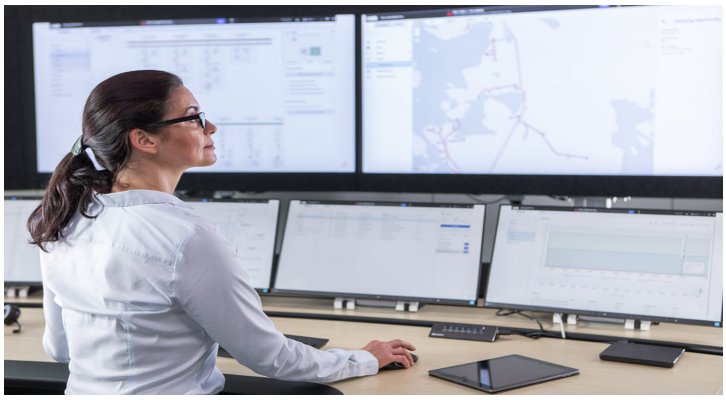
GRID AUTOMATION PRODUCTS

MicroSCADA X SYS600 10.2

Historian Configuration and Administration



Document ID: 1MRK 511 472-UEN
Issued: March 2021
Revision: A
Product version: 10.2

Table of contents

Section 1 Copyrights....5

Section 2 Introduction....7
2.1 This manual....7
2.2 Use of symbols....7
2.3 Intended audience....7
2.4 Related Documents....8
2.5 Document conventions....8
2.6 Document revisions....8

Section 3 Vtrin as a tool to manage the database....9
3.1 Connecting the database....9
3.2 Basic UI components....10
3.3 List display....10
3.4 Tree view....13
3.4.1 Default content....13
3.4.2 Functionality....14
3.5 Generic properties window....17
3.6 Access control....18
3.6.1 Vtrin User Interface....18
3.6.2 Tree Security....18
3.6.3 Role Security....21
3.7 Roles....22
3.8 Text translation....24

Section 4 Advanced Configuration of the Data Collection....27
4.1 Tag and relationships between related classes....27
4.2 Creating tags using Vtrin....35
4.3 Fine tuning using Variables....36

Section 5 Process history in detail....49
5.1 Related classes....49
5.2 Using History collection templates....50
5.3 Adding new History Transformation....51
5.3.1 Adding Trend to Tree item context menu....52
5.4 History Tables configuration – the base for all history storing....53
5.5 Physical History Tables....55
5.6 Recalculating secondary histories....56

Section 6 Administrating a Database in production....57
6.1 Introduction....57
6.2 Environment....57
6.3 Disk Configuration....58

6.4 Database structure....59
6.5 Diagnostics....61
6.5.1 General....61
6.5.2 Component status....61
6.5.3 MessageLog....63
6.5.4 Diagnostics log files....64
6.5.5 Windows Event Log....65
6.6 Services....65
6.6.1 General....65
6.6.2 RTDB-ServiceManager....66
6.6.2.1 Functionality....66
6.6.2.2 Configuration....66
6.6.2.3 Diagnostics....67
6.6.3 RTDB-CVMCServer....67
6.6.3.1 Functionality....67
6.6.3.2 Configuration....67
6.6.3.3 Diagnostics....67
6.6.4 RTDB-Transformator....68
6.6.4.1 Functionality....68
6.6.4.2 Configuration....68
6.6.4.3 Diagnostics....68
6.6.5 RTDB-EcPerfMon....68
6.6.5.1 Functionality....68
6.6.5.2 Configuration....69
6.6.5.3 Diagnostics....69
6.6.6 RTDB-Scheduler....69
6.6.6.1 Functionality....69
6.6.6.2 Configuration....69
6.6.6.3 Diagnostics....69
6.6.7 Vtrin Server....70
6.6.7.1 Functionality....70
6.6.7.2 Configuration....70
6.6.7.3 Diagnostics....70
6.6.8 Vtrin Server RTDB Internal....70
6.6.8.1 Functionality....70
6.6.8.2 Configuration....71
6.6.8.3 Diagnostics....71
6.6.9 VtrinLink....71
6.6.9.1 Functionality....71
6.6.9.2 Configuration....72
6.6.9.3 Diagnostics....72
6.6.10 RTDB-TagConsistencyController....72
6.6.10.1 Functionality....72
6.6.10.2 Configuration....72
6.6.10.3 Diagnostics....73
6.6.11 SimbaServer....73

6.6.11.1 Functionality ... 73
6.6.11.2 Configuration ... 73
6.6.12 RTDB-OpcUaServer ... 73
6.6.12.1 Introduction ... 73
6.6.12.2 Functionality ... 74
6.6.12.3 Configuration ... 74
6.6.12.4 Connecting the server with Unified Automation UA Expert ... 74
6.7 Management tasks ... 80
6.7.1 Starting SYS600 Historian ... 80
6.7.2 Stopping SYS600 Historian ... 81
6.7.3 Restarting, enabling and disabling separate SYS600 Historian services ... 81
6.7.4 Backing Up the SYS600 Historian Database and Software Files ... 82
6.7.5 Restoring the SYS600 Historian Database from Backup ... 84
6.7.5.1 Restoring offline backups ... 84
6.7.5.2 Restoring SYS600 Historian from online backups ... 84
6.7.5.3 Restoring backups from live backups from tape ... 84
6.7.6 Scanning the database tables ... 84
6.7.7 Managing Client Software distribution ... 86
6.7.8 Upgrading Historian ... 87

Section 7 Uninstalling SYS600 Historian ... 89
7.1 Uninstalling ... 89

Section 1 Copyrights

The information in this document is subject to change without notice and should not be construed as a commitment by Hitachi Power Grids. Hitachi Power Grids assumes no responsibility for any errors that may appear in this document.

In no event shall Hitachi Power Grids be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall Hitachi Power Grids be liable for incidental or consequential damages arising from the use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without written permission from Hitachi Power Grids, and the contents thereof must not be imparted to a third party nor used for any unauthorized purpose.

The software or hardware described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Trademarks

ABB is a registered trademark of ABB Asea Brown Boveri Ltd. Manufactured by/for a Hitachi Power Grids company. All other brand or product names mentioned in this document may be trademarks or registered trademarks of their respective holders.

Guarantee

Please inquire about the terms of guarantee from your nearest Hitachi Power Grids representative.

Third Party Copyright Notices

List of Third Party Copyright notices are documented in "3rd party licenses.txt" and other locations mentioned in the file in SYS600 and DMS600 installation packages.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (https://www.openssl.org/). This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Section 2 Introduction

2.1 This manual

This manual provides information on how to configure and administer the data storage of the SYS600 Historian. The configuration of the SYS600 Historian is divided to two stages. The integration of the data is configured in the base system and the addition processing and viewing of the data is configured in the SYS600 Historian database. The integration is described in the application design manual and in the system configuration manual. This manual covers the usage of the SYS600 Historian database.

2.2 Use of symbols

This publication includes warning, caution and information symbols where appropriate to point out safety-related or other important information. It also includes tips to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:

Warning icon indicates the presence of a hazard which could result in personal injury.

Caution icon indicates important information or a warning related to the concept discussed in the text. It might indicate the presence of a hazard, which could result in corruption of software or damage to equipment/property.

Information icon alerts the reader to relevant factors and conditions.

Tip icon indicates advice on, for example, how to design a project or how to use a certain function.

Although warning hazards are related to personal injury, and caution hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all warnings and caution notices.

2.3 Intended audience

This manual is intended for installation personnel, administrators and skilled operators to support installation of the software.

2.4 Related Documents

Name of the document	Document ID
SYS600 10.2 Historian Monitor Configuration	1MRK 511 473-UEN
SYS600 10.2 Historian Operation	1MRK 511 474-UEN
SYS600 10.2 Application Design	1MRK 511 466-UEN
SYS600 10.2 System Configuration	1MRK 511 481-UEN

2.5 Document conventions

The following conventions are used for the presentation of material:

- The words in names of screen elements (for example, the title in the title bar of a dialog, the label for a field of a dialog box) are initially capitalized.
- Capital letters are used for file names.
- Capital letters are used for the name of a keyboard key if it is labeled on the keyboard. For example, press the CTRL key. Although the Enter and Shift keys are not labeled they are written in capital letters, e.g. press ENTER.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the space bar, comma key and so on.
- Press CTRL+C indicates that the user must hold down the CTRL key while pressing the C key (in this case, to copy a selected object).
- Press ALT E C indicates that the user presses and releases each key in sequence (in this case, to copy a selected object).
- The names of push and toggle buttons are boldfaced. For example, click **OK**.
- The names of menus and menu items are boldfaced. For example, the **File** menu.
    - The following convention is used for menu operations: **Menu Name/Menu Item/ Cascaded Menu Item**. For example: select **File/Open/New Project**.
    - The **Start** menu name always refers to the **Start** menu on the Windows Task Bar.
- System prompts/messages and user responses/input are shown in the Courier font. For example, if the user enters a value that is out of range, the following message is displayed: `Entered value is not valid`.
  The user may be told to enter the string MIF349 in a field. The string is shown as follows in the procedure: `MIF349`
- Variables are shown using lowercase letters: sequence name

2.6 Document revisions

Revision	Version number	Date	History
A	10.2	31.03.2021	New document for SYS600 10.2

Section 3 Vtrin as a tool to manage the database

SYS600 Historian is a database system that can be used to collect data from SYS600. Data can be refined with various methods. The data can be visualized as trends and reports can be generated based on the data. The SYS600 Historian is accessed using a separate user interface, called Vtrin.

SYS600 Historian installation does not support Non-English Operating Systems

3.1 Connecting the database

SYS600 User interface, Vtrin, can be started from Windows **Start** menu **All Programs/ABB/ Vtrin**. The dialog displayed in Figure 1 is opened. Depending on the interactive user the content might include the location prefilled. In case the interactive user is same as the user that was used to install SYS600 Historian Vtrin is able to prefill the location string to be similar in Figure 1.

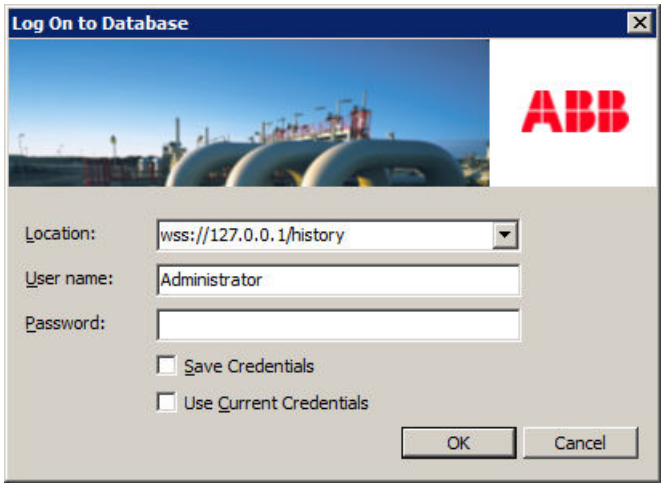


Figure 1: Vtrin connection dialog to connect SYS600 Historian.

In case the interactive user used to open Vtrin first time is not the same as installer user, the contents of Location is empty. The connection to the local installation is wss://127.0.0.1/ history.

The user name and password can be given, or if the user is logged on as a user that is recognized to have an access to the database, (e.g. the database administrator user given in setup phase) these could be left as blank. In that case, Kerberos authentication is attempted to be used. The user can also select **Use Current Credentials** to use Kerberos authentication or check **Save Credentials** to enable server to encrypt and store the credentials into the local machine to make future logins easier. To allow the user to select from multiple locations when Vtrin is used to connect multiple data sources located into multiple computers, Vtrin remembers the databases it has previously successfully connected to.

3.2 Basic UI components

The Figure 2 shows how the Vtrin looks like after being successfully connected.

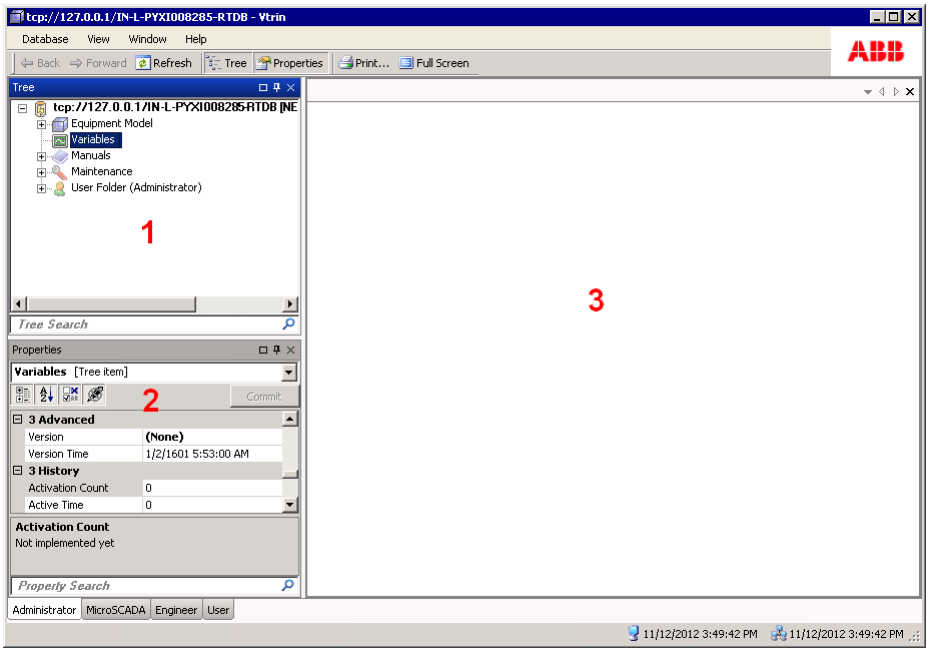


Figure 2: SYS600 Historian has been connected successfully using Vtrin user interface with user with administrator credentials. The main components are main menu, toolbar, tree (1), generic properties window (2) and area for the components opened from tree (3, current blank).

The components on the layout can be dragged, dropped and pinned for different locations as usual. When a user interface component is selected from tree view, it is opened in the blank area. Clicking the Variables tree item opens a list display of the variables that are currently available in the database (Figure 3).

3.3 List display

A list can be used to display, for example, configuration data. As an example, variables are viewed as a list. All the graphical elements are not described here in detail, see Historian Operation manual for more detailed description.

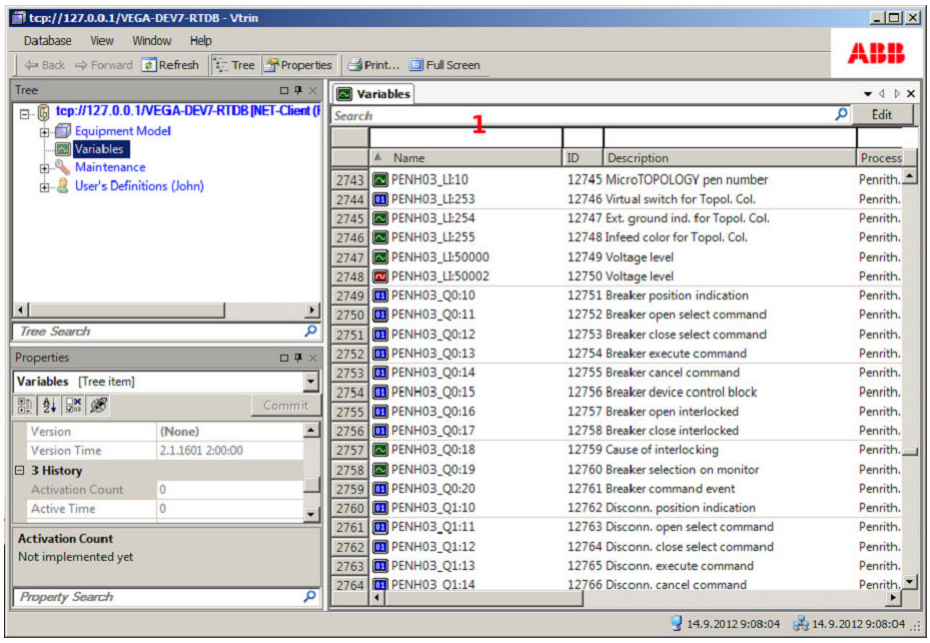


Figure 3: Variables list display showing a set of variables that are generated when SYS600 is connected to SYS600 Historian. Rows can be searched using search fields (1). The top-most search field searches simultaneously from all columns.

Key features of the list display are:

- Shows the selected properties of selected class.
- Arrange the data based on any selected property
- Filter based on any available property. This is done by using the filter box on top of each property. Available filtering wildcards are ? (1 character) or * (0 or more characters). Combining different filtering criteria is done using the comma separator (,). NOT operator is either the - or the ! character. Note that not only string properties can be filtered but as well all other kinds of properties including the time properties.
- Select the content the list is showing. Many times it is good idea to pre-configure different kinds of lists for various use-cases. Configuration is done from the properties of the list view. Properties can be selected from the context menu of the list view (right click from the white empty area of the list, or if the list is fully populated, from the upper left corner of the list view (Figure 5 and Figure 6).

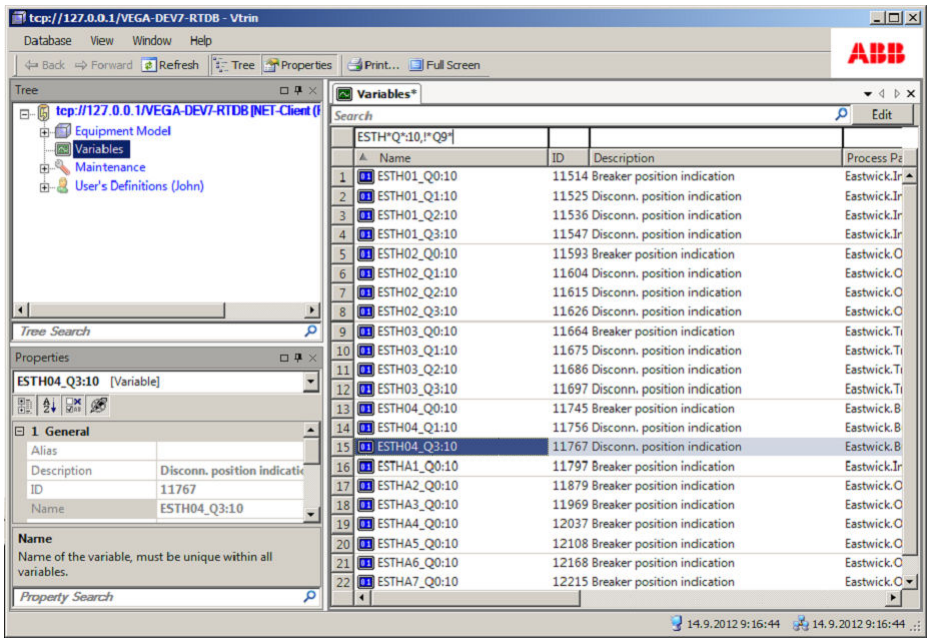


*Figure 4: Applying property filter for name property to find breaker position indications for Eastwick station. The filter used in this example is ESTH*Q*:10,!*Q9*. It shows all breaker position indications except for earth switches (Q9).*

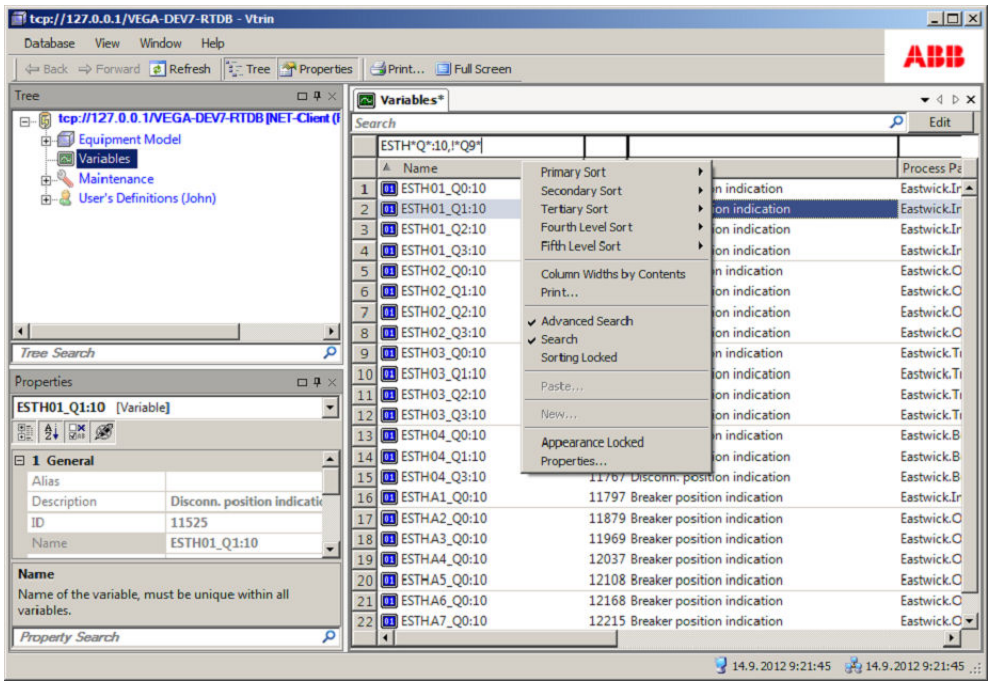


Figure 5: Selecting properties from Variables list. Note that this properties dialog is different to the Generic properties window below the tree that is only showing the properties of currently selected user interface component (Variables).

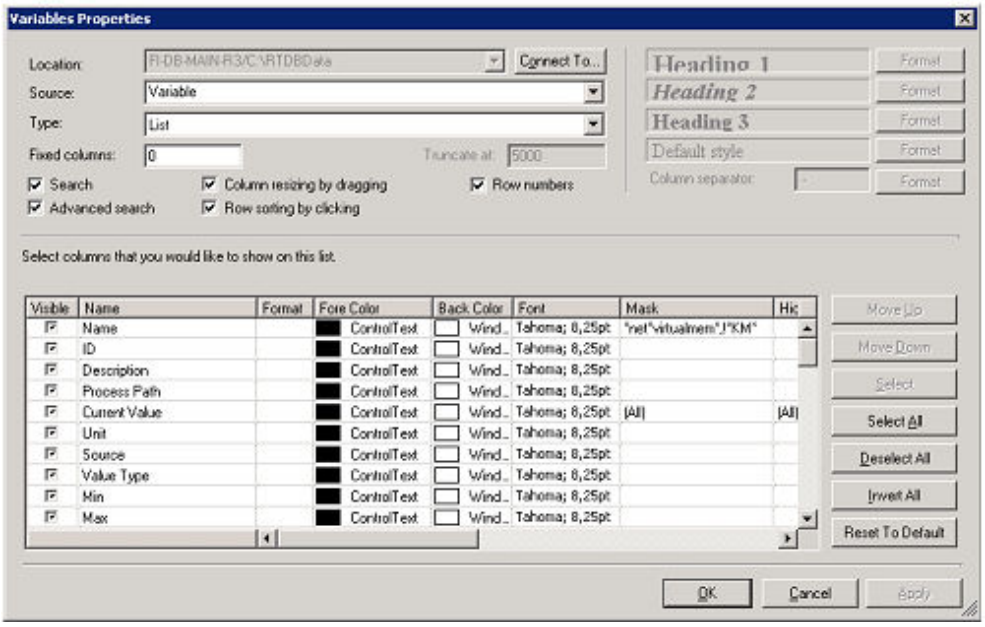


Figure 6: Properties of Variables list view. The Source drop-down has been used to define that list is populated from Variable class. The properties of Variable class are visible in lower part. All properties are not selected to be visible by default (The defaults come from the class definition).

3.4 Tree view

3.4.1 Default content

The tree is one of the key components available in Vtrin. It is located into upper left corner in default layout. The toolbar button **Tree** below the main menu is used to hide or show the tree.

The user interface components available from the tree are stored in the database together with everything else.

The root of the tree shows the name of the data source connected. The default content of the tree depends on the user role. For administrator users, the default content is described below. Under data source name there is Equipment Model that is empty by default. Under this, a hierarchic structure of variables and tags is represented once tag configuration is loaded into database.

Variables list displays all variables available (Section 5.1 and Section 5.6 discuss variables in detail).

Maintenance folder contains lot of useful lists for administration and the content is widely referred in this document. In this section, no details from the content is presented.

The last folder in root component is User's definitions (<user>). This is where user interface elements that should only be visible for a particular user can be created.

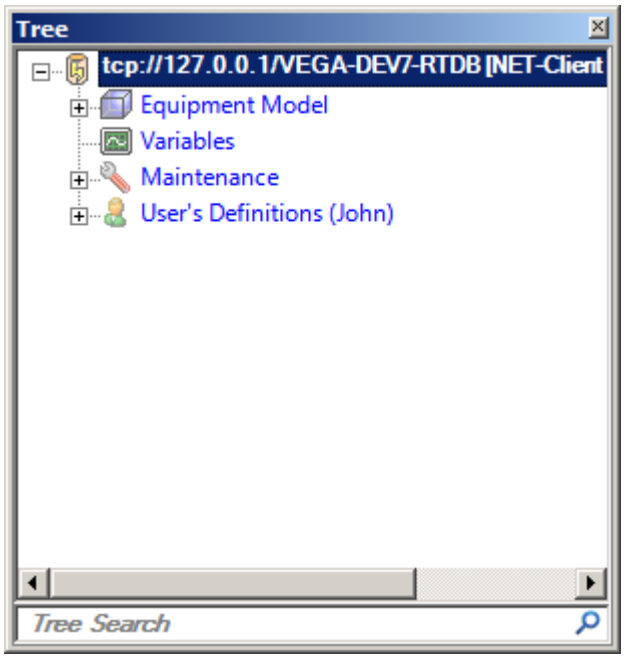


Figure 7: Default layout of tree component showing the content available for user administrator user John in database VEGA-DEV7-RTDB.

3.4.2 Functionality

Any user that has write access to database can add new components into a tree by right-clicking an element and selecting **New Tree Item**. If database connection would be done using read-only access, the color of the tree is blue and new elements cannot be added.

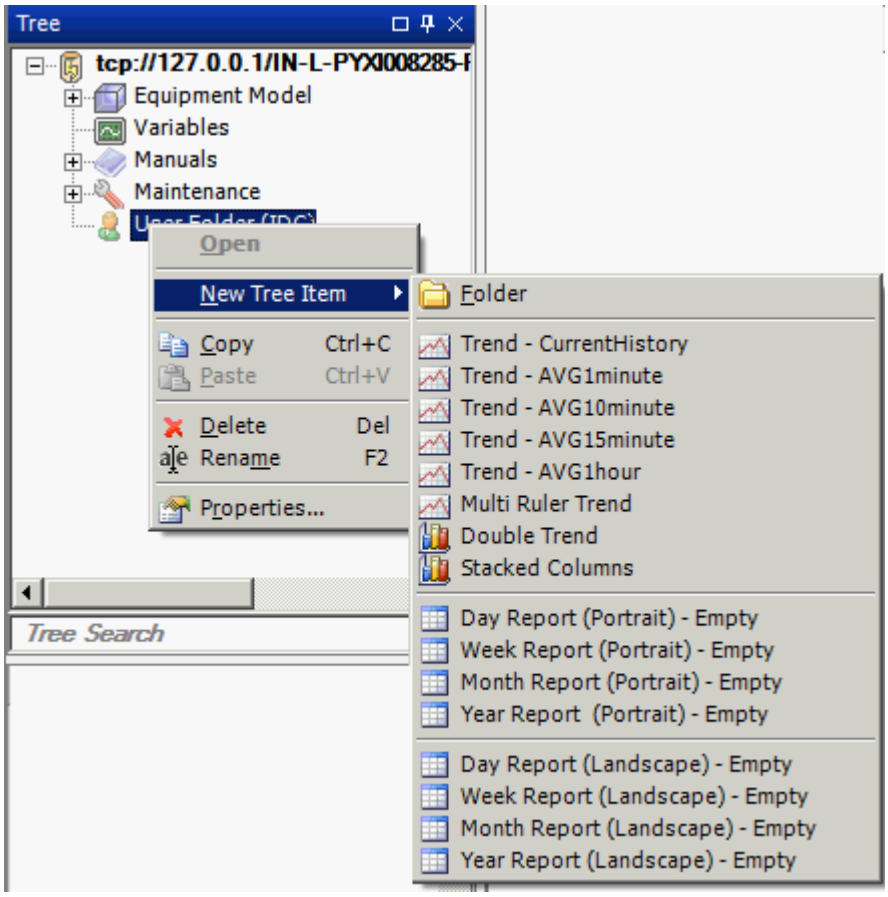


Figure 8: New elements are created from context menu of existing elements.

The content of a tree can be modified using copy-paste operations. When copy is applied to folder the content of the folder is recursively copied.

Selecting properties from the context menu of tree element shows the properties of selected user interface component.

Vtrin can be used to connect several databases by selecting **Database/Connect** from the main menu. Different database connections will be shown on top of each other in Vtrin tree (Figure 9). Disconnecting is done from context menu of root node (Disconnect).

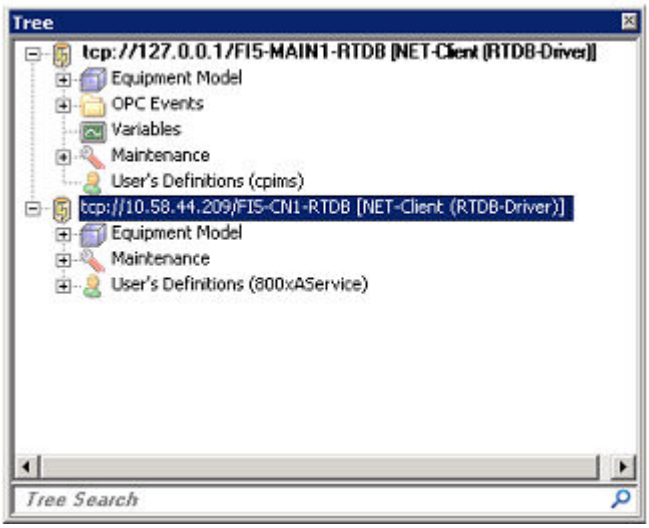


Figure 9: Vtrin connected to two databases.

There are two different kinds of mechanism to bring data to tree:

- creating new folders and UI components using context menu **New** functionality
- creating **virtual tree** from other hierarchical data in database.

Examples from virtual trees are the **Equipment Model** tree in root level and the **Component Status** tree in **Maintenance**. The Figure 10 illustrates the properties of **Component Status** virtual root.

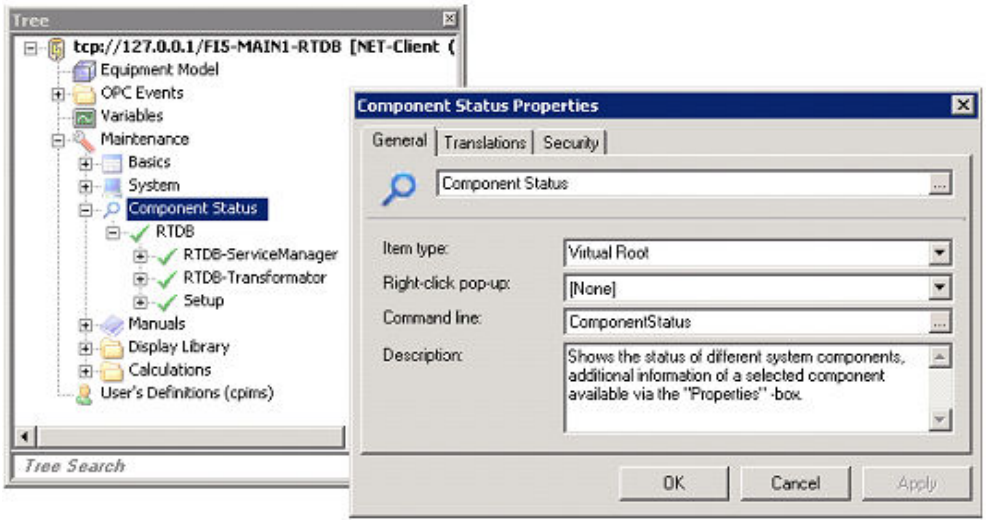


Figure 10: Properties of node Component Status showing Item type as Virtual Root. ComponentStatus is the name of the class (case sensitive) that provides data to virtual tree.

The value of the virtual tree comes from the fact that it can present all kinds of data where parent-child relationship is available. In other words, the class should contain the property Parent that is a type of class reference (to itself) and id property that provides the unique identifier within this database.

A very valuable feature in tree is Tree Search in the bottom of the control. It can be used to filter tree and find components fast without having to know the exact location. Filter is used by writing text into the search area. Search is looking for the string available as any part in the name of the component.

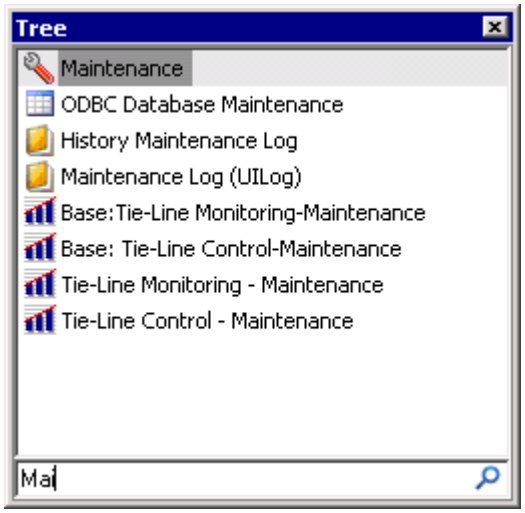


Figure 11: Tree search used to display component where name includes Mai string. Note that the name does not have to begin with that string.

3.5 Generic properties window

The properties view shows the properties of the object that is currently selected. This object can be either GUI element or actual process data stored into database (Figure 12). Many times there are quite many properties and it is advisable to undock the window. The buttons in the top toolbar can also be used to understand the content easier (from left to right):

- Categorize the content
- Sort alphabetically
- Show selected properties only (Default visibility is defined in class definition)
- Include read-only properties

Property search is the undermost component in the property display. It is used simply by starting to write a property name in the search field, and the content of the properties are filtered using the search string provided.

Properties view is useful compared to List view if the number of properties is large, because it lets properties to be aligned on top of each other. Using list display would often require scrolling the display horizontally.

It is also possible to write values to database using the properties window. This requires switching to Edit mode from the upper right corner of the Vtrin. This is similar to what would be done when feeding the values from list display. After, the writable properties are enabled to accept new values. The write operation is completed again from the upper right corner by clicking the **Commit** button and accepting the changes (Figure 13). The **Revert** button can be used to cancel the operation and return back to Normal mode.

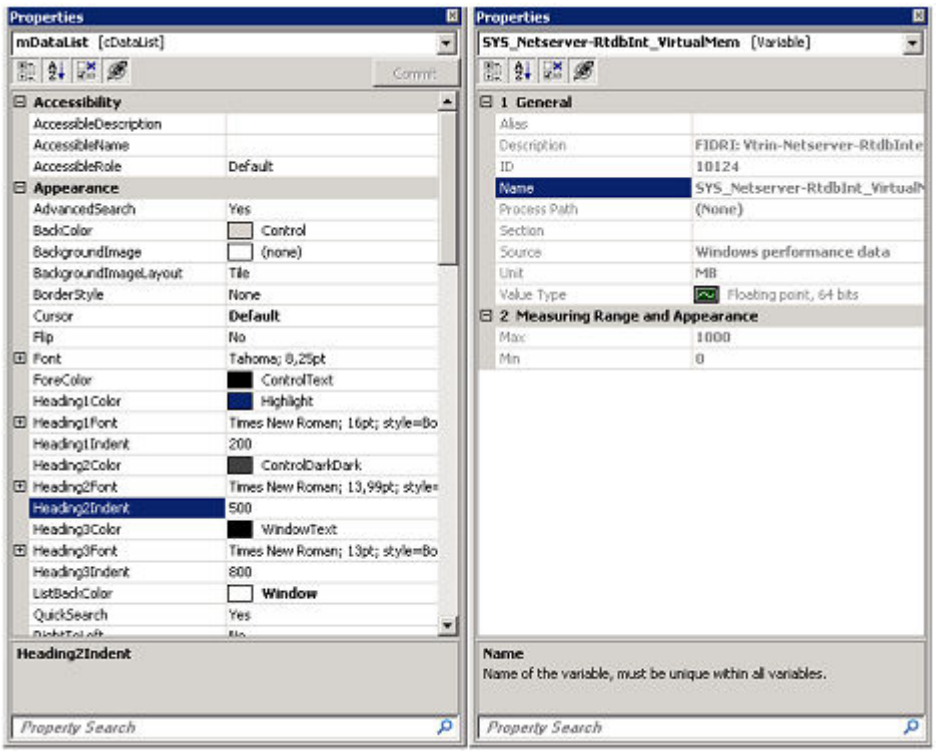


Figure 12: Properties of the list display containing variables (left) vs. properties of the Variable instance SYS_NetServer-RtdbInt_VirtualMem (right).

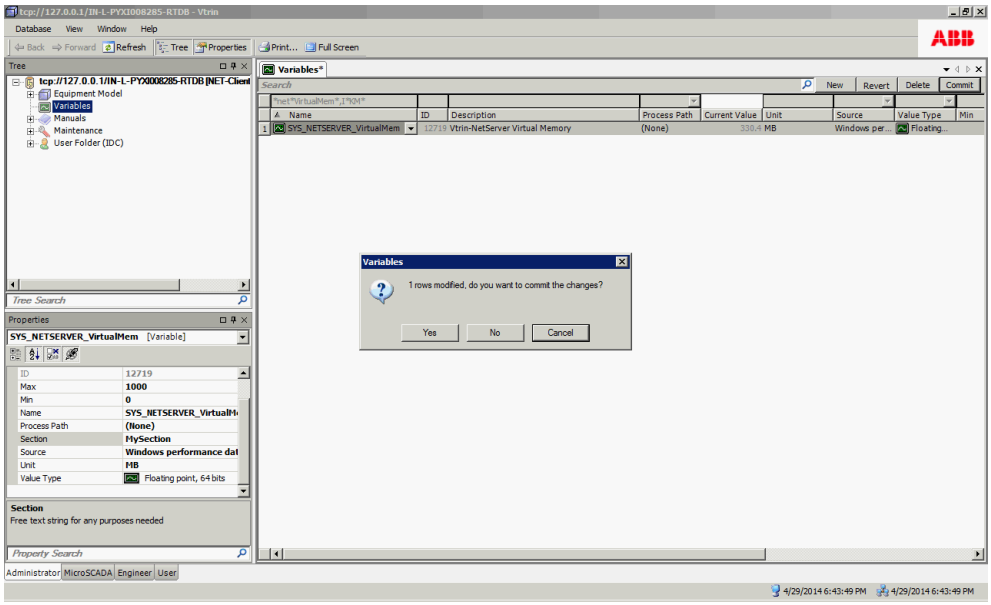


Figure 13: Setting Section property a new value MySection using the properties window.

3.6 Access control

SYS600 Historian installation will create four local user groups in Windows:

Table 1: SYS600 Historian default user groups

Groups	Description
RTDB-admin	All rights to the RTDB-system
RTDB-operator	Operator rights to the RTDB-system
RTDB-readonly	Read-only rights to the RTDB-system
RTDB-robots	Excluded from logging VtrinLib changes

3.6.1 Vtrin User Interface

Three layer access control can be achieved in Vtrin user interface using these groups:

RTDB-admin: Apply Vtrin administrator users (for example Administrator and MicroSCADA) as the member of this group to have full access within Vtrin user interface.

RTDB-operator: Apply Vtrin users (for example Engineer) as the member of this group to have limited access control to the tree view nodes/leafs and to have those hidden.

RTDB-readonly: Apply Vtrin users (for example User) as the member of this group to have mostly Read and Execute access rights.

Fourth group RTDB-robots is meant to be used by non-interactive software. This group member's operations are logged in a limited manner to prevent filling logs unnecessarily. Interactive users should not be included into RTDB-robots group.

3.6.2 Tree Security

Normally, all other users except Vtrin administrator can be connected to the system by Vtrin, and get read-only access to the Vtrin tree (class called TreeNode). It is normal to want to limit

which users can open the Vtrin tree. Users will need at least **Read** and **Execute** rights to tree items. This is done, for example, by first setting no rights to the TreeNode root node to Everyone, and then setting the Read and Execute rights to the individual user who can connect to system. **Security editor** tab can be found in the Properties dialog of the TreeNode.

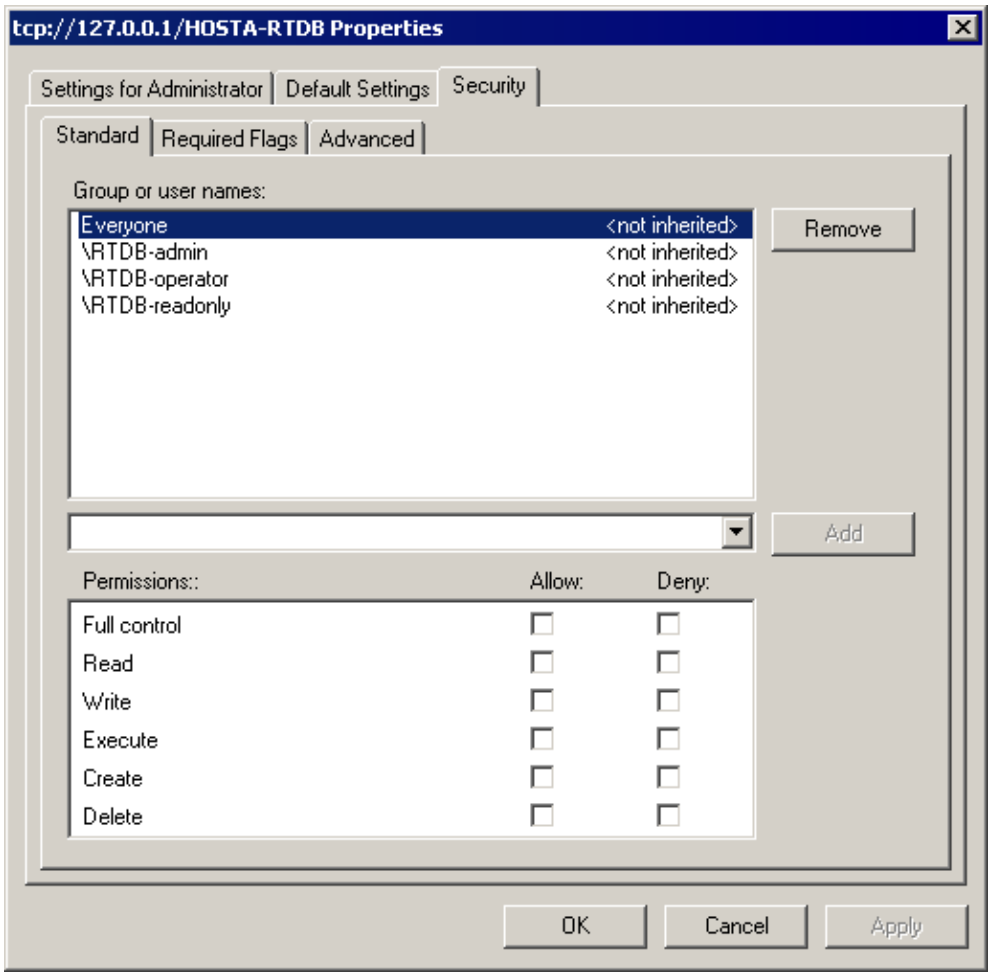


Figure 14: Security to Tree root

All other tree nodes will inherit these settings (except User Definition tree), and all tree items are available to all users (for example Maintenance folder).

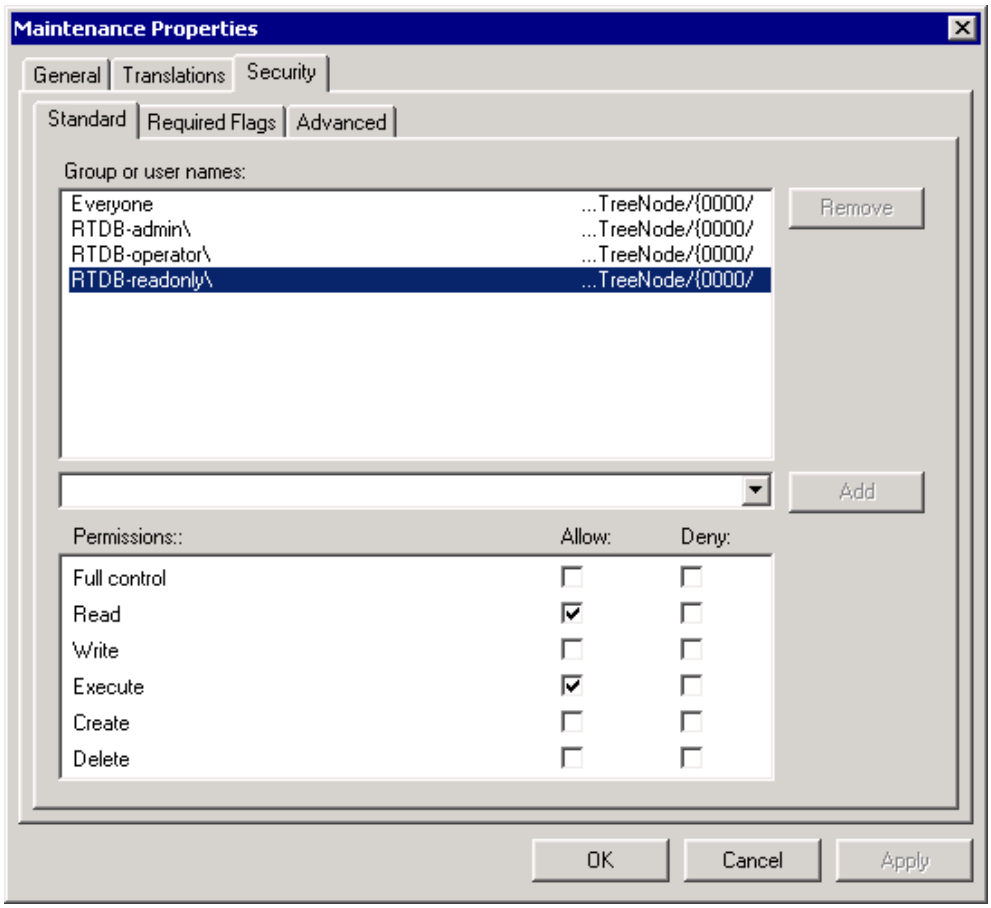


Figure 15: Inherited securities to Maintenance Tree item

In order to limit some displays from certain groups, select an upper node and set No Rights to the desired groups, for example to the Maintenance Tree item. The root definition is not inherited, and inherited definitions can thus be overwritten. Normally, for example the Maintenance folder is wanted to be visible only to limited users.

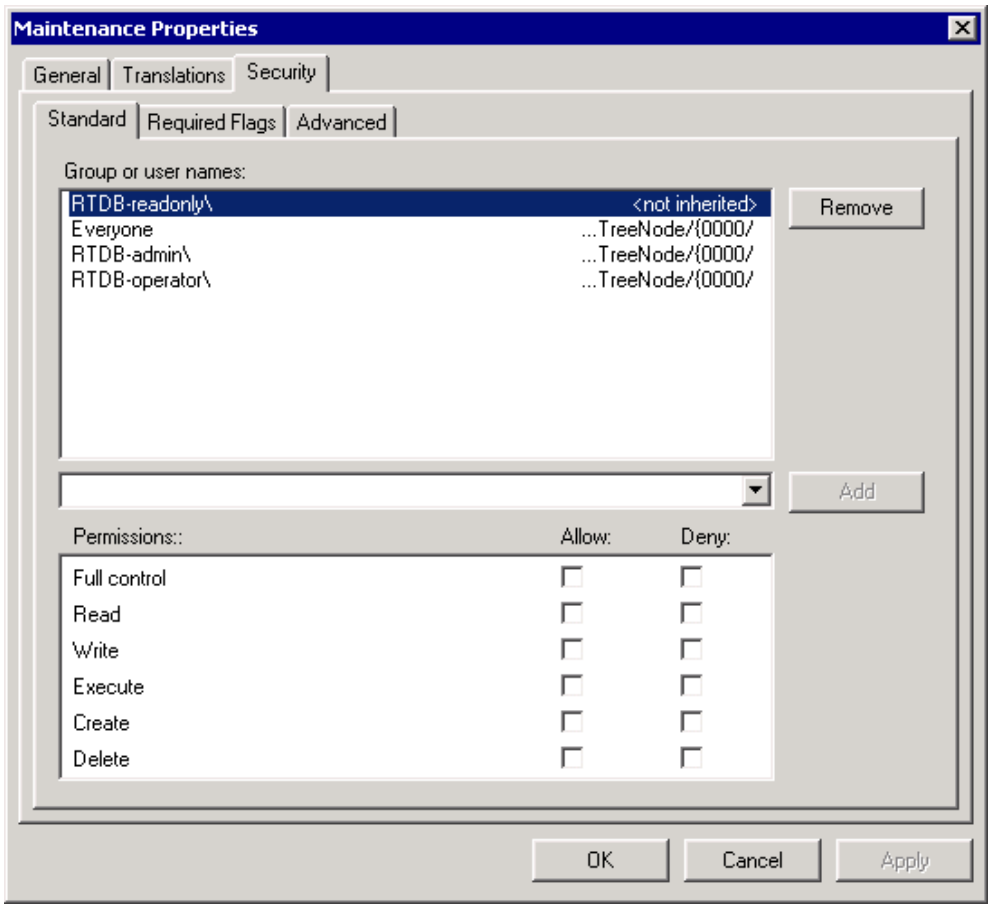


Figure 16: Instance of security definition to RTDB-readonly covers inherited definition

No right group name should match the name given to the root item, or otherwise the root definition is not overwritten. Giving no rights to group Everyone will not work, because it will overwrite only Everyone definition to root, and other definitions will be inherited.

Normally, default trends or template displays are located under Maintenance tree node. If a user does not have rights to Maintenance node, then also the displays under it are unavailable to the user. Solution is to add **Read** and **Execute** rights to the items needed by all users under Maintenance node to user Everyone. In this case, the users cannot see it, but they can use it.

Other rights to TreeNode items:

- Write: tree node definition could be changed (name, type, …) Security editor is visible, but changes are not allowed.
- Create: new tree creation, paste tree items (must give also Write, otherwise useless).
- Delete: delete tree items.

3.6.3 Role Security

The visibility of a display can be restricted for users/groups. Each role can be accessed via the Role tab in the lower left corner of the screen.

Figure 17: Vtrin Role tabs

For example, the role Engineer shall be visible to Vtrin Administrators and Engineers and shall not be visible to other users. User-Engineer is associated with the group RTDB-operator. The

Security Properties of Engineer role shall at least allow **Read** and **Execute** permissions for RTDB-admin and RTDB-operator groups and, on the other hand, Everyone and RTDB-readonly groups shall allow no permission.

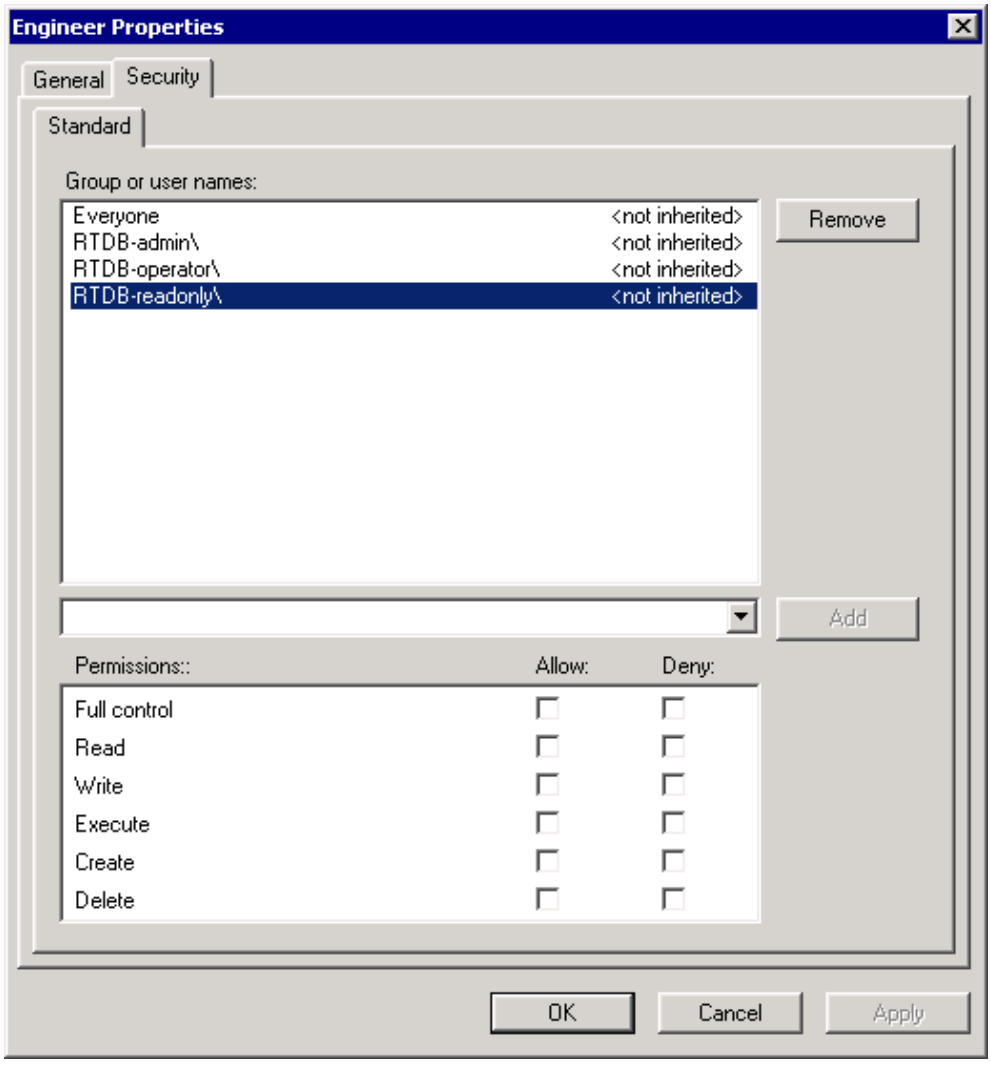


Figure 18: Engineer Role security with no permissions for RTDB-readonly

3.7 Roles

The roles are used for storing the state of the workspace for different users or roles. The state includes selection of visible displays as well as their mutual positioning on the screen. A role can be, for example, Engineer role or User/Operator role describing the tasks intended for certain group of users.

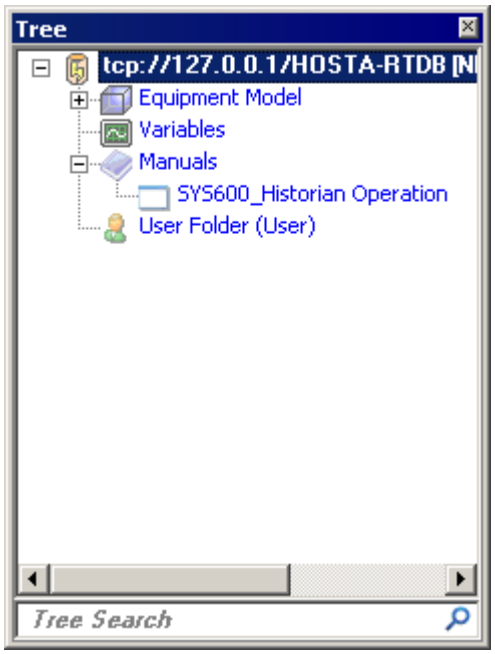


Figure 19: Display visible for role-User

Each role can be accessed via the Role tab in the lower left corner of the screen.

Figure 20: Available Roles

To create a new role, right-click an existing role tab and select **New Role** from the pop-up menu. A dialog will appear prompting for a name for the new role.

The available roles are:

1. Administrator
2. MicroSCADA
3. Engineer
4. User

To create New Role or Delete Role, logon as Administrator.

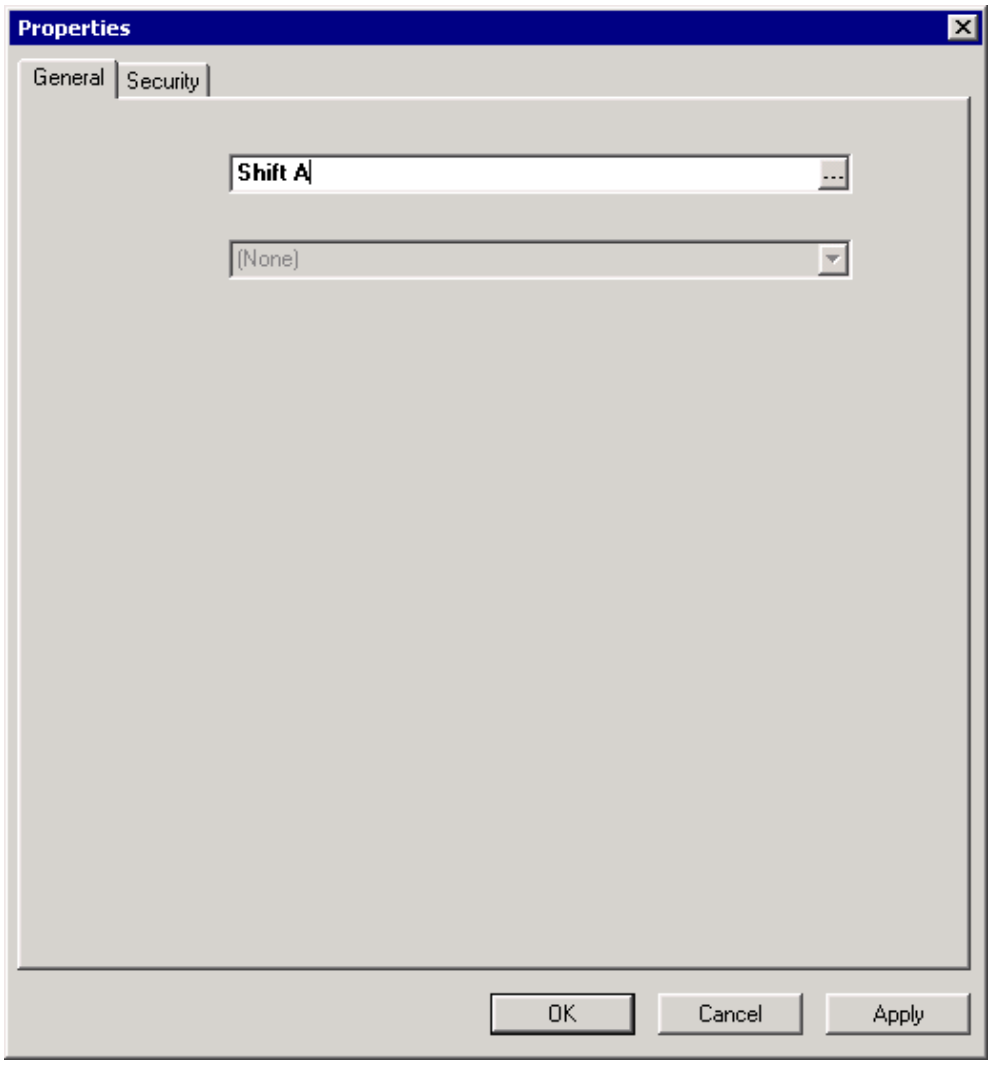


Figure 21: New Role Properties

Enter the new role name into the field and click **OK**. A new role tab will appear in the lower left corner with the name provided. Clicking the tab opens an empty window area. New content can now be added for the new role.

Remember to save the layout of the role once it has been defined. The layout is saved with the **Save Layout** command found in the View menu.

Any changes in the role's layout can be stored as the default layout for the role using the **Save Layout** command. If the command is not taken, the default layout will not be changed and it will be used the next time the role is accessed after Vtrin is restarted.

To delete a role select **Delete Role** from the aforementioned pop-up menu. This erases the layout associated with the role and removes the role tab.

3.8 Text translation

All texts used by SYS600 Historian user interface (Vtrin) are stored in Historian database. To access these texts, in Vtrin Tree **Maintenance/System/Vtrin/Language Strings**. Clear all filters fields and select the filter **(All)** for Language (Figure 22).

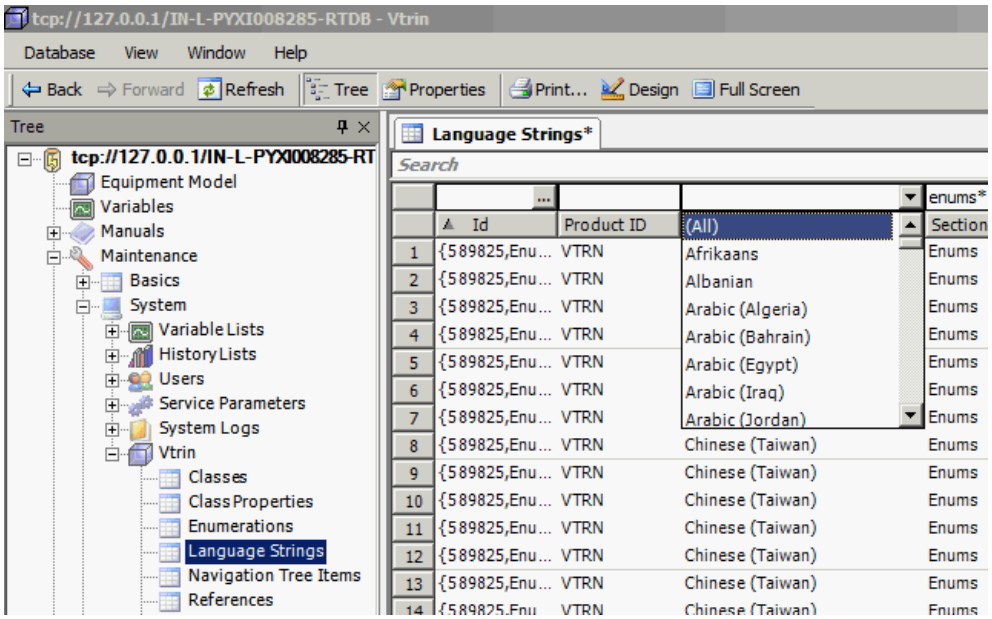


Figure 22: Language Strings in Vtrin

Type in the text that you need to translate in **Text** filter field (Figure 23)

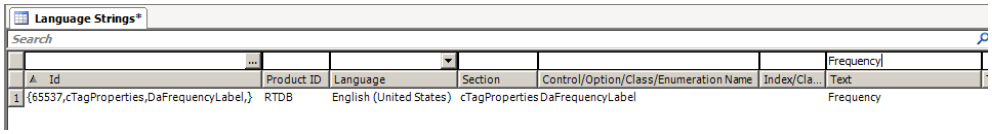


Figure 23: Text selected for translation

CopyProduct ID, Section, Control/Option/Class/Enumeration Name of filter field. Remove Text filter. Add new row **Edit/New**. Fill at least Product ID, Language, Section, Control/Option/Class/Enumeration Name and Text fields. Choose Modified by End User. **Commit** and **Revert** when finished (Figure 24).

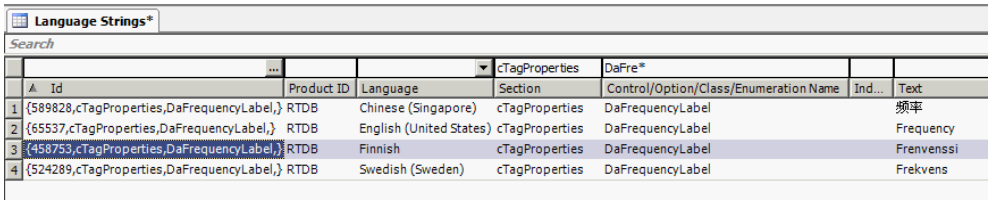


Figure 24: Texts translated

Section 4 Advanced Configuration of the Data Collection

This section contains advanced information about the configuration of the data collection. The basic configuration is done using the SYS600 user interface. The configuration is then automatically propagated to the SYS600 Historian. This section contains information for fine tuning and understanding the configuration done to the SYS600 Historian. This information is not necessary for normal use of the system.

4.1 Tag and relationships between related classes

Tag is an interface class that is used to configure the database to collect and store process data. Usually, no other classes need to be used. The purpose of this section is to introduce tag class and the related functionality in detail, as well as show how different tools can be used to create tags and start collecting process history. When SYS600 Historian is configured, the tags are created automatically. However, it is possible to add additional tags or add more configuration data to the automatically configured tags.

Tag class is used to define:

- How the data should look like in the data base. The name and unique id for these data structures. Defines the data type of stored data. Currently supported data types are double, int64 and string. From numerical data types there are two variants: discrete and linear.
- Where data is collected originally.
- How data is further processed, e.g. compressed and preprocessed before storing to database.
- What kind of upper level histories (secondary logs) are collected from raw history (current history). An example of upper level histories would be averages calculated over one minute (AVG1minute), one hour (AVG1hour) and one day (AVG1day).

Some general properties of Tag class:

- Only unique identifier of tag is Id property (the type of Id is GUID). This makes it possible to create tags in distributed environment without the possibility of name clashes.
- The creator of the Tag can propose a name for the tag. If this name proposal is unique within the local node, it is accepted as is. Otherwise, a renaming algorithm is used to ensure name uniqueness. Name of tag can be changed later on by setting new ProposedName. DisplayName property shows the actual name after the uniqueness algorithm has been applied.
- Once a tag is deleted, references to already stored data are also lost. Although historical data is not deleted, it is impossible to refer and access it. If a new tag is created with the same property values, it is still a new tag and the old history data cannot be found.
- Implementing class instances should never be deleted manually behind the tag. This breaks the Tag instance. E.g. associated variable should not be deleted alone. Deleting tag instance using Vtrin is okay, as Vtrin is able to clean associated instances and references.
- All tag instances are independent of each other.

Figure 25 illustrates the relationships between tag class and implementing classes.

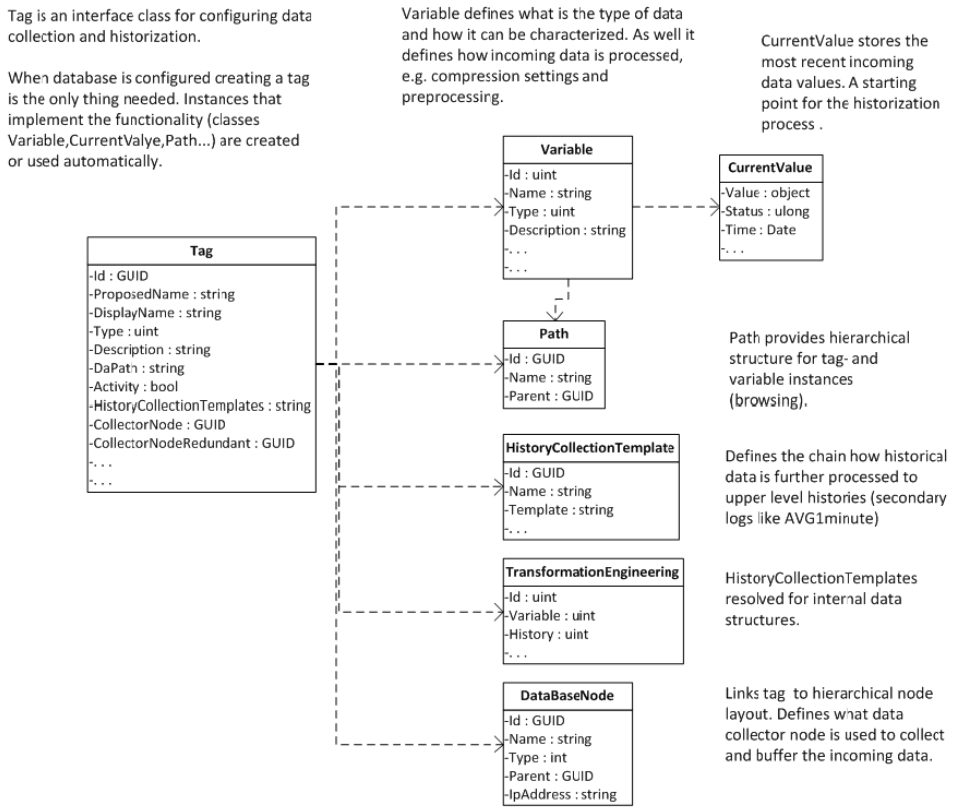


Figure 25: Figure Logical view of Tag class used to configure the database. Relationships between Tag and implementing classes are illustrated using dotted line. Selected set of properties are shown for each class. The full list of the properties in Tag class is presented later on in the table.

The complete definitions of all the properties in Tag class is presented in the table below. The properties are presented in the same order than they are found from Vtrin Tag dialog, and the categorization is based on the tabs available in Vtrin dialog.

Table 2:

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
General						
Id	in/out	GUID	no	no	The only unique id for a Tag. Used to identify tags in hierarchical system.	Usually, it is best to let the system generate the ID. However, for the purposes of RTDB-TagConsistency Controller, an id can be set as well.
ProposedName	in	string32	yes	yes	Proposed name for the Tag	The user may propose a name for the Tag, but because the name must be unique within one node, it might be changed automatically by the system.
DisplayName	out	string32	no	yes	Display name for the Tag. Modifyable trough the property ProposedName.	The unique name of the tag within one node. ProposedName is used as DisplayName if it is unique within the node (Derived from the Variable.Name). If the name is not unique, it is made unique by adding _1..._99 postfix, or finally by using GUID notation.
ExternalName	in	string254	yes	yes	External name for the Tag. The external name can contain any information that is valuable in order to recognize what the tag is presenting in the integrated system.	The object in SCIL syntax.
ExternalId	in	string 254	yes	yes	Another external identifier for tag.	Not used in SYS600 Historian

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ExternalTypeName	in	string 254	yes	yes	Another external identifier for tag.	Not used in SYS600 Historian
Description	in	string128	no	yes	Description text	Derived from Variable class.
Type	in	int32	no	no	Data type	0=float 64, 1=int64, 2=binary 64, 3=text, 4=float 64 discrete
Source	in	uint32	no	yes	Where the input data is originated.	Derived from Variable class.
DataFlowDirection	in	int32	no	yes	Defines if the tag is used to collect data into database [in] (0) or is it used to write values into external data source [out] (1).	
DisplayFormat	in	string32	no	yes	Formatting of the variable value.	For example, 0.0 means that the format will have one digit after the decimal point, whereas 0 means the format consists of a whole number. (Derived from Variable class.)
Unit	in	string16	no	yes	Engineering unit of the tag.	(Derived from Variable class)
SymbolGroup	in	uint16	no	yes	Number of the symbol group that the symbol boxes use to display the binary value. A symbol group could e.g. contain pictures of open and closed valves drawn in the different status colors. The symbol groups are organized as a symbol library. For binary variables only.	(Derived from Variable class)

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
BinaryTextId	in	uint32	no	yes	Binary text translation number.	For example 4=No/Yes Requires product specific mappings for Binary text ids and description strings.
EquipmentPath	in	string	yes	yes	Link between tag and equipment model.	String that uses dot separated format to defined process paths. E.g. MyPlant.MyObject1.MyObject2 Specifying this property creates new Path class instances if such a path does not exist yet. Deleting a tag does not remove the associated Path objects because they can be shared and used by various applications.
Creator	in	string64	yes	yes	Identifies what software component (or user) has created the tag originally.	
NumericalId	out	uint32	no	no	Numerical identification for the Tag inside one node.	Variable identification. Unique within one node. A cross reference between Tag and Variable instances.
DataCollection						
Activity	in	int32	no	yes	Active status of data collection. 0=inactive (default), 1=active, 2=no connection	
DaType (ProtocolType)	in	uint32	no	yes	Not used	
DaTypeInternal	in	uint32	no	yes	Not used	
DaPath	in	string254	yes	yes	Not used	

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
RedundantData Collection	in	bool	no	yes	Whether redundant data collection is used.	
DaTypeRedundant	in	uint32	yes	yes	Not used	
DaPathRedundant	in	string254	yes	yes	Not used	
DaFrequency	in	uint32	no	yes	Not used	
DaDeadband	in	float	no	yes	Not used	
DaItemPath	out	string (unlimited)	yes	no	Not used	
History collection						
HdaType	in	uint32	no	yes	Not used	
HdaPath	in	string254	yes	yes	Not used	
HdaTypeRedundant	in	uint32	no	yes	Not used	
HdaPathRedundant	in	string254	yes	yes	Not used	
HistoryCollectionTemplates	in	string254	yes	yes	Defines the chain of histories to be collected. Available collection chains are defined in the HistoryCollectionTemplate class. Multiple collection templates can be combined using the semicolon separator, e.g. AVG;MIN.	History template defines a secondary collection chain that could be, for example, AVG1minute;AVG1hour;AVG1day, which would collect three average history levels.
HistoryLevels	out	string (unlimited)	yes	no	Identifies what kinds of secondary histories are actually collected based on defined history collection templates.	Different history levels are separated by semicolon.
PrimaryLogHdaItemPath	out	String (unlimited)	yes	no	Not used	
SecondaryLogHdaItemPath	out	String (unlimited)	yes	no	Not used	
Processing						
ValueMin	in	Number (Object)	no	yes	Min engineering unit value	(Derived from Variable)

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ValueMax	in	Number (Object)	no	yes	Max engineering unit value	(Derived from Variable)
ValueDisplayMin	in	Number (Object)	no	yes	Min engineering unit value for displays	(Not used currently)
ValueDisplayMax	in	Number (Object)	no	yes	Max engineering unit value for displays	(Not used currently)
RecordingMethod (StoringMethod)	in	uint32	no	yes	Specifies how input data is stored into current history. Allowed values are 0=No Current History Storing, 1=Default Current History Storing and 2=Clustered Storing. (Clustered storing is not officially supported in current product version, but is still in prototype level.)	(Derived from Variable)
CompressionMethod	in	uint32	no	yes	Defines whether data is compressed or not when stored into current history. Allowed values are 0=No compression or 2=Quantization Lane Compression.	(Derived from Variable)
CompressionError	in	double	no	yes	Specifies the maximum level of error allowed in the compression of the current history values. The maximum error amount is given as a percentage of the range defined in ValueMin and ValueMax properties.	(Derived from Variable)
Hierarchy						

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ConsistencyControlled	in	bool	no	yes	Defines whether or not the tag is distributed in hierarchical system by RTDB-Tag ConsistencyController.	
CollectorNode	in	GUID	yes	yes	Defines which data collector node is used primarily to collect data in hierarchical system.	
CollectorNodeRedundant	in	GUID	yes	yes	Defines which data collector node is used to execute redundant data collection in hierarchical system.	Only available if RedundantDataCollection property is set to true.
SourceNode	in	GUID	yes	yes	Support for multiple main-node layers.	(Not used currently)
ConsistencyFindRoot	in	bool	no	yes	Support for multiple main-node layers. Defines whether this tag should be propagated to only to the first found main node (false) or all the way to last found main node (true).	(Not used currently)
Misc						
Version	in	int32	no	yes	Engineering version number	Version number of the engineering attributes configuration (currently not in use).
ArrayData	in	int32	no	no	Input data is an array	0=Not an array, 1=Time based array, 2=QCS profile (currently not in use)

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ArrayCycle	in	int32	no	yes	Time cycle	Default time cycle for the values in array in 100 nanoseconds for array type data. The default measuring frequency of the tag values (currently not in use).
Status	out	GUID	yes	no	Reference to status instance of the tag.	(Currently nto in use, but reserved for future use.)
RowModificationTime	out	DateTime	no	no	Last modification time of this tag instance.	RTDB-TagConsistency Controller (TCC) and RTDB-ConsistencyController (CC) use this to define what the newest changes are.

4.2 Creating tags using Vtrin

The easiest way to create a Tag using Vtrin is to use the custom made dialog. It can be opened from tag list display by right-clicking the white area and selecting **New** from the context menu, or in the Vtrin tree **Maintenance/Basics/Tag Configurations**. The dialog looks like in Figure 26. The required properties that have to be filled are marked with a red color. Nullable properties are marked with (None). There are five available tabs in the dialog. The only required property, **Proposed Name** that has no default value, can be found in the General tab. To get the first tag created, write, for example, TestTag1 into **Proposed Name** and select **OK** to confirm the creation. This closes the dialog and a new row should be visible in the Tag configuration list. After that, the dialog can be reopened by double-clicking the row, or by right-clicking the row and selecting **Properties** from the drop-down menu. The **Apply** button can be used to commit changes into database without closing the dialog. Clicking **Cancel** closes the dialog without committing the changes into database.

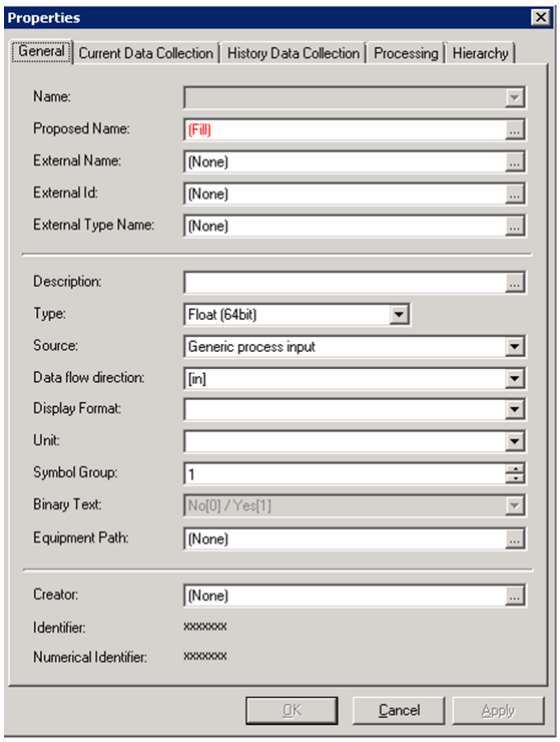


Figure 26: Tag dialog for creating and modifying tags using Vtrin.

Other alternative to creating a tag using Vtrin is to use the standard list display features in **Edit** mode (upper right corner of the tag list display). The benefits of using this instead of the custom made dialog are the ability to use multi-select features of Vtrin list, and the ability to use copy-paste to get create multiple rows at once.

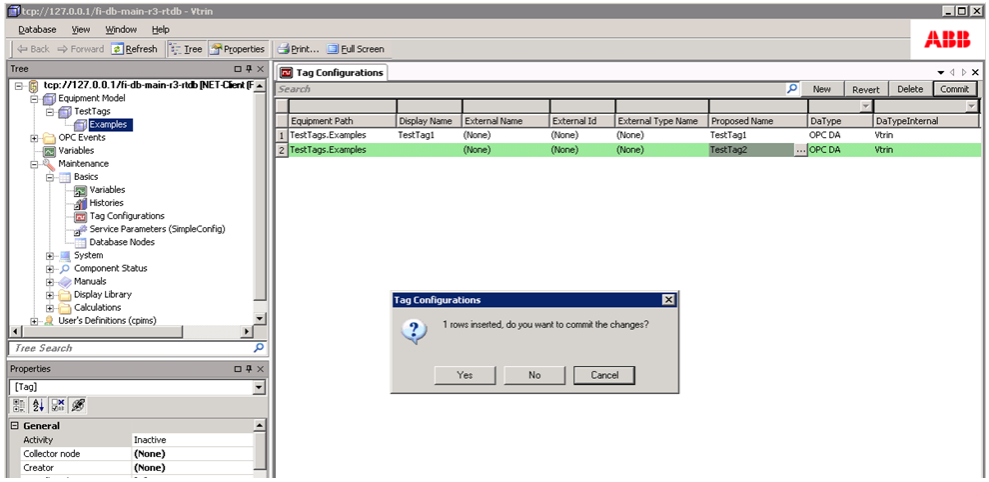


Figure 27: Creating tag TestTag2 in Vtrin list view in Edit mode. The only filled properties are ProposedName and EquipmentPath to provide structure for the tags (Note the Equipment Model in the upper left corner of the main tree).

For a complete description of properties in Tag class, see Section 4.1.

4.3 Fine tuning using Variables

As explained in previous sections, whenever a tag is created, a variable is created as well. Tag class extends Variable class with some additional functionality. In addition to just extending

the functionality of a Variable, Tag class also hides some details from Variable class. The properties of a Tag class that are derived from Variable class are visible in Table 2 (Derived from Variable class notification in the Description column). In cases when some functionalities are only available in the Variable class, the user might want to fine tune the behavior by accessing the Variable properties directly.

Variables list is available from the Vtrin tree root level or from **Maintenance/Basics**. Variables can be created using either standard list functionality (From edit mode by using **New** (and **Commit**), or by using the dedicated dialog to create it. Figure 28, Figure 29 and Figure 30 illustrate the dialog functionalities for Variables.

There are five tabs in the Variable dialog: **General**, **Processing**, **Preprocessing**, **Variable Security** and **Current Value Security**. **General** contains the basic identification of Variable. **Processing** contains definitions for compression, current value processing, substitute value handling and current history storing. The **Preprocessing** tab provides functionality to process input data before forwarding it to current value processing and current history storing. Preprocessing functionalities depend on the type of the variable. The last two tabs contain security definitions for the variable instance and the associated current value. In Vtrin data abstraction layer, there are three different mechanisms to control user access: Access control lists (ACL), Flag protection, and Owner based – Unix-style method. Possibility to configure variable access control using either ACL or flags is available in the Variable dialog. For the current value, only flag-based approach is available due to performance reasons.

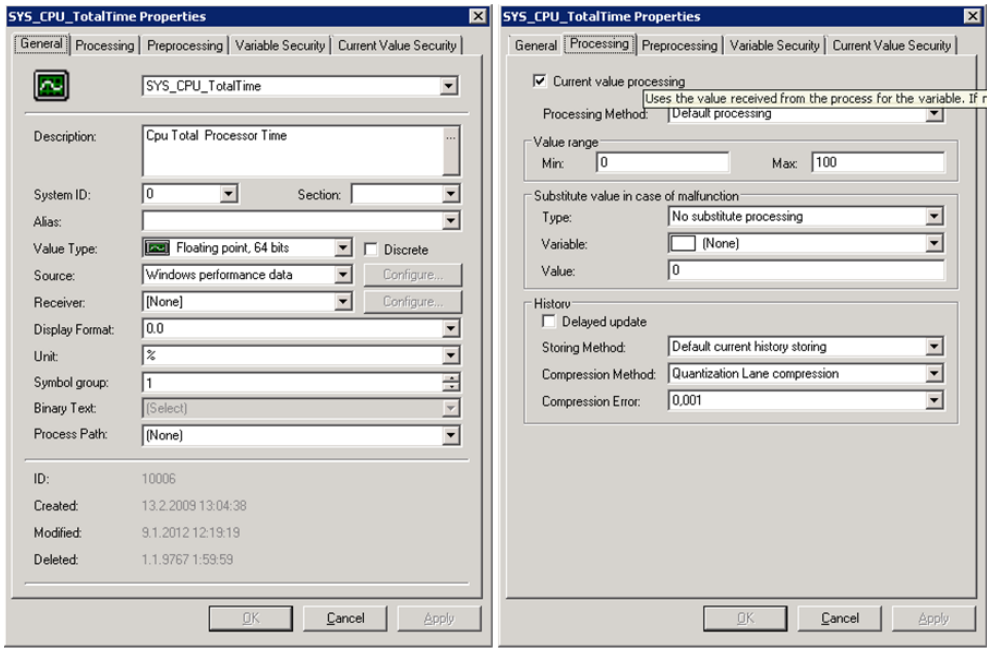


Figure 28: Variable dialog is used to create, modify or view variables in detail. General tab contains the basic identification. Processing tab contains definitions for compression, current value processing, substitute value handling and current history storing.

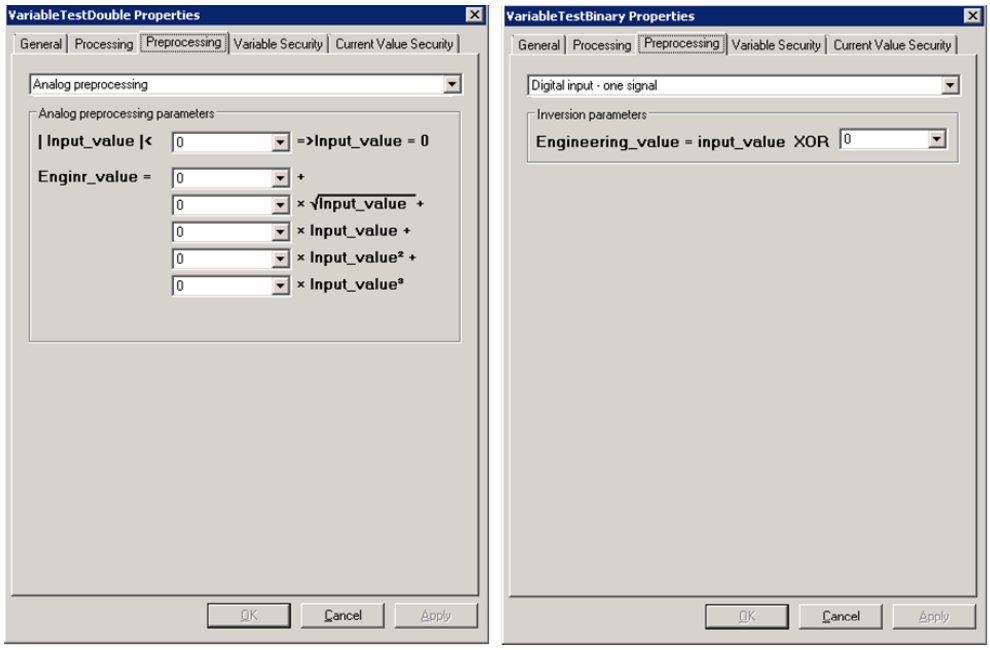


Figure 29: Preprocessing for double variable (left) and for binary variable (right).

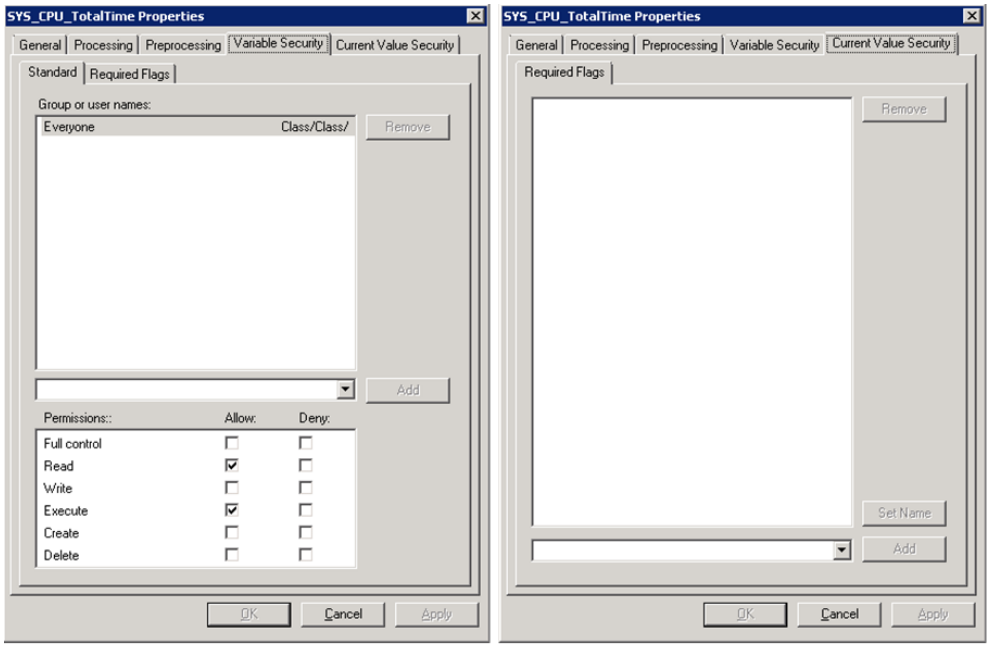


Figure 30: Security definitions for variable (left) and current value (right). ACL and Flag based access control can be applied to variable instance. ACL is not available for current value protection because of performance reasons.

Table 3 summarizes all the properties available in Variable class. Not all properties are available in custom dialog. In such cases, the list display should be used. The most important properties are available in Tag class as well, and whenever a tag instance has been created, the most frequent configuration changes can be done from that side.

Table 3: Properties of Variable class in detail.

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
General						
Id	out	uint32	no	no	The unique id for a Variable, generated by system (autonumber).	Tag instances cross reference with variables using the id of the variable (stored in property NumericalId in Tag)
Name	in	string32	yes	yes	Unique name for a variable inside this database.	DisplayName property of the Tag class presents the same thing. When a variable is part of tag, the Name should not be modified directly, but by using the ProposedName tag instead.
Description	in	string128	no	yes	Description of the variable.	Same as Tag property Description.
Unit	in	string16	no	yes	Engineering unit of the variable.	Same as Tag property Unit.
Section	in	string32	no	yes	Content is system specific.	
Alias	in	string32	no	yes	Secondary name for the variable.	
DisplayFormat	in	string32	no	yes	Formatting of the variable value.	For example, 0.0 means that the format will have one digit after the decimal point, whereas 0 means the format consists of a whole number. Same as the Tag property DisplayFormat.
Source	in	uint32	no	yes	Specifies the system where the input data originated.	Same as the Tag property Source. The default value is Generic Process Input. Other much used values are Calculated and Manually Entered.

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ValueChange-Receiver (Receiver)	in	uint32	no	yes	Specified the system where the data is sent into.	Receiver identification of a value changer notification, for example to an output value in data acquisition system. The SendOutputRequest operation sends a command with the syntax DoOutput;-IDT=name:VALUE=value to the CommandQueue with this receiver id number. By using special configuration with Source and SendOutputMode variables, current value production can also trigger sending the output request.
Type	in	int32	no	yes	Data type. Allowed values are Floating Point 64bit, Integer 64bit, Binary 64bit, Text	Tag property Type combines the Variable properties Type and Discrete.
DiscreteValue	in	bool	no	yes	False if the value of the variable is interpreted as linear between data values. True if the value is interpreted as discrete.	Also affects the upper level history aggregation calculations, not only the presentation of data.
ProcessArea	in	uint64	no	yes	Stores information where the actual signal is located in process.	
SystemId	in	int32	no	yes	OBSOLETE	OBSOLETE

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
SymbolGroup	in	uint16	no	yes	Number of the symbol group that the symbol boxes use to display the binary value. A symbol group could e.g. contain pictures of open and closed valves drawn in the different status colors. The symbol groups are organized as a symbol library. For binary variables only.	Same as Tag property SymbolGroup.
BinaryTextId	in	uint32	no	yes	Binary text translation number.	For example 4=No/Yes Requires product specific mappings for Binary text ids and description strings. Same as Tag property BinaryTextId.
UserStatusId (Aplication status text)	in	uint32	no	yes	UI support to display status (quality) of current value in clear text format.	User status list identification. Variable specific text equivalent to the user status value (the CUS column in CurrentValue or the HUS column in history tables). The status numbers are enumerated in the Enums section of the UIStrings table with the key User Status Translations, and the actual texts in the Enums section with the key User Status Translations(n), where n is the user status id number (for examples, see %RTDBRoot% \Config \RTDB_UIStrings_Insert.sql).

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
Path	in	string	yes	yes	Link between a variable and equipment model.	Defines same thing as EquipmentPath in Tag class. The difference is that Path is a reference to an existing Path instance and cannot be used directly to create paths.
ApplicationType	in	int32	yes	yes		
Created	out	DateTime	no	no	Creation time of the variable.	
Modified	out	DateTime	no	no	The last time the variable has been modified.	
Deleted	out	DateTime	no	no	Time of when the variable has been deleted from database.	When variables are deleted, they are only marked as deleted (by prefixing with ~) and hidden from view by Vtrin. The variables are actually deleted the next time database is restarted.
Processing						
CurrentValue	in/out	object	no	yes	Current value (real-time value) of variable.	Visible in the variable list and in a separate dialog when the variable in double-clicked on the variable list.
ValueMin	in	Number (Object)	no	yes	Min engineering unit value	Same as Tag property ValueMin. Note that if compression is enabled (as it is by default), the values of ValueMin and ValueMax affect the compression result.

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ValueMax	in	Number (Object)	no	yes	Max engineering unit value	Same as Tag property ValueMax. Note that if compression is enabled (as it is by default), the values of ValueMin and ValueMax affect the compression result.
Connected (Current value processing)	in	bool	no	yes	True (default) when current values are normally processed. If set to false, a substitute value is used instead.	
ProcessingMethod	in	uint32	no	yes	0=Default processing or 1=Pure value storing. Default is 0 when value is processed as defined. In pure value storing, the value is stored into database without applying processing.	
RecordingMethod (StoringMethod)	in	uint32	no	yes	Specifies how input data is stored into current history. Allowed values are 0=No Current History Storing, 1=Default Current History Storing and 2=Clustered Storing. Clustered storing is not officially supported in current product version, but is still in prototype level.	Same as Tag property RecordingMethod.

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
CompressionMethod	in	uint32	no	yes	Defines whether data is compressed or not when stored into current history. Allowed values are 0=No compression or 2=Quantization Lane Compression.	Same as Tag property CompressionMethod.
CompressionError	in	double	no	yes	Specifies the maximum error level allowed in the compression of the current history values. The maximum error amount is given as a percentage of the range defined in ValueMin and ValueMax properties.	Same as Tag property CompressionError. Value range is from 0.0 to 1.0. For linear variables, the actual maximum error can in some cases be twice the value. Therefore if the user wants to have absolute maximum error of 1 unit when the measurement limits are 0..100, compression error need top be set at 0.005.
SubstituteMethod	in	uint32	no	yes	Defines if the substitute value handling is used, as well as which kind of handling is performed.	Allowed values are 0=No substitute processing (default), 1=Substitute value, 2=Variable, 3=Variable if valid, otherwise value, and 4=Bad range limit.
SubstituteValue (Value)	in	object	no	yes	If SubstituteMethod is SubstituteValue or Variable if valid, otherwise value, the SubstituteValue is used as a substitute.	

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
SubstituteVariable (Variable)	in	ref var	yes	yes	If SubstituteMethod is Variable or Variable if valid, otherwise value, SubstituteVariable specifies the value used as a substitute.	
Preprocessing						
Preprocessing Method	in	uint32	no	yes		
NoiseFilterFactor	in	double	no	yes		
NoiseFilterEnable	in	bool	no	yes		
AnalogPower0	in	double	no	yes		
AnalogPower1	in	double	no	yes		
AnalogPower2	in	double	no	yes		
AnalogPower3	in	double	no	yes		
AnalogSquareRoot	in	double	no	yes		
AnalogZeroClamping	in	double	no	yes		
Inverted	in	int64	no	yes		
PulseToEngineering	in	double	no	yes		
PulseLowLimit	in	int64	no	yes		
PulseHighLimit	in	int64	no	yes		
PulseEngineerinVariable	in	reference (uint32) to variable instance	yes	yes		
PulseDerivatedVariable	in	reference (uint32) to variable instance	yes	yes		
PulseWeight	in	double	no	yes		
PulseDerivatedCoefficient	in	double	no	yes		
PulseSensorVariable	in	reference (uint32) to variable instance	yes	yes		
StatisticsLimitFunction	in	reference (uint32) to variable instance	no	yes		
Misc						

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ExternalName	in	string128	no	yes	Name in an external system.	This is different from the ExternalName property in Tag class.
CalcActivation	in	int32	no	yes	Reserved, but not in current product scope.	
PreventDisableOnInit	in	bool	no	yes	Value true (1) means that SYS600 Historian will not set the value to invalid during SYS600 Historian restart. This is not needed for manual entered variables because they are never set to invalid at restart.	
SendOutputMode	in	int32	no	yes	Determines the situations when current value production triggers the SendOutputRequest operation. The default value is 0/ ManualEntryForProcessOutput, which means that the trigger is done only if the Source attribute of the variable is ProcessOutput and a manual entry current value is produced. 1/ Always means that the output is triggered always.	

Table continues on next page

Property	In/Out	Type	IsNullable	IsModifyable	Description	Comment
ProducerTable	in	uint32	no	yes	The history table number (HTID) of the table that is used as the origin of the data for this variable. Zero means that the current values are produced as usual.	For example, the number of AVG1hour means that the data is produced to one hour average level directly. This column is only meant to be used by UI programs that want information about the variables where data is not produced using current value mechanism.
SecurityMask	in	int32	no	yes	Place to store flag-based access control information.	
VariableRowId	out		yes	no	Programming support.	
CurrentValueId	out		yes	no	Programming support.	

Section 5 Process history in detail

The following sections describe what kinds of mechanisms there are for configuring how numerical process data is stored into process history tables. Although the configuration is easy through Tag interface class, it is valuable to understand some details as well, especially in situations where default configuration will not meet the requirements well enough.

5.1 Related classes

Figure 31 illustrates the classes related to history configuration. Tag class is an interface class that is used to configure the database. Instances of Variable and TransformationEngineering classes are created automatically when a Tag instance is created and its lifetime is bound to the lifetime of a Tag instance (composition). Tag and Variable classes have been presented in detail in the previous sections.

HistoryCollectionTemplate class defines what kinds of secondary logs are collected for each tag. By default, history collection templates are defined for Average (AVG), Sum of Values (SUM), Minimum (MIN), Maximum (MAX), Deviation (DEV), First Value (FVA), Last Value (LVA), Operating Time (OPT), Startup Count (CNT) and Forecast Averages (FOR).

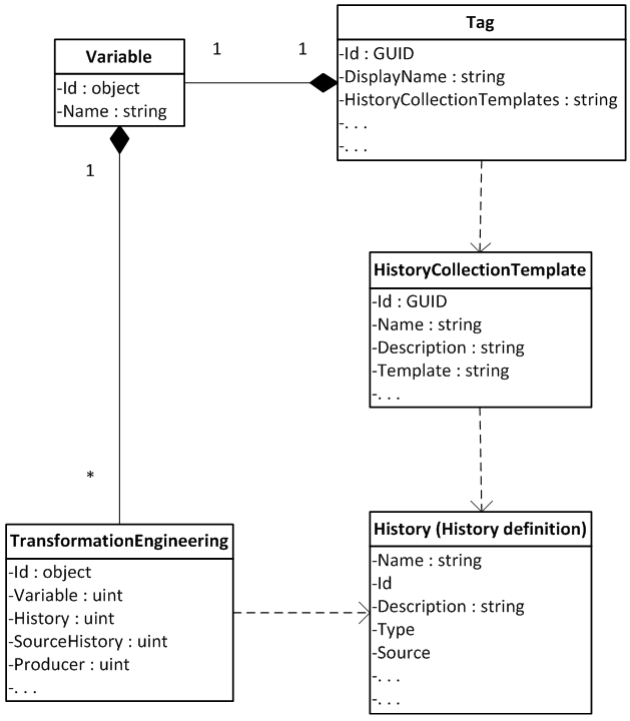


Figure 31: Relationships between classes that define how numerical data is historized into database tables.

The Tag class property HistoryCollectionTemplates is used to configure which history collection templates (zero to N) are applied, i.e. defined to be collected when the tag begins to receive values from an external data source.

Each history collection template defines the chain of Histories using the property Template. The default templates consist of three time periods: 1 minute, 1 hour and 1 day. 1 minute

history is collected from raw history (Current history), 1 hour history is collected from 1 minute history and 1 day history is collected from 1 hour history.

Finally, the History class defines the kinds of physical history tables available in the database. These histories are referred from HistoryCollectionTemplate class by Name. Each history definition defines exactly how data is collected. More detailed information about the HistoryCollectionTemplate and History classes is provided in the following sections.

5.2 Using History collection templates

The list of History collection templates is found in Vtrin in **Maintenance/Basics**. There are also displays for History Tables (History class) and History Collections (Transformation Engineering class).

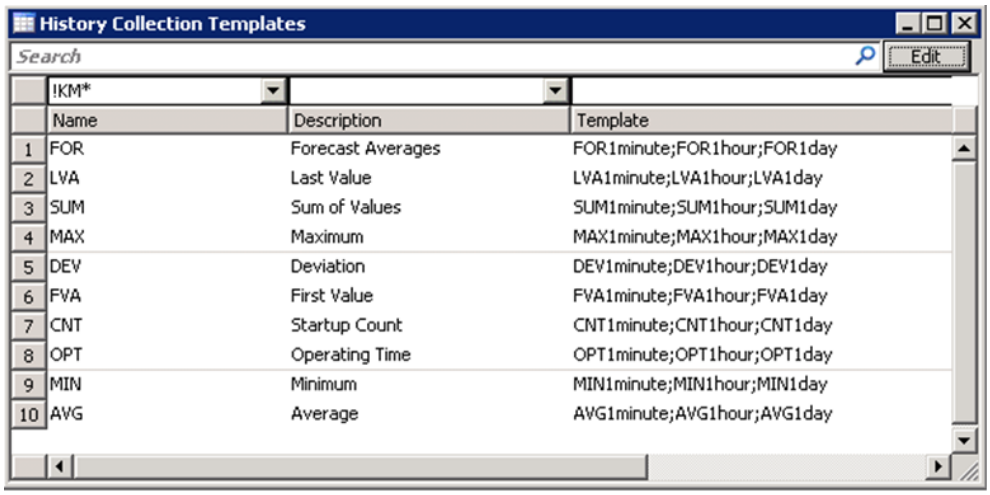
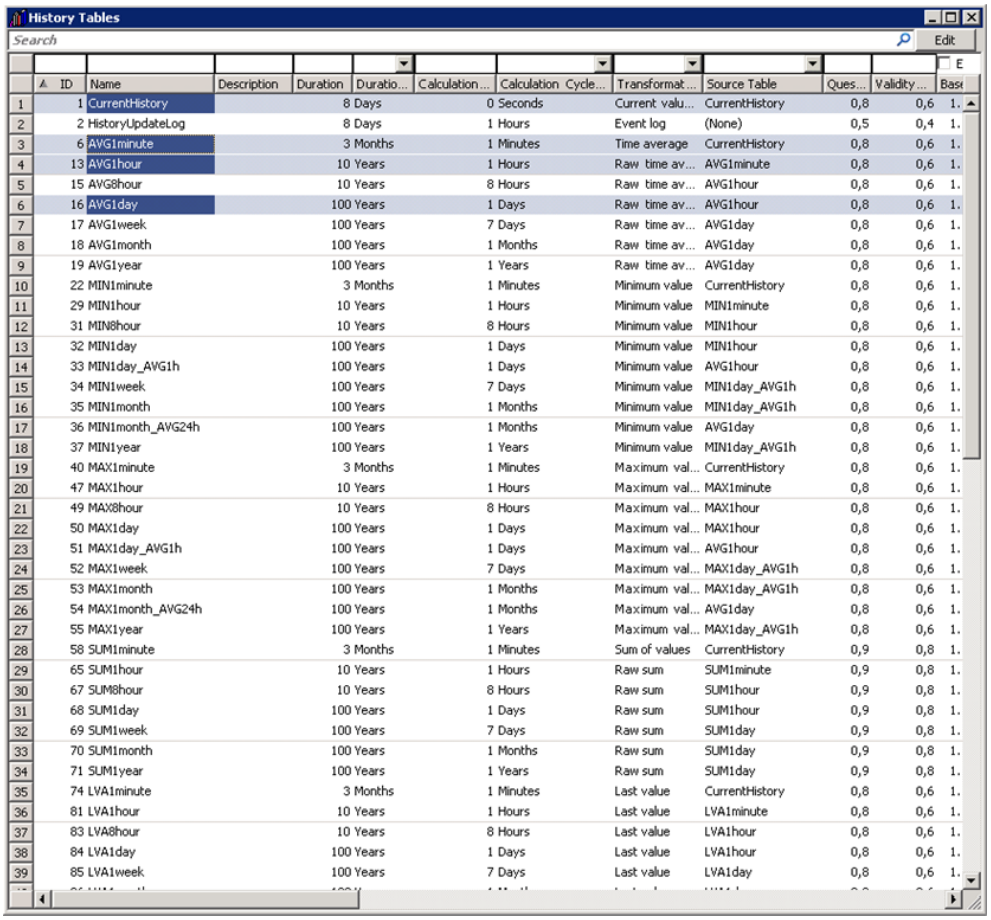


Figure 32: History collection templates that are available by default after install. Note! KM Name property filter that excludes History collection templates that are available to support Knowledge Manager based applications.*

New history collection templates can be created using Vtrin if the templates available by default do not meet all the requirements. When new history collection templates are created, each one is identified uniquely using an Id (GUID) (automatically generated, not visible by default in the list view). The Name of the template is unique as well. The content of Template property is a semicolon-separated list of histories that are collected when the history collection template is applied to a tag instance. The list of the available histories is available in the History tables display. It is important to ensure is that the chain of histories can to produce data, i.e. the history chain is such that each source table is available in the chain. Check the SourceTable property in History Tables list. An example of a chain that is not working would be an AVG_BAD template where the Template chain would be AVG1minute; AVG1day, because AVG1day data is collected from AVG1hour table, which is not included in chain.



	ID	Name	Description	Duration	Duratio...	Calculation...	Calculation Cycle...	Transformat...	Source Table	Ques...	Validity...	Bas
1	1	CurrentHistory		8	Days	0	Seconds	Current valu...	CurrentHistory	0,8	0,6	1.
2	2	HistoryUpdateLog		8	Days	1	Hours	Event log	(None)	0,5	0,4	1.
3	6	AVG1minute		3	Months	1	Minutes	Time average	CurrentHistory	0,8	0,6	1.
4	13	AVG1hour		10	Years	1	Hours	Raw time av...	AVG1minute	0,8	0,6	1.
5	15	AVG8hour		10	Years	8	Hours	Raw time av...	AVG1hour	0,8	0,6	1.
6	16	AVG1day		100	Years	1	Days	Raw time av...	AVG1hour	0,8	0,6	1.
7	17	AVG1week		100	Years	7	Days	Raw time av...	AVG1day	0,8	0,6	1.
8	18	AVG1month		100	Years	1	Months	Raw time av...	AVG1day	0,8	0,6	1.
9	19	AVG1year		100	Years	1	Years	Raw time av...	AVG1day	0,8	0,6	1.
10	22	MIN1minute		3	Months	1	Minutes	Minimum value	CurrentHistory	0,8	0,6	1.
11	29	MIN1hour		10	Years	1	Hours	Minimum value	MIN1minute	0,8	0,6	1.
12	31	MIN8hour		10	Years	8	Hours	Minimum value	MIN1hour	0,8	0,6	1.
13	32	MIN1day		100	Years	1	Days	Minimum value	MIN1hour	0,8	0,6	1.
14	33	MIN1day_AVG1h		100	Years	1	Days	Minimum value	AVG1hour	0,8	0,6	1.
15	34	MIN1week		100	Years	7	Days	Minimum value	MIN1day_AVG1h	0,8	0,6	1.
16	35	MIN1month		100	Years	1	Months	Minimum value	MIN1day_AVG1h	0,8	0,6	1.
17	36	MIN1month_AVG24h		100	Years	1	Months	Minimum value	AVG1day	0,8	0,6	1.
18	37	MIN1year		100	Years	1	Years	Minimum value	MIN1day_AVG1h	0,8	0,6	1.
19	40	MAX1minute		3	Months	1	Minutes	Maximum val...	CurrentHistory	0,8	0,6	1.
20	47	MAX1hour		10	Years	1	Hours	Maximum val...	MAX1minute	0,8	0,6	1.
21	49	MAX8hour		10	Years	8	Hours	Maximum val...	MAX1hour	0,8	0,6	1.
22	50	MAX1day		100	Years	1	Days	Maximum val...	MAX1hour	0,8	0,6	1.
23	51	MAX1day_AVG1h		100	Years	1	Days	Maximum val...	AVG1hour	0,8	0,6	1.
24	52	MAX1week		100	Years	7	Days	Maximum val...	MAX1day_AVG1h	0,8	0,6	1.
25	53	MAX1month		100	Years	1	Months	Maximum val...	MAX1day_AVG1h	0,8	0,6	1.
26	54	MAX1month_AVG24h		100	Years	1	Months	Maximum val...	AVG1day	0,8	0,6	1.
27	55	MAX1year		100	Years	1	Years	Maximum val...	MAX1day_AVG1h	0,8	0,6	1.
28	58	SUM1minute		3	Months	1	Minutes	Sum of values	CurrentHistory	0,9	0,8	1.
29	65	SUM1hour		10	Years	1	Hours	Raw sum	SUM1minute	0,9	0,8	1.
30	67	SUM8hour		10	Years	8	Hours	Raw sum	SUM1hour	0,9	0,8	1.
31	68	SUM1day		100	Years	1	Days	Raw sum	SUM1hour	0,9	0,8	1.
32	69	SUM1week		100	Years	7	Days	Raw sum	SUM1day	0,9	0,8	1.
33	70	SUM1month		100	Years	1	Months	Raw sum	SUM1day	0,9	0,8	1.
34	71	SUM1year		100	Years	1	Years	Raw sum	SUM1day	0,9	0,8	1.
35	74	LVA1minute		3	Months	1	Minutes	Last value	CurrentHistory	0,8	0,6	1.
36	81	LVA1hour		10	Years	1	Hours	Last value	LVA1minute	0,8	0,6	1.
37	83	LVA8hour		10	Years	8	Hours	Last value	LVA1hour	0,8	0,6	1.
38	84	LVA1day		100	Years	1	Days	Last value	LVA1hour	0,8	0,6	1.
39	85	LVA1week		100	Years	7	Days	Last value	LVA1day	0,8	0,6	1.

Figure 33: History tables available. Selected histories are used in history collection template AVG. In each history table it is defined where the data is collected. For example. AVG1minute table is collected from CurrentHistory, and AVG1hour is collected from AVG1minute.

As a result of applying History collection templates into tags, so called transformations are created for the associated variable. These are available from the Vtrin main tree in **Maintenance/Basics/Histories**.

5.3 Adding new History Transformation

New History Transformation can be added, in this case as an example to add 20 minutes average **AVG20minute**, the procedure follows:

Login to Vtrin as Administrator, In Tree **Maintenance/System/History Lists/Histories** Click **Edit** and **New**.

Add the following:

Table 4: Example for adding new History Transformation

Name	AVG20minute
ID	(none)
Duration	1
Duration Unit	Year
Table continues on next page	

Calculation Cycle	20
Calculation Cycle Unit	Minutes
Transformation Type	Time average
Source Table	AVG1minute
Questionable Limit	0.8
Validity Limit	0.6
Base Time	1/1/2300 12:00:00 AM
Execution Interval Unit	Minutes
Execution Interval	1
Latest Transfer	(default)
Next Transfer	(default)
Transfer Interval	4
Transfer Interval Unit	Days
Split Age	8
Split Age Unit	Days
Log Split Length in Rows	0
Split Length	4
Split Length Unit	Months
Owner	(none)
Owner Permissions	128
Group	(none)
Group Permissions	128
Other Permissions	128
Maximum Size in Gigabytes	0
Auto Size Adjust	Database Size
Enable sorting to the trail table	Unchecked
HandleWithCore	Unchecked

Click **Commit/Yes/Revert, Save** and **Close** History tab.

Click on **History Collection Templates** in the Tree, Select Template field of AVG, Right click **Edit** add AVG1minute; AVG10minute;AVG15minute; **AVG20minute**; AVG1hour; AVG1day. Click **Commit/Yes/Revert, Save** and **Close** History Collection Templates tab.

To start AVG20minute transformation for existing AVG history logging, open SYS600 Object Navigator and navigate to **Logging Profiles/By Profile Type (PT) HISTORY**. Open each logging profile object and in case the History Collection Templates (HC) list includes template AVG definition, set the logging profile object out of use by unchecking In Use (IU) and clicking **Apply**. Next set the logging profile object back to use by checking In Use (IU) and clicking **Apply**.

5.3.1 Adding Trend to Tree item context menu

In Vtrin Tree **Maintenance/Display Library/Display Templates**, Right click on **Trend - AVG10minute** and **Copy**, Click on **Display Templates** and **Paste**. Hold CTRL key down and drag and drop the newly pasted **Trend - AVG10minute.** Rename **Trend - AVG10minute** as **Trend – AVG20minute**.

Click to Open **Trend – AVG20minute** template, Right Click on the Graph template area and Open Proprieties. Click on **Items** tab and then on **Past** tab, select History **AVG20minute**. Similarly, Click on **Future** tab, select History **AVG20minute (default)**. Click **OK**, **Save** and **Close Trend - AVG20minute** template.

5.4 History Tables configuration – the base for all history storing

This section describes the features and properties available in History class (History tables list). In most cases, the changes to history tables are not needed, but it is valuable for the user to understand the details and how they affect to history collection process.

Some of the properties of the History tables were already discussed in previous section and in 33. Using the default list view is not always convenient because of the amount of properties available. Instead, the user may choose to use the default properties window or the custom dialog that can be accessed by right-clicking a row in the **History tables** list and selecting **Properties** from the drop-down menu (Figure 34).

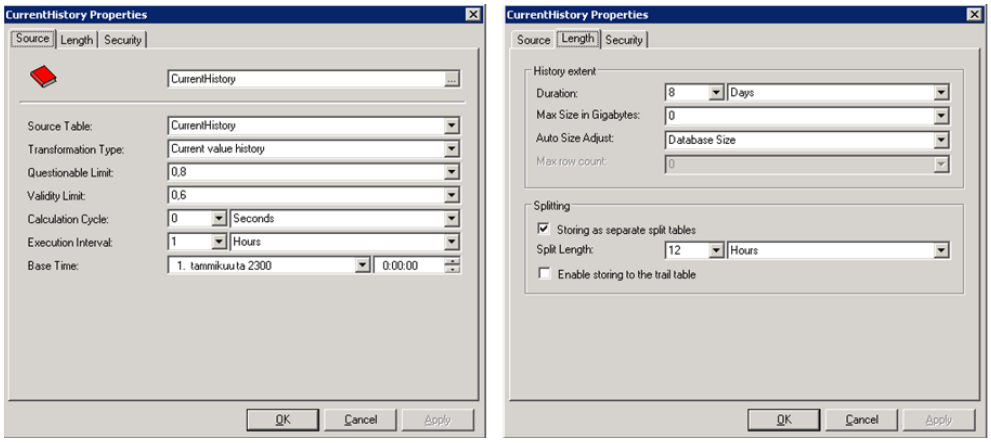


Figure 34: Properties of current the history of the Data collector node. The content of Source and Length tabs.

The first property in the Source tab is the name of the history table. **CurrentHistory** is the most fundamental numerical history table, and all other numerical histories are usually collected from the current history. Current history can also sometimes be referred to as **Raw history**. However, **CurrentHistory** is a more suitable name because often some processing, for example compression, has already been applied to current history data depending on the configuration.

Source Table defines where the data to this history table is collected from. Current history is a special case and the source has been defined to be itself.

Transformation Type defines what kind of data is stored. Again, Current history is a special case and data is defined to be Current value history.

Questionable Limit and **Validity Limit** define how the status of the data is marked with respect to the original data when the data is processed to this history level from the source table.

Calculation Cycle and **Execution Interval** define when the aggregation is running.

Base Time defines how the data is aligned respect to time. The base time that is used by default is selected to be January 1, 2300.

Duration on the **Length** tab controls the extent of the history based on time. Data that is older than defined by this will be removed from the database. It is the responsibility of the RTDB-Transformator service to clean up too old history data. The length of the current history in data collector is 8 days. In the main node, the length of current history would be one year by default.

Auto Size Adjust and **Max size in Gigabytes** control the size of the tables. The **Auto Size Adjust** parameter controls the type of calculation that the history clean-up is based on. If the value is Table size, the cleaning decision is made based on the data in this table only. In that case, the Max size in Gigabytes determines when the cleaning begins. As a result, the data is gradually cleaned up until the desired size is reached. If the maximum size is set to zero, ten no size limitation in use. By default, the **Auto Size Adjust** is set to **Database Size**, in which case the **Max size in Gigabytes** is not interpreted (and is set to 0). This leads to cleaning strategy where the total amount of disk space available is controlling the cleaning. The percentage of the available disk space SYS600 Historian is allowed to use is defined when installing the system. This information is then automatically used to adjust the history lengths and control the cleaning.

Splitting is an optimization and implementation level concept available in SYS600 Historian. All the numerical history, as well as many log tables, are splitted. Splitting means that one database table is divided logically into several smaller tables based on time. This enhances performance and makes many internal implementation techniques possible.

Table can be either splitted or non-splitted. All numerical history tables are splitted by default as the expected amount of data is relatively large. If the **Storing as separate split tables** is checked, the **Split length** (with a unit definition) defines how long the splits are. Naturally, the split length should be smaller than the initial duration of the history.

The split length is not adjustable when the database is running. Run-time modification may corrupt the database and make it unusable.

The user should also be careful in analyzing when split size should be changed. In practice, it is very rare that the system administrator would be required to change splitting settings. Scenarios where this might be needed would be such that the amount of input data would drastically increase from the original plans and requirements. In that case, decreasing the split size might result in better performance. However, reducing the split size might hide parts of the data that already exists in the database. For this reason, the default split size settings are selected to be such that the resulting split tables are still of an acceptable size on the upper bounds of the supported input data rates.

In Figure 34, the split size of the current history on data collector is defined to be only 12 hours. The maximum length of the history is 8 days.

Figure 35 illustrates the history table definition for Average 1 hour table available in main node. Main differences, compared to current history definitions from data collector (Figure 34), are that **Source Table** now refers to AVG1 minute table (that would refer again to current history). The duration of the history is also longer (10 years). This is possible as the amount of data in 1 hour level is considerably smaller than in current history. The **Split length** is defined to be 6 months.

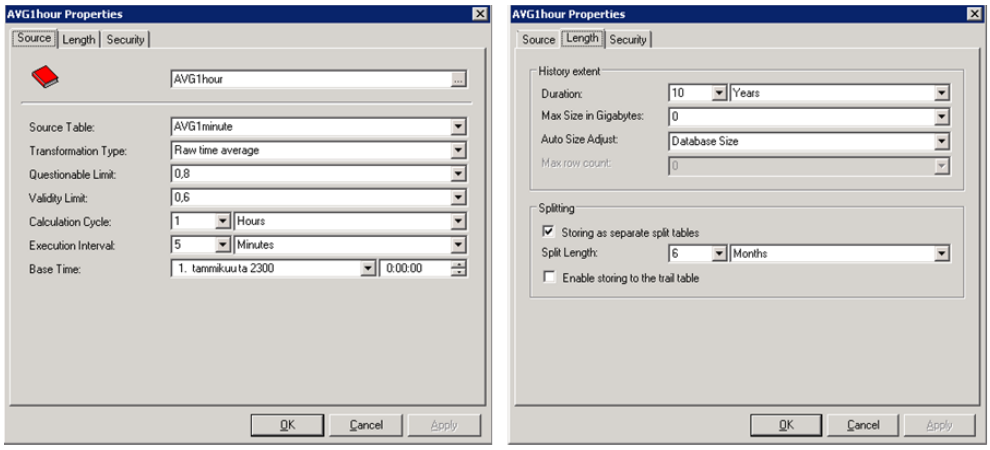


Figure 35: AVG1hour history table definitions from MAIN node.

5.5 Physical History Tables

Figure 36 illustrates the physical table layout for the current history in data collector. Upper level histories use the same layout with the difference that the split length is varying depending on the type of the history.

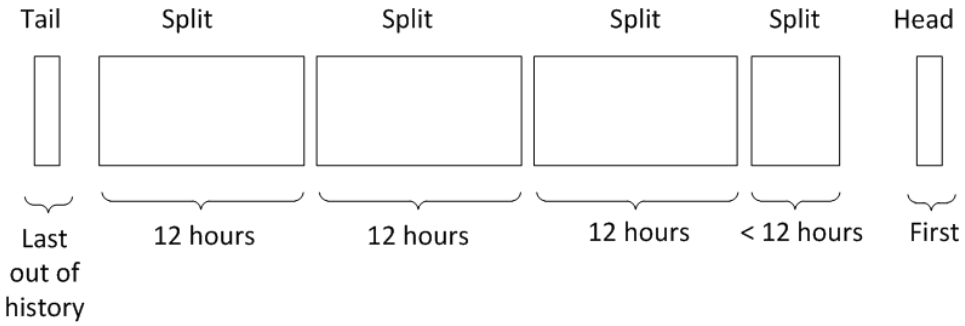


Figure 36: Physical history tables for history storing (current history in data collector node). Split size has been defined to be 12 hours. The first values are stored into the Head table and last value out of the history length is stored into the Tail table. The bulk of the data is stored in several Split tables.

The history data is divided into several Split Segments. The amount and length of the splits is defined by the **History Length** and **Split length** parameters. The start and end time of each split is always known based on these parameters together with the **Base Time** parameter. New input value is written into the Head. When the new value is entered into the Head, the previous value in the head is compressed, if so configured, and written into the newest split. If the split does not exist, it is created on the fly. As the time span of the oldest split has aged out of the defined history length, the entire split is deleted. The latest value of the deleted split is stored into Tail. This is needed in order to support different kinds of history fetch functionalities.

The recommended number of splits is less than 50, but the absolute maximum number allowed is 400 (not configurable).

Exceeding the maximum number of splits limit will stop collecting data to the table in question.

All SYS600 Historian tables, including history split files, are stored into database disk in %APP_DATAPATH%, e.g. into D:\sc\Historian. The split file names includes a date indicating

the time frame the split file covers in yyyymmdd format. The following examples would be split files for December 22, 2011 in data collector. Each 12 hours, starting from midnight, would have one split and an associated table definition. The Head file contains only the base table name while a Tail file has the _Tail postfix.

RTDB_CurrentHistory.TableDefinition

RTDB_CurrentHistory.TableData

RTDB_CurrentHistory.TableIndex

RTDB_CurrentHistory_20111222_0000_P0200.TableDefinition

RTDB_CurrentHistory_20111222_0000_P0200.TableData

RTDB_CurrentHistory_20111222_1200_P0200.TableData

RTDB_CurrentHistory_20111222_1200_P0200.TableDefinition

RTDB_CurrentHistory_Tail.TableDefinition

RTDB_CurrentHistory_Tail.TableData

RTDB_CurrentHistory_Tail.TableIndex

Sometimes it is useful to be able to work in split level. For example, queries using ODBC command line tool praotstx operates directly using the names of the split tables (otherwise the system works against the combination of all tables using logical table names). Also, rtdb_scandb can only be run for one table at a time instead of running it for all tables, which speeds up the administration significantly. Administrator tools like praotstx, rtdb_scandb and rtdb_coreinfo are discussed in detail in Section 6.

5.6 Recalculating secondary histories

Recalculation is needed when historical values are corrected later on, and there are upper level histories that are defined to collect data from history tables where corrections are done. Other reason for recalculations might be that values arrive late to database.

Other terms used for recalculation are recollection and reconsolidation.

Section 6 Administrating a Database in production

6.1 Introduction

After a database has been successfully configured using tags and input data is coming into the system, there are tools to monitor the behavior of the system. This section provides an overview of the various kinds of diagnostic tools that can be used to monitor how database work.

Diagnostics as well as the available services are covered in detail. The purpose of each service, how their behavior can be monitored, and what are the troubleshooting tools and best practices for doing that are also explained in this section.

The administrator should know the answers to, for example, the following questions:

- How to shut down and start up the system
- How to populate the database
- What is the scope of process history for both current history and upper level histories
- Which services exist in the system and what they are doing
- Where to find diagnostic information
- How to take a backup of the database
- How to restore the database from a backup

6.2 Environment

The SYS600 Historian needs a set of system wide environment variables, which have been defined during the installation phase. Table 5 lists the system wide environment variables. Note that usually the administrator should not change the values of the environment variables. In many low level administration tasks it is valuable to know what each environment variable means in order to be able to effectively work from command line.

Table 5: System-wide environment variables in SYS600 Historian production installation.

Name	Value	Description
RTDBRoot	C:\Program Files (x86)\ABB Oy\RTDB	Root directory of the SYS600 Historian software.
CSCommonRoot	C:\Program Files (x86)\ABB Oy \CSCommon	Root directory of the CSCommon software (part of SYS600 Historian product)
CSCommon_TZ	e.g. Europe/Helsinki	Time zone definition
CSCommon_TZDIR	C:\Program Files (x86)\ABB Oy \CSCommon\Data\ZoneInfo	Time zone database location
Path	Includes the Bin directories of SYS600 Historian and CSCommon and <APP_ROOT>\bin	The system Path must contain the directories that contain the EXEs and DLLs needed during runtime.
APP_DATAPATH	D:\sc\Historian	SYS600 Historian database directory
Table continues on next page		

APP_DSN	<COMPUTERNAME>-RTDB	The ODBC data source name of the SYS600 Historian database
APP_ROOT	D:\sc\HistorianApp	Root directory of the application specific software, as well as some configuration files for standard services.
APP_BACKUP_ROOT	E:\HistorianBackup	Backups are stored here.

6.3 Disk Configuration

SYS600 Historian database can reside in any Windows folder. However, in typical installations, the SYS600 Historian database is located on a separate data disk (hard disk drive), and the software is located on another disk (usually System disk C). For performance reasons, it is recommended, whenever possible, to have the disk containing SYS600 Historian database to be formatted using 64kB block size. This is also pointed out in system installer in the setup phase.

The following is a typical disk configuration:

Disk	Description
C:	Operating system and product software
D:	SYS600 Historian Database and project specific Application files. For best database performance, the allocation unit size of this disk should be 64kB.
E:	SYS600 Historian Backup (online backup, essential backup and application backup)

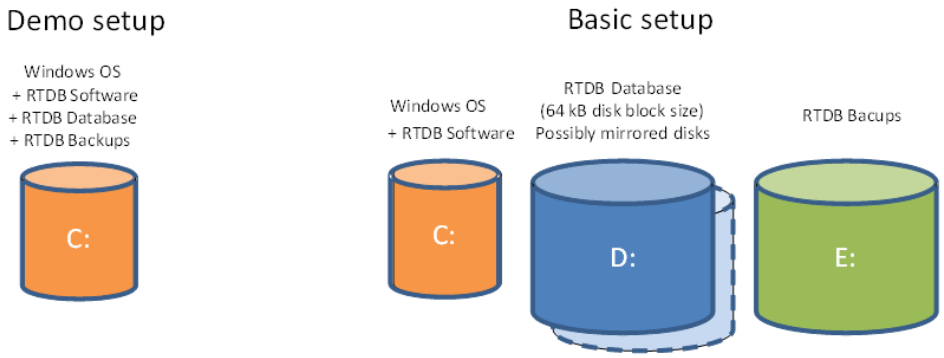


Figure 37: Basic SYS600 Historian disk configurations

It is possible to make more complex and sizeable disk configurations with high redundancy and high performance for SYS600 Historian. A typical larger configuration setup can be seen in Figure 38.

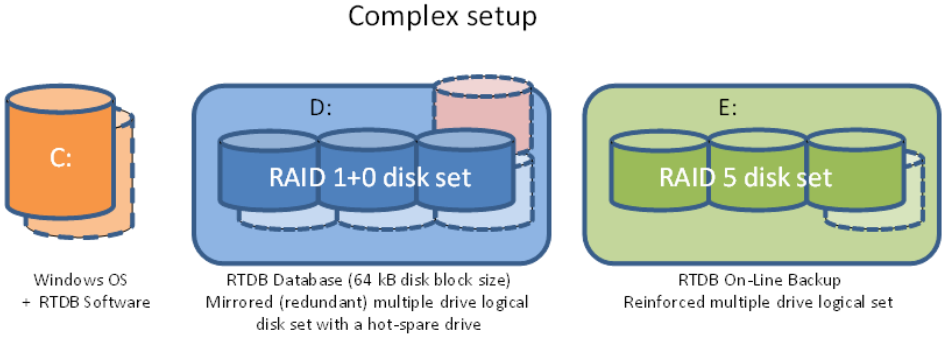


Figure 38: Typical SYS600 Historian disk configurations for a larger system.

The guidelines when planning a SYS600 Historian disk configurations are:

- The SYS600 Historian database must reside in a single Windows folder. If a sizeable database is required, then an adequately sized single logical disk must be provided.
- The size of the database online backup must be at least the same as of the actual database.

The SYS600 Historian database directory location is available in the environment variable %APP_DATAPATH% that refers to the database directory.

RTDB database cannot be partially divided into two different storage places at the same time, i.e. that first part of data exists in one server, and the send part in another server. Queries and data storage shall be always done into one place.

6.4 Database structure

In this document, the Vtrin user interface is used to manage the content of a database and make database configurations. Using Vtrin implies that the database operations are done through the VtrinLib data-abstraction layer. It is good to understand the basics of the relationships between the VtrinLib data-abstraction layer and relational database tables in the core level.

SYS600 Historian is a relational database, and its data is stored into relational tables. The actual database tables and definitions are available in the following directory:

%APP_DATAPATH%

A list of available tables can easily be found by using the command line and running the following query:

dir *.tabledefinition

The amount of available tables is extensive, mostly because there is a separate table for each different process history type (see Section 5.5 for more information on process history tables).

Tables are defined in the files with a filename that ends with .TableDefinition. Actual data is available in .TableData files. Primary indexes are included in .TableData files and .TableIndex files contain secondary indexes. Good tools for understanding table sturcture are, for example:

- RTDB_CoreInfo.exe: information of a selected table (column names, data types and indexes).
- Praotstx.exe: ODBC command line client that can execute SQL queries against the database.

The relational database side illustrates what any relational database essentially looks like. On top of the relational database, there is a VtrinLib data-abstraction layer. Figure 39 illustrates the basic mappings intuitively by combining the UML class model from VtrinLib data abstraction layer into a relational model and actual table data.

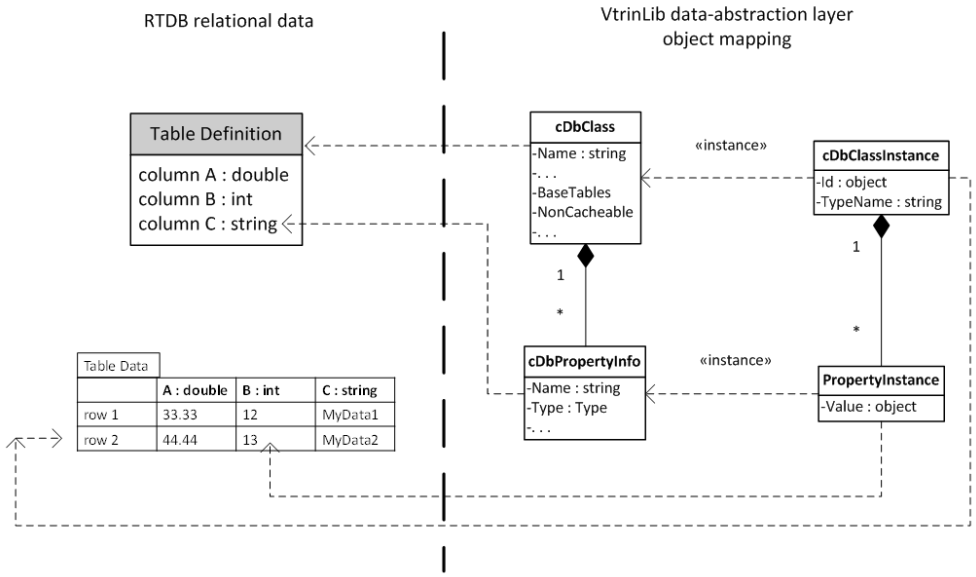
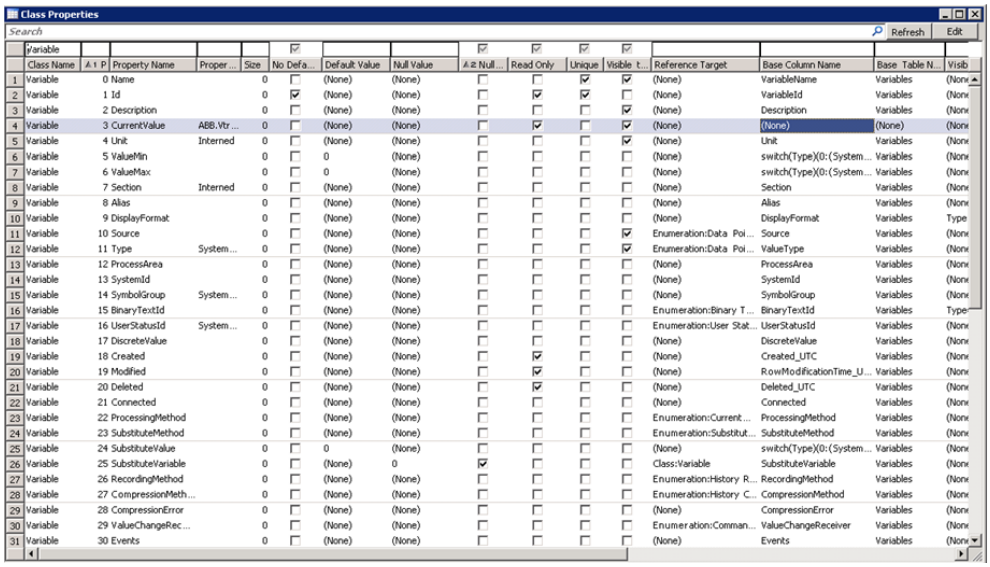


Figure 39: Relationships between SYS600 Historian relational database tables and classes in VtrinLib data-abstraction layer.

The building blocks of VtrinLib object model are class, property info, class instance and property instance. Class and property info together form the type definition part, whereas class instance and property instance are the actual objects. In a simple use case, class is mapped to a single table definition, and property info is mapped to one database column definition. In an instance side, class instance represents the rows in a database table, and each property instance represents a single column in one row. VtrinLib data abstraction layer is, however, able to present more advanced mappings as well. A good example of virtualized class is Tag class, which presents data from several tables, hiding the complexity of the configuration behind a multifunctional, easy to use class.

VtrinLib class definitions are stored into database together with actual process data. Two displays, Classes and Class Properties, are available in the Vtrin tree in **Maintenance/System/Vtrin**. The same folder also contains the language translations for enumerations, class names and class properties. In this document, for example in the tables that specify the contents or Tag and Variable classes in detail, the class properties and classes are referred to by using language independent presentation. Figure 40 illustrates some of the property info definitions for the Variable class.



	Variable														
	Class Name	A↓1 P	Property Name	Proper...	Size	No Defa...	Default Value	Null Value	A↓2 Null...	Read Only	Unique	Visible t...	Reference Target	Base Column Name	Base Table N...	Visib
1	Variable	0	Name		0	☐	(None)	(None)	☐	☐	☑	☑	(None)	VariableName	Variables	(None
2	Variable	1	Id		0	☑	(None)	(None)	☐	☑	☑	☐	(None)	VariableId	Variables	(None
3	Variable	2	Description		0	☐	(None)	(None)	☐	☐	☐	☑	(None)	Description	Variables	(None
4	Variable	3	CurrentValue	ABB.Vtr...	0	☐	(None)	(None)	☐	☑	☐	☑	(None)	(None)	(None)	(None
5	Variable	4	Unit	Interned	0	☐	(None)	(None)	☐	☐	☐	☑	(None)	Unit	Variables	(None
6	Variable	5	ValueMin		0	☐	0	(None)	☐	☐	☐	☐	(None)	switch(Type)0:(System...	Variables	(None
7	Variable	6	ValueMax		0	☐	0	(None)	☐	☐	☐	☐	(None)	switch(Type)0:(System...	Variables	(None
8	Variable	7	Section	Interned	0	☐	(None)	(None)	☐	☐	☐	☐	(None)	Section	Variables	(None
9	Variable	8	Alias		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	Alias	Variables	(None
10	Variable	9	DisplayFormat		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	DisplayFormat	Variables	Type
11	Variable	10	Source		0	☐	(None)	(None)	☐	☐	☐	☑	Enumeration:Data Poi...	Source	Variables	(None
12	Variable	11	Type	System...	0	☐	(None)	(None)	☐	☐	☐	☑	Enumeration:Data Poi...	ValueType	Variables	(None
13	Variable	12	ProcessArea		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	ProcessArea	Variables	(None
14	Variable	13	SystemId		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	SystemId	Variables	(None
15	Variable	14	SymbolGroup	System...	0	☐	(None)	(None)	☐	☐	☐	☐	(None)	SymbolGroup	Variables	(None
16	Variable	15	BinaryTextId		0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:Binary T...	BinaryTextId	Variables	Type
17	Variable	16	UserStatusId	System...	0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:User Stat...	UserStatusId	Variables	(None
18	Variable	17	DiscreteValue		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	DiscreteValue	Variables	(None
19	Variable	18	Created		0	☐	(None)	(None)	☐	☑	☐	☐	(None)	Created_UTC	Variables	(None
20	Variable	19	Modified		0	☐	(None)	(None)	☐	☑	☐	☐	(None)	RowModificationTime_U...	Variables	(None
21	Variable	20	Deleted		0	☐	(None)	(None)	☐	☑	☐	☐	(None)	Deleted_UTC	Variables	(None
22	Variable	21	Connected		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	Connected	Variables	(None
23	Variable	22	ProcessingMethod		0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:Current...	ProcessingMethod	Variables	(None
24	Variable	23	SubstituteMethod		0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:Substitut...	SubstituteMethod	Variables	(None
25	Variable	24	SubstituteValue		0	☐	0	(None)	☐	☐	☐	☐	(None)	switch(Type)0:(System...	Variables	(None
26	Variable	25	SubstituteVariable		0	☐	(None)	0	☑	☐	☐	☐	Class:Variable	SubstituteVariable	Variables	(None
27	Variable	26	RecordingMethod		0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:History R...	RecordingMethod	Variables	(None
28	Variable	27	CompressionMeth...		0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:History C...	CompressionMethod	Variables	(None
29	Variable	28	CompressionError		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	CompressionError	Variables	(None
30	Variable	29	ValueChangeRec...		0	☐	(None)	(None)	☐	☐	☐	☐	Enumeration:Comman...	ValueChangeReceiver	Variables	(None
31	Variable	30	Events		0	☐	(None)	(None)	☐	☐	☐	☐	(None)	Events	Variables	(None

Figure 40: Properties of Variable class available in Maintenance\System\Vtrin

6.5 Diagnostics

6.5.1 General

There is no one single place that would contain the information that is useful in understanding the behavior of the system. However, the administrator should at least understand which component is responsible for a certain functionality in order to be able to know what tools are available, and therefore to understand the behavior in detail. See Section 6.6 for more information on component functionality.

This section provides a general description of the available diagnostic tools. For more information, see Section 6.6.

6.5.2 Component status

Monitoring the condition of the system can be started with Component Status. Component Status is a Vtrin class that is implemented by many SYS600 Historian components, and it is available for browsing from Vtrin in **Maintenance/Component Status**. The component's appearance in a (redundant) Main node after successful installation with two connected data collector nodes is illustrated in Figure 41.

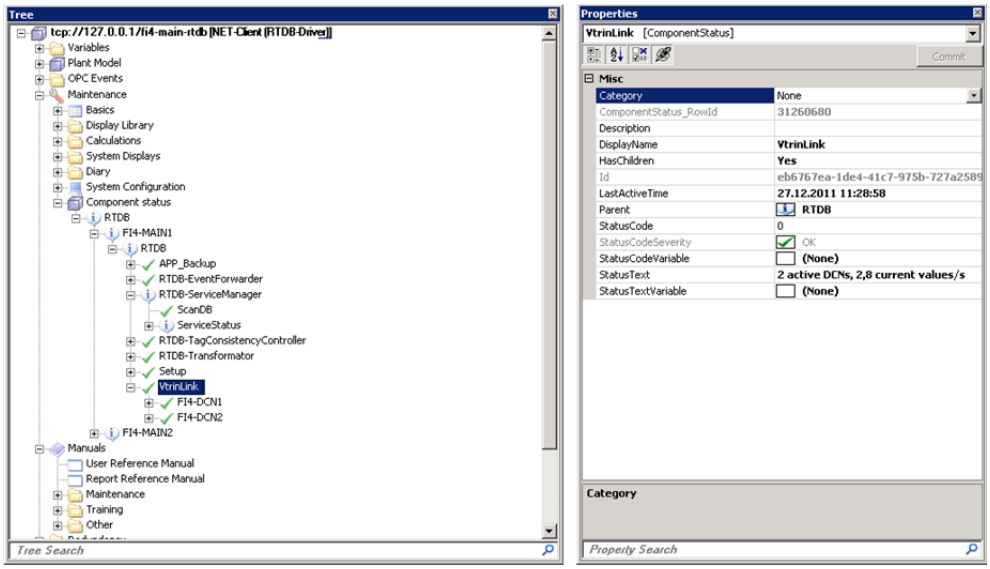


Figure 41: Component status tree in redundant main node. The FI4-MAIN redundant main node has two data collectors connected (FI4-DCN1) and (FI4-DCN2). The generic properties view is used to display the status values for each component status instance available.

A component status has parent-child relationship. One component status has zero or more children and one parent, with the exception of the root level node RTDB that has not parent.

The generic properties window is used to show the status information related to each component. The window is docked below the tree component by default. Status code shows the integer presentation for the status. The icons visible to the user are interpreted based on this information. The status codes that are currently in use are OK, Information, Warning and Error in increasing severity order.

Figure 42: Status codes and icons available for Component status

Component status information is updated by owning a component. For example, a VtrinLink component status and the related children are updated by the VtrinLink service, and the RTDB-TagConsistencyController components are updated by the RTDB-TagConsistencyController service. The three first levels in redundant Main nodes, as well as the two first levels in a normal Main node and data collectors, are updated by the RTDB service manager service (RTDB \FI4-MAIN1\RTDB). It is up to each component to decide how it propagates component status information from a child node to the parent node. In case of errors that need to be resolved, the errors will be visible in the root level as Error. In that case, the administrator should browse from the root level down to those components that are showing errors and resolve the problem. Clues to what the error might be can be found either from StatusCode or from StatusText properties. For more information on error generating services, see Section 6.6. All services are not yet covered by component status. However, the number of supported components will increase in future releases. Detailed information on different components can be found in Section 6.6.

6.5.3 MessageLog

Many services provide tracing information to the MessageLog. The log is available in the Vtrin tree in **Maintenance/System/System Logs**.

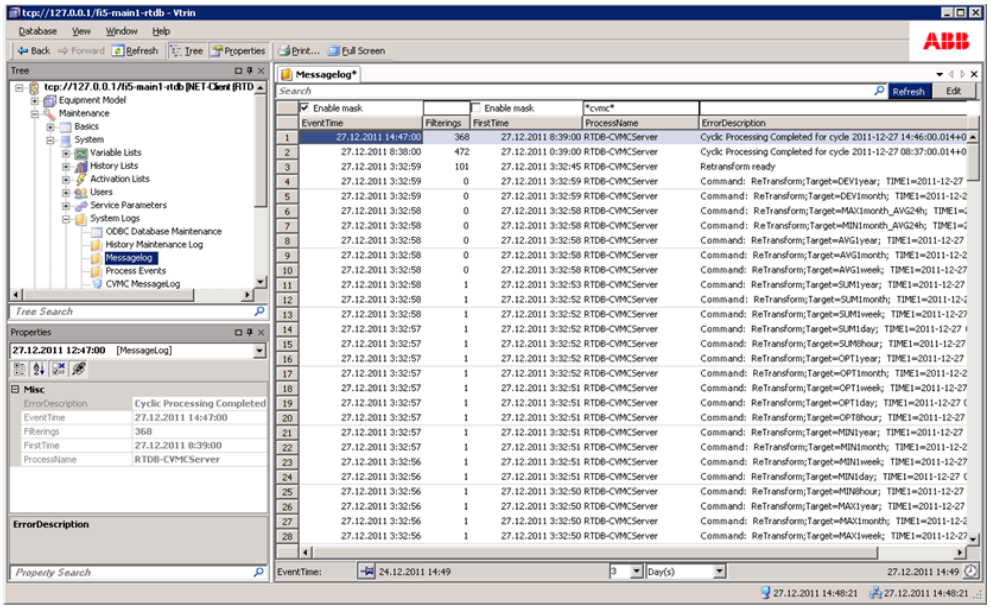


Figure 43: Figure Message log is displaying information from RTDB-CVMC Server service

The log messages can be monitored in the MessageLog display. The MessageLog display is operated as any other list display in Vtrin. Useful filters are, for example, EventTime and ProcessName. Services that produce logging information to message log include SYS600 Historian main service (SRVMGR in ProcessName), RTDB-CVMCServer, RTDB-Transformator and RTDB-EcPerfmon.

Messages in MessageLog can also be fetched and monitored with the RTDB_MsgType tool. This tool can also be used remotely because it uses the ODBC connection to the database. The tool can be used to inspect old diagnostic messages while continuously monitor new messages. The actual diagnostic messages are stored in the table named MessageLog (Vtrin class MessageLog).

It is important to remember that the MessageLog mechanism is a filtering log. Forgetting this can cause confusion even for most experienced system specialists. The filtering mechanism means that similar event messages are filtered out from the log table, and only a filter count is incremented to indicate that there has also been more similar messages. The similarity of the diagnostic messages does not mean that the messages must be identical. Instead the following heuristics is used: the messages are treated as similar if they differ only on numbers, except if the number is after a letter (however, some applications may override the definition of similarity). For example, the messages PM1_SPEED=9.9 and PM1_SPEED=10.1 are treated as similar, but the messages PM1_SPEED=9.9 and PM2_SPEED=9.9 are treated as different (while “PM11_SPEED=9.9” and “PM12_SPEED=9.9” are again treated similar. The similarity is handled within the name of the application (or process) that produces the message, so messages “hello” and “hello” are treated different if they come from different applications (again some applications may override the definition of similarity). The similarity checking is performed within an 8-hour time window (can be set in SimpleConfig). The time window starts when the first message of a kind arrives. The MessageLog table remembers the first time and the last time when the message has arrived. Together with the filter count, the information is usually enough to be useful in determining what has happened and when.

To display the MessageLog messages using MsgType application:

1. Log into the active node
2. Open Diagnostics Tools from the SYS600 Historian control panel (either from desktop or from %APP_DATAPATH%\SYS600 Historian Control Panel)
3. Run any of the prepared MsgType - * commands, for example MsgType – Service Manager, that prints old messages within last 10 hours, and keeps monitoring new messages (note that the amount of old messages printed varies between each tool, for most tools the setting is 10 minutes).
4. The MsgType – Ask is the interactive version. It asks the process name (or part of it), a search string, and how old messages are printed (as seconds).
5. The MsgType – Ask can also be used in more advanced way. Answer the question Give delta seconds [600] with (without the question marks):
    5.1. "600 -c 1f" to display also the first time when the message occurred. The available letters that can be used are: p=ProcessName, 1=FirstTime, f=Filterings, s=Severity. Severity number: 0 = success, 1 = info, 2 = warning, 3 = error.
    5.2. "600 -i 0.5" to use 0.5 second polling interval for the messages instead of the default of 3.

More help can be found by running the command "RTDB_MsgType -?" on the command line.

6.5.4 Diagnostics log files

Most SYS600 Historian services produce a diagnostics log in text files that reside in the Diag subdirectory under the database path:

%APP_DATAPATH>\Diag, for example D:\sc\Historian\Diag.

The name of the diagnostic text file begins with the service name, except that the special characters in the service name are substituted.

Most log files are such that a new log file is generated for every day, and when the number of log files exceeds the predefined limit, cleaning is done starting with the oldest log file.

Diagnostic files can be opened with any suitable text tool e.g. notepad.

Dedicated behavior or tools for diagnostics are documented in Section 6.6. For clarity, some services use a dedicated folder under the folder %APP_DATAPATH%\Diag to store logging information. Such services are RTDB-CalcScheduler, RTDB-EventForwarder, and RTDB-TagConsistencyControllerWindows performance counters.

A useful tool for understanding the behavior of the system is to monitor Windows performance counters. There are system wide performance counters for various resources, as well as counters for specific processes. Monitoring the history of SYS600 Historian related performance counters has been made very easy as a selected set of performance counters have been configured to be collected after setup. Figure 44 illustrates some performance counters available.

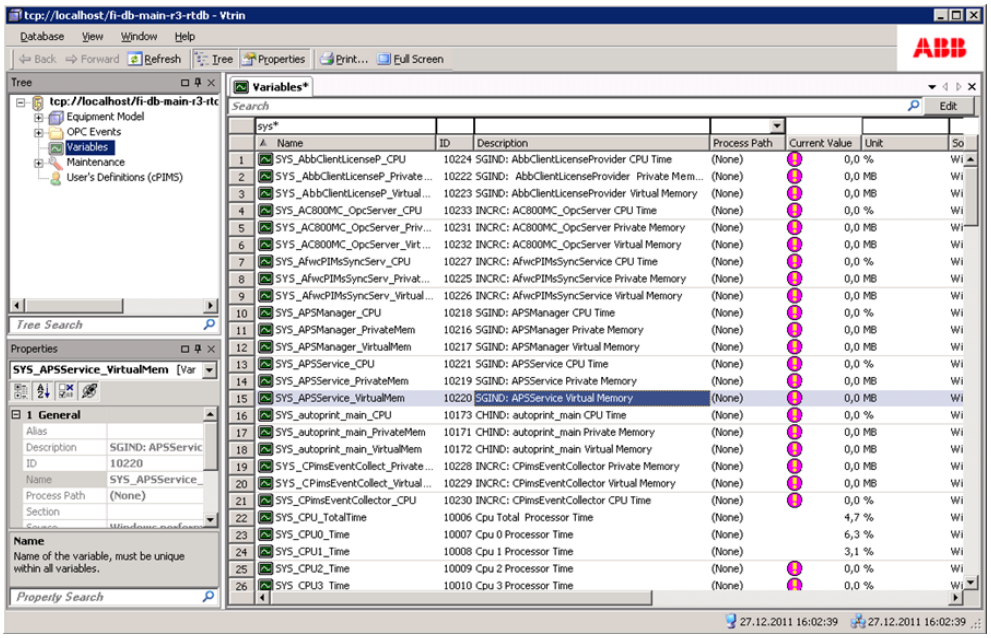


Figure 44: Windows performance counters are found in the Variables list after installation. To display only performance counters, sys filter can be applied to the Name property.*

Depending on the configuration, some performance counters are not collecting data. These are marked with a red exclamation point icon. The easiest way to find the history of a certain variable is through the context menu of variable instance by selecting **Send to/Trend**. The performance data is collected from Windows performance counters by the RTDB-EcPerfMon service. If some important counters are missing, adding new ones can be done by adding new rows to the eccrossrefs-table using the ODBC interface.

6.5.5 Windows Event Log

The SYS600 Historian services and software libraries may write messages into the Windows Event Log in phases where the other diagnostic means are not possible, for example when the MessageLog table of SYS600 Historian database is missing or is corrupted, or when the given data path was incorrect. Furthermore, Windows performs useful logging in case an application crashes.

Windows Event Viewer is started from Windows start menu **All Programs/Administrative tools/Event Viewer**. Windows Administrative rights are needed to access Windows Event Viewer. The messages from SYS600 Historian or messages from application crashes are available in Windows Logs folder in the Application list.

6.6 Services

6.6.1 General

SYS600 Historian software runs in the background as multiple Windows Services. A list SYS600 Historian services can be seen using Windows services.msc (Figure 45).

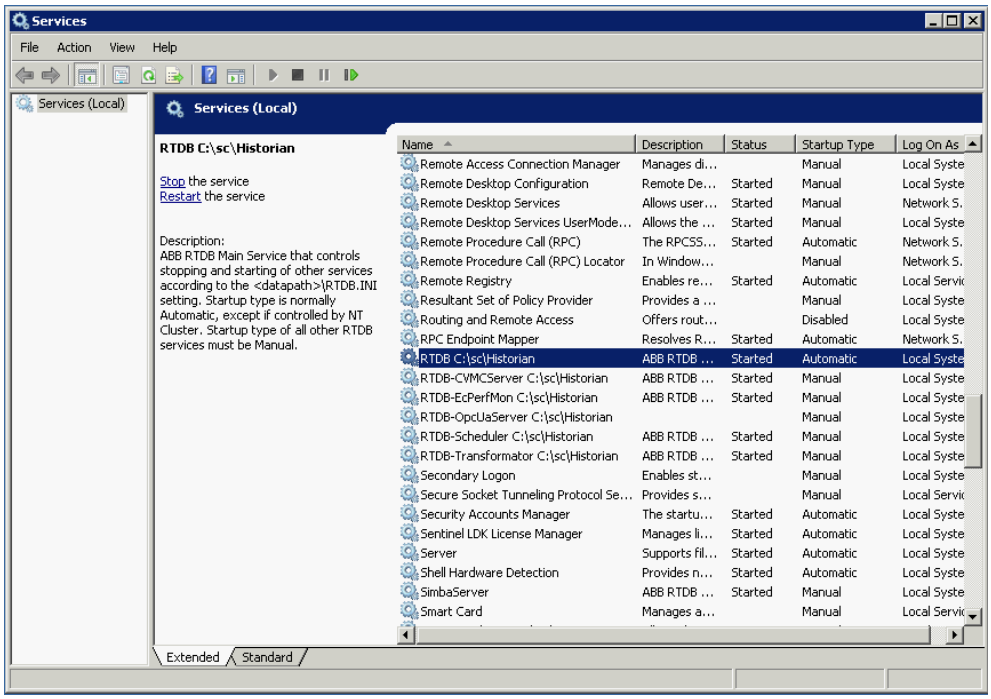


Figure 45: Services.msc shows SYS600 Historian services running.

Most of the service names start with RTDB. In addition to these, there is the SimbaServer (ODBC server) and two Vtrin Servers (Vtrin Server and Vtrin Server RTDB Internal), VtrinLink.

A single computer may have multiple SYS600 Historian databases. Because of this, the database name is included in the names of the SYS600 Historian services. However, installing several SYS600 Historians into same computer is usually not needed and therefore not supported when using standard installers.

The following sections describe in detail what is the functionality the service is responsible for, what the configuration options for each service are, and how to monitor and understand what the service is doing.

6.6.2 RTDB-ServiceManager

6.6.2.1 Functionality

RTDB-ServiceManager (selected service RTDB D:\sc\Historian) is the main service of SYS600 Historian. It runs as a Windows service under local system account, and it is normally defined to be automatically started up during the computer startup (Startup Type is Automatic).

SYS600 Historian Service Manager controls and monitors other SYS600 Historian services.

6.6.2.2 Configuration

A list of the SYS600 Historian services to be managed can be found in the RTDB.INI file of the database directory (%APP_DATAPATH%). SYS600 Historian Service Manager cyclically tries to start other services if they are not running. The monitoring cycle is defined with the Interval_ms setting in the General section of RTDB.INI. This reduces the need for engineering in case of service crashes.

6.6.2.3 Diagnostics

SYS600 Historian Service Manager writes the status of the services into MessageLog (for more information on MessageLog, see Section 6.5.2). Component status (see Section 6.5.3) shows what services are running and what have been set, for example, into disabled state.

6.6.3 RTDB-CVMCServer

6.6.3.1 Functionality

RTDB-CVMCServer is responsible for storing real time input data (current values) into the database. It stores values to current history from current values as well. Before storing values to current history, compression and other (pre)processing is applied if such are configured to be used.

A key component in current value processing is the ring buffer that is used between the data links and RTDB-CVMCServer. SYS600 produces values to a shared memory ring buffer, from where the values are consumed by the RTDB-CVMCServer that actually stores the values into the database. This procedure is a-synchronic and may sometimes be confused when current values are fed using e.g. a Vtrin user interface.

Another task for the RTDB-CVMCServer is to calculate the upper level histories from the current history based on the defined transformations. Together with the RTDB-Transformator it also performs upper level history recalculations when such are requested. In that case, the RTDB-CVMCServer performs the history recalculation for the last period whereas the RTDB-Transformator takes care of all other older periods.

6.6.3.2 Configuration

After installation, no additional configuration is required. The configuration settings are available in SimpleConfig from the Vtrin tree **Maintenance/Basics/Service Parameters (SimpleConfig)**. The Section property can be used with "*RTDB-CVMCServer" to filter the view so that the settings for the RTDB-CVMCServer only are shown. A substantial number of the available parameters are configured based on resources (memory, number of CPUs etc.). It is recommended that these parameters are not changed.

6.6.3.3 Diagnostics

Message log can be used to monitor behavior. For more information on message log, see Section 6.5.3.

It is sometimes advantageous to be able to see diagnostics online from console, and therefore the RTDB-CVMCServer can also be run from console. Before accessing it through console, the service should be set as disabled and stopped from SERVICES.MSC. From Path to executable in services.msc, -service switch should be removed when the service is started as a normal process. The command line could be, for example:

"C:\Program Files\ABB Oy\RTDB\bin\RTDB_CVMCServer.exe" D:\sc\Historian

where D:\sc\Historian is the database directory. Stopping the service from console is done by pressing CTRL+C once. The RTDB-CVMCServer uses a direct server connection to access the database. In order to preserve the contents of the used database tables, the service or the process should always be stopped in a controlled way. Shutting down the service violently, for example with Windows Task Manager, can break the tables. This can cause, for example, that next time the startup time is increased due to the fact that rtdb_scandb has to fix potentially broken tables.

Shutting down the RTDB-CVMCServer might take some time if the size of the database and the number of variables is large. Please be patient and wait for the server to shut down on its own.

6.6.4 RTDB-Transformator

6.6.4.1 Functionality

RTDB-Transformator is responsible for cleaning up history tables that have aged out from the history scope. Log tables are also cleaned by using similar rules.

Together with the RTDB-CVMCServer, RTDB-Transfromator takes care of recalculating upper level histories. In the recalculation process, the RTDB-Transformator is responsible for all but the very latest time period of upper level history.

Monitoring available disk size and the current size of the database is also executed by the RTDB-Transformator. Based on that, the RTDB-Transformator automatically adjusts the history length definitions for tables, where the scope of history is defined to be adjusted based on the available disk size.

6.6.4.2 Configuration

After installation, no additional configuration is required. The configuration settings are available in SimpleConfig from the Vtrin tree **Maintenance/Basics/Service Parameters (SimpleConfig)**. The Section property can be used with "RTDB-Transformator" to filter the view so that the settings for the RTDB-Transformator only are shown. Some of the parameters are configured based on resources (memory, number of CPUs etc.). It is recommended that these parameters are not changed.

6.6.4.3 Diagnostics

Message log can be used to monitor behavior. For more information on message log, see Section 6.5.3.

It is sometimes advantageous to be able to see diagnostics online from console, and therefore, the RTDB-Transformator can be run from console. Before accessing it through console, the service should be set as disabled and stopped from SERVICES.MSC. From Path to executable in the services.msc, -service switch should be removed when service is started as a normal process. The command line could be for example:

"C:\Program Files\ABB Oy\RTDB\bin\RTDB_Transformator.exe" D:\sc\Historian

where D:\sc\Historian is the database directory. Stopping the service from console is done by pressing CTRL+C once. The RTDB-Transformator uses a direct server connection to access the database. In order to preserve the contents of used database tables, the service or the process should always be stopped in a controlled way. Shutting down the service violently, for example with Windows Task Manager, can break the tables. This can cause, for example, that next time the startup time is increased due to the fact that rtdb_scandb has to fix potentially broken tables.

6.6.5 RTDB-EcPerfMon

6.6.5.1 Functionality

Collects data from Windows performance counters based on the definitions in the ec_crossrefs table.

6.6.5.2 Configuration

Configuration is done by using the ec_crossrefs table. Configuration is ready after setup. If the default set of performance counters is not enough and new performance counters are added, one row for the each variable should be created. No Vtrin class mapping exists for this table, but the ODBC interface has to be used to configure the table. The Excel application %APP_ROOT%\Config\Variables\RTDB_Populate_cPIMSPerfMon.xls can be used to create ec_crossrefs rows using the ODBC interface. However, the application is of a low level and not as straightforward to use as it could be.

6.6.5.3 Diagnostics

Message log can be used to monitor behavior. For more information on message log, see Section 6.5.3.

It is sometimes advantageous to be able to see diagnostics online from console. Therefore, the RTDB-EcPerfMon can be run from console. Before accessing it through a console, the service should be set as disabled and stopped from SERVICES.MSC. From Path to executable in the services.msc, –service switch should be removed when service is started as a normal process. The command line could be, for example:

"C:\Program Files\ABB Oy\RTDB\bin\RTDB_EcPerfMon.exe" D:\sc\Historian

where D:\sc\Historian is the database directory. Stopping the service from console is done by pressing CTRL+C once. The RTDB-EcPerfmon uses a direct server connection to access the database. In order to preserve the contents of used database tables, the service or the process should always be stopped in a controlled way. Shutting down the service violently, for example with Windows Task Manager, can break the tables. This can cause, for example, that the next time the startup time is increased due to the fact that rtdb_scandb has to fix potentially broken tables.

6.6.6 RTDB-Scheduler

6.6.6.1 Functionality

The purpose of the service is to run different kinds of scheduled tasks. There are some tasks that the RTDB-Scheduler performs by default, for example tasks related to installing the product, but the users and the system administrator can add project specific tasks to be executed as well.

6.6.6.2 Configuration

After installation, no additional configuration is required. The configuration settings are available in SimpleConfig from the Vtrin tree **Maintenance/Basics/Service Parameters (SimpleConfig)**. The Section property can be used with "RTDB-Scheduler" to filter the view so that the settings for the RTDB-Scheduler only are shown. It is recommended that these parameters are not changed.

New tasks are assigned to the RTDB-Scheduler either by using TimedTasks entries (the associated table and Vtrin class both have the same name) or by using the command queue mechanism.

6.6.6.3 Diagnostics

Message log can be used to monitor behavior. For more information on message log, see Section 6.5.3.

6.6.7 Vtrin Server

6.6.7.1 Functionality

Vtrin Server (also known as Vtrin-NetServer) is a service that provides remote access to Vtrin user interface client. (Local connections are usually also done through Vtrin Server as it provides an additional protection layer for the database tables.)

The Vtrin user interface connects the database by using the Vtrin Server as a very first step when client is started (login details are described in detail in Section 4.1). If the connection string starts with "wss://", the connection is established through Vtrin Server.

6.6.7.2 Configuration

No configuration changes are required after installation. Vtrin Server reads the configuration file at startup from VTRIN-NETSERVER.EXE.CONFIG, which is located in the same directory as the Vtrin Server. The directory can be checked from SERVICES.MSC properties in Path to executable.

The configuration file is in XML and contains, for example, details of which IP addresses and ports are used, what database to connect to, and which buffer sizes and timeouts are available for different kinds of communication scenarios. The meaning of each parameter is described briefly by using XML comments.

Sometimes the service needs to be run interactively from command line. This can be done by disabling the service from SERVICES.MSC and stopping it. The command line can be copied from SERVICES.MSC properties from Path to executable and adding –d to the end. For example:

C:\Program Files\ABB Oy\Vtrin35\Vtrin-NetServer.exe –d

The service is stopped by pressing the ESC button (Note that the Vtrin Server is stopped in a different way from some other services, which are stopped using CTRL+C. Using the same command for Vtrin-NetServer would terminate the program violently and potentially break the database tables).

6.6.7.3 Diagnostics

Log files are saved in the folder %APP_DATAPATH%\diag\. The log files are named by combining Vtrin Server with the date of when the log information was generated, e.g. Vtrin Server_2011-12-31. The content of the file can be verbose, and it also contains much lower level information than what usually is valuable in troubleshooting situations for developers and for customer support.

6.6.8 Vtrin Server RTDB Internal

6.6.8.1 Functionality

Vtrin Server RTDB Internal (also known as Vtrin-NetServer Internal) is a service that provides remote access to other services over the Vtrin data abstraction layer. Vtrin Server RTDB Internal is available both in data collector nodes and in the main node. Having separate servers to handle user interface requests and serving internal communication provides enhanced fault tolerance and better possibilities to optimize internal communication.

Currently there are no differences in behavior on functional level.

6.6.8.2 Configuration

No configuration changes are required after installation. Vtrin Server RTDB Internal reads the configuration file at startup from VTRIN-NETSERVER-RTDBINTERNAL.EXE.CONFIG, which is located in the same directory as the Vtrin Server RTDB Internal. The directory can be checked from the SERVICES.MSC properties in Path to executable.

The configuration file is in XML and contains, for example, details of which IP addresses and ports are used, what database to connect to, and which of buffer sizes and timeouts are available for different kinds of communication scenarios. The meaning of each parameter is described briefly by XML comments.

Sometimes the service needs to be run interactively from command line. This can be done by disabling the service from SERVICES.MSC and stopping it. The command line can be copied from SERVICES.MSC properties from Path to executable and adding –d at the end. For example:

C:\Program Files\ABB Oy\Vtrin35\Vtrin-NetServer-RtdbInternal.exe –d

The service is stopped by pressing the ESC button (Note that the Vtrin Server RTDB Internal is stopped in a different way from some other services, which are stopped using CTRL+C. Using the same command for Vtrin Server RTDB Internal would terminate the program violently and potentially break the database tables).

6.6.8.3 Diagnostics

Log files are saved in the folder %APP_DATAPATH%\diag\. The log files are named by combining Vtrin Server RTDB Internal with the date when the log information was generated, e.g. Vtrin Server_2011-12-31. The content of the file can be verbose, and also contains much lower level information than what usually is valuable in troubleshooting situations for developers and for customer support.

6.6.9 VtrinLink

6.6.9.1 Functionality

VtrinLink is a service that fetches numerical process data (current values and current history) from data collector nodes into the main node. The hierarchical system architecture is described in Section 3.1.

VtrinLink reads the Tag configuration available in the Main node and connects to one or more data collector nodes based on that information. VtrinLink uses the Vtrin Server RTDB Internal to connect data collectors.

VtrinLink uses queued current values (CurrentValue2 class) to get numerical values from data collector as soon as possible. In case of network breaks between the main node and data collector node, VtrinLink uses values stored into current history of the data collector to backfill missing values to the main node once the network break is over. Similar backfill is done if the main server is shut down for some reason.

In case there are redundant tags from redundant data collectors, VtrinLink is connected to both data collectors simultaneously, and data is collected from where it can be provided the fastest.

6.6.9.2 Configuration

After the first data collector has registered itself to the main node, the RTDB-Scheduler starts the post-installation procedure that install services that are needed to manage the communication between the Main node and Data collectors. VtrinLink is one of these services while the other ones are RTDB-TagConsistencyController and RTDB-EventForwarder. This configuration procedure is automatic and the system administrator is not required to perform any additional steps. If the services are not installed automatically after having the main node and at least one data collector installed, the user needs to ensure that the network firewall is not blocking the communication between the main node and data collectors. The complete list of ports that are required to be open is available in installer documentation (The TCP port that SIMBA client in data collector node uses to register itself to Main is 1583/TCP).

The VtrinLink service, as well as RTDB_TagConsistencyController and RTDB-EventForwarder, are stopped every time a new data collector is installed into the hierarchical system, and the RTDB-Scheduler reconfigures and restarts the services automatically.

The configuration files of VtrinLink are available in the folder %APP_ROOT%\Config \VtrinLinkIniFiles\.

The main initialization file VRTINLINK.INI refers to the data collector specific initialization files (VtrinLink_<datacollectorname>). Configuration files are automatically generated and managed by the RTDB-scheduler.

6.6.9.3 Diagnostics

By default, log files are stored in the folder %APP_DATAPATH%\diag\.

The log files for the last ten days are available in the folder. The amount of log files and their naming can be controlled from VRTINLIN.INI. The contents of log files is quite low level information, and it is mainly targeted for troubleshooting.

6.6.10 RTDB-TagConsistencyController

6.6.10.1 Functionality

RTDB-TagConsistencyController (TCC) is responsible for maintaining tag configuration in the hierarchical system. The hierarchical system architecture is described in Section 3.1. Usually, tags are originally created in data collector node, and the RTDB-TagConsistencyController brings the definitions into main node automatically. Furthermore, if the tag configuration is modified, the RTDB-TagConsistencyController synchronizes tags between the nodes.

RTDB-TagConsistencyController executes a full tag consistency check at server startup. After that it relays the events received from Tag class. After network break, a full check is also performed to related data collectors.

When tags are deleted, the RTDB-TagConsistencyController uses the information stored into the TagDeleted class in order to handle deletions that occur during network breaks. Once a tag has been deleted, it cannot be created again with the same Id (GUID) without first removing it from the TagDeleted class of all related nodes.

6.6.10.2 Configuration

After the first data collector has registered itself to the main node, the RTDB-Scheduler starts the post-installation procedure that install services that are needed to manage the communication between Main node and Data collectors. The RTDB-TagConsistencyController is one of these services.

This configuration procedure is automatic and the system administrator is not required to perform any additional steps. If the services are not installed automatically after having the main node and at least one data collector installed, the user needs to ensure that the network firewall is not blocking the communication between the main node and data collectors. The complete list of ports that are required to be open is available in installer documentation (The TCP port that that SIMBA client in data collector node uses to register itself to main is 1583/TCP).

The configuration files of RTDB-TagConsistencyController are available in the folder %APP_ROOT%\Config\TccIniFiles\.

The main initialization file RTDB-TAGCONSISTENCYCONTROLLER.INI refers to the data collector specific initialization files (RTDB-TagConsistencyController_<datacollectorname>). Configuration files are automatically generated and managed by the RTDB-scheduler.

6.6.10.3 Diagnostics

Component status information is available in Vtrin (see the component status description in Section 6.5.2). The main component status RTDB-TagConsistencyController has all the data collectors and the main node as children. Currently, there is only support to see if the network connection to the data collectors is okay.

Log files are stored in the folder:

%APP_DATAPATH%\diag\TagConsistencyController\

There is a separate log file for each day. Log files for last 30 days are preserved. A log file contains information about events generated by the tag class, as well as what actions the RTDB-TagConsistencyController has executed based on consistency checks. Information about possible network breaks between the main node and data collectors is also included.

6.6.11 SimbaServer

6.6.11.1 Functionality

SimbaServer provides ODBC-access to SYS600 Historian for remote clients.

6.6.11.2 Configuration

No changes needed after installation.

6.6.12 RTDB-OpcUaServer

6.6.12.1 Introduction

The RTDB OPC UA server implements partly the OPC Unified Architecture (UA) – a set specifications defined by OPC Foundation. UA is the next generation version from popular OPC industry standard interoperability specifications. UA combines all the previous specifications under one homogenous information model and does this using the best IT security practices and cross platform capabilities. More information about the specifications is available in OPC Foundation’s web pages http://www.opcfoundation.org.

6.6.12.2 Functionality

RTDB OPC UA server is able to publish RTDB current values according to OPC UA specification 1.01. RTDB OPC UA Server supports the monitoring of the current values and reading history values from the variables that store process history. RTDB OPC UA Server does not support alarms or events and it is not possible to write values to MicroSCADA process database via the RTDB OPC UA Server.

SYS600 Historian versions beginning from the version 1.1 support UA server functionality. No client support is currently available.

6.6.12.3 Configuration

SYS600 Historian installation includes RTDB OPC UA server installed as Windows service. Currently installing two service instances into same computer is not supported. After the installation, the OPC UA service starts up is not automatic.

OPC UA service start up: While RTDB service is running in an installed system, on Windows desktop **SYS600 Historian Control Panel/Starting and Stopping/Shortcut to RTDB.INI** using Notepad. Set `RTDB-OpcUaServer c sc Historian=`**ON**, **Save** and **Close** the file. Wait for some time for OPC UA Service to start automatically.

There exists two initialization files that server reads at the startup. RtdbOpcUaDriverConfig.ini defines where the server connects and some other basic configurations like the content that is shown in the address space. Visibility of the aggregated (pre-calculated) history levels is configurable here, for example. By default they are not displayed. RtdbOpcUaServerConfig.xml contains UA-stack and UA SDK related parameters, including security configurations. It is not recommended to change the settings in RtdbOpcUaServerConfig.xml.

6.6.12.4 Connecting the server with Unified Automation UA Expert

The RTDB OPC UA server is connected as any other OPC UA server. Connecting with security policy ‘None’ is not allowed by default and only supported security policy is Basic128Rsa15 with message security mode ‘Sign and encrypt’.

As an example instructions to get connected is demonstrated here using free OPC UA client - UA Expert (version 1.2.2 175) - from Unified Automation. This is currently the most used free test OPC UA client. Freeware download is available from (Registration required):

http://www.unified-automation.com/opc-ua-clients/

After installing UA Expert (Installation is simple **next/next** process) it is started from Windows Start menu or from the desktop. Once the client is started first time a certificate is required to be generated for the client. UA requires both client and server applications to be identified using certificates. To get connected both client and server certificates can be self-signed.

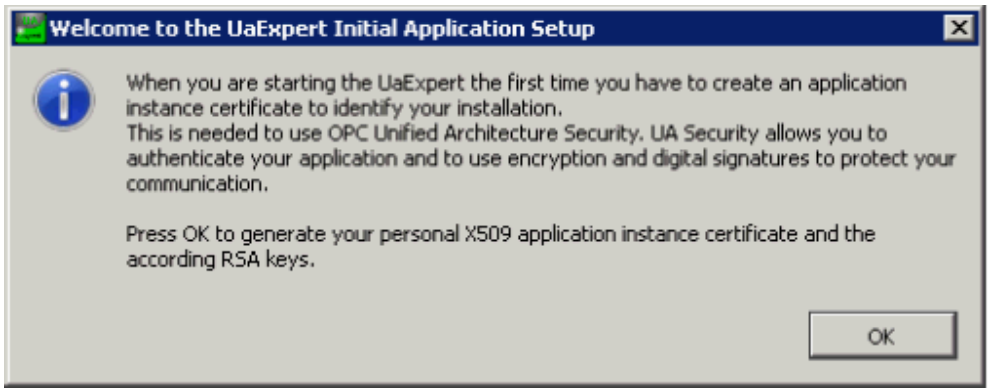


Figure 46: First time starting UA expert require to generate certificate for the client application

Certificate details should be filled as asked. This is one time process only, as certificate will be saved to disk and used later on automatically. (Figure 46 and Figure 47).

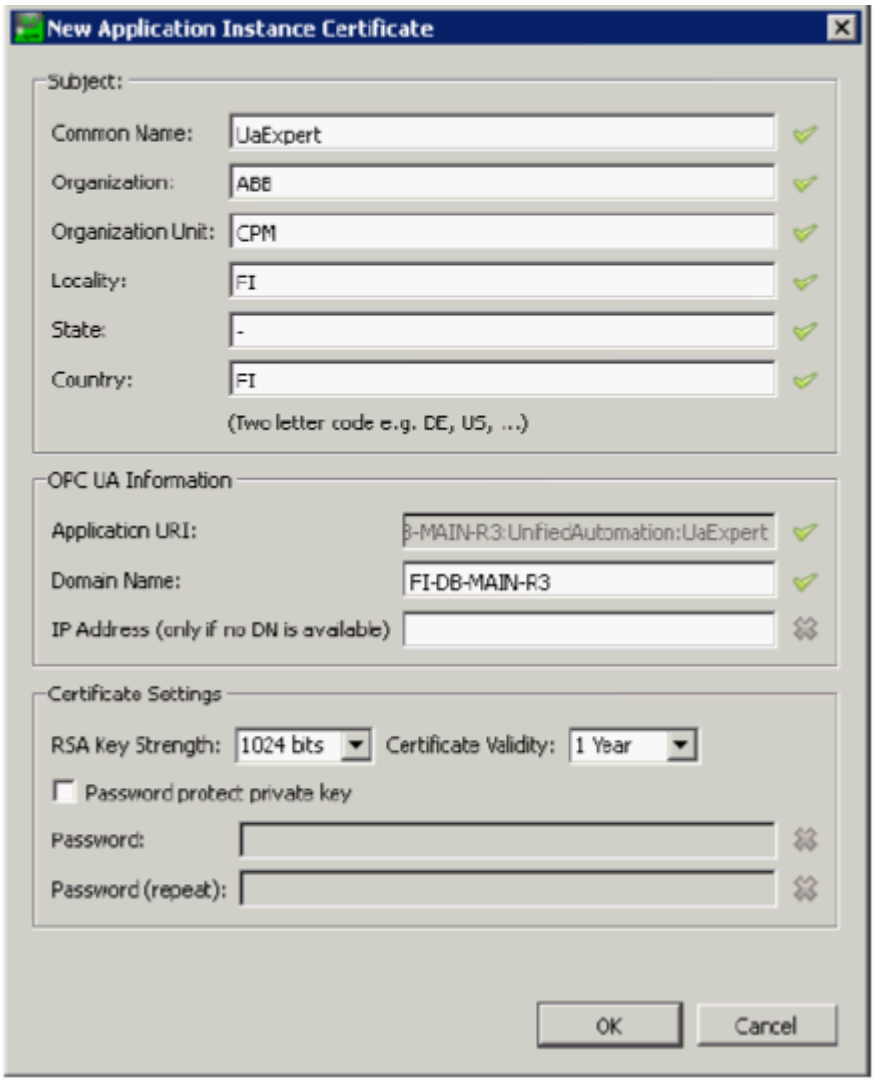


Figure 47: Certificate details for the client application certificate. An example

You can enable downloading updates automatically if internet connection is available (Figure 48)

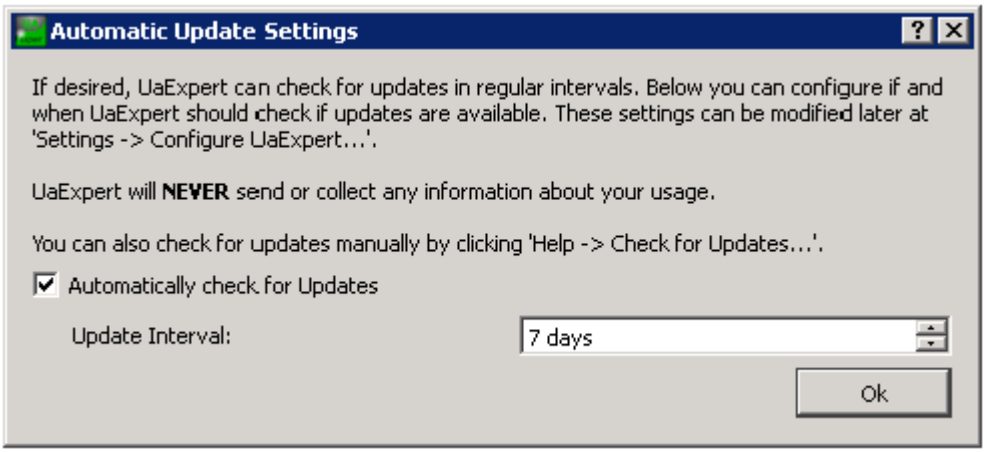


Figure 48: Enabling automatic updates

After these steps UA Expert is ready to use (Figure 49).

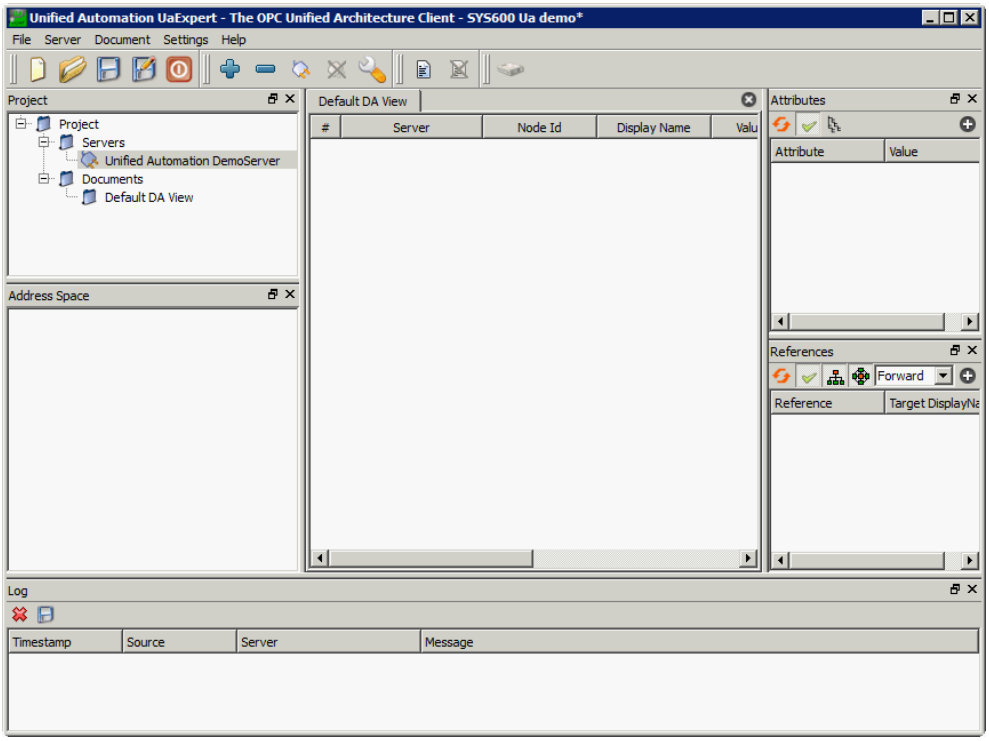


Figure 49: UA Expert is ready to use

Connecting the server starts by adding a new server from the context menu in the Project view **Project/Servers/Add**. As there isn't Discovery server running by default, the server is connected from Advanced tab, by giving the exact Endpoint Url with security and authentication details. The Endpoint Url should be:

opc.tcp://localhost:4841

The port 4841 is the default tcp-communication port in UA specification. Security Setting should be selected to be **Basic128Rsa15** and **Sign & Encrypt**. **Configuration Name** and **Session Name** can be anything that characterizes the connection (Figure 50).

To start Discovery server, refer UA Expert-Quickstart Installation and Configuration Guide.

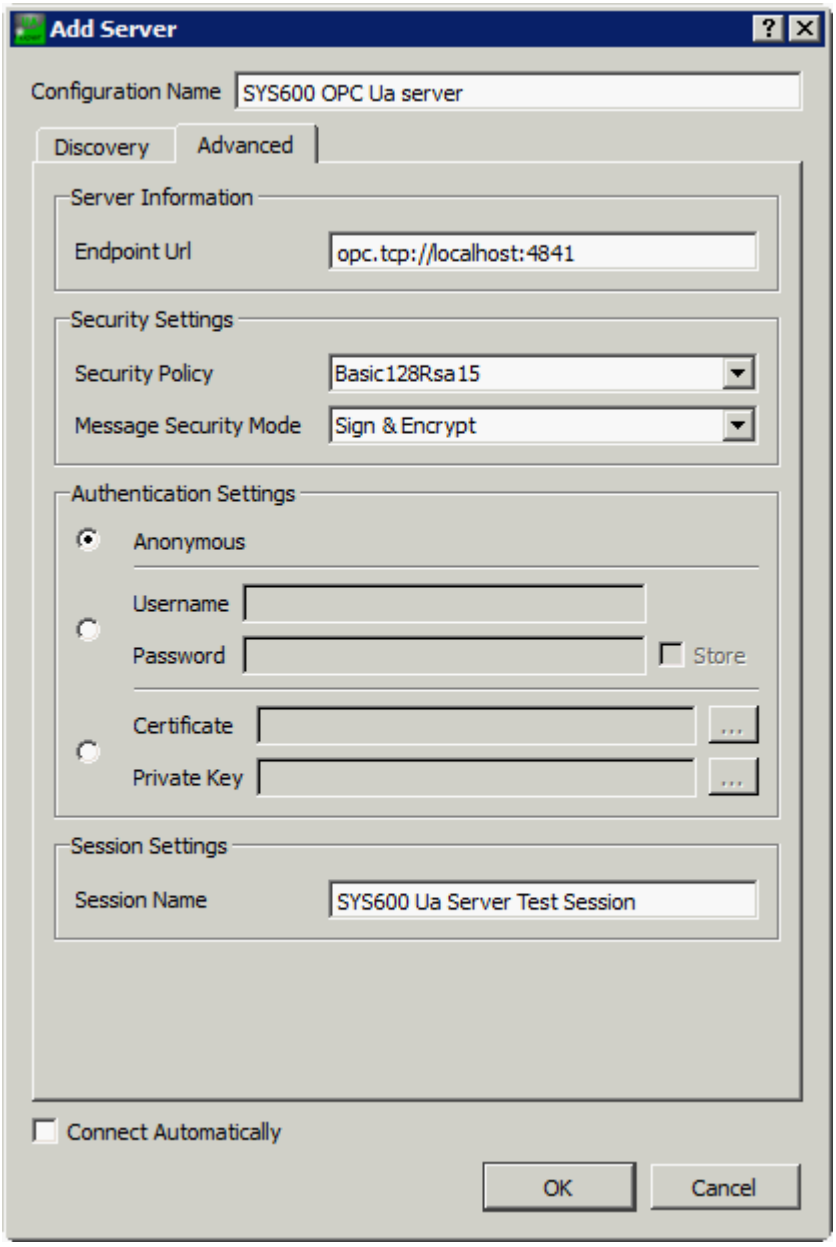


Figure 50: Preparing UA server to be connected

Once the server is configured, it can be connected from the context menu by right Click **Connect** (Figure 51).

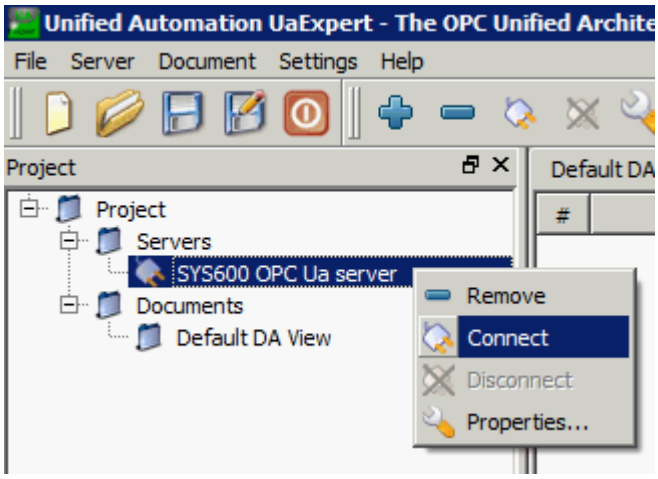


Figure 51: Connecting the UA server

First connect attempt fails with the error `Connecting failed with error 'BadSecurityChecksFailed'` as client application has not been accepted to connect the server. This can be done next by going to server computer and moving client's certificate from the directory

C:\Program Files\ABB Oy\RTDB\Bin\Config\UaSecurity\rejected

to directory

C:\Program Files\ABB Oy\RTDB\Bin\Config\UaSecurity\certs

After that retrying connect should succeed.

Once getting connected the server returns its self-signed application instance certificate to client, and UA Expert asks to accept that (Figure 52). Certificate can be accepted permanently or only for this session.

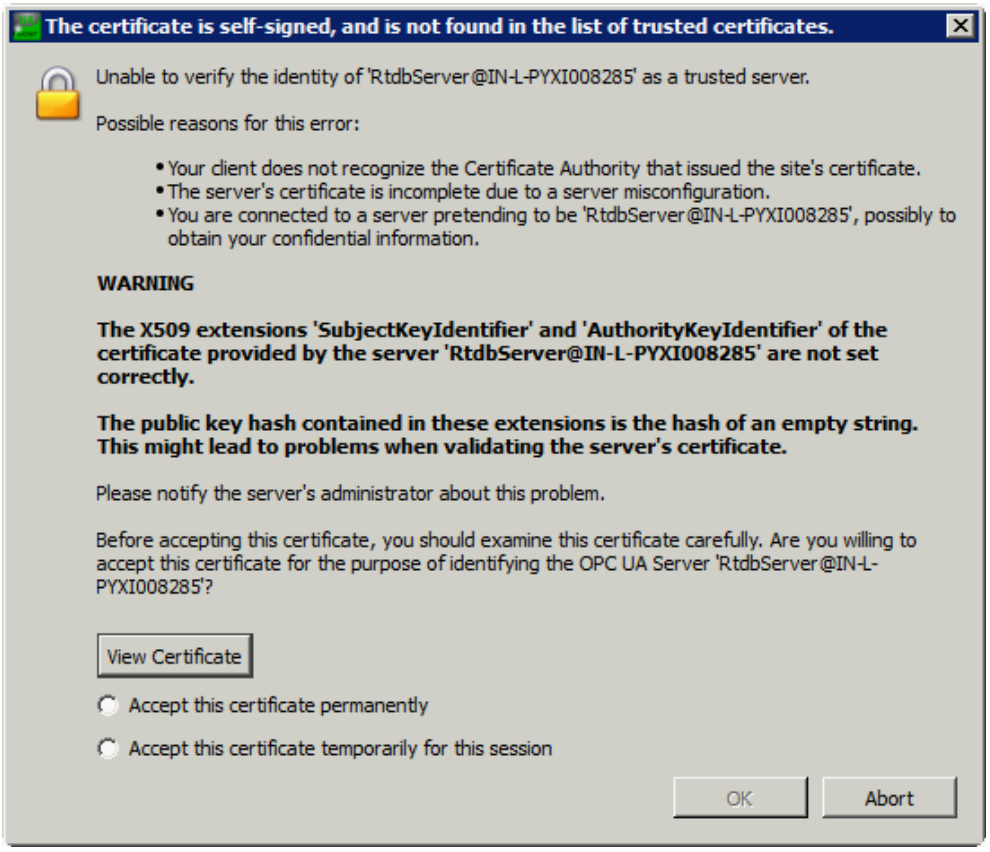


Figure 52: UA expert asks to accept the server application instance certificate

In case the security policy 'None' is used the server is not validating client application instance certificate. Security policy settings can be changed from the file

%APP_ROOT%\Config\OpcIniFiles\RtdbOpcUaServerConfig.xml

None security policy could be added by uncommenting the following lines:

```
<SecuritySetting>
<SecurityPolicy>http://opcfoundation.org/UA/SecurityPolicy#None</
SecurityPolicy>
<MessageSecurityMode>None</MessageSecurityMode>
</SecuritySetting>
```

If the connection is successful it is reported in Log window and Address Space window is populated to show currently available types and objects. If the connection fails it might be due to connection timeout. In this case a message: `Connection failed with error bad timeout` is displayed in Log window, UA Expert uses relatively tight default timeouts to determine that connection cannot be established. As RTDB OPC UA server initializes the device connection when the first OPC UA client connects, it might be that it was not able to do that fast enough. Client side timeout settings can be changed from menu **Settings/Configure/ UaExpert**. Alternatively retry the connection.

If the connection was successful the resulting view is similar to Figure 53.

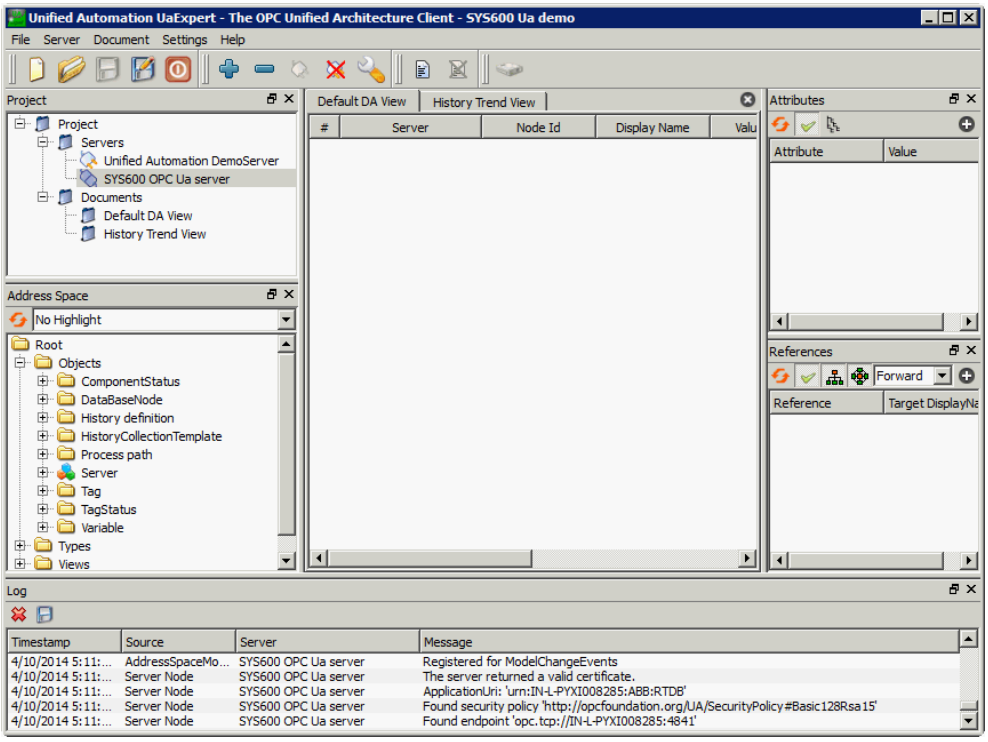


Figure 53: UA Expert is successfully connected with SYS600 OPC UA Server

6.7 Management tasks

6.7.1 Starting SYS600 Historian

SYS600 Historian is started automatically when the computer is started. If SYS600 Historian is not running (e.g. as someone has set it to start manually from SERVICES.MSC), it can be started from the SYS600 Historian Control Panel. SYS600 Historian Control Panel is available in:

```
%APP_DATAPATH%\SYS600 Historian Control Panel\
```

The control panel is also available for all administrators in **Windows Administrative Tools/ SYS600 Historian Control Panel** and in the desktop of the computer where SYS600 Historian is installed.

Starting SYS600 Historian is done as follows:

1. Open the SYS600 Historian Control Panel
2. Go to Starting and Stopping subfolder of the SYS600 Historian Control Panel
3. Run Start SYS600 Historian Control Panel (in 2008 server, or Run as Admin in Windows 7)

At startup, all tables are checked by the RTDB_ScanDB process. Scanning through all tables takes some time and the database is not fully functional before RTDB_ScanDB has finished. The table scanning order is prioritized so that the production of data into the database can start as soon as possible, and that taking Vtrin connections to the database would be possible. The old history splits are scanned in the very last phase as that is usually the least interesting data.

The task manager is fastest tool used to monitor which SYS600 Historian services are running. Most SYS600 Historian processes are prefixed by RTDB_, and the **Processes** tab of Windows Task Manager quickly shows if there are any running SYS600 Historian processes.

The progress of RTDB_SCANDB.EXE can be monitored from the log file SCA*.TEMP in the Windows temp directory (usually located in C:\Windows\temp, but in some systems, located in C:\documents and settings\LocalSystem\local settings\temp). The log file is appended to the log file of the SYS600 Historian main service in the %APP_DATAPATH%\Diag directory after RTDB_SCANDB.EXE has completed.

It is recommended to use the SYS600 Historian Control Panel scripts to start and stop SYS600 Historian, because the control panel provides warnings if the SYS600 Historian is tried to start while it already is started. It also displays warnings at startup if broken tables are found (and automatically fixed).

6.7.2 Stopping SYS600 Historian

It is recommended to shut down SYS600 Historian manually before shutting down Windows. Shutting down the database interactively provides the system administrator with the possibility to monitor the shutdown process.

SYS600 Historian should be shut down from SYS600 Historian Control Panel subfolder Starting and Stopping using the script Stop RTDB. The shutdown script is able to detect if any tables are corrupted. It also warns the user if there are some applications that keep tables open. In case some tables are kept open by a process, the shutdown script suggest away to resolve the conflict (e.g. Kill, Retry or Kill the process forcibly).

If SYS600 Historian is not shut down manually before shutting down Windows, Windows shutdown scripts do it automatically. In that case, however, the system administrator is not able to monitor the process. Windows tries to shut down SYS600 Historian automatically before shutting itself down, and by default waits for up to 30 minutes for all SYS600 Historian related services to stop. Windows continues the shutdown process as soon as shutdown scripts have been completed.

It is recommended to notify the users before stopping SYS600 Historian in production. An example of a shutdown message could be:

“The database %APP_DSN% will be shut down for <reason> at <time>. The system will be back again after 30 minutes. <Administrator name>/tel. 555-5555”

For Vtrin users, this is done easiest by establishing a Vtrin connection to Vtrin Server (connection details wss://127.0.0.1/history) and browsing the Active Users list from Vtrin main tree in Maintenance\System\Users\Active Users. The list shows all Vtrin clients that are currently connected to SYS600 Historian. Right-clicking a user in list and selecting **Send To (Message)** from the drop-down menu sends a message to the selected user. A message can also be sent to all users at once.

SYS600 Historian consists of many service processes, but the SYS600 Historian main service (RTDB <data path>) handles starting and stopping the other SYS600 Historian services. When SYS600 Historian is in the shutdown state, none of the SYS600 Historian services are running. This means that data acquisition services do not produce data to the database, history is not collected, and it is not possible to connect to the database from remote nodes through ODBC. Local direct use of the database is possible, but it should still be avoided.

6.7.3 Restarting, enabling and disabling separate SYS600 Historian services

If a service needs to be stopped for some reason, it is recommended to modify RTDB.INI in the database directory

```
%APP_DATAPATH%\RTDB.INI
```

and set the service state to OFF. After that, the service will be stopped within one minute (the polling cycle can be changed from RTDB.INI, Interval_ms=60000).

The service can also be stopped directly from SERVICES.MSC. If using this mechanism, disabling the service is needed for all services. When stopping the service with NET STOP command or from Windows Services application, the SYS600 Historian service manager would restart the service within one minute by default. This mechanism cannot be used for services that are marked with DisableWhenStop=1. These would be started from disabled state by SYS600 Historian service manager.

The easiest way to restart SYS600 Historian services is to use the Restart command of the Windows Services application. However, a quick restart of Vtrin Server or Vtrin Server RTDB Internal may not always succeed, and the user may have to retry the start, or just let the SYS600 Historian service manager start it automatically when it detects that the service is not running.

6.7.4 Backing Up the SYS600 Historian Database and Software Files

Because most database files of SYS600 Historian are usually kept open by the SYS600 Historian software, a running SYS600 Historian database cannot be backed up simply with Windows Backup. For backing up SYS600 Historian, use one of the following methods:

- Offline database backup
- Full database disk tape backup
- Online backup
- Essential backup
- Application backup
- Product software backup

Offline database backup is done by shutting down SYS600 Historian and taking a copy of the database directory (%APP_DATAPATH%) using any backup tool or the user’s favorite method. The benefits of offline backup are that the concept is easy to understand and that it is independent from SYS600 Historian product.

Full database disk tape backup that is done online using a backup technique that is able to execute without having to lock files (e.g. Windows shadow copy mechanism). In case the database disk breaks up, at least one backup of the original disk is needed in addition to the online copy to be able to fully restore the system. For details, see Section 6.7.5.

Online backup is SYS600 Historian's mechanism that is automatically scheduled to run every day to backup all data from database directory into a backup disk specified in the setup phase (usually, the E: disk is used for the backups). By default, online backup is only configured to run in main nodes, and not in data collectors. This is because the purpose of the data collectors is only to buffer data for short time period. The content of an online copy reflects the latest data in the database directory.

Essential backup contains a copy of the SYS600 Historian database tables that contain configuration information (but does not include history data or logs). Essential backups are stored to backup disk in folder:

```
%APP_BACKUP_ROOT%\EssentialBackup
```

Essential backup is scheduled to run daily by default, but unlike online backup, it is run both in data collector and in main nodes. There are several versions of essential backup available:

- Backup from each day for last 7 days
- Backup from each week during last 4 weeks
- Backup from each month during last 12 months
- Backup from each year that SYS600 Historian has been running

The above is achieved by re-using subfolder names. E.g. a backup directory %APP_BACKUP_ROOT%\EssentialBackup\WeekDay2 would be re-used next Monday, unless it is be time to take a weekly, monthly, or yearly backup.

In addition to the table files, essential backup contains all other files from the database directory and its subdirectories. E.g. the content of the diag subdirectory is copied, which may be valuable when searching for information on what kind of diagnostics various services have been producing in the past.

Essential backups can be especially valuable in cases where the configuration has become invalid due to an invalid user interaction. Having multiple versions from backups spanning a longer time period make it possible to return back to a valid version many times.

If SYS600 Historian is upgraded to newer version, SYS600 Historian software upgrade creates essential backups to same directory, before running the upgrade.

Application backup is a SYS600 Historian mechanism to backup files in folder %APP_ROOT%. This folder contains, for example, calculations if such have been created. Application backup is different from other backups in the sense that old versions of the application files from backup are never automatically deleted. The backup appends the modification timestamp of the file to the target file name. As a result, all old versions from the end of the day can be found in the application backup directory.

Product software backup is not automatically executed by the scheduled SYS600 Historian backup task. Software is usually installed into system disk under folders Program Files and Program Files (x86) (latter one is only available in 64-bit operating systems). However, old version of the product software is backed up when the product software is upgraded to a new version. In software upgrade, the backup goes as a subdirectory under %APP_BACKUP_ROOT %, equipped with the current date in the directory name.

Monitoring backups can be done by using Component status nodes available from the Vtrin tree in Maintenance\Component Status\RTDB\APP_Backup. DiskFree status shows how much space there currently is available in the backup disk (in the **Generic Properties** window, LastActiveTime presents the snapshot time and StatusText the actual free space at time of snapshot). There are separate status nodes for ApplicationBackup, EssentialBackup and OnlineCopy (Online backup). The status is OK (green) when the backup procedure has succeeded in taking a backup as scheduled.

Online backup, essential backup and application backup are executed using the scheduled task:

```
%APP_ROOT%\bin\APP_Run_PrepareDatabaseForOnlineBackup.bat
```

It is up to the RTDB-Scheduler to run the task cyclically. The task definitions for RTDB-Scheduler are available in TimedTasks class (a list is available from Vtrin tree in Maintenance \System\Service Parameters\Timed Tasks). The task used to schedule daily backups can be found by filtering the TaskSpec property with the string *backup*. From the ScheduleTime and IntervalSecs properties, the user can see that backups are executed every day starting from 21:12:20 local server time. Backing up valuable process data can actually cause large relative proportion from the disk load in system. If the base disk load is already high, e.g. because of extensive input data production, the backup procedure might not be completed within one day. In this case, the next backup cycle is skipped and the existing one is completed before next one is started.

To take scheduled daily backup out of use choose **Edit**, replace TaskSpec command QuickRun with command Comment and click **Commit**.

Suggested plan for taking tape backups:

If the user has sophisticated backup software and enough hardware resources, daily backups should be taken from all disks of the SYS600 Historian server. However, backing up the

database disk should exclude the *.table* files, unless the backup software can back them up without locking them.

For systems where the capabilities of backup software or of available hardware resources for tape backups are limited, the following steps should be taken:

- Back up all disks before and after product software upgrades.
- Back up the contents of the backup disk daily.
- Backup up the database disk once a month. Exclude the *.table* files unless the backup software can back them up without locking them.

6.7.5 Restoring the SYS600 Historian Database from Backup

6.7.5.1 Restoring offline backups

To restore a backup that has been taken while SYS600 Historian was shut down is straightforward:

- Shut down SYS600 Historian first.
- Restore the backup save sets from the tape to their original location.
- Restart SYS600 Historian.

6.7.5.2 Restoring SYS600 Historian from online backups

To restore SYS600 Historian online backup that was taken while SYS600 Historian was running requires the following steps:

- Shut down SYS600 Historian
- Restore the backup save sets from the tape to their original location (both the data directory and the OnLineCopy directory).
- Prepare the online copy for restoration from cmd by running:

  1. CD /D %app_backup_root%\onlinecopy

- Running this instructs running the following or similar:
  "C:\Program Files (x86)\ABB Oy\RTDB\config\RTDB_RestoreFromOnlineBackup.bat" "D:\sc\Historian" /from_onlinebackup "E:\HistorianBackup\Onlinecopy"

- Run RTDB_RestoreFromOnlineBackup.bat to replace files in database directory with the files in online backup. When prompted, confirm the action with Y to continue.
- Restart SYS600 Historian.

6.7.5.3 Restoring backups from live backups from tape

To restore a database that has been created by backing up a live database:

- Shut down SYS600 Historian
- Restore the backup save sets from the tape to their original location
- Run a paranoid scan (RTDB_ScanDB –ap *), see Section 6.7.6
- Restart SYS600 Historian.

6.7.6 Scanning the database tables

When SYS600 Historian starts up or shuts down, the RTDB_ScanDB.exe verifies the low level database integrity by running RTDB_ScanDB. The purpose of RTDB_ScanDB is to ensure that the database tables are in good shape when SYS600 Historian is started up again.

The behavior of the RTDB_ScanDB is different during shutdown and startup. In shutdown, it is running in peek mode. The purpose is to find out which tables are closed successfully, which tables are open (for example, because some application has crashed during it has kept the database table open) and which tables are broken. The table is concluded to be broken if the RTDB_ScanDB is not able to open it. The purpose of the scan is to accelerate the starting of SYS600 Historian.

Based on the information received during shutdown and based on the information available in the configuration file:

```
%APP_DATAPATH%\ScanDB.ini
```

RTDB_ScanDB scans through all tables at startup. The scan order is defined in the configuration file and is selected to be such that the tables that are required to be available for input data producing (e.g. current values and variables) and for user interface support (UI classes, UI properties, UI strings) are scanned first. Old split files of historical data are scanned last as this data is seldom needed. If the tables are marked to be corrupted (or after abrupt shutdown of the machine), the corrupted tables are scanned using extensive paranoid scan settings. In order to get even more confidence that data is available as it should, some tables are also scanned using extensive checks even if the table looks fine based on the header information, based on the settings in ScanDB.ini. The settings available in ScanDB.ini file should not be changed without a very good reason or without deep experience of the SYS600 Historian administration.

If the startup is executed after running SYS600 Historian or operating system down in a controlled way, RTDB_ScanDB will perform fast checks. If a computer crashes and a database table is open that time, RTDB_ScanDB enables extensive checks to restore the integrity of the table. However, if a malfunctioning disk or software has corrupted a database table file, RTDB_ScanDB may not notice the error without activating it with so called paranoid mode. In these kinds of situations, or because a low level administration task requires it, RTDB_ScanDB troubleshooting can be run from command line. To get help on different settings, run:

```
RTDB_ScanDB.exe
```

In order to run the RTDB_ScanDB interactively, SYS600 Historian should be stopped first. After that, the command line can be started from the database disk and from a data base directory (cd %app_datapath%) using (elevated) administrator privileges.

The most used commands are:

```
RTDB_ScanDB *
```

which scans through all tables in the current directory,

```
RTDB_ScanDB –ap *
```

which paranoid-scans through the whole database (running paranoid scan through all tables can take long time, depending on how much data there is in the database),

```
RTDB_ScanDB <tablename>
```

for example

```
RTDB_ScanDB variables
```

that only scans a specified table. In production environment, it is good to remember that tables can be copied to temporary folders where RTDB_ScanDB is executed. This makes analyzing and fixing the problems possible while SYS600 Historian is running.

6.7.7 Managing Client Software distribution

Microsoft ClickOnce technology can be used to make the Vtrin client available on another computer besides the actual server. On the server side, the "Vtrin Server" service acts as the web server that serves the ClickOnce installations. However, if Internet Information Server (IIS) is installed on the computer, the IIS serves the installations instead.

On the client side, .Net framework 4.5 needs to be installed (if not installed already). Additionally a self-signed certificate for server authentication created by Historian installation needs to be added to the local computer certificate store for Trusted Root Certification Authorities. It can be done for example by right-clicking the file and selecting correct store manually or by running mmc.exe, adding certificate management snap-in for local computer and importing to correct store.

The certificate is available in the Historian server application setup folder (%APP_ROOT% \Setup\vtrincert.cer) or it can be exported without private key in base-64 encoded format from the server local computer certificate store for Personal certificates (Friendly Name is vtrincert).

Next a connection to following web address needs to be opened using Internet Explorer browser (computer name is required, IP address is not accepted):

https://<Historian computer>/vtrin

This should start the downloading process. Figure 54 illustrates the download dialog.

The **Run** button activates the downloading binaries from the server. After the download is completed, a Vtrin connect dialog is opened automatically (Figure 55). Once the binaries have been downloaded, the new ClickOnce requests to automatically use those without loading new ones from server.

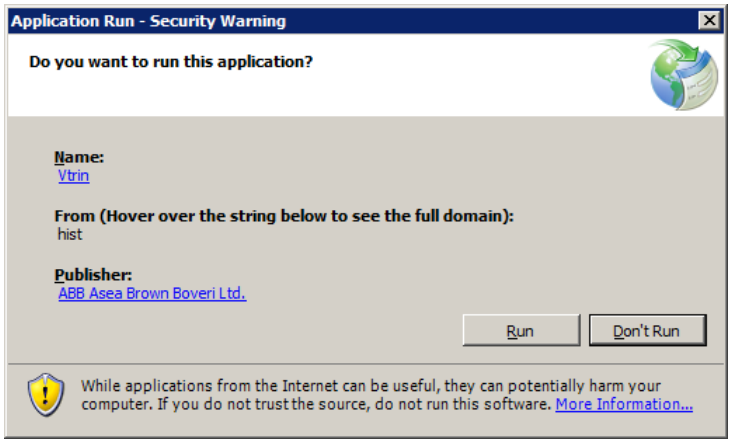


Figure 54: Dialog opened to confirm file download from server.

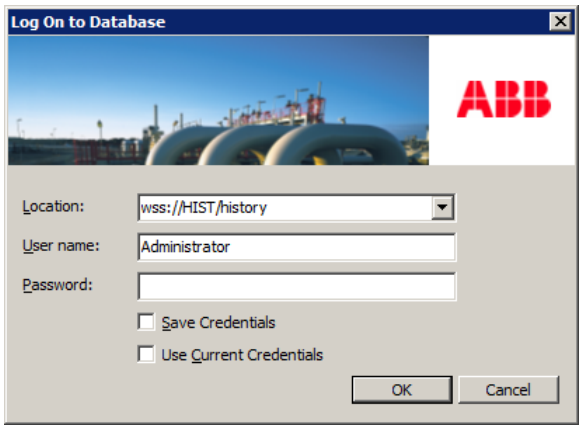


Figure 55: Connect dialog opened after succesfully downloading binaries from server.

Another possibility to distribute Vtrin client is to copy it manually from the server folder C:\Program Files (x86)\ABB Oy\Vtrin to some client computer folder. The required files are VtrinStub.exe, GKS.dll and GKS_x64.dll.

In this case (without adding self-signed server authentication certificate to trusted store) a security warning is displayed when a connection to Historian is opened. It is now possible to let Vtrin trust the certificate. This information will be stored into C:\Users\<username>\AppData\Roaming\ABB\Vtrin\keys.txt file and with the next connection the security warning is not displayed any more.

6.7.8 Upgrading Historian

SYS600 Historian can be upgraded from earlier versions. Over installing SYS600 Historian on an existing old version will automatically upgrade the services and database. The data will also be migrated automatically.

It is advised to take backup of Database and Software Files (see Section 6.7.4) of SYS600 Historian before attempting upgrade

Section 7 Uninstalling SYS600 Historian

7.1 Uninstalling

Open SYS600 Historian Setup application from the Windows Start Menu items for MicroSCADA X Control System.

Choose uninstall operation and click **Uninstall**.

Next uninstall ABB SYS600 Historian product from the Windows Programs and Features list.

Further delete the following files/folders:

```
C:\sc\Historian
C:\sc\HistorianApp
C:\sc\HistorianBackup
C:\inetpub\wwwroot\Vtrin
C:\Program Files\ABB Oy
C:\Program Files (x86)\ABB Oy
C:\ProgramData\ABB\Licensing
C:\Users\<user>\ABB Oy
C:\Users\<user>\AppData\Local\ABB Oy
C:\Users\<user>\AppData\Local\IsolatedStorage
C:\Users\<user>\AppData\Roaming\ABB\Vtrin
SYS600 Historian Control Panel (Desktop shortcut)
Control Panel\All Control Panel Items\Administrative Tools\SYS600
Historian Control Panel
```

Finally reboot the computer.

Hitachi ABB Power Grids
Grid Automation Products
PL 688
65101 Vaasa, Finland

https://hitachiabb-powergrids.com/microscadax

Scan this QR code to visit our website

1MRK 511 472-UEN