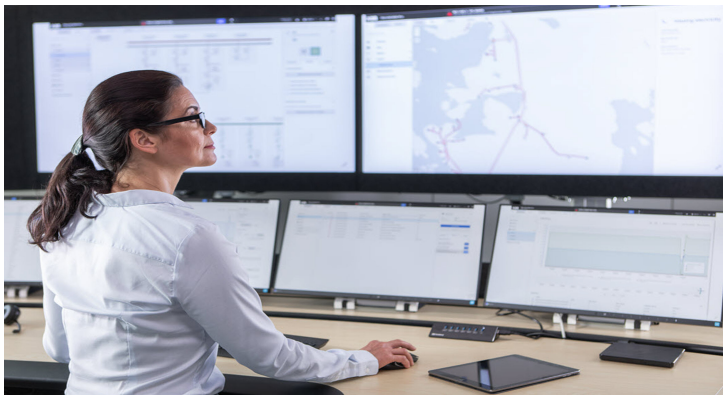
GRID AUTOMATION PRODUCTS

MicroSCADA X SYS600 10.2

OPC Server



Document ID: 1MRK 511 476-UEN
Issued: March 2021
Revision: A
Product version: 10.2

Table of contents

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 About this manual

2.1 General

This manual provides information for application programmers that build up OPC (OLE for Process Control) client applications interacting with SYS600.

The SYS600 system contains three different OPC (OLE for Process Control) server implementations based on the specifications of OPC Foundation. The documents containing these specifications are listed below in Section 2.4.

The three OPC servers are the following:

1. Data Access Server is an implementation of OPC Data Access Custom Interface Standard. It exposes all the data in the SYS600 database to its clients. There is a single instance of the server, which runs as an integrated process within the SYS600 base system. This server is described in Section 3.
2. Alarms and Events Server is an implementation of OPC Alarms and Events Custom Interface Standard. It sends notifications of significant events to its clients. Each SYS600 application runs its own instance of the server as an integrated process within the SYS600 base system. This server is described in Section 4.
3. Application OPC Server is an implementation of OPC Data Access Custom Interface Standard. It is a tunneling server that is usually run in the computer where the client application runs. It exposes the data of the connected Data Access Server to the client(s). The Application OPC Server is used in redundant (HSB) SYS600 systems. It supervises the state of an application in the redundant system. On the event of HSB takeover, it automatically connects to the new hot application. The client applications continue to run undisturbed and without loss of data. This server is described in Section 5.

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

The following SYS600 manuals should be available for reference during the use of this manual:

Name of the manual	Document ID
SYS600 10.2 Application Objects	1MRK 511 467-UEN
SYS600 10.2 System Objects	1MRK 511 482-UEN
SYS600 10.2 Programming Language SCIL	1MRK 511 479-UEN

The SYS600 OPC Server implementation is based on the following documents by OPC Foundation:

Name of the document	Version
OPC Overview	Version 1.0, October 27, 1998
OPC Common Definitions and Interfaces	Version 1.0, October 27, 1998
OPC Data Access Custom Interface Standard	Version 2.05A, June 28, 2002
OPC Alarms and Events Custom Interface Standard	Version 1.10, October 2, 2002
OPC Security Custom Interface	Version 1.0, October 17, 2000

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, for example, 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 Data Access Server

3.1 General

The SYS600 OPC Data Access Server is an implementation of the interface specification OPC Data Access Custom Interface Standard, Version 2.05A, on SYS600 system.

The main features of the Data Access Server include the following:

- All the objects of the SYS600 database are exposed by the server as OPC items. See Section 3.2.
- All the attributes of the SYS600 objects are exposed by the server as OPC items and as OPC item properties. See Section 3.3.
- The server implements three item name spaces for the different needs of different client applications. See Section 3.2.
- The server supports the dynamic object space of SYS600 by implementing special purpose items that the client applications may use to keep track of newly created and deleted objects. See Section 3.5.
- The entire functionality the SYS600 implementation language SCIL offers for reading and writing objects (evaluating object attributes, commands #SET and #MODIFY) is also available by reading and writing OPC items. See Section 3.7.9.
- The full power of the SCIL language is reached by client applications via special purpose OPC items. See Section 3.5.
- The server is capable of delivering all the changes of an item to the client applications by implementing true update rate 0 OPC item groups. See Section 3.4.
- A great effort has been put on the consistency of the data that is seen by the client applications. See Section 3.7.9 and Section 3.7.11.
- The updating of the values of OPC items is highly optimized for fast access and low overhead. See Section 3.4.
- The server fully supports multilingual SYS600 applications. See Section 3.7.1.
- The server supports authentication of SYS600 users. See Section 3.6.
- The server is fully integrated to the SYS600 base system software. It is started and stopped by the SYS600 application.
- Runtime diagnostics about the server and the connected clients is exposed as the base system attribute SYS:BOD (OPC Server Diagnostics). For more information, see the System Objects manual.

This section provides information for application programmers that build OPC data access client applications interacting with SYS600.

3.2 Item names

The Data Access Server implements three different name spaces for OPC items:

1. The Absolute name space exposes all the SYS600 objects to the clients as OPC items. The hierarchy of the name space is predefined by the OPC server. The OPC item names are equal to the SYS600 object names.
2. The Application relative name space exposes the SYS600 objects of the primary application to the clients. The name space is a subset of the absolute name space.
3. The User defined name space offers an alternative view to the SYS600 objects and their attributes. The hierarchy of the name space, as well as the item names are defined by application engineering.

3.2.1 Absolute name space

The root of the absolute name space is the SYS600 system itself.

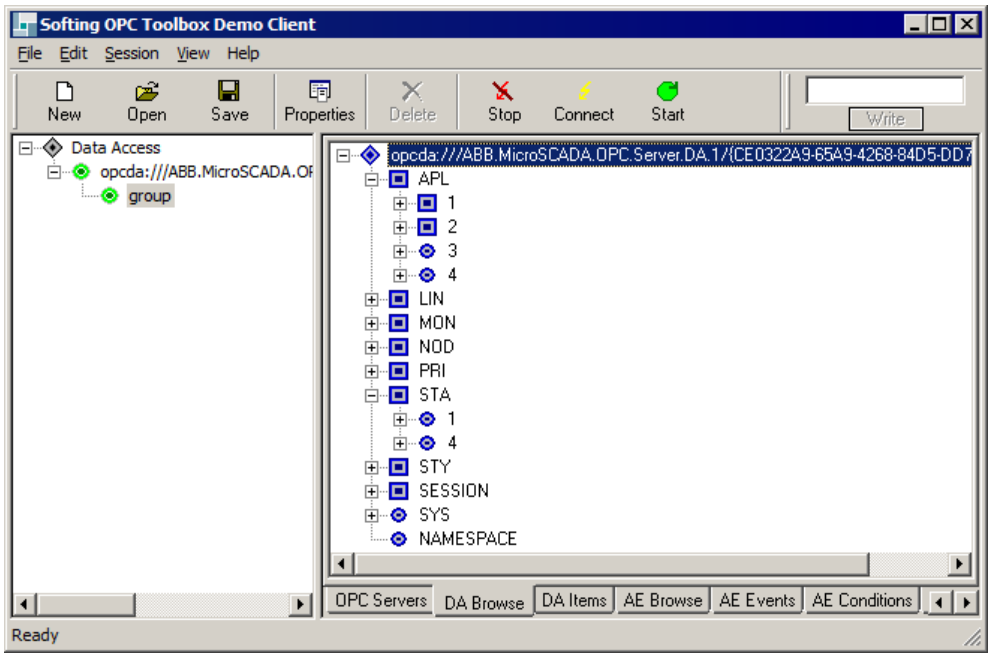


Figure 1: The absolute name space

As seen in Figure 1, the highest level items correspond to the base system (B) objects of the SCIL object name space:

APL	Application objects
LIN	Link objects
MON	Monitor objects
NOD	Node objects
PRI	Printer objects
STA	Station objects
STY	Station type objects
SYS	The system object

The last item seen in Figure 1, the NAMESPACE item, is a special purpose OPC item used to report the changes in the name space, see Section 3.5.

Two of the highest level item branches have been opened to show that there are three applications and four stations defined in the system.

The fully qualified OPC item ID begins with two backslashes (\\) to indicate that it is an absolute name space item ID. The branch item names that make up the path to the item are separated by a single backslash.

Some examples of a valid absolute item ID's are listed below:

\\SYS	The base system object SYS:B
\\STA\3	The base system object STA3:B
\\APL\1	The base system object APL1:B
\\APL\1\P\ABC\1	The process object ABC:1P1

The Windows style naming of absolute name space items (instead of the conventional naming with dot separated names often seen in OPC servers) is selected for three reasons:

- The dot separated names are reserved for the user defined name space.
- A dot is a valid character in SYS600 object names. Depending on the application engineering, it may or may not imply a hierarchy.
- The prefix \\ of the absolute name space item ID's makes a distinction between the absolute and application relative item ID's clear.

The application branches that refer to a HOT application (the branch \\APL\1 in Figure 1) open further to expose the application's data bases. As seen in the table above (\\APL\1\P\ABC\1), the application objects may be accessed by absolute name space item ID's as well.

3.2.2 Application relative name space

The root of the application relative name space is the primary application of the SYS600 system. The primary application is defined by the PA (Primary Application) attribute of the SYS object (SYS:BPA in SCIL). For more information, see the Application Objects manual.

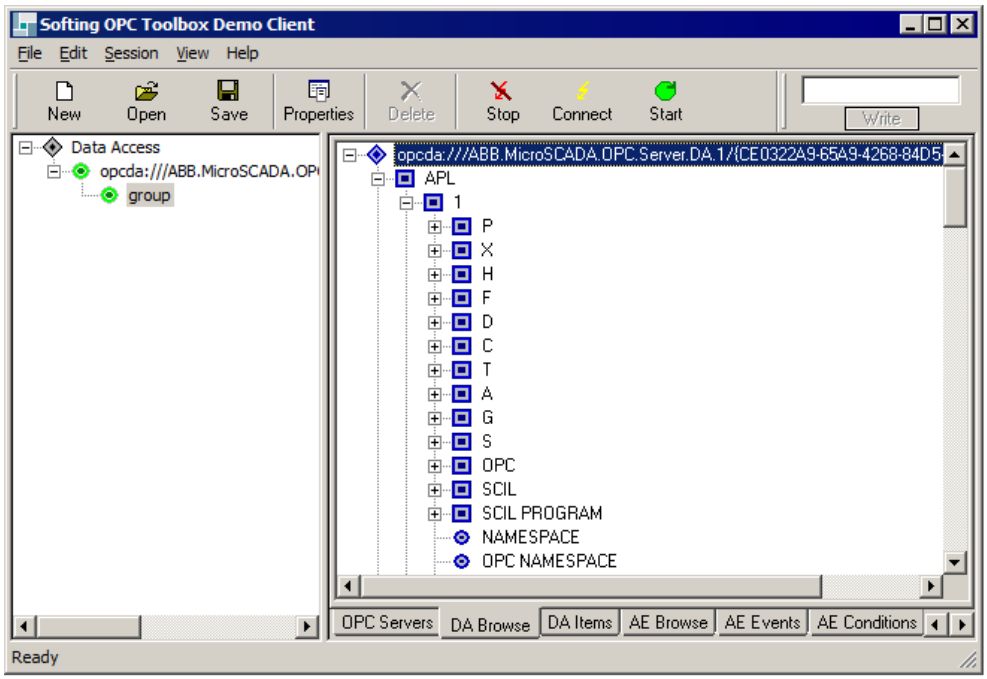


Figure 2: The application relative name space

The highest level items correspond to the application object types:

P	Process objects
H	Event handling objects
X	Scale objects
F	Free type objects

Table continues on next page

D	Data objects
C	Command procedure objects
T	Time channel objects
A	Event channel objects
G	Logging profile objects
S	System objects

The item named OPC serves as the root of the user defined name space.

The items SCIL and SCIL PROGRAM are special purpose OPC items used to interact with the SCIL language, see Section 3.5.

The last two items in Figure 2, the NAMESPACE and the OPC NAMESPACE item, are special purpose OPC items used to report the changes in the name space, see Section 3.5.

The fully qualified OPC item ID begins with a backslash (\) to indicate that it is an application relative item ID. The branch item names that make up the path to the item are separated by a single backslash.

Examples of valid application relative item ID's are listed below:

\	The application itself, that is, APLn:B, where n is the primary application
\P\ABC\1	The process object ABC:P1
\D\ABC	The data object ABC:D

All the application relative item ID's may also be given as absolute item ID's. For example, if the application 1 is the primary application, the item ID's \P\ABC\1 and \\APL\1\P\ABC\1 denote the same SYS600 object.

When creating an OPC item ID for an object that is not located in the primary application, a fully qualified absolute item ID must be used.

The top level branches of the application relative name space may be opened further to list the object names. See Figure 3.

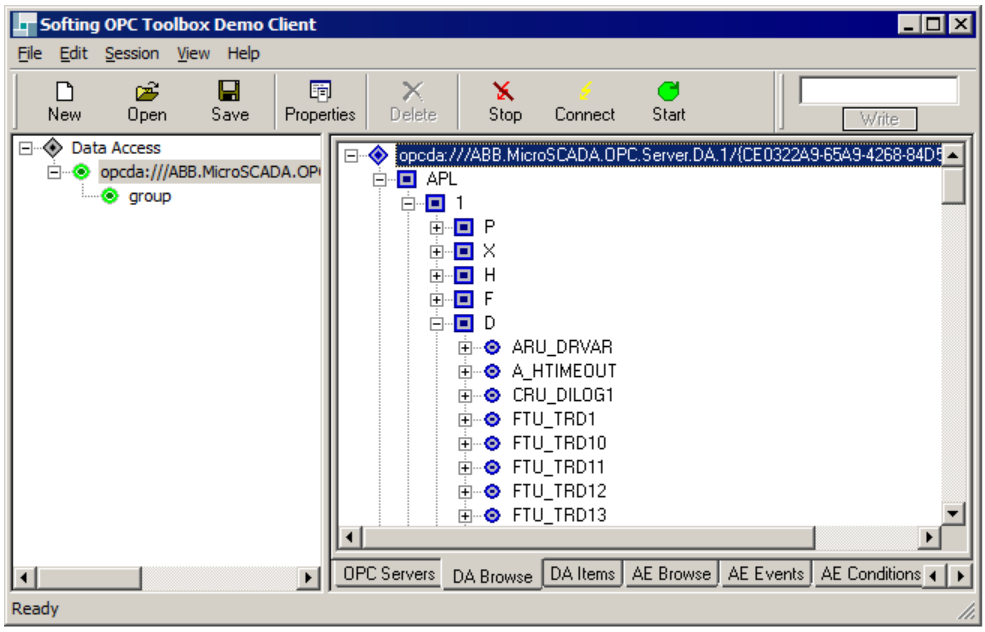


Figure 3: Data objects of the application

In case of process objects, the names may still be opened to list the indices of the object. See Figure 4.

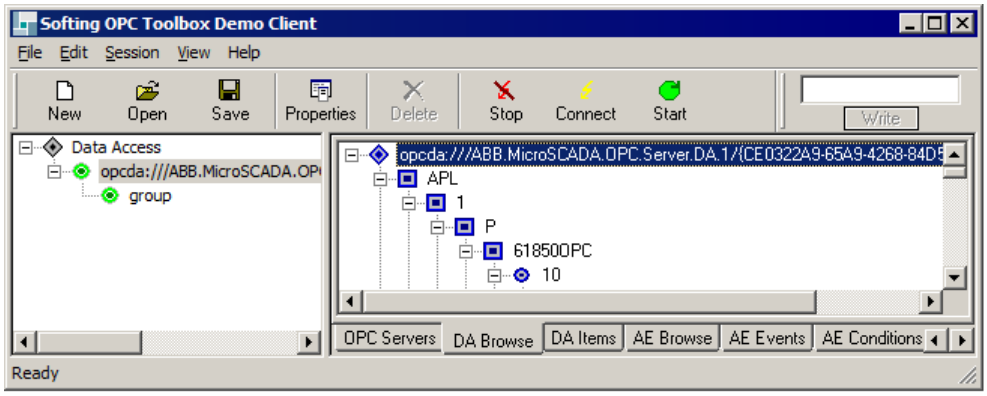


Figure 4: Item path of a process object

The system (S) objects are not shown by the browser. The reason for this is that the OPC Server (or the SYS600 base system) has no way of finding out which objects are configured in the PC-NET program. The S objects are fully managed by PC-NET. However, a client may create an OPC item ID for a system object provided that it knows the name. For example, the following are valid S object item ID's:

\S\STA5	Station 5
\S\STA5	Station 5, a synonym of \S\STA5
\S\NET\2	NET 2

The OPC server accepts all syntactically correct system object items. Only when the object is later accessed, the client may know that the object really exists in the current system.

3.2.3 User defined name space

The root of the user defined name space is the OPC item of the application.

The hierarchy and the item names of the user defined name space are defined by application engineering.

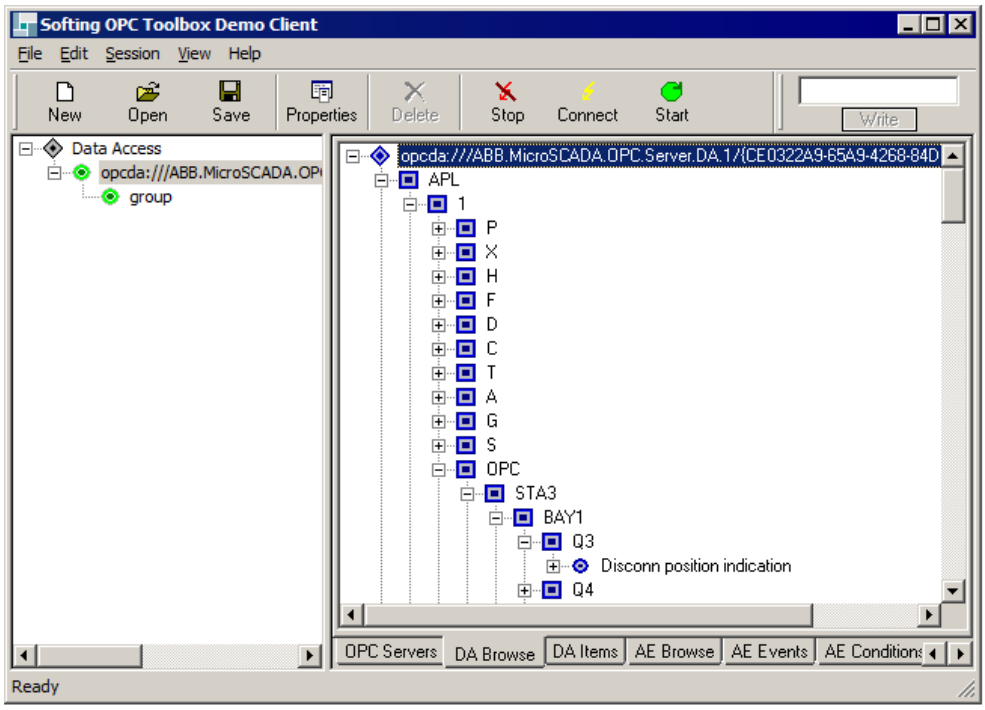


Figure 5: The user defined name space

There are two ways to define the user name space:

1. ON (OPC Item Name) attribute of process, data and command procedure objects, see the Application Objects manual.
2. The SCIL function OPC_NAME_MANAGER, see the Programming Language SCIL manual.

A user defined OPC name creates an alias name for a SYS600 object or one of its attributes.

Contrary to the SYS600 object names, the user defined OPC item names are case-sensitive and may contain any visible characters (except for colons) and embedded spaces. The hierarchy is indicated by dots.

The item ID of the disconnector position indication item depicted in Figure 5 is:

- STA3.BAY1.Q3.Disconn position indication

The same object is reached by using the other two name spaces:

- \\APL\1\OPC\STA3.BAY1.Q3.Disconn position indication
- \OPC\STA3.BAY1.Q3.Disconn position indication

3.2.4 SYS600 object attributes

An OPC name space browser shows the available items to the level of SYS600 objects. However, the SYS600 OPC Data Access Server exposes all the attributes of all the objects to the OPC clients as well. The attributes are shown as the properties of the OPC items as well as separate items.

For more information about creating an OPC item ID for the attribute of a SYS600 object, see Section 3.3.3.

3.3 Item properties

3.3.1 Overview

The SYS600 OPC Data Access Server supports the OPC defined standard properties 1–6 for every OPC item.

Items that correspond to SYS600 database objects have additional, vendor specific properties as shown in Figure 6. These properties match the SCIL attributes of the object exactly.

Some items in the SYS600 item name space are neither readable nor writeable (because they have no object value), but still have a set of properties. All the items that correspond to the base system objects of SYS600 are examples of such items (\\SYS, \\APL\1 etc.).

In summary, all the attributes of all the SYS600 objects, except for system (S type) objects, can be accessed as item properties via the Data Access Server. The attributes of the S type objects are totally managed by the PC-NET program as they are not known by the OPC server (or the SYS600 base system).

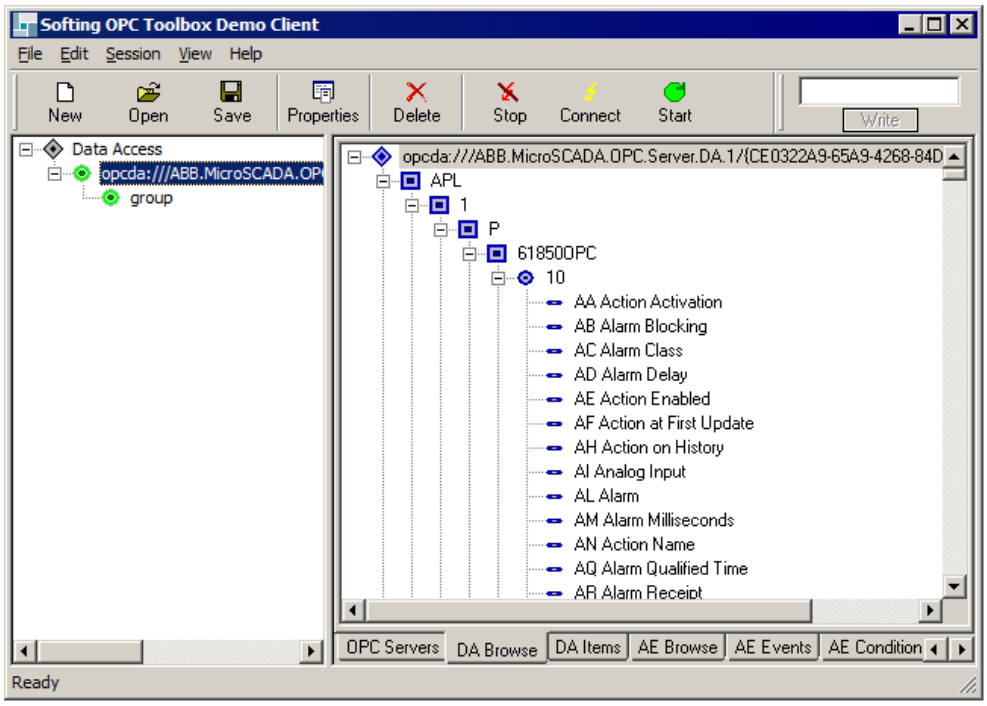


Figure 6: Properties of the process object item \\APL\1\P\61850OPC\10

3.3.2 Property ID's

According to the OPC standard, the property ID's of vendor specific properties are 5000 or greater. In SYS600 OPC Data Access Server, the property ID is calculated from the corresponding two-letter attribute name:

Attribute name	Property ID
AA	5001
AB	5002
BA	5027

Table continues on next page

Attribute name	Property ID
AZ	5026
BB	5028
ZZ	5676

Because the attribute names of SYS600 objects are object type specific, the property IDs are also item type specific. Two properties that have the same ID may or may not mean the same thing, depending on the type of the object that the item corresponds to.

The method IOPCItemProperties::QueryAvailableProperties may be used to find out the properties of an OPC item. The description of the vendor specific attributes contains the two-letter name and a short description of the corresponding attribute, as shown in Figure 6.

The method IOPCItemProperties::GetItemProperties reads the values of specified properties of an item.

3.3.3 Item ID's of the properties

Every vendor specific property of the Data Access Server may also be referred to as a separate item. The ID of the item is formed by appending a colon and the two-letter SCIL attribute name to the item ID of the containing item.

The method IOPCItemProperties::LookupItemIDs may be used to find out the item IDs for the properties.

Some valid item IDs of properties are listed below as examples:

Item ID	SCIL attribute
\P\ABC\1:AA	ABC:PAA1
\\APL\2\P\ABC1:AZ	ABC:2PAZ1
\\SYS:TI	SYS:BTI
\\APL\1:SV	APL1:BSV
\:UV	APL:BUV

If the attribute has an array value (VECTOR type in SCIL), a single element or a slice of elements may be referred to by indexing in the item name:

\\APL\1:SV(5)	APL1:BSV5 or APL1:BSV(5)
\:UV(1..5)	APL:BUV(1..5)

3.3.4 Event properties

Process objects have a few special attributes, called event history attributes, that are related to events and have no persisting value. In SCIL, they are read from the history database using the function HISTORY_DATABASE_MANAGER.

In the Data Access Server, three of the event history attributes are exposed to the clients as event properties:

- CA (Changed Attribute) tells which attribute caused the event logging.
- MX (Message Text) is the language sensitive description of the event.
- US (User Name) is the name of the MicroSCADA user who caused the event.

For example, these event attributes may be accessed as OPC items \P\ABC\1:CA, \P\ABC \1:MX and \P\ABC\1:US.

Because these attributes do not have a persistent value, a zero update rate should be used in their subscription.

In some OPC client applications, these properties may solve the requirements otherwise fulfilled only by connecting to SYS600 via the Alarms and Events Server as well.

For more information about event history attributes, see the Application Objects manual.

3.4 Item values

This section describes the update policy of items, the used data types of OPC item values, and the properties related to the values: Quality, Time stamp and Error codes.

3.4.1 Update policy

The updating of item values is highly optimized for fast access and minimal overhead. This is very important in large SYS600 systems that may contain hundreds of thousands of objects, each having dozens of properties. The number of potential OPC items is several millions. In order to do the required optimization, two update policies are implemented in the Data Access Server: REPORT and POLL.

3.4.1.1 REPORT and POLL policy

The items that have the update policy REPORT are never read cyclically from the SYS600 (polled) by the OPC server. When a client subscribes to a REPORT policy item, the OPC Server subscribes to the SYS600 object it refers to. The SYS600 base system returns the current (initial) value for the item and spontaneously sends any changes of the value to the OPC server.

The items that have the update policy POLL are not updated spontaneously by the SYS600 base system. The OPC server reads the values from SYS600 cyclically or by demand.

The REPORT policy items cause no CPU load in the system when they do not change, while POLL policy items do.

3.4.1.2 Update rate 0

By implementing the REPORT policy, the Data Access Server may offer true update rate 0 groups that are not often seen in other OPC servers. If the update rate of a group is set to 0, the server implementation guarantees that all the changes of the items in the group will be seen by the client, that is, no events are lost.

However, the clients should use update rate 0 groups carefully. Traffic between the server and the client may increase drastically when rapidly changing items are added to the group. For example, it normally makes no sense to subscribe to all the changes of rapidly fluctuating analog input objects.

3.4.1.3 Polled items

The Data Access Server implementation prefers the REPORT policy when it selects the update policy of different items. Almost all the items obey the REPORT policy.

The items that are implemented to use the POLL update policy are the following:

- The special purpose SCIL language items SCIL and SCIL PROGRAM, see Section 3.5.2.
- The system object items (the ones located in the S branch of a SYS600 application), that is, the items that are managed by the PC-NET program.
- The items that refer to an index or an index range of a SYS600 object attribute, for example \:SV4 (the system variable 4 of the primary application) or \\APL\1\D \ABC:OV(1..100) (the 100 first history registrations of the data object ABC in application 1).
- The individual base system object attributes listed in Table 1. These are mainly rapidly changing diagnostic type attributes.

Table 1: The attributes that use the POLL update policy

Object type	Attribute	SCIL name	OPC name	Description
SYS	CD	SYS:BCD	\\SYS:CD	External Clock Data
	CS	SYS:BCS	\\SYS:CS	External Clock Status
	DD	SYS:BDD	\\SYS:DD	DDE Server Diagnostics
	MF	SYS:BMF	\\SYS:MF	Memory Blocks Free
	MU	SYS:BMU	\\SYS:MU	Memory Blocks Used
	OD	SYS:BOD	\\SYS:OD	OPC Server Diagnostics
	RU	SYS:BRU	\\SYS:RU	Report Cache Used
APL	AU	APLn:BAU	\\APL\n:AU	APL-APL Server Queue Used
	EU	APLn:BEU	\\APL\n:EU	Event Queue Used
	HD	APLn:BHD	\\APL\n:HD	Host Diagnostics
	ID	APLn:BID	\\APL\n:ID	Image Diagnostics
	PS	APLn:BPS	\\APL\n:PS	Printer Spool Stop
	QO	APLn:BQO	\\APL\n:QO	Queued Objects
	QU	APLn:BQU	\\APL\n:QU	Queue Used
	RO	APLn:BRO	\\APL\n:RO	Running Objects
	RS	APLn:BRS	\\APL\n:RS	Report Task Stop
	SD	APLn:BSD	\\APL\n:SD	Shadowing Diagnostic Counters
NOD	LT	NODn:BLT	\\NOD\n:LT	Last Transaction
	RT	NODn:BRT	\\NOD\n:RT	Registration Time
LIN	DC	LINn:BDC	\\LIN\n:DC	Diagnostic Counters
PRI	CL	PRIn:BCL	\\PRI\n:CL	Printer Control
	LN	PRIn:BLN	\\PRI\n:LN	Line Number
	QU	PRIn:BQU	\\PRI\n:QU	Queue Length Used
	ST	PRIn:BST	\\PRI\n:ST	Printer State
MON	MS	MONn:BMS	\\MON\n:MS	Monitor Stop

3.4.2 Data types

3.4.2.1 Overview

This section describes the data types used by the SYS600 OPC Data Access Server. The conversion rules between the SCIL data types and the OLE data types are summarized in Table 2 below.

Table 2: Conversion rules between SCIL and OLE data types

SCIL Data Type	OLE Native Data Type	Also accepted when written or subscribed
INTEGER	VT_I4	VT_I1, VT_UI1, VT_I2, VT_UI2, VT_UI4, VT_INT, VT_UINT
REAL	VT_R4	VT_R8
BOOLEAN	VT_BOOL	
TIME	VT_DATE	
TEXT	VT_BSTR	
BIT_STRING	VT_I1 | VT_ARRAY	
BYTE_STRING	VT_UI1 | VT_ARRAY	
VECTOR of INTEGER	VT_I4 | VT_ARRAY	VT_I1 ... VT_UINT | VT_ARRAY, see above
VECTOR of REAL	VT_R4 | VT_ARRAY	VT_R8 | VT_ARRAY
VECTOR of BOOLEAN	VT_BOOL | VT_ARRAY	
VECTOR of TIME	VT_DATE | VT_ARRAY	
VECTOR of TEXT	VT_BSTR | VT_ARRAY	
VECTOR of other data types	VT_VARIANT | VT_ARRAY, see below	
VECTOR of mixed data types	VT_VARIANT | VT_ARRAY, see below	
LIST	VT_VARIANT | VT_ARRAY, see below	

3.4.2.2 Simple data types

The simple data types of SCIL (INTEGER, REAL, BOOLEAN, TIME and TEXT), as well as the vectors of simple data types, have a natural corresponding native data type in OLE (VT_I4, VT_R4, VT_BOOL, VT_DATE and VT_BSTR).

When items of these types are written or subscribed to, some close OLE data types are also accepted, see Table 2. Note that if, for example, an INTEGER item (native data type VT_I4) is added to a group with the requested data type VT_I2, the automatic data type conversion fails if the current value of the item exceeds the range of VT_I2 type.

3.4.2.3 Bit strings and byte strings

BIT_STRING and BYTE_STRING items are represented as arrays of VT_I1 and VT_UI1 data, respectively. Consequently, if integer arrays are written by the client, other array element types than VT_I1 and VT_UI1 should be used. Otherwise, SYS600 may interpret the array as a bit string or a byte string.

3.4.2.4 Vectors

For simple data type vectors, see Section 3.4.2.2 above.

The vectors of non-simple data types, as well as vectors of mixed data types, are represented as arrays of VT_VARIANT. Each element VARIANT may then be of different OLE data type.

There is no support in OLE for passing the status codes of vector elements to the client. The elements with status codes from 1 to 9 (almost OK values) are passed as good values. The elements with a status code 10 or higher are considered as missing, they are converted to OLE data type VT_EMPTY.

The indexing of OLE arrays starts from 1 as in SCIL. For every OLE array passed to the client, the SafeArrayGetLBound function of OLE returns the value 1.

The data type VT_VARIANT | VT_ARRAY is not on the list of the data types that every OPC Server should support. Therefore, general purpose clients may not be capable of displaying this data type.

3.4.2.5 Lists

OLE does not support any data type that corresponds to the LIST type of SCIL.

An artificial mapping is implemented: a list is represented as a VARIANT array, where the element VARIANTs are as follows:

1. Type VT_CY, value 0. This works as a flag to tell that a list value follows. VT_CY (currency) is not otherwise used, therefore it can be used as a flag.
2. Type VT_BSTR, the name of the first attribute.
3. The value of the first attribute, data type according to Table 2.
4. VT_BSTR, the name of the second attribute.
5. The value of the second attribute.
6. And so on.

3.4.2.6 Example

The SCIL data structure

LIST(A = 1, B = VECTOR(1, 2.5), C = LIST(D = FALSE))

is represented as an OLE array of 7 VARIANT type elements described in the table below.

Element #	VT	Value	Element #	VT	Value
1	VT_CY	0			
2	VT_BSTR	"A"			
3	VT_I4	1			
4	VT_BSTR	"B"			
5	VT_ARRAY | VT_VARIANT	array	1	VT_I4	1
			2	VT_R4	2.5
6	VT_BSTR	"C"			

Table continues on next page

Element #	VT	Value	Element #	VT	Value
7	VT_ARRAY | VT_VARIANT	array	1	VT_CY	0
			2	VT_BSTR	"D"
			3	VT_BOOL	0

3.4.3 Time stamps

The time stamps of the OPC item values refer to the time at the data source whenever possible.

For the process object values (attribute OV, Object Value), the time stamp is calculated from the RT (Registration Time) and RM (Registration Milliseconds) attributes of the object. Consequently, the time stamp displays the time of the event in the process station (relay, RTU etc.). If the station does not time stamp the value, the process database of SYS600 sets the attributes from the system clock when the value is received.

For the process object attributes that are closely related to the OV attribute and typically change along with the OV attribute, the time stamp follows the RT and RM attributes as well. These attributes are listed in Table 3.

For the dynamic data object attributes (OV (Object Value), OS (Object Status), RT (Registration Time) and QT (Qualified Registration Time)), the time stamp is calculated from the QT attribute of the object. Consequently, it tells the time when the value was logged.

The time-stamping policy of the OPC items not mentioned above depends on their update policy (see Section 3.4.1):

- The time stamp of an item, whose updating is event based (update policy REPORT) tells the time of the latest change of the value. If the value has not changed since the item was subscribed to, the time stamp tells the time of the initial read, that is, the subscription time.
- If the item is polled (update policy POLL), the time stamp tells the time of the poll that noticed the latest change of the value. The actual change of the value may have happened at any time after the previous poll.

Table 3: OV related process object attributes time-stamped by RT and RM

Attribute	Description
AL	Alarm
AM	Alarm Milliseconds
AQ	Alarm Qualified Time
AS	Alarm State
AT	Alarm Time
AZ	Alarm Zone
BL	Blocked
EP	End of Period
MM	Minimum Time Milliseconds
MQ	Minimum Qualified Time
MT	Minimum Time
OF	Overflow
OR	Out of Range
OS	Object Status
Table continues on next page	

Attribute	Description
RA	Reserved A
RB	Reserved B
RM	Registration Milliseconds
RQ	Registration Qualified Time
RT	Registration Time
SB	Substituted
SE	Selection
SP	Stop Execution
SU	Substitution State
SX	State Text
TM	Test Mode
TS	Topological State
WQ	Warning Qualified Time
XM	Maximum Time Milliseconds
XQ	Maximum Qualified Time
XT	Maximum Time
YM	Alarm On Time Milliseconds
YQ	Alarm On Qualified Time
YT	Alarm On Time
AK, CK, CM, CN, CQ, ID, NS, OQ, QU, SE, SN, VC, VM, VQ and VT	OPC Event (OE) object attributes
LE, LP, LV, LX, ND and NF	Network Topology (NT) object attributes

3.4.4 Quality

3.4.4.1 Process objects

For process object values, the OPC quality of the item value is determined by the OS (Object Status) attribute and some other attributes of the object, see Table 4 below.

Table 4: OPC Quality of process objects

OPC Quality	Condition
OPC_QUALITY_GOOD	OS = 0 or 3, OR = 0, OF = 0, SB = 0
OPC_QUALITY_SENSOR_FAILURE	OS = 1 (FAULTY_VALUE_STATUS)
OPC_QUALITY_LAST_USABLE	OS = 2 (OBSOLETE_STATUS)
OPC_QUALITY_UNCERTAIN	OS = 4 .. 9
OPC_QUALITY_NOT_CONNECTED	OS = 10 (NOT_SAMPLED_STATUS)
OPC_QUALITY_BAD	OS > 10
OPC_QUALITY_EGU_EXCEEDED	OR (Out of Range) = 1 or OF (Overflow) = 1
OPC_QUALITY_LOCAL_OVERRIDE	SB (Substituted) = 1

3.4.4.2 Other objects

For data object values, the quality is determined by the SCIL status code contained in the OS (Object Status) attribute of the object. For all the other OPC items, the quality is determined by the SCIL status code obtained from the evaluation of the item, see Table 5.

Table 5: OPC Quality vs. SCIL status code

OPC Quality	SCIL Status Code
OPC_QUALITY_GOOD	0 (OK_STATUS) or 3 (FAULTY_TIME_STATUS)
OPC_QUALITY_SUB_NORMAL	1 (SUSPICIOUS_STATUS)
OPC_QUALITY_LAST_USABLE	2 (OBSOLETE_STATUS)
OPC_QUALITY_UNCERTAIN	4 .. 9
OPC_QUALITY_NOT_CONNECTED	10 (NOT_SAMPLED_STATUS)
OPC_QUALITY_BAD	> 10

3.4.4.3 All objects

Status code 3 (FAULTY_TIME_STATUS) is reported as OPC_QUALITY_GOOD. The status code implies that the value is good but the time stamp may be inaccurate or bad. In the OPC specification, there is no quality for time stamps. Therefore, status code 3 is regarded as 0 (OK_STATUS).

Whenever the OPC quality is OPC_QUALITY_BAD (non-specific bad quality), a vendor specific error code is supplied to give more information about the failure.

3.4.5 Error codes

In cases where the OPC standard allows the server to return a vendor specific error code, the SYS600 OPC Data Access Server supplies one. The OPC functions that may return vendor specific error codes are the following:

- IOPCItemProperties::GetItemProperties
- IOPCSyncIO::Read
- IOPCSyncIO::Write
- IOPCAsyncIO2::Read
- IOPCAsyncIO2::Write
- IOPCDataCallback::OnDataChange
- IOPCDataCallback::OnReadComplete
- IOPCDataCallback::OnWriteComplete

The OPC standard does not allow vendor specific error codes to be returned by the following functions:

- IOPCItemMgt::AddItems
- IOPCItemMgt::ValidateItems

In case of a failure, these functions are allowed to return only one of the error codes OPC_E_INVALIDITEMID, OPC_E_UNKNOWNITEMID, OPC_E_BADTYPE or E_FAIL. Consequently, the server cannot tell the specific reason why the function failed.

The vendor specific error codes of the SYS600 OPC Data Access Server are basically SCIL error codes generated by the SYS600 base system. According to the OPC rules, the HRESULT type OPC error code is generated by adding the vendor error code mask 0xC0048000 to the SCIL error code. If the OPC error code is 0xC0048806, the corresponding SCIL error code is 0x806 = 2054 (PROF_OBJECT_NOT_IN_USE).

The mnemonic name for a vendor specific error code is obtained by IOPCCommon::GetErrorString or IOPCServer::GetErrorString. As an example, these functions return the string PROF_OBJECT_NOT_IN_USE, when invoked with the dwError parameter value 0xC0048806.

3.5 Special purpose OPC items

Normally, an OPC item refers to a SYS600 object or its attribute.

For special purposes, three other kinds of items are implemented in the SYS600 OPC Data Access Server:

- NAMESPACE items are used to keep track of the dynamic object space of SYS600.
- SCIL items expose the SCIL language to be used by any OPC client.
- SESSION items are client specific items that contain user and login related data of current session.

These items are described in this section.

3.5.1 NAMESPACE items

The OPC item name space of the Data Access Server is dynamic, because SYS600 objects may be created, renamed or deleted at any time.

Some OPC clients, for example a SYS600 Object Navigator, may want to keep track of all existing objects in real time. Because the OPC standard does not offer any means to manage dynamic item name spaces, special purpose OPC items called NAMESPACE and OPC NAMESPACE are implemented for the purpose.

There are several NAMESPACE items in the system, one for the base system objects and one for each hot application. In addition, each hot application has an OPC NAMESPACE item.

3.5.1.1 The NAMESPACE item for the base system objects

The top level NAMESPACE item \\NAMESPACE keeps track of the created and deleted base system (B) objects, see Figure 7.

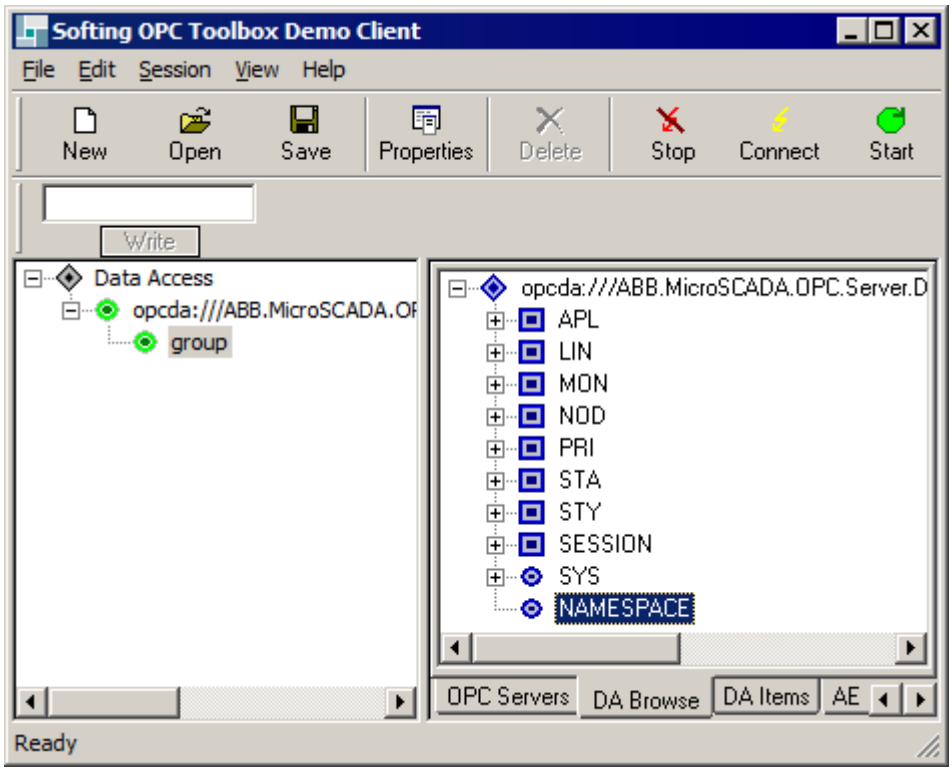


Figure 7: The NAMESPACE item for the base system objects

The item has a text (BSTR) value that tells the latest change in the base system object name space (or is an empty string). The value begins with a character + (created) or - (deleted), followed by the name of the created or deleted OPC item.

For example, the value +\\STA\23 indicates that the station object \\STA\23 (SCIL name STA23:B) has been created.

The base system objects are predefined in the SYS600: APL1 ... APL250, STA1 ... STA50000 etc. However, most of them are typically unused in any practical system.

The Data Access Server considers a base system object to be existing according to the following attribute values:

- APL (Application) objects: TT (Translation Type) <> "NONE"
- LIN (Link) objects: LT (Link Type) <> "NONE"
- MON (Monitor) objects: TT (Translation Type) <> "NONE"
- NOD (Node) objects: LI (Link Number) <> 0
- PRI (Printer) objects: TT (Translation Type) <> "NONE"
- STA (Station) objects: TT (Translation Type) <> "NONE"
- STY (Station Type) objects are predefined, they exist always

Note also that the browser of the Data Access Server shows only the base system objects that exist in the sense described above.

3.5.1.2 The NAMESPACE items for the application objects

Each hot application has two name space items, NAMESPACE and OPC NAMESPACE to report changes in the application object space, see Figure 8.

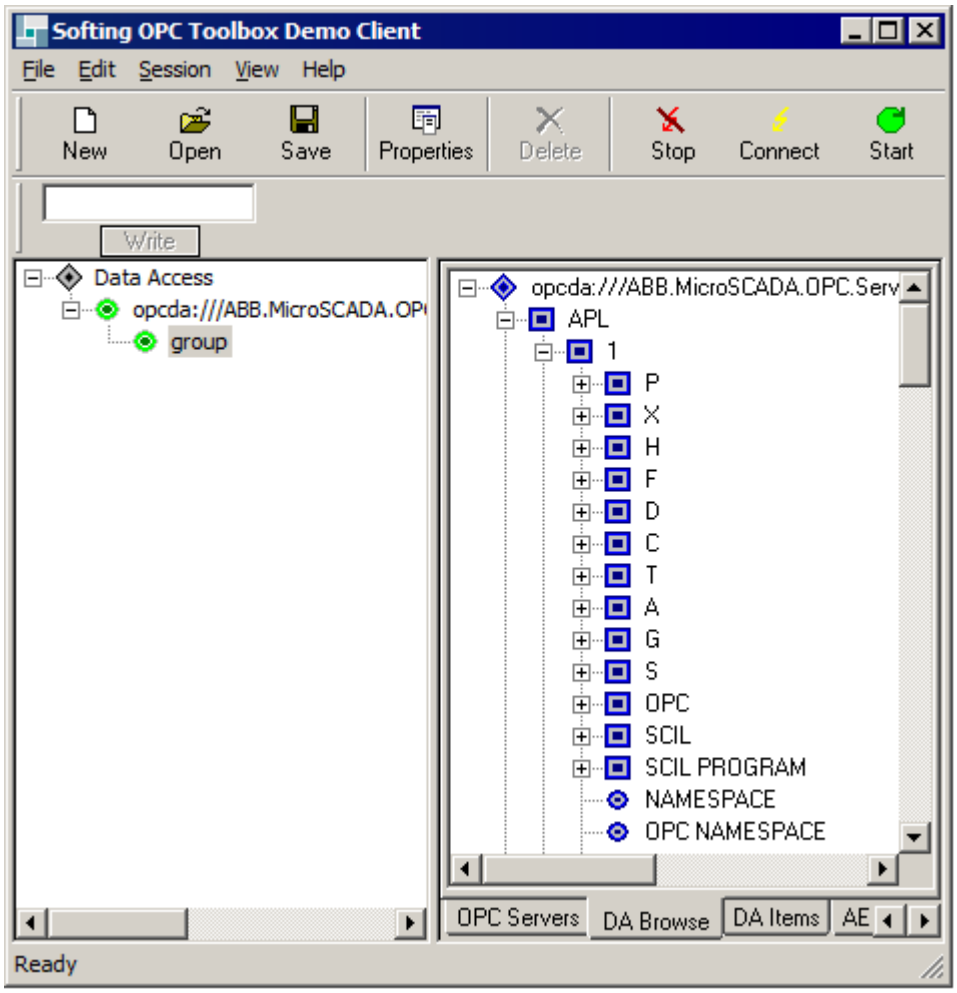


Figure 8: The NAMESPACE items for the application objects

The NAMESPACE item has a text (BSTR) value that tells the latest change in the application object name space (or is an empty string). The value begins with a character + (created) or - (deleted), followed by the name of the created or deleted OPC item.

For example, the value +\\APL\1\P\ABC\3 indicates that the process object \\APL\1\P\ABC \3 (SCIL name ABC:1P3) has been created.

When an object is renamed, two NAMESPACE events are generated: a deleted event for the old name and a created event for the new name.

The OPC NAMESPACE item has a text (BSTR) value that tells the latest change in the user defined OPC item name space (or is an empty string). The value begins with a character + (created) or - (deleted), followed by the name of the created or deleted OPC item.

For example, the value +\\APL\1\OPC\Station1.Feeder2 indicates that the user defined OPC item name Station1.Feeder2 has been created in the application.

3.5.1.3 Using NAMESPACE items

To keep track of all the application objects of application 1, the client application should do the following:

1. Create a group for the name space event(s). It is important to set the update rate of the group to 0, otherwise name space events may be lost.
2. Add the item \\APL\1\NAMESPACE (and/or \\APL\1\OPC NAMESPACE, if user defined OPC item names are used) to the group.
3. By using the interface IOPCBrowseServerAddressSpace, create an image of the current application object space.
4. On each OnDataChange callback of the name space item, update the image accordingly.

Note the order of the steps above: The subscription to the NAMESPACE item must be done before the browsing, otherwise changes that occur during the browsing or shortly thereafter may be lost.

To keep track of all the objects of a SYS600 system, the client application should do the following:

1. Create a group for the name space events. It is important to set the update rate of the group to 0, otherwise name space events may be lost.
2. Add the item \\NAMESPACE to the group.
3. By using the interface IOPCBrowseServerAddressSpace, create an image of existing base system objects.
4. For each existing application object n, subscribe to its AS (Application State) attribute, that is, add item \\APL\n:AS to the group.
5. For each hot application (AS = "HOT"), start keeping track of its objects as described in the example above.
6. Update the image according to the changes reported by the name space items and the \ \APL\n:AS items.

3.5.2 SCIL language items

The SCIL language items expose the full power of the SYS600 programming language SCIL to OPC clients. Each hot application has two read-only items, SCIL and SCIL PROGRAM, for the purpose, see Figure 8.

SCIL and SCIL PROGRAM are not complete item ID's that may be added to a group. The complete item name is formed by appending a SCIL language element, a SCIL expression or a SCIL program, to the name.

The SCIL language items are evaluated in the context of the application where they are located.

3.5.2.1 SCIL item

The SCIL item is used to evaluate any SCIL language expression.

The OPC item id is formed by appending the expression to the SCIL item ID:

- \\APL\2\SCIL\1 + 2 evaluates the SCIL expression "1 + 2" in the context of application 2. It will, of course, always evaluate to the integer value 3.
- \SCIL\APPLICATION_OBJECT_COUNT(0,"D") invokes the APPLICATION_OBJECT_COUNT function in the primary application and evaluates to the number of data objects in the application.

The expression may contain any valid SCIL language characters and it may be of any length, provided that the length of the item id does not exceed 65 535 characters.

SCIL items do not have any canonical (or native) data type, the data type may be determined only by evaluating the item. The canonical data type returned by IOPCItemMgt::AddItems and ValidateItems is VT_EMPTY.

3.5.2.2 SCIL PROGRAM item

The SCIL PROGRAM item is used to execute any SCIL program. The value of the item is the value returned by the program (SCIL statement #return).

The OPC item id is formed by appending the program to the SCIL item id:

- \\APL\2\SCIL PROGRAM\#return 1 + 2 executes the single-line SCIL program "#return 1 + 2" in the context of application 2. It will, of course, always evaluate to the integer value 3.
- \SCIL PROGRAM\#create ABC:P1 = list(PT = 3) creates a new process object in the primary application. The program does not return a value, so the item value will be empty (value type = VT_EMPTY).
- \SCIL PROGRAM\#if HOUR < 12 #return "MORNING" | #else #return "AFTERNOON" evaluates to the text "MORNING" or "AFTERNOON", depending on the hour.

As seen in the last example, the program may contain more than one line. The lines are separated by an NL (New Line) character (ASCII 10) or a CR-NL character pair (CR stands for Carriage Return, ASCII 13). In the example, the separator is shown as |.

The program may contain any valid SCIL language characters and it may be of any length.

The SCIL PROGRAM items do not have any canonical (or native) data type, the data type may be determined only by evaluating the item (executing the program). The canonical data type returned by IOPCItemMgt::AddItems and ValidateItems is VT_EMPTY.

When a SCIL PROGRAM item is added to a group, the SCIL program specified by the item is syntax checked and compiled. If the program does not compile, error code E_FAIL is returned and the item is not added to the group. The OPC standard does not allow vendor specific error codes to be returned by AddItems. Therefore, the server is unable to tell even the SCIL status code of the failure.

When a SCIL PROGRAM item is evaluated, the compiled version of the program is executed. If no execution error is encountered, the value of the item is set to the value returned by the terminating #return statement of the program, or to an empty value if there is no such statement. If the execution terminates to an error, the quality of the item is set to OPC_QUALITY_BAD and the SCIL status code is returned as the auxiliary error code.

Because of the limited error reporting possibilities, it is recommended that any SCIL program to be executed via OPC is first thoroughly tested in the native SYS600 environment.

3.5.2.3 Using SCIL language items

A typical way to use SCIL language items in an OPC client program comprises the following steps:

1. Create an inactive group for the items.
2. Add each item to the group, check for compilation errors of SCIL PROGRAM items.
3. Use IOPCSyncIO::Read (or IOPCASyncIO2::Read) to evaluate the items, when required.

SCIL language items may not be added to a group, whose update rate is 0.

3.5.3 SESSION items

Session items are located in the absolute item branch \\SESSION. They contain information about current SYS600 user session. Each OPC client sees only its own session data.

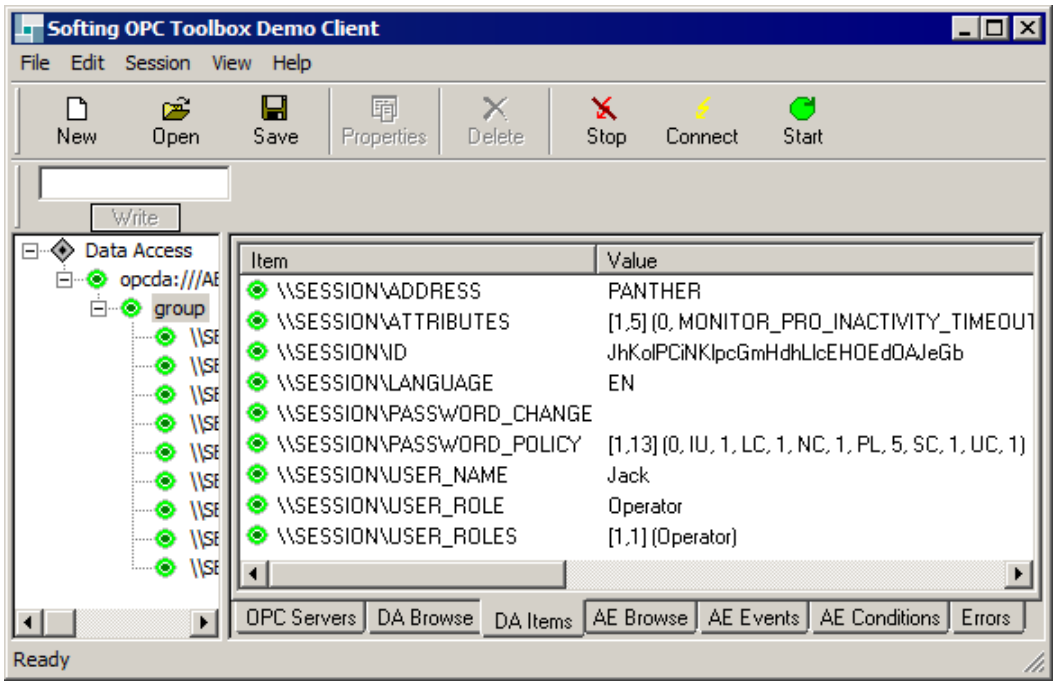


Figure 9: Session items

The items in the \\SESSION branch are the following:

Item name	SCIL data type	Description
ADDRESS	Text	The name or IP address of the client computer
AOR_DATA	Vector	Contains session specific **Area of Responsibility** (AoR) and **Exclusive Access Rights** (EAR) data. The content of the vector varies according to the current AoR configuration.
ATTRIBUTES	List	Monitor Pro/Workplace X specific session timeout attributes For details, see the Programming Language SCIL manual, function USM_SESSION_ATTRIBUTES.
ID	Text	The id of the session
LANGUAGE	Text	Two-letter abbreviation of the user's preferred language
PASSWORD_CHANGE	Text	Reason to change password, if any. For details, see the Programming Language SCIL manual, function USM_PASSWORD_CHANGE.
PASSWORD_POLICY	List	The password policy of the application For details, see the Programming Language SCIL manual, function USM_PASSWORD_POLICY.
USER_NAME	Text	Name of the user
USER_ROLE	Text	The role of the user in this session
USER_ROLES	Text vector	Available roles of the user

3.5.4 Public data items

SYS600 OPC server provides public data to all OPC clients without user authentication. This data is provided via IOPCPublicGroups interface. The name of the public group is "PublicSysGrp". It contains the following SYS600 system specific attributes:

- SYS:PA, Primary Application
- SYS:NN, Node Name
- SYS:AL, Application List
- SYS:CX, Comment Text
- SYS:CT, Current Time
- SYS:RE, Program Revision
- SYS:TF, Time Format
- SYS:WA, Web Address

3.6 User authentication

By default, SYS600 OPC Data Access Server requires OPC client authentication before any process data access can take place. OPC Client authentication requirement is controlled by SYS600 hardening attribute SYS:BHD. OPC Client authentication must be done before any OPC groups are created. Attempt to create OPC groups without authentication will fail with error code E_ACCESSDENIED.

SYS600 OPC Data Access Server implements optional IOPCSecurityPrivate interface for OPC client authentication. IOPCSecurityPrivate interface is designed for access control using private credentials. For implementation details see Section 3.7.13.

3.7 Implementation notes

This section describes details of the implementation of the SYS600 OPC Data Access Server that may be important for the developer of a client application. The notes are listed by interfaces as defined by the OPC Data Access Custom Interface Standard (Version 2.05A) and OPC Security Custom Interface (Version 1.0).

3.7.1 IOPCCommon

3.7.1.1 LocaleIDs

The languages listed in Table 6 are returned by the QueryAvailableLocaleIDs method. This indicates that the locale IDs derived from the listed language ids can be used as an argument for the SetLocaleID method.

However, it is up to the SYS600 applications which languages are actually supported. A SYS600 system can contain several applications that support different languages. Therefore, the OPC server must support virtually all the languages. If the chosen language is not supported by the application, language-sensitive texts are read in English.

The toolkit that the Data Access Server is based on supports some LocaleIDs (English, German, Russian, Hungarian and Swedish). The OPC error codes returned by the GetErrorString method are available in these languages.

Table 6: ISO 639 language identifiers and Windows language ids

Afrikaans	AF	LANG_AFRIKAANS
Albanian	SQ	LANG_ALBANIAN
Arabic	AR	LANG_ARABIC
Armenian	HY	LANG_ARMENIAN
Assamese	AS	LANG_ASSAMESE
Azerbaijani	AZ	LANG_AZERI
Basque	EU	LANG_BASQUE
Byelorussian	BE	LANG_BELARUSIAN
Bengali	BN	LANG_BENGALI
Bulgarian	BG	LANG_BULGARIAN
Catalan	CA	LANG_CATALAN
Chinese	ZH	LANG_CHINESE
Croatian	HR	LANG_CROATIAN
Czech	CS	LANG_CZECH
Danish	DA	LANG_DANISH
Dutch	NL	LANG_DUTCH
English	EN	LANG_ENGLISH
Estonian	ET	LANG_ESTONIAN
Faroese	FO	LANG_FAEROESE
Persian	FA	LANG_FARSI
Finnish	FI	LANG_FINNISH
French	FR	LANG_FRENCH
Georgian	KA	LANG_GEORGIAN
German	DE	LANG_GERMAN
Greek	EL	LANG_GREEK
Gujarati	GU	LANG_GUJARATI
Hebrew	HE	LANG_HEBREW
Hindi	HI	LANG_HINDI
Hungarian	HU	LANG_HUNGARIAN
Icelandic	IS	LANG_ICELANDIC
Indonesian	ID	LANG_INDONESIAN
Italian	IT	LANG_ITALIAN
Japanese	JA	LANG_JAPANESE
Kannada	KN	LANG_KANNADA
Kashmiri	KS	LANG_KASHMIRI
Kazakh	KK	LANG_KAZAK
Korean	KO	LANG_KOREAN
Latvian	LV	LANG_LATVIAN
Lithuanian	LT	LANG_LITHUANIAN
Macedonian	MK	LANG_MACEDONIAN
Malay	MS	LANG_MALAY

Table continues on next page

Malayalam	ML	LANG_MALAYALAM
Marathi	MR	LANG_MARATHI
Nepali	NE	LANG_NEPALI
Norwegian	NO	LANG_NORWEGIAN
Oriya	OR	LANG_ORIYA
Polish	PL	LANG_POLISH
Portuguese	PT	LANG_PORTUGUESE
Punjabi	PA	LANG_PUNJABI
Romanian	RO	LANG_ROMANIAN
Russian	RU	LANG_RUSSIAN
Sanskrit	SA	LANG_SANSKRIT
Serbian	SR	LANG_SERBIAN
Sindhi	SD	LANG_SINDHI
Slovak	SK	LANG_SLOVAK
Slovenian	SL	LANG_SLOVENIAN
Spanish	ES	LANG_SPANISH
Swahili	SW	LANG_SWAHILI
Swedish	SV	LANG_SWEDISH
Tamil	TA	LANG_TAMIL
Tatar	TT	LANG_TATAR
Telugu	TE	LANG_TELUGU
Thai	TH	LANG_THAI
Turkish	TR	LANG_TURKISH
Ukrainian	UK	LANG_UKRAINIAN
Urdu	UR	LANG_URDU
Uzbek	UZ	LANG_UZBEK
Vietnamese	VI	LANG_VIETNAMESE
Uzbek	UZ	LANG_UZBEK
Vietnamese	VI	LANG_VIETNAMESE

3.7.1.2 IOPCCommon::GetErrorString

The method IOPCCommon::GetErrorString may be used to get a description of vendor specific (SCIL) error codes. The description is the mnemonic name of the SCIL status code as returned by the SCIL function STATUS_CODE_NAME.

3.7.2 IOPCServer

3.7.2.1 IOPCServer::AddGroup

When the dwRequestedUpdateRate parameter is set to 0, the server sends all the changes of the items connected to the group. In this case, only items with the update policy REPORT may be added to the group, see Section 3.4.1.

The dwLCID parameter can be chosen from the Table 6. For more information about localization, see Section 3.7.1.

3.7.2.2 IOPCServer::GetErrorString

The method IOPCServer::GetErrorString may be used to get a description of vendor specific (SCIL) error codes. The description is the mnemonic name of the SCIL status code as returned by the SCIL function STATUS_CODE_NAME.

3.7.3 IOPCItemProperties

3.7.3.1 Recommended properties

The recommended properties (property ID's 100 to 399) are not supported by the SYS600 OPC Data Access Server.

3.7.3.2 Vendor specific properties

The SYS600 OPC Data Access Server exposes all the attributes of SYS600 objects as vendor specific properties (property ID's > 5000), see Section 3.3.

3.7.3.3 IOPCItemProperties:: QueryAvailableProperties

The OPC Data Access Custom Interface Standard 2.05A states:

"The expected use of this is that you would pass it an ITEMID such as A100, which represents a 'record' object although you can also pass it a fully qualified ITEMID such as A100.CV or A100.SP. In any case, you will get back a list of all of the other properties related to this item; typically, these are the other properties of the record object. Except for properties 1 - 6 it is not relevant whether the starting ITEMID reflects the record object or one of its property objects. Either way you will get back the same result - that is, the list of properties in the containing 'record' object."

An intentional deviation from the standard is applied here. If the ITEMID refers to an attribute of a SYS600 object, such as \P\ABC\1:CX (the comment text of a process object), QueryAvailableProperties returns the standard properties only. Only if the ITEMID refers to the object itself or its object value attribute (\P\ABC\1, \P\ABC\1:AI or \P\ABC\1:OV), the full list of properties is returned by the function. In SYS600, it makes no sense to state, for example, that the RT (Registration Time) attribute is a property of the comment text.

3.7.3.4 IOPCItemProperties:: GetItemProperties

Whenever a property cannot be read, the server provides the reason as a vendor specific error code in the ppErrors array.

3.7.3.5 IOPCItemProperties:: LookupItemIDs

For each vendor specific property (SYS600 object attribute), the method LookupItemIDs returns an item id that can be added to a group.

For the standard properties, no item ID's are available.

3.7.4 IOPCServerPublicGroups

This optional interface contains by default one predefined public group named "PublicSysGrp". Client can use this predefined group but other public group operations, for example RemovePublicGroup method call, are not supported.

3.7.5 IOPCBrowseServerAddressSpace

This optional interface is fully implemented. A HIERARCHICAL address space is shown.

3.7.5.1 IOPCBrowseServerAddressSpace::BrowseOPCItemIDs

An intentional deviation from the standard is applied here. If the parameter dwBrowseFilterType is set to OPC_BRANCH, the parameters vtDataTypeFilter and dwAccessRightFilter are ignored. This is done because there is a big confusion among the existing OPC clients and servers about the terms "branch", "leaf" and "item". Many clients and servers erroneously assume, that item and leaf are synonymous and that a branch may not be an item.

3.7.5.2 IOPCBrowseServerAddressSpace::BrowseAccessPaths

Access paths are not relevant in SYS600. Therefore, the method BrowseAccessPaths always return S_FALSE as there is nothing to enumerate.

3.7.6 IPersistFile

This interface is irrelevant in SYS600, and therefore it is not implemented.

3.7.7 IOPCItemMgt

3.7.7.1 Blobs

Blobs are not used by the SYS600 OPC Data Access Server.

3.7.7.2 IOPCItemMgt::AddItems and IOPCItemMgt::ValidateItems

If the update rate of the group is 0, not all (otherwise valid) items may be added to group. If the update policy of the item is POLL (see Section 3.4.1), it may not be added to a group with a zero update rate. If attempted, the error code E_FAIL is returned in the ppErrors array.

There are items, whose data type is not known by the SYS600 base system before they are evaluated the first time. Such items include the system objects (S type objects of SCIL), that are totally managed by the PC-NET program, and the special SCIL language items SCIL and SCIL_PROGRAM. In these cases, the canonical data type reported by the methods is VT_EMPTY.

There are even items, whose very existence is not known by the SYS600 base system before they are evaluated for the first time. The system objects (S type objects of SCIL) are such objects. AddItems and ValidateItems only check the syntax of the item ID's. For example, the item ID \S\STA2:XX is always successfully validated. However, it is possible that STA2 is not configured in the PC-NET or that it does not have any attribute named XX.

3.7.8 IOPCGroupStateMgt

3.7.8.1 IOPCGroupStateMgt::SetState

The pLCID parameter can be chosen from the Table 6. For more information about localization, see Section 3.7.1

3.7.9 IOPCSyncIO and IOPCAsyncIO2

3.7.9.1 IOPCSyncIO::Read and IOPCAsyncIO2::Read

In the description of the ppErrors parameter, the standard states:

"NOTE any FAILED error code indicates that the corresponding Value, Quality and Time stamp are UNDEFINED."

However, the MicroSADA OPC Data Access Server supplies the proper values for Quality (OPC_QUALITY_BAD) and Time stamp. For IOPCAsyncIO2::Read, the same applies to the data returned by the OnReadComplete callback.

Whenever possible, Read is implemented as an atomic operation to achieve the consistency of the data. When consequent items within a single Read reside in the same SYS600 database (process, report or base system database), the values are guaranteed to be consistent.

For example, if the two items \P\ABC\1:AI (Analog Input) and \P\ABC\1:AZ (Alarm Zone) are read in one Read, the client may be confident to rely on the two values dating from the same process event. If they are read in two separate Read's, it is possible that a process event occurs in between, and the AZ value does not correspond to the AI value.

3.7.9.2 IOPCSyncIO::Write and IOPCAsyncIO2::Write

Whenever possible, Write is implemented as an atomic operation. When consequent items within a single Write reside in the same SYS600 database (process, report or base system database), the values seen by any reader of the same data are guaranteed to be consistent.

A Write of a single item is functionally equivalent to the SCIL command #SET.

When consecutive items within a Write refer to different attributes of the same SYS600 object, the semantics of the attributes are analyzed and the Write mimics the SCIL command #MODIFY or the so called list #SET command, whenever appropriate.

For example, if a Write sets the attributes UN and OA of a process object \P\ABC\1 to values 2 and 1000, it is equivalent to the SCIL command #MODIFY ABC:P1 = LIST(UN = 2, OA = 1000).

In another example, a single Write sets the following items:

\P\ABC\1:SE = 1

\P\ABC\1:BO = 1

\P\ABC\1:TY = 45

\P\ABC\1:CT = 6

\P\ABC\1:QL = 1

This is equivalent to the SCIL command #SET ABC:PSE1 = LIST(BO = 1, TY = 45, CT = 6, QL = 1)

The command selects the binary output object to be closed.

3.7.10 IOPCAsyncIO and IDataObject

These interfaces are obsolete interfaces defined by the OPC Data Access Custom Interface Standard, Version 1.0. They are supported by the SYS600 OPC Data Access Server, but not as thoroughly tested as the recommended Version 2.0 interfaces IOPCAsyncIO2 and IConnectionPointContainer.

3.7.11 IOPCDataCallback

3.7.11.1 IOPCDataCallback::OnDataChange

The server supplies a vendor specific error code (SCIL status code) in the pErrors array for any item that has the quality OPC_QUALITY_BAD (unspecified bad quality).

The consistency of the data as seen by the client is guaranteed. All the items that have changed due to a single transaction in SYS600 (such as a process event or a SCIL command #SET or #MODIFY) are received by the client in one OnDataChange callback.

As an example, suppose that the client has subscribed to the items \P\ABC\1, \P\ABC\1:AS and \:AC, that is, the value and the alarm state of a process object and the alarm count of the application. If a process event now causes an alarm of the object, the changed values of all the three items are received in a single OnDataChange callback.

3.7.12 IAdviseSink

This interface is an obsolete interface defined by the OPC Data Access Custom Interface Standard, Version 1.0. It is supported by the SYS600 OPC Data Access Server, but not as thoroughly tested as the recommended Version 2.0 interface IOPCDataCallback.

3.7.13 IOPCSecurityPrivate

IOPCSecurityPrivate implementation contains three methods. OPC client uses these methods to authenticate itself for secure connection:

- IsAvailablePriv
  Method is used to indicate if the OPC server requires users to authenticate before any data communication can take place. The return value of this method follows the REQUIRE_OPC_AUTHENTICATION field of the SYS600 hardening attribute SYS:BHD.
- Logon
  Method authenticates the user using provided 'User Name' and 'Password' parameters. The 'User Name' parameter is a combination of SYS600 application number and the user name separated with "\" character.
- Logoff
  Logs off the currently authenticated connection.

Example of OPC client authentication for application number 1 using user name "demo" and password "DEMO":

Logon("1\demo", "DEMO")

For detailed descriptions of each method, see OPC Security Custom Interface specification.

It is recommended that each OPC client authenticates the connection even though it is not required.

Section 4 Alarms and Events Server

4.1 General

The SYS600 OPC Alarms and Events Server (OAES) is an implementation of the interface specification OPC Alarms and Events Custom Interface Standard, Version 1.10, on the SYS600 system.

The main features of the Alarms and Events Server include the following:

- Each SYS600 application has its own instance of OAES.
- The server is fully integrated to the SYS600 base system software. It is started and stopped by the SYS600 application.
- The name space (event sources, areas, event categories, conditions, etc.) of the server is configured by application engineering.
- Conditions are associated with SYS600 process alarms, other events with SYS600 process events.
- The server supports multilingual SYS600 applications. See Section 4.7.7.

This section provides information for application programmers that build up OPC alarms and events (A&E) client applications interacting with SYS600.

4.2 Application design

The design of an A&E application is divided into client and server application design, which are made in parallel.

Sometimes the application is targeted to an existing client application. In this case, the server is designed according to the requirements of the client.

The server design is done by configuring the A&E server name space (events, event sources etc.) in the SYS600 database.

Step-by-step procedure for engineering the SYS600 database for an A&E server is as follows:

1. Design the event categories and their properties: ID, description and event type.
2. Design a grouping of process objects according to A&E events they should generate. Each process object of a group sends identical events (apart from the source name) and has the same condition (if any). Design the event and condition properties for each group (event handling attributes OS and OC).
3. Create an event handling object for each group or extend the existing event handling object with OPC A&E attributes OG, OS and OC.
4. If appropriate, create application default event handling objects for various process object types and/or attributes.
5. Design the event source name hierarchy to be used. If the default delimiters are no good, configure the APL:BOP attribute to define the delimiter characters.
6. Configure the ES (Event Source) attributes of process objects.
7. Connect each event source to its event handling object. This is done by setting the EH attribute of the process object.
8. Set the OE attribute of the application (APL:BOE) to 1. This starts the OAES process.

4.3 Connecting to the server

The class ID (CLSID) of the server, which is needed by clients when connecting to the server, is found in the server's configuration file in the SYS600 computer. The path name of the file is

```
\SC\APL\aplname\APL_\APL_OAES_aplname.INI
```

In the path name, aplname stands for the name of the SYS600 application.

For example, the class ID of the Alarms and Events Server of the application TUTOR is found in file

```
\SC\APL\TUTOR\APL_\APL_OAES_TUTOR.INI
```

The contents of the file might look like

```
CLSID={40838188-4B1F-4C10-A08C-4BD3AF66BBC7}
```

The class ID may also be found by using OPC server browsers of various freeware OPC demo clients.

4.4 Process areas and event sources

The event source names and their hierarchy within process areas are freely configurable by the application. The structure and delimiter characters used in qualified source names depends on the application design.

The fully qualified event source name of a process object is specified by the ES (Event Source) attribute of the object. The process area hierarchy shown by the OPCEventAreaBrowser object is constructed from these source names.

4.5 Conditions

OPC conditions go hand in hand with SYS600 alarms. Each process object that generates SYS600 alarms may have an associated OPC condition defined.

The OPC condition is active while the process object is alarming. The condition is acknowledged when the alarm of the process object is acknowledged. Conditions (process alarms) may be acknowledged by OPC clients and/or by conventional SCIL based tools.

Multiple state conditions are applicable only for the limit alarms and warnings of analog process objects.

The conditions and their attributes are freely configurable during the application design phase.

4.6 Events

Each process object may generate a simple or a tracking event when its value is updated. In addition, changes of several other process object attributes may generate additional events. All of these events and their attributes are freely configurable by the application.

4.7 Implementation notes

This section describes details of the implementation of the SYS600 OPC Alarms and Events Server that may be important for the developer of a client application. The notes are listed by interfaces as defined by the OPC Alarms and Events Custom Interface Standard (Version 1.10).

4.7.1 IOPCCommon

The implementation of this interface is the same as in the Data Access Server. See Section 3.7.1 for details.

4.7.2 IOPCEventServer

All the methods of the interface have been implemented, including the following optional ones:

- QuerySourceConditions
- TranslateToItemIDs
- GetConditionState
- EnableConditionByArea
- EnableConditionBySource
- DisableConditionByArea
- DisableConditionBySource
- CreateAreaBrowser

4.7.2.1 IOPCEventServer::AckCondition

The server strictly checks the arguments pftActiveTime and pdwCookie according to the specification. Therefore, the method returns the error code E_INVALIDARG or OPC_E_INVALIDTIME when these arguments do not exactly match the corresponding attributes of the condition event notification.

4.7.3 IOPCEventServer2

This optional interface has not been implemented.

4.7.4 IOPCEventAreaBrowser

This optional interface has been implemented.

4.7.5 IOPCEventSubscriptionMgt

4.7.5.1 IOPCEventSubscriptionMgt::SetFilter

The OPC specification lists the following five optional filter criteria: event type, event category, severity, process area and event source. All of them have been implemented in the Alarms and Events Server.

4.7.5.2 IOPCEventSubscriptionMgt::SelectReturnedAttributes

The Alarms and Events Server supports one vendor specific attribute, CV (current value) for all event categories. Its attribute ID is 1 and its value is the value of the attribute OV (Object

Value) of the process object at the time of event. Consequently, its value type depends on the type of the process object.

4.7.6 IOPCEventSubscriptionMgt2

This optional interface has not been implemented.

4.7.7 IOPCEventSink

4.7.7.1 IOPCEventSink::OnEvent

The following bits of the wChangeMask member of the event notification are set by the server:

- OPC_CHANGE_ACTIVE_STATE
- OPC_CHANGE_ACK_STATE
- OPC_CHANGE_ACTIVE_STATE
- OPC_CHANGE_QUALITY
- OPC_CHANGE_SUBCONDITION

The following bits are not set:

- OPC_CHANGE_SEVERITY
- OPC_CHANGE_MESSAGE
- OPC_CHANGE_ATTRIBUTE

The szMessage member of the event notification is subject to localization. It is sent in the language specified by the client (using SetLocaleID), or else in the default language of the SYS600 application.

The default value of the szActorID member of the event notification is the name of the MicroSCADA user who caused the event.

Section 5 Application OPC Server

5.1 General

The SYS600 Application OPC Server (AOPCS) is an implementation of the interface specification OPC Data Access Custom Interface Standard, Version 2.05A, on the SYS600 system.

The Application OPC Server is a proxy server that acts as a gateway between an OPC data access client and the Data Access Server of a SYS600 system. It is normally run in the computer where the client application is run and started and stopped automatically by COM. It may also run as a DCOM server in another computer, but hardly any benefit can be seen in such a configuration.

The Application OPC Server is normally used in redundant (HSB) SYS600 systems. Each AOPCS instance is associated with a SYS600 application, the home application of the instance. AOPCS monitors the state of the home application in the redundant system. On the event of HSB switch-over, it automatically connects to the newly hot application. The client applications continue to run undisturbed and without loss of data.

The Application OPC Server can also be used in a non-redundant SYS600 system. This is especially useful if the DCOM connection between the OPC client and the SYS600 Data Access Server is unreliable, since AOPCS can provide seamless handling of connection breaks, including automatic recovery of lost events.

Other reasons for using AOPCS in a non-redundant system include:

- The client application does not support DCOM servers.
- The client application does not want to bind to absolute OPC item names (containing the application number).
- If redundancy (HSB) is later introduced, no changes or reconfiguration are needed in the client application.

Because the Application OPC Server is a proxy server, the details of the OPC implementation of the Data Access Server, described in Section 3, apply to AOPCS as well. A slight difference in the semantics of application relative name spaces is described in Section 5.4. The added functionality is described in Section 5.3.

The home application and its location(s) in the SYS600 system are specified in the configuration file of the AOPCS instance, see Section 5.2.

The Application OPC Server is installed on each computer where client applications are run.

5.2 Configuration

5.2.1 Configuration file

An Application OPC Server instance is configured by the home application specific configuration file AOPCS_instance.INI located in the directory where the program AOPCS.EXE is installed. The name of the SYS600 application is typically used as the server instance name instance.

As an example, the contents of the configuration file AOPCS_DEMO.INI, which defines the access to the SYS600 application DEMO, might be the following:

```
NODE1=SYS1
NODE2=SYS2
APL=DEMO
```

The node may be defined by name or by IP address. The SYS600 application may be defined by name or by application number.

If the names or application numbers of the SYS600 applications in the two SYS600 base systems differ, two APL keywords (APL1 and APL2) are required.

In the following example, IP addresses are used and applications are specified by number:

```
NODE1=12.34.56.78
NODE2=12.34.56.79
APL1=3 APL2=4
```

In the last example, AOPCS is used in a non-redundant SYS600 system:

```
NODE=SYS600NODE
APL=DEMO
```

Two optional keywords, FAILURE_HIDING_TIME and FAILURE_RECOVERY_TIME may be provided to configure behavior during connection breaks. The default values are 60 and -1, respectively. In the following example, hiding time is set to 5 seconds and recovery time to 55 seconds.

```
FAILURE_HIDING_TIME = 5
FAILURE_RECOVERY_TIME = 55
```

For recovery time, value of -1 may be given to indicate no time limit at all.

The effects of these values are described in Section 5.3.

In cases where the SYS600 Data Access server is configured to require authentication but the client does not support OPC security, user ID and password can be provided in the AOPCS configuration file. The password can be omitted to imply a blank password.

```
USER=DEMO
PASSWORD=DEMO
```

Security features are described in more detail in Section 5.3.

5.2.2 Registration

After the configuration file has been created, the server instance must be registered in the registry of Windows. This is done by running AOPCS in command line.

```
aopcs instance -regserver
```

The DEMO instance configured above is registered by the command:

```
aopcs DEMO -regserver
```

After a successful registration, AOPCS adds its newly created COM class ID to the configuration file, for example:

```
CLSID={40838188-4B1F-4C10-A08C-4BD3AF66BBC8}
```

The same command may be used to re-register an AOPCS instance: If the configuration file already contains a CLSID line, AOPCS only registers the instance with this CLSID without generating a new one.

The class ID is used by the OPC client application to connect to the correct instance of the Application OPC Server.

The success or possible errors during registration are reported in the Windows Event Log.

5.2.3 DCOM security settings

After the AOPCS instance has been registered, the DCOM connection between the AOPCS and the SYS600 Data Access Server is configured:

1. If not already created, create a user named "MicroSCADA" on the computer running AOPCS, with the same password as in the SYS600 computers.
2. Start DCOMCNFG and find the list of registered COM servers (**Component Services/ Computers/My Computer/DCOM Config**) Select the newly registered AOPCS configuration and open its properties (**Action/Properties**). In the **Identity** tab, choose the "MicroSCADA" user and enter its password. In the **Security** tab, give the Launch and Activation Permissions to the client user and the MicroSCADA user. This is needed by the automatic DCOM server start-up.
3. Start Local Security Policy (**Control Panel/Administrative tools**). Select **Local Policies/ User Rights Assignment**. Make sure that the MicroSCADA user has the "Access this computer from the network" permission. Select **Local Policies/Security options**. Set "Network access: Sharing and security model for local accounts" to "Classic - local users authenticate as themselves".

In case of insufficient access rights, AOPCS will remain in suspended state due to lack of connection to the SYS600 server.

5.3 Functionality

5.3.1 Start-up

The Application OPC Server does not need to be explicitly started. When an OPC client tries to connect to the CLSID registered for an AOPCS instance, COM will automatically run AOPCS to serve the client. When the last client disconnects, AOPCS will shut down.

Possible errors during the start-up are reported in the Windows Event Log.

5.3.2 States

AOPCS constantly monitors the SYS600 applications specified in the INI file. The availability of a hot SYS600 application determines the state of AOPCS. Clients can query the state by the method IOPCServer::GetStatus.

During start-up, AOPCS will try to connect to a hot application. If none is found within 10 seconds, AOPCS will start in SUSPENDED (OPC_STATUS_SUSPENDED) state.

When a hot SYS600 application is available, AOPCS state is RUNNING (OPC_STATUS_RUNNING). This is the normal operational state.

When neither application of the HSB pair is hot, the SYS600 base systems are down, or their Data Access Servers are not running, the state is SUSPENDED. Most OPC interface calls return E_FAIL.

5.3.3 Security

AOPCS implements the OPC Security Custom Interface, Version 1.0, providing support for SYS600 systems that are configured to require authentication of OPC clients. Only 'private credentials' (interface IOPCSecurityPrivate) are supported. The function IOPCSecurityPrivate::IsAvailablePriv will return TRUE when AOPCS is connected to a SYS600 system at the moment, and that system requires OPC authentication. Otherwise, it will return FALSE.

Security-related behaviour of AOPCS is similar to the SYS600 Data Access Server. OPC groups can only be created after a successful logon. The OPC namespace can be browsed even without authentication.

When the client calls IOPCSecurityPrivate::Logon, AOPCS will convey the call to the SYS600 Data Access server, with the added prefix of client hostname and application number. If the logon is successful, groups can then be created with the IOPCServer::AddGroup function.

If the client calls the AddGroup function without having first logged on, AOPCS will then try to logon to the SYS600 Data Access server with the credentials defined in the AOPCS configuration file. If the logon is successful, the AddGroup function will then complete. In this case, security functionality is completely invisible to the client.

Saving the username and password in plain text in the configuration file is obviously undesirable. This feature is only provided to accommodate for OPC clients that do not support security, and would therefore be incapable of connecting to the SYS600 system otherwise.

Whenever the client has provided credentials by calling the Logon function, AOPCS always uses these credentials and ignores any credentials in the configuration file.

The credentials provided by the Logon function are not saved in AOPCS memory. Instead, AOPCS asks for a 'session ID' from the SYS600 Data Access server. AOPCS uses this ID to logon to the SYS600 Data Access server after a connection break or a HSB switch-over. In other words, the authentication issues related to a HSB switch-over or a connection break are automatically handled by AOPCS.

However, the 'session ID' expires after 10 minutes of inactivity. After a connection break longer than 10 minutes, AOPCS will be unable to logon to SYS600 Data Access server and therefore cannot truly recover from the connection break. In this case, AOPCS will simply disconnect its client. The client can then start over with a new connection and logon.

This 'session ID' functionality only applies in cases where the client has called the Logon function. When using configuration file based credentials, session expiration is not an issue.

5.3.4 HSB switch-over

When the connection to the hot application is lost for a reason or another and the other SYS600 system is available, AOPCS checks whether an HSB switch-over is taking place. If it is, AOPCS waits for the switch-over to complete and reconnects to the newly hot application.

During the reconnection, the following actions are taken:

- The items of cyclic groups (update rate > 0) are reconnected and the values that have changed since the last ones received from the previously hot application are sent to the client of AOPCS.
- The items of event based groups (update rate = 0) are reconnected and a replay request is sent to the SYS600 Data Access Server. The Data Access Server then sends back all the events of these items since the start-up of the newly hot application in the order of their occurrence. AOPCS then conveys them to its client. No events are lost.
- The pending OPC client requests, that is, the ones done during the reconnection, are completed in the newly hot application.

Effectively, the client will see nothing but a brief stop in the data callback flow and some delay in OPC interface calls made during the reconnection.

5.3.5 Connection break

A connection break is detected when AOPCS loses the hot application and no HSB switch-over is taking place. The behavior during a connection break is determined by hiding time and recovery time values set in the configuration file, see Section 5.2.

If connection to a hot application is re-established within hiding time, the connection break will not be visible to the client at all. The reconnection procedure is similar to the reconnection after an HSB switch-over: the event replaying mechanism guarantees that no events are lost for event based groups (update rate = 0).

If the connection is not re-established within hiding time, AOPCS will go to SUSPENDED state. Client callbacks, where all the items are set to UNCERTAIN quality, substatus last known, are sent to the client. The recovery time begins at this moment.

When the connection is later recovered within the recovery time, AOPCS will go back to RUNNING state and items will automatically wake up. Again, no events are lost for event based groups (update rate = 0).

Finally, if the connection cannot be established within recovery time, all items will be set to BAD quality, substatus out of service. When the connection to a hot application is later recovered, items will automatically wake up and the refreshed values are sent to the client. However, lost events will not be recovered since the recovery time has expired.

For clients that depend on the no events lost concept, it is important to understand the meaning of OPC item status. AOPCS guarantees complete event recovery after a connection break as long as item status remains UNCERTAIN, last known. Whenever events are lost for one reason or another, AOPCS reports the loss of events to the client by status BAD, out of service.

The default value of recovery time is -1, that is, no time limit at all. With this setting, items are actually never set to BAD, out of service state during a connection break but will remain in UNCERTAIN, last known state indefinitely. The only factor limiting event recovering in this case is the event buffer size in the SYS600 Data Access Server (currently appr. 60 000 events). If the event replaying after a connection break fails due to a buffer overflow, AOPCS will set items to BAD status to report that events may have been lost. After that, however, a fresh connection (without event replaying) is immediately established and new refreshed item values are sent to the client.

5.4 Item names

As a proxy server, the Application OPC Server exposes the full item name space of the connected Data Access Server to its own clients.

The three item name spaces implemented by the Data Access Server (see Section 3.2) are shown for the clients of the AOPCS as follows:

1. The absolute name space is exactly the same as seen by the direct clients of the Data Access Server. Address space browsers see the absolute name space through the escape item SYS ROOT, located below the root of the address space.
2. The application relative name space is relative to the home application of the Application OPC Server. In the Data Access Server, it is relative to the primary application of the

SYS600 system, which may or may not be the home application of AOPCS. This is the address space seen by address space browsers.

3. The user defined name space is also relative to the home application of the Application OPC Server. Address space browsers see the user defined name space through the escape item OPC, located below the root of the address space.

It is important to use application relative or user defined item names instead of absolute item names when subscribing to the data of the home application. Otherwise, the following two fundamental concepts are lost:

- Because the names and numbers of the applications that make up an HSB pair may be different in the two SYS600 systems, an absolute item reference may work in one system but fail in the other, or it may denote two logically unrelated objects in the two systems.
- Events may be lost after an HSB switch-over or a connection break because the event replaying mechanism only applies to application relative items.

Index

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 511 476-UEN