TEAMCENTER

Embedded Software Solutions

Teamcenter 2412

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow’s idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: **support.sw.siemens.com**

Send Feedback on Documentation: **support.sw.siemens.com/doc_feedback_form**

Contents

Managing Calibration and Configuration Data

Managing the embedded software implementation process

Managing ClearCase Integration

Using binary management

Managing control units

Customizing solutions and exchanging software data

1. Getting started with embedded software solutions

Overview of getting started with embedded software solutions

Embedded Software Solutions comprise different solutions that help to manage functions, product electronic architecture, signals, dependencies, calibration and configuration-related parameter data, embedded software design data, binaries, and control units in My Teamcenter and Structure Manager in the rich client.

Using Embedded Software Solutions, you can view and manage the dependencies between control units, control unit-to-software, and software-to-software. You can release software parts (for example, binaries), perform impact analysis, manage changes, and make the software available for downloading to external applications that flash (write) embedded software onto the control unit.

Embedded Software Solutions allow you to manage the calibration and configuration-related parameter data of embedded systems into enterprise dictionaries. You can define, create, view, update, and delete parameter data. It allows you to group related parameter definitions and associate parameter values to a project. After values are associated, you can configure them using options and features applicable to the project.

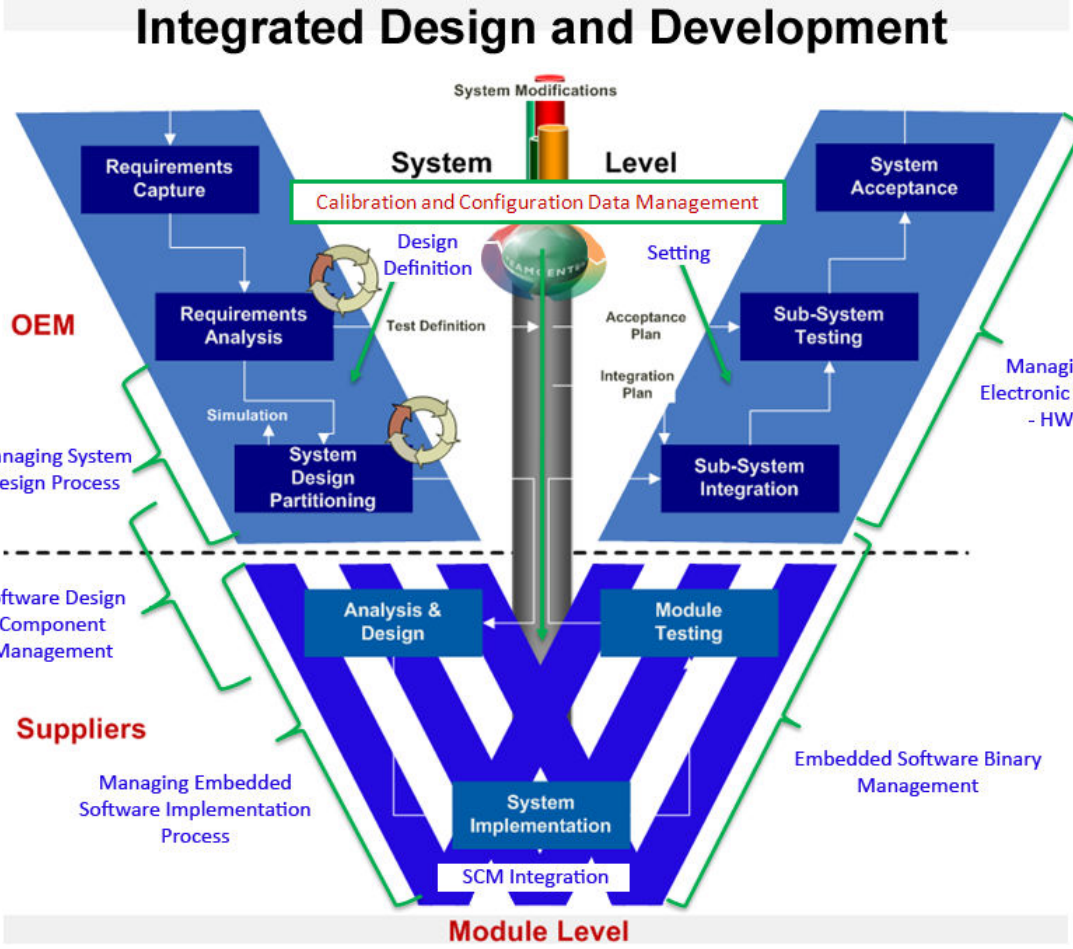
With Embedded Software Solutions, you can also manage design data-related embedded software components and capture the dependencies between the design components. A repository of reusable embedded software components can be managed within Teamcenter. You can also capture traceability from the embedded software binary to the actual source code. You can manage source code data in SCM systems, using the Teamcenter ClearCase Integration solution.

Embedded software development process

Embedded Software Solutions address the rapid expansion of embedded software in the industry. They effectively manage the software development process by integrating it into the whole product life cycle, which includes mechanical, electronic, software, and control system components. This approach helps to accelerate product introduction, lower cost, and improve quality.

Embedded Software Solutions provide a common data model and out-of-the-box manageable objects. The solutions help to specify product structures combining electronic, software, and electrical components and allow tracking interdependencies among the parts composing a system. In addition to helping in integration, the solutions also help to easily configure products and track and manage software changes. This helps determine how the embedded software is used in different variants of a product.

The various solutions involved in the embedded software development process are shown in the following figure.



Embedded Software Solutions

This section provides a brief overview of the various Embedded Software Solutions:

- Systems Engineering and Requirements Management

  Use Systems Engineering and Requirements Management to manage requirements related to the embedded systems. It simplifies requirement development and access. With Systems Engineering and Requirements Management, you can:

  - Identify and develop the requirements of the project at inception.
  - Associate requirements with the product design at an early stage and validate those requirements against the design.
  - Maintain requirements as separate items with specific properties.
  - Establish traceability of requirements to various artifacts of the product.

- Functional design management

  Use this solution to create functions, interfaces of functions, decompose functions, and manage communication between them.

- Product electronic architecture management

  Use this solution to create the overall product electronic architecture. This includes identifying and creating control units, managing messaging protocols and networks, processors, signals, messages, and defining dependencies between the control units.

- Functional distribution

  Use this solution to allocate functions from the functional model to the control units in the product electronic architecture.

- Signal management

  Use this solution to manage communication information (signals and messages) and manage networks within a product's electronic architecture. Signal management also helps you to identify dependencies between control units that participate in the electronic architecture of a given product.

- Dependency management

  Use this solution to specify and manage dependencies between system parts and assess the impact of changes.

- Calibration and Configuration Data Management

  Use this solution to manage calibration and configuration data of embedded software by creating parameter definitions and setting values for parameters.

- Embedded Software Design Data Management

  Use this solution to manage design data of embedded software components, define dependencies, and associate specifications, source code, models, and test cases to software design data component.

- Embedded Software Binary management

  Use this solution to create binary software revisions, establish traceability, and upload data into Teamcenter while making it available for flashing.

- Control unit management

  Use this solution to create control units, associate them to product configuration, and create compatibility reports.

Industry solution matrix

Embedded Software Solutions can be used in various industries and each industry can use the solutions relevant to it in its design, manufacturing, or production process. The following figure represents some industry verticals and how they may use the different solutions.

Capability ▼	Industry			
	High-Tech	Machinery	Aerospace	Automotive
Managing the system design process				
Functional design management	■	■	■	■
Product electronic architecture management	■	■	■	■
Function distribution across control units	■	■	■	■
Signal management	■	■	■	■
Dependency management	■	■	■	■
Managing the embedded software implementation process				
Create software design component	■	■	■	■
Associate source code to software design component	■	■	■	■
Associate reference data to software design component	■	■	■	■
Calibration and configuration data management and map data to software design	■	■	■	■
Define build dependency	■	■	■	■
View existing build dependencies of software design component	■	■	■	■
Remove build dependencies of software design component	■	■	■	■
Modify software design component	■	■	■	■
Revise software design component	■	■	■	■
Delete software design component	■	■	■	■
Link software design component to binary	■	■	■	■
Create composition of software design component	■	■	■	■
Create repository of embedded software component	■	■	■	■
Binary Management				
Create binary software revisions	■	■	■	■
Establish traceability from binary to software design component and parameters	■	■	■	■
Upload binary data into Teamcenter	■	■	■	■
Make software available for flashing	■	■	■	■
Managing control units				
Create control units	■	■	■	■
Identify and associate hardware and software to control units	■	■	■	■
Query for hardware software compatibility	■	■	■	■
Associate control units to product configuration	■	■	■	■
■ Required				
■ Relevant				

Before you begin

For information about versions of operating systems, third-party software, and Teamcenter software configurations that are certified for your platform, see the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com.

Prerequisites	Embedded Software Solutions have the same hardware and software requirements as Teamcenter. If you are implementing ClearCase Integration to manage embedded software, ensure the following:

- Install a supported version of ClearCase. The supported version is:
  - ClearCase 7.1.1

  For instructions on installing ClearCase on your operating system, see the vendor's documentation at http://www.ibm.com
- Ensure you have a valid ClearCase license. If your ClearCase version is a multisite version, then you must also have a multisite license.
- Ensure that your Teamcenter host runs one of the following supported platforms:
  - Microsoft Windows
  - SUSE Linux

  For information about operating system versions certified for Teamcenter, see the see the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com.

  > Note:
  >
  > Management of ClearCase data is supported only in the rich client.

Enable Embedded Software Solutions	To enable you must **install Embedded Software Solutions**.
Configure Embedded Software Solutions	You may be required to set some preferences for configuring Embedded Software Solutions.
Start Embedded Software Solutions	You do not need to manually start Embedded Software Solutions. If you have installed and enabled Embedded Software Solutions, the menu commands appear in My Teamcenter, Platform Designer, and Structure Manager.

Enabling Embedded Software Solutions

Overview of enabling Embedded Software Solutions

Depending on the functionality you require, you must first install one or more of these solutions:

- Embedded Software Manager
- Embedded Software Design Data Management

- Calibration and Configuration Data Management
- SCM ClearCase for Foundation

You must also check for the license and display the menu commands.

Installing Embedded Software Solutions

Install Embedded Software Solutions using Deployment Center

Add the Embedded Software Solutions application to your existing Teamcenter environment.

Procedure

1. Log on to Deployment Center and select the environment to which you want to add Embedded Software Solutions.
2. Go to the **Applications** task. Click **Add or Remove Selected Applications**.
3. In the **Available Applications** panel, use the web browser search to find the following applications:
   - **Calibration and Configuration Data Management**

     Installs Calibration and Configuration Data Management.
   - **Embedded Software Design Data Management**

     Installs Embedded Software Design Data Management.
   - **SCM ClearCase for Foundation**

     Installs Teamcenter ClearCase Integration.
   - **ESM Base**

     Installs base features for Embedded Software.
   - **ESM Processor**

     Installs processor artifacts for Embedded Software.
   - **ESM Software**

     Installs software artifacts for Embedded Software.

Select the application, and then click **Update Selected Applications**.

Deployment Center automatically selects any additional dependent applications.

4. Go to the **Components** task.

5. In the **Selected Components** list, note any remaining components whose configuration status is not **100%**. Select each incomplete component, enter required parameters, and save component settings until all components in the environment show a configuration status of **100%**.

   When all components are fully configured, the **Deploy** task is enabled.

6. Go to the **Deploy** task. Click **Generate Install Scripts** to generate deployment scripts you will use to update affected machines.

   When script generation is complete, note any special instructions in the **Deploy Instructions** panel.

7. Locate deployment scripts, copy each script to its target machine, and then run each script on its target machine.

   For more information about running deployment scripts, see *Deployment Center — Usage*.

Install Embedded Software Solutions features using Teamcenter Environment Manager

Typically, you install the Embedded Software Solutions when you install Teamcenter. However, if Teamcenter is already installed, you can install the Embedded Software Solutions using Teamcenter Environment Manager (TEM).

1. Start TEM.

2. Proceed to the **Features** dialog box.

3. Choose **Teamcenter 9→Extensions→Mechatronics Process Management** and select the following:

   - **Embedded Software Manager** to install Embedded Software Manager
   - **Embedded Software Design Data Management** to install Embedded Software Design Data Management.
   - **Calibration and Configuration Data Management** to install Calibration and Configuration Data Management. **Product Variant** gets automatically installed when you select this option.

Note:

If you are upgrading the Calibration and Configuration Data Management solution, you must manually update the stylesheet by following these steps:

a. Find these XMLRenderingStylesheet type datasets in My Teamcenter– **datasetParmDefBCDRevision**, **datasetParmDefBitDefRevision**, **datasetParmDefBoolRevision**, **datasetParmDefDateRevision**, **datasetParmDefDblRevision**, **datasetParmDefHexRevision**, **datasetParmDefIntRevision**, **datasetParmDefSEDRevision**, **datasetParmDefStrRevision**, **datasetParmGrpValRevision**, **datasetParmDef**, **datasetParmGrpDef**, and **datasetParmGrpDefRevision**.

b. Find these XML stylesheets in *$TC_DATA*– **ParmDefBCDRevision.xml**, **ParmDefBitDefRevision.xml**, **ParmDefBoolRevision.xml**, **ParmDefDateRevision.xml**, **ParmDefDblRevision.xml**, **ParmDefHexRevision.xml**, **ParmDefIntRevision.xml**, **ParmDefSEDRevision.xml**, **ParmDefStrRevision.xml**, **ParmGrpValRevision.xml**, **ParmDef.xml**, **ParmGrpDef.xml**, and **ParmGrpDefRevision.xml**.

c. Copy the content from the XML stylesheets into the corresponding XMLRenderingStyleSheet type datasets.

- **SCM ClearCase for Foundation** to install Teamcenter ClearCase Integration.

  Choose **SCM ClearCase for Foundation** feature. This installs the ClearCase types and sets Teamcenter preferences to enable the integration.

  Note:

  You must have IBM ClearCase installed before you attempt to install the Teamcenter ClearCase Integration. You should also know the ClearCase server name as this is required during installation.

Embedded Software manager business objects installed

During installation, you are prompted to select the business objects you want to use. You can use existing business objects, if you have any, or use the default Teamcenter business objects.

The **Foundation** option in Teamcenter Environment Manager provides the following three options:

- Base
- Processor
- Software

Select the base option to install the default Teamcenter base business objects. The processor and software are optional. You must select the base option if you want to select and install the processor and software options.

If you use existing business objects, you must update the relevant preferences after the installation is complete. If you use the default Teamcenter business objects and select all the three options, Teamcenter Environment Manager automatically installs the business objects and sets the necessary preferences with the values of the installed business objects.

Type	Parent type	Purpose
AppSoftware	**Software**	Represents the application software, for example, **radio**.
Calibration	**Software**	Represents calibration software embedded in a control unit. For example, the software that calibrates the product for a particular context.
ConfigFile	**Software**	Represents configuration software or a file containing information used by the application software. For example, the frequency range of a radio may differ between Europe and the U.S.A. The **radio** application software reads the range from the configuration software to identify the correct range.
PriBootLoader	**Software**	Represents the primary bootloader software that loads operating software into the control unit.
Processor	**Item**	Represents an internal electronic device that runs software. For example, a control unit may contain three CPUs, each of which is modeled as a processor.
SecBootLoader	**Software**	Represents the secondary bootloader software that controls flashing access to the control unit.

Type	Parent type	Purpose
Software	**Item**	Represents the parent type for all embedded software. It is not instantiated and is not visible in the user interface.
Embeds	**ImanRelation**	Represents an association of the software with the processor in the context of a structure. It indicates the embedded software (secondary) that is embedded in a specified processor (primary).
DependentOn	**ImanRelation**	Models dependencies between embedded software within the same control unit or across different control units.
GatewayOf	**ImanRelation**	Models dependencies between processors. A processor that is linked and accessed through another processor (commonly called a *gateway*) is associated with the other processor by this relationship.
MemoryType LOV	**ListOfValues**	Provides the list of values for the **Memory Type** attribute of the processor, for example, **Programmable** and **Reprogrammable**.
ByteOrder LOV	**ListOfValues**	Provides the list of values for the **byteOrder** attribute of the processor, for example, **Little Endian** and **Big Endian**.
Hardware Architecture LOV	**ListOfValues**	Provides the list of values for the **architecture** attribute of the processor, for example, **32 bit** and **64 bit**.
SCM_element_specification	**ImanRelation**	Represents which source code versions were used in generation of a given

Type	Parent type	Purpose
		embedded software binary part.
StruObjAttrOverride	**ImanRelation**	Implies override of the item/ item revision class attributes in a given context.

Embedded software design data management business objects installed

Type	Parent type	Purpose
SwDesignComp	**Design**	Represents the embedded software component and manages the design data and the life cycle of the embedded software component.
SwDesignCompRevision	**DesignRevision**	Represents an object that has source code associated to it as specification in the form of **SCMVersionObject** or a regular dataset. Reference data such as design specifications, test specifications, or behavior specifications can be associated as reference data.

Calibration and configuration data management business objects installed

Type	Parent type	Purpose
CalParm **CalParmRevision**	**Architecture**	Represents all CCDM specific classes, which represent a group of parameters. It holds all the generic nonvalue attributes that are used to define a group of parameters. Objects of this type cannot be created and are hidden in the user interface.
ParmGrpDef **ParmGrpDefRevision**	**CalParm**	Represents the parameter master dictionary, template, or the project specific breakdown. The **ParmGrpDef** class organizes the parameters specific to a particular feature, function, or device. The class can also represent a subgroup

Type	Parent type	Purpose
		of parameters in the hierarchy when the parent group is used for organizational purposes. An extension is attached to the **ParmGrpDefRevision** class so that the associated objects are deleted along with the revision.
ParmDef **ParmDefRevision**	**Item**	Represents a single parameter that is used for calibrating or configuring embedded software. This is the parent class of all parameters. It is not visible in the user interface and you cannot create instances of this object. The attached **CCDMParmTypeForParmDef** LOV has a measurement value that must be identified. The **tableDefinition** property on most subclasses of **ParmDefRevision** have property constants attached with them to enable initialization of the number of rows and columns for the table.
ParmDefInt **ParmDefIntRevision**	**ParmDef** **ParmDefRevision**	Represents the **Integer** data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefIntRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefDbl **ParmDefDblRevision**	**ParmDef** **ParmDefRevision**	Represents the **Double** data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefDblRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefStr **ParmDefStrRevision**	**ParmDef** **ParmDefRevision**	Represents the **String** data type of the parameter. It is a subclass of the **ParmDef** class.

Type	Parent type	Purpose
		The **ParmDefStrRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefBool **ParmDefBoolRevision**	**ParmDef** **ParmDefRevision**	Represents the **Boolean** data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefBoolRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefDate **ParmDefDateRevision**	**ParmDef** **ParmDefRevision**	Represents the **Date** data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefDateRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefHex **ParmDefHexRevision**	**ParmDef** **ParmDefRevision**	Represents the **Hex** (hexidecimal) data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefHexRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefSED **ParmDefSEDRevision**	**ParmDef** **ParmDefRevision**	Represents the **SED** (state encoded) data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefSEDRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values. If the **ParmDefSEDRevision** is created without specifying values for the valid values attribute and with only the domain element name, the initial value attribute will contain the domain element name string that was selected at the time of creation.

Type	Parent type	Purpose
ParmDefBCD **ParmDefBCDRevision**	**ParmDef** **ParmDefRevision**	Represents the binary coded decimal data type of the parameter. It is a subclass of the **ParmDef** class. The **ParmDefBCDRevision** has the **Ccd0TypeRefInitValue** constant associated with it to enable initialization of the table cell values.
ParmDefBitDef **ParmDefBitDefRevision**	**ParmDef** **ParmDefRevision**	Represents the **BitMap** data type of the parameter. It is a subclass of the **ParmDef** class.
BitValue	**POM_object**	Holds the byte/bit information and the name.
BitDef	**POM_object**	Holds the byte/bit information for each bit in a **ParmDefBitDefRevision** class.
ParmGrpVal **ParmGrpValRevision**	**Item** **ItemRevision**	Represents the part solution to the **ParmGrpDef** object belonging to the project specific architecture breakdown. Represents the value solution of the **ParmDefBitDefRevision** object in the project-specific breakdown This class organizes all the **ParmVal** objects, which have values specific to parameters grouped together in **ParmGrpDef** for which this is a solution.
ParmVal	**POM_object**	Carries the value of a parameter. The value is based on a specific version of the **ParmDef** class.
ParmValInt	**ParmVal**	Carries **Integer** type of values of a parameter. The value can be a single integer value or a single-dimension array or a two-dimension array of integer values.
ParmValDbl	**ParmVal**	Carries **Double** type of values of a parameter. The value can be a single double value or a single-dimension array or a two-dimension array of double values.

Type	Parent type	Purpose
ParmValStr	**ParmVal**	Carries **String** type of values of a parameter. The value can be a single string value or a single-dimension array or a two-dimension array of string values.
ParmValBool	**ParmVal**	Carries **Boolean** type of values of a parameter. The value can be a single logical value or a single-dimension array or a two-dimension array of logical values.
ParmValDate	**ParmVal**	Carries **Date** type of values of a parameter. The value can be a single date value or a single-dimension array or a two-dimension array of date values.
ParmValHex	**ParmVal**	Carries **Hex** type of values of a parameter. The value can be a single hex value or a single-dimension array or a two-dimension array of hex values.
ParmValSED	**ParmVal**	Carries **SED** type of values of a parameter. The value can be a single **SED** value or a single dimension array or a two dimension array of **SED** values.
ParmValBCD	**ParmVal**	Carries **BCD** type of values of a parameter. The value can be a single **BCD** value or a single dimension array or a two dimension array of **BCD** values.
ParmValBitDef	**ParmVal**	Carries **BitDef** type of values of a parameter. It holds values for either single bit of a byte or for multiple bits and bytes.
BitValue	**POM_object**	Represents the **Bit** value.
Fnd0GenericConversionRule	**POM_application_object**	Holds all generic non value attributes which are used to define a conversion rule. It is an abstract base class for all conversion rules.
Fnd0FormulaConversion	**Fnd0GenericConversionRule**	Represents a conversion rule depending on an algebraic formula.

Type	Parent type	Purpose
		It calls the algebraic formula to translate the input value into output value.
Fnd0RecordColConversion	**Fnd0Generic ConversionRule**	Represents a table conversion. It contains a list of record conversion to map an input value to an output value.
Fnd0RecordConversion	**POM_object**	Holds all generic non-value attributes that are used to define a record conversion. **Fnd0RecordConversion** is an abstract base class for all record conversions.
Fnd0ValueConversion	**Fnd0Record Conversion**	Represents the mapping from the input value to the output value.
Fnd0RangeConversion	**Fnd0Record Conversion**	Represents the mapping to the output value if the input value is between the minimum value and the maximum value.
Fnd0AlgebraicFormula	**POM_application _object**	Represents algebraic formulas: for example, linear, quadratic, and rational.
Mem0MemoryBlock	**POM_application _object**	Represents the memory block.
Mem0MemoryRecord	**POM_application _object**	Represents the memory record.
Ccd0ParmMemRecord	**Mem0Memory Record**	Represents the parameter memory record.
Mem0MemoryContent	**POM_object**	Represents the memory content.
Mem0MemoryLayout	**Item**	Represents the memory layout.
Ccd0ParmMemLayout	**Mem0Memory Layout**	Represents the parameter memory layout.
Mem0MemoryLayout Revision	**ItemRevision**	Represents a revision of the memory layout.
Ccd0MemoryLayout Revision	**Mem0Memory LayoutRevision**	Represents a revision of the parameter memory layout.

Type	Parent type	Purpose
Ccd0HasInputSource	**ImanRelation**	Represents the input sources of the parameter memory layout.
Ccd0HasOutputSource	**ImanRelation**	Represents the output files of the parameter memory layout.

Calibration and configuration data management LOVs

Name	Description	Type	Attached to property	On business object
CCDMGrp DefRepresents	Valid representations for a parameter group definition. The possible values can be: **Organizational Group** **Packeted Parameter** **Parameter Breakdown** **Parameter Dictionary** **Parameter Group**	String	**Represents**	**ParmGrpDef**
Memory size	List of memory size units. The possible values can be: **Bit** **Byte**	String	**sizeUnits**	**ParmDefRevision**
CCDMParmType ForParmDef	Valid parameter types of parameter definitions. The possible values can be: **Calibration** **Configuration**	String	**parmType**	**ParmDef**

Teamcenter ClearCase Integration business objects installed

Type	Parent type	Purpose
SCMVersionObject	Workspace object	Represents the version of an element of ClearCase in Teamcenter. It can contain single or multiple ClearCase elements. This object uses the configuration context to determine which version of the ClearCase element it is associated with.
SCMElement	Workspace object	Represents the ClearCase element in Teamcenter. It encapsulates the element details, such as relative element path in the version object base (VOB), element type, and the ClearCase server to which the element belongs.
SCMCompound Element	Workspace object	Represents a collection of SCM element objects that are managed by the **SCMVersionObject**.
SCMConfigContext	Workspace object	Represents the SCM server.
SCMServer	POM object	Holds the SCM configuration information used to identify the correct version of the **SCMVersionObject**.

Licensing

You should have a license for:

- Embedded Software Solutions
- Teamcenter ClearCase Integration

  > Note:
  >
  > Ensure that you also have an IBM ClearCase product license.

- Calibration and Configuration Data Management

  > Note:
  >
  > This solution also requires the level of Teamcenter license that enables authoring.

If any of the required licenses are not available when a user attempts to use the respective solution (for example, all the licenses are assigned to other users), Teamcenter displays an error message. If you encounter problems using Calibration and Configuration Data Management, contact your system administrator; the problem may be a licensing issue.

Displaying Embedded Software Solutions menu commands

The Embedded Software Solutions menu commands in Structure Manager and My Teamcenter are not displayed until the solutions are installed. You must also unsuppress the menus for the required user or role. You must have administrator privileges to enable the menu commands using Command Suppression.

If you are installing Embedded Software Manager, the menus are not displayed if the solution is not licensed.

Configuring Embedded Software Solutions

Overview of configuring Embedded Software Solutions

Whether you need to configure the solution depends on how you installed it. If, during installation, you choose to use the default Teamcenter business objects, no further configuration of the Embedded Software Manager is required. Other solutions required additional configuration.

If you choose to use your own existing business objects, you must set the necessary preferences for each solution as described.

Setting Embedded Software Manager preferences

Set these preferences after installing Embedded Software Manager and before users are given access to the system.

- **ESM_Processor_linked_attr**
- **ESM_Processor_programmableInService_attr**
- **ESM_Processor_memoryType_attr**
- **ESM_Processor_byteOrder_attr**
- **ESM_Processor_architecture_attr**
- **ESM_HW_Compatibility_Report_TransferMode**
- **ESM_SW_Compatibility_Report_TransferMode**

- **ESM_HW_Compatibility_Report_HTML_StyleSheet_Dataset**
- **ESM_SW_Compatibility_Report_HTML_StyleSheet_Dataset**
- **ESM_HW_Compatibility_Report_Excel_StyleSheet_Dataset**
- **ESM_SW_Compatibility_Report_Excel_StyleSheet_Dataset**
- **ESM_Compatibility_Report_Excel_Template_Dataset**
- **ESM_processor_direct_child_of_hardware**

Setting Embedded Software Design Data Management preferences

- Add the **SwDesignComp** and **SwDesignCompRevision** objects to the **ICS_classifiable_types** preference.

Setting Calibration and Configuration Data Management preferences

Set these preferences after installing Calibration and Configuration Data Management and before users are given access to the system.

- **ParmDefIntRevision.RENDERING**
- **datasetParmDefIntRevision.REGISTEREDTO**
- **ParmDefDblRevision.RENDERING**
- **datasetParmDefDblRevision.REGISTEREDTO**
- **ParmDefStrRevision.RENDERING**
- **datasetParmDefStrRevision.REGISTEREDTO**
- **ParmDefBoolRevision.RENDERING**
- **datsetParmDefBoolRevision.REGISTEREDTO**
- **ParmDefDateRevision.RENDERING**
- **datasetParmDefDateRevision.REGISTEREDTO**
- **ParmDefHexRevision.RENDERING**
- **datasetParmDefHexRevision.REGISTEREDTO**

- **ParmDefSEDRevision.RENDERING**
- **datasetParmDefSEDRevision.REGISTEREDTO**
- **ParmDefBCDRevision.RENDERING**
- **datasetParmDefBCDRevision.REGISTEREDTO**
- **ParmGrpValRevision.RENDERING**
- **datasetParmGrpValRevision.REGISTEREDTO**
- **ParmDefBitDefRevision.RENDERING**
- **datasetParmDefBitDefRevision.REGISTEREDTO**
- **ParmGrpDef.RENDERING**
- **datasetParmGrpDef.REGISTEREDTO**
- **ParmDef.RENDERING**
- **datasetParmDef.REGISTEREDTO**
- **ParmGrpDefRevision.RENDERING**
- **datasetParmGrpDefRevision.REGISTEREDTO**
- **ProductVariant.RENDERING**
- **datasetProductVariant.REGISTEREDTO**
- **ProductVariant.CREATERENDERING**
- **datasetProductVariantCreate.REGISTEREDTO**
- **ProductVariantIntent.RENDERING**
- **datasetProductVariantIntent.REGISTEREDTO**
- **ProductVariantIntent.CREATERENDERING**
- **datasetProductVariantIntentCreate.REGISTEREDTO**
- **Ccd0ColumnPreference**

- **CCDM_grp_rev_val**
- **TC_relation_required_on_export**
- **ParmGroup_Assigned_Hierarchy_Shown**
- **Parameter_Assigned_Columns_Shown**
- **Parameter_Available_Columns_Shown**
- **OverrideContRev_Assigned_Columns_Shown**
- **OverrideContRev_Available_Columns_Shown**
- **MemoryLayoutRevOrBlock_Assigned_Columns_Shown**
- **MemoryRecord_Assigned_Columns_Shown**
- **MemoryOverrideRecord_Assigned_Columns_Shown**
- **ConvOverrideRecord_Assigned_Columns_Shown**
- **ConvOverrideRecord_Available_Columns_Shown**

Setting Teamcenter ClearCase Integration preferences

Set these preferences after installing the Teamcenter ClearCase Integration and before users are given access to the system.

- **SCM_ClearCase_Server**
- **SCM_ClearCase_UNCO_Keep_Version**
- **SCM_CheckIn_Identical_Version**

Embedded Software Solutions interface

Overview of Embedded Software Solutions interface

There are some unique menu commands available when you install the licensed solutions.

Embedded Software Manager menus

Embedded Software Manager adds several menu commands to Structure Manager and My Teamcenter as follows.

Applications	Menus command	Submenu command	Description
Structure Manager	**View→Embedded Software Explorer→Show Associated Software**	**Embedded** **Uses** **Used By**	Shows associated software by relations.
Structure Manager	**View→Embedded Software Explorer→Show Associated Processor(s)**	**Embedding** **Accessed By** **Gateway**	Shows associated processors by relations.
Structure Manager	**Tools→Embedded Software Manager→Associate Software to**	**Processor** **Software**	Associates software either to a processor or to another software.
Structure Manager	**Tools→Embedded Software Manager→Associate Processor to**	**Gateway**	Associates processor to a gateway.
Structure Manager	**Tools→Embedded Software Manager→Remove Software Association**	**Software**	Removes association between two associated software.
Structure Manager	**Tools→Embedded Software Manager→Remove Processor Association**	**Software** **Gateway**	Removes association between processor-gateway and processor-software.
My Teamcenter	**Tools→Embedded Software Manager→Compatibility Reports**	**Hardware-Software** **Software-Hardware**	Checks hardware-software compatibility.

Embedded Software Design Data Management menus

Embedded Software Design Data Management adds several menu commands to Structure Manager and My Teamcenter as follows

Application	Menu command	Description
My Teamcenter	**File→New→Software Design Component**	Creates a new software design component
Structure Manager	**File→New→Software Design Component**	Creates a new software design component

Calibration and Configuration Data Management menus

Calibration and Configuration Data Management adds several menu commands to Structure Manager, My Teamcenter and Platform Designer, as follows.

Application	Menu command	Description
My Teamcenter	**File→New→Parameter Management**	Creates a parameter definition, parameter group definition, or parameter value group.
My Teamcenter	**File→New→Parameter Management→Parameter Definition**	Creates a parameter definition.
My Teamcenter	**File→New→Parameter Management→Parameter Definition Group**	Creates, revises, updates, and deletes the parameter definition group.
My Teamcenter	**File→New→Parameter Management→Parameter Value Group**	Creates, revises, updates, and deletes the parameter value group.
My Teamcenter	**File→New→Product Variant**	Creates a product variant object.
My Teamcenter	**File→New→Product Variant Intent**	Creates a product variant intent object.
Structure Manager	**File→New→Parameter Management→Parameter Definition Group**	Creates, revises, updates, and deletes the parameter definition group.
Structure Manager	**File→New→Parameter Management→Parameter Definition**	Creates, revises, updates, and deletes the parameter definition.
Platform Designer	**File→New→Architecture Breakdown**	Creates a parameter breakdown.

ClearCase Integration menus

The ClearCase Integration interface has some unique menu commands, which appear in the Organization application and in My Teamcenter.

Application	Menu command	Description
Organization	**Tools→Verify ClearCase Connectivity**	Verifies that ClearCase is connected to Teamcenter.
My Teamcenter	**File→New→SCMVersionObject**	Creates a new **SCMVersionObject**.
My Teamcenter	**Edit→User Setting→SCM Workspace**	Displays SCM workspace options.
My Teamcenter	**Tools→Check-In/Out→Check-Out**	Checks out **SCMVersionObject**.
My Teamcenter	**Tools→Check-In/Out→Check-In**	Checks in **SCMVersionObject**.
My Teamcenter	**Tools→Check-In/Out→Cancel Check-Out**	Cancels **SCMVersionObject** checkout.

Understanding Embedded Software Solutions

Basic concepts about Embedded Software Solutions

Embedded software runs on systems that interact with the physical world in real time. These systems are referred to as embedded systems. Embedded systems are used to control and monitor the operation of machinery and can be found in a variety of applications. These embedded systems are becoming complex and are increasing in size and sophistication. They are made up of many hardware and software components. Most of these components are inter-related. For example, software resides on a processor, or a software component depends on other software for execution, or a hardware component may depend on another hardware component for activation, and so on.

Software for embedded systems is developed using reusable components that can be tracked against a precise source code version. To allow the reuse of the same software binaries in different product variants, the parameters are calibrated and configured after the binaries are generated.

Managing embedded systems using Embedded Software Solutions

With the increasing complexity of embedded systems, the management of enormous amounts of product data is crucial. The product structure with all the embedded components and their unique attributes and relations is preserved, reused, and managed in an embedded system. It is vital to have a data management system that allows the configuration of embedded components with their attributes and manages the dependencies and changes among embedded components and systems. It should also provide a list of compatible components for any given embedded component.

Teamcenter has powerful features to design and manage complex product structures. Using Embedded Software Solutions, you can create custom embedded hardware components like processors, embedded software components such as calibration, bootloaders to enable flashing, and software binaries. You can manage software design components as self-contained entities that represent a particular design and related parameters (for example, source code) of a released software element. With Embedded Software Solutions, you can also create relations between various embedded components such as processor and software, software and software, and processor and processor, and so forth.

Embedded Software Solutions provide a reporting mechanism of compatible embedded components, which allows you to get information about compatible and non-compatible hardware and software components for any embedded software. This helps you to assess the impact of changes and to identify possible changes.

This solution also allows external flashing applications to integrate with Teamcenter. You can download the correct embedded software to the flashing station and then the flashing application can flash it to the appropriate component of the embedded systems.

Calibration and Configuration Data Management allows you to manage dictionaries of embedded software parameters that you can reuse in various products. You can create, manage, and reuse these parameter values in different product variants. They can then be configured with the standard Teamcenter configuration features to unambiguously identify the values for a given variant.

Working in a Multi-Site Collaboration environment

When you share product structures in a Multi-Site Collaboration environment, the following embedded software objects can also be shared:

- Software item

  When a user imports or exports an assembly that includes a software item, Teamcenter transfers the associated relations (**DependentOn**, **Embeds**) and the associated secondary objects.

- Message

  When a user imports or exports an assembly that includes messages, Teamcenter transfers the associated relations (**Source** and **Target** control units, **transmitter**) and the associated secondary objects.

- Processor

  When a user imports or exports an assembly that includes a processor, Teamcenter transfers the associated relations (**Embeds** and **GatewayOf**) and the associated secondary objects.

- Software design component

  When a user imports or exports an assembly that includes a software design component, Teamcenter transfers the associated secondary objects.

- CCDM object

  When a user imports or exports an assembly that includes a CCDM object, Teamcenter transfers the associated parameter definitions, parameter definitions, and parameter value groups

Remote users can modify shared structures only if they have the necessary access privileges.

Working with ECAD integrations

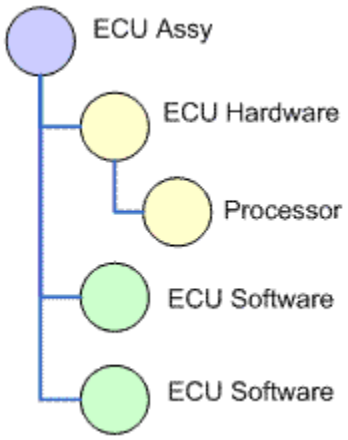
You can use both the Embedded Software Solutions and ECAD (Electronic Computer Aided Design) integration functionality together in a common environment for managing information related to the electronic and software components. Set the **ESM_processor_direct_child_of_hardware** preference to **false** if you want to use Embedded Software Solutions and ECAD together.

Embedded Software Solutions deal with managing software and its dependency on other software and hardware; ECAD deals with managing the hardware components of a control unit or programmable unit.

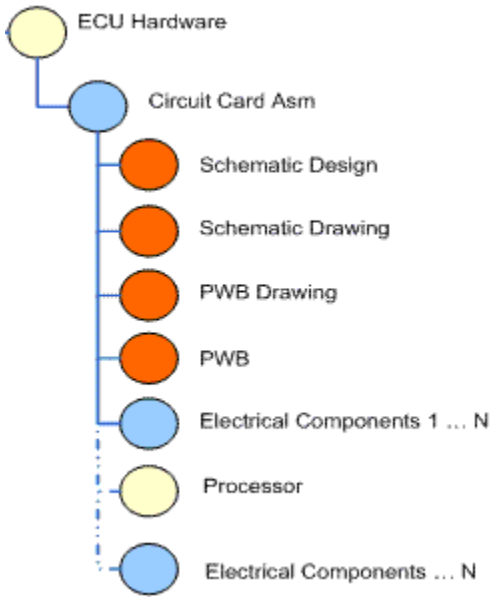
If you design the hardware and software as a package, you can leverage the following benefits:

- Manage hardware ECAD and software data together, referencing the EDA data from the embedded software structure.
- Enable the validation of software on the hardware.
- Ensure concurrent and collaborative development of hardware and software.

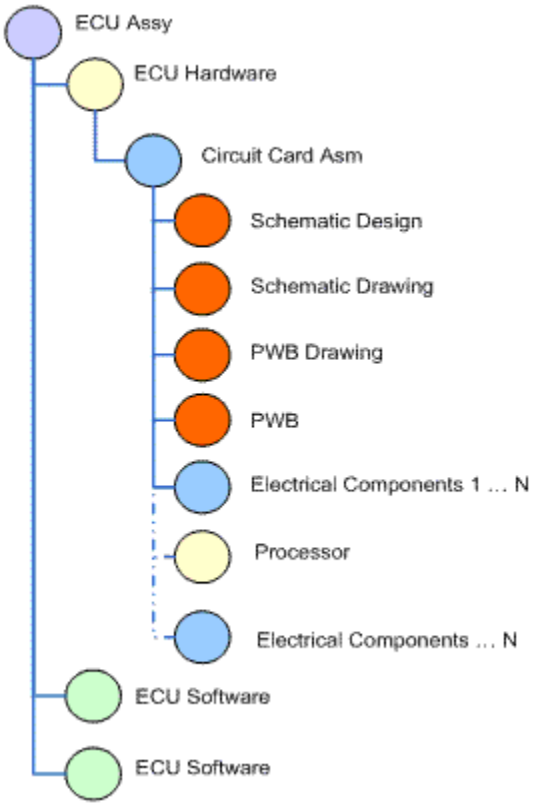
For example, as a software designer, you must know the processor of the control unit hardware that embeds the given software of the control unit. To provide this capability, Embedded Software Solutions must model at least the processor and CPU of the hardware as shown in the following figure.



The hardware may be made up of several other components such as PCB, capacitors, and resistors. These components are modeled as part of the ECAD solution as shown in the following figure.



If both Embedded Software Solutions and ECAD solutions are deployed, the hardware can be expanded to show the detailed components that come from the ECAD solution and software designers can embed the software to the processor under the hardware. In this case, the **ESM_processor_type** preference must be configured to the item type used by the ECAD solution to represent the processor. The resulting structure looks like the example shown in the following figure.



Basic tasks using Embedded Software Solutions

With Embedded Software Solutions, you can:

- Configure various embedded components and manage the associations among them and with other systems.
- Design the embedded components with corresponding compatible devices with ease as the compatible devices for any embedded components are listed. Also, if an embedded component needs to be replaced with a new one, designers can validate the dependent components through compatibility checking.
- Perform a quick impact analysis of interdependent components. Impacted components can be identified in the early stages of design. This reduces the number of recalls associated with embedded systems of any product.
- Store and retrieve the embedded systems information in a PLM environment. You can revise, check out, check in, and release the individual embedded systems, thus enabling the comprehensive data management capabilities for embedded components.
- Configure hardware and software parts separately or as part of the same structure.
- Configure the embedded system in the context of the overall product.
- Manage volumes of messages and signals in networks and associate them with embedded devices within a single source of information.
- Manage repository of reusable software components.
- Determine the compatible network components of any message.
- Share embedded network systems data across manufacturers, suppliers, and vendors. This ensures seamless transfer of embedded systems data and communication about modifications to any of the components.
- Manage reusable calibration and configuration data of embedded systems.
- Manage parameter dictionaries for use in multiple products.
- Set values for parameters and modify and delete them.
- Manage design data (source code, models, object files, and third-party libraries) of embedded software components.
- Create, update, and delete software design data components.

- Associate specifications, source code, and test cases to software design data components, and define dependencies between different software design data components.
- Associate a software design data component to a software binary.
- Create a composition of the software design component.
- Configure, track, and control all hardware and software components that comprise your product and their related product/process definitions.
- Manage references to the ClearCase version object and use Teamcenter for product modeling, workflow, and process management.
- Check out, check in, and create a workflow process.
- Provide access to ClearCase from Teamcenter.
- Manage traceability to the source code from the software binary part of a product. This is helpful in assessing the impact of changes and identifying the source code to change for a particular issue of a software binary.

2. Managing the system design process

Overview of managing the system design process

Before using the Embedded Software Solutions, the Teamcenter administrator creates the necessary projects and assigns user groups to them. The Organization application allows you to create projects and groups and then assign groups or individual users to each project.

After setting up the projects, you can proceed to:

1. Create and manage the functional design.
2. Create and manage the product architecture.
3. Distribute functions across control units in the product architecture.
4. Create and manage signals and messages and define dependencies.

Functional design management

To build and manage the functional model of a system, the system engineer creates functions, decomposes them into further subfunctions, and then defines the dependencies between these functions.

Product electronic architecture management

Overview of product electronic architecture management

Typically, a system engineer follows this process to create the overall product architecture:

- Identifies the number of control units needed and creates them within the product.
- Identifies messaging protocols.
- Creates the required networks and associates control units to the network.
- Distributes functions across the control unit network.
- Defines dependencies between control units.
- Identifies the signals that are consumed and transmitted as part of each frame.
- Assigns each control unit to a design engineer for detailed design work.

Creating control units

Overview of creating control units

To start with, you must identify the number of control units you need in a given product architecture and either find them for reuse or create them.

To create a control unit you must:

1. Create a control unit item.
2. Create hardware corresponding to the control unit.
3. Create processor related to the hardware of the control unit.
4. Create ports of the hardware.

In all these steps you can either search for existing objects or create a new object.

Create a control unit

1. Choose **File→New→Item**.

   The **New Item** dialog box appears.

2. Select the appropriate item type that represents the control unit part at your site.

   > Note:
   >
   > Teamcenter does not provide a specific default item for control units.

Create hardware

1. Choose **File→New→Item**.

   The **New Item** dialog box appears.

2. Select the appropriate item type that represents the hardware to be used in a given control unit part at your site.

   > Note:
   >
   > Teamcenter does not provide a specific default item type for hardware to be used in control units.

3. Add the hardware as a child of the control unit:

    a. Copy the hardware to the clipboard.

    b. Send the control unit to Structure Manager.

    c. Select the control unit and choose **Edit→Paste**.

Create processor under control units

1. Choose **File→New→Item**.

    The **New Item** dialog box appears.

2. Select **Processor** as the item type or a type that was configured to represent a processor while configuring Embedded Software Manager.

    The **New Processor** dialog box appears.

3. Enter the processor name and description, and click **OK** or **Apply**.

    Teamcenter creates the new processor.

    If the processor is linked to the network, the linked attribute should be set to **True**.

    > Note:
    >
    > You can click **Next** to define the master form attributes and assign the processor to a project.

4. Add the processor as a child of the hardware in the control unit:

    a. Copy the processor the clipboard.

    b. Send the control unit to Structure Manager.

    c. Select the hardware under the control unit and choose **Edit→Paste**.

To set the processor ID, enter the processor identifier by double-clicking the **Occurrence Name** cell and typing the appropriate value or defining another processor (a *gateway*) through which the processor is accessed. You can map the processor identifier to the occurrence note by creating an occurrence note type and setting the processor ID using this note type; alternatively, you can map the processor ID to the absolute occurrence ID.

Create ports for the hardware of control units

1. Choose **File→New→Interface Definition**.

2. Select **Connection Terminal** as the type.

3. Copy the ports created in My Teamcenter to the clipboard and send the control unit containing the hardware to Structure Manager.

4. Select the hardware BOM line in Structure Manager, copy the objects from the clipboard, and paste them here.

Create control unit topology

1. Choose **File→New→Item**.

   The **New Item** dialog box appears.

2. Select the appropriate item type that represents the electronic architecture of the product at your site.

   > Note:
   >
   > Teamcenter does not provide a specific default item type for representing the electronic architecture of the product.

Next, create the connection objects for establishing connectivity of the control units.

Create connection object representing a network

1. Choose **New→Connection→Revisable**.

2. Select the **Connection** item type for each type of network for the product.

To use these connections to model connectivity between the control units, you must put them in the product structure of the control unit topology at the same level as the control units.

Associate control unit as a child of the electronic architecture

1. Select the item representing the electronic architecture of the product and send it to Structure Manager.

2. Search for the required control units and copy them to the clipboard.

3. Select the top line in Structure Manager and choose **Edit→Paste**.

4. Search for the required connection objects and copy them to the clipboard.

5. Select the top line in Structure Manager and choose **Edit→Paste**.

The electronic architecture of the product is updated to include all required control units and connection objects. Next, you must establish connectivity between the control units.

Associating ports of control units to the connection object representing the network

In the electronic architecture of the product, control units that must be connected by the same connection are connected by:

1. Selecting the port of the control unit's hardware.

2. Selecting the connection object and then associating the ports created under the control unit to the connection object representing the network, by choosing **Edit→Connect**.

Next, you must establish dependency between the processors to capture how a processor of one control unit is accessed through a processor of another control unit on the network.

Create processor dependency

You can access other processors within the same control unit or across several control units through a gateway processor. A gateway processor has the **isGateway** attribute set to **True**.

1. In Structure Manager, select each of the processors that you want to associate with the gateway processor and copy them to the clipboard.

2. Select the gateway processor and choose **Tools→Embedded Software Manager→Associate Processors To→Gateway**.

   Teamcenter associates all the processors you copied to the clipboard with the gateway processor.

View the processors accessed through a gateway processor

1. In Structure Manager, select the gateway processor.

2. Choose **View→Embedded Software Explorer→Show Associated Processor(s)→Accessed By**.

   Teamcenter highlights the gateway processor and all processors accessed through it.

View the gateway processor associated with a processor

1. In Structure Manager, select the processor.

2. Choose **View→Embedded Software Explorer→Show Associated Processor(s)→Gateway**.

   Teamcenter displays the processor and its gateway processors in the product structure.

Remove the gateway processor associations from a processor

1. In Structure Manager, select the gateway processor in the product structure.
2. Choose **Tools→Embedded Software Manager→Remove Processor Associations**.

   Teamcenter displays the **Remove Processor Association** dialog box, listing all processors accessed through the selected gateway processor.
3. In the dialog box, select all the accessed processors for which you want to remove the association with the selected gateway processor and click **Remove**.

   Teamcenter removes the selected associations.

Distributing functions across control units

When the system engineer distributes the functions identified for the product across different control units, you can use allocations to allocate functions from the functional model to the control units in the control unit structure. Allocations can be made programmatically or manually with the Multi-Structure Manager application.

Signal management

Overview of signal management

The system engineer creates and associates messages and signals when building the structure of the control units. The messages and signals exchanged between control units result from how the functions are distributed across the control units. Use this information to identify the dependencies between control units and to assist in managing the communication matrix.

Create messages and signals

1. Choose **File→New→Signal**.
2. Choose the appropriate type of signal (**Message** or **Signal**).

   The **New Signal** dialog box appears.
3. Enter an ID, revision, name, and description for the signal.
4. Click **Finish**. Teamcenter creates a new signal of the specified type.

Note:

You can click **Next** to define other attributes of the signal and assign it to a project.

Associate messages to control units

1. In the product structure, select a message and one or more objects of the appropriate type, such as a control unit item for a message or a port for a signal. The valid appropriate types are configured in the following preferences:

   - **SIG_asystem_source_rules**
   - **SIG_asystem_target_rules**
   - **SIG_asystem_transmitter_rules**
   - **SIG_pavriable_rules**
   - **SIG_redundant_rules**

2. In Structure Manager, choose **Tools→Signal Manager→Associate Signal To** and then choose one of the following menu commands:

   - **Source**

     The object is the source, transmitter, or origin of the signal.

   - **Target**

     The object is the target, receiver, or destination of the signal.

   - **Transmitter**

     The object carries the message/signal. Select a connection or its subtype with the signal or the message. The selection of any other object is invalid in this role.

   - **Process Variable**

     The process variable associated with the signal. Select the process variable with the signal.

   - **Redundant Signal**

     The signal is redundant. Copy the redundant signal to the clipboard, select the primary signal to associate with the copied redundant signal, and choose the **Tools→Signal Manager→Associate Signal To→Redundant Signal** menu command.

Teamcenter associates the signal and object.

> Note:
>
> My Teamcenter creates signal definitions and provides signal-related definition information. These signal objects can now be reused in multiple designs by including the signal in a given structure. In the context of the structure, the signal can then be associated to source **transmitter** and target **receiver** objects. Teamcenter creates these relations in a given context, so that you do not need to revise the signal even if the source/target objects change from one structure to another.
>
> By including the signal in the structure, you can apply various configurations to the signal and also use them in fine-grained allocation in context of a given structure.

Associate signals to ports of control units

1. In Structure Manager, select the signal and port of the control unit that you created.
2. Choose **Tools→Signal Manager→Associate Signal to→Source**.

Associate messages to connection

1. In Structure Manager, select the message that you created.
2. Select the connection object.
3. Choose **Tools→Signal Manager→Associate Signal to→Transmitter**.

View associations of a signal

1. In Structure Manager, select the signal in the product structure.
2. Choose **View→Signal Explorer** and then choose one of the following menu commands:
   - **Source**
   - **Target**
   - **Transmitter**
   - **Process Variable**
   - **Redundant Signal**

   Teamcenter displays the appropriate associated objects depending on the selected option for the signal.

View received and transmitted signals

1. In Structure Manager, select the object whose signals you want to view.

2. Choose **View→Signal Explorer→Show Signals→Received** to view signals received by the object.

   Choose **View→Signal Explorer→Show Signals→Transmitted** to view transmitted messages.

   Teamcenter displays the object and signals of the chosen type in the product structure. If you select a control unit and choose **Show Source**, Teamcenter displays the control units that transmit messages for which the selected control unit is a target. Similarly, if you choose **Show Target**, Teamcenter highlights control units that are targets of messages of which the selected control unit is a source.

Remove associations from signals

1. In Structure Manager, select the signal in the product structure.

2. Choose **Tools→Signal Manager→Remove Signal Association** and then choose one of the following menu commands:

   - **Source**
   - **Target**
   - **Transmitter**
   - **Process Variable**
   - **Redundant Signal**

   Teamcenter displays the **Remove Signal Association** dialog box containing all the associations of the selected type for the signal, if there is more than one.

   > Note:
   >
   > In the case of a process variable, the **Remove Signal Association** dialog box is not displayed and the association between the process variable and the signal is removed immediately.

3. Select the required association and click **Remove**.

   Teamcenter removes the selected association from the signal.

Dependency management

Overview of dependency management

Typically, the design engineer uses the following process to create software data and model dependencies:

- Creates application software, calibration software, and bootloader software types and adds them under the control unit item.
- Embeds the different software items of the control unit into the processor of the control unit assembly hardware component.
- Creates the following software dependencies:
  - Associates the calibration software to the application software binary.
  - Associates the application software binary to other application software binaries. This dependency may be useful to service personnel, for example, when a vehicle is serviced. If a problem is identified in one of the control units, the technician may decide to flash the latest version of the application software onto the control unit. However, the latest version of the software may work only with specific versions of software on other control units. Consequently, these dependent control units are affected by the change to the faulty control unit. To identify this change, the technician must know the other software in the assembly that is affected by changes to the updated software, so the corresponding control units can be identified and updated at the same time.

Establish software to hardware dependency

You can associate a software item of any type with one or more processors. To associate a software item to a processor, ensure the following criteria are met:

- The parent of the software item must be the same as some ancestor of the processor item.
- The processor must be a direct child of hardware if the value of the **ESM_processor_direct_child_of_hardware** preference is true. If the value is false, then the processor can be some child of a hardware and does not have to be direct child.

1. In Structure Manager, select the software item and the processor to which you want to associate.
2. Choose **Tools→Embedded Software Manager→Associate Software To→Processor**.

   Teamcenter associates all the software items with the processor through an **Embeds** relationship.

Establish software to software dependency

You can associate one or more software items of any type with another dependent software item.

1. In Structure Manager, select each of the software items that depend on a given software and copy them to the clipboard.

2. Select the software item on which the other software types selected in step 1 depend and choose **Tools→Embedded Software Manager→Associate Software To→Software**.

   Teamcenter associates all the software items you copied to the clipboard with the software item on which the items on the clipboard depend via a **DependentOn** relation.

Establishing control unit to control unit dependency

You can manage and define dependency between control units by two mechanisms:

- You can define and manage the messages that are transmitted between two control units by using the out-of-the-box message data type that is transmitted over a connection. The message has a source defined as the source control unit and a target defined as the target control unit using the signal manager functionality.

  To identify the dependent control unit for a given control unit, you can select the control unit and use signal explorer to identify the target and source control units.

- The other form of dependency is through software. Software flashed on one control unit can depend on software flashed on another control unit.

  When a change is being made to a software on a control unit, you can select the software on that control unit in the Structure Manager and choose **View→Embedded Software Explorer→Show Associated Software→Used By** to see which software is dependent on the selected software and which other control units use the selected software.

View the processors associated with a software item

1. In Structure Manager, select the software item in the product structure.

2. Choose **View→Embedded Software Explorer→Show Associated Processor(s)→Embedding**.

   Teamcenter highlights the software item and all the processors on which the selected software is embedded.

View the software associated with a software item

1. In Structure Manager, select the software item in the product structure.

2. Choose **View→Embedded Software Explorer→Show Associated Software** and then choose **Uses** or **Used By** menu command, as appropriate.

   Teamcenter highlights the software item and all the software in which the software item is used or on which it depends.

Remove software associations from a processor

1. In Structure Manager, select the processor in the product structure.

2. Choose **Tools→Embedded Software Manager→Remove Software Association**.

   The **Remove Software Association** dialog box appears, listing all associated software items.

3. In the dialog box, select all the software items that you want to dissociate from the processor and click **Remove**.

Remove associations between software items

1. In Structure Manager, select the software item on which other software items depend in the product structure.

2. Choose **Tools→Embedded Software Manager→Remove Software Association**.

   The **Remove Software Association** dialog box appears, listing all associated software items.

3. In the dialog box, select all the software items that you want to dissociate from the software item and click **Remove**.

3. Managing Calibration and Configuration Data

Working with Calibration and Configuration Data

Overview of managing Calibration and Configuration Data

The Calibration and Configuration Data Management (CCDM) solution allows you to manage the calibration and configuration-related parameter data of embedded systems. You can define, create, view, update, and delete parameter data. You can also group related parameter definitions together and associate parameter values to a project. Using existing Teamcenter applications, such as Structure Manager, My Teamcenter, and Platform Designer, you can create calibration and configuration dictionaries and set up projects by instantiating the dictionaries. The hierarchy of parameter groups is maintained and can be used to collect the applicable parameter values in specific projects, depending on the features of the projects.

Calibration and configuration data are largely used to facilitate context-specific product deployment for different countries, different customer usage, or simply to enable product configuration and calibration through dealer and distributor services. Due to a rising complexity in products and a significant increase in the number of variants to manage, the traditional ways of managing data is no longer viable. Typically, one set of embedded software binaries is used by several variants to manage the variability of the calibration and configuration parameters.

The CCDM solution introduces artifacts to manage all the calibration and configuration data in Teamcenter. It introduces parameter definitions and allows you to group parameters and set values for them. It also uses product variants to represent a given configuration of a product, system, or component.

> Note:
>
> Stylesheet rendering view does not support CCDM objects.

You can use the CCDM solution to do the following basic functions:

- Manage parameters

  Defining your parameters is integral to managing your calibration and configuration data. A calibration and configuration parameter has several attributes that hold different information, like name, size, valid values, value descriptors, and so forth. These attributes are defined and modified so that the parameter definitions can be used in embedded software. Teamcenter allows you to create parameter definition **(ParmDef)** objects and parameter definition group **(ParmGrpDef)** objects. These objects provide definition to the parameters and help in grouping similar parameters.

- Manage parameter dictionaries

You can create a master dictionary of all groups. The master dictionary contains a hierarchical layout of all groups across various projects. It serves as a template that can be used for instantiating groups into individual projects. This ensures that definition data is carried forward into projects and the projects use values that adhere to the specification data.

You use Platform Designer to create the master dictionary in one of two ways:

- Run a custom utility that takes the groups as input.
- Create the group definitions and associated parameter definitions in My Teamcenter. You can then copy the group definitions into the master dictionary structure.

The master dictionary structure is created as a precise structure. A specific master dictionary revision is associated with specific revisions of the definition groups that are themselves precise. Consequently, a given revision of master dictionary structure only contains the revisions of those parameters that are applicable for the particular revision of the dictionary. The master dictionary hierarchy is based on the architecture breakdown.

- Manage conversion rules

  The conversion rule management option allows you to create a conversion rule and associate it to a parameter definition by defining the type, expression, and constants.

- Manage memory layout and memory blocks
- Set parameter values

  Once the product hierarchy and product context are defined, you must set the parameter values. The parameter value setting is done through the parameter value groups. You can create, revise, modify and delete parameter values for projects.

Copying and pasting a CCDM table

The copy and paste feature simplifies the initialization process when creating or modifying a parameter definition item and improves usability. It allows you to copy existing values from a single or multiple table cells and paste them into the desired cells.

You can perform the following copy and paste operations:

- Copy and paste within the same CCDM table.
- Copy from a CCDM table and paste to a different type of CCDM table. For example, you can copy values from the **min** CCDM table and paste them to the **max** CCDM table.
- Copy from a CCDM table and paste to a Microsoft Excel sheet.

- Copy from an Excel sheet and paste to a CCDM table.

  Note:

  When you are copying from an Excel sheet, consider the following points:

  - If the Excel sheet contains only the values of the table, you must hide the description field of the CCDM table before pasting. If the Excel sheet contains both values and descriptions, you must unhide the description field so that the values and descriptions can be pasted accordingly.
  - In Excel, when you copy columns (or rows) that are not adjacent and paste them in a CCDM table, all the cells between the columns (or rows) also get copied and pasted in the CCDM table. For example, if you copy and paste column 1 and column 3, column 2 also gets copied and pasted in the CCDM table.
  - In the **ParmDefSed** type table, when you copy one row from an Excel sheet, you can paste it to only one row in the **ParmDefSed** table. Only one row is pasted even if you select multiple rows for the operation.

- Copy from the **ParmDef_1 CCDM** table and paste into the **ParmDef_2 CCDM** table.
- Copy from the **ParmVal_1 CCDM** table and paste into the **ParmVal_2 CCDM** table.
- Copy from the **ParmDef_1 CCDM** table and paste into the **ParmVal_1 CCDM** table.
- Copy once and paste multiple times to these different targets.

  Note:

  When you paste cells into a selected table area that has a different size or shape (different in the number of columns and/or rows) than the copied ones, the values are rearranged to fit the target cells.

You can copy and paste values by first selecting the cells, then pressing the Ctrl+C key, and finally, the Ctrl+V key. You can also right-click the CCDM table to get the shortcut menu. The options in the shortcut menu are listed in the table below.

Menu	Description
Copy	Copies the selected cells in the table. If the **Show Value Description** check box is selected, the value as well as the description is copied. If the **Show Value Description** check box is cleared, only the value is copied.
Paste	Pastes the contents from the clipboard to the desired cells. If the **Show Value Description** check box is selected, both

Menu	Description
	the value as well as the description are pasted. If the **Show Value Description** check box is cleared, only the value is pasted.
Clear	Resets the cells to empty. If the **Show Value Description** check box is selected, it overwrites the values as well as the descriptions. If the **Show Value Description** check box is cleared, it modifies only the values. Note: For **ParmDefBool**, it changes the values to **False**.
Undo	Restores the table to the previous state, before the change. Once this function is invoked, this menu is disabled. The menu is enabled again, if you paste or make manual modifications.
Next Error	Searches for the next cell with an invalid value, starting from the current cell selection. If the search reaches the end of the table, it restarts the search from the beginning of the table. This menu is enabled when invalid values exist in the CCDM table.
Previous Error	Searches for the previous cell with an invalid value starting from the current cell selection. If the search reaches the beginning of the table, it restarts the search from the end of the table. This menu is enabled when invalid values exist in the CCDM table.

After you paste the values, check that all values are valid. The system highlights all invalid values, such as incorrect type and values out of range, less than the minimum, or more than the maximum. A notification informs you that there are invalid values and gives you the option to undo the paste operation. If you ignore the undo option, you cannot undo the paste operation at a later time and revert the values.

Note:

Sometimes the CCDM table can be very large, and the invalid cells may not be in the visible area of the table. In such cases, you should view the validation result at the top right corner, indicated by icons.

Note:

A few important things that you must keep in mind about CCDM tables:

- The CCDM table has three different validators: the type validator, table validator, and panel validator.

  The type validator pairs the parameter definition to the input type. For example, for **HEX** values, you can only enter values from 0–9 and A–F.

  The table validator is responsible for the logic validations. For example, to validate whether the initial, minimum, and maximum values comply with each other, the table must fulfill the condition min<=init<=max.

  The panel validator validates whether the CCDM tables of the parameter definition contain errors (wrong or missing values).

- You can edit the CCDM table either by double clicking or by pressing the spacebar and the F2 key.

- If you want to copy from a text file to a CCDM table, press the Tab key to separate cells and columns in the same line or row and press the Enter key to separate lines or rows.

Stretching a CCDM table

The table stretch feature improves usability and enables the CCDM table to provide enhanced support for large parameters by allowing you to stretch or collapse a CCDM table, that has the same value in all the cells. This offers a simplified initialization and editing process for parameter definition items.

You can enter the value in a particular cell of a collapsed table, and it automatically populates the same value in the entire table. You can stretch and collapse the table by selecting or clearing the **Collapse** check box.

Note:

By default, if the values in the cells are the same, the CCDM table of a parameter definition item is collapsed. An error message is displayed if you try to collapse a CCDM table that doesn't have the same values in all the cells. The CCDM table is not in the collapsed state for the same value during creation. It is only seen in the **Viewer** pane or the **Properties** page after the object is created.

The following parameter definition items are supported in the CCDM table stretch feature:

- **ParmDefBCD**
- **ParmDefHex**
- **ParmDefDbl**
- **ParmDefInt**
- **ParmDefStr**

- **ParmDefBool**
- **ParmDefDate**

The following parameter definition items are not supported in the CCDM table stretch feature:

- **ParmDefBitDef**
- **ParmDefSED**

Working with parameter definitions

Overview of working with parameter definitions

Parameter definitions provide a definition of your embedded software parameters within Teamcenter. Once created, you can manage parameters by tracing their history of modifications, perform modifications, put them through workflows, attach release status, manage and group these parameters for use in other projects and in the master dictionary, which contains the hierarchical layout of all groups across various projects.

You can create parameter definition objects for the following specific data types:

Data type	Teamcenter type
Integer	**ParmDefInt**
Double	**ParmDefDbl**
String	**ParmDefStr**
Boolean	**ParmDefBool**
Date	**ParmDefDate**
Hex	**ParmDefHex**
SED	**ParmDefSED**
BCD	**ParmDefBCD**
BitDef	**ParmDefBitDef**

Object creation support single value, simple one dimensional array values, and two dimensional array values.

Create a parameter definition

1. From My Teamcenter, choose **File→New→Parameter Management→Parameter Definition**.

The **New Parameter** dialog box appears.

2. Select the parameter type that you want to create from the list displayed.
3. Click **Next**.
4. Enter the following additional parameter information:
   - **ID**
   - **Name**
   - **Revision ID**
   - **Description** (optional)
   - **Unit of measure** (optional)
5. Click **Next**.
6. Enter the **Parameter Type** and any comment.

   Caution:

   Once you have specified the parameter type and Teamcenter has created the object, you cannot change the type.

7. Click **Next**.
8. Enter the additional parameter revision information:
   - If you are creating a parameter definition of **Integer** type, enter **Descriptor**, **Parameter Type**, **Size Units**, **Size**, **Rows**, **Columns**, minimum values, maximum values, and initial values. Optionally, you can also enter values for **isSigned**, **Resolution Numerator**, and **Resolution Denominator**.

     Teamcenter checks the values entered are valid. If invalid values are entered, an error message identifies the invalid value and the reason it is invalid.

     Note:

     For an **Integer** single valued parameter, type **1** as the value for **Rows** and **Columns**.

     For an **Integer** 1-D array valued parameter, type **3** for **Rows** and **1** for **Columns**, assuming that you are creating a 1-D array of size 3.

For an **Integer** 2-D array valued parameter, type **2** for **Rows** and **3** for **Columns**; a 2 X 3 matrix of values is required.

- If you are creating a parameter definition of type **Boolean**, enter **Size Units**, **Size**, **Rows**, **Columns**, and optionally enter the row and column headers and **Initial Values**.

  Note:

  For a **Boolean** single valued parameter, type **1** as the value for **Rows** and **Columns**.

  For a **Boolean** 1-D array valued parameter, type **1** for **Rows** and **2** for **Columns**, assuming that you are creating a 1-D array of size 3.

  For a **Boolean** 2-D array valued parameter, type **2** for **Rows** and **2** for **Columns**; a 2 X 3 matrix of values is required.

- If you are creating a parameter definition of type **BitDef**, enter values for **Parameter Descriptor** and **Size in Bytes**.

  Depending on the size of the bytes, the dialog box gets updated, to allow you to enter details like **Name** and a value for what 0 or 1 represent in each bit in the byte.

- If you are creating a parameter definition of type **Hex**, enter values for **Parameter Descriptor**, **Parameter Type**, **Size Units**, **Size**, **Rows**, **Columns**, and minimum, maximum and initial values. Optionally, you can also enter the row and column headers.

  Note:

  For a **Hex** single valued parameter, type **1** as the value for **Rows** and **Columns**.

  For a **Hex** 1-D array valued parameter, type **3** for **Rows** and **1** for **Columns**, assuming that you are creating a 1-D array of size 3.

  For a **Hex** 2-D array valued parameter, type **2** for **Rows** and **3** for **Columns**; a 2 X 3 matrix of values is required.

- If you are creating a parameter definition of type **BCD**, enter values for **Parameter Descriptor**, **Parameter Type**, **Size Units**, **Size**, **Rows**, **Columns**, and minimum, maximum and initial values. Optionally, you can also enter the row and column headers.

  Note:

  For a **BCD** single valued parameter, type **1** as the value for **Rows** and **Columns**.

  For a **BCD** 1-D array valued parameter, type **3** for **Row Label** and **1** for **Column Label**, assuming that you are creating a 1-D array of size 3.

For a **BCD** 2-D array valued parameter, type **2** for **Rows** and **3** for **Columns**; a 2 X 3 matrix of values is required.

- If you are creating a parameter definition of type **SED**, enter the **Parameter Descriptor**, **Parameter Type**, **Size Units**, **Size**, and **Rows**. Additionally, you can define valid values and initial values. For each state in the valid values, you can enter either the domain element or the value.

  Note:

  For an **SED** single valued parameter, type **1** as the value for **Row Label**.

  For an **SED** 1-D array valued parameter, type **3** for **Row Label**, assuming that you are creating a 1-D array of size 3.

- If you are creating a parameter definition of type **Date**, enter the **Parameter Descriptor**, **Parameter Type**, **Size Units**, **Size**, **Rows**, **Columns**, and minimum, maximum and initial values. Optionally, you can also enter the table descriptors and values.

  Note:

  Minimum and maximum value validation is not applied to the **Date** type parameter.

  For a **Date** single valued parameter, type **1** as the value for **Rows** and **Columns**.

  For a **Date** 1-D array valued parameter, type **3** for **Rows** and **1** for **Columns**, assuming that you are creating a 1-D array of size 3.

  For a **Date** 2-D array valued parameter, type **2** as the value for **Rows** and **3** for **Columns**; a 2 X 3 matrix of values is required.

- If you are creating a parameter definition of type **Double**, enter the **Parameter Descriptor**, **Parameter Type**, **Size Units**, **Size**, **Rows**, **Columns**, **Tolerance**, **Precision**, **Resolution Numerator**, **Resolution Denominator**, and **isSigned**. Optionally, you can also enter the table descriptors and values.

  Precision and tolerance values are used for validations in rounding scenarios.

  Note:

  For a **Double** single valued parameter, type **1** as the value for **Rows** and **Columns**.

  For a **Double** 1-D array valued parameter, type **3** for **Rows** and **1** for **Columns**, assuming that you are creating a 1-D array of size 3.

  For a **Double** 2-D array valued parameter, type **2** for **Rows** and **3** for **Columns**; a 2 X 3 matrix of values is required.

  Resolution evaluated from **Resolution Numerator** and **Resolution Denominator** is used in validating that the minimum, maximum, and initial values are entered such that:

- At least one increment based on resolution is possible between minimum and maximum.
- Initial value is an absolute increment of the resolution to the minimum value and within the specified tolerance. While evaluating tolerance the number of fractional digits used is driven based on the information provided in the **Precision** box.

- If you are creating a parameter definition of type **String**, enter the **Parameter Descriptor**, **Parameter Type**, **Size Units**, **Size**, **Rows**, **Columns**, and initial values. Optionally, you can enter the minimum and maximum value, the table descriptors and values.

  Note:

  For a **String** single valued parameter, type **1** as the value for **Rows** and **Columns**.

  For a **String** 1-D array valued parameter, type **3** for **Rows** and **1** for **Columns**, assuming that you are creating a 1-D array of size 3.

  For a **String** 2-D array valued parameter, type **2** for **Rows** and **3** for **Columns**; a 2 X 3 matrix of values is required.

9. Click **Finish**.

The parameter definition is created with the type you selected.

Save a parameter definition

1. In My Teamcenter, select the revision or the item corresponding to the parameter definition object.
2. Choose **File→Save As**.

   The **Save As** dialog box appears.

   If you select an item, you must select the revision also.
3. Enter values in the **Item ID**, **Revision ID**, and **Name** boxes.
4. Click **Finish**.

A new item of the existing parameter definition object is created. It carries forward all the information from the previous revision. Now, you can modify attributes like size, rows, column values.

Revise a parameter definition

1. In My Teamcenter, select the revision of the parameter definition object.

2. Choose **File→Revise**.

   The **Revise** dialog box appears.

3. Enter the new ID in the **Revision ID** box.

4. Click **Finish**.

A new revision of the existing parameter definition object is created. It carries forward all the information from the previous revision. Now, you can modify any attributes like size, rows, and columns values.

Edit a parameter definition

1. In My Teamcenter, select the parameter object you want to edit and choose **Properties** from the shortcut menu.

   The **Properties** dialog box appears.

2. Click **Check-Out and edit** to edit modifiable properties.

   > Note:
   >
   > You cannot perform a cancel checkout operation on CCDM objects, as work-in-progress history tracking is not enabled for CCDM objects. If you try to cancel a checkout, an error message is displayed and the CCDM object does not revert to its original value.
   >
   > To release a checkout lock, check in the CCDM object and if you need to change any previous edits, check out the object, make the desired changes, and then check in the CCDM object again.
   >
   > To track work-in-progress history, lock the previous revision, create a new revision, and modify the newly created revision.

3. Click **Yes** in the **Check-Out** dialog box.

   You can now edit the **Rows**, **Columns**, **Minimum Value**, **Maximum Value**, and **Initial Value** attributes.

   > Note:
   >
   > If you are modifying a **BitDef** type parameter definition object, you can edit the **Size in Bytes** and **Name** attributes, and the values for 0 and 1.

4. Click **Save** and **Check-In**.

Delete a parameter definition

1. In My Teamcenter, select the parameter definition object you want to delete.
2. Click **Delete** on the toolbar or choose **Edit→Delete**.

   The **Delete** dialog box appears.
3. Click **Yes** to confirm.

   The parameter definition object is deleted.

Working with the parameter definition group

Overview of working with the parameter definition group

The parameter definition group is required to group related parameter definitions and define the parameter master dictionary and template. The parameter definition group object organizes the parameters into various groups based on known usage of related parameters. However, when creating the parameter definition group, you can specify what the object represents.

Parameter groups	Purpose
Parameter Dictionary	Manages the dictionary.
Parameter Breakdown	Manages the parameter breakdown in a given project.
Organizational Group	Manages group of parameters for organizational purposes.
Packeted Parameter	Manages an ordered group of parameters.
Parameter Group	Manages a set of parameters that have value assignment together.

Create a parameter definition group

1. In My Teamcenter, choose **File→New→Parameter Management→Parameter Definition Group**.

   The **New Definition Group** dialog box appears.
2. Select the type of parameter definition group to create and click **Next**.
3. Enter values in the **Item ID**, **Revision ID**, **Name**, and **Architecture Breakdown Element Id** boxes and click **Next**.
4. Select **Parameter Group** as the **Represents** value and click **Finish**.

5. Specify the control engineer, descriptor, specialist, and comment information for the parameter definition group.

6. Click **Finish**.

Override the specialist attribute in a project

Before you the modify the specialist assignment at the project level, ensure that you have created a custom form in My Teamcenter that has attributes of the same name and type as the one that needs to be overridden from the master dictionary's definition group object.

1. Copy the form created in My Teamcenter that will override the **ParmGrpDef** attribute value from the master dictionary. This form has the **Specialist** attribute set to the specialist to whom the group's parameter value assignment task needs to be set.

2. Select the project context in Platform Designer.

3. Select the **ParmDefGrp** object for which data is to be overridden.

4. Open the **Data** panel and select the **Attachments** tab.

5. Select the top-level node, which corresponds to the **ParmGrpDefRevision** object of the selected **ParmDefGrp** object.

6. Choose **Edit→Paste Special**.

   The **Paste** dialog box appears.

7. Select the **StrObjAttrOverride** relation and click **OK**.

The form with the override information is associated to the **ParmGrpDefRevision** object in context of that project.

Save a parameter definition group

1. In My Teamcenter, select the revision or the item corresponding to the parameter definition group object.

2. Choose **File→Save As**.

   The **Save As** dialog box appears.

   If you select an item, you must select the revision also.

3. Enter values in the **Item ID**, **Revision ID**, and **Name** boxes.

4. Enter a value for the new **Architecture Element ID**.

5. Click **Finish**.

A new item of the existing parameter definition group object is created. If the **ValidateReUseInGroups** constant is configured as false for the parameter definition group, then on save as, the children from the previous revision are carried forward to the new revision.

Revise a definition group

1. In My Teamcenter, select the definition group you want to revise.

2. Choose **File→Revise**.

   The **Revise** dialog box appears.

3. Enter values in the **Name** and **Description** boxes and click **Finish**.

A new revision is created and if the group has parameter definitions, they are carried forward to the new revision.

Edit a definition group

1. In My Teamcenter, select the definition group you want to modify and choose **Properties** from the shortcut menu.

   The **Properties** dialog box appears.

2. Click **Check-Out and edit**.

3. Click **Yes** in the **Check-Out** dialog box.

   You can now make changes to the modifiable attributes.

> Note:
>
> You cannot perform a cancel checkout operation on CCDM objects, as work-in-progress history tracking is not enabled for CCDM objects. If you try to cancel a checkout, an error message displays and the CCDM object does not revert to its original value.
>
> To release a checkout lock, check in the CCDM object and, if you need to change any previous edits, check out the object, make the desired changes, and then check in the CCDM object again.
>
> To track work-in-progress history, lock the previous revision, create a new revision, and modify the newly created revision.

4. Click **Save** and **Check-In**.

Add a parameter definition to a definition group

1. In My Teamcenter, select one or more parameter definitions to be added.

   > Note:
   >
   > A parameter can be added to one group only. If you associate a parameter to a group, you cannot associate the same parameter to another group.

2. Choose **Edit→Copy**.

3. Select the definition group to which you want to add the parameter definitions and choose **Send To→Structure Manager** from the shortcut menu.

4. In Structure Manager, select the top line and choose **Edit→Paste**.

5. Click **Save**.

   The parameter definitions are added to the definition group.

If a parameter is already used in another item of the same type, it cannot be added if the **ValidateReUseInGroups** constant of the parent type is set to **true**.

The parameter is added to the group as a precise occurrence.

Remove a parameter definition from a definition group

1. In Structure Manager, open the parameter definition group from which a parameter definition needs to be removed.

2. Select the parameter definition you want to remove.

3. Choose **Edit→Remove**.

4. Click **Save**.

   The parameter definition is removed from the definition group.

Resequence parameter definition within a definition group

1. In Structure Manager, open the parameter definition group in which you want to resequence parameter definitions.

2. Select the parameter definition you want to resequence.

3. Double-click the find number corresponding to the selected parameter definition and enter a new find number.

> Note:
>
> If the selected parameter definition must be sequenced ahead of another reference parameter definition object, enter a find number less than the find number of the reference parameter definition.
>
> If the selected parameter definition must be sequenced after another reference parameter definition object, enter a find number greater than the find number of the reference parameter definition.

4. Click **Save**.

   The parameter definitions are resequenced within the definition group.

Delete a parameter definition group

1. In My Teamcenter, select the parameter definition group object you want to delete

2. Click **Delete** on the toolbar or choose **Edit→Delete**.

   The **Delete** dialog box appears.

3. Click **Yes** to confirm.

   The parameter definition group object is deleted.

Working with a master dictionary

Overview of working with a master dictionary

You can create a master dictionary of all groups. The master dictionary contains a hierarchical layout of all groups across various projects. It serves as a template, which can be used for instantiating groups into individual projects. This ensures that the definition data is carried forward into projects and the projects use values that adhere to the specification data.

Create a master dictionary

1. In My Teamcenter, select the folder where you want to place the master dictionary.

2. Choose **File→New→Parameter Management→Parameter Definition Group**.

   The **Parameter Definition Group** dialog box appears.

3. Select **ParmGrpDef** as the parameter definition group type and click **Next**.

4. Specify the values for **Architecture Element ID** and **Name** for the master dictionary and click **Next**.

5. Set the architecture breakdown properties and click **Next**.

6. Select **Parameter Dictionary** as the value in the **Represents** box and click **Finish**.

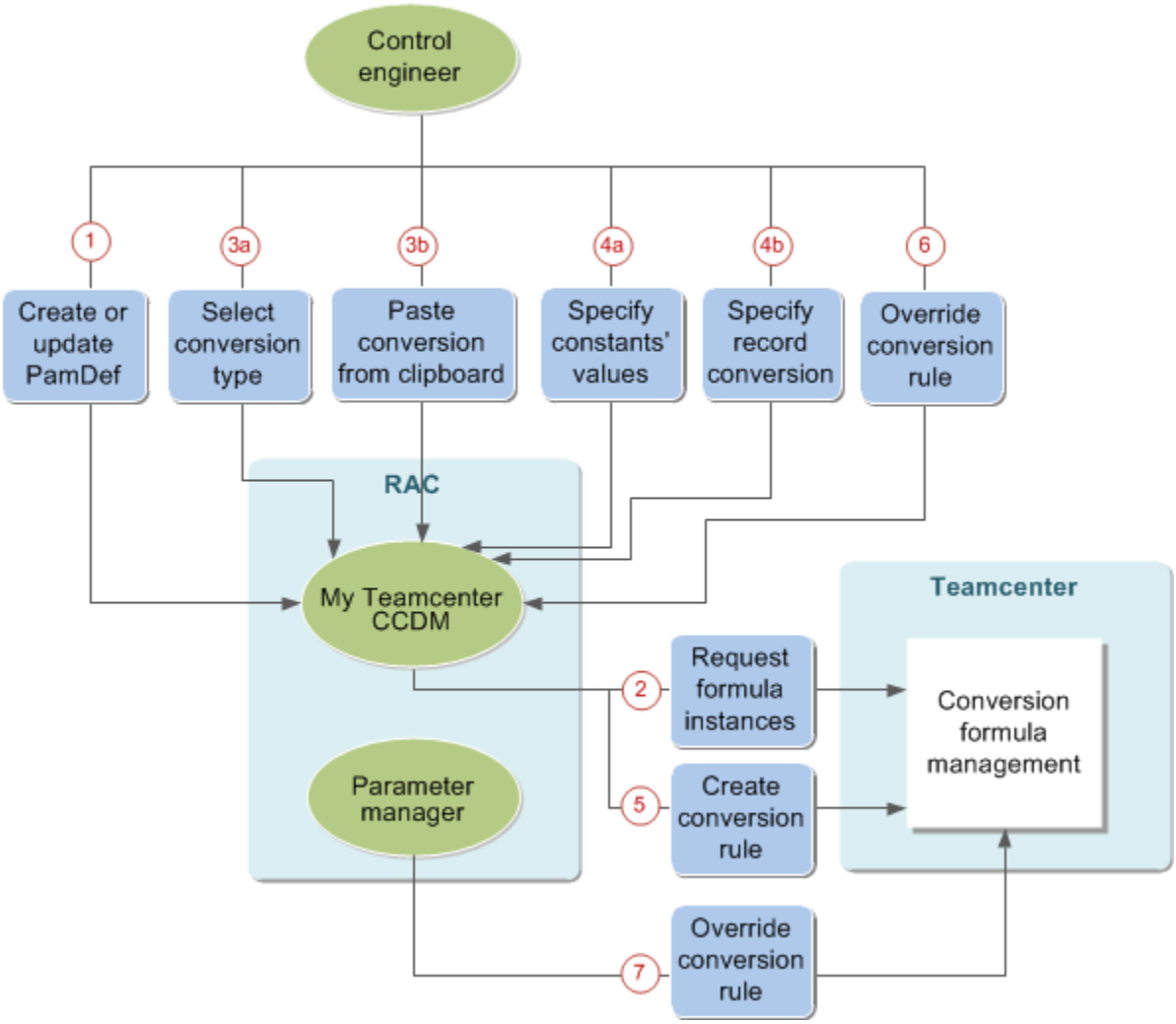
An empty master dictionary is created and placed by default in the **New Stuff** folder in My Teamcenter. You can add the group definition revisions, which have been already created, to the master dictionary. You can copy the group definition revisions from My Teamcenter and paste them into the master dictionary in Structure Manager.

> Note:
>
> It is recommended that you do not add non-parameter definition objects under **ParmGrpDef**. You can customize this by adding values which are either **ParmDef** or its sub-class to the **TCAllowedChildTypes_<ParmGrpDef or SubTypeName>** preference.

Managing conversion rules

Overview of managing conversion rules

A conversion rule consists of specific algebraic formula, a list of constant values, and a list of variables. It allows the system to convert the parameter values into ECU values automatically. According to the input, the conversion rule evaluates the formula and returns a computed result. The conversion rule management option allows you to create a conversion rule and associate it to a parameter definition by defining the type, expression, and constants. You can modify, copy, paste, override, and delete the conversion rule. The various steps in conversion rule management are depicted in the following figure.



There are four conversion types provided out of the box in Teamcenter:

Name	Expression	Constant Names	Variable Names
Identical	**x**	**{}**	**{x}**
Linear	**A*x+B**	**{A, B}**	**{x}**
Quadratic	**A*x*x+B*x+C**	**{A, B, C}**	**{x}**
Rational	**(A*x*x+B*x+C)/ (D*x*x+E*x+F)**	**{A, B, C, D, E, F}**	**{x}**

Define a conversion rule for parameter definition

1. Create a parameter definition and click **Next** after you have entered the additional parameters.

   The **Define Conversion Rule** dialog box appears.

2. In the **Name** box, enter a name for the conversion rule.

3. In the **Type** box, select the conversion type.

4. In the **Description** box, enter a brief description of the conversion rule.

5. In the **Constants** box, specify the constant value.

   Specify a number in decimal format for each constant. The value must be between -2^31 and 2^31–1.

   > Note:
   >
   > If you select the conversion type as **Rational**, then **D**, **E**, and **F** should not be 0.

6. Click **Finish**.

Modify a conversion rule associated to a parameter definition

1. Modify the parameter definition as described in *Edit a parameter definition*.

2. In the **Name** box, enter a new name for the conversion rule.

3. In the **Type** box, change the conversion type.

   For example, from **Linear** to **Rational**.

4. In the **Description** box, enter a brief description of the conversion rule.

5. In the **Constants** box, specify the constants value for the formula.

6. Click **Save** and **Check-In**.

Delete a conversion rule

1. Modify the parameter definition.

2. In the **Type** box, select **Blank**.

   The conversion rule associated with the parameter definition is deleted. The parameter definition does not have any conversion rule.

3. Click **Save** and **Check-In**.

Copy and paste a conversion rule to a parameter definition

1. In My Teamcenter, open the **Properties** dialog of the parameter definition you want to copy.
2. In the **Define Conversion Rule** pane, click the context menu and select **Copy** from the list displayed.
3. Create a parameter definition and click **Next** after you have entered the additional parameters.

   The **Define Conversion Rule** dialog box appears.
4. Click the context menu and select **Paste** from the list displayed.
5. Click **Finish**.

> Note:
>
> Similarly, you can copy a conversion rule and paste it when you are editing a parameter definition.
>
> Also, when you copy and paste a conversion rule to a parameter definition, the system checks whether the copied conversion rule can apply to the parameter type. If not, the system notifies that the conversion type is not matched. A single conversion rule cannot be shared by different parameter definitions.

Managing memory layout or memory block

Overview of managing memory layout or memory block

The memory management feature allows you to create memory layouts and blocks and assign parameters to them using the Software Parameter Manager. Once you have the memory blocks, you can edit them by assigning and removing parameters. You can also delete and copy blocks and override addresses.

Create a memory layout and add a memory block

1. In My Teamcenter, select the dictionary or project for which you want to create a memory layout and choose **Send To→Software Parameter Manager** from the shortcut menu.

   The Software Parameter Manager appears with the dictionary or project and its memory layout.
2. Select the memory layout and choose **Add→Memory Block** from the shortcut menu.

   A new memory block is created and added in the tree. Next, assign parameters to this block.

View, assign, and remove parameters from a memory block or memory layout

1. In the Software Parameter Manager, select the memory block or memory layout you want to view or assign parameters to.

2. View any assigned parameters in the **Assigned Parameters** tab in the right pane.

   In context of a dictionary, there is just one group (group header is hidden) in the assigned parameter view-**Parameter Definition**. In context of a project, there are two groups (group header is visible) in the assigned parameter view-**Parameter Definition** and **Override**.

   Under the **Parameter Definition** group, you can view the following parameter:

   - **Hierarchy** displays the hierarchy of the corresponding memory layout or block.
   - **Object** displays the assigned parameter name.
   - **Conversion Rule** represents the formula expression of the associated conversion rule for the assigned parameter.
   - **Minimum Values** represents the parameter minimal value if any.
   - **Maximum Values** represents the parameter maximal value if any.
   - **Initial Values** represents the parameter initial value if any.
   - **Table Definition** represents the parameter table definition if any.
   - **Valid Values** represents the **SED** valid values.
   - **Initial Value** represents the **SED** initial value.
   - **Bit Definition** represents the **BitDef** definition.
   - **Start Address** displays the start address of the corresponding memory layout/block.

     > Note:
     >
     > This is the absolute address. Start address of a memory layout equals to its start address. The start address for a memory block equals to the memory block offset address plus the memory layout start address.

   - **Offset Address** represents the offset address for the assigned parameter.

   Under the **Override** group, you can view the following parameters:

- **Revision** displays the corresponding override container revision id if any.
- **Offset Address** displays the overridden offset address if any for the assigned parameter.
- **Conversion Rule** represents the formula expression of the overridden conversion rule for the assigned parameter.

3. You can modify the columns and save the new configuration by right-clicking a column and choosing **Column** from the shortcut menu.

   The **Column Chooser** dialog appears for you to change the column configuration.

4. Click **Done** after making the changes and then click **Save Column Configuration** to save this configuration for future.

   To change the new column configuration, click **Reset Column Configuration** and reset the columns to their default configuration.

5. After viewing the parameter, you can assign more parameters by selecting one or more parameters from the **Available Parameters** tab.

6. Click Assign.

   The selected parameters appear in the **Assigned Parameter** tab.

7. To remove a parameter, select the parameter in the **Assigned Parameter** tab and click **Unassign**.

   This deletes the parameter records and memory override records, but keeps the formula override record.

8. Click **Save**.

Edit memory layout or memory blocks

1. In the Software Parameter Manager, check out the memory layout you want to edit by choosing **Check-In/Out→Check Out** from the shortcut menu.

2. If you are editing a memory layout revision, click the **Viewer** tab and edit these properties.

   - Enter values in the **Start Address** and **Mirrored Offset** boxes in **Hex** format.
   - Enter values in the **Input Sources** box. This represents the list of parameter files (datasets) imported into this memory layout revision.
   - Enter values in the **Output Files** box. This represents the list of parameter files (datasets) exported from this memory layout revision.

- Click **Check In**.

3. If you are editing a memory block revision, select the memory block, click the **Viewer** tab and edit these properties.

   - Enter a value in the **Start Address** box in **Hex** format.

   - Click **Check In**.

Copy a memory block or memory layout

1. In My Teamcenter, select the source dictionary whose memory block or layout you want to copy and choose **Send To→Software Parameter Manager** from the shortcut menu.

   The dictionary along with its memory layout and blocks appears in the Software Parameter Manager.

2. In My Teamcenter, select the target dictionary to which you want to copy the memory block to and choose **Send To→Software Parameter Manager** from the shortcut menu.

   The dictionary along with its memory layout and blocks appears in the Software Parameter Manager. You can see it below the source dictionary.

3. Navigate in the source dictionary memory layout and select the block you want to copy.

4. Choose **Copy** from the shortcut menu.

5. Select the target dictionary memory layout and select the correct parent.

6. Choose **Paste** from the shortcut menu.

   The memory block along with its sub-blocks and parameter assignments are copied.

> Note:
>
> You cannot copy a memory layout and paste it under another memory layout.

Delete a memory block or memory layout

1. In the Software Parameter Manager, select the memory block or layout you want to delete.

2. Choose **Delete** from the shortcut menu.

   This deletes the block or layout and its sub-blocks and unassigns any assigned parameters.

> Note:
>
> Even after the deletion, the parameters still remain in the database. If there are any overrides, they are not deleted as the overrides could be authored by someone else.

Override address for parameter assignments

1. Select the memory block in which you want to override the parameter address.

   The parameter details appear on the right pane under the **Parameters** tab.

2. Click **Offset Address** in the **Override** group, to make the box editable.

3. Enter the new offset address and click **Save**.

   The first revision of the override is created and you can see this under **Override Containers** on the left. This container contains all override-related information. You can select the override and view the override record.

Override conversion rule

1. In My Teamcenter, select the project for which you want to override the conversion rule and choose **Send To→Software Parameter Manager** from the shortcut menu.

   The Software Parameter Manager appears with the project.

2. Check out the container revision if it already exists.

3. In the **Assigned Parameters** tab, find the row in the table where the parameter is located for which you want to override the conversion rule.

4. Double-click the **Conversion Rule** cell you want to override.

   The **Define Conversion Rule** dialog box appears.

5. Override the conversion rule by selecting the value to override and click **OK**.

6. Click **Save** and check in the container.

> Note:
>
> If the conversion rule being overridden is for an unassigned parameter, start by finding the parameter in the **Available Parameter** box.

Managing parameter configuration and value setting

Overview of managing parameter configuration and value setting

Parameter value setting is an integral part of managing your calibration and configuration data. You can start by creating project-specific breakdown and then specifying the values for each parameter.

Project-specific breakdown of parameters helps you to create an architectural breakdown of your project at the project setup process and provides placeholders for parameter values. Parameter values can be set for all parameters in a group by creating a parameter value group object. You can also revise and modify the parameter value group to change the values of parameters. You also can attach the parameter value group object to the definition group in the project specific breakdown using the add part to product functionality.

Working with parameter configuration and value setting

Parameter value setting is an integral part of managing your calibration and configuration data. You can start by creating project-specific breakdown and then specifying the values for each parameter.

Project-specific breakdown of parameters helps you to create an architectural breakdown of your project at the project setup process and provides placeholders for parameter values. Parameter values can be set for all parameters in a group by creating a parameter value group object. You can also revise and modify the parameter value group to change the values of parameters. You also can attach the parameter value group object to the definition group in the project specific breakdown using the add part to product functionality.

Working with product variants

Overview of working with product variants

Product variants represent a given configuration of a product. It is part of the product planning process and exists before you create the actual objects in building the product. It comprises the configuration details of the product. The final configured product is based on the product variant configurations.

Teamcenter allows you to create product variants and product variant intent objects. When you create a product variant object, you specify the product context and the configuration. When you create a product variant intent object, you specify the product context intent for which you are creating a product variant intent. The product variant captures the context and configuration of the product and serves as a container for grouping together the various product variant intents.

Technically, the **ProductVariant** object is a subclass of the **CCObject** object, which is used in Teamcenter to group together various structure context objects that reference product structures. The **ProductVariantIntent** object is associated with the actual architecture breakdown structures. Therefore, data stored against the **ProductVariantIntent** objects imply that data is stored against that particular architecture breakdown structure.

The **ProductVariantIntent** object is a subclass of the **StructureContext** object that references the architecture breakdown structure. The **ProductVariantIntent** objects are grouped together by the **ProductVariant** object.

To use **ProductVariant** and **ProductVariantIntent** objects in data exchange using PLM XML, the product context, configuration context, and the product context intent must already exist in the environment prior to importing a PLM XML file, which has **ProductVariant** and **ProductVariantIntent** objects.

Before doing any of the following procedures, ensure you have created the product, set variability, created NVEs, and added the relevant parts to the product.

> Note:
>
> Both, the **ProductVariant** and the **ProductVariantIntent** objects are not extensible.

Create a product variant

1. In My Teamcenter, select the product context that you want to use as the basis of the product variant.

2. Create a configuration context object for the product context:

   a. From My Teamcenter, choose **File→New→Configuration Context**.

      The **New ConfigContext** dialog box is displayed.

   b. Type values in the **Name** and **Description** boxes.

   c. Select a value in the **Revision Rule**, **Variant Rule**, and **Closure Rule** lists.

   d. Click **OK**.

3. In My Teamcenter, select the product context for which you want to create a product variant and copy it to the clipboard.

4. Choose **File→New→Product Variant**.

   The **New Product Variant** dialog box is displayed.

5. Type values in the **Name** and **Description** boxes.

6. Select **Paste** from the product context options.

7. Click **Search** next to the **Configuration Context** box to search for the configuration context you created for the selected product context.

8. Click **OK**.

Save a product variant

1. In My Teamcenter, select the revision or the item corresponding to the product variant object.
2. Choose **File→Save As**.

   The **Save As** dialog box appears.

   If you select an item, you must select the revision also.
3. Enter values in the **Item ID**, **Revision ID**, and **Name** boxes.
4. Enter a value for the new **Architecture Element ID**.
5. Click **Finish**.

A new item of the existing product variant is created. However, the product variant intent associated with the product variant is not carried forward to the new item. This is because one product variant intent cannot be used in multiple product variants.

Edit a product variant

1. In My Teamcenter, select the product variant you want to edit and from the shortcut menu, choose **Properties**.

   The **Properties** dialog box is displayed.
2. Click **Check-out and Edit** to edit modifiable properties.
3. Click **Yes** to confirm.

   You can modify the name and configuration context properties.
4. Click **Save and Check in**.

Note:

You cannot perform a cancel checkout operation on CCDM objects, as work-in-progress history tracking is not enabled for CCDM objects. If you try to cancel a checkout, an error message displays and the CCDM object does not revert to its original value.

To release a checkout lock, check in the CCDM object and, if you need to change any previous edits, check out the object, make the desired changes, and then check in the CCDM object again.

To track work-in-progress history, lock the previous revision, create a new revision, and modify the newly created revision.

Create a product variant intent

1. In My Teamcenter, select the product variant for which you want to create the product variant intent.
2. Choose **File→New→Product VariantIntent**.

   The **New Product Intent** dialog box is displayed.
3. Select **ProductVariantIntent** in the left pane.
4. Type values in the **Name** and **Description** boxes.
5. Select a value from the **Product Context Intent** list.
6. Click **OK**.

The product variant intent is created and associated with the product variant.

Edit a product variant intent

1. In My Teamcenter, select the product variant intent you want to modify and from the shortcut menu, choose **Properties**.

   The **Properties** dialog box is displayed.
2. Click **Check-out and Edit** to edit modifiable properties.
3. Click **Yes** to confirm.

   You can modify the product context intent property.
4. Click **Save and Check in**.

Note:

You cannot perform a cancel checkout operation on CCDM objects, as work-in-progress history tracking is not enabled for CCDM objects. If you try to cancel a checkout, an error message displays and the CCDM object does not revert to its original value.

To release a checkout lock, check in the CCDM object and, if you need to change any previous edits, check out the object, make the desired changes, and then check in the CCDM object again.

To track work-in-progress history, lock the previous revision, create a new revision, and modify the newly created revision.

Create a project-specific breakdown

1. In Platform Designer, select the product context corresponding to which the project-specific breakdown needs to be created.
2. Choose **File→New→Architecture Breakdown**.
3. Select **ParmGrpDef** and click **Next**.
4. Specify values for **Name** and **Description**, and click **Assign**.
5. Set the value of **Represents** to **Parameter Breakdown** and click **Next**.
6. Select the empty project-specific breakdown and click the split window button at the top-right corner.

   This splits the window in two parts with the project-specific breakdown in one pane.
7. Select the empty pane and search for the master dictionary that needs to be used for instantiating the project breakdown.

   You see the project-specific breakdown in the left pane and the master dictionary in the right.
8. Move either all or only the required group definitions under the master dictionary to the clipboard, so that they can be instantiated into the project-specific breakdown.
9. Select the project specific breakdown and click **Paste** on the toolbar to paste the selected groups into the project specific breakdown.

   The **Paste** dialog box is displayed.
10. Select **Process Parents** to copy the complete parent structure.

    Select **Process Children** to copy all the children of the selected element.

A project-specific breakdown is created, and the groups chosen by you are now a part of the project-specific breakdown.

Note:

Generic breakdown is not supported for parameter breakdown.

Create a parameter value group

1. In My Teamcenter, choose **File→New→Parameter Management→Parameter Value Group**.

   The **New Parameter Value Group** dialog box appears.

2. Select the type of group value and click **Next**.

3. Enter values in the **Name**, **ID**, and **Revision** boxes and click **Next**.

4. (Optional) Enter any additional information for parameter value and click **Next**.

5. (Optional) Enter any additional information for parameter value group revision and click **Next**.

6. Search for the project for which value needs to be defined.

   The search results are displayed.

7. Select the project and click **Next**.

8. Choose the category for the product by selecting options under **Architecture** and **Revision Rule**.

9. Click **Architecture Element ID**.

   The element is displayed.

10. Double-click the element to select it and click **Next**.

11. Select the parameter group definition revision from the breakdown and click **Next**.

    The **Parameter Table** with all the parameters associated with the selected parameter group definition is displayed.

12. Select a row specific to the parameter and double click the **Actual Value** cell. You can select only one row and specify values for only one parameter at a given time.

    The **Enter Actual Values for Parameter** dialog box appears, which shows user interface controls in context of each data type, that you specified earlier as row and column descriptors.

13. Enter the values for each parameter and click **OK**.

    The values get updated in the **Actual Value** cell as follows:

    - Single values for all single values attributes.

- Comma-separated values for all simple array attributes, for example, 1,2,3,4.
- Combination of commas and brackets for 2D array attributes, for example, {1,2,3}, {4,5,6}, {7,8,9}. The row is represented by values within the brackets.
- A concatenated bit stream is displayed for the **BitMap** type.

14. Click **Finish**.

A parameter value group is created and values are set for each parameter.

> Note:
>
> You can avoid steps 6 through 11 by using a custom constructor.

Save a parameter value group

1. In My Teamcenter, select the revision or the item corresponding to the parameter value group object.
2. Choose **File→Save As**.

   The **Save As** dialog box appears.

   If you select an item, you must select the revision also.
3. Enter values in the **Item ID**, **Revision ID**, and **Name** boxes.
4. Enter a value for the new **Architecture Element ID**.
5. Click **Finish**.

Revise a parameter value group

1. In My Teamcenter, select the parameter value group revision that you want to revise and choose **File→Revise**.

   The **Revise** dialog box appears.
2. Type a value in the **Revision ID** box and click **Finish**.

A new revision is created and all the parameter value objects associated with the old parameter group value revision are copied to new objects. These new objects are associated to the new revision of the parameter group value. You can now modify the parameter values.

Edit a parameter value group

1. In My Teamcenter, select the parameter value group object that you want to edit and choose **Properties** from the shortcut menu.

   The **Properties** dialog box appears.

2. Click **Check-Out and Edit** to edit the modifiable properties.

   The **CheckOut** dialog box appears.

3. Click **Yes** to confirm.

   > Note:
   >
   > You cannot perform a cancel checkout operation on CCDM objects, as work-in-progress history tracking is not enabled for CCDM objects. If you try to cancel a checkout, an error message displays and the CCDM object does not revert to its original value.
   >
   > To release a checkout lock, check in the CCDM object and, if you need to change any previous edits, check out the object, make the desired changes, and then check in the CCDM object again.
   >
   > To track work-in-progress history, lock the previous revision, create a new revision, and modify the newly created revision.

4. Double-click the **Actual Value** cell to modify the parameter values you want to change.

5. Click **Check-In** and **Save**.

Delete a parameter value group

1. In My Teamcenter, select the parameter value group object that you want to delete.

2. Click **Delete** on the toolbar or alternatively, choose **Edit→Delete**.

   The **Delete** dialog box appears.

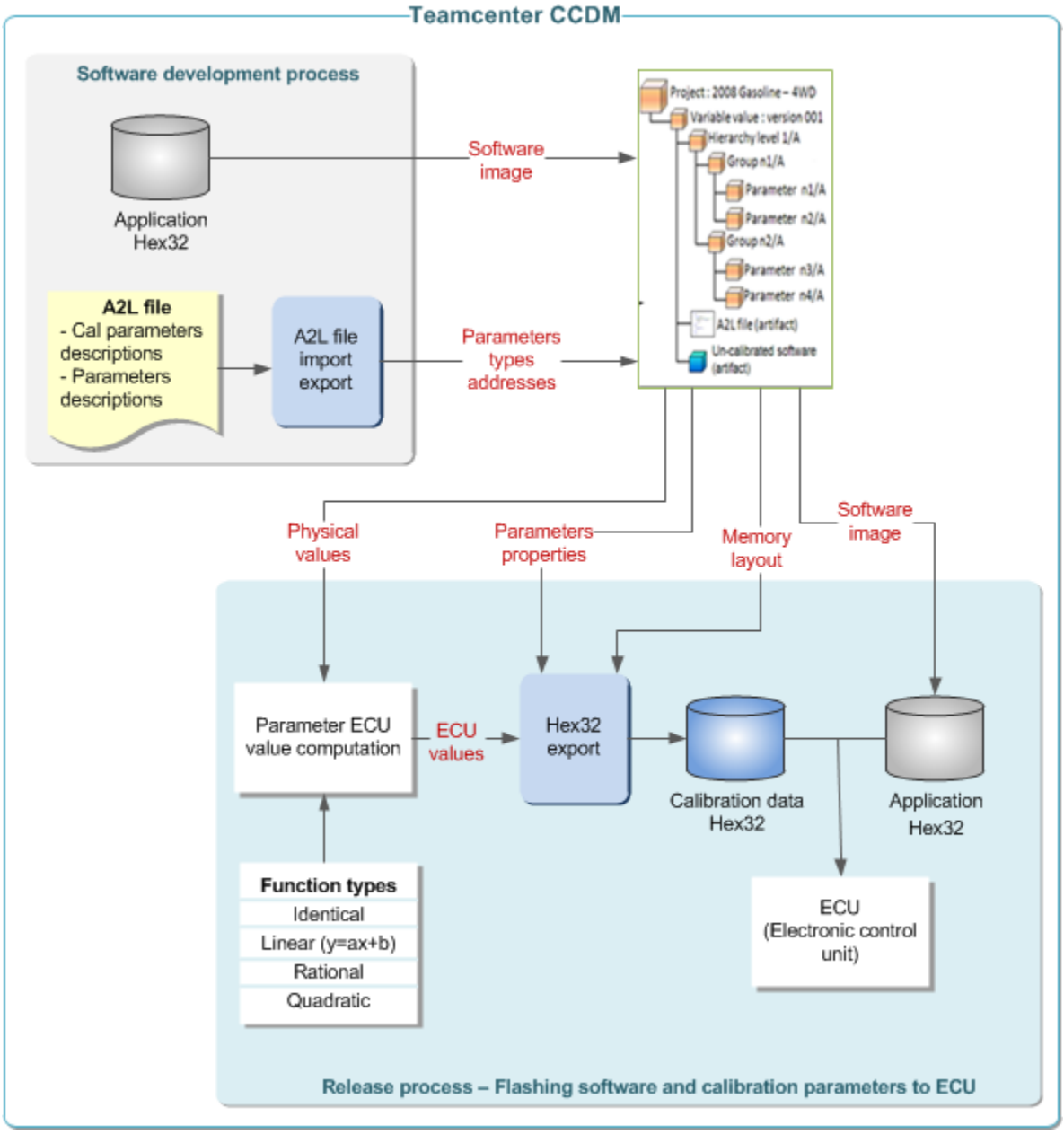
3. Click **Yes** to confirm.

The selected parameter value group object is deleted.

Exporting and importing CCDM data

Overview of exporting and importing CCDM data

The CCDM feature provides export and import capabilities for exchanging parameter definition and value data. The export and import operations are supported for standard file formats like Hex32, CDF, and A2L. It supports export of parameter values and address data for generation of flash files. Teamcenter defines extension points for generating specific flash files. You can implement extensions to generate the standard flash files based on the data in Teamcenter. The following figure captures the export and import process:



Import of data happens through standard files. The import operation supports import of memory layout details and import of conversion rule details. Teamcenter defines extension points for parsing standard files. You can implement extensions to parse the standard files and generate data structures. The data from the generated data structures is updated in the Teamcenter database.

You can export and import CCDM data using the Software Parameter Manager. It allows you to choose the required options, export parameter values from a project, and import portion of parameter definitions in a project, dictionary, memory layout, or override container.

All error/warnings of the import and export operations are logged in a log file. It is generated in the temporary location of the system example *c:\temp*. You can view this log file after the operation is complete.

Export CCDM data

1. In Software Parameter Manager, select the project or product variant intent you want to export and choose **Tools→Export CCDM Data**.

   The **Parameter Export** dialog box appears.

2. Select the memory layout you want to export.

3. Select the override revision.

   > Note:
   >
   > The **Override Container** is populated based on the revisions of the override item that is associated to the selected project.

4. Select the file type.

   > Note:
   >
   > The file type is populated based on those sub-classes of **Ccd0ParmFile** where the **Ccd0FileTypeExchangeSupported** business object constant is set to either **Both** or **Export**.

5. Browse and select the file path where you want to save the exported file.

6. Enter a prefix for the file name.

   > Note:
   >
   > The file name would be the prefix followed by the memory layout name and then the file extension based on the file type you selected. If the long id preference is set, then the file name should not be more than 128 characters. Else, only 32 characters are allowed.

7. Select the **Save Dataset** check box.

   If you select this, the dataset corresponding to the files generated for each selected memory layout is created and associated to the selected project using a specific relation type. One dataset will be created for each memory layout. The dataset will be visible in Software Parameter Manager perspective and associated with its memory layout in the tree.

8. Select the **Overwrite Existing Data** check box.

   If you select this, the system checks whether another dataset exists with the same name. If a dataset with the same name exists, then it overwrites the contents. If this check box is not selected, the system will give you the option to either overwrite or rename.

9. Select the **Show Error Log** check box

   If you select this, the system displays the error log after the operation.

10. Click **Next** if the form options for the selected file type is set to true and **Next** is enabled. Else, click **Finish** to perform the export operation.

11. Enter the form options in the **Export Options** dialog and click **Finish**.

> Note:
>
> You see this dialog only if you had clicked **Next** in the previous step.

Import CCDM data

1. In Software Parameter Manager, select the project, dictionary, memory layout, or override container you want to import data into and choose **Tools→Import CCDM Data**.

   The **Parameter Import** dialog box appears.

2. Select the type of file you want to import.

   The options listed are auto-populated based on those sub-classes of **Ccd0ParmFile** where the **Ccd0FileTypeExchangeSupported** business object constant is set to either **Both** or **Import**.

3. Browse and select the data file to import.

> Note:
>
> When you click **Browse**, you can only select files of those extensions that correspond to the named references of the selected file type.

4. Select the **Save Dataset** check box to display the text file where you can specify the dataset name.

   Note:

   If the **Save Dataset** check box is selected, the dataset corresponding to the specified file is created and associated to the selected object with a specific relation type. If the long id preference is set, then the file name should not be more than 128 characters. Else, only 32 characters are allowed.

5. Select the **Proceed on Error** check box to continue with import even if there is an error.

6. Select the **Show Error Log** check box to display the error log after the operation is complete.

7. Select the **Overwrite Existing Data** check box to override existing data.

8. Click **Click here to Validate Import Data** if the file type you chose did not have any associated form. Then click **Finish** .

   Else, click **Next** for the **Import Options** dialog box.

   Note:

   The **Click here to Validate Import Data** button is visible only if the file type you chose did not have any associated form. This also displays any errors.

9. In the **Import Options** dialog, enter the form options.

10. Click **Click here to Validate Import Data** to validate the import data and click **Finish** to import it.

4. Managing the embedded software implementation process

Overview of managing the embedded software implementation process

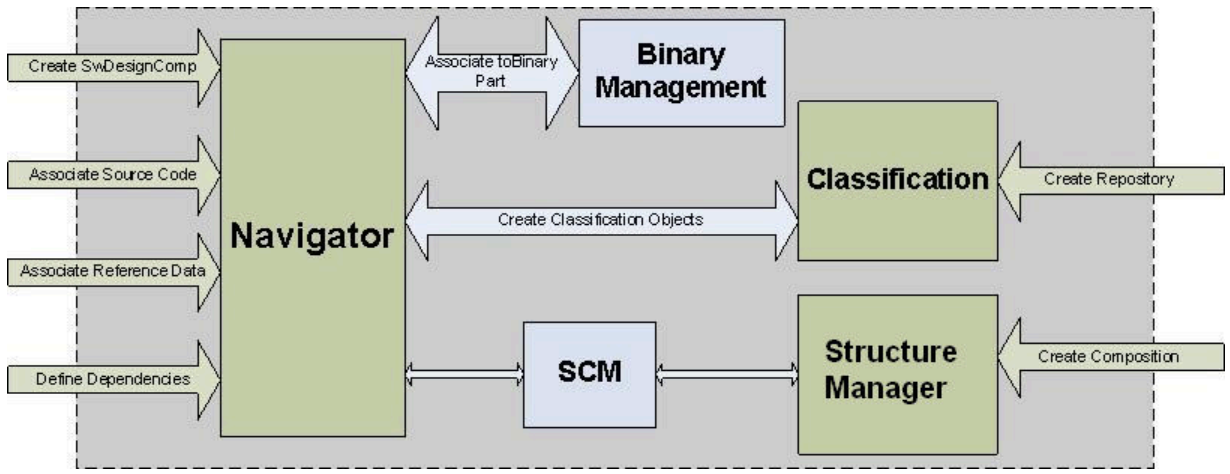
Various industries are fast moving toward a component-based embedded software system development approach. The Embedded Software Design Data Management solution bridges the current industry gaps in the usage of the embedded software components. An embedded software component is a self-contained entity that can be used as a building block in the design of a larger system. It can be used an arbitrary number of times within the same system as well as in different systems.

Use the Embedded Software Design Data Management solution to:

- Manage design data (source code, models, object files, third-party libraries and specifications) of embedded software components.
- Create, update, and delete software design data components.
- Define dependencies and associate specifications, source code, and test cases to software design data components.
- Launch an efficient search and comparison of components based on attributes.

Using this solution, designers can define the composition framework of the embedded software component and developers can define the build dependencies of the embedded software component. The user can link the binary to source code and achieve the binary to source code traceability.

The following figure depicts the functions of the Embedded Software Design Component Management solution in Teamcenter.



Create software design component

The design component is the building block and helps to specify the source code, object files, or libraries, which are used to generate the software binary.

1. In My Teamcenter, choose **File→New→Software Design Component**.

   The **New Software Design Component** dialog box appears.

2. Select the **SoftwareDesignComp** option and click **Next**.

3. Enter a name, description, and ID for the new software design component. The **Object ID** box is populated by default.

4. Click **Finish**.

Associate source code to software design component

1. In My Teamcenter, select the Teamcenter SCM object and choose **Edit→Copy**.

2. Select the revision of the software design component object and choose **Edit→Paste**.

   Because the default paste relation is defined, the SCM object gets associated with the **Specification** relation.

   Alternatively, choose **Edit→Paste Special** and select the **Specification** relationship.

Associate reference data to software design component

You can associate reference data in the form of design documents, test documents, or some other reference data to the software design component using the **References** relation.

1. In My Teamcenter, select the reference object and choose **Edit→Copy**.

2. Select the software design component object and choose **Edit→Paste**.

   The **Paste** dialog box appears.

3. Choose the **References** relationship from the list and click **OK**.

Define dependency

You can define dependencies for building a software component to generate the corresponding binary.

1. In My Teamcenter, select the required software design component object and choose **Edit→Copy**.
2. Select the dependent software design component object and choose **Edit→Paste**.

   The **Paste** dialog box appears.
3. Choose the **Dependent On** relationship from the list and click **OK**.

View existing build dependencies of software design component

- In My Teamcenter, go to the software design component folder in the navigation tree and expand the folder to see the build dependencies.

Remove build dependencies of software design component

1. In My Teamcenter, go to the software design component folder in the navigation tree and expand the folder to see all the build dependencies.
2. Select the dependencies you want to remove.
3. Choose **Edit→Cut**.

Modifying software design component

You can modify the associations, source code, or the reference data by making modifications to the relation object that links the software design component to the reference data or the source code.

Revise software design component

You can create a new revision of the software design component.

1. In My Teamcenter, select the software design component you want to revise and choose **File→Revise**.

   The **Revise** dialog box appears.
2. Make the desired modifications and click **Finish** to generate a new revision of the software design component.

Deleting software design component

You can delete the software design component along with its relations, source code, and reference data.

Link software design component to binary

You can link the software design component to the corresponding software binary or binaries using the **SCM_Element_specification** relationship. Linking these two objects helps to track the source code through the software design component of the corresponding binary.

1. In My Teamcenter, select the software design component object and choose **Edit→Copy**.

2. Select the application software object and choose **Edit→Paste**.

   The **Paste** dialog box appears.

3. Choose the **SCM_Element_Specification** relationship from the list and click **OK**.

Creating a repository of embedded software component

You can create a repository of software design components using the Classification Admin application and then classify it in the Classification application. The repository helps to categorize the existing software design components by the domain, groups, or classes of embedded software components. The process of creating a hierarchy and classifying objects is described here.

1. Ensure that you have added the **SwDesignComp** and **SwDesignCompRevision** objects to the **ICS_classifiable_types** preference, so that the objects get qualified for classification.

2. In Classification Admin, create a hierarchy by defining groups and subgroups as needed.

3. In the attributes dictionary, create a new attribute that classifies the Teamcenter object.

4. Create a class and associate the attribute to this class.

5. Select a Teamcenter object of item type and drag it to the Classification application.

   If there is no corresponding **ICO**, the system prompts you and creates it.

5. Managing ClearCase Integration

Overview of managing ClearCase Integration

ClearCase Integration enables you to incorporate software components created under ClearCase into complex product configurations comprising multiple mechanical, electrical, and software design elements in Teamcenter. By integrating into a single Teamcenter-managed PLM environment, you can configure, track, and control all hardware and software components that compose your product and their related product/process definitions. You can manage versions of software files in ClearCase and use Teamcenter for product modeling, workflow, and process management. Teamcenter manages the artifacts for ClearCase objects but does not act as a parallel vaulting system for ClearCase files. All files are in ClearCase.

Use the rich client to manage the product data of the hardware part of the product and use ClearCase to manage the product data of the software part of the product. With ClearCase Integration, you can access ClearCase information from Teamcenter and query the ClearCase views and versioned object bases (VOB). ClearCase Integration allows you to check out, check in, and create a workflow process.

Understanding ClearCase Integration

Basic concepts about ClearCase Integration

The benefits of ClearCase Integration, how to manage ClearCase users, and connectivity of ClearCase with Teamcenter are explained here.

Benefits of ClearCase Integration

- Enables you to establish a single source of product information that accounts for a product's software design definitions in addition to its mechanical and electrical design definitions.
- Enables cross-discipline teams to participate in an integrated design environment that accounts for the entire product configuration as it evolves through its life cycle.
- Allows software developers to perform their software configuration management tasks in their ClearCase environment and their integrated design tasks in an enterprise PLM environment.
- Allows software development teams to manage traceability to the source code from the software binary part of a product. This is particularly helpful in assessing the impact of changes and identifying which source code needs to be changed for a particular issue.
- Allows software development teams to share their most stable ClearCase data with the rest of the stakeholders in the product life cycle.

- Helps to accelerate the product life cycle, improve quality, and reduce cost by allowing software developers to understand the impact of software design as early in the product life cycle as possible.
- Enables enterprises to tie product requirements into the software design cycle, thereby allowing software development teams to understand a product's quality definitions and thereby, design in quality and design out defects.
- Ensures that the latest product decisions (such as design changes and their related work impacts) are communicated and fully understood by every stakeholder in the product life cycle.

ClearCase and users

ClearCase does not maintain a database of its users. Any user who is logged on to a ClearCase host and can acquire a license can use the software. ClearCase relies on a server host's operating system to establish a user's identity. It is important that the administrator ensures that user identities are consistent on every ClearCase host. A user's identity is defined by the following attributes:

- User ID
- Principal group ID
- Supplementary group IDs (optional)

You can achieve this consistency by using the network-wide databases maintained by the operating system, such as the Windows NT or Active Directory domains on Microsoft Windows.

Teamcenter and ClearCase connection

When you install ClearCase Integration, connectivity is established between Teamcenter and ClearCase. When you create or modify Teamcenter objects that represent ClearCase data, this connectivity keeps the two systems synchronized.

The synchronized processes run in the background without user intervention and make the following checks:

- ClearCase server is accessible.
- ClearCase licenses are available.

What is SCMVersionObject?

SCMVersionObject is a Teamcenter object created to represent versioned objects in an SCM system. In the case of ClearCase Integration, **SCMVersionObject** is used to represent single or multiple ClearCase element versions. For example, for a file element called **abc.txt** in ClearCase, **SCMVersionObject** can be created in Teamcenter representing a specific version of **abc.txt**.

The following table shows the GRM relationship that can exist between **SCMVersionObject** and a business object. It defines the operations and relations that can or cannot exist between a business object and **SCMVersionObject**. For example, you can create **SCMVersionObject** in a folder but not in a dataset.

Business object	**Operations**				**Relations**	
	Create	Edit	Delete	Cut, copy, and paste	Associate	Dissociate
Item	Y	Y	Y	Y	**SCM_Element_Specification**	Remove relation
Item Revision	Y	Y	Y	Y	**SCM_Element_Specification**	Remove relation
SwDesign Comp Revision	Y	Y	Y	Y	**IMAN_Specification**	Remove relation
Folder	Y	Y	Y	Y	Contents	Remove relation
Dataset	N	N	N	N	-	-
Form	N	N	N	N	-	-
GDE	N	N	N	N	-	-

Note:

Deletion of a Teamcenter object that is associated with **SCMVersionObject** does not delete the **SCMVersionObject**. The operation deletes the primary object after removing the relation with the version object.

Basic tasks using ClearCase Integration

To manage ClearCase Integration, you can perform a series of tasks. Typically, these are the basic tasks:

- Verify connectivity and create Teamcenter users of ClearCase data.
- Register user's default view.
- Create **SCMVersionObject**.
- Check out and check in **SCMVersionObject**.
- Create a workflow process for ClearCase data.

Teamcenter ClearCase Integration and Multi-Site

Teamcenter ClearCase Integration supports Multi-Site import and export. The **SCMVersionObject** is the primary object, which gets exported to a remote site during a Multi-Site operation.

Before you work with Multi-Site in a ClearCase Integration environment, ensure the following:

- The value of **SCMVersionObject** is present in these preferences:
  - **TC_publishable_classes**
  - **TC_directly_transferable_classes**
- The **SCM_element_specification** is added to the **TC_relation_export** preference.
- If the ClearCase clients at remote sites are configured for different ClearCase servers, then the **SCM_ClearCase_Server** preference is configured to include the remote site's ClearCase server and the ClearCase sites must be in sync.

All basic Teamcenter operations such as checking in and out and cancelling checkout along with checking in and out and cancelling checkout of ClearCase elements are supported if you have done the transfer of ownership during the export operation.

During an export, when the Item or ItemRevision to which the **SCMVersionObject** is associated through **SCM_Element_specification** relation is exported to the remote site, the **SCMVersionObject** is also exported.

After export, you are able to view the ClearCase Integration information directly on the Teamcenter **SCMVersionObject** at the remote site. Along with other properties, you can view these compound properties:

- SCM server
- SCM element path
- SCM configuration rule
- SCM server type

> Note:
>
> You cannot view the SCM configuration rule at the replica site because **Browse** is unavailable. To view the SCM configuration rule, open the **Properties** dialog box. Scroll bars are displayed when the SCM configuration rule text exceeds the text area space.

Administering ClearCase Integration

Overview of administering ClearCase Integration

As an administrator, after you install ClearCase Integration, you must check the connectivity between ClearCase and Teamcenter, set preferences, and create users.

Verify connectivity with ClearCase

1. Open the Organization application.

2. Choose **Tools→Verify ClearCase Connectivity**.

   If the connection is established, a success message appears. If the connection is not established, Teamcenter displays an error message.

   > Note:
   >
   > The error message could be due to one of the following reasons:
   >
   > - You have not entered the ClearCase server host name correctly.
   >
   >   Ensure that the value entered for the SCM ClearCase server during installation is correct.
   >
   > - The configured ClearCase server user does not match with the ClearCase server to which Teamcenter is configured.
   >
   >   Ensure that the value of the **SCM_ClearCase_Server** preference is populated and correct. The ClearCase server the user is configured to must match with value of the **SCM_ClearCase_Server** preference.
   >
   > - ClearCase environment is not set up correctly.
   >
   >   Ensure that the ClearCase client environment is set up correctly on the system from which the Teamcenter rich client is run. To test, open a command window/shell and invoke **Cleartool hostinfo –l** command. If errors occur, contact your system administrator.
   >
   > To diagnose a connection failure or license issues, you can run ClearCase diagnostics, as described in the ClearCase documentation.

3. If the connectivity check is successful, in My Teamcenter, choose **Edit→User Settings→SCM Workspace** to verify that you have access to the necessary ClearCase views.

Setting preferences

As an administrator, you can set the following ClearCase preferences to suit the requirements at your site:

- **SCM_ClearCase_Server**

  This preference establishes and verifies the connection with the ClearCase server. You can configure multiple ClearCase servers by adding them to this preference.

- **SCM_CheckIn_Identical_Version**

  This preference determines whether Teamcenter runs the SCM checkin command when the checked-out version is identical to the previous version of the element. You can set this to true or false.

- **SCM_ClearCase_UNCO_Keep_Version**

  This preference determines whether .keep files are preserved in ClearCase when you use the uncheckout command. You can set this to true or false.

Create Teamcenter users of ClearCase data

1. Open the Organization application.
2. Click **Users**.
3. Define each Teamcenter user by typing the values in each of the following boxes:
   - **Person Name**
   - **User ID**
   - **OS Name**

     This entry must correspond exactly to the user's operating system name in ClearCase. This allows the user to perform ClearCase actions from Teamcenter.

     Optionally, you can also define a password, a default group, and volume for the user.
4. Click **Create**.

Using ClearCase Integration

Overview of using ClearCase Integration

After the administrator has verified connectivity with ClearCase and set the preferences, as a Teamcenter user, you can register your view and start working with ClearCase Integration. You can create a **SCMVersionObject** and check it out and check it in. You can also undo the checkout and create a workflow process for ClearCase data.

Register your default view

To access data stored in ClearCase versioned object bases (VOBs), you must select the appropriate ClearCase view for yourself. This view is your default view and is used to browse the ClearCase elements.

1. Open My Teamcenter.
2. Choose **Edit→User Setting**.

   The **User Settings** dialog box appears.
3. Click the **SCM Workspace** tab.

   The **Select Default View** pane appears.

   You can choose to see only your views, by selecting the **Show my views only** check box.
4. Choose the view you want to designate as the default view for yourself.
5. Click **OK**.

   Teamcenter checks for the following:

   - The view exists.
   - The view is accessible from your workstation.
   - The view is started on your workstation. If not, it starts the view on your workstation.

The ClearCase view tag is stored as a user preference in Teamcenter to allow quick access to the view information in subsequent sessions.

Create SCMVersionObject

To create **SCMVersionObject**, you must associate the Teamcenter object with either a single ClearCase element or multiple ClearCase elements.

1. Open My Teamcenter.
2. Choose **File→New→SCMVersionObject**.

   The New SCMObject wizard appears.
3. In the **Name** box, type the name of the ClearCase element you want to associate with the Teamcenter object.

4. Choose the **Navigate the VOBS** or **Use Existing Element Object** option.

   - Choose **Navigate the VOBS** to browse and select all the elements you want to associate with the Teamcenter object. You can select multiple SCM elements at one time to add. All selected elements appear in the **Elements** list.

   - Choose **Use Existing Element Object** to select the existing element that you want to associate with the Teamcenter object. In the **Name** box, type the name of the existing element. For a wildcard search, type *****. A list of all existing elements displays for you to choose from.

5. Select **Use Selected View** to choose an element from the current view.

   To select an element from another view, choose the **Use Different View** option and select a view from the list of views displayed.

6. Click **Next**.

   The **Select Configuration** pane appears.

7. Select the configuration by choosing one of the following:

   - The current version of the ClearCase element by clicking **Select Current View Configuration**.

   - The label by clicking **Select Label** and selecting the required label.

   - The branch, for example, **LATEST**, by clicking **Select Branch** and selecting the required branch.

   - The precise version by clicking **Select Version** and selecting the required version.

     Note:

     The **Select Version** option does not appear if you have associated the Teamcenter object with multiple ClearCase elements.

8. Click **Finish**.

Teamcenter creates **SCMVersionObject** by associating the Teamcenter object with the selected ClearCase elements. **SCMVersionObject** holds the ClearCase element name, kind, and configuration context.

Check out SCMVersionObject

You can check out and place a lock only on a view-based **SCMVersionObject**. This gives you exclusive rights to modify the associated data.

1. Open My Teamcenter.

2. Select the **SCMVersionObject** and choose **Tools→Checkin/Checkout→Checkout**.

   The **CheckOut** dialog box appears.

3. In the **Change ID** box, type a value and (optionally) any comments.

   > Note:
   >
   > You can also select the **Check out ClearCase Element as well** check box. If you select this box, you check out both the Teamcenter object and the associated ClearCase element. If you do not select this box, you check out only the Teamcenter object. If you check out the associated ClearCase element, the Teamcenter object uses the current view and its information to access ClearCase and check out the ClearCase element.

4. Click **Yes**.

   > Note:
   >
   > You can verify the checkout by using one of the following ways:
   >
   > - Run the ClearCase explorer for the view on which you are performing ClearCase operations and note the checkout symbol next to the file you checked out.
   > - Open ClearCase command prompt and set the view on which you are working. Run the **lsco** operation on the file you checked out.

Check in SCMVersionObject

After you have finished modifying the **SCMVersionObject**, you can check it back in to ClearCase.

1. Open My Teamcenter.

2. Select the **SCMVersionObject** and choose **Tools→Checkin/Checkout→Checkin**.

   The **CheckIn** dialog box appears.

3. Click **Yes**.

   Teamcenter checks in the Teamcenter object associated with the ClearCase element.

   If the object you are trying to check is identical to the previous version of the object on the server, then the checkin fails and an error appears. In such a case, you should cancel the checkout.

   See the SCM_CheckIn_Identical_Version preference that determines identical checkin behavior.

Note:

Teamcenter uses the ClearCase view (workspace) to access the checked-out ClearCase element and then to check it in to ClearCase. This places the information from the view storage area into the VOB storage area; that is, from a temporary storage into the permanent VOB storage. You can delete the workspace (view) if there are no more ClearCase objects remaining to work on. Thus, the information carried by the Teamcenter object reflects the new location of the ClearCase object.

Note:

If you have checked out the **SCMVersionObject** from Teamcenter and checked in Clearcase Element from the ClearCase explorer, ensure that you check in from Teamcenter, else the **SCMVersionObject** will be shown as checked out in Teamcenter.

Undo the checkout of SCMVersionObject

You can undo a checkout operation if you want the data to return to its previous state. This cancels any changes that you made to the data.

Caution:

If you cancel a checkout operation, any changes you made to the associated data are lost.

1. Open My Teamcenter.
2. Select the **SCMVersionObject**.
3. Choose **Tools→Checkin/Checkout→Cancel CheckOut**.

   The **Cancel CheckOut** dialog box appears.
4. Click **Yes**.

Teamcenter cancels the checkout and data reverts to its previous state. It also restores the original information about the Teamcenter object associated with the ClearCase object.

Create a workflow process for ClearCase data

You can create a workflow to apply a label to the ClearCase elements associated with **SCMVersionObject**. You can use the **SCMApplyLabel Task** template to create a process for any **SCMVersionObject** whose new configuration on completion of a task is a label.

> Note:
>
> Steps 1–10 are administrative tasks, and you must be an administrator user to perform them. The steps after that are regular user actions and any user having ClearCase access can perform them.

Log on as an administrator and complete the following steps:

1. Open the Workflow Designer application.
2. Choose **File→New Root Template**.

   The **New Root Template** dialog box appears.
3. In the **New Root Template Name** box, type **SCMProcess**.
4. Choose **Process** as the template type.
5. Click **OK**.

   In the **Workflow Designer** pane, the **Process Template** list displays the template name.
6. Choose **Edit→Template→SCMApplyLabel Task**.

   The **SCMApplyLabel Task** template appears in Workflow Designer.
7. Link **Start** and **SCMApplyLabel Task** by clicking the **Start** task and dragging it to **SCMApplyLabel Task**.
8. Link **SCMApplyLabel Task** and **Finish** by clicking **SCMApplyLabel Task** and dragging it to the **Finish** task.
9. Select the **Set to Available Stage Template** option.

   A confirmation window appears.
10. Click **Yes**.

Log on as a Teamcenter user and complete the following steps:

1. Open My Teamcenter.
2. Select the **SCMVersionObject**.
3. Choose **File→New→Process**.

The **New Process** dialog box appears.

4. In the **Process Name** box, type a value.
5. From the **Process Template** list, choose **SCMProcess**.
6. Click **OK**.
7. Go to your inbox and expand the **Tasks to Perform** folder.
8. Select the **SCMVersionObject**.
9. Choose **Actions→Perform**.

   The **Perform SCMApplyLabel Task** dialog box appears.

10. Select **Task Instruction** and click **Done**.
11. Click **OK** to perform the task and apply the label.

6. Using binary management

Binary management

You can manage binary with the Software Binary Management Solution by creating the software revision and uploading it to Teamcenter and making it available for flashing. Once in Teamcenter, the binaries may be placed under change control and their history tracked. They can be reused and associated to products, where they can be configured for specific feature combinations.

Create binary software revisions

1. Choose **File→New→Item** and select the appropriate type of software:

   - **AppSoftware**

     Creates an application software item. This type represents the software that implements the actual functionality.

   - **Calibration**

     Creates a calibration software item. This type represents the software that calibrates the application software.

   - **Config File**

     Creates a configuration file software item. This type represents the configuration information on which the application software depends.

   - **PriBootLoader**

     Creates a primary bootstrap loader software item. This type represents the bootloader software of the control unit.

   - **SecBootLoader**

     Creates a secondary bootstrap loader software item. This type represents the software that makes the control unit flashable.

2. Click **Next**. The **New Item** dialog box for the selected software type appears.

3. Enter the identifier and description of the new software item and click **OK** or **Apply**. Teamcenter creates a new software item of the selected type.

Optionally, you can click **Next** again, rather than **OK** or **Apply** to define the master form attributes and assign the software item to a project.

Mapping released binary to a specification and design components

A design engineer uses the data in Teamcenter to generate a software specification that is sent to an internal development group or external supplier to create the necessary binary. The general work process is as follows:

- A design engineer creates a software specification for the software to be flashed to the control unit.
- The design engineer sends the specification to an internal development group or external supplier.
- The development group or supplier develops the software in compliance with established standards.
- The development group or supplier validates the release binary by executing a preconfigured file format checker. Optionally, the binary may be converted to other formats. This step must succeed before the binary is released.
- The development group or supplier creates a release binary and associates it with the software design component in Teamcenter as a named reference. The source code is managed in the source control management (SCM) system and may also be associated with the same software design component.

To allow file format checkers to verify software binaries, you create a dataset and attach the software files to it. As a prerequisite for importing the dataset, you must configure validation of the extension rule for the software binary dataset type; likewise you must configure the conversion extension rule on the post action.

Upload binary data into Teamcenter

1. Ensure that the administrator has configured Teamcenter by creating a dataset type that can be used for managing binary data.
2. Select the software item type revision for which data needs to be managed.
3. Choose **File→New→Dataset**.

   The **New Dataset** dialog box appears.
4. Fill in the required information and select the file corresponding to the selected software item revision you want to upload into Teamcenter.
5. Click **Finish**.

   A new dataset of the selected type is created and the file chosen is uploaded into Teamcenter. The dataset gets associated to the software item revision.

Making software available for flashing

The released binary is downloaded by the flashing application so that the flashing tool can store the binary on the control unit. Sometimes the control unit may be preconfigured and preflashed with the release binary, in which case this step is not necessary. Production personnel or a service technician enters the software part number and revision into the flashing application. This process downloads the binary file for flashing onto the control unit.

> Note:
>
> The user interface for capturing technician inputs is written in the flashing application or any application that is used by the technician. This application has an Application Interface (AI) service client, which interacts with Teamcenter and downloads the binary corresponding to the information passed in the user interface.
>
> For more information about the AI Service, see the *Services Reference* in the Teamcenter HTML Help.
>
> To access the *Services Reference*, install the Teamcenter developer references when you install Teamcenter online help, or go to https://support.sw.siemens.com.

7. Managing control units

Overview of managing control units

After you have created your control units, you search for the hardware and software for the control unit and associate them. You can also generate reports on the compatibility of the hardware and software and finally associate the control unit to the product configuration.

Create a control unit

1. Choose **File→New→Item**.

   The **New Item** dialog box appears.

2. Select **Part** as the item type or select an item type that represents a part at your site.

Identify and associate hardware and software to control units

1. Search for the hardware and software relevant to the control unit you created.

2. Select the hardware and software and copy them to the control unit.

Querying hardware software compatibility

Overview of querying hardware software compatibility

During the life of a product, service engineers may need to change the hardware or the software of the product. If such a change is made, it is necessary to determine that the replacement item is compatible with the existing parts. If the software is changed, the existing hardware must be compatible with the new version of the software. Similarly, if a hardware is changed, the new hardware should be compatible with the existing software, else a new software must be flashed.

To check such things, in Teamcenter you can generate compatibility reports and use them to:

- Identify the compatible software.

- Identify the compatible hardware.

Embedded Software Solutions allow you to generate compatibility reports in HTML format. Default stylesheets are supplied with Teamcenter. You can overwrite the default stylesheets with your own custom stylesheet templates.

To create an HTML report, select the **hw_sw_html_report_template.xsl** template file.

You can extend the content of the report to include more details by configuring the transfer modes used for the generation of reports. If a transfer mode is used to display more information in the report, you should also update the stylesheet to translate the additional information correctly.

The following preferences must be updated when changing the transfer mode and the stylesheet:

- **ESM_HW_Compatibility_Report_TransferMode**
- **ESM_SW_Compatibility_Report_TransferMode**
- **ESM_HW_Compatibility_Report_HTML_StyleSheet_Dataset**
- **ESM_SW_Compatibility_Report_HTML_StyleSheet_Dataset**

Create a report of compatible software

For a defined hardware identifier (for example, a control unit hardware part number), you can search for all the compatible software parts and assemblies.

1. In My Teamcenter, choose **Tools→Embedded Software Manager→Compatibility Report→Hardware-Software**.

   The **Software Compatibility Report** dialog box appears.

2. Enter the item identifier and revision of the hardware item and click **Next**.

3. Select the HTML report format and click **Finish**.

   Teamcenter generates the requested report.

4. Select the table and use the shortcut menu to export this report to Excel. Alternatively, you can copy and paste the table into Excel.

Associating control units to product configuration

After the designer defines the software part in Teamcenter, the engineer can register it for a given product configuration with the Add Part to Product wizard in Platform Designer. You can configure a given control unit to a particular variant and add the control unit variant as a part to the product. Alternatively, you can add individual components such as hardware or software to the product.

8. Customizing solutions and exchanging software data

Overview of customizing solutions and exchanging software data

Embedded software item types are transferred when you export or import structure data in PLM XML format. Using ITK functions, you can customize Embedded Software Solutions.

Exchanging embedded software information

When you export or import structure data in PLM XML format, the following Teamcenter Mechatronics Process Management and embedded software object types are also transferred:

- **Connection** (revisable)
- **GDELink** (nonrevisable connection)
- **Signal**
- **Message**
- **Connection_Terminal** (GDE)
- **Network_Port** (GDE)
- **ProcessVariable** (GDE)
- **Processor** (item)
- **Software**
- **SwDesignComp**
- **SwDesignCompRevision**
- **Parameter Definition** and subclasses.
- **Parameter Definition Group**
- **Parameter Value Group**

When you import or export structure data, CCDM parameter definitions, parameter definition groups, and parameter value groups are transferred.

For exporting product variant and product variant intent objects, use the **MRMAssemblyExport** transfer mode. Add the following property set to the **MRMAssemblyExport** transfer mode:

CLASS ProductVariant PROPERTY *productCtxt* **DO**

Adding this property set ensures that the *productCtxt* attribute is exported out as user data.

Table cells along with cells and definition information are exported as user data elements when property set is added for them.

Customizing Embedded Software Manager

The embedded software manager module provides the following ITK functions that allow you to customize Embedded Software Manager.

- **ESM_is_processor**

  For a BOM line tag, this function checks whether it is a processor BOM line.

- **ESM_is_gateway**

  For a processor BOM line tag, this function checks whether it is a gateway processor BOM line.

- **ESM_is_software**

  For a BOM line tag, this function checks whether it is a software BOM line.

- **ESM_associate_processor_to_software**

  For a processor BOM line and an array of software BOM lines, this method associates a processor and a software with the **Embeds** relation. To save the associations, the BOM window must be saved.

- **ESM_associate_processor_to_processor**

  For a gateway processor BOM line, and an array of processor BOM lines, this method associates one processor to another with the **GatewayOf** relation.

- **ESM_associate_software_to_software**

  For a software BOM line and an array of software BOM lines, this method associates software with the **Dependent On** relation. To save the associations, the BOM window must be saved.

- **ESM_remove_processor_to_software_association**

  For a processor BOM line and an array of software BOM lines, this method removes the **Embeds** relation association between processor and software lines.

- **ESM_remove_processor_to_processor_association**

  For a gateway processor BOM line and an array of associated processor BOM lines, this method removes the **GatewayOf** relation association between the processor lines.

- **ESM_remove_software_to_software_association**

  For a software BOM line and an array of software BOM lines, this method removes the **Dependent On** relation association with the software lines.

- **ESM_ask_embedded_software_of_processor**

  For a processor BOM line, this function gets an array of software BOM lines that are associated to the processor with an **Embeds** relation.

- **ESM_ask_gateway_of_processor**

  For a processor BOM line, this function gets an array of gateway processor BOM lines that are associated as the gateway with the input processor line.

- **ESM_ask_processors_accessedby_processor**

  For a gateway processor BOM line, this function gets an array of processor BOM lines that are accessed through the input gateway.

- **ESM_ask_dependent_software_of_software**

  For a primary software BOM line, this function gets an array of (secondary) software BOM lines that are dependent on the primary software.

- **ESM_ask_software_used_by_software**

  For a (secondary) software BOM line, this function gets an array of (primary) software BOM lines that are used by the secondary software.

In addition, the SIG module provides the following ITK functions that allow you to customize the operations that can be performed on frames, signals, and relations:

- **SIG_ask_signal_source**

  Finds all the sources of a signal or message line tag using the **associated_system** relation.

- **SIG_ask_signal_target**

  Finds all the targets of a signal or message line tag using the **associated_system** relation.

- **SIG_ask_signal_transmitters**

  Finds all the transmitters of a signal or message line tag using the **associated_system** relation.

- **SIG_ask_device_sources**

  Finds all the source devices that transmit messages or signals to the input target device. This function uses the underlying **associated_system** relation between the source and the message or signal.

- **SIG_ask_device_targets**

Finds all the target devices to which the device transmits messages or signals. This function uses the underlying **associated_system** relation between the target and the message or signal.

- **SIG_ask_device_ transmitted_signals**

  Finds all the frames or signals that are transmitted by a source device. This function uses the underlying **associated_system** relation between the source and the message or signal.

- **SIG_ask_device_ received_signals**

  Finds all the messages or signals that are received by a target device. This function uses the underlying **associated_system** relation between the target and the message or signal.

Using a custom constructor for parameter value group creation

You can use the custom command constructor to instantiate and create a new parameter value group object wizard.

When you create the parameter value group, you determine the associated parameter group definition revision for which the group value object has to be created, by following steps 6 through 11 in *Create a parameter value group*. But, if you already have the parameter group definition revision object reference, you can use the custom constructor for creating a parameter group value object.

In the custom code, from where you invoke the group value object creation wizard, use the following command class constructor instead of the default constructor.

- Constructor Name

  NewParamValGrpCommand

- Parameters

  Frame– reference to the parent frame.

  AbstractAIFApplication– reference to the AIF application.

  TCComponentArchitectureRevision– reference to the parameter group definition revision object.

Customizing for CCDM export and import

Overview of customizing for CCDM export and import

The dataset **Ccd0ParmFile** provides only the out of the box functionality to process the CCDM data and provide data for export. Likewise, it only provides out of the box functionality to import the data given in the specified data structure.

For export, the custom dataset has to override the required method **ccd0generateFile** on **Ccd0ParmFile** dataset object to have its own implementation for generating the required file format. The data for export is provided in **ParmFlashGenerationExpData** data structure. Please refer to the Teamcenter root folder *includes\Ccd0ExpImp.h*.

For import, the custom dataset has to override the required method **ccd0Parse** to parse the input file and populate the data in the data structure and the out of the box functionality imports the data. The data for import is in **DictionaryParseOutput** data structure. Please refer to the Teamcenter root folder *includes\Ccd0ExpImp.h*.

Customize CCDM export and import

1. In Business Modeler IDE, create a custom dataset of type **Ccd0ParmFile**.

2. Provide the **FileTypeExchange** supported in the **Ccd0FileTypeExchangeSupported** Business Modeler IDE constant. Choose from**Export**, **Import**, **Both**, or **None**.

3. Provide the required information about the optional form to be used for export in Business Modeler IDE constant **Ccd0ExportOptions**.

4. Provide the required information about the optional form to be used for import in Business Modeler IDE constant **Ccd0ExportOptions**.

5. Override the required operations for the custom dataset based on the **FileTypeExchange** supported.

6. Deploy the changes to the database.

7. Provide the custom implementation for the overridden methods.

8. Build the custom **dll**.