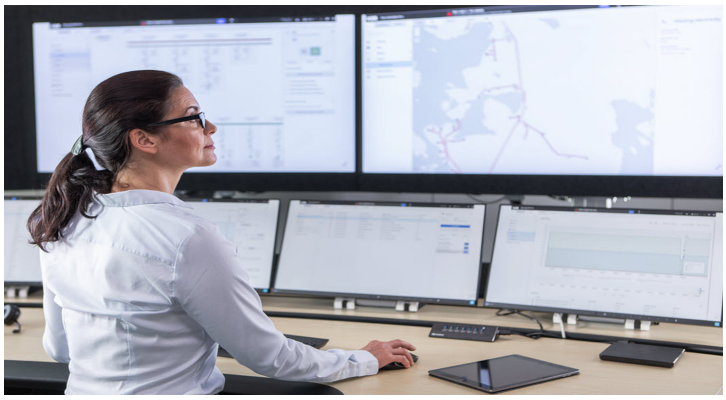
GRID AUTOMATION PRODUCTS

MicroSCADA X SYS600 10.2

System Configuration



Document ID: 1MRK 511 481-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 Introduction

2.1 This manual

This manual provides thorough information on the SYS600 software and hardware installation: base systems, LAN connections, process communication systems, workplaces and peripherals.

This manual provides thorough information on the various configuration settings that you have to make in order to use your SYS600 system. The manual also describes how to use the configuration tools.

The rows in some of the SCIL examples in this manual have been split to two rows. Take this into consideration when copying the SCIL examples.

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 engineers to support configuration and engineering of systems and/or applications.

2.4 Product documentation

Name of the document	Document ID
SYS600 10.2 Application Design	1MRK 511 466-UEN
SYS600 10.2 Application Objects	1MRK 511 467-UEN
SYS600 10.2 COM500*i* Users Guide	1MRK 511 468-UEN
SYS600 10.2 Communication Programming Interface (CPI)	1MRK 511 469-UEN
SYS600 10.2 Connecting LONWORKS Devices	1MRK 511 470-UEN
SYS600 10.2 Cyber Security Deployment Guideline	1MRK 511 485-UEN
SYS600 10.2 DNP V3.00 Master Protocol	1MRK 511 486-UEN
SYS600 10.2 DNP V3.00 Slave Protocol	1MRK 511 487-UEN
SYS600 10.2 External OPC Data Access Client	1MRK 511 471-UEN
SYS600 10.2 IEC 60870-5-101 Master Protocol	1MRK 511 489-UEN
SYS600 10.2 IEC 60870-5-101 Slave Protocol	1MRK 511 490-UEN
SYS600 10.2 IEC 60870-5-103 Master Protocol	1MRK 511 491-UEN
SYS600 10.2 IEC 60870-5-104 Master Protocol	1MRK 511 492-UEN
SYS600 10.2 IEC 60870-5-104 Slave Protocol	1MRK 511 493-UEN
SYS600 10.2 IEC 61107 Master Protocol	1MRK 511 494-UEN
SYS600 10.2 IEC 61850 Master Protocol (OPC)	1MRK 511 495-UEN
SYS600 10.2 IEC 61850 System Design	1MRK 511 475-UEN
SYS600 10.2 Installation and Administration	1MRK 511 496-UEN
SYS600 10.2 Modbus Master Protocol	1MRK 511 497-UEN
SYS600 10.2 Modbus Slave Protocol	1MRK 511 498-UEN
SYS600 10.2 OPC Server	1MRK 511 476-UEN
SYS600 10.2 Operation	1MRK 511 499-UEN
SYS600 10.2 Operation manual for Workplace X	1MRK 511 500-UEN
SYS600 10.2 Pipeline Application Design	1MRK 511 477-UEN
SYS600 10.2 Pipeline Operation	1MRK 511 501-UEN
SYS600 10.2 Process Display Design	1MRK 511 478-UEN
SYS600 10.2 Programming Language SCIL	1MRK 511 479-UEN
SYS600 10.2 Status Codes	1MRK 511 480-UEN
SYS600 10.2 System Configuration	1MRK 511 481-UEN
SYS600 10.2 System Objects	1MRK 511 482-UEN
SYS600 10.2 Visual SCIL Application Design	1MRK 511 483-UEN
SYS600 10.2 Visual SCIL Objects	1MRK 511 484-UEN
SYS600 10.2 Workplace X Data Model	1MRK 511 504-UEN
SYS600 10.2 Workplace X Process Picture Design	1MRK 511 505-UEN
SYS600 10.2 Workplace X View writer's guide	1MRK 511 506-UEN

2.5 Document conventions

The following conventions are used for the presentation of material:

- The words in names of screen elements (for example, the title in the title bar of a dialog, the label for a field of a dialog box) are initially capitalized.
- Capital letters are used for file names.
- Capital letters are used for the name of a keyboard key if it is labeled on the keyboard. For example, press the CTRL key. Although the Enter and Shift keys are not labeled they are written in capital letters, e.g. press ENTER.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the space bar, comma key and so on.
- Press CTRL+C indicates that you must hold down the CTRL key while pressing the C key (to copy a selected object in this case).
- Press ALT E C indicates that you press and release each key in sequence (to copy a selected object in this case).
- 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 you enter a value 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 Functional overview of SYS600

MicroSCADA X Control System SYS600 is a modular and scalable automation product. It is structured into a generic application independent platform and application functionality. SYS600 is designed mainly for Substation Automation and Network Control applications. It scales from compact single computer communication gateway or monitoring applications in substations to hierarchical and redundant network control systems, managing tens to several hundreds of thousands of data points. One of the strengths of the system is the scalability regarding capacity and performance but also regarding functionality. This enables a suitable solution for every need.

3.1 System architecture

The main components of a SYS600 system are:

- System servers
- Communication servers
- Workstations
- Peripheral equipment including printers, GPS clocks, alarm devices
- Communication equipment including switches, routers, modems
- IEDs, process devices, data acquisition units, RTU's, PLC's and so on

The different system components can be used to build everything, including the following:

- A small single computer monitoring system, for example, embedded in a panel PC with touch screen mounted in the door of a cubicle (as shown in Figure 1).
- A hierarchical system in several levels with redundant servers (as shown in Figure 2).

Figure 1: Single computer system in Panel-PC with touch screen

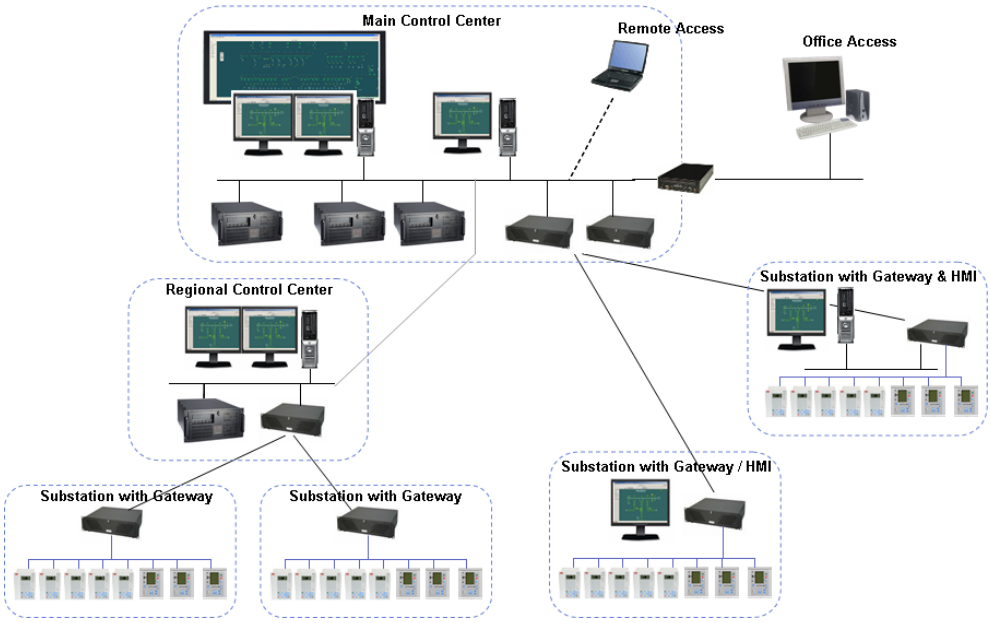


Figure 2: Interconnected systems in several levels

3.2 System server

Most of the different functions in SYS600 are handled by the system server. In each system server, there is one base system that is responsible for the central data processing services. Each base system can include one or several applications, as shown in Figure 3. The application defines the automation functionality and user interface by its own realtime process database, history database, displays and so on. The applications are independent from each other, although they can interact with each other. A typical single computer SA system has one system server with one base system and one application, whereas a large distributed control system can have several servers with several applications each. The applications can be divided according to:

- application area (for example, electricity or district heating).
- tasks (for example, reporting, process display handling, communication gateway).

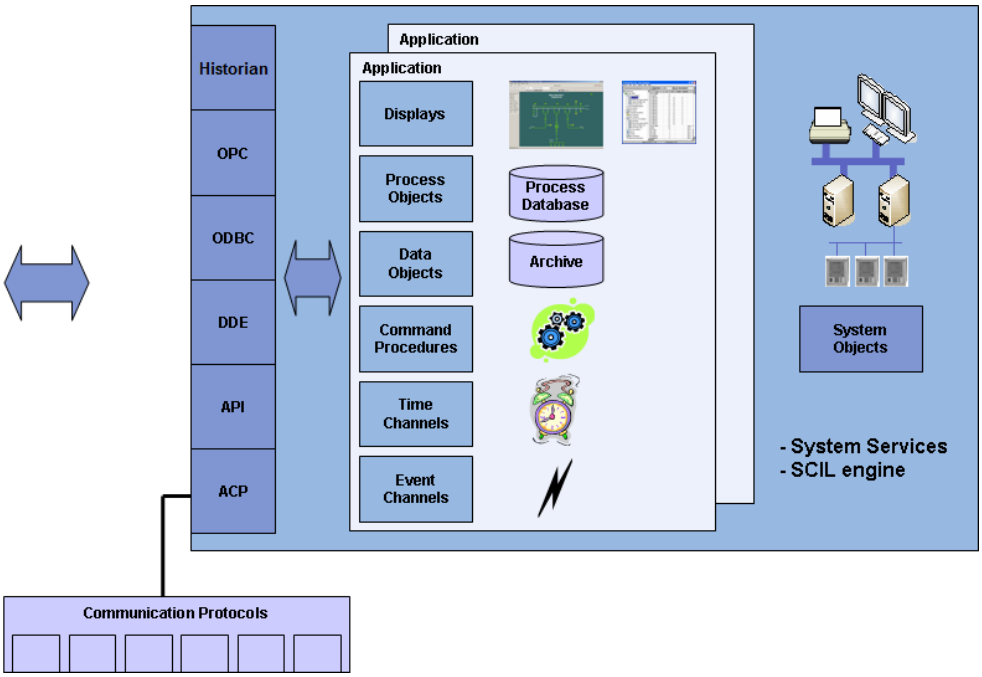


Figure 3: System Server Architecture

The system server can also include the communication services needed for remote communication (communication with an upper level system) and process communication (communication with the process or lower level systems). However, these services can also be located in separate communication servers.

The system server also supports redundancy by a hot stand-by concept for the applications, and with the help of it the availability of the system can be improved.

3.2.1 Application examples

Application examples are shown in the following figures:

Figure 4: System with one system server including all functionality

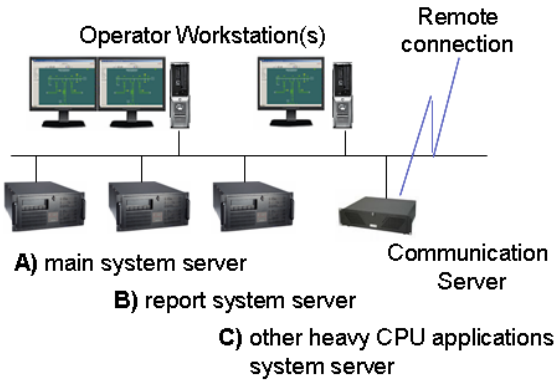


Figure 5: System with several system servers for various tasks

Figure 6: System with hot stand-by servers

3.2.2 System objects

The components of the system and the system server are configured and managed by means of System objects. There is a system object for each connected station (IEDs, RTUs and so on), printer, system node (base system, PC-NET, OPC servers and so on) and application. The most important system object types are:

- System
- Application
- Link
- Node
- Station
- Printer
- Monitor

The system objects have a number of attributes that are used for configuring and monitoring of the system. The system objects are created and managed by SCIL.

For more information, see SYS600 System Objects.

3.2.3 Application objects

Each application contains a number of various application objects. Application objects define the behavior of the application. The application object types are listed below:

- Process objects (and free type process objects)
- Event handling objects
- Scales
- Time channels
- Event channels

- Command procedures
- Data objects
- Logging profile objects

For more information, see SYS600 Application Objects.

3.2.4 Programming language SCIL

The script programming language SCIL plays a central role in the SYS600 system. All objects, both system objects and application objects, are created and managed by SCIL. In addition to that, all supervision and control tasks are executed by SCIL programs. Most of the configuration and engineering tasks are managed by tools whereby the SCIL program is hidden for the user. Also, the SCIL, in the supervision and control tasks, are by default hidden from the user. However, in case the functionality needs to be customized and extended, it can be accomplished by SCIL. SCIL can also be used to develop new user interface applications and dialogs.

For more information, see SYS600 Programming Language SCIL, SYS600 Visual SCIL Application Design and SYS600 Visual SCIL Objects.

3.2.5 Process database

The process database is the storage for all process data related application objects. These are process objects, scales, event handling objects and free type process objects. The process database is a high performance and high capacity real time database that is maintenance free and can be configured online. The objects are created, deleted and managed with SCIL and with dedicated tools.

For more information, see SYS600 Application Objects.

3.2.6 Report database

Report database is the storage for all reporting, calculation and data processing related application objects. These objects are: Time channels, Event channels, Command procedures, Data objects and Logging Profile objects.

For more information, see SYS600 Application Objects.

3.3 Process communication

The task of the process communication is to form a communication link between the SYS600 system server and various process devices, like IEDs, RTUs, PLC and so on. The communication with process devices uses various types of protocols, such as IEC 60870-5-10x, IEC 61850, DNP, Modbus, LON, SPA and so on. Each protocol has its own characteristics and the physical media and interfaces have to be built accordingly. The software interface in SYS600 is handled by a communication unit. The communication unit is dependent on the protocol that is used.

The most common communication units are the PC-NET and the IEC 61850 OPC Server/ External OPC DA Client. The PC-NET is used with most of the SYS600 protocols, except IEC 61850, which is handled by the IEC 61850 OPC Server and the External OPC DA Client. The process communication can be integrated in the system server in compact systems and it can also be allocated to dedicated communication servers.

For process communication, always use shielded communication cables to minimize the effect of electrical disturbances to the communication lines connected to SYS600. The maximum shield is provided when the shield of the communication cable has been wired to the DB9 connectors in both ends. The additional equipment, for example, gender changers or DB9 adaptors, should not be used together with serial cables.

3.3.1 Communication servers

The communication services can be allocated to dedicated communication servers. This can be done under the following circumstances:

- When higher capacity and performance is needed than what one server with everything integrated can handle.
- When more hardware interfaces are needed, for example, serial ports or LON interfaces.
- To minimize impact of hardware failure.
- When redundant communication servers are required.

The communication server typically communicates with the systems server over LAN.

The communication server always includes one or several communication units (PC-NET, IEC 61850 OPC Server), but it can also be configured to include its own process database. When it includes its own process database, it communicates with the system server(s) by means of the process data mirroring function. The reasons for including the process database could be one of the following:

- The communication server provides data to several system servers.
- Higher buffering capacity of the data between the communication server and the system server is needed.
- Local data processing in the communication server is required.

The communication servers can be implemented both as a single computer and as a hot stand-by pair.

The components of the communication server vary depending on the used protocols and on the overall system architecture. The most important components are:

- PC-NET
  It includes protocol drivers for all supported protocols, except IEC 61850 (and those implemented with CPI).
- SYS600 base system
- IEC 61850 OPC Server
  It is an IEC 61850 client that is connected to SYS600 base system over OPC.
- External OPC DA Client
  The OPC DA Client that is used to connect the IEC 61850 OPC Server to the base system.

3.4 Communication gateway

The communication gateway is a system server with an application, that routes messages from process communication to remote communication and vice versa. The gateway application is called COM500i. The communication gateway can have the communication services integrated in it, or it can also use external communication servers for process communication. It can be configured in hot stand-by mode. The application can be freely combined with any other system server functionality including reporting, process displays and IED tools.

3.5 Operator and Engineering Workplaces

Operator workplace provides the means for the operator to supervise and interact with the process with the help of the graphical user interface (GUI). Three different generations of operator workplace types are supported: Classic, Pro and Workplace X. Workplace X is the latest generation of the operator workplace and the other workplaces are supported to provide backward compatibility from older product versions to newer.

Engineering workplace is used for engineering and configuring the system. Two different types of engineering workplaces are supported: the classic workplace and the pro workplace.

Workplaces are always connected to a system server. Each system server can have its local workplaces but the workplaces can also be distributed to other computers and locations. The computer, in which the workplace is used, is called a workstation. Each system server can have up to 50 Workplace X workplaces or 50 pro workplaces and/or 100 classic workplaces simultaneously in use.

Workplace X operator workplace uses web technology with dedicated Workplace X Windows application or with regular web browser that allows both local and remote access over network. No Microsoft's Remote Desktop Services (RDS) are used for these and therefore no related client access licenses are required.

Pro and Classic workplaces can be used for both operation and engineering. The possibilities are defined by the access rights given to the user in question. The remote workplace concept for these is based on Microsoft's Remote Desktop Services (RDS).

3.6 Peripheral equipment

Various peripheral equipment can be connected to the system. The most typical ones are listed below:

- Printers
- Alarm annunciation units
- GPS clocks

3.6.1 Printers

The printers can be connected to the system server either directly, over a LAN via printer servers, or via the process communication system.

A base system can have up to 20 printers of different types: it can be a matrix printer, or a laser printer. Printers can be allocated for different tasks, including alarm and event printout, hard copy, and historical reports. It can be programmed to take over the tasks of another printer automatically.

Printouts can be produced automatically or manually. The layout of a printout can be customized. The main printout types are logs, reports, hard copies and documents. Logs are automatic printouts based on process events. The logs can be directed to one or more printers.

3.6.2 Alarm annunciation units

Alarm annunciation units can be connected to the system to produce visual or audible alarms and a provision to acknowledge alarms. Some units can also supervise the system server and produce alarms in the case of failure.

3.6.3 GPS clocks

For accurate time synchronization, one or several GPS clocks can be connected to the system. The clock can be connected directly over LAN, using standard Operating System functions, or it can be connected via the process communication bus, for example, over IEC 61850.

3.7 System Self Supervision

The System Self Supervision function shows the status of the various system components in a display for easy and fast system maintenance and fault localization. The display shows information about the base system, applications, redundancy, communication lines, IEDs and so on. The system can also receive status information from any device or external software reporting to the Windows event log, for example, disks, power supplies and computer boards. Communication equipment and peripheral devices that support SNMP can also be supervised by using a SNMP-OPC gateway.

3.8 Event handling

An event is a kind of change in the process or in the system, that occurs at a certain moment in time. Event can be caused by changes in the process, actions performed by the user, faults in the system and so on.

3.8.1 Data collection

All data related to the controlled process, including status indications, measurements and commands, are managed by the process database. In addition to these, data related to the system itself, connected IEDs, RTUs, peripheral equipment, user activities and so on can be collected into the process database. Each signal is represented by a process object in the database.

In addition to the object value, the process object holds a number of attributes that gives more information about the value, as well as describes how events are generated based on changes in the process object.

3.8.2 Time tagging

Each process object has a time tag that tells when the object was updated. The time tag typically originates from the source of the data, for example, IED, RTU, and so on. Only if the device, that collects the data, does not support time tags, or if the used communication protocol does not support time tags, then the system itself gives the time tag. The time tag resolution is 1 ms, but the accuracy depends on the time accuracy of the data source.

3.8.3 Event criteria and actions

For each process object, it is possible to specify what kind of actions take place and when. Examples of these criteria and actions are listed below:

- Criteria
  - New value
  - Updated value
  - Value goes up/down

    - Warning state reached
    - Alarm state reached
    - Alarm acknowledged
- Actions
    - Event logging
    - Activation of freely configurable post-processing. The freely configurable post-processing can basically initiate any type of activity in the system.
    - Printouts

3.8.4 Event descriptions

The system also contains descriptive information about each event. This information is shown in the event list. The descriptive information can be formed from two different texts: the state text and the message text.

The state text described the current object state and is dependent on the object value. The state text can, however, also take into account any other attribute of the process object in order to exactly describe the situation.

The message text is formed based on the object value transition, both the old value and the new value.

The event descriptions are defined by event handling objects in the process database. Each process object is associated with an event handling object.

3.8.5 Event logging

All events that are defined to be logged are stored in an event database. The capacity of the event database is only limited by the disk storage capacity. Most process object attributes are stored with each event to enable exact post-analysis of the process state.

3.8.6 Event post-processing

It is possible to initiate some post-processing at an event. The post-processing is implemented by a SCIL program, that can perform any type of tasks, including reading a disturbance record file from an IED, starting a command sequence, sending data to another system and starting an external program.

3.8.7 Event display

The event display shows events stored in the event database. It supports easy sorting and filtering of the events to make the event analysis as easy as possible. It can also be freely configured to display the information, the layout and colouring of the events.

It is also possible to have predefined filters that are easily accessed from toolbars and menus.

3.9 Alarm handling

3.9.1 Alarm

An alarm is a state of a signal, (process object) that is so critical that it is defined to cause special actions in the system in order to notify the operator. Some alarms require an acknowledgement. An alarm can have the following states:

- Active (or persisting)
  - The alarm state is active but the alarm does not need to be acknowledged
- Active unacknowledged
  - The alarm state is active but not yet acknowledged
- Active acknowledged
  - The alarm state is active and acknowledged
- Fleeting
  - The alarm state is no longer active but has not been acknowledged after it became active

The state transitions of an alarm can cause an event in the system, as well as the acknowledgements. In this way, it is possible to analyze the alarm state history with the help of the event display. With proper filtering, the event list can show all alarm activations, deactivations and acknowledgements.

3.9.2 Alarm indication

Alarming signals are represented in various ways in the system. The alarm state information is always available in the process database and can be used for any type of alarm representation. The most typical signals are listed below:

- A specific representation in the process display, for example, a blinking symbol
- The signal representation is colored red, for example, in process displays, lists, dialogs and so on
- An audio signal is played, for example, a horn or sound file of the computer
- The alarm is displayed in the alarm display
- The alarm is displayed in the object dialog of the concerned object

3.9.3 Alarm display

The alarm display shows the alarms of the system in a list. It supports easy sorting and filtering of the events to make the alarm analysis as easy as possible. It can also be freely configured to display the information, the layout and coloring of the alarms.

It is also possible to have predefined filters that can be easily accessed from toolbars and menus.

3.10 Reporting

SYS600 offers many possibilities for data logging and visualisation in form of graphs and reports. There is a built in flexible data logging mechanism capable of cyclic and event based logging of data of various data types. The logging mechanism is built using Application Objects (Data Objects, Time Channels, Command Procedures). There are also Trends and Measurement Reports packages, optimized for power distribution systems, that include predefined Application Objects for the data logging as well as standard displays for reports and Trends. These packages are easy to configure and use but the full flexibility of the Application Objects cannot be fully utilised. The above mentioned functions are always running integrated on the SYS600 system servers. Additionally, SYS600 has a separate Historian function that offers advanced data logging and visualisation features. The Historian is typically used when more advanced visualisation is needed, or when the data logging capacity grows and standalone Historian servers are needed. The integrated features are described in this section, and the Historian is described in Section 3.11.

3.10.1 Data logging

Data that needs to be stored in the system, is stored in data objects. Each data object can hold between 1 and 2 000 000 data entries of the data types: real, integer, text and time. One data object can hold data of one data type only. The total number of data objects and Command Procedures is 2 000 000. So the maximum number of data entries is $4*10^{12}$ (can further be restricted by the license).

If more data needs to be logged and stored SYS600 historian database has to be used.

The data logging can be triggered based on the time (at certain time intervals), based on events, or at any time from a SCIL Program. The raw value for the data logging, which is defined in the data object as a SCIL expression, is typically taken from a process object (a current or voltage measurement). However, it can also be derived from several process objects or other system data, including mathematical formulas. Before the actual value is logged, an additional formula that operates on the new value and the already logged values, can be applied in order to calculate sums, mean values, integral values, differences, derivative values, maximum and minimum values. Additional refinement of the logged data can be achieved by SCIL programming.

Each data object entry has a time tag and status information. The status information is derived from the source for the data. For example, if a process object that is used as a source for the logging has uncertain status, also the data object entry gets the uncertain status. The time tag is the time when the logging occurred.

3.10.2 Data processing

All data objects are fully accessible with SCIL so further data processing can be achieved by SCIL programming. In this way, further data analysis and refinement can be achieved to calculate forecasts, trends, various statistics and so on. The result of the calculations can also be stored in data objects for visualization or other needs.

3.10.3 Trends

The trend display is an application that collects data and visualizes the data in numerical or in graphical form. It is used for follow up and analysis of data in time frames from minutes to months. The data is logged cyclically in intervals of 10 seconds, 30 seconds, 1 minute, 2 minutes, 5 minutes or 10 minutes. The trend display can log and show any type of data available in the process database.

3.10.4 Measurement reports

The measurement reports display is also an application that collects data and visualizes the data in numerical or graphical form. The measurement reports display is used to log and report data during longer periods than the Trend Application, and it is dedicated for Energy, Current, Voltage, Temperature and Frequency reports. The available time ranges for the reports are:

- Hourly report (time resolution: 1, 2, 3, 5, 6, 10, 15, 20 or 30 minutes)
- Daily report (time resolution: 10 minutes)
- Daily report (time resolution: 15 minutes)
- Daily report (time resolution: 30 minutes)
- Daily report (time resolution: 60 minutes)
- Weekly report (time resolution: 1 day)
- Monthly report (time resolution: 1 day)
- Yearly report (time resolution: 1 month)

The storage period for the reports can be up to 6 years. Longer storage periods can be achieved by exporting data to an external reporting database.

3.11 Historian

Historian is a subsystem dedicated for fast and efficient data logging, data refinement and information visualisation. The flexible Historian can store all process data for long periods and refine the data into meaningful information. This gives a clear view of the situation in the primary process, allows for optimized utilization of energy and equipment as well as produces reports and statistics. The information is visualized in the form of various graphs, trends and numerical reports. The numerical reports utilize reporting plugin that provides tools for further data refinement.

3.11.1 System configurations

Historian should primarily be used on a dedicated server. The data sources for Historian are Process Objects or Data Objects of SYS600 Applications. The data is transfer from the SYS600 Application to the Historian over a TCP/IP LAN. One SYS600 application can log data into one or several Historian instances and one Historian instance can receive data from one or several SYS600 Applications.

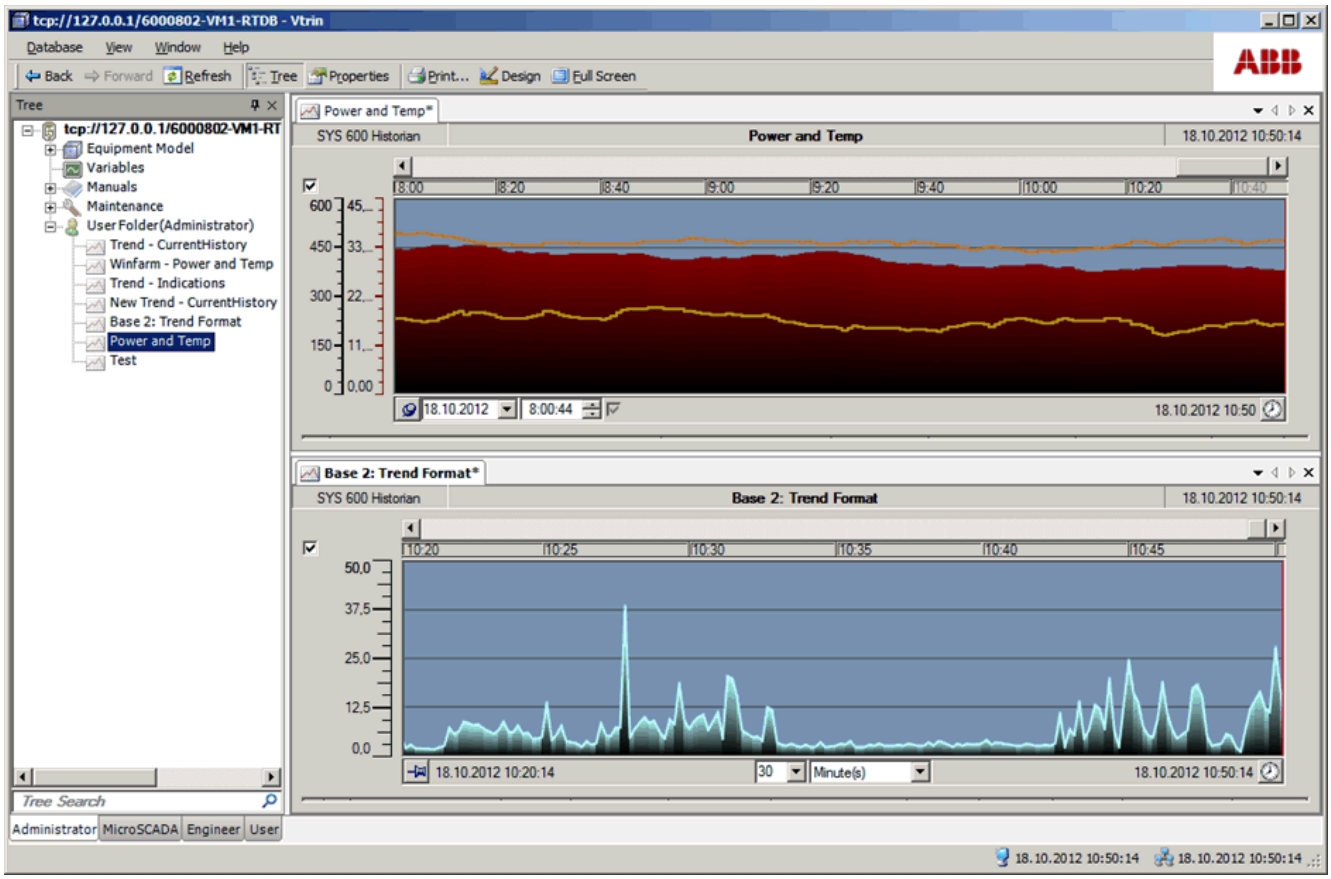
3.11.2 Data logging

Historian logs data from the SYS600 Application and more precisely from Process Objects or Data Objects. The data logging is triggered by the update of the Process Object or Data Object. This means that the data is logged event based, but by updating Process Objects or Data Objects from a Time Channel or from SCIL, also cyclic or manual data logging can be triggered.

Historian logs the raw data (all updates) from the sources during 8 days. The raw data is then refined into long term storage like 1 minute average, 2 minute maximum, 1 hour minimum, etc.

3.11.3 Visualisation

Visualisation of the logged and refined data is done by means of various graphs and tables. The graphs can be freely configured in terms of graphical representation, axes types and orientation, update rates, etc. The numerical tables are utilizing reporting plugin providing flexible configuration and data manipulation possibilities.



3.12 User Activity Logging

SYS600 logs user activity related events. Because users are per application in SYS600, each application does its logging independently of other applications. Events are stored locally by default. Additionally up to 6 external Syslog server definitions can be used.

User activity events are generated also by communication modules such as PC-NET for the application defined in communication configuration. Node, line and station attributes UI (UAL Event identification) and UA (User Activity) controls the sending conditions and the source naming of the generated event. The default naming of event source is based on node, line and station number. See protocol specific manuals for more information.

3.12.1 Local Storage

User Activity Log events of an application are stored in the file APL_UALyyyy.UAL, where yyyy stands for the year. There is one event file per calendar year. In a Hot Stand-by system, the file is subject to shadowing. Administrator users can view local events using User Activity Log list included in SYS600 Monitor Pro.

3.12.2 External Syslog Server Configuration

User Activity Logging functionality is configured by application object attribute UA. UA attribute must be defined when external Syslog servers are used as User Activity Log receivers. For detailed information about application object attribute UA, see SYS600_System Objects manual.

Example of application object attribute UA configuration for two external Syslog servers:

```
#SET APL:BUA = LIST(SOURCE="SYS600",SERVERS=VECTOR(
LIST(TYPE="UDP",IP_ADDRESS="212.226.162.153",PORT=514),
LIST(TYPE="TCP",IP_ADDRESS="212.226.162.154",PORT=1468)))
```

3.12.3 User Activity Log Event Messages

Event message is a data structure that describes an event. Each event type (defined by numeric event id) has a short description. The event descriptions are localizable and the English descriptions are available in the delivered SYS_TEXT.SDB file. The text id's are prefixed by "SYS_UAL_". The following table describes message details and fields received by external Syslog servers.

Table 1: User Activity Log Event message structure

Field	Description
EventId	Predefined identifier of the event type. See table Table 2
Time	Event generation time (UTC), 1 ms resolution
SOE	Sequence number of event
User	The name of the user that generated the event
Severity	Severity of the event as defined by Syslog standard
Source	The source of event
ProductName	The product that generated the event, always "SYS600"
SourceIP	The IPv4 address of the sender of the message. In case where there are multiple network adapters and ip addresses, this field is always set to contain the first/primary ip address of the computer.
ExtraInfo	Event type specific extra information. In cases where the description of the event is not accurate, the extra information field may provide additional information about the event conditions. See table Table 2

If the SYS600 user name is longer than 32 bytes when encoded to UTF-8, it will be truncated in the event message. In the local storage it is stored un-truncated. The contents of the Source field can be configured in the UA attribute of the application object. If not configured, the name of the SYS600 application is used.

3.12.3.1 Generated User Activity Event Messages

The following table lists all the user activity events that are generated by SYS600 base system. In the ExtraInfo column, "Host" means either the computer name or the IP address of the workplace.

Table 2: Generated User Activity Event Messages

Id	Description	ExtraInfo
1110	Log-in successful	Role, Host
1115	Password expired, Log-in successful	Host
1130	Log-in failed - Wrong credentials	Host
1150	Log-in failed - Password expired	Host
1170	Log-in failed 3 times	Host
1210	Log-out (user logged out)	Host
1220	Log-out by user inactivity (timeout)	Reason
Table continues on next page		

Id	Description	ExtraInfo
1730	Admin password reset to factory default	EMERGENCY Login
2110	User account created successfully	User
2120	User account deleted successfully	User
2115	User account enabled successfully	User
2117	User account disabled successfully	User
2160	New role assigned to user successfully	User/Role
2170	User role assignment removed successfully	User/Role
2180	New role created successfully	Role
2190	Role deleted successfully	Role
2210	User password changed successfully	User
2220	Change of user password failed	Reason
2233	User Password change failed - too short	
2235	User Password change failed - policy check failed	
2410	Areas of Responsibility enabled successfully	
2411	Areas of Responsibility disabled successfully	
2430	Exclusive Access Rights enabled successfully	
2431	Exclusive Access Rights disabled successfully	
2435	Exclusive Access Rights reassigned successfully	User
2436	Exclusive Access Rights unassigned successfully	
2437	Exclusive Access Rights auto-assigned successfully	User
2410	Areas of Responsibility enabled successfully	
2411	Areas of Responsibility disabled successfully	
2413	Area of Responsibility created successfully	Area name
2414	Area of Responsibility removed successfully	Area name
2416	User assigned to Area of Responsibility successfully	User, area name
2417	User's Area of Responsibility assignment removed successfully	User, area name
2423	User's Area of Responsibility role changed successfully	User, area name
2425	Area of Responsibility controlled - operator present	Area name
2426	Area of Responsibility left uncontrolled - no operator	Area name
2430	Exclusive Access Rights enabled successfully	
2431	Exclusive Access Rights disabled successfully	
2435	Exclusive Access Rights reassigned successfully	Area name, old owner user -> new owner user
2436	Exclusive Access Rights unassigned successfully	Area name, old owner user
2437	Exclusive Access Rights auto-assigned successfully	Area name, new owner user
2442	User's Exclusive Access Rights role changed successfully	User, area name

3.13 Localization

It is possible to translate the user interface to any language that can fit into the boundaries of the graphical user interface, that is, size, position and direction of texts and icons. The system also supports several languages simultaneously enabling operators to work with the system in their native language. The language is part of the user profile and is selected automatically when the user logs into the system.

3.14 Time synchronization

3.14.1 GPS

GPS time can be used to synchronize the computer in various ways. The most common ways are listed below:

- LAN
- Serial Port
- Dedicated PC Card

When the GPS clock is connected to the LAN, it communicates with the SYS600 computer using SNTP (Simple Network Time Protocol). The SNTP is needed, as there is a SNTP client in the computer that synchronizes the internal clock. If IEC 61850 is used in the system, the IEC 61850 client can also act as a SNTP client. If IEC 61850 is not used, an external SNTP client has to be used. Alternatives are Tardis2000, Yats32 Synchronization applications and Trimble Ace III.

It is preferable to connect the GPS over LAN in IEC 61850 systems, because then the same clock can be used to synchronize the IEDs.

The GPS clock can also be connected to a serial port of the computer. In this case, the clock manufacturer provides driver that handles the synchronization of the computer clock. One example is Meinberg GPS 167, as shown in Figure 7.

Figure 7: Meinberg GPS 167

The GPS signal can also be handled by a dedicated PC Card. Also in this, case the manufacturer provides a driver that handles the synchronization of the computer clock. The GPS170PCI card, as shown in Figure 8, synchronizes the system time of computers with PCI/PCI-X bus interface.

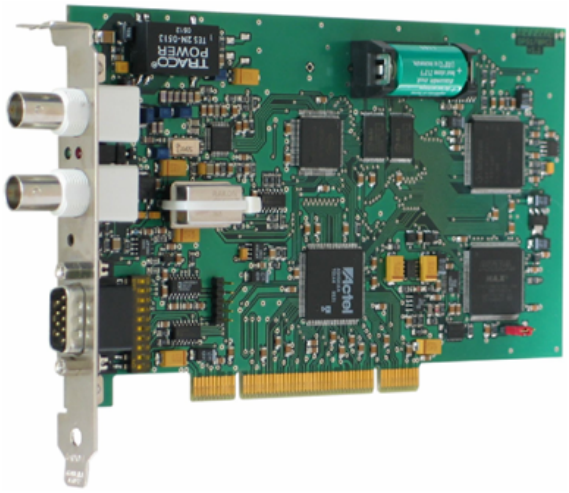


Figure 8: Meinberg's GPS170PCI clock

The accuracy of the time depends on the components participating in the synchronization, including the receiver, the SNTP client and the internal clock of the computer. Typically an accuracy of +- 1 ms can be reached.

3.15 Redundancy

3.15.1 Server redundancy

The availability of a SYS600 system can be further improved by redundant servers configured in hot stand-by mode (HSB). The HSB concept defines redundancy between applications, which means that there is always a pair of applications in a hot stand-by relation. This relation means that one application is active and receiving and processing all the data from the process. It is also managing the displays and providing data for the displays. At the same time, all process data, configuration data and so on, are shadowed over to the stand-by application. The stand-by application will, at all times, be in exactly the same state as the main application. If the computer, in which the main application runs, breaks down, the stand-by application will take over the process communication and continue to manage the process. In this way, the process can be operated and supervised even if one server computer fails.

An HSB application pair can be seen as one logical application by other applications in the SYS600 network. A proxy application is defined to represent the currently active application of the HSB application pair. It automatically keeps track of the state of the HSB state and routes the APL-APL communication requests to the active application.

The HSB configuration can be applied both on system servers and communication servers. When the HSB concept is applied on a communication server, the communication server is equipped with a base system and a local process database. The process data is transferred to the system server using the data mirroring functionality.

3.15.2 Communication redundancy

Redundant communication lines mean that two or several connections between the master and the slave form one logical connection. One of the connections is active, and if the active connection fails, another connection is used instead.

Redundant communication lines are supported for IEC 60870-7-101 slave, RP-570 slave and IEC 60870-7-104 master and slave.

In addition proxy STA objects can be used to provide redundancy for all master protocols. See System Object manual for more details.

3.16 Mirroring

The process data mirroring provides a powerful means for sharing process data in a SYS600 network with minimal engineering effort. This can be used to build hierarchical systems, for example, with one main control center, a number of regional control centers and more local control centers or substations. It can also be used for load distribution in large systems. The data mirroring can also be used to share data between a SYS600 HMI and SYS600 Gateway (COM500*i*), for example, when communication protocols that can have only one master are used. Mirroring can even be used to share process data among totally different kinds of applications. For example, electrical SCADA and district heating SCADA can share some indications, measurements and events.

The advantage of using data mirroring instead of standard communication, like IEC 60870-5-104, is that the data mirroring communication is much more efficient and the required engineering work to build up the system is minimal. In addition, several special functions, such as event buffering during communication breaks and handling of hot stand-by configurations, are automatically taken care of.

Data mirroring takes place between a Host and an Image. The Host is the source for the process data and the Image is a copy of the process data. The SYS600 node that receives the process data, for example, through IEC 61850 or LON, is the Host. This process data can be mirrored to another SYS600 node, which acts as an Image.

The data mirroring function is defined per SYS600 Station object (STA). When a station is configured for data mirroring, all process objects connected to that station are mirrored according to the mirroring definitions of that station. Each station can only have one source for the process data, either the process itself or a Host station in another SYS600 node. However, the process data of one station can be mirrored to up to ten (10) Image stations. A station can also act both as an Image and a Host. In this way a hierarchical system in several levels can be built up.

Although mirroring typically takes place between different computers, it is also possible to mirror data from one application to another in the same base system.

3.17 OPC connectivity

OPC is a defacto interface standard when connecting various devices and systems within the process automation industry. OPC is also becoming more and more used and accepted also in other application areas and within the power industry. It defines the exchange of data between a server and a client. The server provides data and the client uses the data. The client can also write data (commands, parameters and so on) to the server.

SYS600 provides OPC connectivity both in forms of clients and servers, that is, SYS600 can receive data from a device or system that provides its data through an OPC server, as shown in Figure 9. SYS600 can also provide its data to another system that acts as a client.

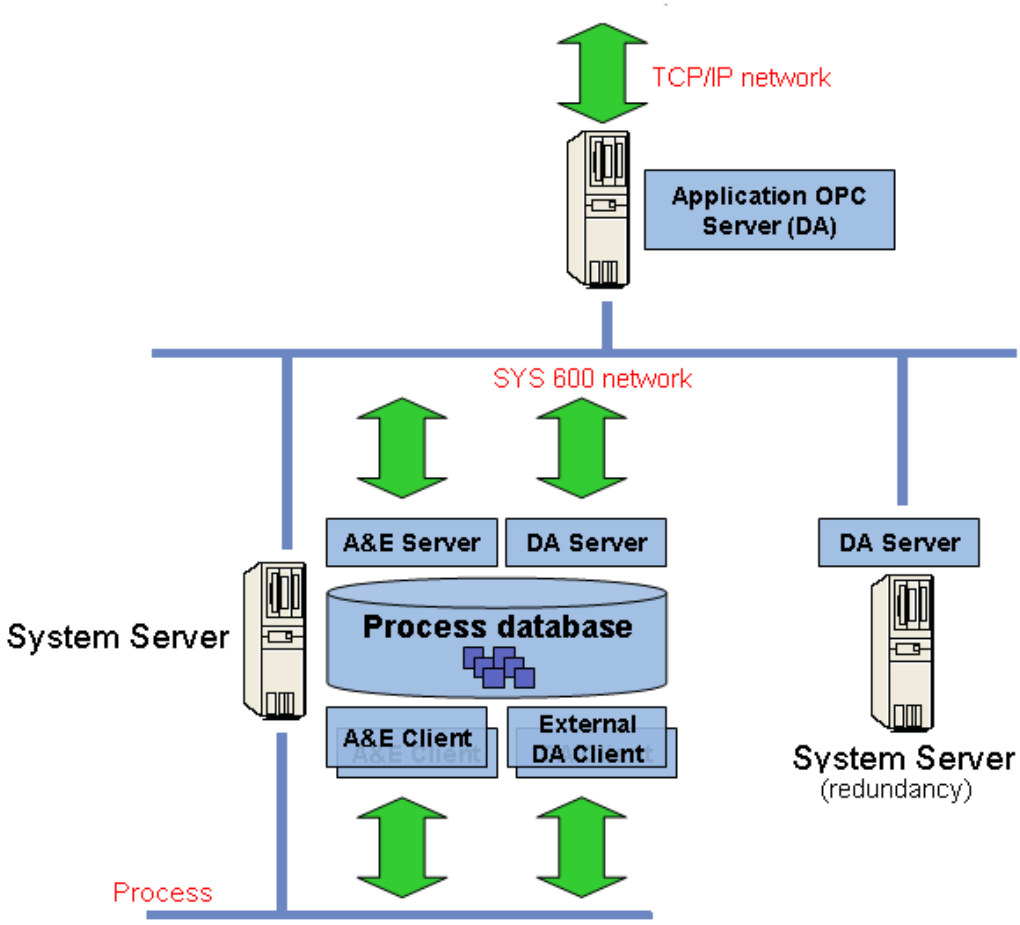


Figure 9: OPC connectivity

3.18 Capacity and performance scalability

The SYS600 system is highly scalable with regards to capacity and performance. This allows systems of significant differences in size to be built, starting from small monitoring systems with tens of IOs to large systems with hundreds of thousands of IOs.

The capacity and performance of the system is mainly affected by the computer processing capacity, which can be adjusted in two ways:

- by using computer(s) with various processing capacity
- by using various numbers of computers

3.18.1 Computer capacity

The computer type can be selected in order to match the capacity requirements of the system. The most important parameters are listed below:

- CPU performance and number of CPU cores
- RAM capacity
- Disk capacity and type

The most important system characteristics, that should be considered when designing the system, are listed below:

- Process communication load
- Number of simultaneous workplaces
- Intensity of archiving, calculations and reporting
- Possible other system specific functions

3.18.2 Distributing processing capacity

In this context, one system is characterized by one common image of the process. In SYS600 terms this means, that one common process database and one common event archive, which allows all alarms and events, is to be managed in one alarm/event list. If this one common process image is not required, the system can be built up of several independent sub-systems, for example, with common workstations but with individual workplaces for the different sub-systems. In the system, there is always one server that hosts the complete process database. This server can, of course, be redundant (HSB) as described in a separate section. However, the process database is very efficient and can handle tens of thousands of updates per second. Functions, that can be distributed, are process communication (PC-NET, IEC 61850 OPC Server & Client), Archiving, Reporting, Workplace processing and other post-processing activities. The functions are distributed so that so that each computer is allocated to its own task, and the process data is mirrored between the computers by means of the Mirroring function in SYS600.

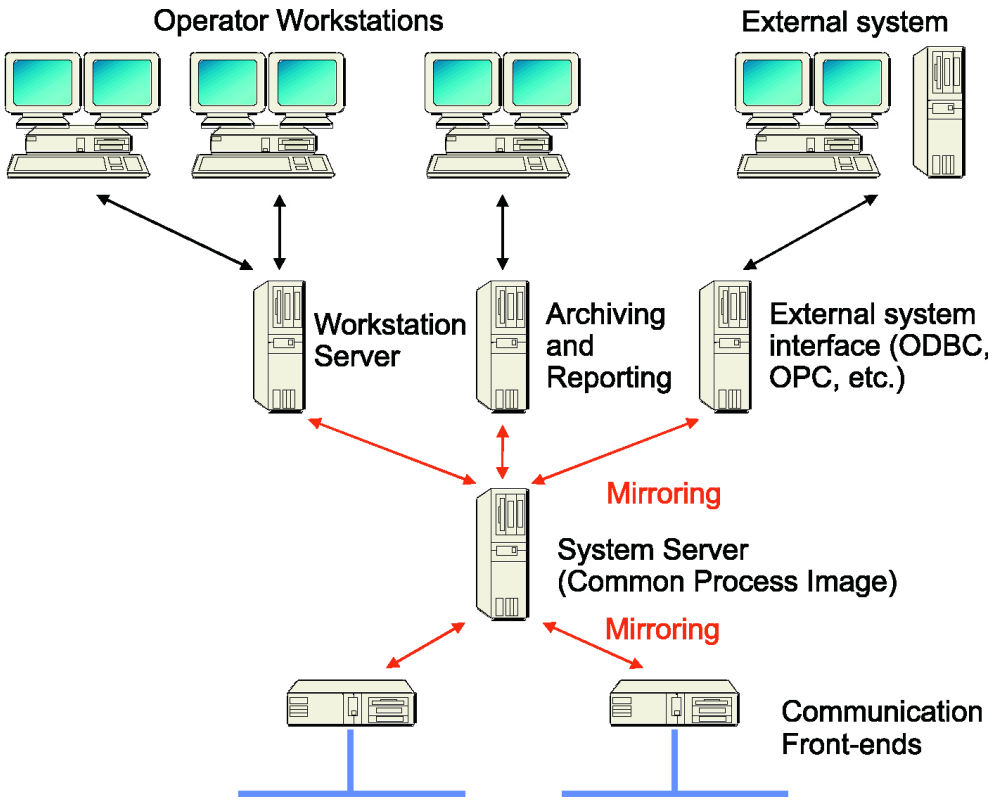


Figure 10: Example with distributed system servers

Figure 10 is an example of a system, where different functions have been distributed to achieve a higher capacity. The process image of the system server can be mirrored up to ten different servers. The number of communication front-ends connected to the system server is mainly limited by practical factors and the system server capacity. All nodes connected to the system by means of mirroring functionality have SYS600 installed.

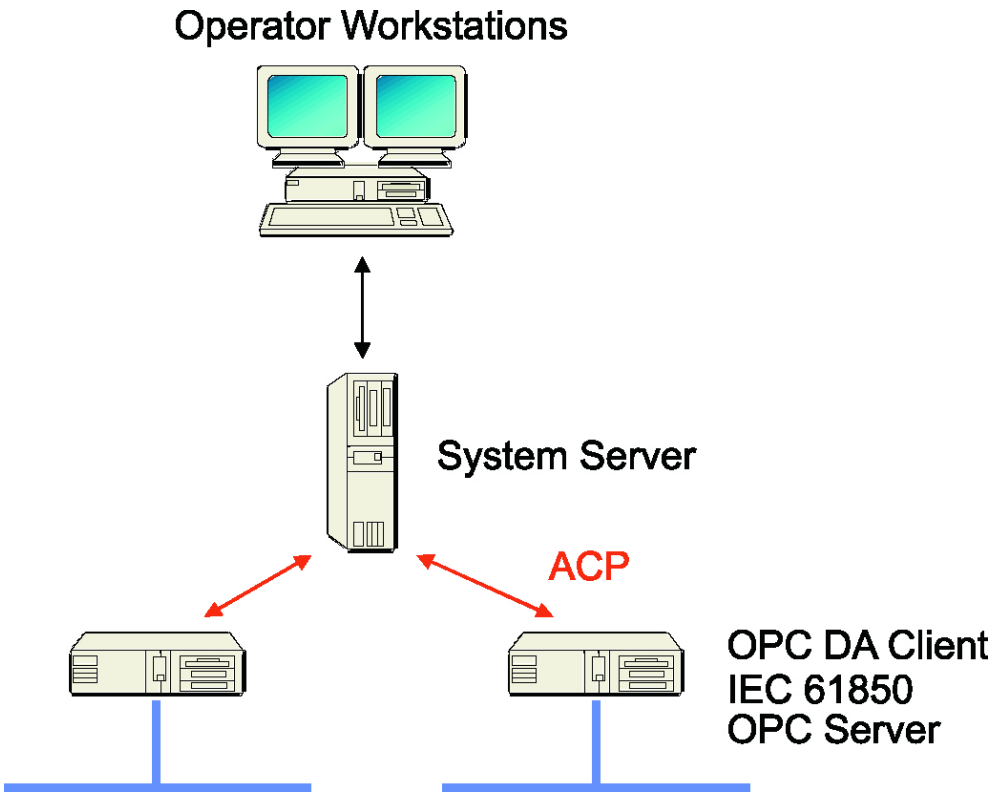


Figure 11: Example with distributed IEC 61850 front-ends

The communication front end can also be distributed in other ways, depending on the communication protocols used. In an IEC 61850 system, the External OPC DA Client and the IEC 61850 OPC server can run in its own computer. In addition, protocols or protocol converters implemented with CPI (Communication Programming Interface), can run in its own computer. For more information on building the IEC 61850 system, see SYS600 IEC 61850 System Design.

3.19 Cyber security

The electric power grid automation has changed significantly and continues to change with technological advancements. The new generation of control systems is more and more based on open standards and commercial technology, e.g. Ethernet and TCP/IP based communication protocols such as IEC 60870-5-104, DNP 3.0 and IEC 61850. This change in technology has brought huge benefits from an operational point of view, but it has also introduced cyber security concerns known from office or enterprise IT systems. Cyber security extensions according to IEC/TS 62351 are supported in IEC 60870-5-104 and DNP 3.0 protocols of SYS600 product. See protocol specific manuals and related sections in this manual for more information.

SYS600 Cyber security guideline collects instructions to harden the system. For example, there is detailed information about port numbers that can be used to configure hardware firewall correctly, or disabling devices (USB ports, CD/DVD drives etc.), which are not used. For more information, see SYS600 Cyber Security Deployment Guideline.

3.20 Product licensing

The functionality of the SYS600 product is dependent on the product license. The license describes the functions that can be used, the size or capacity of a specific function, as well as the time during which the product can be used.

The license is delivered as a .paf file (product authorization file) and is installed in the product by means of a dedicated tool.

The license is locked to a specific hardware with the help of one or several hardware keys. The license includes information about which hardware keys are used. If one of the specified hardware keys is present, the license is enabled. If the hardware keys are missing, the license is disabled. The hardware key is a USB stick that is installed into a USB port of the computer.

The hardware keys are identified by a unique ID number that is printed on the USB stick. The corresponding number is also used in the license.

The License can define if the hardware key is installed in the local computer or in another SYS600 computer in the same SYS600 network.

If the hardware lock becomes invalid during operation, SYS600 will work without restrictions for one week. During this time, the hardware lock should be fixed.

The following changes in the status of the hardware lock are reported:

1. Changes in the status of the hardware lock: valid/invalid
2. Missing hardware key
3. Found hardware key

Section 4 Configuration

The SYS600 system configuration is composed of objects and definitions for the base systems and process communication units as shown in Figure 12.

The main components for the configuration are the following:

- Each base system contains a set of base system objects that specify the base system itself and its environment. During operation, the base system objects are in the primary memory of the base system computer. The base system objects are created with SCIL commands when MicroSCADA X base system is started. They can be added and modified during operation.
- Each process communication unit contains a set of system objects that specify the unit itself and its environment. During operation, the system objects are in the memory of the PC (for example, process communication type PC-NET). Typically, these system objects have been configured by SCIL programs. Otherwise, the default values given by the process communication units will be applied. The system objects can be added and modified during system operation.
- Process communication units of type IEC 61850 with External OPC DA Client are configured using the Communication Engineering Tool for IEC 61850 OPC Server or the IET Data Loader if IET600 is used for the configuration.

4.1 Configuring system server

As a rule, when a device is added to the MicroSCADA X system, several configuration modules are affected. For example, when a process unit (station) is connected to a NET, additions and modifications are required in:

- base system which uses the device (base system objects)
- communication unit to which the device is directly connected to (system objects)

Concerning PC-NET and communication protocols, the configuration work is done with the System Configuration Tool. It automatically provides default values, which can be changed if needed.

The MicroSCADA X system configuration can be changed any time. However, in some cases, like when a new station is added to the system, a shutdown and restart is required for activating the changes.

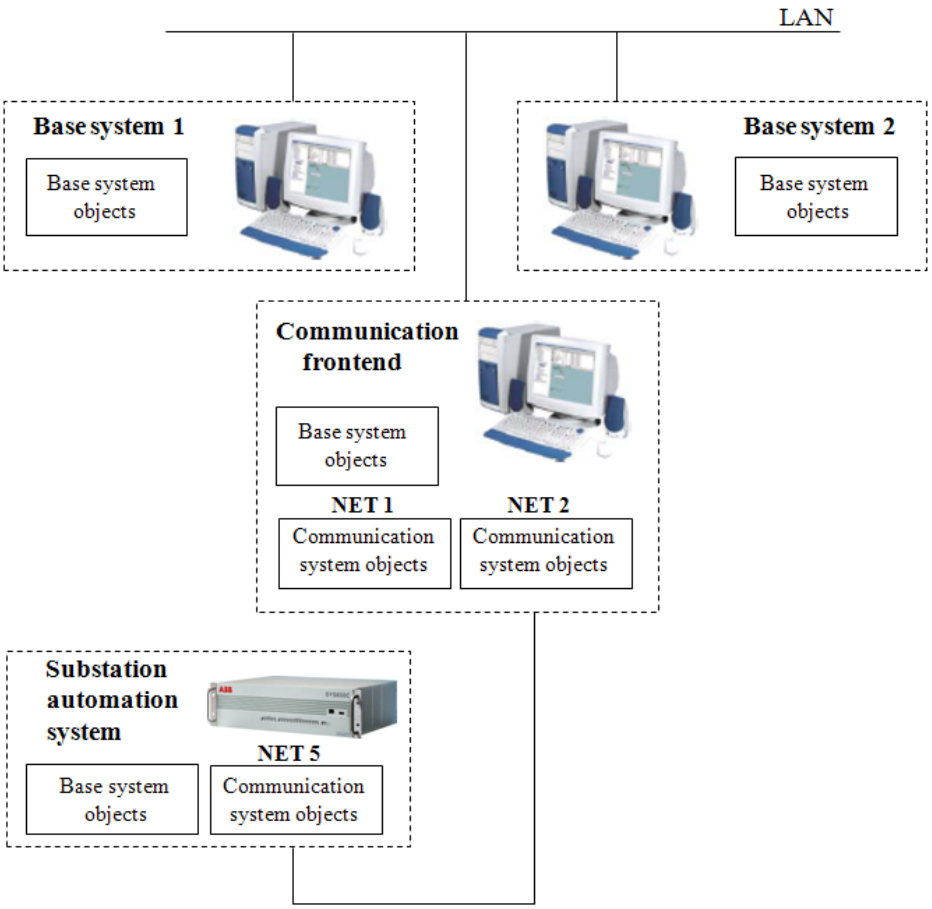


Figure 12: The configuration software modules in MicroSCADA X

4.1.1 Hardware and operating system

MicroSCADA X supports Intel® x86 compatible processors. For more information on system requirements and the full list of supported operating system, see SYS600 Installation and Administration Manual.

MicroSCADA X can be installed as a part of Windows domain, but it cannot be installed to the domain server. A Windows domain is a logical grouping of computers that share common security and user account information. This information is stored in a master directory database (SAM) which resides on a Windows server designated as a domain controller.

It is recommended to use MicroSCADA in a stand alone server to avoid complicated errors.

4.1.2 Base system (SYS)

The configuration of the MicroSCADA X base system is defined in the SYS_BASCON.COM configuration file. The configuration file SYS_CONFIG.PAR is a text file containing settings of memory parameters that cannot be set with SCIL (for more information, see Section 4.1.2.2). The file is read at system start up before the execution of SYS_BASCON.COM.

4.1.2.1 Base system objects

The file SYS_BASCON.COM is a text file containing SCIL statements for creating the base system (B) objects. The system base software package contains a SYS_BASCON.COM template file. The template file contains all the significant statements for configuring both single and hot stand-by systems. During installation, SYS_BASCON$COM is copied to SYS_BASCON.COM if the SYS_BASCON.COM does not previously exist.

The contents of the SYS_BASCON$COM file is listed below. Some configuration definitions have been excluded by commenting them. They can be taken into use by removing the comment sign (;) in front of the SCIL lines that create the base system object.

To edit the current SYS_BASCON.COM, click **SYS600 Control Panel > Configuration > Edit system configuration (SYS_BASCON.COM**) link.

The contents of the current SYS_BASCON.COM can be indirectly edited with the System Configuration Wizard. The wizard enables the elementary configuration and is specifically intended for persons unfamiliar to SCIL programming. The wizard is started with the SYS600 Control Panel, click **SYS600 Control Panel > Configuration > System configuration wizard...** link.

For more information, see Section 5.1.

The template file **SYS_BASCON$COM** consists of three different sections: a variable definition section, a statements section, and a project specific section. The actual configuration definitions are made in the variable definition section. The statements section contains the SCIL language statements for creating all the base system objects. This section is not intended to be modified by a project engineer. The last section in the end of the file is reserved for project specific and site specific configuration code.

The contents of the template file SYS_BASCON$COM is presented in Section 6.1.

4.1.2.2 Memory configuration

The SYS600 base system applies the policy of pre-allocating the virtual memory to be used during the whole session. Pre-allocation of the virtual memory is a usability issue: it ensures that the kernel, once successfully started, is able to request the memory resources it needs regardless of what happens in the system.

Memory allocation by demand may cause the computer to run in Virtual memory low state, which severely degrades the performance and, in the worst case, makes the system totally unusable.

Memory pools
The global memory pool is accessed by all SYS600 base system processes. It is used for process and report databases, execution queues, inter-process communication and so on. The pool is pre-allocated and fixed in size. If it is exhausted, a hot stand-by switchover is initiated. In a non-HSB system, the kernel tries to continue, but the application may not function properly.

A private memory pool is owned and used by one thread only. Private memory pools contain SCIL objects, such as variables, SCIL programs and Visual SCIL objects, and other private data of the thread. Currently, the main threads of classic monitor processes (PICN and PICV) as well as the main threads of reporting (REPR) and printer spooler (PRIN) processes have a private memory pool. In addition, the client specific threads of the OPC Data Access Server (OPCS) have their own private memory pool. Private memory pools have a configured maximum size to grow to. If the maximum size is exceeded, the running SCIL program is terminated.

The pools are configured in the configuration file SYS_CONFIG.PAR, which is read at system startup before the execution of SYS_BASCON.COM. If the SYS_CONFIG.PAR file does not exist,

the default values are used. A template, SYS_CONFIG$PAR, is copied to \sc\sys\active\sys_ during the installation of SYS600. SYS_CONFIG.PAR can be edited with any text editor.

Remember to remove the comment sign (;) in front of the lines.

SYS_CONFIG.PAR
SYS_CONFIG.PAR may contain the following parameters:

MEMORY_POOL_SIZE
This parameter specifies the size of the global memory pool in megabytes.

When a 32 bit Windows version is used, recommended values are 256, 384, 512, 768 and 1024. The default value is 256 (MB). The maximum possible value is slightly dependent on the operating system and the installed hardware and software. In any system, it is possible to define a 1 GB (1024 MB) pool, or a little larger.

When a 64 bit Windows version is used, bigger global memory pools may be specified. Recommended values extend to 1536, 2048 and 2560. The maximum size process and report databases (both containing 2 000 000 objects) require a 2560 MB (2,5 GB) pool.

For example, the line MEMORY_POOL_SIZE = 512 sets the size of the global memory pool to 512 MB.

MAX_PRIVATE_MEMORY_POOL_SIZE
This parameter specifies the maximum size of the private memory pool of any thread. The default value is 128 (MB). Recommended values are 32, 64, 128 and 256.

The content of the SYS_CONFIG$PAR is:

```
; File: Sys_config.par
; Description: Configuration of static base system parameters
; Commented lines (with leading ';') show default values
;-----------------------------------------------------------------
;
;MEMORY_POOL_SIZE = 256 ;Global memory pool size (MB)
; ;Recommended values: 256, 384, 512, 768, 1024
; ;(,1536, 2048, 2560 if 64 bit OS)
;MAX_PRIVATE_MEMORY_POOL_SIZE = 128 ;Max private memory pool size (MB)
; ;for SCIL threads (processes picn.exe, picv.exe, repr.exe, opcs.exe)
; ;Recommended values: 128, 256
;
;Obsolete parameters -- still supported but not to be used in new
applications:
;
;MEMORY_POOL_ADDRESS = 30000000 ;Memory pool start address
;ANALOG_SWITCH_STATE_OPEN = 2 ;The semantics for MicroTOPOLOGY of AI
;ANALOG_SWITCH_STATE_CLOSED = 1 ;process objects used for indicating the
;ANALOG_SWITCH_STATE_MIDDLE = 0 ;state of a switching device
```

Tuning memory pool sizes
If a classic monitor process (PICN or PICV) requires more memory than the memory pool size allows, the dialog **SCIL Application Error/Memory Pool Exhausted** is displayed. The dialog displays a critical error with information about the pool, which caused the error. The information is either "Private memory pool exhausted" or "Global memory pool exhausted". Report (REPR) and printer spool (PRIN) processes report their memory pool problems in the Notification Window and SYS_LOG.CSV.

If the private pool is exhausted, increase the value of MAX_PRIVATE_MEMORY_POOL_SIZE and restart SYS600, or optimize the picture, Visual SCIL dialog or command procedure to use less private pool memory. Large SCIL data structures, such as long vectors, are most likely

consumers of memory. If the global pool is exhausted, increase the value of MEMORY_POOL_SIZE.

The current usage of pools may be checked by evaluating the SCIL function MEMORY_POOL_USAGE, for example, in the Test Dialog. The function takes one argument, either “GLOBAL" or "PRIVATE", which specifies the pool to be investigated. For example, MEMORY_POOL_USAGE("GLOBAL") evaluates to a list value, where attribute USED tells the number of bytes used in the global memory pool and attribute FREE the number of available free bytes. When the databases have been loaded and the system is in its normal running state, the values USED and FREE should be of the same size class. If FREE is much smaller than USED (less than half of USED), increase the MEMORY_POOL_SIZE parameter by 50 or 100 %.

Oversized pools do not give any performance benefit on the system; they only increase the required size of the paging file. Increase the memory pool sizes only if the system has reported a memory pool shortage error or the MEMORY_POOL_USAGE function shows that the pool is nearly full.

The size of the physical memory (RAM) of the computer does not affect the pool sizes in any way.

The memory pools are allocated from the paging file ('swap file') of the computer. With currently available disks, there is no need to optimize the size of the paging file to the smallest possible. It is recommended that paging file size is set to at least 4 gigabytes.

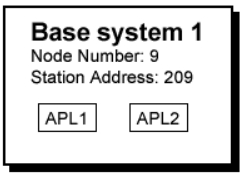
Report cache
The base system maintains one run-time cache of recently used disk-resident objects. The memory space required by the cache is allocated from the global memory pool.

The Report Cache contains history data of data objects and SCIL programs of command procedure objects. The size of the cache is defined by the RC (Report Cache Size) attribute of the SYS object (SYS:BRC). The recommended size in the SYS_BASCON$COM template is 8 MB. If the application intensively uses large data objects, some enhancement in performance may be achieved by using a larger value.

4.1.3 Applications (APL)

The application data is stored in the SYS600 base system computer as a directory branch under the application directory apl. For example, the application software of the local application sample is stored in the directory \sc\apl\sample. Figure 13 presents an example of the definition of two applications.

The application directory branch with its subdirectories should exist before a local application can be defined in the SYS600 base system configuration (for more information, see Section 4.1.3.3).



Base system configuration

Base system

```
#CREATE SYS:B = LIST(-
     ND = 9,-
     SA = 209,-
     ...)
```

Application 1:

```
#CREATE APL1:B = LIST(-
     NA = "SAMPLE",-
     AS = "HOT",-
     MO = (-1,-1,-1,-1,-1),-
     TT = "LOCAL")
```

Application 2:

```
#CREATE APL2:B = LIST(-
     NA = "TUTOR",-
     AS = "HOT",-
     MO = (-1,-1,-1,-1,-1),-
     TT = "LOCAL")
```

Figure 13: Example of the fundamental definition of a base system and the definition of two local applications

4.1.3.1 Configuring APL objects

The attributes of APL object are described in the System Objects manual.

At least one local application should be created in SYS_BASCON.COM, given a name (NA), set to LOCAL (TT) and to HOT (AS) and mapped for at least one monitor (MO).

If the primary application is not defined while creating the SYS object (the attribute PA), then the application that is created first in SYS_BASCON.COM is the default application. If no application number is given while opening a classic monitor, the default application is chosen. Likewise, if no application number is given when using the program interface, the default application is addressed.

4.1.3.2 Mapping devices

Monitors, printers and stations can be mapped for an application, which means that the application recognizes the devices under logical numbers. The station mapping, for instance, specifies the station numbers under which the application recognizes the stations. The station mapping has the following format:

APLn:BSTi = j

i	The logical station numbers as known to the application.
j	The real STA object numbers of the stations.

The printers and stations have a default mapping, which means that each logical application recognizes them under the real object numbers. Therefore, the printer and station mapping is needed only if the application for some reason needs to know the devices under logical numbers. If there are no obstacles, let the logical numbers be the same as the object numbers (that is i = j), do not change the default values of printer and station mapping.

4.1.3.3 Adding applications

To add a MicroSCADA application, follow the instructions given below:

1. Click **Control Panel** to open **Control MicroSCADA Applications** dialog (as shown in Figure 14).
2. Click **Add** button and type in the application name with capital letters (up to 10 characters).
3. Click **OK**.

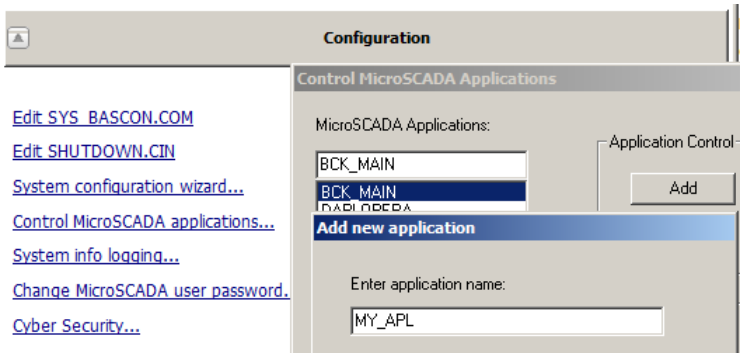


Figure 14: Adding an application

4. Depending on the usage of the application, prepare it for LIB 500.
5. Define application characteristic in file SYS_BASCON.COM.
6. When MicroSCADA is started for the next time, application definitions are taken in use.

4.1.3.4 Removing applications

To remove a MicroSCADA application, follow the steps below:

1. Stop running MicroSCADA.
2. Click **Control Panel** to open **Control MicroSCADA Applications** dialog.
3. Select application from the list and click **Remove**.
4. Confirm the operation.
5. Remove application definitions from the SYS_BASCON.COM.

4.1.4 Configuring license protection

When a license with a remote hardware key is installed in a system, node diagnostics must be set up to the system(s) that are to be searched for the installed hardware key.

Node diagnostics is enabled by setting the Diagnostic Interval (DI) attribute of the base system node:

```
#SET NODn:BDI = 30
;'n' is the node number of the SYS600 system
;that has (or may have) the dongle installed.
;Diagnostic interval is 30 seconds.
```

To check the status of the current license, open the License Tool and click the **Status** button.

4.2 Configuring Workplace X

Workplace X can be opened either locally in the server computer, in a remote computer or in mobile device. It can be opened using dedicated Workplace X Windows application or with a web browser. Workplace X uses HTTP/HTTPS to communicate with the web server in the

server computer. Most of the Workplace X configuration is related to the web server. Web server configuration is described in Section 4.4 section. Workplace X client configuration is described in Operation Manual for Workplace X.

4.3 Configuring Classic, Monitor Pro and Monitor Pro+ workplaces

A workstation is a computer with mouse, keyboard and one or more physical monitors that are connected with the MicroSCADA system. Connection to MicroSCADA system is established when user opens a monitor. The monitor can be either Monitor Pro+, Monitor Pro or Classic Monitor. Workplace can be either on a server computer or on a remote computer. If the workplace is on a remote computer, connection to the server computer is established over the network using remote client. Remote client means that programs of the workplace run on the server computer, whereas graphical output and mouse/keyboard input of the processes happen on the client computer. The promoted technology between MicroSCADA server computer and remote workplace computer is Windows Remote Desktop Services (RDS).

4.3.1 Classic Monitor

Classic monitor is a program for showing operator graphics that is implemented by using graphical resources provided by SCIL/Visual SCIL environment that is proprietary for MicroSCADA technology.

4.3.2 Monitor Pro

Monitor Pro is an application that is implemented by using graphical resources provided by the Windows operating system. Monitor Pro extends the functionality of Classic Monitor by introducing new features for process display navigation, such as zooming, panning and decluttering. In addition, features of Windows graphical user interface, like menu and toolbar customization by using drag/drop, Windows standard dialogs for open, save, fonts, printing support and so on, are supported more widely in application frame.

4.3.3 Monitor Pro+

Monitor Pro+ offers similar functionality as Monitor Pro using web technology.

4.3.4 Configuring the server for workplaces

To enable RDP connections to server, add Remote Desktop Services role to the server. This can be done in **Control Panel/Administrative tools/Manage Your Server**.

For information about Remote Desktop Licensing, see Microsoft website, Windows Server help.

4.3.5 Configuring client computer for workplace

Client software should be installed on client computer to connect to the server by using RDP protocol before a connection can be established. Client for Windows Remote Desktop Services is normally installed by default with Windows operating system. Normally, it can be found from **Start > Programs > Accessories > Communications > Remote Desktop Connection**. See Microsoft website if the client program is not installed.

4.3.6 Executing commands on client computer

Usually, some server initiated commands, like acknowledging audible alarms, automatically opening RDP session, and so on, should be executed on the client computer. These can be arranged by using third party software PsExec. For more information about PsExec, contact customer support.

4.3.7 Windows Remote Desktop services server (formerly known as terminal server)

Remote Desktop Services (RDS) is a component of Microsoft Windows operating systems. It allows a user to access applications on a remote computer over a network connection. Remote Desktop Services is Microsoft's take on server centric computing. Based on the Remote Desktop Protocol (RDP), Terminal Services was first introduced in Windows NT 4.0 Terminal Server Edition. Next Windows Server products have introduced several improvements and new features. Since Windows Server 2008 R2 Terminal Services has been renamed to Remote Desktop Services. RDS in Windows Server operating systems provides a new option for MicroSCADA monitor deployment. Windows Remote Desktop services are used to access older MicroSCADA X Monitor Pro/Pro+ applications from remote workstations.

4.3.8 Defining MON objects

4.3.8.1 Base system configuration

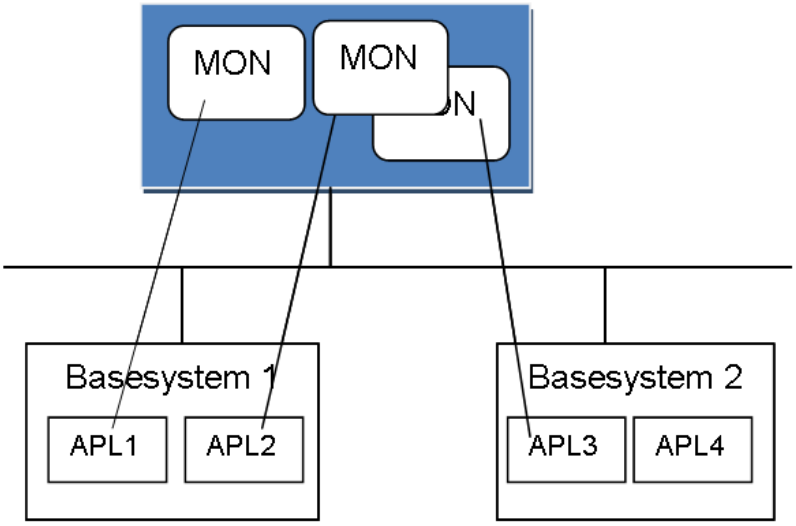
Each classic monitor should be defined as a MON object in the connected base system. Pro Monitors do not need MON objects. The MON objects should be mapped for the applications that use them. The monitor objects are defined equally, whether they are opened on the base system monitor or on a workplace. Figure 15 shows an example of a workplace with three classic monitors opened to three applications in two separate base systems.

Make the following object definitions in each base system:

1. Create MONn:B objects, one for each classic application monitor that is opened on the base system monitor or on connected workplaces. Assign the MON objects the following attributes:
   DT = "VS"
   TT = "LOCAL"
   Up to 100 MON objects can be created per base system.
2. Define monitor numbers for each application by setting the APLn:BMO attribute to -1 by using freely chosen monitor numbers as indexes.
   Example:
   Application 1 is created with the following definition:

   ```
   @MON_MAP(1..5) = -1
   #CREATE APL1:B = LIST (-     ....     MO=%MON_MAP     ....     )
   ```

   Now the value of indexes 1 ... 5 of attribute MO is -1 (APL1:BMO(1..5) = (-1,-1,-1,-1,-1)), which means that monitor numbers 1 ... 5 can be opened to view application 1.



Configuration of basesystem 1:
Monitors:
`#LOOP_WITH I = 1..20`
`  #CREATE MON'I':B = LIST(-`
`  DT = "VS",-`
`  TT = "LOCAL")`
`#LOOP_END`
	Configuration of basesystem 2:
	Monitors:
	`#LOOP_WITH I = 1..20`
	`  #CREATE MON'I':B = LIST(-`
	`  DT = "VS",-`
	`  TT = "LOCAL")`
	`#LOOP_END`

Application 1:
`@MON_MAP(1..20) = -1`
`#CREATE APL1:B = LIST (-`
`  ....`
`  MO=%MON_MAP`
`  ....`
`  )`
	Application 3:
	`@MON_MAP(1..20) = -1`
	`#CREATE APL3:B = LIST (-`
	`  ....`
	`  MO=%MON_MAP`
	`  ....`
	`)

Application 2:
`@MON_MAP(1..20) = -1`
`#CREATE APL2:B = LIST(-`
`  ....`
`  MO=%MON_MAP`
`  ....`
`)
	Application 4:
	`@MON_MAP(1..20) = -1`
	`#CREATE APL4:B = LIST(-`
	`  ....`
	`  MO=%MON_MAP`
	`  ....`
	`)

Figure 15: Example of a workplace that is connected to two base systems and four applications

4.3.9 Monitor Pro and Monitor Pro+ configuration

For information on command line support, see SYS600 Application Design.

4.3.9.1 Process Display Specific Configuration Files

4.3.9.2 Process Display Specific Menu Files

In addition to Process Display specific menu files, there can be application and user specific Process Display menu files. The items are separated in the menu structure.

- Application Specific
  Application specific Process Display menu files are in the folder [Appl path]\PAR\APL \PROCESS\MENU. Application specific Process Display menu files are common for all Process Displays.
- User Specific
  User specific Process Display menu files are in the folder [User path]\PROCESS\MENU. If the folder exists, the contents of application specific Process Display menu folder is not investigated at all. This makes it possible to skip the application specific Process Display menu files by just defining an empty user specific one.

4.3.9.3 Visibility Shortcut Files

If visibility files have been defined, the following operations are blocked:

- **File > Open** menu item will not allow user to open any Process Display.
- Process Display drag and drop to application window is not allowed.
- Custom commands concerning opening Process Displays is blocked.

4.3.9.4 Application Specific

The shortcut files in [Appl path]\PAR\APL\PROCESS\TOOLBAR_SHORTCUTS are shown to all users in application Process Displays toolbar in Monitor Pro. If this folder is not empty (excluding user specific folders), the files in [Appl path]\PICT\ are left investigated. Other Process Displays than the ones in application Process Displays toolbar are not allowed to open.

For backwards compatibility, the existing files in the old folder are moved programmatically to a new folder when Monitor Pro is started.

4.3.9.5 User Specific

The shortcut files in [User path]\PROCESS\TOOLBAR_SHORTCUTS\ are shown only to appropriate user, in application Process Displays toolbar in Monitor Pro. If this folder is not empty, the files in [Appl path]\PICT\ folder and application specific shortcut files are left investigated. Other Process Displays than the ones in application Process Displays toolbar are not allowed to open.

For backwards compatibility, the existing files in the old folder are moved programmatically to a new folder when Monitor Pro is started.

4.3.9.6 Process Display menu

A **Process Display** menu displays both the common parts for the process pictures and the specific parts for the currently active process picture.

The menu structure is similar to the Windows **Start** menu. The menu commands are organized as folders and files in the file system. Therefore, no special tool is needed to configure the menu. The configuration is done by organizing the files, such as programs, documents or shortcuts and the directories in a file system. For example, a menu command can be:

- a file
- a folder containing submenu commands
- a link
- a shortcut to a file
- a shortcut to a directory
- an Internet hyperlink

4.3.9.7 Defining shortcuts to Process Displays

The user access rights can be restricted by showing the required Process Displays as shortcuts. The user can only open the Process Displays shown in a toolbar, if the shortcut files have been defined.

The following operations are unavailable for the user:

- Opening Process Displays by using the menu operation **Main/Open File**.
- Dragging Process Displays to an application window.
- Custom commands related to opening the Process Displays are blocked.

Shortcut files should be present at the operating system level. Shortcuts cannot be created by copying the picture.

Application specific Process Display shortcuts are shown to all users. The shortcut files are located in Appl path]\PAR\APL\PROCESS\TOOLBAR_SHORTCUTS. The user is not allowed to open Process Displays that are not defined on toolbar of the application specific Process Displays.

User specific Process Display shortcuts are shown individually to each user. The user specific shortcut files are located in [User path]\PROCESS\TOOLBAR_SHORTCUTS. The user is not allowed to open Process Displays that are not defined on the toolbar of the user specific Process Displays.

If the user specific shortcut files exists, the application specific files are ignored.

4.3.9.8 Process Display Context Menus

MENUS directory:

The MENUS directory consists of directories, such as all, objecttypes and instances.

- Menu commands, which are to be shown on each shortcut menu, should be stored under the directory all.
- The directory objecttypes consists of all object types that a station overview can have. Menu commands that each object type can have are also located here.
- It is also possible that an instance of a breaker might have some menu commands that are instance specific, such as a figure of a breaker, online video stream, maintenance log or a web link to the manufacturers home page. In that case, the menu commands are located in \<unique object id>.
- The menu for all objects are located in <apl>\MENUS\all.

SYS600 supports two-letter language codes. When generating the menu structure, use the /culture <language> option. When the option is selected, the menu commands are displayed under the culture specific folder.

Table 3: Menu command descriptions

Name	Description	Path
<apl>	Path of the application	For example, C:\sc\apl\510_403_1
<object type>	For more information about object types, see the SYS600 Application Design manual	<apl>\MENUS\objecttypes\ <object type>(common menu for an object type)
<display name>	Name of the display file (without extension .v)	

Menu structures are not shadowed. If a HSB system is used, the menus have to be manually copied between hot stand-by computers after the modifications have been done. Otherwise, the directories are deleted during switch over.

4.3.9.9 Troubleshooting

Table 4: Possible problems with context-sensitive shortcut menus

Description of the problem	Possible cause	Solution
Only a description of a shortcut menu is displayed. A message "No items" or the icon indicates that the shortcut menu is empty.	There are no commands in the menu directory, that is, the menu is empty.	Check that the application has a menu structure. For more information, see SYS600 Application Design. Furthermore, check whether a culture has been defined similar to the user for the menu structure.
The folder is displayed as a menu command and can be browsed. The folder does not contain submenu commands.		
The command Open With dialog is displayed when running a menu command.	The file type of the menu command is unknown. The file has not been associated with any program.	Use the command Open With dialog to select the program in which you want to open the file.
A menu command is disabled.	User is not authorized to run the menu command or a link target does not exist.	Check the authorization level or the link target.
The custom object type does not have menu commands.	A menu structure was not generated for the custom object type or the type does not have an SAObjectType attribute.	Add a menu structure manually for the custom object type and ensure that the object has an SAObjectType attribute.
An instance specific menu structure is not created for a symbol.	No view was defined when the menu structure was generated or the symbol's Object Name field is empty.	Use the Display Builder to set the Object Name field for the symbol. Run the menu generator and use a viewpath argument to generate menus for instances.

4.3.9.10 OpenRemoteDesktop

The OpenRemoteDesktop program can be used for opening a connection from a workstation to the active server in the MicroSCADA HSB system. The program that runs on client computer inspects both servers, detects active server of HSB pair and establishes a terminal server session to it.

The program transfers audible alarms from the server computer to the workstation computer.

The program makes the following checks on the servers of the HSB pair when initializing the connection:

- The server connection is checked using connection state query to SYS600 OPC DA Server over (D)COM. The check is done for both servers.
- If MicroSCADA is running on the servers, the program detects which one of the servers contains the application that is in the HOT state.
- After finding an appropriate HSB server that has the main application in the HOT state, the program opens the Remote Desktop connection to that server.
- After opening the Remote Desktop session, the program stays active as a tray application, and subscribes OPC items for the application state and an audible alarm for monitoring.
- If the application goes to another state than HOT, the program automatically restarts a procedure to find out the active server and to establish the connection.
- If an audible alarm occurs on the server, the program flashes the audible alarm icon in the application tray and plays the alarm that is configured in the configuration file.

Configuring RDP files
To configure RDP files:

1. Start the remote desktop client program (RDP client configuration) by selecting **Start/All Programs/Accessories/Remote Desktop Connection** (in Windows 7), or running the command `mstsc` from the command line.
2. In the RDP configuration, define <drive>:\sc\prog\sa_lib\Framewindow.exe as the startup program.
3. Define the working directory to <drive>:\sc\prog\sa_lib.
4. Define the parameters that close Monitor Pro without message boxes when application switches to COLD state. To define the parameters, use the following command line arguments:<drive>:\sc\prog\sa_lib\Framewindow.exe -sdown:closemonitorpro -silentexit

INI file configuration
The program uses the configuration file OpenRemoteDesktop.ini that must be located in same directory as the program file. The INI file should contain following required and optional parameters in the option group OpenRemoteDesktop:

SERVER1	IP number or node name of primary server.
SERVER2	IP number or node name of secondary server.

The following parameters are optional:

LANGUAGE	Language that will be used for finding the localized resource DLL. It is possible to give this information also from command line using -lang command line argument.
RDP_RECONNECT	If set to TRUE it is checked whether mstsc process (RDP session) is running. If such a process is running, new RDP session won't be created for the last known HOT application. Default value is FALSE so by default such checking won't be executed.
RDP1	Remote Desktop Protocol (RDP) file defining session to primary server.
RDP2	Remote Desktop Protocol (RDP) file defining session to secondary server.
RDP1_1...RDP1_9	Additional Remote Desktop Protocol (RDP) files defining session to primary server.
RDP2_1...RDP2_9	Additional Remote Desktop Protocol (RDP) files defining session to secondary server.
Table continues on next page	

SERVER1_APP	Application number where to connect on primary server. Default value is 1.
SERVER2_APP	Application number where to connect on secondary server. Default value is 1.
AUDIBLE_ALARM_OPC_ITEM	The OPC item name for audible alarm process object. The OPC item should be given without attribute definition, i.e. the item name must be \\APL\1\P \ACK_SOUND\1 rather than \\APL\1\P\ACK_SOUND \1:OV.
BALLOON_TIMEOUT	The timeout (in milliseconds) how long the balloon notification stays on the screen. Note that the minimum and maximum timeout defined by the operating system are 10000 and 30000 milliseconds (respectively), so values that are too large are set to the maximum value and values that are too small default to the minimum value.
BASE_SYSTEM_WAIT_TIME	The time (in milliseconds) that is waited for the base system to start. With large applications there might be need to increase the value. Default value is 5000 milliseconds.
AUDIBLE_ALARM_WAIT_TIME	The time (in milliseconds) to wait for the audible alarm OPC item subscription. Default value is 1000 milliseconds.
AUDIBLE_CONNECTED	WAV audio file name for SYS600 connected.
AUDIBLE_DISCONNECTED	WAV audio file name for SYS600 disconnected.
AUDIBLE_ALARM_FILE_1	WAV audio file name for alarm sound 1.
AUDIBLE_ALARM_FILE_2	WAV audio file name for alarm sound 2.
AUDIBLE_ALARM_FILE_3	WAV audio file name for alarm sound 3.
AUDIBLE_ALARM_FILE_4	WAV audio file name for alarm sound 4.
AUDIBLE_ALARM_FILE_5	WAV audio file name for alarm sound 5.
AUDIBLE_ALARM_FILE_6	WAV audio file name for alarm sound 6.
AUDIBLE_ALARM_FILE_7	WAV audio file name for alarm sound 7.
CLOSE_DISCONNECTED	Determines if active RDP session window is automatically closed when the session is disconnected. Default value is TRUE.

The RDP session opening is not obligatory, OpenRemoteDesktop.exe can also be used for transferring audible alarms only. In this case the RDP parameters can be left empty in the INI-file.

Example contents of the configuration file:

```
[OpenRemoteDesktop]
LANGUAGE = EN
SERVER1 = 192.168.225.132
SERVER2 = 192.168.225.133
RDP_RECONNECT = FALSE
RDP1 = SYS_1.rdp
RDP1_1 = SYS_1_1.rdp
RDP1_2= SYS_1_2.rdp
RDP1_3 = SYS_1_3.rdp
RDP2 = SYS_2.rdp
RDP2_1 = SYS_2_1.rdp
RDP2_2 = SYS_2_2.rdp
RDP2_3 = SYS_2_3.rdp
SERVER1_APP = 1
SERVER2_APP = 1
AUDIBLE_ALARM_OPC_ITEM = \\APL\1\P\ACK_SOUND\1
```

```
BALLOON_TIMEOUT = 0
BASE_SYSTEM_WAIT_TIME = 5000
AUDIBLE_ALARM_WAIT_TIME = 1000
AUDIBLE_CONNECTED = C:\Windows\Media\Windows Ringin.wav
AUDIBLE_DISCONNECTED = C:\Windows\Media\Windows Ringout.wav
AUDIBLE_ALARM_FILE_1 = C:\Windows\Media\ding.wav
AUDIBLE_ALARM_FILE_2 = C:\Windows\Media\chimes.wav
AUDIBLE_ALARM_FILE_3 = C:\Windows\Media\recycle.wav
AUDIBLE_ALARM_FILE_4 = C:\Windows\Media\ringout.wav
AUDIBLE_ALARM_FILE_5 = C:\Windows\Media\chord.wav
AUDIBLE_ALARM_FILE_6 = C:\Windows\Media\tada.wav
AUDIBLE_ALARM_FILE_7 = C:\Windows\Media\notify.wav
CLOSE_DISCONNECTED = TRUE
```

Audio alarm handling

The program monitors the value of the audio alarm process object on the server computer. When the value of the process object changes to non-zero, audio alarm starts to play on the client computer. When the audio alarm sound is on, the icon of the program in the task bar changes to red animated loudspeaker. Logical name and index of the process object can be configured in the INI file by defining the OPC item ID. The value of the process object defines what audio sound will be played. For example, when the process object value is 1, the audio file defined by the parameter AUDIBLE_ALARM_FILE_1 is played, when the value is 2, the audio file defined by the parameter AUDIBLE_ALARM_FILE_2 is played and so on. The audio alarm sound stops when the process object value changes back to 0. The audio alarm continues to play until the process object value is set to 0. The value can be set to 0, for example, in the command procedure that is executed by clicking a toolbar button.

Indications

The following icons indicate the status of the application in the tray area of the task bar:

Icon	Description
	The connection has been successfully established to the active server.
	The connection to the active server has been lost.
	Indicators for an audible alarm. The connection has been successfully established to the active server, and the audible alarm is on.

When the application state changes, for example when the connection is established to the server, the status changes are indicated to the user by showing a balloon in the tray area:

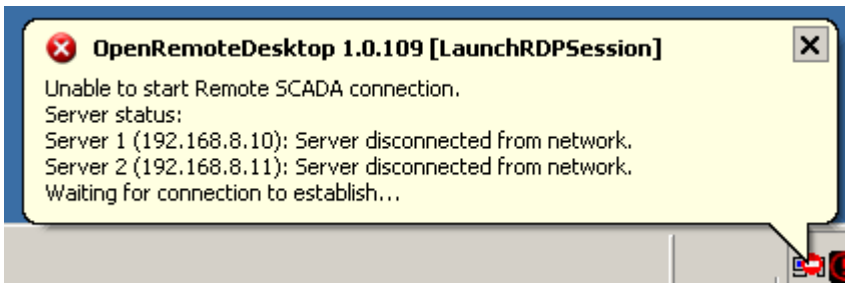


Figure 16: Server is disconnected from the network

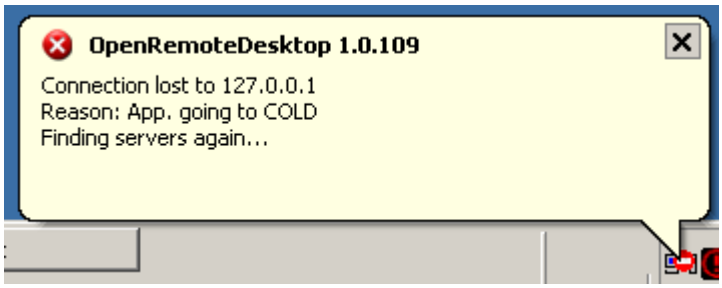


Figure 17: Finding servers again

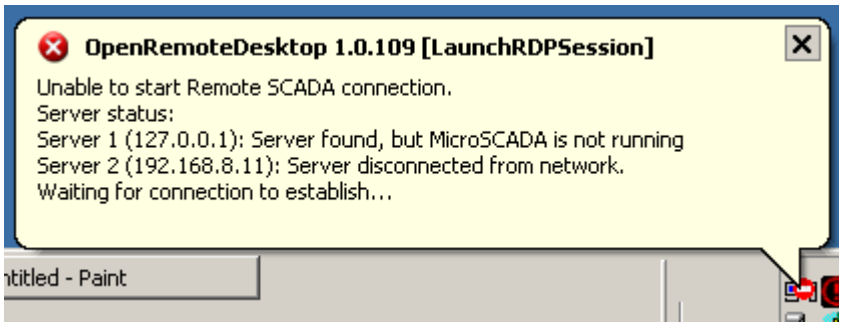


Figure 18: Server is found, but MicroSCADA is not running

The message "Server found, but MicroSCADA is not running" can also mean that the workstation user does not have the authority to access named pipes on the server computer or the access is blocked by the firewall. To test the access rights, run net view \\<host> command, for example:

net view \\10.58.125.47

To solve the problem, log in to the workstation computer as a user that has sufficient access rights to the server computer.

Program status
Check the status of the program by right-clicking the icon in the taskbar.

Figure 19: Program status

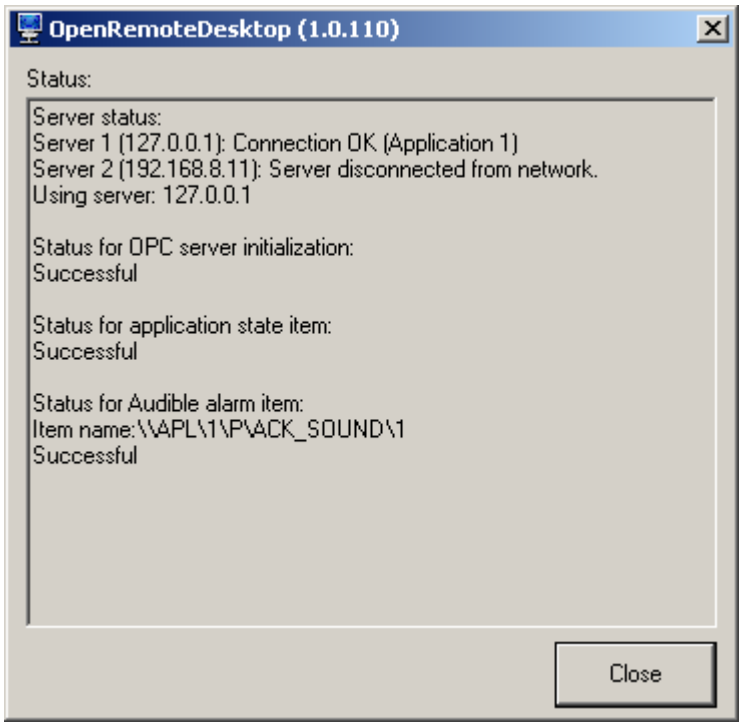


Figure 20: OpenRemoteDesktop status

Closing OpenRemoteDesktop
Select **Exit** from the context menu of the tray icon.

When the program is closed, monitors are not opened to the server.

Closing Remote Desktop Server session
If the Remote Desktop server session is closed, it can be reopened by double-clicking the icon on the taskbar. If the user tries to restart the program when another instance of the program is already running, RDP sessions will be re-started unless RDP_RECONNECT is set to TRUE.

4.3.9.11 Configuring RDP session

Configure a startup program for a RDP session:

1. Open the **Remote Desktop Connection** dialog and select the **Programs** tab.
2. Enter the full path and program name and the starting folder:
   Program path and file name:
   ```
   <drive letter>:\sc\prog\sa_lib\framewindow.exe
   ```
   Start in the following folder:
   ```
   <drive letter>:\sc\prog\sa_lib
   ```

Figure 21 shows the **Remote Desktop Connection** dialog with Monitor Pro as a startup program.

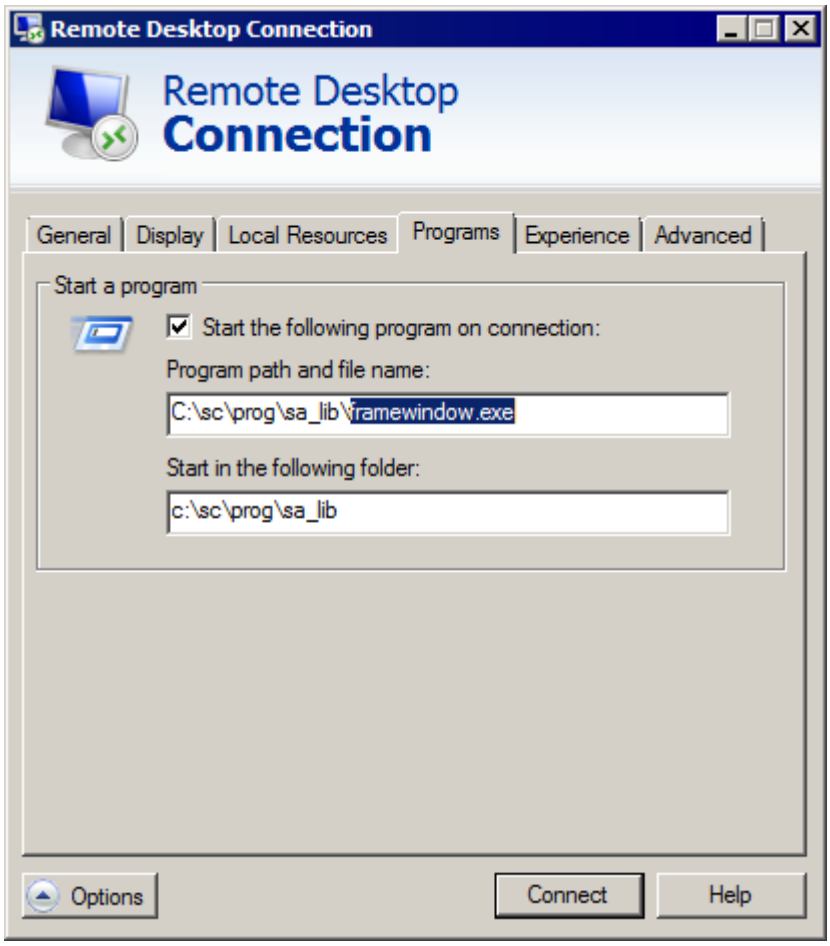


Figure 21: Remote Desktop Connection dialog

4.3.9.12 Opening Monitor Pro automatically at application startup

Monitor Pro can be configured to start automatically when an application state becomes HOT.

This example shows how this configuration can be done.

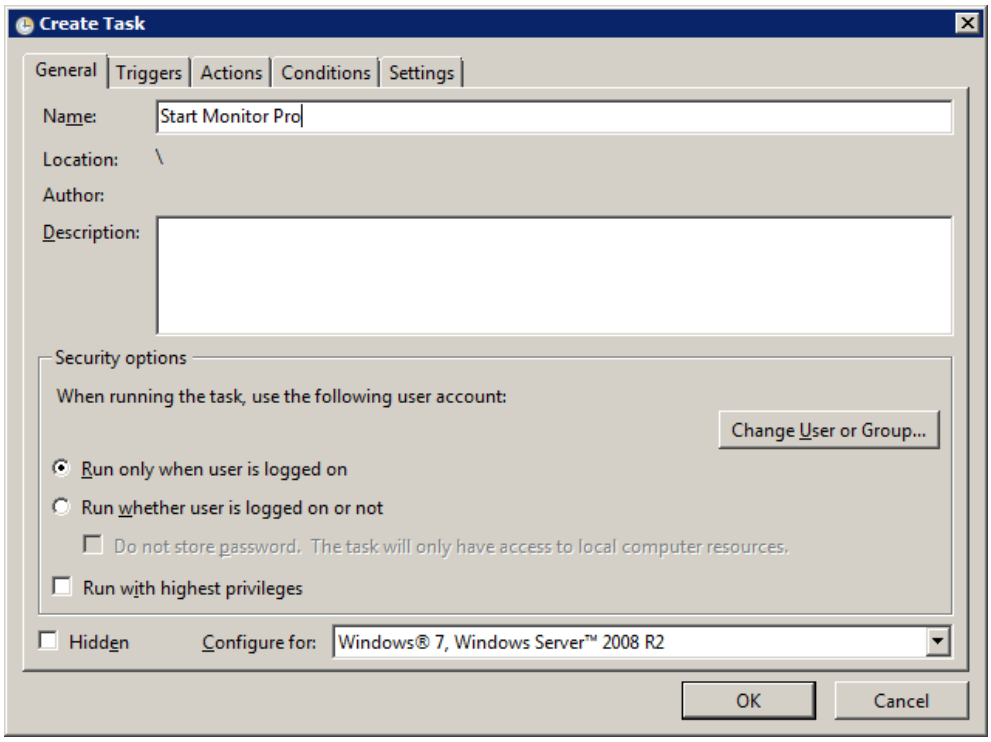
1. Add Trigger Event Generation
   Add the following line into APL_INIT_1 procedure. This procedure is executed when this application state changes to HOT. In HSB systems the same command should be added into APL_INIT_H, which is executed when switch-over occurs.

   ```
   @callEvent=ops_call("start /b eventcreate /t information /so SCIL /l
   application /id 4 /d ""Application 1 is hot"" ")
   ```

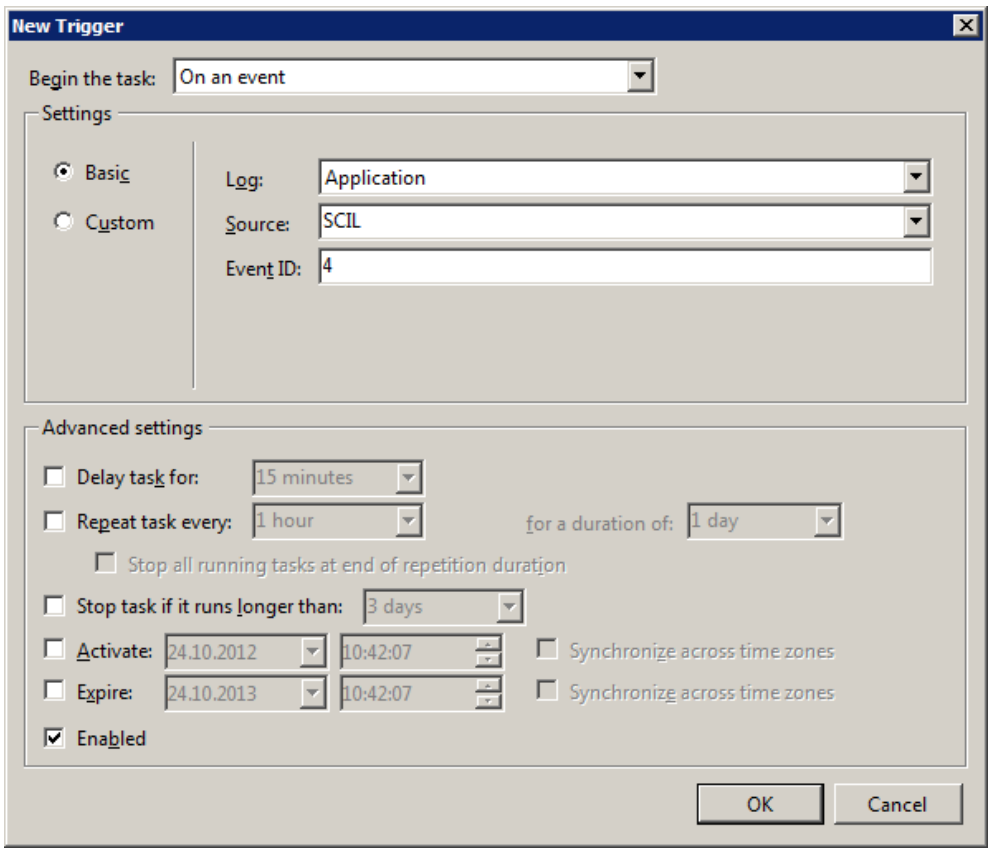
   In this example, values /so SCIL, /id 4 are used later but they may differ from the ones used here. Used event ID must be an integer between 1 and 1000. For more information about eventcreate, see Microsoft Windows documentation.

2. Configure Windows Task Scheduler
   Start Task Scheduler (**Control Panel/System and Security/Administrative tools/Task Scheduler**) and create a new task.

   2.1. Start configuring task by giving it a descriptive name.

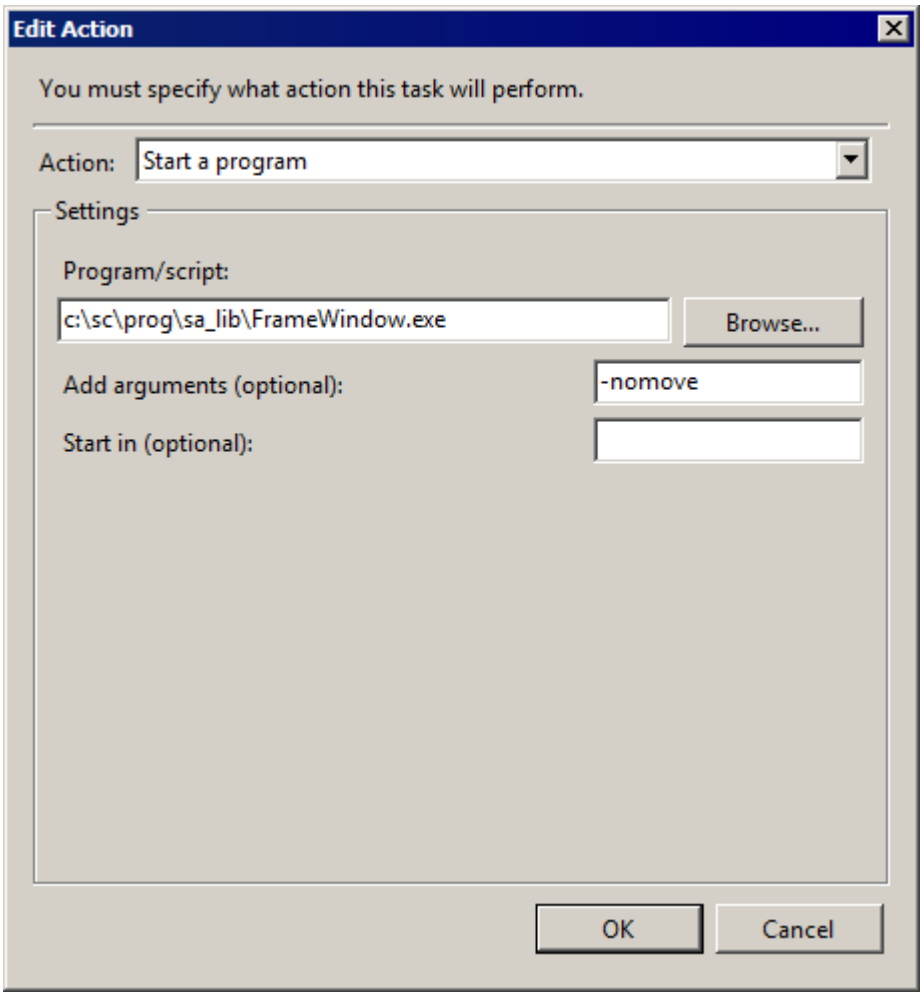


2.2. Create a new Trigger which will activate this task. Value for these settings comes from the event create call added to procedure code.



2.3. Create a new action, which determines what happens when this task is activated. In this case, start the FrameWindow.exe process from the appropriate location. The framewindow start parameter as e.g. -nomove needs to be added to the "Add

arguments" field. When using Monitor Pro+, the "<system drive>:\Program Files (x86)\ABB\MicroSCADA Pro\FrameWindow" program path must be used.



4.4 Configuring Web Server

SYS600 10 includes a web server which must be configured to enable following features

- Workplace X
- Monitor Pro+
- SYS600 Tool Launcher, IET Data Loader, View Builder and other local tools

4.4.1 Localhost Use Only

For systems where web server shall be only allowed to be accessed locally, e.g. when Workplace X is not used at all or it is only used locally in the same computer where SYS600 base system is running, the web server can be configured to allow only local connections. This is configured in the Web Server Configuration -section of SYS600 Control Panel with Localhost Use Only -checkbox. When the checkbox is checked, the web server only allows local connections and the communication is configured automatically to use HTTP protocol that does not use encryption or require certificate management. Web server can be accessed using localhost as host name. In case the default HTTP port number 80 is already reserved by some other software in the computer, the used port number can be changed using the Web Server Port parameter in SYS600 Control Panel.

4.4.2 Remote & Local Use (Workplace X)

When Workplace X is used, the Web server needs to be typically configured to allow access also from remote computers. This is configured in the Web Server Configuration -section of SYS600 Control Panel with following parameters

- Localhost Use Only -checkbox must be unchecked to allow also remote access
- Use HTTPS -checkbox specifies whether secured HTTPS or unsecure HTTP communication shall be used between web server and clients. It is strongly recommended to always use secure communication when remote connections are allowed.
- Web Server Certificate -parameter specifies the certificate used for secured HTTPS communication when Use HTTPS -checkbox is checked. See more information about certificates in Web Server Certificate section below.
- Web Server Port –parameter specifies the port which web server is listening to. By default, HTTPS uses port 443 and HTTP uses port 80. In case the computer already has some other software that is using these port numbers, then another free port number must be chosen and used when connecting to the web server. Note that enabling or disabling encryption will change port number back to the default value. No other process running in the system should use same port number. SYS600 Control Panel does not check or enforce this and the startup of SYS600 will result in web server being unreachable if there is a conflict between SYS600 and some other process. In that case either conflicting software or SYS600 must be reconfigured to use other port number and the SYS600 must be restarted for the changes take effect.
- Web Server Hostname –parameter specifies the host name used by the local tools to access the web server. When secure communication is used, the hostname must be included in the selected certificate. Hostname can be also used for accessing the web server remotely. Remote computer or mobile device needs to know the hostname e.g. by setting up a Domain Name Server that can resolve the hostname to IP Address. It is possible to use also IP address as hostname, but it needs to be also included in that case in the certificate.
- Redirect HTTP -checkbox specifies if the client requests to HTTP URL are automatically redirected to HTTPS-port

SYS600 web server is using HTTP Strict-Transport-Security when HTTPS mode is enabled. For more info, see 'https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security'. This means that if HTTPS is successfully used, further HTTP usage will be prevented. Cached settings must be deleted in order to enable HTTP usage.

- SYS600 Workplace X: Delete folder %APPDATA%\ABB\MicroSCADA Pro\WebMonitor
- SYS600 Monitor Pro+: Delete folder sc\TEMP\cef
- Standalone Chrome: Clear the browsing data and delete domain security policies related to SYS600 by opening 'chrome://net-internals/#hsts'. Use the Delete domain security policies section and enter the SYS600 web server host name into the Domain field.

4.4.3 Hot-Standby Configuration

For hot-standby systems, additional configuration attributes must be set to enable Workplace X switch between the systems. The address/host name of the own web server shall be set to the Web Address (SYS:BWA) attribute in each system. Additionally the shadowing partner web address (web server hostname) shall be set per application by using Shadowing Partner Web Address (APL:BSA) attribute. In single systems the SYS:BWA or APL:BSA attributes should not be set. For more information about attributes, see SYS600 System Objects.

Secured communication (HTTPS) shall be used for Workplace X in hot-standby configuration to enable automatic switch between the systems.

4.4.4 Web Server Certificate

When web server is configured to use secure communication by checking the Use HTTPS - checkbox, a web server certificate (SSL) is required. Web server certificate binds the cryptographic key to web server domain name/hostname and organization information (that is, company name). Certificate is used to encrypt the communication between the web server and clients, and for ensuring that client is connected to a trusted web server. There are multiple ways to create and manage the web server certificates. Two possibilities are described below. If customer has existing infrastructure for certificate creation and management, it is typically the preferred way of managing also the SYS600 web server certificates. Note, that SYS600 needs to be restarted if the used certificate is changed.

The expiration date of the active Web Server certificate is displayed in the SYS600 Control Panel and SYS600 Notify windows. Expired certificate disables SSL communication and thus prevents Web Server usage.

4.4.4.1 Using SYS600 Self-Signed Certificates

SYS600 self-signed certificates can be used e.g. for testing when certificates for the final installation are not yet available or in small system setups, where only one or few certificates are needed. SYS600 control panel searches Personal folder under the Local Computers account in the Windows Certificate Store for suitable certificates. If no suitable certificate is found, SYS600 Control Panel will create and install self-signed certificate named with the full computer name that includes the domain information. Certificate will include this name as web server hostname. Certificate with private key is installed into the Personal store under the Local Computer account and without private key into Trusted Root Certificate Authorities store. This certificate will be automatically selected in the Web Server Certificate field of the SYS600 Control Panel.

To make the SYS600 self-signed certificate trusted in the Workplace X clients, it needs to be installed to the client computer. To do this, export the certificate (.CER) from SYS600 computer Trusted Root Certificate Authorities store and copy it to the client computer and install it to the Trusted Root Certificate Authorities store.

Do not export the certificate with the private key from the Personal store. The certificate with private key should be kept safe and with limited access. Only the certificate without private key (.CER) shall be distributed to client computers.

Some browsers require specific settings to trust the certificates installed to the Windows Trusted Root Certificate Authorities store. For example, Firefox requires that security.enterprise_roots.enabled setting is set to true.

In case new self-signed certificate needs to be created (for example, computer name is changed and web server hostname needs to be updated), this can be done from the SYS600 Control Panel by clicking the "Generate SYS600 Default Certificate...". This will replace the old certificate with new one.

Microsoft limits a NetBIOS name to be 15 characters in length. Using computer names longer than 15 characters is not supported by SYS600 self-signed certificates.

4.4.4.2 Using Certificates from Customer or 3rd Party Certificate Authority

Certificate shall be signed by a trusted Certificate Authority and contain a fully qualified domain name (e.g. SYS600 computer name) known by the clients accessing the server.

Certificates must contain a private key without a passphrase and have "Server Authentication" purpose explicitly enabled.

Requesting a certificate
In a case that a new certificate is needed a certificate request must be generated and sent to a Certificate Authority. Certificate Authorities provide their own instructions how certificate request should be provided. A Certificate Request can be generated with a MMC-tool provided with the Windows Operating System.

Run MMC and add Certificates -snap-in into the console:

1. Run mmc.exe from <system drive>:\Windows\System32 folder as **Administrator**
2. Click **File > Add/Remove Snap-in...**
3. Select **Certificates** and click **Add >** button and a new dialog should appear
4. Select **Computer account** and click **Next >** button
5. Select **Local computer: (the computer this console is running on)** and click **Finish**
6. Click **OK** in the main dialog

Prepare and send a Certificate Request:

1. Browse into a **Personal > Certificates** folder in MMC
2. Right click onto middle panel and select **All Tasks > Advanced Operations > Create Custom Request...** and click **Next** in the new dialog
3. Select the option **Proceed without enrollment policy** and click **Next**
4. Leave **Template** and **Request format** to their default values if not otherwise instructed by a Certificate Authority and click **Next**
5. Click **Details** of the **Custom request** and then click **Properties**
6. Open the **Extensions** tab and **Extended Key Usage (application policies)** section. Select **Server Authentication** option and click **Add >** button to add it as Selected option
7. Open the **Private Key** tab, click **Key Options** and set **Key size** to 2048 and check **Make private key exportable** checkbox and click **OK**
8. Fill out all remaining details according to instructions provided by the Certificate Authority and click **OK**
9. Click **Next**
10. Select **File format** according the instructions provided by the Certificate Authority and select a name and a location for the file to be saved and click **Finish**
11. Provide the file created to the Certificate Authority according instructions provided

Installing certificate

1. Open up Windows Explorer and double click on the acquired certificate and click on the **Install Certificate...** and an import wizard should pop up
2. Select **Local Machine** as location and click **Next** button
3. Select **Place all certificates in the following store** and click **Browse**. Select **Personal** folder
4. Click **Next** button on the wizard and complete the installation by clicking **Finish** button

Installed valid certificate appears in SYS600 Control Panel Web Server Certificate dropdown list. Select the certificate to be used by the web server from the list.

4.5 Configuring process communication

The configuration of process communication is divided into the following configuration tasks:

- Configuring communication system objects in base system
- Configuring process communication units (PC-NET, External OPC DA Client, Modbus slave, CPI applications)

In general, the process communication units cannot be created or accessed from any application before the corresponding system objects in base system has been configured. The

definition method of these system objects depends on used process communication unit type.

The required base system object types are:

- LINn:B, which defines a communication route from the base system to the defined node.
- NODn:B, which in this context defines the node of process communication unit itself.
- STAn:B, which defines a station within a defined node.
- PRIn:B, which defines a printer within a defined node.

If the used process communication unit is of type PC-NET and System Configuration Tool is used, these base system objects are created automatically. If the System Configuration Tool is not used, or the used process communication units are other than PC-NET, see Section 4.5.2.2 and Section 4.5.2.6 or to PC-NET start-up with SCIL commands in Section 4.5.2.1. System Configuration Tool supports only process communication units of type PC-NET.

These objects can be also be configured using SYS_BASCON.COM or with SCIL using statement #CREATE.

4.5.1 Configuring communication system objects in base system

The following system objects are needed for each process communication unit:

4.5.1.1 Links (LIN)

A link is a data transmission line to another base system, a NET unit or a device. Each link is defined by a LINn:B object (n = 1 ... 20). A base system can have the following links:

- One link of type LAN. The process communication unit may be directly connected through LAN link. The LAN link is used between base systems and the process communication unit may be connected to another base system (indirect connection).
- One link of type INTEGRATED for each configured PC-NET. This link type is used only by PC-NET process communication unit and it is created by the System Configuration Tool. The PC-NET.EXE process is started when the LIN:B object of type INTEGRATED is created.

4.5.1.2 Node (NOD)

Nodes are directly or indirectly connected base systems and process communication units. The nodes are defined by NODn:B objects (n = 1 ... 250). A node definition is needed for:

- Communication to the communication units. Each process communication unit which is recognized by the base system must be defined as a node. These node definitions are described in SYS600 System Objects manual.
- Reading and writing attributes. A node is primarily specified by the used connection link and the station address of the node. If a node is only indirectly connected to the base system, the link to the node is the link to the nearest intermediate node. The link object must have been defined before the node can be defined.

Node definition is also used to define another base system in a network.

4.5.1.3 Example

A process communication unit uses LAN link and is configured to be node 3 and its address is 203. The first step in the configuration is to create a LIN:B object of type LAN (if not yet created for the base system communication). In this situation, the link number can be selected. The next step is to create NOD3:B object for the process communication unit and

assign selected link number to the NOD3:BLI attribute of the created object. The node address 203 is assigned to the attribute NOD3:BSA.

This configuration must be present before the process communication unit in node 3 can be accessed. For more information on system objects of type NOD and LIN, see SYS600 System Objects. The process communication unit is PC-NET and System Configuration Tool is used, these base system objects are created automatically, otherwise the NOD and LIN need to be created with SCIL. The example above would then require the following lines:

```
#CREATE LIN1:B = LIST(LT = "LAN", TR = "TCPIP")
#CREATE NOD3:B = LIST(LI = 1, SA = 203)
```

4.5.2 Configuring process communication units

The process communication unit types are listed below:

- PC-NET
- External OPC DA Client
- Modbus slave
- Other CPI-connected applications

Most of the communication protocols are supported by the PC-NET process communication unit. For more information on attribute PO and a complete list of protocols, see SYS600 System Objects. Communication unit of type External OPC DA client is used with OPC connected protocols such as IEC61850. With External OPC DA client, Modbus slave and other CPI-connected applications not part of SYS600 product, LAN link is used as the communication route between base system and the communication unit.

CDC-II slave is deprecated from version 10.0 since it used special hardware not supported in latest Windows versions and modern computers.

4.5.2.1 Configuring PC-NET

The recommended way to configure process communication unit of type PC-NET is to use System Configuration Tool. While creating the full configuration, it provides a set of possible selections in each step. In practice, these selections are mainly protocol specific line type and station types. It is also possible not to use the System Configuration Tool and create the line and station configuration using SCIL. The protocol specific manuals contain examples of how this is done with each protocol. This method is often used when a MicroSCADA system is updated to a newer release and the amount of changes to the system is tried to minimize.

When the PC-NET program is started, it reads the initial configuration file PC_NET.CF1, which is a text file located in the \sc\sys\active\sys_ directory. It defines the basic communication nodes and addresses to enable the communication to an application that downloads the total configuration.

When a PC-NET configuration is created with the System Configuration Tool, the tool produces two data files: SYSCONF.INI and SIGNALS.INI. When the system is started, it reads the mentioned files and creates a file PC_NET.CF1 automatically. To create system objects, the System Configuration Tool automatically creates the file SYS_BASE.SCL, which is executed at system start-up. After the PC-NET has started, the system executes the file SYS_NET.SCL to configure the PC-NET. The file is automatically created by the System Configuration Tool. A step-by-step description of the System Configuration Tool operation is described in Section, PC-NET Startup with System Configuration Tool. This information is rarely needed, and in practice, the system configuration can be entered and controlled without knowledge of the internal operation of the tool.

In case the NET node number is bigger than 50, the default addresses of the MI attributes may overlap with some other NET nodes and the default values may have to be changed. See the MI attribute description in SYS600 System Objects manual for details.

Start-up definition file PC_NET.CF1
When PC-NET process starts, it always reads the start-up configuration file PC_NET.CF1. This file is generated automatically by the System Configuration Tool. If the configuration is loaded with SCIL, it may be necessary to edit this file. The following PC_NET.CF1 file is included in the MicroSCADA X Control System SYS600 delivery as a default configuration:

```
local_node.sa=203 ; The station address of the PC-NET.
local_node.nn=3 ; The node number of the PC-NET.
ext_node(1).sa=209 ; The station address of the base system.
ext_node(1).nn=9 ; The node number of the base system.
ext_apl(1).nn=9 ; The node number of the base system.
ext_apl(1).an=1 ; An application in the base system.
```

In case the PC_NET.CF1 file is missing when the PC-NET is started, a default configuration becomes valid.

PC-NET start-up with System Configuration Tool
The information given in this section describes the internal operation of the System Configuration Tool. Usually, this is transparent for the user.

The System Configuration Tool creates procedures for automatic start-up and configuration of the PC-NET. The automatic starting/configuration can be switched on or off. Manual starting/stopping of the PC-NET can be done in online mode. The automatic starting and configuration of the PC-NET works in the following way:

- A command procedure SYS_INIT_1:C is connected to the event channel APL_INIT_1:A as the first secondary object. If the list of the secondary objects is full, the last one is removed and a warning is generated (notify window, log file).
- The command procedure SYS_INIT_1:C calls a text file (StartPC-NET.SCL) which starts the PC-NET. The program in the text file first updates the sys_/PC_NET.CF1 file and then starts the PC-NET by setting the corresponding base system link object type to INTEGRATED. The PC_NET.CF1 file is updated in the following way:

The PC-NET sends a system message to the own application when it is started. This message is received by a process object to which an event channel, SYS_NET'net_number'D:A, is connected. This event channel calls a command procedure SYS_NET'net_number'D:C. If the process object exists (for example created by LIB5xx) and has an event channel connected to it, all the objects connected to that event channel is moved to the SYS_NET'net_number'D:A event channel as secondary objects. In other cases, the System Configuration Tool automatically creates a process object SYS_NETD:P('net_number'), to which the event channel SYS_NET'net_number'D:A is connected.

The command procedure SYS_NET'net_number'D:C checks the message coming from the PC-NET. If this is the start message (10001), the PC-NET is configured according to the information entered in the System Configuration Tool.

If the system configuration contains many PC-NETs, then for each of the PC-NET processes to be started by System Configuration Tool, an instance specific copy of PC_NET.CF1 file can be found from the same folder. These files follow the file naming convention DEBUG'net_instance_number'.CF1, for example, file name DEBUG1.CF1 is used for the first PC-NET process instance.

All the possible error messages that occur during the start-up or configuration of the PC-NET are shown in the notify window. They are logged into the SYS_LOG.CSV log file, which can be viewed using some text editor of Windows operating system.

PC-NET start-up with SCIL commands
A command procedure for online reconfiguration of PC-NET can be started as follows:

- When a PC-NET unit is restarted, it sends the system message 10001 to all the defined applications (by default to process object address 6000 + NET no), provided that the application is running. The system message can be used for updating a process object which activates an event channel, which in turn starts a command procedure with reconfiguration commands. For more information describing the System messages from PC-NET units, see Section 4.5.2.1.7 and Section 4.5.2.1.8.
- When the connection between PC-NET and an application recovers after a break, PC-NET sends the system message 1000 + APL no. to the application (by default to address 6050 + NET no.). This message can be used for conditional start of reconfiguration procedures, that is, reconfiguration takes place if PC-NET has been restarted, not if the application has been out of use. This can be checked for example by reading a system object attribute configured online. If online configuration changes are valid, PC-NET has not been out of operation.

Reconfiguration commands could also, for example, be included in the command procedures started by the event channels APL_INIT_1 and APL_INIT_2, (APL_INIT_H in hot stand-by systems, see SYS600 Application Objects). However, a PC-NET unit can be restarted even though the application is not restarted.

The protocol specific manuals contains examples for the configuration script for each protocol. In principle, following step are needed for every protocol:

1. Define the NET line to be used by assigning it the wanted protocol.
2. Give the line its communication properties by means of the line attributes.
3. Create the station(s) by giving it an object number and assigning it the line number.
4. Set the attributes of the created object.
5. Take the line and the device into use.

In SCIL, communication system objects are created and deleted using NET attributes, see SYS600 System Objects. When adding a device, the NET line should first be defined. NET lines are defined by the NET line attribute PO. The used hardware device is generally defined with attribute SD, which, for example, may refer to certain serial port, PCLTA card or system device.

Defining external nodes (NET)
All the connected base systems and communication units are defined as external nodes (NET objects). This applies also to base systems and communication units, which are only indirectly connected via other communication units.

The primary external node, that is, the PC-NET that communicates via integrated link is defined in PC_NET.CF1 file and the corresponding attribute values are updated. If there is a need to define other nodes, the configuration of NET node attribute NE need to be configured.

Defining applications (APL)
As a rule, all the applications in all base systems, which are directly or indirectly connected to the communication unit, should be defined to the NET unit as APL objects. The defined applications can be configured to receive spontaneous messages from the stations and system messages generated by NET.

In order to define or redefine an application in a connected base system:

1. Define Application, an APL object, in the preconfiguration or online by means of the NETn:SSY attribute. For more information, see SYS600 System Objects.
2. Assign it to the following attributes for the communication supervision. For more information, see SYS600 System Objects.

From SCIL, the SW and SU attributes are accessed as NETn:S attributes. NET supervises all its application connections by cyclically reading the DS attribute of all known applications at the interval calculated from the SU attribute. If an application does not reply, an error message is

produced and the application is suspended. This happens when the base system is closed, when the application has been set to COLD, the application does not exist, or the connection is faulty or disturbed, or the communication does not work. When an application has been suspended, the RTUs connected to that application are not polled until the communication with the application has been re-established.

If the defined application is not running in a base system directly connected to the PC-NET but is running in another node, the NOD:B and LIN:B objects should define the route to the destination node. This route is usually a LAN link, but this may also be a serial line ACP.

Online configuration changes
The online configuration changes can be done in the online mode of the System Configuration Tool, with SCIL from a Test Dialog or from a command procedure.

The online changes take effect immediately. However, if the PC-NET unit is stopped and restarted, the online changes are lost and the preconfiguration is restored. Online changes which need to be permanent, and are not made in the preconfiguration should, therefore, be included in a command procedure which is executed each time the PC-NET unit is restarted.

When SCIL is used, the attributes are accessed with the object notation according to the format:

OBJnn:Sati

Table 5: Object notation table

'nn'	Object number (device number)
'at'	Attribute name
'i'	The possible index

OBJ in this context may be STA referring to a station object or PRI referring to a printer object. For more detailed information about the object notation, see SYS600 System Objects.

The attributes are written with the #SET command according to the format:

```
#SET OBJnn:SATi = value
```

The line attributes can be changed with the SCIL command #SET:

```
#SET NETnn:Sati = value
'i' Line number
```

For detailed information on each attribute, see SYS600 System Objects or protocol specific configuration manuals.

System messages - base system configuration
To use the system messages in an application, follow the instructions given below:

1. Create a fictitious process object of type ANSI analog input and set the Unit Number (UN) attribute to 0. The system message codes of the device is registered as the object value of this object.
2. Set the objects Object Address (OA) attribute equal to the Message Identification (MI) attribute and set the Switch State (SS) attribute to Auto.
3. Select direct scale (1-1).

Define the consequential operations by using the event, alarm and printout attributes. For more information on alarm generation, activation of event channel and automatic updating in pictures, for printout activation and for including event history in the event list, see SYS600 Application Objects.

The default values of MI attributes for each station type are presented in the System Objects manual and in the protocol specific manuals. The defaults listed below are protocol independent:

Table 6: Message Identification (MI) default value table

Object	Message	MI default value
NET itself	General messages, for example start-up messages Value: status code	6000 + NET no.
	Application supervision Value: APL no.= failure 1000 + APL no. = recovery (APL no. as known to NET)	6050 + NET no.
NET line	All NET line messages	6000 + 100 NET no. + line no.

System messages - PC-NET configuration
When a system message is caused by a system object, it is directed to the application specified by the Message Application (MS) attribute of the object. The code of the message is updated as the object value for a fictitious process object with the Object Address (OA) attribute value equal to the value of the Message Identification (MI) attribute. In case the NET node number is bigger than 50, the default addresses of the MI attributes may overlap with some other NET nodes and the default values may have to be changed. See the MI attribute description in SYS600 System Objects manual for details.

To achieve system message handling in the communication unit:

1. Set the Message Application (MS) attribute of the system object to the number of the receiving application.
2. Set the Message Identification (MI) attribute of the system object to the value of object address of the receiving object. The MI attribute has object dependent default values which are also the recommended values, and should generally not be changed. The default value is used when the system object is defined online and the MI attribute is not explicitly set, or if the MI attribute is set to 0 in the preconfiguration. The default values are shown in the Table 6 above, in the System Objects manual and protocol specific manuals.

The transmission of system messages from individual objects can be enabled or disabled by the System Message Enabled (SE) attribute of the objects. The system message generation should only be disabled in special cases, for example, if the base system application program often executes commands, which cause unwanted system messages.

The SE attribute exists for the PC-NET Node and it is accessed by NETx:SSE. This attribute controls the transmission of the system messages from all objects configured to the node. This attribute can also be used to enable the updating of the binary status points.

4.5.2.2 Configuring IEC 61850 with External OPC DA Client

External OPC DA Client and IEC 61850 OPC Server (=IEC 61850 Client) are included as a separate executables in the SYS600 product. External OPC DA Client is communicating to SYS600 base system via LAN link and acting as one communication node in a MicroSCADA network. In other words, the related base system objects has to be created in the file and the OPC namespace of the IEC 61850 OPC Server has to be mapped into process objects. The required station type for units is SPA. In case of IEC 61850 OPC server, the .ini file containing the mapping from OPC server items to process objects is created by SCL Importer tool or IET Data Loader. With other OPC servers, the mapping between OPC items and process objects can be done using External OPC DA Client Configuration Tool.

The creation of the base system nodes and station objects are supported in System Configuration Tool, see Section 5.4.7.

For description of the different topologies and configuration details of this IEC 61850 communication, see SYS600 IEC 61850 System Design. Otherwise, the configuration of External OPC DA Client has been described in SYS600 External OPC Data Access Client and IEC 61850 OPC Server in SYS600 IEC 61850 Master Protocol (OPC).

4.5.2.3 Configuring Modbus slave

Modbus slave protocol is also supported by a separate executable which is connected to base system via LAN link. The configuration of the base system objects described in Section 4.5.1 are needed before the communication between base system and the modbus slave executable is established.

For description of the configuration details of this process communication unit type, see SYS600 Modbus Slave Protocol.

4.5.2.4 Configuring IEC 61850 Server

IEC 61850 Server is a separate executable which is connected to base system via LAN link. The configuration of the base system objects described in Section 4.5.1 are needed before the communication between base system and IEC 61850 Server executable is established.

For description of the configuration details of this process communication unit type, see SYS600 IEC 61850 Server manual.

4.5.2.5 Configuring IEC 60870-6 (ICCP)

ICCP is a separate executable which is connected to base system via LAN link. The configuration of the base system objects described in Section 4.5.1 are needed before the communication between base system and ICCP executable is established. This part of the configuration is similar whether the ICCP operates as client or server.

The creation of the base system nodes and station objects are supported in System Configuration Tool, see Section 5.4.9.

For description of the configuration details of this process communication unit type, see SYS600 IEC 60870-6 (ICCP) Protocol manual.

4.5.2.6 Configuring other CPI-connected applications

CPI (Communication Programming Interface) is a software library for connecting an application to the MicroSCADA base system. Each application instance using CPI as an interface is seen as node in SYS600 network and corresponding NODx:B and LINx:B need to be created.

For a description on how the CPI interface is used and how the application instance is seen from the base system, see SYS600 Communication Gateway, COM500*i* User's Guide.

4.5.2.7 Selected configuration examples for PC-NET

Base system network using serial ACP
For performance reasons, there should generally not be more than three communication units in a series between a base system and a communicating device. These are base system, workplace, printer or RTU.

When communication units and base systems are connected to a network, each NET unit and each base system in the network should be defined as a node in each others' NET unit and base system.

In order to connect two communication units through serial lines, make the following definitions in each of the unit:

1. Select a line for the connection and define it with the ACP protocol as follows:

PO	1
MS	System message application
MI	System message object address
BR	Baud rate
PY	Parity
SB	Number of stop bits
RD	Read data bits
TD	Transmission data bits
ER	Embedded response
RE	Redundancy
TI	Timeout length in seconds (1 + 2400/BR)
NA	NAK limit
EN	ENQ limit
PS	Buffer pool size

   The communication attributes (BR, and so on) should have the same values as the corresponding parameters in the connected communication unit. If the NET is a PC-NET, the line numbers 1...4 are available. These lines corresponds to the COM ports. When selecting one of these lines for the ACP protocol (setting the PO attribute of the line to 1), the line number cannot be used for any of the LON channels.

2. Define an External node, a NET object, on the ACP line for the connected communication unit:

Device type	NOD
Device number	The node number of the connected communication unit
LI, Line number	The number of the selected line
SA	Station address of the connected communication unit

   Though two communication units are connected indirectly via another unit, they should be defined to each other. Make the following definitions in each of the units.

3. Define an External node (NET object) connected to the line to the nearest communication unit:

Device type	OD
Device number	The node number of the indirectly connected communication unit
LI, Line number	The line to the nearest NET unit in the series
SA	Station address of the indirectly connected communication unit

Each NET unit connected to a base system via one or more other units should be defined to the base system as a node (NODn:B objects):

1. Create a NODn:B base system object corresponding to the indirectly connected communication unit. The NOD object number ('n') should be the same as the node number of the communication unit. The NOD object is given the following attribute values:

LI	Link number (= LIN object number)
	This is the link to the nearest communication unit
SA	Station address of the indirectly connected communication units

   Even if there is no communication between the base system and the indirectly connected NET, the node definition is necessary for the system diagnostics, online configuration and system maintenance.

2. Define an External node (NET object) on the line to the nearest communication unit:

Device type	NOD
Device number	The node number of the indirectly connected base system
LI, Line number	The line to the nearest communication unit in the series
SA	Station address of the indirectly connected base system

3. Define an application for each application in the indirectly connected base system. Figure 22 shows an example of a network of two communicating NETs and two base systems. The table below shows the configuration of the NETs and base systems. The example includes only the definitions which are of importance for this particular configuration, and that have not been described in the previous sections.

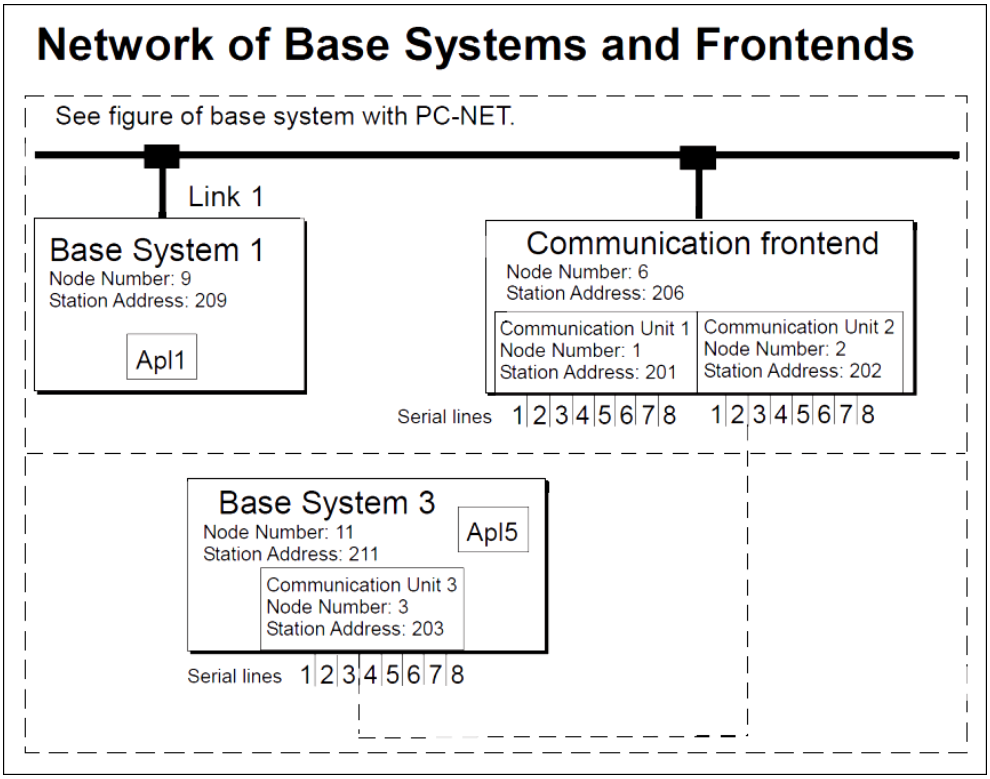


Figure 22: Example of a configuration with interconnected base systems and NETs

Configuration of Communication unit 1		
See Figure 22.		
External node 9 (Base system 1)		
	Device type:	NOD
	Device number:	9
LI	Line number:	13
IU	In use:	1
SA	Station addr. (Dec.):	209
Application 1		
	Device type:	APL
	Device number:	1
	Translated APL number:	1
	Node number:	9
IU	In use:	1
SW	Reply timeout:	5
SU	Suspension time:	60
Configuration of Communication unit 2		
See Figure 22.		
Line 2 (ACP line)		
PO	Protocol	1
IU	In use:	1
MS	Message application:	1
MI	Message ident:	6202
LT	Link type:	1
BR	Baud rate:	9600
SB	Stop bits:	1
PY	Parity:	2
RD	Receiver data bits:	8
TD	Transm. data bits:	8
RE	Redundancy:	2
TI	Timeout length:	3
NA	NAK limit:	3
EN	ENQ limit:	3
ER	Embedded response:	1
RP	Reply poll count:	1
PS	Buffer pool size:	30
External node 3 (Communication unit 3)		
	Device type:	NOD
	Device number:	3
Table continues on next page		

Configuration of Communication unit 1		
LI	Line number:	2
IU	In use:	1
SA	Station addr. (Dec.):	203
External node 9 (Base system 1)		
	Device type:	NOD
	Device number:	9
LI	Line number:	13
IU	In use:	1
SA	Station addr. (Dec.):	209
External node 11 (Base system 3)		
	Device type:	NOD
	Device number:	11
LI	Line number:	2
IU	In use:	1
SA	Station addr. (Dec.):	211
Application 1		
	Device type:	APL
	Device number:	1
	Translated APL number:	1
	Node number:	9
IU	In use:	1
SW	Reply timeout:	5
SU	Suspension time:	60
Application 5		
	Device type:	APL
	Device number:	5
	Translated APL number:	5
	Node number:	11
IU	In use:	1
SW	Reply timeout:	5
SU	Suspension time:	60
Configuration of Communication unit 3		
See Figure 22.		
Line 3 (ACP line)		
PO	Protocol:	1
IU	In use:	1
MS	Message application:	5
MI	Message ident.:	303

Table continues on next page

Configuration of Communication unit 1		
LT	Link type:	1
BR	Baud rate:	9600
SB	Stop bits:	1
PY	Parity:	2
RD	Receiver data bits:	8
TD	Transm. data bits:	8
RE	Redundancy:	2
TI	Timeout length:	3
NA	NAK limit:	3
EN	ENQ limit:	3
ER	Embedded response:	1
RP	Reply poll count:	1
PS	Buffer pool size:	30
External node 2 (Communication unit 2)		
	Device type:	NOD
	Device number:	2
LI	Line number:	3
IU	In use:	1
SA	Station addr. (Dec.):	202
External node 9 (Base System 1)		
	Device type:	NOD
	Device number:	9
LI	Line number:	13
IU	In use:	1
SA	Station addr. (Dec.):	209
Application 1		
	Device type:	APL
	Device number:	1
	Translated APL number:	1
	Node number:	9
IU	In use:	1
SW	Reply timeout:	5
SU	Suspension time:	60
Application 5		
	Device type:	APL
	Device number:	5
	Translated APL number:	5
	Node number:	11

Table continues on next page

Configuration of Communication unit 1		
IU	In use:	1
SW	Reply timeout:	5
SU	Suspension time:	60

Configuring stations using RP570 master protocol

Base system configuration:

In order to connect SYS600 network to RTUs with RP570 protocol, the following definitions are required in the base system, which uses the station:

1. Create a STAn:B object with the following attributes:

ND	The node number of the NET unit to which the RTU directly connected.
ST	RTU
TN	Corresponding STA object number in the communication unit.
TT	EXTERNAL

   For more information on the attributes, refer to the SYS600 System Objects manual.
   The STAn:B object definition is not necessary if the default station type defined by SYS:BDS is RTU and the default node defined by SYS:BDN is the NET unit to which the RTU is connected and the mapping is direct.
   However, if no STAn:B object is defined, the station cannot be handled by the MicroSCADA X tool pictures.

2. If needed, map the station for the application which uses it with the APLn:BST attribute. Station mapping is necessary only if the logical number is other than the STAn:B object number, which is the default mapping.
   The logical station number is the Unit Number (the UN attribute) of the process objects defined for the station. For more information, refer to the System Objects manual.

PC-NET configuration:

Perform the configuration definitions described below in the NET unit to which the station is directly connected. It is assumed that the NET unit has been defined to the base system as a NODn:B object, and that the base system has been defined to the NET unit as an external node.

1. Select a line for the station (several RTUs can be connected to the same line) and define it with the RP 570 protocol:

PO	7
LT	0 (RS232) or 1 (modem line)
IU	1
MS	The application receiving system messages
MI	The object receiving system messages
BR	Baud rate, should be the same as in the RTU
PY	2
RD	8
TD	8
SB	1

Table continues on next page

PS	Buffer pool size
DE	CTS delay in milliseconds
EN	Enquiry limit time in milliseconds
PD	Poll delay in milliseconds
PP	Polling of suspended stations
RP	Number of consecutive polls
TI	Timeout length in seconds
HT	Timeout in milliseconds for start of response reception (default = 700 ms)
RI	Time delay in milliseconds before enabling a line after a message. Default = 0. A time delay should be used if NET's transmission echoes back into the receiver.
RK	RTS keep up padding characters, refer to the SYS600 System Objects manual.

2. Make sure that the application which receives spontaneous messages from the station (the station attribute AS) is defined as an APL object.
3. Define a station of type RTU connected to the RP 570 line:

Device type	4
LI	Selected line number
AL	1
AS	The number of the connected application
MS	The application receiving system message
MI	The object receiving system messages
SA	RP570 station address (= the address in the RTU)
RT	Reply timeout in seconds

   If several stations are connected to the same line, define the stations with the same line number (LI).
   The NET unit recognizes an automatically created station, STA0, as broadcast station. The broadcast station notates all S.P.I.D.E.R. RTUs connected to the same NET.

4. Synchronize the RTU200 clock with the clock of the NET unit at start-up by setting the SY attribute, for example, #SET STAn:SSY (supposing that the NET clock has been synchronized before). By using the broadcast station number, all RTUs connected to one NET can be synchronized simultaneously.
5. If needed, change the AW attribute of the RP 570 line (refer to the System Objects manual). This is normally not necessary.

A SYS600 revision beginning with 8.2B supports the configuration of hierarchical RTU structures. Define the sub-RTUs as STA objects in NET and in the base system, in the same way as an RTU connected directly to NET as described in the manual. The only difference between the directly connected RTUs and sub-RTUs is the STAn:SHR attribute, refer to the System Objects manual. For STA objects corresponding to sub-RTUs, the HR attribute is the station address of the RTU one level above in the hierarchy.

The data of ERMFD and ERMIR telegrams is converted into bit stream values, which are sent to the process database.

In order to register the data in the process database, define bit stream type process objects with the following object addresses:

For ERMFD : 2304 + block nr

For ERMIR : 1792 + block nr

ERMFD data coding in process object

Bit stream object value:

bytes 1..4:	VALUE (least significant byte first)
byte 5:	STATUS with time quality and so on, copied from RP 570 telegram
bytes 6..7:	RELATIVE TIME (least significant byte first)
bytes 8..9:	NUMBER (least significant byte first)
byte 10:	CAUSE OF TRANSMISSION
byte 11:	FORMAT
Registration time: stored in RT attribute as normal	

ERMIR data coding in process object

Bit stream object value:

byte 1:	VALUE (least significant byte first)
byte 2:	BIT NUMBER
byte 3:	INDICATION TYPE
byte 4:	STATUS with time quality and so on, copied from RP 570 telegram
bytes 5..6:	RELATIVE TIME (least significant byte first)
bytes 7..8:	NUMBER (least significant byte first)
byte 9:	CAUSE OF TRANSMISSION
Registration time: stored in RT attribute as normal	

The coding of each field, when not explicitly described above, follows the RP 570 telegram.

There are two methods of building an RTU configuration. The RTU configuration can be performed independently of SYS600, which means, the SYS600 process object definition is built separately with no help from the RTU configuration files. Alternatively, the RTU configuration can be built via SYS600, which means, the SYS600 engineer can use the configuration in the process object definitions. Changes in the SYS600 process database can then be loaded down to the RTUs.

RTU configuration:

See SYS600 System configuration manual of MicroSCADA 8.2 for complete process of RTU Configuration and the configuration of the process objects to the MicroSCADA application.

The mentioned manual refers to tools which are not necessarily supported by the MicroSCADA X product family.

Configuring stations using ANSI X3.28 protocol
Base system configuration:

Connecting a station using the ANSI X3.28 protocol (ANSI station) to the SYS600 network requires the following definitions in the base system which uses the station:

Create a STAn:B (n = 1 ... 2047) object with the following attributes:

ND	The node number of the NET unit to which the station is directly connected.
ST	STA
TN	STA object number in the NET unit.
TT	EXTERNAL

The STAn:B object definition is not necessary if the default station type, defined by SYS:BDS, is STA and the default node, defined by SYS:BDN, is the NET unit to which the station is connected to and if the mapping is direct.

If needed, map the station to the application which uses it with the APLn:BST attribute. Station mapping is necessary only if the logical number is other than the STAn:B object number, which is the default mapping. The logical station number is the Unit Number (the UN attribute) of the process objects defined for the station (refer to the System Objects manual).

Configuration for SRIO device:

SRIO system parameters

By changing the SRIO 1000M system parameter values, the application programmer can affect general features of the SRIO 1000M program. The system parameters are located in the address area from 3000 upwards.

Table 7 shows some examples of system parameters, each of which occupies one word. For further information about SRIO system parameters, refer to the SRIO manuals.

Table 7: Examples of systems parameters

Word 0 (address 3001): Spontaneous event data transmission
1 = Enabled
0 = Disabled
Word 1 (address 3002): Spontaneous transmission of changed data in database
1 = Enabled
0 = Disabled
Word 2 (address 3003): Store command
1 = Start storing the configuration data into EEROM
0 = No meaning
Word 3 (address 3004): Analog data format
0 = 32 bit integer
1 = 3-digit BCD
2 = 6-digit BCD
Word 4 (address 3005): Analog data scaling
1, 10, 100, 1000 (default) or 10000
Word 5 (address 3006): Time polling interval
30 .. 30000 seconds (default = 60 s)

#SET STA1:SME3001=0

Disable spontaneous process data transmission.

#SET STA1:SME3002=1

Store SRIO 1000M configuration data into EEROM memory.

#SET STA1:SME3003=1

Analog values to be coded as 3-digit BCD numbers.

SRIO object parameters:

The SRIO object parameters allow the MicroSCADA X applications to read and write the definitions of data items, data groups and event data polling.

The start address of object parameters is 5000 in the default configuration. SRIO can contain up to 500 objects.

The following attributes are defined for each data item (start address refers to the start address within the object parameter area, that is add 5000 to each address):

Attributes	Start address
ANSI address	0
Bus number	500
SPACOM address	1500
Data type/format	4500
Delta value/bit mask (32 bits)	5500
Status word (16 bits)	6500

Example
A SCIL command procedure for the creation of an AI type SRIO object:

Defining variables

```
@OBJ_IND = Object nr (index) in the SRIO database
@ANSI_A = Object address in the MIcroSCADA database
@BUS = Bus number
@SPA_A = SPA address as a 6 word vector (see the SRIO manuals)
@DTYPE = Data type
@DFORM = Data format
@DELTA = Delta value
@STATUS = Status word as an integer
```

Defining constants

```
@OB_PAR_I = 5000 @ANSI_A_I = %OB_PAR_I
@BUS_I = %OB_PAR_I + 500
@SPA_A_I = %OB_PAR_I + 1500
@DATA_T_F_I = %OB_PAR_I + 4500
@DELTA_I = %OB_PAR_I + 5500
@STATUS_I = %OB_PAR_I + 6500
```

Creating object

```
#SET STA1:SME (%ANSI_A_I+%OBJ_IND) = %ANSI_A
#SET STA1:SME (%BUS_I+%OBJ_IND) = %BUS
@SPA_STADR = %SPA_A_I + 6 * %OBJ_IND
#SET STA1:SME (%SPA_STADR..(%SPA_STADR+5)) = %SPA_A
@D_T_F_ADR = %DATA_T_F_I + 2 * %OBJ_IND
@DATA_T_F(1) = %DTYPE @DATA_T_F(2) = %DFORM
#SET STA1:SME (%D_T_F_ADR..(%D_T_F_ADR + 1))=%DATA_T_F (1..2)
@DELTA_S_A = %DELTA_I + 2 * %OBJ_IND ;32-BIT ADDRESS
#SET STA1:SME (%DELTA_S_A) = %DELTA
#SET STA1:SME (%STATUS_I+%OBJ_IND) = %STATUS
```

A data group can consist of 10 data items, and there can be up to 100 data groups. The data group definition indicates the ordinal numbers of the data items in the group. The data group definitions are found from the address (7000 + object parameter area start address). Above is an example of a SCIL command procedure that defines a data group.

The event data polling can contain up to 300 SPA bus slave units (100 slaves/bus, in the address range starting from 8000 + object parameter area start address). The following features of each object to be event polled are defined:

- Bus number
- Unit number
- Unit type
- Status

Examples of creating a SRIO data group with SCIL

Defining variables

```
@GROUPNR = Number of the group to be created
@MEMBERS = Vector containing the ordinal numbers of the group members in
the SRIO 1000M database.
```

Defining constants

```
@GROUPDEFSA = 12000
@GROUPLEN = 10
```

Creating the data group

```
@MEMBCOUNT = LENGTH (%MEMBERS)
@STARTADR = %GROUPDEFSA + %GROUPNR * %GROUPLEN
@ENDADR = @STARTADR + %MEMBCOUNT - 1
#SET STA1:SME (%STARTADR..%ENDADR) = %MEMBERS
```

Auto-dialing in serial protocols
Auto-dialing can be used on all NET serial lines with the following protocols:

- ANSI X3.28 Half Duplex or Full Duplex protocols
- ACP (Application Communication Protocol)
- Modbus
- Alpha
- 61107
- IEC
- RP 570 master and slave
- SPA
- IEC 60870-5-101 master and slave
- IEC 60870-5-103 master
- DNP 3.0 master and slave

The example below describes the configuration using SCIL. The usage of the System Configuration Tool is recommended.

Auto-dialing is useful for the following reasons:

- For the connection of remote stations with infrequent data transfer
- For taking a reserve line into use

Auto-dialing is possible in both directions.

The auto-dialing line can be defined in pre-configuration. However, the auto-dialing feature cannot be pre-configured as it needs to be configured and taken into use online.

Create the line in pre-configuration or online. Depending on the device(s) connected to the line, set the Protocol (PO) attribute to 1 for the ANSI X3.28 Full Duplex protocol, 2 for the ANSI X3.28 Half Duplex protocol, 25 for Modbus RTU mode master protocol and 26 for the IEC 61107 protocol.

The auto-dialing feature for a line can be added by using a tool or SCIL. The dial-up modem has to be connected to the line while defining the auto-dialing feature.

To define the auto-dialing with SCIL:

1. Take the line out of use by setting the In Use (IU) attribute of the line to 0, for example:
   #SET NET1:SIU5 = 0
2. Set the ACE (AC) attribute of the line to 1, for example:
   #SET NET1:SAC5 = 1
3. If the NET unit is supposed to answer incoming calls which is always the case on RP 570 lines, set the Remote Calls Enabled (RC) attribute to 1, for example:
   #SET NET1:SRC5 = 1
4. If an automatic break of the connection is wanted after a specified time, set the Connection Time Limited (CL) and Connection Time (CT) attributes, for example:
   #SET NET1:SCL5 = 1
   #SET NET1:SCT5 = 500
   which means that the connection is broken automatically after 500 seconds.
5. If needed, set the Radio Disconnection Delay (DD), Pulse Dialling (PU), Radio Connection Wait Time (RW) and ACE AT S Register (SR) attributes. See SYS600 System Objects.
6. Set the In Use (IU) attribute of the line to 1, for example:
   #SET NET1:SIU5 = 1

To dial up a workplace or RTU from a NET:

Set the Connection (CN) attribute in an application program as follows:

#SET NETn:SCN'line' = "phone"

or when dialing a station:

#SET NETn:SCN'line' = "phoneSstation"

where

line	Line number
phone	Phone number of the receiver
station	Station number of the receiver

Dialing is done while the line is in use (IU = 1).

When the NET is dialing, system messages with codes 16107, 16208 or 16825 (depending on the protocol) are generated. If a station is dialing, the codes 16108, 16209 or 16826 are generated. A failed dial-up generates the code 16704.

The connection to an RTU is broken automatically under the following circumstances:

- RTU becomes inoperable
- RTU hangs up
- when the RTU is the dialling part
- RTU has nothing to send (after two subsequent CCR2 responses)

In addition to these, the connection can be broken automatically according to the Connection Time (CT) attribute. If the connection is not broken automatically, break it by setting the Connection (CN) attribute to 0:

#SET NETn:SCN'line' = 0

A succeeded hang-up generates a system message with code 16733. If the hang-up failed, the code 16702 or 16703 is generated. The status codes 16106, 16107 and 16810 indicate that disconnection has started.

Example

```
#SET NET1:SCN5 = "1234567"
```

Dialing station 11 (STA11):

```
#SET NET1:SCN5 = "1234567S11"
```

Breaking the connection:

```
#SET NET1:SCN5 = ""
```

4.5.2.8 Secure authentication using IEC/TS 62351-5

Secure authentication according to IEC/TS 62351-5 is supported with the following protocols:

- DNP3.0 master and slave
- IEC 60870-5-104 master and slave
- IEC 60870-5-101 master and slave (balanced mode only)

With DNP3.0, the implementation follows IEEE1815-2010 (V2 mode) and IEEE1815-2012 (V5 mode). With IEC60870-5-101/104, the implementation follows IEC/TS 60870-5-7 Ed 1.0. Protocols operating in TCP/IP mode, i.e. DNP3.0 and IEC60870-5-104, the TLS (Transport Layer Security) defined by IEC/TS 62351-3 may be used together with secure authentication. For TLS configuration details see protocol specific manuals. This section describes the key storage creation only for application layer authentication i.e. IEC/TS 62351-5. It is recommended to read this section before the key handling process for system is planned.

A separate Authority Tool is needed to create an encrypted database for user sets and update keys for each station object using secure authentication. In SYS600 version 10 and later, this Authority Tool is delivered as part of the SYS600 package but it must explicitly installed before it can be used. With older versions, Authority Tool is delivered separately. Configuration steps 1a/1b below give instructions for the installation of the tool. The feature described in this section cannot be used without a key storage database created with this tool. An on-line help is provided with the Authority Tool. Authority Tool is used for configuring the application layer authentication only and not for the TLS defined in IEC/TS 62351-3.

The usage of secure authentication in the mentioned protocols protects the systems from unauthorized access and helps to reveal possible attacks. IEC/TS 62351 part 3 and part 5 describes the addressed threats in detail. It is assumed here that the reader of this section knows the principles and motives of IEC/TS 62351.

If the secure authentication feature is used, it is very important to keep all symmetric keys (Update keys, Authority Certification Key) in secret, otherwise the benefit of the usage of the feature is compromised. There are two alternative engineering workflows for the handling of the symmetric keys:

1. Authority Tool is used in external key handling computer

   This option is more complicated but also more safe since the secret keys are visible only in one place. This option also supports centralized key handling process better. See configuration steps 1a-5a and related flowchart picture below. This option is recommended if system is operated as Hot Stand-by.

2. Authority Tool is used in SYS600 computer

   This option is more simple but the secret keys are distributed to multiple locations. Persons who have access to the SYS600 computer and have password for the Authority Tool may open the key storage used by the system and see the currently active symmetric keys, including user specific Update Keys thought the negotiation had happened in asymmetric manner. When this option is used, it is strongly recommended that the Authority Certification Key is not organization wide but specific for the system in question only. This option is recommended if system is configured for compatibility tests. See configuration steps 1b-2b and related flowchart picture below.

Steps 6-13 of the engineering process describes the actual creation of the key storage. The key storages used by the PC-NET communication modules are encrypted in both alternatives with an encryption key is bound to the computer in question. The created keystorage cannot be opened with an Authority Tool in another computer. Thus, transferring the key storage file or taking a back-up is not a security risk itself. From the engineering of point view, the usage of secure authentication in asymmetric mode is easier since all the keys visible during engineering are public keys and those need not to be kept in secret.

Following pictures describe the main steps of the configuration of the secure authentication of both alternative. If possible, the testing of the communication channels and signal engineering separately without secure authentication is worth to be considered. The configuration of the key storage can be made concurrently with other testing activities. When other tests are completed, the secure authentication can be enabled and tested. Configuring the secure authentication should be started with the master and then moving to the slave configuration. If asymmetric mode is used, the master and slave configurations should be done in a parallel manner.

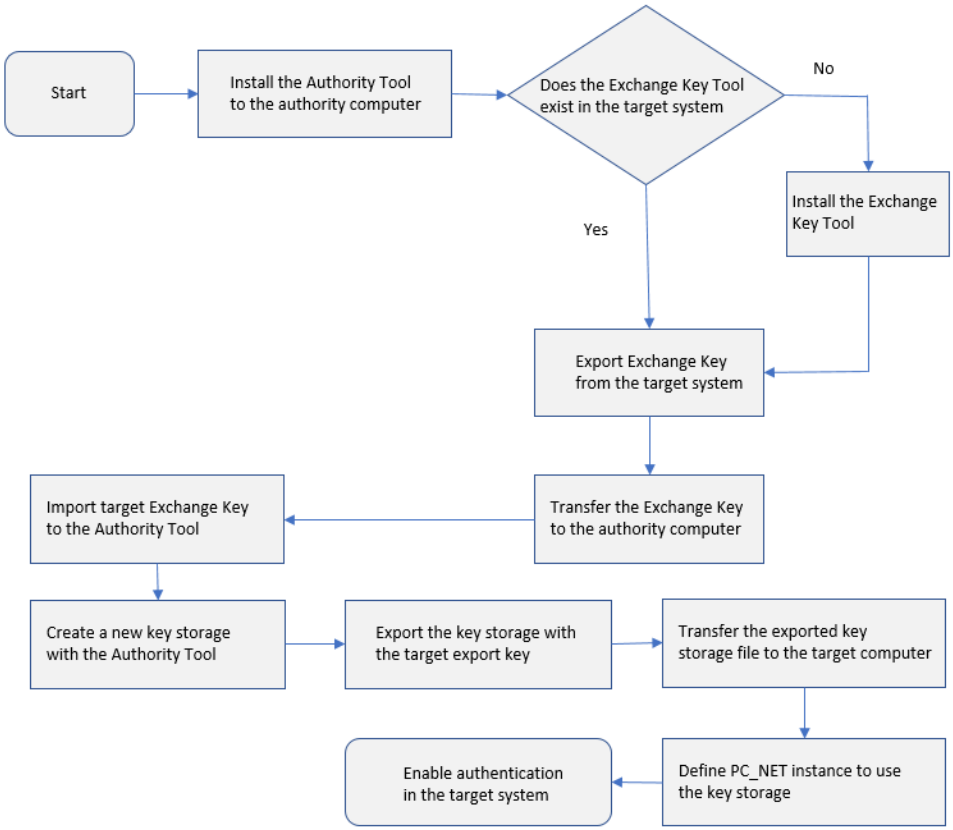


Figure 23: Secure authentication configuration using external computer (Steps 1a-5a)

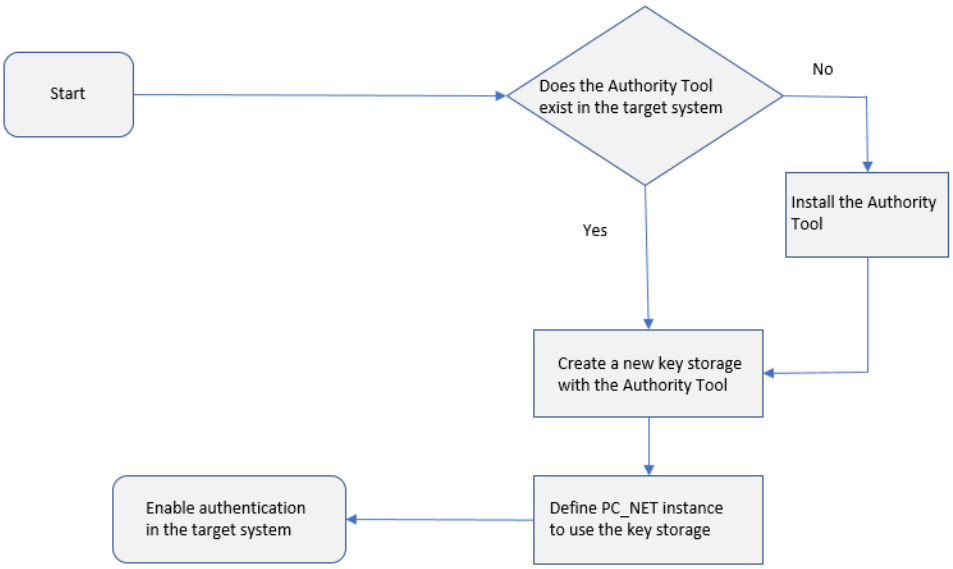


Figure 24: Secure authentication configuration using SYS600 computer (Steps 1b-2b)

In SYS600 version 9.4FP1 or newer, the supported algorithms in DNP3.0 are :

- SYMMETRIC_AES128_SHA1_HMAC (symmetric mode)
- SYMMETRIC_AES256_SHA256_HMAC (symmetric mode)
- ASYMMETRIC_RSA1024_DSA_SHA1_HMAC_SHA1(asymmetric mode)

In SYS600 version 10.0 or newer, the supported algorithms in IEC 60870-5-101/104 are :

- SYMMETRIC_AES256_SHA256_HMAC (symmetric mode)
- ASYMMETRIC_RSA2048_DSA_SHA256_HMAC_SHA256 (asymmetric mode)
- ASYMMETRIC_RSA3072_DSA_SHA256_HMAC_SHA256 (asymmetric mode)

The creation of the stations and user sets in Authority Tool does not separate the differences between protocols but provides all options. Please do not assign modes or algorithms which are not supported by the protocol in use.

In case some other algorithm is needed, contact the nearest Hitachi ABB Power Grids representative.

The principal sequence of the configuration is described below. Most of the steps are the same for versions V2 and V5 (DNP3.0) and for master and slave. Steps for IEC 60870-5-101/104 are similar to DNP3.0 V5, but if predefined Update Keys are used, the steps are similar to DNP3.0 V2. Steps 8 and 9 are different depending on the used authentication mode and the master/slave role of the station object in question. The naming of the fields follows IEEE1815 but if the system is connected to a system from a different vendor, the naming may also differ. If V5 authentication (DNP 3.0) of pre-defined Update Keys (IEC60870-5-101/104 is used and the station objects uses symmetric update key change mode, the instructions given for asymmetric mode can be ignored (and vice versa).

Authority Tool is used in external key handling computer:

1a. Install the Authority Tool to a separate computer in a safe place. Identify the persons who may have access to the key information kept in secrecy (Authority Certification key and Update keys). Copy the file \sc\stool\SysConf\Authoritytool.exe from SYS600 computer and install it to the key handling computer by starting the executable and following the installation instructions from Step 6 in section 'Section 4.5.2.8.4' below.

2a. Install the Exchange Key Tool to each computer running PC-NET communication module with secure authentication. The installation steps are described in the section 'Section 4.5.2.8.4' below.

3a. Export an Exchange Key using the Exchange Key Tool in the computer running PC-NET communication module with secure authentication. See the section 'Section 4.5.2.8.5' for the usage details of the Exchange Key Tool.

4a. Transfer the public Exchange Key file to the computer where Authority Tool is used.

5a. Import the Exchange Key file to the Authority Tool using menu bar (Key Management -> Import Exchange Key) and proceed to step 6 for key storage creation.

Authority Tool is used in SYS600 computer:

1b. Install the Authority Tool to SYS600 computer. The instructions for the Authority Tool installation are described in the section 'Section 4.5.2.8.4' below. Identify the persons who may have access to the key information kept in secrecy (Authority Certification key and Update keys).

2b. Proceed to step 6 for key storage creation.

The sequence of the keystorage configuration (steps 6 to 13) is described below. Most of the steps are the same for versions V2 and V5 (DNP3.0) and for master and slave. Steps for IEC 60870-5-101/104 are similar to DNP3.0 V5, but if predefined keys are used the steps are similar to DNP3.0 V2. With IEC 60870-5-101/104, it is possible to use the name "Default" (starting with capital letter) as default user = 1, this setting must be the same both in master and slave systems. Steps 8 and 9 are different depending on the used authentication mode and the master/slave role of the station object in question. The naming of the fields follows IEEE1815 but if the system is connected to a system from a different vendor, the naming may also differ. If V5 authentication is used and the station objects uses symmetric update key change mode, the instructions given for asymmetric mode can be ignored (and vice versa).

Key Storage creation:

6. From the system configuration, identify the station objects which will use the secure authentication.

7. With the Authority Tool create a key storage for the computer running the PC-NET communication module with secure authentication. The same key storage can be used by multiple PC-NET instances in a same computer but one PC-NET instance can use only one key storage. It is also possible to have a separate key storage for each PC-NET instance in the same computer.

8. Create necessary users, user sets and station object to the key storage. Assign user numbers and names as needed. The station names are freely selectable but their content must be the same in both master and slave (technically, matching not necessary with V2 but is recommended). The field name may also be called as 'outstation name' when connected to a third party system. Station identification should match the value of the ZT attribute of the corresponding station object in the System Configuration in SYS600.
If DNP3.0 secure authentication V2 is configured or IEC 60870-5-101/104 protocol with predefined keys is used, both master and slave key stores need to have the same exact users (names and roles).

   If DNP3.0 secure authentication V5 is configured or the used protocol is IEC 60870-5-101/104 and the station object is connected to a slave line, create a user set which contains no other users but "Common" (present as default, name defined in DNP3 standard). With IEC 60870-5-101/104, the default user = 1 can also be named as "Default". Define the created user set in the station creation. This is because in DNP3.0 secure authentication V5 and in IEC 60870-5-101/104, the users are created on-line using the protocol. When the COM500*i* system is connected to NCC, before the update keys are negotiated, the users and their roles for the STA object in COM500*i* are created as they are defined in the NCC.
   If DNP3.0 secure authentication V5 is configured or the used protocol is 60870-5-101/104 and the station object is connected to a master, a Role, Role Expiry Interval and Update Key Change Method must be given to each user. The given update key change method should follow the update key mode accepted by the slave system (symmetric/asymmetric). In most cases, the Update Key Change Method is the same for each user. See attributes ZO and CR in the protocol specific manuals for more information about the roles and required permissions for each operation.
   If the Update Key Change Method is set to 'SYMMETRIC_AES128_SHA1_HMAC' or 'ASYMMETRIC_RSA1024_DSA_SHA1_HMAC_SHA1' (possible only in DNP 3.0), the update key length may be set to 16 bytes, with other selections to 32 bytes. If the session key wrapping algorithm is AES-256, the update key length must always be 32 bytes. When the update key length is changed for users in station level and a red mark is visible in the Update Key field, selecting the users from 'Selected' column and pressing of the 'Generate Update Keys for Selected users' button is needed.

Table continues on next page

9. Stations using DNP3.0 secure authentication V2 or IEC 60870-5-101/104 with predefined Update Keys:

   1. For stations connected to slave lines:
      Enter the Update Key manually for each user in each station object according to the settings in the master end. This applies in all cases, i.e. when the master is SYS600 or when it is a third party system.
   2. For stations connected to master lines:
      Generate the Update Key for each user in each station object. The configured user names must match the users created to MicroSCADA (case sensitive). If the name is not found in the key storage for the station object, the user is not able to send control commands to the station. Make a clear-text copy of the settings of each station (e.g. a screenshot) and store them in a safe place. These copies are used when the corresponding configuration is made to the slave devices using their own tools (= third party system).

   Stations using IEC 60870-5-101/104 or DNP3.0 secure authentication V5:

   1. For stations connected to slave lines:
      If symmetric mode is used, in 'Stations' level of the Authority Tool, set Update Key Mode of the station(s) to 'Symmetric' and paste the 'Authority Certification Key' provided from the master/authority system. The Authority Certificate Key can be different for each station as long as they match the ones in the master system. If the used update key method is 'SYMMETRIC_AES128_SHA1_HMAC' (possible only in DNP3.0), the authority certification key length may be 16 bytes (128 bits) or 32 bytes (256 bits). This is dependent on the functionality of the remote system, see attribute ZV index 0 for the compatibility flag. With other symmetric update key algorithms, the authority certification key length is 32 bytes (256 bits).
      If asymmetric mode is used, in 'System' level of the Authority Tool, import 'Authority Public Key' provided from the master/authority system. In 'Stations' level, set Update Key Mode to 'Asymmetric', select Station(s) and press 'Generate Station Key pair(s)' to generate asymmetric key pairs for stations. Export the outstation public key using button 'Export station public key(s)' to be used the in the master system. If multiple slave lines are used in asymmetric mode and those use different authorities, the slave lines must configured to different PC_NET instances and they must use separate key storages.
   2. For stations connected to master lines:
      If symmetric mode is used, in 'System' level of the Authority Tool, select the wanted Authority Certification Key length and press 'Generate' button to generate an Authority Certification Key. Copy the key to be provided for the slave system, but keep it secured. In 'Stations' level, select station(s) and set the Update Key Mode to 'Symmetric'. The generated Authority Certification Key from 'System' level should be assigned for each station automatically. In case a different Authority Certification Key is needed for some of the remote IEDs, a dedicated authority certification key can be pasted manually for any created station (in 'Stations' level). The authority certification key visible in 'Station' level is always the one used in communication. If the used update key method is 'SYMMETRIC_AES128_SHA1_HMAC' (DNP 3.0 only), the authority certification key length may be 16 bytes (128 bits) or 32 bytes (256 bits). This is dependent on the functionality of the remote system, see attribute ZV index 0 for the compatibility flag. With other symmetric update key algorithms, the authority certification key length is 32 bytes (256 bits).
      If asymmetric mode is used, in 'System' level of the Authority Tool, select wanted Key Pair Length and press 'Generate' to generate an Authority Key pair and export 'Authority Public Key' for the slave system. In 'Stations' level, set Update Key Mode to 'Asymmetric', select Station and press 'Import outstation public key' to import the public key of the slave system. In asymmetric mode the Authority Certification Key is not used.

10. Save the key storage database and make a back-up. In case of a IEC60870-5-101/104 slave or slave using DNP3.0 secure authentication V5, the back-up can be used to restore a situation where no users have been created for the slave. If this done, the master must repeat the 'User Add' operation and the update key negotiation for each user.
    In case the key storage handling is made in the separate authority computer, export (i.e. encrypt) the saved key storage database using the Exchange Key from the target computer running SYS600. Use menu selections File -> Export To and from the opening dialog, select 'Select Target site exchange key'. Name the file according to the role of the target computer and transfer the exported key storage database file to the target SYS600 computer.

Table continues on next page

11. Define the name and the path of the key storage to the KS attribute of the NET Node. If the target system is an HSB system, the private-public key pair created in System A must be imported to System B by using the Exchange Key Tool of System B before the key storage can be used. The password given when the key pair was created will be requested before the importing. If IEC60870-5-101/104 or DNP 3.0 secure authentication v5 is used, the key storage file need to be shadowed, see section 'Using key store in hot stand-by system for information'. With DNP 3.0 secure authentication v2, shadowing is not needed because the key storage file is not updated by pc_net.
12 For stations connected to master lines, the corresponding settings must be done to the slave devices using the device specific tools. If the slave is running SYS600, the instructions are listed above. Activate and test the secure authentication by setting the ZA attribute of the corresponding station object. For more information, see protocol specific manuals.
13. Repeat steps from 2 (a or b) to 12 for each computer running a PC-NET communication module with secure authentication.

Limitations

- One PC-NET instance can use only one key storage
- A DNP3.0 station object can use either V2 authentication of V5 authentication but not both.
- A station object can use either symmetric or asymmetric update key change method but not both.

Troubleshooting

In case there are problems when connecting to another system, the following tips may help:

- In Authority Tool a red mark besides the update key field may be visible when the update key length has been changed. In this situation, it is necessary to select all users from 'Selected' column and press the button 'Generate Update Keys for Selected users'.
- When setting up the system, more UAL events related to error situations are visible when the UA attribute of the STA object is set to its highest reporting level. See UA attribute description from the protocol manuals for details.
- When SYS600 is operating as a master and it is clear that the slave device has lost its user information, the users can be deleted and recreated using station attribute NU. See protocol specific manual for details. When deleted, only the users created by the master are deleted. If new users have been created from the front panel of the IED, they are not deleted.
- Communication log can be taken using any network analyzer (TCP and UDP) or serial analyzer (RS-232 / RS-485). In case the TLS encryption is used in TCP mode, the unencrypted communication log can be recorded using the protocol analyzer of the PC-NET, see line attribute AO in the SYS600 System Objects manual.
- Authentication diagnostics in the on-line mode of the System Configuration Tool provides information if some operation fails repeatedly. If the failed operation is directly related to certain user, repeating the same operation and recording the changes in the authentication diagnostic counters for that user provides helpful information for the analysis.
- In case the used protocol is IEC 60870-5-101/104 and key negotiation for user 1 fails, remote device may expect the user = 1 be named as "Default". Check the remote and configuration and enter the name "Default" for user = 1 by selecting "Users" in the AuthorityTool. This setting has an effect in the whole key storage, which means that DNP3 communication cannot be configured to the same key storage.
- If the used key storage database file size changes to 0kb after the file is updated, it could mean that the PFX file is not correctly bound to the user account. Start the Exchange Key Tool in that computer and log in with the key user role. If a note appears that advices to bind the PFX file (see Figure 25), rebind the PFX file using the instructions given in the part 'Installation of the Exchange Key Tool' in this section.

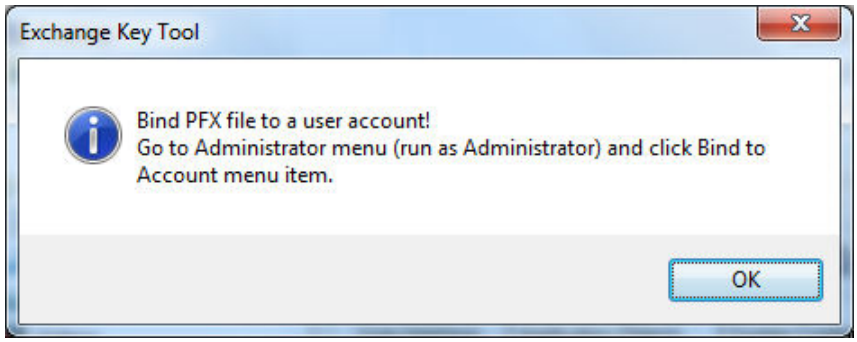


Figure 25: Bind PFX file

Notes and tips related to handling of key storage files

- If Authority Tool is used in SYS600 computer, it is important to keep Authority Tool password in secrecy and avoid using the same Authority Certification Key in multiple geographical locations. Unauthorized access to key storage reveals the secret keys and reconfiguration of the system is becomes necessary.
- An exported key storage file cannot be used in another computer but must be exported again using the Exchange key from the new target computer.
- Take a good care of the Authority Tool database because if it is lost, the exported databases from the same key storage file cannot be utilized as a back-up.
- Key storage may contain Station objects and user sets which are not used by the system configuration yet. This will help to test the system step by step.
- It is safest to create at least one key storage for each geographical location. If one of those is compromised, the information cannot be utilized for an attack to another location.
- Same communication line in PC-NET may contain stations that use authentication and that do not.
- If SYS600 is used both in the master and slave end, the same key storage can be exported twice and no manual entering of the user numbers and update keys are needed for key storage used in the computer operating as slave, i.e. in the COM500*i* computer. Each key storage must still be exported using the Exchange Key from the target computer. This practice is applicable only with DNP3 secure authentication V2.
- Temporary key storages can be created to test the system's functionality with secure authentication.
- If a 'User Add' operation is made for an existing user in slave using V5 or IEC60870-5-101/104, it is handled as 'User Change'.
- If master protocol is used and the remote device does not support on-line creation of users or there is a need to use only one user (e.g. Single User role in DNP 3.0), it is possible to map the command sending from the interactives users to this one user. See protocol specific manual, attribute ZV, index 0 for further information. If this setup is used, the key storage configuration becomes easier but it is not possible log in the remote system who actually made e.g. a control command. This setup does not make the system more vulnerable to external attacks.

Installation of the Exchange Key Tool or Authority Tool

Exchange Key Tool is delivered as part of the SYS600 version 9.3FP2 or newer and Authority Tool is delivered as part of the SYS600 version 10.0 or newer. The installation of these tools is needed only if the secure authentication, as described in this Section 4.5.2.8, is used on the computer in question. If Authority Tool is used in SYS600 computer, the installation of the Exchange Key Tool is not needed.

The installation of the Authority Tool or Exchange Key Tool requires a .PFX file (Personal Exchange Format). It strongly recommended to create a system or customer specific PFX file for Authority Tool or Exchange Key Tool installation. Customer specific PFX files can be created using system management tools such as Microsoft System Center Configuration

Manager. Alternatively, see e.g. 'https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-

and-2008/cc753127(v=ws.10)' or 'https://blogs.msdn.microsoft.com/kaevans/2016/08/12/using-powershell-with-certificates'.

OpenSSL or Windows SDK can be used to create PFX files, too.

The installation steps are as follows:

1. Open Tool manager as administrator, select **System Configuration** Tab
2. Select **Edit/Insert Tool/User Defined**
   (if no previous SYS600 installation, select 'Exchange key' or 'Authority Tool' from the 'installed tools' list and skip steps 3 & 4)
3. Select **Browse VSO**\sc\Stool\SysConf\EEK.VSO (Exchange Key Tool installation)
   or
   Select **Browse VSO**\sc\Stool\SysConf\AUTHORITYTOOL.VSO (Authority Tool installation)
4. Give Icon description "Exchange Key" or "Authority Tool", press **OK**
5. Double-click the created icon
6. Select **administrator**

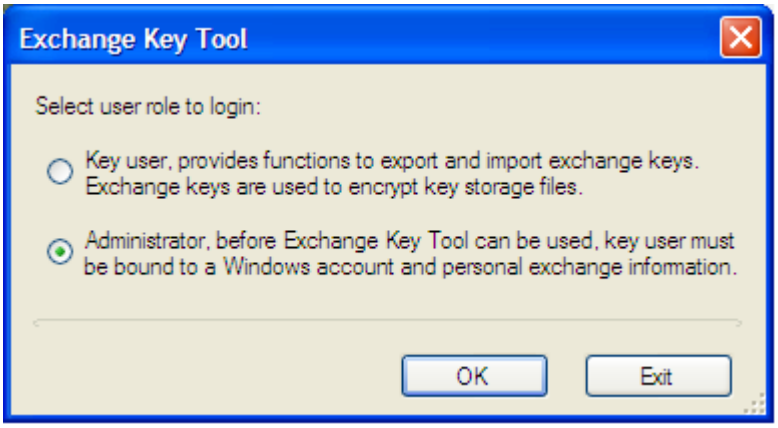


Figure 26: Role selection dialog in Exchange Key Tool or Authority Tool

7. Select **administrators/Bind to Account** from the Menu bar
8. Enter a username, e.g. "keyuser" to the Select or Create Username field. If a new username is given, a Windows user with this name will be created
9. Press **Import new** to select a PFX file, select the PFX file (see the creation instructions above)
10. Enter the required password for the PFX file
11. Select the imported PFX from the list and press **Bind**

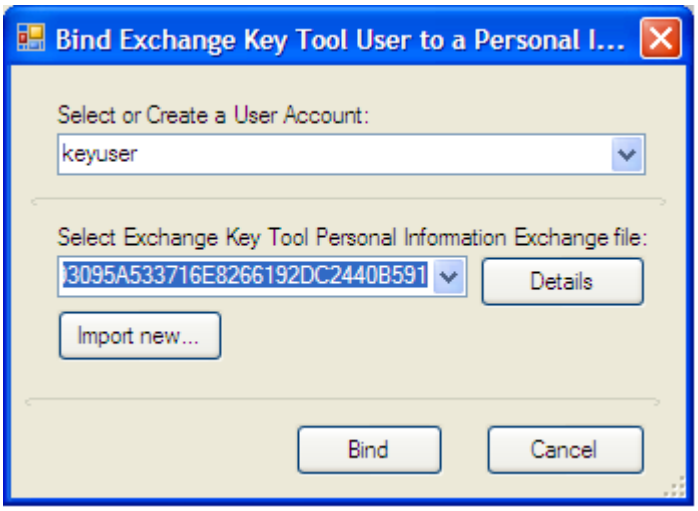


Figure 27: Bind Tool user to PFX

12. If a new username was given in step 8, a password for the created Windows user must be entered in this phase.
13. Binding of the Tool to a Windows user should be successful. Close the tool. If the installed tool was Exchange Key Tool, see next section for Exchange key file importing and exporting. New password for the created user must be given when the tool is started for the first time as 'Key user'.

In case the password changing fails and tool reports 'Access denied', it is worth to check that directory \ProgramData\Microsoft\Crypto\RSA \MachineKeys has 'Write' access rights for the user in question.

Exchange key file importing and exporting

These steps are not needed if Authority Tool is used in SYS600 computer and system is not a HSB system. If system is a HSB and Authority Tool is used in SYS600 computer, same steps need to be done using Authority Tool.

When private key is exported for System B in HSB setup, the it worth to consider persons who have access to created .pvkey file and password. Together with the keystorage file from a running system, keystore can be opened in an external computer and secret keys become visible. Keep .pvkey file in a safe place and do not copy it to the SYS600 system computer, it is not needed by SYS600 after it has been imported using Exchange Key Tool or Authority Tool of System B.

Start Exchange Key Tool (or Authority Tool if used locally).

1. Double-click the icon.
2. Select Key user
3. Enter the password. If the Exchange Key Tool is started for the first time as a Key user after having been installed and/or bound to an account, the Old Password is the password of the Windows user given in step 12 of the installation phase.
   Enter a new password. When a new password is entered, the tool closes itself and needs to be restarted. In this case, proceed from step 1.
4. Key exporting and importing
   4.1. If an Exchange Key needs to be exported (single system or System A in a HSB configuration), select **Key Management/Export Exchange Key** and select the type of the key:

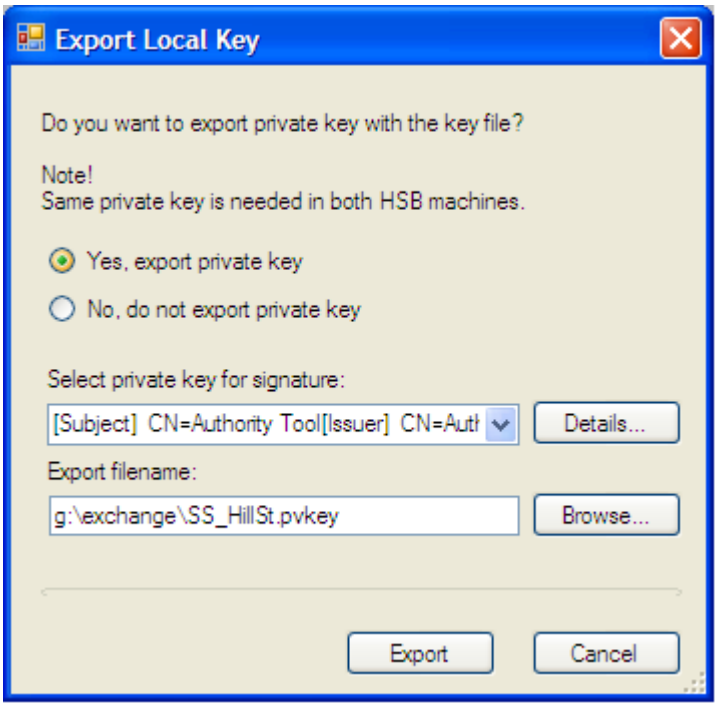


Figure 28: Key exporting

- An exchange key and a private key are needed if the key storage is used in HSB system. Password must be given to protect the private key. If private key is exported, the created .pvkey file should not be stored to the SYS600 computer (see note above).
- A private key is not needed if the key storage is used in a single system.
- Press **Browse** and define the filename.
- Press **Export**. Store the created file, for example to a USB stick.
- The exchange key will be used as an encryption key for all key storages in the computer in question (or the computer-pair in HSB system).
- Transfer the exchange key file to the computer where Authority Tool is used. In case of a HSB system, also transfer the private key file to the parallel system.

4.2. If a private key needs to be imported (System B in a HSB configuration), select **Key Management/Import Exchange Key**. Insert for example a USB stick, press **Browse** and select the key pair file to be imported.
Press **Import** and enter the password given to protect the private key. The imported key will be used as an encryption and decryption key for all key storages in the computer in question.

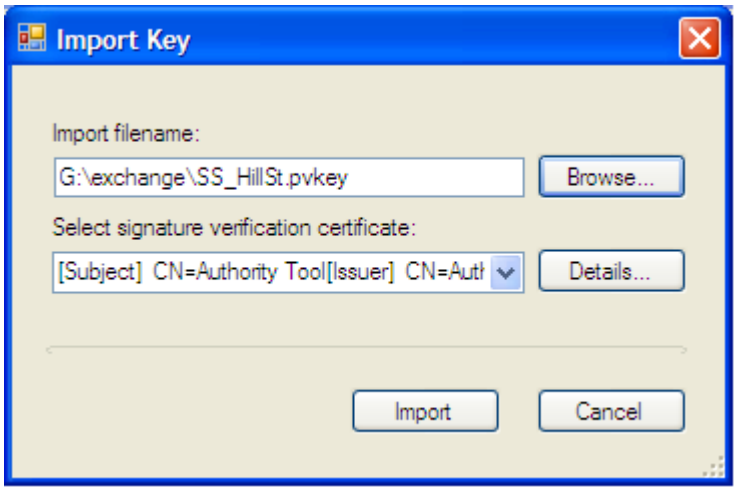


Figure 29: Key importing

Using key store in hot stand-by system

After importing the A computers private key to the B computer, as instructed above, a key storage exported from the Authority Tool with the A computers exchange key, can now be opened in the B computer. In HSB system both computers should have exactly the same key stores in case a switch-over takes place. A key store is updated when new secure authentication user is created or update or session keys are changed. The key store updating happens when the communication is running so the key store file needs to be explicitly shadowed by the SCIL application.
Shadowing of a single file should be done with 'SHADOW_FILE' SCIL command (for more information about the command see section 10.16 in SYS600 Programming Language SCIL manual). For a key store to be shadowed from one HSB computer to another a command procedure, that is executed for example when the key store is updated, can be used.
When a switch-over happens in HSB system, the shadowed key store needs to be updated also to the PC_NET. Updating should be done in the APL_INIT_H command procedure by setting the used nodes Key Storage File (KS) attribute first to empty ("") and then setting the key store file name. All of the affected stations Authentication Used (ZA) attribute needs to be set to 'disabled' during the KS setting and then set to the correct authentication mode again.

Example 1:

SCIL command for shadowing a file:

```
@return_value = SHADOW_FILE("PICT/hsb_key_store.dat")
```

Example 2:

Example of the lines that update the Key Storage File (KS) attribute of the node when a switch-over takes place. The following lines should be added to the 'APL_INIT_H' command procedure.

```
#SET STA'x1':SZA = 0
#SET STA'x2':SZA = 0
#SET NET'X':SKS = ""
#SET NET'X':SKS = "c:\sc\apl\main\PICT\hsb_key_store.dat"
#SET STA'x1':SZA = 2
#SET STA'x2':SZA = 2
```

4.5.2.9 Secure communication using TLS (IEC 62351-3)

Secure communication uses TLS (Transport Layer Security) and SSL (Secure Sockets Layer) protocols to encrypt the protocol communication. In MicroSCADA the TLS secure communication is supported with the following protocols:

- DNP3.0 TCP/IP master and slave
- IEC 60870-5-104 master and slave

For full confidentiality, the TLS secure communication should only be used with the secure authentication explained in section 4.3.2.7.

The TLS is configured using SCIL commands or System Configuration Tool in MicroSCADA. The secure communication requires certification files which are used to authenticate the communicating devices. There are 3 ways the certificate files can be acquired:

1. Certificates can be made (for example using OpenSSL) or they can be bought from a third-party provider.
2. Hitachi ABB Power Grids personnel can create the certificates.
3. Certificates can be created as self-signed certificates using the MicroSCADA. This is explained in detail below.

After the certificates are acquired, the secure communication can be started. The secure communication functionality is activated for a station when both CI attribute indices are set. CI index 1 defines the certificate and private key file and CI index 2 defines the trusted certificate authority file.

The certificate files location can be chosen freely, but it is recommended to place them in a folder with limited access rights to improve the security of the system. For example use a folder where only the MicroSCADA user has access rights.

It should be noted that all connections in a PC-NET instance must use same certificate and key file and trusted certificate authority files. If different certificate must be used, it must be configured to another PC-NET instance

If the certificate file contains multiple certificates, the CN index 3 is used for selecting a certain certificate that will be used. In CN(3) attribute the Certificate Name, write the Common Name, Country Code and Organization name needs to be listed and they should have the exact values that the wanted certificate has. If only one certificate exists in the certificate file, CN(3) can be left empty.

With CV attribute it possible to configure how the TLS communication works. Recommended configuration is to set the CV attribute index 3 to 1, which means that the connection is closed if the remote certificate cannot be authenticated. This prevents connections from sources with unknown certificates. Another configuration that is recommended to check is the CV attribute index 2. It defines if self-signed certificates are created. If certificates already exist, the CV(2) should be set to 0. In case the setting is 2, ‘Always’, the existing certificate is overwritten with a new certificate when the communication is started. More detailed descriptions of the configuration attributes are given in the related protocol manuals.

Creating the self-signed certificates using MicroSCADA

Creation of self-signed certificates is configured and set using the protocol attributes. The attributes can be configured using SCIL commands or with System Configuration Tool. Detailed descriptions of the attributes can be found in the protocol manuals. Self-signed certificates can be created with the following steps:

1. Configure IEC 60870-5-104 or DNP3 LAN communication between master and slave MicroSCADA computers.
2. In both computers write to the stations CI attribute index 1 a path and filename of the wanted certificate file. The file type should be ‘.pem’. At this moment the CI attribute index 2 can be set to the same as index 1. The file location can be chosen freely, but it is recommended to use a folder to which only the MicroSCADA user has access rights.

The folders in the path must exist beforehand.

3. If wanted, a passphrase can be set to the created certificate file using the CP attribute. After the certificate has been created, the opening of the certificate with MicroSCADA is prevented if the passphrase in the CP attribute is not correct.
4. Give the certificate file a name, country code and organization name using the CN attribute indices 1, 2 and 4. The CN(1), or Common Name, should be the IP address which the station listens to. The CN(2), or the Country Code, should be given in a form of two capital letters. CN(4), the Organization, should be a single name without spaces. The CN(3) can be left empty because there are no multiple certificates in the certificate file.
5. From CV attribute the index 2 should be set to 1 'If not Found'. This indicates that a self-signed certificate is created if the certificate described in CI(1) doesn't exist already.
6. CV(3) should be set to 1 'Close Connection'. This closes the connection if the remote certificate cannot be authenticated.
7. Other changes to the default setting can be done, but they don't affect the self-signed certificate creation and thus are not mentioned here.
8. Now the communication should be started. The certificates described in the CI(1) are now created to the given folders. However, the secure communication should fail at this point because the remote certificates cannot be authenticated. The communication can be stopped.
9. Make copies of the created master and slave certificates and rename them. Open them with a text editor and remove the private key parts from them and save the files. Now they work as a certificate authority (CA) files that are used to authenticate remote certificates.
10. Move the created CA files to the other computers certificate folder. Change the CI attribute index 2 value to match the new CA files.
11. Now start the communication again. The TLS secure communication should start and authenticate the remote certificates correctly.

Definition of certificate revocation list (CRL)

In IEC60870-5-104, it is possible to define a CRL (Certificate Revocation List) based revocation checking for remote certificates. CRL checking can be taken into use with following steps:

1. Configure IEC 60870-5-104 connection to use TLS and enable CRL usage using attribute CV, index 9. Values 1 and 2 enable CRL usage.
2. Configure system to download certificate revocation list file cyclically from the certificate authority in use. A suitable interval may be e.g. once a day, in a middle of the night. The example below shows how the CRL file downloading can be made using SCIL. The file location can be chosen freely, but it is recommended to use a folder to which only the MicroSCADA user has access rights.

The folders in the path must exist beforehand. It is recommended that the CRL file is stored in local file system.

3. After a successful transfer, (re)define the file and path to STA object using attribute RL (Revocation List). See SCIL example below. Redefining the file activates it usage, too. RL writing does not disconnect existing TLS connections. Checking activates when full TLS handshake takes place for the next time. Please note that if session resumption is used, big session resumption time may delay the next full TLS handshake.

```
; Connect this procedure to 24-hour time channel
#error continue
@local_crl1="c:\sys600_certs\pcnet1_conns.der"
@tls_sta = 1
@net = STA'tls_sta':BND
@x=ops_call("del 'local_crl1'")
@x=ops_process("powershell iwr -outf 'local_crl1' http://crl.cainuse.com/
crloftheday.crl","", "SERVER"); Win10&Server2016
;@x=ops_process("bitsadmin /transfer myJob /download /priority normal
http://crl.cainuse.com/crloftheday.crl 'local_crl1'","", "SERVER") ; Win7
#pause 2
```

```
@x=read_text("'local_crl1'",1,0)
@st=status
 #if %st == 0 #then #block
     #SET STA'tls_sta':SRL1="'local_crl1'"
     @st=status
      #if %st == 0 #then #block
        @x = console_output("New CRL file 'local_crl1' activated for
NET'net'")
      #block_end
      #else #block
          @x = console_output("New CRL file 'local_crl1' activation
failed for NET'net'")
    #block_end
#block_end
#else #block
   @x = console_output("CRL file 'local_crl1' download failed")
#block_end
```

HSB instructions
TLS secure communication can be used with master and slave HSB setups. Due to HSB pair computers having a separate IP addresses, they both must have their own certificate and private key files. The remote device communicating with a HSB pair must have a certificate authority file that can authenticate both HSB computer certificates. This is done by having multiple certificates in the certificate authority file.

When creating self-signed certificates for HSB setup, follow the instructions given above for both HSB pair computers. This means that first create self-signed certificates for the first HSB computer and the remote computer, and then create self-signed certificates for the second HSB computer and the remote computer. After this combine the two certificate authority files created for the remote computer by copy-pasting both certificate texts into same .pem file. Now the remote computer can authenticate certificates from both HSB computers.

If both master and slave sides have HSB setups, each computer needs to be able to authenticate both remote computers and thus have both remote computers certificates in the certificate authority file. If self-signed certificates needed to be created, the instructions given above should be followed for each computer pair. After this, each computer should have two certificate authority files which need to be combined by copy-pasting both certificate texts into same .pem file. Now each computer can authenticate certificates from both remote HSB computers. It is also possible to use option 'Close connection, no IP-address checking' in CV attribute index 3 in both computers, this will allow the communication from both remote IP-addresses if valid certificate is presented by the remote.

In TLS secure communication, by setting the CV attribute index 3 to 1 or 2, ‘Close Connection’ or 'Close connection, no IP-address checking', the communication is closed if the remote certificate cannot be authenticated. This is the recommended setting in both HSB pair computers, so that neither one accepts a faulty certificate.

4.5.3 Distributed process communication units

In principle, the process communication unit may be directly connected to any base system node in MicroSCADA X network. The application containing the corresponding process database may be in another base system node, and all data sent from the process communication unit is transmitted through the LIN objects between these nodes. The used protocol is ACP (Application Communication Protocol). The base system between the process communication unit and the upper base system routes the ACP messages in both directions. The STAn:B objects are created both to the routing base system and to the base system running the database.

A commonly used system setup is based on the distribution of the PC-NET process communication unit to separate computers. In this configuration, the primary link between the base systems is a LAN link. For more information on this configuration, see Section 4.5.3.1.

If the process communication unit is configured to contain slave protocols, it is recommended that the unit is directly connected to the base system that contains the application receiving the process data. In other words, it is not recommended that, for example, the COM500*i* application refers to STAn:S objects running in another computer.

4.5.3.1 Distributed PC-NETs

There are many reasons why it is necessary to divide the PC-NETs to operate in a separate computer or multiple separate computers:

- Computer hardware limitations of LON or serial cards
- Decreasing the problems caused by a computer failure
- Process communication causes CPU load
- Redundancy is required in the process communication level

The base system directly connected to the PC-NET usually contains no process database. Figure 30 presents the system containing a hot stand-by base system that contains a process database and three separate computers for process communication. In the system, the CPU load caused by the process communication is divided to three CPUs. Furthermore, if a hardware failure occurs in some of the computers, the rest of the system is still under control.

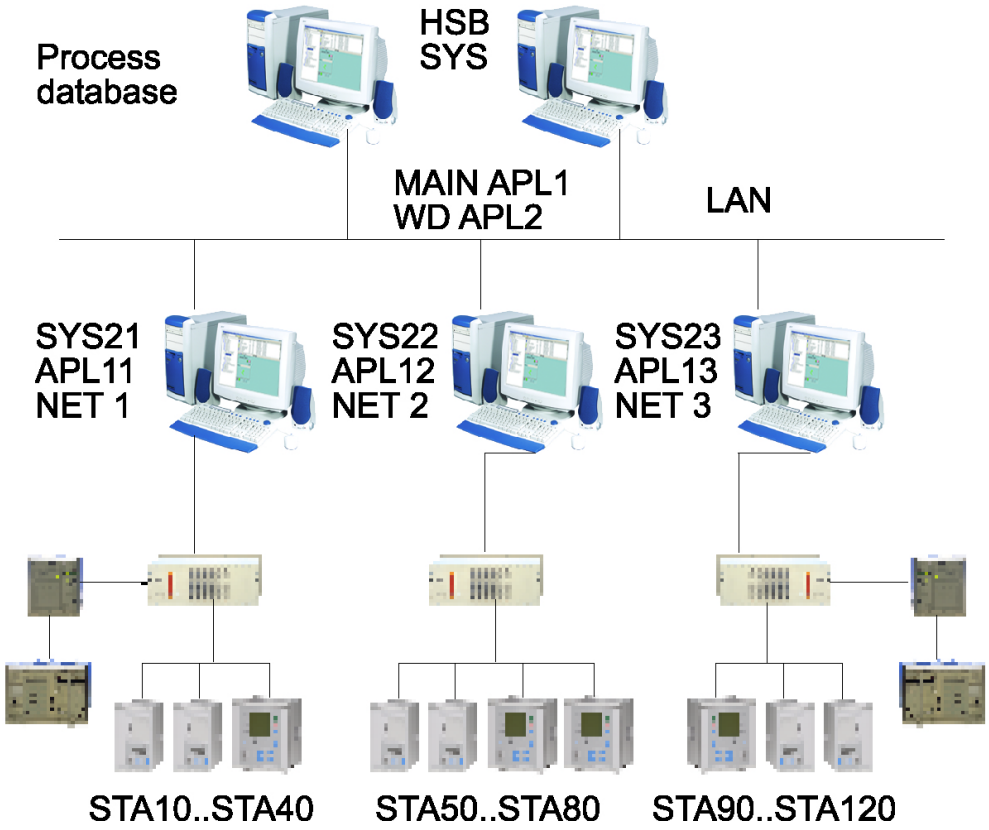


Figure 30: PC-NET configuration

The communication front-end base systems SYS21..SYS23 running APL11..APL13 only routes the ACP messages between PC-NET nodes and APL1. In the system start-up or in the hot stand-by switch the APL1 is defined to be in either of the base systems containing the database.

Figure 30 describes a situation in which there is only one PC-NET running in each computer. In practice, each of these computers may contain multiple PC-NET instances and various sets of lines using different protocols. Furthermore, each of these computers may also be doubled

using hot stand-by configuration. The watchdog APL object needed in hot stand-by configuration is APL2.

The system may also contain mirroring at the process database level.

The System Configuration Tool should be used in each of the computers running the PC-NETs. The configuration in the base system running APL 11 can be the same as presented in Figure 31.

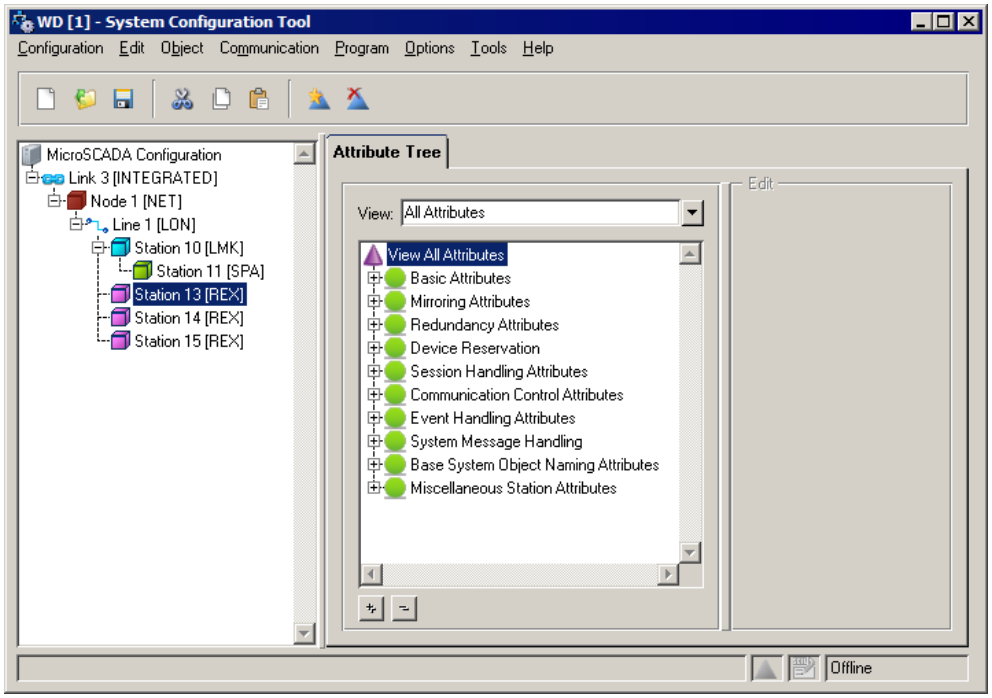


Figure 31: System Configuration Tool

The corresponding STAn:B objects should be configured to the base systems that contain the process database. The selected part from the SYS_BASCON.COM file for the system described above would be:

```
.
.
;
;Other Base System Nodes
;
#local NOD_Numbers = vector(21,22,23),- ;for example vector(11,12),-
    NOD_Names = vector("10.10.10.21","10.10.10.22", "10.10.10.23"),-
    NOD_Addresses = vector(221,222,223)
.
.
;
;Gateway Nodes
;
#local GW_NOD_Numbers = vector(1,2,3),- ;for example vector(20,22),-
            GW_NOD_Addresses = vector(201,202,203)

.
.
;An example of STA object definitions for NCC level (MR = "IMAGE")
stations
;
#local Stations  = ( 10, 11, 13, 14, 15, 16 )
#local Sta_ST    = ( "LMK", "SPA", "REX", "REX", "REX", "REX" )
#local Sta_Nodes = ( 1, 1, 1, 1, 1, 1 )
#local Sta_MR    = ( "NONE", "NONE", "NONE", "NONE", "NONE", "NONE" )
```

```
#local Sta_H_Apl = ( 0, 0, 0, 0, 0, 0 )
#local Sta_H_UN  = ( 0, 0, 0, 0, 0, 0 )
#local Sta_I_APL = (vector(15),vector(15),vector(15),-
vector(15),vector(15),vector(15) )
#local Sta_I_UN  = (vector(2005),vector(2012),vector(2031),-
vector(2042),vector(2051),vector(2091))
.
.
```

The definition file above will create NOD:B objects for base systems and PC-NET nodes. After this, the STAn:B objects are created for each station object created to the PC-NET nodes. This configuration should match with the configurations entered with the System Configuration Tool.

When a hot stand-by switch, i.e. a take-over occurs during run-time, the main application changes to HOT state in the adjacent base system. In this situation, a procedure SHADMAPNET is executed and PC-NET nodes are informed that the application is running in another base system node. The used attribute is NET node attribute SY.

4.6 Configuring System Self Supervision

System Self Supervision (SSS) provides information about the status of the software and hardware components of the MicroSCADA X system. This information appears as supervision events in the Event and Alarm Lists. Typically, the dedicated System Self Supervision process display should become constructed to show the status information in the graphical view by using the supervision symbols with the status symbol appearance and color semantics. Supervision symbols are dynamically updated, and thus always reflect the real status of the supervised system. Supervision symbols also provide an access point for launching the appropriate Supervision Control Dialog for monitoring more detailed information about the selected object, or dialogs used for controlling operations, such as performing manual switch-over in the redundant system. Standard authority handling mechanisms are applied to supervision controlling operations.

For more information on the System Self Supervision functionality from the operator's point of view, see SYS600 Operation Manual. For information on the configuration steps from the project engineer's point of view, see SYS600 Application Design manual. SYS600 Application Design manual also includes information on how to adjust the System Self Supervision functionality to project-specific needs.

To introduce the supervision logic in application:

1. Keep the supervision logic for each component as simple as possible. Introducing more application objects into the re-processing chain of the same supervision event means that there will be delays and the granularity of the operation will be split to several places.
2. Represent the status information in a binary way so that one value represents the good status and another value represents the bad status. In the SYS600 process database, use Binary Input type process objects for this. Use naming convention for supervision process objects as it is easier to recognize them among the other application process objects.
3. When detailed information is needed for certain system component, include detailed pictures or displays providing the detailed information on request.
4. Use Windows operating system events by using OS_EVENT predefined event channel and re-route the information into the process objects.
5. If the system contains devices capable for sending SNMP messages in the SYS600 network, use the SNMP-OPC gateway solution. The solution helps to map the supervision information into process objects by using the subscription of related OPC items from the SNMP-OPC Server namespace. See manuals SYS600 Application Design and SYS600

External OPC Data Access Client for more information about the configuration of SNMP OPC Server feature of SYS600.

6. When engineering the system supervision picture, re-use the symbols found from the SYS600 Display Builder Palette. Include the color semantics representing the status information either beside the symbol or inside the symbol, when feasible.

4.7 Configuring communication gateway

Before gateway engineering can start, the application has to be prepared for the gateway functionality. This is done in the Signal X-Reference Tool as shown in Figure 32. Once the application is prepared, Tool Manager should be restarted as shown in Figure 33. If the Signal X-Reference tool has been opened from Tool Launcher, the Signal X-References tool restarting is enough (Figure 34). After restarting of the Tool Manager or the Signal X-References tool, COM500*i* creates all the necessary application objects, such as event and time channels. Also, command procedures are created automatically, as well as the directory \sc\apl\ <name> \com500 that is used for storing cross-reference files and parameter files.

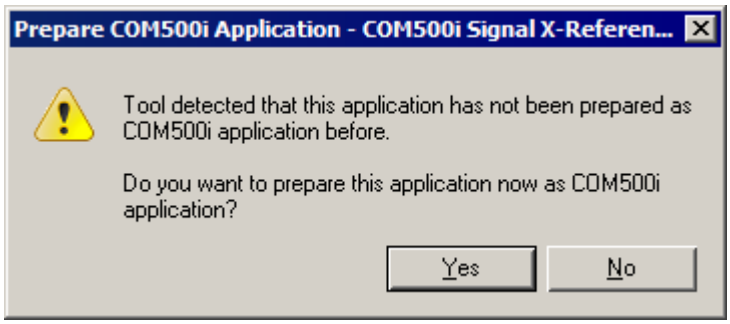


Figure 32: Prepare COM500i application

Figure 33: Application prepared successfully

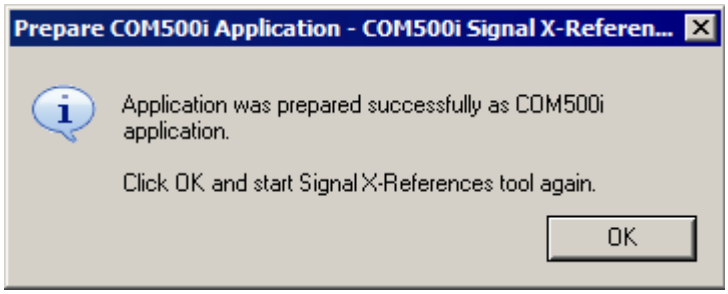


Figure 34: Restart Signal X-Reference tool

4.7.1 SYS_BASCON.COM modifications

Command procedures of COM500*i* use parallel queues. COM500*i* applications must be introduced in SYS_BASCON.COM.

```
#local COM500 = vector(TRUE)   ;TRUE = COM500i application,
                                   ;FALSE = not COM500i application
…
#if COM500(j) #then -   ;COM500i Application
        Apl_Permanent = merge_attributes(Apl_Permanent, -
                  list(PQ = 16,-      ;Number of parallel queues
                   QD = (1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1)))
                   ;Parallel queue dedication
```

4.7.2 Gateway license

The gateway functionality can be used if the value in the gateway field is not zero. The value also shows how many NCC lines can be configured in the Signal X-Reference Tool.

Figure 35: Gateway license

4.8 Configuring peripheral equipment

The following devices have a connection from MicroSCADA and are therefore defined in SYS_BASCON.COM:

- Printers

4.8.1 Configuring printers

To connect a printer directly to the base system computer:

1. Connect the printer to a parallel port or a serial port. Printers connected to the parallel port of the base system computer can be placed maximum 3 meters from the base system computer. Serial lines allow the connection lines to be up to 15 meters without modem.
2. Configure the printer to the operating system as described in the Windows manuals. Define the printer as "shared" if it is going to be used by several base systems or other Windows computers on the LAN.
3. Select the connection mode on the printer.
   - Select Parallel Mode if the printer is connected to the parallel port.
   - Select Serial Mode if it is connected to a serial port.
4. Configure the printer in the base system as a PRIn:B object. If the printer is used by several base systems or by programs other than MicroSCADA X on the same or other computers, set the printer's OJ attribute to 1.

Printers that are used by several base systems should be defined in all base systems, both regarding the operating system configuration and the MicroSCADA X configuration.

4.8.1.1 LAN connection

Printers connected to a base system computer or LAN should be configured in all base systems that will use the printers.

4.8.1.2 NET connection

A printer connected to a PC-NET communication unit can be used by all base systems connected to the same network. The printer should be defined both in the base systems that uses it and in the PC-NET unit to which it is directly connected as shown in Figure 36. It is assumed that the PC-NET unit has been defined to the base system as a NODn:B object.

Include the following definitions in each base system which will use the printer:

1. Create a PRIn:B base system object with at least the following attributes:

TT	LOCAL
ND	The node number of the NET unit to which the printer is directly connected.
TN	The device number of the printer in the NET unit to which it is directly connected.
DT	COLOR, NORMAL, or TRANSPARENT. Select NORMAL if the printer will be used exclusively for black and white character-based printouts. Select COLOR for all other types of picture based printout. Even if the printout will be black and white, COLOR is preferable as this mode provides a more picture-like printout by exchanging graphical characters to printer specific characters. Select TRANSPARENT if the printout will be SCIL defined.
DC	NET

For more information on the attributes of the PRI object, see SYS600 System Objects. Optional features are defined by the following attributes:

LP	Lines per page
QM	Queue length maximum
OD	Output destination: PRINTER, LOG (disk files) or BOTH
LD, LL, LF	Printer log attributes, specify the management of log files The attributes are meaningful, if OD = LOG or BOTH

For more information on the attributes of the PRI object, see SYS600 System Objects.

2. If needed, map the printer for an application with the APLn:BPR attribute. The printer mapping is required only when using a logical printer number that is not the same as the printer object number.

Only the printers mapped with logical printer numbers 1 ... 15 can be used as alarm and event printers. Printer 15 is reserved for event lists.

Include the following definitions in the NET unit to which the printer is directly connected:

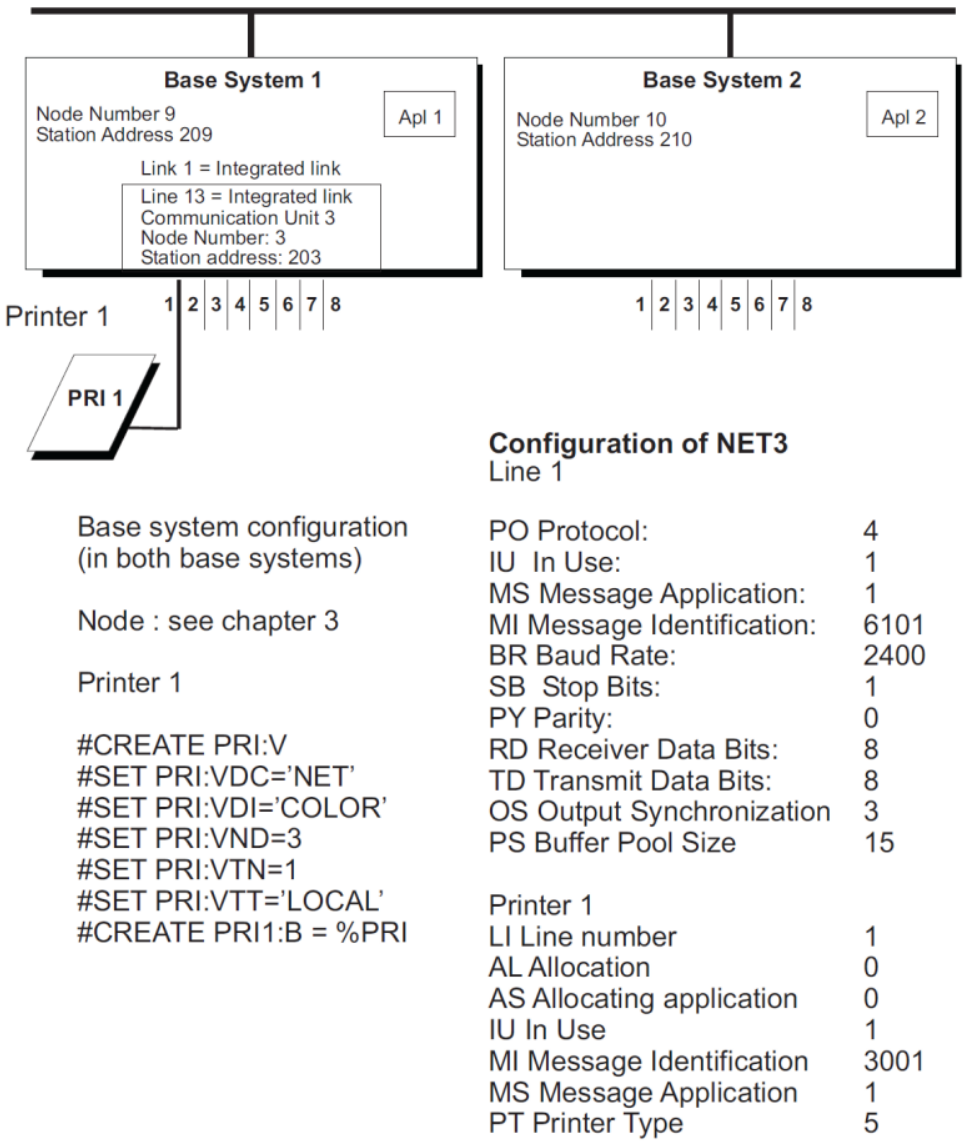


Figure 36: Configuration of a printer connected to NET unit

1. Select a line for the printer and define the line with the ASCII protocol:

PO	4
IU	1
LT	0
MS, MI	System message handling, see Section 4.5.2.1.8.
PS	Buffer pool sizes.
BR	Baud rate (recommended value 2400)
PY	0
RD	8
Table continues on next page	

SB	1
TD	8
OS	Output synchronization, see SYS600 System Objects.

2. Define a printer (a PRI object) on the selected printer line with the attributes:

LI	The number of the selected line
IU	1
MI, MS	System message handling
AL, AS	0 (the printer reservation is handled automatically by the base system
PT	Printer type: 1 = character-based, black-and-white 2 = transparent 3 = pixel based, black-and-white 5 = character-based, black-and-white, graphical characters replaced by printer characters 6 = Facit 4544 7 = pixel-based, color

For more information on the attributes of the PRI object, see SYS600 System Objects.

When a base system is started, its default application sends a message to the printers (form feed). Therefore, make sure that these applications are defined in the NETs.

4.9 Configuring time handling

Time synchronization of the SYS600 system and the connected IEDs is necessary in order to interpret correctly any time-stamped information provided by the system. The time-stamped information can be, for example, the event information from the IEDs.

The SYS600 system uses the internal clock of the computer as the source of time. Depending on the configuration and the system structure, this clock may be synchronized from an external source such as GPS, a radio clock or another SYS600 system. The GPS clock reference device is connected through a SLCM card via a LON line or using an external application. When the SYS600 system operates as a communication gateway, the synchronization is often received from the network control center through the used communication line. If IEC 61850 OPC server is used in MicroSCADA PC, SNTP time synchronization can be used.

The same internal clock is used when the real IEDs connected to the SYS600 system are synchronized through the communication lines in process communication units. In most of the communication protocols, there is a predefined method to synchronize the IEDS. For more information about the synchronizing methods, see the protocol specific manuals. When SYS600 is used to synchronize the IEDS, the synchronization command is usually initiated cyclically from the application running in the base system. The related station object attribute is SY. In some protocols, there is a possibility for the IEDS to request a time synchronization from the master.

The process communication units running in the same computer use the same system clock, and therefore there is no need to make separate synchronization for the unit. This is different from the DCP-NET units used with the older SYS base system versions where the units had a separate clock in the used hardware.

When operating as a communication gateway, the network control center may operate in a different time zone. The corresponding compensation attribute is TZ which exists both in SYS:B object and NET:S object (only for process communication unit PC-NET).

4.9.1 Configuring time synchronization

The time synchronization characteristics usually vary from one system to another and are strongly dependent on the customer and/or the IEDS requirements. The used communication protocol also has an effect. For example, with SPA-protocol the time synchronization command is sent every second. With some other protocols, the interval of the time synchronization may be hours.

The following steps should be considered when the time synchronization concept for a SYS600 system is planned:

1. Find out the system requirements.
2. Resolve the clock reference for the planned system.
3. Resolve IEDS requirements with the used communication protocol.
4. Define the synchronization intervals for IEDs or IEDS groups.
5. Define the time synchronization details of the used protocol.

In step 2, it is defined if the clock of the SYS600 computer is synchronized from an external source. This source could be a network control center or a radio clock device connected to a communication line in PC-NET, an external application using SNTP server and/or GPS device, a SLCM-card connected to LON star coupler and so on.

If the time source is connected to the communication line of the PC-NET, some configuration tasks may be required. For IEC60870-5-101/104 and DNP3.0 protocols, see the station attribute TC in corresponding slave protocol manuals. For LON protocol, see SYS600 System Objects. The time zone compensation is done with the SYS:BTZ attribute.

If the time source is an external application or a special device, see the corresponding manuals for more information.

In step 3, the requirements of the used IEDs are defined. These requirements may define the minimum frequency of the incoming time synchronization or it may require that the synchronization should be preceded by a delay measurement. The IEDs may also define if they accept the time only with or without date or only as a broadcast message. It is also possible to synchronize the device from an external time source.

Steps 4 and 5 require SYS600 application programming. With protocols implemented to process communication unit PC-NET, a common practice is to define a time channel or a set of time channels that are executed with a defined interval. A command procedure which actually initiates the synchronization is then connected to the time channel.

Example for DNP3.0 master protocol:

```
#LOOP_WITH I=20..25
#IF STA'I':SIU==1 AND STA'I':SOS==0
#THEN #BLOCK
#SET STA'I':SSY=(1,0) ; direct, no time delay measurement
#PAUSE 10
#BLOCK_END
#LOOP_END
```

If the procedure is attached to a time channel executed in 1 hour intervals, the IEDs related to STA20..STA25 are synchronized once every hour. The same station object attribute SY is used for time synchronization in most protocols implemented to PC-NET. For more information, see the protocol specific manuals and SYS600 System Objects manual.

4.9.2 Configuring time zone and daylight saving

To adjust computer clock and MicroSCADA applications to daylight saving time:

1. Open **Control Panel/Date and Time**.
2. Click the **Time Zone** tab to change your zone.
   - To perform the above action, click the drop-down arrow and select the current zone.
3. Select **Automatically adjust clock for daylight saving changes**.
4. Open **Monitor Pro/Tools/Options**.
5. Click **Daylight Settings/Automatically adjust applications for daylight saving changes**.

4.9.3 Time zone and daylight saving history

The base system maintains a history of time zone and daylight saving time rules obeyed in the site. The history is found in the SYS_TIME.PAR file located in folder \SC\SYS\ACTIVE\SYS_.

The history is used for two purposes:

- To display old time tags correctly.
  The site may have been moved from a time zone to another, or daylight saving time may have been applied differently in the past. As the base system stores the time tags in UTC time, the history is needed to correctly convert the tags to the local time of the moment of event.
- To time future actions correctly.

The base system transfers the current time zone and daylight saving time information from the operating system at each system startup and maintains the history automatically. However, there are a few cases, where the history should be edited explicitly:

- Time zone and/or daylight saving time settings are changed and it is not acceptable to shut down and restart the system for this reason.
- Prior to SYS 500 revision 8.4.4, the time tags were stored in local time. If a revision 8.4.3 application or older is upgraded and the time zone and/or daylight saving time settings have been changed during the age of the application, the old settings should be copied to the base system to display old time tags correctly.
- If it is known that time zone and/or daylight saving time settings are to be changed in the future, it is useful to transfer the new settings to the base system in advance. Then, the base system is able to correctly time the once-only time channels scheduled after the coming change of settings. In addition, other local/UTC time conversions of future time tags are done correctly.

The explicit editing of SYS_TIME.PAR is done with SCIL function TIME_ZONE_RULES, see SYS600 Programming Language SCIL.

As there are some complications of local/UTC time handling, it is recommended that engineering of a new application is done in a PC that uses the time zone and daylight saving time settings of the target site.

4.9.4 Configuring the representation of dates

There are two places to configure representation of time and date in Monitor Pro:

- TF-attribute
  With TF-attribute the following conventions are available:
  yy-mm-dd hh:mm:ss, when TF=0
  dd-mm-yy hh:mm:ss, when TF= 1
  mm-dd-yy hh:mm:ss, when TF= 2
  The default value for time format can be configured where node for the base system is created.

```
Extract from SYS_BASCON.COM
#local Sys_Modify = list(-
      -; Communication Attributes
       ER = 0,- ;Enable Routing 0=disabled, 1=enabled
       TI = 10,- ;Timeout Length
      -; Time Handling Attributes
       TM = "SYS",-  ;Time Master, SYS or APL
       TR = "LOCAL",-  ;Time Reference, LOCAL or UTC
       TF = 0) ;Time Format
                -; 0 = yy-mm-dd hh:mm:ss
                -; 1 = dd-mm-yy hh:mm:ss
                -; 2 = mm-dd-yy hh:mm:ss
```

Like Alarm list and Event list, Monitor Pro applications follow time format defined by TF.

- Initialization file FRAMEWINDOW.INI
  FRAMEWINDOW.INI is a user specific file and it is located in the \sc\apl\'apl name'\PAR \'user name' directory. The representation order of time and date can be configured in the DAYFORMAT section of the above file. The configuration applies for the Monitor Pro status bar. For example:
  [DAYFORMAT] FreeDateTimeField=%Y-%m-%d %H:%M:%S
  The default representation style in the status bar follows the TF attribute setting. If DAYFORMAT is defined, it will be used.
  All possible options for configuring DAYFORMAT are:
  - Year
    %y, Year without century
    %Y, Year with century
  - Month
    %b, Abbreviated month name
    %B, Full month name
    %m, Month as number (01 – 12)
  - Week
    %V, ISO 8601 week number (01-53)
    %W, Week of year as decimal number with Monday as first day of week (00 – 53)
    %U, Week of year as decimal number with Sunday as first day of week (00 – 53)
  - Day
    %d, Day of month as number (01 – 31)
    %j, Day of year as number (001 – 366)
    %w, Day of week as number (0 – 6; Sunday is 0)
    %a, Abbreviated weekday name
    %A, Full weekday name
  - Hour
    %H, Hour in 24-hour format (00 – 23)
    %I, Hour in 12-hour format (01 – 12)
    %p, Current locale's A.M./P.M. indicator for 12-hour clock
  - Minute
    %M, Minute as number (00 – 59)
  - Second
    %S, Second as number (00 – 59)
  - Misc
    %c, Date and time representation appropriate for locale
    %x, Date representation for current locale
    %X, Time representation for current locale
    %z, %Z, Either the time zone name or time zone abbreviation, depending on the registry settings. No characters if the time zone is unknown.

4.10 Configuring networks

Each NET unit connected to a base system via one or more units should be defined to the base system as a node (NODn:B objects):

1. Create a NODn:B base system object corresponding to the indirectly connected communication unit. The NOD object number ('n') should be the same as the node number of the communication unit. The NOD object is given the following attribute values:
   LI = Link number (= LIN object number). The link to the nearest communication unit.
   SA = Station address of the indirectly connected communication units.
   Even if there is no communication between the base system and the indirectly connected NET, the node definition is necessary for the system diagnostics, online configuration and system maintenance.
   Correspondingly, each base system connected to a NET unit indirectly via other units should be defined to the NET unit as a node.
2. Define an External node (NET object) on the line to the nearest communication unit:
   Device type = NOD
   Device number = The node number of the indirectly connected base system.
   LI, Line number = The line to the nearest communication unit in the series.
   SA = Station address of the indirectly connected base system.
3. Define an application for each application in the indirectly connected base system.

4.10.1 Example

Figure 37 shows an example of a network of two base systems and a Front-end system containing two communication units. Application 1 communicates to process true Front-end system, and APL1 and APL5 (in Base system 2) communicate with each other.

The example includes only the definitions which are of importance for this particular configuration.

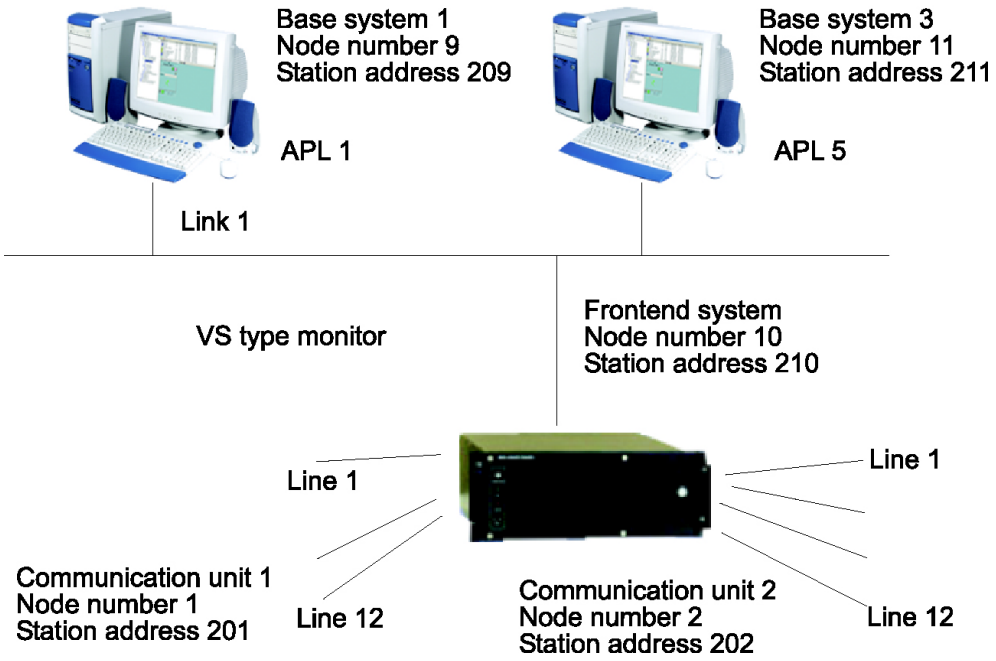


Figure 37: Network of two base systems and a Front-end system containing two communication units

4.10.2 Configuring base system 1

```
.
#local LAN_Link = 1 ;LAN link number
.
.
;
;Other Base System Nodes
;
#local NOD_Numbers = vector(10, 11),-
 NOD_Names = vector("10.10.10.10", "10.10.10.11"),-
 NOD_Addresses = vector(210, 211)
;
;Gateway Nodes
;
#local GW_NOD_Numbers = vector(1,2),-
 GW_NOD_Addresses = vector(201,202)

.
.
;
;External Applications
;
#local Ext_Apl_Numbers = vector(5),- ;for example vector(15,16),-
 Ext_Apl_NA = vector("APL5"),-
 Ext_Apl_ND = vector(11),-
 Ext_Apl_SN = vector(0),- ;Shadowing partner application
 Ext_Apl_TN = vector(5),-
 Ext_Apl_EM = 5000,- ;Maximum length of mirroring queue
 Ext_APl_EP = "KEEP" ;Event queue overflow policy, "DISCARD" or
          "KEEP"
```

4.10.3 Configuring Base system 2

```
.
#local LAN_Link = 1 ;LAN link number
.
.
;
;Other Base System Nodes
;
#local NOD_Numbers = vector(9),-
 NOD_Names = vector("10.10.10.9"),-
 NOD_Addresses = vector(209)
;
;Gateway Nodes
;
#local GW_NOD_Numbers = vector(),-
GW_NOD_Addresses = vector()
. . ; ;External Applications
;
#local Ext_Apl_Numbers = vector(1),-
;for example vector(15,16),-
 Ext_Apl_NA = vector("APL1"),-
 Ext_Apl_ND = vector(9),-
 Ext_Apl_SN = vector(0),- ;Shadowing partner application
 Ext_Apl_TN = vector(1),-
 Ext_Apl_EM = 5000,- ;Maximum length of mirroring queue
 Ext_APl_EP = "KEEP" ;Event queue overflow policy, "DISCARD" or
          "KEEP"
```

4.10.4 Configuration of Front-end system

```
.
#local LAN_Link = 1 ;LAN link number
.
.
;
;Other Base System Nodes
;
#local NOD_Numbers = vector(9),-
 NOD_Names = vector("10.10.10.9"),-
 NOD_Addresses = vector(209)
;
;Gateway Nodes
;
#local GW_NOD_Numbers = vector(),-
 GW_NOD_Addresses = vector()
.
.
;
;External Applications
;
#local Ext_Apl_Numbers = vector(1),-
;for example vector(15,16),-
 Ext_Apl_NA = vector("APL1"),-
 Ext_Apl_ND = vector(9),-
 Ext_Apl_SN = vector(0),- ;Shadowing partner application
 Ext_Apl_TN = vector(1),-
 Ext_Apl_EM = 5000,- ;Maximum length of mirroring queue
 Ext_APl_EP = "KEEP" ;Event queue overflow policy, "DISCARD" or
         "KEEP"
```

4.10.5 Encrypted Communication

Communication between SYS600 9.4 and later is encrypted. Each node has an identity. Identity is a 64 character hexadecimal string. When a connection between nodes is established nodes sends their identities to each other. The connection is accepted if the identity matches the identity stored for the node. The identity is stored in the NODn:BID (Node's Identity) attribute. If the identity of the connecting node doesn't match the identity stored in the NODn:BID the new identity is stored in the NODn:BPI (Node's Pending Identity) attribute. The pending identity can be copied to node's identify using Accept Identify button in the Base System Object Navigator tool.

If the NODn:BID is empty, the new identity can be accepted automatically. This is done by setting REQUIRE_KNOWN_ACP_CERTIFICATE to FALSE.

When the system is set up, the REQUIRE_KNOWN_ACP_CERTIFICATE should be set to TRUE. This makes sure that hostile clients can't connect to the system.

Identities are stored in SYS\ACTIVE\SYS_\KEYS\KEYS file.

4.10.5.1 Compatibility with 9.3 FP3 and earlier

By default the system doesn't accept unencrypted communication from the network. To allow unencrypted communication set the REQUIRE_ENCRYPTED_ACP hardening switch to NONE. If this setting is applied, make sure that the firewall only accepts connections to port 21845 from addresses where the version with older systems are.

Earlier it was possible that the networking configuration was not symmetric. Newer versions requires that in a connection between systems both must have node configured for the other system.

4.10.6 Configuring Local Area Networks (LAN)

To connect a base system to a LAN, create a LINn:B object with the following attributes (for more information, see SYS600 System Objects):

LT = LAN

TR = TCPIP

All workplaces and base systems can use the same LIN object.

4.10.6.1 LAN nodes

In the LAN network, each connected base system and workplace has a LAN node name or number. The LAN node names are used in the SYS600 configuration to achieve communication between the base systems, between the base systems and the LAN connected devices. Use static IP addressing, which indicates that the computer is manually configured to use a specific IP address since using DHCP for IP assignment is not verified. For more information about Base System Configuration, see Figure 38:

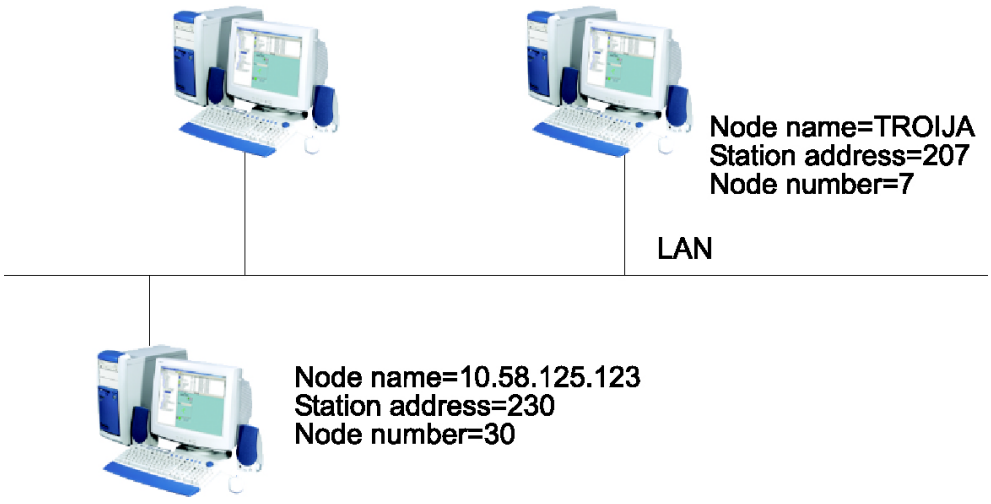


Figure 38: Example of a SYS600 configuration for a LAN

```
.
#local LAN_Link = 1                        ;LAN link number
.
.
;
;Other Base System Nodes
;
#local NOD_Numbers = vector(7,30),-
       NOD_Names = vector("TROIJA", "10.58.125.123"),-
       NOD_Addresses = vector(207,230)
```

4.10.7 Communicating between applications

The object data in one application can be read and written from another application with object notations. This communication is called APL-APL communication.

Communication between applications in the same base system (i.e. between two local applications) is achieved by application mapping (the APLn:BAP attribute).

Communication between applications in separate base systems requires that the base systems are physically connected to each other either through LAN or through direct serial lines. The configuration and communication principles are the same regardless of the route between the base systems. The communicating base systems are identified to each other by node numbers and station addresses as well as the link to the nearest node. The route through the network does not need to be defined.

4.10.7.1 Local applications

Example:

Suppose that application 'a' needs to read and write data in application 'b' in the same base system, as shown in Figure 39. Application 'b' should then be introduced to application 'a' by means of application mapping (For more information, see SYS600 System Objects):

```
#SET APLa:BAPi = b
```

where:

'i' The logical application number under which application 'a' recognizes application 'b'

To avoid complexity, it is recommended to let the logical number be the same as the object number of the application, that is 'i' = 'b'. For example, setting #SET APL1:BAP2 == 2 means that APL2 is recognized to APL1 by the logical application number 2. In application 1 it is possible to read object data in application 2, for example with the notation: OBJ:2POV1.

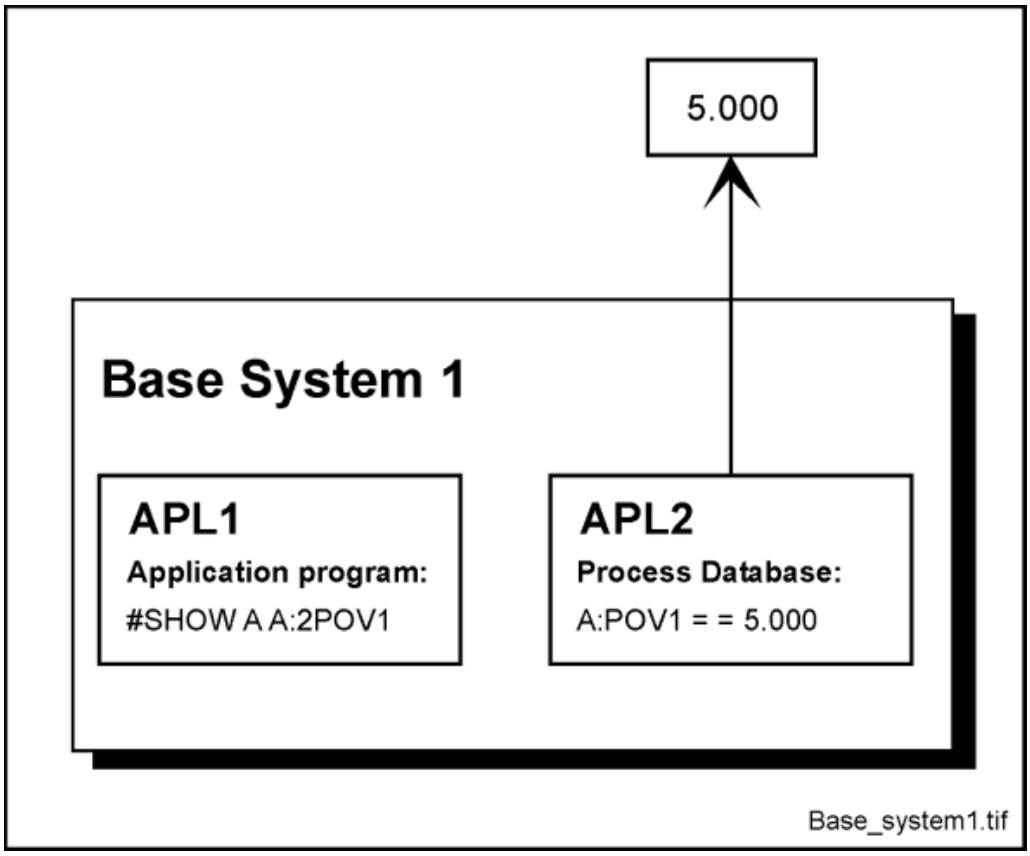


Figure 39: Illustration of the data communication between applications in the same base system

4.10.7.2 Applications in separate base systems

Example:

Suppose that application 'a' in base system 1 needs to read and write data in application 'b' in base system 2. The following configurations are required in base system 1:

1. Create a LINn:B object for the link to the base system 2 (if it does not already exist).
2. Create a NODn:B object representing base system 2, where 'n' is the node number of base system 2.

The NODn:B object should be assigned at least the following attributes:

LI	The number of the link to base system 2 (the LINn:B object number, see above)
SA	Station address of base system 2

In addition, if LAN is used:

NN	LAN node name of base system 2 (see Section 4.10.6).

3. Create an external application, an APLn:B object, referring to application 'b' in base system 2. For clarity, use the same object number ('n') as the application object number in base system 2 created in APLb:B. Assign the APLb:B object the following attributes:

TT	EXTERNAL
ND	Node number of base system 2
TN	Application object number in base system 2 ('b')

4. Map the external application in base system 1 to the communicating application 'a', by setting APLa:BAPi = b, where 'i' is the logical application number with which application 'a' recognizes application 'b'. If there are no obstacles, let the logical number be the same as the object number of the application (that is 'i' = 'b').

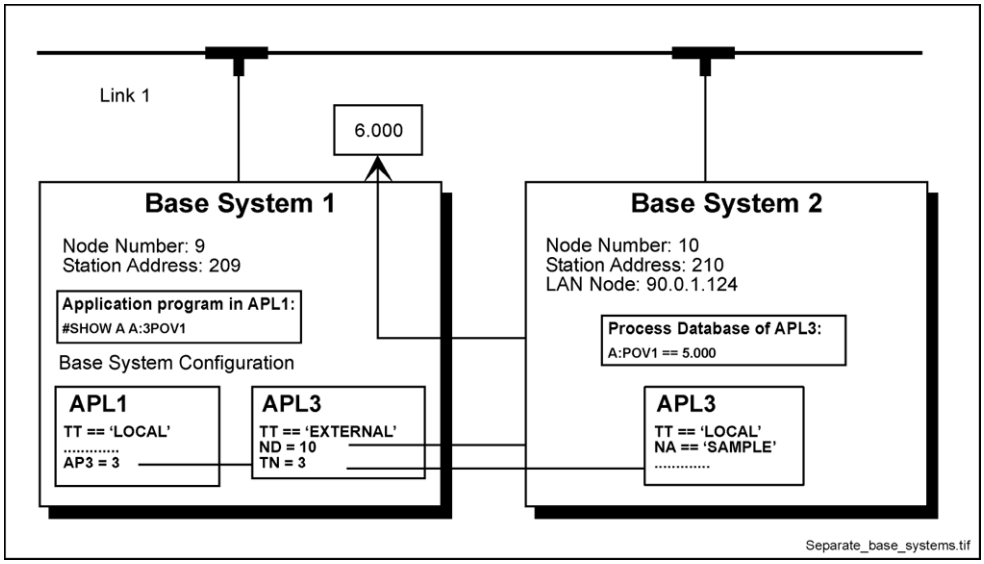


Figure 40: Illustration of the configuration and data communication between two applications situated in separate base systems

LAN link configuration
Refer to Figure 40.

```
;LAN link:
#CREATE LIN1:B=LIST(-
         LT="LAN",-
         TR="TCPIP")
;Node for Basesystem 2:
#CREATE NOD10:B=LIST(-
         LI=1,-
         SA=210,-
         NN="90.0.1.124")
;Application 1:
#CREATE APL1:B=LIST(-
         AP3=3)
;Application 3:
#CREATE APL3:B=LIST(-
        TT="EXTERNAL",-
        ND=10,-
       TN=3)
```

4.10.8 LAN Teaming Procedure

This section describes the procedure for making a LAN Teaming of network adapters. Intel® LAN teaming feature offers many various modes, such as AFT, SFT, ALB, RLB, SLA and IEEE 802.3ad DLA.

Intel® LAN teaming is not supported in all Windows versions, e.g. in Windows Server 2016. Ensure the current situation for the required OS and adapter type from the Intel® website. Microsoft Windows NIC teaming may be considered as an alternative to Intel® LAN teaming.

4.10.8.1 Teaming Modes

4.10.8.2 Adapter Fault Tolerance

Adapter fault tolerance protects the system against failure in cable, failure in switch port connected to an adapter, and adapter fault. Adapter fault tolerance provides more than one backup standby link which supports up to 8 adapters in the teaming link. In case of active link failure, the backup standby link takes it over. In case of a failure of an active link, the primary adapter will transfer the MAC address and the layer-3 address (IP Address) to a failover secondary adapter (standby link). AFT is preferred in the small substation applications with only a few connected IEDs.

AFT Requirements:

- All adapters should be connected to the same hub or switch.
- Disable STP (Spanning Tree Protocol) in the switch connected to teamed adapters.
- All teamed adapters must be in the same subnet.

Example of configuration:

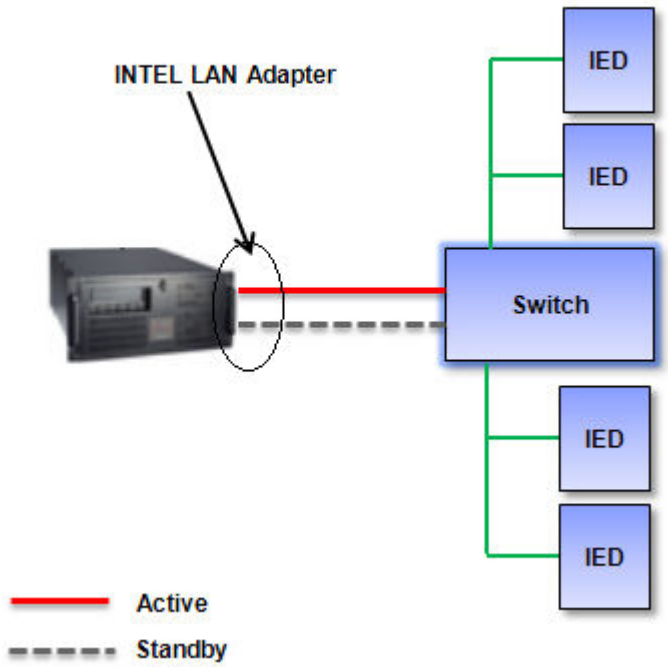


Figure 41: AFT Network Configuration for the small substation application.

4.10.8.3 Switch Fault Tolerance

The Switch Fault Tolerance feature of the Intel® network adapter provides redundancy across the separate switches. It supports a maximum of two adaptors in the team. In this configuration, one port will remain active and the other in the standby mode. When the active

port loses the link, the secondary port backup standby link gets activated. SFT is preferred in the medium/large substation application.

SFT Requirements:

- Both adapters should be connected to the separate switches.
- One switch is redundant so that when the active port fails, all traffic is forwarded via the redundant switch. STP (Spanning Tree Protocol) should be enabled in all of the switches connected to the network to ensure that loops are avoided. However, switch ports connected to the adapter should be configured as Edge Port or Fast Port.
- Do not put clients to link partner switches because they will not pass traffic to the partner switch if failure occurs.
- All teamed adapters must be in the same subnet.

Example of configuration:

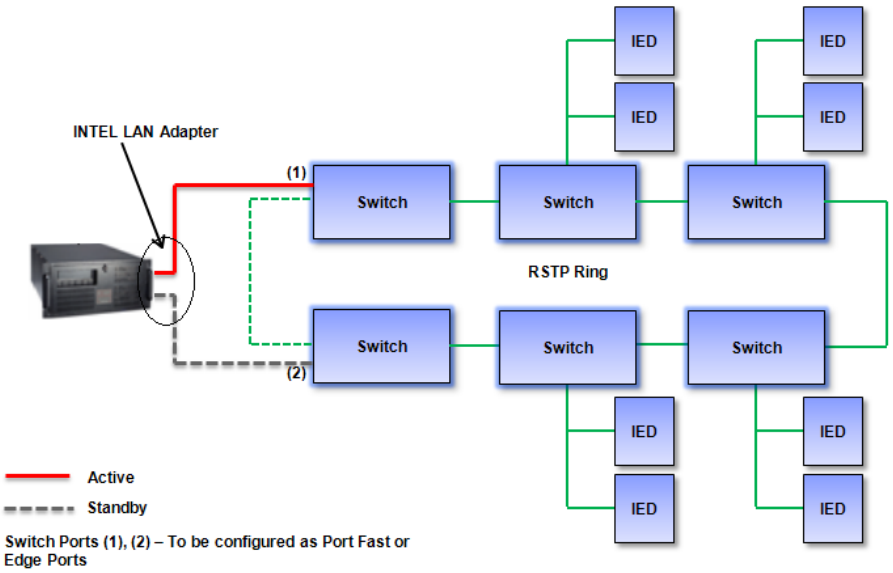


Figure 42: SFT Network Configuration for the medium/large substation application.

A network may be expanded to accommodate a larger number of SYS600C to link partner switches.

4.10.8.4 Configuring Adapter Drivers for Teaming feature

To configure the Intel® Teaming Feature:

1. Intel® LAN Drivers must be installed in the system.
2. Open **Control Panel**.
3. Select **Device Manager**.
4. Expand **Network adapters** and double-click on **Intel(R) PRO/1000 PT (PF) Dual Port Server Adaptor**.
5. The **Properties** windows appear. Select the **Teaming** tab.
6. Select **Team this adapter with other adapters** and click **New Team**.
7. The **New Team wizard** dialog box appears. Specify a name for the team, for example Team #1, and click **Next**.

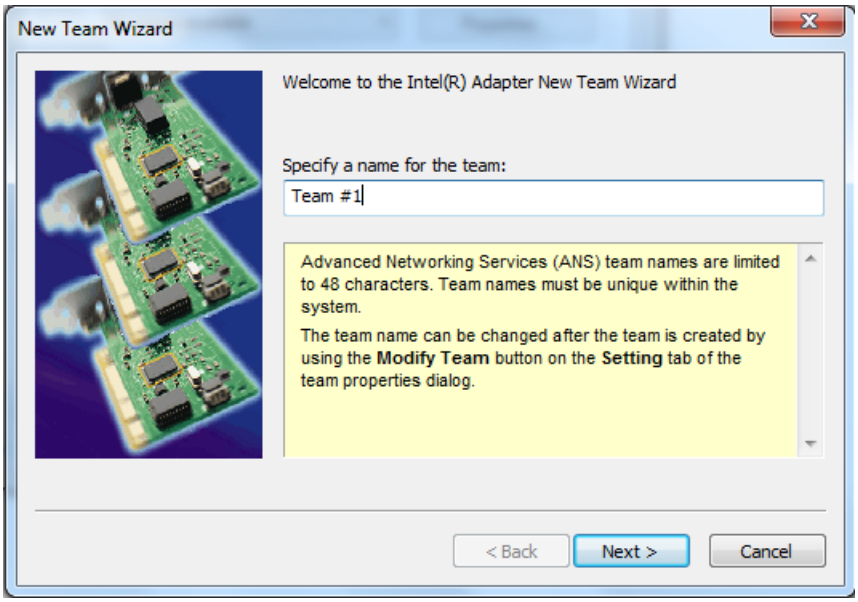


Figure 43: Naming of the team virtual adaptor

8. Select both **Intel(R) PRO/1000 PT Dual Port Server Adapters**. Click **Next**.

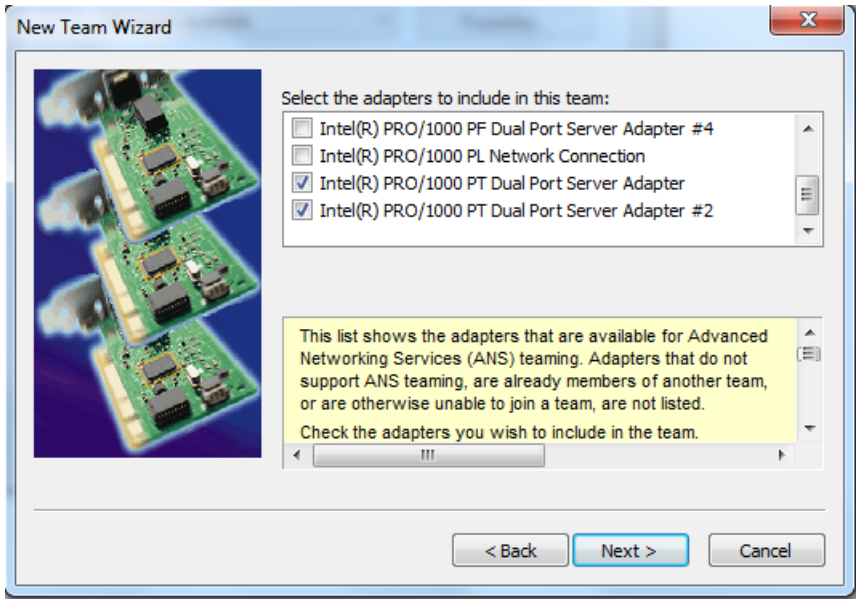


Figure 44: Selection of the team member adapters

9. Select either **Switch Fault Tolerance** or **Adaptor Fault Tolerance** depending on the required team mode and click **Next**.

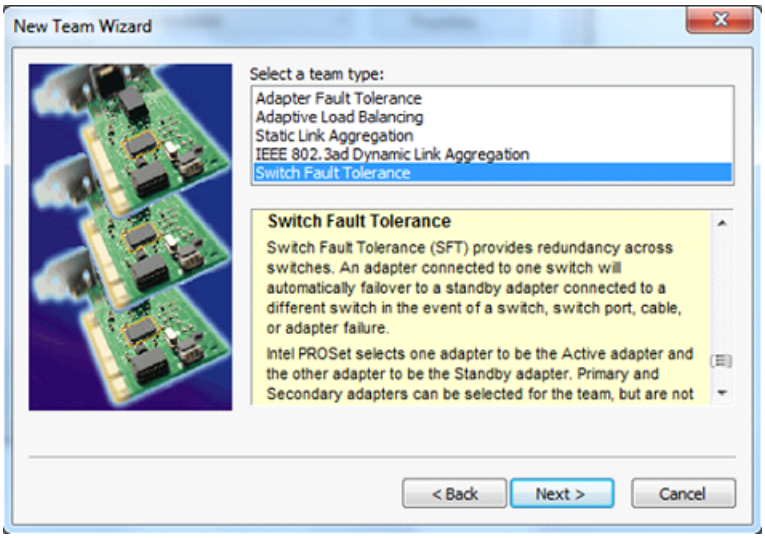


Figure 45: Selection of the teaming mode

10. Click **OK** to finish driver configuration.
11. Click **OK** to close the **Properties** dialogs.
12. The Team is now configured.

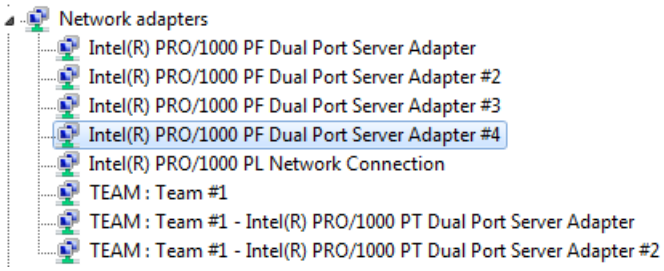


Figure 46: Team #1 Virtual Adapter has been created

13. Virtual adaptor named TEAM: Team #1 appears in the adaptor list.

For further information about Intel® LAN Teaming features, refer documentation on corresponding website.

4.11 Configuring redundancy

4.11.1 Hot stand-by base systems

In a hot stand-by base system, two base system computers are interconnected via a LAN in a redundant relationship, where one or both base systems are prepared for fast takeover at system break-down in the other base system. An application in one base system operates as the hot application, while an identical application in the other base system is a stand-by application. The stand-by application is maintained by a continuous shadowing (copying) of data from the hot application.

When a fault occurs in the primary base system (the base system containing the hot application), the shadowing application in the stand-by base system is started and takes over all the operational functions. After recovery and restart of the former primary base system, it can either be used as a stand-by base system, whereby the former stand-by base system is the primary base system, or the base systems can be returned to their original tasks.

During normal operation, the two base systems can function independently, each running one or more applications, for example, electrical energy distribution and district heating. Alternatively, one base system can be reserved exclusively for stand-by duty. Both base systems can contain several applications connected with an application in the other base system in a shadowing relationship. In the following description, it is assumed that the base systems contain only one shadowing application pair, but the same principles apply to systems with several shadowing applications.

4.11.1.1 Functional description

During normal operation, the running application in the primary base system continuously sends shadowing data to an identical application in the stand-by base system. Shadowing means that the following data is copied from the running application to the stand-by application:

- All changes of files, including deletions, on disk under the application subdirectories (APL_, PICT, FORM, etc.). The files in the application root directory are not copied.
- Changes of application data (process and report data) stored in RAM. Other changes, such as changes of cache memories, monitor states, printer spool and execution queues, are not copied.

Neither the SYS_ folder nor any other folder outside the application directory is shadowed. Therefore, if customizations are done in these folders, they are lost at the takeover.

File management done outside the SYS600 program is not shadowed. Therefore, always use SYS600 tools instead of operating system utilities for file management (copy, delete, rename, and so on).

An update of an object constitutes a transaction. A transaction consists of atoms, which are the smallest quantity of data modified in RAM. The primary base system collects the transactions in a buffer. The data in the buffer is transmitted to the standby system as long messages (64 kB). A transmission is performed when the oldest transaction in the buffer has been there for a preselected time (the Shadowing Flush Time (SF) attribute, default = 100 ms), or when the buffer is full. This means that data is transmitted more often in situations when the databases are updated frequently.

Besides the shadowing data messages, the hot application cyclically sends diagnostic commands and time synchronization commands to the stand-by application. If it does not receive an acknowledgement to the messages, the connection to the stand-by application is regarded as broken, and the shadowing halts until the connection is re-established.

The watchdog application in the stand-by base system monitors the diagnostic commands and messages from the hot application. It starts an event channel if message or diagnostic commands are not received within a specified time. The event channel starts a command procedure, which examines the situation and performs a switch-over if needed.

Switch-over means that the former stand-by application is switched to hot application. When the stand-by application is set to HOT, an event channel APL_INIT_H is started (instead of APL_INIT_1 and 2). The event channel may be used, for example, for reconfigurations and updating. Apart from an ordinary application start-up, no process data is copied from disk to RAM. The watchdog application in the new primary base system tries to connect to the former hot application (which is now regarded as stand-by application) cyclically with a specified time interval. At switch-over, the workstation programs (Monitor Pro and Classic Monitor) must be restarted by application programs or manually.

When the shadowing communication between the hot and stand-by application is established, the stand-by application sends a list of all files under the stand-by application subdirectories to the hot application. The list contains the name, creation time, modified time and size of

each file. The hot application makes a similar list of its own files. Comparing the two lists produces a list of actions to be taken in order to get the files of the stand-by-application identical to the files of the hot application. Files that exist only in the hot application are copied to the stand-by application, files that exist only in the stand-by application are deleted and files that are different are rewritten to the stand-by application. Identical files are just preserved. Two files are considered identical, if their name, creation time, modified time and size are identical. Executing the action list is called 'file dump'. Likewise, all application data of the hot application stored in RAM is copied to the stand-by application ('RAM dump'). While the RAM data is copied, which may take some seconds depending on the application, the running application is out of operation. A new switch-over may be obtained manually by a simulated error, for example, by setting the primary main application to cold.

4.11.1.2 Configuring hot stand-by systems

Minimum configuration requirements are:

- Two complete base systems connected to a LAN, each including at least two applications: one main application, which is a part in the hot stand-by relation, and one watchdog application which is dedicated for monitoring the main application and performing a switch over when needed.
- A local area network (LAN), TCP/IP.
- A standard watchdog application software package in each base system. The watchdog software package contains command procedures and data objects for monitoring the operation and reconfiguring at switch-over.

Options:

- Additional applications in both base systems.
- Operator workplaces.

The following procedure describes the steps for configuring a hot stand-by base system:

1. Install the MicroSCADA X Technology software for both base systems as described in the Installation and Commissioning Manual.
2. Edit SYS_BASCON$COM template and rename it to SYS_BASCON.COM for both base systems.
3. Start base system that should have the hot application.
4. Install standard watchdog application software.
5. Define external watchdog application and start the main application.
6. Start stand-by base system.
7. Install standard watchdog application software in the stand-by base system.
8. Define external watchdog application software in the stand-by base system.
9. Edit command procedures in the watchdog applications for both base systems.

4.11.1.3 SYS_BASCON.COM in hot stand-by systems

The MicroSCADA X software delivery includes a version of SYS_BASCON.COM template, SYS_BASCON$COM, which contains all the necessary configuration definitions for a hot stand-by, too. The file is listed in Appendix A.1.1.

To turn on the hot stand-by functionality in SYS_BASCON.COM, change the value of the variable Hot_Standby to TRUE in the beginning of the file.

```
#local Hot_Standby = TRUE ;Hot Stand-by enabled/disabled
```

Except for the node numbers, the SYS_BASCON.COM files of both base systems can be identical.

```
#local This_Node_is = BS_Nodes(1) ;This system 1 or 2 (Always 1 for
single system)
```

The standard base system configuration defines that both main applications are COLD when the base systems are started, only the watchdog applications are running.

The principles for the initial configuration of a hot stand-by base system in SYS_BASCON.COM are shown in Figure 47.

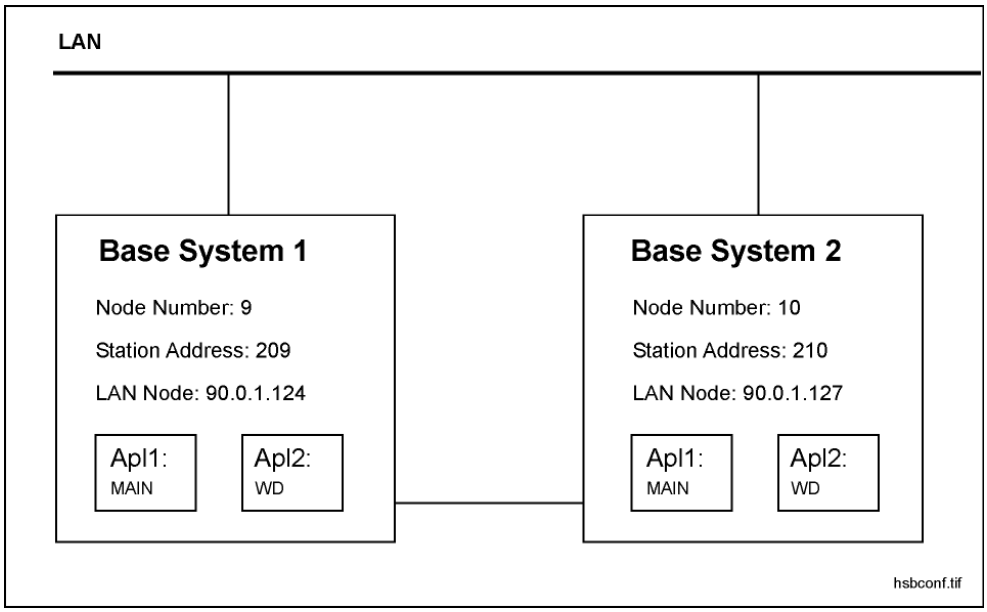


Figure 47: Example of two redundant base systems

The example in following Table 8 illustrates only the attributes and parameters that are significant for hot stand-by. A significant definition from the WebUI HSB discovery point of view is the BS_Web_Addresses in SYS_BASCON.COM.

```
#local BS_Web_Addresses = vector("90.0.1.124","90.0.1.127") ;Base System
Web Addresses
```

Based on the definition above the base system Web Address WA of base system 1 will be set to "90.0.1.124" and the base system Web Address of base system 2 will be set to "90.0.1.127" in SYS_BASCON.COM. Respectively the Shadowing Partner Web Address SA of the shadowing application in base system 1 will be set to "90.0.1.127" and the Shadowing Partner Web Address of the shadowing application in base system 2 will be set to "90.0.1.124" in SYS_BASCON.COM.

For more information on the shadowing attributes and other application attributes, see SYS600 System Objects manual.

Table 8: Start-up configurations for two redundant base systems

Configuration of base system 1:	Configuration of base system 2:
Base system: • SH = 1 • WA = "90.0.1.124"	Base system: • SH = 1 • WA = "90.0.1.127"
Application 1 (internal): • NA = "MAIN" • AS = "COLD" • SA = "90.0.1.127" • SN = 3 • SR = 1 • SI = 100 • SW = 2 • SY = 0 • AP = (1,2,3,4)	Application 1 (internal): • NA = "MAIN" • AS = "COLD" • SA = "90.0.1.124" • SN = 3 • SR = 1 • SI = 100 • SW = 2 • SY = 0 • AP = (1,2,3,4)
Application 2 (internal, default application): • NA = "WD" • AS = "HOT"	Application 2 (internal, default application): • NA = "WD" • AS = "HOT"
Application 3 (external): • NA = "ADJ_MAIN" • TT = "EXTERNAL" • ND = 10 • TN = 1	Application 3 (external): • NA = "ADJ_MAIN" • TT = "EXTERNAL" • ND = 9 • TN = 1
Application 4 (external): • NA = "ADJ_WD" • TT = "EXTERNAL" • ND = 10 • TN = 2	Application 4 (external): • NA = "ADJ_WD" • TT = "EXTERNAL" • ND = 9 • TN = 2

4.11.1.4 Watchdog application

The watchdog application software package handles the following procedures for all hot stand-by applications within a base system:

- When the base system is started, the application checks which main application was operating last and sets the application state (AS) to HOT and the shadowing state (SS) to HOT_SEND.
- During the operation, the application monitors the messages sent from the hot application. If no messages are received in a specified time defined by the Shadowing Receive Timeout (SR) attribute, a switchover is started and the stand-by application is set to HOT and shadowing is started (SS = HOT_SEND) when the connection is re-established.
- If the hot system does not get acknowledgments from the stand-by system, it regards the connection as broken and the shadowing stops (SS = NONE). The watchdog application then checks the connection by sending commands cyclically (with a few minutes interval) to the stand-by system, and starts shadowing (SS = HOT_SEND) when the connection is re-established.

Installing Watchdog application
To install the watchdog application package:

1. Enter the **Base System Object Navigator** on the **System Configuration** page of the **Tool Manager**.
2. Select **Base Objects (SYS)** from the object tree.
3. Click the **Tools** menu and click the **HSB Management...** item to open the **Shadowing Object Management** dialog. The following information is displayed.
   - Installed Package
     Version: The name and revision date of a previously installed package.
     Status: The status of the installed package, running or not running.
   - Disk Package
     Version: The hot stand-by software package installed on a disk.
     Status: The status of the disk package (OK, DISK PACKAGE NOT FOUND or FILE READ ERROR (SHAD_VERS.CIN): error number).
4. Click **Install** to install the watchdog software package. The installation creates a set of command procedures, data objects and time channels. When the installation is complete, the name and revision date of the package appears in the **Installed Package Version** field.

Editing the command procedures of the watchdog application
The watchdog application package contains a set of command procedures. The following command procedures can be freely customized, whereas the others should not be edited.

While editing the command procedures, the existing part of the contents should be left as it is and the modifications are added to the end of the command procedure.

SHADUSR	The generation of alarms and events in the situations when: • Hot stand-by transmission starts • File and RAM dump is ready • Connection is lost to the receiver (in the stand-by system) • Takeover starts • Change of state occurs in the partner application
SHADMAPMON	The shifting of classic monitors at takeover, for example, mapping monitors for the main application, or opening application windows. For more information on how to open application windows, see SYS600 Installation and Administration Manual.
SHADMAPNET	Reconfiguration of the communication units at takeover. This is currently not the primary method to use for the reconfiguration. In case the PC-NET process communication units are distributed, the redefinition of the APL objects of the PC-NET units should be done using the SY attribute of the PC-NET node. This can be done by editing the SHADMAPNET procedure or from a separate procedure connected to the APL_INIT_H event channel of the main application.
SHADGOHOT	Specifies whether the main application is allowed to be set HOT when a lost connection has been discovered. The command procedure can contain a check of the error, for example, if the communication disturbance is due to a communication fault on the LAN connection to the stand-by system. Then no switchover should be performed. By default, the application is set to HOT.
SHADREMHOT	Specifies whether the main application is allowed to remain HOT when also the standby application is HOT. Such situation can occur at a LAN break. By default, the application remains HOT.

4.11.1.5 Starting shadowing

To define the external watchdog application and enable hot stand-by:

1. In the Shadowing Object Management dialog of the Base System Object Navigator, click the page **Shadowing Applications**. A list of HSB applications is shown in the dialog.
2. Click the row of the local main application to be modified and click the **Edit** button.
3. Check the **HSB** checkbox to set the HSB On.
4. Select the external watchdog application in the drop-down list and click **OK** (or **Apply** and **Cancel**).
5. Repeat steps 2-4 for each main application.

File dump and shadowing will start after the HSB has been enabled for the main applications in both base systems.

A takeover should not be attempted before the file dump is completed.

The file dump of an application is completed when the shadowing state attribute SP = HOT_SD.

4.11.1.6 Configuring proxy applications

The system configuration file SYS_BASCON.COM contains SCIL statements for creating proxy applications. All the engineer has to do is to edit the proxy statements to enter the numbers, names and the numbers of the two true applications represented by the proxy. The statements are listed in the beginning of the file SYS_BASCON.COM in the variable definition part of the file. The example below shows a configuration of two proxy applications. The definition will create proxy applications 101 and 102. The name the proxy application 101 will be PRO_1 and it will represent true applications 13 and 15. Respectively, the name the proxy application 102 will be PRO_2 and it will represent true applications 14 and 16.

```
;
;Proxy Applications
;
#local Proxy_Apl_Numbers = vector(101,102),- ;for example
                  vector(101,102),-
Proxy_Apl_NA = vector("PRO_1","PRO_2"),-
Proxy_Apl_HS = vector(vector(13,15),vector(14,16))
```

4.11.2 Hot stand-by with OPC client and servers

The principles of the HSB concept in systems including the OPC client and servers connected to IEC 61850 process devices is described in the SYS600 IEC 61850 System Design manual. The following sections describe the recommended configuration sequences for External OPC Data Access Client start-up and how the related stations should become activated for the communication during the switch over.

4.11.2.1 Starting an External OPC Data Access Client

External OPC DA Client should be started from the watchdog (WD) application. The recommendation for the HSB systems is that all communication components, for example, External OPC DA Client, PC-NET related protocols or any communication protocol based on the Communication Protocol Interface (CPI) should be started in this way. In practice, this startup logic is included in the command procedure triggered from APL_INIT_1 event channel. An example snapshot is as follows:

```
;APL_INIT_1:C in HSB systems for WD application
...
#do STOP_OPC_DA_CLIENT_INSTANCE:C  ;Stop previous External OPC DA Client
instance
```

```
#exec_after 10 START_OPC_DA_CLIENT_INSTANCE:C ;Start OPC client after a
delay
```

The application logic for handling External OPC Data Access client stopping and starting has been included in a separate command procedure of the WD application. The reason for stopping the External OPC Data Access client is related to a situation when MicroSCADA X SYS600 may have been stopped abnormally due to, for example, computer failure. In these circumstances, the standard routines related to the shutdown of SYS600 base system's SHUTDOWN.CIN execution or stopping the application (APL_CLOSE event channel execution) has been handled normally.

An example of these command procedures is as follows:

```
;STOP_OPC_DA_CLIENT_INSTANCE:C in HSB systems for WD application
...
@opc_status = ops_call("C:\sc\prog\OPC_Client\DA_Client\daopccl.exe -id
conf -stop", 0)
;START_OPC_DA_CLIENT_INSTANCE:C in HSB systems for WD application
...
@opc_status = ops_call("C:\sc\prog\OPC_Client\DA_Client\daopccl.exe -id
conf -start C:\sc\sys\active\sys_\config.ini", 0)
```

For detailed information about the usage of DAOPCCL.EXE and its parameters, see SYS600 External OPC Data Access Client.

4.11.2.2 Activating station communication

Before the OPC item updates are propagated to the MicroSCADA X SYS600 process database, the communication station(s) included in the External OPC DA Client configuration need to be activated. In a hot stand-by system, this should be done by the main application. Typically this application logic is included in the command procedure triggered from APL_INIT_1 and APL_INIT_H event channels.

```
;APL_INIT_1:C / APL_INIT_H:C in HSB systems for MAIN application
...
#exec STAS_IN_USE:C
```

The station communication can be activated as described in the following command procedure:

```
;STAS_IN_USE:C in HSB systems for MAIN application
...
#error continue
@i_Sta_Numbers = (81,82,83,84,85)
#loop_with lc = 1..length(%i_Sta_Numbers)
    @i_Sta_Number = %i_Sta_Numbers(%lc)
    #set sta'i_Sta_Number':sIU = 1     ;station in use
    #set sta'i_Sta_Number':sUP = 1     ;update points
#loop_end
```

4.11.3 Hot stand-by with PC-NET

In communication protocols supported by the PC-NET process communication unit, the switch-over situation in HSB concept requires SCIL programmable actions. This will guarantee the communication to continue in the HOT system without interruptions. When the system goes into stand-by mode, it deactivates the communication to stations and, in case of slave protocols, to NCCs. Correspondingly, when the system goes to HOT state, the communication to the stations and NCCs is activated. The PC-NET processes are running all the time, including when system is in stand-by mode.

4.11.3.1 PC-NET configuration

The PC-NET configuration made with the System Configuration Tool should be entered from the watchdog (WD) application. With this approach, the configured PC-NET instances are automatically started when MicroSCADA is started and the PC-NET instances are already configured when the MAIN application goes hot in all situations.

When the system configuration is entered from the WD application, the AS and MS attributes of the STA objects created for the PC-NET will also refer to the WD application. The AS attribute defines the application that will receive process data messages and the MS defines the application where the system messages from the station object will be sent for. In order to map the incoming process data and the status messages to the main application, the following configuration should be entered to the configuration of the NET node. In this example, the number of the main application is 1 and the number of the WD application is 2.

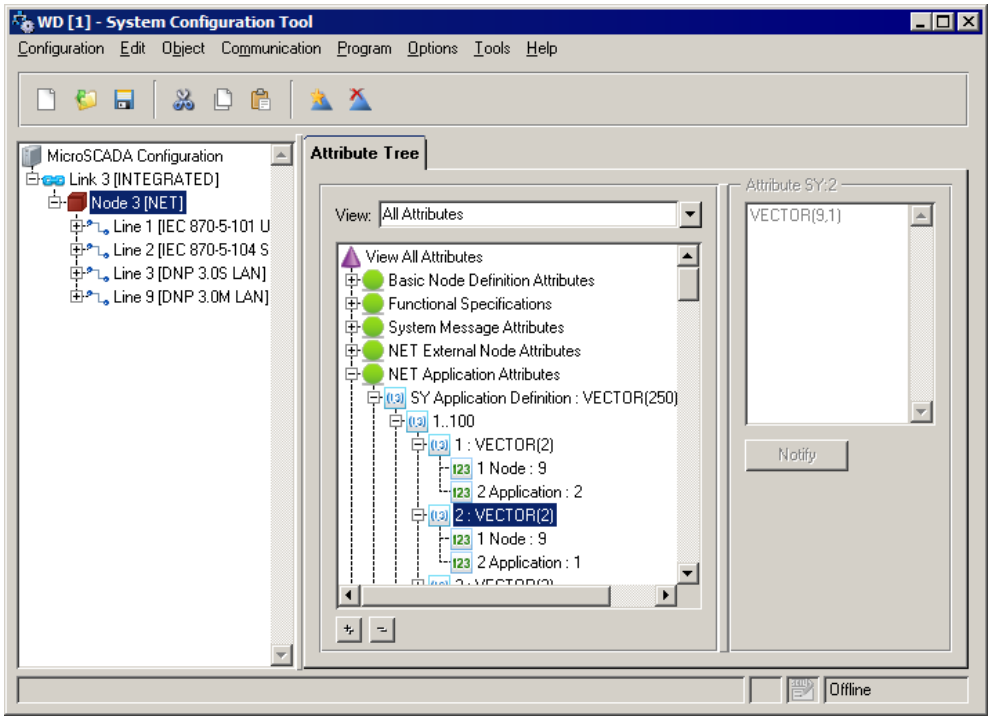


Figure 48: System Configuration Tool window

An alternative method is to add the following script to the User-Defined program of the NET node.

```
@MAIN_APL = 1
@WD_APL = 2

#SET NET'i_Net_Number':SSY'WD_APL'= (SYS:BND, %MAIN_APL)
#SET NET'i_Net_Number':SSY'MAIN_APL'= (SYS:BND, %WD_APL)
```

This script should be present in HSB configuration, otherwise the process data will be sent to the WD application. Furthermore, all lines created to PC-NET process communication units should be configured to have IU=0, which means that all lines should be out of use by default.

4.11.3.2 Activating communication

The communication to the station and the NCCs should be activated when the state of the main application turns to HOT. In practice, a command procedure attached to event channels APL_INIT_1 and APL_INIT_H of the main application should loop all lines created to PC-NET nodes and take the line into use.

The LON protocol (if created for the system) requires special handling. With LON protocol, the stations objects should be taken into use after the line has been taken into use. In practice, this means that all the mentioned procedure should contain a block.

```
#SET NET3:SIU1 = 1         ;LON line 1 in NET3
#SET STA20:SIU = 1         ;REX station connected to LON line 1
#SET STA21:SIU = 1         ;REX station connected to LON line 1
.
#SET STA80:SIU = 1         ;LMK station connected to LON line 1
```

No other station types except for REX and LMK need to be taken out of use and back to use when the communication activation and deactivation is required.

The event channel APL_INIT_1 in the main application is executed when the system is started and the state of the main application is HOT immediately after startup. APL_INIT_1 is not executed in normal take-over. The event channel APL_INIT_H is activated when the take-over occurs and the state of the main application changes from COLD to HOT. For more information about the predefined event channels APL_INIT_1 and APL_INIT_H, see SYS600 Application Objects.

With the COM500*i* application, the communication activation procedure should be executed before the COM_COMINI_H and COM_COMINI procedures in APL_INIT_H and APL_INIT_1.

4.11.3.3 Deactivating communication

When the main application turns to COLD state, it is necessary to deactivate all communication. The deactivation is made from command procedure connected to the event channel APL_CLOSE of the main application. The behavior is opposite compared with the activation presented above.

Like in the activation, the LON protocol (if created to the system) requires special handling. In this case, it is required that the stations objects should be taken out of use before the line has been taken into use.

```
#SET STA20:SIU = 0     ;REX station connected to LON line 1
#SET STA21:SIU = 0     ;REX station connected to LON line 1
.
#SET STA80:SIU = 0     ;LMK station connected to LON line 1
#SET NET3:SIU1 = 0     ;LON line 1 in NET3
```

No other station types except for REX and LMK need to be taken out of use and back to use when the communication activation and deactivation are required.

4.11.4 Hot stand-by with Modbus slave

In the HSB setup with modbus slave, the basic idea is that when the system goes to stand-by state, the communication is deactivated by killing the instances of MODBUS_SLAVE.EXE. When the system is started or the main application goes from COLD to HOT state, the communication is activated by starting the modbus slave instances.

This approach is different from the description in Modbus slave protocol manual of MicroSCADA. However, shutting down the instance is required in hot stand-by setup, in order to control the fallback switches of the NCC line correctly.

4.11.4.1 Modbus slave configuration

This example should be applied to required amount of Modbus slave lines, that is, NCC connections from COM500*i*. The following steps are required to configure 3 modbus slave lines:

1. Create the following directories:
   - \sc\prog\modbus_slave\s1
   - \sc\prog\modbus_slave\s2
   - \sc\prog\modbus_slave\s3
2. Copy MODBUS_SLAVE.EXE and CONFIG.INI to each of these created directories.
3. Rename the MODBUS_SLAVE.EXE to MDS1.EXE, MDS2.EXE and MDS3.EXE.

   The renaming is needed in order to shut down the instances in a controlled way. If there is only one instance of the modbus slave, the renaming is not necessary and the shutting down (Step 6) can be done using the name MODBUS_SLAVE.EXE.
4. Edit the CONFIG.INI in each directory and make the corresponding configuration to the base system, as instructed in Modbus Slave Protocol manual of MicroSCADA. The number of the main application should be given to the configuration item application_number.
5. Create a BAT-file MDS1.BAT with the following contents:

   ```
   cd C:\sc\prog\modbus_slave\s1 mds1
   ```

   Create a similar BAT-file for each instance to their own subdirectory.
6. Add the following script to a code procedure executed from event channels APL_INIT_1 and APL_INIT_H of the main application.

   ```
   ;MD_SLAVE_START:C
   @MDS1_STATUS = OPS_CALL("C:\sc\prog\modbus_slave\s1\MDS1.BAT",0)
   @MDS2_STATUS = OPS_CALL("C:\sc\prog\modbus_slave\s2\MDS2.BAT",0)
   @MDS3_STATUS = OPS_CALL("C:\sc\prog\modbus_slave\s3\MDS3.BAT",0)
   ```
7. Add the following script to a code procedure executed from event channel APL_CLOSE of the main application.

   ```
   ;MD_SLAVE_STOP:C
   @MDS1_STATUS = OPS_CALL("taskkill /IM mds1.exe /F",0)
   @MDS2_STATUS = OPS_CALL("taskkill /IM mds2.exe /F",0)
   @MDS3_STATUS = OPS_CALL("taskkill /IM mds3.exe /F",0)
   ```
8. Define the NCC connections to the Signal X-reference tool in the base system. The entered RTU station numbers should be equal to values entered to CONFIG.INI files, item stn_no_1.

   COM500*i* initializes the modbus databases in the order NCC1, NCC2. and so on. If some of the modbus slaves requires faster communication start-up after a takeover, it is preferred to configure it to a smaller NCC number compared to the others.

4.11.4.2 Activating communication

The communication to the NCCs will be activated when the state of the main application turns to HOT. In this situation, if the configuration step 5 in Section 4.11.4.1 is completed, instances MDS1.EXE, MDS2.EXE and MDS3.EXE should be found from the process list of the computer. The operation of each of these instance can be tested as instructed in Modbus Slave Protocol manual.

Furthermore, the starting and stopping of the modbus instances can be tested by executing the command procedures in steps 5 and 6 in Section 4.11.4.1.

The event channel APL_INIT_1 is executed when the system is started and the state of the main application is HOT immediately after the start-up. APL_INIT_1 is not executed in normal take-over. The event channel APL_INIT_H is activated when the state of the main application changes from COLD to HOT. For more information about the predefined event channels APL_INIT_1 and APL_INIT_H, refer to the Application Objects manual.

The communication activation procedure should be executed before the COM_COMINI_H and COM_COMINI procedures in APL_INIT_H and APL_INIT_1.

4.11.4.3 Deactivating communication

When the main application turns to COLD state, it is necessary to deactivate all communication. In this situation, if the configuration step 6 in Section 4.11.4.1 is completed as instructed, instances MDS1.EXE, MDS2.EXE and MDS3.EXE will disappear from the process list of the computer and the communication to the NCCs are at least temporarily stopped.

For more information about the predefined event channel APL_CLOSE, refer to the Application Objects manual.

4.11.5 Hot-standby with ICCP and IEC 61850 Server

Configuration of the Hot-standby with ICCP and IEC 61850 Server has been described in protocol specific manuals, that is, in SYS600 IEC 60870-6 (ICCP) Protocol and SYS600 IEC 61850 Server.

4.11.6 Hot stand-by with CPI applications

The general requirement in the HSB setup with CPI applications is that if the takeover occurs, the CPI application instances in the computer going from HOT to COLD should not disturb the communication of the system going to HOT state.

In practice, the following rules should be applied:

1. In all serial line protocols, the DTR pin of the used serial port should be set to non-signaled state when the main application goes to COLD state. Correspondingly, the DTR pin of the used serial port should be set to signaled state when the main application goes to HOT state. The controlling of the DTR pin is needed to control the fallback switches between the application and the remote device. If the CPI application instance provides an attribute which controls the DTR pin and also activates/deactivates the communication, this attribute can be used from the predefined event channels APL_INIT_1, APL_INIT_H and APL_CLOSE, using the logic described in Section 4.11.3. If the CPI application instance does not provide an attribute which controls the DTR pin and the communication, the killing of the instance while the system is in stand-by state may be necessary. In this situation, the runtime behavior follows the idea presented in Section 4.11.4.
2. In all LAN protocols operating as TCP/IP client or using UDP/IP, all communication to the remote devices should be deactivated when the system is in stand-by state. In a remote device, any communication from the stand-by system may disturb the communication with the adjacent HOT system. Whether the CPI application provides the runtime control for this or not, the principles presented in rule 1 should be used.
3. In all LAN protocols operating as TCP/IP server, the listening socket of the protocol should be deactivated while the system is in stand-by state or at least the CPI application should close the TCP connection as quickly as possible in this state. Any activity in this state will make the selection algorithm in remote end more complicated and slower. If the CPI application does not provide control for the listening socket, killing the instance while the system is in stand-by state may be necessary.

The rules 2 and 3 need not be followed if it is known that the remote device is capable of handling concurrent connections.

With the COM500*i* application, the communication activation procedure should be executed before the COM_COMINI_H and COM_COMINI procedures in APL_INIT_H and APL_INIT_1.

4.11.7 Hot stand-by with communication gateway COM500*i*

This section describes the principles of the HSB concept in systems including COM500*i* communication gateway.

Figure 49 is a typical HSB system with gateway functionality. Both COM500*i* computers consist of their own SYS600 base system and COM500*i* gateway functionality. When a fault occurs in the primary base system including the HOT application, the shadowing application in the stand-by base system is started and it takes over all the operational functions. For more information, see Section 4.11.1.

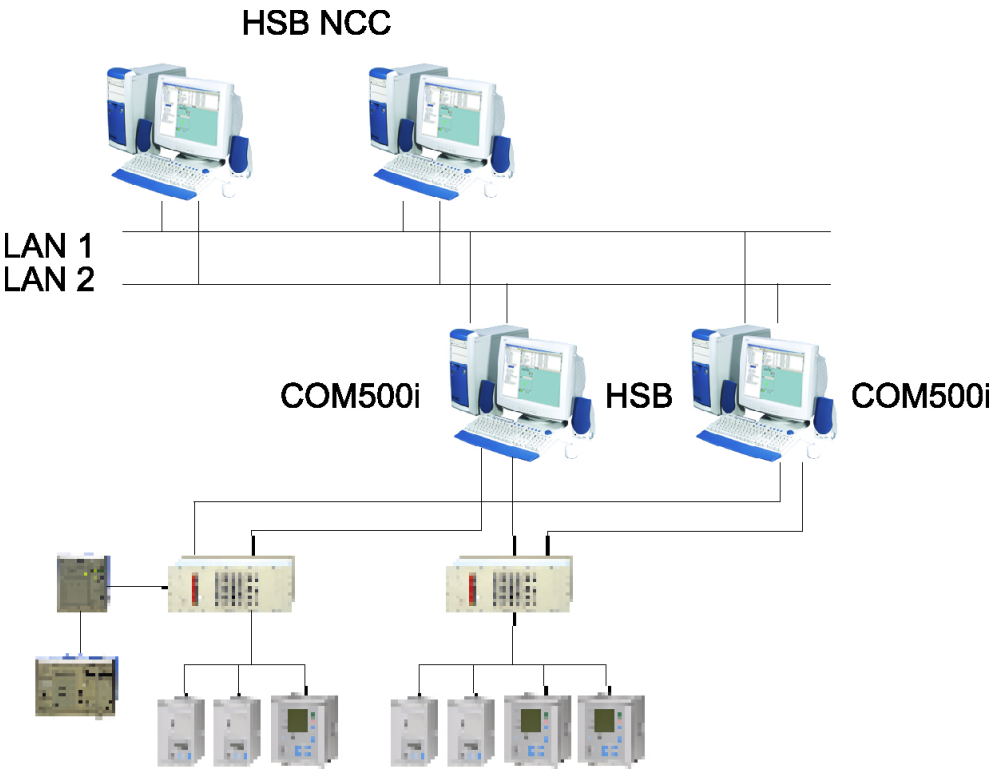


Figure 49: Typical HSB system with COM500i

4.11.7.1 Limitations

- No keep-alive connection from NCCs to stand-by COM500*i*
- Switch is initiated by the hot stand-by application of COM500*i* and not by the NCCs
- Events can be lost or doubled when switch-over occurs in COM500*i*

4.11.7.2 Configuring communication gateway COM500*i*

SYS_BASCON.COM
Set the variable COM500 to TRUE for each COM500*i* application.

```
#local COM500 = vector(TRUE) ;TRUE = COM500i application,
                                                    ;FALSE = not COM500i
application
```

APL_INIT_H
APL_INIT_H is activated when the switch-over occurs and the state of the main application changes from COLD to HOT. COM500*i* automatically writes command #DO COM_COMINI_H:C to the APL_INIT_H command procedure when the main application is opened for the first time after installation. The COM_COMINI_H command procedure starts the initialization of COM500*i* after switch-over as shown in Figure 50.

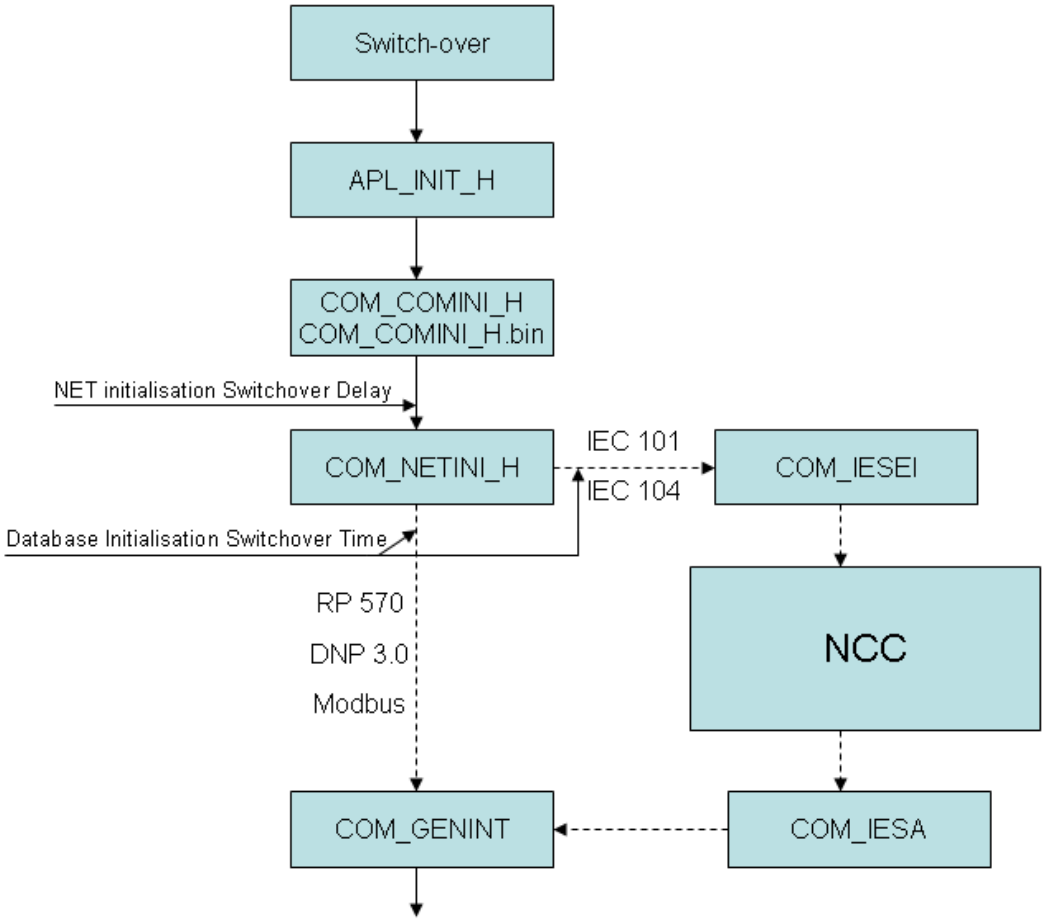


Figure 50: Initialisation of COM500i after switchover

Communication units
For more information, see Communication units configuring in Section 4.11.2, Section 4.11.3, Section 4.11.4 and Section 4.11.6.

Parameters page of signal X-Reference

NET initialization switchover delay
This defines the time (second) after which the initialization of the protocol converters in NET started. This parameter should be set to be the time from switchover to the moment when all the NET lines and stations have been set to in use. The default value is 0 s.

Database initialization switchover time
This defines the time in seconds after which NET database initialisation is started (DNP 3.0 and RP 570) and Database Initialized message in sent to the NCCs IEC 60870-5-101/104. The default value is 0 s.

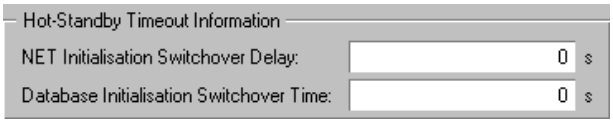


Figure 51: Hot stand-by timeout information

4.12 Configuring mirroring

Process database mirroring is defined on station (STA:B) object level. A station in SYS600 that is connected to a PC-NET or some other data source (host station) is connected to one or more image stations located in other applications, usually in other MicroSCADA X machines.

One host station can have up to 10 image stations. In a hierarchical mirroring system, each image station can in turn act as a host to upper-level image stations. The role of a station object and its mapping to a station located in the external application are defined by station object attributes.

The application containing the host station is called host application (of the station) and the application containing the image station is called the image application (of the station). Note however that an application can have both host and image stations, so it can act in different roles for different stations.

The process objects of a host station and an image station are mapped according to their object address (OA and OB) or source name (IN or IN/EH of OPC based objects). If the object address or the source name of a process object is identical in the host and image database, the objects are considered to denote the same signal in the station device. The logical names of the process objects can be different in different databases.

All the process objects in an image database that are in use (IU = 1) have the switch state AUTO (SS = 2) and map to an in-use AUTO state process object in a host database are subject for mirroring. No new process object attributes are used to configure mirroring communication.

An image application subscribes to the events of process objects in its process database. The image database can only contain a subset of addresses found in the host database, the uninteresting signals can be dropped from the communication load.

The mirroring function contains the following sub-functions:

1. The host application replicates messages from the station device to each image application that has subscribed to the object address.
2. The process commands (#SET and #GET) executed in an image application are routed to be executed by the host application. The changed OV value is sent to the image applications by the host.
3. Any access of STA:S attributes in an image application is routed to be executed by the host application.
4. The host application replicates the system messages from the NET to each image application that has subscribed to the system messages.

The mirroring communication between the host and image application is implemented as APL-APL communication. Consequently, LAN, WAN and serial communication can be used. APL-APL communication between the host and image applications must be configured to enable mirroring.

Communication between the host and the image is buffered and the communication breaks are handled automatically. The events that have occurred during the break are sent when the connection is re-established.

The hot stand-by configurations are supported and the switch overs are handled automatically without losing any events.

Mirroring can be disabled or enabled on a host/image application pair basis by means of APL object attributes. When mirroring is disabled, the host buffers events the same way as during other types of communication breaks.

Significant mirroring events, such as established or lost connections and configuration mismatches, are reported to the application via application events (event channels APL_EVENT and HOST_ADDRESS_MISSING).

Diagnostic counters, implemented as APL object attributes, help monitor the traffic between the host and image applications.

Because it is possible to create very large image applications by using the mirroring function, the maximum number of STA base system objects (MAX_STATION_NUMBER in SCIL) is 50 000.

4.12.1 Station mapping

There are three station (STA:B) attributes that define the role and addressing of a station in the mirroring network. The MR (Mirroring Role) attribute defines the role of the station:

- MR = HOST: A host station that transmits the process data to one or more image stations defined by the attribute IS.
- MR = IMAGE: An image station that receives the process data from the host station defined by the attribute HS.
- MR = BOTH: An image station that receives the process data from the host station defined by the attribute HS. Furthermore, it acts as a host station to the image stations defined by the attribute IS.
- MR = NONE: The station does not participate in mirroring (default).

The HS (host station) attribute of an image station object defines where the corresponding host station is found. It has a list value with the following attributes:

APL	The number of the (usually external) host application
UN	The unit number of the host station in the host application

The IS (Image Stations) attribute of a host station object defines where the corresponding image stations are found. The attribute is a vector of up to 10 list values with the following attributes:

APL	The number of the (usually external) image application
UN	The unit number of the image station in the image application

4.12.2 Process messages

In principle, all the messages from the station device to the host database are replicated by the host database and sent to the image applications that have subscribed to the object address. For load control, however, some measurement events may have been dropped. For more information, see Section 4.12.7. In a hierarchical mirroring network, the image application can also act as a host and re-replicate the messages and send them further to their upper-level image applications.

The substituted values of the process objects (the ones written by SCIL along with the SU attribute) are handled as real process values, i.e. they are subject to mirroring as well. This feature can be used to send mirroring events by SCIL. Define an AUTO state process object

with a pseudo-address (an address having no real counterpart in the process station) and write to it by using the following notation:

```
#SET ABC:P1 = LIST(OV = 1, SU = 1)
```

4.12.3 Process commands

Process commands, i.e. the #GET commands and the #SET commands of the OV (BO, DO, AO or BS) attribute of an AUTO state process object are sent to the host application which executes them on behalf of the image application. In a hierarchical mirroring network, the commands are delivered to the lowest level host. If the command is successful, the new OV value is distributed as a mirroring event to the host database and all the image databases. If it is unsuccessful, the status of the failed command is returned to the controlling SCIL program in the image application.

When a process command is executed by the host application, the new OV value is mirrored to all image databases.

4.12.4 System object (STA:S) communication

Evaluation of STA:S object attributes in an image application, as well as the SET command (#SET) and GET command (#GET), are routed via the mirroring mechanism to the lowest level host application, which executes the request on behalf of the image application. The results (the status of SET and GET command and the result of evaluation) are back-routed to the image application. The host database and other image applications are affected only if the setting/getting/evaluation indirectly generates messages from the station.

Because of the routing via the mirroring mechanism, the tools that communicate with a station via its system object attributes may be run in an image application without any SCIL code changes.

4.12.5 System messages

System messages from the NET are delivered to image applications in a similar way as process messages. The difference in configuration is that in the host application, the system messages are always sent to virtual unit number 0.

In the image application, a non-zero unit number must be reserved for each host whose system messages are received. This unit then represents the virtual unit 0 of the host. As in process messages, the process objects within this unit and the host unit 0 are mapped by their object addresses. In the STA object of the image database, the unit is mapped to unit 0 of the host application by setting HS = LIST(APL = host_application, UN = 0). When the image application starts up, it subscribes to the system messages (object addresses) of the unit in the same way as it subscribes to the process messages.

In the host application, no STA objects related to system message mirroring are needed because system messages are always received to unit 0. Instead, the image stations that receive system messages are listed in a new application attribute IS (Image Stations for System Messages).

The IS attribute of the host application is similar to the IS attribute of a host station. It is a vector of up to 10 list values, which define the image stations mapped to the system messages of this host application.

4.12.6 Subscriptions

The communication between the host and image is subscription-based. When the image application successfully connects to the host, it scans through its process database and sends a list of object addresses that it is interested in, i.e. the addresses that:

- are in use.
- are in AUTO switch state.
- belong to a unit (station) that is connected to the host.

When the host receives a subscription, it immediately sends back the current value of the object (with CT, cause of transmission, set to INTERROGATED). If the requested object address is not found in the host database, an ADDRESS_MISSING event is sent back.

An address is unsubscribed when a mirrored process object in the image database is:

- deleted.
- turned out of use.
- set out of switch state AUTO.

On the other hand, when an in-use AUTO object is created, the image application automatically subscribes to its events.

When an object with subscriptions to it is deleted or its state switched from in-use AUTO state in the host database, an ADDRESS_MISSING event is sent to all the subscribers. When a new process object is created (or switched to in-use AUTO state), a NEW_ADDRESS event is sent to the image applications. They can then decide to subscribe to its events.

The database of the image application can only be a subset of the host database, thereby reducing the required communication rate.

Each image application does its own subscription. The subscriptions can be different.

4.12.7 Buffering and communication breaks

The events to be sent to image applications are buffered by the host system. Each external application that serves as an image application in mirroring has its own event queue.

- The EM attribute (Event Queue Length Maximum) of the application defines the maximum length of the queue.
- The EU attribute (Event Queue Used) shows the current length of the queue.
- The EP attribute of the APL object (Event Queue Overflow Policy) specifies the policy to be followed when the maximum length is reached.

The two different event queue overflow policies are defined below:

- EP = DISCARD: The queue is destroyed, an overflow message is sent to the image and the communication is stalled. In this case, the image application does a general interrogation to the host database, but some events can be lost.
- EP = KEEP: The events are not allowed to be lost. In this case, the process communication between the host application and PC-NET is slowed down just as if the EU attribute of the host application would have reached EM.

The KEEP policy is considered only during the established communication between the host and image. If the limit is reached during a communication break, the DISCARD policy is used.

When the connection to the image application is lost, the host only buffers events without trying to send them. When the image application (or its HSB partner, if there has been a take-over at image site) succeeds in re-establishing the connection, it sends the sequence number

of the last received message to the host and requests retransmitting of newer events. If the host still has the requested events in its buffer, it sends them and no events are lost.

If the requested events are no longer available because the queue length has reached its maximum during the break or because the host has been down, the image application does a new subscription and events can be lost.

During a communication break, the process objects in the image database are marked as old by setting the object status value to 2.

The load control in the communication is done by reducing the rate of measurement events. Measurement event means a process message to an analog input process object when all the following conditions are met:

- The object has a real value representation. Integer valued AI objects, that is the ones with IR = 1, are not considered as measurements.
- The event is a measurement event according to the load control policy of the station.
- The object address is not included in the list of analog event addresses (attribute AE) of the station.

The LP (Load Control Policy) attribute of the station (STA:B) object defines which analog events can be considered as measurement events according to the description above. The attribute can take one of the following four values:

KEEP_ALL_ANALOGS	No analog messages are taken as measurements.
KEEP_TIME_STAMPED_ANALOGS	The messages that are not time-stamped by the station are taken as measurements.
KEEP_NO_ANALOGS	The analog messages are taken as measurements whether they are time-stamped or not.
DEFAULT	The LP attribute of the corresponding STY object is applied.

The station type (STY) objects have a similar LP attribute as well. STY:BLP defines the default policy for all the stations of the type.

For the station types that have the DB attribute value STA, the DEFAULT policy is equivalent to KEEP_TIME_STAMPED_ANALOGS. Otherwise the default policy is KEEP_NO_ANALOGS. For more information about the LP attribute of station and station type objects, see SYS600 System Objects.

The AE (Analog Events) attribute of the station (STA:B) object is defined in the host system. Its value is an integer or text vector of any length and it contains a list of the analog input object addresses (OA) or OPC item names (IN) within the station that are not to be taken as measurements in the sense described above.

4.12.8 Hot stand-by

HSB switchovers are automatically taken care of and no SCIL command procedures are involved.

To be able to do this, the base system should know which external applications make up an HSB pair. Therefore, the SN (Shadowing Number) attribute of the external applications that participate in mirroring should be set. Either of the two applications numbers can be defined as the APL attribute of the HS or IS attribute of the stations participating in the mirroring.

For example, if in an image system the external host applications 7 and 8 make up an HSB pair, the SN attribute of APL7 should be set to 8 and the SN attribute of APL8 should be set to 7. Either 7 or 8 can be defined as the APL attribute of the HS attribute of the stations located in the host.

No events are lost due to HSB switch-overs because the mirroring event queues are shadowed by the host application and the events are sequence-numbered. After an HSB switchover of the host or the image application, the image application asks the host to retransmit all the events that are newer than the last received event.

4.12.9 Disabling mirroring

In an image system, the mirroring communication with a particular host application can be temporarily disabled by setting the HE (Host Enabled) attribute of the host's APL object to 0. Mirroring is re-enabled by setting the attribute back to 1.

In a host system, the mirroring communication with a particular image application can be temporarily disabled by setting the IE (Image Enabled) attribute of the image's APL object to 0. Mirroring is re-enabled by setting the attribute back to 1.

While the mirroring is disabled, the host buffers events breaks, just like during other types of communication, and sends them to the image when the mirroring is enabled again.

4.12.10 Application events

Various significant mirroring events are reported to the application via application event channels APL_EVENT and HOST_ADDRESS_MISSING. For full description of these event channels, see SYS600 Application Objects.

HOST_ADDRESS_MISSING is used in an image application to log the object addresses that, according to the image database, should be mirrored but are not found in the host database.

APL_EVENT is used both in the host and in the image application.

The events reported by the event channel in the image application are the following:

Source	Event	Description
UN	MISSING	The connection to the host station cannot be established because of a mismatch in STA object configuration between the image and the host.
	LOST	The connection to the host station is lost because the mirroring configuration (either MR or IS) in the host has been changed.
	FOUND	The connection to the host station is established because the mirroring configuration in the host has been changed.
	HOST_LOST	A LOST event for the unit has occurred in the intermediate level host. This event is generated instead of the LOST unit to tell the application that there is nothing wrong with the mirroring configuration between this application and the intermediate level application, but the configuration mismatch is detected on a lower level.
	HOST_FOUND	The configuration problem lower in the mirroring hierarchy has been fixed.
HOST	CONNECTED	The connection to the host is established.
	LOST	The connection to the host is lost.
	DISCONNECTED	The connection to the host has been lost because there are no stations connected to the host anymore.
	RECONNECTED	The connection to the host is re-established without losing any events.
	OVERFLOW	The event buffer of the host has overflown. Events have been lost.

Table continues on next page

Source	Event	Description
	DOWN	The connection to the host has been disconnected by the host, because the host application is shutting down.
	DISABLED	The communication with the host has been disabled by setting APL:BHE to 0.
	ENABLED	The communication with the host has been enabled by setting APL:BHE to 1.
	HOST_DISABLED	The communication with the host has been disabled by the host (by setting APL:BIE to 0).
	HOST_ENABLED	The communication with the host has been enabled by the host (by setting APL:BIE to 1).

The events reported by the event channel in the host application are the following:

Source	Event	Description
IMAGE	CONNECTED	The connection to the image is established.
	LOST	The connection to the image is lost.
	DISCONNECTED	The image application has disconnected the mirroring session.
	RECONNECTED	The connection to the image is re-established without losing any events.
	OVERFLOW	The event buffer for the image has overflown. Events have been lost.
	BLOCKING	The event buffer for the image is full, but because of the defined event queue overflow policy KEEP, the buffer is not discarded. The connection is now blocking (or slowing down) the communication between the SYS and the NET in order not to lose any events in the image application.
	NON_BLOCKING	The event buffer for the image is not full anymore. The connection does not slow down the communication between the SYS and the NET anymore. This event is generated when the length of the event queue (EU) has dropped below 90 % of its maximum (EM).
	DISABLED	The communication with the image has been disabled by setting APL:BIE to 0.
	ENABLED	The communication with the image has been enabled by setting APL:BIE to 1.
	IMAGE_DISABLED	The communication with the image has been disabled by the image (by setting APL:BHE to 0).
	IMAGE_ENABLED	The communication with the image has been enabled by the image (by setting APL:BHE to 1).

4.12.11 Configuration examples

The main steps of the mirroring configuration procedure are:

1. Create a node for each base system in the mirroring system (and a LAN link).
2. Create an external application for each image in the host system and an external application for each host in the image system.
3. Define the mirroring attributes for each station: the mirroring role (MR) of a station, the image stations (IS) which are to receive events from the host for the host stations and the host station (HS) for image stations.
4. Raise the amount of APL-APL servers (APL:BAA) of each mirroring application to 10. In most real applications, a lower value would do as well, but the cost of 10 servers is low compared to finding out the smallest usable value. If, however, a lower value is preferred, the following rule can be used. In a host application set the APL:BAA attribute to 10 or two times the number of connected image applications, whichever is lower. In an image

application, set the AA attribute to 10 or two times the number of connected host applications, whichever is lower.

5. Copy/create the process objects of the image application.

The process database of the image system can be a subset of the host process database. All process objects that are of interest can be copied from the host to the image.

Six example configurations are described in the following. The first example describes a simple system where process events are mirrored from a host to an image. The second is an example of a system where a redundant image system receives process events from several hosts. The third example contains a redundant image which receives data from redundant hosts. Station numbering convention is demonstrated in the fourth example. The fifth example demonstrates local mirroring and the sixth one is an example of hierarchical mirroring.

4.12.11.1 One host, one image

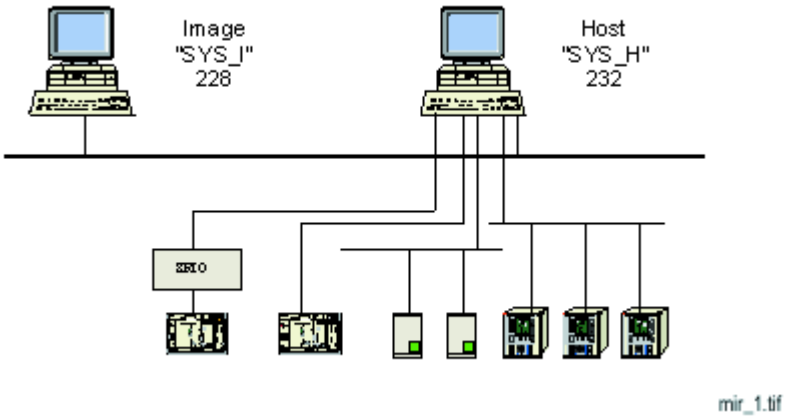


Figure 52: Simple mirroring system

This is a basic configuration. Process database events of a host base system are mirrored to an image base system.

Configuring the host base system
The configuration of the host base system is described first. Mirroring requires a base system node and external application additions in SYS_BASCON.COM. A LAN link, link number 1, is assumed to exist already. In this example, the node number of the host base system is 232 and the node name is SYS_H. The node number of the image base system is 228 and the node name is SYS_I.

A base system node for the image:

```
#CREATE NOD228:B = LIST(- ;Node for SYS_I
     LI = 1,-
     NN = "SYS_I",-
     SA = 228)
```

An external application to represent the image:

```
#CREATE APL2:B = LIST(-
     TT = "EXTERNAL",-
     NA = "IMAGE1",-
     ND = 228,-
     TN = 1)
```

Mirroring attributes of the host stations can be defined in the user-defined programs of the System Configuration Tool. The definition can be written in the user-defined program of each

station, or definitions can be grouped in the user-defined program of the net node. If the System Configuration Tool is not used, the mirroring attributes can be defined. The definition should be written for each mirroring station. The definitions for unit 51 serve as an example in the following:

```
#SET STA51:BMR = "HOST"
#SET STA51:BIS = VECTOR(LIST(APL=2, UN=51))
```

The host application is connected to one image application, so there should be at least two APL-APL servers.

```
#SET APL1:BAA = 2
```

The host part of the mirroring configuration is now ready.

Configuring the image base system
A base system node for the host:

```
#CREATE NOD232:B = LIST(- ;Node for SYS_H
     LI = 1,-
     NN = "SYS_H",-
     SA = 232)
```

An external application to represent the host:

```
#CREATE APL2:B = LIST(-
     TT = "EXTERNAL",-
     NA = "HOST1",-
     ND = 232,-
     TN = 1)
```

The mirroring configuration additions of the stations in the image base system can be written in SYS_BASCON.COM.

Mirroring attributes for stations (station 51 as an example):

```
#SET STA51:BMR = "IMAGE"
#SET STA51:BHS = LIST(APL=2, UN=51)
```

The image application receives messages from one host, which defines that the number of APL-APL servers should be at least 2.

```
#SET APL1:BAA = 2
```

Now the configuration of the image system is ready. Both base systems can now be started and the process objects which are of interest are copied from the host to the image.

System messages
Some additional configuration is required to get the system messages from the NET to the image. In the host application, the attribute IS should be defined to introduce the image station which is to receive system messages.

```
#SET APL1:BIS = vector(list(APL=2, UN=91))
```

In the image system, the respective station, here unit number 91, should be created to receive system messages from the host.

```
#CREATE STA91:B = LIST(-
     TT = "EXTERNAL",-
     ST = "RTU",-
     MR = "IMAGE",-
     HS = LIST(APL=2, UN=0)
     TN = 91)
```

This unit 91 in the image base system now represents the virtual unit 0 of the host, and system messages from the NET are delivered to the image application.

4.12.11.2 Two hosts, redundant image

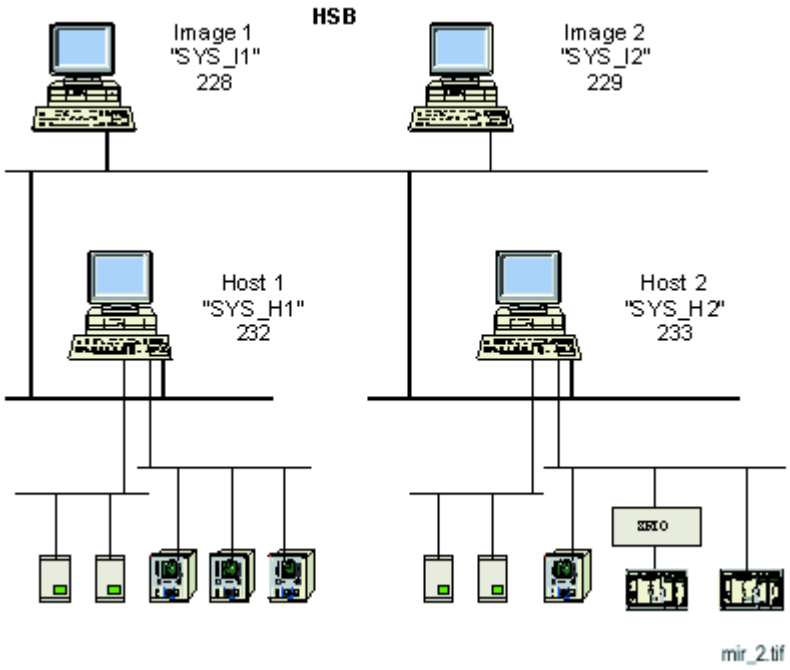


Figure 53: Redundant image, two hosts

In this example, a redundant image base system receives process updates from two host base systems.

- The node number of the image base system 1 is 228 and the node name is SYS_I1.
- The node number of the redundant image base system 2 is 229 and the node name is SYS_I2.
- The node number of the host 1 base system is 232 and the node name is SYS_H1
- The node number of the host 2 base system is 233 and the host name is SYS_H2

The configuration of host base systems is presented first.

Configuring the host base system
The base system nodes for the image base systems are required and should be created in SYS_BASCON.COM of each host base system.

```
#CREATE NOD228:B = LIST(- ;Node for SYS_I1
     LI = 1,-
     NN = "SYS_I1",-
     SA = 228)
 #CREATE NOD229:B = LIST(- ;Node for SYS_I2
      LI = 1,-
      NN = "SYS_I2",-
      SA = 229)
```

An external application should be created for each image. The following code is added in SYS_BASCON.COM of each host base system. Note the attribute SN which defines the application number of the shadowing partner. Here the external applications 2 and 3 make up a HSB pair.

```
#CREATE APL2:B = LIST(-
      TT = "EXTERNAL",-
      NA = "IMAGE1",-
      ND = 228,-
      SN = 3,-   ;Shadowing partner
      TN = 1)

#CREATE APL3:B = LIST(-
      TT = "EXTERNAL",-
      NA = "IMAGE2",-
      ND = 229,-
      SN = 2,-   ;Shadowing partner
      TN = 1)
```

Mirroring attributes of the host stations can be defined in the user-defined programs of the System Configuration Tool. The definition can be written in the user-defined program of each station, or definitions can be grouped in the user-defined program of the net node. If the System Configuration Tool is not used, the mirroring attributes can be defined. The definition should be written for each mirroring station. An example of the definitions for unit 51 is given below:

```
#SET STA51:BMR = "HOST"
#SET STA51:BIS = VECTOR(LIST(APL=2, UN=51))
```

The host application serves one image application, so there should be at least two APL-APL servers.

Now the mirroring configuration of the hosts is completed.

```
#SET APL1:BAA = 2
```

Now the mirroring configuration of the hosts is completed.

Configuring the redundant image base system
Make the modifications in one configuration file and then copy the results to the configuration file of the redundant base system.

A node should be created for each host base system.

```
#CREATE NOD232:B = LIST(-   ;Node for host 1 (SYS_H1)
     LI = 1,-
     NN = "SYS_H1",-
     SA = 232)

#CREATE NOD233:B = LIST(-   ;Node for host 2 (SYS_H2)
     LI = 1,-
     NN = "SYS_H2",-
     SA = 233) An external application should be created for each host
base system.

#CREATE APL5:B = LIST(-
     TT = "EXTERNAL",-
     NA = "HOST1",-
     ND = 232,-
     TN = 1)

#CREATE APL6:B = LIST(-
     TT = "EXTERNAL",-
     NA = "HOST2",-
     ND = 233,-
     TN = 1)
```

The mirroring configuration additions of the stations for the image base systems can be written in SYS_BASCON.COM.

Mirroring attributes for the stations, here the configuration for stations 51 and 53, is presented as an example. Station 51 receives messages from host 1 and station 53 from host 2.

```
#SET STA51:BMR = "IMAGE"
#SET STA51:BHS = LIST(APL=5, UN=51)

#SET STA53:BMR = "IMAGE"
#SET STA53:BHS = LIST(APL=6, UN=53)
```

The image application receives messages from two hosts, so there should be at least four APL-APL servers.

```
#SET APL1:BAA = 4
```

The mirroring configuration of the image base systems is now ready. All base systems can now be started and process objects can be copied from the hosts to the hot image.

Overlapping unit numbers

In a mirroring system where process events are gathered from several existing hosts, it is possible that the same unit number exists in several hosts. Therefore, after the process objects have been copied, the overlapping unit numbers should be changed in the image application. When defining mirroring attributes for the station, this should be noticed in the host base system.

For example, if unit 2 exists both in host 1 and host 2, the unit number of the process objects from host 2 should be changed to any valid value which is not in use. Here the new unit number in the image application can be 3.

The mirroring definitions for station 2 are:

```
#SET STA2:BMR = "HOST"
#SET STA2:BIS = VECTOR(LIST(APL=2, UN=2))
```

in host 1 and

```
#SET STA2:BMR = "HOST"
#SET STA2:BIS = VECTOR(LIST(APL=2, UN=3))
```

in host 2.

In the image base system a new STA object, station 3, should be created with the appropriate mirroring attribute values:

```
#CREATE STA3:B = LIST(-
     TT = "EXTERNAL",-
     ST = "RTU",-
     MR = "IMAGE",-
     HS = LIST(APL=6, UN=2)
     TN = 3)
```

4.12.11.3 Redundant host, redundant image

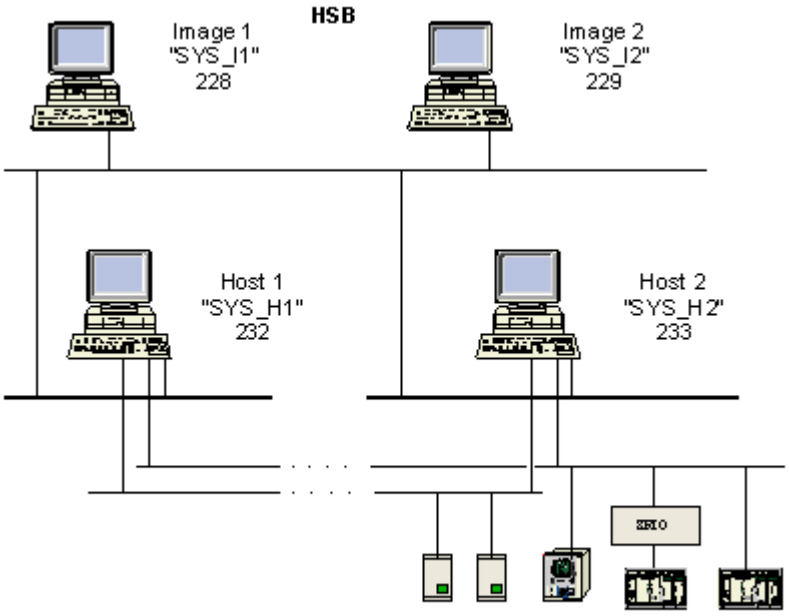


Figure 54: Redundant image, redundant host

In this example, a redundant image base system receives process updates from redundant host base system.

- The node number of the image base system 1 is 228 and the node name is SYS_I1.
- The node number of the redundant image base system 2 is 229 and the node name is SYS_I2.
- The node number of the host base system is 232 and the node name is SYS_H1
- The node number of the redundant host base system is 233 and the host name is SYS_H2

The configuration of host base system is presented first.

Mirroring configuration of the host base system
The base system nodes for the image base systems are required and should be created in SYS_BASCON.COM of each host base system.

```
#CREATE NOD228:B = LIST(- ;Node for SYS_I1
     LI = 1,-
     DI = 10,-
     DT = 5,-
     DF = 1,-
     NN = "SYS_I1",-
     SA = 228)
#CREATE NOD229:B = LIST(- ;Node for SYS_I2
     LI = 1,-
     DI = 10,-
     DT = 5,-
     DF = 1,-
     NN = "SYS_I2",-
     SA = 229)
```

An external application is created for each image. The following code can be added in SYS_BASCON.COM of each host base system. Note the attribute SN which defines the application number of the shadowing partner. Here the external applications 2 and 3 make up a HSB pair.

```
#CREATE APL2:B = LIST(-
     TT = "EXTERNAL",-
     NA = "IMAGE1",-
     ND = 228,-
     SN = 3,- ;Shadowing partner
     TN = 1)
#CREATE APL3:B = LIST(-
     TT = "EXTERNAL",-
     NA = "IMAGE2",-
     ND = 229,-
     SN = 2,- ;Shadowing partner
     TN = 1)
```

Mirroring attributes of the host stations can be defined in the user-defined programs of the System Configuration Tool. The definition can be written in the user-defined program of each station, or definitions can be grouped in the user-defined program of the net node. If the System Configuration Tool is not used, the mirroring attributes can be defined in SYS_BASCON.COM. The definition should be written for each mirroring station. An example of the definitions for unit 51 is given below:

```
#SET STA51:BMR = "HOST"
#SET STA51:BIS = VECTOR(LIST(APL=2, UN=51))
```

In the host application, the attribute IS should is defined to introduce the

image station which is to receive system messages.

```
#SET APL1:BIS = vector(list(APL=2,UN=91)))
```

The host side of the mirroring configuration is now ready.

Configuring the redundant image base system
Make the modifications in one configuration file and then copy the results to the configuration file of the redundant base system.

A node should be created for each host base system.

```
#CREATE NOD232:B = LIST(- ;Node for host 1 (SYS_H1)
     LI = 1,-
     DI = 10,-
     DT = 5,-
     DF = 1,-
     NN = "SYS_H1",-
     SA = 232)
#CREATE NOD233:B = LIST(- ;Node for host 2 (SYS_H2)
     LI = 1,-
     DI = 10,-
     DT = 5,-
     DF = 1,-
     NN = "SYS_H2",-
     SA = 233)
```

An external application should be created for each host base system. Note the attribute SN which defines the application number of the shadowing partner. Here the external applications 2 and 3 make up a HSB pair.

```
#CREATE APL2:B = LIST(-
     TT = "EXTERNAL",-
     NA = "HOST1",-
     ND = 232,-
     SN = 3,-
     TN = 1)

#CREATE APL3:B = LIST(-
     TT = "EXTERNAL",-
     NA = "HOST2",-
```

```
     ND = 233-
     SN = 2,-
     TN = 1)
```

Mirroring attributes for the image stations should be defined. Mirroring attributes for station 51 are presented as an example in the following.

```
#SET STA51:BMR = "IMAGE"
#SET STA51:BHS = LIST(APL=2, UN=51)
```

In the image system, the station is created to receive system messages from the host.

```
#CREATE STA91:B = LIST(-
     TT = "EXTERNAL",-
     ST = "RTU",-
     MR = "IMAGE",-
     HS = LIST(APL=2, UN=0),-
     TN = 91)
```

The image side of the mirroring configuration is now ready.

4.12.11.4 Station numbering convention in a mirroring system

This example illustrates the configuration of a system where the same unit number is used in several hosts, and the messages coming from these units are delivered to one application in an image base system.

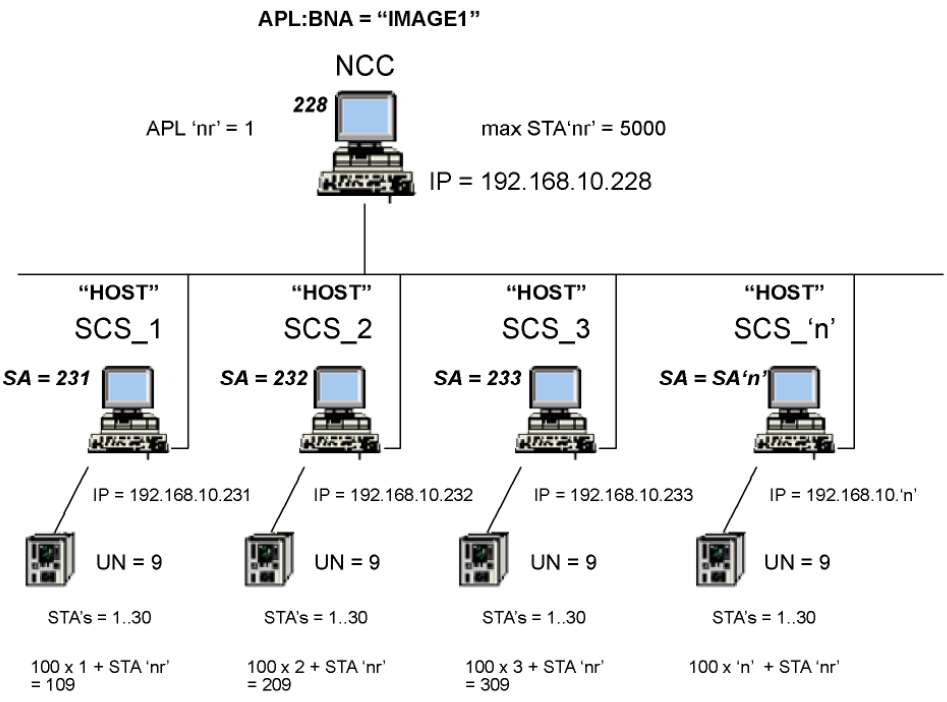


Figure 55: Same unit number in n image applications

Configuring the host base system
Mirroring-related configurations for host 1 (SCS_1).

```
#CREATE LIN:V = LIST(- ;Link to other SYS or LAN frontend
        LT = "LAN") ;Link type
#CREATE LIN2:B = %LIN
```

```
#CREATE NOD228:B = LIST(- ;Node for NCC
     LI = 2,-
    NN = "192.168.10.228",-;if name used, remember define \etc\HOSTS -
table
    SA = 228)

#CREATE APL2:B = LIST(-     ;Mirroring image appl.
    TT = "EXTERNAL",-
    NA = "IMAGE1",-
    ND = 228,-
    TN = 1)

#SET STA9:BMR = "HOST"
#SET STA9:BIS = VECTOR(LIST(APL=2, UN=109)
```

Number of APL-APL servers.

```
#SET APL1:BAA = 2
```

Mirroring-related configurations for host 'n' (SCS_'n').

```
#CREATE LIN:V = LIST(-       ;Link to other SYS or LAN frontend
        LT = "LAN")            ;Link type
#CREATE LIN2:B = %LIN

#CREATE NOD228:B = LIST(-     ;Node for SYS
    LI = 2,-
    NN = "192.168.10.228",-    ;if name used, remember define \etc\HOSTS -
table
    SA = 228)

#CREATE APL2:B = LIST(-     ;Mirroring image appl.
    TT = "EXTERNAL",-
    NA = "IMAGE1",-
    ND = 228,-
    TN = 1)

#SET STA9:BMR = "HOST"
#SET STA9:BIS = VECTOR(LIST(APL=2, UN=709)); UN=100x'n'+STA'nr'
```

Number of APL-APL servers.

```
#SET APL1:BAA = 2
```

Configuring the image base system

Mirroring-related configurations for the image.

```
#CREATE LIN:V = LIST(- ;Link to other SYS or LAN frontend
        LT = "LAN") ;Link type
#CREATE LIN2:B = %LIN
#CREATE NOD231:B = LIST(- ;Node for SCS_1
    LI = 2,-
    NN = "192.168.10.231",- ;if name used, remember define \etc\HOSTS -
table
    SA = 231)
    ....
    #CREATE NOD237:B = LIST(-;Node for SCS_'n'
    LI = 2,-
    NN = "192.168.10.237",- ;if name used, remember define \etc\HOSTS -
table
    SA = 237)    ;SA=230+'n'
    #CREATE APL3:B = LIST(-     ;Mirroring host appl.
    TT = "EXTERNAL",-
    NA = "SCS_1",-
    ND = 231,-
    TN = 1) ;Appl. number
    ....
```

```
    #CREATE APL7:B = LIST(- ;Mirroring host appl.
    TT = "EXTERNAL",-
    NA = "SCS_7",- ;'n'=7
    ND = 237,- ;230+'n'
    TN = 1)  ;Appl. number

#CREATE NOD231:B = LIST(- ;Node for SCS_1
    LI = 2,-
    NN = "192.168.10.231",- ;if name used, remember define \etc\HOSTS -
table
    SA = 231)

....
#CREATE NOD237:B = LIST(- ;Node for SCS_'n'
    LI = 2,-
    NN = "192.168.10.237",- ;if name used, remember define \etc\HOSTS -
table
    SA = 237)    ;SA=230+'n'

#CREATE APL3:B = LIST(- ;Mirroring host appl.
    TT = "EXTERNAL",-
    NA = "SCS_1",-
    ND = 231,-
    TN = 1)  ;Appl. number

....
#CREATE APL7:B = LIST(- ;Mirroring host appl.
    TT = "EXTERNAL",-
    NA = "SCS_7",- ;'n'=7
    ND = 237,- ;230+'n'
    TN = 1)  ;Appl. number
```

Station STA109 receives messages from unit 9 of host 1 (SCS_1) and STA209 from unit 9 of host 2 (SCS_2).

```
#SET STA109:BMR = "IMAGE"
#SET STA109:BHS = LIST(APL=3, UN=9)

#SET STA209:BMR = "IMAGE"
#SET STA209:BHS = LIST(APL=4, UN=9)

#SET STA309:BMR = "IMAGE"
#SET STA309:BHS = LIST(APL=5, UN=9)
....
#SET STA709:BMR = "IMAGE"
#SET STA709:BHS = LIST(APL=7, UN=9)
```

Number of APL-APL servers.

```
#SET APL1:BAA = 10
```

Because it is possible to create very large image applications by using the mirroring function, the maximum number of STA base system objects (MAX_STATION_NUMBER in SCIL) is 50 000.

4.12.11.5 Local mirroring

Both the host and the image are in the same base system. One application is the host application connected to the process, and the other is the image application. In addition to the stations connected to the process, the corresponding image stations must be created as well. In this example, the image station number is 1000 + the host stations number.

Mirroring configuration of the host
Mirroring attributes of the host stations can be defined in the user-defined programs of the System Configuration Tool. The definition can be written in the user-defined program of each station, or definitions can be grouped in the user-defined program of the net node. The

definition must be done for each mirroring station. An example of the definitions for unit 51 is given below:

```
#SET STA51:BMR = "HOST"
#SET STA51:BIS = VECTOR(LIST(APL=2, UN=1051))
```

Mirroring configuration of the image
Mirroring attributes of the image stations can be defined in the configuration file SYS_BASCON.COM.

The station 1051 receives messages from the host station 51 in application 1. Therefore, the mirroring attributes for station 1051 are the following:

```
#SET STA1051:BMR = "IMAGE"
#SET STA1051:BHS = LIST(APL=1,UN=51)
```

4.12.11.6 Hierarchical mirroring

In this example, the node number of the substation base system is 232, and the node name is SUBS. The regional control center base system node number is 230, and the node name is REGIONCC. Finally, the main control center base system node number is 228, and the node name is MAINCC.

Mirroring attributes can be defined in the user-defined programs of the System Configuration Tool. The definition can be written in the user-defined program of each station, or definitions can be grouped in the use-defined program of the net node. The definition must be done for each mirroring station. See an example of the definitions for unit 51 below.

Mirroring definitions in the substation:

This is the host base system.

```
#SET STA51:BMR = "HOST"
#SET STA51:BIS = VECTOR(LIST(APL=2, UN=51))
```

Create an external application (image).

```
#CREATE APL2:B = LIST(-
        TT = "EXTERNAL",-
        NA = "REGION",-
        ND = 230,-
        TN = 1)
```

Create a node for the image.

```
#CREATE NOD230:B = LIST(-   ;Node for the Regional Control Center
        LI = 1,-
        DI = 10,-
        DT = 5,-
        DF = 1,-
        NN = "REGIONCC",-
        SA = 230)
```

Now the mirroring configuration of the substation is ready.

Mirroring definitions of the regional control center
The units in the regional control center can both receive messages from the substation (the host) and transmit messages to the main control center (image). The mirroring role MR of such stations must be BOTH.

Mirroring attributes for station 51:

```
#SET STA51:BMR = "BOTH"
#SET STA51:BHS = LIST(APL=2, UN=51)
#SET STA51:BIS = VECTOR(LIST(APL=3, UN=51))
```

Create two external applications, one for the image and another one for the host.

```
#CREATE APL2:B = LIST(-
        TT = "EXTERNAL",-
        NA = "SUBS",-
        ND = 232,-
        TN = 1)

#CREATE APL3:B = LIST(-
        TT = "EXTERNAL",-
        NA = "MAIN",-
        ND = 228,-
        TN = 1)
```

Create nodes.

```
#CREATE NOD228:B = LIST(- ;Node for the Main Control Center
        LI = 1,-
        DI = 10,-
        DT = 5,-
        DF = 1,-
        NN = "MAINCC",-
        SA = 228)

#CREATE NOD232:B = LIST(- ;Node for the Substation
        LI = 1,-
        DI = 10,-
        DT = 5,-
        DF = 1,-
        NN = "SUBS",-
        SA = 232)
```

The mirroring configuration of the regional control center is now ready.

Mirroring definitions of the main control center
Image configuration additions can be written in SYS_BASCON.COM of the main control center base system. The node number of the base system is 228, and the node name is MAINCC.

Mirroring attributes for station 51

```
#SET STA51:BMR = "IMAGE"
#SET STA51:BHS = LIST(APL=2, UN=51
```

Create an external application for the regional control center (host).

```
#CREATE APL2:B = LIST(-
        TT = "EXTERNAL",-
        NA = "REGION",-
        ND = 230,-
        TN = 1)
```

Create a node for the host.

```
#CREATE NOD230:B = LIST(-  ;Node for the Regional Control Center
        LI = 1,-
        DI = 10,-
        DT = 5,-
        DF = 1,-
        NN = "REGIONCC",-
        SA = 230)
```

The main control center configuration is now ready as well.

4.13 Configuring OPC connectivity

The usage of OPC communication between OPC client and server requires that Distributed COM (DCOM) has been configured accordingly in the Windows operating systems.

The following figure describes all the different locations, where OPC connectivity can be reached via OPC client and server software with the SYS600.

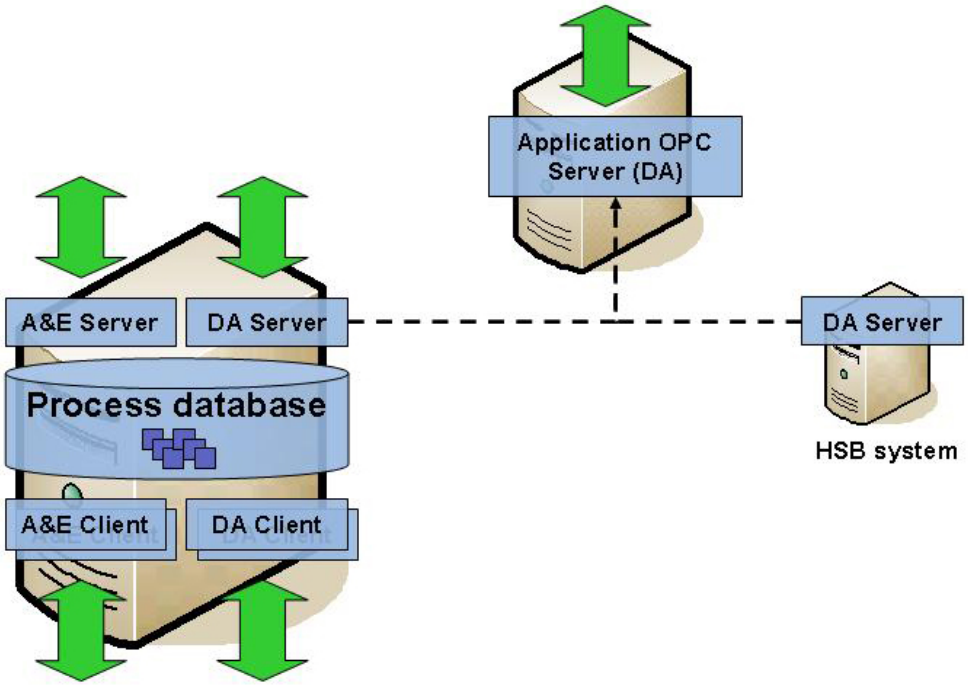


Figure 56: OPC Summary

During the SYS600 installation, the DCOM settings for the usage of OPC communication has been configured automatically into the target computer.

The role of DCOM settings is to make distributed applications secure by using the extensible security framework provided by Windows operating systems. This is possible via storing the access control lists for detailed components into registry of target computer. It is possible to see the DCOM settings by using the DCOM configuration tool (**Start/Run/DCOMCNFG**).

Detailed steps required for the DCOM settings are described in the SYS600 Installation and Administration manual.

Do not forcefully unload the user registry at user logoff
There have been situations where OPC clients in SYS600 or DMS600 (e.g. Communication Engineering Tool (CET) or DMS600 Server Application) have had various symptoms of malfunction e.g. establishing connection to OPC server, error dialogs or even a crash. This has normally occurred when MicroSCADA service has been started or shutdown e.g. HSB takeover. The error code 0x800703FA or message "Illegal operation attempted on a registry key that has been marked for deletion" is related to this. In addition a Windows Application Event ID 1530 is generated indicating Windows registry issues. This is a problem in some COM+ server applications such as MicroSCADA X OPC servers.
Workaround is to not forcefully unload the user registry at user logoff. This is a setting that has to be set manually:
1. Start gpedit.msc
2. Go to Computer Configuration > Administrative Templates > System > User Profiles
3. Click on the key "Do not forcefully unload the users registry at user logoff"
4. Change the key from "Not Configured" to "Enabled"
In HSB redundant system this has to be configured to both computers. See also https://support.microsoft.com/en-us/kb/2287297.

4.14 Configuring Process Display Note links

It is possible to configure whether links are enabled and whether the adding/removing of links is possible.

To configure links:

1. Click the Process Display Note object. A **Process Display Note** dialog is displayed.
2. Click **Configure** in the **Process Display Note** dialog (button available only for system managers).

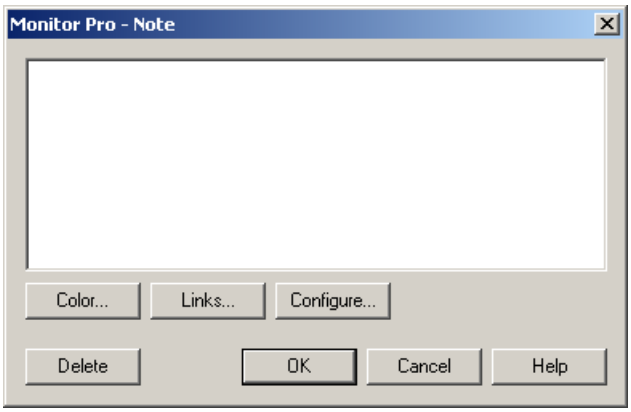


Figure 57: The Process Display Note dialog

3. The **Customize Note** dialog opens.

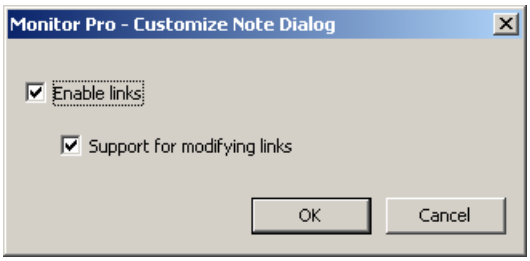


Figure 58: The Customize Note dialog

4. Select whether to enable or disable links and click **OK**. If support for modifying links is disabled, the **Add** and **Remove** buttons in the **Note Links** dialog are disabled.

4.15 Icon storage and configuration

Icons are stored in the \sc\sa_lib\defaults\icons\ folder. Each icon file includes all of the needed icon sizes. The application specific .ini file [appl]\par\apl\framewindow.ini defines the icon mappings, i.e. which icons belong to which menu items and toolbar buttons.

If definitions are not found in the application specific ini-file, the global .ini file \sc\prog\sa_lib \default_framewindow.ini is used.

The section name is MPROICONS, the key name is the icon file name and the key value is the menu or toolbar button id, see the example below:

```
[MPROICONS]

AlarmDisplayTemplate1.ico=32900,32902
```

The icon size is also defined in the ini-file. The system checks the user specific file first, and if no definitions are found the application specific file is checked and then the global ini-file definitions. For icon size definitions there is a possibility for system or user specific definitions. If user specific values are not defined (no value found), the system specific values are used. The section name is MPROICONSIZE, see example below.

```
[MPROICONSIZE]

IconsSize=16x16
```

The key name is IconSize. Possible size definitions are 16x16, 24x24, and 32x32.

4.15.1 Customizing icon mappings

To find out the menu item or toolbar button id:

1. Open the **Customize** dialog by selecting **Settings/Customize**.
2. Click on a menu item or a toolbar button.
3. Add icon mapping to framewindow.ini (see Section 4.15).

The id can be seen on the status bar as numeric presentation, for example 62227 in Figure 59.

Figure 59: Icon ID on status bar

It is also possible to define an icon graphic for the disabled state of menu items or toolbar buttons. This is done by defining an icon with the same name as the active state icon, but adding a string "Disabled" into the name. For example, corresponding to the

AcknowledgeAlarm.ico icon file there is a file named AcknowledgeAlarmDisabled.ico to handle the disabled state of the icon. All icons are defined in the \sc\sa_lib\defaults\icons\ folder. Using and defining disabled state of icons as well is recommended to make the presentation of the disabled state of the icon or toolbar button more clear.

Icons can be reset by selecting **Reset Icons...** from the **Settings** menu. When this is done, all the icon files in the \sc\sa_lib\defaults\icons\ folder are checked and the mapping to each menu item and toolbar button is done based on the information found in \sc\prog\sa_lib \default_FrameWindow.ini file as explained above.

4.16 Configuring Customize Search dialog

It is possible to make more specific configurations to the Search dialog in the **Monitor Pro - Customize Search** dialog. The configurations are global, which means that they affect all users. Configuring requires system administrator user rights. With the correct authorization the **Configure...** button in the Search dialog is enabled.

To configure the Process Display search:

1. Open the **Monitor Pro - Search** dialog by clicking the icon on the Process Display toolbar. The **Search** dialog opens.
2. Click the **Configure...** button, see Figure 60. The **Monitor Pro - Customize Search** dialog opens.

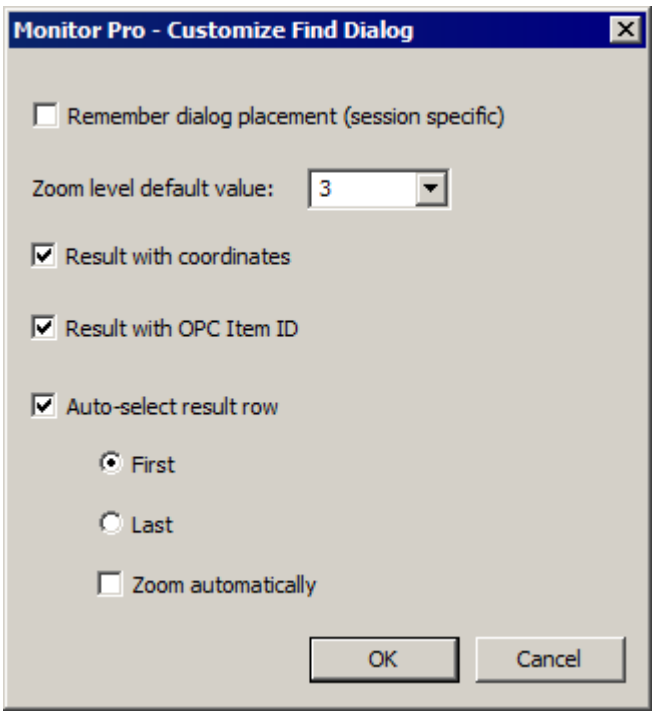


Figure 60: Customize Search dialog

3. Set the configuration in the **Customize Search** dialog, see Figure 60. The following options are available:

- **Remember dialog placement (session specific)**: Checking this box makes Monitor Pro open the dialog to the same location during the whole session until logout.
- **Zoom level default value**: Defines the zoom level used when the located object is selected from the result row and viewed in the Process Display.
- **Result with coordinates**: Checking this box makes the coordinates of the object visible on the **Search** dialog result row.
- **Result with OPC Item ID**: Checking this box makes the OPC Item ID visible on the result row.
- **Auto-select result row**: When this box checked and either **First** or **Last** radio button is selected, either the first or last result row is automatically selected in the **Search** dialog results. The **Zoom automatically** box can also be checked automatically to make Monitor Pro automatically zoom on the selected result.

4. Click **OK** to take the configurations into use.

4.17 Configuring Monitor Pro dialog graphics

The bitmap (ABB logo) for the Splash, Login and About Monitor Pro dialogs is created based on the \sc\sa_lib\defaults\misc\DialogLogo.bmp file. If the bitmap is replaced, the new bitmap dimensions should be 382(width) x 72(height).

4.18 Configuring SYS600 Monitor dialog graphics

When opening SYS600 Monitor, the first visible object can be defined in file \sc\apl\'aplname`\ APL_\Apl_Def.txt. This option can be used when an application specific login dialog is opened or when user login is not wanted at all. For example, the login dialog from LIB4 can be taken in use by doing the following:

1. Copy BGU_LOGIN.VSO from SA_LIB/BASE/BBONE/USE to APL/'APL NAME'/APL_
2. Make this definition to Apl_Def.txt

```
    ;define starting picture
    @i_monitor_number = mon:ban
    #if mon'i_monitor_number':bcx == "<LIB500/INVISIBLE_MONITOR>" #then
                #block
    ;This monitor is reserved for the relay tools. Monitor is
                invisible
     @Start_Object_Type = "Dialog" ;Defines the fist object as a VS
dialog
     @Start_Dialog_File = "b_use/bgu_invisible_mon.vso" ;Needed if
                Start_Object_Type == "Dialog"
     @Start_Dialog_Tag = "Main" ;Needed if Start_Object_Type ==
"Dialog"
     @Start_Dialog_Type = "VS_Main_Dialog" ;Needed if
Start_Object_Type ==
                "Dialog"
     #block_end
    #else_if MON:BDT<>"VS" #then #block
     @Start_Object_Type = "Picture" ;Defines the fist object as a
picture
     @Start_Picture_Name = "APL_START"
     ;Needed if Start_Object_Type == "Picture"
     #block_end
    #else #block
     @Start_Object_Type = "Dialog" ;Defines the fist object as a VS
dialog
     @Start_Dialog_File = "b_use/bgu_login.vso" ;Needed if
Start_Object_Type ==
                "Dialog"
     @Start_Dialog_Tag = "Main" ;Needed if Start_Object_Type ==
```

```
"Dialog"
     @Start_Dialog_Type = "VS_Main_Dialog" ;Needed if
Start_Object_Type ==
                  "Dialog"
    #block_end
```

In order to have Tool Manager as the first object:

```
   @Start_Object_Type = "Dialog" ;Defines if the first object is a
picture or
                  VS dialog
    @Start_Picture_Name = "APL_START" ;Needed if Start_Object_Type ==
"Picture"
    @Start_Dialog_File = "Sys_Tool/ToolMgr.vso" ;Needed if
Start_Object_Type ==
                  "Dialog"
    @Start_Dialog_Tag = "Main" ;Needed if Start_Object_Type ==
"Dialog"
    @Start_Dialog_Type = "VS_Main_Dialog" ;Needed if Start_Object_Type
==
                  "Dialog"
    @Start_Dialog_Attributes= vector("Default_Path=""Sys_Tool""",
                  "Notify_Parent_On_Exit =
false","Delete_Parent_On_Exit=true")
```

4.19 Configuring login dialog shortcut key

A shortcut can be created for the login dialog by defining the shortcut key in [SHORTCUT] of the \sc\prog\sa_lib\default_framewindow.ini file:

```
[SHORTCUTS] ;Keyboard shortcut for showing the login dialog (Ctrl+
[character]).
            If not defined using the default value Ctrl+L LoginDlg=
```

4.20 Configuring custom Process Display Notes

Add custom Process Display Notes that are accessible from**Tools/Notes** menu in Monitor Pro.

The menu items are created based on the files that are found in the following folders:

- System Notes in \sc\prog\graphicsEngine\lib\views\
- Appication Notes in \sc\apl\[appl]\par\apl\Notes\
- User Notes in \sc\apl\[appl]\par\[user]\Notes\

Custom Process Display Note file name should start with CustomNoteMarkerSymbol - prefix. Only .sd files are accepted. Remove the Bounding Box definition from the subdrawing before saving it to a file in Display Builder. Icons for the menu items are created based on the corresponding .ico file using the same naming convention.

Localization of the menu items is done in the same way as, for example, for Custom Commands. For more information, see SYS600 Application Design manual. If localizations have not been defined the default localizations from the resource file are used.

The following authorization levels are honored with custom Process Display Notes:

- Authorization level 5 in GENERAL authorization group to add/modify System Notes.
- Authorization level 2 in PRO_NOTEMARKER_HANDLING/GENERAL authorization group to add/modify Application Notes.
- Authorization level 1 in PRO_NOTEMARKER_HANDLING/GENERAL authorization group to add/modify User Notes.

Notice that user specific custom Process Display Notes are not visible to other users. Use only system or application specific custom Process Display Notes.

Section 5 Configuration tools

5.1 System Configuration Wizard

The System Configuration Wizard offers a tool for the user to configure the elementary information of a single or hot stand-by system even if the project engineer is not familiar with SCIL language.

To start the wizard:

1. Open the SYS600 Control Panel
2. Click **Configuration > System configuration wizard...**
3. The **System Configuration Wizard** dialog is displayed. Click **Next**.
4. The license information of the current system is displayed on the next dialog. Click **Next**.
5. The actual configuration starts on the next dialog. Select the base system type. To configure a single system, select **Single System**. To build a hot stand-by system, select **Hot Stand-by System**. The OPC Server is normally enabled.

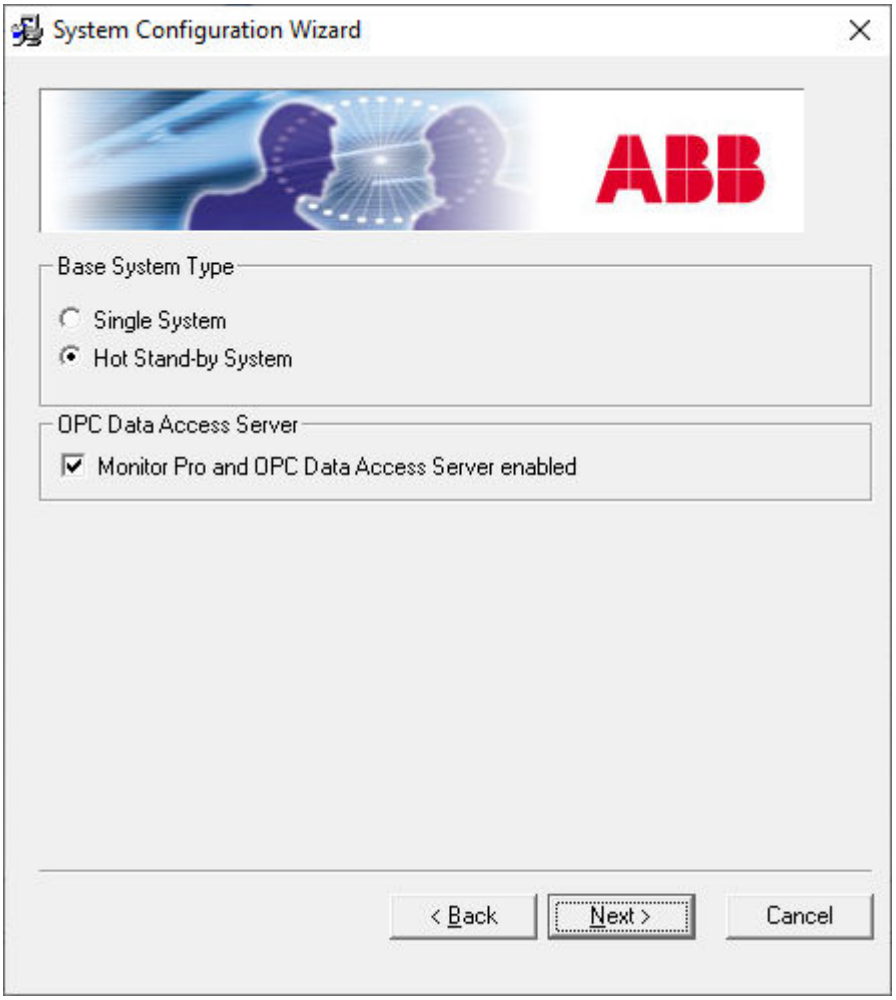


Figure 61: Base system type selection

5.2 Configuring single base system

To configure a single base system:

1. Select **Single System**.
2. On the next dialog, enter the base system node name, node number, station address and web address of the base system node.

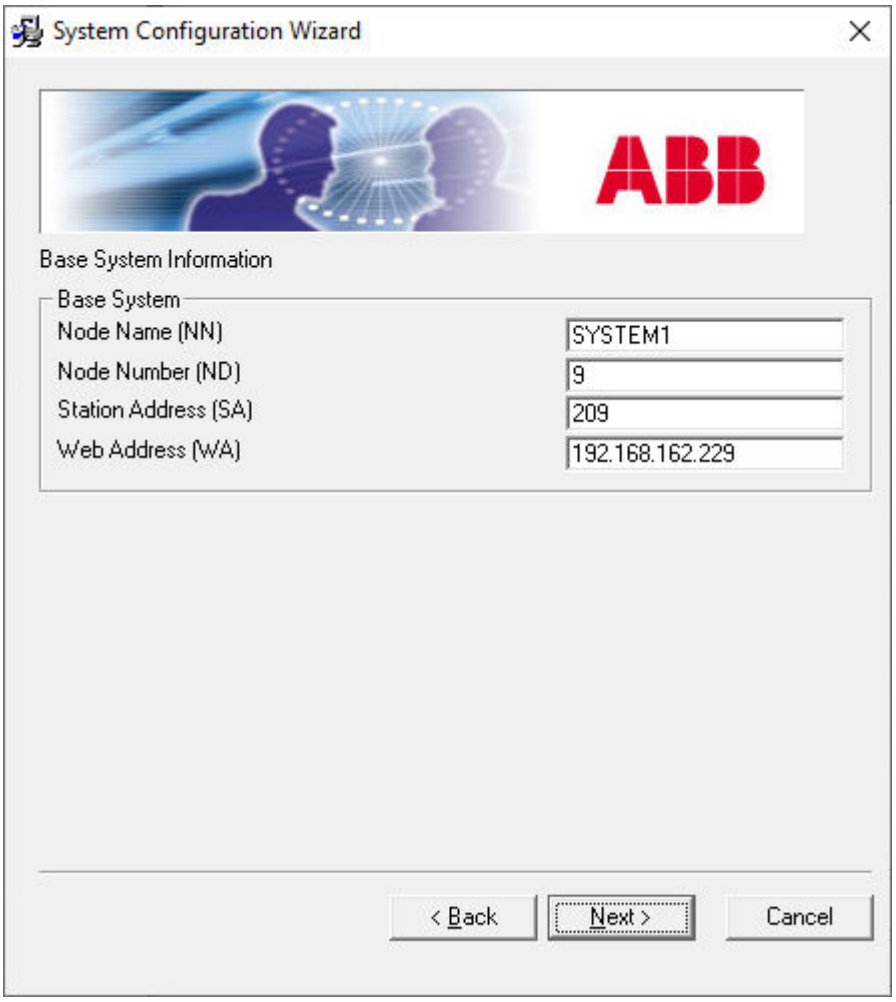


Figure 62: Single base system information

3. Enter the application name. In case of a COM500*i* application, select **COM500i Application enabled**. If the OPC Alarms and Events server is required, select **OPC Alarms and Events Server enabled**. Application Backup can be enabled.

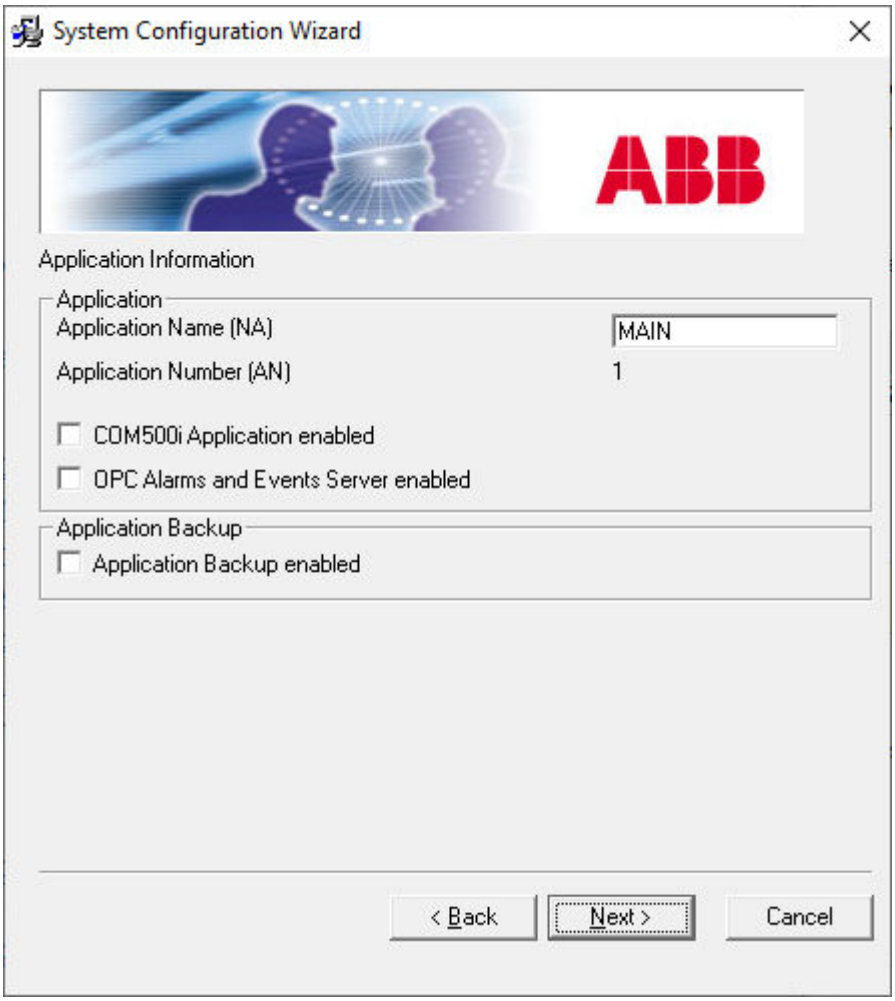


Figure 63: Application information

4. All the required information has now been entered for the elementary configuration of a single base system. The system is ready to be started. The wizard starts the system, if **Start the application** is selected.

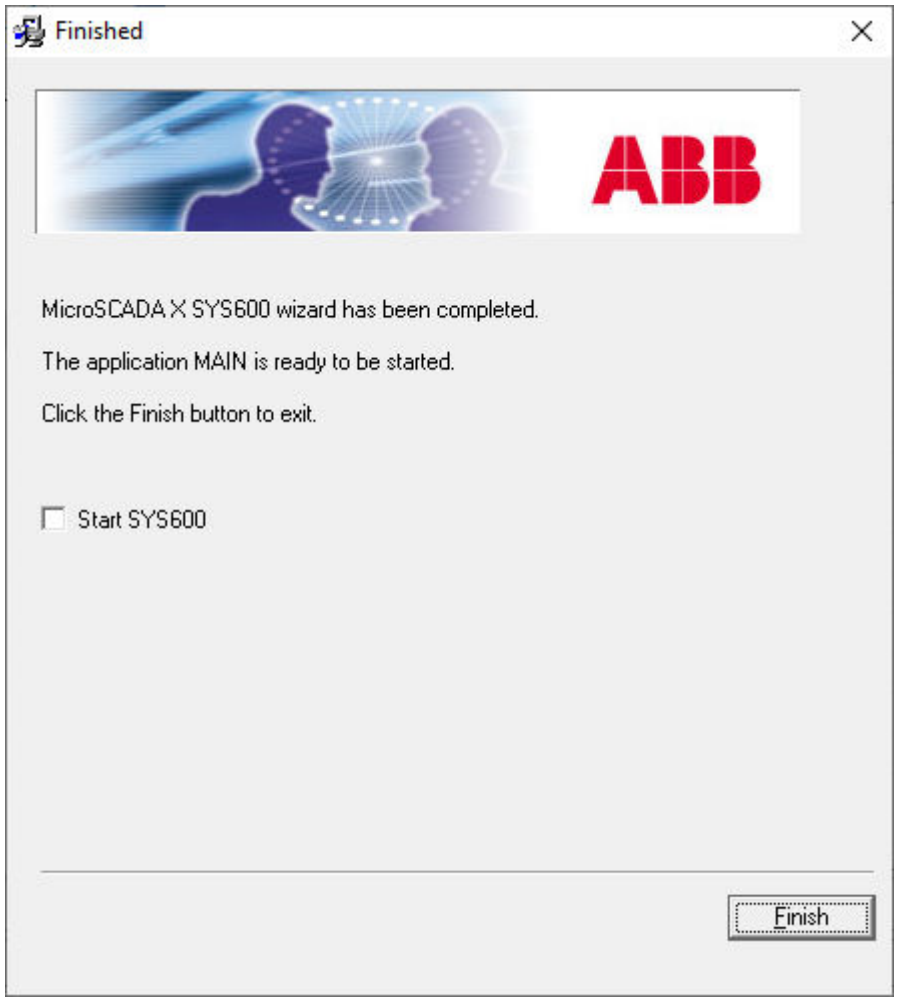


Figure 64: Single base system configuration finished

5.3 Configuring hot stand-by base system

To configure a hot stand-by system:

1. Select the base system type **Hot Stand-by System**.

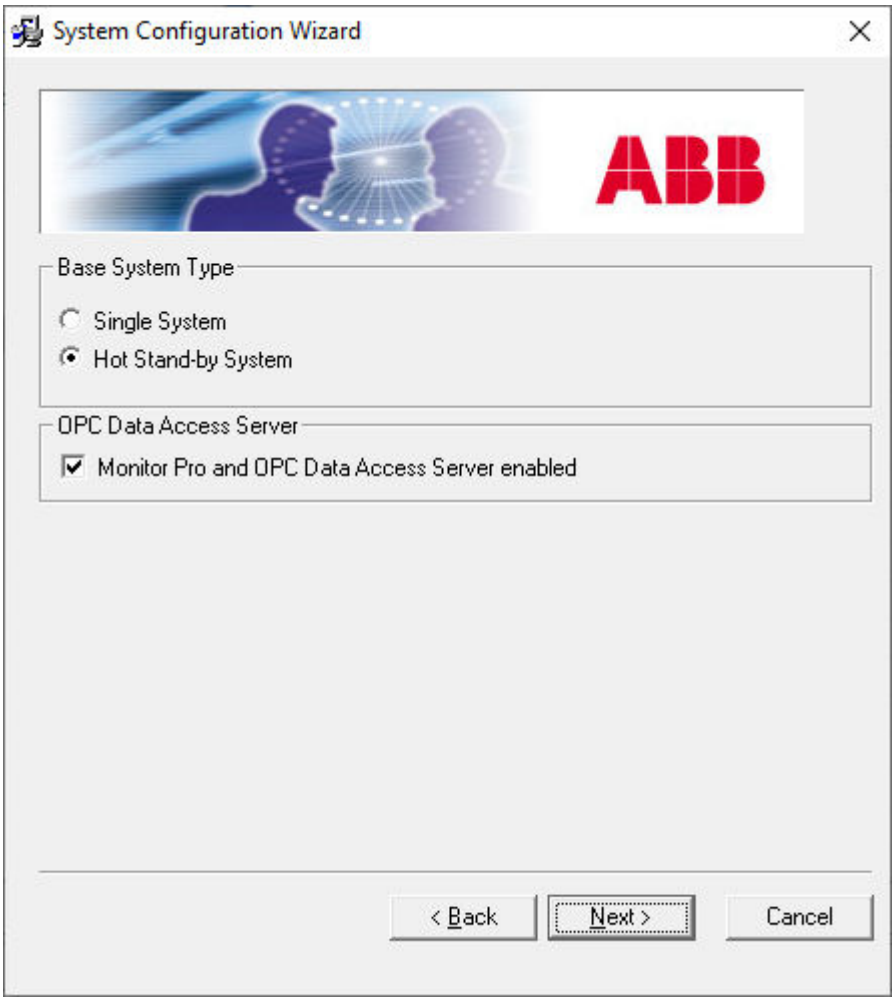


Figure 65: Base system type selection

2. Enter the node name, node number and station address for both base systems. Enter web address for both base systems to support WebUI seamless switch over. Select one of the two base systems as the current node.

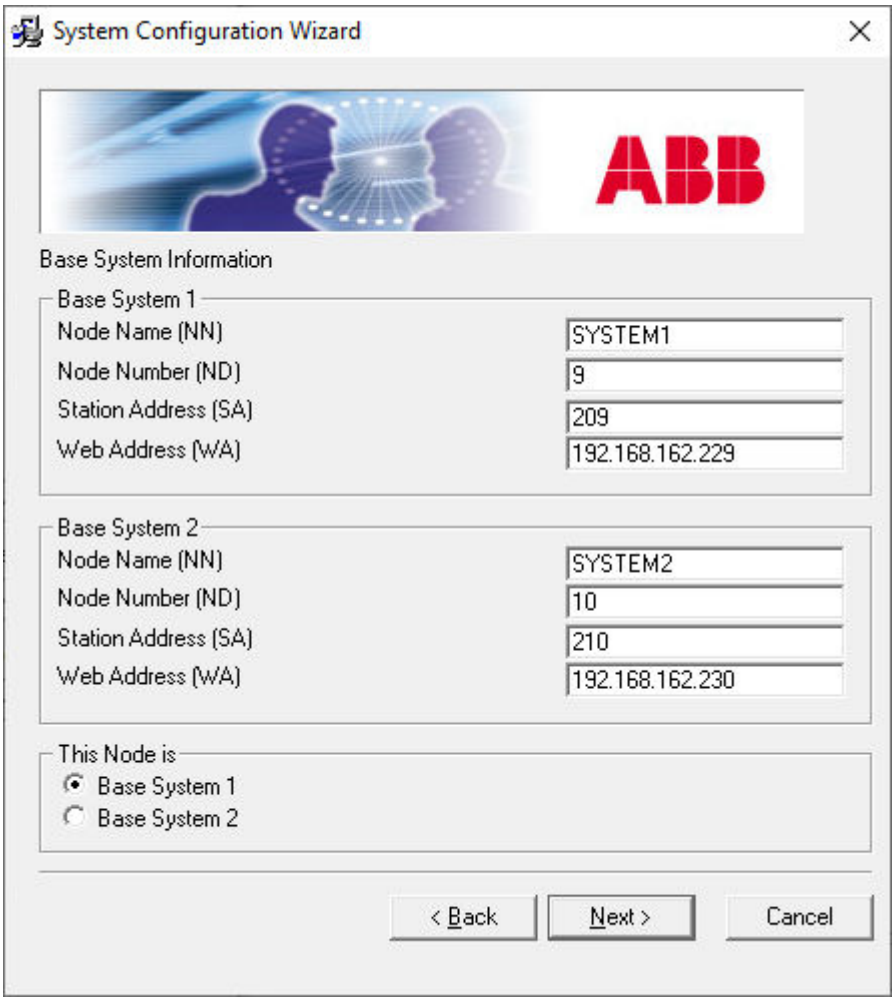


Figure 66: Hot Stand-by system information

3. The next dialog defines the application information. In case of a COM500*i* application, select **COM500i Application enabled**. If the OPC Alarms and Events server is required, select **OPC Alarms and Events Server enabled**. Application Backup can be enabled.

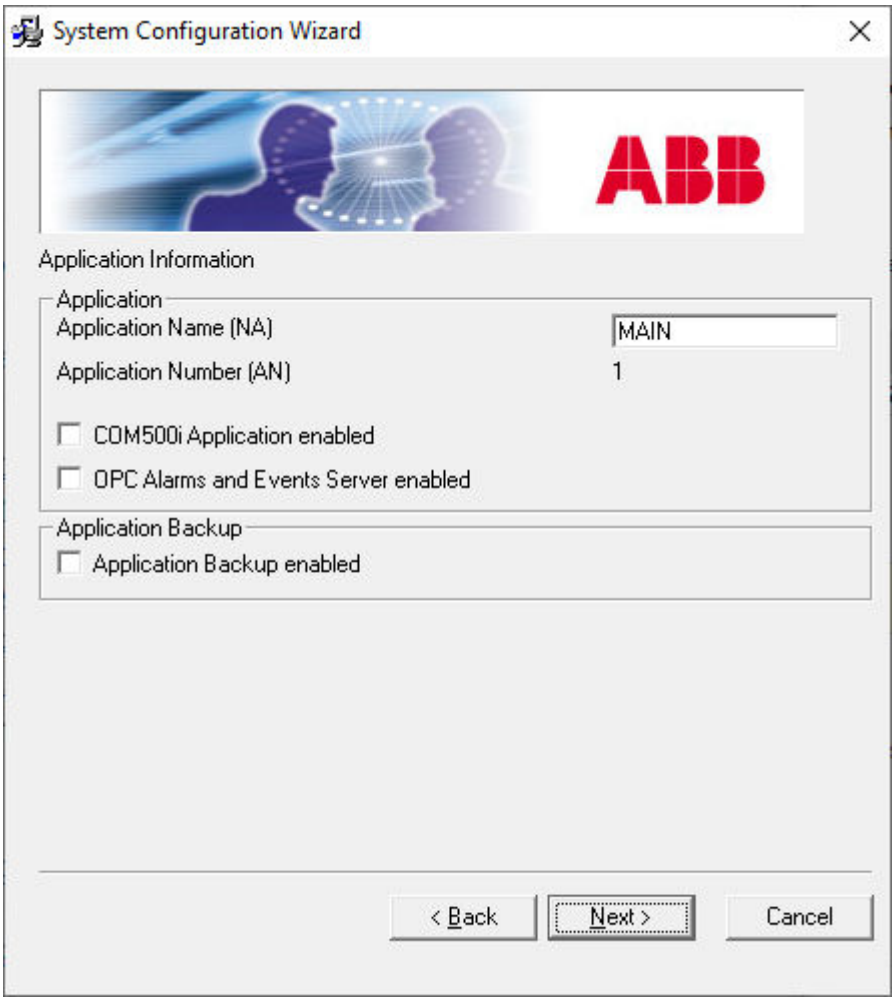


Figure 67: Application information

4. All the required information for the elementary configuration of a hot stand-by base system has now been entered. The system is ready to be started. The wizard starts the system if **Start the application** is selected.

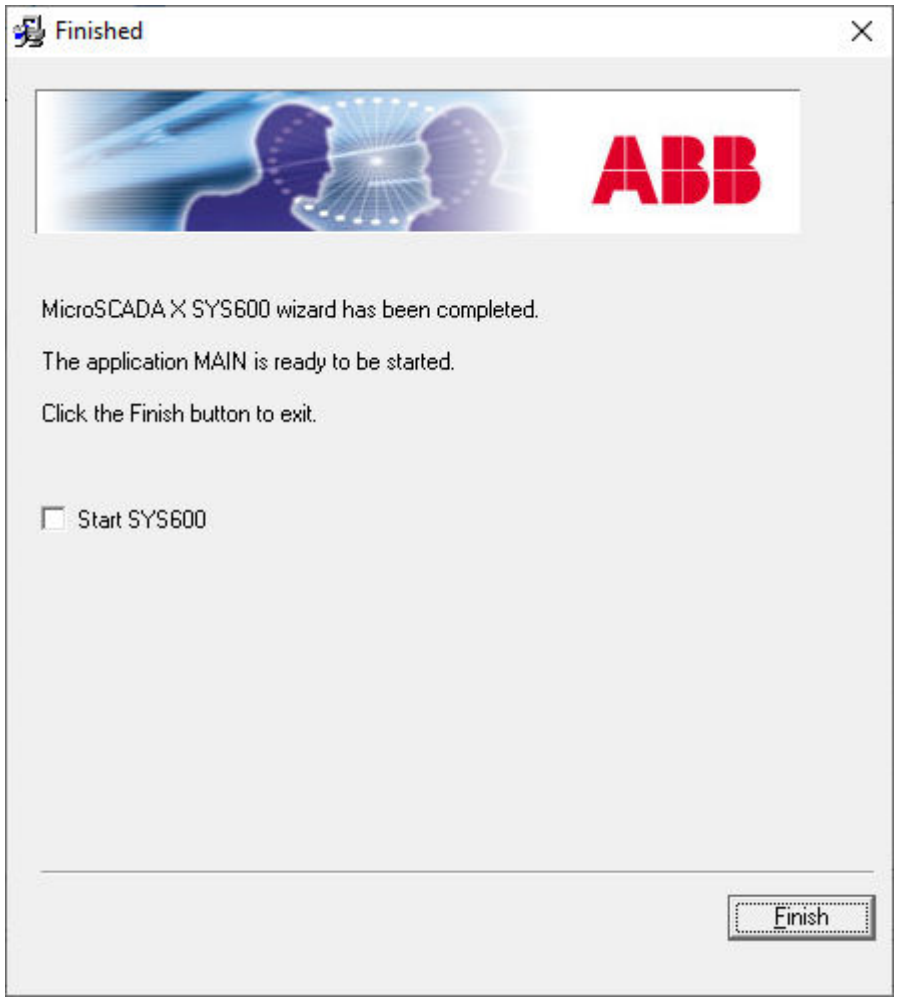


Figure 68: Hot Stand-by system configuration finished

If an upgrade project the old SYS_BASCON.COM is not compatible with the wizard, the following dialog will be displayed. Select **Yes** and the configuration will continue as described above.

Figure 69: Incompatible SYS_BASCON.COM dialog

The wizard writes the configuration information into the SYS_BASCON.COM file. The lines inside the block Quick configuration are affected.

```
    ;File: Sys_bascon.com
    ;Desription:  Standard Base system configuration file
    ; for Single and Hot Stand-By systems
    ; Version 10.0
    ;-------------------------------------------------
    ;
    ;The quick configuration block below is written by the configuration
wizard or
    ;it can be modified manually.
    ;
    ;Quick configuration begin
    ;
    #local Hot_Standby = TRUE ;Hot Stand-by enabled/disabled
    #local Apl_Backup = FALSE ;Backup enabled/disabled
    #local Apl_Names = vector("MAIN") ;Application Name vector, main
applications
    #local BS_Names = vector("SYSTEM1","SYSTEM2") ;Base System Node Names
    #local BS_Nodes = vector(9,10) ;Base System Node Numbers
    #local BS_Addresses = vector(209,210) ;Base System Station Addresses
    #local BS_Web_Addresses = vector("192.168.162.229","192.168.162.230")
                                                ;Base System Web Addresses
    #local This_Node_is = BS_Nodes(1) ;This system 1 or 2 (Always 1 for
single system)
    ;
    #local COM500 = vector(TRUE)  ;TRUE = COM500i application,
                                                ;FALSE = not COM500i
application
    #local OPC_Server_Enabled = TRUE  ;OPC Server enabled/disabled
    #local OPC_AE_Server_Enabled = vector(FALSE) ;OPC A&E Server enabled/
disabled
    ;
    ;Quick configuration end
    ;
```

The wizard also saves the configuration into the SYS_BASCON.INI file.

```
    [System Type]
    Hot_Standby=FALSE
    OPCS=TRUE
    This_Node=1
    [Base System]
    Name=SYSTEM1
    Address=209
    Node=9
    Web_Address=192.168.162.229
    [Base System 2]
    Name=SYSTEM2
    Address=210
    Node=10
    Web_Address=192.168.162.230
    [Application]
    Name=MAIN
    Number=1
    COM500=FALSE
    OPC_AE=FALSE
    Apl_Backup=FALSE
```

When the wizard is started, it reads the contents of the SYS_BASCON.INI file. The wizard does not read the SYS_BASCON.COM file. Thus it does not recognize modifications which are made in SYS_BASCON.COM with Notepad or any other editor.

5.4 System Configuration Tool

The System Configuration Tool is used for configuring the IEC 61850 client, PC-NET, and ICCP communication engines in MicroSCADA X system. With PC-NET, all protocol configuration excluding keys used with IEC TS 62351-5 security extensions can be configured using System Configuration Tool. With IEC 61850 Client and ICCP, configuration is made for node and station level and the actual communication configuration is made with different tools. See corresponding protocol manuals for details.

5.4.1 Starting System Configuration Tool

To start the System Configuration Tool:

1. Click the System Configuration tab in the SYS600 **Tool Manager** dialog.
2. Double-click the **System Conf** tool icon, as shown in Figure 70.

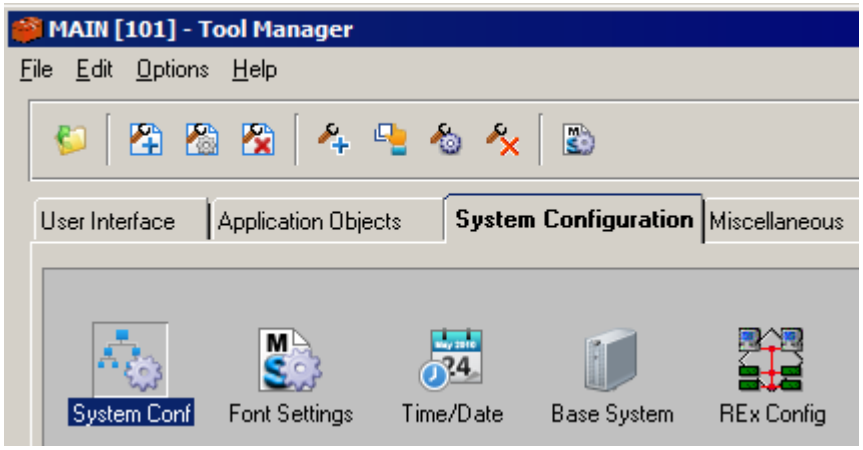


Figure 70: System Configuration Tool icon

The **System Configuration Tool** dialog includes a menubar and a toolbar. To make the toolbar visible, select **Settings/Toolbar Visible**. Below the toolbar, there is an object tree on the left side, an attribute tree in the middle and an attribute editing area on the right side. In addition to these, there is an information text bar and a status bar at the bottom of the page, as shown in Figure 71.

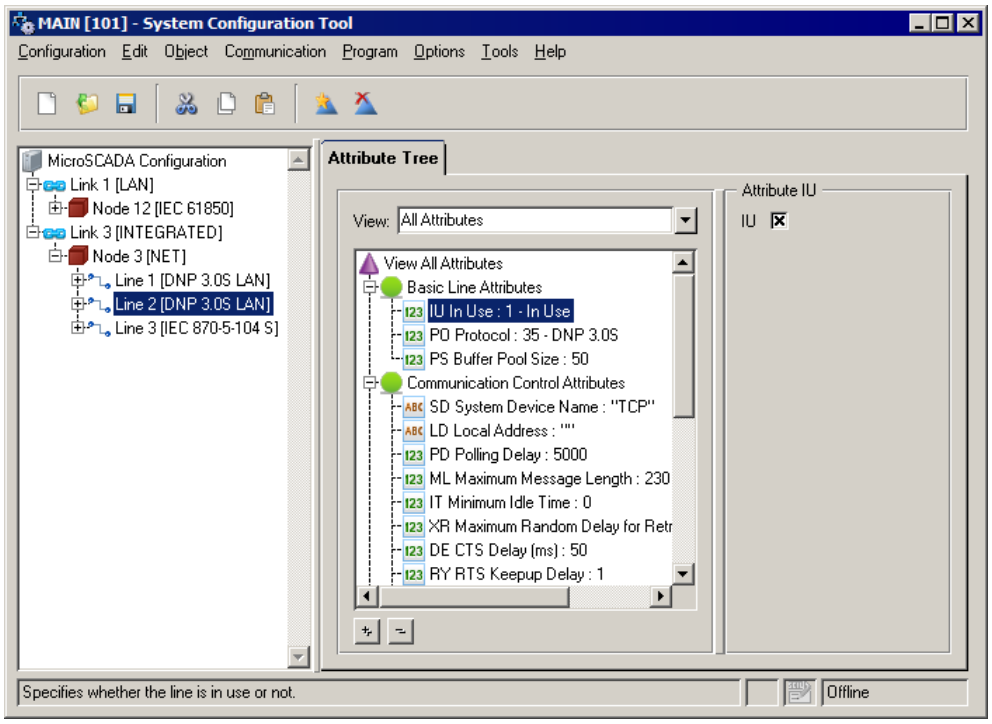


Figure 71: System Configuration Tool dialog

5.4.2 Handling objects and attributes

When selecting an object from the object tree and **All Attributes** is selected as the **View** option, all the attributes linked to it are shown in the attribute tree (as shown in Figure 72).

The tool gives default values to the attributes.

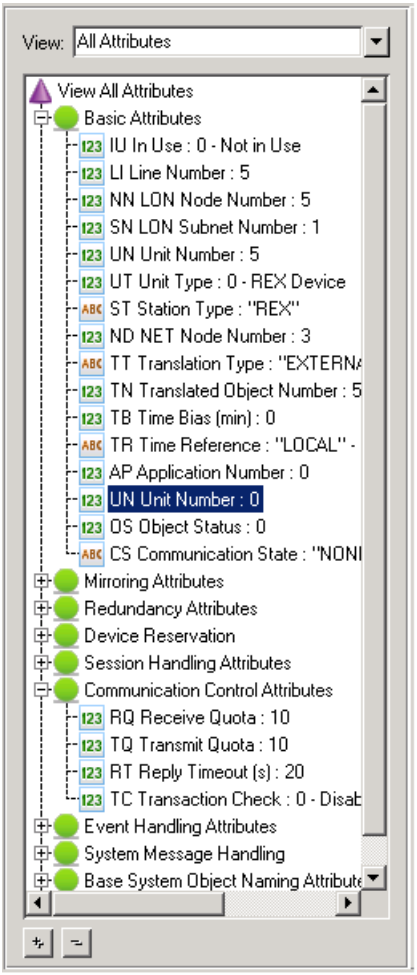


Figure 72: MicroSCADA X Configuration item attributes in the expanded attribute tree

The working order from left to right:

1. Select an object in the **Object Tree**.
2. Click the **+** button under the attribute tree to expand all the attribute groups.
3. Select the attribute to be configured.
4. Change the value in the editing area.
5. Press ENTER.

In the attribute editing area, the on/off values have a check box. A clear check box indicates Off (0) and a selected check box indicates On (1). For integer values, there is a numeric spin box in the editing area, as shown in Figure 73.

The attribute tree is updated when changes are made in the editing area.

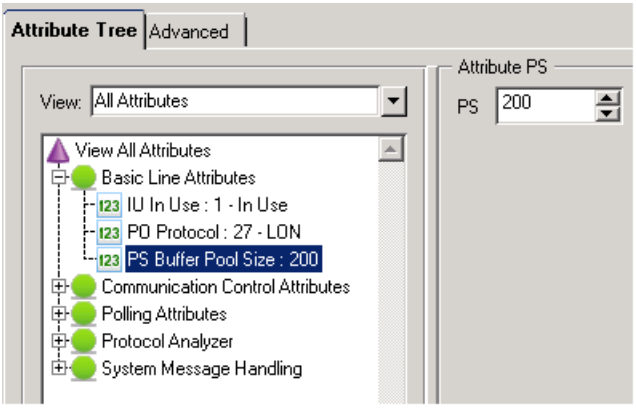


Figure 73: Editing the PS attribute value with the numeric spin box

If the value in the editing area is dimmed, editing is not allowed.

5.4.2.1 NET Node station address

For communication units, the default SA attribute value is 200 + node number. If node number is set bigger than 55, the default SA attribute value set by the System Configuration Tool is 255.

5.4.3 Saving configurations

If a configuration from a former MicroSCADA release is read into the System Configuration Tool, it can be saved with the **Configuration/Save Active** command. The configuration is saved in the following default files: SYSCONF.INI and SIGNALS.INI.

The configuration is available when MicroSCADA 8.4.2 or subsequent SYS_BASCON.COM (sys_bascon$com) template is in use.

5.4.4 Creating a new configuration

From the menu bar, select **Configuration/New**. This command opens a configuration that is delivered with the System Configuration Tool. It includes an Object tree with Link 1 (LAN), Link 3 (INTEGRATED) and Node 3 (NET), as shown in Figure 74.

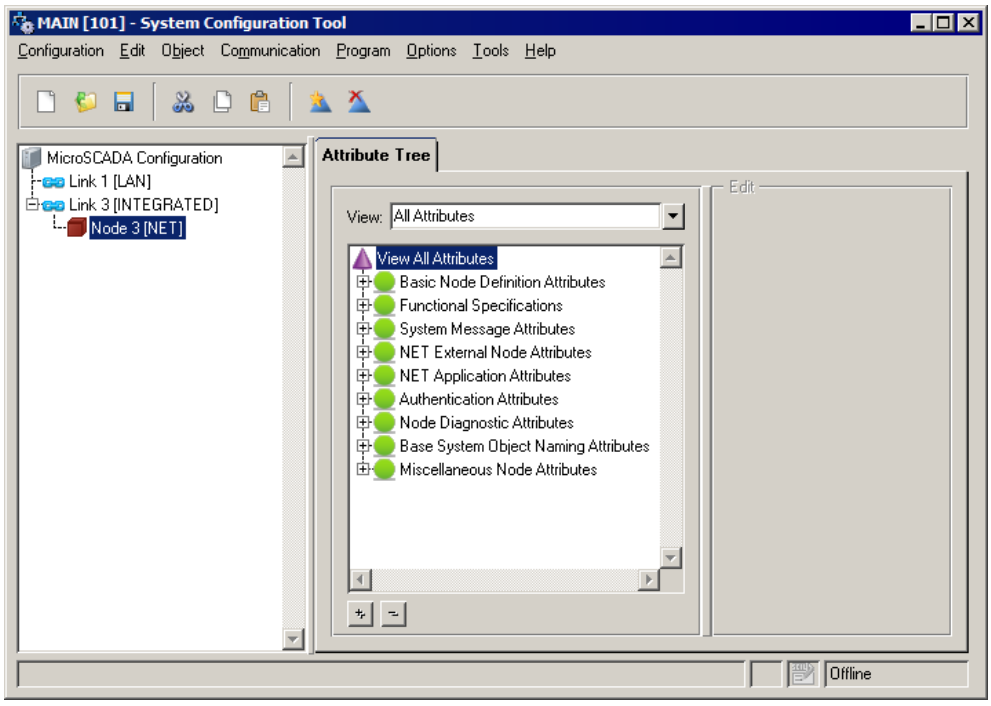


Figure 74: New configuration

If a configuration is already open in the tool, the entire configuration data is cleared from the tool and the contents of the Object tree is replaced with the default configuration. To save the open configuration, copy or rename the SYSCONF.INI and SIGNALS.INI files in the sys/active/sys_ folder.

5.4.4.1 Adding new objects

1. From the object tree, select a parent object for the new object, as shown in Figure 75.

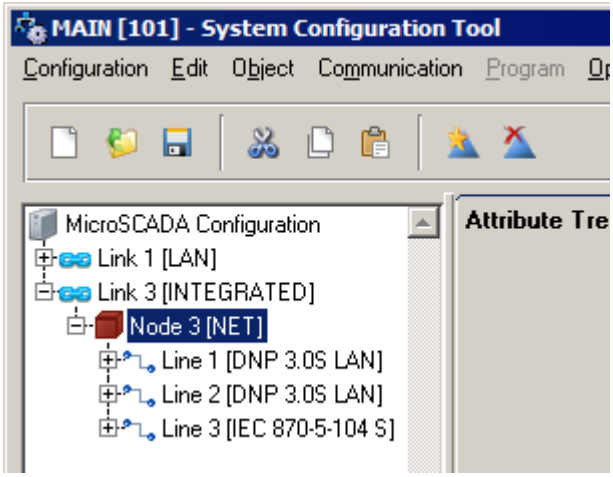


Figure 75: Node 3 (NET) selected to be the parent object

2. After selecting a parent object, there are three ways of adding objects to the configuration:
   - Keyboard command CTRL+N
   - Menu bar command **Object/New**, as shown in Figure 76.

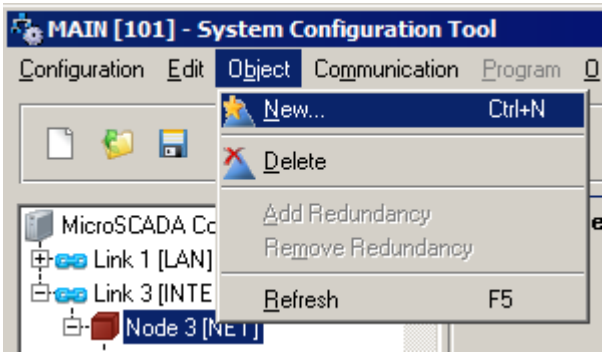


Figure 76: New object is added by using the menu bar command

- The **Object** creation tool icon in the toolbar

Figure 77: New object icon

3. Select the object type and click **Insert**.

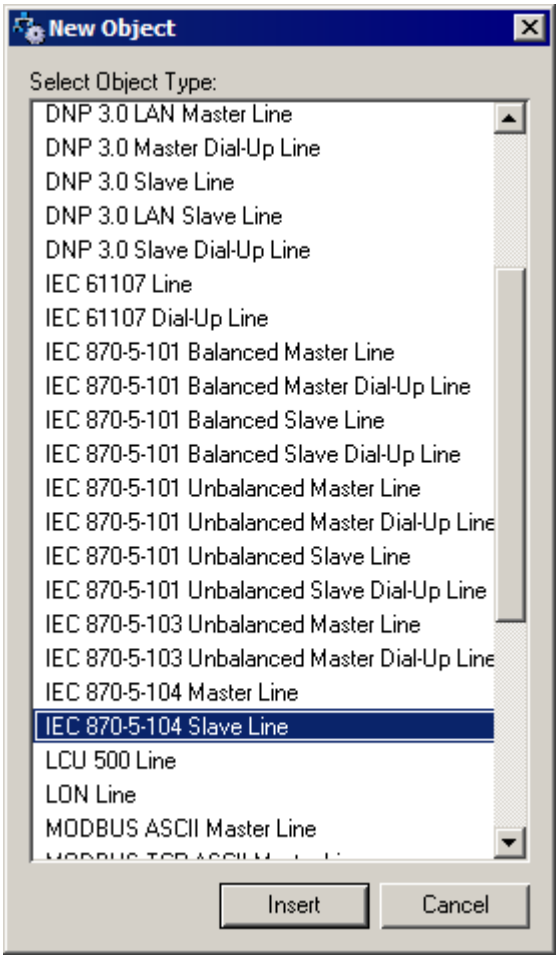


Figure 78: New line is added to the configuration

4. Enter the object number in the text box and click **OK**, as shown in Figure 79.

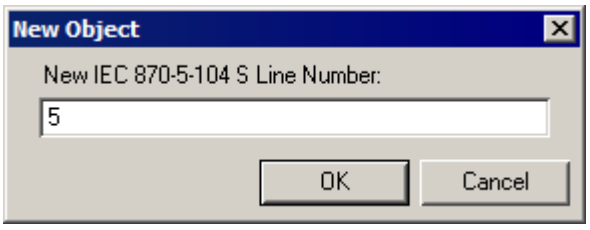


Figure 79: Number five is entered for the new line object

The new object is added to the object tree.

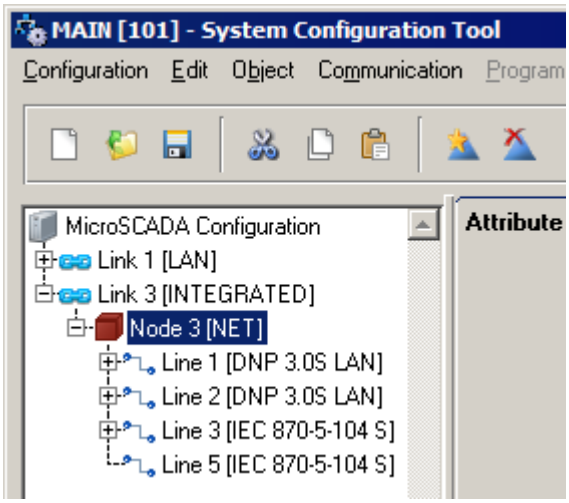


Figure 80: The new line in the object tree

5.4.4.2 Deleting objects

Objects can be deleted by following the steps given below:

1. Select the object from the Object tree.
2. Click the **Object** menu from the menu bar.
3. Select the object and click **Delete**.

If the object includes user-defined SCIL programs or signals, they are deleted as well.

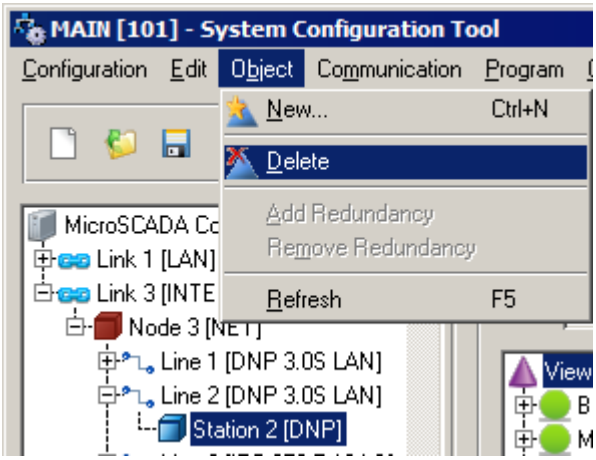


Figure 81: Station 2 is deleted

The main object (MicroSCADA Configuration object) cannot be deleted.

5.4.4.3 Adding a redundant line

The tool supports adding redundant line for IEC 60870-5-101 slave and RP570 slave lines.

1. Select the IEC 60870-5-101 Slave Line or RP570 Slave Line from the object tree.
2. Select **Object/Add Redundancy**, as shown in Figure 82.

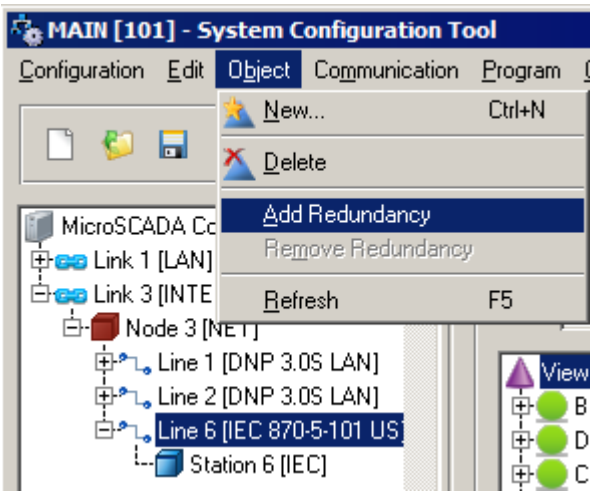


Figure 82: Adding a redundant line

3. Enter the line number of the redundant line in the field, as shown in Figure 83.

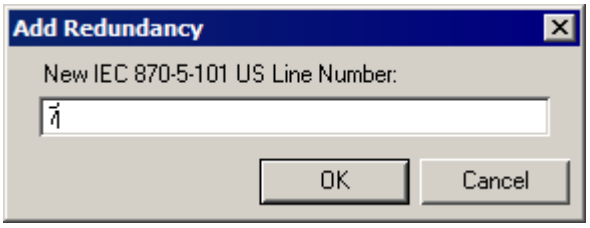


Figure 83: Enter the line number of the redundant line

The redundant line is added to the object tree, as shown in Figure 84.

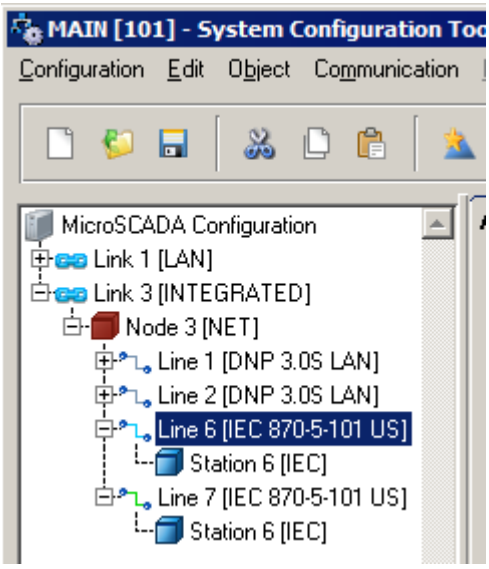


Figure 84: Redundant line configuration

5.4.4.4 Deleting a redundant line

1. To delete a redundant line (IEC 60870-5-101 slave or RP570 slave), select the line to be deleted, as shown in Figure 85.
2. Select **Object/Remove Redundancy** from the menu bar, as shown in Figure 85.

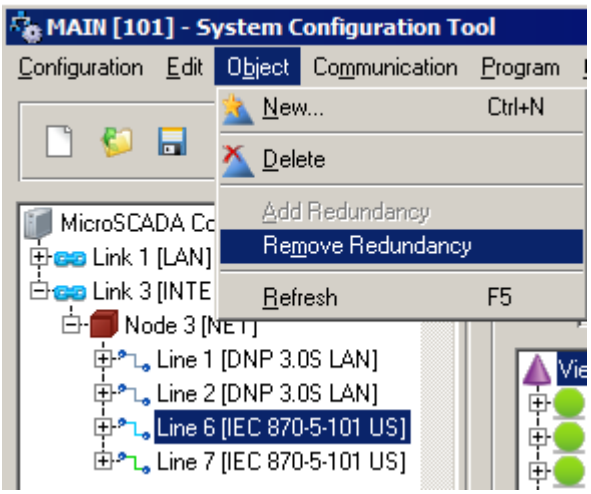


Figure 85: Deleting a redundant line

5.4.5 Configuring dial-up

Some communication lines, for example ANSI X3.28, can be configured to use a dial-up communication. Dial-up protocols are identified in the New Object list when communication line is added to the configuration. In the object tree, an icon is used for dial-up representation and a set of autodialing attributes can be seen in the attribute tree for the selected dial-up communication line, as shown in Figure 86.

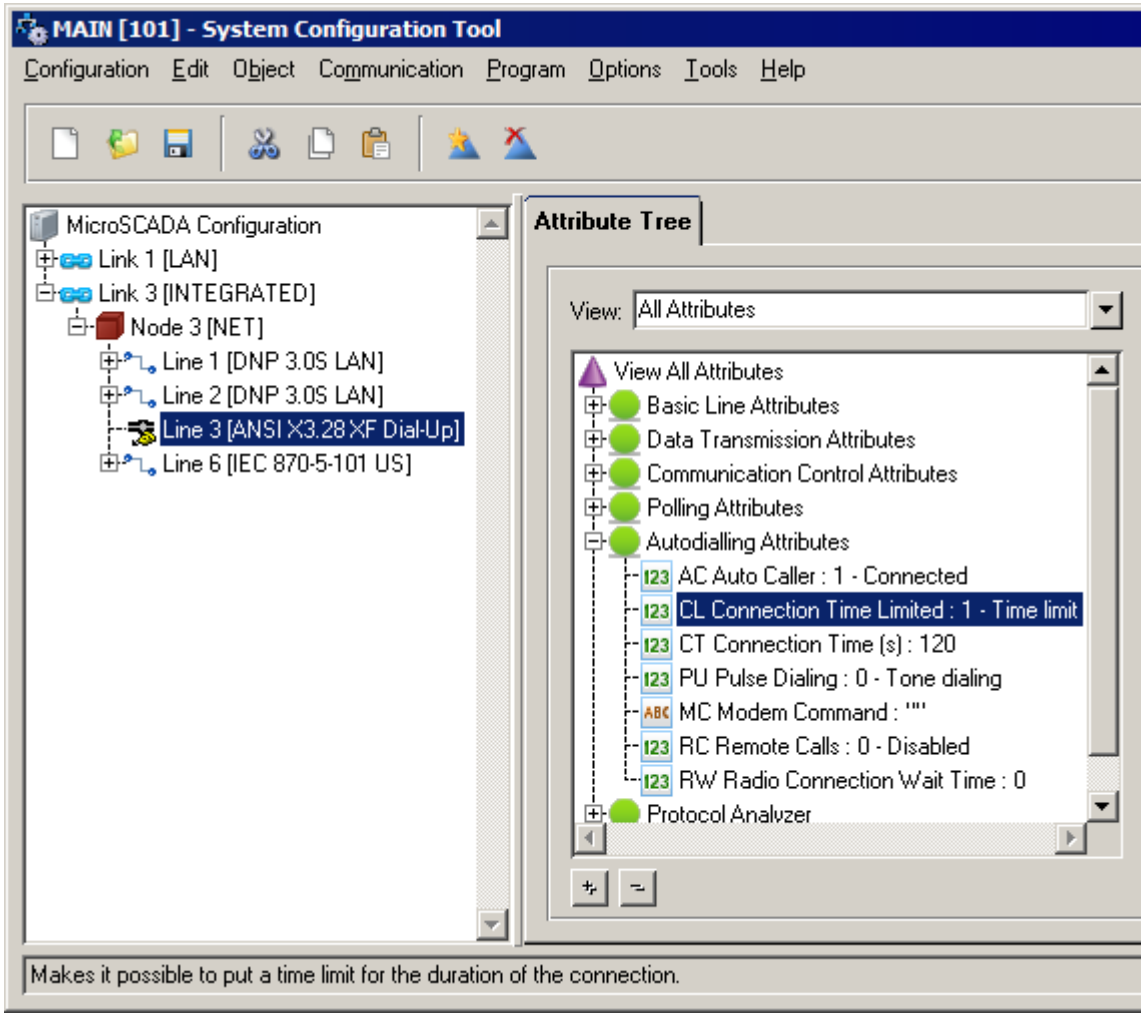


Figure 86: Dial-up configuration

In online mode, the auto-caller state information is displayed on the **Diagnostics** page. It can be used for the dial-up communication line.

If the specified communication port does not contain a modem or the modem is switched off, the communication line cannot be successfully configured into the communication system (PC-NET). When this occurs, the status codes 10003 NETP_TIMEOUT_WHILE_WAITING_ACKNOWLEDGE or 152 SCIL_NET_COMMUNICATION_TIMEOUT are displayed in the SYS600 Notification dialog during the configuration.

5.4.6 Station redundancy

The implementation of station redundancy is described in SYS600 System Objects manual.

The System Configuration Tool supports configuring the redundancy attributes of primary and secondary STA objects and supports adding proxy STA objects.

5.4.6.1 Configuring primary and secondary stations

The tool includes Redundancy Attributes view. Redundancy role (RR) and proxy station number (RS) define primary and secondary stations.

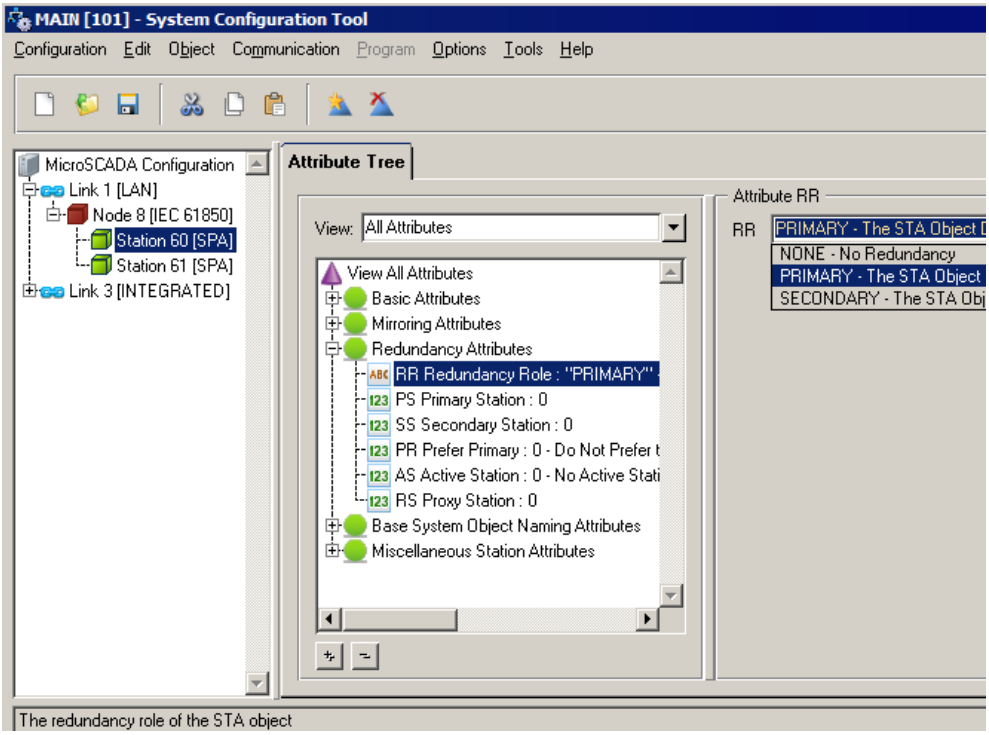


Figure 87: Redundancy Attributes

5.4.6.2 Adding a proxy STA object

To add a proxy STA object:

1. Add the Proxy Station level by selecting **Object/New/Proxy Station**, and then **Insert**.

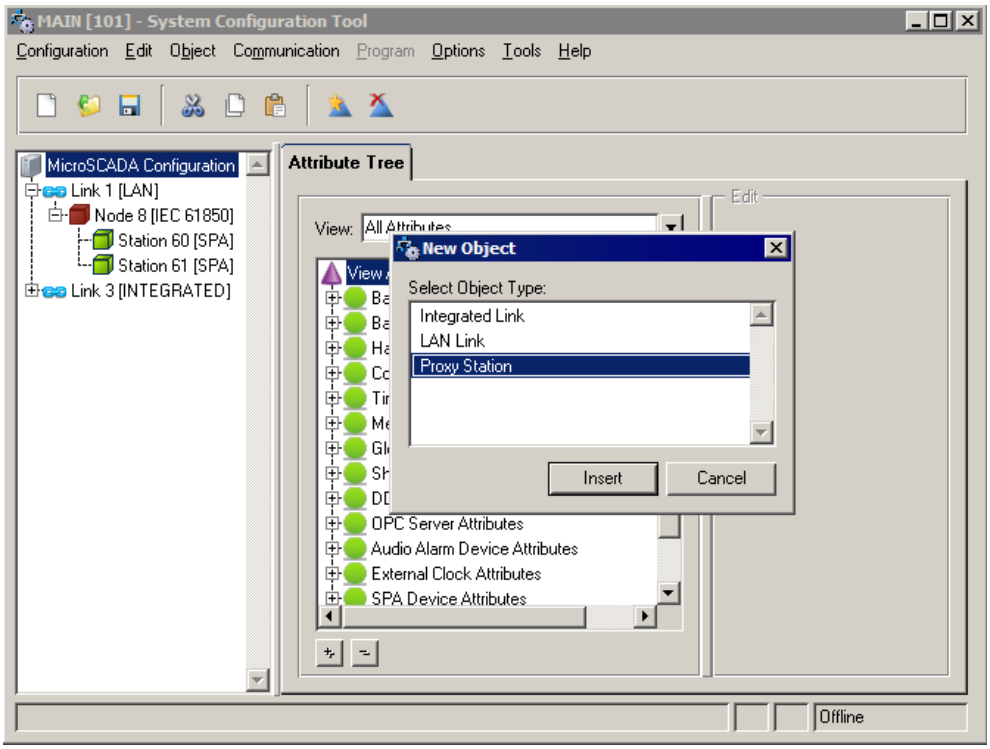


Figure 88: Proxy Station level

2. Add a proxy station by selecting **Object/New**, and select the appropriate station type and enter the station number.
3. Define the primary station (PS) and the secondary station (SS) for the station.

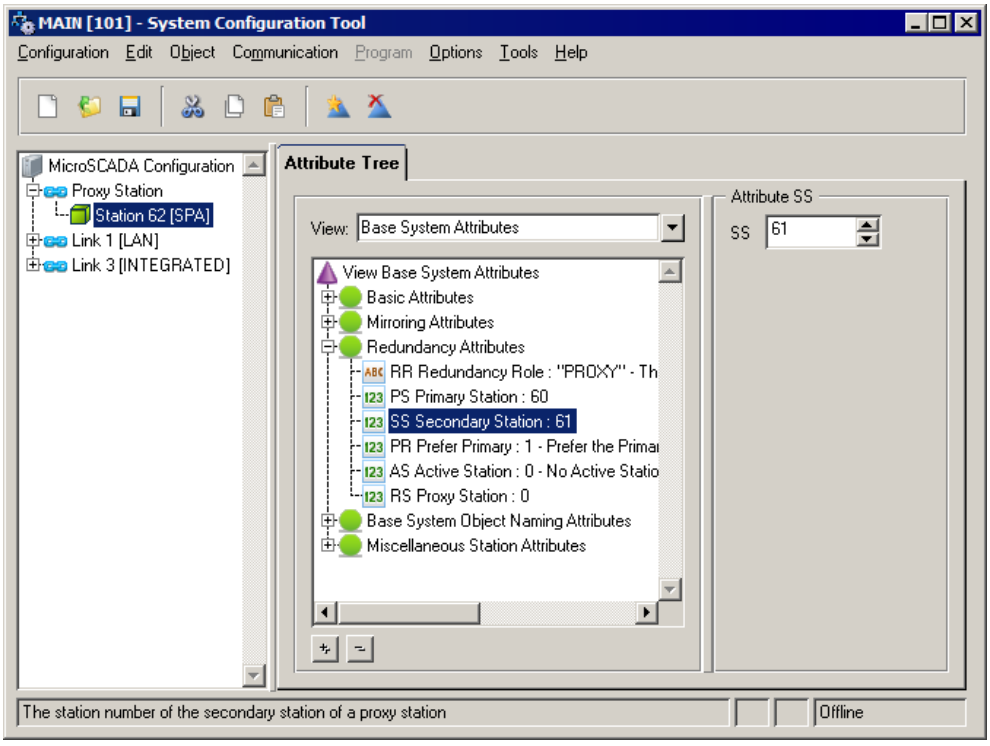


Figure 89: Proxy station configuration

5.4.7 Configuring External OPC DA Client and IEC 61850 OPC Server

The tool supports configuration of the IEC 61850 base system objects for External OPC DA Client.

To configure the IEC 61850 base system objects:

1. Select **Link 1 [LAN]** from the Object tree.
2. Select **Object/New** from the menu bar.
3. Select **IEC 61850 Node** object and click **Insert**.
4. Enter the object number in the text box and click **OK**.
5. Select **Node 2 [IEC 61850]** from the Object tree.
6. Select **Object/New** from the menu bar.
7. Select **IEC 61850 Station** object and click **Insert**.
8. Enter the object number in the text box and click **OK**.

Repeat the steps 7 and 8 above for adding as many IEC 61850 Stations as there are IEDs connected to External OPC DA Client Configuration Tool. After adding IEC 61850 Stations 12, 22, 32 and 42, the configuration appears as shown in Figure 90.

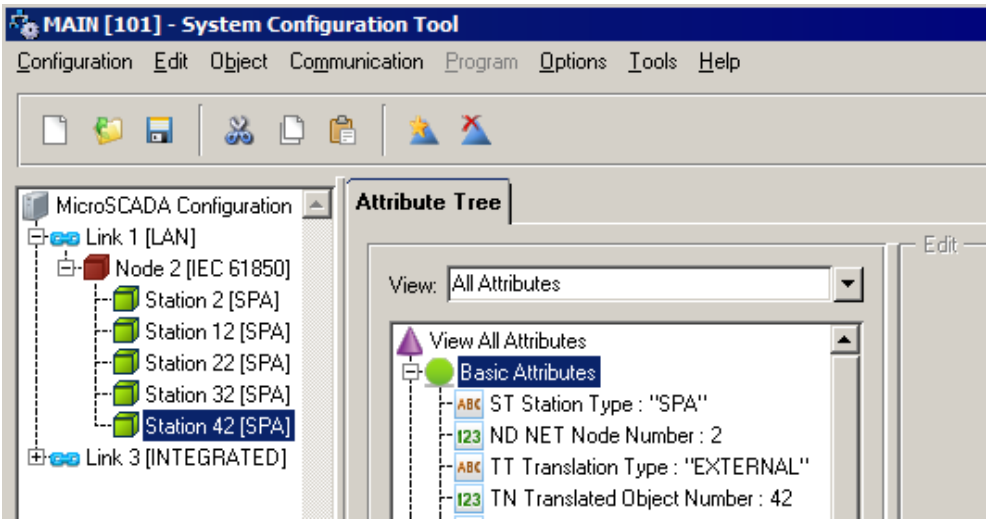


Figure 90: IEC 61850 base system object configuration

Each instance of the External OPC DA Client communication engine represents the collection of the IEC 61850 Node object and one or multiple IEC 61850 Stations. If the system contains multiple instances of External OPC DA Clients, then the appropriate number of IEC 61850 Node objects has to be added to LAN Link.

The configuration of LAN Link object (LIN:B) with the appropriate number should always be included in the SYS_BASCON.COM file. Due to the characteristics of LAN Link object, the tool will not create this object when configuring the MicroSCADA system. The appearance of LAN Link object in the tool indicates the object relationships within the system configuration.

For more information on the External OPC DA Client configuration, see External OPC Data Access Client manual.

5.4.8 Configuring IEC 61850 Server

To configure the base system objects for IEC 61850 Server:

1. Select **Link 1 [LAN]** from the Object tree.
2. Select **Object/New** from the menu bar.
3. Select object **IEC 61850 Server Node** and click **Insert**.
4. Enter the object number in the text box and click **OK**.
5. Select the created **Node [IEC 61850 Server]** from the Object tree.
6. Select **Object/New** from the menu bar.
7. Select **IEC Station** object and click **Insert**.
8. Enter the object number in the text box and click **OK**.

Repeat the steps 7 and 8 above for adding as many stations as needed. Since this station is operating as an interface towards COM500*i*, usually only one station should be created for the node. Node and station object numbers must be in line with the configuration imported using IET Data Loader.

IEC 61850 Server configuration is shown in Figure 91.

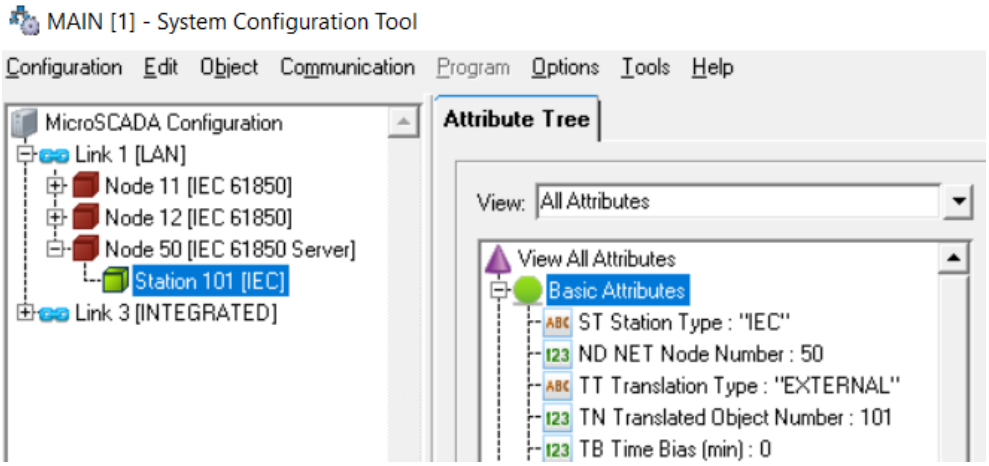


Figure 91: IEC 61850 server base system object configuration

Each instance of the IEC 61850 server communication engine represents the collection of the IEC 61850 server Node object with usually one IEC station connected to it. Having multiple instances of IEC 61850 server engines and nodes are possible.

The configuration of LAN Link object (LIN:B) with the appropriate number should always be included in the SYS_BASCON.COM file. Due to the characteristics of LAN Link object, the tool will not create this object when configuring the MicroSCADA system. The appearance of LAN Link object in the tool indicates the object relationships within the system configuration.

For more information about the configuration, see SYS600 IEC 61850 Server manual.

5.4.9 Configuring ICCP

To configure the base system objects for ICCP communication:

1. Select **Link 1 [LAN]** from the Object tree.
2. Select **Object/New** from the menu bar.
3. Select object **IEC 60870-6 (ICCP) Node** and click **Insert**.
4. Enter the object number in the text box and click **OK**.
5. Select the created **Node [ICCP]** from the Object tree.
6. Select **Object/New** from the menu bar.
7. Select **IEC ICCP Station** object and click **Insert**.
8. Enter the object number in the text box and click **OK**.

Repeat the steps 7 and 8 above for adding as many ICCP Stations as needed. In case if ICCP is operating as a server, that is, NCC of type ICCP will be created to COM500*i*, one ICCP station should be created for the node. In case if ICCP is operating as a client, one station object should be created for each configured RCC (Remote Control Center). This means that if the instance operates both as ICCP server and client, there will be one station object for COM500*i* and additionally, one station object for each RCC. The station object numbers are defined in the ICCP Configurator tool.

ICCP configuration is shown in Figure 92.

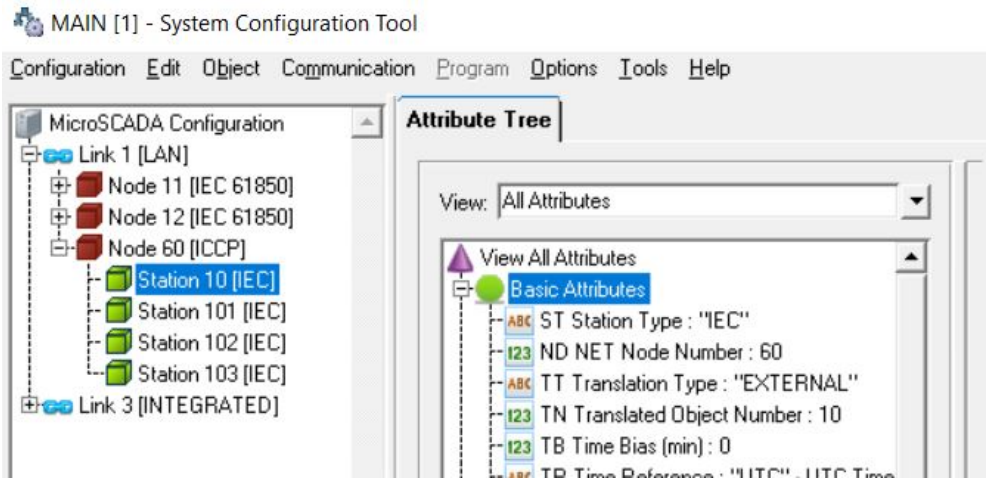


Figure 92: ICCP base system object configuration

Each instance of the ICCP communication engine represents the collection of the ICCP Node object and one or multiple ICCP Stations. Having multiple instances of ICCP engines and node are not currently supported.

The configuration of LAN Link object (LIN:B) with the appropriate number should always be included in the SYS_BASCON.COM file. Due to the characteristics of LAN Link object, the tool will not create this object when configuring the MicroSCADA system. The appearance of LAN Link object in the tool indicates the object relationships within the system configuration.

For more information about the configuration, see SYS600 IEC 60870-6 (ICCP) Protocol manual.

5.4.10 Saving as a default configuration

The default configuration is stored in a configuration file called SYSCONF.INI.

To open the default configuration file, select **Configuration/Open Active**. The default configuration is loaded in the tool.

The tool opens in the off-line mode, which is shown in the status bar.

To save a configuration as the default configuration, select **Configuration/Save Active**. The configuration currently open in the tool is saved as the default configuration in the SYSCONF.INI file.

The configuration can be saved at any time and the saving can be done in both online and off-line mode.

To restore the previous configuration, select **Configuration/Restore Previous Configuration**.

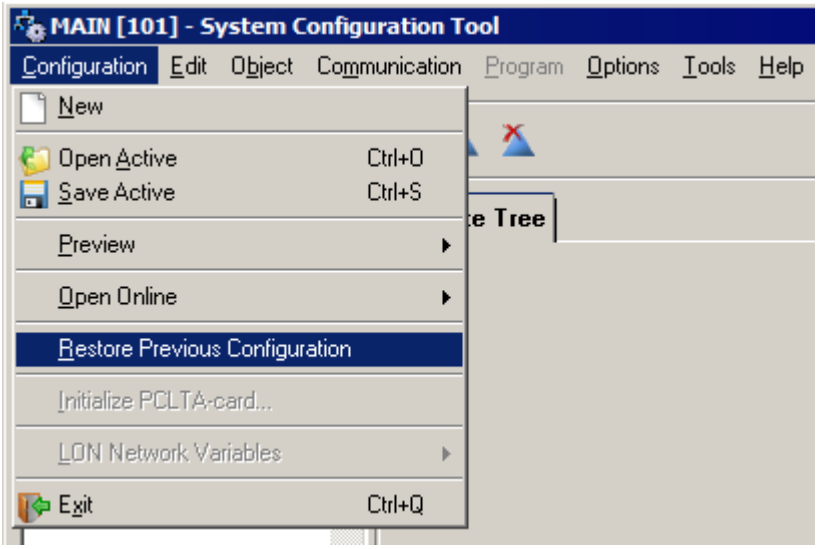


Figure 93: Restore previous configuration

5.4.11 Online configuration

The online configuration is the current running configuration in the SYS600 system.

5.4.11.1 Loading online configuration

Load the current SYS600 system configuration in the tool either all at once or node by node.

To load the current configuration all at once, select **Configuration/Open Online/All**, as shown in Figure 94.

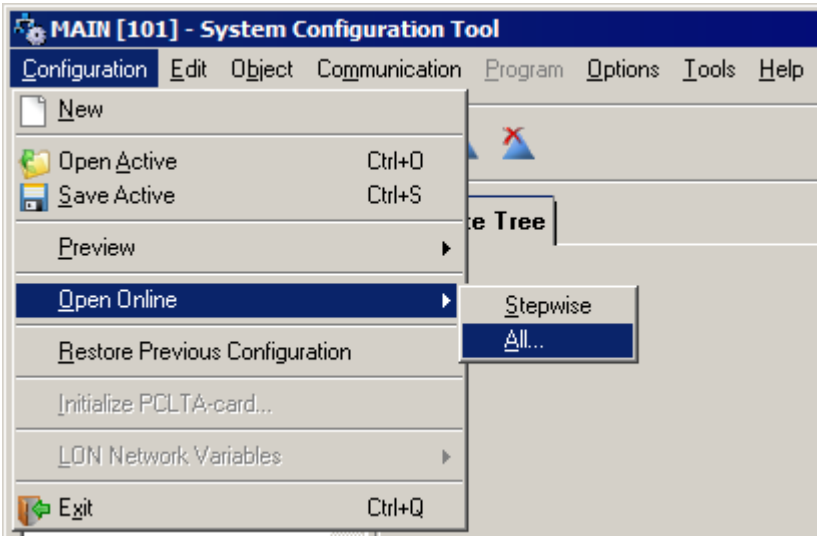


Figure 94: Open Online configuration 1

Loading the online configuration all at once can be a lengthy operation under the following circumstances:

- When the configuration consists a great amount of devices
- When number of devices are located behind slow communication lines or do not respond at all

Thus, it is recommended to open the current online configuration stepwise, for example, the actual loading is not done until the node is expanded. To load the current configuration step by step, select **Configuration/Open Online/Stepwise**, as shown in Figure 95.

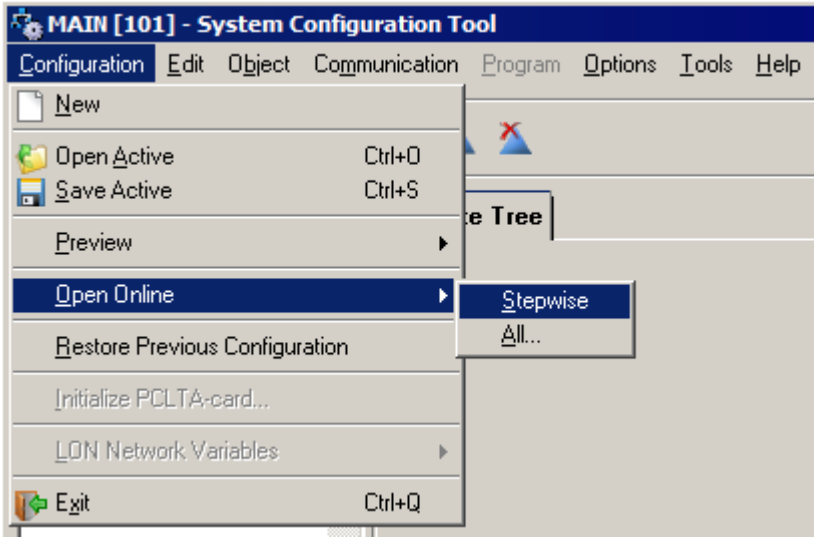


Figure 95: Open Online configuration 2

After either of the above action, the System Configuration Tool is changed to online mode. The background color of the object and attribute trees are set to Lavender and the text in the lower-right corner is changed to Online when online mode is selected.

Under MicroSCADA Configuration node there is a node called Station Type Definitions, as shown in Figure 96. This object includes all the different station types, which are displayed when the Station Type Definitions node is expanded. It is not possible to delete this object.

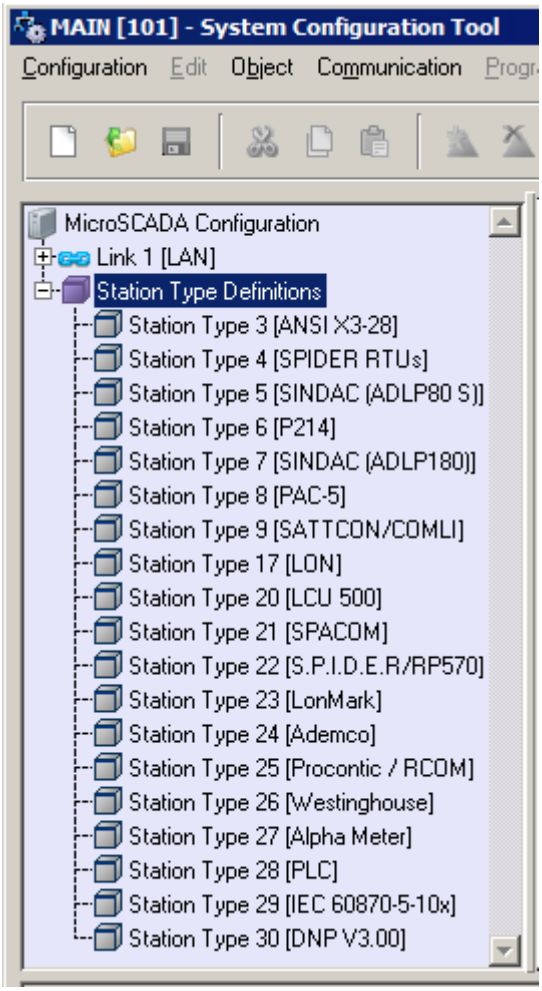


Figure 96: Station type definitions in the online configuration

5.4.11.2 Saving online configuration

If the online configuration is loaded using through **Configuration/Open Online/All**, it can be saved using the **Configuration/Save Active** command. The following notification dialog is displayed on the window:

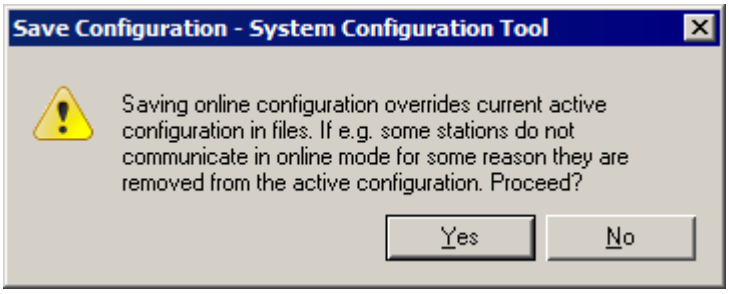


Figure 97: Dialog informing the user that saving online configuration overrides current configuration files

- Click **Yes** to override the current active configuration in the System Configuration Tool and save the online configuration as the default configuration.
- Click **No** to cancel the saving operation. If the menu bar command **Configuration/Save Active** is selected, the configuration should include a Link object and a NET Node object related to the link.

If the INTEGRATED Link object and NET Node object are not included in the object tree, the PC-NET does not start up successfully. The tool informs the user about the invalid PC-NET configuration, see Figure 98. Save the configuration where only IEC 61850 Node objects are included by clicking **Yes**. If the configuration must be valid for PC-NET, click **No** and add the necessary objects to the object tree.

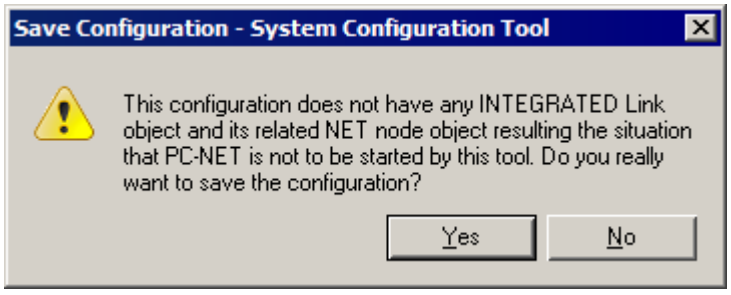


Figure 98: Dialog informing the user that the configuration doesn't include PC-NET

5.4.12 Taking configuration in use and out of use

When taking LONWORKS lines and stations in use in the PC-NET, it is essential for the line to be taken in use before any station (on that specific line) is taken in use. Likewise, all the stations must be taken out of use before the line is taken out of use.

To take the configuration in use, change the IU (In Use) attribute values to In Use mode in the System Configuration Tool.

1. From the menu bar, select **Configuration/Open Active** if the configuration is not open already.
2. In the Object tree, select the desired line.

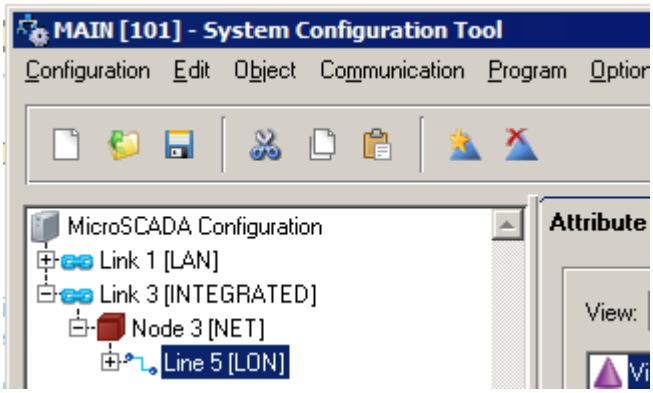


Figure 99: LON line number five is selected in the Object tree

3. Double-click **Basic Line Attributes** or click the **+** sign in the Attribute tree. This expands the **Basic Line Attributes** group to display all the attributes in it.

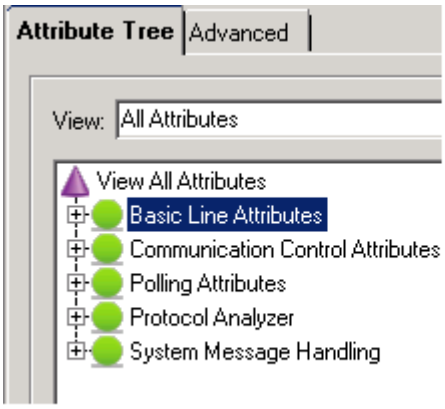


Figure 100: Line five (LON) attribute groups

4. If the IU (In Use) attribute value is 0 (Not In Use), change it to 1 (In Use) in the following way:
   - In the Attribute tree, click the IU attribute line.
   - In the attribute editing area, select the **IU** check box (In Use state).

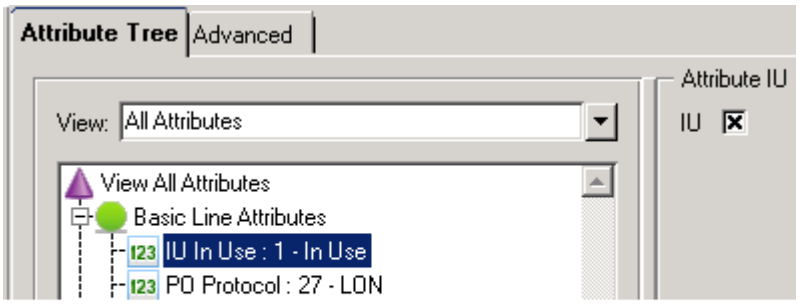


Figure 101: IU Attribute in the In Use (1) state

5. Select **Configuration/Save Active** from the menu bar.

After taking the line in use, take the stations in that line in use as well.

5.4.13 Reallocating stations

It is possible to cut, copy and paste the already defined objects in the configuration tree. When an object is cut, it is also deleted from the configuration tree.

During the cutting/copying and pasting action, all the related information is copied and reallocated. This includes attribute values, possible user-defined SCIL programs (stations, NET Lines and NET Nodes) and signals (data points for REx, LMK, SPA, STA, PLC and DNP stations).

5.4.13.1 Cutting and copying stations

1. Select the desired object be to cut or copied from the configuration tree.
2. Select **Edit/Cut** or **Edit/Copy** from the menubar.

The selected object is cut or copied to the clipboard.

During cutting/copying the contents of the signal data for the REx, LMK, SPA, STA, PLC and DNP stations is assigned to the clipboard.

Cutting an object is not possible if the selected object includes child objects.

5.4.13.2 Pasting stations

1. In the configuration tree, select the parent object for the object on the clipboard.
2. Select **Edit/Paste** from the menu bar.

The pasted object is a child object for the selected parent object.

During the **Edit/Paste** sequence, the possible signal data is taken into use from the clipboard. This concerns REx, LMK, SPA, STA, PLC and DNP stations only.

The System Configuration Tool guards against incorrect configuration: it is not possible to paste a SPA device directly under a LON line (an LMK device is needed) or to paste an LMK device under a SPA line.

The configuration object that is copied into the clipboard can be pasted several times. The pasted object number collection is based either on the definition of the minimum and maximum object numbers (for example from 1 to 10) or on the definition of individual object numbers (for example 4, 5, 8, 10). The **Paste as Range** function can be found in the **Edit** menu.

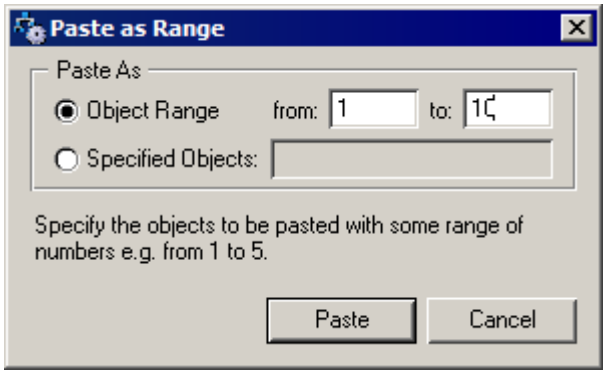


Figure 102: Minimum object number is defined to be 1 and the maximum object number 10

If the copied object includes a set of child objects (for example, copied LMK station includes several SPA stations), the pasting of the object (LMK station) does not include pasting of the child objects (SPA stations). The child objects need to be copied separately.

System Configuration Tool includes error handling during the pasting of objects.

5.4.14 Previewing

The contents of a currently open configuration file can be displayed in the tool using the **Preview** function. In this function, the data is shown in SCIL clauses.

To display the configuration data, select **Configuration/Preview**, as shown in Figure 103. The SCIL clauses are displayed in the SCIL Editor.

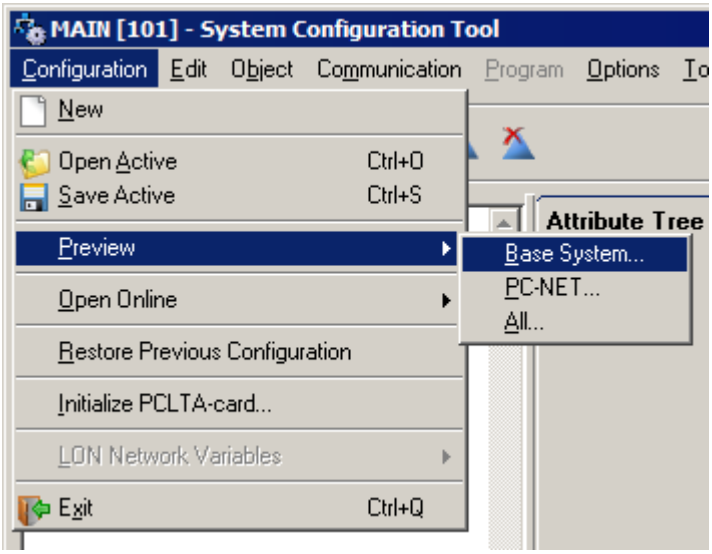


Figure 103: Preview options

SCIL programming is not possible by using the **Preview** function.

5.4.15 User-defined programs

It is possible to make user-defined SCIL programs for the NET Node, NET Line and Stations. With these programs, lines and process units can be modified with features, which are not yet supported by the configuration tool. For the NET, protocols and devices that are not yet supported for the lines in the System Configuration Tool can be created.

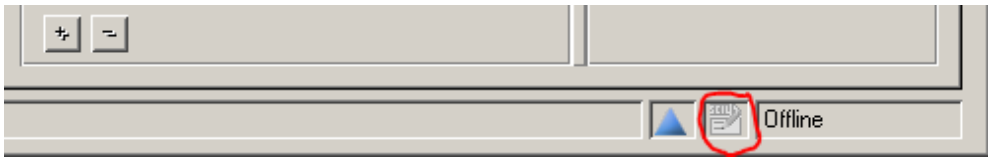


Figure 104: Symbol for the user-defined programs is disabled

Information for user-defined SCIL programs with the following meanings can be found in the status bar of the System Configuration Tool:

- If an enabled symbol exists, the selected object includes a user-defined SCIL program.
- If a disabled symbol exists, it is possible to include a user-defined SCIL program for the selected object, but nothing has been attached yet.
- If no symbol exists, it is not possible to include a user-defined SCIL program for the selected object.

To edit user-defined programs:

1. Select the object to be modified.
   If the symbol exists in the status bar, the SCIL program can be edited or a new one can be created.
2. Select **Program/User-Defined**, as shown in Figure 105.

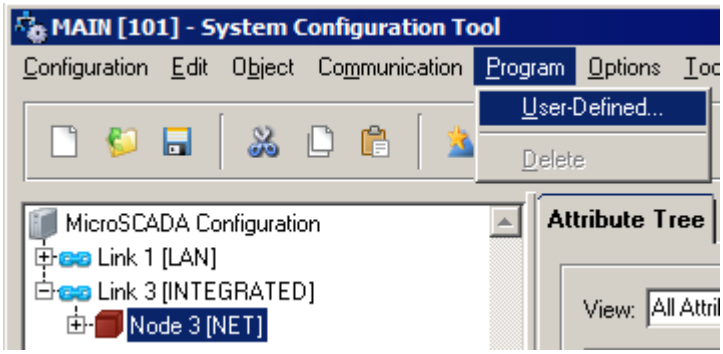


Figure 105: SCIL Editor is opened

3. Edit the program using the variables listed in the comments of the program.

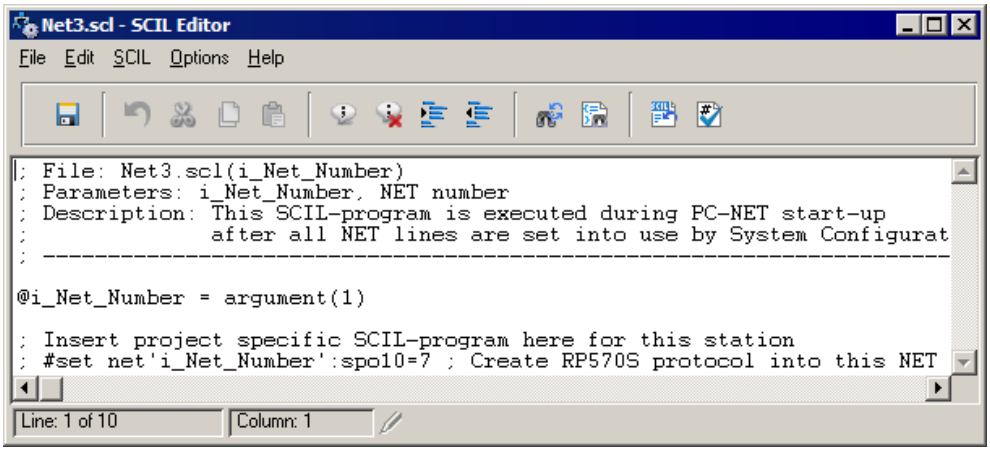


Figure 106: NET3.SCL file in the SCIL Editor

4. Update and exit the program editor.

5.4.16 Sending general object handling command

This attribute is included in the System Configuration Tool when the tool is used in online mode.

1. Select a REX station in the Object tree.
2. Select **Communication/General Object Handling command** to open the **General Object Handling command** dialog, as shown in Figure 107 and Figure 108.

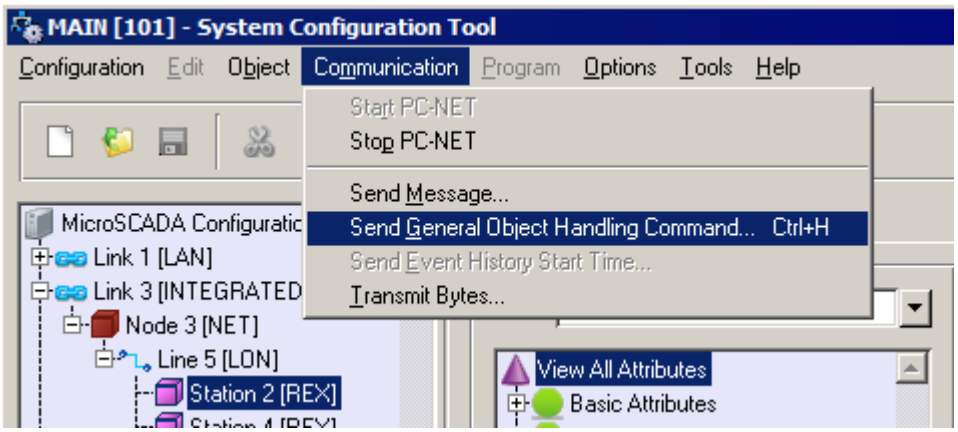


Figure 107: General Object Handling command dialog is displayed

3. Type the appropriate values.
4. Click **Send** to send the command to the selected REX station. The **Close** button closes the dialog without sending any command.
   For example:

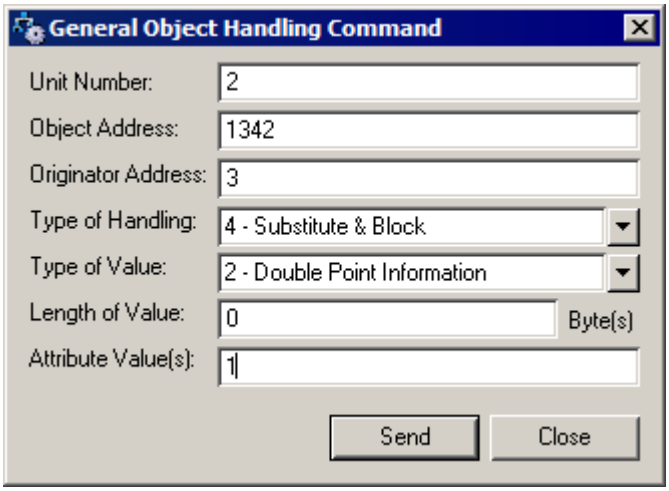


Figure 108: General Object Handling command dialog with example values

Entering the same information as in Figure 108 and clicking the **Send** button or pressing ENTER sends the following SCIL command to REX station number one:

```
#SET STA1:SGO = (1, 1342, 3, 4, 2, 0, 1)
```

5.4.17 Defining general environment definitions

The attribute tree definitions and PC-NET start-up delay time can be set in the **Environment** dialog.

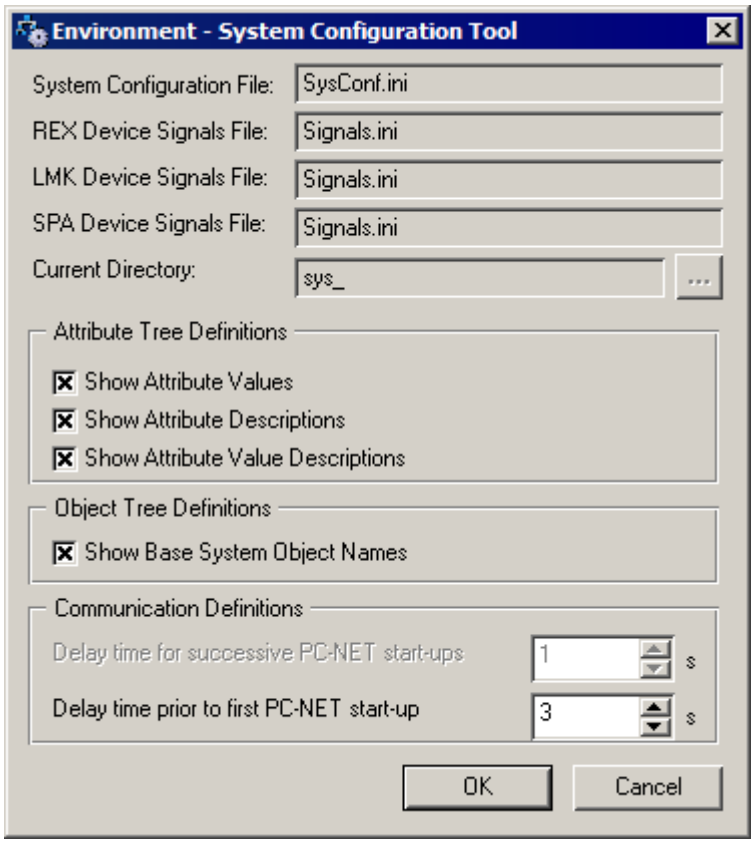


Figure 109: Environment dialog of System Configuration Tool

The setting of delay time for successive PC-NET start-ups has meaning only when more than one PC-NET is installed. In a single PC-NET configuration the setting is disabled in the dialog.

5.4.18 System monitoring

System Self Supervision is always dedicated to a certain SYS600 application, which includes sets of command procedures, event channels, time channels, process objects, data objects and parameter files. System Self Supervision functionality can be enabled in the SYS600 application by either of the following ways:

- By installing the Application status standard function from the SA_LIB\Supervision category
- By selecting the enabled state from the **System Self Supervision** dialog in the System Configuration Tool

To open the **System Self Supervision** dialog, select **Settings/System Self Supervision** in the System Configuration Tool, as shown in Figure 110.

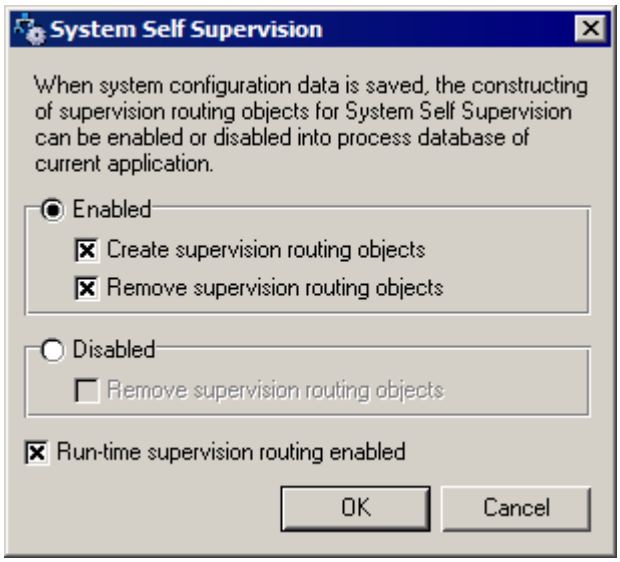


Figure 110: Enabling and disabling the System Self Supervision

When the System Self Supervision functionality is enabled in SYS600 application, the System Configuration Tool does not create supervision routing objects for all the included configuration objects by default. Therefore, the user needs to select the appropriate option from the dialog. To remove the supervision routing objects from the previously included configuration objects, it is also required to set that option in the **System Self Supervision** dialog.

If no Application status standard function is installed from the SA_LIB\Supervision category when System Configuration Tool is accessed for the first time and this dialog is opened, the System Self Supervision is in the disabled state. By default, removing supervision routing from all the previously included configuration objects requires to set that option in the **System Self Supervision** dialog.

If the **System Self Supervision** dialog is accessed when previous SYS_BASCON.COM template is being used, an information dialog is displayed. To enable the system self supervision routing, a new attribute B_SSS_MECH_IN_USE needs to be added in the base system object definition (SYS:B). An example of this attribute can be found from the new template in the file SYS_BASCON$COM.

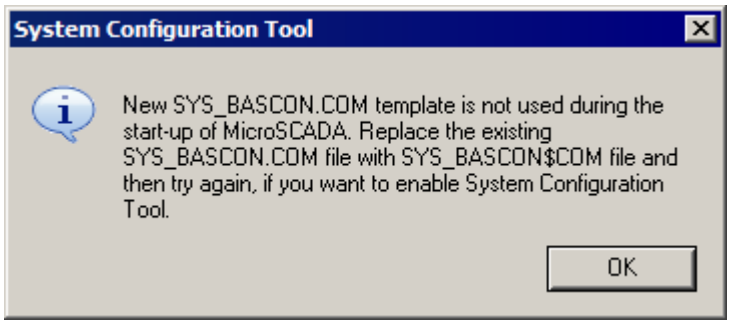


Figure 111: Dialog asking to replace the current SYS_BASCON.COM template to enable the System Self Supervision

When the old SYS_BASCON.COM is used during the start-up of SYS600, the editing of the **System Self Supervision** dialog is disabled.

If a new SYS_BASCON.COM template is used during the start-up of SYS600, it is possible to stop and start the run-time supervision routing in the application. To stop and start the run-time supervision routing, use the **Run-time supervision routing enabled** check box in the

bottom of the **System Self Supervision** dialog. An information dialog displays the message of whether the action was successful or not.

5.4.18.1 Supervision log

The System Configuration Tool includes access to Supervision Log. To enter the **Supervision Log** dialog, select **Tools/Supervision Log** from the menu bar.

The **Supervision Log** displays all the different events in SYS600 and the Windows system. Different log types are:

- Common system messages
- Unknown process objects
- System events from operating system
- Security events from operating system
- Application events from operating system

To select the log type, click **Log** from the menu bar and select the appropriate log type from the menu items. It is possible to set a different filter condition, for example, events from a certain station number, for the events shown in the view.

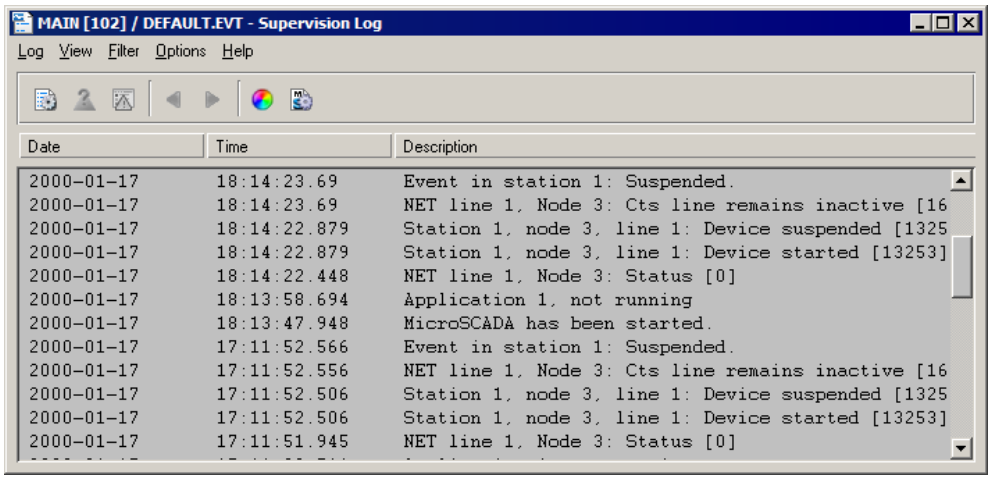


Figure 112: The main view of Supervision Log

5.4.18.2 Supervision Filter Editor

The Supervision Filter Editor can be used for profiling the appearance of the system self supervision information. Open the editor by selecting **Tools** > **System Self Supervision Filter Editor**. The Supervision Filter Editor displays all the categories supported by the System Self Supervision functionality:

- Communication lines
- Stations
- Communication nodes
- Printers
- System events
- Operating system events
- Application events
- LON Clock master

The editor is aware of the active system configuration and displays only the relevant instances of communication lines and station types in accordance with the active configuration.

The appearance of system self supervision profile defines whether the information is collected as a log entry, or whether the appropriate event appears in the Event and Alarm List. Define the settings for each supervision event by selecting the appropriate check boxes in the tool. The default supervision profile is included in the tool.

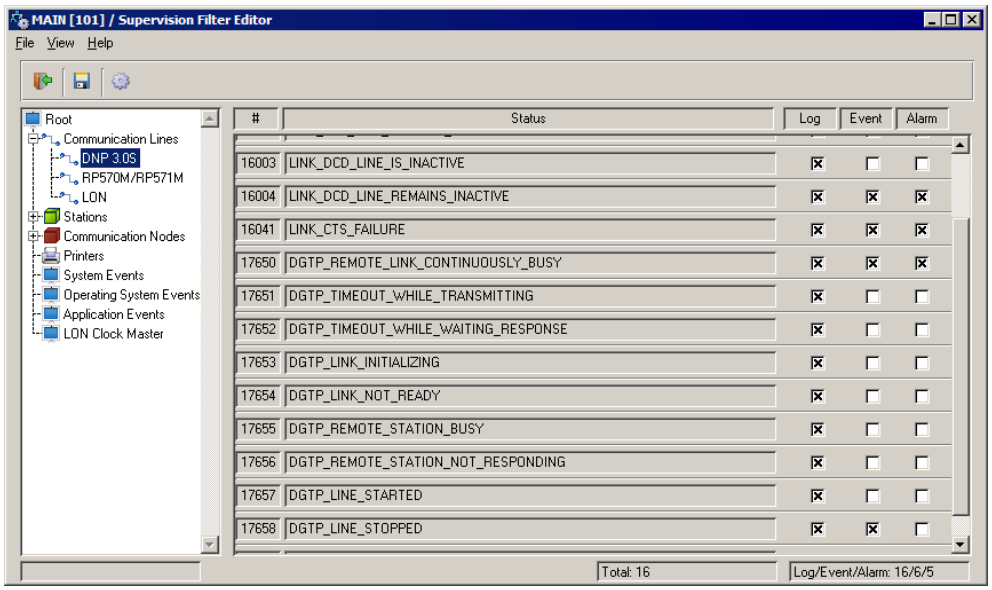


Figure 113: SYS600 Supervision Filter Editor in the System Configuration Tool

5.4.19 Signal engineering

For some PC-NET station types, additional signal engineering information must be configured in order to correctly update the process objects in the MicroSCADA process database. The signal configuration occurs either by using the subtools started by selecting **Tools/Signal Engineering**, or on the **Advanced** tab that appears for STA, PLC and DNP stations.

The following signal engineering characteristics are supported by the tool:

- SPA points for REx stations
- LON points for LMK stations (including LON Clock Master and LON Star Coupler devices)
- SPA points for SPA stations
- Memory areas for STA stations
- Topic items for PLC stations
- Data points for DNP stations

System Configuration Tool is integrated to subtools for handling signal information for stations. For each station type, there is a corresponding configuration tool for managing signal information. To start the subtools, select **Tools/Signal Engineering** from the menu bar. The configuration dialog opens. The dialog includes all the signal information for the selected station.

To transfer the signal information from the subtool, select **Configuration/File/Update** from the subtool's menu bar. Information is also transferred to the System Configuration Tool, when **Configuration/File/Exit** is selected. In each of the subtools, there are options to cut, copy and paste signal information.

5.4.19.1 Indicator for signal information

In the status bar of the System Configuration Tool, there is an indicator for signal information with the following meanings:

- If there is an enabled symbol, the selected object includes signal information.
- If there is a disabled symbol, it is possible to include signal information for the selected object, but no signals are created yet.
- If there is no symbol, it is not possible to include signal information for the selected object.

Figure 114: Indicator shows that the selected object includes signal information

The following features are common to all devices:

- When a station is selected in the configuration tree, the attribute area is updated.
- Select **Tools/Signal Engineering** from the menu bar to see the signal information of the selected station. This operation opens the station **Configuration** page.
- Manage the signals with the **Add**, **Edit** and **Delete** buttons in the **Configuration** page. Signal items can be edited only when the System Configuration Tool is in offline mode. In online mode the buttons **Add**, **Edit** and **Delete** are disabled and the signal configuration can be viewed but not modified.

Add/Edit
Add and **Edit** buttons open the signal **Add/Edit** dialog for entering or changing the signal information. The user interface of this dialog depends on the station type.

OK
The **OK** button accepts the entered values into the signal list of the device and closes the **add/edit** dialog.

Cancel
The **Cancel** button cancels the add/edit operation and closes the **Add/Edit** dialog.

Apply
The **Apply** button accepts the entered values into the signal list without closing the dialog.

- When a device configuration tool is closed, the signals related to the selected device are transferred to the System Configuration Tool. When **Configuration/Save Active** is selected, these signals are saved into the configuration files and they become a part of the configuration data. The device signals are interpreted automatically when the NET communication is starting.
- The SCIL commands which are created from the device signals can be seen by selecting **Configuration/Preview** from the System Configuration Tool menu bar.

To edit the signal information:

1. In the Object Tree, select the station to be engineered.
2. Select **Tools/Signal Engineering** from the menu bar, as shown in Figure 115.

The station configuration page opens for editing.

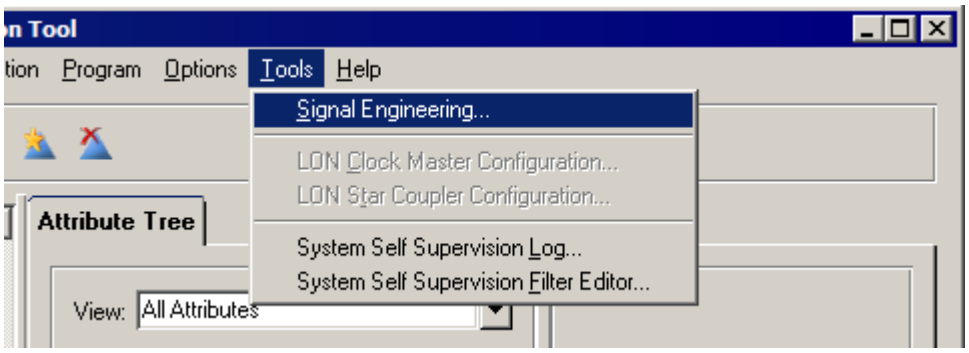


Figure 115: The station configuration page is opened

5.4.19.2 REX, LMK and SPA stations

For more information about the signal engineering for the REX, LMK and SPA stations, see SYS600 Connecting LONWORKS Devices.

5.4.19.3 Topic configuration for PLC stations

Topic configuration is done in the Advanced page for the PLC stations. For more information about the signal engineering for the PLC stations, see SYS600 Modbus Master Protocol.

5.4.19.4 Configuring data points for DNP stations

DNP 3.0 protocol provides versatile possibilities for data polling. In DNP 3.0, the data polling can be configured in a different way in each DNP 3.0 master device. The data polling for DNP master station is defined in the **Advanced** page of the System Configuration Tool, as shown in Figure 116.

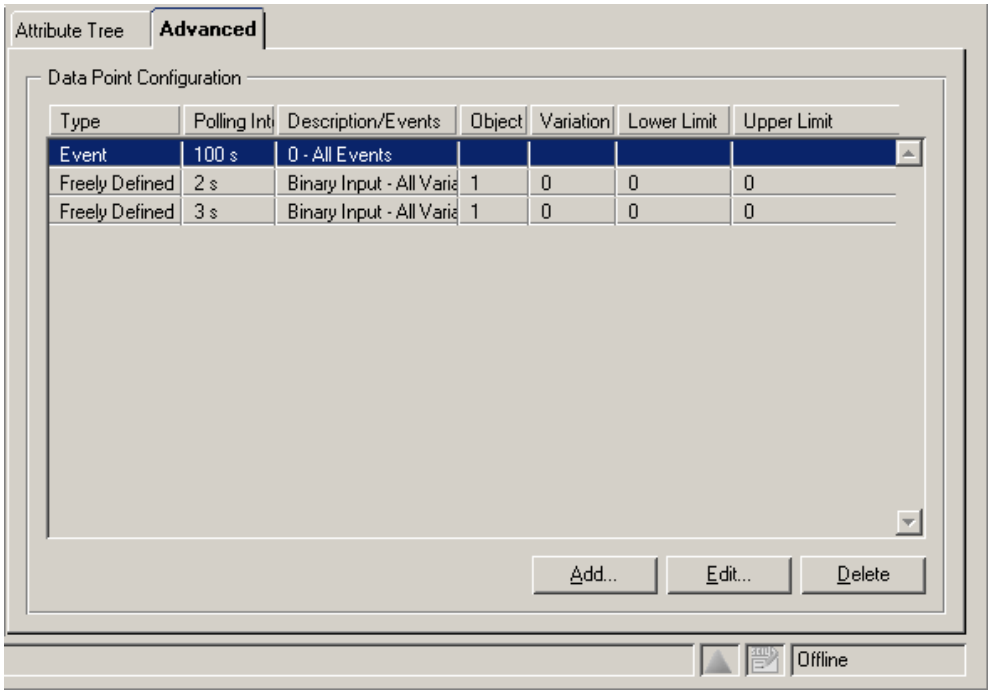


Figure 116: The data point configuration of DNP station in the Advanced page of the System Configuration Tool

To add a new item, click the **Add** button. This action opens the **Add Data Point Item** dialog. In this dialog, the default type is event poll. If the DNP station already contains the defined event poll item, the default type is always a freely defined poll. If the DNP station already includes the maximum number of data point items, the **Add** button is disabled as there can be maximum of fifty freely defined data point items for one DNP device.

To delete the existing data point items, select the appropriate item in the list and click **Delete**. Before the delete operation is done, a notification dialog is displayed to the user. Click **Yes** to delete the selected data point item and refreshes the list. Click **No** to cancel the deletion.

To edit the existing data point item, select the appropriate item in the list and click **Edit** or double-click the data point item. The selected items are displayed in the Data Point Configuration Editor with the existing definitions, as shown in Figure 117. In this dialog, the poll type, polling interval, object, variation, description, number of events and lower/upper limit of index range are defined.

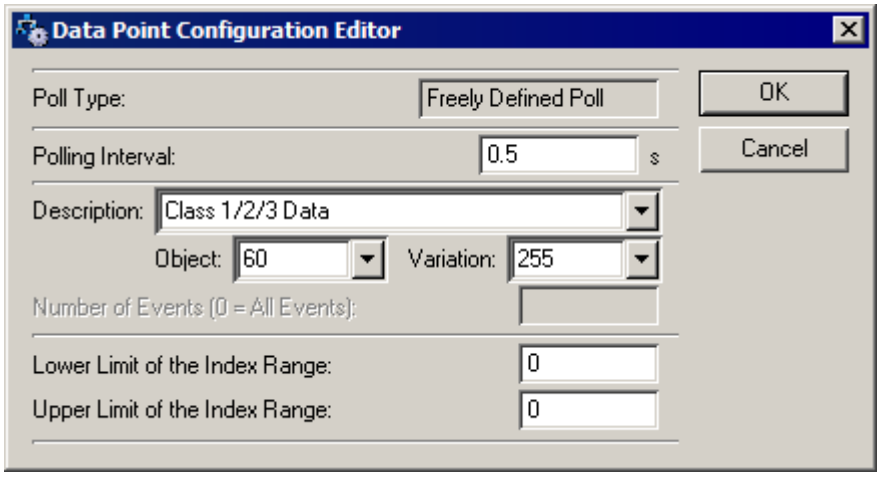


Figure 117: Editing the existing data point item

Poll Type
Pole Type specifies the poll type of a data point item. There may be one event poll and maximum 50 freely defined polls for a DNP station.

Polling Interval
Polling Interval specifies the polling interval as seconds. Setting this parameter to zero stops the poll, which is the default for freely defined poll. For event poll, the default is 100.

Description, Object and Variation
This is a combination of Object and Variation specifies the information element structure for a data point item. It is also possible to select the information element structure directly from the Description list. In both cases, only the relevant Object and Variations appear in the lists.

Number of Events
Number of Events specifies the number of events to be polled. Value 0 indicates that all events are to be polled. Default value is 0.

Lower Limit of Index Range
Lower Limit of Index Range specifies the lower limit of the index range. If 0, all data points with the given data object type and variation are polled. Default value is 0.

Upper Limit of Index Range
Upper Limit of Index Range specifies the upper limit of the index range. If 0, all data points with the given data object type and variation are polled. Default value is 0.

All the above definitions are applicable to the freely defined poll, except the Number of Events definition. For an event poll, only the Polling Interval and Number of Events are applicable.

5.4.19.5 Configuring memory areas for STA stations

For STA stations, the memory area configuration for data items is defined in the Advanced page, as shown in Figure 118.

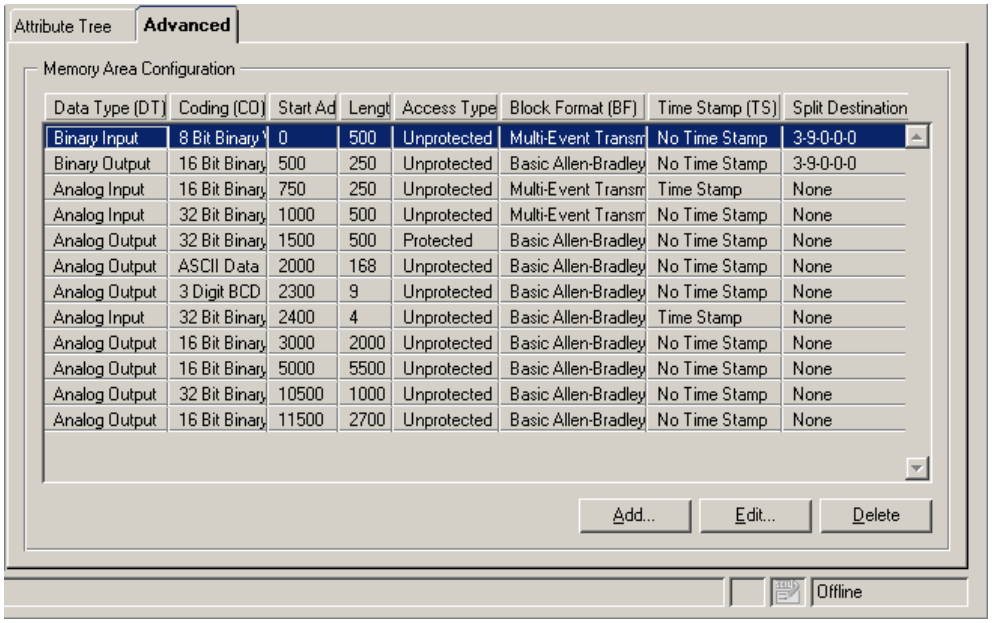


Figure 118: The memory area configuration of STA station in the Advanced page of the System Configuration Tool

To add a new item, click the **Add** button, which opens the **Add Memory Area Item** dialog shown in Figure 119. In this dialog, the default type is binary input or the type of the last added item. If STA station already includes 30 items, the **Add** button is disabled, as the maximum number of the memory area items for each STA device is 30.

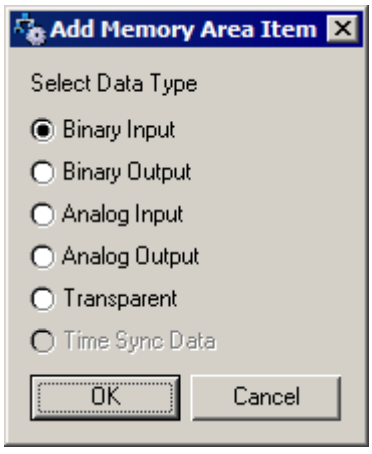


Figure 119: New memory area item, Binary Input, is added to the STA station

To delete the existing memory area items, select the appropriate item in the list and click **Delete**. Before the deletion is done, a notification dialog is displayed to the user. Click **Yes** to delete the selected memory area item and refreshes the list. Click **No** cancel the delete operation.

To edit the existing memory area item select the appropriate item in the list and click **Edit** or double-click the memory area item. The selected items are displayed in the Memory Area Configuration Editor with the existing definitions, as shown in Figure 120. In this dialog, the data type, coding, start address, length, access type, block format, time stamp and split destination are defined.

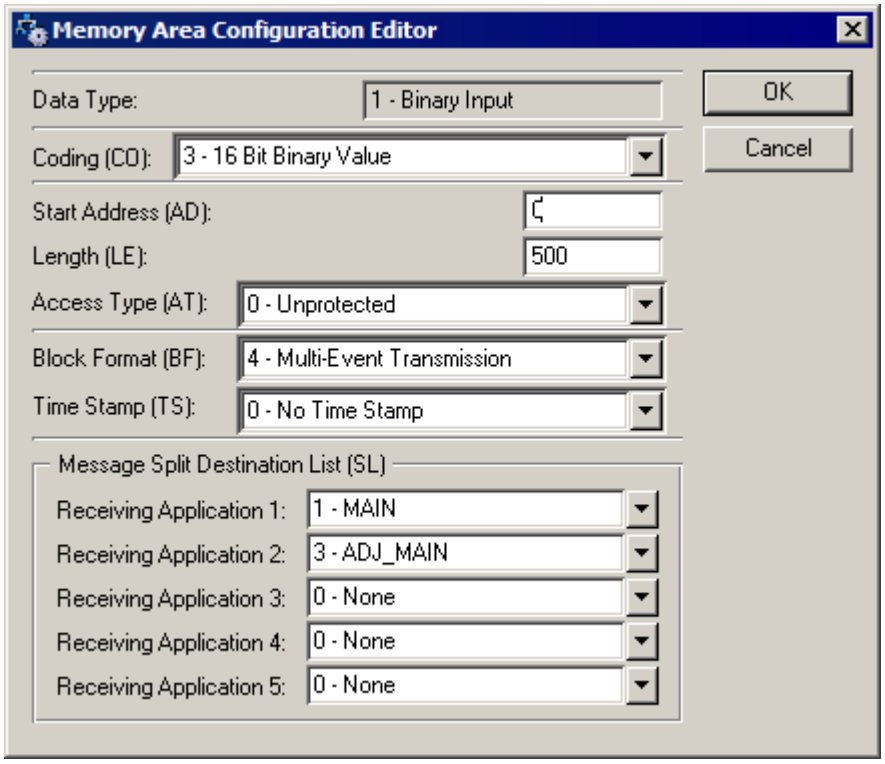


Figure 120: Editing the existing memory area item

Data Type
Data Type specifies the data type of process objects. The following data types of STA device are available: Binary Input, Binary Output, Analog Input, Analog Output, Transparent and Time Sync Data.

Coding
Coding specifies the coding of the data elements in the address interval defined by the memory area. The value of CO attribute tells the communication program how to interpret the data of the memory area.

Values:

1 - 8 Bit Binary Value

2 - 12 Bit Binary Value

3 - 16 Bit Binary Value

4 - 32 Bit Binary Value

5 - 3 Digit BCD Value

6 - 4 Digit BCD Value

7 - Not in Use

8 - Not in Use

9 - 32 Bit Floating Point Value

10 - ASCII Data

11 - 16 Bit Integer Value

12 - 32 Bit Integer Value

Start Address
Start Address specifies the word address of the first word's memory area. Value range: 0 - 32767.

Length
Length specifies the number of words in the memory area. Value range: 0 - 32767.

Access Type
Access Type defines whether the write commands directed to this memory area are protected or unprotected. The attribute is relevant only to Allen Bradley stations. Values:

0 = Unprotected

1 = Protected

Block Format
Block Format states if the spontaneous command messages from the station use the basic format of the protocol, or if an additional address field is used. Values:

1 = Basic Allen-Bradley

2 = Special 1 (the message contains a second word address, which is a BCD coded octal number)

3 = Special 2 (the message contains a second, binary word address)

4 = Multi-Event Transmission (allows transmission of many events with non-continuous addresses in the same telegram)

Time Stamp
Time Stamp states whether the time tagged information is included in spontaneous commands from the station. Values:

0 = No Time Stamp

1 = Time Stamp

Message Split Destination List
Message Split Destination List defines the applications that will receive the message. Receiving applications can only be defined if Message Split attribute of station is defined (SP > 0).

5.5 Base System Object Navigator

The Base System Object Navigator tool is an online tool that provides the following common functionalities:

- Recognizing of the base system objects in SYS600 system
- Viewing of the base system related attributes and their values
- Editing of the base system object related attribute values
- Adding of base system objects

For further information on Base System Object Navigator, see SYS600 System Objects.

As the tool is an online tool, the modified attribute values or added base system objects affect only the running system. If there is need to configure the system permanently, the changes should be made to base system configuration files (SYS_BASCON.COM).

5.6 Communication Engineering tool

MicroSCADA X SYS600 includes a Communication Engineering tool for IEC 61850 OPC Server. This tool is used for the configuration tasks before the use of IEC 61850 OPC Server can be started in the system.

The IET Data Loader in combination with IET600 can also be used to configure the IEC 61850 OPC Server. If system contains IEC 61850 as a NCC protocol, the configuration is possible only using IET600.

5.6.1 Building an object tree

Building an object tree is possible by using Project Explorer of CET, and the purpose is to create the hierarchical communication structure for the project. The required steps for when using Project Explorer in CET are described in the MicroSCADA X IEC 61850 Master Protocol manual.

5.6.2 Configuring objects

For each object found in the object tree, there are set of properties that can be adjusted by using Object Properties of CET. In this way, the communication details, such as the IP Address of the Communication Port for IEC61850 Subnetwork can be set. For further information on configuring object properties, see the SYS600 IEC 61850 Master Protocol (OPC).

5.6.3 Using object tools

When an object is selected in the object tree, the applicable object tools can be found from the **Tools** menu and object tree's **Context** menu. These tools are used when building the object tree, or later on when performing operations with these objects. Examples of such a tools are: SCL Import, Management and Online Diagnostics. For further information, see SYS600 IEC 61850 Master Protocol (OPC).

Appendix A Appendix

1.1 SYS_BASCON$COM template file

All variables are defined in the beginning of the file. The variable definitions section is also divided to groups. The first group is called the Quick configuration part, which includes definitions that can be modified either with the wizard or manually.

```
    ;File: Sys_bascon.com
    ;Description: Standard Base system configuration file
    ; for Single and Hot Stand-By systems
    ; Version 10.0
    ;-------------------------------------------------
    ;
    ;The quick configuration block below is written by the configuration
wizard
                or
    ;it can be modified manually.
    ;
    ;Quick configuration begin
    ;
    #local Hot_Standby = FALSE ;Hot Stand-by enabled/disabled
    #local Apl_Backup = FALSE ;Backup enabled/disabled
    #local Apl_Names = vector("MAIN") ;Application Name vector, main
               applications
    #local BS_Names = vector("SYS_1","SYS_2") ;Base System Node Names / IP
               Addresses
    #local BS_Nodes = vector(9,10) ;Base System Node Numbers
    #local BS_Addresses = vector(209,210) ;Base System Station
               Addresses
    #local BS_Web_Addresses = vector("","") ;Base System Web
               Addresses
    #local This_Node_is = BS_Nodes(1) ;This system 1 or 2 (Always 1 for
single
               system)
    ;
    #local COM500 = vector(FALSE) ;TRUE = COM500i application, FALSE = not
               COM500i application
    #local OPC_Server_Enabled = TRUE ;OPC Server enabled/disabled
    #local OPC_AE_Server_Enabled = vector(FALSE) ;OPC A&E Server
               enabled/disabled
    ;
    ;Quick configuration end
    ;
```

The second group of variable definitions is called the Basic configuration part and it includes all the definitions for base system object configuration. The code includes examples to help the project engineer in the configuration work.

```
    ;Basic configuration begin
    ;
    ;Application numbers. Empty vector = automatic numbering from 1 to
               max_application_number
    ;Hot Stand-by Application numbers in the order:
    ;(MAIN1, MAIN2, ... , WATCH-DOG, ADJ MAIN1, ADJ MAIN2, ... , ADJ
               WATCH-DOG)
    ;An example of three main applications
    ;#local Apl_Names = vector("MAIN1","MAIN2","MAIN3")
    ; MAIN1
    ; . MAIN2
    ; . . MAIN3
    ; . . . WD
```

```
        ; . . . . ADJ_MAIN1
        ; . . . . . ADJ_MAIN2
        ; . . . . . . ADJ_MAIN3
        ; . . . . . . . ADJWD
        ; . . . . . . . .
        ;#local Apl_Numbers = vector(1, 2, 5, 6, 7, 8, 11, 12)
        ;
        #local Apl_Numbers = vector()

        ;Image Stations for System Messages (max. 10 for each main
                  application)
        #local Apl_Image_Stations = vector()
        ;An example. Three applications with one image station each
        ;#local Apl_Image_Stations = vector(vector(list(APL = 15, UN =
                  91)),-
        ; vector(list(APL = 16, UN = 92)),-
        ; vector(list(APL = 17, UN = 93)))

        #local LAN_Link = 1 ;LAN link number
        #local Number_of_Printers = 0 ;Number of Printers
        #local Pri_Numbers = vector(1,2)
        #local Pri_Device_Names =
                  vector("\\PrintServer1\Printername1","\
\PrintServer2\Printername2")
        #local Number_of_VS = 6 ;Number of VS monitors

        ; Base system node whitelisting
        #local BS_Nodes_WL = vector("","") ;for example
                  vector("","10.123.45.62")
         ;when This_Node_is = BS_Nodes(1)
        ;
        ; Other Base System Nodes
        ;
        #local NOD_Numbers = vector(),- ;for example vector(11,12),-
         NOD_Names = vector("NOD_11","NOD_12"),-
         NOD_Addresses = vector(211,212)

        ; Other Base system node whitelisting
        #local OBS_Nodes_WL = vector("","") ;for example
                  vector("10.123.45.67",("10.123.45.68","10.123.45.69"))
        ;
        ; Gateway Nodes or nodes of the remote communication units
        ;
        #local GW_NOD_Numbers = vector(),- ;for example vector(20,22),-
         GW_NOD_Addresses = vector()

        ; Gateway node whitelisting
        #local GW_Nodes_WL = vector("")
        ;
        ; OPC DA and OPC A&E Nodes
        ;
        #local OPC_DA_NOD_Numbers = vector()
        #local OPC_DA_NOD_RN = vector()
        #local OPC_DA_NOD_OP = vector()

        ; An OPC DA example, two OPC DA nodes
        ;#local OPC_DA_NOD_Numbers = vector(101, 102);Node numbers
        ;#local OPC_DA_NOD_RN = vector(0, 0) ;Routing node: 0 = current SYS
                  node
        ;#local OPC_DA_NOD_OP = vector(list(CI =
                  "{CE0322A9-65A9-4268-84D5-DD7A17E94C56}",-
        ; US = "username",-
        ; PW = "password",-
        ; SN = "DA Server 1",-
        ; SK = "",- ;Currently recognized: "AC 800"
         ;ABB AC 800 series
        ; GR = vector(-
        ; list(IG = "Group name",-
```

```
    ; UR = 0,-
    ; PD = 0.0))),-
    ; list(CI = "{2E565242-B238-11D3-842D-0008C779D775}",-
    ; US = "username",-
    ; PW = "password",-
    ; SN = "DA Server 2",-
    ; SK = "",- ;Currently recognized: "AC 800"
     ;ABB AC 800 series
    ; GR = vector(-
    ; list(IG = "Group name",-
    ; UR = 0,-
    ; PD = 0.0))))

    #local OPC_AE_NOD_Numbers = vector()
    #local OPC_AE_NOD_RN = vector()
    #local OPC_AE_NOD_OP = vector()

    ; An OPC A& E example, two OPC A&E nodes
    ;#local OPC_AE_NOD_Numbers = vector(111, 112);Node numbers
    ;#local OPC_AE_NOD_RN = vector(0,0) ;Routing node: 0 = current SYS
              node
    ;#local OPC_AE_NOD_OP = vector(list(CI =
              "{386CAD37379842BA-A6BD-086A022CE803}",-
    ; US = "username",-
    ; PW = "password",-
    ; SN = "AE Server 1",-
    ; SK = "",- ;Currently recognized: "AC 800"
     ;ABB AC 800 series
    ; AA = 0 ),-
    ; list(CI = "{68AEC2B0-93CD-11D1-94E1-0020AFC84400}",-
    ; US = "username",-
    ; PW = "password",-
    ; SN = "AE Server 2",-
    ; SK = "",- ;Currently recognized: "AC 800"
     ;ABB AC 800 series
    ; AA = 0 ))

    #local OPC_Nod_DI = 20 ;Diagnostic interval
    #local OPC_Nod_DT = 2 ;Diagnostic timeout
    ;
    ; OPC Stations
    ;
    #local OPC_DA_Station_Numbers = vector();Station numbers
    #local OPC_DA_Station_Nodes = vector() ;The node numbers of the node
object
              that specifies
     ;the OPC server of the station
    ;An OPC DA station definition example
    ;#local OPC_DA_Station_Numbers = vector(1,2)
    ;#local OPC_DA_Station_Nodes = vector(101,102)

    #local OPC_AE_Station_Numbers = vector();Station numbers
    #local OPC_AE_Station_Nodes = vector() ;The node numbers of the node
object
              that specifies
     ;the OPC server of the station
    ;An OPC A&e station definition example
    ;#local OPC_AE_Station_Numbers = vector(11,12)
    ;#local OPC_AE_Station_Nodes = vector(111,112)
    ;
    ;External Applications for mirroring or for APL-APL communication
    ;
    #local Ext_Apl_Numbers = vector(),- ;for example vector(15,16),-
     Ext_Apl_NA = vector("EXT_1","EXT_2"),-
     Ext_Apl_ND = vector(11,12),-
     Ext_Apl_SN = vector(12,11),- ;Shadowing partner application
     Ext_Apl_TN = vector(1,1),-
     Ext_Apl_EM = 5000,- ;Maximum length of mirroring queue
```

```
     Ext_APl_EP = "KEEP" ;Event queue overflow policy, "DISCARD" or
              "KEEP"
    ;
    ;Proxy Applications
    ;
    #local Proxy_Apl_Numbers = vector(),- ;for example
              vector(101,102),-
     Proxy_Apl_NA = vector("PRO_1","PRO_2"),-
     Proxy_Apl_HS = vector(vector(13,15),vector(14,16))
    ;
    ;Application mapping
    ;
    #local Apl_Mapping
    ;
    ;Base System Object SYS attributes
    ;
    #local Sys_Modify = list(-
     -; Communication Attributes
     ER = 0,- ;Enable Routing 0=disabled, 1=enabled
     TI = 10,- ;Timeout Length
     -; Security Attributes
     HD = list(DENY_EXTERNAL_API_ACCESS = TRUE,-
     REQUIRE_ACP_IP_WHITELISTING = TRUE,-
     REQUIRE_ENCRYPTED_ACP = "NETWORK",-
     REQUIRE_KNOWN_ACP_CERTIFICATE = FALSE,-
     REQUIRE_OPC_AUTHENTICATION = TRUE,-
     RUN_EXTERNAL_API_AS_READ_ONLY = TRUE,-
     RUN_OPC_GUEST_AS_READ_ONLY = TRUE),-
     -; Time Handling Attributes
     TM = "SYS",- ;Time Master, SYS or APL
     TR = "LOCAL",- ;Time Reference, LOCAL or UTC
     TF = 0,- ;Time Format
     -; 0 = yy-mm-dd hh:mm:ss
     -; 1 = dd-mm-yy hh:mm:ss
     -; 2 = mm-dd-yy hh:mm:ss
     -; Memory Handling Attributes
     RC = 8192,- ;Report Cache Size(kB)
     -; Audio Alarm Device
     AA = 0,- ;Audio Alarm Address, 0 = No audio alarm board
     AD = "NONE",- ;Audio Alarm Device, "NONE" = No audio alarm board
     AW = 3,- ;Audio Watchdog Cycle, seconds
     -; SPA device attributes
     SD = "",- ;SPACOM Driver Name
     SP = 0,- ;SPACOM Protocol, 0 = Direct connection of SPACOM not
              allowed
     -; MS-STOOL Settings
     SV = (0,- ;System Variables
     list(t_System_Configuration_File = "sys_/SysConf.ini",-;PC-NET
Configuration
     ;information
     b_Conf_Mech_In_Use = TRUE,- ;enables/disables start-up
              configuration
     b_SSS_Mech_In_Use = TRUE,- ;enables/disables system self supervision
     -;routing
     t_Version = "9.4")),-
     -; Operating System Event Handles Attributes
     OE = 0,- ;1=Enabled, 0=Disabled
     OT = (Bit_Mask(0,1,2,3,4),- ;Application events (Bit 0=ERROR,
1=WARNING,
     ;2=INFORMATION, 3=AUDIT_SUCCESS, 4=AUDIT_FAILURE)
     Bit_Mask(0,1,2,3,4),- ;System events (Bit 0=ERROR, 1=WARNING,
     ;2=INFORMATION, 3=AUDIT_SUCCESS, 4=AUDIT_FAILURE)
     Bit_Mask(0,1,2,3,4))) ;Security events (Bit 0=ERROR, 1=WARNING,
     ;2=INFORMATION, 3=AUDIT_SUCCESS, 4=AUDIT_FAILURE)
    ;
    ; Node Object NOD attributes
    ;
    #local Nod_Modify = list(-
```

```
 DF = 1,- ;Diagnostic event from first FOUND (0=System event not
 -;generated, 1=System event generated)
 DI = 20,- ;Diagnostic interval
 DT = 5) ;Diagnostic timeout
;
; Application Object APL attributes
;
#local Apl_Modify = list(-
 TT = "LOCAL",- ;Translation type
 AS = "COLD",- ;Application state
 PQ = 5,- ;Parallel queues
 AA = 10,- ;Number of APL-APL servers (1..10)
 SR = 1,- ;Shadowing maximum receive wait time in seconds
 HP = "DATABASE",- ;History Logging Policy ("DATABASE", "EVENT_LOG",
          "NONE")
 EE = 1,- ;System Events & Operating System Events (1=Enabled,
          0=Disabled)
 TD = vector(),- ;Application specific text databases
 LA = Apl_LA,- ;Application Language
 LS = vector(Apl_LA),- ;Languages Supported
 PN = list(LOCAL=5432,REMOTE=5432),- ;External Database Port
          Numbers
 QM = (- ;Maximum queue lengths:
 10000,- ; Time channel queue
 10000,- ; Event channel queue
 15000,- ; Parallel queues
 10000),- ; Delayed exec queue
 EM = 5000,- ;Maximum process event queue length
 DI = vector(),-
 DT = vector(),-
 -; UAL Configuration
 -; UA = list(SOURCE="UAL Logging",-
 -;

SERVERS=vector(list(TYPE="UDP",IP_ADDRESS="111.111.111.1",PORT=514),-
 -; list(TYPE="TCP",IP_ADDRESS="111.111.111.2",PORT=1468))),-
 WS = 0,- ;Windows Single Sign-On
 -; 0 or FALSE = Disabled
 -; 1 or TRUE = Enabled
 CE = 0) ;Centralized Account Management
 ; 0 = SYS600 User Account Management
 ; 1 = Centralized Account Management
 ; 2 = Centralized Account Management with role integration

#local Apl_DI = 0,- ;Application diagnostic interval (0 =
          Disabled)
 Apl_DT = 5 ;Application diagnostic timeout
;
; Printer Object PRI attributes
;
#local Pri_Modify = list(-
 -; Common attributes
 TT = "LOCAL",- ;Translation Type
 DC = "LINE",- ;Device Connection
 DT = "TRANSPARENT",- ;Device Type
 -; Printer connection attributes
 ND = 0,- ;Node Number of NET unit (to be defined if DC = "NET")
 -; Printout attributes
 LP = 0,- ;Lines per Page
 PN = 0,- ;Page Number
 -; Printer queue attribute
 QM = 1000,- ;Queue Length Maximum
 -; Printer log attributes
 LD = "",- ;Log Directory
 LL = "DAY",- ;Log Length
 LF = 1000,- ;Log Flush Timeout
 OD = "PRINTER",- ;Output Destination
 -;
```

```
     OJ = 1,- ;Open on Job Basis
     -;
     CX = "") ;Comment Text
    ;
    ;Host stations are typically defined with the PC-NET system
configuration
              tool.
    ;Image stations, gateway stations etc. can be defined here.
    ;
    #local Stations = vector()
    #local Sta_Nodes = vector()
    #local Sta_ST = vector()
    #local Sta_MR = vector()
    #local Sta_H_Apl = vector()
    #local Sta_H_UN = vector()
    #local Sta_I_Apl = vector()
    #local Sta_I_UN = vector()

    ;An example of STA object definitions for NCC level (MR = "IMAGE")
stations
              in the following
    ;
    ;#local Stations = ( 1005, 1012, 1031, 1042, 3051, 91 )
    ;#local Sta_ST = ( "PLC", "SPA", "IEC", "STA", "REX", "STA" )
    ;#local Sta_Nodes = ( 0, 0, 0, 0, 0, 0 )
    ;#local Sta_MR =
              ("IMAGE","IMAGE","IMAGE","IMAGE","IMAGE","IMAGE")
    ;#local Sta_H_Apl = ( 13, 13, 13, 13, 13, 13 )
    ;#local Sta_H_UN = ( 5, 12, 31, 42, 51, 0 )
    ;
    ;An example of STA object definitions for intermediate level (MR =
"BOTH")
              stations in the following
    ;
    ;#local Stations = ( 1005, 1012, 1031, 1042, 3051, 91 )
    ;#local Sta_ST = ( "PLC", "SPA", "IEC", "STA", "REX", "STA" )
    ;#local Sta_Nodes = ( 0, 0, 0, 0, 0, 0 )
    ;#local Sta_MR = ("BOTH", "BOTH", "BOTH", "BOTH", "BOTH", "BOTH"
              )
    ;#local Sta_H_Apl = ( 13, 13, 13, 13, 13, 13 )
    ;#local Sta_H_UN = ( 5, 12, 31, 42, 51, 0 )
    ;#local Sta_I_APL =

(vector(15),vector(15),vector(15),vector(15),vector(15),vector(15) )
    ;#local Sta_I_UN =

(vector(2005),vector(2012),vector(2031),vector(2042),vector(2051),vector(2
091))

    ;
    Basic configuration end
    ;
```

The third group is the Variable declarations part, which includes local variable declarations. A project engineer can add new variables to the end of this part for the site-specific configuration code.

```
    ;
    ;Variable declarations
    ;
    #local Sys,- ;The base system object
     Sys_Permanent = list(),- ;Permanent definitions of the base system
              object
     Sys_Supersede = list(),- ;Definitions to replace obsolete kernel
default
              attribute values
     Sys_ND,- ;The node number of the base system computer
```

```
     Sys_OP,- ;The state of the OPC Data Access server
     Sys_SA,- ;The station address of the base system
     Sys_SH,- ;The state of the shadowing
     Sys_WA,- ;The web address of the base system
     Adj_Sys_ND,- ;The node number of the adjacent base system
              computer
     Adj_Sys_SA ;The station address of the adjacent base system
              computer

    #local Nod,- ;The NOD object
     Nod_Permanent = list(),- ;Permanent definitions of the node
              object
     Nod_Supersede = list(),- ;Definitions to replace obsolete kernel
default
              attribute values
     Nod_NN,- ;The LAN node name of the node or the TCP/IP internet
              address
     Nod_Nb,- ;The node number
     Nod_SA,- ;The station address of the node
     Nod_RN,- ;The node number of the routing node
     Nod_OP,- ;OPC server configuration
     Nod_WL = vector() ;Node whitelisting

    #local Sta,- ;The STA object
     Sta_Nb,- ;The station number
     Sta_ND ;The node number of the communication unit to which the
station is
              connected

    #local Apl,- ;The APL object
     Apl_AP,- ;Application mapping
     Apl_Count,- ;Application count
     Apl_Permanent = list(),- ;Permanent definitions of the application
              object
     Apl_Supersede = list(),- ;Definitions to replace obsolete kernel
default
              attribute values
     Apl_Nb,- ;The application number
     Apl_IS,- ;Image stations for system messages
     Apl_OE,- ;OPC A&E server enabled
     Apl_SA,- ;Shadowing partner web address
     Apl_SC,- ;Shadowing connection time
     Apl_SF,- ;Shadowing flush time
     Apl_SI,- ;Shadowing diagnostic interval
     Apl_SQ,- ;Shadow the event channel queue
     Apl_SY,- ;Time synchonization interval
     Apl_SV = vector() ;System variables of the application

    #local Pri,- ;The PRI object
     Pri_Nb,- ;The printer number
     Pri_Permanent = list(),- ;Permanent definitions of the printer
              object
     Pri_Supersede = list(),- ;Definitions to replace obsolete kernel
default
              attribute values
     Printer_Mapping ;Printer mapping

    #local First_Free_Mon,- ;Variable for monitor mapping definition
     Mon,- ;The MON object
     Monitor_Mapping ;Monitor mapping

    #local Global_Paths,- ;
     Standard_Paths ;

    #local m, n ;Control variables of the loop
    #local Result ;

    #local Global_Variables,- ;Local variables used to delete global
```

```
          variables
     Global_Variable ;
```

The statements section of the SYS_BASCON.COM file contains the SCIL program for creating base system objects, start applications and PC-NET. This section is not intended to be modified by a project engineer.

```
    ;Initialize application numbers if not done yet
    ;
    #if Hot_Standby #then Apl_Count = 2 * (length(Apl_Names) + 1)
    #else_if Apl_Backup #then Apl_Count = 2 * (length(Apl_Names) ) +
              1
    #else Apl_Count = length(Apl_Names)

    #if length(Apl_Numbers) == 0 #then #block
     #loop_with n = 1 .. Apl_Count
     Apl_Numbers(n) = n
     #loop_end
    #block_end

    Apl_Mapping = append(append(Apl_Numbers, Ext_Apl_Numbers),
              Proxy_Apl_Numbers)

    ;Definitions to replace obsolete kernel default attribute values with
              up-to-date values
    ;
    Sys_Supersede = list(-
     DN = 3,- ;Default NET Node Number
     DS = "SPA",- ;Default STA type
     PA = Apl_Numbers(1)) ;Primary application = the first main
              application
    Nod_Supersede = list()
    Apl_Supersede = list()
    Pri_Supersede = list()

    ;Shadowing attributes
    Apl_SC = 120 ;Shadowing maximum connect time in seconds
    Apl_SF = 100 ;Shadowing flush time in milliseconds
    Apl_SI = Apl_SF ;Shadowing diagnostic interval
    Apl_SQ = 0 ;Shadow the Event Channel Queue (0 = Disabled, 1 =
Enabled)
    Apl_SY = 60 ;Shadowing time synchronization interval in seconds

    ; ********************************************
    ; ********************************************
    ; Do not edit contents below this line -
    ; except project specific part in the end
    ; ********************************************
    ; ********************************************
    Sys_Modify = merge_attributes(Sys_Modify, Sys_Supersede)
    Nod_Modify = merge_attributes(Nod_Modify, Nod_Supersede)
    Apl_Modify = merge_attributes(Apl_Modify, Apl_Supersede)
    Pri_Modify = merge_attributes(Pri_Modify, Pri_Supersede)

    #if Hot_Standby #then Sys_SH = 1
    #else_if Apl_Backup #then Sys_SH = 1
    #else Sys_SH = 0

    #if OPC_Server_Enabled #then Sys_OP = 1
    #else Sys_OP = 0

    ;*********************************************
    ;Statements for creating Base System Objects (B)
    ;*********************************************
    #if not Hot_Standby #then This_Node_is = BS_Nodes(1) ;to be sure!
```

```
#case This_Node_is
 #when BS_Nodes(1) #block
 Sys_ND = BS_Nodes(1)
 Sys_SA = BS_Addresses(1)
 Sys_WA = BS_Web_Addresses(1)
 #if Hot_Standby #then #block
 Adj_Sys_ND = BS_Nodes(2)
 Adj_Sys_SA = BS_Addresses(2)
 Apl_SA = BS_Web_Addresses(2)
 #block_end
 #block_end
 #when BS_Nodes(2) #block
 Sys_ND = BS_Nodes(2)
 Sys_SA = BS_Addresses(2)
 Sys_WA = BS_Web_Addresses(2)
 Adj_Sys_ND = BS_Nodes(1)
 Adj_Sys_SA = BS_Addresses(1)
 Apl_SA = BS_Web_Addresses(1)
 #block_end
#case_end

;*********************************************
;Base System Object (SYS)
;*********************************************
Standard_Paths = do(read_text("/STool/Def/Path_Def.txt"))

Sys_Permanent = list(-
 ND = Sys_ND,- ;Node number
 SA = Sys_SA,- ;Station address
 WA = Sys_WA,- ;Web address
 SH = Sys_SH,- ;Shadowing enabled
 OP = Sys_OP,- ;OPC Server
 FS = "NEVER",- ;File synchronization criteria
 PH = Standard_Paths)

Sys = merge_attributes(Sys_Modify, Sys_Permanent)

#create SYS:B = Sys

;*********************************************
;LAN link (LIN)
;*********************************************
#create LIN'LAN_Link':B = list(LT = "LAN")

;*********************************************
;Base system nodes (NOD) for Hot Stand-by system
;*********************************************
#if Hot_Standby or Apl_Backup #then #block
 Nod = list()
 #loop_with n = 1 .. length(BS_Nodes)
 Nod_Nb = BS_Nodes(n)
 Nod_SA = BS_Addresses(n)
 Nod_NN = BS_Names(n)
 Nod_WL = BS_Nodes_WL(n)
 Nod_Permanent = list(SA = Nod_SA, NN = Nod_NN, LI = LAN_link, WL =
          Nod_WL)
 Nod = merge_attributes(Nod_Modify, Nod_Permanent)
 #create NOD'Nod_Nb':B = Nod
 #loop_end
#block_end

;*********************************************
;Other base system nodes (NOD) e.g. for Mirroring
;*********************************************
Nod = list()
#loop_with n = 1 .. length(NOD_Numbers)
 Nod_Nb = NOD_Numbers(n)
 Nod_SA = NOD_Addresses(n)
```

```
 Nod_NN = NOD_Names(n)
 Nod_WL = OBS_Nodes_WL(n)
 Nod_Permanent = list(SA = Nod_SA, NN = Nod_NN, LI = LAN_link, WL =
          Nod_WL)
 Nod = merge_attributes(Nod_Modify, Nod_Permanent)
 #create NOD'Nod_Nb':B = Nod
#loop_end

;*********************************************
;Gateway nodes (NOD)
;*********************************************
Nod = list()
#loop_with n = 1 .. length(GW_NOD_Numbers)
 Nod_Nb = GW_NOD_Numbers(n)
 Nod_SA = GW_NOD_Addresses(n)
 Nod_WL = GW_Nodes_WL(n)
 Nod_Permanent = list(SA = Nod_SA, LI = LAN_link, WL = Nod_WL)
 Nod = merge_attributes(Nod_Modify, Nod_Permanent)
 #create NOD'Nod_Nb':B = Nod
#loop_end

;*********************************************
;Printers (PRI)
;*********************************************
Printer_Mapping(1 .. max_printer_number) = 0
#do read_text("sys_/pr_default.dat") ;Control Sequences for the
transparent
          printer
#loop_with n = 1 .. Number_of_Printers
 Pri_Nb = Pri_Numbers(n)
 Printer_Mapping(Pri_Nb) = Pri_Nb
 Pri_Permanent = list(CS = %CS, HF = %HF, SD =
          Pri_Device_Names(n))
 Pri = merge_attributes(Pri_Modify, Pri_Permanent)
 #create PRI'Pri_Nb':B = Pri
#loop_end

;*********************************************
;Classic Monitors (MON)
;*********************************************
First_Free_Mon = 1
Monitor_Mapping(1 .. max_monitor_number) = 0
#loop_with n = 0 .. (Number_of_VS - 1)
 Mon = First_Free_Mon
 First_Free_Mon = First_Free_Mon + 1
 Monitor_Mapping(Mon) = -1
 #create MON'Mon':B = list(TT = "LOCAL", DT = "VS")
#loop_end

;*********************************************
;Applications (APL)
;*********************************************
Global_Paths = do(read_text("sys_/sys_bascon.scl"),
          "GLOBAL_PATHS")

;The usage of OI & OX -attributes
Apl_SV(15) = list(-
 Process_Objects=list(-
 OI=list(-
 Title1=VECTOR("Substation"),-
 Title2=VECTOR("Bay"),-
 Title3=VECTOR("Device"),-
 Title4=VECTOR(""),-
 Title5=VECTOR(""),-
 Length1=10,-
 Length2=15,-
 Length3=5,-
 Length4=0,-
```

```
 Length5=0,-
 Field1=VECTOR("STA"),-
 Field2=VECTOR("BAY"),-
 Field3=VECTOR("DEV"),-
 Field4=VECTOR(""),-
 Field5=VECTOR("")),-
 OX=list(-
 Title1=VECTOR("Object text"),-
 Length1=30)))

#if Hot_Standby #then #block

 ;*** LOCAL WATCHDOG ***
 Apl_Nb = Apl_Numbers(length(Apl_Names) + 1)
 Apl_Permanent = list(-
 NA = "WD",- ;Application name
 AS = "HOT",- ;Application state
 PQ = 2,- ;Parallel queues
 HP = "NONE",- ;History Logging Policy ("DATABASE", "EVENT_LOG",
          "NONE")
 EE = 1,- ;System Events & Operating System Events (1=Enabled,
          0=Disabled)
 MO = Monitor_Mapping,- ;Monitor mapping
 PR = Printer_Mapping,- ;Printer mapping
 AP = spread(APL'Apl_Nb':BAP, Apl_Mapping, Apl_Mapping)) ;Application
          mapping

 Apl = merge_attributes(Apl_Modify, Apl_Permanent)

 #loop_with n = 1 .. max_application_number
 Apl_AP = Apl.AP(n)
 #if Apl_AP > 0 #then #block
 Apl.DT(n) = Apl_DT
 Apl.DI(n) = Apl_DI
 #block_end
 #loop_end

 #create APL'Apl_Nb':B = Apl

 ;*** ADJACENT WATHDOG ***
 Apl_Nb = Apl_Numbers(2 * (length(Apl_Names) + 1))
 #create APL'Apl_Nb':B = list(-
 NA = "ADJ_WD",- ;Application name
 TT = "EXTERNAL",- ;Translation type
 ND = Adj_Sys_ND,- ;Node number
 TN = Apl_Numbers(length(Apl_Names) + 1)) ;Translated object
          number

 #loop_with m = 1 .. length(Apl_Names)

 ;*** LOCAL HSB APPLICATION ***
 Apl_Nb = Apl_Numbers(m)
 #if length(OPC_AE_Server_Enabled) < m #then Apl_OE = 0
 #else_if OPC_AE_Server_Enabled(m) #then Apl_OE = 1
 #else Apl_OE = 0
 #if length(Apl_Image_Stations) < m #then Apl_IS = vector()
 #else APL_IS = Apl_Image_Stations(m)
 Apl_Permanent = list(-
 NA = Apl_Names(m),- ;Application name
 AS = "COLD",- ;Application state (started by WD)
 OE = Apl_OE,- ;OPC A&E Server
 PH = Global_Paths,- ;Global paths
 SV = Apl_SV,- ;System variables (reserved)
 SA = Apl_SA,- ;Shadowing Partner Web Address
 SC = Apl_SC,- ;Shadowing maximum connect time in seconds
 SF = Apl_SF,- ;Shadowing flush time in milliseconds
 SI = Apl_SI,- ;Shadowing diagnostic interval
 SY = Apl_SY,- ;Shadowing time synchronization interval in
```

```
          seconds
 SQ = Apl_SQ,- ;Shadow the Event Channel Queue
 SN = Apl_Numbers(length(Apl_Names) + 1 + m),- ;Shadow
          application
 SW = Apl_Numbers(length(Apl_Names) + 1),- ;Shadow watchdog
 IS = Apl_IS,- ;Image Stations for System Messages
 MO = Monitor_Mapping,- ;Monitor mapping
 PR = Printer_Mapping,- ;Printer mapping
 AP = spread(APL'Apl_Nb':BAP, Apl_Mapping, Apl_Mapping)) ;Application
          mapping

 #if m <= length(COM500) #then #block

 #if COM500(m) #then - ;COM500i Application
 Apl_Permanent = merge_attributes(Apl_Permanent, -
 list(PQ = 16,- ;Number of parallel queues
 QD = (1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1))) ;Parallel queue dedication

 #block_end

 Apl = merge_attributes(Apl_Modify, Apl_Permanent)

 #loop_with n = 1 .. max_application_number
 Apl_AP = Apl.AP(n)
 #if Apl_AP > 0 #then #block
 Apl.DT(n) = Apl_DT
 Apl.DI(n) = Apl_DI
 #block_end
 #loop_end

 #create APL'Apl_Nb':B = Apl

 ;*** ADJACENT HSB APPLICATION ***
 Apl_Nb = Apl_Numbers(length(Apl_Names) + 1 + m)
 #create APL'Apl_Nb':B = list(-
 NA = substr("ADJ_" + Apl_Names(m), 1, 10),- ;Application name
 TT = "EXTERNAL",- ;Translation type
 ND = Adj_Sys_ND,- ;Node number
 TN = Apl_Numbers(m)) ;Translated object number

 #loop_end

#block_end
#else #block ; *** Single System ***

 #loop_with m = 1 .. length(Apl_Names)

 Apl_Nb = Apl_Numbers(m)
 #if length(OPC_AE_Server_Enabled) < m #then Apl_OE = 0
 #else_if OPC_AE_Server_Enabled(m) #then Apl_OE = 1
 #else Apl_OE = 0
 #if length(Apl_Image_Stations) < m #then Apl_IS = vector()
 #else APL_IS = Apl_Image_Stations(m)
 Apl_Permanent = list(-
 NA = Apl_Names(m),- ;Application name
 AS = "HOT",- ;Application state
 OE = Apl_OE,- ;OPC A&E Server
 PH = Global_Paths,- ;Global paths
 SV = Apl_SV,- ;System variables (reserved)
 IS = Apl_IS,- ;Image Stations for System Messages
 MO = Monitor_Mapping,- ;Monitor mapping
 PR = Printer_Mapping,- ;Printer mapping
 AP = spread(APL'Apl_Nb':BAP, Apl_Mapping, Apl_Mapping)) ;Application
          mapping

 #if m <= length(COM500) #then #block

 #if COM500(m) #then - ;COM500i Application
```

```
     Apl_Permanent = merge_attributes(Apl_Permanent, -
     list(PQ = 16,- ;Number of parallel queues
     QD = (1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1))) ;Parallel queue
              dedication

     #block_end

     ;System Events & Operating System Events Enabled in Primary
Application
     #if Apl_Nb == SYS:BPA #then Apl_Permanent =
merge_attributes(Apl_Permanent,
              list(EE = 1))

     ;Application Backup Enabled
     #if Apl_Backup #then #block
     Apl_Permanent = merge_attributes(Apl_Permanent, list(SN =
              Apl_Numbers(length(Apl_Names) + m + 1),-
     SW = Apl_Numbers(length(Apl_Names) + 1)))
     #block_end

     Apl = merge_attributes(Apl_Modify, Apl_Permanent)

     #loop_with n = 1 .. max_application_number
     Apl_AP = Apl.AP(n)
     #if Apl_AP > 0 #then #block
     Apl.DT(n) = Apl_DT
     Apl.DI(n) = Apl_DI
     #block_end
     #loop_end

     #create APL'Apl_Nb':B = Apl

     #loop_end

    #block_end

    ;***********************************************
    ;Application Backup Enabled
    ;***********************************************
    #if Apl_Backup #then #block

     ;Watchdog application
     Apl_Nb = Apl_Numbers(length(Apl_Names) + 1)
     Apl_Permanent = list(-
     NA = "WD",- ;Application name
     AS = "HOT",- ;Application state
     PQ = 2,- ;Parallel queues
     HP = "NONE",- ;History Logging Policy ("DATABASE", "EVENT_LOG",
              "NONE")
     EE = 1,- ;System Events & Operating System Events (1=Enabled,
              0=Disabled)
     MO = Monitor_Mapping,- ;Monitor mapping
     PR = Printer_Mapping,- ;Printer mapping
     AP = spread(APL'Apl_Nb':BAP, Apl_Mapping, Apl_Mapping)) ;Application
              mapping

     Apl = merge_attributes(Apl_Modify, Apl_Permanent)

     #loop_with n = 1 .. max_application_number
     Apl_AP = Apl.AP(n)
     #if Apl_AP > 0 #then #block
     Apl.DT(n) = Apl_DT
     Apl.DI(n) = Apl_DI
     #block_end
     #loop_end

     #create APL'Apl_Nb':B = Apl
```

```
 ;Backup applications

 #loop_with m = 1 .. length(Apl_Names)
 Apl_Nb = Apl_Numbers(length(Apl_Names)) + m + 1
 Apl = list(-
 TT = "LOCAL",- ;Translation Type
 NA = substr("BCK_" + Apl_Names(m), 1, 10),- ;Application name
 AS = "COLD",- ;Application state
 SN = Apl_Numbers(m),- ;Shadow application
 SW = Apl_Numbers(length(Apl_Names) + 1),-
 AP = spread(APL'Apl_Nb':BAP, Apl_Mapping, Apl_Mapping)) ;Application
          mapping
 #create APL'Apl_Nb':B = Apl
 #loop_end
#block_end

;*********************************************
;External Applications e.g. for Mirroring
;*********************************************
#loop_with n = 1 .. length(Ext_Apl_Numbers)
 Apl_Nb = Ext_Apl_Numbers(n)
 Apl_Permanent = list(-
 TT = "EXTERNAL",-
 NA = Ext_Apl_NA(n),- ;Application name
 ND = Ext_Apl_ND(n),- ;Host node number
 SN = Ext_Apl_SN(n),- ;Shadowing partner
 TN = Ext_Apl_TN(n),- ;Host application number
 EM = Ext_Apl_EM,- ;Max. length of mirroring queue
 EP = Ext_Apl_EP) ;Event queue overflow policy
 #create APL'Apl_Nb':B = Apl_Permanent
#loop_end

;*********************************************
;Proxy Applications
;*********************************************
#loop_with n = 1 .. length(Proxy_Apl_Numbers)
 Apl_Nb = Proxy_Apl_Numbers(n)
 Apl_Permanent = list(-
 TT = "PROXY",-
 NA = Proxy_Apl_NA(n),- ;Application name
 HS = Proxy_Apl_HS(n) ) ;HSB application pair
 #create APL'Apl_Nb':B = Apl_Permanent
#loop_end

;*********************************************
;Station Types
;*********************************************
Result = do(read_text("sys_/sys_bascon.scl"), "STATION_TYPES")

;*********************************************
;Node, Link for PC-NET & Stations
;*********************************************
Result = do(read_text("Sys_Tool/Create_C.scl"), "BASE_SYSTEM")

;*********************************************
;OPC DA nodes
;*********************************************
Nod = list()
#loop_with n = 1 .. length(OPC_DA_NOD_Numbers)
 Nod_Nb = OPC_DA_NOD_Numbers(n)
 Nod_RN = OPC_DA_NOD_RN(n)
 Nod_OP = OPC_DA_NOD_OP(n)
 Nod = list(NT = "OPC_DA", RN = Nod_RN, OP = Nod_OP, DI =
          OPC_Nod_DI)
 #create NOD'Nod_Nb':B = Nod
#loop_end

;*********************************************
```

```
;OPC AE nodes
;*********************************************
Nod = list()
#loop_with n = 1 .. length(OPC_AE_NOD_Numbers)
 Nod_Nb = OPC_AE_NOD_Numbers(n)
 Nod_RN = OPC_AE_NOD_RN(n)
 Nod_OP = OPC_AE_NOD_OP(n)
 Nod = list(NT = "OPC_AE", RN = Nod_RN, OP = Nod_OP, DI =
          OPC_Nod_DI)
 #create NOD'Nod_Nb':B = Nod
#loop_end

;*********************************************
;OPC DA stations
;*********************************************
Sta = list()
#loop_with n = 1 .. length(OPC_DA_Station_Numbers)
 Sta_Nb = OPC_DA_Station_Numbers(n)
 Sta_ND = OPC_DA_Station_Nodes(n)
 Sta = list(ST = "OPC", ND = Sta_ND, TT = "EXTERNAL")
 #create STA'Sta_Nb':B = Sta
#loop_end

;*********************************************
;OPC AE stations
;*********************************************
Sta = list()
#loop_with n = 1 .. length(OPC_AE_Station_Numbers)
 Sta_Nb = OPC_AE_Station_Numbers(n)
 Sta_ND = OPC_AE_Station_Nodes(n)
 Sta = list(ST = "OAE", ND = Sta_ND, TT = "EXTERNAL")
 #create STA'Sta_Nb':B = Sta
#loop_end

;*********************************************
;Stations for Mirroring etc.
;*********************************************
Sta = list()
#loop_with n = 1 .. length(Stations)
 Sta_Nb = Stations(n)
 Sta = list(TT = "EXTERNAL",-
 ST = Sta_ST(n),-
 ND = Sta_Nodes(n),-
 TN = Stations(n),-
 MR = Sta_MR(n),-
 HS = list(APL = 0, UN = 0),-
 IS = vector(list(APL = 0, UN = 0)))
 #if Sta_MR(n) == "HOST" or Sta_MR(n) == "BOTH" #then #block
 #loop_with m = 1 .. length(Sta_I_Apl(n))
 Sta.IS(m) = list(APL = Sta_I_Apl(n)(m), UN = Sta_I_UN(n)(m))
 #loop_end
 #block_end
 #if Sta_MR(n) == "IMAGE" or Sta_MR(n) == "BOTH" #then #block
 Sta.TN = 0
 Sta.HS = list(APL = Sta_H_Apl(n), UN = Sta_H_UN(n))
 #block_end
 #create STA'Sta_Nb':B = Sta
#loop_end

;*********************************************
;Delete global variables
;*********************************************
Global_Variables = variable_names
#loop_with n = 1 .. length(Global_Variables)
 Global_Variable = Global_Variables(n)
 #delete 'Global_Variable':V
#loop_end
```

```
;*********************************************
;Project specific configuration after this line
;*********************************************
```

If there is a configuration error in the SYS_BASCON.COM file, the system might not start. Check the messages from the Notification Window or SYS_LOG.CSV.

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

Scan this QR code to visit our website

https://hitachiabb-powergrids.com/microscadax

1MRK 511 481-UEN