MicroSCADA Pro
SYS 600 9.2
OPC Server

Technical Description
1. About this manual ..........................................................................................5
   1.1. Copyrights ..........................................................................................5
   1.2. Trademarks ..........................................................................................5
   1.3. General ...............................................................................................5
   1.4. Related documents .............................................................................6
   1.5. Document revisions .............................................................................6

2. Data Access Server .....................................................................................7
   2.1. General ...............................................................................................7
   2.2. Item names ..........................................................................................7
      2.2.1. Absolute name space ....................................................................8
      2.2.2. Application relative name space .................................................10
      2.2.3. User defined name space ...............................................................13
      2.2.4. SYS 600 object attributes .............................................................14
   2.3. Item properties ....................................................................................14
      2.3.1. Overview ......................................................................................14
      2.3.2. Property ID's ...............................................................................15
      2.3.3. Item ID's of the properties .............................................................16
      2.3.4. Event properties ..........................................................................16
   2.4. Item values ..........................................................................................17
      2.4.1. Update policy ...............................................................................17
      2.4.2. Data types ...................................................................................18
      2.4.3. Time stamps ...............................................................................21
      2.4.4. Quality .........................................................................................22
      2.4.5. Error codes ...............................................................................23
   2.5. Special purpose OPC items ..................................................................24
      2.5.1. NAMESPACE items .......................................................................24
      2.5.2. SCIL language items ....................................................................27
   2.6. Implementation notes ..........................................................................29
      2.6.1. IOPCCommon ..............................................................................29
      2.6.2. IOPCServer ..................................................................................31
      2.6.3. IOPCItemProperties .....................................................................31
      2.6.4. IOPCServerPublicGroups .............................................................32
      2.6.5. IOPCBrowseServerAddressSpace .................................................32
      2.6.6. IPersistFile ...................................................................................33
      2.6.7. IOPCItemMgt ..............................................................................33
      2.6.8. IOPCGroupStateMgt .................................................................33
      2.6.9. IOPCSyncIO and IOPCAsyncIO2 ...............................................34
      2.6.10. IOPCAsyncIO and IDataObject ................................................35
      2.6.11. IOPCDataCallback .................................................................35
      2.6.12. IAdviseSink ...............................................................................35

3. Alarms and Events Server ..........................................................................37
3.1. General ................................................................. 37
3.2. Application design .................................................. 37
3.3. Connecting to the server .......................................... 38
3.4. Process areas and event sources ......................... 38
3.5. Conditions .............................................................. 38
3.6. Events ................................................................. 38
3.7. Implementation notes .............................................. 39
    3.7.1. IOPCCommon ................................................. 39
    3.7.2. IOPCEventServer ........................................... 39
    3.7.3. IOPCEventServer2 ......................................... 39
    3.7.4. IOPCEventAreaBrowser .............................. 39
    3.7.5. IOPCEventSubscriptionMgt ................. 39
    3.7.6. IOPCEventSubscriptionMgt2 .................. 40
    3.7.7. IOPCEventSink ........................................... 40

4. Application OPC Server ........................................ 41
    4.1. General ........................................................... 41
    4.2. Configuration .................................................... 41
    4.3. Functionality ..................................................... 43
    4.4. Item names ....................................................... 45

5. Index ........................................................................ 47
1. About this manual

1.1. Copyrights

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

In no event shall ABB Oy be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Oy be liable for incidental or consequential damages arising from 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 ABB Oy, 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.

Copyright © 2006-2008 ABB Oy
All rights reserved.

1.2. Trademarks

Registrations and trademarks used in this document include:

Windows: Registered trademark of Microsoft Corporation.

1.3. General

This manual provides information for application programmers that build up OPC (OLE for Process Control) client applications interacting with SYS 600.

The SYS 600 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 1.4.

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 SYS 600 database to its clients. There is a single instance of the server, which runs as an integrated process within the SYS 600 base system. This server is described in Chapter 2.

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 SYS 600 application runs its own instance of the server as an integrated process within the SYS 600 base system. This server is described in Chapter 3.

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) SYS 600 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 Chapter 4.

1.4. Related documents

The following SYS 600 manuals should be available for reference during the use of this manual:

<table>
<thead>
<tr>
<th>Name of the manual</th>
<th>MRS number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Objects</td>
<td>1MRS756175</td>
</tr>
<tr>
<td>System Objects</td>
<td>1MRS756177</td>
</tr>
<tr>
<td>Programming Language SCIL</td>
<td>1MRS756176</td>
</tr>
</tbody>
</table>

The SYS 600 OPC Server implementation is based on the following documents by OPC Foundation:

<table>
<thead>
<tr>
<th>Name of the document</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC Overview</td>
<td>Version 1.0., October 27, 1998</td>
</tr>
<tr>
<td>OPC Common Definitions and Interfaces</td>
<td>Version 1.0., October 27, 1998</td>
</tr>
<tr>
<td>OPC Data Access Custom Interface</td>
<td>Version 2.05A., June 28, 2002</td>
</tr>
<tr>
<td>Standard</td>
<td></td>
</tr>
<tr>
<td>OPC Alarms and Events Custom Interface</td>
<td>Version 1.10, October 2, 2002</td>
</tr>
</tbody>
</table>

1.5. Document revisions

<table>
<thead>
<tr>
<th>Version</th>
<th>Revision number</th>
<th>Date</th>
<th>History</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>9.2</td>
<td>11.10.2006</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>9.2</td>
<td>28.03.2008</td>
<td>Changes for software revision 9.2 SP1</td>
</tr>
</tbody>
</table>
2. Data Access Server

2.1. General

The SYS 600 OPC Data Access Server is an implementation of the interface specification OPC Data Access Custom Interface Standard, Version 2.05A, on the SYS 600 system.

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

- All the objects of the SYS 600 database are exposed by the server as OPC items. See Section 2.2.
- All the attributes of the SYS 600 objects are exposed by the server as OPC items and as OPC item properties. See Section 2.3.
- The server implements three item name spaces for the different needs of different client applications. See Section 2.2.
- The server supports the dynamic object space of SYS 600 by implementing a few special purpose items that the client applications may use to keep track of newly created and deleted objects. See Section 2.5.
- The whole functionality that the SYS 600 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 2.6.9.
- The full power of the SCIL language is reached by client applications via some special purpose OPC items. See Section 2.5.
- 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 2.4.
- A great effort has been put on the consistency of the data that is seen by the client applications. See Sections 2.6.9 and 2.6.11.
- The updating of the values of OPC items is highly optimized for fast access and low overhead. See Section 2.4.
- The server fully supports multilingual SYS 600 applications. See Section 2.6.1.
- The server is fully integrated to the SYS 600 base system software. It is started and stopped by the SYS 600 application.

This chapter provides information for application programmers that build up OPC data access client applications interacting with SYS 600.

2.2. Item names

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

1. The absolute name space exposes all the SYS 600 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 SYS 600 object names.

2. The application relative name space exposes the SYS 600 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 SYS 600 objects and their attributes. The hierarchy of the name space, as well as the item names are defined by the application engineering.

### 2.2.1. Absolute name space

The root of the absolute name space is the SYS 600 system itself. As seen in Fig. 2.2.1.-1, the highest level items correspond to the base system (B) objects of the SCIL object name space:

- **APL**: Application objects
- **IND**: Input device 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 Fig. 2.2.1.-1, the NAMESPACE item, is a special purpose OPC item used to report the changes in the name space, see Section 2.5.
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 SYS 600 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.

The application branches that refer to a HOT application (in Fig. 2.2.1.-1, the branch \APL\1) open further to expose the application data bases. They are described in the next section in more detail. As seen in the example above (\APL\1\P\ABC\1), the application objects may be accessed by absolute name space item ID’s as well.
2.2.2. Application relative name space

The root of the application relative name space is the primary application of the SYS 600 system. The primary application is defined by the PA (Primary Application) attribute of the SYS object (SYS:BPA in SCIL). See the Application Objects manual.

The highest level items correspond to the application object types:

- **P**: Process objects
- **H**: Event handling objects
- **X**: Scale objects
- **F**: Free type objects
- **D**: Data objects
- **C**: Command procedure objects
- **T**: Time channel objects
- **A**: Event channel 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 2.5.
The last two items seen in Fig. 2.2.2.-1, the NAMESPACE and the OPC NAMESPACE item, are special purpose OPC items used to report the changes in the name space, see Section 2.5.

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.

Some examples of valid application relative item ID’s are listed below:

- \ The application itself, i.e. 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 SYS 600 object.

If you want to create 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, as seen in Fig. 2.2.2.-2.

*Fig. 2.2.2.-2 Data objects of the application*

In case of process objects, the names may still be opened to list the indices of the object, as seen in Fig. 2.2.2.-3.
Fig. 2.2.2.-3  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 (nor the SYS 600 base system) has no means to find out, which objects are configured in the PC-NET program. The S objects are fully managed by the 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\STA\5   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.
2.2.3. User defined name space

Fig. 2.2.3.-1 The user defined name space

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

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

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_MANAGE R, see the Programming Language SCIL manual.

A user defined OPC name actually creates an alias name for a SYS 600 object or one of its attributes.

Contrary to the SYS 600 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 breaker position item depicted in Fig. 2.2.3.-1 is Tipperary.Feeder 1.Breaker.Position
The same object is reached by using the other two name spaces:

- \APL\1\OPC\Tipperary.Feeder 1.Breaker.Position
- \OPC\Tipperary.Feeder 1.Breaker.Position

### 2.2.4. SYS 600 object attributes

An OPC name space browser shows the available items to the level of SYS 600 objects.

However, the SYS 600 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 creating an OPC item ID for the attribute of a SYS 600 object, see Section 2.3.3.

### 2.3. Item properties

#### 2.3.1. Overview

The SYS 600 OPC Data Access Server supports the OPC defined standard properties 1 - 6 for every OPC item.

Items that correspond to SYS 600 database objects have additional, vendor specific properties as shown in Fig. 2.3.1.-1. These properties match exactly the SCIL attributes of the object.

Some items in the SYS 600 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 SYS 600 are examples of such items (\SYS, \APL\1 etc.).

In summary, all the attributes of all the SYS 600 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; they are not known by the OPC server (or the SYS 600 base system).
Property ID's

According to the OPC standard, the property id’s of vendor specific properties are 5000 or greater.

In SYS 600 OPC Data Access Server, the property id is calculated from the corresponding two-letter attribute name:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Property ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>5001</td>
</tr>
<tr>
<td>AB</td>
<td>5002</td>
</tr>
<tr>
<td>AZ</td>
<td>5026</td>
</tr>
<tr>
<td>BA</td>
<td>5027</td>
</tr>
<tr>
<td>BB</td>
<td>5028</td>
</tr>
<tr>
<td>ZZ</td>
<td>5676</td>
</tr>
</tbody>
</table>

Because the attribute names of SYS 600 objects are object type specific, also the property ID’s are 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 Fig. 2.3.1.-1.

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

2.3.3. 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 ID’s for the properties.

Some valid item ID’s of properties are listed below as examples:

<table>
<thead>
<tr>
<th>Item ID</th>
<th>SCIL attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>\P\ABC\1:AA</td>
<td>ABC:PAA1</td>
</tr>
<tr>
<td>\APL\2:P\ABC1:AZ</td>
<td>ABC:2PAZ1</td>
</tr>
<tr>
<td>\SYS:TI</td>
<td>SYS:BTI</td>
</tr>
<tr>
<td>\APL\1:SV</td>
<td>APL1:BSV</td>
</tr>
<tr>
<td>:UV</td>
<td>APL:BUV</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Item ID</th>
<th>SCIL attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>\APL\1:SV(5)</td>
<td>APL1:BSV5 or APL1:BSV(5)</td>
</tr>
<tr>
<td>:UV(1..5)</td>
<td>APL:BUV(1..5)</td>
</tr>
</tbody>
</table>

2.3.4. 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, two 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.

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

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 SYS 600 via the Alarms and Events Server as well.

For details of event history attributes, see the Application Objects manual.
2.4. 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.

2.4.1. Update policy

The updating of item values is highly optimized for fast access and minimal overhead. This is very important in large SYS 600 systems, which may contain hundreds of thousands of objects, each having tens 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.

REPORT and POLL policy

The items that have the update policy REPORT, are never cyclically read from the SYS 600 (polled) by the OPC server. When a client subscribes to a REPORT policy item, the OPC Server subscribes to the SYS 600 object it refers to. The SYS 600 base system returns the current (initial) value for the item, and after that, sends spontaneously any changes of the value to the OPC server.

The items that have the update policy POLL, are not spontaneously updated by the SYS 600 base system. The OPC server reads the values from SYS 600 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.

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, i.e. no events are lost.

However, the clients should use update rate 0 groups carefully. The 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.

Polled items

The Data Access Server implementation prefers the REPORT policy when it selects the update policy of different items. As a matter of fact, 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 2.5.2.
- The system object items (the ones located in the S branch of a SYS 600 application), i.e. the items that are managed by the PC-NET program.
- The items that refer to an index or an index range of a SYS 600 object attribute, for example \\SV4 (the system variable 4 of the primary application) or
• The individual base system object attributes listed in Table 2.4.1-1. These are mainly rapidly changing diagnostic type attributes. No client is likely to want to see all their changes.

### Table 2.4.1-1 The attributes that use the POLL update policy

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

### 2.4.2. Data types

#### Overview

This section describes the data types used by the SYS 600 OPC Data Access Server. The conversion rules between the SCIL data types and the OLE data types are summarized in Table 2.4.2-1 below.
### Table 2.4.2-1  Conversion rules between SCIL and OLE data types

<table>
<thead>
<tr>
<th>SCIL Data Type</th>
<th>OLE Native Data Type</th>
<th>Also accepted when written or subscribed</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>VT_I4</td>
<td>VT_I1, VT_UI1, VT_I2, VT_UI2, VT_UI4, VT_INT, VT_UINT</td>
</tr>
<tr>
<td>REAL</td>
<td>VT_R4</td>
<td>VT_R8</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>VT_BOOL</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>VT_DATE</td>
<td></td>
</tr>
<tr>
<td>TEXT</td>
<td>VT_BSTR</td>
<td></td>
</tr>
<tr>
<td>BIT_STRING</td>
<td>VT_I1</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>BYTE_STRING</td>
<td>VT_UI1</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>VECTOR of INTEGER</td>
<td>VT_I4</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>VECTOR of REAL</td>
<td>VT_R4</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>VECTOR of BOOLEAN</td>
<td>VT_BOOL</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>VECTOR of TIME</td>
<td>VT_DATE</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>VECTOR of TEXT</td>
<td>VT_BSTR</td>
<td>VT_ARRAY</td>
</tr>
<tr>
<td>VECTOR of other data types</td>
<td>VT_VARIANT</td>
<td>VT_ARRAY, see below</td>
</tr>
<tr>
<td>VECTOR of mixed data types</td>
<td>VT_VARIANT</td>
<td>VT_ARRAY, see below</td>
</tr>
<tr>
<td>LIST</td>
<td>VT_VARIANT</td>
<td>VT_ARRAY, see below</td>
</tr>
</tbody>
</table>

### 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, also some close OLE data types are accepted, see Table 2.4.2-1. Note however, 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.

### 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, SYS 600 may interpret the array as a bit string or a byte string.

### Vectors

For vectors of simple data types, see ‘Simple data types’ 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.

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 VARIANT’s 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 2.4.2-1.
4. VT_BSTR, the name of the second attribute.
5. The value of the second attribute.
6. And so on.

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.

<table>
<thead>
<tr>
<th>Element #</th>
<th>VT</th>
<th>Value</th>
<th>Element #</th>
<th>VT</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VT_CY</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>VT_BSTR</td>
<td>&quot;A&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>VT_I4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>VT_BSTR</td>
<td>&quot;B&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VT_ARRAY</td>
<td>array</td>
<td>1</td>
<td>VT_I4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>VT_VARIANT</td>
<td></td>
<td>2</td>
<td>VT_R4</td>
<td>2.5</td>
</tr>
<tr>
<td>6</td>
<td>VT_BSTR</td>
<td>&quot;C&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2.4.3. 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 SYS 600 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 2.4.3-1.

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 2.4.1):

- 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, i.e. 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.
2.4.4. Quality

For the 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 2.4.4-1 below.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Alarm</td>
</tr>
<tr>
<td>AM</td>
<td>Alarm Milliseconds</td>
</tr>
<tr>
<td>AQ</td>
<td>Alarm Qualified Time</td>
</tr>
<tr>
<td>AS</td>
<td>Alarm State</td>
</tr>
<tr>
<td>AT</td>
<td>Alarm Time</td>
</tr>
<tr>
<td>AZ</td>
<td>Alarm Zone</td>
</tr>
<tr>
<td>BL</td>
<td>Blocked</td>
</tr>
<tr>
<td>EP</td>
<td>End of Period</td>
</tr>
<tr>
<td>MM</td>
<td>Minimum Time Milliseconds</td>
</tr>
<tr>
<td>MQ</td>
<td>Minimum Qualified Time</td>
</tr>
<tr>
<td>MT</td>
<td>Minimum Time</td>
</tr>
<tr>
<td>OF</td>
<td>Overflow</td>
</tr>
<tr>
<td>OR</td>
<td>Out of Range</td>
</tr>
<tr>
<td>OS</td>
<td>Object Status</td>
</tr>
<tr>
<td>RA</td>
<td>Reserved A</td>
</tr>
<tr>
<td>RB</td>
<td>Reserved B</td>
</tr>
<tr>
<td>RM</td>
<td>Registration Milliseconds</td>
</tr>
<tr>
<td>RQ</td>
<td>Registration Qualified Time</td>
</tr>
<tr>
<td>RT</td>
<td>Registration Time</td>
</tr>
<tr>
<td>SB</td>
<td>Substituted</td>
</tr>
<tr>
<td>SE</td>
<td>Selection</td>
</tr>
<tr>
<td>SP</td>
<td>Stop Execution</td>
</tr>
<tr>
<td>SU</td>
<td>Substitution State</td>
</tr>
<tr>
<td>SX</td>
<td>State Text</td>
</tr>
<tr>
<td>WQ</td>
<td>Warning Qualified Time</td>
</tr>
<tr>
<td>XM</td>
<td>Maximum Time Milliseconds</td>
</tr>
<tr>
<td>XQ</td>
<td>Maximum Qualified Time</td>
</tr>
<tr>
<td>XT</td>
<td>Maximum Time</td>
</tr>
<tr>
<td>YM</td>
<td>Alarm On Time Milliseconds</td>
</tr>
<tr>
<td>YQ</td>
<td>Alarm On Qualified Time</td>
</tr>
<tr>
<td>YT</td>
<td>Alarm On Time</td>
</tr>
</tbody>
</table>

Table 2.4.3-1  OV related process object attributes time-stamped by RT and RM
Table 2.4.4-1  OPC Quality of process objects

<table>
<thead>
<tr>
<th>OPC Quality</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC_QUALITY_GOOD</td>
<td>OS = 0, OR = 0, OF = 0, SB = 0</td>
</tr>
<tr>
<td>OPC_QUALITY_SENSOR_FAILURE</td>
<td>OS = 1 (FAULTY_VALUE_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_LAST_USABLE</td>
<td>OS = 2 (OBSOLETE_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_UNCERTAIN</td>
<td>OS = 3 .. 9</td>
</tr>
<tr>
<td>OPC_QUALITY_NOT_CONNECTED</td>
<td>OS = 10 (NOT_SAMPLED_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_BAD</td>
<td>OS &gt; 10</td>
</tr>
<tr>
<td>OPC_QUALITY_EGU_EXCEEDED</td>
<td>OR (Out of Range) = 1 or OF (Overflow) = 1</td>
</tr>
<tr>
<td>OPC_QUALITY_LOCAL_OVERRIDE</td>
<td>SB (Substituted) = 1</td>
</tr>
</tbody>
</table>

For data objects, 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 2.4.4-2.

Table 2.4.4-2  OPC Quality vs. SCIL status code

<table>
<thead>
<tr>
<th>OPC Quality</th>
<th>SCIL Status Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC_QUALITY_GOOD</td>
<td>0 (OK_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_SUB_NORMAL</td>
<td>1 (SUSPICIOUS_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_LAST_USABLE</td>
<td>2 (OBSOLETE_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_UNCERTAIN</td>
<td>3 .. 9</td>
</tr>
<tr>
<td>OPC_QUALITY_NOT_CONNECTED</td>
<td>10 (NOTSAMPLED_STATUS)</td>
</tr>
<tr>
<td>OPC_QUALITY_BAD</td>
<td>OS &gt; 10</td>
</tr>
</tbody>
</table>

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.

2.4.5. Error codes

In every case where the OPC standard allows the server to return a vendor specific error code, the SYS 600 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
Unfortunately, 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 SYS 600 OPC Data Access Server are basically SCIL error codes generated by the SYS 600 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.

2.5. Special purpose OPC items

Normally, an OPC item refers to a SYS 600 object or its attribute.

For special purposes, two other kinds of items are implemented in the SYS 600 OPC Data Access Server:

- NAMESPACE items are used to keep track of the dynamic object space of SYS 600.
- SCIL items expose the SCIL language to be used by any OPC client.

These items are described in this section.

2.5.1. NAMESPACE items

The OPC item name space of the Data Access Server is dynamic, because SYS 600 objects may be created, renamed or deleted at any time.

Some OPC clients, for example a SYS 600 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.

Actually, 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.

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 Fig. 2.5.1.-1.
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.

Actually, the base system objects are predefined in the SYS 600: 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”
- IND (Input Device) 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 always ‘exist’

Note also that the browser of the Data Access Server shows only the base system objects that exist in the sense described above.
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 Fig. 2.5.1.-2.

![Diagram of Softing OPC Toolbox Demo Client](image)

**Fig. 2.5.1.-2** 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.
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 SYS 600 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, i.e. 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.

2.5.2. SCIL language items

The SCIL language items expose the full power of the SYS 600 programming language SCIL to OPC clients. Each hot application has two read-only items, SCIL and SCIL PROGRAM, for the purpose, see Fig. 2.5.1.-2.

SCIL and SCIL PROGRAM are not actually 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.

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.

**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:

• \"\"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. Unfortunately, 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 SYS 600 environment.

**Using SCIL language items**

A typical way to use the 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.

⚠️ If you want to evaluate SCIL language items cyclically (in an active group), make sure not to overload the base system with too frequent lengthy calculations.

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

### 2.6. Implementation notes

This section describes some details of the implementation of the SYS 600 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).

#### 2.6.1. IOPCCommon

**LocaleIDs**

The languages listed in Table 2.6.1-1 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 SYS 600 applications which languages are actually supported. A SYS 600 system can contain several applications, which 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 a few LocaleIDs (English, German, Russian, Hungarian and Swedish). The OPC error codes returned by the GetErrorString method are available in these languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>ISO Code</th>
<th>Windows Language Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans</td>
<td>AF</td>
<td>LANG_AFRIKAANS</td>
</tr>
<tr>
<td>Albanian</td>
<td>SQ</td>
<td>LANG_ALBANIAN</td>
</tr>
<tr>
<td>Arabic</td>
<td>AR</td>
<td>LANG_ARABIC</td>
</tr>
<tr>
<td>Armenian</td>
<td>HY</td>
<td>LANG_ARMENIAN</td>
</tr>
<tr>
<td>Assamese</td>
<td>AS</td>
<td>LANG_ASSAMESE</td>
</tr>
<tr>
<td>Azerbaijani</td>
<td>AZ</td>
<td>LANG_AZERI</td>
</tr>
<tr>
<td>Basque</td>
<td>EU</td>
<td>LANG_BASQUE</td>
</tr>
</tbody>
</table>

Table 2.6.1-1 ISO 639 language identifiers and Windows language ids
### Table 2.6.1-1 ISO 639 language identifiers and Windows language ids

<table>
<thead>
<tr>
<th>Language</th>
<th>Windows Language ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byelorussian</td>
<td>LANG_BELARUSIAN</td>
</tr>
<tr>
<td>Bengali</td>
<td>LANG_BENGALI</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>LANG_BULGARIAN</td>
</tr>
<tr>
<td>Catalan</td>
<td>LANG_CATALAN</td>
</tr>
<tr>
<td>Chinese</td>
<td>LANG_CHINESE</td>
</tr>
<tr>
<td>Croatian</td>
<td>LANG_CROATIAN</td>
</tr>
<tr>
<td>Czech</td>
<td>LANG_CZECH</td>
</tr>
<tr>
<td>Danish</td>
<td>LANG_DANISH</td>
</tr>
<tr>
<td>Dutch</td>
<td>LANG_DUTCH</td>
</tr>
<tr>
<td>English</td>
<td>LANG_ENGLISH</td>
</tr>
<tr>
<td>Estonian</td>
<td>LANG_ESTONIAN</td>
</tr>
<tr>
<td>Faroese</td>
<td>LANG_FAEROESE</td>
</tr>
<tr>
<td>Persian</td>
<td>LANG_FARSI</td>
</tr>
<tr>
<td>Finnish</td>
<td>LANG_FINNISH</td>
</tr>
<tr>
<td>French</td>
<td>LANG_FRENCH</td>
</tr>
<tr>
<td>Georgian</td>
<td>LANG_GEORGIAN</td>
</tr>
<tr>
<td>German</td>
<td>LANG_GERMAN</td>
</tr>
<tr>
<td>Greek</td>
<td>LANG_GREEK</td>
</tr>
<tr>
<td>Gujarati</td>
<td>LANG_GUJARATI</td>
</tr>
<tr>
<td>Hebrew</td>
<td>LANG_HEBREW</td>
</tr>
<tr>
<td>Hindi</td>
<td>LANG_HINDI</td>
</tr>
<tr>
<td>Hungarian</td>
<td>LANG_HUNGARIAN</td>
</tr>
<tr>
<td>Icelandic</td>
<td>LANG_ICELANDIC</td>
</tr>
<tr>
<td>Indonesian</td>
<td>LANG_INDONESIAN</td>
</tr>
<tr>
<td>Italian</td>
<td>LANG_ITALIAN</td>
</tr>
<tr>
<td>Japanese</td>
<td>LANG_JAPANESE</td>
</tr>
<tr>
<td>Kannada</td>
<td>LANG_KANNADA</td>
</tr>
<tr>
<td>Kashmiri</td>
<td>LANG_KASHMIRI</td>
</tr>
<tr>
<td>Kazakh</td>
<td>LANG_KAZAK</td>
</tr>
<tr>
<td>Korean</td>
<td>LANG_KOREAN</td>
</tr>
<tr>
<td>Latvian</td>
<td>LANG_LATVIAN</td>
</tr>
<tr>
<td>Lithuanian</td>
<td>LANG_LITHUANIAN</td>
</tr>
<tr>
<td>Macedonian</td>
<td>LANG_MACEDONIAN</td>
</tr>
<tr>
<td>Malay</td>
<td>LANG_MALAY</td>
</tr>
<tr>
<td>Malayalam</td>
<td>LANG_MALAYALAM</td>
</tr>
<tr>
<td>Marathi</td>
<td>LANG_MARATHI</td>
</tr>
<tr>
<td>Nepali</td>
<td>LANG_NEPALI</td>
</tr>
<tr>
<td>Norwegian</td>
<td>LANG_NORWEGIAN</td>
</tr>
<tr>
<td>Oriya</td>
<td>LANG_ORIYA</td>
</tr>
<tr>
<td>Polish</td>
<td>LANG_POLISH</td>
</tr>
<tr>
<td>Portuguese</td>
<td>LANG_PORTUGUESE</td>
</tr>
<tr>
<td>Punjabi</td>
<td>LANG_PUNJABI</td>
</tr>
<tr>
<td>Romanian</td>
<td>LANG_ROMANIAN</td>
</tr>
<tr>
<td>Russian</td>
<td>LANG_RUSSIAN</td>
</tr>
<tr>
<td>Sanskrit</td>
<td>LANG_SANSKRIT</td>
</tr>
<tr>
<td>Serbian</td>
<td>LANG_SERBIAN</td>
</tr>
<tr>
<td>Sindhi</td>
<td>LANG_SINDHI</td>
</tr>
<tr>
<td>Slovak</td>
<td>LANG_SLOVAK</td>
</tr>
<tr>
<td>Slovenian</td>
<td>LANG_SLOVENIAN</td>
</tr>
</tbody>
</table>
**MicroSCADA Pro**

**OPC Server**

**Technical Description**

---

| **Table 2.6.1-1 ISO 639 language identifiers and Windows language ids** |
|-----------------|-----------------|
| **Spanish**     | **ES**          |
| **Swahili**     | **SW**          |
| **Swedish**     | **SV**          |
| **Tamil**       | **TA**          |
| **Tatar**       | **TT**          |
| **Telugu**      | **TE**          |
| **Thai**        | **TH**          |
| **Turkish**     | **TR**          |
| **Ukrainian**   | **UK**          |
| **Urdu**        | **UR**          |
| **Uzbek**       | **UZ**          |
| **Vietnamese**  | **VI**          |

---

**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.

---

**2.6.2. IOPCServer**

**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 2.4.1.

The ‘dwLCID’ parameter can be chosen from the Table 2.6.1-1. For more information on localization, see Section 2.6.1.

**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.

---

**2.6.3. IOPCItemProperties**

**Recommended properties**

The recommended properties (property ID’s 100 to 399) are not supported by the SYS 600 OPC Data Access Server.

**Vendor specific properties**

The SYS 600 OPC Data Access Server exposes all the attributes of SYS 600 objects as vendor specific properties (property ID’s > 5000), see Section 2.3.
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 - i.e. the list of properties in the containing ‘record’ object.”

Here, an intentional deviation from the standard is applied. If the ITEMID refers to an attribute of a SYS 600 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 SYS 600, it makes no sense to state, for example, that the RT (Registration Time) attribute is a property of the comment text.

IOPCItemProperties:: GetItemProperties

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

IOPCItemProperties:: LookupItemIDs

For each vendor specific property (SYS 600 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.

2.6.4. IOPCServerPublicGroups

This optional interface is not implemented, because public groups are not relevant in SYS 600.

2.6.5. IOPCBrowseServerAddressSpace

This optional interface is fully implemented. A HIERARCHICAL address space is shown.
MicroSCADA Pro
OPC Server
Technical Description

IOPCBrowseServerAddressSpace::BrowseOPCItemIDs

Here, an intentional deviation from the standard is applied. 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.

IOPCBrowseServerAddressSpace::BrowseAccessPaths

Access paths are not relevant in SYS 600. Therefore, the method BrowseAccessPaths always return S_FALSE (there is nothing to enumerate).

2.6.6. IPersistFile

This interface is irrelevant in SYS 600, it is not implemented.

2.6.7. IOPCItemMgt

Blobs

Blobs are not used by the SYS 600 OPC Data Access Server.

IOPCItemMgt::AddItems and IOPCItemMgt::ValidateItems

Note 1: 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 2.4.1), 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.

Note 2: There are items, whose data type is not known by the SYS 600 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.

Note 3: There are even items, whose very existence is not known by the SYS 600 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.

2.6.8. IOPCGroupStateMgt

IOPCGroupStateMgt::SetState

The ‘pLCID’ parameter can be chosen from the Table 2.6.1-1. For more information on localization, see Section 2.6.1
2.6.9. IOPCSyncIO and IOPCAsyncIO2

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.

Read is implemented as an atomic operation to achieve the consistency of the data, whenever possible. When consequent items within a single Read reside in the same SYS 600 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.

IOPCSyncIO::Write and IOPCAsyncIO2::Write

Write is implemented as an atomic operation, whenever possible. When consequent items within a single Write reside in the same SYS 600 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 SYS 600 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.
2.6.10. **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 SYS 600 OPC Data Access Server, but not as thoroughly tested as the recommended Version 2.0 interfaces IOPCAsyncIO2 and IConnectionPointContainer.

2.6.11. **IOPCDDataCallback**

**IOPCDDataCallback::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 SYS 600 (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, i.e. 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.

2.6.12. **IAdviseSink**

This interface is an obsolete interface defined by the OPC Data Access Custom Interface Standard, Version 1.0. It is supported by the SYS 600 OPC Data Access Server, but not as thoroughly tested as the recommended Version 2.0 interface IOPCDDataCallback.
3. Alarms and Events Server

3.1. General

The SYS 600 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 SYS 600 system.

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

- Each SYS 600 application has its own instance of OAES.
- The server is fully integrated to the SYS 600 base system software. It is started and stopped by the SYS 600 application.
- The name space (event sources, areas, event categories, conditions, etc.) of the server is configured by application engineering.
- Conditions are associated with SYS 600 process alarms, other events with SYS 600 process events.
- The server supports multilingual SYS 600 applications. See Section 3.7.7.

This chapter provides information for application programmers that build up OPC alarms and events (A&E) client applications interacting with SYS 600.

3.2. Application design

The design of an A&E application is divided into the client and the 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 SYS 600 database.

Step-by-step procedure for engineering the SYS 600 database for an A&E server:

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.
3.3. **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 SYS 600 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 SYS 600 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.

3.4. **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.

3.5. **Conditions**

OPC conditions go hand in hand with SYS 600 alarms. Each process object that generates SYS 600 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.

3.6. **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 these events and their attributes are freely configurable by the application.
3.7. Implementation notes

This section describes some details of the implementation of the SYS 600 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).

3.7.1. IOPCCCommon

The implementation of this interface is the same as in the Data Access Server. See Section 2.6.1 for details.

3.7.2. IOPCEventServer

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

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

IOPCEventServer::AckCondition

The server strictly checks the arguments pftActiveTime and pdwCookie according to the specification (unlike some other OPC A&E servers). 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.

3.7.3. IOPCEventServer2

This optional interface has not been implemented.

3.7.4. IOPCEventAreaBrowser

This optional interface has been implemented.

3.7.5. IOPCEventSubscriptionMgt

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.
IOPCEventSubscriptionMgt::SelectReturnedAttributes

The Alarms and Events Server supports one vendor specific attribute, CV (for '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.

3.7.6. IOPCEventSubscriptionMgt2

This optional interface has not been implemented.

3.7.7. IOPCEventSink

IOPCEventSink::OnEvent

**Note 1:** 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

**Note 2:** 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 SYS 600 application.
4. Application OPC Server

4.1. General

The SYS 600 Application OPC Server (AOPCS) is an implementation of the interface specification OPC Data Access Custom Interface Standard, Version 2.05A, on the SYS 600 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 SYS 600 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) SYS 600 systems. Each AOPCS instance is associated with a SYS 600 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 SYS 600 system. This is especially useful if the DCOM connection between the OPC client and the SYS 600 Data Access Server is unreliable, since AOPCS can provide seamless handling of connection breaks, including automatic recovery of lost events.

Some other reasons for using AOPCS in a non-redundant system might be:

• The client application does not support DCOM servers.
• For a reason or another, 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 Chapter 2, apply to AOPCS as well. A slight difference in the semantics of application relative name spaces is described in Section 4.4. The added functionality is described in Section 4.3.

The home application and its location(s) in the SYS 600 system are specified in the configuration file of the AOPCS instance, see Section 4.2.

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

4.2. Configuration

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 SYS 600 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 SYS 600 application DEMO, might be the following:

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

The node may be defined by name or by IP address. The SYS 600 application may be defined by name or by application number.

If the names or application numbers of the SYS 600 applications in the two SYS 600 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 SYS 600 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 4.3.

**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.

**DCOM security settings**

After the AOPCS instance has been registered, the DCOM connection between the AOPCS and the SYS 600 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 SYS 600 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.

   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 SYS 600 server.

**4.3. Functionality**

**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.

**States**

AOPCS constantly monitors the SYS 600 applications specified in the INI file. The availability of a hot SYS 600 application determines the state of AOPCS. Clients can query the state by the method IOPCServer::GetStatus.

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

When a hot SYS 600 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, or the SYS 600 base systems are down, or their Data Access Servers are not running, the state is SUSPENDED. Most OPC interface calls will simply return E_FAIL.

**HSB switch-over**

When the connection to the hot application is lost for a reason or another and the other SYS 600 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 SYS 600 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, i.e. 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.

**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 4.2.

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, e.g. 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 SYS 600 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.

4.4. 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 2.2) 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 SYS 600 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 SYS 600 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.
6. Index

Symbols
#MODIFY .................................................................34
#SET .............................................................................34

A
Absolute name space .........................................................7, 45
Access paths ..................................................................33
Alarms and Events Server .............................................37, 41
AOPCS ..........................................................................41
Application relative name space ................................7, 45

B
BIT_STRING ................................................................... 19
Blobs ..............................................................................33
BOOLEAN ...........................................................................19
BYTE_STRING ...................................................................19

C
Conditions ........................................................................38

D
Data Access Server .........................................................7
Data types ....................................................................18

E
Error codes .......................................................................23
Event properties ..........................................................16
Event sources ..............................................................38

F
FAILURE_HIDING_TIME .............................................42
FAILURE_RECOVERY_TIME ........................................42

H
HSB ...............................................................................41

I
IAdviseSink ...................................................................35
IDataObject ...................................................................35
INTEGER .......................................................................19
IOPCAsyncIO ..................................................................35
IOPCAsyncIO2 ..............................................................34
Read 23, 34
Write 23, 34
IOPCBrowseServerAddressSpace ................................32
MicroSCADA Pro
OPC Server
Technical Description

BrowseAccessPaths 33
BrowseOPCItemIDs 33
IOPCCommon ......................................................... 29, 39

GetErrorString 24, 31
IOPCDataCallback .................................................. 35

OnDataChange 23, 35
OnReadComplete 23
OnWriteComplete 23
IOPCEventAreaBrowser ............................................. 39
IOPCEventServer .................................................... 39

AckCondition 39
CreateAreaBrowser 39
DisableConditionByArea 39
DisableConditionBySource 39
EnableConditionByArea 39
EnableConditionBySource 39
GetConditionState 39
QuerySourceConditions 39
TranslateToItemIDs 39
IOPCEventServer2 ................................................... 39
IOPCEventSink ....................................................... 40

OnEvent 40
IOPCEventSubscriptionMgt

SelectReturnedAttributes 40
SetFilter 39
IOPCEventSubscriptionMgt2 ....................................... 40
IOPCGroupStateMgt ................................................. 33

setState 33
IOPCItemMgt .......................................................... 33

Addltems 24, 33
ValidateItems 24, 33
IOPCItemProperties ................................................. 31

GetItemProperties 16, 23, 32
LookupItemIDs 16, 32
QueryAvailableProperties 16, 32
IOPCServer ............................................................ 31

AddGroup 31
GetErrorString 24, 31
MicroSCADA Pro
OPC Server
Technical Description

IOPCServerPublicGroups ............................................................ 32
IOPCSyncIO .................................................................................. 34

  Read 23, 34
  Write 23, 34

IPersistFile .................................................................................... 33
Item properties ................................................................................ 14

L
LIST .............................................................................................. 20
LocaleID ......................................................................................... 29

N
NAMESPACE .................................................................................... 24
NAMESPACE items .......................................................................... 24

O
OAES .............................................................................................. 37
OPC NAMESPACE ........................................................................... 24

P
POLL ............................................................................................... 17
Process areas .................................................................................. 38
Properties ......................................................................................... 14
Property ID ....................................................................................... 15
Public groups ................................................................................. 32

Q
Quality ......................................................................................... 22, 23
QueryAvailableLocaleIDs .............................................................. 29

R
REAL ............................................................................................... 19
Recommended properties .............................................................. 31
REPORT .......................................................................................... 17

S
SCIL ............................................................................................... 27
SCIL error codes ........................................................................... 24
SCIL language items ....................................................................... 27
SCIL PROGRAM .............................................................................. 28
SetLocaleID ..................................................................................... 29
Simple event .................................................................................... 38

T
TEXT ............................................................................................... 19
TIME ............................................................................................... 19
Time stamps .................................................................................... 21
Tracking event ................................................................................ 38
U
Update policy .......................................................... 17
User defined name space .............................................. 8, 45

V
VECTOR .................................................................................. 19
Vendor specific event attributes ........................................ 40
Vendor specific properties ................................................. 14, 31