serie 'OUTDOORTEMP', '2005-07-25 00:00:00', '2005-07-25 00:01:00', 5
```

In following example, some sample data with the sinus signal is used as when illustrating storage of analog data under section ASig_Store. For this example some transients has been added to the sample data.
Five periods are requested from this sample data. For these periods the following result will be returned.

- In period 1 there is a first value (which is also the same as the min value), a status change, and a max value.
- In period 2 there is a first value, a min value and a max value.
- In period 3 there is a first value (which is also the same as the max value), and a min value.
- In period 4 there is a first value, a min value and a max value.
- In period 5 there is a first value (which is the same as the min value) and a max value (which is the same as the last value)

This example do also visualize that when different type of values will coincide, the duplicate information will be filtered away.



TimeSerieDate	CalcValue	Quality	Period	Type
2005-07-25 00:00	0,007005753	192	1	First and Min
2005-07-25 00:08	0,825991262	8	1	Quality change
2005-07-25 00:10	2	192	1	Max
2005-07-25 00:12	1,000134347	192	2	First
2005-07-25 00:13	-2	192	2	Min
2005-07-25 00:13	1,008808418	192	2	Max
2005-07-25 00:25	0,045567239	192	3	First and Max
2005-07-25 00:25	0,022533524	192	3	Min
2005-07-25 00:33	-0,780278515	192	4	First
2005-07-25 00:39	-0,997128696	192	4	Min
2005-07-25 00:44	-0,817025727	192	4	Max

2005-07-25 00:51	-0,114535402	192	5	First
2005-07-25 00:54	0,243971978	192	5	Max and Last

The red curve (the base signal) is selected with:

```
Sig_Select_Serie 'OUTDOORTEMP', '2008-01-22 12:00:00', '2008-01-22 14:00:00', 200
```

The cyan curve (the 1 hour calculated average signal) is selected with:

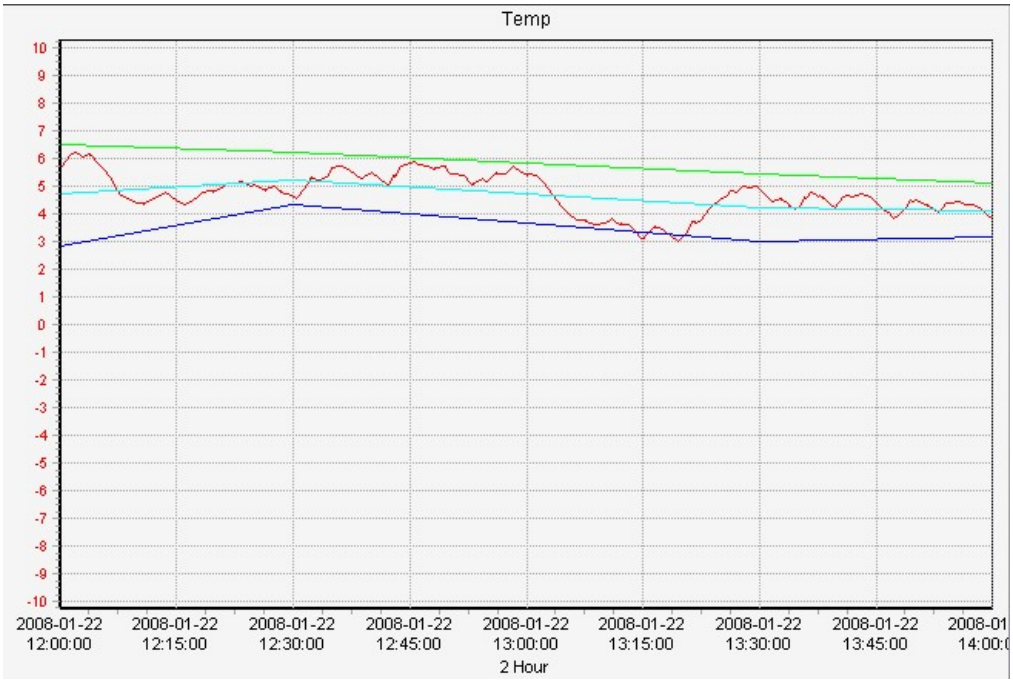
```
Sig_Select_Serie 'OUTDOORTEMP', '2008-01-22 12:00:00', '2008-01-22 14:00:00', 200, 3, 3600, 1
```

The green curve (the 1 hour calculated max signal) is selected with:

```
Sig_Select_Serie 'OUTDOORTEMP', '2008-01-22 12:00:00', '2008-01-22 14:00:00', 200, 2, 3600, 1
```

```
The blue curve (the 1 hour calculated min signal) is selected with:
Sig_Select_Serie 'OUTDOORTEMP', '2008-01-22 12:00:00', '2008-01-22 14:00:00', 200, 1, 3600, 1
```

The average, max and min values are returned with its timestamps in the middle of the calculated period.



2.1.6 Scheduled jobs

2.1.6.1 AutArch_DBPurge

This job is scheduled to run once per hour.
The job executes the procedure DBPurge which checks the percent of free space in the AutArch DB. If free space is les than 2.5% of the total DB size, the oldest data will be deleted until there is at least 2.5% free space.

This procedure will only run if the database file has a fixed size. The procedure takes its configuration from the DBPurgeStat table and also writes back some information to that same table.

2.1.7 Formulas used for TSQL calculated signals

NOTE! Since AutArch 2.0 TSQL calculations are not recommended. Instead it is recommended to use Python calculations.

The information in this appendix is based on the information in the SQL Server Books Online.
To learn more about python calculated signals, see the Python editor chapter.

2.1.7.1 Operators

2.1.7.1.1 Arithmetic Operators

2.1.7.1.1.1 + (Add)

Adds two numbers. This addition arithmetic operator can also add a number, in days, to a date.

Syntax
expression + expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression of any of the data types in the numeric category except the bit data type.

Result Types
Returns the data type of the argument with the higher precedence. For more information, see Data Type Precedence.

2.1.7.1.1.2 - (Subtract)

Subtracts two numbers. This subtraction arithmetic operator can also subtract a number, in days, from a date.

Syntax
expression - expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression of any of the data types of the numeric data type category except the bit data type.

Result Types
Returns the data type of the argument with the higher precedence. For more information, see Data Type Precedence.

2.1.7.1.1.3 * (Multiply)

Multiplies two expressions (an arithmetic multiplication operator).

Syntax
expression * expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression of any of the data types of the numeric data type category except the datetime or smalldatetime data types.

Result Types
Returns the data type of the argument with the higher precedence. For more information, see Data Type Precedence.

2.1.7.1.1.4 / (Divide)

Divides one number by another (an arithmetic division operator).

Syntax
dividend / divisor

Arguments
dividend

Is the numeric expression to divide. dividend can be any valid Microsoft® SQL Server™ expression of any of the data types of the numeric data type category except the datetime and smalldatetime data types.

divisor

Is the numeric expression to divide the dividend by. Divisor can be any valid SQL Server expression of any of the data types of the numeric data type category except the datetime and smalldatetime data types.

Result Types
Returns the data type of the argument with the higher precedence. For more information about data type precedence, see Data Type Precedence.

If an integer dividend is divided by an integer divisor, the result is an integer that has any fractional part of the result truncated.

Remarks
The actual value returned by the / operator is the quotient of the first expression divided by the second expression.

2.1.7.1.1.5 % (Modulo)

Provides the remainder of one number divided by another.

Syntax
dividend % divisor

Arguments
dividend

Is the numeric expression to divide. dividend must be any valid Microsoft® SQL Server™ expression of the integer data type category. (A modulo is the integer that remains after two integers are divided.)

divisor

Is the numeric expression to divide the dividend by. divisor must be any valid SQL Server expression of any of the data types of the integer data type category.

Result Types
int

Remarks
The modulo arithmetic operator can be used in the select list of the SELECT statement with any combination of column names, numeric constants, or any valid expression of the integer data type category.

2.1.7.1.2 Bitwise Operators

2.1.7.1.2.1 & (Bitwise AND)

Performs a bitwise logical AND operation between two integer values.

Syntax
expression & expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression of any of the data types of the integer data type category. expression is an integer parameter that is treated and transformed into a binary number for the bitwise operation.

Result Types
Returns an int if the input values are int, a smallint if the input values are smallint, or a tinyint if the input values are tinyint.

Remarks
The bitwise & operator performs a bitwise logical AND between the two expressions, taking each corresponding bit for both expressions. The bits in the result are set to 1 if and only if both bits (for the current bit being resolved) in the input expressions have a value of 1; otherwise, the bit in the result is set to 0.

The & bitwise operator can be used only on expressions of the integer data type category.

If the left and right expressions have different integer data types (for example, the left expression is smallint and the right expression is int), the argument of the smaller data type is converted to the larger data type. In this example, the smallint expression is converted to an int.

2.1.7.1.2.2 | (Bitwise OR)

Performs a bitwise logical OR operation between two given integer values as translated to binary expressions within Transact-SQL statements.

Syntax
expression | expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression of any of the data types of the integer data type category. expression is an integer that is treated and transformed into a binary number for the bitwise operation.

Result Types
Returns an int if the input values are int, a smallint if the input values are smallint, or a tinyint if the input values are tinyint.

Remarks
The bitwise | operator performs a bitwise logical OR between the two expressions, taking each corresponding bit for both expressions. The bits in the result are set to 1 if either or both bits (for the current bit being resolved) in the input expressions have a value of 1; if neither bit in the input expressions is 1, the bit in the result is set to 0.

The | bitwise operator requires two expressions, and it can be used on expressions of only the integer data type category.

If the left and right expressions have different integer data types (for example, the left expression is smallint and the right expression is int), the argument of the smaller data type is converted to the larger data type. In this example, the smallint expression is converted to an int.

2.1.7.1.2.3 ^ (Bitwise Exclusive OR)

Performs a bitwise exclusive OR operation between two given integer values as translated to binary expressions within Transact-SQL statements.

Syntax
expression ^ expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression of any of the data types of the integer data type category, or of the binary or varbinary data type. expression is an integer that is treated and transformed into a binary number for the bitwise operation.

Note Only one expression can be of either binary or varbinary data type in a bitwise operation.

Result Types
Returns an int if the input values are int, a smallint if the input values are smallint, or a tinyint if the input values are tinyint.

Remarks
The bitwise ^ operator performs a bitwise logical ^ between the two expressions, taking each corresponding bit for both expressions. The bits in the result are set to 1 if either (but not both) bits (for the current bit being resolved) in the input expressions have a value of 1; if both bits are either a value of 0 or 1, the bit in the result is cleared to a value of 0.

The ^ bitwise operator can be used only on columns of the integer data type category.

If the left and right expressions have different integer data types (for example, the left expression is smallint and the right expression is int), then the argument of the smaller data type is converted to the larger data type. In this example, the smallint expression is converted to an int.

2.1.7.1.3 Comparison Operators

2.1.7.1.3.1 = (Equals)

Compares two expressions (a comparison operator). When you compare nonnull expressions, the result is TRUE if both operands are equal; otherwise, the result is FALSE. If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

Syntax
expression = expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. Both expressions must have implicitly convertible data types. The conversion depends on the rules of data type precedence. For more information, see Data Type Precedence.

Result Types
Boolean

2.1.7.1.3.2 > (Greater Than)

Compares two expressions (a comparison operator). When you compare nonnull expressions, the result is TRUE if the left operand has a higher value than the right operand; otherwise, the result is FALSE. If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

Syntax
expression > expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. Both expressions must have implicitly convertible data types. The conversion depends on the rules of data type precedence. For more information, see Data Type Precedence.

Result Types
Boolean

2.1.7.1.3.3 < (Less Than)

Compares two expressions (a comparison operator). When you compare nonnull expressions, the result is TRUE if the left operand has a lower value than the right operand; otherwise, the result is FALSE. If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

Syntax
expression < expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. Both expressions must have implicitly convertible data types. The conversion depends on the rules of data type precedence. For more information, see Data Type Precedence.

Result Types
Boolean

2.1.7.1.3.4 >= (Greater Than or Equal To)

Compares two expressions (a comparison operator). When you compare nonnull expressions, the result is TRUE if the left operand has a higher or equal value than the right operand; otherwise, the result is FALSE. If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

Syntax
expression > = expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. Both expressions must have implicitly convertible data types. The conversion depends on the rules of data type precedence. For more information, see Data Type Precedence.

Result Types
Boolean

2.1.7.1.3.5 <= (Less Than or Equal To)

Compares two expressions (a comparison operator). When you compare nonnull expressions, the result is TRUE if the left operand has a lower or equal value than the right operand; otherwise, the result is FALSE. If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

Syntax
expression = < expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. Both expressions must have implicitly convertible data types. The conversion depends on the rules of data type precedence. For more information, see Data Type Precedence.

Result Types
Boolean

2.1.7.1.3.6 <> (Not Equal To)

Compares two expressions (a comparison operator). When you compare nonnull expressions, the result is TRUE if the left operand is not equal to the right operand; otherwise, the result is FALSE. If either or both operands are NULL and SET ANSI_NULLS is set to ON, the result is NULL. If SET ANSI_NULLS is set to OFF, the result is FALSE if one of the operands is NULL, and TRUE if both operands are NULL.

Syntax
expression < > expression

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. Both expressions must have implicitly convertible data types. The conversion depends on the rules of data type precedence. For more information, see Data Type Precedence.

Result Types
Boolean

2.1.7.2 Functions

2.1.7.2.1 Mathematic functions

2.1.7.2.1.1 ABS

Returns the absolute, positive value of the given numeric expression.

Syntax
ABS (numeric_expression)

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Types
Returns the same type as numeric_expression.

2.1.7.2.1.2 ACOS

Returns the angle, in radians, whose cosine is the given float expression; also called arccosine.

Syntax
ACOS (float_expression)

Arguments
float_expression

Is an expression of the type float or real, with a value from -1 through 1. Values outside this range return NULL and report a domain error.

Return Types
Float

2.1.7.2.1.3 ASIN

Returns the angle, in radians, whose sine is the given float expression (also called arcsine).

Syntax
ASIN (float_expression)

Arguments
float_expression

Is an expression of the type float, with a value from -1 through 1. Values outside this range return NULL and report a domain error.

Return Types
Float

2.1.7.2.1.4 ATAN

Returns the angle in radians whose tangent is the given float expression (also called arctangent).

Syntax
ATAN (float_expression)

Arguments
float_expression

Is an expression of the type float.

Return Types
Float

2.1.7.2.1.5 ATN2

Returns the angle, in radians, whose tangent is between the two given float expressions (also called arctangent).

Syntax
ATN2 (float_expression , float_expression)

Arguments
float_expression

Is an expression of the float data type.

Return Types
Float

2.1.7.2.1.6 CEILING

Returns the smallest integer greater than, or equal to, the given numeric expression.

Syntax
CEILING (numeric_expression)

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Types
Returns the same type as numeric_expression.

2.1.7.2.1.7 COS

A mathematic function that returns the trigonometric cosine of the given angle (in radians) in the given expression.

Syntax
COS (float_expression)

Arguments
float_expression

Is an expression of type float.

Return Types
Float

2.1.7.2.1.8 COT

A mathematic function that returns the trigonometric cotangent of the specified angle (in radians) in the given float expression.

Syntax
COT (float_expression)

Arguments
float_expression

Is an expression of type float.

Return Types
Float

2.1.7.2.1.9 DEGREES

Given an angle in radians, returns the corresponding angle in degrees.

Syntax
DEGREES (numeric_expression)

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Code Values
Returns the same type as numeric_expression.

2.1.7.2.1.10 EXP

Returns the exponential value of the given float expression.

Syntax
EXP (float_expression)

Arguments
float_expression

Is an expression of type float.

Return Types
Float

2.1.7.2.1.11 FLOOR

Returns the largest integer less than or equal to the given numeric expression.

Syntax
FLOOR (numeric_expression)

Arguments

numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Types
Returns the same type as numeric_expression.

2.1.7.2.1.12 LOG

Returns the natural logarithm of the given float expression.

Syntax
LOG (float_expression)

Arguments
float_expression

Is an expression of the float data type.

Return Types
Float

2.1.7.2.1.13 LOG10

Returns the base-10 logarithm of the given float expression.

Syntax
LOG10 (float_expression)

Arguments
float_expression

Is an expression of the float data type.

Return Types
Float

2.1.7.2.1.14 PI

Returns the constant value of PI.

Syntax
PI ( )

Return Types
Float

2.1.7.2.1.15 POWER

Returns the value of the given expression to the specified power.

Syntax
POWER (numeric_expression , y)

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

y

Is the power to which to raise numeric_expression. y can be an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Types
Same as numeric_expression.

2.1.7.2.1.16 RADIANS

Returns radians when a numeric expression, in degrees, is entered.

Syntax
RADIANS (numeric_expression)

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Types
Returns the same type as numeric_expression

2.1.7.2.1.17 RAND

Returns a random float value from 0 through 1.

Syntax
RAND ([seed])

Arguments
seed

Is an integer expression (tinyint, smallint, or int) that specifies the seed value. If seed is not specified, Microsoft® SQL Server™ 2000 assigns a seed value at random. For a given seed value, the result returned is always the same.

Return Types
float

2.1.7.2.1.18 ROUND

Returns a numeric expression, rounded to the specified length or precision.

Syntax
ROUND (numeric_expression , length [, function])

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

length

Is the precision to which numeric_expression is to be rounded. length must be tinyint, smallint, or

int. When length is a positive number, numeric_expression is rounded to the number of decimal places specified by length. When length is a negative number, numeric_expression is rounded on the left side of the decimal point, as specified by length.

function

Is the type of operation to perform. function must be tinyint, smallint, or int. When function is omitted or has a value of 0 (default), numeric_expression is rounded. When a value other than 0 is specified, numeric_expression is truncated.

Return Types
Returns the same type as numeric_expression.

Remarks
ROUND always returns a value. If length is negative and larger than the number of digits before the decimal point, ROUND returns 0.

2.1.7.2.1.19 SIGN

Returns the positive (+1), zero (0), or negative (-1) sign of the given expression.

Syntax
SIGN (numeric_expression)

Arguments
numeric_expression

Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type.

Return Types
float

2.1.7.2.1.20 SIN

Returns the trigonometric sine of the given angle (in radians) in an approximate numeric (float) expression.

Syntax
SIN (float_expression)

Arguments
float_expression

Is an expression of type float.

Return Types
Float

2.1.7.2.1.21 SQUARE

Returns the square of the given expression.

Syntax
SQUARE (float_expression)

Arguments
float_expression

Is an expression of type float.

Return Types
float

2.1.7.2.1.22 SQRT

Returns the square root of the given expression.

Syntax
SQRT (float_expression)

Arguments
float_expression

Is an expression of type float.

Return Types
float

2.1.7.2.1.23 TAN

Returns the tangent of the input expression.

Syntax
TAN (float_expression)

Arguments
float_expression

Is an expression of type float or real, interpreted as number of radians.

Return Types
float

2.1.7.2.2 Conversion functions

2.1.7.2.2.1 CONVERT

Explicitly converts an expression of one data type to another.

Syntax
CONVERT (data_type [(length)] , expression [, style])

Arguments
expression

Is any valid Microsoft® SQL Server™ expression. For more information, see Expressions.

data_type

Is the target system-supplied data type, including bigint and sql_variant. User-defined data types cannot be used. For more information about available data types, see Data Types.

length

Is an optional parameter of nchar, nvarchar, char, varchar, binary, or varbinary data types.

Example:
Use this function to convert an integer to a float
CONVERT(float, 123)

2.1.7.3 Conditional Data Processing

2.1.7.3.1 CASE

Evaluates a list of conditions and returns one of multiple possible result expressions.

CASE has two formats:

The simple CASE function compares an expression to a set of simple expressions to determine the result.

The searched CASE function evaluates a set of Boolean expressions to determine the result.
Both formats support an optional ELSE argument.

Syntax
Simple CASE function:

```
CASE input_expression
    WHEN when_expression THEN result_expression
        [ ...n ]
    [
        ELSE else_result_expression
    ]
END
```

Searched CASE function:

```
CASE
    WHEN Boolean_expression THEN result_expression
        [ ...n ]
    [
        ELSE else_result_expression
    ]
END
```

Arguments
input_expression

Is the expression evaluated when using the simple CASE format. input_expression is any valid Microsoft® SQL Server™ expression.

WHEN when_expression

Is a simple expression to which input_expression is compared when using the simple CASE format. when_expression is any valid SQL Server expression. The data types of input_expression and each when_expression must be the same or must be an implicit conversion.

n

Is a placeholder indicating that multiple WHEN when_expression THEN result_expression clauses, or multiple WHEN Boolean_expression THEN result_expression clauses can be used.

THEN result_expression

Is the expression returned when input_expression equals when_expression evaluates to TRUE, or Boolean_expression evaluates to TRUE. result expression is any valid SQL Server expression.

ELSE else_result_expression

Is the expression returned if no comparison operation evaluates to TRUE. If this argument is omitted and no comparison operation evaluates to TRUE, CASE returns NULL. else_result_expression is any valid SQL Server expression. The data types of else_result_expression and any result_expression must be the same or must be an implicit conversion.

WHEN Boolean_expression

Is the Boolean expression evaluated when using the searched CASE format. Boolean_expression is any valid Boolean expression.

Result Types
Returns the highest precedence type from the set of types in result_expressions and the optional else_result_expression. For more information, see Data Type Precedence.

Result Values
Simple CASE function:
Evaluates input_expression, and then, in the order specified, evaluates input_expression = when_expression for each WHEN clause.

Returns the result_expression of the first (input_expression = when_expression) that evaluates to TRUE.

If no input_expression = when_expression evaluates to TRUE, SQL Server returns the else_result_expression if an ELSE clause is specified, or a NULL value if no ELSE clause is specified.
Searched CASE function:
Evaluates, in the order specified, Boolean_expression for each WHEN clause.

Returns result_expression of the first Boolean_expression that evaluates to TRUE.

If no Boolean_expression evaluates to TRUE, SQL Server returns the else_result_expression if an ELSE clause is specified, or a NULL value if no ELSE clause is specified.

2.1.7.4 Datatypes

Data types used in the AutArch database.

2.1.7.4.1 BigInt

64-bit signed integer.
Integer (whole number) data from -2^63 (-9223372036854775808) through 2^63-1 (9223372036854775807).

2.1.7.4.2 Bit

Integer data with either a 1 or 0 value.

2.1.7.4.3 DateTime

Date and time data from January 1, 1753 through December 31, 9999, to an accuracy of one three-hundredth of a second (equivalent to 3.33 milliseconds or 0.00333 seconds).
Values are rounded to increments of .000, .003, or .007 seconds.

2.1.7.4.4 Float

64-bit floating point precision.
Floating precision number data from -1.79E + 308 through 1.79E + 308.

2.1.7.4.5 Int

32-bit signed integer.
Integer (whole number) data from -2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647).

2.1.7.4.6 NVarChar

Variable-length Unicode data with a maximum length of 4,000 characters.

2.1.7.4.7 Real

32-bit floating point precision.
Floating precision number data from -3.40E + 38 through 3.40E + 38.

2.1.7.4.8 SmallInt

16-bit signed integer.
Integer data from 2^15 (-32,768) through 2^15 - 1 (32,767).

2.1.7.4.9 TinyInt

8-bit unsigned integer.
Integer data from 0 through 255.

2.1.7.4.10 VarChar

Variable-length non-Unicode data with a maximum of 8,000 characters.

2.1.8 Common property descriptions

2.1.8.1 A

2.1.8.1.1 ACKED

This field is defined in the OPCAE standard as “Acknowledged” indication. If “Active”, the condition has been acknowledged
The ACKED condition is a state of the “NewState” values. 4 = OPC_CONDITION_ACKED.

Valid for: Events
Data type: bit

2.1.8.1.2 AckReqd

Abbreviation for OPCAE standard field, “AckRequired”. See separate description for this field.

2.1.8.1.3 AckRequired

OPCAE Standard field.
An indicator as to whether or not an acknowledgement is required. Many event notifications related to conditions do not normally require an acknowledgment, e.g. the receipt of an acknowledgment or

the transition to the inactive state. Furthermore, some conditions may be configured (using facilities outside the scope of this specification) to not require acknowledgment even for transitions into the condition, or for transitions among sub-conditions (e.g. transition into LevelAlarm or transition from HighAlarm to HighHighAlarm). In this case, it is the responsibility of the server to automatically place the condition into the Acknowledged state, since an acknowledgment will never be received.

Valid for: Events
Data type: bit

2.1.8.1.4 ACTIVE

OPCAE Standard field.
The ACTIVE condition is a state of the “NewState” values. 1 = OPC_CONDITION_ACTIVE.
The state indicates that the associated object is in ACTIVE condition.

Valid for: Events
Data type: bit

2.1.8.1.5 ActiveRunTime

This parameter is to display for the user which state that is the active and that is interesting as “operating state”.
It determines the active runtime for the digital signal, 0 (zero) means that the signal is active if the value is 0 (zero) and vice versa.
0 – If 0 is the active runtime.
1 – If 1 is the active runtime.

Valid for: Digital
Data type: tinyint

2.1.8.1.6 ActiveTime

Used in Ev_Store procedure as ActiveTime. This is the same field as EventActiveTime. See description for this attribute in this appendix.

Valid for: Events
Data type: DateTime

2.1.8.1.7 ActorID

OPCAE Standard field.
ActorID is the identifier of the OPC Client which acknowledged the condition, which is maintained as the AcknowledgerID property of the condition. This is included in event notifications generated by condition acknowledgments.

Valid for: Events
Data type: nvarchar(50)

2.1.8.1.8 AsyncRead

This parameter defines that reading from OPCServer as Asynchronous, which means that only changes are sent, and not cyclical updates. Async read is the only supported method.

Valid for: Analog, Digital
Data type: bit

2.1.8.2 B

2.1.8.2.1 BackColor

The BackColor is an Event log property that can be used to store the events background color. The color should be represented as a HTML color specification.

Valid for: Event
Data type: varchar(10)

2.1.8.2.2 BackLog

BackLog is the time in milliseconds, used to retrieve a previous value when the signal has an updated value that the compression will allow to be stored (outside MaxDivergence). The previous value is then stored at “number of BackLog” milliseconds before the new value, if the BackLog time is less than the last logged value.

Without BackLog the trend for the signal might be unrepresented.

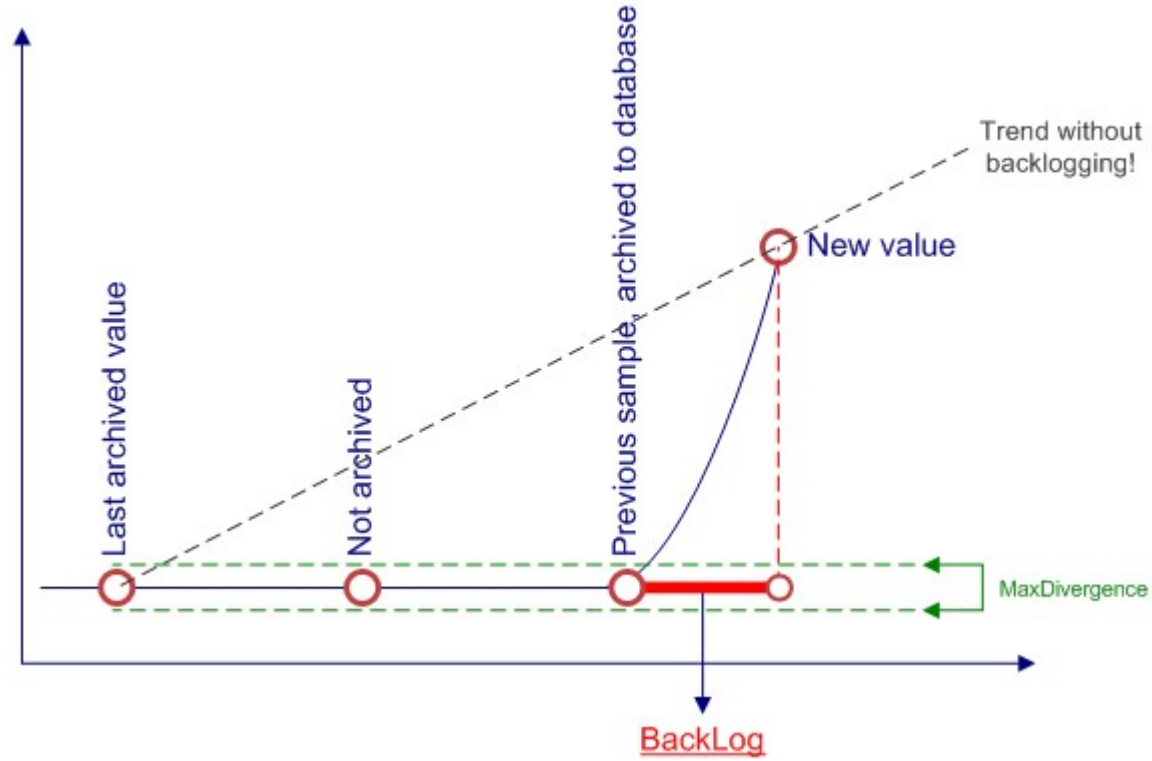
If BackLog is set to 0 (zero) no BackLogging will occur.

The BackLog value point is flagged with quality bit 512.

Valid for: Analog
Data type: bigint

More...

2.1.8.2.2.1 BackLog sample



2.1.8.3 C

2.1.8.3.1 CalcEnabled

CalcEnabled property determines if calculation is enabled or not.
0 – Calculation not enabled.
1 – Calculation enabled.

Valid for: Analog, Digital, MultiValue
Data type: bit

2.1.8.3.2 CalcFormula

The user written formula used for calculations. The formula calculation needs one or several input signals in the formula of which at least one input signal need to have the CalcTrigger set. The CalcTrigger triggers the calculation when the value of the signal changes.

The signals that are used in the formula are stored in an relational table but they are edited by writing the signal names surrounded by "{" and "}" in the following way:

```
{SignalName1} + {SignalName2}
```

Valid for: Analog, Digital
Data type: varchar(2000)

2.1.8.3.3 CalcReferenceTime

A reference time for when the calculation will be triggered. Used as reference time for time calculations.
When using many average calculation tags, the calculations can be spread in time. This is to avoid accumulation of concurrent calculations. To spread the calculations, the CalcReferenceTime property can be used. The default is that calculation occurs at the beginning of next period using `@@CalcReferenceTime = '2001-01-01 00:00:00'` but if following setting is used instead `@@CalcReferenceTime = '2001-01-01 00:00:10'` the calculation is moved to take place 10 seconds later.

Valid for: Analog
Data type: datetime

2.1.8.3.4 CalcType

Determines the type of calculation to execute for calculation signals.

0 – a user written formula (CalcFormula property)
2 – time calculated by average value
4 – time calculated by maximum value
5 – time calculated by minimum value
10 – Python calculated
11 – Average periodic
12 – Sum periodic

Valid for: Analog, Digital, MultiValue
Data type: tinyint

2.1.8.3.5 CalculationInterval

The interval for a time calculated TSQL-formula to be executed, in seconds. The CalculationInterval property is ignored if the calculation is a formula calculated signal.

Valid for: Analog
Data type: bigint

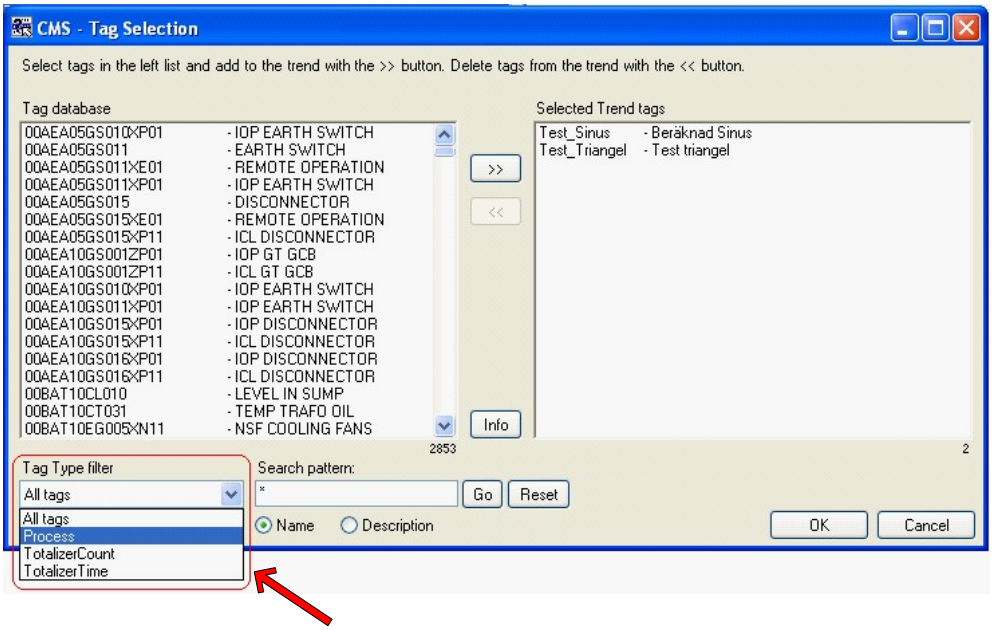
2.1.8.3.6 Category1

The category1 property is used for categorizing tags and filtering functions in viewing applications.

Valid for: Analog, Digital
Data type: nvarchar(20)

More...

2.1.8.3.6.1 Category1 sample



2.1.8.3.7 ChangedBy

The user name that made the last configuration change to this signal.
All changes made to the signal are persisted in the SignalRev table.

Data type: nvarchar(200)

2.1.8.3.8 ChangeDiff

The ChangeDiff property is the difference between ServieLimitChanges and ServiceChanges. Which means it is the number of times the signal can change before it reaches service limit (ServiceLimitChanges).

Valid for: Digital
Data type: bigint

2.1.8.3.9 ChangeMask

OPCAE Standard field.
ChangeMask indicates to the client which properties of the condition have changed, to cause the server to send the event notification.
Indicates to the client which properties of the condition have changed, to have caused the server to send the event notification. It may have one or more of the following values:

OPC_CHANGE_ACTIVE_STATE = 1 (The condition's active state has changed.)
OPC_CHANGE_ACK_STATE = 2 (The condition's acknowledgment state has changed.)
OPC_CHANGE_ENABLE_STATE = 4 (The condition's enabled state has changed.)
OPC_CHANGE_QUALITY = 8 (The ConditionQuality has changed.)

OPC_CHANGE_SEVERITY = 16 (The severity level has changed.)
OPC_CHANGE_SUBCONDITION = 32 (The condition has transitioned into a new sub-condition.)
OPC_CHANGE_MESSAGE = 64 (The event message has changed (compared to prior event notifications related to this condition).)
OPC_CHANGE_ATTRIBUTE = 128 (One or more event attributes have changed (compared to prior event notifications related to this condition).)

If the event notification is the result of a Refresh, these bits are to be ignored.
For a “new event”,OPC_CHANGE_ACTIVE_STATE is the only bit which will always be set. Other values are server specific. (A “new event” is any event resulting from the related condition leaving the Inactive and Acknowledged state.)

Valid for: Events
Data type: tinyint

2.1.8.3.10 ChangeTime

The timestamp for when the value of the signal changed. This applies to both digital and analog signals.

Valid for: Analog, Digital
Data type. Datetime

2.1.8.3.11 ChangeValue

Value of the archived signal.

Valid for: Analog, Digital
Data type: real or float (for Analog, depending on database installation setting), tinyint (for Digital)

2.1.8.3.12 ConditionName

OPCAE Standard field.
The name of the condition related to this event notification.

(The QueryConditionNames method gives clients a means of finding out the specific condition names which the event server supports for the specified event category. This method would typically be invoked prior to specifying an event filter. Condition names are server specific.
The number of condition names returned will vary depending on the sophistication of the server, but is expected to be less than 30 for most servers, making this interface more appropriate than a custom enumerator.
It is expected that the results of the Query will be fairly 'stable' in most situations. However, the Server is in fact allowed to change the available selection at any time. Therefore, a Client should do (or at least allow as an option) a fresh Query every time a selection is to be presented to the end user.)

Valid for: Events
Data type: nVarChar(200)

2.1.8.3.13 ConditionQuality

OPCAE Standard field.
ConditionQuality Indicates the quality of the underlying data items upon which this condition is based.
Condition Quality is stored as “Quality” in AA Eventlog table.

Valid for: Events
Data type: smallint

2.1.8.3.14 Cookie

OPCAE Standard field.
Server defined cookie associated with the event notification. This value can be used by a client when acknowledging the condition. This value is opaque to the client.

Valid for: Events
Data type: int

2.1.8.3.15 CreationTime

The CreationTime property is a timestamp when the signal was first created.
The timestamp is generated by system and cannot be changed.

Valid for: Analog, Digital, Events
Data type: datetime

2.1.8.3.16 Custom01 - 05

The Custom01 - 05 are free signal properties to be used by customer.

Valid for: Analog, Digital
Data type: nvarchar(100)

2.1.8.3.17 CustomEvDef01-03

The CustomEvDef01 - 03 are free Event definition properties to be used by customer.

Valid for: Event
Data type: nvarchar(50)

2.1.8.3.18 CustomEvLog01-03

The CustomEvLog01 - 03 are free Event log properties to be used by customer.

Valid for: Event
Data type: nvarchar(50)

2.1.8.4 D

2.1.8.4.1 DeadBand

Deadband (in % of signals measure range) that signals in this group will have as exception deadband parameter. Changes within this deadband will not be sent as updated values from the OPCServer to the OPCClient.

Valid for: Events
Data type: smallint

2.1.8.4.2 DisplayFormat

The DisplayFormat property sets the number of decimals for viewing applications, such as Trendview application.
The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted.
For example, when the value 14059,42450 is used in different display formats, see following table.

Valid for: Analog, Digital, Events

Data type: nvarchar(20)

More...

2.1.8.4.2.1 DisplayFormat sample

DisplayFormat	Display result in TrendView
0	14059
1	14059,4
2	14059,42
3	14059,425
4	1,4059E+004
5	1,40594E+004
6	1,405942E+004
7	1,4059425E+004
8	1,40594245E+004
9	1,405942450E+004
10	1,4059424500E+004
-1	1E+004
-2	1,4E+004
-3	1,41E+004
-4	1,406E+004
-5	14059
-6	14059,4
-7	14059,42
-8	14059,425
-9	14059,4245
-10	14059,42450

2.1.8.4.3 DisplayFormat2

The DisplayFormat2 property sets the number of decimals for viewing applications, such as Trendview application when using an alternative engineering unit.
The usage in TrendView application is for setting number of decimals (positive number) or number of digits (negative number) in total before displaying in power of ten. Default value for TrendViewer application if not configured at each tag is -5. This default parameter can be changed in configuration file if wanted.
For example, when the value 14059,42450 is used in different display formats, see following table.

Valid for: Analog, Digital, Events
Data type: nvarchar(20)

More...

2.1.8.5 E

2.1.8.5.1 ENABLED

OPCAE Standard field.
This field is defined in the OPCAE standard as “enabled” state. If “Active”, then the condition or area

has been enabled. The condition is currently being checked by the OPC Event Server.
The ENABLED condition is a state of the “NewState” values. 2 = OPC_CONDITION_ENABLED.

Valid for: Events
Data type: bit

2.1.8.5.2 EndDate

End date of time range used as input parameter in procedures

Valid for: Analog, Digital, Events
Data type: datetime

2.1.8.5.3 EndTime

End time of time range used as input parameter in procedures

Valid for: Analog, Digital, Events
Data type: datetime

2.1.8.5.4 EventActiveTime

OPCAE Standard field.
ActiveTime is the time of the transition into the condition or sub-condition which is associated with this event notification. This time corresponds to SubCondLastActive property of the associated OPCCondition object and is used to correlate condition acknowledgements with a particular transition into the condition/sub-condition.

Valid for: Events
Data type: DateTime

2.1.8.5.5 EventBlocking

The EventBlocking property activates/inactivates events from being logged to the database.
0 = Event is not blocked
1 = Event is blocked and no data is archived.
This attribute can be set anytime and changes take effect directly on archiving.

Valid for: Events
Data type: bit

2.1.8.5.6 EventCategory

OPCAE Standard field.
EventCategories define groupings of events supported by an OPC Event server. Examples of event categories might include “Process Events”, “System Events”, or “Batch Events”. Event categories may be defined for all event types, i.e. Simple, Tracking, and Condition-Related. However, a particular event category can include events of only one type. A given Source (e.g. “System” or “FIC101”) may generate events for multiple event categories. Names of event categories must be unique within the event server. The definition of event categories is server specific and is outside the scope of this specification. The name of the event category is included in every event notification. Event subscriptions may be filtered based on event category.

Valid for: Events
Data type: nvarchar(200)

2.1.8.5.7 EventCategoryID

EventCategoryID as a numeric value describing the Category of an Event.

Valid for: Events
Data type: int

2.1.8.5.8 EventDesc

Additional field not specified in OPCAE standard. This field is used by Siemens PCS7 system that uses this field as own defined customer attributes.

Valid for: Events
Data type: nvarchar(1000)

2.1.8.5.9 EventID

The EventID is a unique ID that identifies the event and is generated by the system.

Valid for: Events
Data type: int

2.1.8.5.10 EventServerNode

Additional field not specified in the OPCAE standard. This field can be used to specify the collecting eventservernode when there are several eventservers connected to one AA system. The name of the eventserver is specified in the OPCAEClient applications config file. The parametername in the config file is “CollEventServerNode”.

Valid for: Events
Data type: nvarchar(200)

2.1.8.5.11 EventStatus

Additional field not specified in OPCAE standard. Used for Siemens PCS7 system that uses this as own defined customer attributes.

Valid for: Events
Data type: varchar(10)

2.1.8.5.12 EventTime

The timestamp for when the event was logged.

Valid for: Events
Data type: datetime

2.1.8.5.13 EventType

OPCAE Standard field.
EventType specifies what type of event it is.
Predefined according to the standard is:
OPC_SIMPLE_EVENT
OPC_CONDITION_EVENT
OPC_TRACKING_EVENT
OPC_ALL_EVENTS

Valid for: Events
Data type: tinyint

2.1.8.5.14 ExportFlag

The ExportFlag property is set to mark the signal for export. It is used by the AAExport application for exporting data to encrypted exportfiles. By setting combinations for this value, different export scope can be performed.
0 – no export

1 – signal export
The flag is used in a bit coded way that makes it possible to flag a signal to be exported by different export jobs, se the AAExport documentation for examples.

Valid for: Analog, Digital
Data type: smallint

2.1.8.6 F

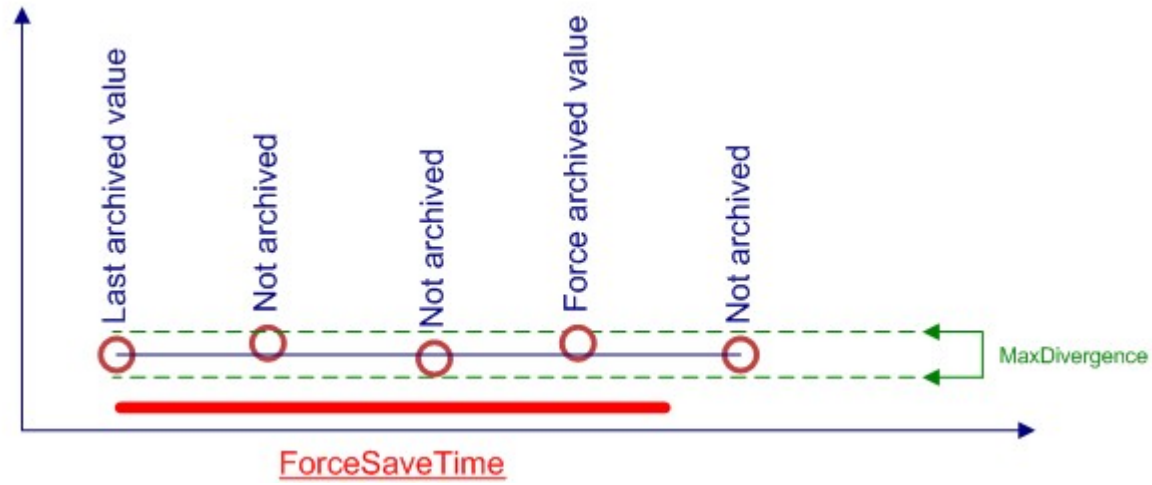
2.1.8.6.1 ForceSaveTime

The ForceSaveTime property is a time, in seconds, for how long new values can be rejected due to the compression function. If the ForceSaveTime has passed since last archived value, the previous value will be archived (by force).

Valid for: Analog, Digital
Data type: bigint

More...

2.1.8.6.1.1 ForceSaveTime sample



2.1.8.6.2 ForceUpdRate

ForceUpdRate is the update rate in milliseconds for signals related to this server to be checked if MaxForceSaveTime has been passed for any tags.
It also serves the purpose of moving last update time from this server to be used when presenting realtime data.

This time is “kept alive” by an active pulse from the OPCClient. The Frequency of this pulse must not be longer than the ForceUpdRate.
The Frequency of the pulse is set in the OPCDA applications configuration file.
The parameter in the configuration file is “ChkSigForceUpdates_ms” and is entered in ms.

Valid for: Analog, Digital
Data type: bigint

2.1.8.6.3 ForeColor

The ForeColor is an Event log property that can be used to store the events foreground color.
The color should be represented as a HTML color specification.

Valid for: Event
Data type: varchar(10)

2.1.8.7 G

2.1.8.7.1 GroupingPoint for calculations

GroupingPoint is used to decide where to put the timestamp for a time calculated TSQL-formula.
In procedures for storing formulas and internally in the database the following codes are used:
1 – The calculated value and timestamp is placed in the beginning of the calculated period.
2 – The calculated value and timestamp is placed in the middle of the calculated period.
4 – The calculated value and timestamp is placed at the end of the calculated period.

Valid for: Analog
Data type: tinyint

2.1.8.7.2 GroupingPoint for data retrieve

GroupingPoint is used to decide where to put the timestamp..
In procedures for retrieving data the following codes are used:
0 – The calculated value and timestamp is placed in the beginning of the calculated period.
1 – The calculated value and timestamp is placed in the middle of the calculated period.
2 – The calculated value and timestamp is placed at the end of the calculated period.

Valid for: Analog
Data type: tinyint

2.1.8.8 I

2.1.8.8.1 InCalc

The InCalc property sets whether the signal is in a calculation. It is also referred to as CalcTrigger.

1 – in calculation.
0 – not in calculation.

Valid for: Analog, Digital
Data type: bit

2.1.8.8.2 Increment

Input parameter to Sig_Select_Calc procedure specifying the time interval on interpolated values.
If increment input is a positive value, the increment parameter is in seconds between each record.
If increment input is a negative value, the increment parameter is in milliseconds.

Valid for: Analog, Digital
Data type: smallint

2.1.8.8.3 Interpol

If the Interpol property is set, the values are interpolated by straight line between each value.
If the property is not set the values are plotted as steps. The Interpol property is only used with the return values from the Sig_Select_Calc and the Sig_Select_Serie procedures.

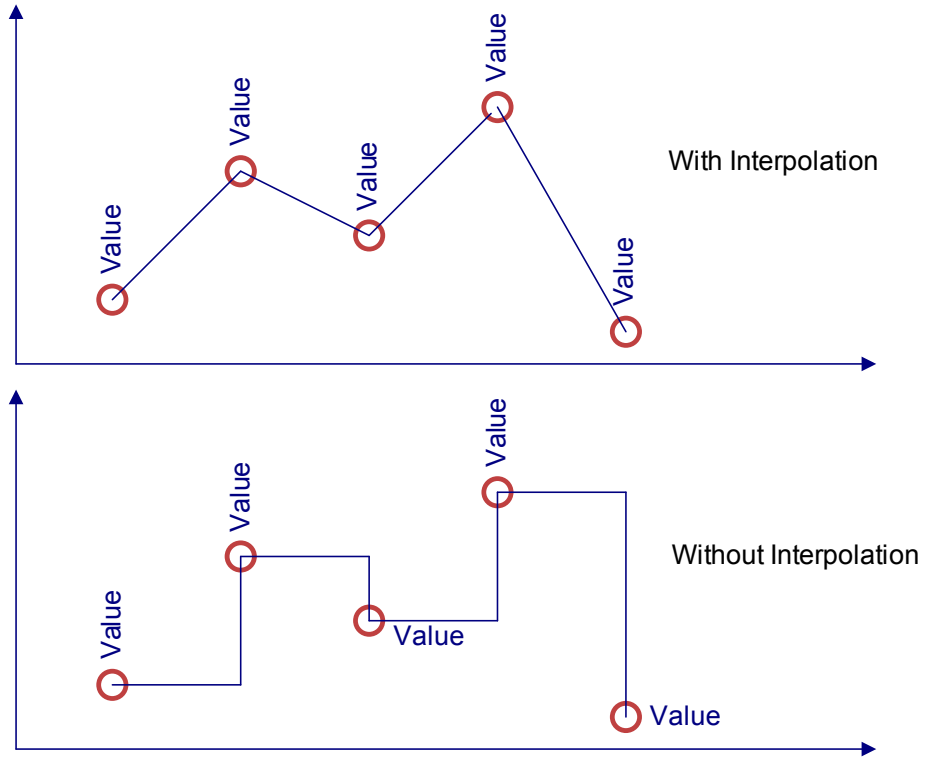
0 – No interpolation
1 – Interpolation

Valid for: Analog, Digital

Data type: bit

More...

2.1.8.8.3.1 Interpol sample



2.1.8.9 L

2.1.8.9.1 LastChangeTime

Timestamp for when the Last value was logged.

Valid for: Analog, Digital
Data type: datetime

2.1.8.9.2 LastQuality

Quality value for Last logged value logged.

Valid for: Analog, Digital
Data type: smallint

2.1.8.9.3 LastValue

The value of the last logged value.

Valid for: Analog, Digital
Data type: float (for Analog values), tinyint (for Digital values)

2.1.8.9.4 LiveRequest

The last time a client requested the last value of this signal.

Valid for: Analog
Data type: datetime

2.1.8.9.5 LogBlocking

The LogBlocking property blocks a signal from being logged to the database.
When LogBlocking is activated or deactivated, the OPCDA application needs to be restarted to effectuate the changes. The OPCDA application excludes a blocked tag from scheduling.

0 – Signal is not blocked
1 – Signal is blocked, and no data is archived.

Valid for: Analog, Digital
Data type: bit

2.1.8.10 M

2.1.8.10.1 MaxDivergence

MaxDivergence property is used for setting the range on compression function.

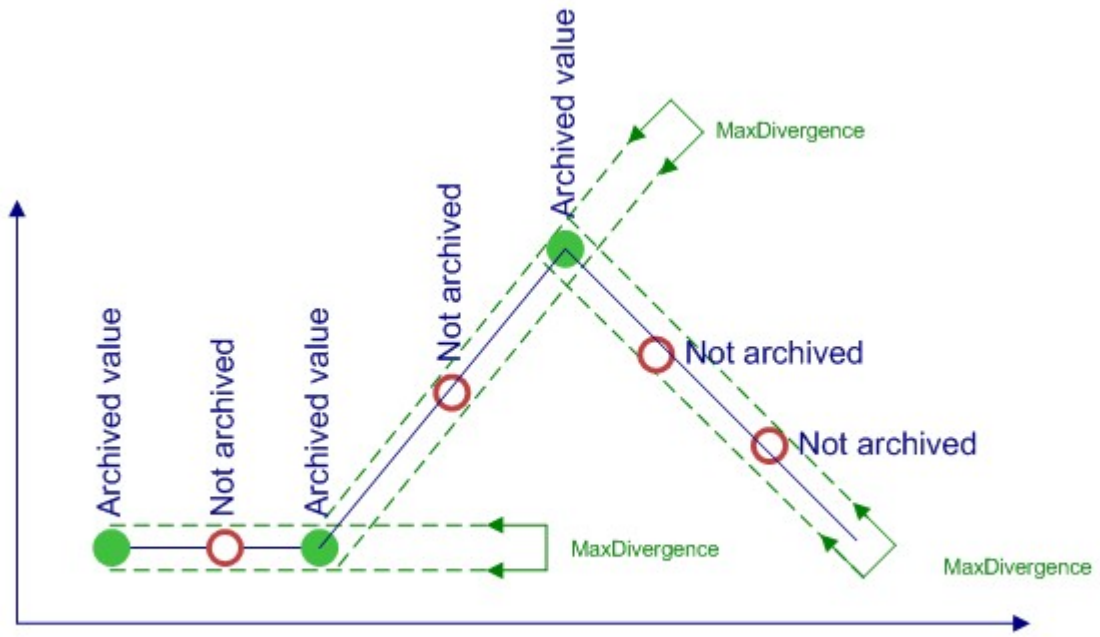
If the derivate value between the last archived value and the last received value are within MaxDivergence the value will not be archived.

The MaxDivergence can be set manually by an administrator or automatically by the CompTune application. The CompTune application performs an advanced signal analyze of each signal to find the accuracy level of the signal. When finding that with good accuracy, tuning of the compression can be made.

Valid for: Analog
Data type: float

More...

2.1.8.10.1.1 MaxDivergence sample



2.1.8.10.2 Message

Message text which describes the event. For condition-related events, this will generally include the description property of the active sub-condition.

Valid for: Events
Data type: nvarchar(1000)

2.1.8.11 N

2.1.8.11.1 New State

OPCAE Standard field
NewState is A WORD bit mask of three bits specifying the new state of the condition:
OPC_CONDITION_ACTIVE (= 2),
OPC_CONDITION_ENABLED (= 1) ,
OPC_CONDITION_ACKED (= 4)

Valid for: Event
Data type: tinyint

2.1.8.12 O

2.1.8.12.1 OPCDAClientName

The name of the OPCDA client instance connected to collect data from this OPCServer.
The OPCDAClient name must be equal with the parameter specifying this in the configuration file of the OPCDA client application.

Valid for: Analog, Digital
Data type: nvarchar(50)

2.1.8.12.2 OPCGroupID

A unique ID identifying the OPC group and is generated by the system.

Valid for: Analog, Digital
Data type: smallint

2.1.8.12.3 OPCGroupName

Name of the OPC group.

Valid for: Analog, Digital
Data type: nvarchar(50)

2.1.8.12.4 OPCItemID

The signals item id on the OPC server.

Valid for: Analog, Digital
Data type: nvarchar(1000)

2.1.8.12.5 OPCServerID

Unique ID identifying the OPC server and is generated by the system.

Valid for:
Data type: smallint

2.1.8.12.6 OPCServerName

Name of the OPC server that are connected. This name is case sensitive. Make sure that the name is entered correct.

Valid for. Analog, Digital
Data type: nvarchar(200)

2.1.8.12.7 OPCServerNode

The computer name or IP address where the OPC Server is located that are connected.

Valid for: Analog, Digital
Data type:nvarchar(50)

2.1.8.12.8 Origin

Describes the origin of the data for this signal.

Data type: nvarchar(200)

2.1.8.13 Q

2.1.8.13.1 Quality

The Quality attribute represents the quality state for a signals value.
The low 8 bits of the Quality flags are defined as three fields: Quality, Substatus and Limit status.
The 8 Quality bits are arranged as follows: QQSSSSLL.

Valid for: Analog, Digital, Events
Data type: smallint

More...

2.1.8.13.1.1 Quality sample

QUALITY Status

QQ	Description
0 – Bad value	The value is bad for reasons specified by Substatus.
64 – Uncertain	Uncertain quality for reasons indicated by Substatus.
128 – N/A	
192 – Good	The value is good.

Substatus for BAD Quality

SSSS	Description
0 – Non-specific	The value is bad but no specific reason is known.
4 – Config. Error	There is some server specific problem with the config.
8 – Not connected	The input is required to be logically connected to something but is not.
12 – Device failure	A device failure has been detected.
16 – Sensor failure	A sensor failure had been detected.
20 – Last known value	Communications have failed, but the last known value is available.
24 – Comm. Failure	Communications have failed. There is no last known value is available.
28 – Out of service	The block is off scan or otherwise locked. This quality is also used when the active state of the item or the group containing the item is InActive.
32 – Waiting for initial data	This substatus is only available from OPC DA 3.0 or newer servers.
36 - 60	Reserved for future use.

Substatus for UNCERTAIN Quality

SSSS	Description
64 – Non-specific	There is no specific reason why the value is uncertain.
68 – Last Usable Value	The returned value should be regarded as 'stale'. Note that this differs from a BAD value with Substatus 5 (Last Known Value). That status is associated specifically with a detectable communications error on a 'fetched' value. This error is associated with the failure of some external source to 'put' something into the value within an acceptable period

	of time.
72 – 76	Not used by OPC
80 – Sensor not accurate	Either the value has 'pegged' at one of the sensor limits (in which case the limit field should be set to 1 or 2) or the sensor is otherwise known to be out of calibration via some form of internal diagnostics (in which case the limit field should be 0).
84 – Engineering units exceeded	The returned value is outside the limits defined for this parameter. Note that in this case the 'Limits' field indicates which limit has been exceeded but does NOT necessarily imply that the value cannot move farther out of range.
88 – Sub-normal	The value is derived from multiple sources and has less than the required number of Good sources.
92 - 124	Reserved for future use.

Substatus for GOOD Quality

SSSS	Description
192 – Non-specific	The value is good. There are no special conditions.
196 – 212	Not used.
216 – Local override	The value has been Overridden. A manually entered value has been 'forced'.
220 – 252	Reserved for future use.

Limit status

LL	Description
0 – Not limited	The value is free to move up or down.
1 – Low limited	The value has 'pegged' at some lower limit.
2 – High limited	The value has 'pegged' at some high limit.
3 – Constant	The value is a constant and cannot change.

The high 8 bits of the Quality Word are available for vendor specific use. If these bits are used, the standard OPC Quality bits must still be set as accurately as possible to indicate what assumptions the client can make about the returned data. In addition it is the responsibility of any client interpreting vendor specific quality information to insure that the server providing it uses the same 'rules' as the client.

AutArch specific Quality state:
256 – Interpolated point (for example used in AutArch Export)
512 – Backlog point
1024 – Polled value from the data source
2048 – Manual data
4096 – Override data
8192 – Startup/Shutdown

2.1.8.14 R

2.1.8.14.1 RangeMax

RangeMax is used to set the default max range for the signal when opened in viewing applications, e.g. Trendviewer.

Valid for: Analog
Data type: float

2.1.8.14.2 RangeMax2

RangeMax2 is used to set the default max range for the signal when opened in viewing applicatios, e.g. Trendviewer and using an alternative engineering unit.

Valid for: Analog
Data type: float

2.1.8.14.3 RangeMin

RangeMin is used to set the default min range for the signal when opened in viewing applicatios, e.g. Trendviewer.

Valid for: Analog
Data type: float

2.1.8.14.4 RangeMin2

RangeMin2 is used to set the default min range for the signal when opened in viewing applicatios, e. g. Trendviewer and using an alternative engineering unit.

Valid for: Analog
Data type: float

2.1.8.14.5 RejectSaveTime

RejectSaveTime overrides the compression function and rejects any values to be archived to the database, during this time (in seconds).

Valid for: Analog, Digital
Data type: bigint

More...

2.1.8.14.5.1 RejectSaveTime sample

2.1.8.14.6 Responsible

The name of the responsible data broker for each signal. Usually the same as the OPCDAClientName that the data broker is pointing to.

2.1.8.15 S

2.1.8.15.1 ServiceChanges

The number of times the signal has changed from 0 (zero) to 1 (one), since last service.

Valid for: Digital
Data type: bigint (in def table) or int (in log table)

2.1.8.15.2 ServiceLimitChanges

The limit of number of times the signal can change from 0 (zero) to 1 (one), since last service.

Valid for: Digital
Data type: bigint

2.1.8.15.3 ServiceLimitRunTime

The limit of the total operating time, in seconds, that the signal is allowed to be active (0 or 1 depending on configuration of ActiveRunTime), since last service.

Valid for: Digital
Data type: bigint

2.1.8.15.4 ServiceRunTime0

The total time, in seconds, that the signal has been 0 (zero), since the last service.

Valid for: Digital
Data type: float (in def table) or int (in log table)

2.1.8.15.5 ServiceRunTime1

The total time, in seconds, that the signal has been 1 (one), since the last service.

Valid for: Digital
Data type: float (in def table) or int (in log table)

2.1.8.15.6 ServiceRunTimeActive

The total time, in seconds, that the signal has been active (0 or 1 depending on configuration of ActiveRunTime), since last service.

Valid for: Digital
Data type: int

2.1.8.15.7 ServiceTime

Timestamp when last service was registered by user.

Valid for: Digital
Data type: datetime

2.1.8.15.8 Severity

OPCAE Standard field.
The severity value is an indication of the urgency of the sub-condition. This is also commonly called 'priority', especially in relation to process alarms.
Values will range from 1 to 1000, with 1 being the lowest severity and 1000 being the highest.
Typically, a severity of 1 would indicate an event which is informational in nature, while a value of 1000 would indicate an event of catastrophic nature which could potentially result in severe financial loss or loss of life.

Valid for: Event
Data type: smallint

More...

2.1.8.15.9 SignalDesc

The description of the signal

Valid for: Analog, Digital, Event
Data type: nvarchar(255)

2.1.8.15.10 SignalDesc2

The description of the signal in a alternative language.

Valid for: Analog, Digital, Event
Data type: nvarchar(255)

2.1.8.15.11 SignalID

Unique ID identifying the signal and is generated by the system.
The ID cannot be changed. If signals are imported to the database, new SignalID's will be created if necessary.

Valid for: Analog, Digital, Event, MultiValue
Data type: smallint

2.1.8.15.12 SignalName

A unique, user input, name for the signal.
The SignalName can be changed.

Valid for: Analog, Digital, Events, MultiValue

Data type: nvarchar(1000)

2.1.8.15.13 SignalName2

The name of the signal in a alternative language.

Valid for: Analog, Digital, Events, MultiValue
Data type: nvarchar(1000)

2.1.8.15.14 SignalType

Bitmasked number to set the functionality of the tag.
Bit:
1 – Digital signal
2 – Analog Signal
4 – Signal has events
8 – Text (for future use)
16 – Calculated
32 – Out
64 – Extend last value
128 – Signal has Multi values. Can be combined with Analog or Digital.
256 – Signal is periodic. The periodicty is set in the CalcFormula field.
512 – Disable recalc
1024 – Disable import
2048 – Disable correction
4096 – AutArch system diagnostic tag

Valid for: Analog, Digital, Events, MultiValue
Data type: smallint

2.1.8.15.15 SourceOffset

SourceOffset and SourceScale is used to rescale an input value if the input has a wrong measuring scale.

If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula:

```
The Stored Value = The Input Value * SourceScale + SourceOffset
```

Valid for: Analog
Data type: float

2.1.8.15.16 SourceRangeMax (Reading)

SourceRangeMax is a calculated value based on SourceScale and SourceOffset.
If SoureScale is not equal 0 then The signals RangeMax is rescaled and returned as SourceRangeMax.

Valid for: Analog
Data type: float

2.1.8.15.17 SourceRangeMax (Writing)

If SourceRangeMin and SourceRangeMax is used when creating or updating a Signal this value can be used to calculate and set the correct values of:
SourceScale and SourceOffset based on RangeMin and RangeMax.

The SourceScale is calculated according to:
SourceScale = (RangeMax - RangeMin) / (SourceRangeMax - SourceRangeMin)

And the SourceOffset is calculated according to:
SourceOffset = RangeMin - (SourceRangeMin * SourceScale)

Valid for: Analog
Data type: float

2.1.8.15.18 SourceRangeMin (Reading)

SourceRangeMin is a calculated value based on SourceScale and SourceOffset.
If SoureScale is not equal 0 then The signals RangeMin is rescaled and returned as SourceRangeMin.

Valid for: Analog
Data type: float

2.1.8.15.19 SourceRangeMin (Writing)

If SourceRangeMin and SourceRangeMax is used when creating or updating a Signal this value can be used to calculate and set the correct values of:
SourceScale and SourceOffset based on RangeMin and RangeMax.

The SourceScale is calculated according to:
SourceScale = (RangeMax - RangeMin) / (SourceRangeMax - SourceRangeMin)

And the SourceOffset is calculated according to:
SourceOffset = RangeMin - (SourceRangeMin * SourceScale)

Valid for: Analog
Data type: float

2.1.8.15.20 SourceScale

SourceScale and SourceOffset is used to rescale an input value if the input has a wrong measuring scale.

If SourceScale is not equal 0, the value is rescaled before it is stored to the database according to the following formula:

```
The Stored Value = The Input Value * SourceScale + SourceOffset
```

Valid for: Analog
Data type: float

2.1.8.15.21 SourceUnit

The SourceUnit is mapped to the Signals ValueUnit, e.g °C

Valid for: Analog
Data type: nvarchar(50)

2.1.8.15.22 SubConditionName

OPCAE Standard field.
Name of current sub-condition, for multi-state conditions.
For a single-state condition, this contains the condition name.
Valid for: Events

Data type: nvarchar(200)

2.1.8.16 T

2.1.8.16.1 TargetFormula

This field is not used.

Valid for: Analog
Data type: varchar(2000)

2.1.8.16.2 TargetOffset

TargetOffset and TargetScale is used to convert an Signals value to an alternative Unit

If TargetScale is not equal 0, the value is converted according to the following formula:
```
The Converted Value = The Stored Value * TargetScale + TargetOffset
```

Valid for: Analog
Data type: float

more...

2.1.8.16.3 TargetScale

TargetScale and TargetOffset is used to convert an Signals value to an alternative Unit

If TargetScale is not equal 0, the value is converted according to the following formula:
```
The Converted Value = The Stored Value * TargetScale + TargetOffset
```

Valid for: Analog
Data type: float

more...

2.1.8.16.4 TargetUnit

TargetScale and TargetOffset is used to convert an Signals value to an alternative Unit

If TargetScale is not equal 0, the value is converted according to the following formula:
```
The Converted Value = The Stored Value * TargetScale + TargetOffset
```

Valid for: Analog
Data type: nvarchar(50)

2.1.8.16.5 TdrLevel

Defines on which flank the trigger Signal should trig the Trigged Data Recorder

1 = The Signal triggers when going from 0 to 1
0 = The Signal triggers when going from 1 to 0

Valid for. Analog and Digital signals
Data type: bit

2.1.8.16.6 TdrLogSignals

A list of signals that should be logged by the Trigged Data Recorder when trigged by this signal. The signals are stored in an relational table but they are edited by writing an list of signal names in the following way:
```
{SignalName1}{SignalName2}{SignalName3}...
```

Valid for: Analog, Digital
Data type: nvarchar(2000)

2.1.8.16.7 TdrOPCGroupID

A unique ID identifying the Trigged Data Recorder OPC group and is generated by the system.

Valid for: Analog, Digital
Data type: smallint

2.1.8.16.8 TdrOPCGroupName

A unique Name identifying the Trigged Data Recorder OPC group.

Valid for: Analog, Digital
Data type: nvarchar(50)

2.1.8.16.9 TdrOPCItemID

The signals item id on the OPC server for a Trigged Data Recorder signal.

Valid for: Analog, Digital
Data type: nvarchar(1000)

2.1.8.16.10 TdrPostBuffer

The amount of data that will be saved to the database after the signal has been trigged by the Trigged Data Recorder.
The time is specified in fractions of seconds.

Valid for. Analog and Digital signals
Data type: real

2.1.8.16.11 TdrPreBuffer

The amount of data that will be saved to the database before the signal has been trigged by the Trigged Data Recorder.
The time is specified in fractions of seconds.

Valid for. Analog and Digital signals
Data type: real

2.1.8.16.12 TdrTrigger

Defines if the Signal is a trigger for the Trigged Data Recorder.

1 = The Signal is a trigger.

Valid for. Analog and Digital signals
Data type: bit

2.1.8.16.13 TimeDiff

The difference, in seconds between ServiceLimitRuntime – ServiceRunTimeActive.

Valid for. Digital
Data type: bigint

2.1.8.16.14 TotalChanges

The total number of times the signal has changed from 0 (zero) to 1 (one).

Valid for: Digital
Data type: bigint (in def table) or int (in log table)

2.1.8.16.15 TotalRunTime0

The total number of seconds the signal has been 0 (zero), since the first value was logged.

Valid for: Digital
Data type: float (in def table) or int (in log table)

2.1.8.16.16 TotalRunTime1

The total number of seconds the signal has been 1 (one), since the first value was logged.

Valid for: Digital
Data type: float (in def table) or int (in log table)

2.1.8.16.17 TotalRunTimeActive

The total number of seconds the signal has been active (0 or 1 depending on configuration, TotalRunTime0 or TotalRunTime1), since the first value was logged.

Valid for: Digital
Data type: int

2.1.8.16.18 TriggerUpdRate

The update rate in milliseconds for time calculated signals related to this server, to be triggered for new calculation.

This time must be shorter than the minimum average time configured for any tags.
This time is “kept alive” by an active pulse from the OPCClient. The Frequency of this pulse must not be longer than the TriggerUpdRate.

If no OPCClient is connected. The OPCServer “Internal” can be used instead. The Internal server is triggered by an internal scheduler, but the update rate can not be less than 60s for this Internal server.

Valid for: Analog, Digital
Data type: bigint

2.1.8.16.19 TuneCount

The number of processed compression auto tunings. For internal use only.

Valid for: Analog
Data type: smallint

2.1.8.16.20 TuneStatus

The TuneStatus property is used by the CompTune application to determine the status of the compression for the signal. CompTune will change this property as different stages of the tuning are performed.

The value is bit coded as:
1- Represents that the signal should collect data for tuning analyze
2 - Represents that the database has collected enough RAW data for this signal to start the tuning analyze.
4 - Represents that the signal has finished its tuning analyze.
8 - Represents that the tuning analyze has Checked the signals tuning.
16 - Represents that the tuning is Enabled for this signal.

32 - Represents that the value of the reference tag is to low and the database has stopped collecting RAW data.
16384 - Represents that the signal is imported by XMZ-import.

Valid for: Analog
Data type: smallint

2.1.8.17 U

2.1.8.17.1 UpdateRate

Rate in milliseconds to update the OPC tag group with from the OPCServer. The rate is used when setting up the scheduling towards the OPCServer.

Valid for: Analog, Digital
Data type: int

2.1.8.18 V

2.1.8.18.1 ValueName

ValueName gives the possibility to use a description for the valuetype, e.g. “Power”.
(For future use by connecting applications.)

Valid for: Digital
Data type: nvarchar(50)

2.1.8.18.2 ValueName0

The ValueName0 property is used to provide a description for when signal is 0 (zero), such as “CLOSED” or “OFF”.

Valid for: Digital
Data type: nvarchar(50)

2.1.8.18.3 ValueName1

The property is used to provide a description for when signal is 1 (one), such as “OPEN” or “ON”.

Valid for: Digital
Data type: nvarchar(50)

2.1.8.18.4 ValueUnit

The unit for the signal, e.g m/s, kPa etc.

Valid for: Analog
Data type: nvarchar(50)

2.1.8.18.5 ValueUnit2

The unit for the signal, e.g m/s, kPa etc. in a alternative language

Valid for: Analog
Data type: nvarchar(50)

2.1.9 Install instructions

2.1.9.1 Prerequisites

The AutArch database requires Microsoft SQL Server 2008 or newer.

In the current version of AutArch Microsoft SQL Server 2014 Standard is included.
Basic prerequisites (for full list of prerequisites see link below):

- DotNet Framework: 3.5 and 4.0
- Windows Powershell
- Disk space: minimum 6GB
- RAM: minimum 4GB
- Processor speed: minumum 2.0 GHz
- Operating sytem: Windows Server 2008 R2 SP1, Windows Server 2012, Windows Server 2016, Windows 7 SP1, Windows 8, Windows 10

A complete list of prerequisites for Microsoft SQL Server 2014 can be found here: https://technet.microsoft.com/en-us/library/ms143506%28v=sql.120%29.aspx?f=255&MSPPError=-2147217396

2.1.9.2 Installation

The AutArch database can be installed in an existing or in a new Microsoft SQL Server instance.

Install the database by starting "AutArchDatabaseSetup.exe" and follow the steps in the wizard.
If a new Microsoft SQL server should be installed make sure that you are using the SQL setup package included in the AutArch release.

2.1.9.3 Update

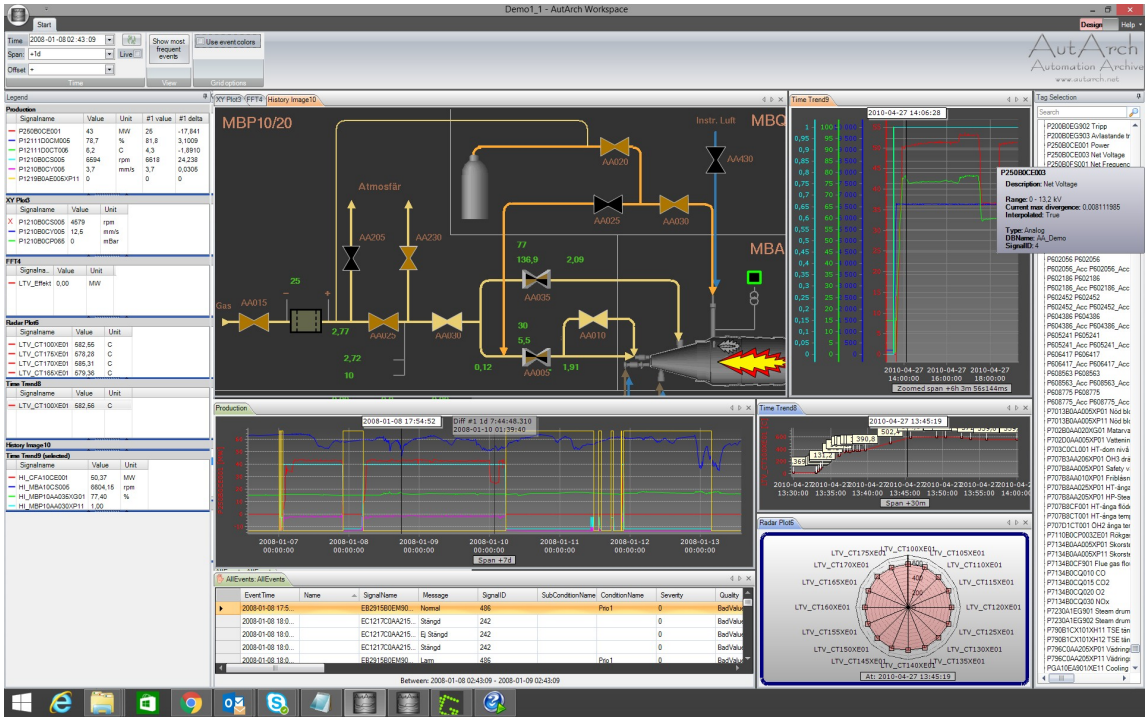
To update the Microsoft SQL server please see recommendations from Microsoft.

The AutArch database is updated by running Global update from Service monitor.

2.2 Workspace

2.2.1 Overview

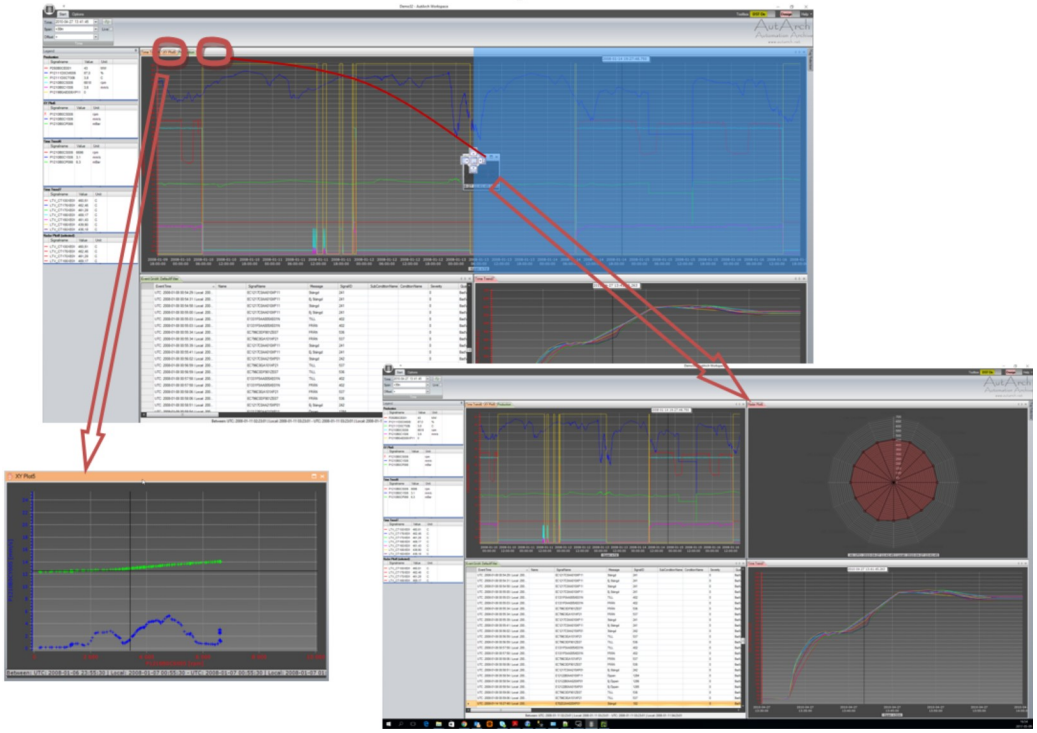
The AutArch Workspace is an application with several tools to view historical data in the database.
The user can create and save several different workspaces for different needs.
Picture below is a sample of a workspace with several different presentations in combinations that will be explained further on in this manual.



2.2.2 General Instructions

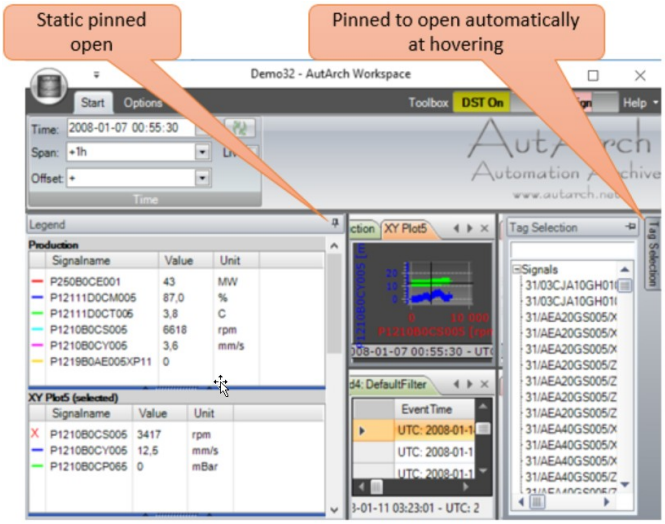
2.2.2.1 Work Space and Dockable windows

The work space works with "dockable windows" that enables a very flexible layout. Any window can be dragged and dropped at any place at the work space area. Windows moved out from a workspace do still have it's related tag list and legend in main workspace. All configurations regarding layout and disposition of the windows are stored when saving a workspace file.



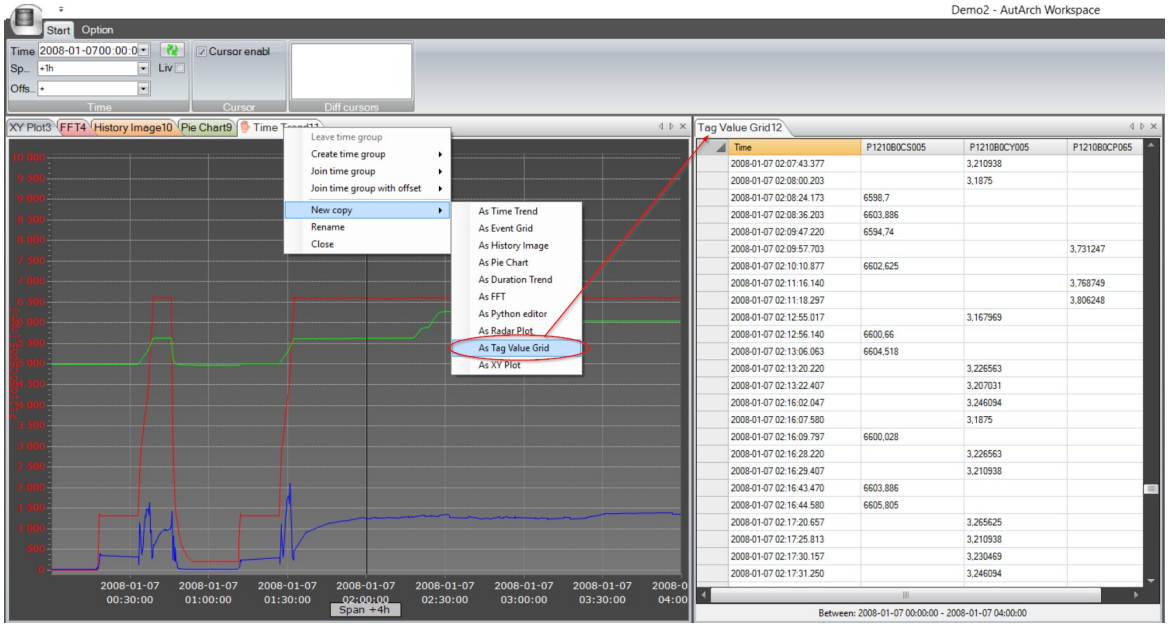
Legends and tag list can be dragged to other positions, but normally is to "pin" them open or dynamically opened when hovering over the panes. The ribbon bar on top can be hidden by double

click on "start" or "option folder" and opened permanent again by double click or dynamically by hovering.



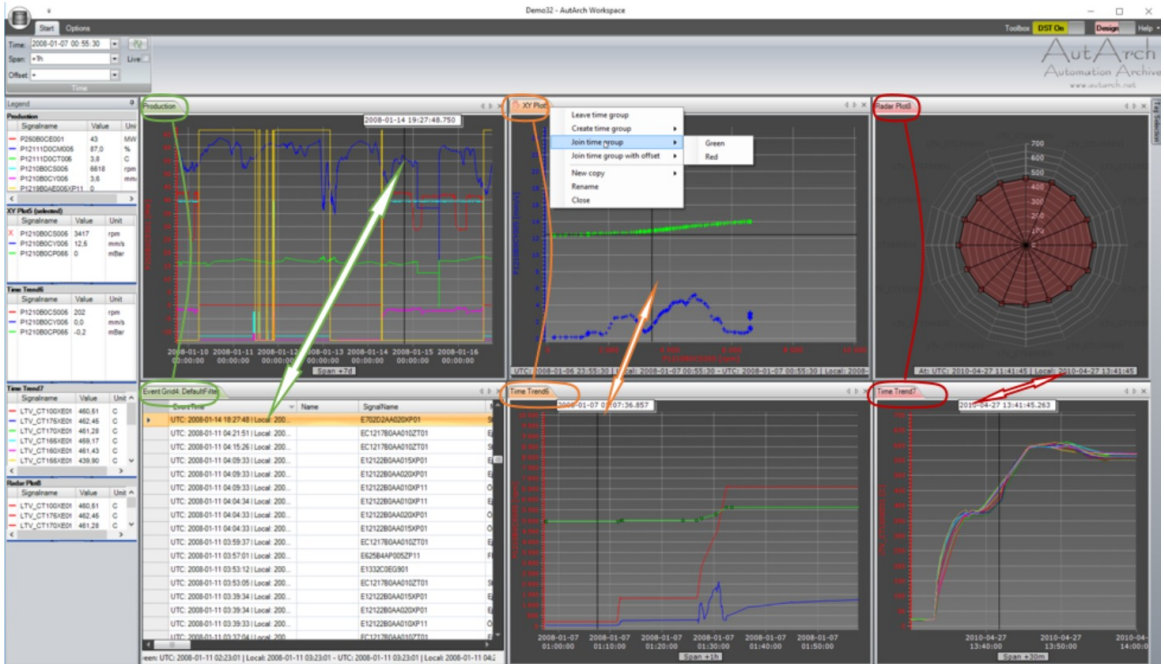
2.2.2.2 Copy presentation

A presentation can be copied by right clicking the presentation name and selecting new copy. The copy will contain the same signals as the original presentation and the same time period. The copy object can be any presentation possible to present the values into (except Dashboards or other types that requires additionally configuration. Picture below shows how to create a Tag Value grid from a time trend.



2.2.2.3 Timegroups

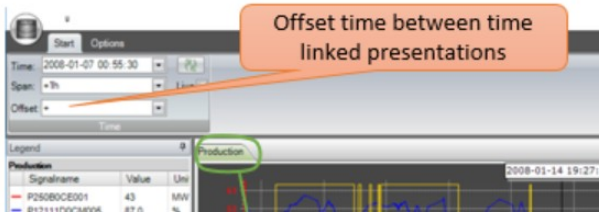
Presentations can be added to time groups by right clicking on the trend name and select Create or Join time group. Time groups are identified by colors.
All trends within a time group will use the time, moving the cursor in one trend automatically moves the cursor in all trend in the time group. Moving cursor in e.g. time trend related to event grid will reposition actual event to same time as cursor position in time trend. Panning a time trend will pan time linked trends as well.



Different presentation objects can interact in a different ways in time groups.

2.2.2.3.1 Offset

The offset option can be used for presentations in a timegroup. The presentation will then have the time from the timegroup + selected offset.



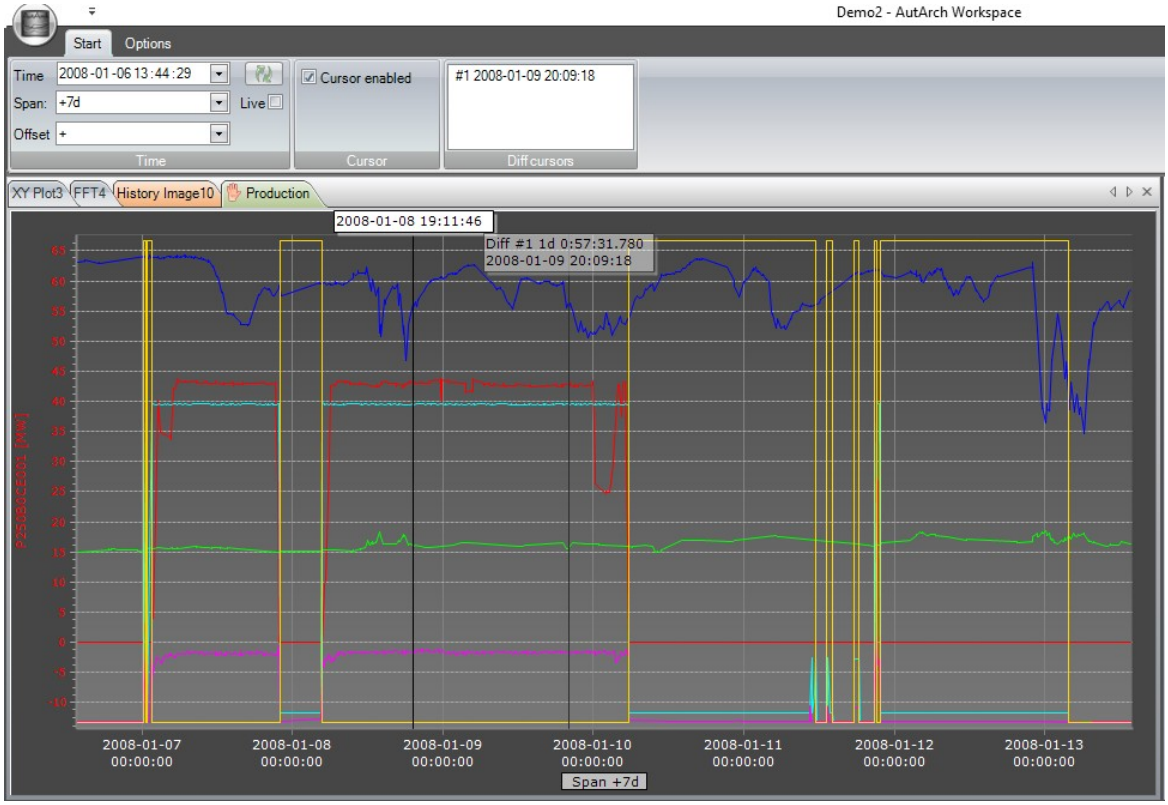
A trend can also join a time group with a time offset. The presentation will then keep it's time and the offset will be calculated automatically compared to the joined timegroup.

2.2.3 Presentations

User presentations are according to following picture and sub sections for end user presentation. Different user presentations and several instances of each presentation can be combined in one workspace. Following sections will give a very brief description of each presentation. More detailed descriptions and functions for each presentation can be found in "General Instructions" section.

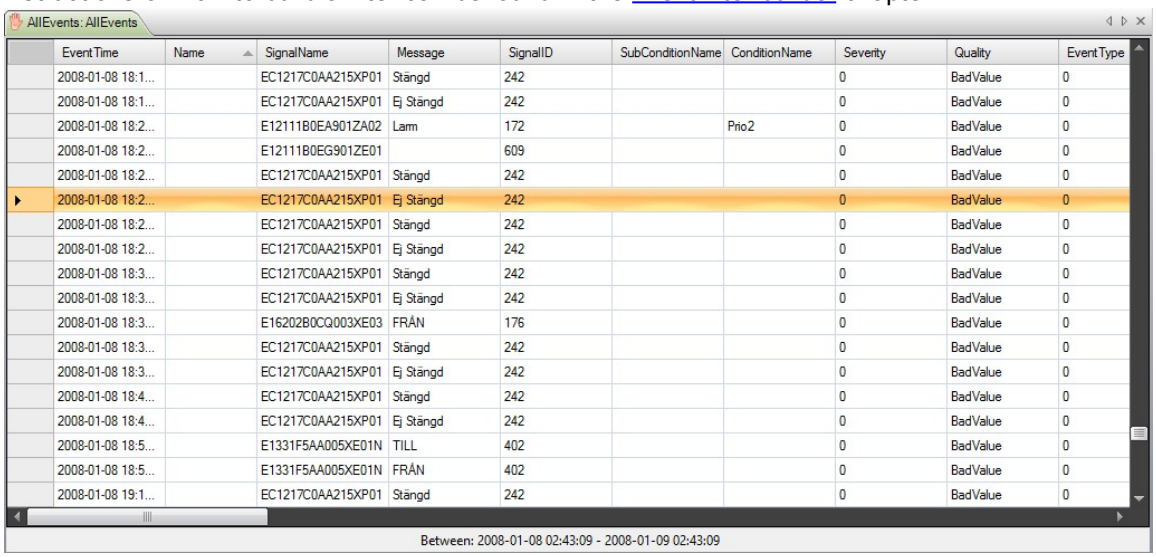
2.2.3.1 Time trend

The time trend is a basic trend which displays the selected tags values on a time axis. Trend is configured by dragging tags from taglist.



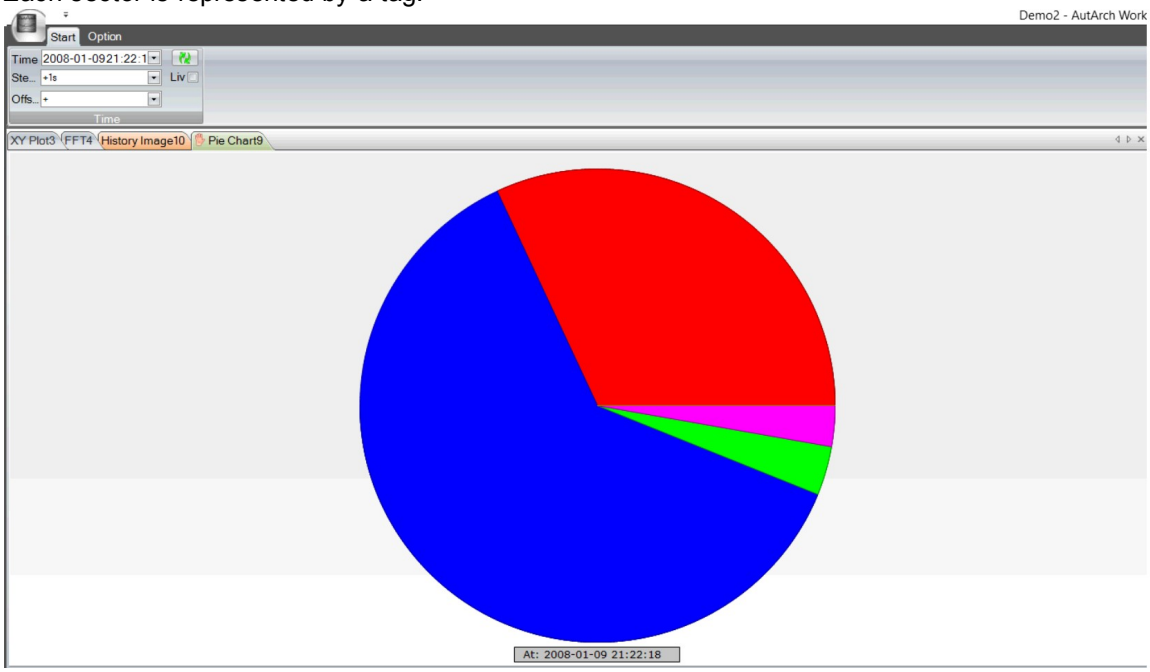
2.2.3.2 Event grid

The event grid is used to show alarms and events.
Select a filter from the Tag selector to filter out the events to show.
Instructions on how to build a filter can be found in the Event filter builder chapter.



2.2.3.3 Pie chart

The pie chart illustrates the numerical proportion between tags.
Each sector is represented by a tag.



2.2.3.4 Dashboardeditor

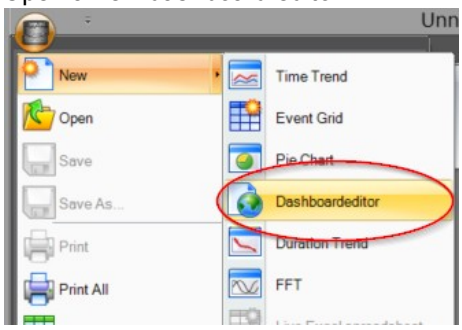
Dashboard presentations are mainly for presenting KPI data in a more customer adapted graphical way. The user can add background pictures and add presentations on top of background. Dashboards are published to a web server (RestProvider) and can be viewed in AutArch Workspace or any web browser.

The Dashboard needs a REST provider installed, that is configured to serve data from the desired database.

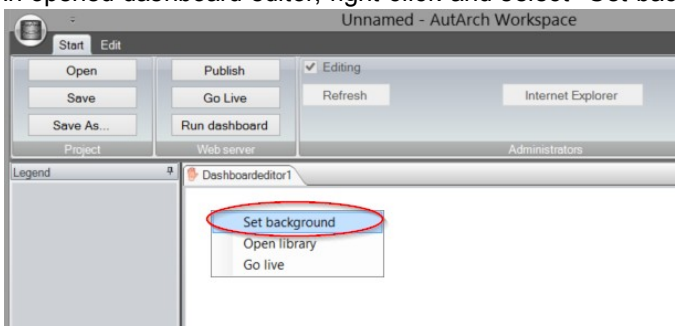
(To open a dashboard in a web browser the url is "http://'server name':'port'//AADashBoard/GetDashboard/'Dashboard name'", where 'server name' is replaced by the name or IP of the server running the RestProvider, 'port' is replaced by the network port (standard: 4717) and 'Dashboard name' is replaced by the name of the dashboard.)

If I want to create a sample dashboard that present weather data on a map. Following steps can be used.

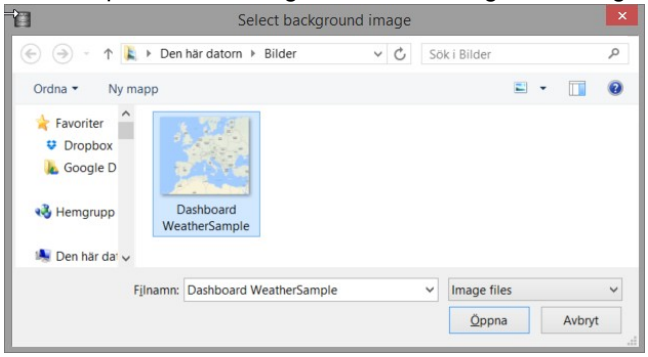
1. Open a new dashboard editor



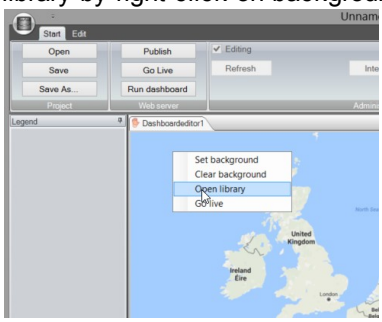
2. In opened dashboard editor, right click and select "Set background"



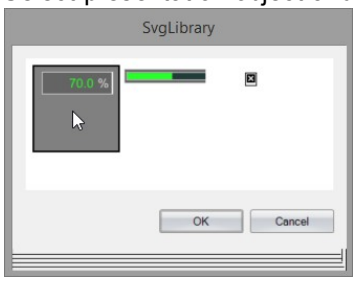
3. Select a picture as background. The background image decides the size of the dashboard.



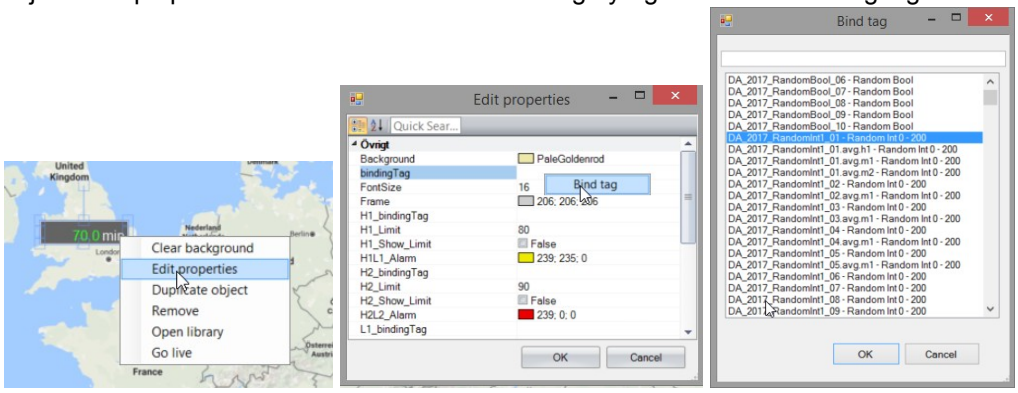
4. The background of he dashboard is now set. Add a value by inserting a value presenter from library by right click on background and select "Open library"



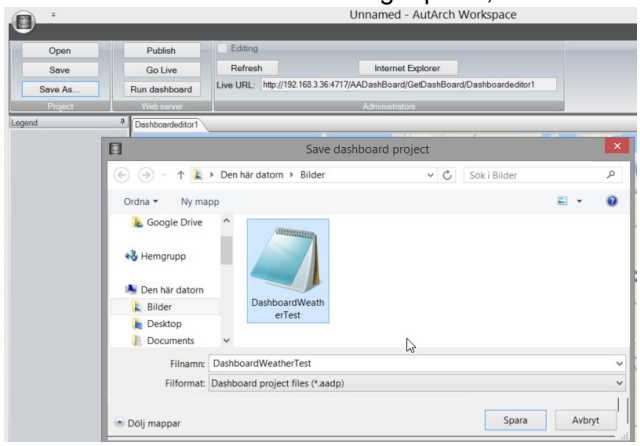
5. Select presentation object and press "OK".



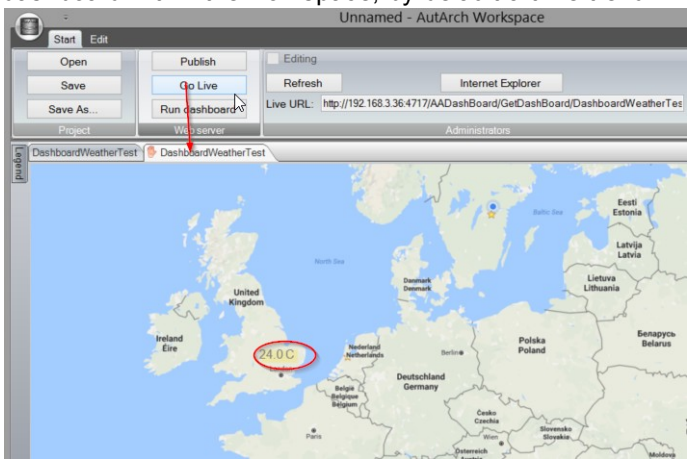
6. The presentation object is now placed on the dashboard. Move it by dragging the object. Format and connect it to database tag by right click on object and select "Edit properties". Format the object with properties and connect to database tag by right click on "bindingTag".



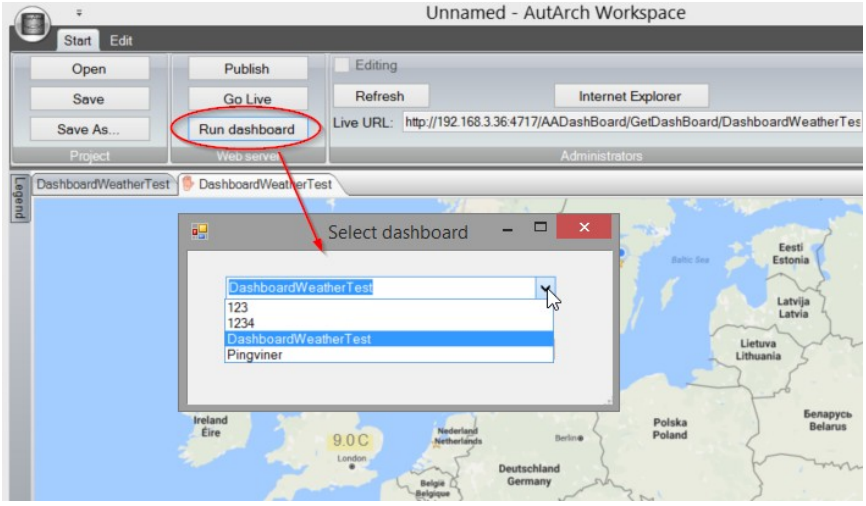
7. When formatted and located at right place, use "Save As" to save the dashboard.



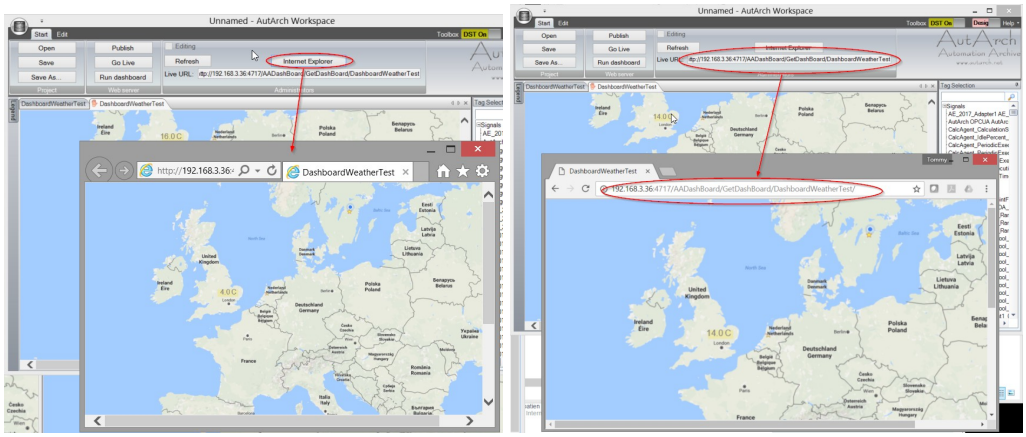
8. When saved, the dashboard can be "published" and after published "Go live" to present the actual published dashboard with values in real time. Note that this is the "view" mode. To make changes, changes to dashboards can only be made at source presentation. (Checkbox "Editing" displays whether selected dashboard is in edit mode or not.) If clicking on a presentation value in a dashboard within the workspace, by default a time trend will be opened with selected tag



9. If another dashboard is requested to be presented, "Run dashboard" button can be used and a selection of existing dashboards will appear to select from.

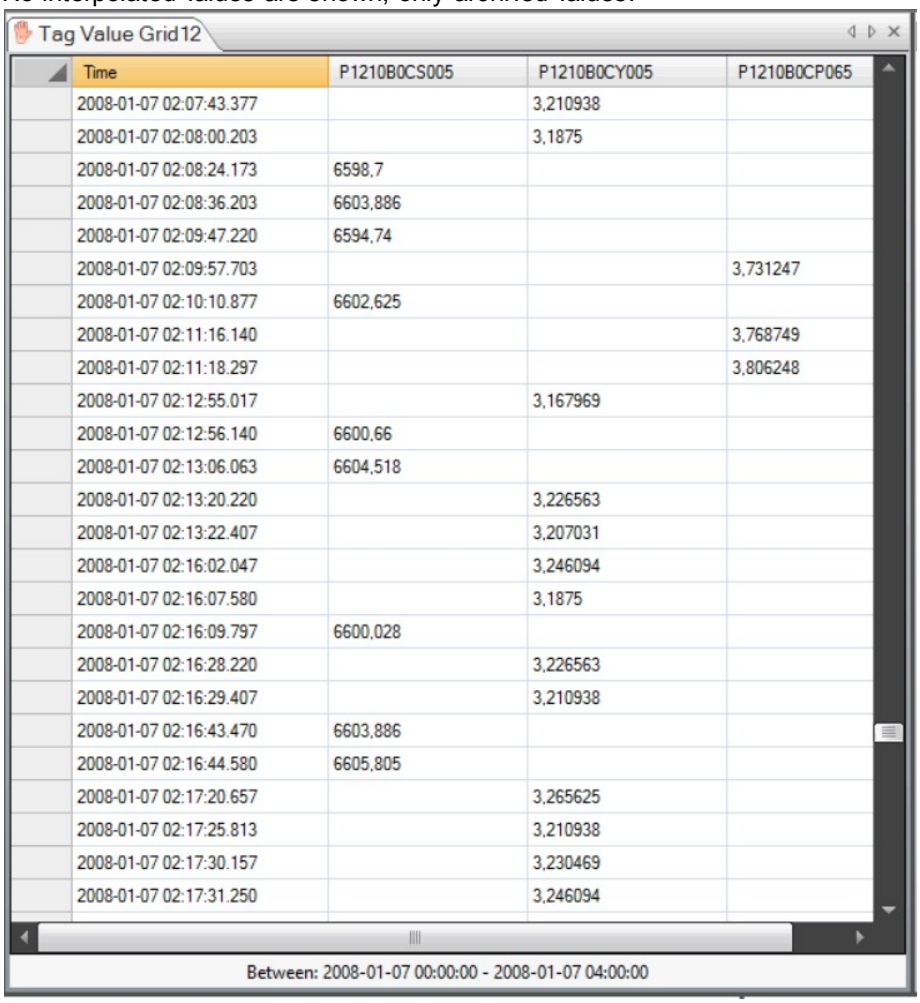


10. To view dashboard in a web browser. Use button "Internet Explorer" that will open actual dashboard in an Internet explorer session. Another option is to use "Live URL" and copy link to a browser of any kind.



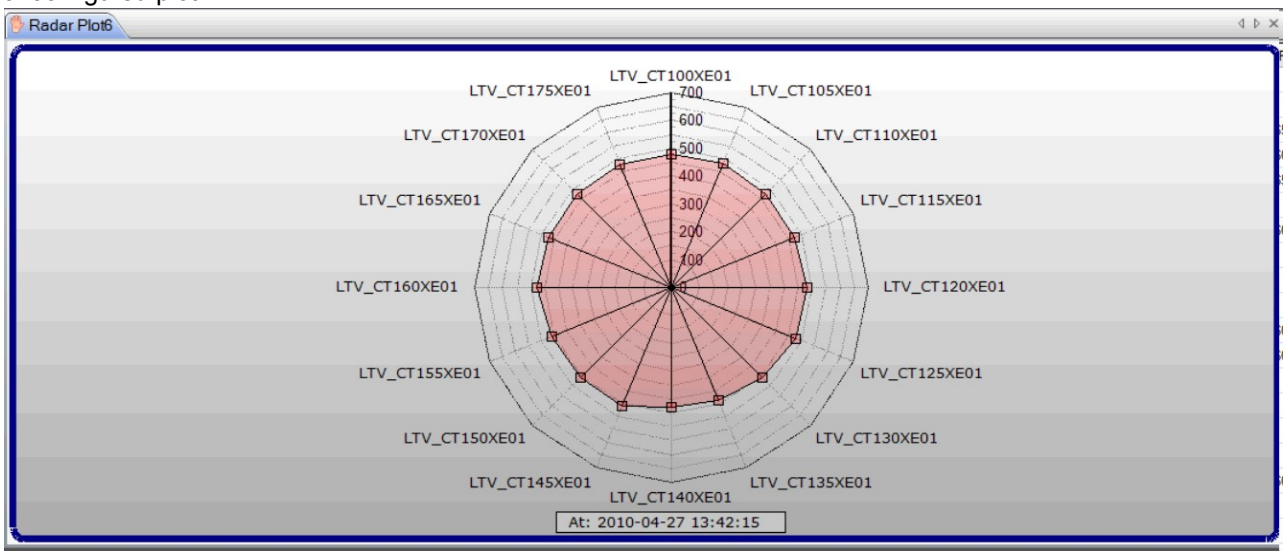
2.2.3.5 Tag Value grid

The tag value grid displays value points of selected tags.
No interpolated values are shown, only archived values.



2.2.3.6 Radar plot

A radar plot will plot added tags evenly spread on a 360 degree radar area. Each tags value will be scaled on it's axis from center of circle (minimum) to outer circle (maximum).
The trend type can be used to easy detect deviations on a "normal state" by having a unregular form of configured plot.



2.2.3.7 XY plot

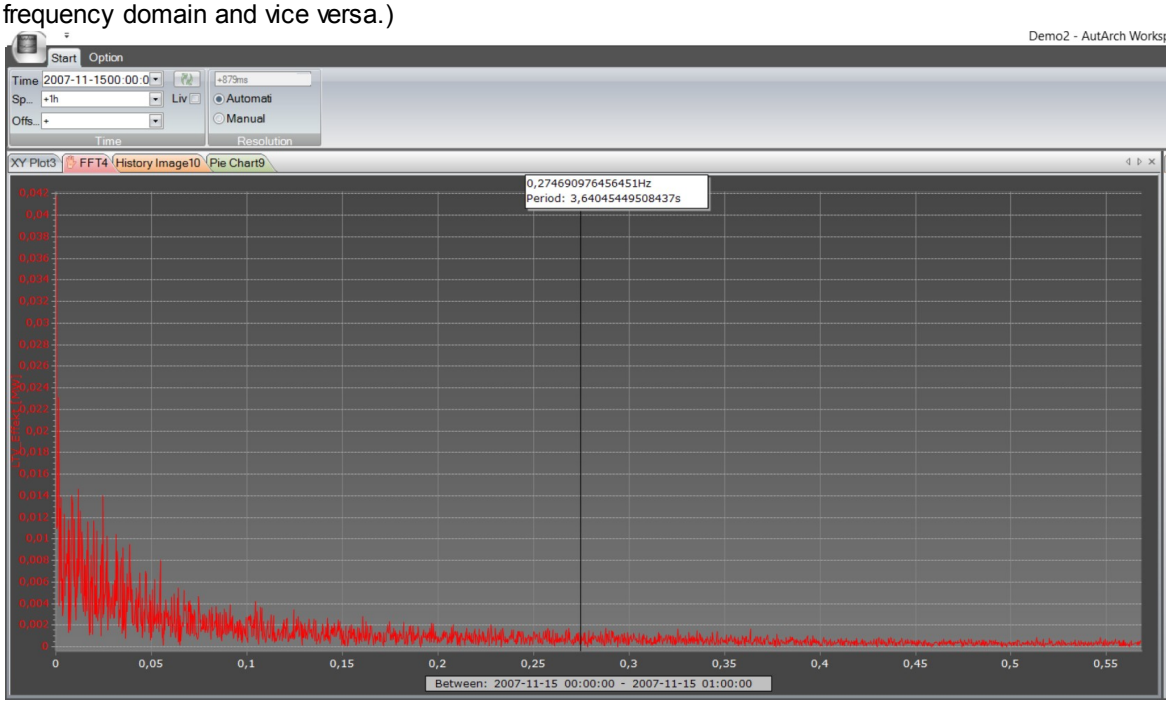
An XY plot is used for plotting one or several tags value related to another reference tag for a certain time span. The reference tag is plotted at X-Axis and the tags related on y-axis. I.e. each value at the reference X tag is representing a Y value at same time.
Picture below displays a vibration in an engine (Blue) related to speed (Red) on X-Axis.

2.2.3.8 Mikon

Mikon is another EMI (Enterprise Manufacturing Info) system within Prevas system suite. See separate documentation for this system. Mikon has been integrated with AutArch and this presentation type is to open a Mikon presentation.

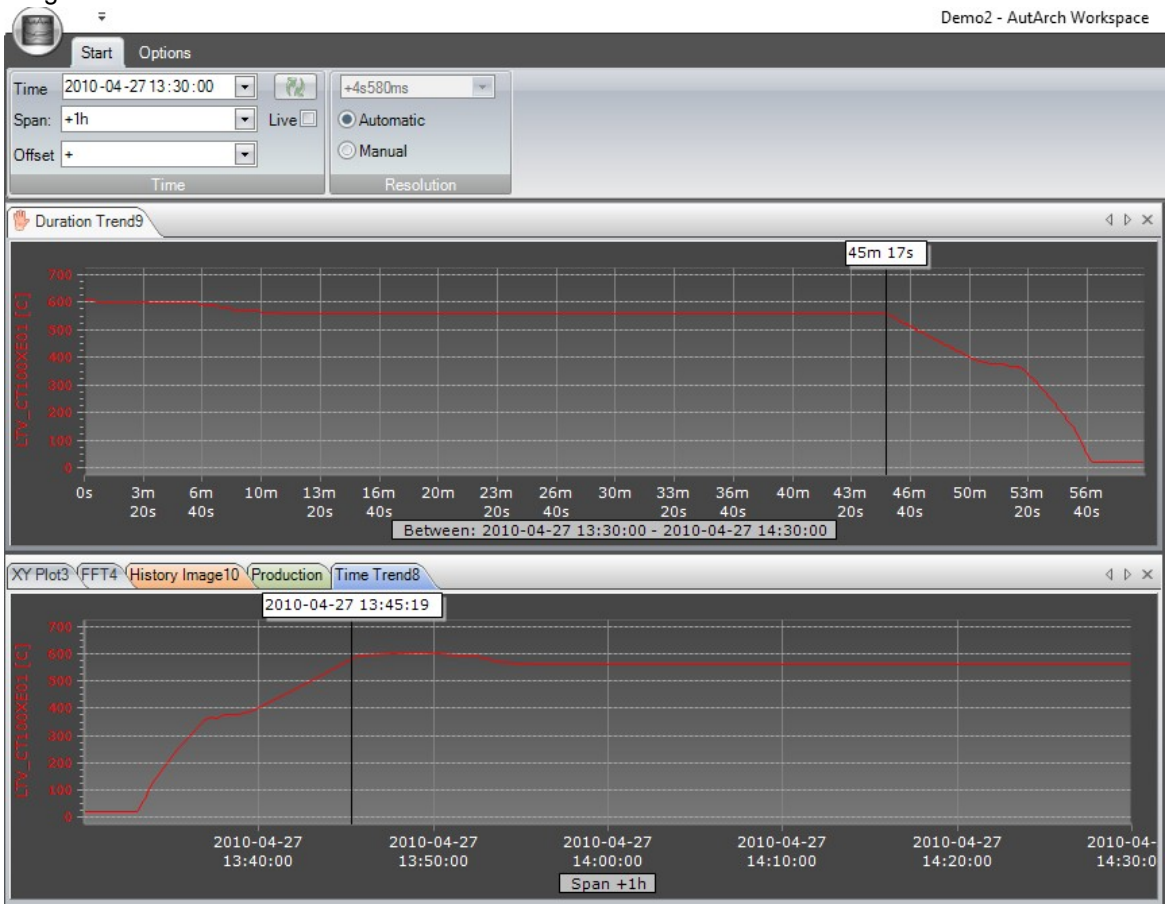
2.2.3.9 FFT

A FFT trend displays a FFT analyze of a signal. (Wiki definition of FFT = A fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.)



2.2.3.10 Duration trend

The duration trends sorts all value points from high to low.
This trend can be used to see for how long time the selected tags has been above a certain value in the selected time frame.
In picture below, a duration trend of same tag and time selection as a time trend below is displayed to explain how the trend types differs. The bottom standard time trend shows values sorted by time. The upper duration sorts values sorted by value. The duration trend can be used for e.g. visualizing usage or bottlenecks.



2.2.4 Tools

2.2.4.1 Admin

The admin tool is used to configure signals.
Configuration is made in grid. Any configuration can easily be copied between the admin tool and any other text editor.

A description of any column can be shown as a tooltip by hovering above the column header.

2.2.4.1.1 Description of tabs

2.2.4.1.1.1 All Signals (Simple View)

The signals view tabs consists a list of all tags and a few selected configurations.
To add a new signal to the system, start typing the new signal name in an empty row. Then fill in its various properties, for example; analog or digital signal type, OPC id, interpolation mode etc. Lastly, click 'Save changes'.

2.2.4.1.1.2 All Signals (Full View)

The signals view tabs consists a list of all tags and all available configuration.
To add a new signal to the system, start typing the new signal name in an empty row. Then fill in its various properties, for example; analog or digital signal type, OPC id, interpolation mode etc. Lastly, click 'Save changes'.

2.2.4.1.1.3 Aggregations

The aggregation tab displays all tags in any aggregation tree.

New aggregations can easily be created by right-clicking on any signal in the simple, full or aggregation view.
When creating a aggregation it can also be reaggregated by checking "Start aggregating from". Values will then be calculated for the signal from selected time.

Aggregations are commonly created with a fixed set of rules allowed aggregation levels and a suffix to add to the aggregated signal name.
To change allowed periods and suffix , click on "Aggregate options..." in the ribbon bar.

2.2.4.1.1.4 Event tags

The event tags tab displays a simplified list with only event tags and configuration related to events.

2.2.4.1.1.5 Opc servers

The Opc server tab is used to configure how a Recording node connects to a server.
- ID - Unique identifier, is set automatically when saving a new server.
- Archive - In which archives the configuration is stored, is set automatically when saving a new server.
- Server name - The name of the server
- Server node - The computer name or IP address where the server runs (usually set to localhost since the adapters are often installed on the same machine as the server).
- Client name - An identifier used by the Recording node to lookup it's configuration (also known as Responsible).

2.2.4.1.1.6 Opc groups

The Opc group tab is used to configure what groups will be handled by a Recording node.
All collected tags needs to be assigned to a group. Groups can also be used for other tags as a pure grouping function.

- ID - Unique identifier, is set automatically when saving a new server.
- Archive - In which archives the configuration is stored, is set automatically when saving a new server.
- Name - The AutArch internal name of the group.
- Update rate - Sample rate of the tags in the group.
- Server client name - Name of the Opc server (see opc server tab) to which the group is connected.

2.2.4.1.1.7 Unit conversions

Unit conversions is used when importing values.
If the tag in the import file has an unit that does not match the unit configured in the database all values will be converted according to the rules from this table.

2.2.4.1.2 Operations

2.2.4.1.2.1 Freeze filters and sorting

When this button is in the active state, the grid will stop continuously filtering and sorting the rows. This means that a row will keep its position in the grid even after editing a cell that normally is affected by filtering or sorting.
This button should be in the active state whenever a user wants to edit cells or add new rows to the grid.

The "freeze filters and sorting" button is located on the ribbon menu's Start-tab, in the Edit section.

2.2.4.1.2.2 Add new tags

Common tags are created by writing into any of the All Signals tabs (full or simple).
Required information for a signal is SignalName and signal type (analog/digital).

Before adding new tags, the "Freeze filters and sorting" button should be in the active state.
Otherwise newly added rows will be resorted and possibly be hidden because of an active filter.

To verify that recording for the new signals works correctly please use the ServiceMonitor tool.

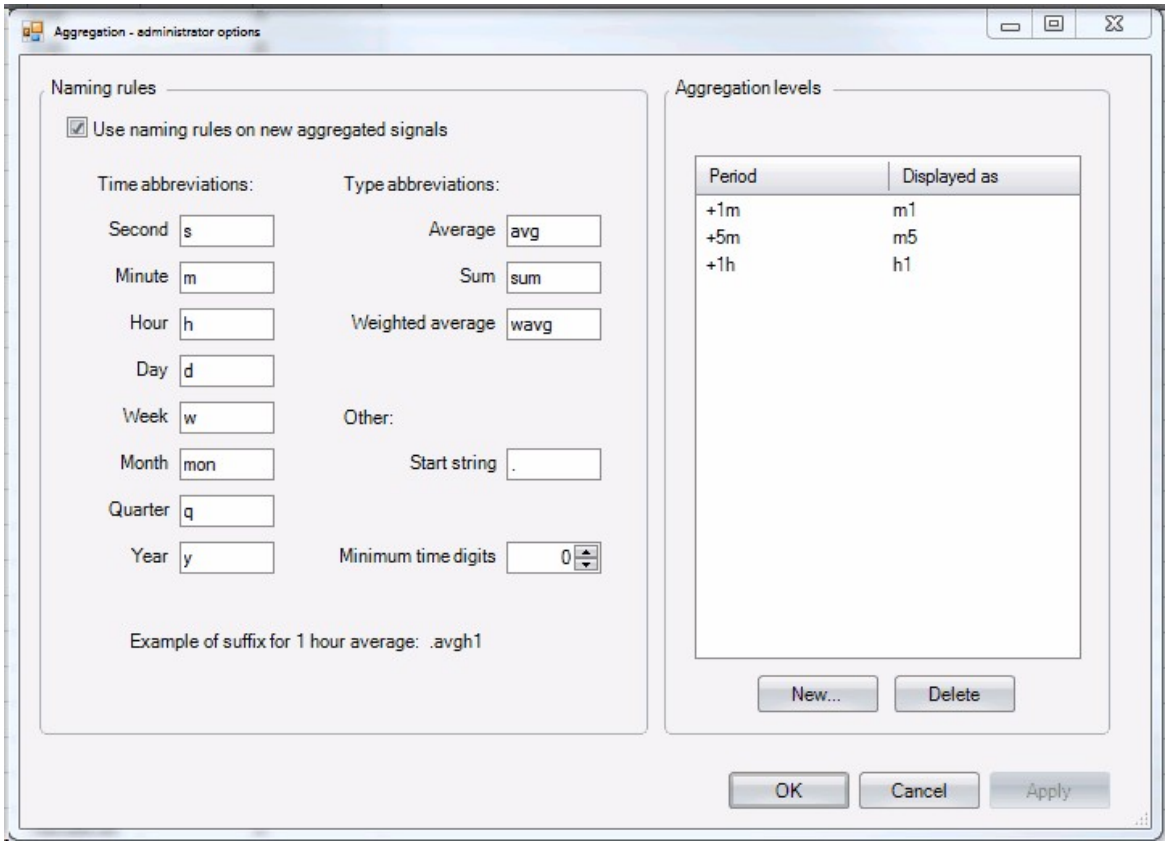
2.2.4.1.2.3 Add aggregation

Aggregations can be created in Simple, Full or Aggregation views.

Aggregations are created by right-clicking on any tag and select aggregate. Appropriate options will be displayed according to level and naming rules.

The naming rules are used to easier name all aggregations equally.
The name for a new aggregation will be suggested accordingly to the specified rules, raw signal name + aggregation suffix.

The aggregation levels are used to configure which levels of aggregations will be allowed in the system.
The configured levels will be available in the context menu when creating a new aggregation.



2.2.4.1.2.4 Add empty periodic tags

Empty periodic tags are created by right-clicking anywhere in the Simple or Full view.

Empty periodic tags does not have any automatic data creation. Instead they are used as target tags from periodic python calculation or imports.

2.2.4.2 Python Editor

The Python editor is a tool used to create calculated tags in the database.

2.2.4.2.1 Python Editor instructions

2.2.4.2.1.1 Create a calculated tag

1. Open the Python editor in AutArch Workspace.
2. Enter the calculation formula.
3. Enter time or tag triggers.
   - Tag triggers will start the calculation every time the trigger tag receives a new value.
   - Time triggers will start the calculation at selected intervals.
4. Test the calculation.
   - The test tool lets the user know if there are any errors in the calculation.
5. Press "Save as" to save the calculated tag to the database.

2.2.4.2.1.2 Recalculate

Normally the calculation starts to write values to the database from the time when it is saved to the database.

To use the calculation on older values the Recalculate option can be used.
1. Open the calculation in the python editor.

2. Set desired start time and span
3. Press Recalculate.

2.2.4.2.1.3 Global variables

Variables are normally treated as local variables that only exists in the calculation where they are used.

Global variables are shared with all calculations.
To make a variable global type: global VariableName

There is one predefined global variable, its name is 'dto' (short for data transfer object). It contains some data transferred from the hosting .NET-scope that are useful for the internal workings of the python AutArch implementation. Some variables can be useful for a user writing code too:

dto.time	Contains the current calculation evaluation time, as an AADateTime-object.
dto.StartTime	Contains the start time of the subsequent result from this calculation, as an AADateTime -object. In case of a non-periodic signal, this is exactly the same time as 'dto.time'. In case of a periodic signal, this contains the start time of the period.

2.2.4.2.2 Functions, Properties & Operators_2

2.2.4.2.2.1 Properties for T() objects_2

Properties that returns the type value point (and can therefore be chained):

T(" ")	Retrieves a representation of time, value and quality (referred to as a 'value point') for the selected tag.
T(" ").LastValueBeforeNow	Retrieves the last value point before the current evaluation time. This must be used in self-referential calculations to get the same result during a recalculation as when calculated the first time.
T(" ").Prev	Retrieves the previous value point.
T(" ").Next	Retrieves the next value point.
T(" ").LastPositiveEdgeFromNow	Returns the value point where the last positive edge occured (could be now)
T(" ").LastPositiveEdgeBeforeNow	Returns the value point where the last positive edge occured (before now)
T(" ").LastNegativeEdgeFromNow	Returns the value point where the last negative edge occured (could be now)
T(" ").LastNegativeEdgeBeforeNow	Returns the value point where the last negative edge occured (before now)

Properties that returns bool, decimal or AADateTime:

T(" ").Value	Only retrieves the tags last value, stripped of its quality. This is a decimal type.
T(" ").Time	Only retrieves the time from the tags last value point. This is a type called AADateTime.
T(" ").Bool	Returns True if the tags last value is not equal to 0.
T(" ").IsGood	Returns True if the last value point has a good quality.
T(" ").HasData	Returns True if the tags last value point has a valid value.
T(" ").IsEdge	Returns True if the retrieved value represents a change in value from the value before.
T(" ").IsNegativeEdge	Returns True if the retrieved value represents a decrease in value from the value before.

T(" ").IsPositiveEdge	Returns True if the retrieved value represents an increase in value from the value before.

References:
AADateTime

2.2.4.2.2.2 Functions for T() objects_2

T(" ").At(string) retrieves a (possibly interpolated) value point at specified time.

The string can be entered as either an absolute time or a relative time.

Example absolute time: "2014-10-16 16:00:00.000", "2014-10-16 16:00:00" or "2014-10-16 16:00".

Relative time can be entered as "s" (second), "m" (minute), "h" (hour), "d" (day) or "M" (month). A combination of the above is also possible. Ex. "-1h30m" will return the value at 1 hour and 30 minutes before the system time.

2.2.4.2.2.3 Time functions_2

BeginningOfYear()	Returns an AADateTime that is the current local time truncated to the nearest whole year back.
BeginningOfMonth()	Returns an AADateTime that is the current local time truncated to the nearest whole month back.
BeginningOfWeek()	Returns an AADateTime that is the current local time truncated to the nearest whole week back.
BeginningOfDay()	Returns an AADateTime that is the current local time truncated to the nearest whole day back.
BeginningOfHour()	Returns an AADateTime that is the current local time truncated to the nearest whole hour back.
BeginningOfMinute()	Returns an AADateTime that is the current local time truncated to the nearest whole minute back.
BeginningOfSecond()	Returns an AADateTime that is the current local time truncated to the nearest whole second back.
BeginningOfYear(AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole year back.
BeginningOfMonth(AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole month back.
BeginningOfWeek(AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole week back.
BeginningOfDay(AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole day back.
BeginningOfHour(AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole hour back.
BeginningOfMinute (AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole minute back.
BeginningOfSecond (AADateTime)	Returns an AADateTime of the entered time truncated to the nearest whole second back.
IsNewYear()	Returns True if it has become a new year since the last value was written.
IsNewMonth()	Returns True if it has become a new month since the last value was written.
IsNewWeek()	Returns True if it has become a new week since the last value was written.
IsNewDay()	Returns True if it has become a new day since the last value was written.
IsNewHour()	Returns True if it has become a new hour since the last value was written.
IsNewMinute()	Returns True if it has become a new minute

since the last value was written.

References:
AADateTime

2.2.4.2.2.4 Arithmetic operators_2

The arithmetic operations below can be performed with either; two value points, one value point and one decimal, or two decimals.
When performed with at least one value point the result will be a new value point with a quality that is a combination of the ingoing value points.

+	Addition	
-	Subtraction	
*	Multiplication	
/	Division	
**	Exponent	Performs exponential (power) calculation on operators, ex 10**3 = 1000

The arithmetic operations below can only be performed with two decimal values:

%	Modulus	Division that returns the remainder, ex. 10%3 = 1
//	Floor Division	Division without decimals, ex 10//3 = 3

2.2.4.2.2.5 Available python modules_2

The python modules that are preinstalled are:
- BaseHTTPServer
- Bastion
- CGIHTTPServer
- ConfigParser
- Cookie
- DocXMLRPCServer
- HTMLParser
- MimeWriter
- Queue
- SimpleHTTPServer
- SimpleXMLRPCServer
- SocketServer
- StringIO
- UserDict
- UserList
- UserString
- _LWPCookieJar
- _MozillaCookieJar
- __future__
- _abcoll
- _pyio
- _strptime
- _threading_local
- _weakrefset
- abc
- aifc
- antigravity
- anydbm
- argparse
- ast
- asynchat
- asyncore
- atexit

- audiodev
- base64
- bdb
- binhex
- bisect
- calendar
- cgi
- cgitb
- chunk
- cmd
- code
- codecs
- codeop
- collections
- colorsys
- commands
- compileall
- contextlib
- cookielib
- copy
- csv
- ctypes
- decimal
- difflib
- dircache
- dis
- distutils
- doctest
- dumbdbm
- dummy_thread
- dummy_threading
- email
- encodings
- ensurepip
- filecmp
- fileinput
- fnmatch
- formatter
- fpformat
- fractions
- ftplib
- functools
- genericpath
- getopt
- getpass
- gettext
- glob
- gzip
- hashlib
- header
- heapq
- hmac
- htmlentitydefs
- htmllib
- httplib
- ihooks
- imaplib

- imghdr
- importlib
- imputil
- inspect
- io
- json
- keyword
- lib2to3
- linecache
- locale
- logging
- macpath
- macurl2path
- mailbox
- mailcap
- markupbase
- md5
- mhlib
- mimetools
- mimetypes
- mimify
- modulefinder
- multifile
- multiprocessing
- mutex
- netrc
- new
- nntplib
- ntpath
- nturl2path
- numbers
- opcode
- optparse
- os
- os2emxpath
- pdb
- pickle
- pickletools
- pipes
- pkgutil
- platform
- plistlib
- popen2
- poplib
- posixfile
- posixpath
- pprint
- profile
- pstats
- py_compile
- pyclbr
- pydoc
- pydoc_data
- quopri
- random
- repr
- rexec

- rfc822
- rlcompleter
- robotparser
- runpy
- sched
- sets
- sgmllib
- sha
- shelve
- shlex
- shutil
- site
- smtpd
- smtplib
- sndhdr
- sqlite3
- sre_compile
- sre_constants
- sre_parse
- ssl
- stat
- statvfs
- string
- stringold
- stringprep
- struct
- subprocess
- sunau
- sunaudio
- symbol
- sysconfig
- tabnanny
- tarfile
- telnetlib
- tempfile
- textwrap
- this
- threading
- timeit
- toaiff
- token
- tokenize
- trace
- traceback
- types
- unittest
- urllib
- urllib2
- urlparse
- user
- uu
- uuid
- warnings
- wave
- weakref
- webbrowser
- whichdb

- wsgiref
- xdrlib
- xml
- xmllib
- xmlrpclib
- zipfile

2.2.4.2.2.6 AADateTime_2

All dates and times from the AutArch system available to the python environment are represented as instances of an object named 'AADateTime'.
The object contains two properties; named 'UTC' and 'Local' which contains the date and time as a .NET DateTime object, represented in UTC time zone and the local time zone.

2.2.4.2.3 Python example code

Enter topic text here.

2.2.4.2.3.1 Accumulator

Given a signal we want to accumulate, here named "X", we can create a new calculation named "XAcc" with the following code:

```
result = T("XAcc").LastValueBeforeNow.Value + T("X")
```

Note the use of 'LastValueBeforeNow'. Since this calculation is self-referencing it has to use 'LastValueBeforeNow' in order to produce the same result during a recalculation as in real time. Also note the use of '.Value'. Since this calculation depends partly on the previous result, we don't want the quality of that result to affect the current calculation.

Set the calculation to 'Real time' and to trigger on the signal "X" by dragging the signal "X" from the tag selector to the python triggers area.

2.2.4.2.3.2 Simulated sinus

To create a calculation that outputs a simulated sinus signal with amplitude 10 and a 1 minute periodicity; create a calculation, set it to trigger periodically (for example with a period of 2 seconds), and use this code:

```
result = 10.0 * sin(dto.time.Local.Second * 6.28 / 60.0)
```

References:
dto.time

2.2.4.3 Service monitor

The service monitor is a tools used to easily monitor all services in the AutArch system.

Most administration of the services can be done by simply right-clicking any services.
For example:
- Start/Stop the services
- See logs for the service
- Edit the services configuration
- Update service

2.2.4.3.1 Operations

2.2.4.3.1.1 Redundancy wizard

To use the redundancy wizard first install 2 or 4 RecordingNodes (install 4 nodes if separated adapter/brokers are necessary). Skip configuring NodeName, BrokerName1&2 and BrokerHost1&2.

Start the redundancy wizard, select the services and enter requested configuration.

2.2.4.3.1.2 Global update

The Global update wizard is started from the Service monitor plugin in AutArch Workspace.
It is always recommended to use the latest version of AutArch Workpace.
To update all services all the computers where any service is installed needs to be listed and visible in the Service monitor.

The Global update is used to update:
- Database
- Services
- Report binaries
- Report StandardTemplate (and configuration)

2.2.4.3.1.3 Installing services

AutArch contains a number of different services.
- RecordingNode
- CalculationAgent
- ReportGenerator
- ServiceWatcher
- RestProvider
- ChaosMonkey

Before installation make sure that SupervisionServer is installed on all computers where any service will be installed.
Start Workspace and open the tool ServiceMonitor.
Add the computers where services will be installed by right-clicking in the white area and click "Add new computer..", enter the computer name or IP-address.

To install a service right-click on the SupervisionServer for the requested computer and select "New -> AutArch Service" and enter necessary configuration in the following dialogs.

2.2.4.3.1.4 Verifying status of services

All services are displaying their current statuses.
If a service is experiencing an error a red cross is displayed in the Service monitor and the latest error text is displayed in the Error cell. Too see all active errors, right click on the service and select "Get all errors".

To investigate an error even further, the best way is to read the log files.
Right click on the service and select "View latest log" or "Get logs" to retrieve all log files.

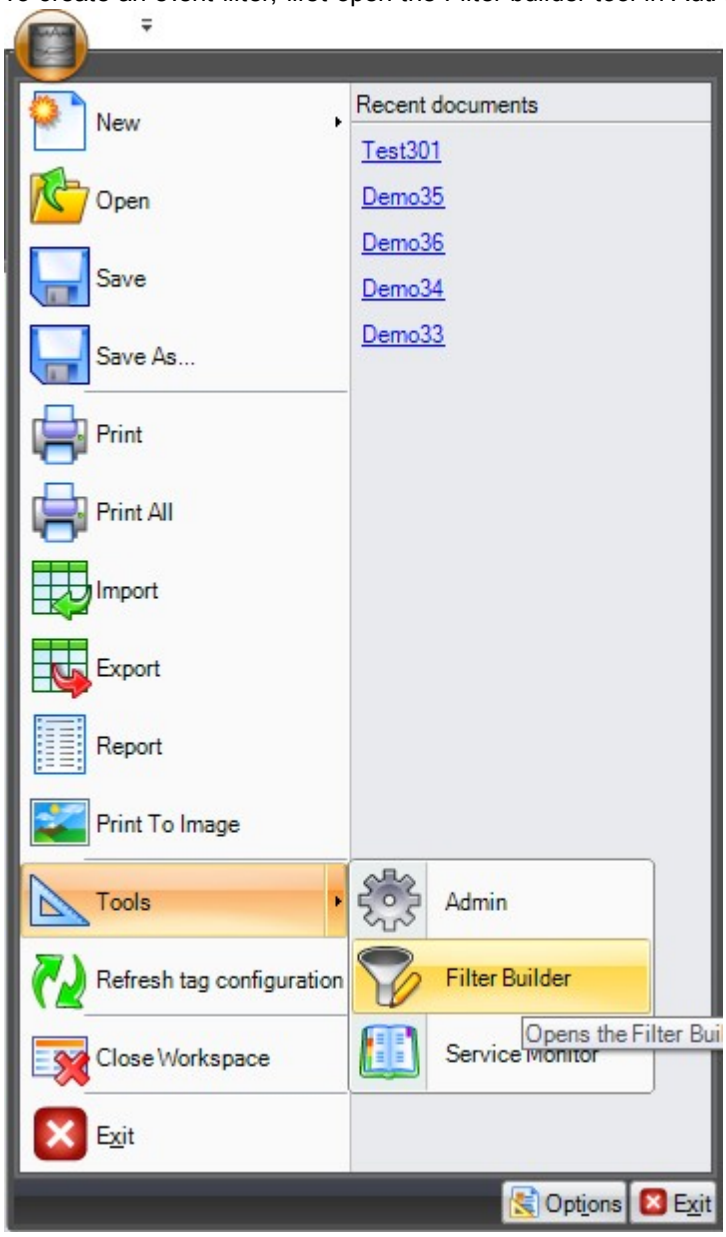
The details view is available for any Recording node service running as an adapter.

The details view is used to investigate the subscription status for all tags.
The view displays all tags handled by the adapter as well as their last values, last qualities and last timestamps.
If any tag does not have a last value it probably indicates that there is something wrong with the subscription of that tag.

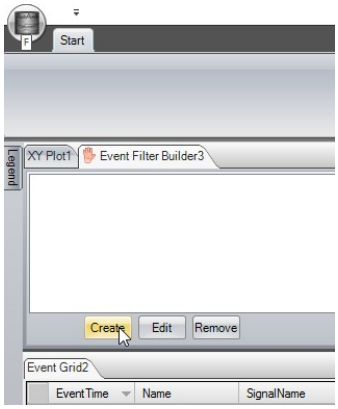
Recording nodes usually updates their tag configuration once per minute.
Tags added during runtime does not automatically show up in the details view. The view needs to be closed and reopened.

2.2.4.4 Event filter builder

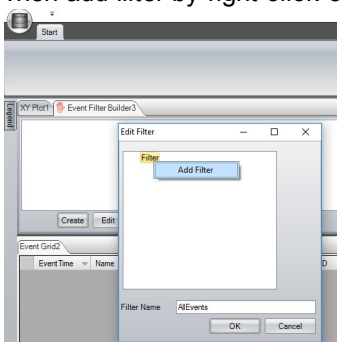
The filter builder tool is used to create filters for the event grid.
To create an event filter, first open the Filter builder tool in AutArch Workspace.



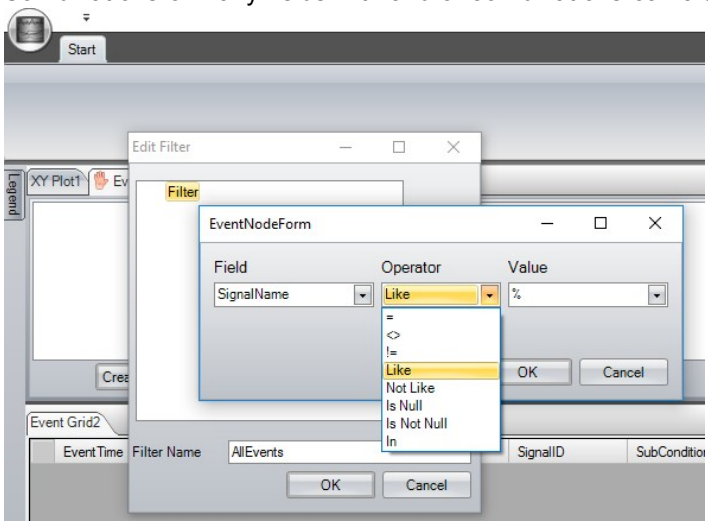
At the bottom left corner of the tool press the Create button.



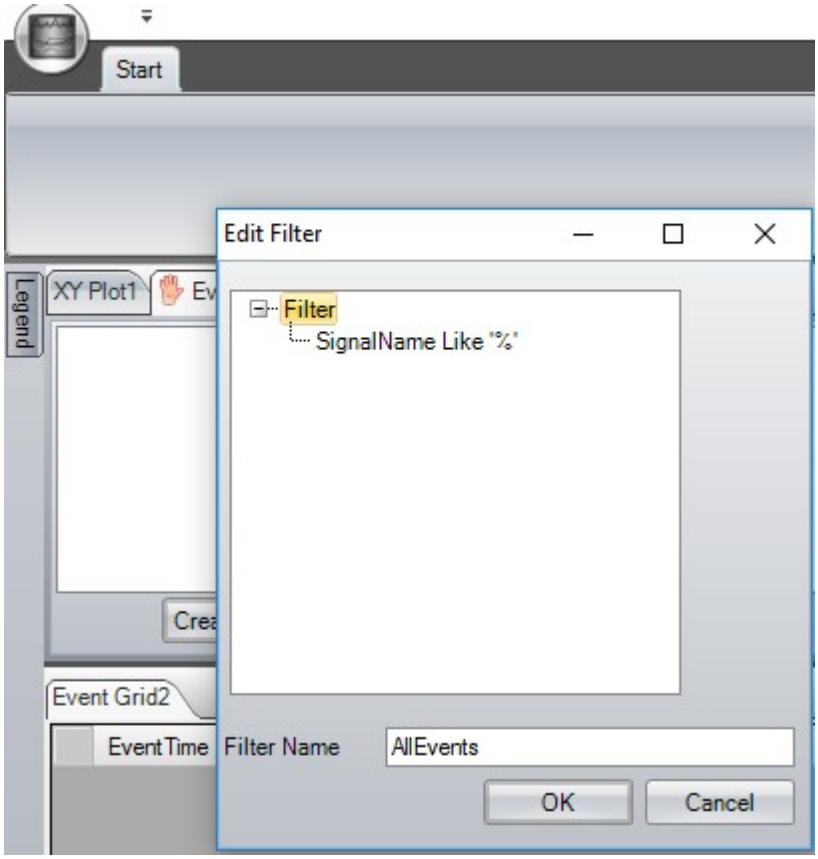
Name the filter with appropriate name. In the "Filter Name" text box. (In this sample "AllEvents")
Then add filter by right click on Filter in tree.



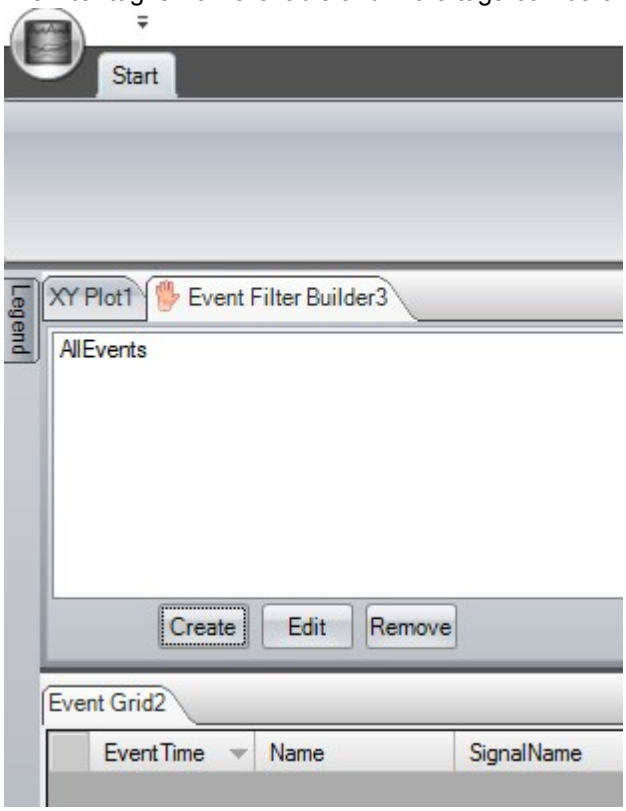
Any field can be used to set an operator and value in combination to create required filter.
Combinations of many fields with and/or combinations can also be used.



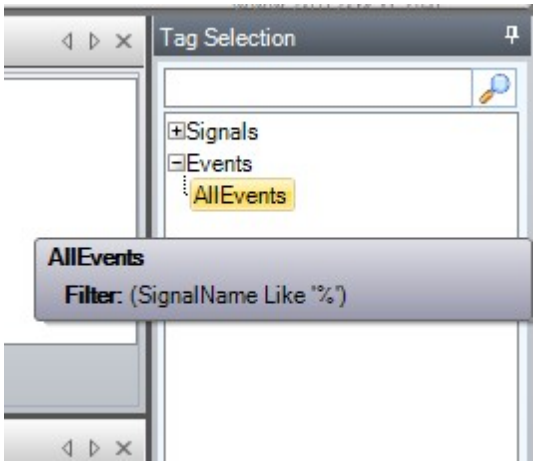
When finished, press ok to store the filter as a filter tag.



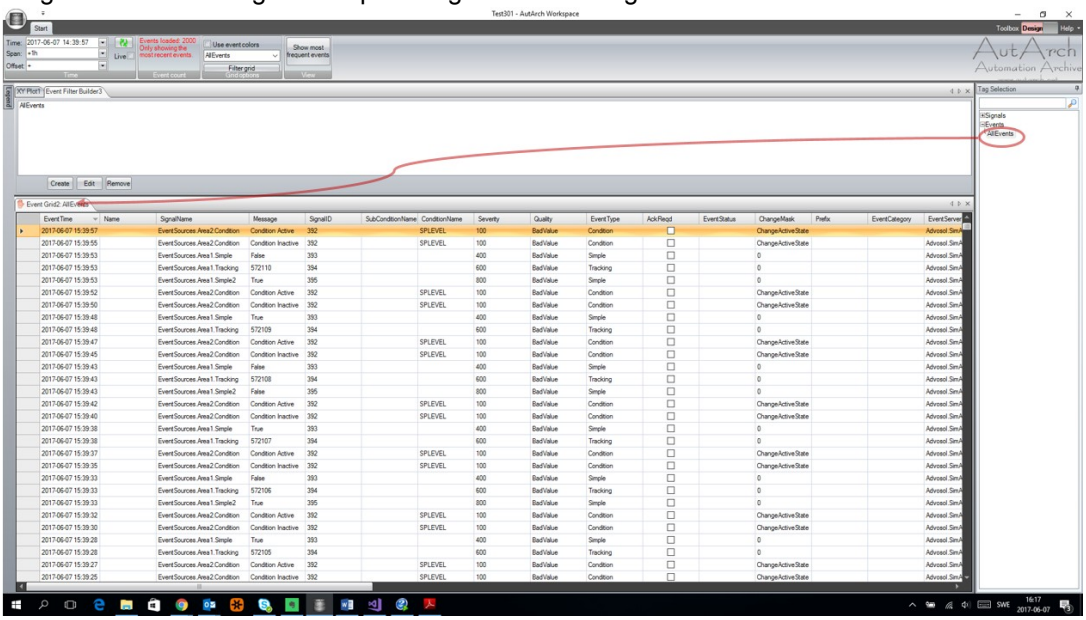
The filter tag is now available and more tags can be created, edited or removed in the filter builder.



When a filter tag is created, it will appear in Tag Selection tree under Events

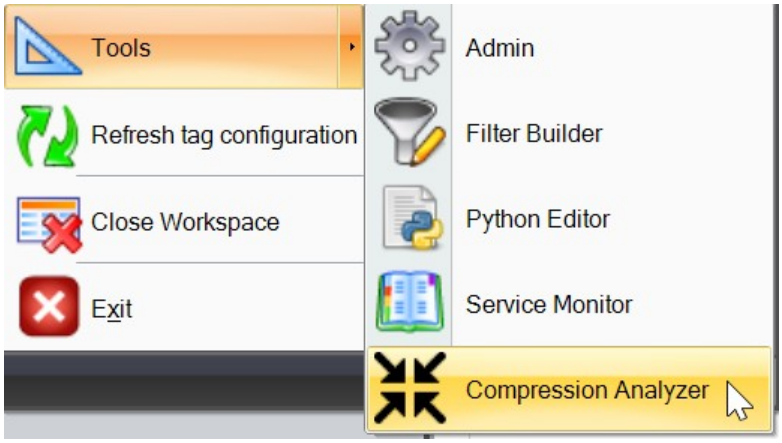


Drag the event filter tag and drop the tag into an event grid.



2.2.4.5 Compression analyzer

The compression analyzer tool is used to manually configure and validate compression settings. The tool consists of a grid with all tags in the connected database and a trend where the user can see the result of new compression settings compared to the current settings.



Find the Compression Analyzer in the Tools submenu.

2.2.4.5.1 Compression analyzer overview

Description of the different areas of the user interface:

Time period settings

Sets the time period to look at in the database. Note that a value too large in the "Span" setting will have a negative effect on the performance of this tool. Normally a setting of at most a couple of days is sufficient and will not affect database performance notably.

Quickly hide irrelevant tags

Quickly hide tags that cannot (or should not) use compression. The different tags that quickly can be shown/hidden are:

- AutArch diagnostic tags - internal system tags that should not be compressed
- Digital tags - cannot be compressed
- Periodic tags - cannot be compressed
- Calculated tags - should most often not be compressed
- Already approved tags - quickly hide tags that already have had their compression fine tuned and manually approved

Automatic calculation of compression settings

Enter a desired compression rate and have the computer automatically suggest a compression setting for each of the current tags in the grid.

Get info about archived values

For each of the current tags in the grid, the number of archived values is fetched for the requested time period and a theoretical compression rate is calculated.

Save button

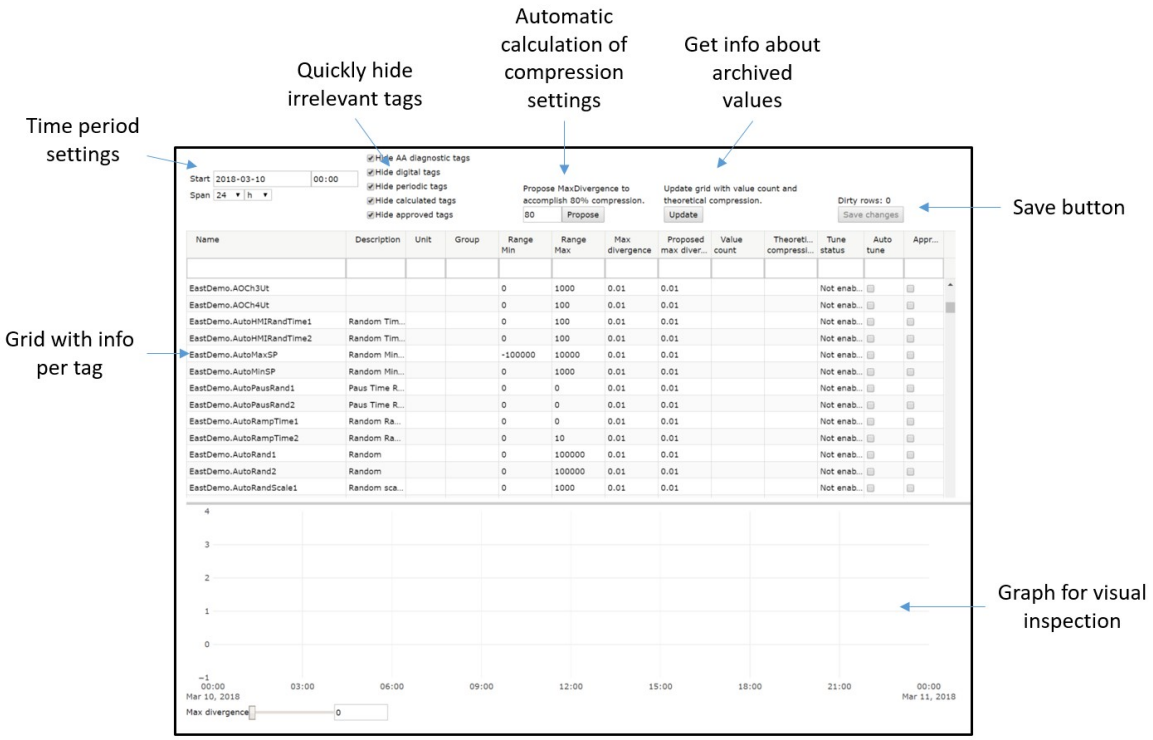
Saves all proposed max divergences, autotune changes and approved changes that are marked as "dirty". Dirty cells in the grid will have a different background color.

Grid with info per tag

A grid showing all tags from the workspace's currently selected database, except the tags checked in the quick hide area. The grid supports text filtering in multiple columns, and sorting in all text or numerical columns.
The user editable columns are: "Proposed max divergence", "Auto tune" and "Approved".
If a row is double clicked then a visual graph will appear in the graph area for that tag and the selected time period.

Graph for visual inspection

By double clicking a tag in the main grid, the value graph for the selected time period will appear in this area. The graph of archived values is overlayed by another graph showing how the values will be affected by the currently proposed max divergence. The proposed max divergence can be directly changed by dragging the "Max divergence"-slider in this area, or by typing in a value by hand in the numerical box beside it, or by using the stepping arrows in the same numerical box.



2.2.4.5.2 Main grid overview

Name	Description	Unit	Group	Range min	Range max	Max divergence	Proposed max divergence	Value count	Theoretical compression	Tune status	Auto tune	Appr-oved
eastdemo2.*ge*pre												
EastDemo2.GE01_Prediction	Machine lea...		EastDemo...	0	100	0.01	0.01			Not enab...	☐	☐
EastDemo2.GE01_PredictionMax	Machine lea...		EastDemo...	0	100	0.5	0.5			Not enab...	☐	☐
EastDemo2.GE01_PredictionMin	Machine lea...		EastDemo...	0	100	0.5	0.5			Not enab...	☐	☐
EastDemo2.GE01_PredMaxDiff	Machine lea...		EastDemo...	0	100	1	1			Not enab...	☐	☐
EastDemo2.GE01_PredMinDiff	Machine lea...		EastDemo...	0	100	1	1			Not enab...	☐	☐

Columns

Name	The tag name. See SignalName
Description	The description text of the tag. See Detailed description
Unit	The unit of the tag.
Group	The data collection group the tag belongs to. See OPCGroupName
Range min	The minimum value of the tag's measuring range.
Range max	The maximum value of the tag's measuring range.
Max	The current Max divergence as configured in the database.

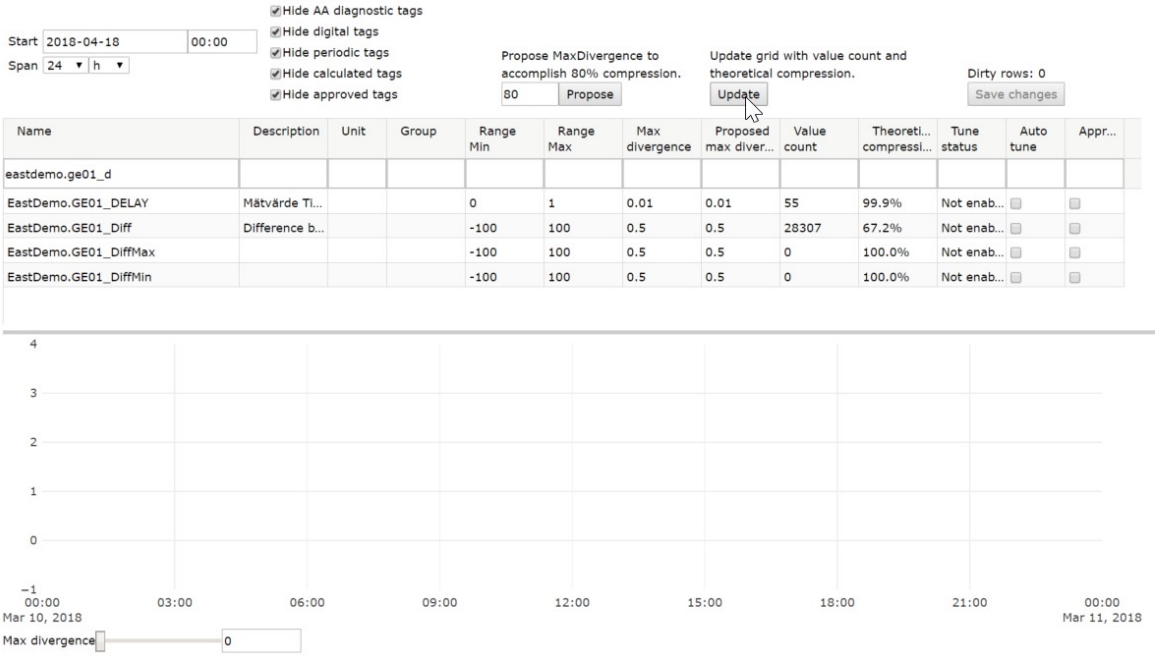
divergence	
Proposed max divergence	The max divergence that the compression analyzer tool proposes. This value can be automatically proposed by clicking the Propose-button above the grid. Or the value can be adjusted manually by bringing the tag up for inspection in the graph area, by double clicking the tag in the grid.
Value count	When the update-button has been clicked, or the tag is brought up in the graph area - then this column counts how many archived values there are in the database for the selected time period. But when "proposed max divergence" is changed, then this column shows how many values there would have been in the database for the selected time period if the proposed max divergence had been used instead.
Theoretical compression	A calculated value that shows the theoretical compression rate at the selected max divergence. 0% means no compression at all - all values are saved to the archive. 100% means full compression - almost no values are saved to the archive.
Tune status	A column displaying the current CompTune status. For example: "Not enabled", "Waiting for reference condition", "Collecting data", "Analyzing", "Finished", "Finished & checked" or "Invalid parameters".
Auto tune	Check this checkbox to set a tag in auto tune mode. The tag will then be automatically tuned by CompTune
Approved	Check this checkbox to set a tag's status as "compression rate is manually approved". All manually approved tags can be hidden from the grid using the quick filters.

Filtering

All columns except "Tune status", "Auto tune" and "Approved" can be filtered. The filter is a combination of wildcards and regular expressions.
* matches any tokens. For example; "a*c" matches "ac", "abc" and "a12345c123", but not "ab".
? matches 0 or 1 token. For exmaple; "a?c" matches "ac" and "abc", but not "abbc".

2.2.4.5.3 Identifying tags in need of review

1. Filter out a subset of tags by writing in the filter row of the grid. See an example in the picture below.
2. Select a suitable time period.
3. Press the Update button to fetch the number of values that has been stored during the selected timespan. See the picture below.
4. Sort the list on the "Theoretical compression" column.
   (Theoretical compression describes, in precent, how many values that have been compressed compared to the theoretical max number of values based on the signals sampling rate)
5. Signals with a very low theoretical compression is probably collecting too much data and should be reviewed manually.
6. Signals with a very high theoretical compression might be compressing to much data.
   Since there is no way to test a lower compression setting on signals that already have been compressed the best way to handle these signals is to manually set the compression lower and then validate them again at a later time.

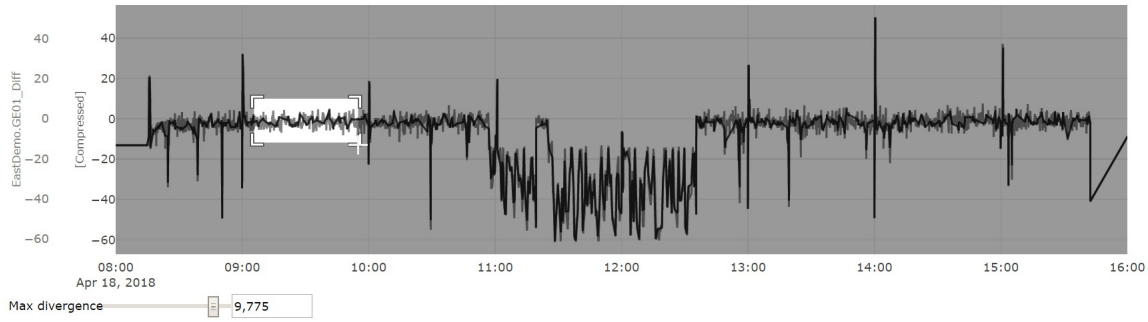


2.2.4.5.4 Automatically propose a new MaxDivergence

1. Filter out a subset of tags by writing in the filter row of the grid. See an example in the picture below.
2. Select a suitable timespan.
3. Enter the requested compression rate, in percent.
   The requested compression rate specifies how much values should be removed.
4. Click on the Propose button. See the picture below.
5. The tool will propose a new MaxDivergence. This only applies to tags with a theoretical compression below the requested compression rate.
6. Save the new MaxDivergence by clicking on the "Save changes" button.

2.2.4.5.5 Manually test and configure new compression settings

1. Select a suitable timespan.
2. Double-click on any tag in the grid.
3. A time trend will open displaying the current values for the selected signal as well as an overlayed graph for the values that would be kept with the new compression settings.
4. The trend is zoomable by selecting different areas with the mouse. See the picture below. The zoom is reset by double clicking anywhere in the trend area.
5. Enter a new MaxDivergence by moving the slider at the bottom of the trend, or by editing the numerical box directly.
6. Save the new MaxDivergence by clicking on the "Save changes" button.



2.2.5 Import/Export data

The Import and Export tool in AutArch workspace is used to manually import and export signal data and events from the database.
Data and events can be exported either as encrypted .XMZ files or Excel .xls files.

Excel files can be exported with either:
- Archived values - Exports the actual values stored in the database (with optional interpolated values at the start-/endtime of the export)
- Interpolated values - Interpolates values at chosen interval.

2.2.6 Report admin tool

The report tool in AutArch workspace is used to administrate reports.
The tool can be used to:
- Create new report templates.
- Schedule reports.
- Run reports manually.
- Open generated reports.

The actual configuration of the report templates are made in Excel and require the Report Excel plugin to be installed.
The scheduled generation of reports are performed by a Report generator.

Reports created by the Report generator are stored in the database.
Manually generated reports are only stored as a temporary file on the computer and the user can then choose to save it to a more permanent location.

2.2.7 Install instruction

2.2.7.1 Prerequisites

- DotNet Framework: 3.5 and 4.0
- Disk space: minimum 300MB
- RAM: minimum 2GB
- Processor speed: minumum 2.0 GHz

- Operating sytem: Windows XP, Windows Server 2003 or newer

2.2.7.2 Installation

AutArch Workspace can be distributed with either a ClickOnce package or a single setup.

Clients installed with the ClickOnce package is updated automatically when a new version is available on the shared drive.
All clients installed with the single setup has to be updated manually.

2.2.7.2.1 ClickOnce

AutArch Workspace can be distributed with ClickOnce technology which means that the Workspace only needs to be installed once, updates are then installed automatically.

To install AutArch Workspace put the entire ClickOnce folder on a shared drive that is accessible from all planned clients.
Start the installation by running "Setup.exe" on each client.

The first client that installs the Workspace will be asked to enter connection setting to the database. These settings are then shared with all other clients.

2.2.7.2.2 Single setup

Install AutArch Workspace by starting "AutArchWorkspaceSetup.exe" and follow the steps in the wizard.
The first time AutArch Workspace is started the user will be asked to enter connection settings to the database.

2.2.7.3 Update

2.2.7.3.1 ClickOnce

Copy the new ClickOnce package to the share from which the workspace is installed, overwrite any existing files.
All clients will automatically get updated the next time they are started.

2.2.7.3.2 Single setup

Update AutArch Workspace by starting the new version of "AutArchWorkspaceSetup.exe" and follow the steps in the wizard.

2.3 SupervisionServer

2.3.1 Overview

The SupervisionServer supervises all other AutArch services and need to be installed on every computer where other services are planned to be used.

2.3.2 Install instructions

2.3.2.1 Prerequisites

- DotNet Framework: 3.5 and 4.0
- Hard-disk space: minimum 300MB
- RAM: minimum 2GB
- Processor speed: minumum 2.0 GHz
- Operating sytem: Windows XP, Windows Server 2003 or newer

2.3.2.2 Installation

To install the SupervisionServer start the "AutArch SupervisionServer Setup.exe" and follow the steps in the wizard.

2.3.2.3 Update

The Supervision server is updated by running Global update from Service monitor.

2.3.2.3.1 Update version prior to 3.0

Versions prior to 3.0 cannot be updated with Global update.
These have to be updated manually by running the "AutArch SupervisionServer Setup.exe" locally on each computer.

2.4 Services

2.4.1 Overview

There are multiple services available in AutArch.
Each of the will be individually described below.

AutArch Services are installed as a "CentralService" to make them easier to configure and maintain.
A CentralService can be used to host multiple services.

2.4.2 Services

2.4.2.1 RecordingNode

Recording nodes are used for collecting data and events from different systems.
Recording nodes is basically divided into two separated entities, adapters and brokers.

Commonly the brokers run in the same process as adapters but they can also run stand-alone if necessary.

2.4.2.1.1 Adapter

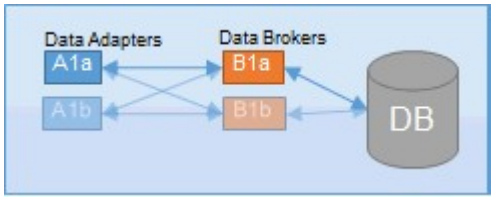
Adapters are the interface between the server (provider of data) and the AutArch system.
Adapters receives or polls data from the server, transforms it to an AutArch valuepoint and passes it on to a broker.

AutArch contains multiple different adapters for collecting from different systems.
- OPCDA
- OPCAE
- OPCUA
- Siemens S7 PLC

2.4.2.1.2 Broker

Brokers have several responsibilities.
- Fetching configuration for adapters
- Handling redundancy
- Compressing data
- Storing data to the database

2.4.2.1.3 Redundancy



Basics of the redundancy for recording nodes:
- Brokers communicate with each other to decide who is master/slave
- The slave broker works in the same way as the master except that it does not store any data to the database. Instead it keeps the data until it has made sure that the master broker has successfully saved it.
- Both adapters sends their data to both of the brokers. The brokers decides which of the adapters should be considered as master.
- Master/slave switches can occur without any noticeable delay in the data-flow.

2.4.2.1.4 Buffer

Both adapters and brokers buffer their data in case of connections issues.

All values are put in a buffer, the buffer primary stores values in the memory.
In case the memory gets full (i.e. when an adapter cannot send the values to the broker or the broker cannot store the values to the database) the values are written to local files.

All values are always handled in order of time, the oldest values are always sent first.

2.4.2.2 CalculationAgent

Calculation agents are used to automatically execute python calculations stored in the systems.

When saving a calculation in the system it is automatically assigned to a responsible depending of it's input tags.
The Calculation agent can be configured to handle one or multiple responsibilities.

2.4.2.3 ServiceWatcher

Service watcher is used to supervise that non-AutArch services.
The service watcher periodically inspects if configured services are running, if not it starts them.

The service watcher can also be used to inspect old AutArch OPCDA and OPCAE services (pre 2.0).

2.4.2.4 ReportGenerator

The report generator is used to automatically execute the scheduled reports in the system.
The reports are written in Excel and since it is not supported to run Excel from a service the report generator has to run as a service.

The generator opens the report and runs it according to it's configuration.
The result file is stored in the database and can be viewed in the report tool in AutArch Workspace.

Before the report generator can execute any report an Excel plugin needs to be installed. The user will be prompted for this the first time a report template is opened from the report tool or by the generator.

2.4.2.5 RestProvider

The REST provider is a service that responds to GET- and POST-calls over http on a specified port. The default port is 4716.
This component also contains a web server for dashboards, default port 4717.

2.4.2.5.1 API reference

All GET-methods are called in the following manner:

URL style:
http://hostaddress:4716/AAProvider/GetSerieValues?tagName={tagName}&startTime={startTime}&endTime={endTime}&nSamples={nSamples}

jQuery style:

```
$.getJSON('http://hostaddress:4716/AAProvider/GetSerieValues?callback=?', {
   tagName: tagName,
   startTime: startTime.getTime(),
   endTime: endTime.getTime(),
   nSamples: 500
}, function(data) {
});
```

All POST-methods are called in the following manner:

jQuery style

```
$.ajax({
   type: "POST",
   url: 'http://hostaddress:4716/AAProvider/SaveMultiValue',
   dataType: 'json',
   contentType: "application/json; charset=utf-8",
   data: JSON.stringify({
      tagName: "Name",
      multiValue: {
         String01: "Some string",
         DateTime01: "2018-05-21 08:00",
         Decimal01: 7.1
      }
   }),
   success: function () {
      console.log("Success");
   },
   error: function (jqXHR, exception) {
      console.log("Error " + jqXHR.status);
   }
});
```

2.4.2.5.1.1 Time formats

Given the following JavaScript date, in a timezone of GMT+1:
var d = new Date("2017-11-19 08:00");

It can be sent to the REST provider from JavaScript these ways:
d.toLocaleString("sv-SE")
 +- - 2017-11-19 08:00:00
d.toISOString()
 +- - 2017-11-19T07:00:00.000Z
d.toJSON()
 +- - 2017-11-19T07:00:00.000Z
d.getTime()

+- - 1511074800000

The REST provider also accepts these formats:
Logi UTC
+- - 2017-11-19T07:00:00.000
Logi UTC with offset
+- - 2017-11-19T06:00:00.000{+1h}

2.4.2.5.1.2 GET-methods

- `List<TagJson> GetTags();`
- `List<ValuePointJson> GetArchivedAndEnclosingValues(string tagName, string startTime, string endTime)`

Since v3.2.8
- `List<ValuePointJson> GetLogValues(string tagName, string startTime, string endTime)`

Since v3.3.0
- `List<ValuePointJson> GetSerieValues(string tagName, string startTime, string endTime, int nSamples);`
- `List<ValuePointJson> GetSerieValuesByString(string tagName, string startTime, string endTime, int nSamples);`
- `List<TagValueCountJson> GetValueCounts(string tagNames, string startTime, string endTime)`

Since v3.3.0
- `List<ComparisonValuePointJson> GetComparisonValues(string tagName, string startTime, string endTime, string filterOnValue);`
- `HttpStatusCode SetValue(string tagName, string startTime, string endTime, double dValue);`
- `HttpStatusCode SaveLogValue(string tagName, string time, double value, ushort quality);`
- `HttpStatusCode SetApplicationParameter(string application, string category, string name, string value);`
- `TextResponseJson GetApplicationParameter(string application, string category, string name);`
- `List<TextResponseJson> GetApplicationParameters(string application, string category);`
- `List<TextResponseJson> GetRolesForUser(string userName);`
- `List<SimpleJson> GetLogiRolesForUser(string userName);`
- `TimespanResponseJson GetTotalTime(string tagName, string startTime, string endTime, bool highbit);`
- `NumericResponseJson GetTotalChanges(string tagName, string startTime, string endTime);`
- `NumericResponseJson GetSum(string tagName, string startTime, string endTime);`
- `List<DigitalValuePointJson> GetInterpolatedDigitalValues(string tagName, string startTime, string endTime, string period);`
- `ValuePointJson GetLastValue(string tagName);`

- `List<ValuePointJson> GetLastValues(string tagNames);`
- `List<AnnotatedValuePointJson> GetTransactionLog(string tagName, string startTime);`
- `List<EventJson> GetEvents(string startTime, string endTime, string filter, int maxEventCount);`
- `List<TextResponseJson> GetTextAndValidIntervalFromMultiColumn (string tagName, string multiColumnName, string startTime, string endTime);`
- `List<MultiValueJson> GetMultiValues(string tagName, string startTime, string endTime);`
- `Stream GetExportFile(ExportParametersJson exportParams, string exportType);`

2.4.2.5.1.3 POST-methods

- `HttpStatusCode SaveMultiValue(string tagName, MultiValueJson multiValue);`
- `HttpStatusCode UpdateMultiValue(MultiValueJson multiValue);`
- `HttpStatusCode ImportXlsxFile(ImportInfoJson importInfo);`
- `ImportInfoJson ParseXlsxFileForImport(Stream fileStream);`

2.4.2.5.1.4 Data transfer classes

Enter topic text here.

```
{
	Name: string,
	SignalID: int,
	ValueUnit: string,
	SignalType: int,
	RangeMax: double,
	RangeMin: double,
	OPCGroupName: string,
	MaxDivergence: float,
	TuneStatus: int
}
```

```
{
	Value: double,
	Quality: ushort,
	Time: long,
	IsoTime: string
}
```

Shares members with ValuePointJson. Extra members are:

```
{
	Comment: string,
	ChangedIsoTime: string,
	ChangedBy: string
}
```

Shares members with ValuePointJson. Extra members are:

```
{
	NextValue: double,
	NextQuality: ushort,
	NextTime: double,
	NextIsoTime: string
}
```

Shares members with ValuePointJson. Extra members are:

```
{
	ActiveServiceRuntime: double,
	ActiveTotalRuntime: double,
	InactiveServiceRuntime: double,
	InactiveTotalRuntime: double,
	ServiceChanges: double,
	TotalChanges: double
}

{
	SignalID: int,
	MultiValueID: long,
	CreationTime: string,
	String01: string,
	String02: string,
	String03: string,
	String04: string,
	String05: string,
	String06: string,
	String07: string,
	String08: string,
	String09: string,
	String10: string,
	DateTime01: string,
	DateTime02: string,
	Decimal01: double,
	Decimal02: double,
	Decimal03: double,
	Decimal04: double
}

{
	SignalID: int,
	ValueCount: int
}

{
	Time: long,
	IsoTime: string,
	LastTime: long,
	LastIsoTime: string
}
```

Shares members with TimestampedResponseJson. Extra members are:

```
{
	Value: double
}
```

Shares members with TimestampedResponseJson. Extra members are:

```
{
	ValueText: string,
	Key: string
}
```

Shares members with TimestampedResponseJson. Extra members are:

```
{
	IsoTimeSpan: string,
	TimeSpan: string,
	TotalMilliseconds: double
}

{
	Value: string
}
```

```
{
        Name: string,
        DatabaseName: string,
        Category1: string,
        Custom01: string,
        Custom02: string,
        Custom03: string,
        Custom04: string,
        Custom05: string,
        EventID: int,
        ConditionName: string,
        Message: string,
        EventCategory: string,
        Severity: int,
        SubConditionName: string,
        AckReqd: bool,
        EventTime: long,
        EventActiveTime: long,
        Quality: ushort,
        EventStatus: string,
        ActorID: string,
        EventCategoryID: int,
        BackColor: string,
        ForeColor: string
}

{
        TagList: List<string>,
        Start: AADateTime,
        End: AADateTime,
        UseInterpolation: bool,
        InterpolationSpanMs: long,
        UseFraming: bool
}

{
        Key: string,
        ImportOffset: TimePeriod,
        Tags: List<ImportTagInfo>,
        Comment: string
}

{
        TagName: string,
        FirstValueTime: AADateTime,
        LastValueTime: AADateTime,
        DisableImport: bool
}
```

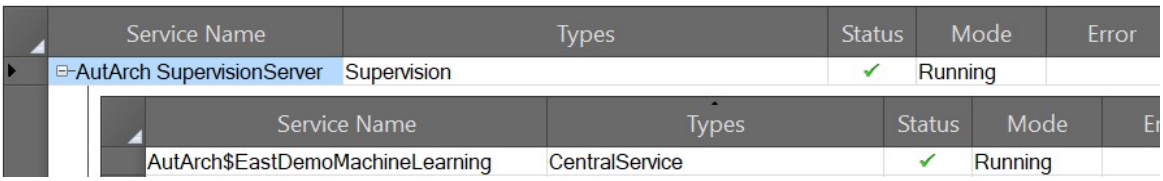
2.4.2.6 MachineLearning

2.4.2.6.1 Technical details

The Machine Learning consists of a Windows service that implements a feed-forward neural network. The inputs to the network can be selected among all the tags configured in the system. Real-time data as well as delayed data can be selected. So, for example, the current value of some signals plus their values 10 seconds ago can be fed to the network.
Any number of outputs from the network can be configured. Each output writes its data to an existing tag configured in the system, and can thus be further processed; for example real-time-filtered or generating alarms using the system's python script engine.

2.4.2.6.2 Runtime

The Autarch node performing the Machine Learning algorithms is installed and supervised by the Service Monitor tool in the AutArch Workspace application.
The picture below shows a typical view from the Service Monitor during runtime:



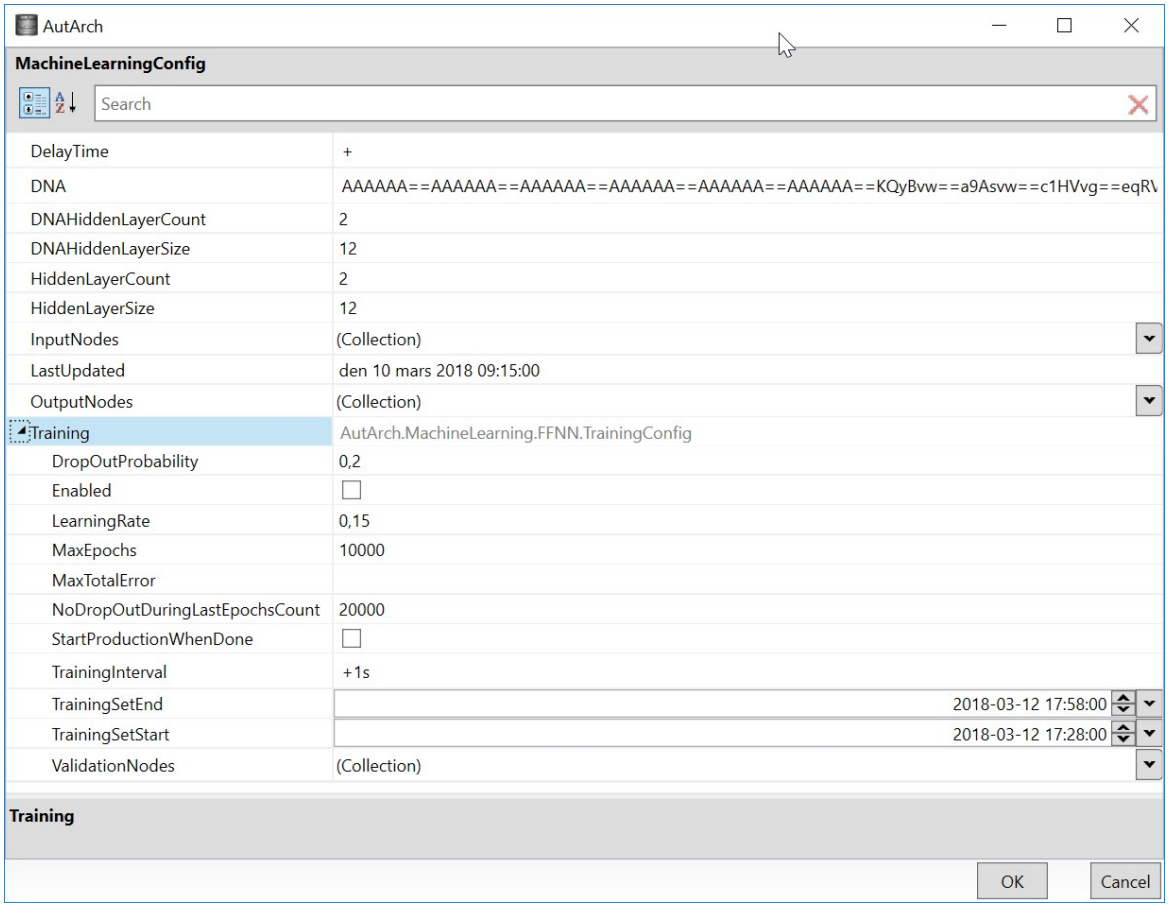
A typical view in the service monitor of a running machine learning node.

2.4.2.6.3 Training

The training is supervised from a simple user interface, where you also have the power to modify the behaviour of the training. The training can be set to stop from several conditions, or manually, and can manually be resumed at a later time.

The machine learning training monitor.

Before training starts, the network size and inputs and outputs must be configured. This can be done from the Service Monitor as well.



2.4.3 Install instruction

2.4.3.1 Prerequisites

- DotNet Framework: 3.5 and 4.0
- Hard-disk space: minimum 200MB
- RAM: minimum 2GB
- Processor speed: minumum 2.0 GHz
- Operating sytem: Windows XP, Windows Server 2003 or newer

2.4.3.2 Installation

Services are installed from the Service monitor tool in Workspace.

2.4.3.2.1 RecordingNode redundancy configuration

The RecordingNode redundancy configuration can be setup manually for each service or by using the redundancy wizard in ServiceMonitor.

2.4.3.3 Update

All services are updated by running Global update from Service monitor.

2.4.3.3.1 Update services version 1.x.x and 2.x.x

Updates for older services are not as automated.
Therefore more manual steps are required to create as little disturbance as possible..

1. Write down configuration of existing services.
2. Install new services (without starting them) according to the configuration from the old.
3. Start the new services and stop the old ones.

4. Uninstall the old services.

2.4.3.3.1.1 Configuration needed for recording services

- Database connection settings
- Redundancy settings
- Buffer directory
- OPCDAClientName/Responsible
- Adapter type

2.5 LiveExcel

2.5.1 Overview

LiveExcel is a plugin to Excel used to fetch data from an AutArch database.
The plugin is mainly used by using the "New copy as LiveExcel" feature from AutArch Workspace.

2.5.2 Install instructions

2.5.2.1 Prerequisites

- DotNet Framework: 3.5 and 4.0
- Hard-disk space: minimum 200MB
- RAM: minimum 2GB
- Processor speed: minumum 2.0 GHz
- Operating sytem: Windows XP, Windows Server 2003 or newer
- Excel 2010 or newer

2.5.2.2 Installation

AutArch LiveExcel is distributed with ClickOnce technology which means that the Excel plugin only needs to be installed once, updates are then installed automatically.

To install AutArch LiveExcel put the ClickOnce folder on a shared drive that is accessible from all planned clients.
Start the installation by running "Setup.exe".

2.5.2.3 Update

Copy the new ClickOnce package to the share from which the workspace is installed, overwrite any existing files.
All clients will automatically get updated the next time they are started.

2.6 Report

2.6.1 Overview

AutArch Reports are based on a template created in Excel.
The templates can then be generated at a schedule by a ReportGenerator service or manually from the report tool in AutArch Workspace.
The generation will output either an PDF or XLS file.

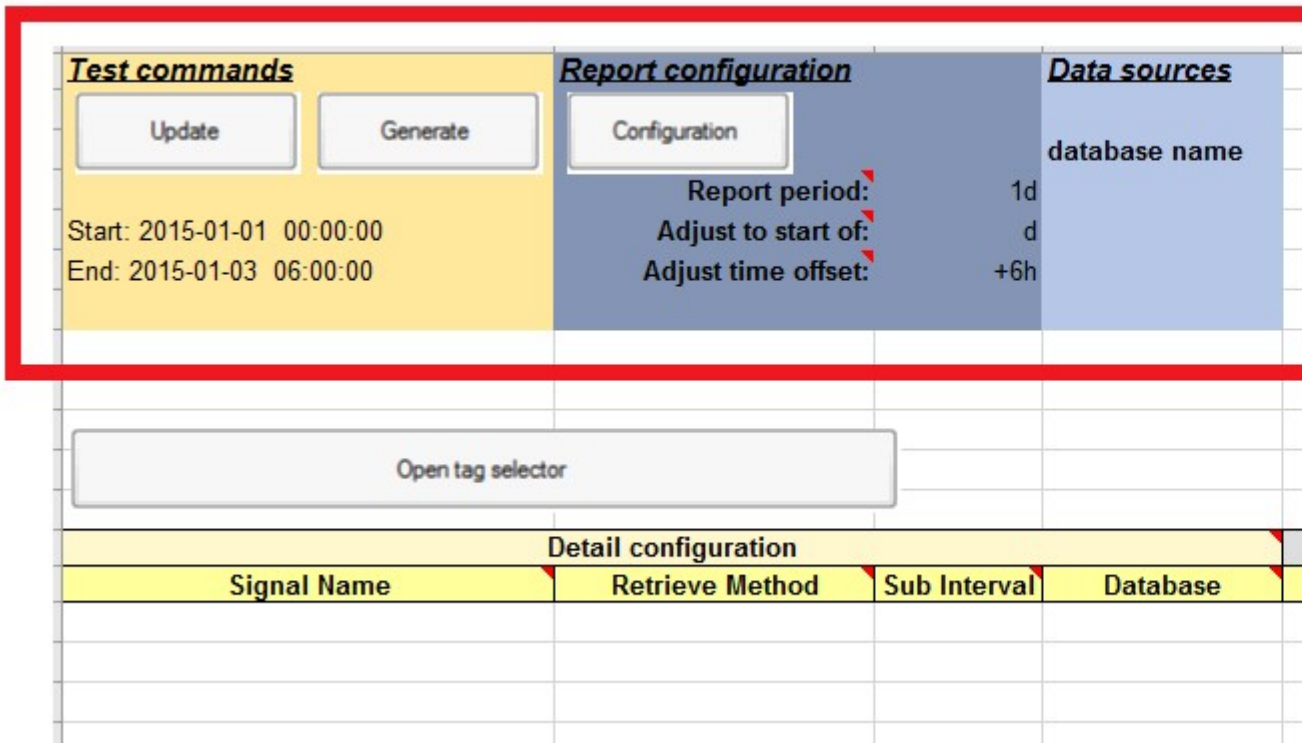
The Excel template is divided into a DataRetrieve sheet where all data will be retrieved and a Layout sheet which will be the sheet used for the result file.
Data can be moved between the DataRetrieve sheet and the Layout sheet using common excel formulas or vba-scripts.

2.6.2 Instructions

2.6.2.1 DataRetrieve sheet

The DataRetrieve sheet can be divided into three different parts.
- General report configuration.
- Data retrieve configuration.
- Update status information.

2.6.2.1.1 General report configuration

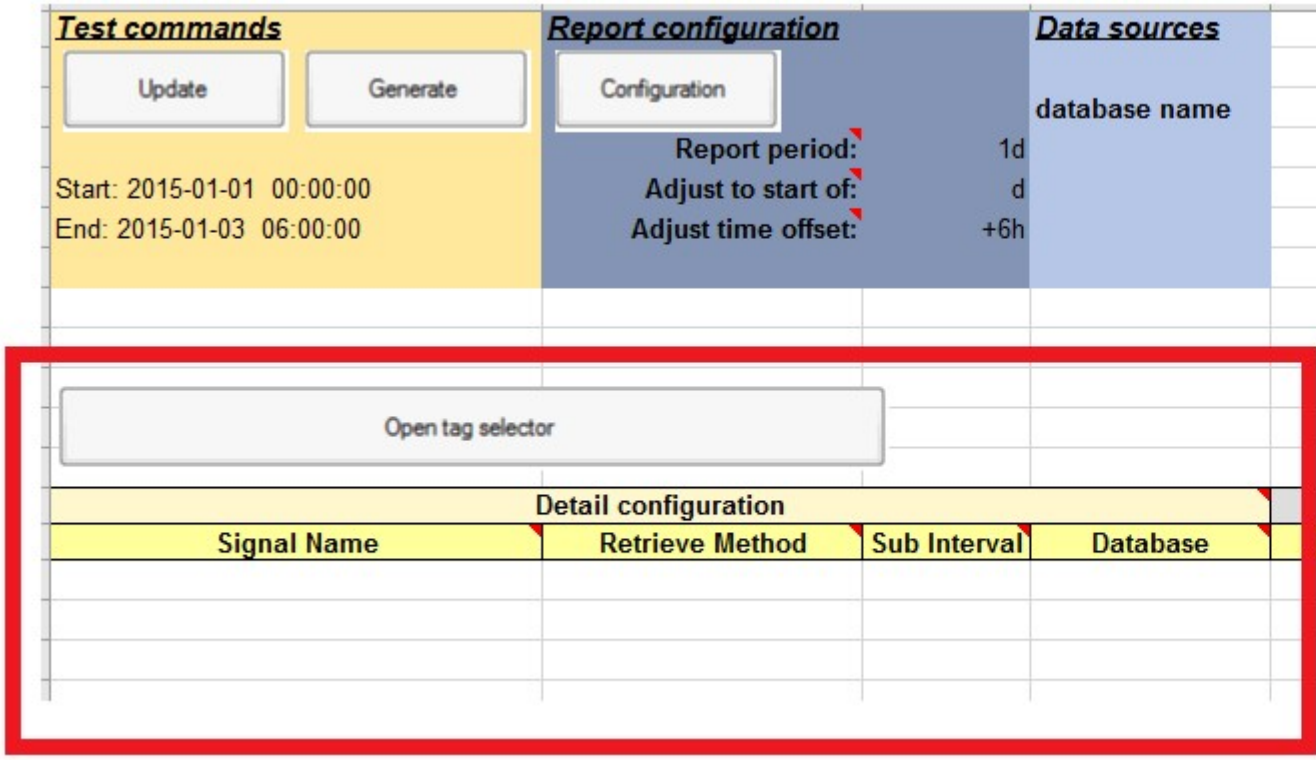


The cells in this configuration are read-only, instead the buttons should be used.

Buttons for testing generating the report.
- Update button fetches all data for signals configured below.
- Generate button generates the report in the same way as a Report generator will.

- Configuration button opens the general configuration dialog. The configuration dialog got three different tabs.
-- Time: Time settings used when the report is executed by a Report generator. These settings can be overridden when generating the report from the Report admin tool.
-- Data sources: Configuration where data should be retrieved.
-- Other: Other general configurations.

2.6.2.1.2 Data retrieve configuration



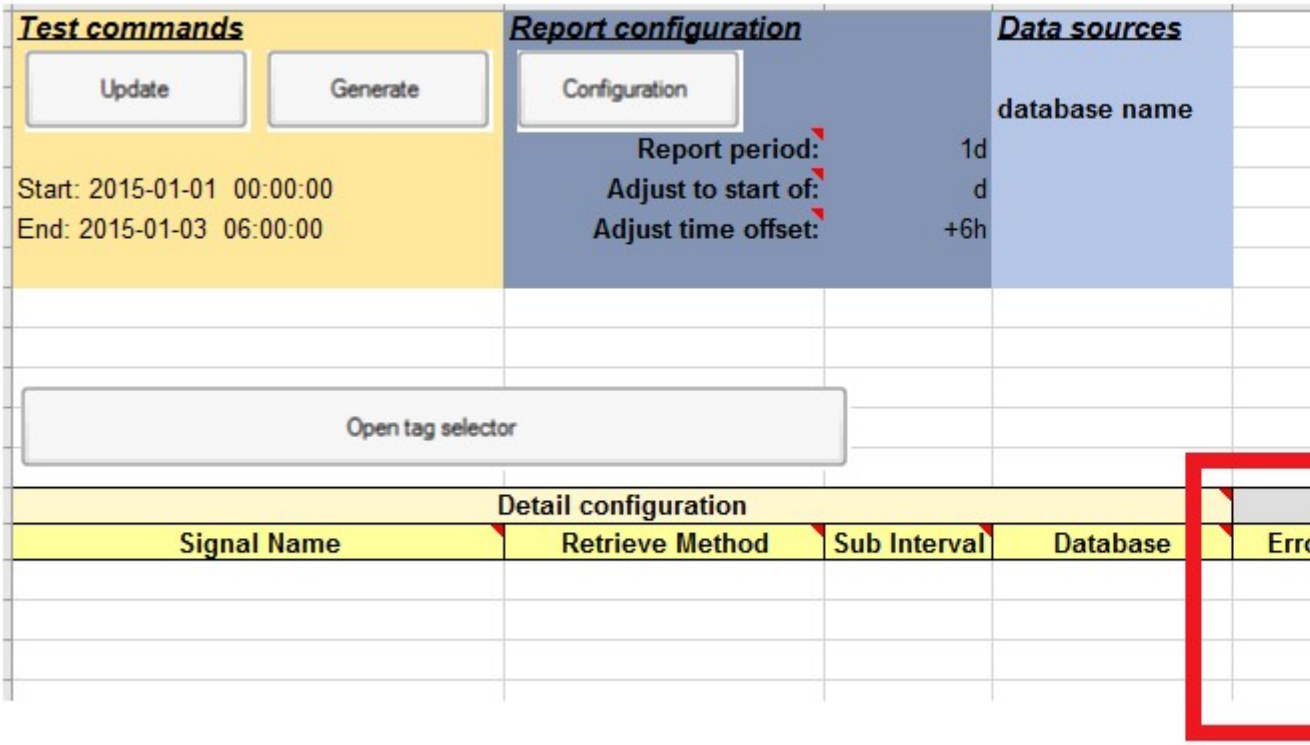
The data retrieve section is used to specify what data should be retrieved.
Tags can be added by either writing in the cells or using the tag selector.

2.6.2.1.2.1 Adding a trend

Trends can be added by right-clicking an empty row in the retrieve configuration and selecting "Add Excel plot".
You will then be asked to select what tags should be in the trend (by selecting the tag names in retrieve configuration) and where the plot should be positioned (usually in the layout sheet).

Before adding a trend the tags that should be in it needs to be configured. It is recommended to use the retrieve method "Series" when fetching data for a trend.

2.6.2.1.3 Update status information



This section displays some information while the data is fetched.
The data itself will be added in columns to the right. "Data Field " columns tells in which column the data is.

2.6.2.2 Layout sheet

The layout sheet is what will be displayed in the result file after the report has been generated.
Data can be moved to this sheet using either common excel formulas or vba-scripts.

Use the Page Break Preview feature in Excel to see how the generated report will look.

2.6.3 Install instructions

2.6.3.1 Prerequisites

- DotNet Framework: 3.5 and 4.0
- Hard-disk space: minimum 200MB
- RAM: minimum 2GB
- Processor speed: minumum 2.0 GHz
- Operating sytem: Windows XP, Windows Server 2003 or newer
- Excel 2010 or newer

2.6.3.2 Installation

The binaries for AutArch report is checked in to the FileStore tables in the database.
The same binaries are used for the editor and the generator.
The binaries and templates for AutArch report can be checked in by either Global update (recommended) or the File repository tool in AutArch Workspace.

The first time a report template is opened.
AutArch Report is installed automatically the first time a report template is opened from the Workspace Report tool or by a Report Generator.

2.6.3.2.1 Adding a customised StandardTemplate and configuration

To be documented.....

2.6.3.3 Update

The report binaries are updated by running Global update from Service monitor.
Note: ReportGenerator and Workspace might have to be updated too.

2.7 XMZ Import/Export

2.7.1 XMZ Export

The purpose of the AutArch Export application is to either perform scheduled daily exports of data from site for domestic use, or manually export selected data from one AutArch system to another AutArch system.

The Export application creates several data files during the export, normally one file per day and data type (Signal data and event data). These files are structured in a compressed internal format.

The files can then be imported by either the AutArch import application or other import applications to other systems.

For scheduled exports the application is executed in “Automatic mode” which means that the application uses the configuration file for its setup and it uses the “ExpLog” table in the AutArch database to keep track on what data has been exported and what data shall be exported.

The AutArch export application can also be executed from the command prompt, this is referred to as manual export.
To schedule the application, the standard function in Windows called “scheduled tasks” can be used.

2.7.1.1 Using XMZ Export

The application is a command line application, therefore there is no icon to start the application. The application is started by opening a command window and change the active directory to where the application is installed.
Select the menu command “Start” -> “Run” and enter “cmd”.
In the Command window enter:

```
cd  "C:\Program Files\AutArch\AutArchExp"
```

(Enter the directory where the application is installed).
To start the application enter “AutArchExp.exe” at the command prompt and add the command line switches that are relevant.

When the program starts it reads the configuration file and command line parameters.
If the command line parameter “/A” is used (the normal behavior) the export history in database is read to find the latest export time. The program then creates export files according to the configuration.
When export is finished the export history is updated in the database. When the program is used for the first time in Automatic mode the configuration parameter “FirstExp” is used to calculate a start date from today.

When using other command line parameters, the program executes in manual mode and exports data according to the parameters.

2.7.1.2 Exported data

During export following data is exported.

2.7.1.2.1 Signals

For signals the following data is exported:

- Configuration data (selectable)
  - OPC Server configuration.
    All properties for all configured OPC Servers and all properties for all OPC Tag groups in the AutArch database.
  - Signal properties.
    All properties for signals that match the export filter and have data that is relevant to the selected time window.
- Signal data
  - All analog signals, including data, time, quality and compression statistics that match the export filter and are relevant to the selected time window.
  - All digital signals, including data, time, quality and calculated operation statistics that match the export filter and are relevant to the selected time window.

2.7.1.2.2 Events

For events the following data is exported:

- Event data
  - Tag definition data for the tag that the events are related to that match the export filter and are relevant to the selected time window.
  - Event data for events that match the export filter and are relevant to the selected time window.

2.7.1.3 Command line parameters

The command line arguments are used for overriding the of default behavior.

2.7.1.3.1 /?

This parameter displays the command line parameters and version information.

2.7.1.3.2 /A

Automatic mode, the program uses the parameters from the configuration file.
This is the normal mode that should be used when scheduling the application.

2.7.1.3.3 Manual mode

When using any of the following switches, the application changes to manual mode. To perform a manual export, at least one of the “/S”, “/s” or “/E” together with the “/T” switch has to be used.

2.7.1.3.4 /S filter

Manual export of signals with signal name filter.
E.g. `"/S MBA%"`

2.7.1.3.5 /s bitmask

Manual export of signals that are marked for export only.
E.g. `"/s 1"`

2.7.1.3.6 /E filter

Manual export of events and alarms with optional filter.
E.g. `"/E MBA%"`

2.7.1.3.7 /T from to

Manual export of data between selected time span.
E.g. `"/T 2006-01-02 2006-01-03"`
A time span has to be added during manual export.

2.7.1.3.8 /D

Manual export of data to selected directory.
e.g. `"/D C:\Temp"`
In manual mode the files are written to the configured "ManualExpDir" if not overridden by the "/D" switch

2.7.1.3.9 /C

Manual export of configuration data for selected Signals and/or Events.

2.7.1.3.10 /I

Interpolate start and end values for each period.

2.7.1.3.11 /IS

Interpolate only start values for each period.

2.7.1.3.12 /IE

Interpolate only end values for each period.

2.7.1.3.13 /K

Used to override the default XMZKey
e.g. `"/K AutArch1`

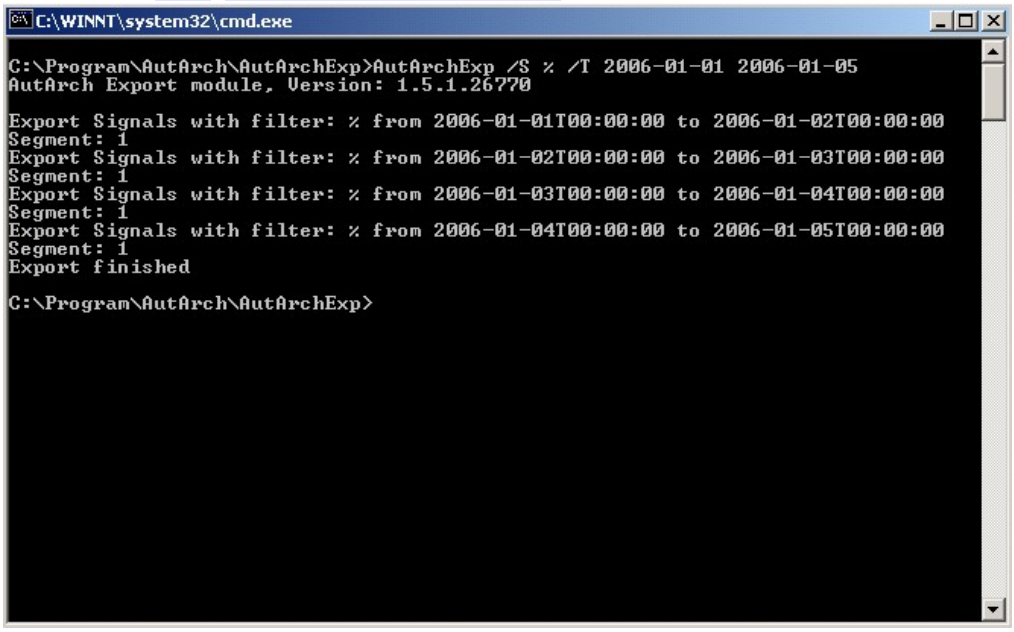
The key must be exaclty 8 characters long.

2.7.1.4 Examples

Examples on how to Export data.

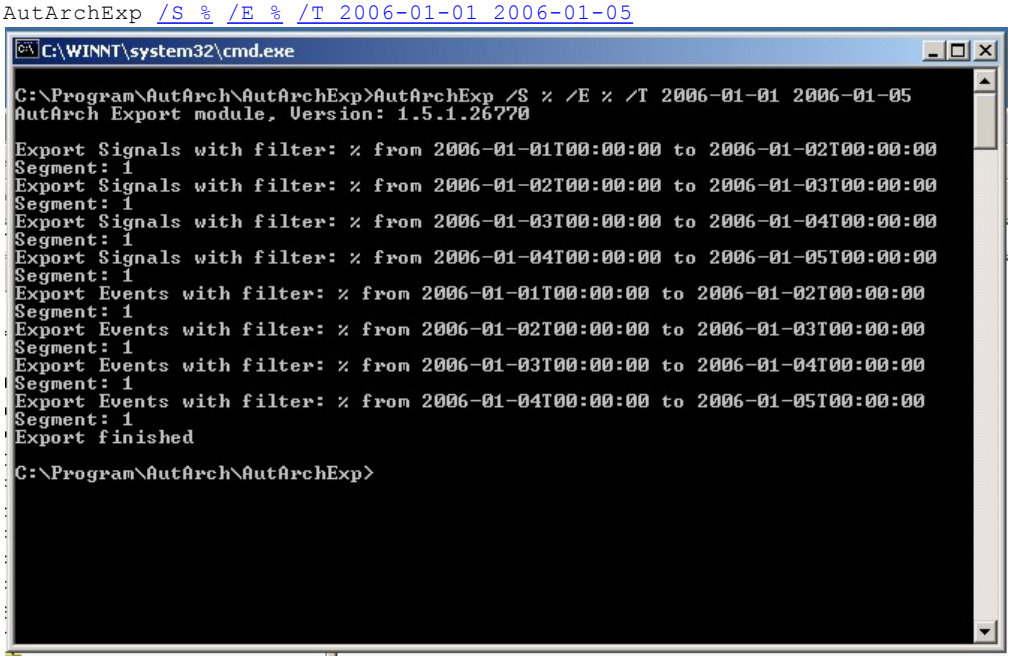
2.7.1.4.1 Export of all signals for four days

```
AutArchExp /S % /T 2006-01-01 2006-01-05
```



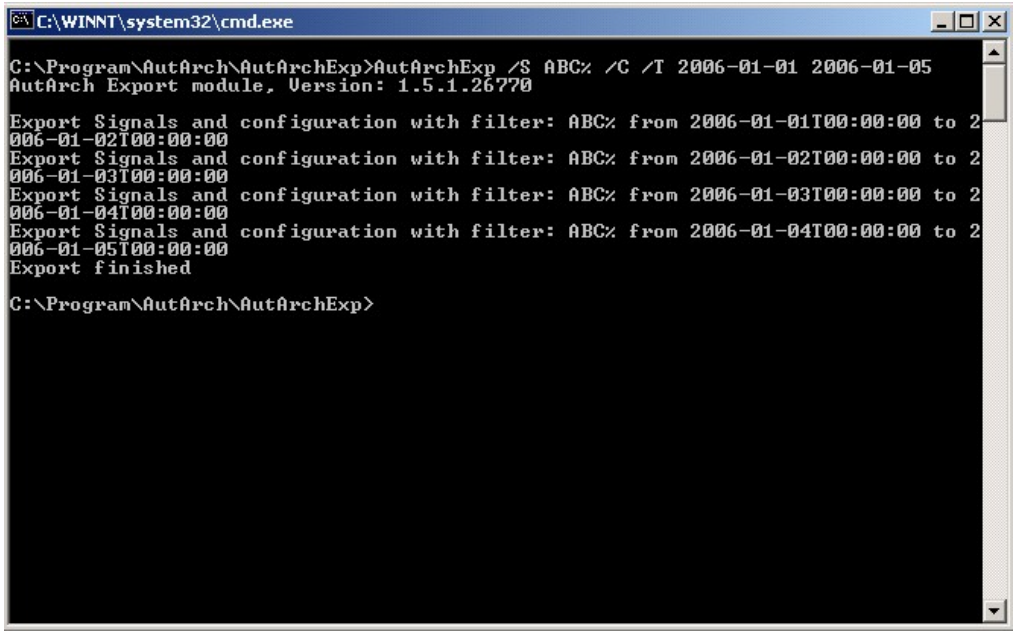
2.7.1.4.2 Export of all signals and events for four days

```
AutArchExp /S % /E % /T 2006-01-01 2006-01-05
```



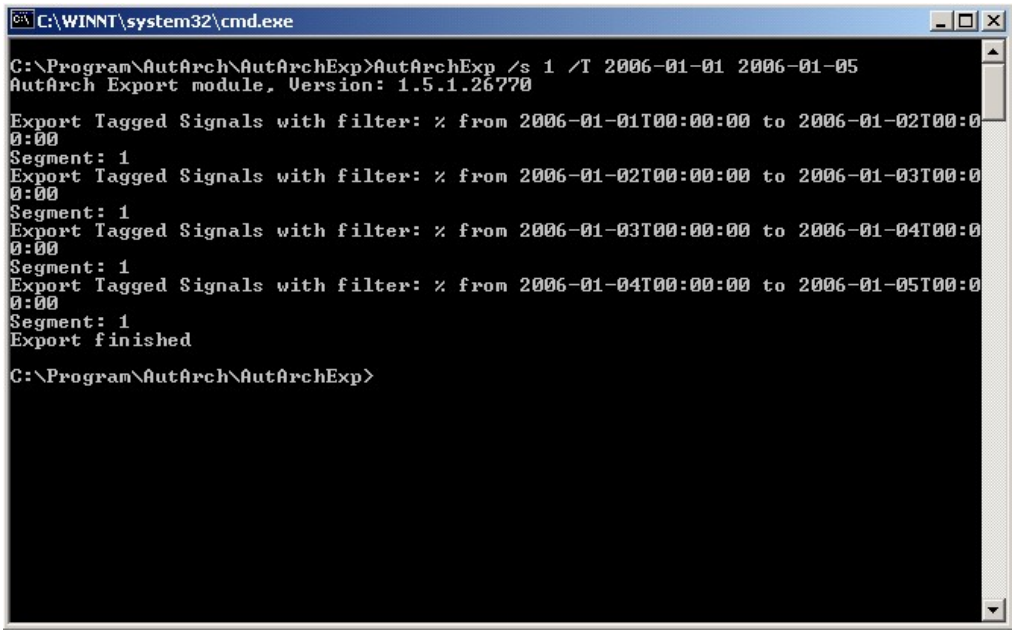
2.7.1.4.3 Export of some signals for four days w ith configuration data

```
AutArchExp /S ABC% /C /T 2006-01-01 2006-01-05
```



2.7.1.4.4 Export of all tagged signals for four days

```
AutArchExp /s 1 /T 2006-01-01 2006-01-05
```



2.7.1.5 Tagged signal export

The tagged signal filter works as a bit mask filter on the “ExportFlag” field in the “SignalDef” table. ExportFlags are configured with the AutArch Admin application.

2.7.1.5.1 Example

Here is an example how the Tagged signal mask filter works. Following sample data with different types of export flags are used.

SignalName	ExportFlag
OUTDOORTEMP	0
INDOORTEMP	1
OUTDOORHUMIDITY	1
INDOORHUMIDITY	3
WATERTEMP	7

If the tagged signal mask “1” is used
E.g. `AutArchExp /s 1 /T 2006-01-01 2006-01-05`

The export application will export the following signals:
INDOORTEMP, OUTDOORHUMIDITY, INDOORHUMIDITY and WATERTEMP.

If the tagged signal mask “2” is used
E.g. `AutArchExp /s 2 /T 2006-01-01 2006-01-05`

The export application will export the following signals:
INDOORHUMIDITY and WATERTEMP

If the tagged signal mask “4” is used
E.g. `AutArchExp /s 4 /T 2006-01-01 2006-01-05`

The export application will export the following signal:

WATERTEMP

In this way some signals in the database can be configured to be exported in several export jobs.

2.7.2 XMZ Import

The purpose of the AutArch Import application is to either perform scheduled daily imports of data from site for domestic use, or manually import selected data from one AutArch system to another AutArch system.

Export files (compressed and encrypted XMZ files or normal XML files with AutArch export format) can then be imported by either the AutArch import application or other import applications to other systems.

For scheduled imports, the application is executed in “Automatic mode” which means that the application uses the configuration file for its setup. It uses the “ImpLog” table in the AutArch database to report what data has been imported.

AutArch import application can also be executed from the command prompt. This is referred to as manual import.
To schedule the application, the standard function in Windows called “scheduled tasks” can be used.

2.7.2.1 Using XMZ Import

The application is a command line application, therefore there is no icon to start the application.
The application is started by opening a command window and changes the active directory to where the application is installed.
Select the menu command “Start” -> “Run” and enter “cmd”.
In the Command window enter:

```
cd  "C:\Program Files\AutArch\AutArchImp"
```

(Enter the directory where the application is installed).
To start the application enter “AutArchImp” at the command prompt and add relevant command line switches.

At start of the program, the configuration file and command line parameters are read. If command line parameter “/A” is used, (the normal behavior) the program analyzes the import directories for files according to the configuration. If the program finds any export files in the directories, these files will then be imported in chronologic order.

When using other command line parameters, the program executes in manual mode and imports data according to the parameters.

2.7.2.2 Imported data

During import following data is imported.

2.7.2.2.1 Signals

For signals the following data is imported:

- Configuration data (selectable)
  - OPC Server configuration.
    All properties for all configured OPC Servers and all properties for all OPC Tag groups in the AutArch database.
  - Signal properties.
    All properties for signals.
- Signal data
  - All analog signals, including data, time and quality.
  - All digital signals, including data, time, quality and calculated operation statistics.

2.7.2.2.2 Events

For events the following data is exported:

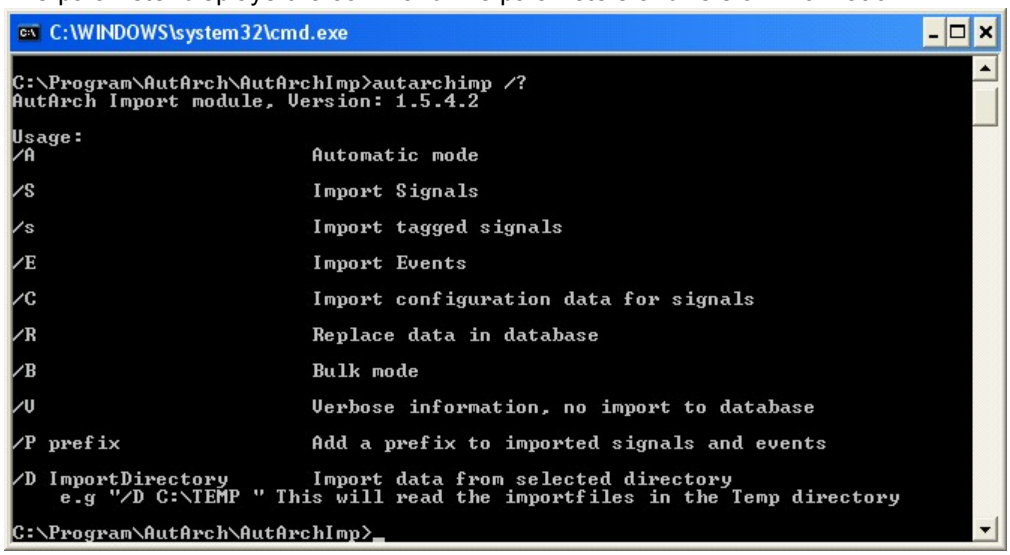
- Event data
  - Tag definition data for the event tag.
  - Event data for events.

2.7.2.3 Command line parameters

The command line arguments are used to override default behavior.

2.7.2.3.1 /?

This parameter displays the command line parameters and version information.



2.7.2.3.2 /A

Automatic mode, the program uses parameters from configuration file.
This is the normal mode that should be used when scheduling the application.

2.7.2.3.3 Manual mode

When using any of the following parameters, the application changes to manual mode.
To perform a manual import, at least one of the "/S", "/s" or "/E" has to be used.

2.7.2.3.4 /S

Manual import of signal files.
E.g. "/S"

2.7.2.3.5 /s

Manual import of tagged signal files.
E.g. "/s"

2.7.2.3.6 /E

Manual import of alarm and event files.
E.g. "/E"

2.7.2.3.7 /C

Manual import of configuration data for selected signals and/or events.

2.7.2.3.8 /R

Replace the data in the database by first deleting data in the database with the same time span as the import files and then importing the data.
This parameter makes it possible to import data older than the data in the database.
When importing data in this mode, no recompression of data is done by the database and no calculation of runtime is done on digital signals.

2.7.2.3.9 /B

Bulk mode inserts the data into the database in batches.
The size of a batch is all the signal values for one signal in one file.
This is a much faster method but it can only be used in combination with /R Replace mode.
The data to import must be valid. If invalid values exist the whole batch is ignored.

2.7.2.3.10 /V

Verbose mode, in this mode the program will only list information from the import files and does not import any data.

2.7.2.3.11 /P prefix

This parameter adds a prefix to the imported signal name.
E.g. `"/P ABC`

This is useful when importing data from several different sites into one database.

2.7.2.3.12 /D directory

Manual import of files from the selected directory.
E.g. `"/D C:\Temp"`
In manual mode, the files are imported from to the configured "ManualImpDir" if not overridden by the "/D" switch.

2.7.2.3.13 /K

Used to override the default XMZKey
e.g. `"/K AutArch1`

The key must be exaclty 8 characters long.

2.7.2.3.14 /X

This switch is used to add an extra value with quality bad, at the end of the imported data.
This is used when importing and inserting the imported data in an period that already contains data and you don't want to create linked lines between the imported data an your previously stored data.

2.7.3 XMZ property descriptions

2.7.3.1 AppEventDir

This configures the directory for where to look for AppEvent files.

Valid for: XMZ Export
Data type: String

2.7.3.2 AppEventTagName

This configures the signal name for storing AppEvents.

Valid for: XMZ Export
Data type: String

2.7.3.3 BulkMode

If this parameter is True, data is inserted into the database in batches.
The size of the batch is all the signal values for one signal in one file.
This is a much faster method but it can only be used in combination with OwerwriteData mode.
The data to import must be valid. If invalid values exist the whole batch is ignored.

Default value: False

Valid for: XMZ Import
Data type: Boolean

2.7.3.4 CommandTimeoutTime

This shall normally be left to the default value i.e. 10 minutes.

Default value: 600

Valid for: XMZ Export, XMZ Import
Data type: int

2.7.3.5 ConfigurationData

This switch configures if configuration data shall be exported/imported in automatic mode.

Default value: True

Valid for: XMZ Export, XMZ Import
Data type: Boolean

2.7.3.6 CreateCounterTags

This configures AutArch Export to include some extra Signals in the Export file for storing statistics of the export.

Default value: False

Valid for: XMZ Export
Data type: Boolean

2.7.3.7 DailyBackupDir

If this parameter is used, the application copies the signal and event files to this directory in automatic mode. The export application will empty the directory before copying the files. The purpose of this parameter is to make it possible for an independent application to backup these files to external media.

Valid for: XMZ Export
Data type: String

2.7.3.8 DeleteImportedFiles

If this parameter is True, the import files are deleted after import. If the parameter is False, the import files are moved to the “Imported” directory.

Default value: False

Valid for: XMZ Import

Data type: Boolean

2.7.3.9 EventCopyDir

The directory where copies of event files shall be written during automatic export.

Valid for: XMZ Export
Data type: String

2.7.3.10 ExportAppEvents

This switch configures if AutArch Export should look for AppEvent files and include these as AutArch Events in the Export file.

Default value: False

Valid for: XMZ Export
Data type: Boolean

2.7.3.11 EventValuesCounterTagName

Configures the Signal name for storing the number of event values for each exported day.

Valid for: XMZ Export
Data type: String

2.7.3.12 EventSegmentsCounterTagName

Configures the Signal name for storing the number of event file segments for each exported day.

Valid for: XMZ Export
Data type: String

2.7.3.13 Events

This switch configures if alarms and events shall be exported or imported.

Default value: True

Valid for: XMZ Export, XMZ Import
Data type: Boolean

2.7.3.14 EventImportedDir

The directory where the imported event files will be moved to after an import.

Default value: D:\DC\Imp\CMS\Imported

Valid for: XMZ Import
Data type: String

2.7.3.15 EventImpDir

The directory where the application searches for event import files during automatic import.

Default value: D:\DC\Imp\CMS

Valid for: XMZ Import
Data type: String

2.7.3.16 EventFilter

This parameter can be used to filter which Events that shall be exported. The filter operates on the SignalName and uses standard “SQL Like” syntax. E.g. ABC% exports all Events beginning with ABC.

Default Value: %

Valid for: XMZ Export
Data type: String

2.7.3.17 EventExpDir

The directory where event files shall be written during automatic export.

Valid for: XMZ Export
Data type: String

2.7.3.18 FirstExp

This parameter is used when the application is executed in automatic mode for the first time. It is used for calculating the first day to export from the database. The default value “2” means that the application starts with exporting the two last days of data. When the application once has run in automatic mode, it will use export statistics from the “ExpLog” table in AutArch database to calculate starting point of next export.

Default value: 2

Valid for: XMZ Export
Data type: Int

2.7.3.19 Interpolate

This switch configures if signals should Interpolate start and end values for each period.

Default value: True

Valid for: XMZ Export
Data type: Boolean

2.7.3.20 MinFreeDiskSpace

The minimum of free disk space required to allow export to run.

Default value:1 000 000 000
(˜0.9 Giga byte)

Valid for: XMZ Export
Data type: Int

2.7.3.21 MaxRecords

This parameter controls the maximum of records in one export file. This is used for limiting the size of export files. If there are more data, the files are split into segments. The split never occur within one signals data. Therefore the actual size per segment can vary. By using a larger number more RAM memory is used.

Default value: 100 000

Valid for: XMZ Export

Data type: Int

2.7.3.22 ManualImportedDir

The directory where the manual imported files will be moved to after a manual import.

Default value: D:\DC\Imp\Manual\Imported

Valid for: XMZ Import
Data type: String

2.7.3.23 ManualImpDir

The directory where the application searches for import files during manual import.
This can be overridden by the command line switch “\D”

Default value: D:\DC\Imp\Manual

Valid for: XMZ Import
Data type: String

2.7.3.24 ManualExpDir

This is the directory where export files are placed during manual export. This can be overridden by the command line switch “\D”

Default value: D:\DC\Exp\Manual

Valid for: XMZ Export
Data type: String

2.7.3.25 OwerwriteData

If this parameter is True, data in database is overwritten by data in the import file.
Data in the database is first deleted by using the import files time span.

If the switch is False, imported data can only be added at the end of the database.
During this mode data is also re compressed by the database compression algorithm and digital signals runtime accumulators are recalculated.

Default value: False

Valid for: XMZ Import
Data type: Boolean

2.7.3.26 Prefix

If this parameter is set to a text string, that string is used to set a prefix to all signal names during import.
This can be used for merging data from two sites into one destination database.

Default value: (empty string)

Valid for: XMZ Import
Data type: String

2.7.3.27 SQL_Password

The encrypted password to connect to the AutArch database.

Default value: lBqfwjpM5RIA

Valid for: XMZ Export, XMZ Import
Data type: String

2.7.3.28 SQL_ConnectionString

This is the SQL server connection string for connecting to the AutArch database.
This is normally configured by the setup program but can be manually edited concerning: “Data Source” and “Initial Catalog”.

Valid for: XMZ Export, XMZ Import
Data type: String

2.7.3.29 Site

This is the name of the site and is used as a check during import to avoid importing wrong files into a database.

Valid for: XMZ Export, XMZ Import
Data type: String

2.7.3.30 SignalValuesCounterTagName

Configures the Signal name for storing the number of signal values for each exported day.

Valid for: XMZ Export
Data type: String

2.7.3.31 SignalSegmentsCounterTagName

Configures the Signal name for storing the number of signal file segments for each exported day.

Valid for: XMZ Export
Data type: String

2.7.3.32 Signals

This switch configures if signals shall be exported or imported in automatic mode.

Default value: True

Valid for: XMZ Export, XMZ Import
Data type: Boolean

2.7.3.33 SignalImportedDir

The directory where the imported signal files will be moved to after an import.

Default value: D:\DC\Imp\CMS\Imported

Valid for: XMZ Import
Data type: String

2.7.3.34 SignalImpDir

The directory where the application searches for signal import files during automatic import.

Default value: D:\DC\Imp\CMS

Valid for: XMZ Import
Data type: String

2.7.3.35 SignalFilter

This parameter can be used to filter which signals that shall be exported. The filter operates on the SignalName and uses standard “SQL Like” syntax. E.g. ABC% exports all signals beginning with ABC.

Default value: %

Valid for: XMZ Export
Data type: String

2.7.3.36 SignalExpDir

The directory where signal files shall be written during automatic export.

Valid for: XMZ Export
Data type: String

2.7.3.37 TaggedSignals

This switch configures if tagged signals shall be exported or imported in automatic mode.
See examples

Default value: True

Valid for: XMZ Export, XMZ Import
Data type: Boolean

2.7.3.38 TaggedSignalMask

This parameter is used to filter which tagged signals that shall be exported. The filter operates as a bit mask on the ExportFlag field in the SignalDef table.
See examples

Default value: 1

Valid for: XMZ Export
Data type: Int

2.7.3.39 TaggedSignalImpDir

The directory where the application searches for tagged signal import files during automatic import.

Default value: D:\DC\Imp\RMS

Valid for: XMZ Import
Data type: String

2.7.3.40 TaggedSignalExpDir

The directory where tagged signal files shall be written when exporting in automatic mode.

Valid for: XMZ Export
Data type: String

2.7.3.41 XMZKey

The key is used for encrypting/decrypting the XMZ export/import files.

The value can be stored in the configuration file either encrypted or not.

Default value: AutArch1

Valid for: XMZ Export, XMZ Import
Data type: String

2.7.4 Logging

During import/export, the applications writes information to the ImpLog/ExpLog table in AutArch database.
The information in this table is only used as a log of all imported data.

An information event is also written to the normal Windows event log, and if any errors would occur, these will also be written to Windows event log named AutArch.

2.7.5 XMZ files

The export/import files are written in a compressed internal format.
The naming of the files is as follow:

`TTYYYY-MM-DD_nnn_SSSS.XMZ`

TT	Two characters for data type for signal data this will be **SD**, for tagged signal data **TD** and for Alarms and events it will be **AE**.
YYYY	Four digits representing the year.
MM	Two digits (including leading zero) representing the month.
DD	Two digits (including leading zero) representing the day.
_	Underscore (only for readability).
nnn	Segment number, used if several files are needed for one day.
_	Underscore (only for readability).
SSSS	Site name or collector name (for traceability when transferring files).
_YYYY-MM-DD	*If the Eventexport has embeded AppEvents this extra date indicates that AppEvents exist up till that date.*
.XMZ	File extension, XMZ for compressed and encrypted files.

For manually exported files the file name also includes the start time for the exported data.
TTYYYY-MM-DD_HHMISS_nnn_SSSS.XMZ

HHMMSS	Six digits (Hour, minute and second).

The last file per day is renamed to ".LAST.XMZ"

2.7.6 Scheduling import/export

Following is an example of how a schedule can be added that runs the import application.

- In the “Windows control panel” select “Scheduled tasks” and click “Add scheduled task”.
- Browse to the directory where AutArch import is installed and select the AutArchImp.exe file, and add the command line parameter “/A”.
- Select an appropriate schedule e.g. once per day.
- Select the time when to run the task.
- Enter a username and password for the task (use a username that has normal rights to write files in the import directories).

2.8 Status

AutArch status is a small tray application that periodically checks some basic status.
The tray application is usually minimized but if any problem is detected a popup will be shown.

AutArch status checks that:
- Recording are working as expected. - Last value in the database is progressing.
- Database jobs are performed.

2.9 CompTune

To be documented.....

3 Backup and restore

All important data in an AutArch system is stored in the database.
This describes different ways to backup the database and to restore it if necessary.

3.1 Microsoft SQL backup

To be documented.....

3.2 Full hard drive backup

To be documented.....

3.3 XMZ Import/Export

Please see the XMZ Import/Export chapter.

4 Release Notes

4.1 Known bugs and limitations

- Workspace does not scale well in newer operating systems and high dpi.
- Sometimes when opening the contextmenu for a service in ServiceMonitor the menu is opened for the wrong service.
  As a temporary workaround a label is added to the top of the menu showing wich service is selected.
- Minor GUI problems in Workspace.
- All system stat tags gets qualified to the first node in OPCServer list
- It is not possible to disable diagnostic tags for anything except calculation agent.
- Memory leak in calculation. As temporary workaround the ChaosMonkey can be used to restart the agents at desired interval.

4.2 Releases

4.2.1 3.3.0

4.2.1.1 New features

- Compression analyzer: New tool for reviewing and configuring compression settings.
- New test platform for long running tests, Long running platform.

- Upgraded IronPython (used in python calculations) from version 2.7.3 to 2.7.8
- Improved performance of XMZ export from databases with large dbo.Change tables.
- Database patch 3.0.1.06
- ServiceMonitor genereates a default config if requested config file does not exist, #153
- Configuration dialogs in ServiceMonitor is not modal anymore, #155
- Improved zoom in trends with high density of values, #149
- Major performance improvement of XY Plot, #233
- Multiple improvements of TagValueGrid, #252
- Listing third party components in About dialog according to their MIT licenses, #140
- Values imported from Excel gets AAManual quality, #244
- Excluded unnecessary files from setup packages to reduce size, #248
- Button too freeze the grid when adding or changing items in Admin, #151
- GetLogValues method in RestProvider, #215
- Possible to disable the "Save Workspace" dialog when closing the Workspace, #209
- DataAdapters now fully support collection of double precision, #211
- MaxDivergence is always set to 0.5% of specified range, #191
- Major improvement of memory usage in Workspace, #208
- All event filters are available in combo box above grid, #182
- TimeSelector Offset is disabled when not in any timegroup, #183

4.2.1.2 Fixed bugs

- Solved a memory leak in Calculation agent
- Installation of AutArch Workspace adds the correct user rights to the installation folder, #146
- ServiceMonitor configuration file is always created in the installation folder
- Events were not removed from the event grid in live mode, #158
- Values were not removed from TagValueGrid in live mode, #189
- Live subscription of events did not stop when closing event grid, #159
- Unexpected behaviour when adding or subtracting time over a DST-shifting, #162
- User did not get sufficient rights to database jobs, #251, #239
- Workspace could crash when editing a dashboard before it was fully loaded, #249
- ReportGenerator failed to generate report if the template was open in Excel, #178
- TagValueGrid could stop working in live, #207, #210
- XY Plot could stop working in live, #160, #157
- Radar Plot could stop working in live, #156
- An old value could get repeated by SigForceUpdate, #213
- TimeGroup with TagValueGrid did not work, #193
- EventGrid menu of selected columns was not correct when opening saved workspace, #92
- Improved selection of EventGrid columns, #91

4.2.2 3.2.8

4.2.2.1 New features

- System alarm check at startup is configurable, #129.
- SQL Patch 3.0.1.03
- Max sweep size configurable for Calculation agent.
- Max end time configurable for Calculation agent.
- Excel export includes milliseconds in timestamps.
- Opening a default config in ServiceMonitor if requested config does not exist.

4.2.2.2 Fixed bugs

- Tags not added to plugin when starting workspace from command line, #130.
- Workspace does not appear on top when started, #126.
- Calculation agent stopped progressing in time, #132.
- TimePeriod and AADateTime is configurable in ServiceMonitor.

4.2.3 3.2.7

4.2.3.1 New features

4.2.3.2 Fixed bugs

- Fixed problems in XMZ export.

4.2.4 3.2.6

4.2.4.1 New features

- Improved data fetching performance.
- Improved Calculation agent performance.
- Improved recalc performance.
- Start ClickOnce published Workspace with arguments.
- Multi-value html editor.
- Machine learning agent.
- Machine learning training html ui.
- Updated codesign certificate.
- Improved data fetching performance

4.2.4.2 Fixed bugs

4.2.5 3.2.5

4.2.5.1 New features

- Event filter for OPCAE and OPCUA.

4.2.5.2 Fixed bugs

4.2.6 3.2.4

4.2.6.1 New features

4.2.6.2 Fixed bugs

- Fixed problems in XMZ export.

4.2.7 3.2.3

4.2.7.1 New features

4.2.7.2 Fixed bugs

- Fixed problems in XMZ export.

4.2.8 3.2.2

4.2.8.1 New features

- Database patch 3.0.0.16.
- Moved columns in the Admin tool.
- Possible to store a custom config for report StandardTemplate.
- Timestamps in report includes milliseconds.
- Password dialog for admin tools in Workspace.
- Rename of tags supported in Admin tool.
- Can insert a comment when importing excel values.
- Possibility to exclude multiple clients from AutArch Status.
- Possibility to set allowed diff time individually in AutArch Status.

4.2.8.2 Fixed bugs

- Multiple GUI fixes in Workspace.
- Multiple GUI fixes in report excel template.
- File association with Workspace files (.aaws) was not working.
- It was not possible to disable diagnostic tags for calculation agent.
- StandardTemplate for report was not imported correctly with GlobalUpdate.
- Caclculation agents with COB calculations could run intervals smaller than configured amount.
- Calculations stored from Python editor got wrong end time.
- Supervision server could restart a service that was currently starting.
- Log all command from ServiceMonitor was not working.
- XY-plot works in live mode.
- XMZ export could possible create an incorrect end interpolated value.

4.2.9 3.2.1

4.2.9.1 New features

4.2.9.2 Fixed bugs

- OPCDA Client sometimes attempted to reconnect to the server while the client was shutting down.
- Multiple GUI fixes in Workspace.

4.2.10 3.2.0

4.2.10.1 New features

- Dashboard officially supported.
- Report officially supported.
- LiveExcel officially supported.
- New GUI and official support for Excel export/import.
- Database version 3.0.0.13
- Event subscription through OPCUA.
- New setup package for AutArch database with SQL 2014
- MultiValue tags.
- GlobalUpdate can now be used to patch database and is replacing AutArchDatabasePatcher.
  (AutArchDatabasePatcher is still working but will not be maintained)
- OPCServer and OPCGroup configuration in Admin are equalized with other tabs.
- Admin search is using wildcards instead of regex.
- User-friendly configuration dialog when installing ClickOnce Workspace.

4.2.10.2 Fixed bugs

- CalculationAgent config sometimes gets corrupted.
- Comparison of datetime rounded to SqlDatetime gave unexpected result.
- Memory leak in CalculationAgent.
- SupervisionServer does not automatically set itself to auto-start.
- Adapter awaits broker connection for 10 seconds at startup before reading offline config.
- Calculation of tags with property ExtendLastValue produced incorrect values.
- OPCAE Recording Node did not run SigForceUpdate which caused AutArch Status to alarm.
- Minor GUI improvements in AutArch Workspace.

4.2.11 3.1.2

4.2.11.1 New features

- Database version 3.0.0.08

4.2.11.2 Fixed bugs

- Not polling values for tags with subscription problems.
- Extra poll attempt after successfully setting up subscriptions.
- Reduced unnecessary logging.
- Memory leak in CaclulationAgent.

4.2.12 3.1.1

4.2.12.1 New features

Last value only stays in collector service for max 10 minutes #41

4.2.12.2 Fixed bugs

- Validation of the value insurance platform is not working.

4.2.13 3.1.0

4.2.13.1 New features

- Calculation Agent is now fully tested and approved
- All RecordingNodes poll values for all tags when started
- Improved Broker performance
- Cleaned all configuration files so that they only contain what is necessary.
- Added comments to configuration files
- AutArch components logs their own errors to eventlog in database.
- Database version 3.0.0.04

4.2.13.2 Fixed bugs

- Invallid char in XMZ Export, #23
- Error serializing TuneStatusEnum in offline config, #25
- Error serializing PythonInof in offline config, #2

4.2.14 3.0.0

4.2.14.1 New features

- Improved performance and resolved bugs by moving calculations from the DataBroker to CalculationAgent.
- To make installation simpler all nodes are now hosted in a CentralServices
- The old Admin application is completely replaced by the new tool included in Workspace.
- Improved test coverage with new testplatforms

4.2.14.2 Fixed bugs

- Problems with calculation order of Python calculations
- Performance issues in DataBroker

4.2.15 2.0.0

4.2.15.1 New features

- Complete rebuild of Service Framework including new OPC clients
- OPCClients can react on new tag configuration and apply it in runtime without restart
- S7Adapter (fastlogger) used for faster sampling of values from Siemens S7 systems
- Redundant adapters/brokers
- Easier creation of aggregated tags
- Calculations written in Python

4.2.15.2 Fixed bugs

5 Tests

The AutArch system is tested in multiple automated and manual ways. The different test platforms is briefly described below.
Before a new AutArch version is released all necessary tests are performed. A full list of performed tests and their result can be found in the release package.

5.1 Manual tests

Before a release a predefined list of tests are performed and documented, the document can be found in the release package.
The development team also tests continuously as the system develops.

5.2 Automated tests

The automated tests are executed by TeamCity build agents.
The tests are executed by a schedule and can also be trigged manually.

5.2.1 Unit tests

Unit tests consists of several fast test to give the developer team a quick overview.

Unit tests are executed at every new commit and once every night.

5.2.2 Integration tests

Integration tests are somewhat slower and tests the integration between different AutArch components.

Integrationa tests are executed once every two hours if there are pending changes and once every night.

5.2.3 Value insurance platform

The value insurance platform is used to insure that values are collected, calculated and refined in a correct way and without losing any valuable data.

The platform is also used to make sure that the redundancy of recording nodes are working as expected.

The test starts with a database without any values.

Recording nodes are configured to collect data from a server that is publishing a known set of values in the course of 12 hours.

During this time the nodes of a redundant pair is periodically killed to make sure redundancy switches occurs without any loss of data.

After 12 hours the data is analyzed to make sure that all expected values have been collected without any loss, calculations have been performed and data is refined as expected.

The platform runs once a day. It starts at 18:00 and is validated at 07:00.

If there are any pending changes the platform will update itself.

The platform can also be configured to run at a higher speed, up to 12x the normal speed which gives a runtime of one hour.

5.2.4 Availability platform

The availability platform is mainly used to make sure the system can restore itself after all kinds of disturbances.

It is also used to make sure that the platform is able to handle changes in the system, i.e. configuration changes, during runtime.

The platform is running continuously, after a testrun is completed a new one is triggered automatically.

If there are any pending changes the platform will update itself.