TEAMCENTER

Mechatronics Process Management

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

Getting Started

Getting started 1-1
Overview of Mechatronics Process Management 1-1
Mechatronics Process Management applications 1-1
Mechatronics Process Management related applications 1-2

Basic concepts

What is Mechatronics Process Management? 2-1
Mechatronics Process Management data model 2-2
- About Mechatronics Process Management data model 2-2
- Functional model 2-2
- Physical model 2-2
- Logical model 2-2

Mechatronics Process Management design integrations 2-3
Fundamental objects you work with 2-3
STEP AP212 and KBL data models 2-5
Working with allocations 2-7
- About allocations 2-7
- Analyzing allocations 2-8
- Creating allocations 2-9
- Configuring allocations 2-10
- Releasing allocations 2-10

Basic tasks

Creating an electrical functional model 3-1
- Creating an electrical functional model 3-1
- Create functions 3-2
- Create function composition 3-2
- Managing information exchange between functions 3-2

Creating an electrical logical model 3-4
Creating an electrical physical model 3-5
Building views 3-6
Working with wiring harnesses 3-7
- Typical integration process of electrical routing application, ECAD application, and Teamcenter 3-7
- Identifying the top-level product 3-8
- Identifying the publishing information 3-8
- Separating electrical connectivity data and mechanical information 3-10
- Assigning electrical components to mechanical connectors 3-10

Defining a route 3-10
Create allocations 3-10
Working with data dictionary 3-11

About data dictionary 3-11
Data dictionary basic tasks 3-11
Data dictionary functional flow 3-11
Interacting with external applications 3-13
Interacting with external applications overview 3-13
Updating schematics with changes made to connectivity data 3-15
Completing the electrical product structure with routes 3-15
Restructuring a Mechatronics structure 3-16
Restructuring a Mechatronics structure 3-16
Insert a level in a Mechatronics structure 3-19
Remove a level from a Mechatronics structure 3-19
Move a node in a mechatronics structure 3-20
Replace a node in a Mechatronics structure 3-22
Split an occurrence in a Mechatronics structure 3-24
Fix in-structure associations 3-24
Tracking edits to a mechatronics structure 3-25
Working with Teamcenter ClearCase Integration 3-26
Managing embedded software 3-27
Administering PLM XML data transfers 3-27
PLM XML data transfers overview 3-27
Export electromechanical structure 3-29
Viewing ECAD design 3-29
Viewing ECAD design 3-29
View a PCB file 3-29
Marking up ECAD design 3-29

Mechatronics Process Management applications

Mechatronics Process Management applications and integrations 4-1
Wiring Harness Design Tools Integration 4-1
Embedded Software Solutions 4-2
Structure Manager 4-3
Multi-Structure Manager 4-4
My Teamcenter 4-4

Mechatronics Process Management data model

About Mechatronics Process Management data model 5-1
Using the abstract classes 5-1
Representing Mechatronics Process Management components 5-3
Representing components 5-3
Representing electrical components 5-3
Representing signals and process variables 5-4
Representing electrical interfaces 5-4
Representing electrical connections 5-5
Representing routes 5-5
Representing allocations 5-7
Modeling relationships 5-8

Multidisciplinary Associations

Collaborating across disciplines using Multidisciplinary Associations 6-1
Setting up and using multidisciplinary associations: participants and roles 6-2
Installing the Multidisciplinary Associations feature 6-3
How to install the Multidisciplinary Associations feature 6-3
Installation of the Multidisciplinary Associations feature 6-5
Administering and configuring the Multidisciplinary Associations objects 6-5
Controlling access to Multidisciplinary Associations objects 6-5
Customizing the client to view MDThreads 6-8
Associate component types other than library elements, items, and item revisions 6-8
Creating and managing Multidisciplinary Associations 6-8
Linking design components using MDThread linking 6-8
Create an association between design components 6-10
Search for MDThreads 6-11
Delete an MDThread 6-11
Edit MDThreads 6-11
Linking design instances 6-12
Linking component usages using MDInstance linking 6-12
What happens when a linked product structure assembly model object undergoes changes? 6-14
Working with notifications in Multidisciplinary Associations 6-15
How notifications work 6-15
Configure a site for notifications 6-15
Create a new multidisciplinary collaboration project 6-16
Subscribing to events for notifications 6-17
Querying for notifications 6-17
Responding to notifications 6-17
Deleting notifications 6-17
Working with Multidisciplinary Associations in a multisite environment 6-18
Exchanging multidisciplinary data with remote sites 6-18
Exchange multidisciplinary data through Multi-Site Collaboration 6-18
Exchange multidisciplinary data using TcXML 6-19
Exchange multidisciplinary data using a Briefcase 6-19
Associate domain information with design components 6-19
Teamcenter services for Multidisciplinary Associations 6-20
Teamcenter services for application model design instances 6-20
Teamcenter services for product structure assembly model design instances 6-22
Teamcenter services to update MDThread and Mdo0MDThread objects 6-23
Teamcenter services to search for MDThread objects 6-23
Teamcenter services for notifications 6-24
Teamcenter services for domain information 6-24

1. Getting Started

Getting started

Teamcenter Mechatronics Process Management solution refers to the integration of mechanical, electrical, electronics, and embedded software technologies into a single electromechanical product. This solution enables you to establish a collaborative environment for developing these electromechanical products. It provides centralized data, workflow, tools, and process management for integrated development. Teamcenter includes several applications and integrations that allow you to manage all related data in a single product structure.

Overview of Mechatronics Process Management

Prerequisites	This document is intended for administrators or users who maintain Mechatronics Process Management data in Teamcenter 2412. It assumes the reader already has a general understanding of the Teamcenter product.
Enable Mechatronics Process Management	Some of the applications and integrations used to manage Mechatronics Process Management data may require licenses in addition to the standard Teamcenter product license. For details about licensing, contact your Siemens Digital Industries Software representative. Following applications and integrations may need to be enabled: • Wiring Harness Design Tools Integration With Teamcenter • Embedded Software Solutions
Configure and start Mechatronics Process Management	Following applications and integrations used to manage Mechatronics Process Management data may require specific configuration. These applications must be started after Teamcenter is running. • Wiring Harness Design Tools Integration With Teamcenter • Embedded Software Solutions

Mechatronics Process Management applications

The following Mechatronics Process Management applications are bundled with Teamcenter.

Application	Description
Wiring Harness Design Tools Integration With Teamcenter	Provides extensions to the standard Teamcenter data model to support external designing tools. It allows you

Application	Description
	to integrate your existing ECAD or MCAD tools with Teamcenter to exchange data. You can create and edit harness data in an ECAD (Electrical Computer Aided Design) application such as Mentor Graphics and manage the same data in Teamcenter. After you have integrated the harness data in Teamcenter, you can control it with process management and configuration management.
Embedded Software Solutions	Comprises of different solutions that help manage software files in ClearCase, work with 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.

Mechatronics Process Management related applications

The following Teamcenter applications help you manage Mechatronics Process Management data.

Application	Description
Structure Management on Rich Client — Usage	Allows you to create generic product structures that can be configured to show the product structure at a particular time or for a certain unit (for example, a particular customer variant). You can create, revise, and delete Mechatronics Process Management related items, including items (assemblies and components), signals, connections, process variables, and embedded software. You can also create and manipulate **Realized By** and **Implemented By** relationships between items.
Multi-Structure Management	Allows you to create and view allocations between objects in product views. For example, you can relate signals in the design view to the physical wiring harness in the physical assembly view. You can also relate lines in the functional structure to the logical structure or the logical structure to lines in the physical structure.
Teamcenter Basics	Allows you to create, revise, and delete Mechatronics Process Management-related items, including process variables, revisable and non-revisable connections, signals, and harness objects. My Teamcenter includes an ECAD

Application	Description
	viewer, which allows you to view and mark up printed circuit board (PCB) designs.
Software Parameter Manager	Allows you to manage embedded software parameters and their location into hardware controller memory space. You can segregate controller memory into multiple segments which in turn can be divided into blocks. Software Parameter Manager provides the framework to import parameter addresses and conversion rules as well as to generate the flash file for each memory segment.

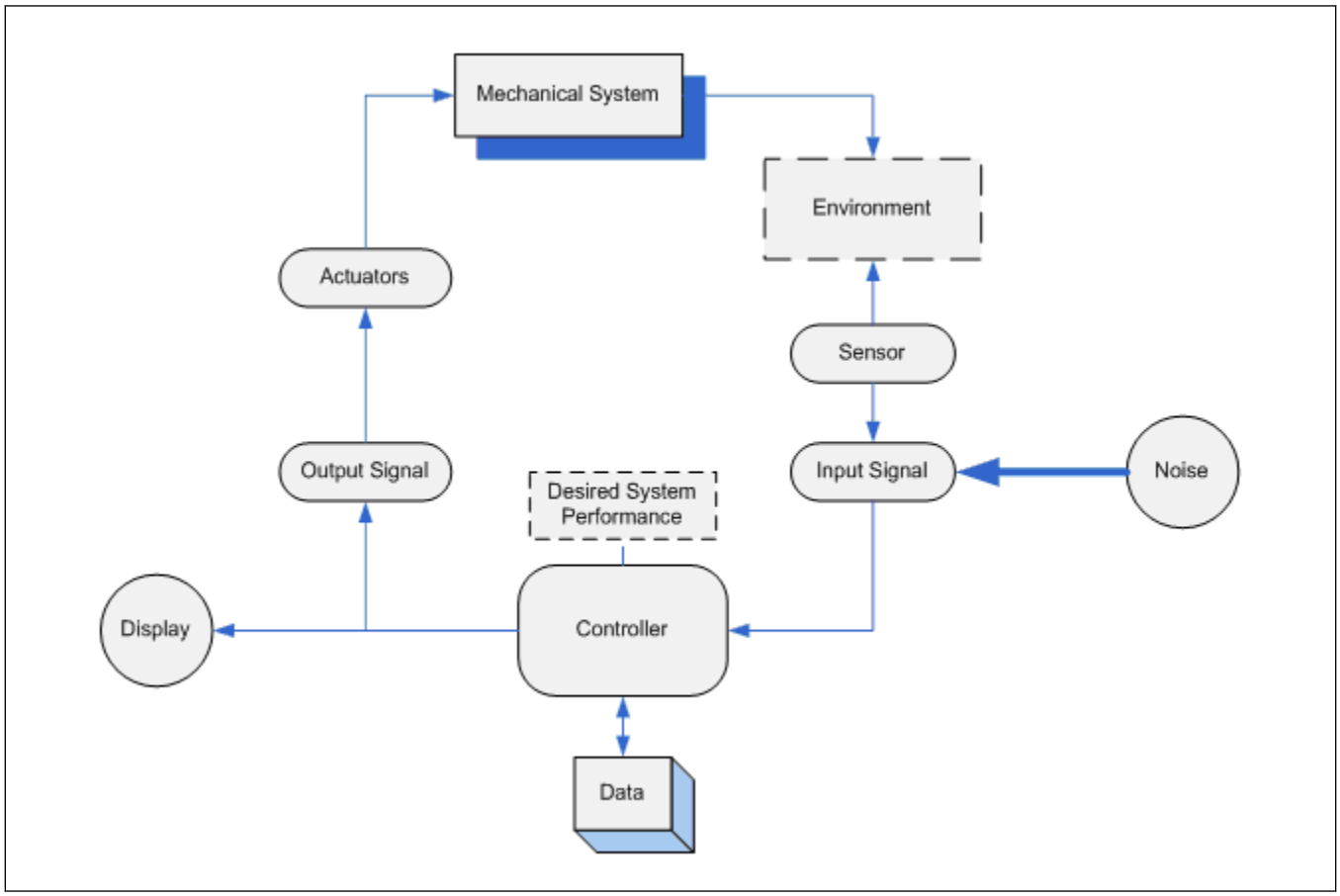
2. Basic concepts

What is Mechatronics Process Management?

Mechatronics Process Management combines mechanical, electrical, electronics, and embedded software technologies. Complex software-driven electronics play a major role in developing advanced features of many products. To address the product development issues that arise from these complexities, Mechatronics Process Management solutions facilitate a collaborative environment that enables different departments to work together as they develop products comprised of multiple mechanical, electronic, electrical, and software interconnecting components.

Mechatronics Process Management systems are widely used in all industries because they offer more flexible design and operation, increased speed, and precise performance. Teamcenter provides extensions to permit the management of Mechatronics Process Management product structures. These extensions comply with the applicable STEP models, including AP212, AP210, AP203, AP214, and KBL.

A model of a typical Mechatronics Process Management system is shown in the following figure.



Mechatronics system

As shown in the figure, Mechatronics Process Management enables elements from several different domains to work together. The data model helps product manufacturers to reduce or eliminate product quality and reliability issues by defining and managing the relationships and dependencies between all parts, options, and variants in the product structure. Teamcenter offers a rich set of functionality to support Mechatronics Process Management design.

Teamcenter supports several relationships, including device to connector, connection to cable or wire, and signal to network or connection. You can also capture relationships between the functional, logical, and physical product structures with *allocations* between components in one view and the corresponding components in another view.

It also provides APIs that allow you to create and configure third-party wire harness applications, for example, with Mentor Graphics Capital Harness. You can then manage the wire harness data as part of the product structure in Teamcenter.

Mechatronics Process Management data model

About Mechatronics Process Management data model

Mechanical, software, electronics and control (electrical and electrical interconnect) life cycles are interdependent in the Mechatronics Process Management data model. The model supports electromechanical design and process within the Teamcenter environment to share data for collaboration in the life cycle of Mechatronics Process Management systems. The data model integrates design environments so data and processes can be defined, stored, managed, and shared across applications and domains. Teamcenter has developed the data model using industry-standard data models such as STEP AP212, AP210, AP203, AP214, and KBL. The Mechatronics Process Management data model is built on the standard data model and supports functional, physical, and logical models.

Functional model

The functional model supports the representation of functions, function revisions, ports, and connectivity between functions (networks), including the associated signals. The model provides a predefined item type called **Functionality** for working with the objects.

Physical model

The physical model supports the representation of physical connectors, devices and associated terminals, physical connectivity between terminals with wires and cables, and the route associated with wires and cables.

Logical model

The logical model supports the representation of logical connectors and devices, logical interfaces of connectors and devices, and the connectivity between interfaces (connections).

Mechatronics Process Management design integrations

Mechatronics Process Management data model integrates a range of domain-specific design tools and their data in a unified design environment, ensuring that you can leverage your preferred applications and processes within a familiar and unified design environment. By integrating tools and libraries and sharing data across domains, design teams can facilitate product validation sooner. This lowers the risk of product failures caused by undetected cross-domain issues. The integrated tools are categorized as follows:

- Mechanical design integration

  Teamcenter supports most of the popular MCAD tools. By leveraging Teamcenter multi-CAD capabilities, designers can work with elements from applications and share across multiple domains.

- Electronic design integration

  Teamcenter enables ECAD teams to increase productivity by integrating design flows, managing their designs, fabricating and assembling data, and sharing data across multiple domains. Teamcenter also allows you to check in and check out data and manage fabrication data, assembly data, derived files, and extracted BOMs.

- *Embedded Software Solutions* design integration

  Teamcenter integrates ClearCase to support product configuration and helps you manage and control source code development. It enables you to manage source code versions from Teamcenter and trace modules to original requirements and resultant binaries. Teamcenter also helps manage software design component by managing software design data embedded into system components. It enables the use of configuration and integrated change management applications, build processes, software product lines, options, and variants.

- *Wiring Harness Design Tools Integration With Teamcenter*

  The Teamcenter wire harness data model enables design teams to define and manage wire harnesses employing multiple configuration options and variants from a single wire harness design. This robust data management capability enables design teams to improve design efficiency. Using a data model based on various aspects of STEP AP212, AP210, AP203, AP214, and KBL, Teamcenter transfers, stores, and manages all of your logical design, physical design, and BOM data.

Fundamental objects you work with

Teamcenter includes several fundamental objects that you can use to model parts and functions in your design. Not all of these objects are required and can be omitted, depending on the components in your design.

- Electrical components

Electrical components can include various electrical devices, connectors, splices, interconnects, wires, cables, and other electrical accessories. In general, you can categorize all these components as parts and model them as *items* in Teamcenter. The data model does not provide any specific item types to model electrical components. However, you can define your own item types. You can use Teamcenter Classification to manage these objects.

The KBL harness model defines item types to represent various types of electrical parts, including wires, harnesses, and modules. These item types are not installed by default, but you can load them manually. In a functional model, electrical components may be *functions* or *functional subsystems*.

- Electrical interfaces

  Electrical interfaces are exposed by electrical components. For example, the terminals associated with an electrical connector are the interfaces of the connector. You can use *item elements* to model electrical interfaces. You can manage, release, and instantiate item elements in an electrical product structure.

- Electrical connections

  Connections define general connectivity between a set of item elements or components in a product structure. You can use a connection to define logical connectivity between interfaces of the functions or electrical connectors. Teamcenter provides two predefined connection types for use in electromechanical product structures.

  - *Network connection* to model connectivity between interfaces of two or more functions.

  - *Connection* to model logical or physical connectivity between interfaces of two or more electrical connectors or devices.

  The AP212 model separates the connectivity data from the device that implements a connection. Teamcenter allows you to separate connectivity data in a similar manner to the AP212 model.

  You can use the **ImplementedBy** relationship to capture the association between a logical connection and a physical device that implements the connection.

- Signals

  Signals represent information or messages transmitted between electrical connectors or devices. Signals can be instantiated in a product structure and associated with electrical product constituents such as connections, terminals, and connectors to capture various relationships; for example, the transmitter of the signal and the receiver of the signal. You can also define custom signal types to meet specific needs.

- Allocations

  Allocations represent mappings from one view in a product structure to another. For example, you can build allocations from functions, networks, and ports in a functional view to devices,

connections, and terminals in a physical view. Allocations represent directional relationship between specific occurrences in two different views. They are independent of the structures they map and are managed independently. You create allocations as a set to map one structure to one or more other structures; you then define an allocation map or context to group them together. You can also configure allocations independent of the structures they map.

- Routes

  The route information defines the physical route that a wire or cable traverses through the product.

- Relationships

  Relationships help in associating components within a given structure.

STEP AP212 and KBL data models

To allow the sharing of design and process with other applications, Teamcenter supports several industry standard data models, including STEP AP212, AP210, AP203, AP214, and KBL.

An acronym for Standard for the Exchange of Product Model data, STEP (formal name: Industrial Automation Systems and Integration–Product Data Representation and Exchange International Standard, ISO 10303) is a standard that provides a framework through which industries can exchange and share product information within and between enterprises. It defines standard data exchange protocols that allow engineering data developed under one automated design tool to be read and manipulated by design teams using different automated tools. STEP includes a number of standard application-oriented data models for product description known as application protocols (APs).

STEP AP212 describes the information necessary to customize electrotechnical products, according to the requirements of the installation and share it between the parties involved in its design, installation, and commissioning. It defines the context, scope, and information requirements for the exchange of design and installation information of electrotechnical equipment and specifies the integrated resources necessary to satisfy those requirements. Electrotechnical systems may be used in plants, buildings, or transportation systems such as cars or ships. AP212 does not impose any restriction on the usage of these systems. It covers equipment for power transmission, distribution and generation, electrical machinery, electric light, electric heat, control, and automation systems.

Teamcenter also supports KBL (KabelBaumListe – wire harness description list), which is a standard for wiring models that enhances the STEP AP212 model. The KBL object model provides high-level components that represent wires, harnesses, and parts in a vehicle that hosts the harness. This model is a subset of the AP212 model and it represent wire harness data in a similar manner.

The KBL model provides specific object types for representing electrical components, including terminals, wires, connectors, and housings. It fully supports the NX wire harness model and allows a wire protection occurrence to be associated to multiple route segments.

The following types are provided to support the KBL functionality:

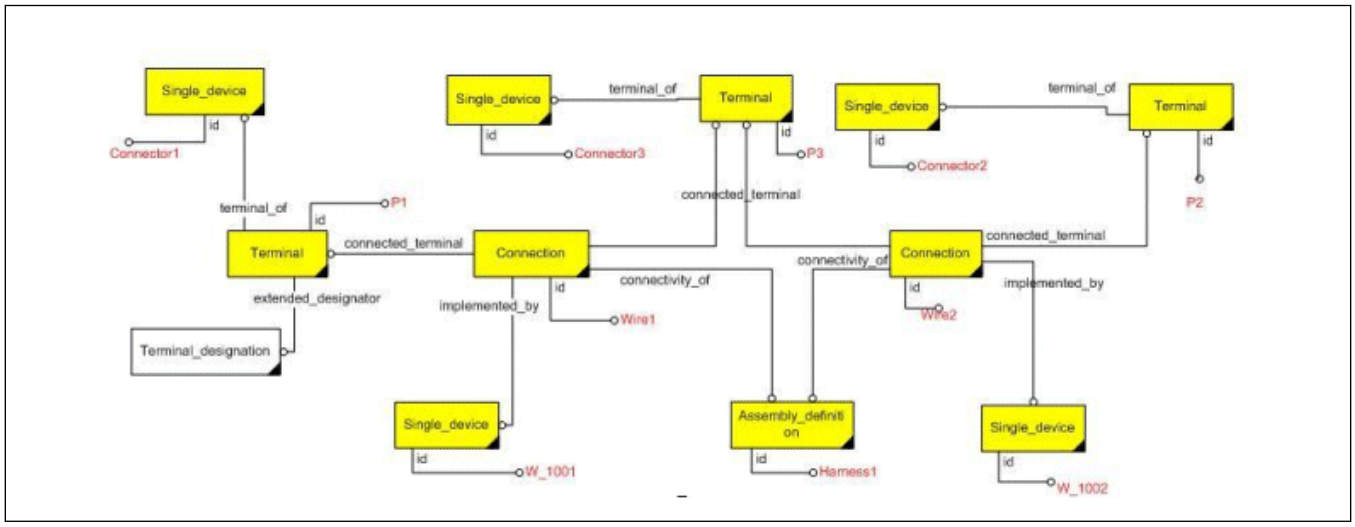
Type	Purpose
HRN_Harness	A harness is an assembly of insulated conductors formed to a predetermined pattern or configuration.
HRN_Module	A module is a physical part of the harness and is electrically defined by one or more module groups including the required harness furniture.
HRN_Fixing	A fixing is a component with which the harness is fixed.
HRN_Accessory	An accessory is any supplementary portion of a connector that helps a harness perform its function.
HRN_CoPackPart	A part is supplied and installed with the wiring harness but without any electrical connection.
HRN_AssemblyPart	An assembly part is a component that contains other subordinate objects.
HRN_WireProtect	Wire protection is a mechanism to describe harness wrappings.
HRN_ConHousing	Con housing is a nonpopulated connector without addressed cavities.
HRN_Cavity	A cavity is a defined space in a housing for the location of an electrical terminal or a cavity plug or seal. It can be empty.
HRN_Slot	A slot is a mechanism to group the cavity objects of a connector housing.
HRN_GeneralWire	A general wire is a physical wire, performing an electrical connection. It may define a single wire or a multicore wire.
HRN_Core	A core is a single conductor of a multicore wire, including its isolation.
HRN_CavityPlug	A cavity plug is a watertight nonelectrical object that fills an empty cavity.
HRN_CavitySeal	A cavity seal is a watertight nonelectrical object that fills a populated cavity.
HRN_GenTerminal	A general terminal object is a device used to terminate a conductor that is usually fixed to a post, stud, chassis, or other conductor to establish an electrical connection.

Type	Purpose
HRN_Cable	A cable is one or more wires bound together, typically in a common sheath. The individual wires may be covered or insulated.
HRN_Shield	A shield acts as a protection against damage and is mounted on cables and wires.

The following is an example of electrical connectivity data.

Wire ID	From connector	From pin	To connector	To pin	Part number	Harness
Wire1	Connector1	P1	Connector3	P3	W_1001	Harness1
Wire2	Connector2	P2	Connector3	P3	W_1002	Harness1

The AP212 application reference model (ARM) instance diagram corresponding to this example is shown in the following figure.



AP212 ARM instance diagram

Working with allocations

About allocations

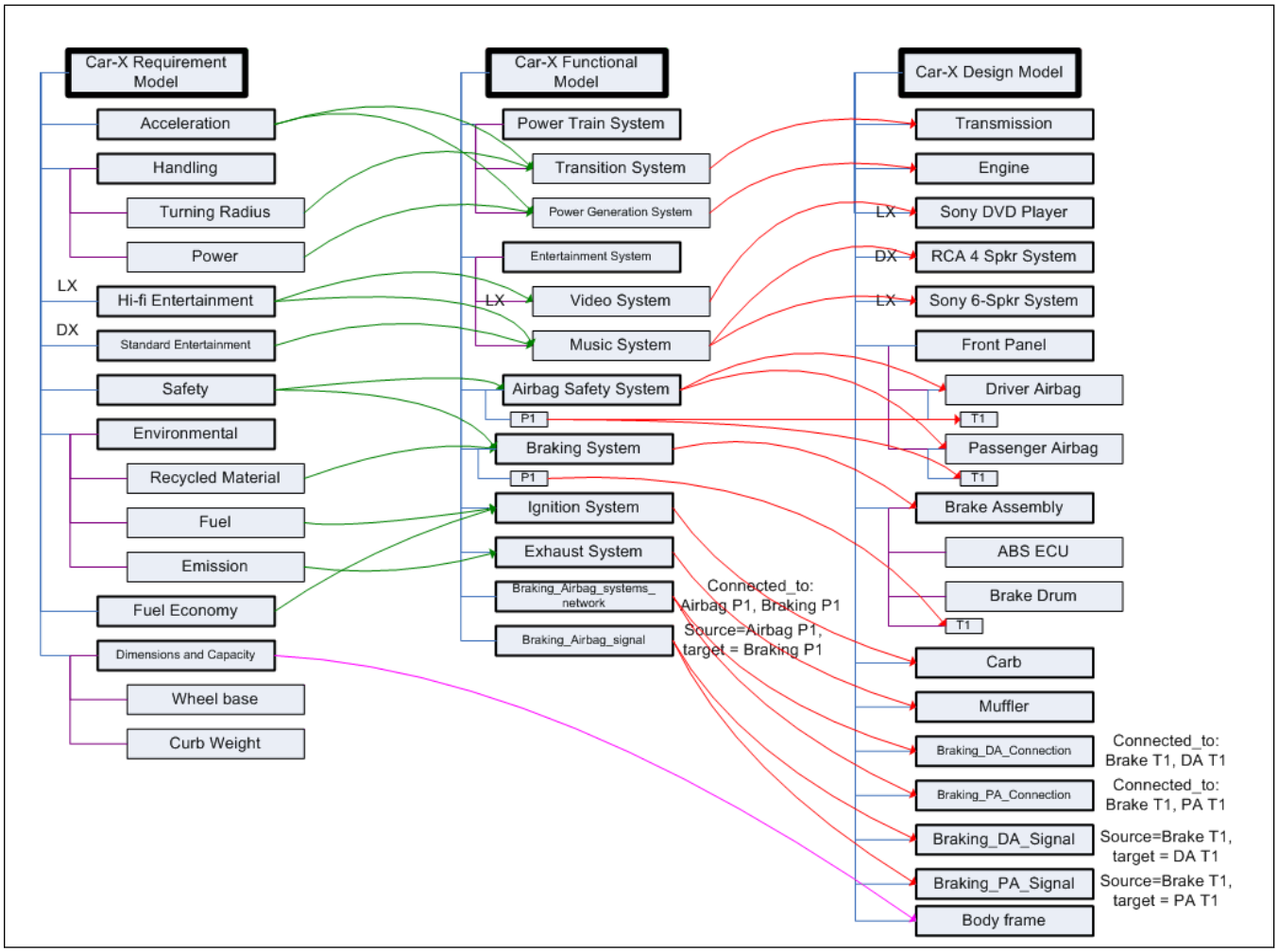
Allocations represent a mapping from one view in a product structure to another view. Use allocations to create separate views of a product and then map the views to each other. They allow you to link the actual parameters on physical parts to the requirements on the functional and logical models.

Analyzing allocations

During the development process, a product evolves through various revisions of the product structure and its components. It also evolves through various representations. For example, during system analysis and engineering, you express the representation of the product in terms of a set of requirements the product must satisfy and a set of functions it is expected to perform. As the product design matures, you may identify other representations, including the logical structure, the assembly structure, and the manufacturing structure. All these representations are related and a change to one representation may necessitate changes to other representations. For example, a change to one of the product requirements may cause the product to perform an additional function. A new physical component is added to perform this function and, in turn, a new manufacturing process is needed to assemble this component into the product. You can capture, track, and manage these dependencies between representations with allocations.

You can create a Teamcenter *view* to model each of these representations, for example, a requirements view, a functional view, and a physical assembly view. Alternatively, depending on your product and working practices, you might create an electrical design view, an electrical harness view, and an assembly view.

The following figure shows an example of how you can create three allocation maps that map the requirements, functional, and assembly views of a product. Note that allocations are made between the functional systems in the functional view (for example, the music system and the video system) and the corresponding physical systems in the assembly view.



Allocations between views

Creating allocations

You can create allocations at various levels of a structure, but in each case they specify an object of interest—one that performs a particular function. It is possible that every item revision in a structure could contain an allocation. For example, this may be necessary for manufacturing purposes, where all items must be consumed.

Because allocations are independent of the structures they map, you can create and manage them independent of the structures themselves. For example, a functional designer may design the functional model, whereas a mechanical engineer designs the physical model. When both structures are complete, a systems engineer may map the functional model to the physical model. The systems engineer may trade off studies between allocating functions to different components and perform the actual assignments.

Note:

The model is the representation of a particular aspect of the product, not the CAD model. For example, the functional model of a car represents a product containing an air conditioning system, power braking, power steering, and other functionality. In Teamcenter, the model is represented by the product structure.

When you create an allocation, Teamcenter automatically assigns it a unique identifier. The identification scheme is determined by user exits set by the Teamcenter administrator.

Because you can configure structures in Teamcenter, you can create allocations between variable product structures. Allocations are valid only in a particular context and therefore all allocations between two structures are part of a group, and Teamcenter evaluates them as a unit. The allocations tie together various components and revisions of items in the structures.

Optionally, a customizer can create subtypes of allocations and allocation maps using the Business Modeler IDE. The Teamcenter administrator can set preferences that determine the allocation subtypes that are valid in a particular allocation map type and, similarly, the allocation maps that are valid between structure type (BOM view types). The administrator also determines whether each allocation subtype is valid for:

- One source and one target.
- One source and multiple targets.
- Multiple sources and one target.
- Multiple sources and multiple targets.

Configuring allocations

When you create allocations, the structures you work with are typically unconfigured products. You can then create allocations between many items in the structure; multiple allocations can originate from a source item and be allocated to several target items. If the source item is configured, the allocation is assumed to be configured with the source item. If the target item is missing, Teamcenter displays an error because all allocated items must be allocated to a target.

Releasing allocations

When can I release allocations?

You can release allocations independent of their associated structures or items. Similarly, you can release the structures or items without releasing their allocations.

Releasing an item with an allocation to a physical component

Allocations are independent of the structures they allocate. The allocations are modeled in their own environment and must form a valid set. Releasing an item on either side of the allocation has no effect on the allocation itself.

> Note:
>
> The allocated target component is identified as a property on the source structure. Because the allocation is independent of source and target release status and the allocation can be created or deleted on a released object, the allocation information (property value) on the source component can change, regardless of its release status.
>
> Teamcenter may allocate the same function structure to multiple physical components. Consequently, the **allocated to** information on the source may differ, depending on the context of allocation.

Releasing an allocation

You can release allocations and, therefore, they may be given a status and possibly an effectivity. The release status may configure allocations in one of two ways:

- The release status and effectivity on the allocation configures the allocation from the revision rule.
- Each of the allocations can be associated with an incremental change order that determines effectivity. Teamcenter uses the effectivity on the incremental change order to configure the allocation.

If you add status to an allocation, you can apply various effectivities to this relationship independent of the structures themselves. This provides flexibility when applying several allocations to the structure.

Releasing allocation maps

Because an allocation map is a workspace object, you can release it. Similarly, you can release allocation map revisions, which is necessary because users typically work with allocation map revisions, not allocation maps. After an allocation map revision is released, you can define additional security rules that depend on the release status, allowing users to control a set of allocations as a whole. For example, you can define a rule that prevents users from modifying a set of allocations once it has been approved.

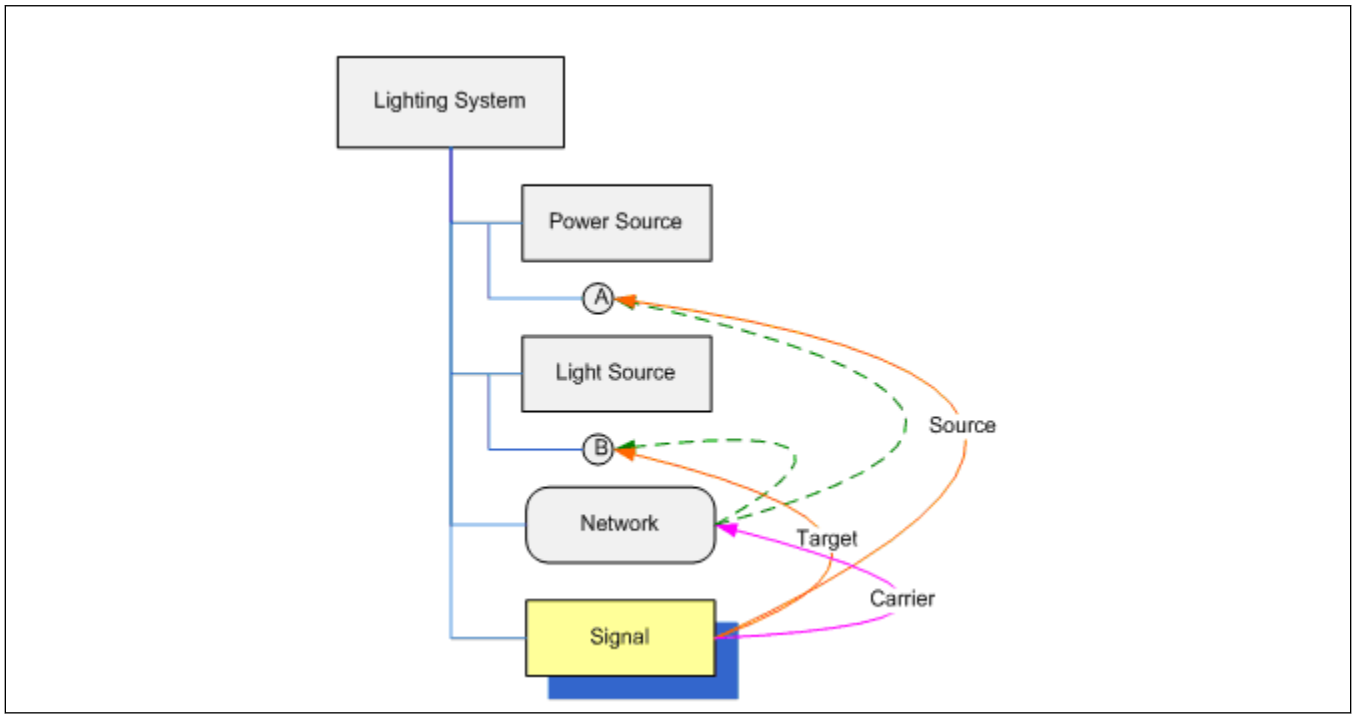
3. Basic tasks

Creating an electrical functional model

Creating an electrical functional model

You can use Teamcenter to represent the functional breakdown of an electromechanical product. The functional breakdown of a car consists of various functional subsystems, for example, the music system and the braking system. In turn, the functional subsystems may contain various individual functions of the subsystems. In a functional model, electrical components can be functions or functional subsystems.

For example, the following figure shows how you can represent an automobile lighting system as a functional structure in Teamcenter.



Electrical functional model

Teamcenter provides the following types of objects to allow you to create functional models.

- **Functionality**

  Represents functions.

- **Network_Port**

Represents interfaces of functions to the external world.

- **Network**

  Represents the connectivity element linking various functions through their interfaces.

- **Signal**

  Represents the message passed between source and target via the carrier.

Create functions

1. In Structure Manager, choose **File→New→Item**.

2. Select **Functionality** as the **Type**.

Create function composition

1. In Structure Manager, open the function.

2. Copy the functions involved in the breakdown, and paste it in Structure Manager to the appropriate function representing the system/sub-system.

3. Select the function in Structure Manager, and choose **File→New→Interface Definition→Network_Port** to create ports for the function.

Managing information exchange between functions

Information exchange between control units

The system engineer creates and associates messages and signals when building the structure of the control unit. The messages and signals exchanged between the control units result from how the functions are distributed across the control units.

Create a signal

1. Select a folder or a containing object for the signal.

2. Choose **File→New→Signal...** or click **Signal** on the toolbar.

3. Select the appropriate type of signal from the list.
   The valid types are:

   - **Signal**
   - **Message**

- **PSSignal**

4. In the **New Signal** dialog box, enter the following information for the signal.

Field name	Definition	Valid Values
ID	The signal ID (required). Leave the field blank to automatically assign an ID.	Alphanumeric string (1-128 characters)
Revision	The signal revision (required). Leave the field blank to automatically assign a revision.	Alphanumeric string (1-128 characters)
Name	The signal name (required).	Alphanumeric string (1-128 characters)
Description	Description of the signal.	Alphanumeric string (1-240 characters)
Unit of Measure	The unit of measure used for the signal.	Select from the list of values.

5. (Optional) Click **Next** to assign the signal to a project.

6. Click **Finish**.

   A new signal of the specified type is created.

Associate signal to functions as source/target

1. Select a message and one or more objects of the appropriate type (for example, an ECU item for a message or a port for a signal) in the product structure. The valid appropriate types are configured in the **SIG_asystem_source_rules** and **SIG_asystem_target_rules** preferences.

   For details on how to set these preferences, see the *Teamcenter Environment Variables*.

2. Choose **Tools→Signal Manager→Associate Signal To** and then choose one of the following 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 or signal. Select a connection or its subtype with the signal or message. 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.

Create a network

1. In Structure Manager, in function decomposition, select the appropriate function representing the system/sub-system or the top level and choose **File→New→Item→Network**.

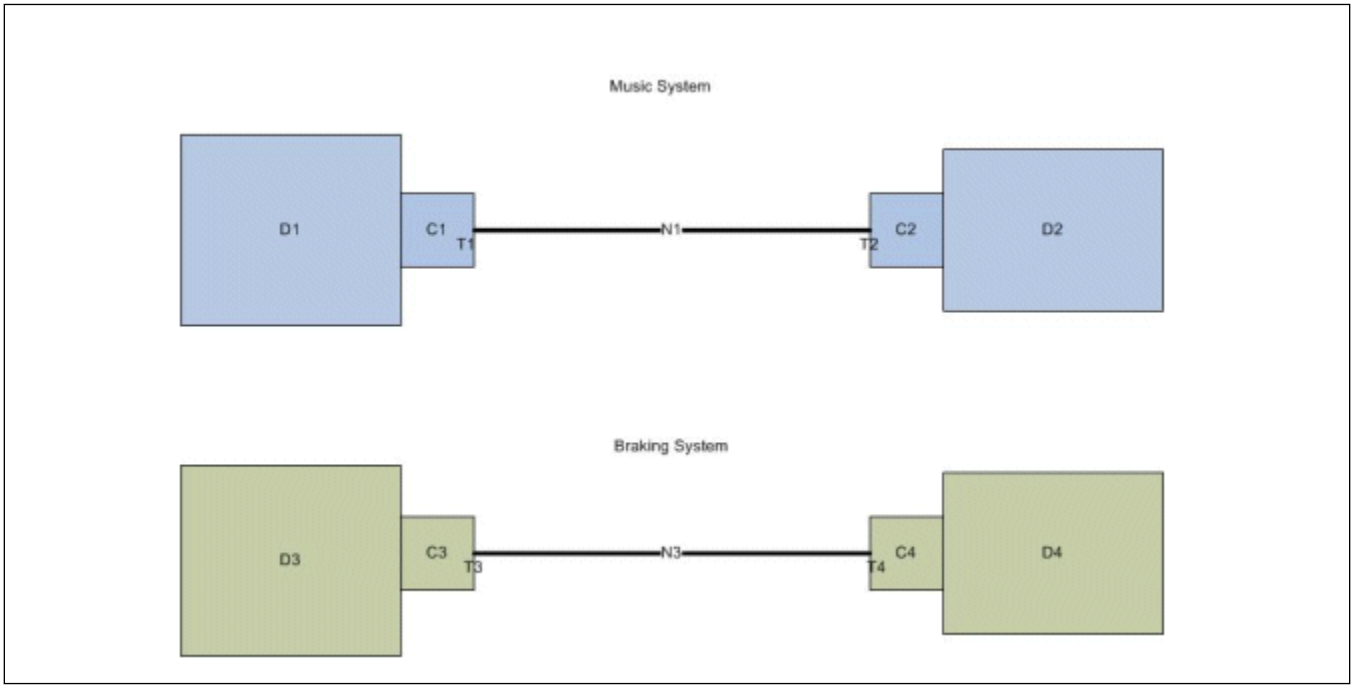
2. Copy the **Network** object and paste it in Structure Manager.

3. Select the ports of function that are in the same function network and the network object.

4. Choose **Edit→Connect** to associate the ports of function to create a network of functions.

Creating an electrical logical model

You can use Teamcenter to represent logical electrical connectivity in your product structure. For example, the logical connectivity for the functional subsystems shown in the figure may consist of a set of electrical devices, connectors, and logical connections between their interfaces. The following figure shows the logical electrical diagram, in which D1 and D2 are electrical devices and C1 and C2 are the associated electrical connectors. T1 and T2 are the electrical interfaces of C1 and C2.

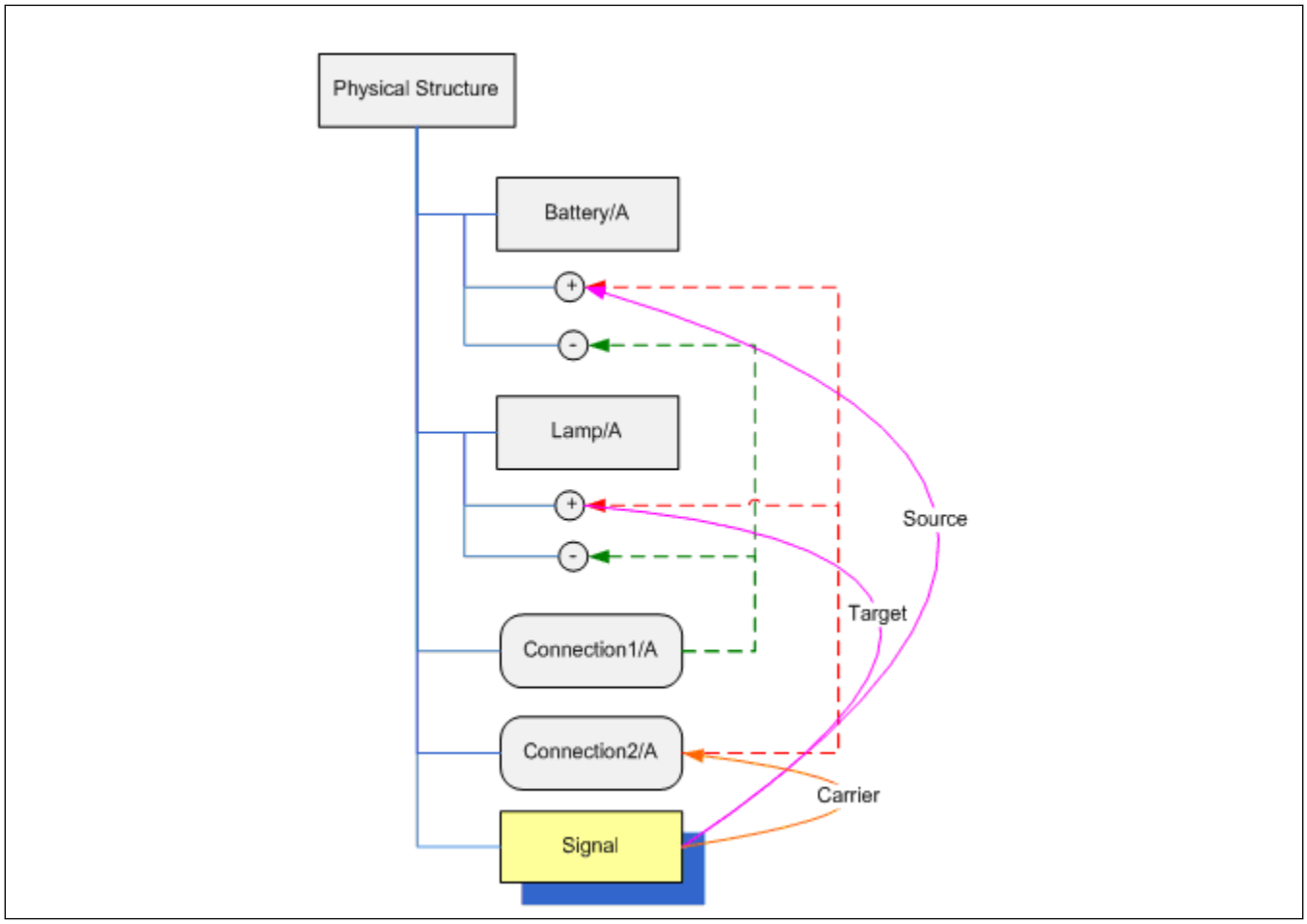


Functional schematic diagram

To represent electrical connectivity, you can create allocations between the functional model and the logical model.

Creating an electrical physical model

The physical model represents the actual physical objects or devices in a product, for example, wires, harnesses, electrical devices and connectors, and the interconnections among them. In Teamcenter, the physical model is represented as a structure of components containing devices and other physical entities as shown in the example in the following figure.



Electrical physical model

You can instantiate connections in the electrical view of the product and define connectivity by associating an instance of a connection with occurrences of terminals. You can also associate connection instances with instances of wires and cables that implement the connections in the context of the electrical structure.

You can represent wires, harnesses, and other electrical components as items or any subtype of items that meet your business requirements.

You can also capture the 3D mechanical assembly structure corresponding to the product in a separate view type of the product. You can then build allocations between the electrical connectors and devices in the electrical view to the mechanical assembly view.

Building views

You can build dedicated view types of structures to represent electrical designs.

- Electrical design views represent various subsystems of the product, for example, a music system or braking system. You can add datasets with attached graphical diagrams in SVG format in these views. The diagrams may contain functional connectivity diagrams or physical wiring connection diagrams.

- Electrical harness views model one or more harnesses and electrical connectors or devices. These views represent the tangible implementation of a functional design at an appropriate level of granularity for the product, for example, a left door harness or an instrument panel harness. You initially populate this view from an ECAD application with connectivity information and logical device connectors. Use a 3D electrical application, such as NX Electrical Routing, to subsequently add physical data and complete the assignment of logical connectors to physical parts.

- Mechanical assembly views contain 3D representations of physical devices and connectors. These views represent the packaging information associated with the product. The top-level item of this view should contain all the devices that the harness must be routed to or around. You typically populate this view from a 3D MCAD or electrical application such as NX Electrical Routing.

Typically, each project in an ECAD application includes one or more designs that comprise functional, logical, or physical models. It also contains an electrical BOM. For example, a vehicle project may include a design for each harness that is fitted in the vehicle.

Working with wiring harnesses

Typical integration process of electrical routing application, ECAD application, and Teamcenter

The following is a typical process for working with an integration between Teamcenter, an ECAD application, and a 3D electrical routing application.

1. Author electrical connectivity information and create the associated datasets in the ECAD application, for example, Mentor Graphics Capital Harness. The electrical information may be the functional design, system design, or harness design.

2. When the designs are complete, publish the data in the Teamcenter database. You can also apply an appropriate release status to this data.

3. A mechanical designer uses a 3D electrical application such as NX Electrical Routing to add more details to the electrical data, for example, wire lengths and route data.

4. Use the ECAD application to search for published data in Teamcenter and retrieve the data modified by the mechanical designer.

5. Use the ECAD application to reconcile the changes made to the connectivity data and make additional edits if necessary.

6. Publish the finalized data and schematic datasets into Teamcenter.

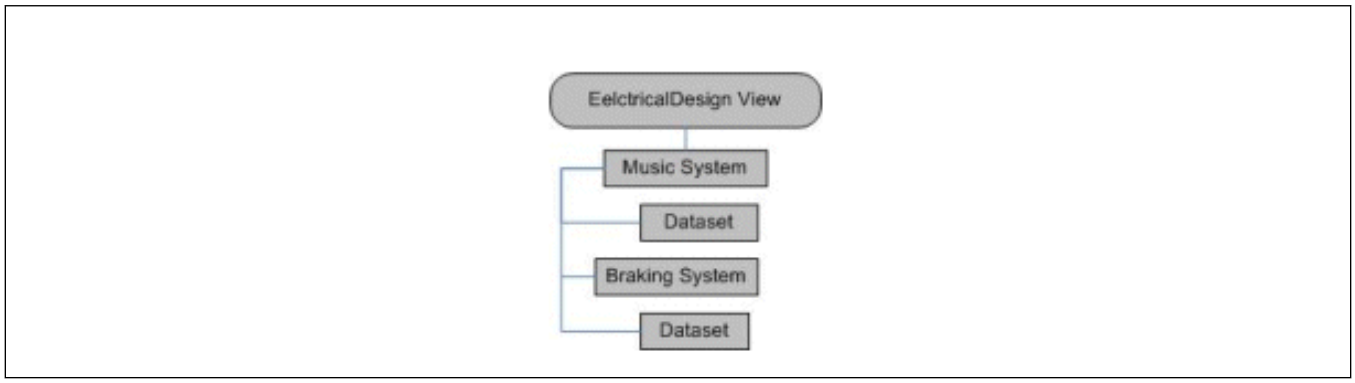
Identifying the top-level product

Because the design view, harness view, and other views associated with an electromechanical product represent various aspects of the same product, you must identify a top-level product that ties all these views together. You then publish this top-level product to Teamcenter with the associated views. Typically, you define the top-level product early in the design process in ECAD applications when you create a project to design the connectivity data. This product must be modeled in Teamcenter as an *item*; you can choose to represent it as a specific subtype of an item. You can specify a unique identifier for the item in the ECAD application, or Teamcenter can automatically generate a unique identifier when the item is published to it.

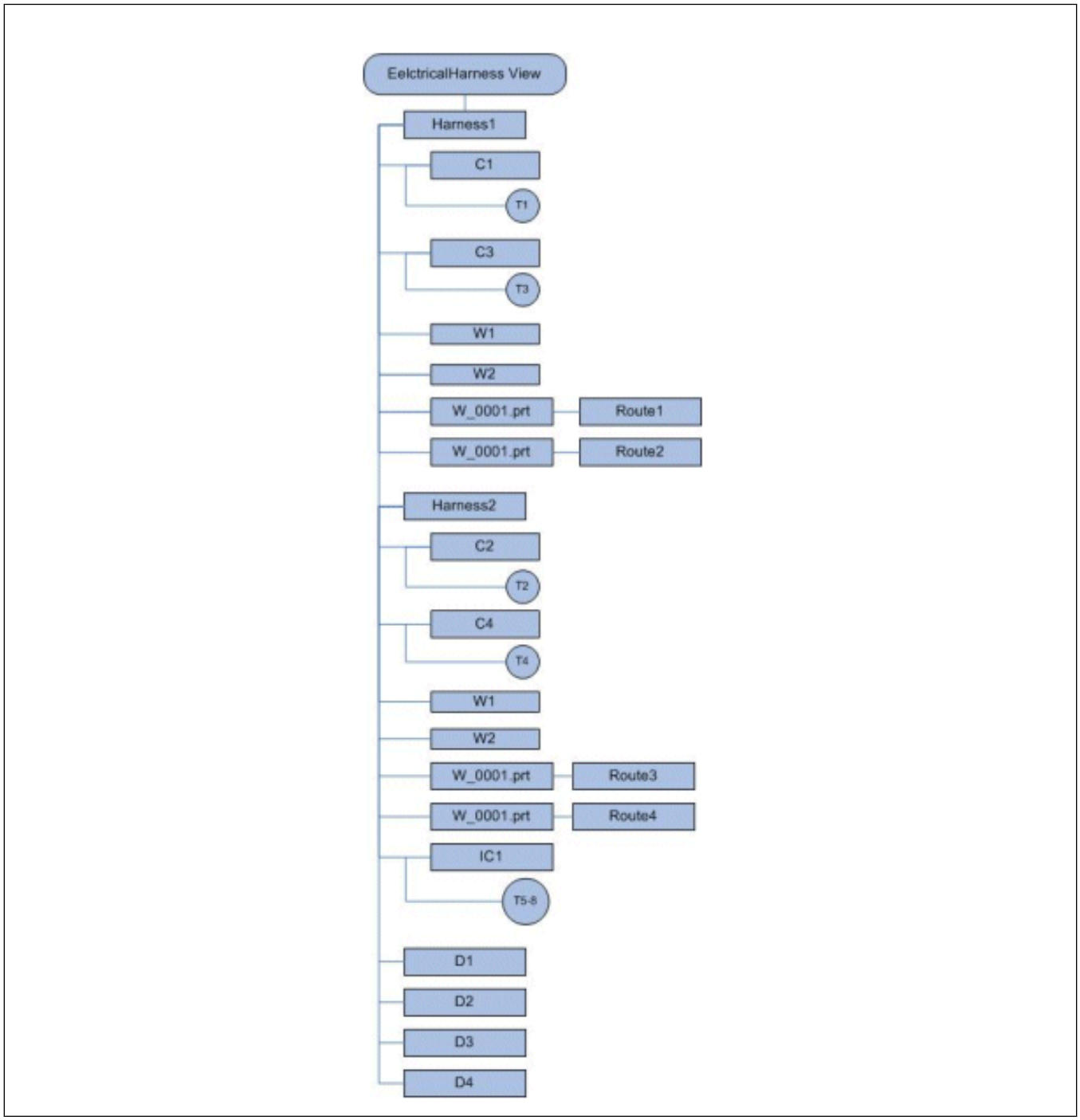
Identifying the publishing information

The ECAD application may publish the connectivity data in the electrical harness view and also in the design view consisting of electrical subsystems. It may publish this information in Teamcenter after it is complete and approved or while the data is still in the design phase. After the electrical data is published in Teamcenter, it can be submitted for an approval workflow process and configuration management. Approved designs may be released to the packaging engineers, who incorporate the harnesses into the mechanical design with an application such as NX Electrical Routing. The data published in Teamcenter may also contain options and variants created in the ECAD application.

Examples of the electrical design views and electrical harness views that you create are shown in the following figures.



Electrical design view



Electrical harness view

If necessary, you can publish routing topology associated with wires designed in an ECAD application to the electrical harness view. Typically, routing topology is authored in a 3D electrical application such as NX Electrical Routing. The top-level product may already exist in Teamcenter if the mechanical view of the top-level product is published by another application such as 3D MCAD. In this case, you can add further views specified in the ECAD application to the existing item.

Separating electrical connectivity data and mechanical information

When the electrical data of the product is separate from its mechanical data, you can build allocations between the electrical view and mechanical view to capture the assignments between the logical connectors and the physical electrical connectors. This type of product data representation allows you to create the electrical connectivity data and the mechanical assembly data separately, often in different applications. This technique allows you to manage the electrical and assembly structures independently.

Assigning electrical components to mechanical connectors

The data published to the electrical harness view of the product consists of logical connectors, devices, and connections grouped by harnesses. You can allocate the logical electrical connectors to physical connector components from the mechanical assembly view by assigning allocations. Alternatively, the logical electrical connector and the physical connector component may share the same absolute occurrence identifier for the allocated occurrences. You can create these assignments in Teamcenter or in a 3D electrical application integrated with Teamcenter.

Defining a route

A route specifies the physical path of a device, connection, or wire in the context of its use in a particular structure. The context of the route object associated with a connection or device is the same as the context where the association is defined. An association between a route and a connection or device may exist in the context of the immediate parent assembly of the connection or device, any intermediate parent assembly, or in the context of the top-level assembly. When a route exists in the context of the immediate parent assembly of a connection, it stays in the same position relative to its immediate parent. The absolute position of the route depends on the position of its immediate parent in the top-level assembly.

When a route exists in the context of the top-level assembly, you must associate it to the instances of connections or devices because different instances of a connection or device may have different physical routes in the context of the top-level assembly. In this case, you must associate routes to the absolute occurrences of connections or devices in the product structure.

Create allocations

You can create allocations between objects in views.

1. Create the necessary views independently in Structure Manager and populate them with items. For example, you can create requirements, functional, and physical assembly views. You can apply variant rules or other techniques to configure these structures, as described in *Getting Started With Product Structure*.

2. Create an allocation map (context) to group each set of allocations that you want to map two views. For example, create one map to group all allocations between the requirements view and the functional view and another map to group allocations between the functional view and the physical view.

3. In the context of the appropriate allocation map, create allocations between individual components. Each allocation can map two components and is directional; one component is the *source* and the other is the *target*.

Working with data dictionary

About data dictionary

In typical automotive and aerospace functional and logical design activities, key building blocks or components of the designs are reused to save time and effort. To facilitate the reuse of these building blocks or components in multiple designs across several projects or programs, Teamcenter provides a central organizational repository. Designers can easily search and find components based on specified criteria and reuse them. This central organizational repository is referred to as a data dictionary. It allows you to organize data by your own criteria with the goal of easy reuse.

Data dictionary basic tasks

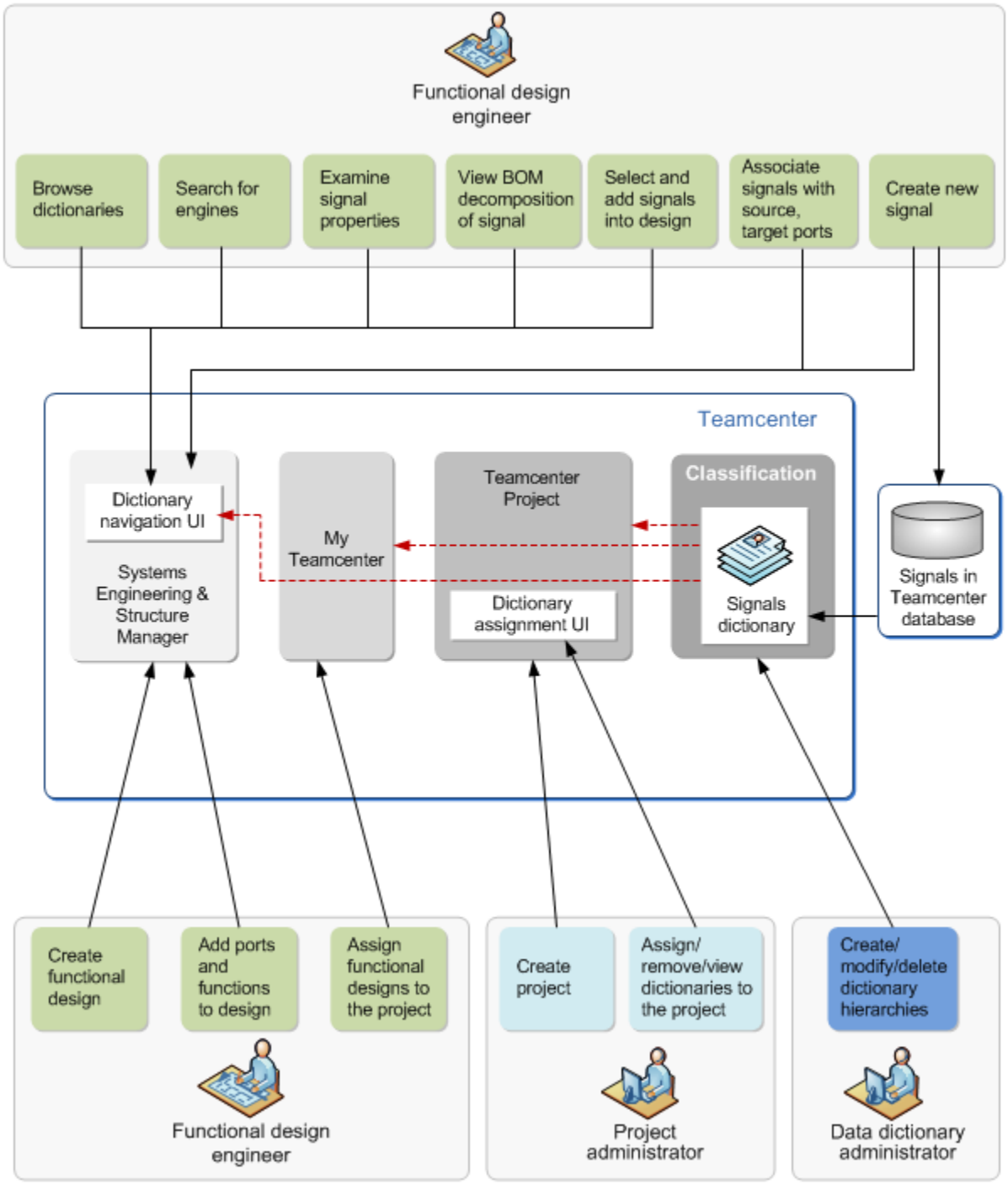
You can perform the following basic tasks using a data dictionary:

- Create a data dictionary and set it up as a library
- Modify and delete the data dictionary
- Define dictionary access control
- Search and view data based on certain criteria
- Create new signals and add signals to the dictionary
- Add signals from the dictionary to BOM
- Set effectivity and release signals

Data dictionary functional flow

The key applications that work with a data dictionary include Classification Admin, Systems Engineering and Requirements Management, Project, and Structure Manager. You can model the data dictionary using Classification Admin. It allows you to build a hierarchy for organizing data.

The following figure shows the various applications used to perform the tasks based on the roles.



Data dictionary applications and roles

- Data dictionary administrator

  The data dictionary administrator receives requests for setting up or providing information on dictionaries. They work out of the Classification Admin application and performs the following key activities:

    - Creates a dictionary hierarchy
    - Modifies a dictionary hierarchy
    - Modifies attributes on a dictionary
    - Deletes a dictionary
- Project administrator

  Works out of the Teamcenter Project application and performs the following key activities:

    - Creates/modifies/deletes a project
    - Assigns/removes users/groups/roles that work on a project
    - Assigns/removes/views dictionaries for a project
    - Assigns functional designs to the project
- Functional design engineer

  Works out of the Systems Engineering and Requirements Management application for the functional design activities and performs the following key activities:

    - Creates and modifies designs
    - Searches/views/adds objects from available dictionaries
    - Adds signals, ports, and functions
    - Associates signals and ports

Interacting with external applications

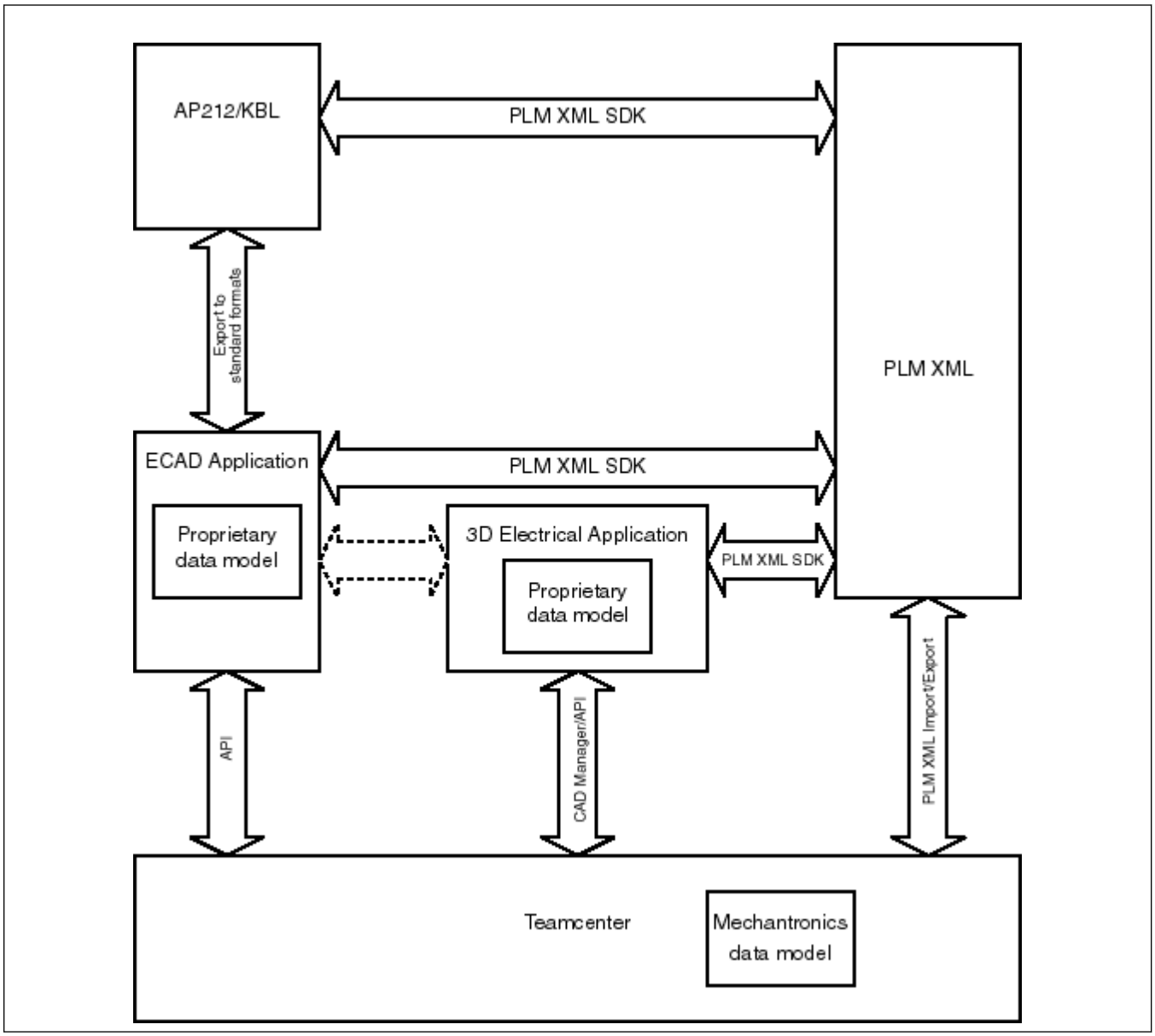
Interacting with external applications overview

You can publish information to Teamcenter, share and exchange wire harness data with other applications, such as MCAD and ECAD, or work with Teamcenter features like Workflow using either PLM XML or SOA.

Teamcenter acts a central repository of master wire harness data and external ECAD and MCAD applications can create and modify the harness design data in Teamcenter. This enables seamless transfer of harness design between ECAD-Teamcenter-MCAD without data loss and redundancy. You can exchange wire harness data, including options and variants associated with MCAD and ECAD with

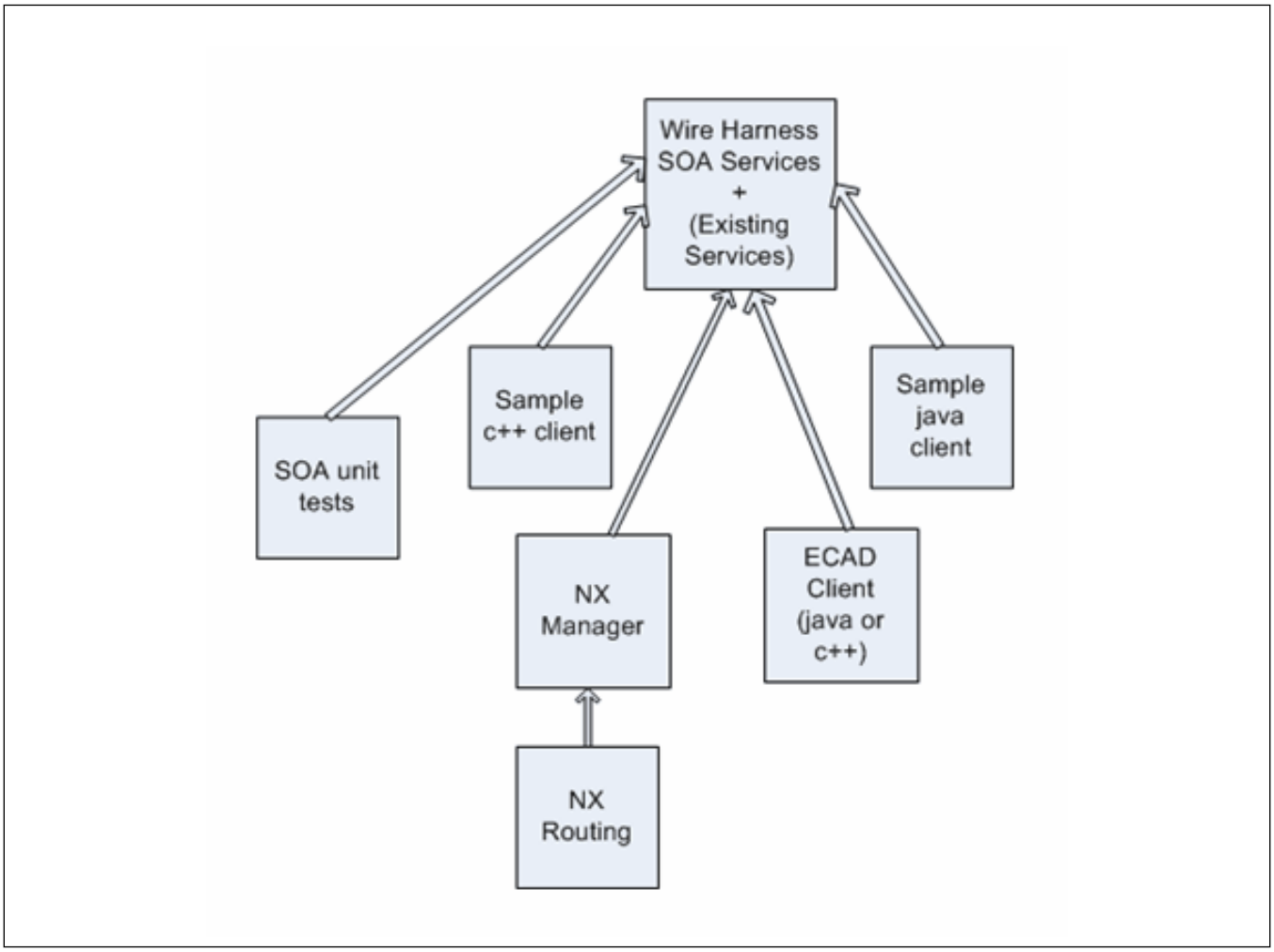
Teamcenter. ECAD and MCAD applications can exchange electrical information and Teamcenter can capture design options created in electrical authoring applications. The SOA services facilitate data exchange with MCAD and ECAD applications.

The following figure shows the Wiring Harness Design Tools Integration using PLM XML for interaction.



Wiring Harness Design Tools Integration using PLMXML

The following figure shows the Wiring Harness Design Tools Integration using SOA for interaction.



Wiring Harness Design Tools Integration using SOA

Updating schematics with changes made to connectivity data

Typically, ECAD applications obtain wire lengths and bundle cross-sections from 3D electrical applications and update the ECAD schematic data accordingly. In cases where both an ECAD application and a 3D electrical application are integrated with Teamcenter, you can update the ECAD schematic data by sending a PLM XML representation of the updated connectivity data from Teamcenter to the ECAD application.

Completing the electrical product structure with routes

You can manage the physical routes associated with the wires and cables. You can publish the route data into the Teamcenter electrical harness view from the ECAD application or the 3D electrical application. The ECAD application retrieves the route data with the connectivity data as part of electrical product structure. This data can be used to reconcile and optimize the schematic design.

Restructuring a Mechatronics structure

Restructuring a Mechatronics structure

You can restructure a representation, including a BOM view, occurrence group, structure context, or composition. You must revise a frozen product structure before restructuring. Restructuring edits the product structure in downstream views (for example, manufacturing) while preserving the derived occurrence structure and data related to specific occurrences of parts and assemblies. Restructuring is disabled for product structures that contain CAD designs because it can make the CAD data invalid.

During restructuring operations, Teamcenter maintains the integrity of incremental changes, classic or modular variants, and structure relationships. Teamcenter displays warnings when it encounters absolute occurrences attributes and data.

If you have edits pending to a product structure, you must save the edits before you open the structure in another application.

In addition to restructuring, you can edit individual properties on any line, subject to the following limitations:

- Restructuring is not permitted on lines that have pending edits.
- Property edits are associated with a relative occurrence and are marked as pending until they are saved to the database.
- Property edits are highlighted only if you use the column editor. If you use other methods of changing properties (for example, the **Properties** dialog box), these edits are not visually highlighted in the properties table. However, Teamcenter still retains the details of such edits until you save or revert them.
- Use the **PS_structure_change_condition** preference to specify actions as structure edits. For example, by default, changing a reference designator is not considered as a structure edit, but you can add this action to the preference. NX requires reference designator changes to be considered as structure edits.
- If any note in the list of notes is edited, the **All Notes** field shows a **...** icon with a red strike-through. It does not show the exact original value.
- Edits to the absolute or relative transformation matrix are not highlighted.
- You cannot edit the first property column (**BOM Line**).
- If you cut more than one BOM line to the clipboard and then modify the BOM lines on the clipboard, this action changes the ownership of the remaining BOM lines. For example, if you cut two BOM lines and then remove one of these lines from the clipboard, the status of the remaining line changes from

pending cut to *pending copy*. If you want to modify the BOM lines that are the subject of a cut action, repeat the cut action on the required BOM lines, rather than modifying the contents of the clipboard.

- If you remove a line that contains one or more unsaved changes to its substructure, the system does not automatically save those changes.

- By default, if you cut or copy a line and then paste it to a new location, incremental change elements (ICEs) are not copied. This may necessitate significant manual recreation of data if you are cutting or copying many lines together. To automatically copy ICEs, the administrator must set two Business Modeler IDE constants:

  - **Fnd0EnableIceCarryOver** business object constant

    When moving, copying, or assigning a line from one location to another, this constant determines if the ICEs are carried forward. You must set this constant to **true** on both the source location's parent and the target location's parent.

  - **Fnd0AttrICEsToExclude** property constant

    Defines the occurrence attributes that Teamcenter does not copy to the target location for occurrence attribute changes.

  These settings apply to in-context changes made to structure lines, their attachments, and their occurrence attributes.

> Note:
>
> Restructuring primitive actions include:
>
> - Removing a level (removing a line and keeping child lines)
> - Inserting a level (including pasting a line as a parent of selected lines)
> - Moving a line to a new location (for example, cut and paste actions)
> - Splitting an occurrence
> - Replacing data in context
>
> Only the last two actions make copies of the absolute occurrence data; the other actions share the existing absolute occurrence data.

In certain cases, restructuring may cause valid reports of broken links, as shown in the following examples:

Example 1:

```
A
+-----------B
            +-----------C (APN1 in context of A)
                        +-------------D
```

If you remove level **C**, the structure becomes:

```
A
+-----------B
            +-----------D
```

The link to **C** is lost. This is correctly reported as a broken link.

Example 2:

```
A
+---------E
+---------B
          +---------C
                    +-----------D(APN1 in context of B)
```

If you move **D** to **E**, it is outside of the context of **B** and **APN1** is lost. This is correctly reported as a broken link.

Example 3:

```
A
+-----------B
|           +-----------C
|                       +-------------D(APN2 in context of A)
+-----------B
|           +-----------C
|                       +-------------D(APN1 in context of A)
+-----------D
+-----------D
```

If you move **C** to **D**, the system cannot determine which **D** to move the APN to without user interaction. It skips the APN and logs an error message.

Example 4:

```
A
+-----------B
            +-----------C
                        +-------------D(APN1 in context of B)
```

If you remove level **B**, it becomes:

```
A
+-----------C
            +-----------D
```

The link to **D** is lost. This is correctly reported as a broken link.

Insert a level in a Mechatronics structure

You can create an item and insert it in the current structure as a new level below the selected line. The number of relative occurrences of the children is preserved. If you select more than one line, they must share the same parent.

1. Select the parent line of the new level and choose **Edit→Insert Level**.

   Teamcenter displays the **Insert Level** dialog box.

2. Enter the item identifier and other attributes of the new item. If the inserted item is new, you must insert it with a quantity of **1**.

3. On completion, click **OK** or **Apply**.

   Teamcenter creates the new item and inserts it as a new level. The selected branches become children of the inserted level, while the inserted level becomes a branch of the original parent. All existing variant conditions, notes, absolute occurrences, and other data is moved down with the selected branches. The default quantity of the new level is **1**, meaning that no quantity change occurs.

> Note:
>
> In most cases, associations between objects (such as **Connected To** relations) are correctly maintained or are reassociated. However, after editing a Mechatronics structure, always choose the **Fix In-Structure Associations** command to identify any invalid associations.
>
> You cannot insert a level if the selected line represents a generic design element (GDE).

> Note:
>
> You can only insert a level if the line represents a standard business object type. If the line represents a custom type, copy it to the clipboard and choose **Paste Special**. Teamcenter pastes it as a new level above the currently selected line.

Remove a level from a Mechatronics structure

1. Select the affected line and choose **Edit→Remove** or click **Remove**  .

   Teamcenter displays the **Remove** dialog box.

2. (Optional) Select **Keep subtree** to reattach the children of the removed level to the next higher level parent line.

3. Click **Yes** to confirm removal of the line. The total number of instances is preserved at the end of this action. Any options of the removed lines are moved up and variant conditions are merged.

If you try to remove a level that would result in option definitions becoming inconsistent (for example, options that are referenced by a parent line), Teamcenter displays an error message.

Caution:

If you select a line for removal and edit tracking is enabled, the line is displayed in red with a strike-through until you commit the edits. Do not attempt to edit or work with this line, or you may obtain unpredictable results. If you want to edit or work with a line that is marked for removal, revert the removal of the line by choosing **Edit→Revert Edit** first.

Similarly, do not edit or work with a marked line in another structure editor such as Multi-Structure Manager. Always complete and save your work on the structure in Structure Manager before you open it in another structure editor.

If you remove one of the lines that is the context of a Mechatronics relation between primary and secondary lines, the relation itself is also removed.

Note:

In most cases, associations between objects (such as **Connected To** relations) are correctly maintained if you select **Keep subtree** or are reassociated. However, after editing a Mechatronics structure, always choose the **Fix In-Structure Associations** command to identify any invalid associations.

You cannot insert a level if the selected line represents a generic design element (GDE).

Move a node in a mechatronics structure

You can move a selected node from one branch to another. All substructure and occurrence data moves with the node.

1. Move a node using any of the following methods:

   - Cut and paste using the **Edit→Cut** and **Edit→Paste** menu commands.
   - Cut and paste using the Ctrl+X and Ctrl+V shortcut keys.

   Caution:

   Do not try to move a node by dragging the line to its new position. Teamcenter performs a copy action when you drag a line.

Teamcenter displays the **Paste** dialog box.

2. Do one or more of the following:

   - Change the item ID and revision ID.
   - Change the view type, if applicable.
   - Select if the line should be pasted as a component of the selected assembly line, as a substitute for the selected line, or as a new level above the selected line.
   - Specify the number of occurrences, quantity per occurrence, and find number.

3. Click **OK** or **Apply** to complete moving the line.

Caution:

If you cut a line and edit tracking is enabled, the line is displayed in red with a strike-through until you commit the edits. Do not attempt to edit or work with this line, or you may obtain unpredictable results. If you want to edit or work with a line that is marked as cut, revert changes to the line by choosing **Edit→Revert Edit** first.

Similarly, do not edit or work with a marked line in another structure editor such as Multi-Structure Manager. Always complete and save your work on the structure before you open it in another structure editor.

Note:

In most cases, associations between objects (such as **Connected To** relations) are correctly maintained or are reassociated. However, after editing a mechatronics structure, you should always choose the **Fix In-Structure Associations** command to identify any invalid associations.

Any lines representing GDEs are renumbered after moving a node.

If you move both the primary and secondary lines of a mechatronics relation out of context, the relation is removed. However, if you move both the primary and secondary lines of a mechatronics relation to any level below the context, the relation is not removed. It is listed in the **Fix In-Structure Associations** dialog box when you choose **Tools→Fix In-Structure Associations**, allowing you to fix the invalid association.

If you remove the secondary line of a mechatronics relation out of context, the relation itself is removed. Refresh the primary line to see the updated BOM column properties.

If the line has an associated variant condition and you move it out of the scope of the option referenced in the variant condition, the variant condition is not retained.

If you move a node that has absolute occurrence overrides out of the scope of the context of the overrides, the system does not retain the override values after the move.

Replace a node in a Mechatronics structure

You can replace an item representing a node in the structure with another item. All data associated with the original node is preserved.

> Note:
>
> Your system administrator sets the **PS_replace_with_substructure** preference to determine if any substructure below the node is replaced. If this preference is **true**, the node and its entire substructure (if any) are replaced without prompting. If it is **false**, Teamcenter displays an error message and does not complete the replacement action.

1. Select the line to replace and choose **Edit→Replace Node** or click  .

   Teamcenter displays the **Replace Node** dialog box.

2. Enter the item identifier and other attributes of the item that replaces the existing item.

3. Click **OK** or **Apply**.

   Teamcenter replaces the existing item.

   If you have edit highlighting turned on, the number of the original part is shown in red, strike-through text.

> Note:
>
> In most cases, associations between objects (such as **Connected To** relations) are correctly maintained or are reassociated. However, after editing a Mechatronics structure, you should always choose the **Fix In-Structure Associations** command to identify any invalid associations.
>
> If the child of the copied item includes **RealizedBy**, **ImplementedBy**, or **RedundantSignal** as the primary association, the corresponding child in the replaced assembly points to the secondary association of the original (copied) assembly.
>
> You can replace a line that represents a generic design element (GDE) with another GDE line but not with a line that represents another object.
>
> You can replace Mechatronics items with any type of item. However, Siemens Digital Industries Software recommends you use the same item type as the replaced item or one of its subtypes for the replacement item. Use of incompatible types may result in invalid relationships between occurrences in the product structure. The following table lists possible Mechatronics items for replace operations.

Mechatronics items for replace operations

Object	Recommended types for replace
Functionality	**Functionality** or its subtype
Connection	**Connection** or its subtype
Network	**Network** or its subtype
PSConnection	**PSConnection** or its subtype
Signal	**Signal** or its subtype
PSSignal	**PSSignal** or its subtype
Message	**Message** or its subtype
HRN_Accessory	**HRN_Accessory** or its subtype
HRN_AssemblyPart	**HRN_AssemblyPart** or its subtype
HRN_CavityPlug	**HRN_CavityPlug** or its subtype
HRN_CavitySeal	**HRN_CavitySeal** or its subtype
HRN_CoPackPart	**HRN_CoPackPart** or its subtype
HRN_ConHousing	**HRN_ConHousing** or its subtype
HRN_Fixing	**HRN_Fixing** or its subtype
HRN_GenTerminal	**HRN_GenTerminal** or its subtype
HRN_GeneralWire	**HRN_GeneralWire** or its subtype
HRN_Harness	**HRN_Harness** or its subtype
HRN_Module	**HRN_Module** or its subtype
HRN_WireProtect	**HRN_WireProtect** or its subtype
AppSoftware	**AppSoftware** or its subtype
Calibration	**Calibration** or its subtype
ConfigFile	**ConfigFile** or its subtype
PriBootloader	**PriBootloader** or its subtype
SecBootloader	**SecBootloader** or its subtype
Processor	**Processor** or its subtype

Split an occurrence in a Mechatronics structure

You can split a line that represents several occurrences into two branches. The new branch and the original (changed) branch initially have the same notes, variant conditions, and other data, but you can subsequently modify them independently. The quantity on the original line before the split must be greater than 1.

1. Select the occurrence line and choose **Edit→Split Occurrence**.

   Teamcenter displays the **Split Occurrence** dialog box.

2. Enter the quantity for the new line that results from the split and click **OK** or **Apply**.

> Note:
>
> You cannot split an occurrence if the line represents a logical object, for example, **Connection**, **Signal**, or **Software**. You *can* split interfaces and other Mechatronics objects, for example, **HRN_GeneralWire** and **Processor**.
>
> In most cases, associations between objects (such as **Connected To** relations) are correctly maintained or are reassociated. However, after editing a Mechatronics structure, you should always choose the **Fix In-Structure Associations** command to identify any invalid associations.
>
> If you split an occurrence line that represents a GDE, Teamcenter increments the GDE number. The quantity of a GDE line must always be an integer.
>
> If the child of the original occurrence includes **RealizedBy**, **ImplementedBy**, or **RedundantSignal** as the primary association, the corresponding child occurrence of the split occurrence points to the secondary association of the original occurrence.

Fix in-structure associations

When you manually edit a structure containing Mechatronics elements such as signals and connections, you may inadvertently create invalid associations between elements. To avoid multiple, potentially misleading warnings, Teamcenter does not display error messages during the editing session. However, once you have completed any editing session during which you chose the **Insert Level**, **Remove**, **Move** or **Replace** commands, manually check for and remove any invalid associations.

1. Select the top line of the edited structure and choose **Tools→Fix In-Structure Associations→Current Level** or **Tools→Fix In-Structure Associations→All Levels**.

   Teamcenter displays a dialog box containing a list of invalid associations. The scope of the invalid associations depends on whether you chose the current level or all levels.

2. Select the first invalid association and click **Remove**.

   Teamcenter removes the selected association from the structure.

3. Repeat the previous step for each of the other invalid associations in turn.

> Note:
>
> Any associations that you do not remove manually remain in the structure.

Tracking edits to a mechatronics structure

If your administrator sets the **PSE_Display_Pending_Edits** preference to **true**, edits to the product structure tabular display are highlighted visually until you save them to the database.

If necessary, you can override this preference setting locally for your current session by selecting the **Display Markups for pending edits** command.

Highlighting edits feature is available only in Structure Manager, not other structure editor applications.

> Note:
>
> This feature is not compatible with the BOM markup feature, and you cannot use the two features together.
>
> Ensure you select this feature when you want to edit the BOM directly, not when you want to create markups (propose changes).
>
> You cannot convert pending BOM edits into markups.

> Tip:
>
> Enable highlighting only when you want to specifically track edits you are making; otherwise, the display may become difficult to interpret.
>
> Close all Structure Manager windows before changing this option to avoid having open windows where edits are not highlighted. (Changing this option does not affect existing windows.)

Edits indicated are as follows:

- Additions

  Displayed in green italic font.

- Removals

  Displayed in red with a strike-through line.

- Property edits

  Displayed with the original values in red with a strike-through line.

If you have pending edits to the structure, you can do any of the following:

- Click **Revert** to undo all pending edits to the selected line.

  All edit marks are removed from the line.

- Click **Save** to commit all pending edits to the database.

  The structure is redisplayed with no edit marks.

- Click **Revert All** to cancel all pending edits.

  The structure is redisplayed with no edit marks.

> Note:
>
> You cannot compare structures with pending edits with the **BOM Compare** command. However, you can expand the structure manually to view the effect of pending edits.
>
> Likewise, you cannot highlight pending edits if you enable incremental changes.

> Note:
>
> You cannot create new associations (for example, a **Connected To**, **Implemented By** or **Embeds** relation) to a line that is displayed in red with a strike-through line.
>
> When you save pending edits, any associations to deleted lines are also deleted. If you revert pending edits, all associations are maintained.

Working with Teamcenter ClearCase Integration

Teamcenter 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. You can manage versions of software files in ClearCase and use Teamcenter for product modeling, workflow, and process management.

Use Teamcenter ClearCase Integration to:

- Create an **SCMVersionObject**.
- Check out an **SCMVersionObject**.
- Check in an **SCMVersionObject**.
- Create a workflow process for ClearCase data.

Managing embedded software

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. They comprise related hardware and software components. For example, software resides on a processor, while a software component may depend on another software component for execution; likewise, a hardware component may depend on another hardware component for activation.

You can create a product structure for the embedded system, which allows its unique attributes and relations to be stored, reused, and managed in Teamcenter. You can manage embedded software by using Embedded Software Solutions.

Use *Embedded Software Solutions* to:

- Model embedded hardware components, like processors.
- Model embedded software components, like calibration software.
- Model real time operating systems, like bootloaders.
- Model software binaries.
- Manage configuration and calibration related parameter data for embedded systems.
- Manage design data of embedded software components.
- Create relations between the embedded components, for example, processor to software, software to software, and processor to processor.
- Compile reports about embedded components and compatible hardware to software components for any embedded software.
- Integrate external flashing applications with Teamcenter.

Administering PLM XML data transfers

PLM XML data transfers overview

The PLM XML export import functionality in Teamcenter allows you to import objects from external systems or export objects to external systems. A PLM XML file is created as the mechanism for exchanging data between Teamcenter and the external application. It supports the transfer of objects such as items, datasets, BOMs, forms, folders, and system data (for example, business rules, organization data, type definitions, lists of values, and saved queries). The standard PLM XML schema supports electromechanical objects.

Transfer mode objects define the import or export context by applying closure (traversal) rules, filter rules, and property set rules to the input or output data. These rules may be stored in the database as a static set or they may be applied only to a specific session. It is important to use the correct transfer mode with PLM XML export and import operations to accomplish the data transfer of desired entities.

Use the **MechatronicsFoundationDataExport** transfer mode to export ECAD data from Teamcenter. Use the **IncrementalImport** transfer mode to import ECAD data into Teamcenter.

When you select the **MechatronicsFoundationDataExport** transfer mode, the following objects and information are transferred:

- Signals
- Associated systems of a signal object [source, target, transmitter]
- Connections
- Connection information
- **ImplementedBy** information
- **RealizedBy** information
- Interfaces
- Allocations
- Allocation maps
- Routes
- Nodes
- Segments
- Route location
- Process variables
- Signal values
- Various KBL types

To export Embedded Software Solutions related objects and relations, add the following closure rules:

- `TYPE BOMLine TYPE * PROPERTY bl_embeds_lines_tags PROCESS+TRAVERSE`

- `TYPE BOMLine TYPE * PROPERTY bl_dependentOn_lines_tags PROCESS+TRAVERSE`
- `TYPE BOMLine TYPE * PROPERTY bl_gatewayOf_lines_tags PROCESS+TRAVERSE`

Export electromechanical structure

1. Run the **plmxml_export** command line utility, as described in the *Teamcenter Utilities*.
2. Choose the **Tools→Export** menu command in My Teamcenter.
3. Choose the **Tools→Export** menu command in Structure Manager.

Viewing ECAD design

Viewing ECAD design

After you translate your native CAD PCB file to the XFATF neutral file format, you can view the file using the *ECAD viewer* in My Teamcenter. The *ECAD viewer user interface* includes several viewing frames, toolbars, and shortcut menu options. The viewer includes an assembly tree that allows you to quickly browse to components and assess their position in the assembly. You can also search for components in large designs by name. An integral toolbar allows you to adjust the view and mark up the PCB.

When you open PCB images, two types of information are available: a textual representation of the PCB assembly and a visual representation of the board. The assembly appears on the assembly page and the visual representation on the Viewing window. On the assembly page, information is displayed in a structured or hierarchical format. This format represents data about specific components and layer details that are associated with the open PCB. In the Viewing window, PCB layers and components are displayed following default viewing characteristics (for example, background color).

View a PCB file

1. Find and highlight the desired PCB type from the home folder.
2. On the right side of the My Teamcenter interface, click the **Viewer** tab in the data pane area.

   The file is loaded into the ECAD viewer.

Marking up ECAD design

The *ECAD viewer* provides the facility to mark up PCB files. It allows you to maintain written communication and collaboration with both internal and external team members by adding text and graphic markups, and saving and displaying your markups. The viewer provides robust textual markup options and specialized markup tools.

You can create markup profiles to create sets of markup preferences. When you create markup profiles, you specify a profile name and choose markup appearance preferences that are used for that specific markup set.

When you work with PCB markups, you can:

- Add a text markup.

  A text markup may provide vital information or direction to other team members.

- Add predefined text markups.

  Adding predefined markups help if you are typing the same markup text repeatedly. You can automate this process by selecting markup text from an existing text file.

- Automatically translate text messages.

  Several standard PCB design text messages are predefined and translated in languages supported by Siemens Digital Industries Software. When you open a PCB containing one of these predefined text messages, the text is automatically translated to the default language of the computer that opened the file.

- Add to an existing markup.

  You can update an existing text or graphic markup. This helps maintain revisions to an existing markup without losing the previous markup information.

- Display predefined text automatically.

  If you are accessing predefined text options regularly, you can display predefined text automatically.

- Add graphic markups.

  You can use lines and graphics to enhance and clarify your design.

- Save and display your markups.

  After you have added or revised markups, you can save them and also display them as icons or as fully expanded text and graphics.

4. Mechatronics Process Management applications

Mechatronics Process Management applications and integrations

The Teamcenter applications and integrations that allow you to manage Mechatronics Process Management data are:

- Wiring Harness Design Tools Integration
- Embedded Software Solutions
- Structure Manager
- Multi-Structure Manager
- My Teamcenter

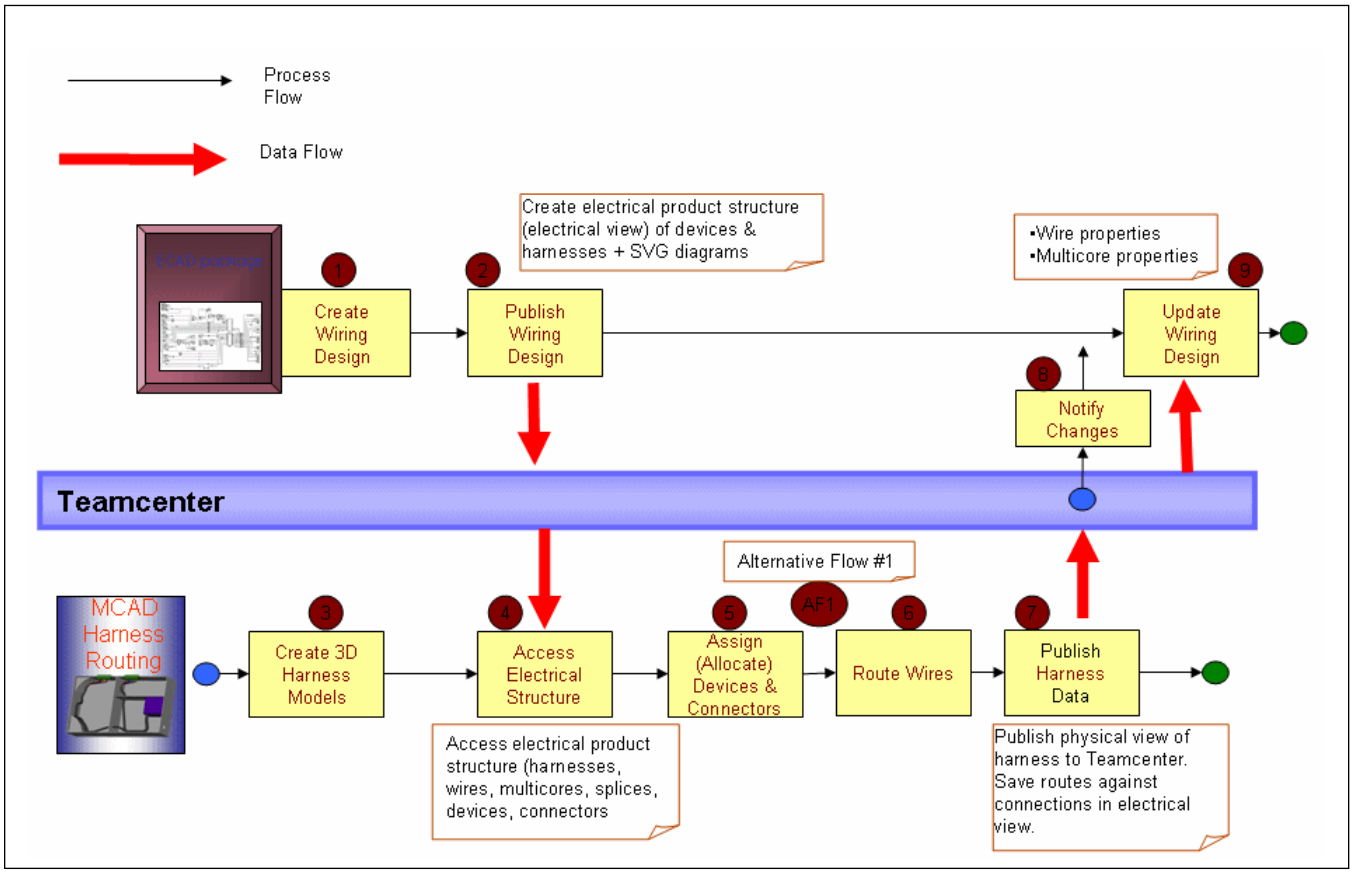
Wiring Harness Design Tools Integration

The *Wiring Harness Design Tools Integration With Teamcenter* allows you to create and edit harness data in an ECAD application and manage the same data in Teamcenter. Use the integration to:

- Facilitate data exchange (options and variants) with ECAD and NX applications.
- Publish electrical connection data created in the ECAD application into Teamcenter.
- Search the harness application for data already published in Teamcenter.
- Retrieve data from Teamcenter back into the harness application to edit, or to synchronize with changes that were made in Teamcenter.
- Define associations between assignments and allocations created in the harness application for harness component data and the corresponding data in the mechanical routing application.

Data is transferred between applications in PLM XML format. The data transfer may be initiated manually or automatically.

The following figure shows an overview of the integration.



Wiring Harness Design Tools Integration overview

Embedded Software Solutions

Embedded Software Solutions help manage binary software in the Structure Manager and My Teamcenter. Use the Embedded Software Solutions to:

- Configure, track, and control all hardware and software components that make up your product and their related product/process definitions.
- Manage versions of software files in ClearCase and use Teamcenter for product modeling, workflow, and process management. 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.
- Access ClearCase information from Teamcenter and query the ClearCase views and versioned object bases (VOBs).
- 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.

- Quickly perform an 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.
- 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 design data (source code, object files, 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 component and define dependencies between different software design data components.
- Associate a software design data component to a binary component.
- Create a composition of the software design component.
- Create, update, delete, import parameter data, group related parameter definitions together, and associate parameter value to projects. You can manage the memory of all the ECUs and generate a flash file and export it.

Structure Manager

Structure Management on Rich Client — Usage allows you to create generic product structures that can be configured to show the product structure at a particular time or for a certain unit. Use Structure Manager for creating and managing the electrical and mechanical elements of the design to:

- Create or modify items that represent electrical or mechanical components or assemblies.
- Create, view, or modify electrical signals, and relate them to physical components. You can associate a signal line in an electromechanical structure to another line as a source, target, transmitter, process variable, or redundant signal.
- Create, modify, or substitute item elements to represent ports, interfaces, terminals, or process variables associated with a signal, and relate them to physical components.
- Create or modify interface definitions to represent connections or network ports, and relate them to physical components.
- Create or modify revisable connections or nonrevisable connections. You can also associate the connection with the connected objects.
- View binary software information in the same way as you view data about hard parts.
- Show or hide all lines that are connected by the selected connection.
- Create, view, and manipulate **Implemented By** and **Realized By** relationships.

Multi-Structure Manager

Multi-Structure Management allows you to manage any structure type compatible with Teamcenter, including product structures and manufacturing process structures. Use Multi-Structure Manager to:

- Collect a subset of data for comparison and analysis purposes, such as for a design study.
- Capture the state of any structure or part of a structure for subsequent retrieval and viewing.
- Collect a subset of data to send to an external application, for example, Tecnomatix, NX, or third-party applications.
- Open multiple BOM structures, configure these as you would in Structure Manager, perform an accountability check, and then save all of this to a collaboration context.

My Teamcenter

My Teamcenter is the workspace you use to manage your product information. My Teamcenter includes an ECAD Viewer, which allows you to view and mark up printed circuit board (PCB) designs. You can find this when you select a PCB design in the XFATF neutral file format in the Teamcenter application's navigation tree and click the Viewer tab. You can also use the Application Interface Viewer to manage transfers of electromechanical data between Teamcenter and external ECAD systems.

Use My Teamcenter to:

- Manage the dependencies between the electrical and mechanical elements of the design, including connections, signals, and item elements.
- Create folders to organize commonly referenced objects and relationships.
- View the contents of your **Home** folder, **My Worklist**, **My Saved Searches**, and **My Links**.
- Perform and track tasks.
- Send and receive e-mail.
- Open objects, automatically launching the related Teamcenter application.
- Search for objects, both in your local environment and at remote sites, using predefined queries.
- Compare search results to other searches or collections.
- Create and manage items, item revisions, and datasets.

5. Mechatronics Process Management data model

About Mechatronics Process Management data model

The data model in Teamcenter includes abstract classes that allow you to model Mechatronics Process Management objects, including items, item elements, BOM views, occurrences, and allocations. If you want to configure or customize your Teamcenter environment, an understanding of the data classes is essential.

You can use the abstract classes to create many types of models including system models, functional models, logical models, physical models, and manufacturing models. Use BOM views to create multiple view types for each type of model. For example, you can define a view type to represent the functional model and another view type to represent the physical model. You can then build allocations to link various view types of a product to capture the associations between the components from one view to another.

Using the abstract classes

You can use the following abstract classes to model Mechatronics Process Management objects:

- Item

  Items are generic, manageable, revisable, releasable objects that you can use in a product structure. In a physical model, use items to represent parts. When integrating with a CAD system, items typically represent physical components instantiated in an assembly structure. When integrated with an ECAD system, use items to represent electrical components such as electrical devices, connectors, and wires. You can also use items to represent system modules and electrical assemblies, for example, electrical harnesses consisting of connectors and wires.

  In functional models, use items to represent the functional breakdown of the product. In this case, you can use an item to represent a function. Teamcenter provides a predefined item type called **Functionality** for this purpose. You can also define custom types of an item to meet specific needs.

  You can substitute Mechatronics items by using the same item type or its subtypes for the **Substitute** operation. The following table lists the possible Mechatronics items for the **Substitute** operation:

Object	Recommended types for Substitute
Functionality	**Functionality** or its subtype
Connection	**Connection** or its subtype
Network	**Network** or its subtype

Object	Recommended types for Substitute
PSConnection	**PSConnection** or its subtype
Signal	**Signal** or its subtype
PSSignal	**PSSignal** or its subtype
Message	**Message** or its subtype
HRN_Accessory	**HRN_Accessory** or its subtype
HRN_AssemblyPart	**HRN_AssemblyPart** or its subtype
HRN_CavityPlug	**HRN_CavityPlug** or its subtype
HRN_CavitySeal	**HRN_CavitySeal** or its subtype
HRN_CoPackPart	**HRN_CoPackPart** or its subtype
HRN_ConHousing	**HRN_ConHousing** or its subtype
HRN_Fixing	**HRN_Fixing** or its subtype
HRN_GenTerminal	**HRN_GenTerminal** or its subtype
HRN_GeneralWire	**HRN_GeneralWire** or its subtype
HRN_Harness	**HRN_Harness** or its subtype
HRN_Module	**HRN_Module** or its subtype
HRN_WireProtect	**HRN_WireProtect** or its subtype
HRN_Cable	**Cable** or its subtype
HRN_Shield	**Shield** or its subtype
AppSoftware	**AppSoftware** or its subtype
Calibration	**Calibration** or its subtype
ConfigFile	**ConfigFile** or its subtype
PriBootloader	**PriBootloader** or its subtype
SecBootloader	**SecBootloader** or its subtype
Processor	**Processor** or its subtype

- Item element

  Item elements or general design elements are manageable, nonrevisable, releasable objects that represent a feature or an aspect of an item. In electromechanical product representations, use item elements to model ports, interfaces, terminals, or process variables associated with a signal. The Teamcenter model provides predefined **Network_Port** and **Connection_Terminal** types to model

functional and physical interfaces. You can also define custom types of an item element to meet specific needs.

- Connection

  Connections link a set of item elements or components in a product structure. Use a connection to define logical connectivity between interfaces, physical connectivity between terminals, and functional connectivity between ports. Teamcenter provides two predefined connection types for use in electromechanical product structures:

  - **Network connection** models connectivity between ports of two or more functions.
  - **Connection** models logical or physical connectivity between interfaces of two or more electrical connections or terminals of two or more electrical devices.

- Signal

  Signals are representations of messages. Messages are transmitted in functional and physical models. You can instantiate signals in a product structure and associate them with other electrical product constituents, including connections of functions, terminals, and connectors, to capture various relationships such as the transmitter of the signal and the receiver of the signal. You can also define custom signal types to meet specific needs.

Teamcenter also allows you to enforce restrictions on the usage and instancing of these objects in an electromechanical product structure. For example, you can tie valid item element types to a connection type with preferences.

Representing Mechatronics Process Management components

Representing components

Teamcenter provides several objects that allow you to represent components of an electromechanical product.

Representing electrical components

Electrical components in the context of wire harness modeling include various electrical devices, connectors, splices, interconnects, wires, cables, and other electrical accessories. In general, you can categorize all these components as parts and model them as *items* in Teamcenter. The Teamcenter data model does not provide any specific item types to represent electrical components but you can define your own item types. You can also use the Classification application to manage these objects.

The KBL harness model defines item types to represent various types of electrical parts, including wires, harnesses, and modules. These item types are not defined by default but can be installed as an option during installation.

In a functional model, electrical components may be *functions* or *functional subsystems*. To manage these objects, the Teamcenter model provides a predefined item type called **Functionality**, a subtype of **item** that you can use to model the functional breakdown of a product. It also provides the **FunctionalityRevision** item type, which corresponds to a particular revision of a **Functionality** object.

Representing signals and process variables

Signals represent information or messages transmitted between electrical connectors or devices, for example, an electrical signal passing through the wires joining a battery and a lamp. The Teamcenter implementation of a signal conforms closely to the AP212 definition.

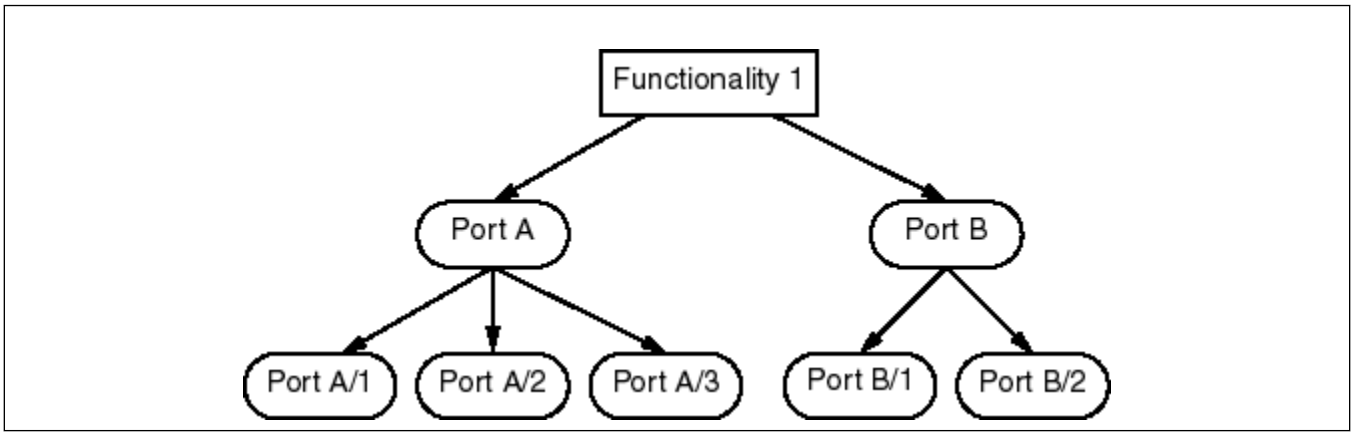
Signals can be instantiated in a product structure and associated with electrical product constituents such as connections, terminals, and connectors to capture various relationships, for example, the transmitter of the signal and the receiver of the signal.

A *process variable* is a parameter used to control or monitor a process, which may serve as an input, output, or control of the system. To be processed by the electromechanical system, process variables must be converted to signals.

Representing electrical interfaces

Electrical interfaces are exposed by electrical components. For example, the terminals associated with an electrical connector are the interfaces of the connector. The Teamcenter data model provides *item elements* to model electrical interfaces. You can manage, release, and instantiate item elements in an electrical product structure.

An example of hierarchical ports defined in a function is shown in the following figure.



Ports in a functions

The following objects allow you to model electrical interfaces:

Object	Purpose
Interface Definition	Models an interface of a product that can be connected, for example, a parallel port on a computer.
Network_Port	It can be used to model interfaces of a product in a functional model.
Connection_Terminal	It can be used to model interfaces of a product in a physical model.

Representing electrical connections

Connections define general connectivity between a set of item elements or components in a product structure. You can define logical connections, functional networks, and physical connections.

The AP212 model separates the connectivity data from the device that implements a connection. The Teamcenter data model allows for separation of connectivity data similar to the AP212 model.

Teamcenter also provides a **TC_Implemented_By** relationship to capture the association between a logical connection and a physical device that implements the connection.

The following objects allow you to model electrical connections:

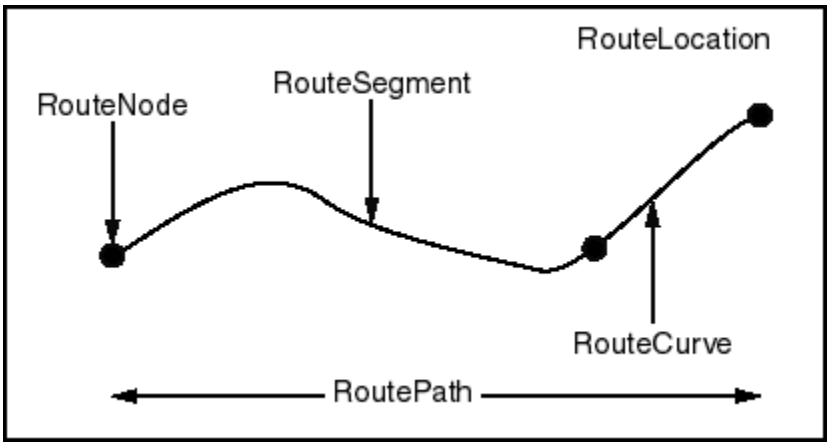
Object	Purpose
PSConnection	Models connectivity between one or more interface definition objects, for example, connectivity between a parallel port of a computer and parallel port of a printer.
Network	A subtype of **PSConnection**. You can use it to model connectivity in a functional model.
Connection	A subtype of **PSConnection**. You can use it to model connectivity in a physical model.
TC_Link	Models connectivity between one or more interface definition objects. It serves the same functional purpose as **PSConnection** except that you cannot revise it.

Representing routes

Route information defines the physical route that a wire or cable traverses through a product. The following objects allow you to model routes:

Object	Purpose
RouteNode	Models a point in space.
RouteSegment	Defines the segment of a path with a start node and an end node. You define nodes with **RouteNode** objects. You can optionally define the shape of the segment with a **RouteCurve** object. If you do not define a **RouteCurve** object, Teamcenter assumes that the segment shape is a straight line between the start node and the end node.
RouteCurve	Models the 3D shape (**b_spline curve**) associated with a **RoutePath** object or **RouteSegment** object.
RoutePath	Models a 3D or 2D physical path associated with a wire. Define a **RoutePath** object as a set of contiguous **RouteSegment** objects. You can also define it with a set of **RouteNode** objects, in which case an optional **RouteCurve** object defines the shape of the path. When you do not define a **RouteCurve** object, Teamcenter assumes that the path consists of straight line segments between **RouteNodes** objects.
RouteLocation	Models a region of space. You can associate objects such as **RouteSegments**, **RouteNodes**, and other electrical items with a **RouteLocation** object to identify the region of space in which they are located.
RouteLocationRev	Models a revision of a **RouteLocation** object.

The following figure shows how these objects apply to a wire or cable.



Route objects

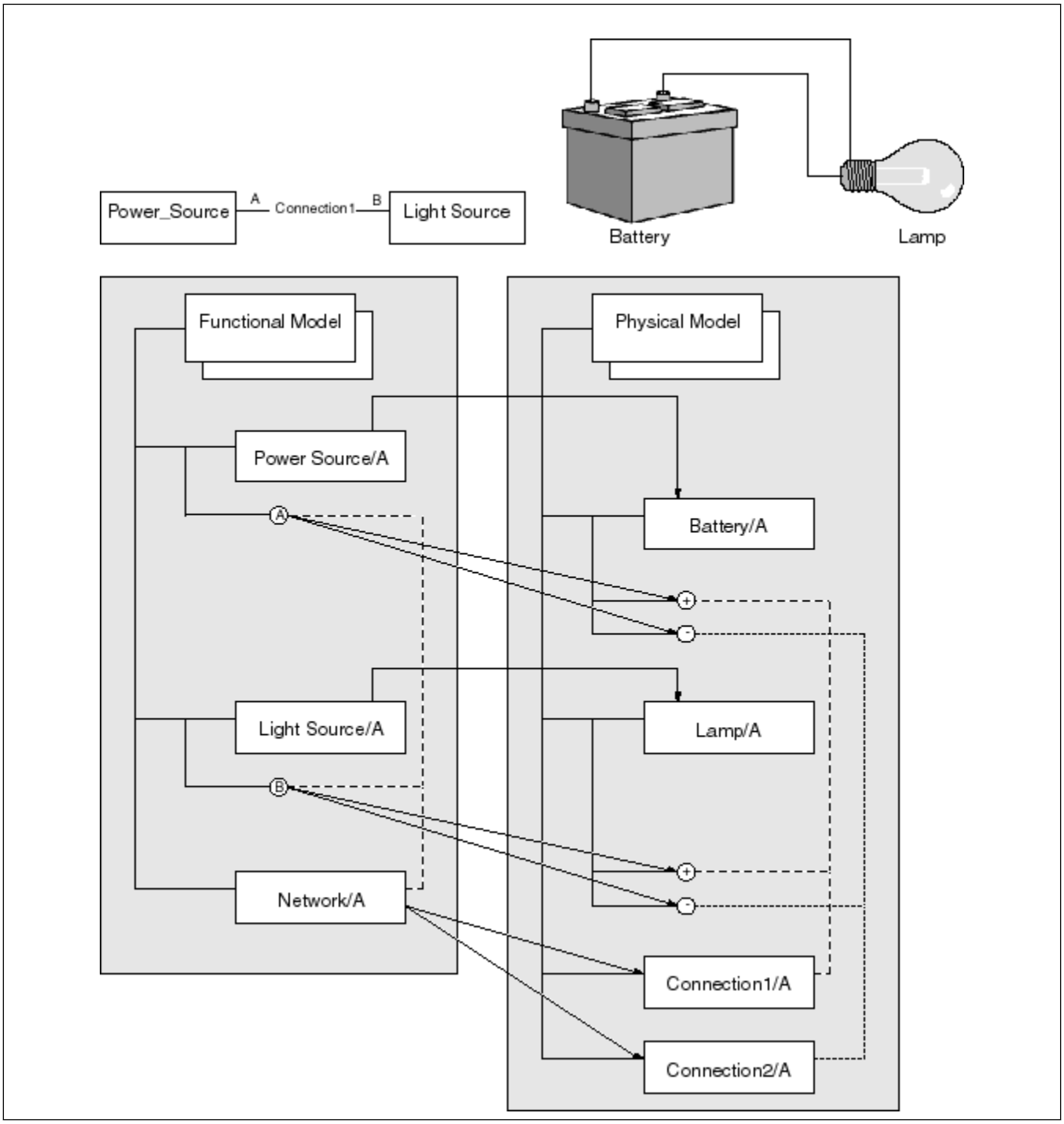
Representing allocations

Allocations represent mappings from one view in a product structure to another. For example, you can build allocations from functions, networks, and ports in a functional view to devices, connections, and terminals in a physical view. Allocations represent directional relationships between specific occurrences in two different views. They are independent of the structures they map and are managed independently. You create allocations as a set to map one structure to another, and define an **AllocationMap** object to group them together. Allocation maps are revisable and are subclassed from an item. You can also configure allocations independently of the structures they map.

The following objects allow you to model allocations:

Object	Purpose
Allocation	Maps components between different product structures. For example, you can use this object to model the association between components in the functional structure and the physical structure of a product.
AllocationMap	Groups a set of allocations created in the same context.

The following figure shows an example of how to create allocations between the functional model and the physical model.



Allocation objects

Modeling relationships

The following relationships allow you to associate fundamental objects:

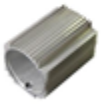
Relationship	Purpose
Implemented By	Specifies the devices that implement a particular connection.
Realized By	Specifies the interfaces that realize interfaces exposed at a lower level in the assembly structure of a product.
Connected To	Specifies the interfaces connected by a connection.
Routed By	Specifies the route associated with a connection or device.
Device To Connector	Specifies the connectors associated with a device in an electromechanical product.
Assigned Location	Specifies the location (region of space) associated with a routing topology object or product constituent.
Associated System	Specifies the associated system (for example, transmitter or receiver) for a particular signal.
Redundant Signal	Specifies a signal that serves as a redundant signal for another signal.
Process Variable	Specifies the process variable associated with a signal.
SCM_Element_Specification	Associates the Teamcenter SCM object with the binary to establish traceability.
Embeds	Represents an association of the software with processor in the context of a structure. It indicates the embedded software (secondary) that is embedded in a specified processor (primary).
GatewayOf	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.
DependentOn	Models dependencies between embedded software within the same ECU or across different ECUs.

6. Multidisciplinary Associations

Collaborating across disciplines using Multidisciplinary Associations

The design and production of most modern products involves components from multiple disciplines. Designers of multidisciplinary products may work in different teams, but their designs have significant interdependence. For example, assembly line designers design the general assembly line, while automation designers design the robots, sensors, and other line-station objects for the assembly line. Both sets of users work in different design projects and may use different designing tools. However, they must ensure that their designs are compatible with those created by the designers from the other discipline. Ensuring compatibility during the design stage is critical to avoid rework that costs time and money.

Consider an example of an electrical motor design project. It requires close collaboration between the mechanical and electrical disciplines. Electrical motors use components such as casings and rotors from the mechanical components library and components such as stators and capacitors from the electrical components library.

Mechanical components library	Electrical components library
 Rotor 01	 Stator 01
 Rotor 02	 Stator 02
 Rotor 03	 Capacitor 01
 Casing 01	 Capacitor 02
 Casing 02	 Capacitor 03

The stator of the electrical motor must be physically and functionally compatible with the casing and the rotor. In such multidisciplinary projects, when designers decide to use a component in their design, they must be aware of its compatibility with the other components in the design.

Although data is centrally managed in a common repository, the development may occur in isolation resulting in a disconnect between the different disciplines. A mechanism is therefore required:

- To identify compatible components from the available components library.
- To notify all the stakeholders of updates.

The Multidisciplinary Associations feature provides collaboration between various disciplines during the product design phase. Using this, you can group compatible objects and their usages in a design. This grouping helps designers identify the components that are compatible with those used in their design.

Multidisciplinary Associations facilitate the sharing of accurate and timely information by generating notifications in the case of design updates. Notifications are available through subscriptions to all designers working on the project. The notifications ensure that component updates are transparent across disciplines.

This feature leverages the domain information of components. In some cases, design components do not have any domain information associated with them. This makes it difficult for the users to trace the parent domain of the components. For such components, you can use the **associate_domain_data** utility to assign domain information.

Setting up and using multidisciplinary associations: participants and roles

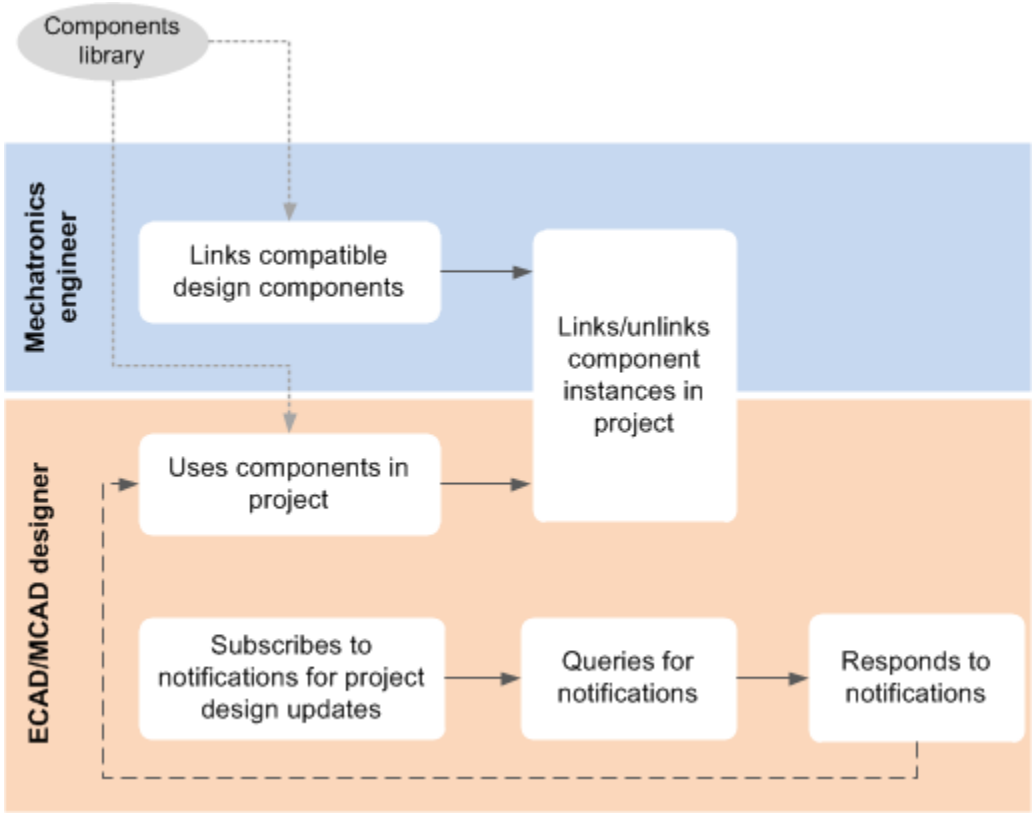
The multidisciplinary collaboration projects involve various Teamcenter and design applications. Teamcenter services are available for linking and notification-related tasks.

The Teamcenter administrator is responsible for installing, customizing, and maintaining the multidisciplinary associations feature.

The Mechatronics engineers manage the multidisciplinary objects. They the compatible components together using a grouping mechanism called **MDThread linking**.

Designers work with design applications to create and update design components and save their designs in Teamcenter. When they use components in a design, design instances are created. Designers or mechatronics engineers link these design instances using a grouping mechanism called **MDInstance linking**.

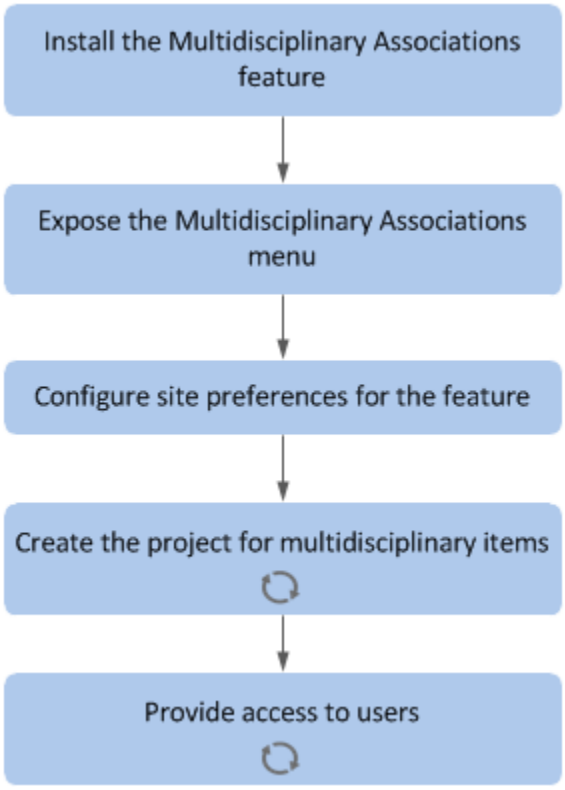
When designs undergo any changes, notifications are generated. Designers must subscribe to these notifications based on their project requirements because notifications are available only through subscriptions. Once subscribed, designers can query for and respond to notifications. Notifications are supported only for application model.



Installing the Multidisciplinary Associations feature

How to install the Multidisciplinary Associations feature

The Teamcenter administrator is responsible for installing, customizing, and maintaining the Multidisciplinary Associations feature. The administrator exposes the Multidisciplinary Associations menu using the Command Suppression application. The administrator also creates the multidisciplinary project that defines the scope of notifications and controls user access.



Multi-Disciplinary Associations is an optional Teamcenter feature. You can install it when you install Teamcenter. Alternatively, you can add this feature to an existing Teamcenter installation. You need Teamcenter administrator privileges to install the Multidisciplinary Associations feature.

If you are installing this feature with a new instance of Teamcenter, follow the detailed steps listed in the *Teamcenter installation documentation*. When you reach the **Features** panel in TEM, proceed as follows:

1. In the **Features** panel, expand **Extensions→Mechatronics Process Management**.
2. Select **Multi-Disciplinary Associations** from the **Mechatronics Process Management** list.
3. Enter the relevant information in the subsequent panels.
4. In the **Confirmation** box, click **Start**.

   TEM installs the Multi-Disciplinary Associations feature.

To add this feature to an existing Teamcenter instance, perform the following steps.

1. Start Teamcenter Environment Manager (TEM).

   Launch Teamcenter Environment Manager. In the Windows start menu, click **Programs→Teamcenter 11**, right-click **Environment Manager**, and choose **Run as administrator**.

2. In the **Maintenance** panel, select **Configuration Manager**.

3. In the **Configuration Maintenance** panel, select **Perform maintenance on an existing configuration**.

4. Proceed to the **Features** panel.

5. In the **Features** panel, expand **Extensions→Mechatronics Process Management**.

6. Select **Multi-Disciplinary Associations** from the **Mechatronics Process Management** list.

7. Enter the relevant information in the subsequent panels.

   > Tip:
   >
   > For information about any TEM panel, click the help button  .

8. In the **Confirmation** box, click **Start**.

   TEM installs the Multi-Disciplinary Associations feature.

Installation of the Multidisciplinary Associations feature

The Multidisciplinary Associations feature can also be installed using Teamcenter Environment Manager (TEM). The feature can be selected in the extensions list on the **Features** panel.

Administering and configuring the Multidisciplinary Associations objects

Controlling access to Multidisciplinary Associations objects

The access to multidisciplinary association objects such as **MDThreads** and **MDInstances** is controlled by three authorization rules: **MDOAdministration**, **MDIAdministration**, and **MDONotificationAdministration**. These rules are applied to the user groups or roles and not to individual users. The users must be added to one of the authorized groups and roles to receive access to Multidisciplinary Association features.

Control access to linking components using MDThreads linking

The **MDOAdministration** authorization rule is used to control access to the **MDThreads** related tasks. Only users that belong to a group added to **MDOAdministration** authorization rule can access the **Manage Multi-disciplinary Associations** dialog box and perform the following tasks:

- Associate design components to create an **MDThread**.

- Search for the **MDThread**.
- Add and remove design components (edit the **MDThread**).
- Delete the **MDThread**.

You can add a user-group or role to the authorization rule as follows:

1. Go to **Authorization**.
2. In the **Authorization** application pane, expand the **Organization** tree and click the group or role to whom you want to grant or deny access.
3. Select **MDOAdministration** from the **Applications with Read Only Access** list to grant access to the group or a role in a group. Click the right-arrow button to move the utility to the **Applications with Full Access** list.

   > Tip:
   >
   > If the **Applications with Read Only Access** list is empty, click any group or role symbol in the **Organization** tree to refresh the list.

4. Click **Save**.

Control access to linking the usages of components using MDInstance linking

The **MDIAdministration** authorization rule is used to grant the user access to **MDInstance** related tasks. Unlike **MDThreads**, **MDInstances** are managed using services. Only users who belong to a group added to the **MDIAdministration** authorization rule can perform the following tasks:

- Link design instances to create **MDInstances**.
- Unlink design instances from **MDInstances**.
- Search for the impacted design instances in the case of a linked design instance undergoing a change.

You can add a user-group or role to the authorization rule as follows:

1. Go to **Authorization**.
2. In the **Authorization** application pane, expand the **Organization** tree and click the group or role to whom you want to grant or deny access.

3. Select **MDIAdministration** from the **Applications with Read Only Access** list to grant access to the group or a role in a group. Click the right-arrow button to move the utility to the **Applications with Full Access** list.

   > Tip:
   >
   > If the **Applications with Read Only Access** list is empty, click any group or role symbol in the **Organization** tree to refresh the list.

4. Click **Save**.

Add users to the MDO Notification Administration group

The **MDONotificationAdministration** authorization rule is used to add user groups to the Notification administration group. Only users who belong to a group added to the **MDONotificationAdministration** authorization rule can subscribe to notifications. Multidisciplinary Associations notification administrators can also delete the notifications using the **mdonotification_cleanup** utility.

You can add a user-group or role to the authorization rule as follows:

1. Go to **Authorization**.

2. Click the **Applications** link in the **Quick Links** section of the navigation pane.

3. In the **Authorization** application pane, expand the **Organization** tree and click the group or role to whom you want to grant or deny access.

4. Select **MDONotificationAdministration** from the **Applications with Read Only Access** list to grant access to the group or a role in a group. Click the right-arrow button to move the utility to the **Applications with Full Access** list.

   > Tip:
   >
   > If the **Applications with Read Only Access** list is empty, click any group or role symbol in the **Organization** tree to refresh the list.

5. Click **Save**.

Display the Multidisciplinary Associations menu

You can manage the **MDThread** objects in Teamcenter using the user interface. through a dialog box. The commands for this dialog box are suppressed by default. You must expose these commands using **Command Suppression**. They are then accessible from My Teamcenter. To display the Multidisciplinary Associations menu:

1. Go to **Command Suppression**.

2. In the **Command Suppression** pane, select **My Teamcenter** from the **Applications** list.

3. Select the desired group, subgroup, or role from the **Organization** list.

4. Select **Tools→Manage Multi-Disciplinary Associations** from the **Suppress Commands** tree.

5. Click **Show**.

   The red line disappears from the **Manage Multi-Disciplinary Associations** node.

6. Choose **File→Close**. Select **Yes** when you are prompted to save the changes.

Customizing the client to view MDThreads

You can view the **MDThreads** and associated design components as child objects in My Teamcenter. You can also view **MDThreads** in the **Viewer** and **Summary** tabs. This is done using the following preferences.

View MDThreads and design components in My Teamcenter — You can view **MDThreads** and associated design components as child objects in My Teamcenter. Set the **Mdo0MDThread_DefaultChildProperties** preference value to **Mdo0MDTAssociation**.

View MDThreads in summary tab — Set the **Mdo0MDThread.SUMMARYRENDERING** preference value to **MDThreadSummary**.

Associate component types other than library elements, items, and item revisions

Out of the box, **MDThreads** can be used to associate three component types: item, item revisions, and library elements. You can associate other component types by adding them to the applicable stylesheets and business constants.

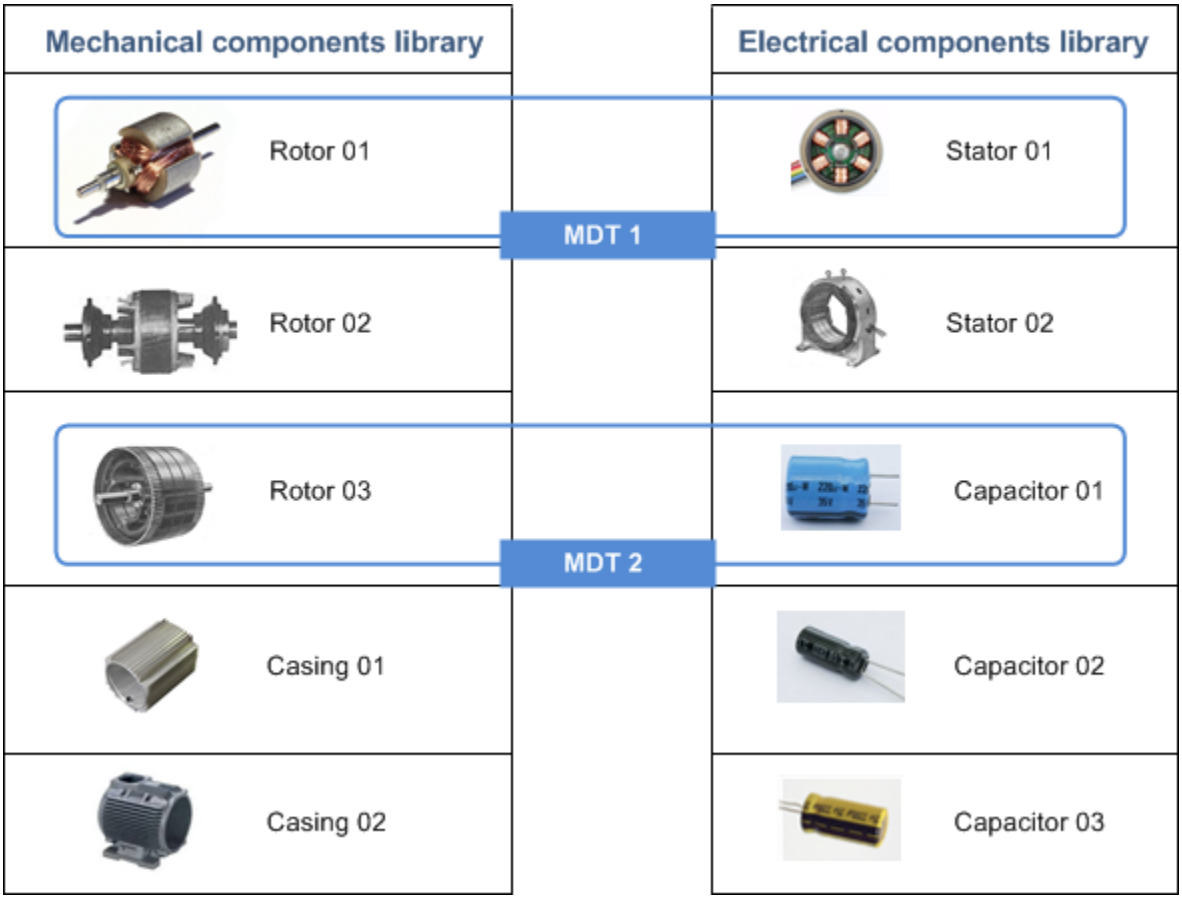
If you create **MDThreads** using the Teamcenter client, you must add the new component types to the **MDThreadCreate.xml** and **MDThreadSummary.xml** stylesheets along with the **Mdo0ValidMDTAssocTypes** business constant.

If you create **MDThreads** using Teamcenter services, to support more component types, you must add the new component types to the **Mdo0ValidMDTAssocTypes** business constant.

Creating and managing Multidisciplinary Associations

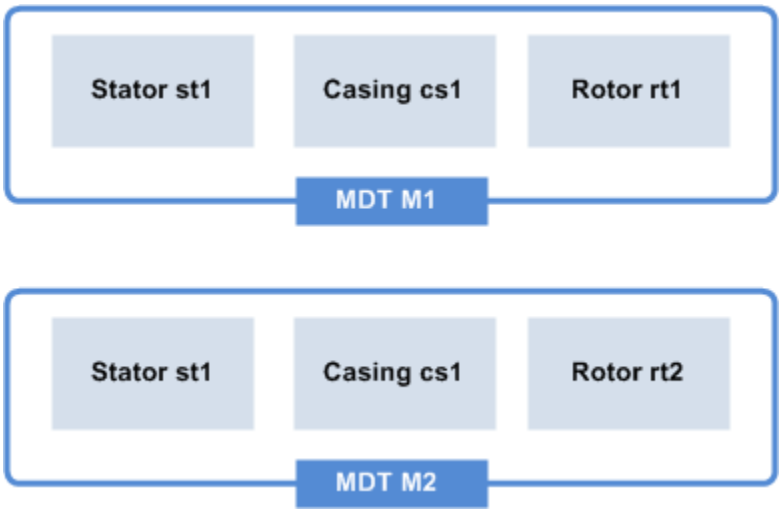
Linking design components using MDThread linking

The linking of multidisciplinary components is called **MDThread** linking. It is a grouping mechanism to identify compatible components, for example, the casing, stator, and rotor of an electrical motor.



By default, **MDThread** linking supports three object types: library elements, items, and item revisions. Using **MDThreads** linking, designers can view all associated design components. In doing so, designers can identify the components that are compatible with the other components in their designs. **MDThread** linking also facilitates notifications when any design component associated with an **MDThread** is used in a design.

For example, an electrical motor project called **motor1** uses casing and rotor components from the mechanical components library, and a stator component from the electrical components library. **MDThread** objects are used to group the casing, stator, and rotor components. The casing **cs1** and stator **st1** are compatible with two rotors: **rt1** and **rt2**. This means that two design combinations are possible: **cs1, st1, rt1** and **cs1, st1, rt2**. These compatible design components are linked using two **MDThreads**: **MDT1** and **MDT2**.



Using this **MDThread** linking, the designer of stator **st1** can see that the stator is compatible with casing **cs1** as well as with two rotors: **rt1** and **rt2**.

When stator **st1** is used in a design, the designers must select a compatible casing and a rotor. Similarly, if a rotor or a casing is used in the design, notifications are generated, which can be viewed by all the impacted designers.

Create an association between design components

The **MDThread** is used to associate two or more design components. As **MDThread** is a linkage between multiple design components, you must link at least two design components to create a new **MDThread**. By default, item, item revisions, and library elements are supported for **MDThread** associations. The procedure remains the same for any design components.

1. In My Teamcenter, select **Tools→Manage Multi-Disciplinary Associations**.

2. Drag the items (or item revisions) from **My Teamcenter** to the **Associated Designs** area in the **Manage Multi-Disciplinary Associations** dialog box.

   Alternatively, you can copy the item and click **Paste** in the **Associated Designs** area.

3. Enter a name and description for the **MDThread**.

4. Click the **Save** button.

   If the **Save** button is disabled, confirm that you have a minimum of two design components in **Associated Designs** and that the **Name** field is populated.

   The new **MDThread** is added to the **Worklist**.

Search for MDThreads

1. In My Teamcenter, select **Tools→Manage Multi-Disciplinary Associations**.

2. To search for an **MDThread** by name, enter the name in the **Search Multi-Disciplinary Associations** area.

3. To search by associated components, drag the item (or item-revision) from **My Teamcenter** to the **Designs** box in the **Search Multi-Disciplinary Associations** area.

   Alternatively, copy the item and click **Paste** .

4. Click **Search** .

The following two services are also available to search for **MDThread** objects.

- Use the **searchMDO3** service to search for **MDThread** objects by linked design components.
- Use the **searchForArtifactsUsingInstances3** service to search for **MDThread** objects by design instances.

Delete an MDThread

The **MDThread** object is an association between at least two design components. If you have less than two associated design components, the **MDThread** object is automatically deleted. To delete an **MDThread** object:

1. In My Teamcenter, select **Tools→Manage Multi-Disciplinary Associations**.

2. Search for the and select the **MDThread** object from **Search Results**.

3. Click **Delete**.

   The deleted **MDThread** is removed from the **Worklist**.

Edit MDThreads

1. In My Teamcenter, select **Tools→Manage Multi-Disciplinary Associations**.

2. Search for the and select the **MDThread** object from **Search Results**.

3. Click **Edit**.

4. To remove an associated design component from the **MDThread**, select the component in **Associated Designs** and click **Cut**.

5. To associate a new design component, drag the item (or item-revision) from **My Teamcenter** to the **Associated Designs** area.

   Alternatively, you can copy the item and click **Paste** in the **Associated Designs** area.

6. Click **Save**.

Linking design instances

Linking component usages using MDInstance linking

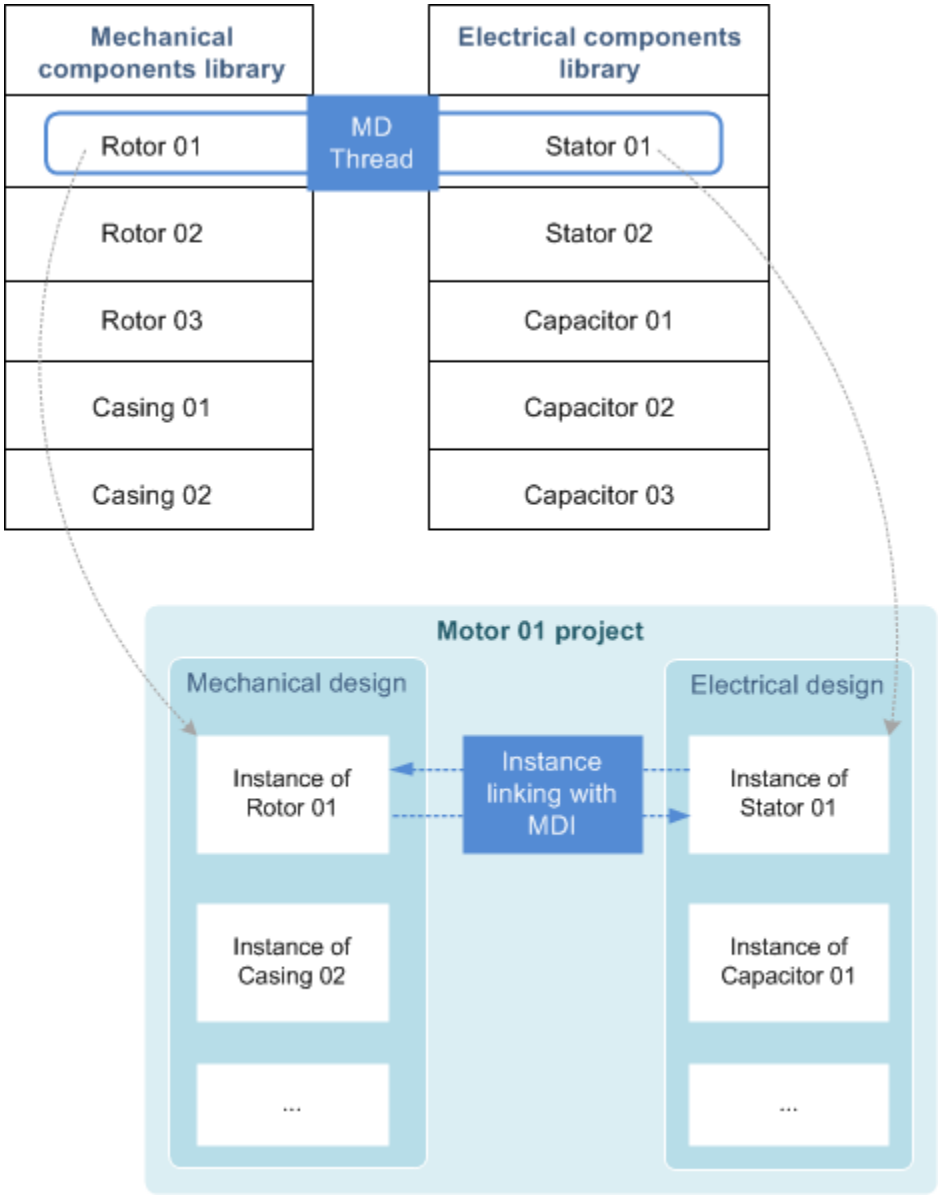
MDInstance linking is used to link the usages of components in a design. Every time a library component is used in a design, a new instance is created. These instances are linked using the **MDInstance** linking.

Linking is supported for both application model and product structure assembly model. It identifies impacted instances.

When a linked instance is updated, notifications are generated for it. Notification generation is supported only for design instances in application model.

When a design component is used in an application model, a new usage is recorded and a *create notification* is generated. Similarly, when an existing instance is updated, a *modify notification* is generated. The other designers in the project can view these notifications and react to them by modifying their designs if needed.

> Example:
>
> An electrical motor project, **motor1**, uses mechanical components such as casings and rotors, and electrical components such as stators and capacitors. An **MDThread** object is used to group a compatible casing, stator, and rotor and an **MDInstance** object is used to link the usages of these components in the **motor1** project.



A mechanical engineer working on the mechanical design for **motor1** uses the component **casing01**. This creates an instance **motor1casing01** and a *notification* is generated for the instance.

The electrical engineer working on the electrical design for **motor1** seeks relevant notifications from the project. The system returns information stating that **casing01** has been selected for this project. **MDThread** linking shows that the electrical components **stator01** and **stator02** are both compatible with **casing01**. The electrical engineer decides on using **stator02** for the **motor1** project. This results in the creation of another instance, **motor1stator02**, and another *notification* is generated.

The two instances **motor1casing01** and **motor1stator02** define the usages of **casing01** and **stator02** in the **motor1** project. A user with MDI administrator privileges links these instances with each other and creates an **MDInstance** link.

Similarly, other component instances in the project are added to the same **MDInstance** linking object. Additionally, when a linked instance is modified, the **MDInstance** linking object helps to generate a *notification*. The designers can query for notifications of both kinds, when instances are created or modified. If a notification indicates an impact, the designer may modify their designs or ignore the notification.

Only users belonging to the **MDIAdministration** authorization rule are authorized to link or unlink the instances.

Use the following **linking services** for **MDInstance** linking-related tasks:

- **linkDesignInstances** or **linkDesignInstances2**
- **unlinkDesignInstances**
- **searchForLinkedDesignInstances** or **searchForLinkedDesignInstances2**

Use the following **linking services** for **MDInstance** linking-related tasks for product structure assembly model:

- **link4GAndBVRDesignInstances**
- **unlink4GAndBVRDesignInstances**
- **getLinkedInstancesForBVR** and **getLinkedInstancesFor4G**

What happens when a linked product structure assembly model object undergoes changes?

The following table shows how the changes to structure assembly model objects impact the linking instances.

Update to structure assembly model instance	Impact on MDI link
Product structure assembly design instance's underlying object is deleted	The link instances from the deleted object to any other objects are deleted.
Product structure assembly design instance is removed from the structure.	The link instances from the removed object to any other objects are deleted.
Product structure assembly design instance's underlying object is revised	The MDI link is carried to new revision and **needsValidation** is set to true and **isValidated** is set to false.

Update to structure assembly model instance	Impact on MDI link
Restructure operation such as **Insert Level**, **Remove Level**, or **Move To** is performed on the product structure assembly design instance.	The MDI link is updated to reflect that the structure is updated. The **needsValidation** is set to true and **isValidated** is set to false.
The underlying object of product structure assembly design instance is replaced with a new object.	The MDI link is carried to all revisions of the replacing item and **needsValidation** is set to true and **isValidated** is set to false.

Working with notifications in Multidisciplinary Associations

How notifications work

The notifications functionality is based on subscriptions. When design components, linked using **MDThread** objects, are used in a design, a notification indicating the component's usage is generated.

The scope of notifications is restricted to a project. The project must be created with **Multi-Disciplinary** as the collaboration category. Notifications are generated only for components that are a part of this project.

Notifications are also generated when design instances are removed from or modified in a design. These notifications are based on **MDInstance** linking. Notifications can be updated and searched for using Teamcenter services.

Only users belonging to the **MDONotificationAdministration** group are authorized to subscribe to notifications and work with notification-related utilities such as **configure_mdo_subscription** and **mdonotification_cleanup**.

Configure a site for notifications

The following preferences must be set to configure the notifications functionality.

- Set the **TC_subscription** preference value to **ON**. This is required to enable the subscription mechanism. Notifications are not generated unless this preference is set.

- Set the **TC_subscriptionmgrd_sleep_minutes** preference value to **1**. This preference defines the time between two consequent runs of the subscription manager daemon. The subscription manager daemon is a separate background process that reads the events from the database. The default value is 10 minutes which must be reset to 1 minute.

- Set the **TC_subscriptionmgrd_max_subscriptions_to_dispatch** preference to value **499**. This preference setting is required only for performance testing if you are using **MDO management SOA sample client**.

- Set the **Mdo0Notification_SourceObject** preference to enable *create notifications*. This preference defines the class name followed by the attribute name for that class. Doing so identifies the underlying design component. For example, the preference value **Cpd0DesignModelElement:Cpd0source_object** implies that for the class name **Cpd0DesignModelElement**, the attribute **Cpd0source_object** should be referred to.

- Set the **Mdo0Notification_PresentedParent** preference to enable *create notifications* for model-to-model realization events. Design instances from one application model can be realized into another application model. Specify the **Mdo0Notification_PresentedParent** preference value in the following format: class name followed by the presented parent attribute, for example, **Cpd0DesignModelElement:cpd0presented_parent**.

- Set the **MdoNotification_TypesSupportedForCreate** preference to support notifications for the object type irrespective of **MDThread** associations. By default, *create notifications* are supported only for design instances whose components are associated to an **MDThread**. Once you add an item type to this preference, when an object of this item type is created, a *create notification* is generated, irrespective of **MDThread** association. To do so, add the name of the item type to the preference in the following format: **MdoNotification_TypesSupportedForCreate=Ptn0Partition**.

  If you want to set *replace notifications* for new business objects, a subclass of model elements, you can do so using the setter() function in BMIDE. The subclass of the model element must override the setter() for the underlying source object of the model element. The setter() function is used to set a flag on the **Mdl0ModelElement** using **setMdl0SourceObjectChangeStatus()**.

In addition to setting these preferences, you must:

- Ensure that the subscription manager daemon is running to process the notification events.

- Create a project with the collaboration category as **Multi-Disciplinary**, and associate all the design components to it.

Create a new multidisciplinary collaboration project

Create notifications are generated when a component is used in a design. To set *create notifications*, you must first create a project with **Multi-Disciplinary** as the collaboration category. You must then assign all the component designs to this project.

For example, you create three new motor projects: **motor01, motor02,** and **motor03** but you select **Multi-Disciplinary** as the collaboration category only for projects **motor01** and **motor03**. In this case, *create notifications* are triggered only for **motor01** and **motor03** and not for the **motor02** project.

1. Click the **Definition** tab in the **Project Administration** quick link.

2. Enter **Name**, **ID**, and **Description** for the project.

3. Select **Multi-Disciplinary** as the collaboration category.

4. Click **Modify** for an existing project, or click **Create** for a new project.

Subscribing to events for notifications

Notifications are available by subscriptions only. There are no default subscriptions. You must subscribe to the events for which you wish to receive notifications. Using the **configure_mdo_subscription** utility, you can subscribe to the notifications by object type and event type. For example, you can subscribe to *create notifications* by specifying event type as **Create**. Following object types and event types are supported:

Object type	Event type
targettype = Mdl0ModelElement	-eventtypes = _Create
-targettype = Ptn0Functional	-eventtypes = _Create
-targettype = Mdl0ModelElement	-eventtypes = Mdl0Unrealize_Model_Element
-targettype = Mdl0ModelElement	-eventtypes = Mdl0Revise_Model_Element
-targettype = Mdl0ModelElement	-eventtypes = Mdl0Replace_Model_Element
-targettype = Mdl0ModelElement	-eventtypes = Mdl0Attach_Attribute_Group
-targettype = Mdl0ModelElement	-eventtypes = Mdl0Detach_Attribute_Group
-targettype = Rlz0RealizationContainer	-eventtypes = Rlz0Create_Rlz_Container
-targettype = Rlz0RealizationContainer	-eventtypes = Rlz0Delete_Rlz_Container

Querying for notifications

You can query for *create notifications* and *modify notifications* based on either the design instance or the project context or a combination of both. The queries are executed using the following Teamcenter services: **qryNotificationsByDesignOrProject**, **qryNotificationsByImpactedDesign**, **qryNotificationsByOriginDesign**.

Responding to notifications

A notification about an object might trigger a reaction from the designers of impacted objects. If the event has an impact, the designers may update their designs based on the notification. In there is no impact, the designer may ignore the notification. Similar to querying for notifications, responding to them may also be a voluntary and optional task based on the processes decided by the project team. This response is recorded using the **updateMDONotifications** service.

Deleting notifications

The administrator users who are added to the **MDONotificationAdministration** authorization rule can delete notifications. This is done using the **mdonotification_cleanup** utility. The utility can delete all the notifications simultaneously or selectively based on the following criteria:

- Notifications prior to the specified date

- Notifications filtered by the action that triggered the notification
- Notifications filtered by the component for which notifications are created

The utility can be run in report mode as well. This mode provides notification details such as design instance, the action performed on the design instance, and the **MDInstance** to which the design instance is linked.

Working with Multidisciplinary Associations in a multisite environment

Exchanging multidisciplinary data with remote sites

Teamcenter supports data exchange for **MDThreads** and associated design components. When a design component is exchanged between sites, the associated design components are exchanged as well.

For **MDInstance** related data exchange, a design instance is the starting point. You select the design instance and initiate the data exchange. When the design instance is exchanged, the associated **MDInstance** object and all linked design instances are also exchanged. Additionally, all notifications associated with the design instance are also exchanged.

For, **MDThreads**, the user can select either **MDThreads** or an associated design component as the starting point for data exchange. In either case, the **MDThreads** and all associated design components are exchanged. The prerequisites for an **MDThreads** related data exchange are as follows:

- The user belongs to the required administrator group.
- The **MDThread** is released.
- Both sites are configured for a multisite exchange.

Exchange multidisciplinary data through Multi-Site Collaboration

You can use Multi-Site Collaboration environment to share **MDThreads** and associated design components data with other sites. You can either start with the **MDThread** object or an associated design component for data exchange.

1. Select the **MDThread** object or an associated design component for data exchange.
2. Export the selected object from your site to a remote site.
3. Import the objects from the remote site.
4. Transfer ownership of the object you own to the remote site.

Exchange multidisciplinary data using TcXML

For data exchange using site consolidation and TcXML, the required transfer mode and closure rules must exist in Teamcenter. The transfer mode **TIEMDOExportDefaultTM** is available for data exchange using Teamcenter XML.

1. Select the **MDThread** object or an associated design component for data exchange.

2. Export the selected object from the source site to the destination site, using the **tcxml_export** utility.

3. Import the data at the target site, using the **tcxml_import** utility.

Exchange multidisciplinary data using a Briefcase

You can use the Briefcase feature to exchange the **MDThread** object, associated components, or design instances.

1. Select the **MDThread** object or an associated design component that you wish to export and then choose **Tools→Export→To Briefcase**.

2. In the **Export to Briefcase dialog box**, enter the required information, and click **Yes** to begin the export process.

3. Select the object that you wish to import, and then choose **Tools→Import→From Briefcase**.

Associate domain information with design components

A multidisciplinary product uses design components from various domains. Typically, object types are used to classify business or domain objects. In some cases, the object type information is incomplete or the objects are not organized appropriately by type. In such cases, the domain information is not available. This restricts the system from filtering the queries by domain. You can use certain tools to associate and identify domain information of the business objects in Teamcenter.

The domain information of a design component is always associated with the object type **Items** and not with **Item Revisions**. The following tasks related to domain-component association are supported:

- Associating an instance of a design component with a domain

- Querying for domain information using Teamcenter services

Add domains to the Mechatronics domains list — The **MECH_domain_list** defines the list of mechatronics domains, which can be associated with design components. By default, this list has two domains: **automation** and **mechanical**. Users can add more domains, using preferences.

Associate a domain to an instance of a design component — When design components are not properly categorized by types, it may not be possible to use associations at the item type level. In such cases, you can associate the domain with an individual design component, using the **associate_domain_data** utility. For example, in a motor project, the rotor, casing, and stator are from different domains. You can use the **associate_domain_data** utility to associate the casing with the **mechanical** domain and the stator with the **electrical** domain.

Query for domain information, using custom exit — For the sites that use a proprietary method for identifying the domain of design components, a **User_get_domain** custom exit is provided. The logic for a custom exit can be defined by the user, and it can be used to identify the domain of the design component. The exit is used by the **getDomainOfObjectOrType** service for domain queries.

Associate components to domains based on the item type — If design components are segregated by item types, the domain of the components can be identified based on item types.

By default, domains are not associated with item types. However, if you can identify the domain of an object based on item type, you can set a domain-specific preference with the item type. For example, if the **MECH_domain_types_Automation** preference is populated with the item type **PartRevision**, all design components of the item type **PartRevision** are identified as being in the automation domain. Similarly, if the item type **HRN_ConHousingRevision** is associated with **MECH_domain_types_Mechanical**, all the design components of the item type are associated with the mechanical domain.

Query for domain information using Teamcenter services — The **getDomainOfObjectOrType** service is provided for domain queries. When the query is raised, it is resolved in the following order based on:

1. Item type
2. Custom exit logic
3. Design component

The domain information is leveraged by the notifications functionality as well. If the domain information is available, then whenever a notification query is executed for a design component, it also returns the domain information. This facilitates filtering of notification information by domain.

Teamcenter services for Multidisciplinary Associations

Teamcenter services for application model design instances

Use the following Teamcenter services for **MDInstance** linking between only application model instances:

- **linkDesignInstances**

- **linkDesignInstances2**
- **unlinkDesignInstances**
- **searchForLinkedDesignInstances**
- **searchForLinkedDesignInstances2**
- **updateMDInstanceToMDThreadLink**

There are two types of instance linking: precise and imprecise.

Precise linking implies that a major revision of a design instance (Model Element) is associated with another design instance. Therefore, if instances A and B are linked using precise linking, any updates to A will impact B and any updates to B impact A.

Imprecise linking means that the changes will impact all revisions of design instances. For example, if A and B are linked using imprecise linking, all revisions of A and B get impacted by the changes. Any change to any revision of A will impact all revisions of B and any change to any revision of B will impact all revisions of A.

The **linkDesignInstances** service creates precise or imprecise linking of the design instances supplied in the input. You can specify the context for instance linking using the **linkDesignInstance** service. This means that linking of instances is valid only for the context specified by the service. If context is not specified, then default is **common** context. **Tc_project** is the context object.

The **linkDesignInstances2** service creates precise or imprecise linking of the design instances supplied in the input with the specified **MDThread** object. The **MDThread** input is optional. If specified, this service links the **MDThread** to the design instance, else it works exactly as the **linkDesignInstance** service.

The **unlinkDesignInstances** service dissociates the linked design instance from the **MDInstance** linking object.

The **searchForLinkedDesignInstances** service searches for the design instances that are linked with the design instance specified in the service.

The **searchForLinkedDesignInstances2** service searches for the design instances that are linked with the design instance specified in the service. It also returns the information about the linked **MDThread** object, its validation status, and the relevant and irrelevant domains for the searched design instance. If the configuration is specified in the search, all linked instances for the specified configuration appear in the search results.

The design instances are linked using the **MDInstance** objects, and design components are linked using the **MDThread** objects. The related **MDInstance** and **MDThread** objects can be linked together using the **updateMDInstanceToMDThreadLink** service, which associates the **MDThread** object to the specified **MDInstance** object.

The design components are linked using the **MDThread** objects, and design instances are linked using the **MDInstance** objects. The related **MDThread** and **MDInstance** objects can be linked together using the **updateMDInstanceToMDThreadLink** service, which associates the **MDThread** object to the **MDInstance** object.

Only users belonging to the **MDIAdministration** authorization rule are authorized to link or unlink the design instances with the **MDInstance** object. When two or more design instances are linked using services, it results in the creation of an **MDInstance** linking object. Linking is supported for both application model and product structure assembly model. Linking of design instances is required for generating notifications.

Teamcenter services for product structure assembly model design instances

In a multidisciplinary product development environment, designers from different disciplines work on their own designs. They need to link their design instances to other discipline's design instances to track any impact of the design changes. The designs can be application model based or product structure assembly based.

Use the following Teamcenter services for **MDInstance** linking with product structure assembly model (BVR model) instances:

- **link4GAndBVRDesignInstances**
- **unlink4GAndBVRDesignInstances**
- **updateMDInstanceToMDThreadLink**
- **getLinkedInstancesFor4G**
- **getLinkedInstancesForBVR**

The **link4GAndBVRDesignInstances** service links two or more design instances from multidisciplinary designs using a common **Mdo0MDInstance** object. It links a product structure assembly instance to:

- Application model design instances
- Other product structure assembly design instances

The **unlink4GAndBVRDesignInstances** service unlinks one or more design instances which are linked together through a common **Mdo0MDInstance** object. The **Mdl0ModelElement** of application model design instance and product structure assembly based design instances are supported.

The **getLinkedInstancesForBVR** service searches for linked design instances for the given product structure assembly model design instance. For the input design instance, the service will return all the linked application model and product structure assembly model design instances.

The **getLinkedInstancesFor4G** service searches for linked design instances for the given application model design instance. For the input design instance, the service will return all the linked product structure assembly model and application model design instances.

Teamcenter services to update MDThread and Mdo0MDThread objects

Use the following Teamcenter services to update the **MDThread** and **Mdo0MDThread** objects:

- **queryNeedsValidationLink2**
- **updateLinksToValidated2**

When design components are revised, the association between the **MDThread** object and design components changes and the status may need to be updated from *needs validation* to *validated*. Similarly, when design instances of application model are revised or the underlying object of product structure assembly design instance are revised, the association between **MDInstance** object and linked design instances may need to be validated. The following two services are used for validation:

The **queryNeedsValidationLink2** service searches for **Mdo0MDThread** association and **Mdo0MDInstance** association that have **mdo0NeedsValidation** attribute set as true on the association.

The **updateLinksToValidated2** service updates **mdo0IsValidated** attribute to true on **Mdo0HasInstanceAssociation** between the given **Mdo0MDInstance** and its design instances. It also updates **mdo0IsValidated** attribute to true on **Mdo0MDTAssociation** between the given **Mdo0MDThread** and its design artifacts.

Teamcenter services to search for MDThread objects

Use the following Teamcenter services to search for **MDThread** objects:

- **searchMDO3**
- **searchForArtifactsUsingInstances3**

The **searchMDO3** service searches for **MDThread** objects that are linked with the design components specified in the service. It also returns the other compatible design components linked with the searched **MDThread** object and its validation status. Optionally, the search can be filtered by domain, and only design components with the matching domain appear in the search results. If the domain filter is left empty, all design components that meet the search criteria appear in the results along with the relevant and irrelevant domain information.

The **searchForArtifactsUsingInstances3** service searches for **MDThread** objects using using design instance of application model or underlying object of the product structure assembly design instance as the search parameter. The design instance has an underlying design component that is linked to the **MDThread** objects. The service returns the **MDThread** objects that are linked to these design components. It also returns the other compatible design components linked with the searched **MDThread** object, their validation status, and the relevant and irrelevant domain information.

The **searchForArtifactsUsingInstances3** service can optionally use the domain information as search filter criteria. If multiple design instances are specified in the input, the service returns all **MDThread** objects containing the underlying design components.

Teamcenter services for notifications

Notifications are supported only for application model. Use the following Teamcenter services for queries for notifications:

- **qryNotificationsByImpactedDesign**
- **qryNotificationsByDesignOrProject**
- **qryNotificationsByOriginDesign**

Services support querying for *create notifications* (notifications generated when a component is used in a design) and *modify notifications* (notifications generated when a design instance is updated) by design instance or project context, or both. You can refine the queries by applying the following criteria:

- Search only *modify notifications*
- Search both *create notifications* and *modify notifications*
- Filter search by component type.
- Filter search by design
- Filter search by project
- Filter search by domain

Use the **updateMDONotifications** service for the designer's response to notifications. The designers can optionally respond to a *create notification* or a *modify notification*. In the response, the designers provide the details of their reaction and specify the original notification object to which they reacted.

Teamcenter services for domain information

Implement the **getDomainOfObjectOrType** service for domain-related queries. The service returns the domain information of the specified object or a list of objects. If object type is specified in the input, the domain information about the object type is returned.

Use the **updateDomainRelevancy** service to add and remove the list of relevant and irrelevant domains with the design component.