

System 800xA Engineering

Process Engineering Tool Integration

System Versions 6.0

Power and productivity
for a better world™



System 800xA Engineering

Process Engineering Tool Integration

System Versions 6.0

NOTICE

This document contains information about one or more ABB products and may include a description of or a reference to one or more standards that may be generally relevant to the ABB products. The presence of any such description of a standard or reference to a standard is not a representation that all of the ABB products referenced in this document support all of the features of the described or referenced standard. In order to determine the specific features supported by a particular ABB product, the reader should consult the product specifications for the particular ABB product.

ABB may have one or more patents or pending patent applications protecting the intellectual property in the ABB products described in this document.

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

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

This document and parts thereof must not be reproduced or copied without written permission from ABB, 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. This product meets the requirements specified in EMC Directive 2004/108/EC and in Low Voltage Directive 2006/95/EC.

TRADEMARKS

All rights to copyrights, registered trademarks, and trademarks reside with their respective owners.

Copyright © 2004-2014 by ABB.
All rights reserved.

Release: August 2014
Document number: 3BUA000184-600

Table of Contents

About this User Manual

General	9
User Manual Conventions	9
Feature Pack	10
Warning, Caution, Information, and Tip Icons	10
Terminology.....	11
Released User Manuals and Release Notes	11

Section 1 - Getting Started

Introduction	13
Data Sources	14
File	14
Database	14
Logging.....	15
Transformations.....	16
Licensing and Security	19

Section 2 - Workflows

Introduction	21
Control Builder Workflow	22
Step 1: Create and Assign Applications	22
Step 2: Synchronize Control Structure.....	23
Step 3: Synchronize Functional Structure.....	23
Step 4: Synchronize Control Structure Again	23
Step 5: Optionally Synchronize Other Structures	23

Pure CB Workflow 23

 Workflow Selection 24

 Variable Creation 24

 Scope and Naming Convention of Variables 26

 Mapping Information 28

 Switching Between Control Builder Workflow and Pure CB Workflow 29

 Subvariable Creation..... 29

Function Diagram Workflow 32

 Step 1: Create and Assign Function Diagrams 32

 Step 2: Synchronize Control Structure..... 33

 Step 3: Synchronize Functional Structure..... 33

 Step 4: Synchronize Control Structure Again..... 33

Selection of Workflow 34

Foundation Fieldbus (FF) Workflow..... 38

Section 3 - Mapping

Introduction 41

Operation 42

 Mapping Control Properties..... 47

 Expert Mapping and Blank Object Type GUID 48

Function Designer Template 50

 Allocation of Function Diagrams to Applications and Applications to Controllers54

Section 4 - Data Transfer

Introduction 57

Operation 57

 Data Comparison 58

 Setting Up Start Object For Compare 59

 Partial Compare on 800xA Tree View 62

 Data Transfer..... 64

Data Transfer in ExpressSync Mode..... 64

Data Transfer in ExpertSync Mode..... 67

Object Reconciliation..... 69

 CreatedByPETI Aspect..... 70

Object Categorization.....	71
Renamed Object	72

Section 5 - CAEX Tree Editor

Introduction	81
Running the CAEX Tree Editor	81
Manipulating Structure Views	81
Inserting Nodes	82
Deleting Nodes.....	83
Modifying Node Attributes	83
Moving Nodes	83
Saving Changes	84

Section 6 - Import/Export Utility

Introduction	85
Import/Export	90
Export Data	90
Import Data	91
Object Type Definitions.....	92
SmartPlant Foundation Adapter	96
Data Flow	96

Section 7 - Command Line Interface

Introduction	101
--------------------	-----

Appendix A - Default Mapping File Configuration

Introduction	103
--------------------	-----

Appendix B - Configuring ABB Hardware in INtools/SPI

Controller Configuration	141
Configuring ABB Cards/Strips.....	142
Configuring Profibus Devices	144
Configuring HART Devices	146
Creation of FF Template.....	147

Index

About this User Manual

General



Any security measures described in this User Manual, for example, for user access, password security, network security, firewalls, virus protection, etc., represent possible steps that a user of an 800xA System may want to consider based on a risk assessment for a particular application and installation. This risk assessment, as well as the proper implementation, configuration, installation, operation, administration, and maintenance of all relevant security related equipment, software, and procedures, are the responsibility of the user of the 800xA System.

The Process Engineering Tool Integration product (the product) is a standalone application tool that provides seamless data exchange between Intergraph SmartPlant Instrumentation (INtools/SPI) and ABBs 800xA System. INtools/SPI manages and stores the history of the control system and provides a single source of plant information that can be easily accessed and updated. It ensures consistency across different instrument tasks and deliverables.

The product provides as much data exchange as possible between the basic, process, and instrumentation engineering phase and the control engineering phase (data exchange in one direction). The product keeps the process and control engineering data consistent over the entire life cycle of a plant (bidirectional data exchange, single point of data entry).

This instruction provides procedures for configuring and operating the software.

User Manual Conventions

Microsoft Windows conventions are normally used for the standard presentation of material when entering text, key sequences, prompts, messages, menu items, screen elements, etc.

Feature Pack

The Feature Pack content (including text, tables, and figures) included in this User Manual is distinguished from the existing content using the following two separators:

Feature Pack Functionality_____

<Feature Pack Content>

Feature Pack functionality included in an existing table is indicated using a table footnote (*) :

*Feature Pack Functionality

Unless noted, all other information in this User Manual applies to 800xA Systems with or without a Feature Pack installed.

Warning, Caution, Information, and Tip Icons

This User Manual includes Warning, Caution, and Information where appropriate to point out safety related or other important information. It also includes Tip to point out useful hints to the reader. The corresponding symbols should be interpreted as follows:



Electrical warning icon indicates the presence of a hazard that could result in *electrical shock*.



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



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



Information icon alerts the reader to pertinent facts and conditions.



Tip icon indicates advice on, for example, how to design your 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, fully comply with all Warning and Caution notices.

Terminology

A complete and comprehensive list of terms is included in *System 800xA System Guide Functional Description (3BSE038018*)*. The listing includes terms and definitions that apply to the 800xA System where the usage is different from commonly accepted industry standard definitions and definitions given in standard dictionaries such as Webster's Dictionary of Computer Terms.

Released User Manuals and Release Notes

A complete list of all User Manuals and Release Notes applicable to System 800xA is provided in *System 800xA Released User Manuals and Release Notes (3BUA000263*)*.

System 800xA Released User Manuals and Release Notes (3BUA000263)* is updated each time a document is updated or a new document is released. It is in pdf format and is provided in the following ways:

- Included on the documentation media provided with the system and published to ABB SolutionsBank when released as part of a major or minor release, Service Pack, Feature Pack, or System Revision.
- Published to ABB SolutionsBank when a User Manual or Release Note is updated in between any of the release cycles listed in the first bullet.



A product bulletin is published each time *System 800xA Released User Manuals and Release Notes (3BUA000263*)* is updated and published to ABB SolutionsBank.

Section 1 Getting Started

Introduction

Begin by launching the product. Click the PETI icon in the **ABB Start Menu**. For more information about the ABB Start Menu refer *System 800xA, Tools (2PAA101888*)* manual.



If there is no icon, the product is installed to the directory **<Program Files(x86)>\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration** and the executable file is in the subfolder **<PETI>\bin**.



The license for the product is incorporated into the CLS (central licensing system) of 800xA. If the product is not licensed on the workstation, an error will occur. Refer to [Licensing and Security](#) on page 19 for more information.



The product uses large amounts of memory when synchronizing large INtools/SPI databases. The available memory on the machine must be evaluated before using the product.

When the start window appears, two options are available to the user:

- Mapping (refer to [Section 3, Mapping](#) for more information).
- Transfer Data (refer to [Section 4, Data Transfer](#) for more information).

Select the appropriate 800xA Environment (Production or Engineering) from the drop-down list box for synchronization with the INtools/SPI database.



The product does not allow synchronization with the Load-Evaluate-Go (LEG) 800xA environment.

Once either of these two options are chosen, the user then selects a data source.

Data Sources

The product supports File and Database types of data sources for INtools/SPI.



If the user needs to initially specify or change the location of the web service used by the product to connect to the INtools/SPI database, select the **Set Server Name** link below the Database option and make the changes.

File

The data is in the form of a **CAEX** file that has been generated from an INtools/SPI database. Select the **CAEX** file that represents the INtools/SPI data source.

Database

The default server is the local host (the server on which the product is installed). To connect to a different server, perform the following:

1. Click on **Set Server** link.
2. Enter the name of the server on which the web-service is running.

The data source is a web service that is connected to an INtools/SPI database (online data source). By selecting this option, the product connects to the INtools/SPI database using the web service and then displays a list of the INtools/SPI PLANT, INtools/SPI AREAS, and their corresponding INtools/SPI UNITS that are available in the INtools/SPI database ([Figure 1](#)).

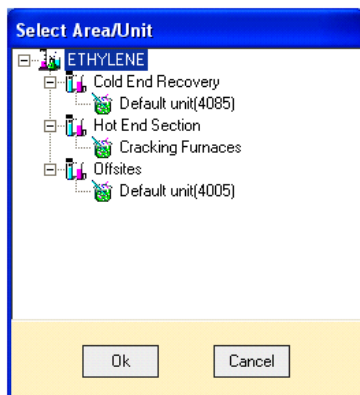


Figure 1. Database Source Selection

Select a particular PLANT, AREA, or underlying UNIT for the web service to retrieve the data for the selection from the INtools/SPI database.



The user can only work with data from one INtools/SPI PLANT, AREA or its underlying UNITS when running the product using the Database option. If the user selects a PLANT, a warning appears about potentially long processing times. Once the user has selected an INtools/SPI data source, the name of the data source and the associated mapping file are displayed at the bottom of the windows.

Once the user has started using the product, options are available to switch between the Mapping and Transfer Data options or start over by choosing a new INtools/SPI data source from the main menu as shown in [Figure 2](#).

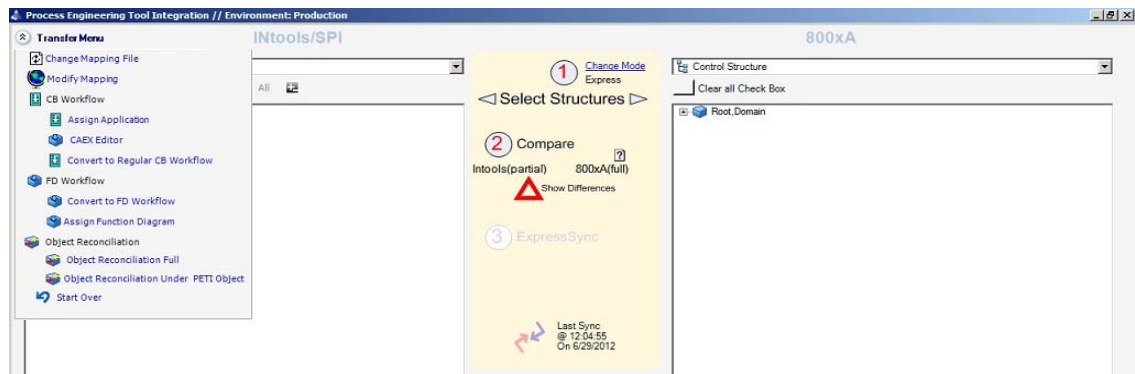


Figure 2. Main Menu Options

Logging

All user actions and errors generated during transfer of data from PETI to 800xA are written to a log file named **PETILog.xml** that is saved in the following path:

C:\ProgramData\ABB\Process Engineering Tool Integration\logs

When the log file reaches its maximum size, PETI fails to log an event in the file. This also results in **out of memory exception** error, see [Figure 3](#).

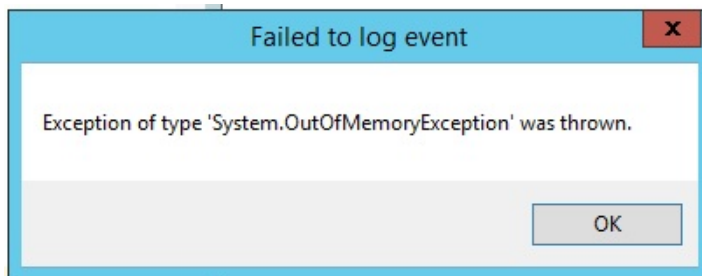


Figure 3. Out Of Memory Exception

To start logging the events in to the log file, perform the following workaround:

1. Rename the **PETILog.xml** file. For example, it can be renamed as **PETILog_Backup1.xml**.
2. Create a new *xml* file and name it **PETILog.xml**.

All the actions performed in PETI will be logged in the new **PETILog.xml** file.

Transformations

The CAEX transformation utility within the product is used for transforming CAEX files. It is intended to transform files that are exported by an engineering tool such as INtools/SPI so that the file fits the needs for the product importing it into System 800xA. [Figure 4](#) shows this representation.

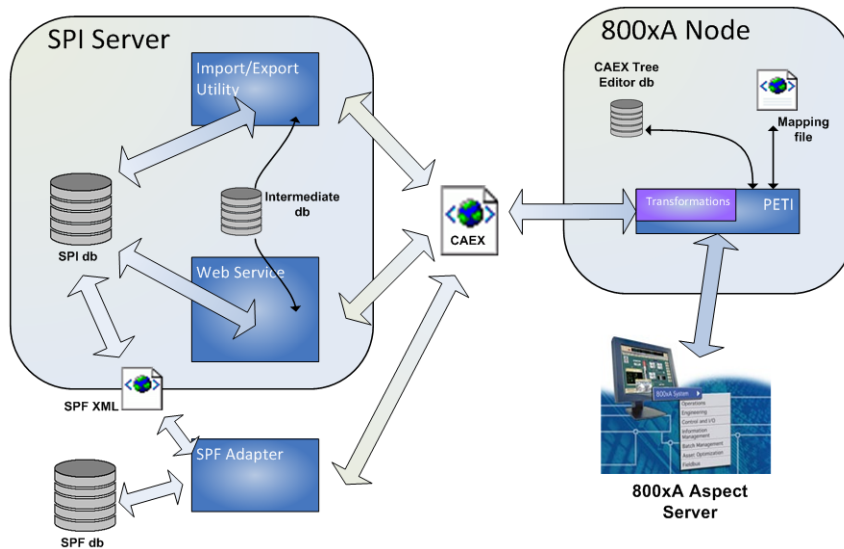


Figure 4. Transformation

The user can view transformations as an individual operation on the incoming CAEX file. Several transformations can be run in sequence. The transformations themselves have to be created only once and are then reused. A batch file of transformations can be created as described in [Operation](#) on page 19 and can be used as a preprocessor for the incoming INtools/SPI data (CAEX file) on a project.

Transformations are available that change attribute values, object names, and object types as well as transformations that delete objects or change the hierarchy of the objects. [Table 1](#) gives a list of transformations that are currently available with the

product.

Table 1. Transformation

#	Transformation	Description
1	Edit attribute value by script	Sets the specified attributes to the value returned by a script, if the script returns a value.
2	Edit type by script	Changes the specified type to the return value of a script.
3	Double an object	Creates duplicates of specific objects. The duplicates will have the specified strings appended to their names and the specified type
4	Structure objects	Creates a new level of the specified object between the first and second levels of the specified structure, but only when the second level object has an attribute that matches the specified attribute.
5	Rearrange I/O cards to be remote	Makes the specified S800/S900 objects Remote I/O.
6	Attribute mapping	Copies the attribute values from the attributes in the From columns to the attributes in the To columns.
7	Delete objects by script	Deletes all objects for which the script returns true. If the script returns false for a particular object, that object is not deleted.
8	Remove unused types in library	Cleans up types that are no longer used.
9	Delete objects by attribute value	Deletes all objects that have the specified attribute with a value that is different from the specified attribute value.
10	Synchronize object names	Appends _Sig to all attribute names of all objects that end with the _Sig.
11	Copy attribute value	Searches all objects and copies the specified source attribute to the sink attribute.
12	Delete objects by type	Deletes all objects of the specified types.
13	Loop pattern matching	Groups objects into loops based on their types.



If a property value is modified using the transformer then the user will not be able to write back the value to INtools/SPI using the bidirectional transfer.

Operation

To start the transformation within the product, perform the following:

1. Launch the product.
2. From the Start window:
 - a. Check the **Apply CAEX Transformations** check box.
 - b. Select the Source (**File** or **Database**).
3. To create or edit an existing transformation (.cts) file, click on the **Create New Transformation File** button. When done, close the Transformer window to return to the product window. The transformation file will be displayed in the drop-down list box.
4. To use an existing transformation file without any change, choose from a list of previously accessed files by selecting it from the drop-down list box. Or, click the **Browse** button and navigate to the file.
5. Once the transformation file has been selected, click the **Continue** button. When complete, the main window of the product appears.

Licensing and Security

The product uses the CLS for licensing and user authentication in 800xA. The product supports two types of 800xA System users:

- Application
- System

The users have access to the functionality as described in [Table 2](#).

Table 2. User Security

	Application User	System User
Express Sync	×	×
Expert Mode	—	×
Override Dataflow Direction	—	×
Edit Mapping File	—	×



An Application/System User does not have an Operator Role by default. It is recommended to add the Operator Role to Application/System User.

The product supports two levels of licenses:

- Base
- Advanced

The licenses are incorporated into the 800xA Engineering Licensing group in the CLS using feature names called PETI_BASE and PETI_CREATE.

The available functionality based on the licenses is described in [Table 3](#).

Table 3. Licensing

	Base	Advanced
View Differences	×	×
Update Objects/Properties	×	×
Create New Objects	—	×

If the product is not licensed on the workstation, the user will see an error message when the application is launched. When successfully licensed, the product licenses can be viewed using the CLS tool on the 800xA workstation.

Section 2 Workflows

Introduction

There are four workflows that can be used (Figure 5) when configuring 800xA:

- [Control Builder Workflow](#) on page 22
- [Pure CB Workflow](#) on page 23
- [Foundation Fieldbus \(FF\) Workflow](#) on page 38

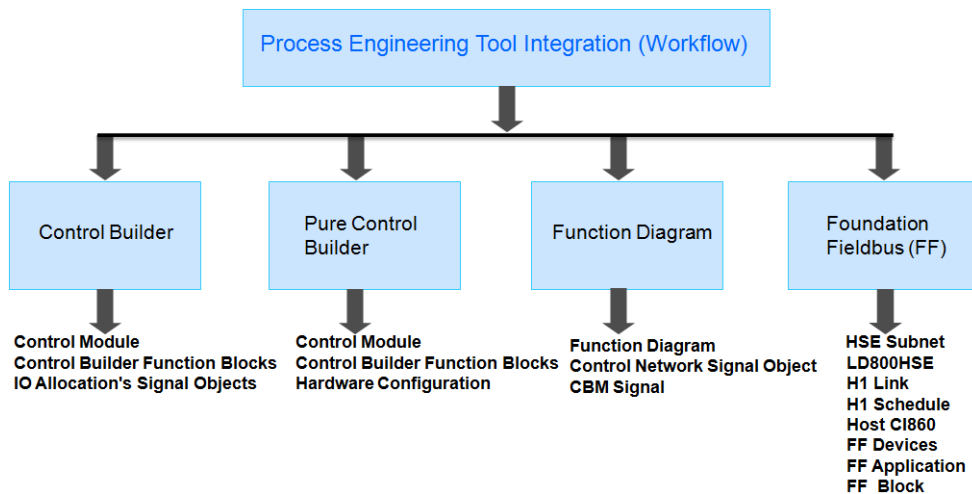


Figure 5. Workflow



PCDeviceLib Workflow has been renamed to Control Builder Workflow.



Foundation Fieldbus (FF) workflow in [Figure 5](#) applies to the System 800xA 5.1 Feature Pack release only.

Control Builder Workflow

There are four steps for the Control Builder Workflow. The workflow steps have to be executed manually by a user.

Step 1: Create and Assign Applications

Create applications and assign tags and loops that are listed as INtools/SPI objects to the applications. The product creates the tags in the INtools/SPI database in the 800xA system as Control Module objects under Applications. However, in order for the control module objects to be successfully created, every tag in the INtools/SPI database need to be assigned to an Application. Therefore, before the user can execute a Data Transfer that involves the creation of new Control Module objects, the user needs to assign the new objects to an Application. In order to assign the objects, select the **Assign Application** link from the main menu.

The user can create new Applications by clicking on the **New** button. Objects can be assigned/unassigned to any existing Applications by performing the following:

1. Select an Application by clicking on the name in the list.
2. Click one or more objects from the Unassigned List (select multiple items using standard Windows methods if desired).
3. Click >> to move the objects to the Assigned List.
4. Repeat the [Step 1](#) through [Step 3](#) to unassign objects if desired and click <<.
5. Click **OK** to save the configuration or **Cancel** to abort.



Click on the **Summary** link in the dialog to view all tags with their assignments. The summary view also displays if the tags have already been created in 800xA in a previous session.

Step 2: Synchronize Control Structure

The control modules get created in Control Structure underneath in the following location: **Control Network > Control Project > Application**. Control Hardware also gets created in the Control Structure.

Step 3: Synchronize Functional Structure

Control Project and Control modules created in earlier step are inserted in the Functional Structure and signal objects are created underneath the control modules as applicable.

Step 4: Synchronize Control Structure Again

Signal objects created in Functional structure in the earlier step are inserted in the hardware (channel IO assignment) back in the control structure.

Step 5: Optionally Synchronize Other Structures

Objects are created in other structures such as Location Structure, Documentation Structure, or Asset Structure, as specified in the CAEX data file.

Pure CB Workflow



Sample CAEX, map, and application assignment files for the Pure CB workflow can be found in the following location: \\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\Sample CAEX.

Sample files that include other structures for Pure CB workflow can be found in the following location: C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\Sample Other Structures.

The Pure CB workflow is similar to the Control Builder workflow in that control modules are created in the Control Structure. Instead of connecting an I/O channel to a control module through a signal object, a variable is used.

Workflow Selection

To select the Pure CB workflow, click on **Convert to Pure CB Workflow** as shown in [Figure 6](#).

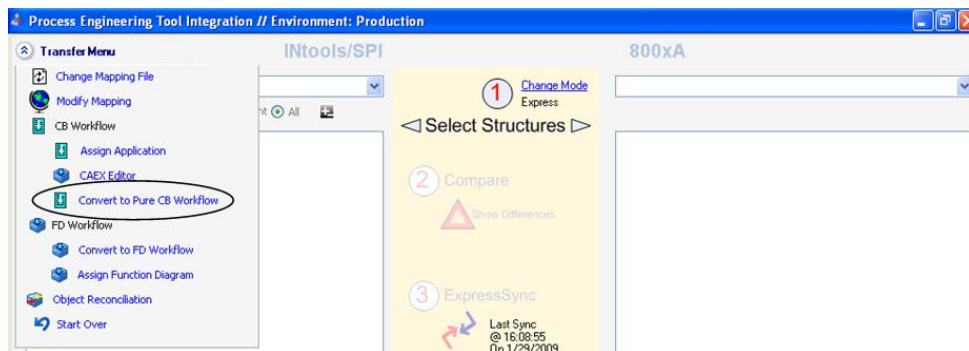


Figure 6. Pure CB Workflow

Variable Creation

There is no change in the way CAEX file is created from an INtools/SPI database for the Pure CB workflow. The CAEX file contains a Functional Structure section in which a signal object is added for each control module.

For each signal object that should be inserted into an I/O device as specified in the CAEX file, a variable will be created either on the Application level or on the Single Control Module level depending on the parent control module of the signal object. For example, if the control module in the Control Structure is a child of an application object, then a global variable is created in that application object as shown in [Figure 7](#). However, if the parent of the control module is a Single Control Module object, then the user can specify whether the variable to be created is global or local. A local variable is one that is created on the parent Single Control Module level. Refer to [Scope and Naming Convention of Variables](#) on page 26 for more

information.

	Name	Data Type	Attributes	Initial Value	I/O Address	Action
1	vg_x1_FT_1501_IO	RealIO	retain		HOT_CTRL2	
2	vg_x1_FT_1234_IO	RealIO	retain		HOT_CTRL2	
3	vg_x1_FT_1602_IO	RealIO	retain		HOT_CTRL2	
4	vg_x1_PT_1569	RealIO	retain		HOT_CTRL1	
5	vg_x1_FT_15021_IO	RealIO	retain			
6	vg_x1_PT_1234	RealIO	retain		HOT_CTRL1	
7	vg_x1_FT_15022_IO	RealIO	retain		HOT_CTRL1	
8						
9						

Global Variables Variables

Row 1, Col 1 Waibiu Cheng

Figure 7. Global Variables on Application Level

The variable created is connected to a port of the control module (I/O) as shown in [Figure 8](#). It is also be connected to a channel of an I/O device as shown in [Figure 9](#). The connection to a channel is made based on the information contained in the CAEX file for the signal object. The connection to a port of a control module requires additional mapping information. Refer to [Mapping Information](#) on page 28 for more information.

	Name	Data Type	Initial Value	Parameter	Direction
1	Name	string[20]		'1-FT-1234'	in
2	Description	string[40]		'RLS FLOW LOOP 1234'	in
3	InteractionPar	TransmitterPar	Default		in
4	IO	RealIO		appPETI.vg_x1_FT_1234_IO	in
5	HSICmd	HSICmdTransmit	Default		in
6	Out	ControlConnectic	Default		in
7	PCCH	PCC	Default		in
8	PCCHConf	dword	cPI.Priority		in
9	PCCHH	PCC	Default		in
10	PCCHHConf	dword	cPI.Priority		in
11	PCCHHH	PCC	Default		in

Parameters

Row 1, Col 4 Waibiu Cheng

Figure 8. Connection of a Variable to a Port of a Control Module

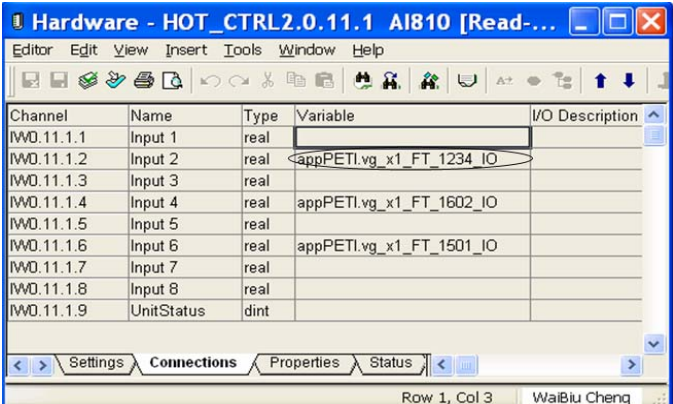


Figure 9. Connection of a Variable to a Channel of an I/O Device

Scope and Naming Convention of Variables

Variables for the Pure CB workflow may be created as global or local variables. Global variables are created on the Control Application level and local variables are created on the Single Control Module level.

For control modules that are placed directly under the Control Modules object in the Control Structure, the variables that they reference are always global variables.

For control modules that are placed under a Single Control Module object, the variables that they reference may be either global variables or local variables. The selection is made by the user via the Variables tab of the software as shown in

Figure 10.

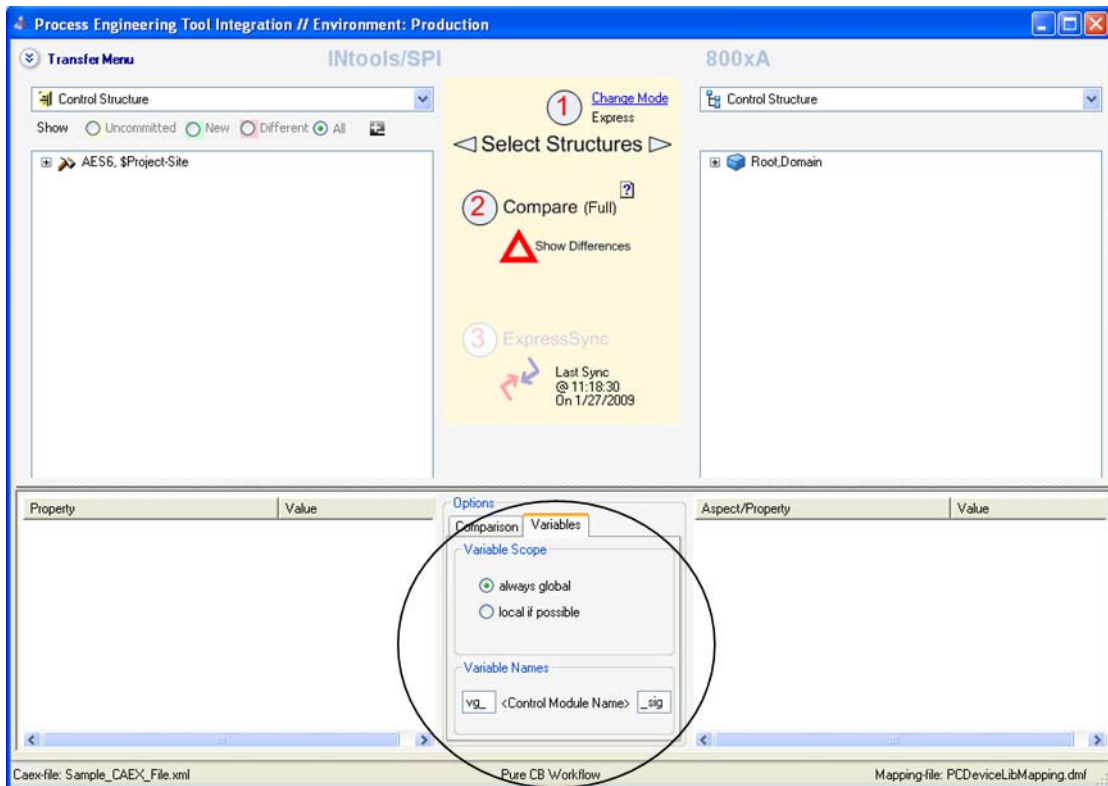


Figure 10. Specifying Scope and Naming Conventions of Variables

The user may also format the variable names by adding a prefix and/or a suffix to the name of the control module on which the variable names are based. For example, if a variable is to be connected to port IO of control module 1-FT-1234, the variable name is $\langle \text{prefix} \rangle 1_FT_1234_IO \langle \text{suffix} \rangle$, where $\langle \text{prefix} \rangle$ and $\langle \text{suffix} \rangle$ are specified through the same menu on the Variables tab as shown in Figure 10. With the specifications shown in Figure 10, the variable is named

vg_1_FT_1234_IO_sig and is created on the Control Application level as a global variable.



A prefix and/or a suffix is applicable to the Function Block and the Control Module based variables. They are not applicable for FF variables.

Mapping Information

To specify which port of the control module the variable is connected to, additional mapping information is needed. This is done with a specific CAEX property called IO_Signal as shown in Figure 11.

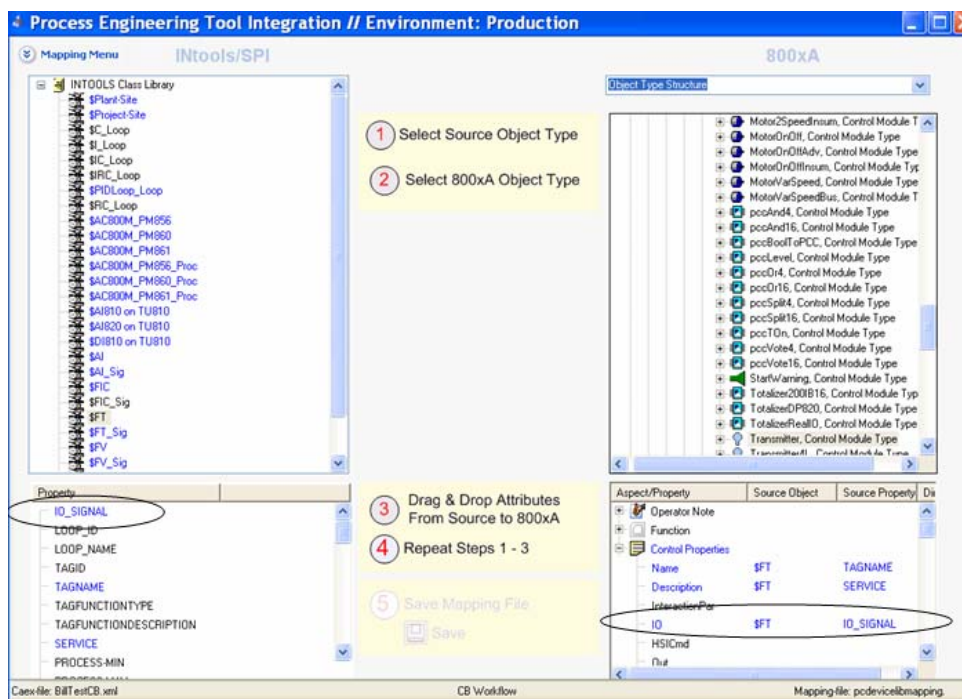


Figure 11. Port Identification via Mapping

The IO_Signal should be mapped to a port (a Control Properties aspect property) of a control module object type. The product checks which port the property IO_Signal is mapped to and connect the variable accordingly.



There is no value associated with this CAEX property. It exists simply to identify the port of a control module to which a variable is to be connected.

Switching Between Control Builder Workflow and Pure CB Workflow

Switching from Control Builder Workflow to Pure CB Workflow requires transformation of the CAEX file. The switching is irreversible. Switching from Control Builder Workflow to Pure CB Workflow only impacts the way the objects are created in 800xA. The CAEX file itself is not affected. Therefore, it is possible to switch from Pure CB Workflow back to Control Builder Workflow.

To switch back, click on the option **Convert to Regular CB Workflow**.

There is no difference in the Compare step for the Control Builder Workflow and Pure CB Workflow. Therefore, the user can execute the Compare step for the Control Structure in Control Builder Workflow, switch to Pure CB Workflow, execute the Transfer step, and the variables will be created in the Control Structure.

Subvariable Creation



Sample files for the Pure CB workflow with subvariables can be found in the following location: C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\Sample SGO.

The Pure CB workflow supports signal group object (SGO). For example, a variable ((application name).vg_x1_FV_1501_IO) is created on the application level and inserted in the control module I/O port, similar to a simple signal object for the following structure (control module - SGO - signal objects):

```
1-FV-1501
    1-FV-1501_SGO
        1-FV-1501_SGO_FB
        1-FV-1501_SGO_SP
```

The difference here is that there are now two signal objects that are associated with the control module. In this case, two subvariables, (application name).vg_x1_FV_1501_IO.FB and (application name).vg_x1_FV_1501_IO.SP, are created by the product and inserted in two separate I/O channels as shown in [Figure 12](#) and [Figure 13](#), based on the information contained in the CAEX file for

the two signal objects.



Figure 12. Insertion of Subvariable FB in an I/O Card

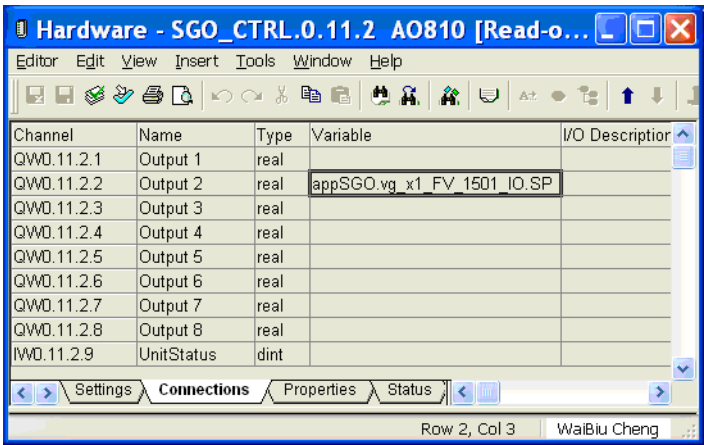


Figure 13. Insertion of Subvariable SP in an I/O Card

As a composite object, the parent and children of an SGO must be mapped as a group as shown in [Figure 14](#) and [Figure 15](#).

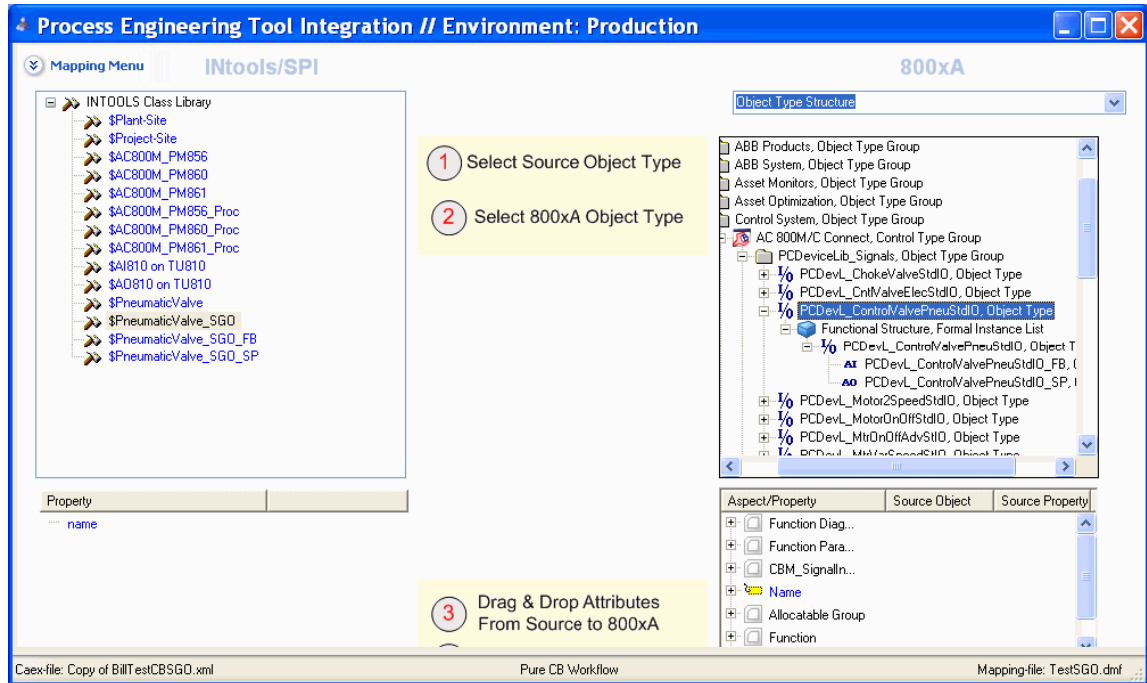


Figure 14. Mapping of Signal Group Object - Parent

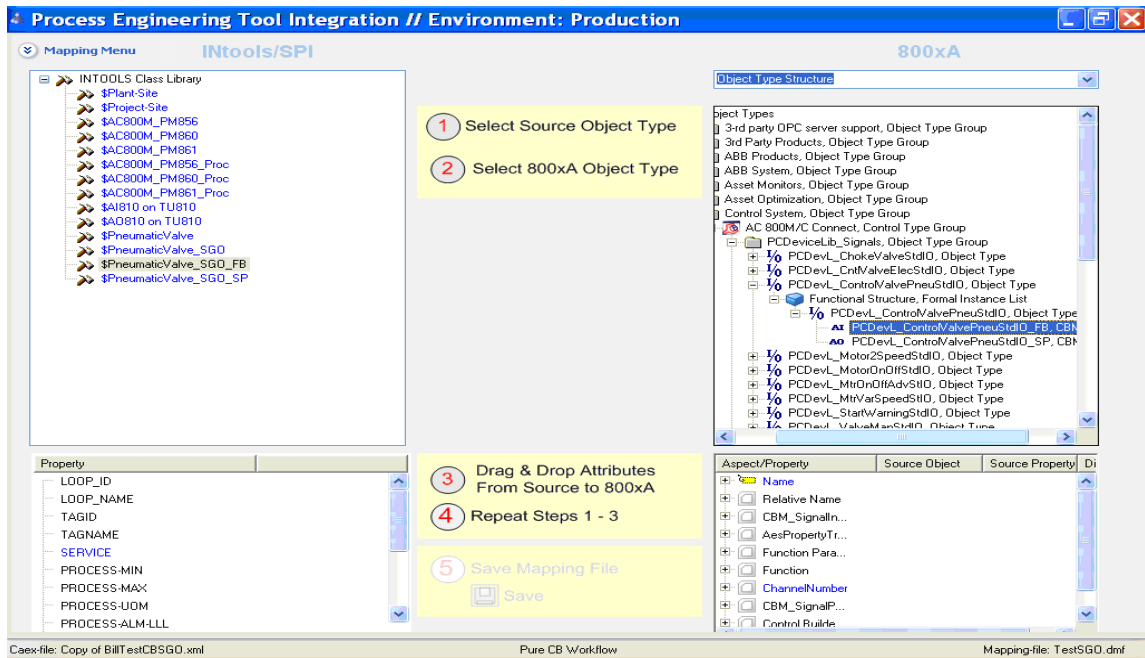


Figure 15. Mapping of Signal Group Object - Child

Function Diagram Workflow

There are four steps for the Function Diagram Workflow. The workflow steps have to be executed manually by a user.



Step 1 is optional. Refer to [Selection of Workflow](#) on page 34 for more information.



To work with Function Diagram Workflow, select **Convert to FD workflow** in the **Transfer Menu** before compare and transfer operations.

Step 1: Create and Assign Function Diagrams

The first step is to create function diagrams of desired types. The loops/tags and signals that are listed as INtools/SPI objects are then assigned to these newly created function diagrams. In order to assign the objects to the Function Diagrams, select the Assign Function Diagram link from the main menu.

The user can create new Function Diagrams by clicking on the **New** button. The user then provides a name for the Function Diagram and selects the type of diagram from the drop-down list box.

The Function Diagram Type drop-down list box is populated with the different types of Function Diagrams that are present in the **Objects Types > Functional Planning > Diagram Types > Based on Diagram Templates** node in the Object Type Structure in 800xA.

Objects can be assigned/unassigned to any existing Function Diagram doing the following:

1. Select a Function Diagram by clicking on the name in the list.
2. Click one or more objects (Control Modules and Signal objects) from the Unassigned List (select multiple items using standard Windows methods if desired).
3. Click >> to move the objects to the Assigned List.
4. Repeat [Step 1](#) through [Step 3](#) to unassign objects if desired and click <<.
5. Click **OK** to save the configuration or **Cancel** to abort.

Step 2: Synchronize Control Structure

Control Network and Control Project are created in the Control Structure. Control Hardware is created as per the information in INtools/SPI.

Step 3: Synchronize Functional Structure

Control modules and signals get created underneath the Function diagrams as specified in the [Step 1](#) above.

Step 4: Synchronize Control Structure Again

Signal objects created in Functional structure in the earlier step are inserted in the hardware (channel IO assignment) back in the Control Structure.

Selection of Workflow

The configuration data that is extracted from the INtools/SPI database (either via Web Service or via Import Export Utility in the form of a CAEX file) is always for the Control Builder workflow. To convert it into Functional Diagram workflow, perform one of the following:

1. Select **Convert to FD Workflow** (Figure 16).
2. Select **Assign Function Diagram** (Figure 16).

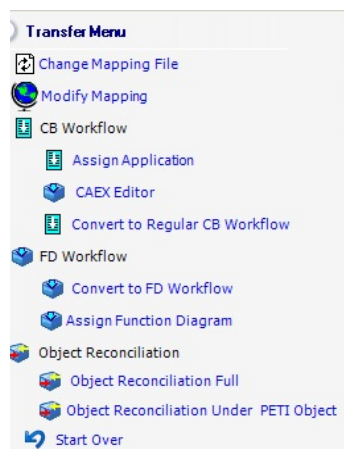


Figure 16. Options for Converting from Control Builder Workflow to Function Diagram Workflow

With the Convert to Function Designer Workflow, the structures for the Control Builder workflow as shown in Figure 17 (without plant, area, and unit objects – refer to Section 6, Import/Export Utility for more information) and Figure 18 (with plant, area, and unit objects) are converted to Function Designer workflow as shown in Figure 19 and Figure 20.



No Function Diagram is inserted in the Functional Structure.

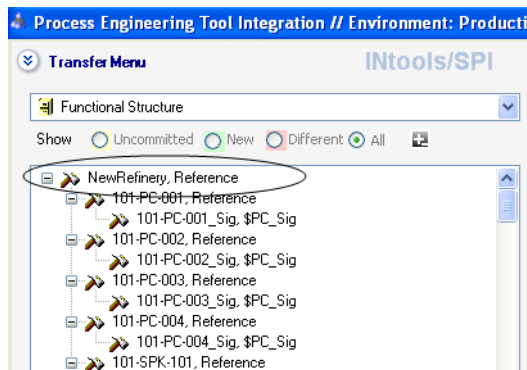


Figure 17. Functional Structure in Control Builder Workflow without Plant, Area, and Unit Objects

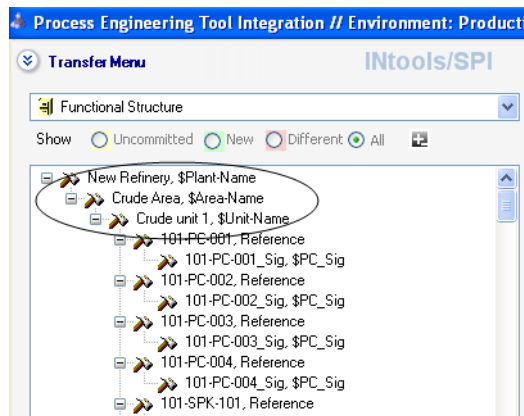


Figure 18. Functional Structure in Control Builder Workflow with Plant, Area, and Unit Objects

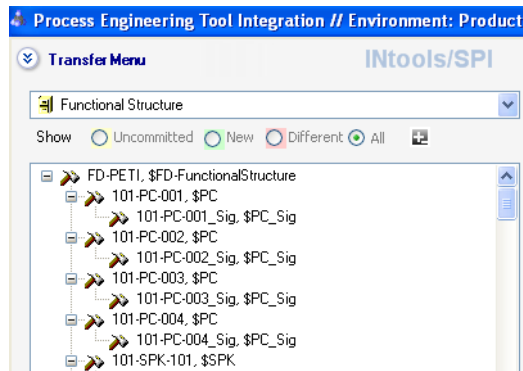


Figure 19. Conversion of Structures in Fig.16 from Control Builder to Function Diagram Workflow without Function Diagram Insertion

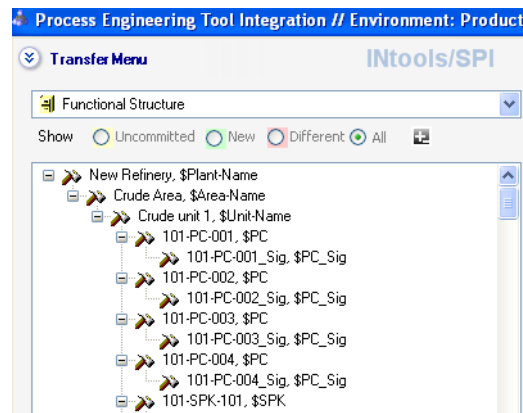


Figure 20. Conversion of Structures in Fig. 17 from Control Builder to Function Diagram Workflow without Function Diagram Insertion

The menu option Assign Function Diagram inserts a Function Diagram in the Functional Structure as shown [Figure 21](#) in and [Figure 22](#). The function diagram is specified in the Assign Function Diagram dialog as described in [Function Diagram Workflow](#) on page 32. If no specification is made, then a Function Diagram named

FDGeneric is inserted.

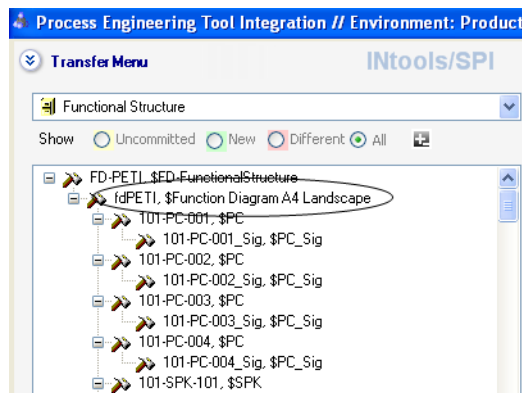


Figure 21. Conversion of Structures in Fig. 16 from Control Builder to Function Diagram Workflow with Function Diagram Insertion

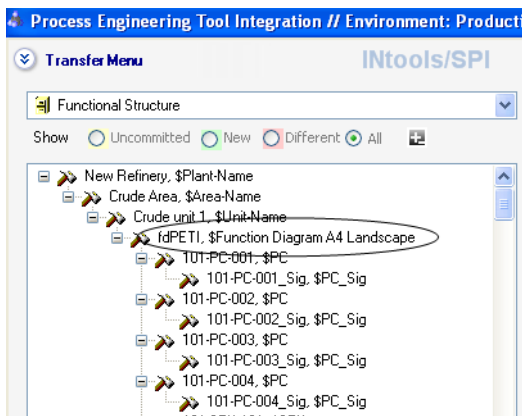


Figure 22. Conversion of Structures in Fig. 17 from Control Builder to Function Diagram Workflow with Function Diagram Insertion

Once the configuration data has been converted to the Functional Diagram workflow, the group of options under Control Builder Workflow in the Main Menu will not be available to the user (the user cannot convert the data from Functional Diagram workflow to Control Builder workflow). To revert back to the Control

Builder workflow, the user will have to reconnect to the Web Service or export a new CAEX file from the product Import Export Utility.



The workflow currently selected is indicated in the information bar at the bottom of the display.

Foundation Fieldbus (FF) Workflow

Foundation Fieldbus (FF) workflow supports instantiation of FF objects from the Object Type Structure and templates in the Control Structure. It also supports signal assignment and updating FF aspect property values.

Following steps describes the FF workflow. The workflow steps have to be executed manually by a user:

1. Launch PETI and select the CAEX file containing FF configuration data.
2. Select Mapping file containing CAEX Object to 800xA FF Objects mappings.



Sample CAEX and mapping files are available at: \\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\Sample FF.

In the Mapping menu, following CAEX FF Objects are mapped to 800xA FF Objects in Object Type Structure:

- HSE Subnet
- ABB LD800HSE
- H1 Link
- H1 Schedule
- HSE Host CI860
- FF Devices (e.g. Yokogowa, Yamatake)

Following CAEX FF objects are mapped to 800xA FF Objects in Control Structure:

- Application
- Block

Figure 23 shows an example of CAEX object type FF_Application mapped to 1xA1 FF Application in the Control Structure.

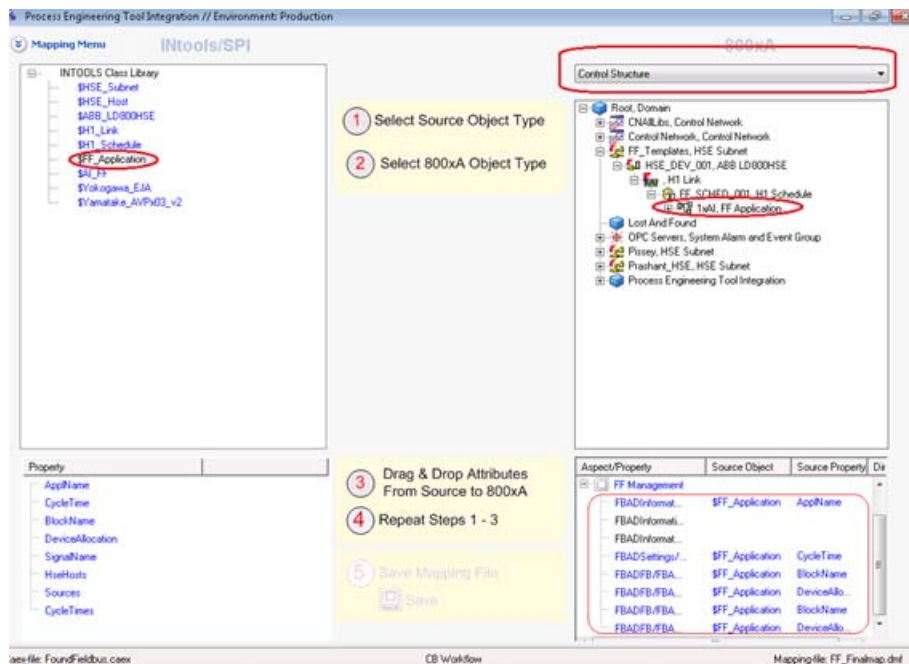


Figure 23. Example of FF_Application Mapping

For more information on mapping, refer to [Section 3, Mapping](#).

3. Go to transfer menu and transfer the data.

For more information on data transfer, refer to [Section 4, Data Transfer](#).



While transferring data an error message **Object created but some properties could not be written** is displayed.

This is a valid error message, and occurs because the FF Application object containing pointers to the FF function blocks which are linked to existing FF devices; hence FF devices must be instantiated before FF objects.

To overcome this error, perform compare and transfer operation in PETI again with **Include property comparison** option enabled. This updates the remaining properties in 800xA system.

Alternate Workflow

Block the transfer of FF Application and FF Blocks in the first run (compare and transfer operation), and then instantiate the two types of FF objects during the second run.

Figure 24 displays the FF hierarchy created under PETI folder or object:

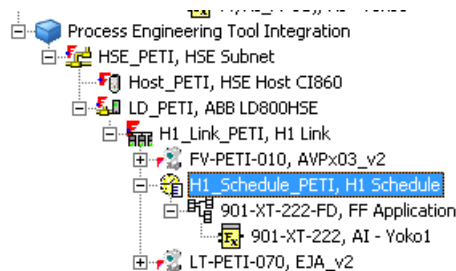


Figure 24. FF Hierarchy



In the input database, if the user specifies names of Control Builder variables corresponding to FF variables, then they are created at application level. However the user has to manually connect the global variables to individual CI860 channels.



PETI uses CDM generated CAEX files for FF, because the Import Export utility does not support importing of FF configuration from SPI data base.

Section 3 Mapping

Introduction

The mapping function of the product is used to map various INtools/SPI object types and properties to corresponding object types in the 800xA System. The associated direction of data-transfer (from INtools/SPI to 800xA or vice versa) is stored in a mapping file. The mapping information saved in a mapping file can be loaded later during data transfer. The mapping file itself is saved on the workstation running the product as a DMF file.

The mapping function can be categorized into two types:

- **Object Mapping** - This specifies INtools/SPI object type A is mapped to ABB object type B (when INtools/SPI object type A needs to be synchronized, the product will create ABB Object Type B). For example:

INtools/SPI Object Type	ABB Object Type
\$FT	PCDeviceLib\Transmitter

- **Attribute Mapping** - This specifies INtools/SPI Object Type A/Attribute A1 is mapped to ABB Object Type B/Attribute B1 (member of Aspect X1). The values of the two attributes are synchronized by the product. For example:

INtools/SPI Object Type	INtools Attribute	Direction	ABB Object Type	ABB Aspect	ABB Property
\$FT	SERVICE	►	Transmitter	Name	Description



The user specifies the mapping in the product by mapping the attributes between the INtools/SPI object types and the ABB object type. The object mapping is then handled automatically by the product and is not explicitly specified by the user. Therefore, the specification of the first Attribute Mapping between an INtools/SPI object type and ABB object type adds the Object Mapping to the mapping file. Additionally, the deletion of the last Attribute Mapping between an INtools/SPI object type and ABB object type automatically deletes the corresponding Object Mapping.

Operation

Once the product has been started, perform the following:

1. Select the **Mapping** option from the Start window.
2. Select the source of the INtools/SPI data to be mapped.



The source can be either a file in a **CAEX** format or the INtools/SPI database that is connected to the product via a web service.

If the user selects the **File** option, there is a prompt to select a **CAEX** file generated from the INtools/SPI database that is to be mapped.

If the user selects the **Database** option, then the product connects to the web service and consequent INtools/SPI databases on the server specified in the opening dialog (refer to [Data Sources](#) on page 14 for more information).

If the data source has never been mapped, then the user is prompted to select a mapping file that is used to map the data. The user can select the default mapping file provided with the installation as a starting point for the mapping. The application remembers the associated mapping file name from that point onwards. The selection of the mapping file has to be done only the first time a particular data source is used. The product will automatically load the associated mapping file for future sessions and display the Mapping Window. The default mapping file provided during installation is for objects contained in ABBs **PCDeviceLib** library.

In the Mapping Window, the INtools/SPI Class Hierarchy appears on the left side and the 800xA object hierarchy appears on the right side as tree views. Expanding the tree on the INtools/SPI side displays all the INtools/SPI object types that occur

in the mapped INtools/SPI data source as shown in [Figure 25](#).

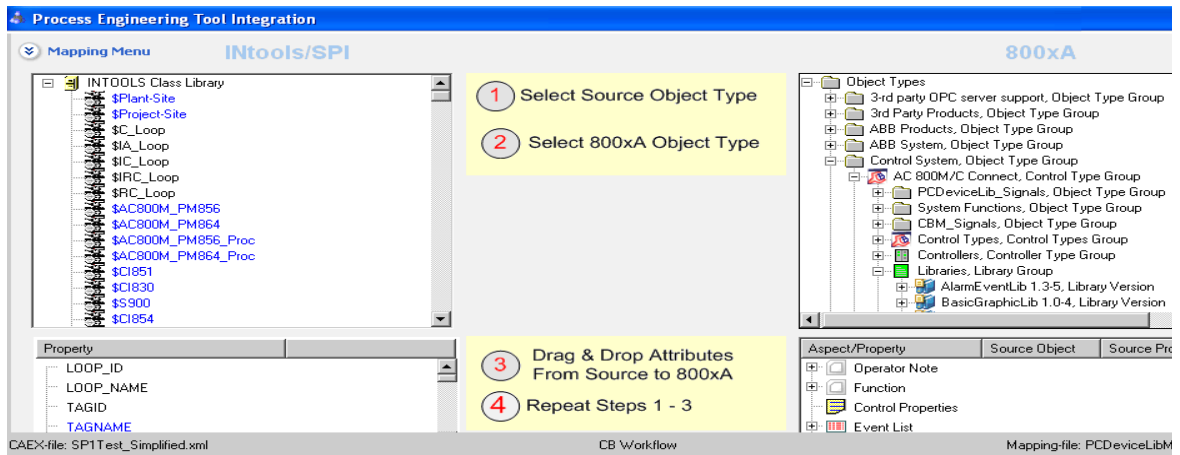


Figure 25. Mapping Object Hierarchy

In order to map a particular INtools/SPI object to a corresponding 800xA object, select the INtools/SPI object type from the left side. Once the object has been selected, the various properties of that object appear in the property box in the lower left frame. These are the various INtools/SPI properties that can be mapped to analogous properties of the various 800xA objects.

Once an INtools/SPI object type has been selected, select a corresponding 800xA object type from the tree on the 800xA side that needs to be mapped to that INtools/SPI object. Once the 800xA object type has been selected, the various properties grouped by the various aspects of that 800xA object will appear in the property box on the lower right side. The aspects can then be expanded to view the individual properties available under that aspect.



Already mapped INtools/SPI objects and properties are shown in blue. If the user selects an INtools/SPI object type that has been mapped, the product shows the corresponding 800xA object type in the tree that has been mapped to that INtools/SPI object type. However, if the user selects an unmapped INtools/SPI object type, the 800xA object tree will collapse and show only the top level.

In order to start mapping the properties between the INtools/SPI object and the corresponding 800xA object, select the property to be mapped on the INtools/SPI

side and drag-and-drop it to the corresponding property on the 800xA side. Once the property has been mapped, the INtools/SPI property name will appear next to the 800xA property in the 800xA property box. When an INtools/SPI property is dropped onto a 800xA property to be mapped, the color of the 800xA property as well as the parent aspect changes to blue as shown in [Figure 26](#).

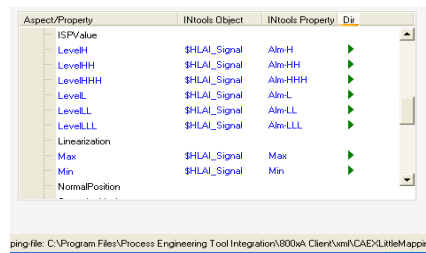


Figure 26. Mapping Properties

All mapped properties and aspects are shown in blue. All unmapped properties and aspects are shown in black. The user must specify the direction of the property data transfer between the INtools/SPI object and the corresponding 800xA object. The different available transfer directions are as follows:

Icon	Direction
	INtools/SPI > 800xA - this direction can be used to Create Objects in 800xA as well as updated property values in 800xA.
	800xA > INtools/SPI - this direction can only be used to update property values in INtools/SPI.
	Unknown - this direction is used when the user is uncertain of the direction at the time of mapping and would have to specify the transfer direction at the time of the execution of the transfer on an instance by instance basis.
	Blocked - this direction is used when the 800xA object is automatically created when a parent object is created but the mapping is required for the product to know the type of 800xA object that would be created automatically.

For example, PM865 HI/TP830 processor that is automatically created by Control Builder at location 0 when an AC 800M HI Controller is created.



This is different from the creation of a child object when a parent is created (PIDLoop with children Tx, PID, PCV). In this case, the PM865 HI processor is not a child of the AC 800M HI Controller as shown in [Figure 27](#).



Figure 27. Parent/Child Example

By default, when a property is initially mapped, the direction is set to Unknown. The user can modify the direction of data transfer by clicking on the direction icon to toggle between the three options. This process is repeated for each property that needs to be mapped between the INtools/SPI object and the corresponding 800xA object.

In order to delete a mapping between properties, the user can select the mapped property on the 800xA side and on the keyboard, press **Delete**.

Once all the required properties have been mapped, the user must save the changes made to the Mapping file by clicking on the **Save** icon and then specifying the name of the Mapping file. The user can either overwrite an existing Mapping file or save the file under a new name. The saved Mapping file can be used to synchronize the data between the INtools/SPI data source and the 800xA System.



The mapping file can be exported to a well formatted report document by clicking on **File > Mapping Print-Preview**. This will create a formatted HTML document that can be viewed on screen using Internet Explorer. It can also be sent to printer.

In certain situations, special mapping is required for the product to synchronize the data correctly and are described as follows:

- In case of a processor (PM856, PM860, etc.)([Figure 28](#)), the user cannot map to the Name property of the Name aspect since the processor is always automatically named 0 by Control Builder. Therefore, often the INtools/SPI

Name property is mapped to the Description property in the Name aspect. As described previously, at least one property/attribute needs to be mapped for the product to synchronize the objects.



Figure 28. Processor Mapping

- In case of a I/O board (AI810, DO810, etc.)(Figure 29), the user must map the INtools/SPI module_no property to the Name property of the Name aspect since the I/O board must be named by the module number in 800xA. Mapping the INtools/SPI name can cause problems if the INtools/SPI name is not the module number but a string such as AI820. As described previously, at least one property/attribute needs to be mapped for the product to synchronize the objects.

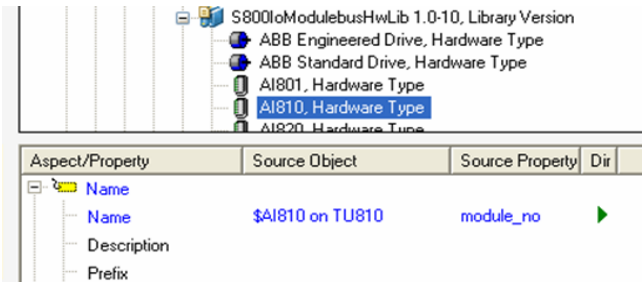


Figure 29. I/O Board Mapping

Mapping Control Properties

The product allows the user to synchronize INtools/SPI data with the properties in the Control Properties aspect for 800xA Control Modules, Controller, and other hardware objects. The access to the Control Properties aspect is made using the CBOpenIF interface since the ABBOBJECTAutomation interface does not have access to the properties in this aspect.



The product only provides access to the Parameters of the Control Modules and not Variables. Therefore, the mapping will only provide access to the properties that are parameters and the product synchronization will modify/access the values of the _Par_Conn parameter connections (Name_Par_Conn, Description_Par_Conn, etc.). Access to variable initial values (PCCH_Init_Val, etc.) is not supported by the product.

Mapping to the Control Properties aspect is handled differently depending upon the type of Control Module object being mapped to and are described as follows:



Control properties of the particular libraries (e.g. PCdevlib, reuse lib.) are exposed in PETI while object mapping only when these libraries are used or loaded in CBM project. The control properties of the control module cannot be read until all the libraries are loaded into CBM.

- Simple Control Module – Mapping to the Control Properties aspect of a simple control module (PCDeviceLib\Transmitter) is performed by directly mapping to the properties in the Control Properties aspect where the different available parameter connections are listed (Figure 30).

Aspect/Property	Source Object	Source Property	Dir
Operator Note			
Function			
Control Properties			
Name	\$FT	TAGNAME	▶
Description	\$FT	SERVICE	▶

Figure 30. Simple Control Module Mapping

- Child Control Module – In case of a child control module object that is part of a larger composite object (PCDeviceLib\ControllerPIDLoop\Tx), the Control

Properties aspect will list all the parameter connections for the child object that are available at the top level control module object and are connected internally (Figure 31).

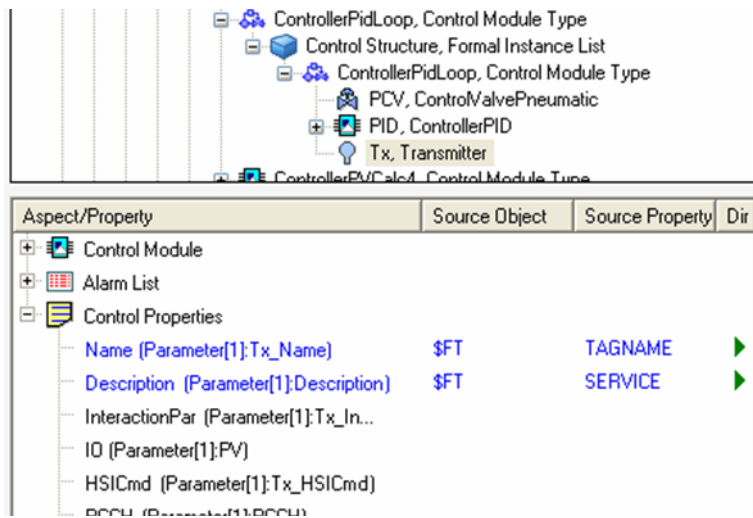


Figure 31. Child Control Module Mapping



The Control Properties of the Child Control Module object are listed as:

- Name(Parameter[1]:Tx_Name) or in general terms,
Child_Parameter_Name (Paramater[TopLevel]: Top_Level_Parameter_Name

The Top Level specifies the number of levels above the current child where the top parent (or ancestor) is present (1 for parent, 2 for grandparent, etc.). When the user maps an INtools/SPI property to one of properties in the Control Properties aspect for a child control module, the product accesses and modifies the value of the parameter at the top ancestor level based on the parameter information that is in the mapping file.

Expert Mapping and Blank Object Type GUID



A sample ExpertMapping.xml file can be found in the following location: \\ABB Industrial IT\\Engineer IT\\Process Engineering Tool Integration\\bin.

There are occasions where a property is accessible only when the object has been instantiated. For example, the Fieldbus Management aspect of Hart device objects is

not accessible in the Object Type Structure, but is accessible in the Control Structure once the object has been instantiated. To map to such properties, the user may edit a special file (ExpertMapping.xml) which is found in the folder where the executable files of the product are installed. A sample copy of the file is shown (Figure 32).

```
<?xml version="1.0" ?>
<ExpertMappingTable>
  <ObjectMapping ABBObjectTypePath="[Object Type Structure]Object
Types/Field Devices/HART Transmitter/ABB Generic HART Transmitter"
  ABBObjectTypeGUID="">
    <AspectMapping ABBAspectName="Fieldbus Management" ABBAspectGuid="">
      <AttributeMapping ABBAttributeName="ParentChannel" />
    </AspectMapping>
  </ObjectMapping>
  <ObjectMapping ABBObjectTypePath="[Object Type Structure]Object
Types/Field Devices/HART Actuators/ABB Generic HART Actuator"
  ABBObjectTypeGUID="">
    <AspectMapping ABBAspectName="Fieldbus Management" ABBAspectGuid="">
      <AttributeMapping ABBAttributeName="ParentChannel" />
    </AspectMapping>
  </ObjectMapping>
</ExpertMappingTable>
```

Figure 32. Sample Code

For the Expert Mapping file as shown above, the product will make the ParentChannel property of the Fieldbus Management aspect available for mapping whenever the object type in the 800xA path ([Object Type Structure]Object Types/Field Devices/HART Transmitter/ABB Generic HART Transmitter) is selected on the mapping page.

Other aspects and properties which can be made accessible only by Expert Mapping include the Control Properties aspect of controllers for IP addresses and the Control Properties aspect of Ethernet objects for IP address, subnet mask, and Enable Channel properties. Mappings for these two examples are included in the sample ExpertMapping.xml file.

To facilitate usage, the GUID of the object type (ABBObjectTypeGUID) does not need to be specified in the Expert Mapping file. The product searches for the object type in 800xA based on its object type path.



The object type path must include a library version number in order for the product to uniquely identify the object type.

For example:

[Object Type Structure]Object Types/Control System/AC 800M\C
Connect/Libraries/Hardware/S800IoModulebusHwLib 1\0-4/AI810.

The functionality of identifying object type GUID based on object type path applies to regular map files. In cases where an object type is not yet available in the Object Type Structure, but that its future object type path is already known, the user may add the mappings manually in a regular map file, with the object type GUID left blank. For example, the following mapping is legitimate when manually added to a map file:

```
<ObjectMapping CAEXObjectPath="/CAEXFile/SystemUnitClassLib
[GlobalSystemUnitLibName='INTOOLS Class
Library']/SystemUnitClass
[SystemUnitClassName='$PneumaticValve']"
ABBObjectTypePath="[Object Type Structure]Object
Types/Control System/AC 800M\C Connect/Libraries/
PCDeviceLib 5\2-0/Control Module
Types/ControlValvePneumatic" ABBObjectTypeGUID=""
MappingCondition="Normal"
ABBObjectTypePathFS=""
ABBObjectTypeGUIDFS="" />
```

Function Designer Template

In addition to instantiating from the Object Type Structure, the product supports instantiating (copying) from the Functional Structure when the Function Designer workflow is selected.

As shown in [Figure 33](#), CAEX object type Copy_FD is mapped to FIC1000_FD which is a Function Diagram object in the Functional Structure. After executing the product comparison and data transfer, a new function diagram object based on FIC1000_FD and renamed as FIC2020_FD is created as shown in [Figure 34](#).

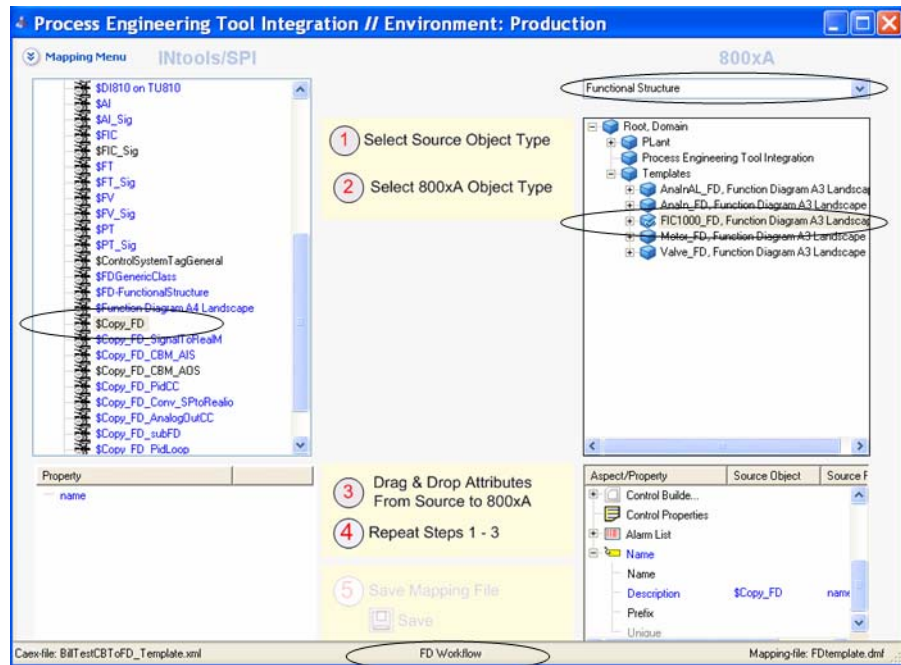


Figure 33. Mapping to Functional Structure

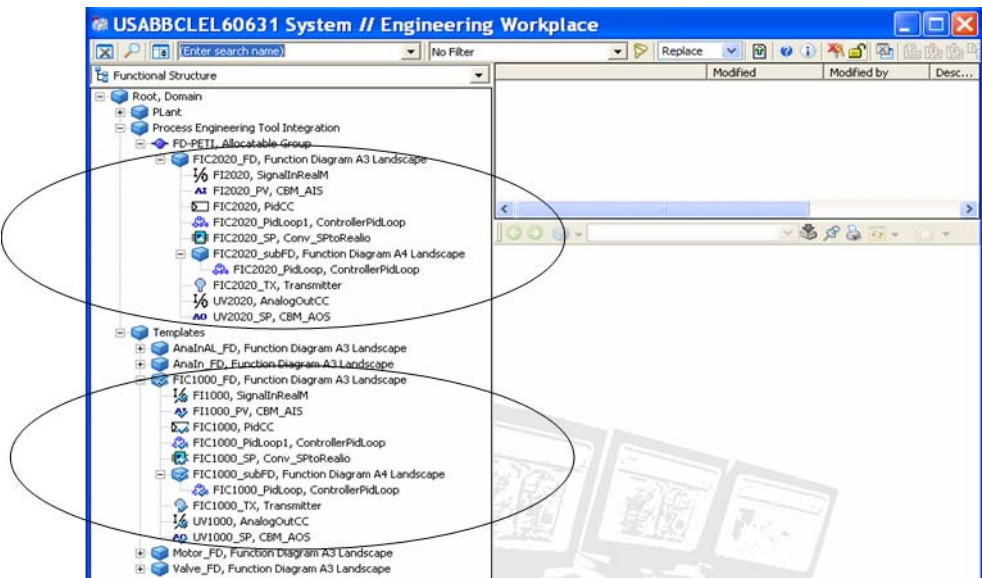


Figure 34. Copying Function Diagram Object

Similar to composite objects, all constituent objects that reside in the function diagram should also be mapped as shown in Figure 35..

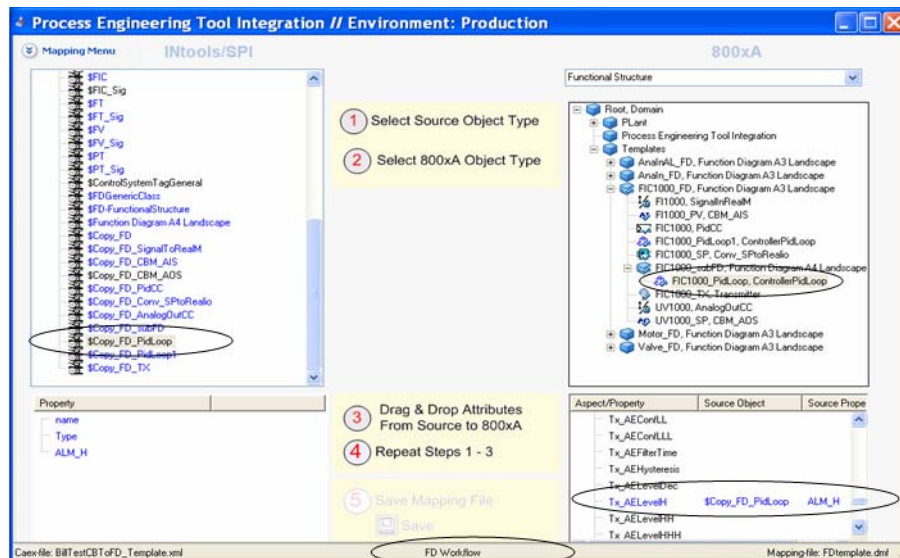


Figure 35. Mapping for Constituent Objects in Functional Diagram

If not, the constituent objects will still be created as the base function diagram is copied, and will be renamed as part of the function diagram automatic naming process. There will be error messages in the Transfer Action List. For example, the list as shown in Figure 36 indicates that object UV2020_SP was not mapped. The

properties of the unmapped objects will also not be updated.

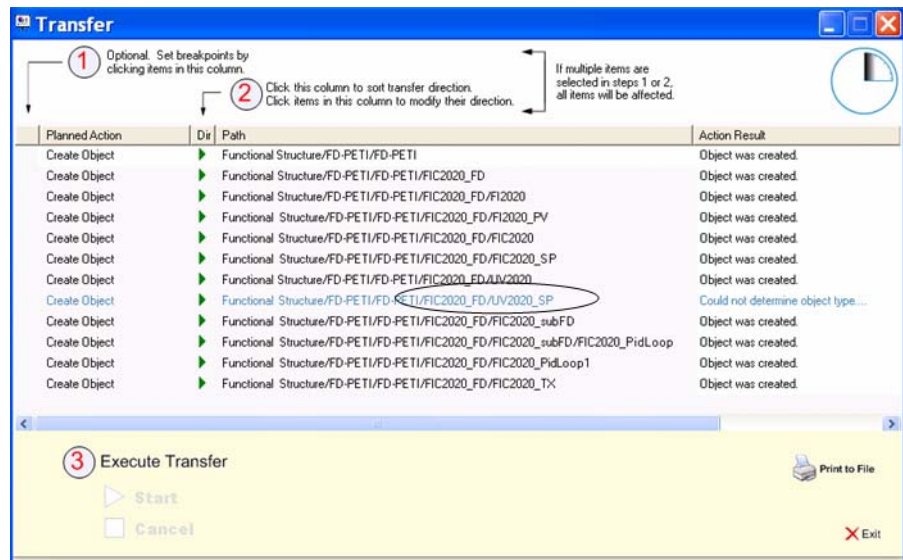


Figure 36. Sample Transfer

Once the mappings are done, the execution of the product will proceed as if the objects were mapped to the Object Type Structure. There is no other difference discernible to the user.



Sample files for instantiating from FD templates can be found in the following location: C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\FD Template Demo.

Allocation of Function Diagrams to Applications and Applications to Controllers

While instantiating a Function Diagram from a template, the user may optionally allocate the Function Diagram to a control application and allocate the control application to a controller.

To do this, perform the following steps:

1. Add two attributes to the Function Diagram object in the CAEX file and set their values to the names of the control application and the controller to which the function diagram and control applications are to be allocated.



In CDM, add two properties to the Function Diagram object and map them to the Control Application Name field and the Controller Name field of the Access table.

2. Run the software and navigate to the mapping page.
3. Map the new attributes to the CBApplication and CBController properties of the Allocatable Group aspect of the Function Diagram.



The CBApplication and CBController properties are grayed out indicating that these properties are not directly writable, but they are still mappable.

4. Start the transfer process for the Functional Structure to instantiate the function diagram (refer to [Section 4, Data Transfer](#) for more information). The corresponding objects in the Control Structure are created under the specified control application. If the control application does not exist, it will also be created by the software. The control application will also be allocated to the specified controller, if it exists.



The allocation of a Function Diagram to an application must be done upon the instantiation of the Function Diagram. However, the allocation of the application to a controller may be done in a separate software run. In that case, set the Compare Attribute flag for the software Compare step in order to identify the missing allocation and the software will allocate the application to the specified controller in a subsequent data transfer step. Refer to [Section 4, Data Transfer](#) for more information on Compare and Data Transfer steps.

Section 4 Data Transfer

Introduction

The Data Transfer function of the product is used to synchronize the data between the INtools/SPI data source and the 800xA System. The data synchronization can involve the creation of new objects as well as changes in values of properties of existing objects.

The product allows bidirectional data transfer between the INtools/SPI data source and 800xA System. The direction of the data transfer is determined by the direction specified in the mapping file for the INtools/SPI data source. Refer to [Section 2, Workflows](#) for mapping information.

Operation

Once the product has been started, perform the following:

1. Select the **Transfer Data** option from the Start window.
2. Select the source of the INtools/SPI data to be transferred.
3. Select the Workflow to be used during the data synchronization. Refer to [Section 2, Workflows](#) for more information.



Checking the **Process Tags only (ignore loops)** check box in the Select Source dialog retrieves only tags from INtools/SPI database. Loops are completely ignored and all tags irrespective of their loop assignments are made available to the user.

The data is now loaded into the product and the main window appears. Depending on the security privileges of the user, the option of synchronizing the data between the INtools/SPI data source and the 800xA System in either the ExpressSync mode or the ExpertSync mode is available.

For more information about access to these modes based on user security, refer to [Licensing and Security](#) on page 19. The Data Transfer function is divided into two parts:

- [Data Comparison](#) on page 58.
- [Data Transfer](#) on page 64.

During the Function Diagram Data Transfer, an error message may appear as shown in [Figure 37](#). To overcome this issue, the user should select Convert to FD workflow option (refer [Selection of Workflow](#)), compare and then transfer in FD workflow.



Figure 37. Function Diagram Data Transfer Error

Data Comparison

The product compares the data in the INtools/SPI data source and the 800xA System and then displays the differences between the two systems. The differences are displayed as shown in [Table 4](#).

Table 4. Color Codes

Color	Description
Green	New objects on either side.
Red	Different values for the same object/property on either side.
Yellow	Uncommitted objects/properties on either side.
White	Completely synchronized objects between both sides.

The following steps occur during comparison (Figure 38):

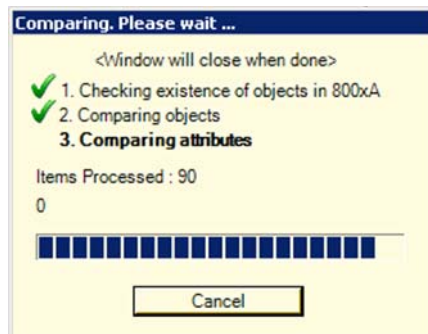


Figure 38. Comparison

1. The product first checks for the existence of the objects in 800xA.



The product checks for the objects in all branches of the Root node.

2. The product identifies the new objects that need to be created in 800xA. To limit the search to a branch of the structure, refer to [Setting Up Start Object For Compare](#) on page 59 for more information. If an object already exists but is of an object type that is different from the one intended, the object will be marked as new. However, it will not be created in 800xA during Data Transfer. Instead, an error message will be displayed to indicate such discrepancy.
3. For existing objects in 800xA, the product compares the property values and creates a related transaction if the values are not the same.



The last step of property comparison is optional and can be left out (by unchecking the check box on the Compare dialog) to improve performance. The property comparison is a lengthy step and may require 10 - 15 minutes of processing for large systems.

Setting Up Start Object For Compare

If a tree node is not selected in the INtools/SPI tree and the Compare icon is clicked on, the entire tree will be compared with the corresponding hierarchy in 800xA. The Compare label will be appended with the word Full to indicate that it is a full

comparison (Figure 39).

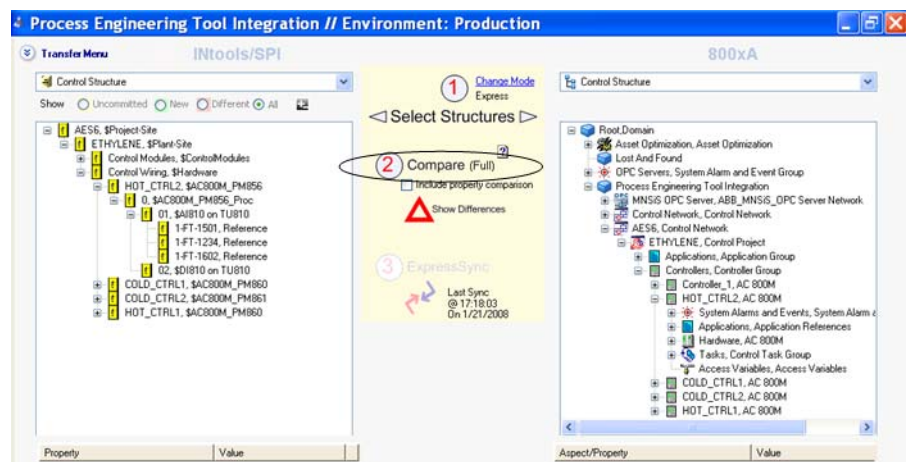


Figure 39. Full Comparison

To limit the comparison to a specific branch of the tree, click on a tree node to select it as a start object. The Compare label will be appended with the word Partial to indicate that it is only a partial comparison. When the Compare icon is clicked on, only that branch of the tree headed by the selected object will be compared. For example (Figure 40), controller HOT_CTRL2 is selected, therefore, only that

controller and its children will be compared.

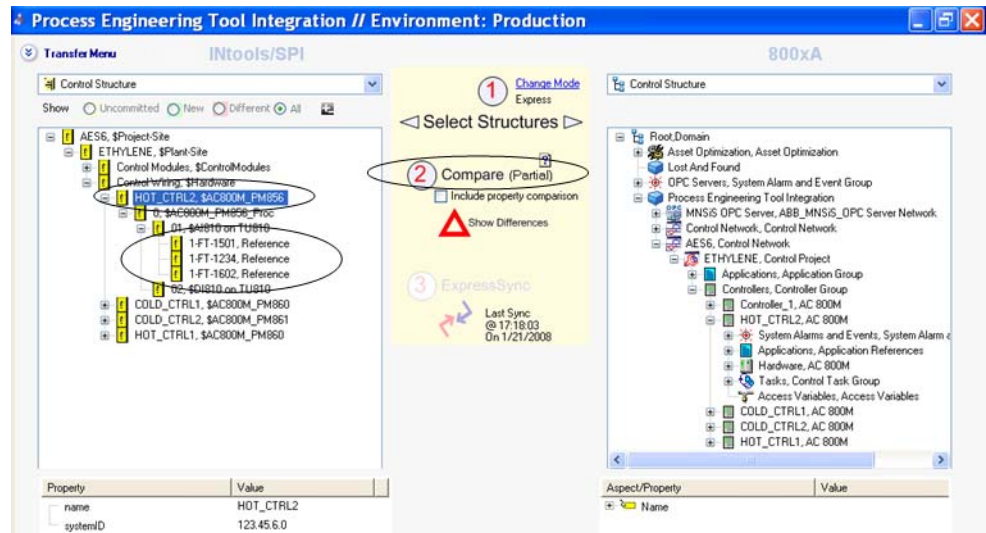


Figure 40. Partial Comparison



After a partial comparison, only that branch of the tree is available for synchronization. For example, Figure 40 shows only controller HOT_CTRL2 and its children being synchronized.

If the parent of the selected object already exists in 800xA, then the selected object and its children will be created or updated as children of the corresponding parent 800xA object.

If the parent of the selected object does not exist in 800xA and the current structure is Control Structure, then the user may activate the Advanced Mode of Synchronization and select the parent object in 800xA by dragging and dropping the selected INtools/SPI object on it.

For the other structures where Advanced Mode of synchronization is not available, an error message will be displayed, indicating that the selected object is not valid for partial synchronization even though the partial comparison has been completed.

The start object selection is also applicable to attribute comparison of existing 800xA objects that have been created by the product. For example in Figure 40,

only the attributes of HOT_CTRL2 and its children will be compared with the corresponding 800xA objects.

For object comparison, all created and inserted objects in the selected branch will be compared. For attribute comparison, however, inserted objects (labeled as Reference in the CAEX tree) are not compared. For example in [Figure 40](#), the attributes of signal objects such as 1-FT-1501 will not be compared. Instead, they are compared as part of the Functional Structure since the signal objects are created in that structure.

To switch back to full comparison after selecting a start object, click on the top tree node to select the entire structure. In cases where there are multiple top nodes, switch to another structure and switch back to the intended structure to clear the node selection for full comparison.

Partial Compare on 800xA Tree View

The partial compare feature enables the user to reduce the overall time to compare the objects by selecting only the required objects in the 800xA tree view.



Do not use Partial Compare for Object Reconciliation.



Partial compare feature compares only the objects present in the selected tree / subtree in the 800xA window.



Partial compare feature performed on 800xA tree view saves a lot of time.

Follow the steps below to compare the selected objects in the 800xA Tree View window:

1. Select the required structure (Example: Control Structure) in both INtools/SPI and 800xA windows as shown in [Figure 41](#).
2. Click **Comparison** tab under **Options**.
3. Select **Include 800xA partial comparison** check box.

- Click **Yes** in the appeared warning message box.

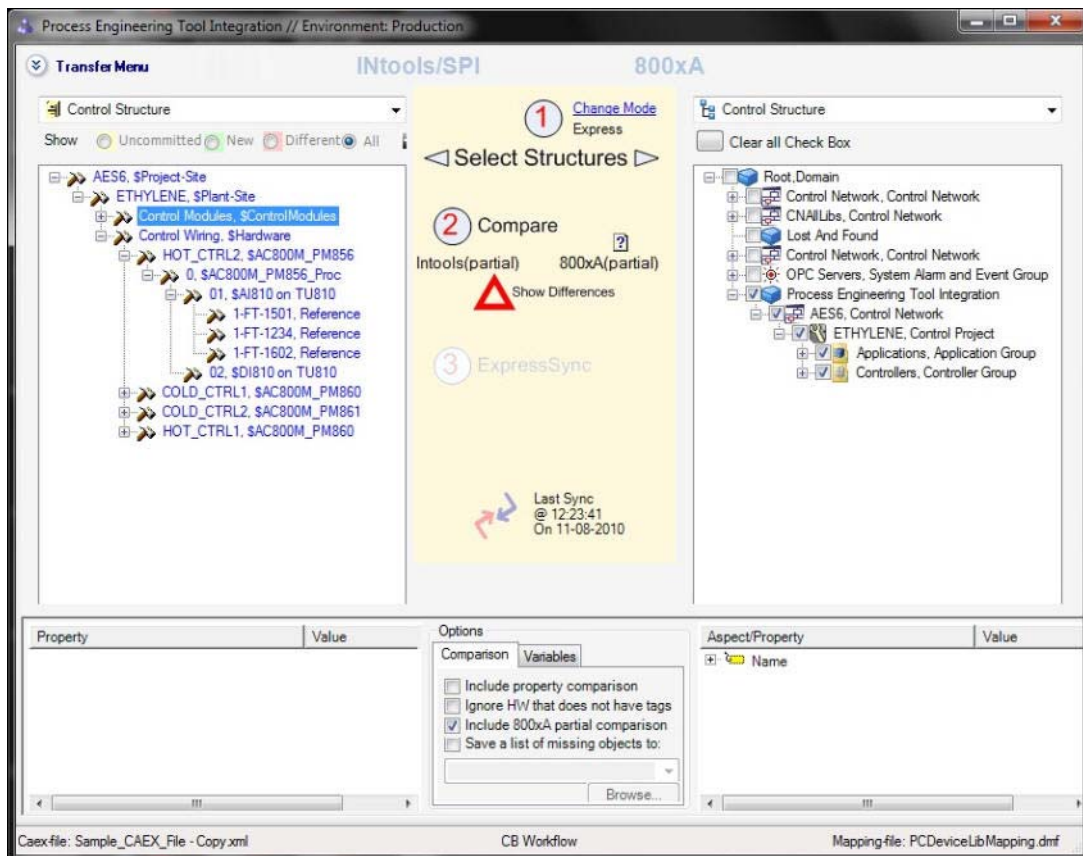
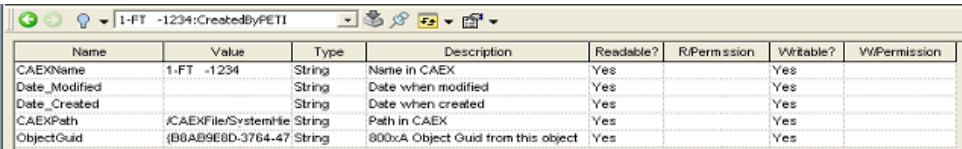


Figure 41. INtools/SPI and 800xA Object Comparison Window

- In the 800xA window select the required tree / sub tree, from where the comparison needs to be done. For example refer to Figure 41.
- Click **Compare**.
- Click **Clear all Check Box** to clear all the selected check boxes.

Data Transfer

Once the user accepts the differences presented and then synchronizes the data. The new objects are created in the 800xA System as well as updating the values of properties on either side depending on the data transfer direction specified in the mapping file. All objects created by the product contain the **CreatedByPETI** aspect. The details of the aspect are shown in the [Figure 42](#).



Name	Value	Type	Description	Readable?	R/Permission	Writable?	W/Permission
CAEXName	1-FT -1234	String	Name in CAEX	Yes		Yes	
Date_Modified		String	Date when modified	Yes		Yes	
Date_Created		String	Date when created	Yes		Yes	
CAEXPath	CAEXFile\SystemFile	String	Path in CAEX	Yes		Yes	
ObjectGuid	{B8AB9E8D-3764-47}	String	800xA Object Guid from this object	Yes		Yes	

Figure 42. CreatedByPETI Aspect

Data Transfer in ExpressSync Mode

The ExpressSync Mode of the Data Transfer function allows the user to synchronize the data between the INtools/SPI data source and the 800xA System similar to the data synchronization performed by a Personal Digital Assistant (PDA) with a computer. The data in the two systems are compared and then synchronized to remove the differences. The synchronization may involve the creation of new objects if necessary in the 800xA System in the same hierarchy as that of the INtools/SPI data source. The user does not have the option to override the hierarchy of the new objects. The synchronization may also involve updating the values of properties of objects on either side. The direction of the update is initially based on the direction specified in the mapping file and the user has the option to override the direction of the update provided the user is an 800xA System user.

The user then follows a series of steps to perform the Data Comparison and Data Transfer operations in order to synchronize the data between the INtools/SPI data source and the 800xA Systems. Perform the ExpressSync operation as follows:



The following steps are for 800xA Systems using ABB's PCDeviceLib objects.

1. Select the Control Structure on both the INtools/SPI side and the 800xA side.

- Once the structures have been selected, the **Show Differences** link is activated. Click on the link to see the differences between the INtools/SPI data source and the 800xA System. The differences are colored according to the color codes specified in [Table 4](#).



If the user selects different structures on each side, the product will generate an error message.



The user can expand/collapse the data in the tree structures on either side by clicking on the plus and minus buttons.

- Once the differences between the two sides have been viewed, click the **ExpressSync** icon to list the transactions of the data from the two sides. The product displays the Transfer window ([Figure 43](#)) which shows the transactions that are executed in [Step 4](#).

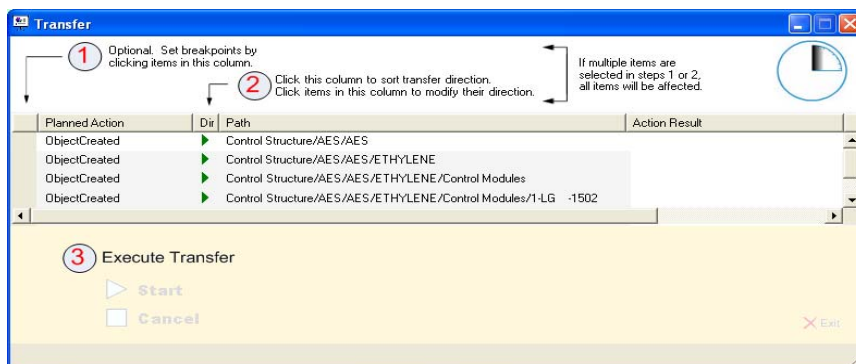


Figure 43. Data Synchronization Steps

These transactions can include the creation of new objects on the 800xA side, insertion of objects on the 800xA side, or changes to the values of properties of objects on either side. Additionally, the direction of each transaction is displayed indicating which side is affected by that particular transaction (INtools/SPI or 800xA). The direction is determined by the configuration of the mapping file. If the user has System User privileges in the 800xA System,

the option to override the transfer direction of any of the individual execution transactions is available.



New Objects can only be created on the 800xA side. The product does not create new objects on the INtools/SPI side.



For transactions that have an Unknown data direction specified, the user is prompted to provide the data transfer direction.

4. Click on **Start** to execute the list of transactions in the Transfer window. The window shows the progress of each step in the **Action Result** column. The overall progress is shown by the progress bar at the bottom of the window. The user can cancel the execution of the transactions by clicking on the **Cancel** link at any time during the progress of the execution. Once the execution has been completed, click on the **Exit** link to close the Transfer window.



The user is not able to click on **Start** until all the unknown directions have been specified.

5. The user can click on **Show Differences** again to verify the updates that were performed in the execution of the transfer steps.
6. Select the Functional Structure on both sides and repeat [Step 2](#) through [Step 4](#) to synchronize the data between the two sides.
7. Once the Functional Structure synchronization is complete, select the Control Structure again on both sides and repeat [Step 2](#) through [Step 4](#) to complete the synchronization of the data.
8. Open the Engineering Workplace of the 800xA System. Right-click on the **Controllers, Controller Group** under the Process Engineering Tool Integration area and select **Advanced >Write Allocation into CBM**. This will insert the Global variables created for the 800xA PCDeviceLib Control Module objects into the 800xA System.

The ExpressSync Mode synchronization of data between the INtools/SPI data source and the 800xA application is now complete.



Clicking on the green direction arrow toggles between the states as described in Table 1. In addition to the three states described, user can choose the red circle with line through it and block the selected transaction(s) from executing.



Select multiple transactions using standard Windows methods, then click the direction column to change direction on multiple items

Data Transfer in ExpertSync Mode

The ExpertSync mode of data synchronization between the INtools/SPI data source and the 800xA System is for expert users that have a good understanding of the data structures in the 800xA System. The functionality of the ExpertSync mode is very similar to the functionality of the ExpressSync mode with one difference. The expert user has the ability to drag-and-drop new objects from the INtools/SPI side to any allowed location on the 800xA side changing the hierarchy of the objects on the 800xA side from that on the INtools/SPI side. The product will then create the new objects on the 800xA side at the location of the dragged-and-dropped objects.

The user can select the ExpertSync mode by clicking on the **Change Mode** link at the top of the product window.



The ExpertSync mode is only available to users that have System User privileges in the 800xA System. The mode is not available to users with only Application User rights. Refer to [Licensing and Security](#) on page 19 for more information.

Perform the ExpertSync operation as follows:



The following steps are for 800xA Systems using ABB's PCDeviceLib objects.

1. Select the Control Structure on both the INtools/SPI side and the 800xA side.
2. Once the structures have been selected, the **Show Differences** link is activated. Click on the link to see the differences between the INtools/SPI data source and the 800xA System. The differences are colored according to the color codes specified in [Table 4](#).



If the user selects different structures on each side, an error message is generated.



The user can expand/collapse the data in the tree structures on either side by clicking on the plus and minus buttons.

3. The user has the option to drag-and-drop objects from the INtools/SPI data structure to any allowed location in the 800xA data structure. Once the objects have been dragged-and-dropped, they appear on the 800xA side.



If the user does not drag-and-drop objects in the ExpertSync mode, the objects are created in the same hierarchy as the INtools/SPI data source similar to the functionality in the ExpressSync Mode.

Once the differences between the two sides have been viewed, click on the **Commit Changes** icon to start the synchronization of the data between the two sides. The product will display the Transfer window, which displays the transactions that can be executed to synchronize the data. These transactions may include the creation of new objects on the 800xA side, insertion of objects on the 800xA side, or changes to the values of properties of objects on either side. Additionally, the direction of each transaction is displayed indicating which side is affected by that particular step (INtools/SPI or 800xA). The direction is determined by the configuration in the mapping file. If the user has System User privileges in the 800xA System, then the option to override the transfer direction of any of the individual execution transactions is available.



New Objects can only be created on the 800xA side. The product does not create new objects on the INtools/SPI side.



For transactions that have an Unknown data direction specified, the user is prompted to provide the data transfer direction.

4. Click on **Start** to proceed the execution of the listed transactions in the Transfer window. The window shows the progress of each step in the **Action Result** column. The overall progress is shown by the progress bar at the bottom of the window. The user can cancel the execution of the transactions by clicking on the **Cancel** link at any time during the progress of the execution. Once the execution has been completed, click on the **Exit** link to close the Transfer window.



The user is not able to click on **Start** until all the unknown directions have been specified.



Pause the sequence of execution of the transactions in the Transfer window by placing a Breakpoint at any of the individual steps in the execution list. When the product encounters a Breakpoint, the execution will pause at that step and then click on **Continue** to proceed the execution of subsequent steps.

5. The user can click on **Show Differences** again to verify the updates that were performed in the execution of the transfer steps.
6. Select the Functional Structure on both sides and repeat [Step 2](#) through [Step 4](#) to synchronize the data between the two sides.

7. Once the Functional Structure synchronization is complete, select the Control Structure again on both sides and repeat [Step 2](#) through [Step 4](#) to complete the synchronization of the data.
8. Open the Engineering Workplace of the 800xA System. Right-click on the **Controllers, Controller Group** under the Process Engineering Tool Integration area and select **Advanced > Write Allocation into CBM**. This will insert the Global variables created for the 800xA PCDeviceLib Control Module objects into the 800xA System.

The ExpertSync Mode synchronization of data between the INtools/SPI data source and the 800xA application is now complete.

Object Reconciliation

The Object Reconciliation utility has several purposes:

- To reconcile 800xA objects that were created by the software with CAEX objects that have been renamed in the SPI database and CAEX files since the last transfer.
- To reconcile 800xA objects that were created by the software with CAEX objects that have been moved within or deleted from the SPI database and the CAEX files since the last transfer.
- To reconcile CAEX objects with 800xA objects that were manually created and thus do not have the CreatedByPETI aspect.

Execute the following steps to perform object reconciliation:

1. Select the Workflow to be used during the data synchronization.
2. Select **Transfer Data** option from the Start window.
3. Select the source of the INtools/SPI data to be transferred.
4. In the **Transfer Data** window, select **Control Structure** for both - INtools/SPI and 800xA.

5. Click **Show Differences** to calculate the differences between both sides.
6. Click **Transfer Menu** as shown in [Figure 44](#).

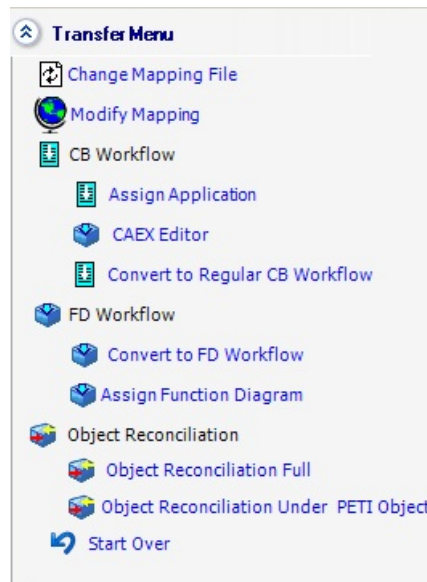


Figure 44. Transfer Menu List

7. Under **Object Reconciliation** option, select either of the following:
 - **Object Reconciliation Full** - This functionality allows reconciliation of all objects located under the control structure.
 - **Object Reconciliation Under PETI Object** - This functionality allows reconciliation of objects located only under PETI object.

The Object Reconciliation window will be displayed as shown in [Figure 46](#).

CreatedByPETI Aspect

All 800xA objects that are created by the software have the CreatedByPETI aspect as shown in [Figure 45](#). The following properties of the aspect are relevant to object reconciliation:

- CAEXPath - The full CAEX path that ends with the object name.
- CAEXID - The full CAEX path that ends with the object ID.

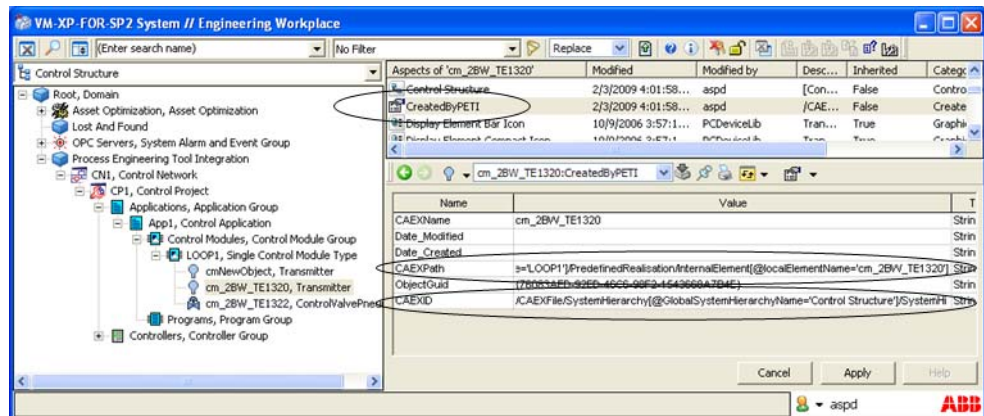


Figure 45. CreatedByPETI Aspect

Object Categorization

Objects that need to be reconciled are categorized into renamed objects, moved or deleted objects, and new objects. They are listed on separate tab pages of the Object Reconciliation utility as shown in Figure 46, Figure 47, and Figure 48. They are categorized as follows:

- Upon activation, the Object Reconciliation utility adds all the new CAEX objects that have been identified in the software Compare step to the Reconcile New Objects tab page as new CAEX objects for which counterparts could not be found in the 800xA system.
- The Object Reconciliation utility then goes through all the 800xA objects under object Process Engineering Tool Integration in the current structure and for each object:
 - Get its CreatedByPETI aspect. If there is no CreatedByPETI aspect, add the object to the Reconcile New Objects tab page. The object is a new 800xA object.

- If there is a CreatedByPETI aspect, get its CAEXPath property value and checks if the corresponding object exists in the CAEX file. If so, do nothing. The CAEX object has neither been renamed nor moved.
- If the corresponding CAEX object cannot be found by its CAEXPath, try to find it by its CAEXID. If found, add the 800xA and the CAEX objects to the Reconcile Renamed objects tab page. The CAEX object has been renamed.
- If the CAEX object cannot be found either by its CAEXPath or CAEXID, then add the 800xA object to the Moved or Deleted Objects tab page. This is a tab page with only one list for the 800xA objects that have a CreatedByPETI aspect, but for which the CAEX object cannot be found. The CAEX object may have been moved or deleted.

Renamed Object

Basic Steps

The basic steps for reconciling renamed objects are as follow:

1. Click to open the Reconcile Renamed Objects tab page as shown in [Figure 46](#).

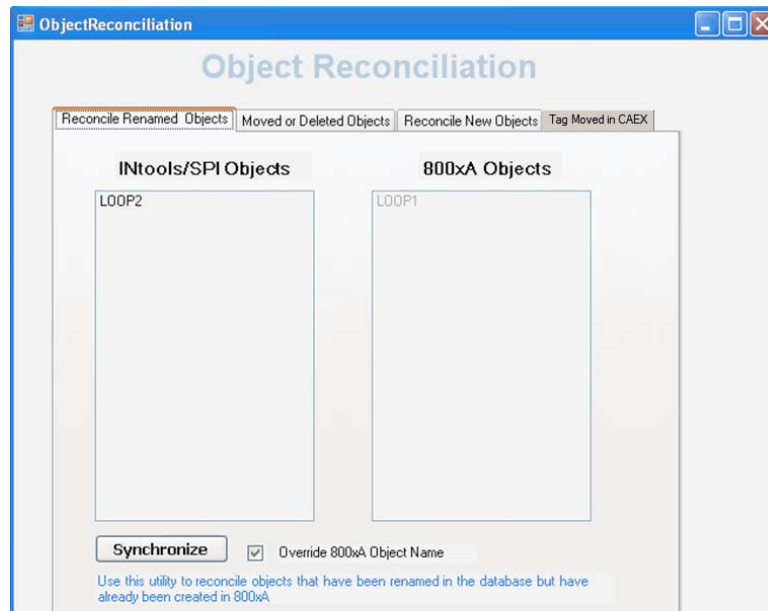


Figure 46. Reconciling Renamed Objects

2. Select an object from the INtools/SPI Objects list box. The corresponding 800xA object based on object ID will be auto-selected.
3. Enable the check box to update the name of the 800xA object that has just been selected.
4. Click **Synchronize** to reconcile the two selected objects.



The Object Reconciliation utility supports selection of multiple objects.

Renamed Objects on Multiple Levels

The reason for properties CAEXPath and CAEXID to include the full CAEX path is to facilitate searching for the objects in the CAEX file. Without the full CAEX paths, the software Compare step takes longer to execute. This works fine for object

reconciliation if the renamed objects are confined to only one level of the structure. However, if any of the parent objects of a CAEX object is renamed, the CAEX path of that CAEX object is changed. Consequently, the 800xA object that has been created for it fails to find it as previously described. The 800xA object is included in the Moved or Deleted Objects list even though the CAEX object has neither been moved nor deleted. It is the parent of the CAEX object that has been renamed.

Renamed Objects in Two Different Structures (Example - Control Builder Workflow)

If a control module in the Control Structure is renamed in the CAEX file, the inserted control module in the Functional Structure is also renamed. Consequently, its child signal object in the Functional Structure will have a different CAEX path resulting in the same discrepancy as previously described.

Recursive Reconciliation of Renamed Objects

In view of the scenarios previously described, the user needs to do multiple object reconciliations. The Object Reconciliation utility supports recursive reconciliation for each structure. It also supports multiple selections. After the user starts the reconciliation process for the selected renamed objects, the software renames the 800xA objects and updates the CAEXPath and CAEXID properties of all their child 800xA objects. It then automatically re-categorizes the objects as previously described and displays any child objects that are now correctly re-categorized as renamed objects.

For each structure, the utility renames the created objects only. The inserted objects are renamed when their created counterparts in the other structure are renamed by the utility. Consequently, the objects should be reconciled in the same order that they were created. The user should reconcile the Control Structure first, then the Functional Structure, and finally the Control Structure.

Moved or Deleted Objects

A CAEX object may have been deleted from the CAEX file. It may have been moved around within the CAEX file. Moving a CAEX object with change its CAEX path and those of its child objects. Consequently, the corresponding 800xA objects will be correctly categorized as Moved and Deleted Objects as previously

described and shown in Figure 47.

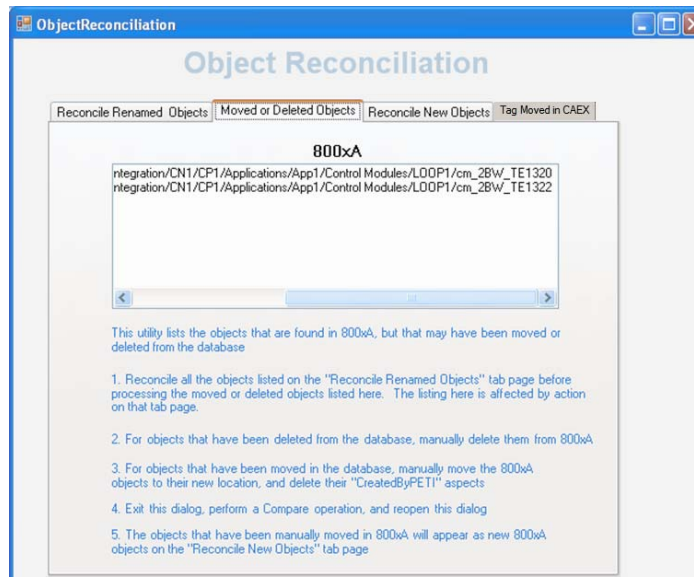


Figure 47. Moved or Deleted Objects

Since the software does not actually move the objects in the 800xA system, the user needs to go through the list of Moved and Deleted Objects, find the objects in the 800xA system, then either move them to their new location or delete them from the 800xA system as indicated in the CAEX file.

When the objects in 800xA are manually moved to their new location, the CreatedByPETI aspects of those objects should be manually deleted as they now contain invalid information. The objects will then be re-categorized as new object by the Object Reconciliation utility when it is activated again. They can then be reconciled as such.

Since the list of Moved or Deleted Objects is affected by action on the objects listed as Renamed objects, the Renamed objects should be reconciled first. Both lists will be updated when the Renamed objects are reconciled. When there is no more Renamed object remains, the objects listed as Moved and Deleted may then be manually processed.

After moving the 800xA objects, the user should then rerun the utility and check if any child objects need to be reconciled. If so, reconcile the renamed objects first and then the moved and deleted objects.

New Objects

As previously described, an object may be categorized as both moved and new by the Object Reconciliation utility. Therefore, after both the renamed and moved objects have been reconciled, the user should stop the Object Reconciliation utility, run the Compare step, and then restart the Object Reconciliation utility to get an updated list of New Objects.

When there are no more entries on the Reconcile Renamed Objects and Moved or Deleted Objects tab pages, the user may proceed to reconcile the new objects. The user can then select one object from the CAEX list and select one object that is deemed to be the same object from the 800xA list and click **Synchronize** to synchronize the two objects (Figure 48).

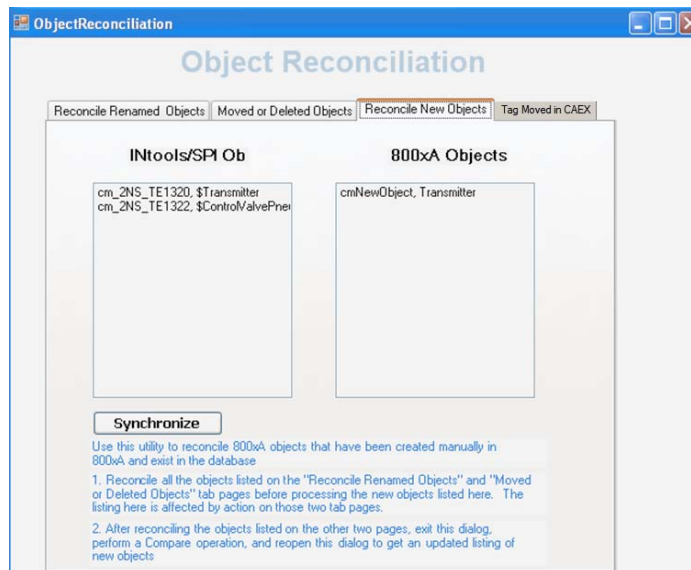


Figure 48. Reconciling New Objects

Tag Moved in CAEX

Perform the following steps to delete the duplicate tag:

1. From the Object Reconciliation utility window, click **Tag Moved in CAEX**.
2. Compare the CAEX Path and the 800xA Path as shown in [Figure 49](#). The duplicate tag moved in the CAEX path (from Application, Single Control Module, Controller (Hardware) and the existing 800xA path will be displayed under **Tag Moved in CAEX** tab.



This functionality works only for pure CB workflow.

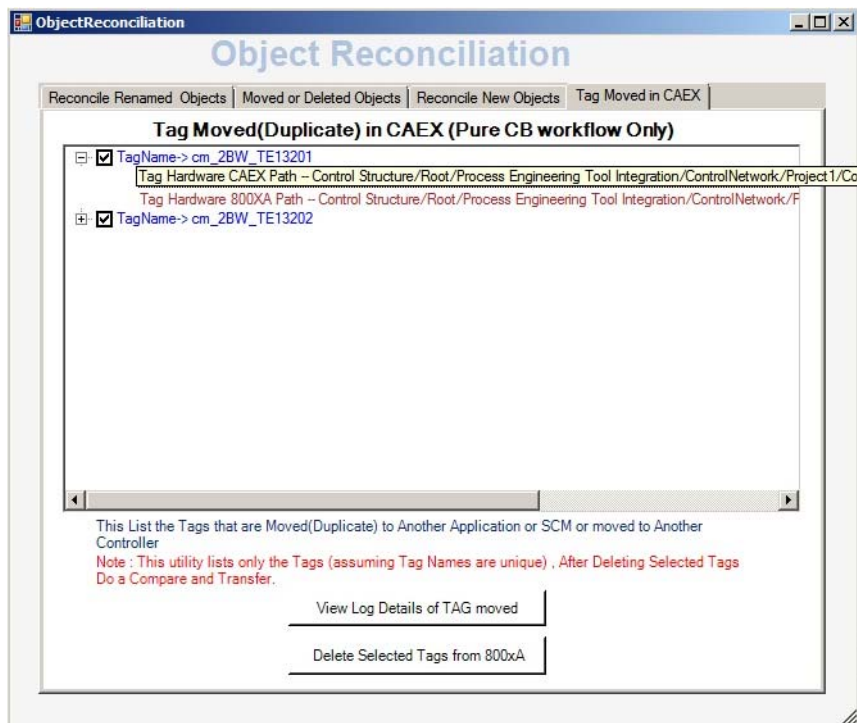


Figure 49. Tag Moved in CAEX

3. Select the tag to be deleted and then click **Delete Selected Tags from 800xA**.
4. Click **Yes** on the dialog box as shown in [Figure 50](#).

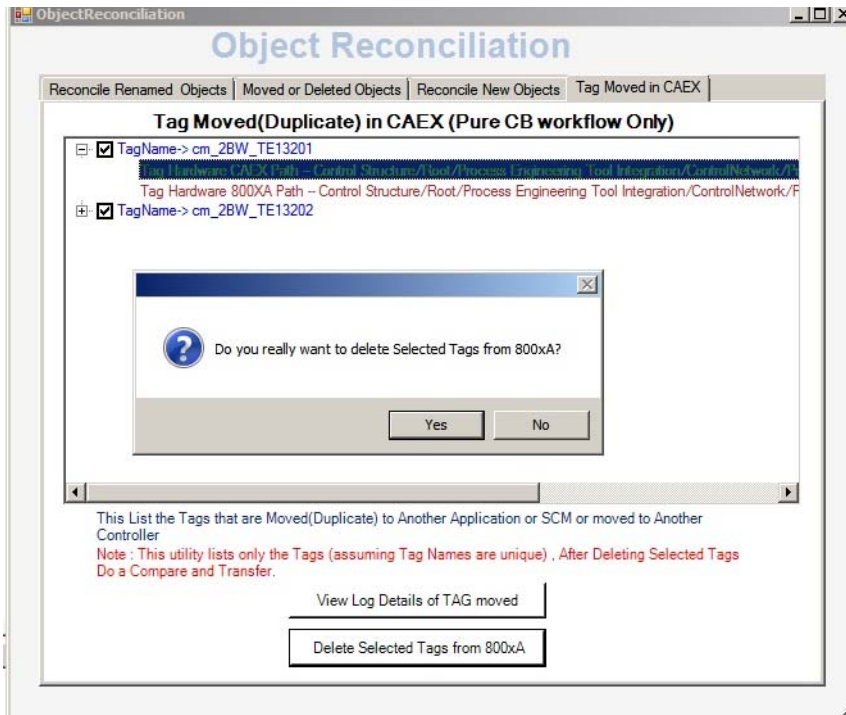


Figure 50. Dialog Box

Reconciliation

The steps for reconciling renamed, moved, deleted, new objects, tag moved in CAEX on multiple levels and in multiple structures are as follows:

1. Perform a Compare step for the Control Structure.
2. Activate the Object Reconciliation utility and recursively reconcile the renamed objects until there is no more entry listed on the Reconcile Renamed Objects tab page.

3. Manually move or delete the 800xA objects as identified by the Object Reconciliation utility on the Moved or Deleted Objects tab page.
4. Exit the Object Reconciliation utility.
5. Repeat [Step 1](#) through [Step 4](#) until there are no more renamed, moved, or deleted objects to be reconciled.
6. Activate the Object Reconciliation utility, reconcile the new objects that are identified on the Reconcile New Objects tab page and then exit the Object Reconciliation window.
7. Repeat [Step 1](#) through [Step 6](#) for the Functional Structure.
8. Repeat [Step 1](#) through [Step 6](#) for the Control Structure again

For Function Diagram workflow, there may not be any object in the Control Structure that needs to be reconciled. In that case, the Object Reconciliation utility needs to be run only for the Functional Structure.

Change Management

Change Management functionality allows users to track modifications done on INtools/SPI and 800xA tools. This functionality records CAEX file path, user name, time stamp, tag name, status, tag path in CAEX file, and tag path in 800xA. These log files are stored in a .xlsx file format.

For viewing the updated **ObjectReconcile** log file, in the object reconciliation utility window, from **Tag Moved in CAEX** tab, click **View Log Details of TAG moved** as shown in [Figure 51](#).

The default location of the **ObjectReconcileLog.xlsx** files are *C:\Program Files (x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\bin\logs*.



If **ObjectReconcileLog.xlsx** exceeds the limit size (~2MB), then the existing file will be automatically renamed to **ObjectReconcileLogBackup.xlsx** and the new log details will be recorded in **ObjectReconcileLog.xlsx** file.

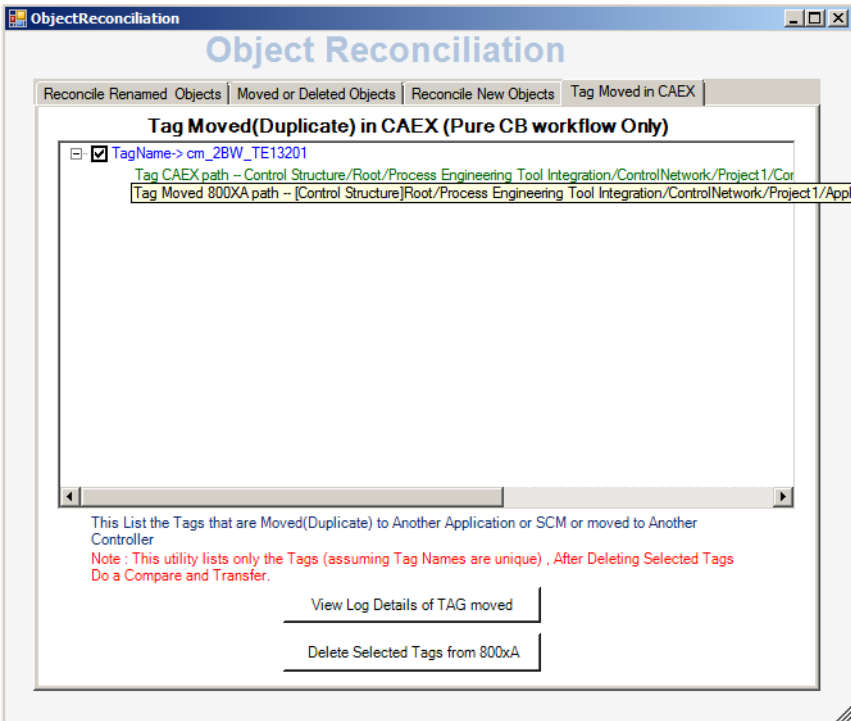


Figure 51. Object Reconciliation Utility Window

Section 5 CAEX Tree Editor

Introduction

CAEX Tree Editor is a tool that allows the user to create a composite object hierarchy and display data in various formats, insert and delete nodes, modify existing nodes, define node attributes, save any Tree Editor work that has been performed, export any CAEX Tree to a database and further edit that database.



It is assumed that Microsoft Access has been installed and is available for use to open any *.mdb (Microsoft Data Base) file.

Running the CAEX Tree Editor

To execute the tool, select the CAEX Editor link from the Main Menu. The current CAEX data will be loaded into the Editor and will be ready for user action.

Manipulating Structure Views

To switch between the Control Structure tree view and the Functional Structure tree view, use the radio buttons located above the large white field and select the button corresponding to the desired view.



Any time the view is switched, the tree view that had been previously active is collapsed to the root node.

Inserting Nodes

To insert a node, select the following:

1. Right-click on the parent of the desired node and select **New** from the drop-down context menu that appears (Figure 52).

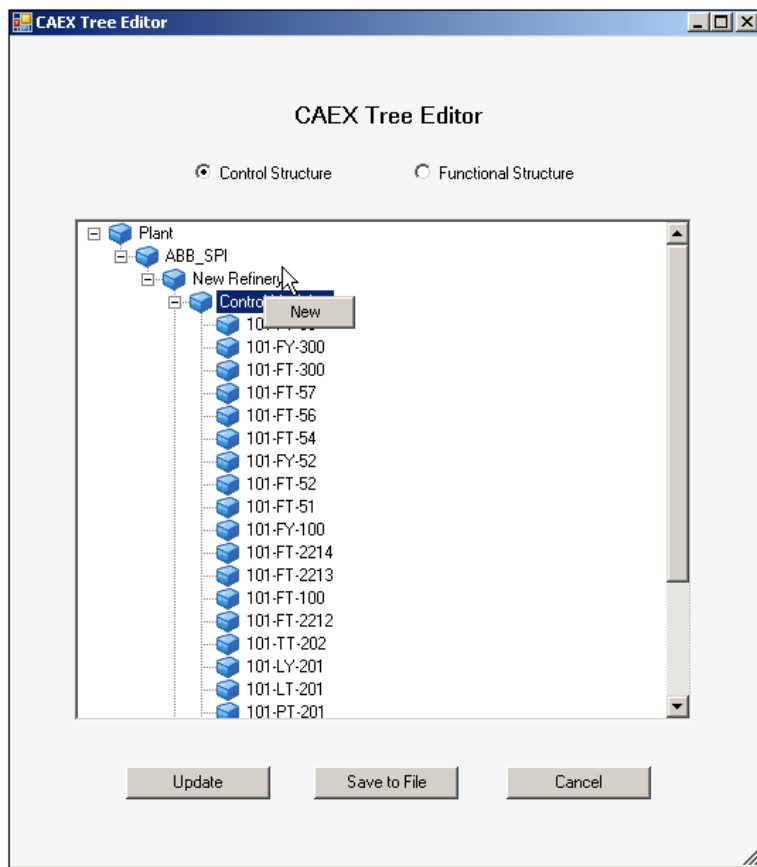


Figure 52. Insert New Node



Nodes that are Control Modules or Signal Objects cannot be deleted and the CAEX Tree Editor will display an error if this action is performed.

2. When a New is selected, the user will be prompted to input both the Node Name and the Node Object Type. Click **OK** to accept any changes made to the node.

Deleting Nodes

To delete a node, select the following:

1. Right-click on the desired node and select **Delete** from the drop-down context menu that appears.



Nodes that are Control Modules or Signal Objects cannot be deleted and the CAEX Tree Editor will display an error if this action is performed.

Deleting a node will also recursively delete any descendents to that node.

Modifying Node Attributes

To modify a node, select the following:

1. Right-click on the desired node and select **Edit** from the drop-down context menu that appears.



Nodes that are Control Modules or Signal Objects cannot be deleted and the CAEX Tree Editor will display an error if this action is performed.

2. When Edit is selected, the user will be prompted to modify both the Node Name and the Node Object Type. Click **OK** to accept any changes made to the node or its type.

Moving Nodes

To move a node, select the following:

1. Select and hold on the desired node the drag the node to a desired parent node. Doing so will place the node that has been moved directly under this new parent, next to any exist siblings.



Moving a node under one of its descendents is not possible and the CAEX Tree Editor will display an error if this action is performed.

Saving Changes

To save the changes made in the Tree Editor, do one of the following:

1. Select the **Update** button. This will save the edited hierarchy to the CAEX data in memory and update the CAEX tree view on the Data Transfer page and it will save the hierarchy to an Access .mdb database file. The .mdb file will receive a name identical to the filename of the CAEX file and the .mdb file itself will be located in the same folder as the CAEX file.

OR

2. Select the **Save to File** button. In addition to the action as described for the Update option, this option will also save the edited data to the CAEX file.



Note that for Web Service, this option is the same as the “Update” option since there is no CAEX file in this case.

Selecting the **Cancel** button will exit the Editor without saving the changes made.

Section 6 Import/Export Utility

Introduction

The standalone application (the product DB Import Export Utility) interfaces with the INtools/SPI database. It is used to transfer data to and from the INtools/SPI database in the form of a CAEX file in case the Web service connection is not available.



When a CAEX file is used as the INtools data source, the product overwrites the same file with the changes that are required to be done in the INtools database. The user has to be diligent in using the same file for repeated sessions and also use the same file for the Import option in this utility to update INtools database.

The standalone application connects to the INtools/SPI database specified in the INtools/SPI INI file. Execute the following steps on the node that contains Intergraph SmartPlant Instrumentation (INtools/SPI) software:

1. Launch the INtools Import Export Utility. The application resides in the <PETI Executables> directory on the workstation running the INtools/SPI software.

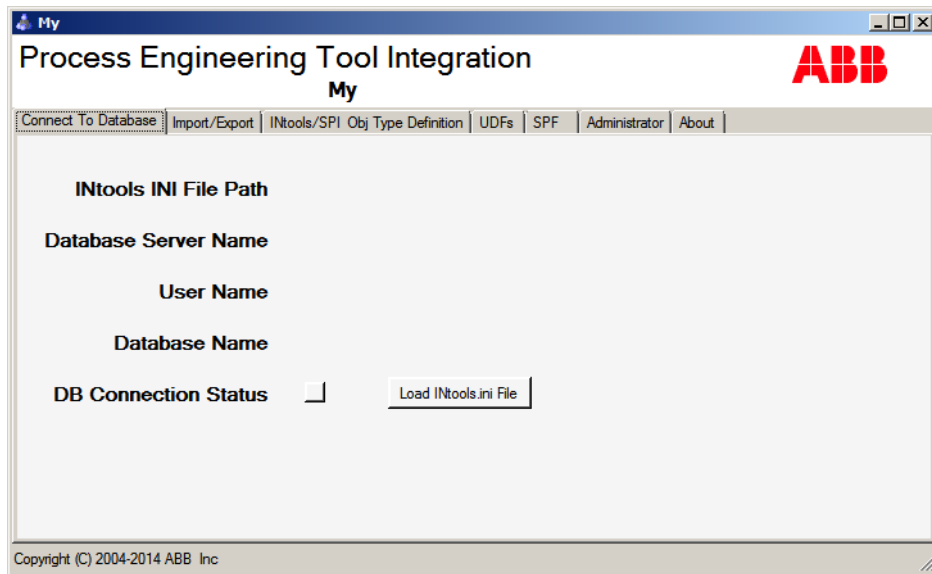


Figure 53. Standalone Application

2. Click **Load INtools.ini File** ([Figure 53](#)).

3. Select the desired .ini file from the Local Disk and click **Open**.

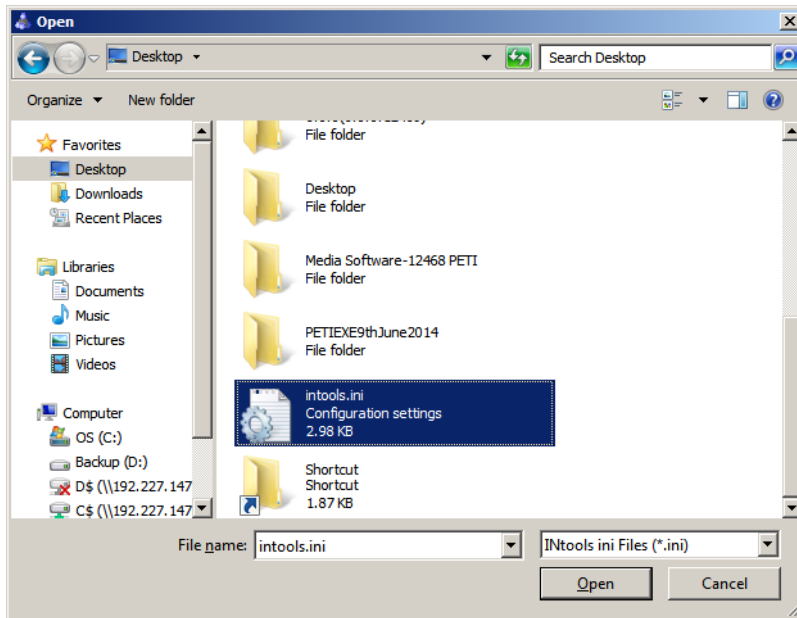


Figure 54. Selecting .ini File from a Location on the Local Disk

4. Click the **Administrator** tab.



The **Administrator** tab is meant for database maintenance and/or administrative tasks. Do not attempt to use the **Administrator** tab feature unless instructed to do so during an installation or by an ABB support person.

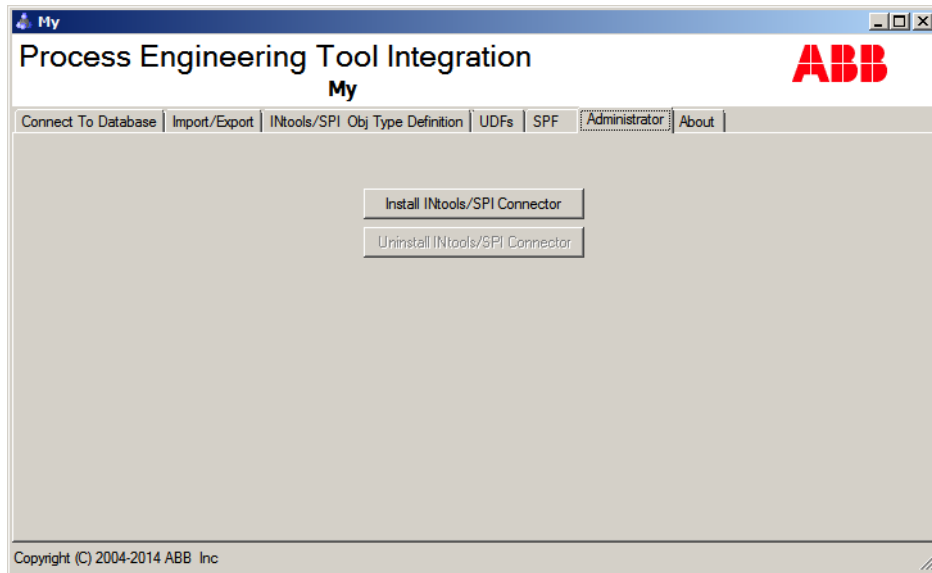


Figure 55. Administrator Tab

The standalone application is used to install or uninstall the INtools/SPI Connector software, which includes the database utilities that are required to import/export data from the INtools/SPI database.

5. Click **Install INtools/SPI Connector**, see [Figure 55](#).
6. On the message displayed click **OK**.

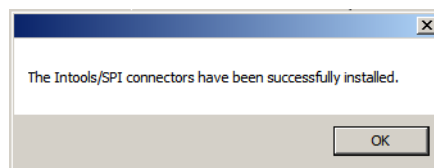


Figure 56. Confirmation Message

The status button turns green indicating the INtools Import Export utility connected to the INtools/SPI database successfully.

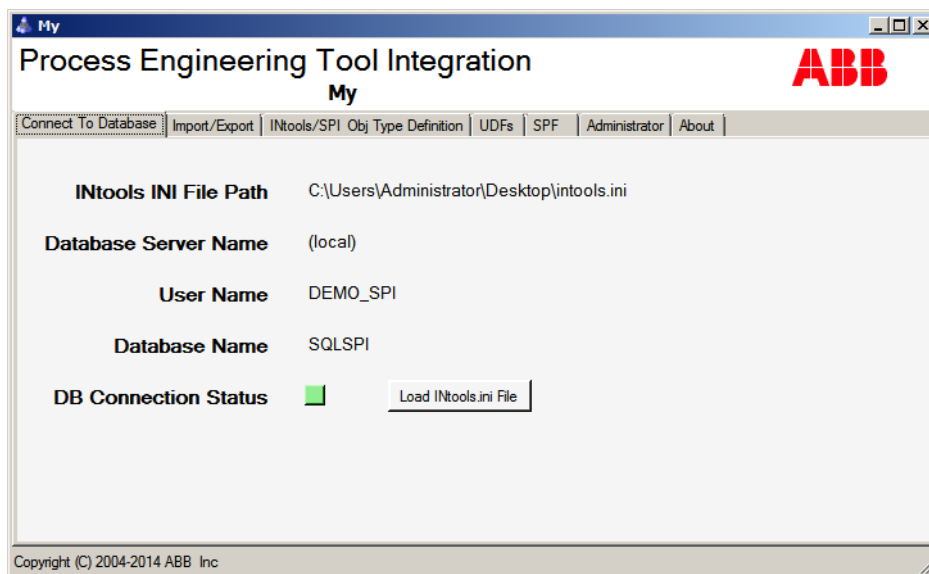


Figure 57. Green Button Indicating the Connection



INtools/SPI data can not be read using Web Services or the CAEX file without successful installation of the connector.

Use the **Import/Export** tab (see [Figure 53](#)) to export the INtools data to a CAEX file, or import data from a CAEX file to update the INtools database.

Use the **INtools/SPI Obj Type Definition** tab (see [Figure 53](#)) to specify the criterion for establishing the various INtools/SPI class types to be used by the software. This is done by selecting the appropriate INtools/SPI Component property as the INtools/SPI Object Type from the drop-down list box.

The **INtools/SPI Obj Type Definition** tab (see [Figure 53](#)) can also be used to override the INtools/SPI Class definition of any tag in the INtools/SPI database on an instance basis. The user can load the INtools/SPI tag information into the grid and then override the 800xA Object Type for any tag to something other than the default INtools/SPI Object Type. The software will then use the updated Object Type when generating the CAEX information for those tags. This will allow the user to create different 800xA Object Types for tags that have the same INtools/SPI

Object Types in the INtools/SPI database by specifying the appropriate mapping information.



The user can apply filters for quick access to specific tags.

Use the **User Defined Fields** tab (see [Figure 53](#)) to specify the inclusion of various INtools/SPI User Defined Fields (UDFs) into the information retrieved for Instruments and Panels. The UDFs appear as additional properties of the corresponding INtools/SPI Object Types and can be mapped to the appropriate 800xA Object properties for data synchronization.

Import/Export

The Import/Export tab page ([Figure 58](#)) is where the user may export data from the INtools/SPI database or import data from a CAEX file to update the INtools/SPI database.

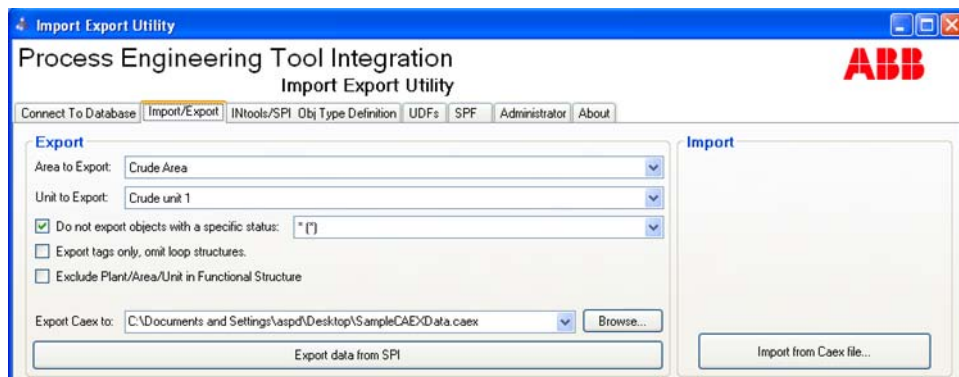


Figure 58. Import/Export Tab

Export Data

The steps for exporting data from the INtools/SPI database are as follow:

1. Select the area to export from the drop down list.
2. Select the unit to export from the drop down list.

3. To exclude tags that have certain attributes from the export file, check the Do Not Export Elements With A Specific Status and then select a specific status from the drop down list as shown in Figure 59.



Figure 59. INtools/SPI Object Status

4. Check the Export Tags Only, Omit Loop Structures to exclude loop information from the export file.
5. Check the Exclude Plant/Area/Unit In Functional Structure to exclude plant, area, and unit objects from the Functional Structure in the export file.
6. Browse for a file path and click on the Export Data From SPI bar.

Import Data

To import CAEX data, click on the Import From Caex File bar and enter the file path of the file.

Object Type Definitions

The Import/Export utility allows the user to configure the DB Import Export Utility to override default object-type and append new fields to the tags.



Using INtools/SPI Obj Definition feature in the product Import/Export Utility is an entirely optional step and should be used only if object type mapping is to be overridden on an instance basis and/or new fields are to be added to the list of properties available from INtools/SPI.

The user can select Instrument Type, System IO Type, Instrument Type + System IO Type (uses an '_' to concatenate the two properties), or Control System Tags type from the drop down list as shown in Figure 60. The fields are available in INtools/SPI and are typically filled in (by EPC engineer) during engineering. Depending upon the user choice, the Product creates the object types, which can then be mapped to the object types in 800xA.

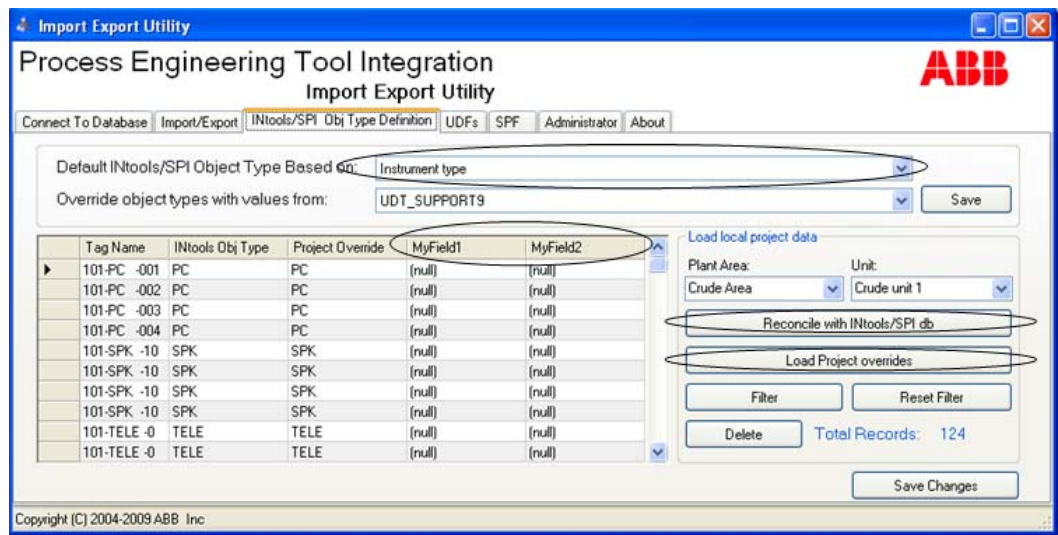


Figure 60. Object Type Definitions

- 7. A blank MS-Access database (PETI.mdb) is provided with the installation of the Product Import/Export utility. Users can append ad-hoc columns (MyField1, MyField2 - refer to Figure 61) to the table named CAEXobjectTypes within the database. The values in these columns for

respective tags appear as properties in the product (refer to [Figure 62](#)).



Do not modify the default columns provided within the table. The application will crash if any modifications are made.

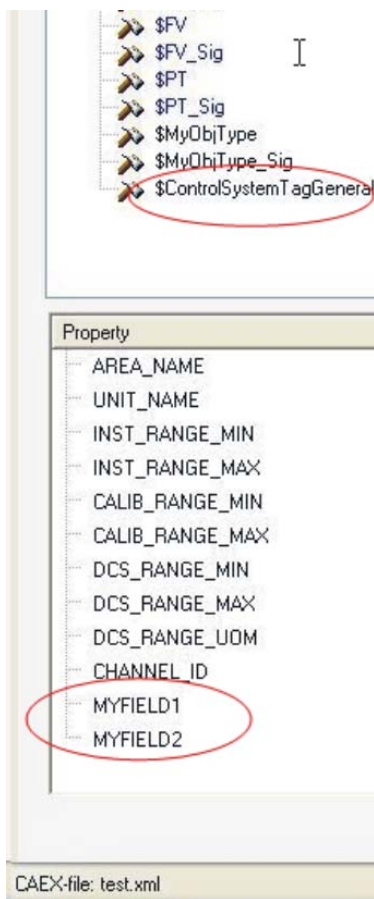


Figure 61. AdHoc Fields



Figure 62. Mapping In Process Engineering Tool Integration

8. Select the **Load Project Overrides** button (Figure 60) to load the MS-Access database PETI.mdb from the local hard-disk. There are no records initially.
9. Select the **Reconcile with INtools/SPI db** button to obtain tag names and their object types from INtools/SPI database as shown in Figure 60. If records already exist within the PETI.mdb database then the fresh data from

INtools/SPI is reconciled with the existing records. Total number of records are shown in the dialog.

User can filter the data by clicking on the **Filter** button. Click on **Reset filter** to view all data. User can edit only the Project Override field and any ad-hoc fields which are added to the table. Tag Name and INtools Obj Type columns are read-only and cannot be modified by the user. Click **Save changes** button to save any edited information.

Perform the following workflow:

1. If new properties that are missing in INtools/SPI are to be added then open:
C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process
Engineering Tool Integration\PETI.mdb file in MS-Access.
2. Open the CAEXObjectTypes table in design view.
3. Add new column(s) with their appropriate data-types to the list. See the warning above. Save the database and Exit MS-Access.
4. Start the product Import Export Utility.
5. Load the desired intools.ini file.
6. Go to the **INtools Obj Type Def** tab.
7. Select the desired object type basis from the drop-down list and save the configuration by clicking on the **Save** button next to the drop-down list.
8. Click on the **Load project overrides**.
9. Initially there will be no records.
10. Select the desired area and unit.
11. Click on the **Reconcile with INtools/SPI db** button. This will populate the data-grid.
12. Override object types for desired tags.
13. Enter information for new columns (if any) added in [Step 3](#).
14. Click on the **Save Changes** button to save the information.

Load to SPF or INtools/SPI, the publish flow is as follows:

CAEX file > Load from CAEX > Publish to SPF > XML file

Retrieve From SPF

To retrieve from SPF, perform the following:

1. Launch the PETI Import Export Utility.
2. From the PETI Import/Export Utility window, click the **SPF** tab.
3. Click the **Retriever from SPF** button in the bottom left corner.
4. From the Select Data XML File window, select a Data XML file.
5. Click **Open**.
6. From the Select Meta-data XML File window, select a Meta-data XML file.
7. Click **Open**.
8. From the Select Tombstones XML File window, click the **Cancel** button.

9. Click the **Close** button once the files have been retrieved successfully. Data will appear in the Current Data Set as shown in [Figure 64](#).

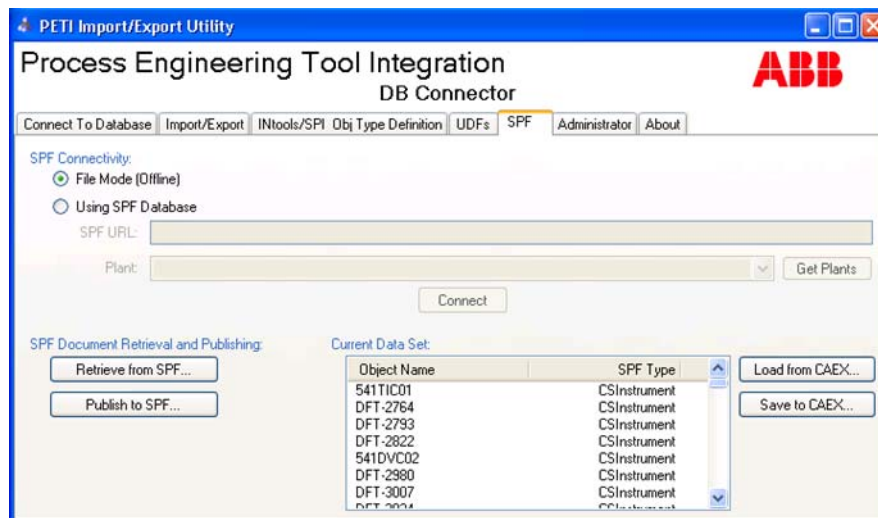


Figure 64. Current Data Set

10. Click the **Save to CAEX** button.

Publish to SPF

To publish to SPF, perform the following:

1. Launch the PETI Import Export Utility.
2. From the PETI Import/Export Utility window, click the **SPF** tab.
3. Click the **Load from CAEX** button to load data from CAEX file and publish into SPF. Data will appear in the Current Data Set as shown in [Figure 64](#).
4. Click the **Publish to SPF** button.

- From the Publish to File window, specify the path in the textbox in bottom left corner of the window to save file at designated location as shown in [Figure 65](#).

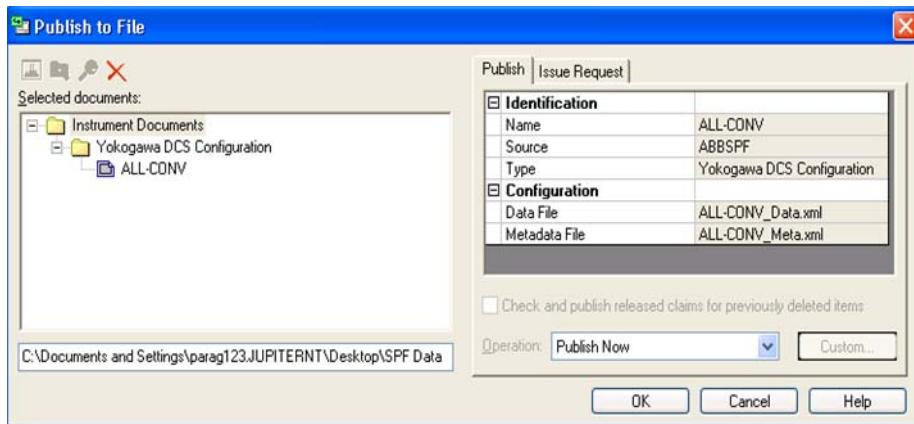


Figure 65. Publish to File Text Path



Sample files for SPI can be found in the following location: C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\Sample SPF.

Section 7 Command Line Interface

Introduction

To run the product in batch mode, an xml file that contains all the required information is needed. A sample file is shown as follows:

```
<?xml version="1.0"?>
<BulkTasks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" QuietMode="true" Version="1.0">
  <ImportDefinition PathToMappingFile="C:\User\WorkFiles\pcdeviceibmapping.dmf"
Workflow="CBM-Workflow">
    <ImportTasks StructureName="Control Structure" CompareAttributes="false" />
    <ImportTasks StructureName="Functional Structure" CompareAttributes="false" />
    <ImportTasks StructureName="Control Structure" CompareAttributes="true" />
    <ImportFiles PathToCaex="C:\User\WorkFiles\SampleCAEX.xml"
BasePathOfLogFile="C:\User\WorkFiles\SampleCAEXLog.xml">
        <Assignment ParentName="dummy">
            <TagName>tagname</TagName>
        </Assignment>
    </ImportFiles>
  </ImportDefinition>
</BulkTasks>
```

The product will use the map file as specified by the PathToMappingFile attribute and the CAEX file as specified by the PathToCaex attribute. The CompareAttributes attribute corresponds to the check box in the product labeled as include property comparison. The log file as specified by the BasePathOfLogfile attribute is currently not used.

The Workflow attribute sets the workflow for the product execution. Possible values are:

- CBM-WorkFlow.
- Pure CBM WorkFlow.
- FD WorkFlow.



If the CAEX file is in Function Designer Workflow format, then the Function Designer Workflow will be used and the Workflow attribute will be ignored. Control Builder Workflow and Pure CBM Workflow are interchangeable.

The product will execute Compare and Data Transfer for each of the structures as specified by the ImportTasks elements. In the above example, the product will process the Control Structure, then the Functional Structure, and finally the Control Structure again.

Although no user intervention is required, the Compare Progress and Transfer Progress dialogs will still appear. Therefore, the user may cancel a Compare or a Data Transfer as appropriate. The next action specified in the xml file will then commence. Canceling the entire batch process is currently not supported.

The command line to start executing the tasks that are specified in the xml file has the following syntax:

```
PETI [Filename | /b Filename |/?]
```

```
/b Filename
```

Start PETI to execute in batch mode the commands in the given file. Short form is '/b'. Long form is '/bulk'.

```
/Filename
```

Start PETI in interactive mode with the given CAEX file open

```
/?
```

Show the help information for the Command Line Interface

Some sample command lines are given as follow:

```
"C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool  
Integration\bin\peti" /b cli.xml
```

```
"C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool  
Integration\bin\peti" C:\User\WorkFiles\SampleCAEX.xml
```

```
"C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool  
Integration\bin\peti" /?
```



Sample files for Command Line Interface can be found in the following location:
C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool
Integration\XML\Sample Command Line Interface.

Appendix A Default Mapping File Configuration

Introduction

Provided in [Table 5](#) is the default mapping file configuration for Process Control Device Library (PCDeviceLib) 5.2 and INtools/SPI objects that comes with the installation of the product.

Table 5. Object Mapping

INtools/SPI Objects	800xA Object
AC 800M	AC 800M
AC800M_PM856	AC 800M
AC800M_PM856_Proc	TP830
AC800M_PM860	AC 800M
AC800M_PM860_Proc	TP830
AC800M_PM861	AC 800M
AC800M_PM861_Proc	TP830
AC800M_PM864	AC 800M
AC800M_PM864_Proc	TP830
AI	Transmitter
AI_Sig	CBM_AIS
AI810 on TU810	AI810
AI820 on TU810	AI820

Table 5. Object Mapping (Continued)

INtools/SPI Objects	800xA Object
AI830	AI830
AI835	AI835
AI845 on TU810	AI845
AI910N on TU921N	AI910* (AI4)
AI930N on TU921N	AI930* (AI4H A)
AO	ControlValvePneumatic
AO_Sig	PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP
AO_Signal	PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP
AO810 on TU810	AO810
AO820 on TU810	AO820
AO845 on TU810	AO845
CI801	CI801
CI830	CI830
CI851	CI851
CI854	CI854
Controller	ControllerPID
Controller_PM864 / TP830	TP830
Controller2	ControllerPID
ControlSystemTag_AO	ControlValvePneumatic
ControlSystemTag_DO	Valve
ControlSystemTag_HLAI	Transmitter
DI	DigitalInput

Table 5. Object Mapping (Continued)

INtools/SPI Objects	800xA Object
DI_Sig	CBM_DIS
DI810 on TU810	DI810
DI811	DI811
DI814	DI814
DI820 on TU810	DI820
DI821	DI821
DI830	DI830
DI831	DI831
DO	DigitalOutput
DO_Sig	CBM_DOS
DO810 on TU810	DO810
DO814	DO814
DO815	DO815
DO820 on TU810	DO820
DO821	DO821
DP820 on TU810	DP820
FC	ControllerPID
FD-FunctionalStructure	Allocatable Group
FDGenericClass	A4 Landscape
FE	Transmitter
FE_Sig	CBM_AIS
FI	Transmitter
FI_Sig	CBM_AIS

Table 5. Object Mapping (Continued)

INtools/SPI Objects	800xA Object
FIC	ControllerPID
FT	Transmitter
FT_HART	add(real)
FT_HART_Dev	ABB Generic HART Transmitter
FT_Sig	CBM_AIS
Function Diagram A3 Landscape	Function Diagram A3 Landscape
Function Diagram A3 Portrait	Function Diagram A3 Portrait
Function Diagram A4 Landscape	Function Diagram A4 Landscape
Function Diagram A4 Portrait	Function Diagram A4 Portrait
Function Diagram Legal Landscape	Function Diagram Legal Landscape
Function Diagram Legal Portrait	Function Diagram Legal Portrait
Function Diagram Letter Landscape	Function Diagram Letter Landscape
Function Diagram Letter Portrait	Function Diagram Letter Portrait
FV	ControlValvePneumatic
FV_Sig	CBM_AOS
FY	ControlValvePneumatic
FY_HART	add(real)
FY_HART_Dev	ABB Generic HART Actuator
FY_Sig	CBM_AOS

Table 5. Object Mapping (Continued)

INtools/SPI Objects	800xA Object
Generic	Product Class
HLAI	Transmitter
HLAI_Sig	CBM_AIS
HLAI_Signal	CBM_AIS
LC	ControllerPID
LI	Transmitter
LI_Sig	CBM_AIS
LIC	ControllerPID
LLAI	Transmitter
LLAI_Sig	CBM_AIS
Loop	ControllerPidLoop
LT	Transmitter
LT_Sig	CBM_AIS
LV	ControlValvePneumatic
LV_Sig	CBM_AOS
LY	ControlValvePneumatic
LY_Sig	CBM_AOS
M_PB	ABB_TFx12_PA
Module_AI810 on TU810	AI810
Panel_HOT_END_CTRL	AC 800M
PC	ControllerPID
PI	Transmitter
PI_Sig	CBM_AIS

Table 5. Object Mapping (Continued)

INtools/SPI Objects	800xA Object
PIC	ControllerPID
PIDLoop_Loop	ControllerPidLoop
Plant-Site	AC800M
PneuValve	ControlValvePneumatic
PneuValveSignal	PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_FB
PROC_	TP830
PROC_PM856 / TP830	TP830
PROC_PM860 / TP830	TP830
PROC_PM861 / TP830	TP830
PROC_PM864 / TP830	TP830
Project-Site	Control Network
PT	Transmitter
PT_Sig	CBM_AIS
PV	ControlValvePneumatic
PV_Sig	CBM_AOS
Root	Control Network
S900	S900
TC	ControllerPID
TE	Transmitter
TE_Sig	CBM_AIS
TI	Transmitter
TI_Sig	CBM_AIS

Table 5. Object Mapping (Continued)

INtools/SPI Objects	800xA Object
TIC	ControllerPID
Transmitter	Transmitter
TT	Transmitter
TT_Sig	CBM_AIS
TV	ControlValvePneumatic
TV_Sig	CBM_AOS
Unit	Control Program
Valve	ControlValvePneumatic

The attribute mapping is provided in [Table 6](#).

Table 6. Attribute Mapping

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
PneuValveSignal					
AC 800M	name		AC 800M	Name	Description
AC800M_PM856	name		AC 800M	Name	Name
AC800M_PM856_Proc	name		TP830	Name	Description
AC800M_PM860	name		AC 800M	Name	Name
AC800M_PM860_Proc	name		TP830	Name	Description
AC800M_PM861	name		AC 800M	Name	Name
AC800M_PM861_Proc	name		TP830	Name	Description

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
AC800M_PM864	name		AC 800M	Name	Name
AC800M_PM864_Proc	name		TP830	Name	Description
AI	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
AI	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
AI	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
AI	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
AI	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
AI	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
AI	SERVICE		Transmitter	Control Properties	Description
AI	TAGNAME		Transmitter	Control Properties	Name
AI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
AI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
AI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
AI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
AI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
AI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
AI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
AI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
AI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
AI810 on TU810	module_no		AI810	Name	Name
AI820 on TU810	module_no		AI820	Name	Name
AI830	module_no		AI830	Name	Name
AI835	module_no		AI835	Name	Name
AI845 on TU810	module_no		AI845	Name	Name
AI910N on TU921N	module_no		AI910* (AI4)	Name	Name
AI930N on TU921N	module_no		AI930* (AI4H A)	Name	Name
AO	TagName		ControlValvePneumatic	Name	Name
AO_Sig	CHANNEL_ID		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	ChannelNumber	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
AO_Sig	PROCESS-ALM-H		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelH
AO_Sig	PROCESS-ALM-HH		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelHH
AO_Sig	PROCESS-ALM-HHH		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelHHH
AO_Sig	PROCESS-ALM-L		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelL
AO_Sig	PROCESS-ALM-LL		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelLL
AO_Sig	PROCESS-ALM-LLL		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelLLL
AO_Sig	PROCESS-MAX		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	Max

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
AO_Sig	PROCESS-MIN		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	Min
AO_Signal	CHANNEL_ID		PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_SP	ChannelNumber	Name
AO_Signal	PROCESS-ALM-H		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelH
AO_Signal	PROCESS-ALM-HH		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelHH
AO_Signal	PROCESS-ALM-HHH		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelHHH
AO_Signal	PROCESS-ALM-L		PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelL
AO_Signal	PROCESS-ALM-LL		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelLL

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
AO_Signal	PROCESS-ALM-LLL		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	LevelLLL
AO_Signal	PROCESS-MAX		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	Max
AO_Signal	PROCESS-MIN		PCDevL_ControlValvePneuStdIO->PCDevL_ControlValvePneuStdIO_SP	CBM_SignalParameter	Min
AO810 on TU810	module_no		AO810	Name	Name
AO820 on TU810	module_no		AO820	Name	Name
AO845 on TU810	module_no		AO845	Name	Name
CI801	module_no		CI801	Name	Name
CI830	module_no		CI830	Name	Name
CI851	module_no		CI851	Name	Name
CI854	module_no		CI854	Name	Name
Controller	TagName		ControllerPID	Name	Name
Controller2	TagName		ControllerPID	Name	Name
ControlSystemTag_DO	TagName		Valve	Name	Name
DI	SERVICE		DigitalInput	Control Properties	Description
DI	TAGNAME		DigitalInput	Control Properties	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
DI_Sig	CHANNEL_ID		CBM_DIS	ChannelNumber	Name
DI_Sig	PROCESS-ALM-H		CBM_DIS	CBM_SignalParameter	LevelH
DI_Sig	PROCESS-ALM-HH		CBM_DIS	CBM_SignalParameter	LevelHH
DI_Sig	PROCESS-ALM-HHH		CBM_DIS	CBM_SignalParameter	LevelHHH
DI_Sig	PROCESS-ALM-L		CBM_DIS	CBM_SignalParameter	LevelL
DI_Sig	PROCESS-ALM-LL		CBM_DIS	CBM_SignalParameter	LevelLL
DI_Sig	PROCESS-ALM-LLL		CBM_DIS	CBM_SignalParameter	LevelLLL
DI_Sig	PROCESS-MAX		CBM_DIS	CBM_SignalParameter	Max
DI_Sig	PROCESS-MIN		CBM_DIS	CBM_SignalParameter	Min
DI810 on TU810	module_no		DI810	Name	Name
DI811	module_no		DI811	Name	Name
DI814	module_no		DI814	Name	Name
DI820 on TU810	module_no		DI820	Name	Name
DI821	module_no		DI821	Name	Name
DI830	module_no		DI830	Name	Name
DI831	module_no		DI831	Name	Name
DO	TagName		DigitalOutput	Name	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
DO_Sig	CHANNEL_ID		CBM_DOS	ChannelNumber	Name
DO_Sig	PROCESS-ALM-H		CBM_DOS	CBM_SignalParameter	LevelH
DO_Sig	PROCESS-ALM-HH		CBM_DOS	CBM_SignalParameter	LevelHH
DO_Sig	PROCESS-ALM-HHH		CBM_DOS	CBM_SignalParameter	LevelHHH
DO_Sig	PROCESS-ALM-L		CBM_DOS	CBM_SignalParameter	LevelL
DO_Sig	PROCESS-ALM-LL		CBM_DOS	CBM_SignalParameter	LevelLL
DO_Sig	PROCESS-ALM-LLL		CBM_DOS	CBM_SignalParameter	LevelLLL
DO_Sig	PROCESS-MAX		CBM_DOS	CBM_SignalParameter	Max
DO_Sig	PROCESS-MIN		CBM_DOS	CBM_SignalParameter	Min
DO810 on TU810	module_no		DO810	Name	Name
DO814	module_no		DO814	Name	Name
DO815	module_no		DO815	Name	Name
DO820 on TU810	module_no		DO820	Name	Name
DO821	module_no		DO821	Name	Name
DP820 on TU810	module_no		DP820	Name	Name
FC	SERVICE		ControllerPID	Name	Description
FC	TAGNAME		ControllerPID	Name	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
FD-FunctionalStructure	name		Allocatable Group	Name	Name
FDGenericClass	name		A4 Landscape	Name	Description
FE	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
FE	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
FE	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
FE	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
FE	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
FE	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
FE	SERVICE		Transmitter	Control Properties	Description
FE	TAGNAME		Transmitter	Control Properties	Name
FE_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
FE_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
FE_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
FE_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
FE_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
FE_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
FE_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
FE_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
FE_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
FI	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
FI	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
FI	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
FI	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
FI	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
FI	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
FI	SERVICE		Transmitter	Control Properties	Description

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
FI	TAGNAME		Transmitter	Control Properties	Name
FI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
FI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
FI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
FI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
FI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
FI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
FI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
FI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
FI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
FIC	SERVICE		ControllerPID	Control Properties	Description
FIC	TAGNAME		ControllerPID	Control Properties	Name
FT	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
FT	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
FT	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
FT	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
FT	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
FT	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
FT	SERVICE		Transmitter	Control Properties	Description
FT	TAGNAME		Transmitter	Control Properties	Name
FT_HART	TAGNAME		add(real)	Name	Description
FT_HART_Dev	CHANNEL_ID		ABB Generic HART Transmitter	Fieldbus Management	ParentChannel
FT_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
FT_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
FT_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
FT_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
FT_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
FT_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
FT_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
FT_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
FT_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
Function Diagram A3 Landscape	name		Function Diagram A3 Landscape	Name	Name
Function Diagram A3 Portrait	name		Function Diagram A3 Portrait	Name	Name
Function Diagram A4 Landscape	name		Function Diagram A4 Landscape	Name	Name
Function Diagram A4 Portrait	name		Function Diagram A4 Portrait	Name	Name
Function Diagram Legal Landscape	name		Function Diagram Legal Landscape	Name	Name
Function Diagram Legal Portrait	name		Function Diagram Legal Portrait	Name	Name
Function Diagram Letter Landscape	name		Function Diagram Letter Landscape	Name	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
Function Diagram Letter Portrait	name		Function Diagram Letter Portrait	Name	Name
FV	SERVICE		ControlValvePneumatic	Control Properties	Description
FV	TAGNAME		ControlValvePneumatic	Control Properties	Name
FV_Sig	CHANNEL_ID		CBM_AOS	ChannelNumber	Name
FV_Sig	PROCESS-ALM-H		CBM_AOS	CBM_SignalParameter	LevelH
FV_Sig	PROCESS-ALM-HH		CBM_AOS	CBM_SignalParameter	LevelHH
FV_Sig	PROCESS-ALM-HHH		CBM_AOS	CBM_SignalParameter	LevelHHH
FV_Sig	PROCESS-ALM-L		CBM_AOS	CBM_SignalParameter	LevelL
FV_Sig	PROCESS-ALM-LL		CBM_AOS	CBM_SignalParameter	LevelLL
FV_Sig	PROCESS-ALM-LLL		CBM_AOS	CBM_SignalParameter	LevelLLL
FV_Sig	PROCESS-MAX		CBM_AOS	CBM_SignalParameter	Max
FV_Sig	PROCESS-MIN		CBM_AOS	CBM_SignalParameter	Min
FY	SERVICE		ControlValvePneumatic	Control Properties	Description

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
FY	TAGNAME		ControlValvePneumatic	Control Properties	Name
FY_HART	TAGNAME		add(real)	Name	Description
FY_HART_Dev	CHANNEL_ID		ABB Generic HART Actuator	Fieldbus Management	ParentChannel
FY_Sig	PROCESS-ALM-H		CBM_AOS	CBM_SignalParameter	LevelH
FY_Sig	PROCESS-ALM-HH		CBM_AOS	CBM_SignalParameter	LevelHH
FY_Sig	PROCESS-ALM-HHH		CBM_AOS	CBM_SignalParameter	LevelHHH
FY_Sig	PROCESS-ALM-L		CBM_AOS	CBM_SignalParameter	LevelL
FY_Sig	PROCESS-ALM-LL		CBM_AOS	CBM_SignalParameter	LevelLL
FY_Sig	PROCESS-ALM-LLL		CBM_AOS	CBM_SignalParameter	LevelLLL
FY_Sig	PROCESS-MAX		CBM_AOS	CBM_SignalParameter	Max
FY_Sig	PROCESS-MIN		CBM_AOS	CBM_SignalParameter	Min
Generic	name		Product Class	Name	Name
HLAI	TagName		Transmitter	Name	Name
HLAI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
HLAI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
HLAI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
HLAI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
HLAI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
HLAI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
HLAI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
HLAI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
HLAI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
HLAI_Signal	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
HLAI_Signal	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
HLAI_Signal	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
HLAI_Signal	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
HLAI_Signal	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
HLAI_Signal	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
HLAI_Signal	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
HLAI_Signal	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
HLAI_Signal	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
LC	SERVICE		ControllerPID	Name	Description
LC	TAGNAME		ControllerPID	Name	Name
LI	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
LI	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
LI	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
LI	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
LI	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
LI	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
LI	SERVICE		Transmitter	Control Properties	Description
LI	TAGNAME		Transmitter	Control Properties	Name
LI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
LI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
LI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
LI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
LI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
LI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
LI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
LI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
LI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
LIC	SERVICE		ControllerPID	Control Properties	Description
LIC	TAGNAME		ControllerPID	Control Properties	Name
LLAI	TagName		Transmitter	Name	Name
LLAI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
LLAI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
LLAI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
LLAI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
LLAI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
LLAI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
LLAI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
LLAI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
LLAI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
LT	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
LT	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
LT	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
LT	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
LT	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
LT	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
LT	SERVICE		Transmitter	Control Properties	Description

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
LT	TAGNAME		Transmitter	Control Properties	Name
LT_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
LT_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
LT_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
LT_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
LT_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
LT_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
LT_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
LT_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
LT_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
LV	SERVICE		ControlValvePneumatic	Control Properties	Description
LV	TAGNAME		ControlValvePneumatic	Control Properties	Name
LV_Sig	CHANNEL_ID		CBM_AOS	ChannelNumber	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
LV_Sig	PROCESS-ALM-H		CBM_AOS	CBM_SignalParameter	LevelH
LV_Sig	PROCESS-ALM-HH		CBM_AOS	CBM_SignalParameter	LevelHH
LV_Sig	PROCESS-ALM-HHH		CBM_AOS	CBM_SignalParameter	LevelHHH
LV_Sig	PROCESS-ALM-L		CBM_AOS	CBM_SignalParameter	LevelL
LV_Sig	PROCESS-ALM-LL		CBM_AOS	CBM_SignalParameter	LevelLL
LV_Sig	PROCESS-ALM-LLL		CBM_AOS	CBM_SignalParameter	LevelLLL
LV_Sig	PROCESS-MAX		CBM_AOS	CBM_SignalParameter	Max
LV_Sig	PROCESS-MIN		CBM_AOS	CBM_SignalParameter	Min
LY	SERVICE		ControlValvePneumatic	Control Properties	Description
LY	TAGNAME		ControlValvePneumatic	Control Properties	Name
LY_Sig	PROCESS-ALM-H		CBM_AOS	CBM_SignalParameter	LevelH
LY_Sig	PROCESS-ALM-HH		CBM_AOS	CBM_SignalParameter	LevelHH
LY_Sig	PROCESS-ALM-HHH		CBM_AOS	CBM_SignalParameter	LevelHHH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
LY_Sig	PROCESS-ALM-L		CBM_AOS	CBM_SignalParameter	LevelL
LY_Sig	PROCESS-ALM-LL		CBM_AOS	CBM_SignalParameter	LevelLL
LY_Sig	PROCESS-ALM-LLL		CBM_AOS	CBM_SignalParameter	LevelLLL
LY_Sig	PROCESS-MAX		CBM_AOS	CBM_SignalParameter	Max
LY_Sig	PROCESS-MIN		CBM_AOS	CBM_SignalParameter	Min
M_PB	TAGNAME		ABB_TFx12_PA	Name	Description
PC	SERVICE		ControllerPID	Name	Description
PC	TAGNAME		ControllerPID	Name	Name
PI	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
PI	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
PI	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
PI	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
PI	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
PI	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
PI	SERVICE		Transmitter	Control Properties	Description

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
PI	TAGNAME		Transmitter	Control Properties	Name
PI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
PI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
PI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
PI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
PI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
PI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
PI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
PI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
PI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
PIC	SERVICE		ControllerPID	Control Properties	Description
PIC	TAGNAME		ControllerPID	Control Properties	Name
PIDLoop_Loop	name		ControllerPidLoop	Name	Name
Plant-Site	name		AC800M	Name	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
PneuValveSignal	channelid		PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_FB	ChannelNumber	Name
PneuValveSignal	PROCESS-ALM-L		PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_FB	CBM_SignalParameter	LevelL
PneuValveSignal	PROCESS-ALM-LL		PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_FB	CBM_SignalParameter	LevelLL
PneuValveSignal	PROCESS-ALM-LLL		PCDL_ControlValvePneuStdIO->PCDL_ControlValvePneuStdIO_FB	CBM_SignalParameter	LevelLLL
PROC_	name		TP830	Name	Description
PROC_PM856 / TP830	name		TP830	Name	Description
PROC_PM860 / TP830	name		TP830	Name	Description
PROC_PM861 / TP830	name		TP830	Name	Description
PROC_PM864 / TP830	name		TP830	Name	Description
Project-Site	name		Control Network	Name	Name
PT	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
PT	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
PT	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
PT	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
PT	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
PT	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
PT	SERVICE		Transmitter	Control Properties	Description
PT	TAGNAME		Transmitter	Control Properties	Name
PT_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
PT_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
PT_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
PT_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
PT_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
PT_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
PT_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
PT_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
PT_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
PV	SERVICE		ControlValvePneumatic	Control Properties	Description
PV	TAGNAME		ControlValvePneumatic	Control Properties	Name
PV_Sig	CHANNEL_ID		CBM_AOS	ChannelNumber	Name
PV_Sig	PROCESS-ALM-H		CBM_AOS	CBM_SignalParameter	LevelH
PV_Sig	PROCESS-ALM-HH		CBM_AOS	CBM_SignalParameter	LevelHH
PV_Sig	PROCESS-ALM-HHH		CBM_AOS	CBM_SignalParameter	LevelHHH
PV_Sig	PROCESS-ALM-L		CBM_AOS	CBM_SignalParameter	LevelL
PV_Sig	PROCESS-ALM-LL		CBM_AOS	CBM_SignalParameter	LevelLL
PV_Sig	PROCESS-ALM-LLL		CBM_AOS	CBM_SignalParameter	LevelLLL
PV_Sig	PROCESS-MAX		CBM_AOS	CBM_SignalParameter	Max

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
PV_Sig	PROCESS-MIN		CBM_AOS	CBM_SignalParameter	Min
S900	module_no		S900	Name	Name
TC	SERVICE		ControllerPID	Name	Description
TC	TAGNAME		ControllerPID	Name	Name
TE	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
TE	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
TE	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
TE	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
TE	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
TE	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
TE	SERVICE		Transmitter	Control Properties	Description
TE	TAGNAME		Transmitter	Control Properties	Name
TE_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
TE_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
TE_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
TE_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
TE_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
TE_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
TE_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
TE_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
TE_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
TI	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
TI	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
TI	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
TI	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
TI	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
TI	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
TI	SERVICE		Transmitter	Control Properties	Description

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
TI	TAGNAME		Transmitter	Control Properties	Name
TI_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
TI_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
TI_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
TI_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
TI_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL
TI_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
TI_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
TI_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
TI_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
TIC	SERVICE		ControllerPID	Control Properties	Description
TIC	TAGNAME		ControllerPID	Control Properties	Name
Transmitter	PROCESS-MAX		Transmitter	Control Module	IO.Parameters.Max
Transmitter	TagName		Transmitter	Name	Name

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
TT	PROCESS-ALM-H		Transmitter	Control Properties	AELevelH
TT	PROCESS-ALM-HH		Transmitter	Control Properties	AELevelHH
TT	PROCESS-ALM-HHH		Transmitter	Control Properties	AELevelHHH
TT	PROCESS-ALM-L		Transmitter	Control Properties	AELevelL
TT	PROCESS-ALM-LL		Transmitter	Control Properties	AELevelLL
TT	PROCESS-ALM-LLL		Transmitter	Control Properties	AELevelLLL
TT	SERVICE		Transmitter	Control Properties	Description
TT	TAGNAME		Transmitter	Control Properties	Name
TT_Sig	CHANNEL_ID		CBM_AIS	ChannelNumber	Name
TT_Sig	PROCESS-ALM-H		CBM_AIS	CBM_SignalParameter	LevelH
TT_Sig	PROCESS-ALM-HH		CBM_AIS	CBM_SignalParameter	LevelHH
TT_Sig	PROCESS-ALM-HHH		CBM_AIS	CBM_SignalParameter	LevelHHH
TT_Sig	PROCESS-ALM-L		CBM_AIS	CBM_SignalParameter	LevelL

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
TT_Sig	PROCESS-ALM-LL		CBM_AIS	CBM_SignalParameter	LevelLL
TT_Sig	PROCESS-ALM-LLL		CBM_AIS	CBM_SignalParameter	LevelLLL
TT_Sig	PROCESS-MAX		CBM_AIS	CBM_SignalParameter	Max
TT_Sig	PROCESS-MIN		CBM_AIS	CBM_SignalParameter	Min
TV	SERVICE		ControlValvePneumatic	Control Properties	Description
TV	TAGNAME		ControlValvePneumatic	Control Properties	Name
TV_Sig	CHANNEL_ID		CBM_AOS	ChannelNumber	Name
TV_Sig	PROCESS-ALM-H		CBM_AOS	CBM_SignalParameter	LevelH
TV_Sig	PROCESS-ALM-HH		CBM_AOS	CBM_SignalParameter	LevelHH
TV_Sig	PROCESS-ALM-HHH		CBM_AOS	CBM_SignalParameter	LevelHHH
TV_Sig	PROCESS-ALM-L		CBM_AOS	CBM_SignalParameter	LevelL
TV_Sig	PROCESS-ALM-LL		CBM_AOS	CBM_SignalParameter	LevelLL
TV_Sig	PROCESS-ALM-LLL		CBM_AOS	CBM_SignalParameter	LevelLLL

Table 6. Attribute Mapping (Continued)

INtools/SPI Object	INtools/SPI Attributes	Dir.	800xA Object	800xA Aspect Name	800xA Attribute
TV_Sig	PROCESS-MAX		CBM_AOS	CBM_SignalParameter	Max
TV_Sig	PROCESS-MIN		CBM_AOS	CBM_SignalParameter	Min
Unit	name		Control Program	Name	Name
Valve	TagName		ControlValvePneumatic	Name	Name



Foundation Fieldbus Workflow creates Foundation Fieldbus objects in 800xA from CAEX file that is generated by CDM.

Appendix B Configuring ABB Hardware in INtools/SPI

Controller Configuration

To configure the ABB controllers, perform the following (refer to [Figure 66](#) for more information):

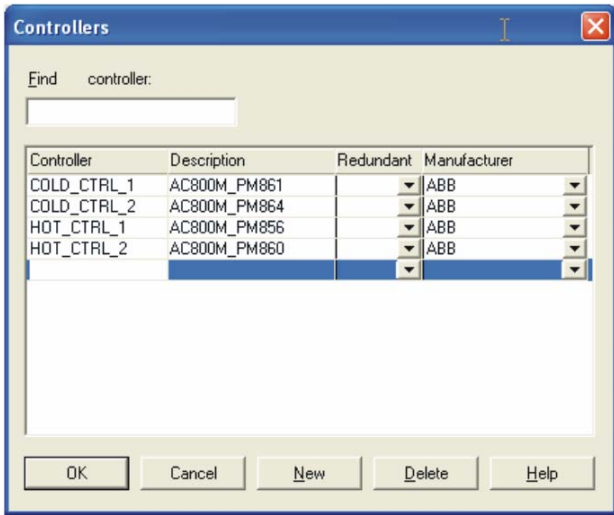


Figure 66. Find Controllers

1. Enter the Controller name in the **Controller** column.
2. Enter the type of controller under the description column. The product will use this field to determine the INtools/SPI class type of the controller. Examples are shown in [Figure 66](#) to distinguish between the different processor types for AC 800M controllers.

3. Make sure to select the **Manufacturer** for the controller.
4. The <controller description>_PROC (AC800M_PM864_PROC) type processor will be created by the Product.



Currently, only one type of processor will be created for controller. This can be either mapped to processor for S800 or communication box for S900 but not both. In case, a controller has both S800 & S900 IO cards the product will automatically create only one type. The other will have to be manually dragged (and dropped) in expert mode to the proper hierarchical location within the 800xA tree view in the product.

Configuring ABB Cards/Strips

1. Select the appropriate ABB controller from the drop-down list under the **Controller/Processor** list as shown in [Figure 67](#).

Figure 67. I/O Card Properties - ABB Controller

- Specify the position of the card in the controller under the **Module** field as shown in Figure 68.

The screenshot shows the 'I/O Card Properties' dialog box with the 'General' tab selected. The 'I/O card' field contains 'AI820'. The 'Panel' field contains 'CRACKING'. In the 'Control system details' section, 'System I/O type' is set to 'HLA' and 'Module' is set to '01', which is circled. 'Controller / Processor' is set to 'HOT_CTRL1'. There are checkboxes for 'Set within a distant cabinet' and 'Define a redundant I/O'. Below, there are sections for 'Primary location' and 'Secondary location', each with fields for 'Cabinet', 'Rack', 'Position', and 'System cable'. The 'Primary location' fields are filled with 'CRACKING', '*', and '*' respectively. At the bottom are buttons for 'OK', 'Cancel', 'Revisions...', 'Delete', and 'Help'.

Figure 68. I/O Card Properties Module

- For I/O cards connected to Profibus S800 and S900 communication interface cards, specify the position of the I/O card in the communication interfaces under the Series field as *master card type*/**<master card position>**, *slave card type*/**<slave card position>**.

For example, CI854/2, CI801/3 as shown in Figure 69. Together with Figure 68, the sample I/O card will be connected to position 1 of a Profibus slave card of type CI801, which in turn is connected to position 3 of a Profibus master card of type CI854, which is then connected to position 2 of controller HOT_CTRL1. The hierarchy in 800xA will be:

Controller > 2, CI854 > 3, CI801 > 1, AI820

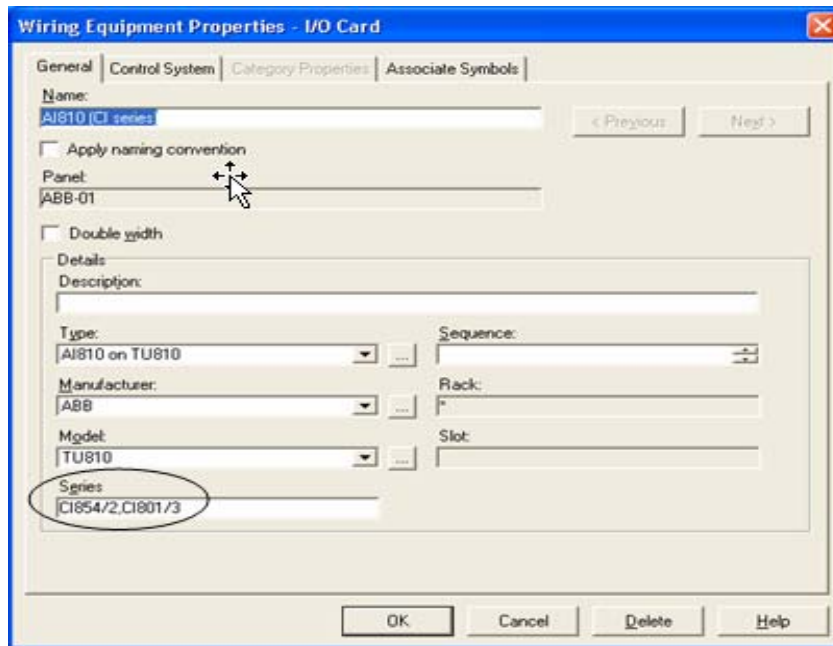


Figure 69. I/O Card Properties Module

If the Series field is blank or contains illegal data, the I/O card will be connected to the Modulebus. The hierarchy in 800xA will then be:

Controller > 0, (processor type) > 11, ModuleBus > 1, AI820.

Configuring Profibus Devices

In 800xA, Profibus objects are created only in the Control Structure and only under an appropriate master Profibus card such as CI851 and CI854. Other Profibus cards such as CI801 and CI830 are slave cards and cannot be used for Profibus device connections.

Although there are two types of Profibus devices (ProfibusDP and ProfibusPA as configured in INtools/SPI) the difference between them is transparent to 800xA. Therefore, only ProfibusDP is currently supported by the product.

In INtools/SPI, the I/O Type of a Profibus DP device is ProfibusDP. If the corresponding Instrument Type is VFD, then it is a virtual tag of a Profibus DP device. Virtual tags are currently not supported by the product.

To configure Profibus DP devices in INtools/SPI, perform the following steps:

1. In the Fieldbus Segment Manager, create a DP segment and name the segment as follows: *Profibus master card type*/**<position of Profibus master card in a controller>**/**<name of the controller>**. For example, CI854/3/HOT_CTRL1 as shown in Figure 70.

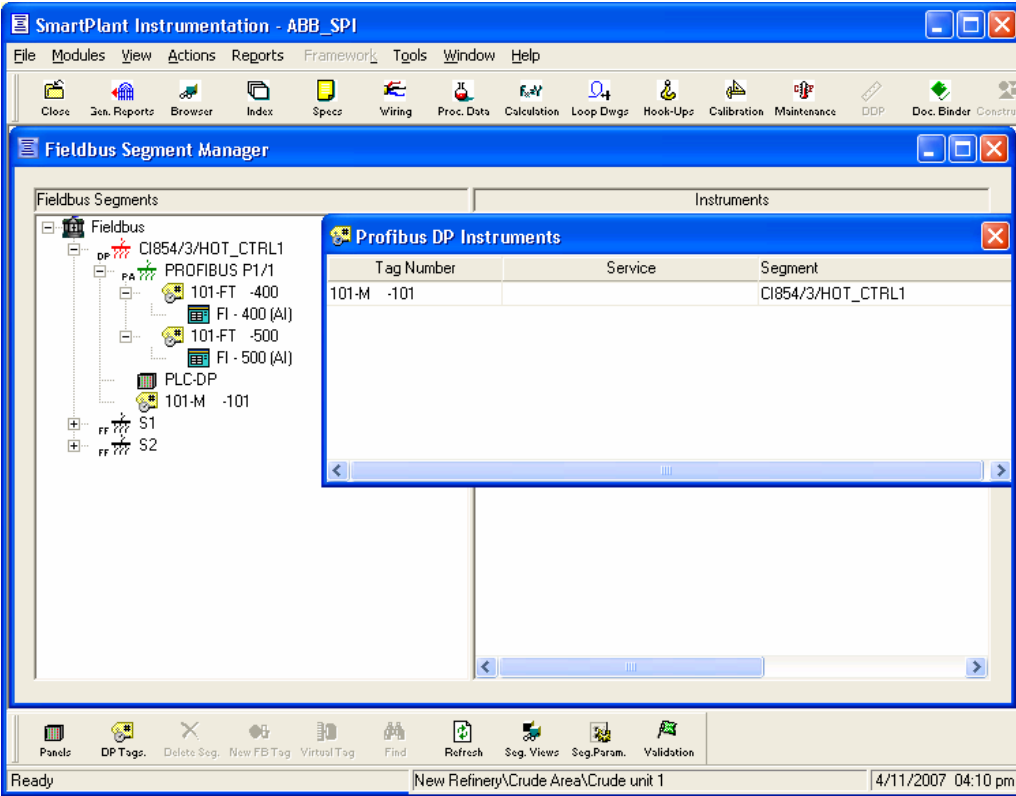
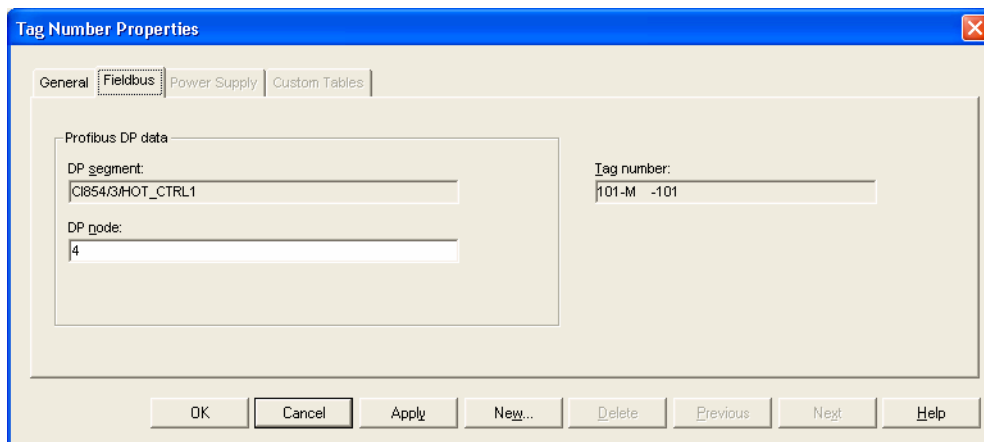


Figure 70. Profibus Segment Manager

2. Drag and drop the Profibus device from Profibus DP Instrument onto the segment as shown in [Figure 70](#).
3. Specify the position of the Profibus device in the Profibus master cards as shown in [Figure 71](#).



The screenshot shows the 'Tag Number Properties' dialog box with the 'Fieldbus' tab selected. The 'Profibus DP data' section contains the following fields:

- DP segment: CI854/3/HOT_CTRL1
- DP node: 4
- Tag number: I101-M -101

At the bottom of the dialog box, there are buttons for OK, Cancel, Apply, New..., Delete, Previous, Next, and Help.

Figure 71. Fieldbus Device Properties

Configuring HART Devices

In INtools/SPI, the I/O Type of a HART device is either HART AI or HART AO. If the corresponding Instrument Type is VFD, then it is a virtual tag of a HART device. Virtual tags are currently not supported by the product.

The configuration of a HART device for 800xA is similar to that of a conventional tag. The HART device is allocated to a channel of an appropriate I/O card. The I/O card, as discussed above, will have sufficient information to determine the position of the HART device in the 800xA hierarchy.



Sample files that include Profibus and Hart objects can be found in the following location: C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool Integration\XML\Sample Profibus Hart.

Creation of FF Template

Follow the steps to create a FF template in Control Structure:

1. In Engineering Workplace, create the following FF objects in Control Structure and provide appropriate names:

FF Object Type	Sample Object Name
HSE Subnet	FF_Template
ABB LD800HSE, HSE Host CI860	HSE_DEV_001, HSE_Host_001
H1 Link	FF_H1_L1_001
H1 Schedule	FF_SCHED_001
FF Application	1xAI

2. Select the created FF Application object from the Control Structure.
3. Select FF Management aspect.
4. Click **Open Project** to launch **FF Builder**.
5. In FF Builder right-click **HSESubnet** Object and select **Reserve nodes recursively**.
6. Select an H1 Link object from the tree view and add the required FF device type.



For more information on FF Device configuration refer *System 800xA, Device Management Device Library Wizard* (2PAA102573*).

7. Select FF application and click the Link view tab placed at the bottom of the FF Builder.
8. Drag and drop the required **input / output** blocks (e.g AI / AO) into the graphics area from link view tab. Assign signals to **input / output** (e.g AI) blocks as displayed in [Figure 72](#).

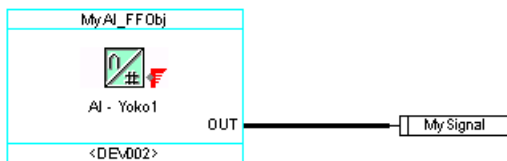


Figure 72. Assigning Signal

9. Right-click **AI** objects, go to Properties -> Enter a valid name (e.g. **MyAI_FFObj**).

10. Click **Save** and close the FF Builder.

FF Template is created in control structure as shown in Figure 73.

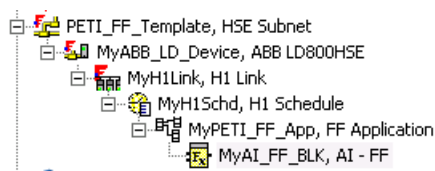


Figure 73. FF Template

For more information on FF builder, refer to *System 800xA, Device Management FOUNDATION Fieldbus Configuration* (3BDD0129*).



Users can import sample **AFW** file containing FF objects available at:
 C:\Program Files(x86)\ABB Industrial IT\Engineer IT\Process Engineering Tool
 Integration\XML\Sample FF.

Numerics

800xA 9, 19, 41

A

AREAS and UNITS 14

Attribute

Mapping 41, 109

C

CAEX 81

Deleting nodes 83

Inserting nodes 82

Modifying node attributes 83

Moving nodes 83

Running the editor 81

CAEX file 14, 85

CAEX tree editor 81

Command line 101

Command line interface 101

Comparison 59

Configuration

Controllers 141

Default mapping file configuration 103

HART devices 146

Profibus devices 144

Configuring HART devices 146

Configuring profibus devices 144

Control builder workflow 22

Controller configuration 141

Controllers 141

Core functions 9

Product overview 9

Creating a Control builder workflow 22

Creating a function diagram workflow 32

D

Data comparison 59

Data sources

Database 14

File 14

Data transfer

ExpertSync mode 67

ExpressSync mode 64

Database data source 14

Default mapping file 103

Delete

Mapping 45

Deleting nodes in the CAEX tree editor 83

E

Executable file 13

Export/import 85

F

File

CAEX 85

Mapping 103

PETI executables 85

File database source 14

Function designer 50

Function designer template 50

Function diagram workflow 32

Creating a function diagram 32

G

Getting started 13

Data sources 14

Executable file 13

Licensing 13

Main options 13

H

HART 146
HTML file
 PETIlog 15

I

Icons
 800xA to INtools/SPI 44
 Blocked 44
 INtools/SPI to 800xA 44
 Unknown 44
Import/export 85
Import/export utility 92
Inserting nodes in the CAEX tree editor 82
Interface 101
INtools/SPI 9
INtools/SPI obj type definition tab 89

L

Licensing 20

M

Main options
 Data transfer 13
 Mapping 13
Mapping
 Attribute 41, 109
 Database 42
 Delete 45
 File 42
 Object 41, 103
 Overview 41
Mapping attribute 109
Mapping file 103
Mapping objects 103
Mapping operation 42
Memory usage 13
Modifying node attributes in the CAEX tree editor 83
Moving nodes in the CAEX tree editor 83

O

Object
 Mapping 41
Object mapping 103
Objects
 Mapping 103
Operation
 Mapping 42
 Transfer data 57
Overview 9
 800xA 9
 INtools/SPI 9

P

PCDeviceLib workflow
 Creating a PCDeviceLib 22
 Synchronizing control structure 23
 Synchronizing functional structure 23
PLANT, AREA, UNIT 15
Product overview 9
 Core functions 9
Profibus 144
Pure CB Workflow 23
Pure CB workflow
 Mapping information 28
 Variable 24

R

Running the CAEX tree editor 81

S

Security 19 to 20
 User 20
Standalone application
 Functionality 85

T

Template 50
Transfer data operation 57

U

UNITS and AREAS 14

User security 20

Utility

 Import/export 92

V

Variable 24

W

Workflow 21

 Control builder 22

 Function diagram 21, 32

 PCDeviceLib 21

 Pure CB 23

Contact us

www.abb.com/800xA
www.abb.com/controlsystems

Copyright© 2014 ABB.
All rights reserved.

3BUA000184-600

Power and productivity
for a better world™

