SIEMENS

Teamcenter Connector for Mendix 1.0

Developer's Guide

Contents

Getting started with Teamcenter Connector

Introduction

Teamcenter is a virtual gateway to your company's product information, connecting all who need to collaborate with product and process knowledge. Teamcenter enables you to digitally manage your product and manufacturing data in the context of the product life cycle.

Teamcenter Connector for Mendix enables Mendix developers to access product data from Teamcenter or create and modify product data in Teamcenter.

This documentation provides guidance on using Teamcenter Connector for Mendix. It assumes that you are familiar with Mendix concepts, processes, and terminology for application development.

Prerequisites

To use Teamcenter Connector for Mendix, you need the following:

- Mendix Business Modeler 7.23.5
- Appropriate Teamcenter licenses
- A running and accessible Teamcenter instance

Download Teamcenter Connector, the sample application, and dependencies

Teamcenter Connector for Mendix is available on the Mendix App Store. To add the connector to your project:

1. Open your Mendix Modeler project.
2. Click the AppStore icon  on the menu bar to access the Mendix App Store.
3. Search or navigate to the Teamcenter Mendix Connector and click the link.
4. On the Teamcenter Connector for Mendix page, click Download.
   In the **Import Module** dialog box, click Import.

The connector is imported in your project in the **App Store Modules** folder.

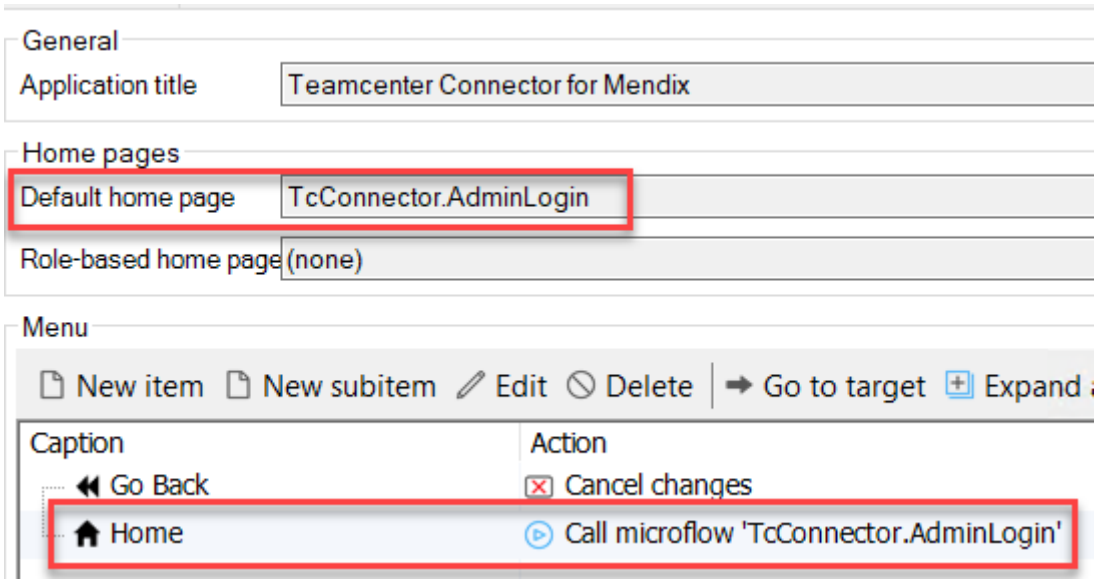
Similarly, download the following applications from the Mendix App Store:

- (Optional) Siemens PLM Software UI Resources. This application should be in the project that contains the Teamcenter Connector.
- (Optional) Teamcenter Sample Application. The Sample Application contains sample microflows. You *do not* require to download the Sample Application in the same project that also contains the Teamcenter Connector.

Configure to connect to Teamcenter

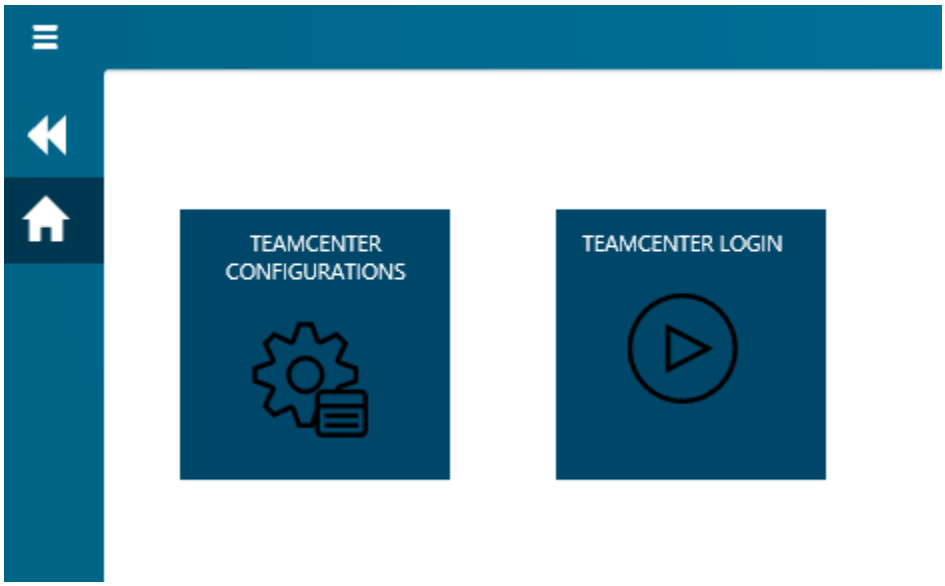
To help administrators configure the connection to Teamcenter, the Teamcenter Connector contains configuration microflows and pages. To configure the connection to Teamcenter, the Mendix administrator must do the following:

1. In the navigation, update the Default home page and the Home menu item to connect to the AdminLogin microflow.



2. Run your project.

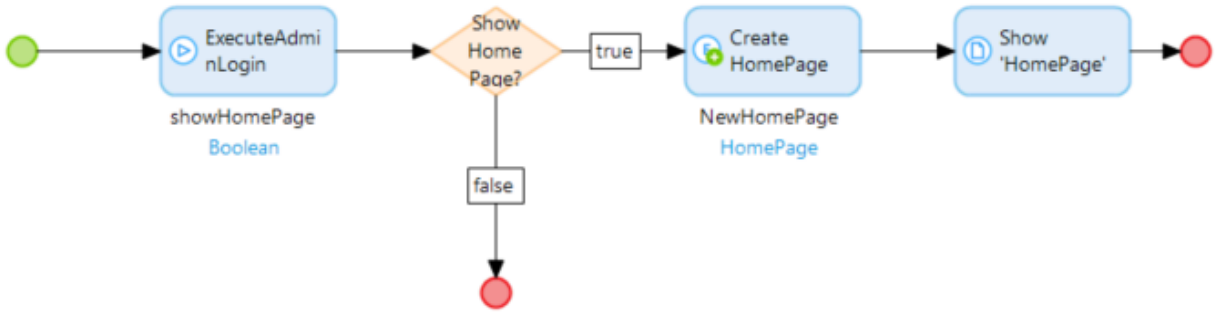
The browser displays the AdminHomePage.



3. Click the **TEAMCENTER CONFIGURATIONS** tile in the web browser.
4. Click **New** in the Teamcenter Environment Configuration page.
5. In the Add Teamcenter Configuration dialog box, specify the required fields and click **Save**.
6. If you have enabled SSO, ensure that you add the **Add RegisterRequestHandlers** microflow to your startup microflow.
7. Once you have updated the Teamcenter configuration information, log on to Teamcenter using the **TEAMCENTER LOGIN** icon.

If the previous steps, the configuration shows the Home page. If you want to customize the page that you finally see, create a new microflow that contains the **ExecuteAdminLogin** service. Update your **Default home page** and the **Home** menu item to connect to the microflow you created.

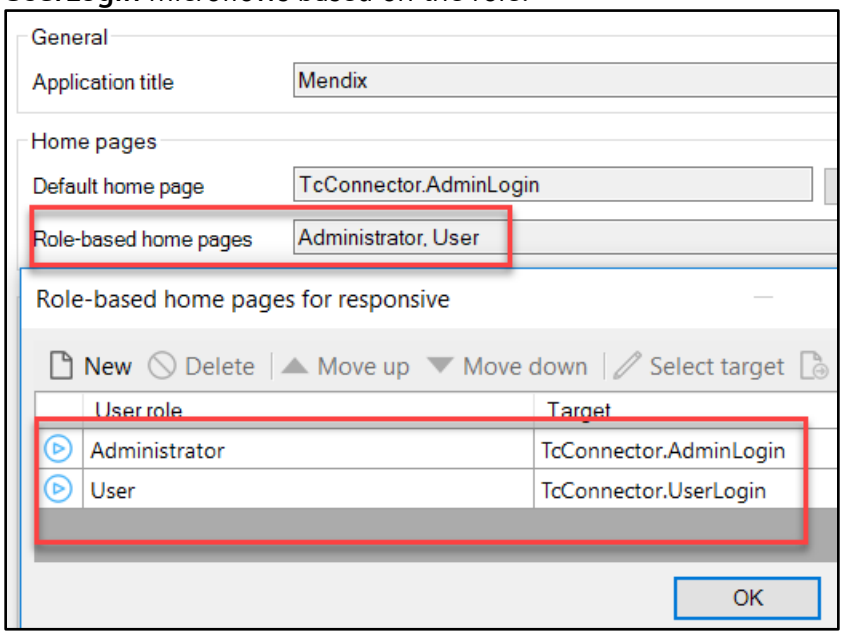
The Teamcenter Sample Application has a microflow called **MyAdminLogin** that uses the **ExecuteAdminLogin** service. Refer to that microflow for help.



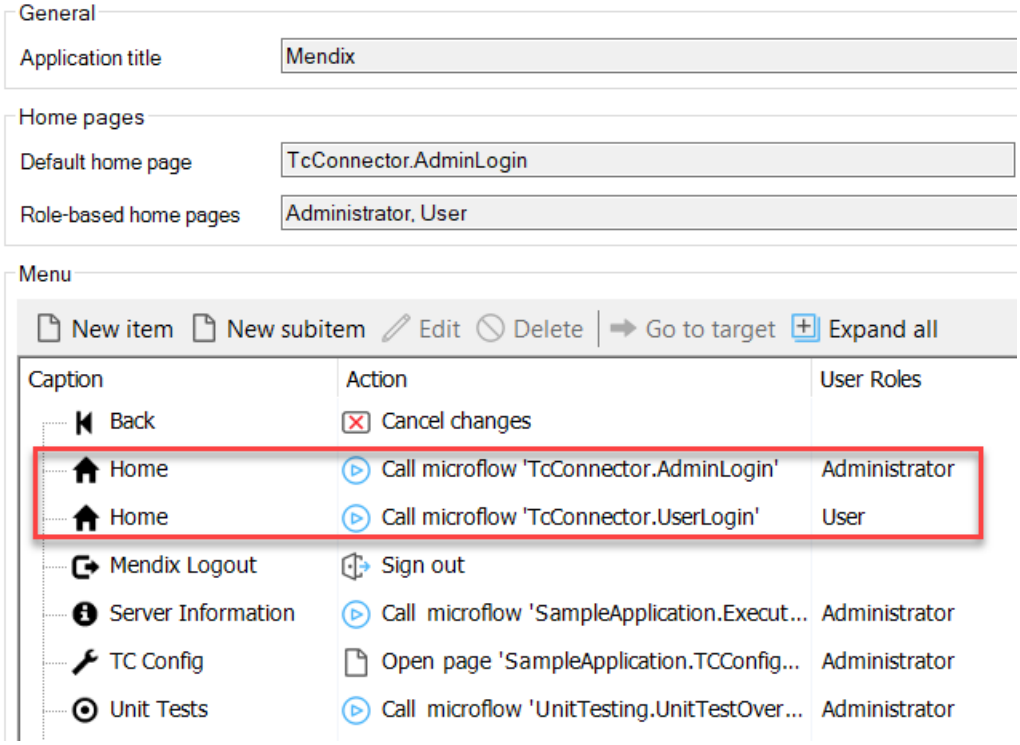
Configure to connect to Teamcenter in production mode

You can use the **AdminLogin** and the **UserLogin** microflows to configure the security of the Teamcenter Connector when the security level of your Mendix project is set to **Production.** Configure the security as follows:

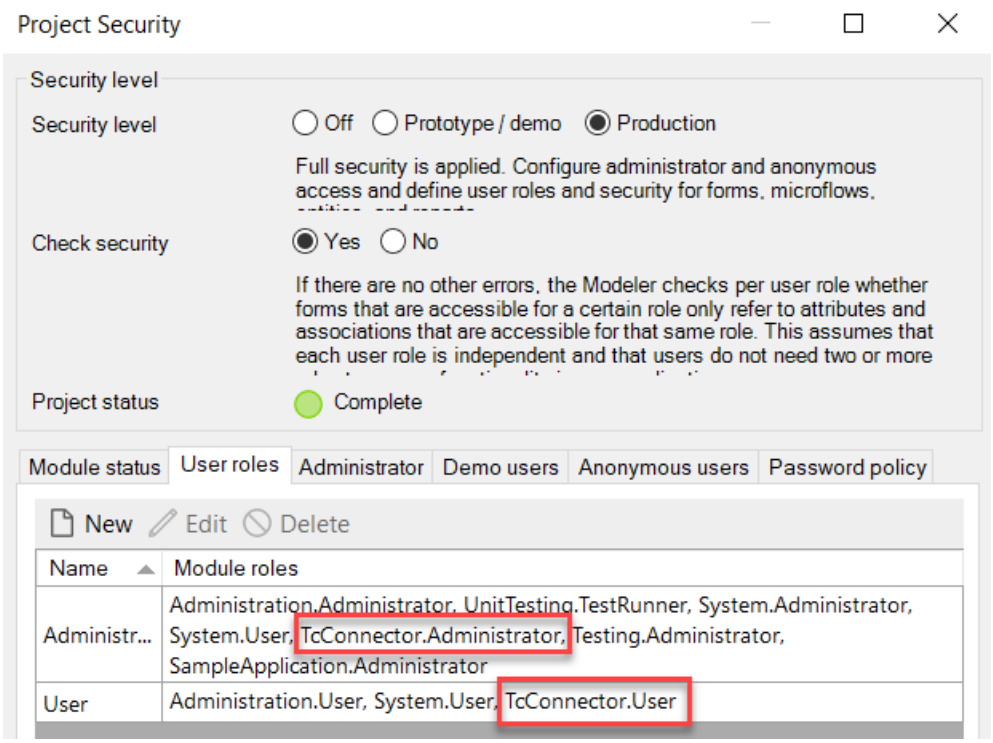
1. In navigation, update the **Role-based home pages** to connect to **AdminLogin** and **UserLogin** microflows based on the role.



2. In navigation, update the Home menu items to include the **AdminLogin** and **UserLogin** microflows.



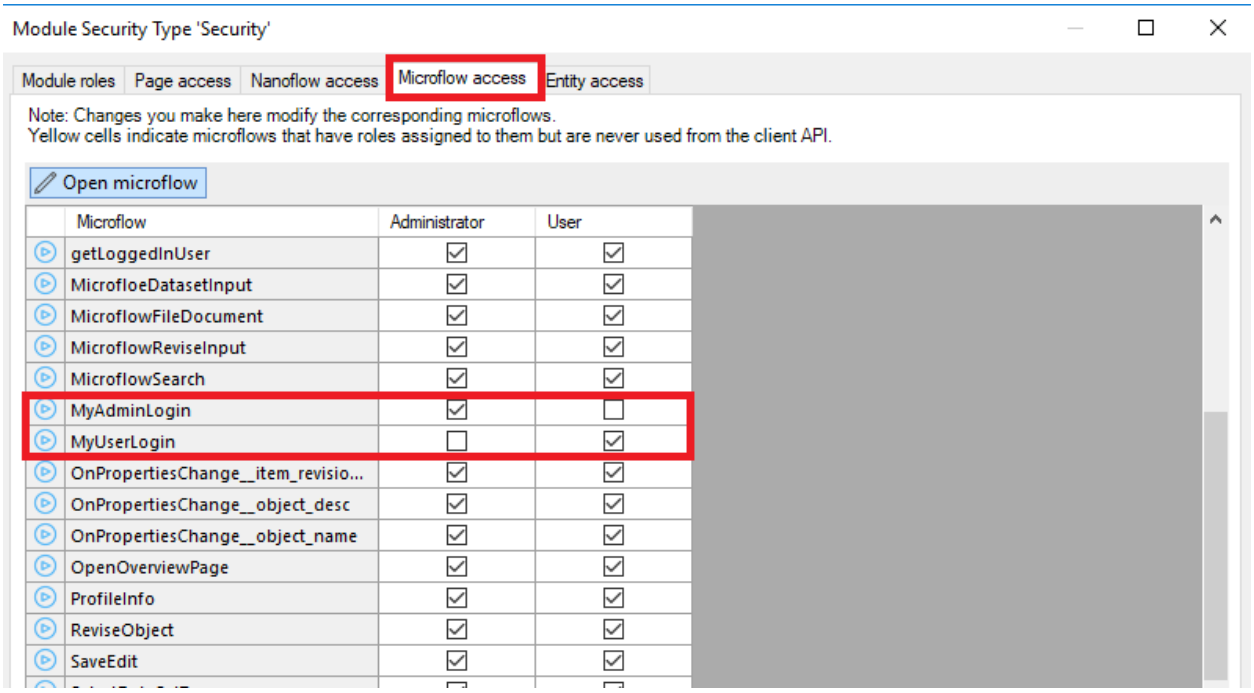
3. In the security page of your project, add the respective administrator and user roles in the **User roles** tab.



4. Run your project, and ensure that administrators and users see their configured login pages.

In the previous steps, the configuration shows the Home page. If you want to customize the page that you finally see, duplicate the **AdminLogin** and the **UserLogin** microflows.

In the security page of Teamcenter Connector, give appropriate access to the new microflows you created.



Update your **Default home page** and the **Home** menu item to connect to the microflow you created.

Understanding the Teamcenter Connector Domain Model

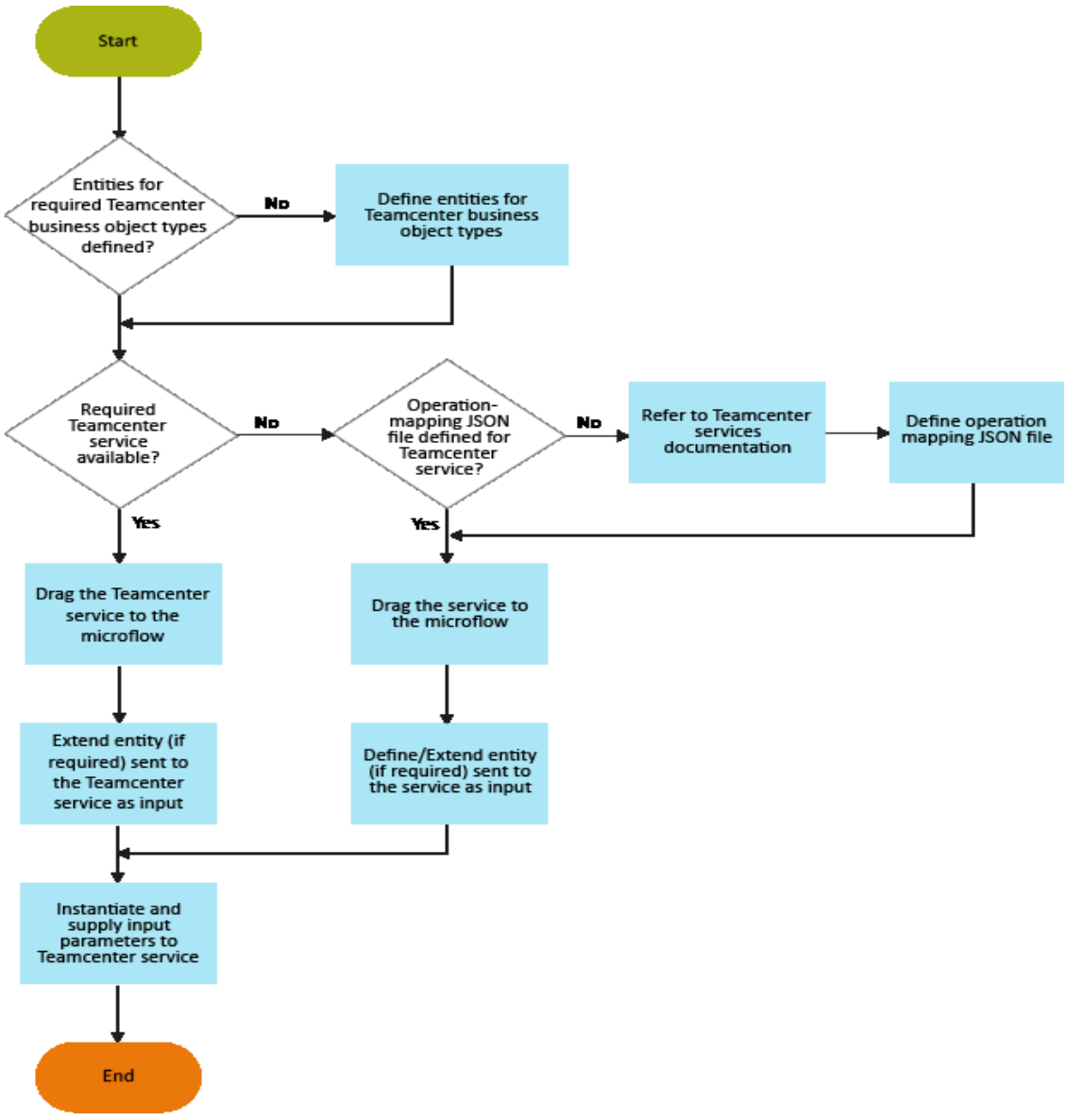
The Domain Model is a data model that describes the information in your application domain in an abstract way. It is central to the architecture of your application. The Domain Model consists of entities and their relations that are represented by associations.

The Teamcenter Connector for Mendix Domain Model represents Teamcenter business object types and their properties.

You can view the Teamcenter Connector Domain Model by navigating to TcConnector→Domain Model from the **Project Explorer**.

To export the Domain Model documentation, right-click the **Project Explorer** and choose **Export documentation**. The Domain Model documentation is exported as an HTML file.

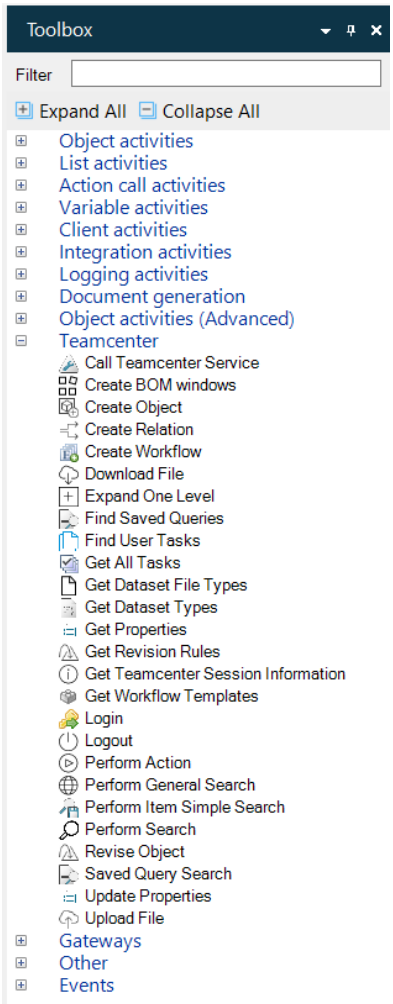
Using Teamcenter services through Teamcenter Connector: process workflow



Using Teamcenter services included with Teamcenter Connector

Introduction

Teamcenter services are provided through Java actions. You can see the available services in the **Teamcenter** section of the **Toolbox.**



Supported services

For information about the supported Teamcenter services, see Teamcenter Connector services.

Process to use Teamcenter services available in the Teamcenter Connector

1. Design a microflow as per your business logic.
2. Drag a service from the Teamcenter category of the toolbox into the microflow.
3. Specify input parameters for the Teamcenter service.
4. Specify how data is retrieved.
5. Test your application.

Example of using the available Teamcenter services

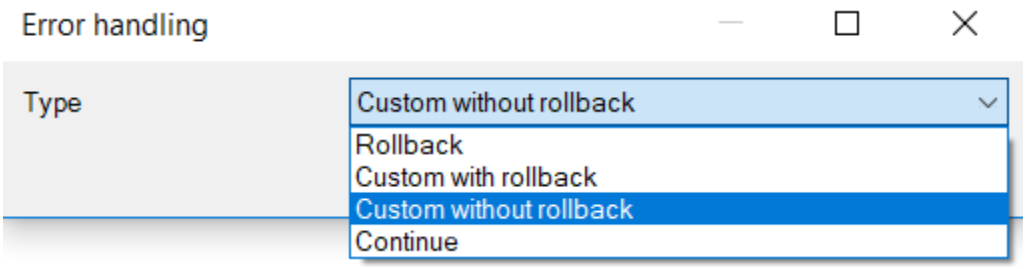
The Sample Application has microflows that use the available services such as **CreateItem** and **Search**. Download the Sample Application from the Mendix App Store and import it into your project.

How to handle errors

It is a good practice to set up error handling on all your Teamcenter services in a microflow.

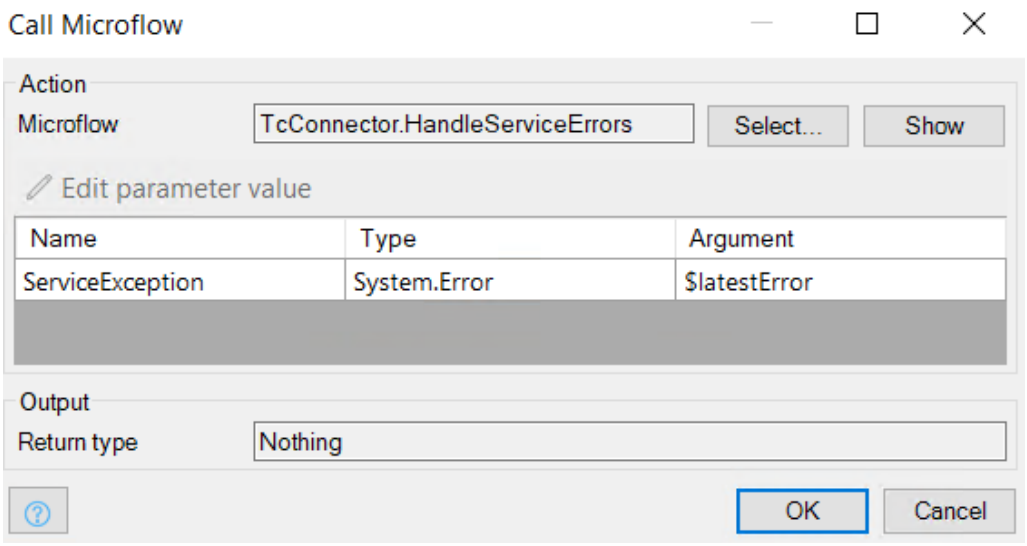
Use the **HandleServiceErrors** microflow to handle errors. To handle errors:

1. In your microflow, right-click your Teamcenter service and choose **Set error handling**.
2. In the **Error handling** dialog box, choose the error handling component.

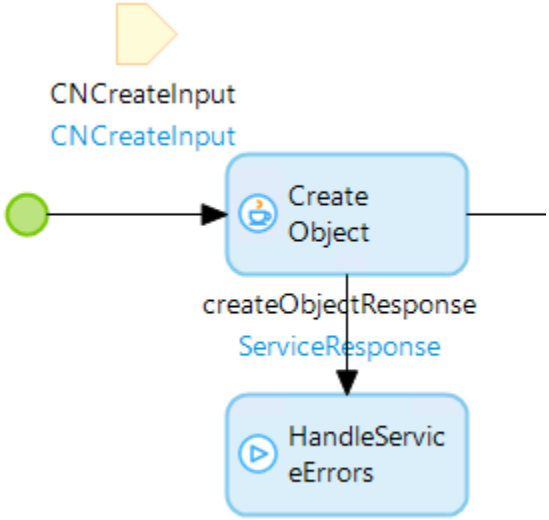


3. Drag the **Microflow call** service from the **Toolbox** to your microflow.
    a. Update the **Action** section of the microflow and select the **TcConnector.HandleServiceErrors** microflow.

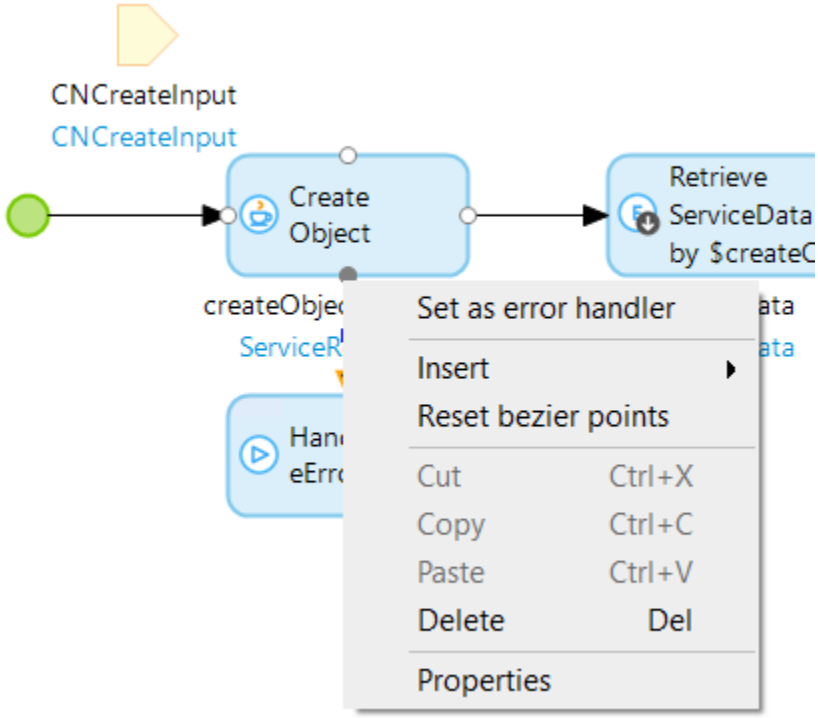
b. Update the **ServiceExeption** parameter and specify the type of error.



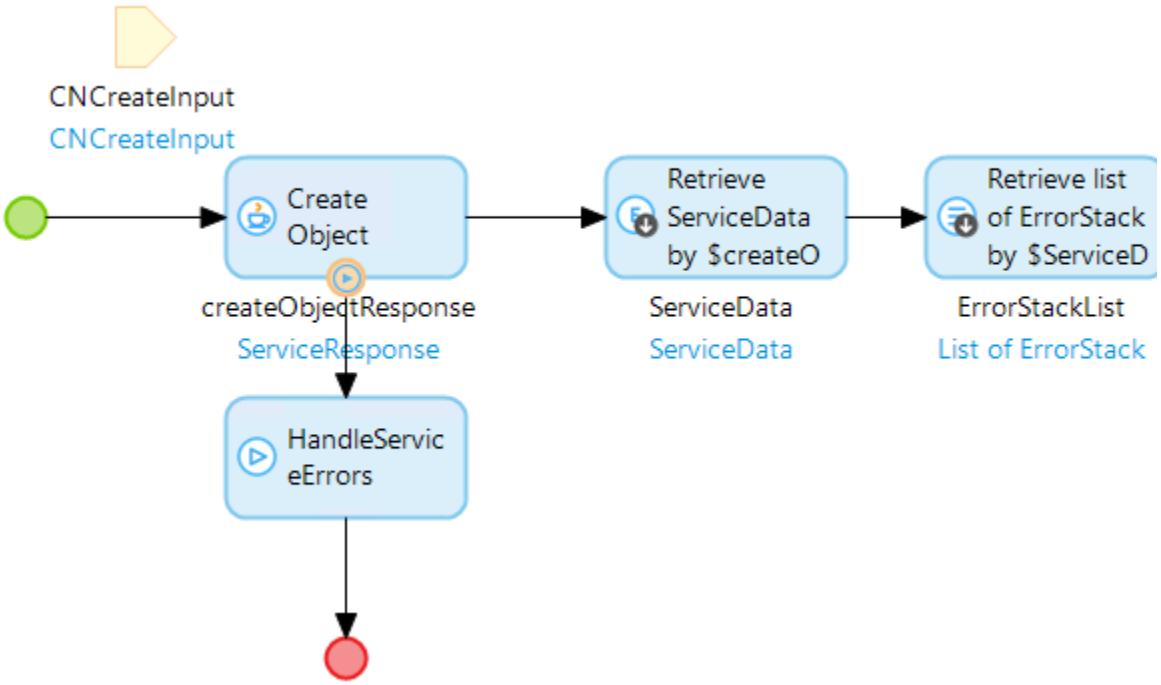
4. Connect the **Microflow call** service with the service from where the error will originate.



5. Right-click the anchor point of the Teamcenter service and choose **Set as error handler**.



6. Specify an end event for your **Handle Service Errors** service.



Extending the Domain Model

Guidelines to extend the Teamcenter Connector Domain Model

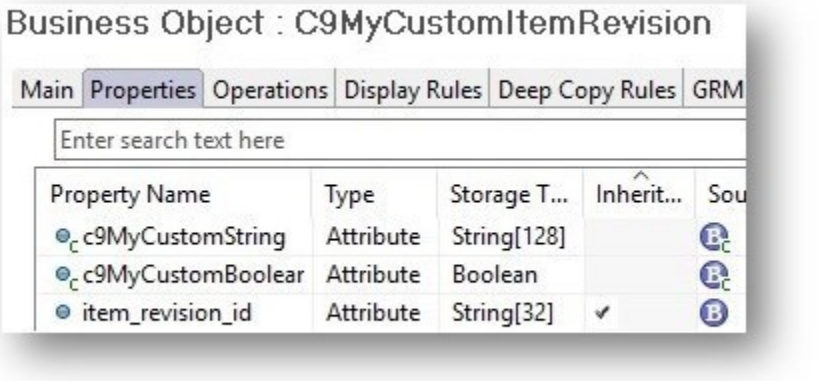
The Domain Model in Mendix consists of entities, associations, and annotations. It is analogous to the Teamcenter data model. When you want to extend the Domain Model, ensure that:

- You extend the Domain Model in a separate module and not the Teamcenter Connector Domain Model.
- The entities and associations must match the corresponding Teamcenter object type names and their properties. You can find Teamcenter object types and properties in the Teamcenter Developer Documentation or in the Business Modeler IDE application in Teamcenter.

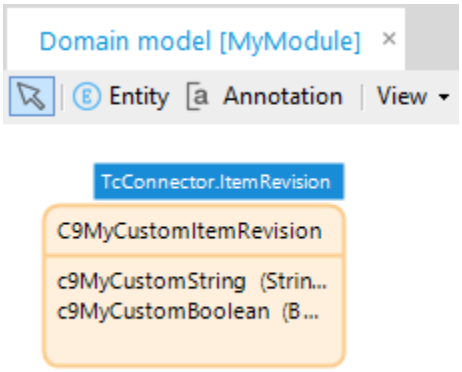
Example: Extend the domain model to access additional information

To access other object types or properties which are not already defined, you must add their definition to your app's Domain Model. This process is the same regardless of whether the object or property is OOTB or custom.

The Teamcenter BMIDE view of a custom business object is as follows. It is a child of the **ItemRevision** object and contains two new properties.



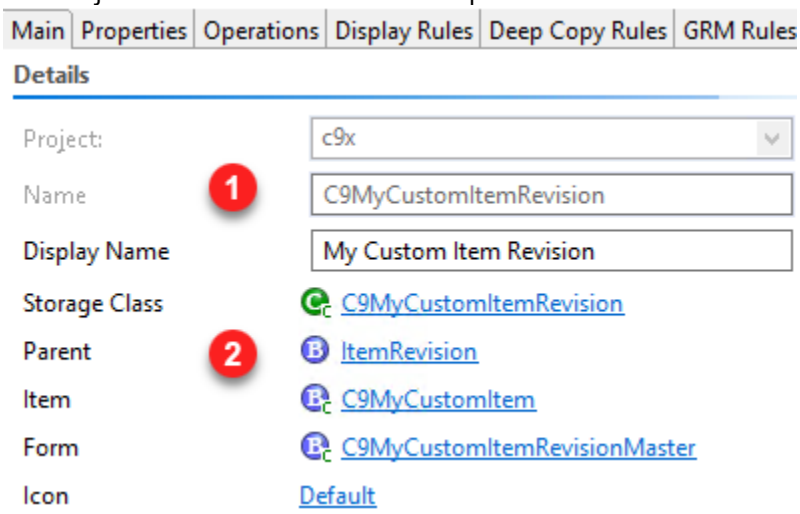
The Mendix Domain Model entity that you must create to retrieve these properties is as follows.



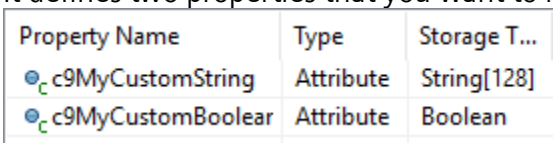
I. Examine the Teamcenter object

Examine the Teamcenter object and determine which properties you want to retrieve.

a. This object[1] has **ItemRevision** as its parent[2].

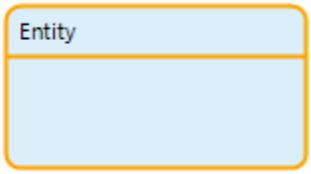


b. It defines two properties that you want to retrieve.

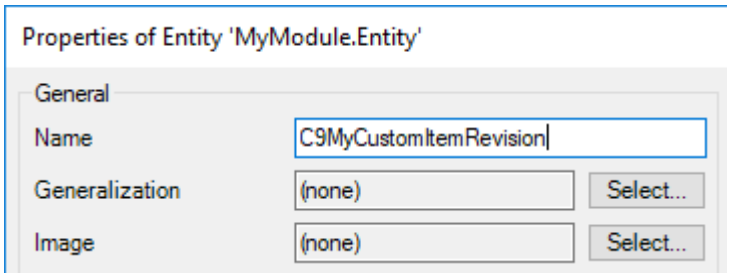


II. Create a new domain entity

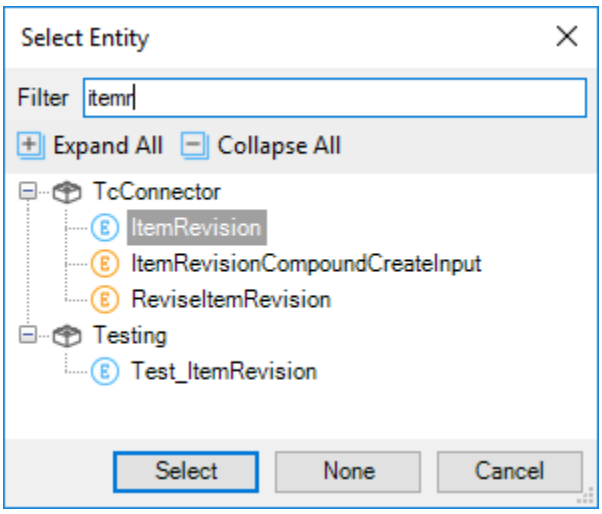
1. In your module's Domain Model, create a new entity.



2. Change the entity's name to match the Teamcenter object type name.

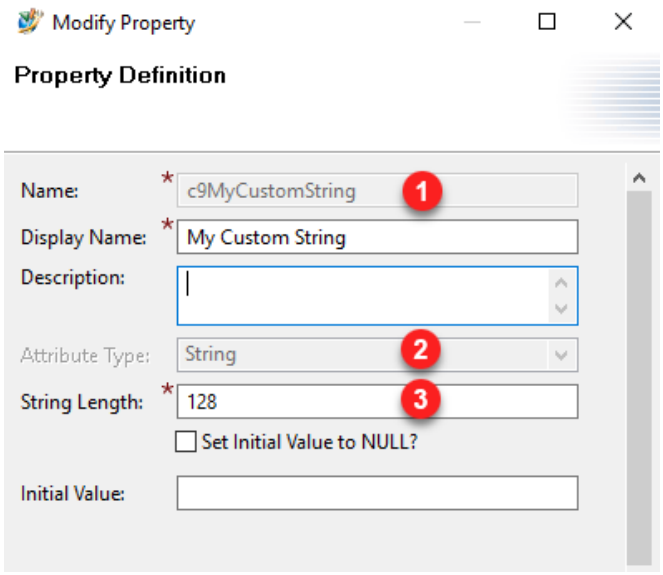


3. Define the new entity's *generalization* to match the Teamcenter parent object type.

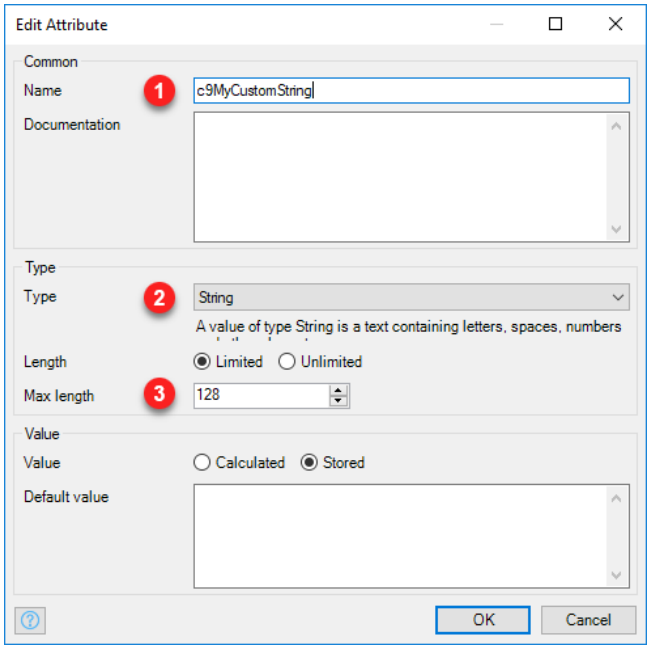


III. Define new attributes to match the Teamcenter properties

1. Match the Teamcenter property name[1], attribute type[2], and other parameters[3] to create the Mendix attribute.
    - Teamcenter

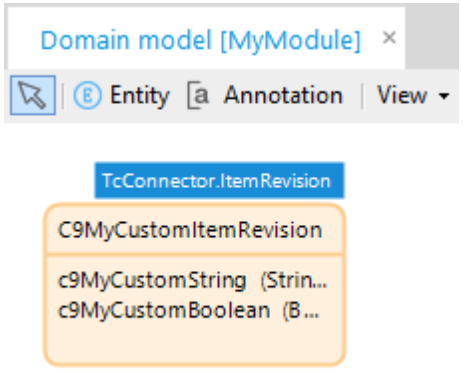


    - Mendix



IV. Process complete

You can now use your new Mendix domain entity and its attributes.



Repeat this process for each object whose properties you want to work with.

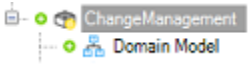
Process to extend the domain model while using available Teamcenter services

- Create the required entities in a separate module as a subtype of an existing entity (specialized entity).
- For example, if you want to create an entity for the *ChangeNoticeRevision* business object, you can create it as a subtype of the *ItemRevision* entity. In this case, the *ChangeNoticeRevision* entity is a specialized entity based on the *ItemRevision* entity.
- If the required entity is available but the required properties are not available, create a subtype of the entity in a new module and then add the required properties to the new entity.
  Caution
  Siemens recommends that you always create new entities and not make any changes to the entities that come with Teamcenter Connector.
- Design a microflow as per your business logic.
- Drag a service from the **Teamcenter** section of the toolbox into the microflow.
- Specify inputs for the Teamcenter service.
- Instantiate and specify input parameters to the Teamcenter service.
- Specify business object mapping to the Teamcenter service.
- Specify how data is retrieved.

- Test your application.

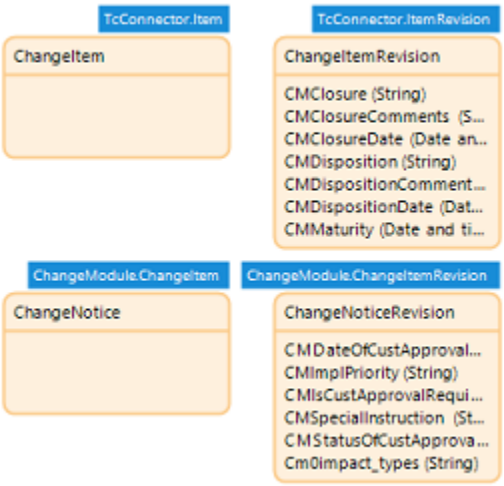
Example: Extend the domain model to call a service to create a Change Notice Revision

1. Create a module that represents the Change management domain. Skip this step if the module exists.



2. Define the Domain Model.

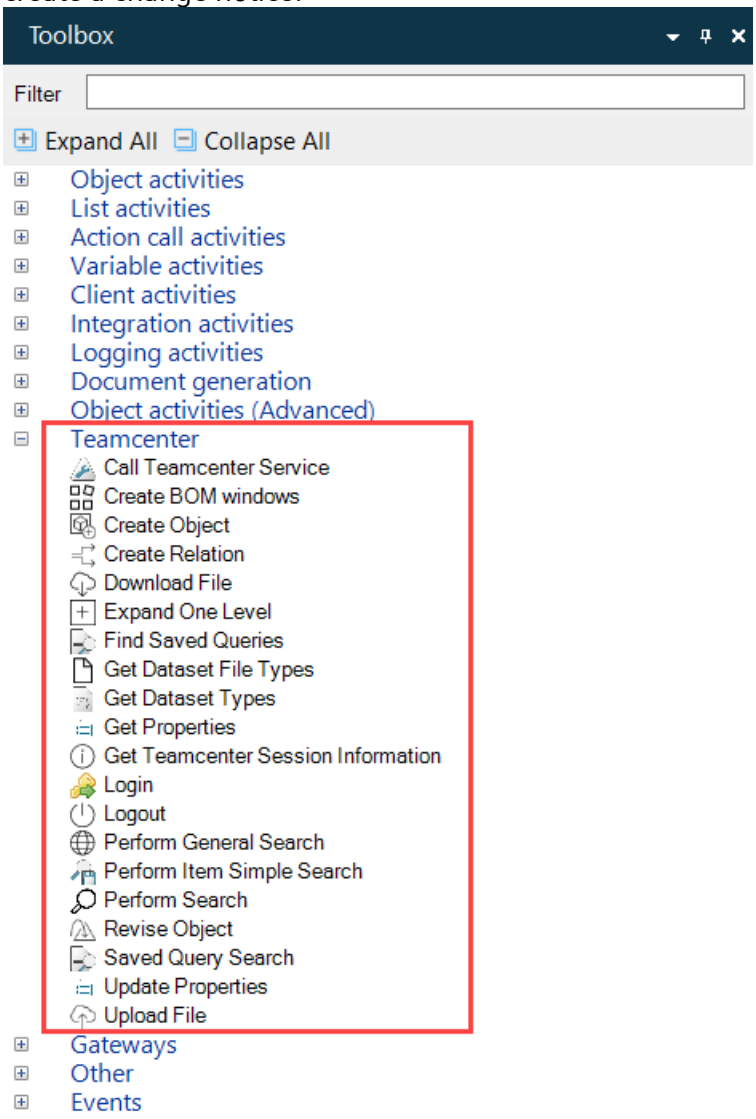
   Create ChangeNotice and ChangeNoticeRevision entities based on the TcConnector.Item and TcConnector.ItemRevision entities. Skip this step if the entities exist.
   Ensure that the names and properties of the defined entity match the corresponding Teamcenter business object type name and properties. The Reference properties on Teamcenter object types must be represented as associations in the Mendix Domain Model.



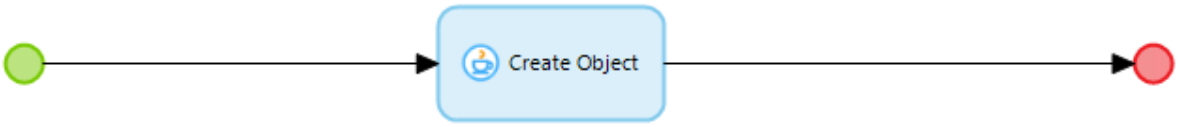
   Tip: You can find Teamcenter business objects and their properties in the Teamcenter Developer Documentation.

3. Plan your microflow and decide what services you want to use. The Teamcenter services are available in the microflow Toolbox under the **Teamcenter** category. For creating a Change Notice Revision, the microflow typically consists of the following activities:

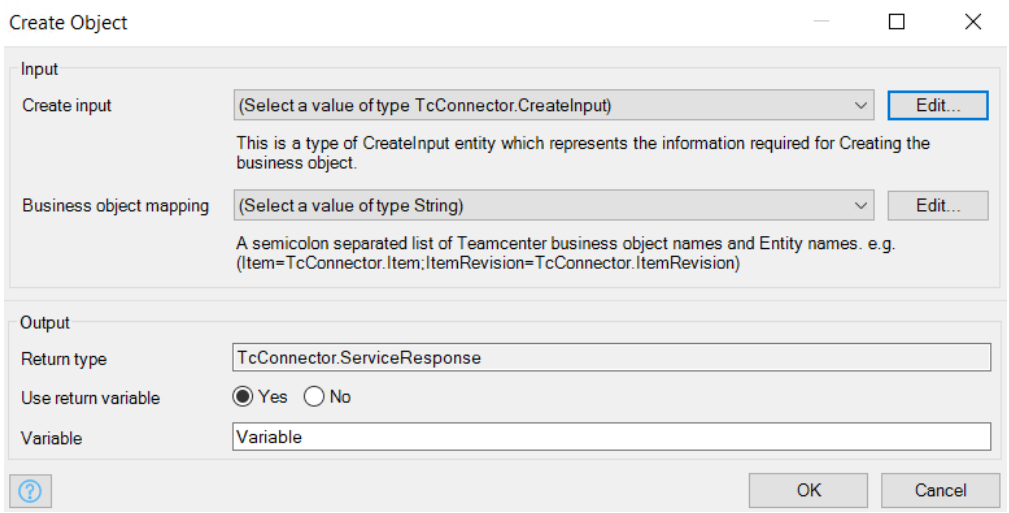
4. Select the appropriate Teamcenter service and drag it to your microflow.
   For example, you can use the **Create Object** service under the Teamcenter category to create a change notice.



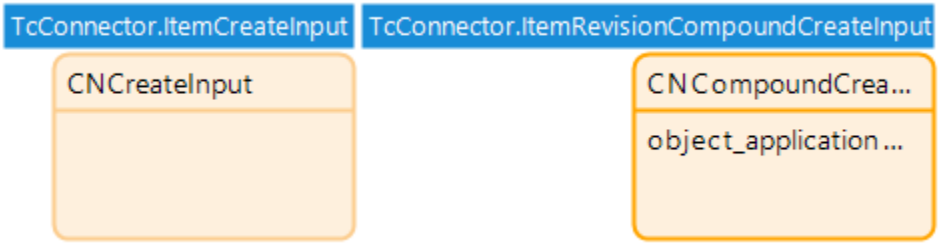
   Your microflow now appears as follows:



5. In your microflow, specify the input parameters that the Teamcenter service will use. The **CreateObject** service requires two input parameters:
    - An input variable or entity that contains the information required to create the change notice.
    - The mapping between Teamcenter business object names and Mendix entities. In the following example, the input entity, is the default **TcConnector.CreateInput** parameter.

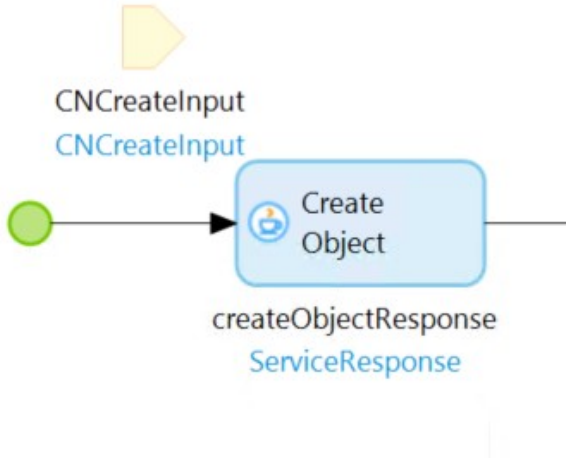


6. In the previous step, an existing input entity served as the input parameter. If you need specific properties to be sent as an input, you must create new input entities or *specialize* the default input entity and instantiate it.
For example, to send properties specific to Change Notice and Change Notice Revision, you must specialize the **CreateInput** entity as follows:
    - In your Change Management module, create the entities **CNCreateInput** and **CNCompoundCreateInput** deriving from the **ItemCreateInput** and **ItemRevisionCompoundCreateInput** entities that are available in the Teamcenter connector.



    - Provide input parameters to your microflow. When creating input parameters, ensure that you instantiate it if necessary.

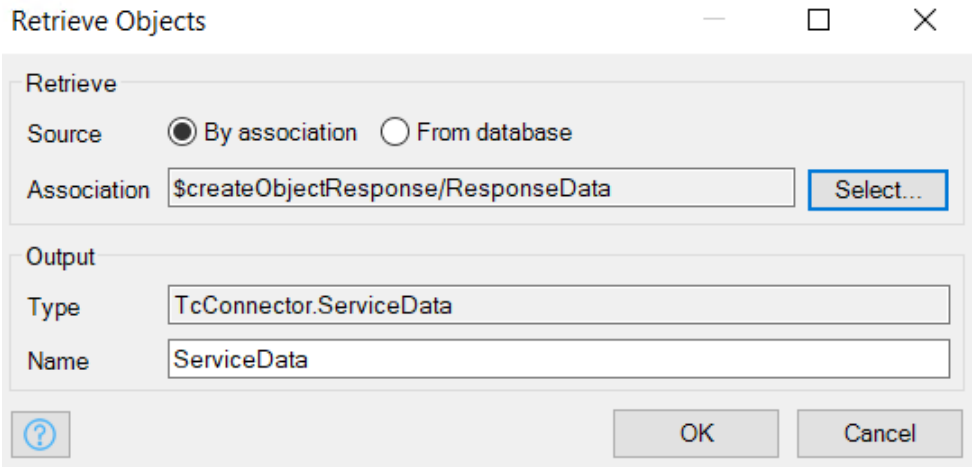
Your microflow appears as follows:



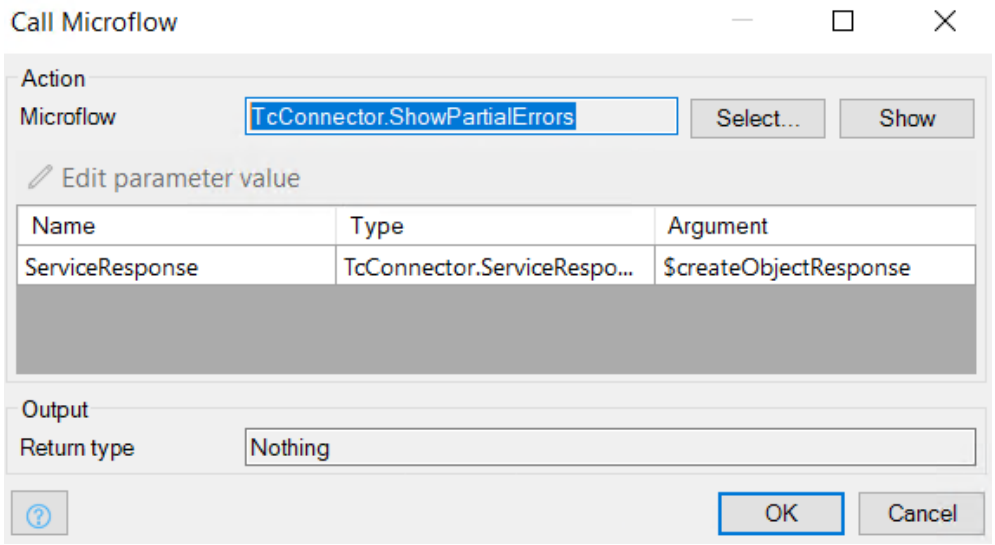
The input parameter contains the instantiated entities **CNCreateInput** and **CNCompoundCreateInput.**

7. You must also specify the **Business object mapping**, which is the mapping between an entity and the corresponding Teamcenter business object types that the microflow is dealing with.
   For example, the Teamcenter business object **ChangeNotice** is represented by the **ChangeManagement.ChangeNotice** entity and **ChangeNoticeRevision**, by the **ChangeManagement.ChangeNoticeRevision** entity. Thus, the mapping can be specified as:
   `'ChangeNotice=ChangeManagement.ChangeNotice;ChangeNoticeRevision=ChangeManagement.ChangeNoticeRevision'.`

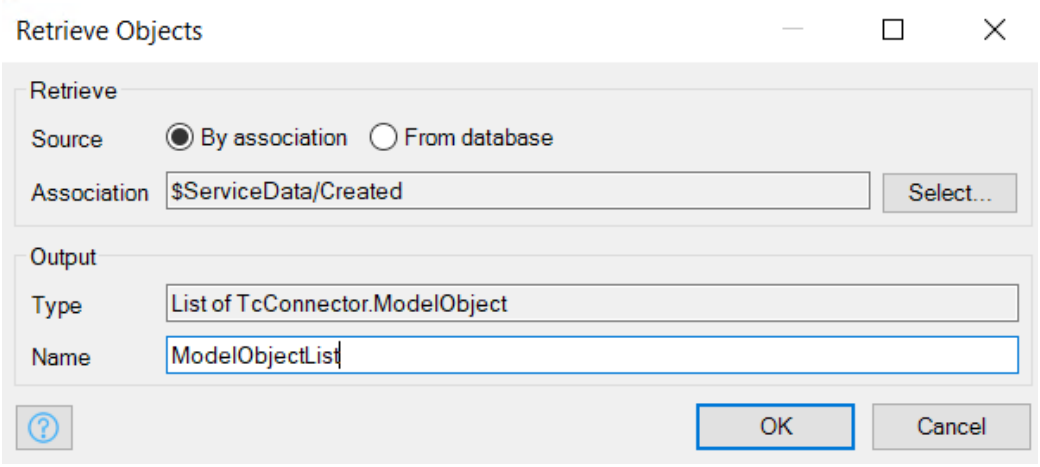
8. Retrieve the objects by dragging the **Retrieve** action to your microflow.
    - To retrieve response data use the $**createObjectResponse/ResponseData** association.



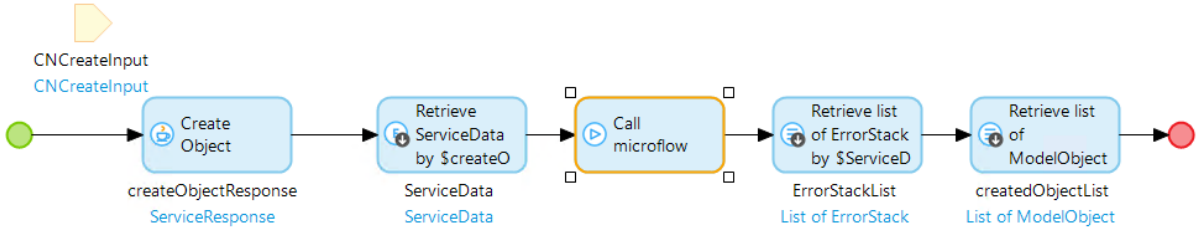
    - To retrieve partial error data, drag the **Microflow call** action into your microflow and call the use the **TcConnector.ShowPartialErrors** microflow.



- To retrieve created objects use the $ServiceData/Created association.



Your microflow appears as follows:



9. Test your microflow.

Using Teamcenter services not included in Teamcenter Connector

The Teamcenter connector provides two APIs for calling any service operation that is part of Teamcenter Services.

- `CallTeamcenterService` — Java action

  This Java action can be used directly in any microflow and maps the Mendix domain entities to or from the JSON data structures that are used by Teamcenter Services operations.

- `TcConnection.callTeamcenterService` — Java method

  This Java method can only be used from within the Java code and requires that the caller create and parse JSONObjects directly for the call. The TcConnector class also exposes an entity version of the `callTeamcenterService` method that matches the Java action.

Both APIs have general error handling and take care of the Teamcenter server session management. The Login service operation must be called before calling any other service operation using these APIs. The ShowLoginPage and ExecuteLogin Microflows perform this function.

Using the Java action

The `CallTeamcenterService` Java action is the entry point for calling Teamcenter service operations from a microflow. This Java action allows the developer to create a custom mapping between the Teamcenter service operation and the Mendix Domain Model entities. This Java action takes the following arguments:

- **Service Name**: The name of the Teamcenter service operation
- **Input Argument**: The entity containing the input data for the service operation.
- **Response Object**: The entity instance to which the service operation response will be written. This may be any entity type that extends from ServiceResponse.
- **Operation Mapping**: The mapping definition for this service operation.
- **Business Object Mapping**: The mapping of Teamcenter business object names to Mendix entity names.

Operation mapping

This maps a Teamcenter service operation request and a response each to the Mendix entities. The mapping is defined as a JSON document, either a as a file or provided as an input string to the CallTeamcenterService Java action. The JSON schema for an operation mapping document is as follows:

```
{
ServiceOperation:       The operation name (Core-2011-06-Session/login).
InputType:        The Entity type to map the input from.
ResponseType:           The Entity type to map the response to.
ObjectMapping:          The business object mappings.
OperationInput:   The template for the service operation input. Must
represent the complete service operation input.
OperationResponse: The template for the service operation response. May
contain only the specific elements of the response that are mapped.
}
```

Within the **OperationInput** and **OperationResponse** templates, values are either hard coded or use the **$Input** or **$Response** substitution keywords. These substitution keys have the following syntax:

```
$Input[/Association]</Attribute>[;Instruction]
$Response[/Association]</Attribute>
```

Where:

- **Association** — Optional association name on the given entity type. Multiple associations can be sequenced, each separated by a '/'.
- **Attribute** — Optional attribute name on the given entity type.
- **Instruction** — Optional instruction to be applied to the substitution. Multiple instructions can be used, each separated with a semicolon. The supported instructions are:
- `AttributeAsArray` — single valued JSONArray for each attribute value.
- `DateFormat=Format` — Use the custom date format for serializing date attributes.
- ignoreNull – Use to omit the key if the value of that key is null.

Examples of which attributes that will be mapped:

- `$Input`
  The full entity
- `$Input/TcConnector.itemRev`
  The entity referenced by the `TcConnector.itemRev` association
- `$Input/TcConnector.user/person`
  The *person* attribute on the referenced entity (`TcConnector.user` association)
- `$Input;DateFormat=MM/dd/yyyy`
  The full entity, with any date attributes serialized in the format `mm/dd/yyyy`

Note: This is a JSON document, so the forward slash '/' character in any quoted string must be escaped '\/'.

A sample mapping JSON document:

```
{
    "ServiceOperation": "Cad-2007-01-StructureManagement\/createBOMWindows",
    "InputType":        "TcConnector.CreateBomWindowInput",
    "ResponseType":     "TcConnector.CreateBomWindowResponse",
    "ObjectMapping":    "BOMLine=TcConnector.BOMLine",
    "OperationInput":
    {   "info":
        [{  "clientId": "CreateBOMWindows",
            "item":     "",
            "itemRev":  "$Input\/TcConnector.itemRev",
            "bomView":  "",
            "objectForConfigure": "",
            "activeAssemblyArrangement":"",
            "revRuleConfigInfo":
            {   "clientId": "",
                "revRule":  "$Input\/TcConnector.revRule",
                "props":
                {   "unitNo":  -1,
                    "date":    "",
                    "today":   true,
                    "endItem": "",
                    "endItemRevision": "",
                    "overrideFolders":
                    [{  "ruleEntry": "",
                        "folder": ""
                    }]
                }
            }
        }]
    },
    "OperationResponse":
    {   "output":
        [{"bomLine": "$Response\/TcConnector.createBomWindowResponseBOMLine"
        }]
    }
}
```

ExpandGRMRelationsForPrimary Example:

The SOA request and response structure for 'Core-2007-09-DataManagement/expandGRMRelationsForPrimary' are as follows:

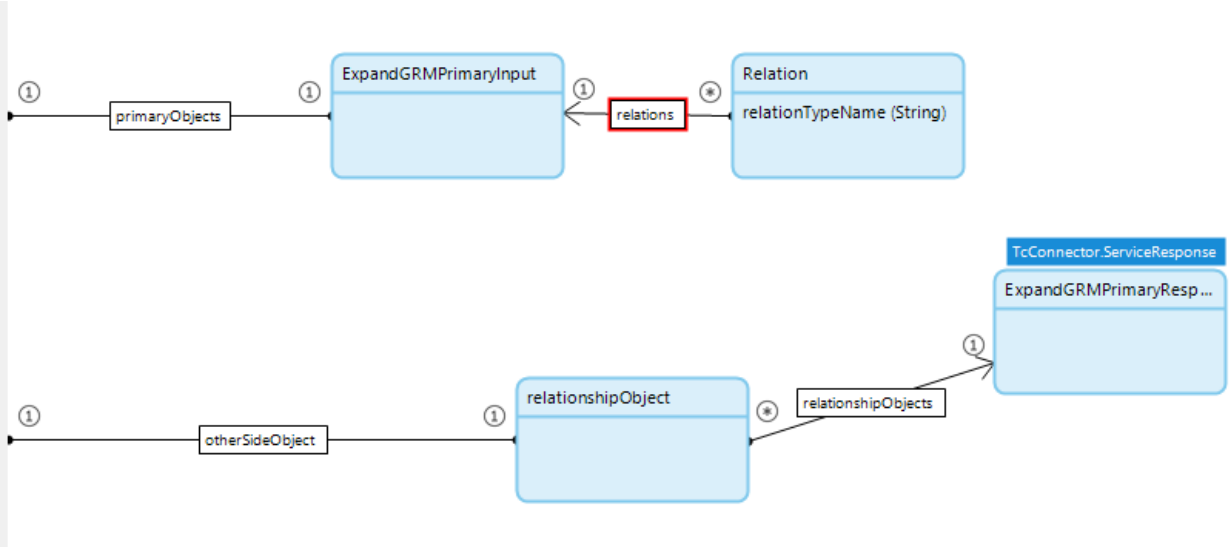
Library: Core
Service: DataManagement
Year: 2007-09
Url: Core-2007-09-DataManagement/expandGRMRelationsForPrimary
Soa Dependency Inclusion: "Teamcenter.Soa.Core._2007_09.DataManagement.expandGRMRelationsForPrimary"
Request:

```
{
  primaryObjects: "JModelObject[]",
  pref:
  {
    expItemRev: "bool",
    returnRelations: "bool",
    info:
    [{
      relationTypeName: "String",
      otherSideObjectTypes: "String[]"
    }]
  }
}
```

Response:

```
{
  output:
  [{
    inputObject: "JModelObject",
    relationshipData:
    [{
      relationshipObjects:
      [{
        otherSideObject: "JModelObject",
        relation: "JModelObject"
      }],
      relationName: "String"
    }]
  }],
  serviceData: "JServiceData",
}
```

To create the operation mapping, refer to the request and response entities and substitute the corresponding entity attributes and associations for the key or the structure. In this example we have created the entities structure as follows:



Request Entities:

1. **ExpandGRMPrimaryInput** – Input entity.
2. **Relation** – Represents the info structure in the request.
3. **relations** – Many-to-one association between ExpandGRMPrimaryInput and Relation entities.
4. **primaryObjects** – One-to-one association with list of model objects represents the **primaryObjects** key in request

Response Entities:

1. **ExpandGRMPrimaryResponse** – Response Entity derived from ServiceResponse
2. **relationshipObject** – Represents relationshipObjects structure in the response.
3. **relationshipObjects** – many to One Association between ExpandGRMPrimaryResponse and relationshipObject entities.
4. **otherSideObject** – One to One association with ModelObject represents the otherSideObject key in the response.

Following is the operation mapping file for expandGRMRelationsForPrimary SOA call:

```
{
  "ServiceOperation": "Core-2007-09-DataManagement/expandGRMRelationsForPrimary",
  "InputType": "TcConnectorSample.ExpandGRMPrimaryInput",
  "ResponseType": "TcConnectorSample.ExpandGRMPrimaryResponse",
  "ObjectMapping": "",
  "OperationInput": {
    "primaryObjects": [ "$Input\/TcConnectorSample.primaryObjects/TcConnector.ListOfModelObjects" ],
    "pref": {
      "expItemRev": false,
      "returnRelations": true,
      "info": [ "$Input\/TcConnectorSample.relations" ]
    }
  },
    "OperationResponse": {
        "output": [{
                "relationshipData": [{
                        "relationshipObjects": "$Response\/TcConnectorSample.relationshipObjects"
                    }
                ]
            }
        ]
    }
}
```

CreateWorkflow Example:

The SOA request and response structure for 'Workflow-2014-10-Workflow/createWorkflow' are as follows:

Library: Workflow

Service: Workflow

Year: 2014-10

Url: Workflow-2014-10-Workflow/createWorkflow

Soa Dependency Inclusion: "Teamcenter.Soa.Workflow._2014_10.Workflow.createWorkflow"

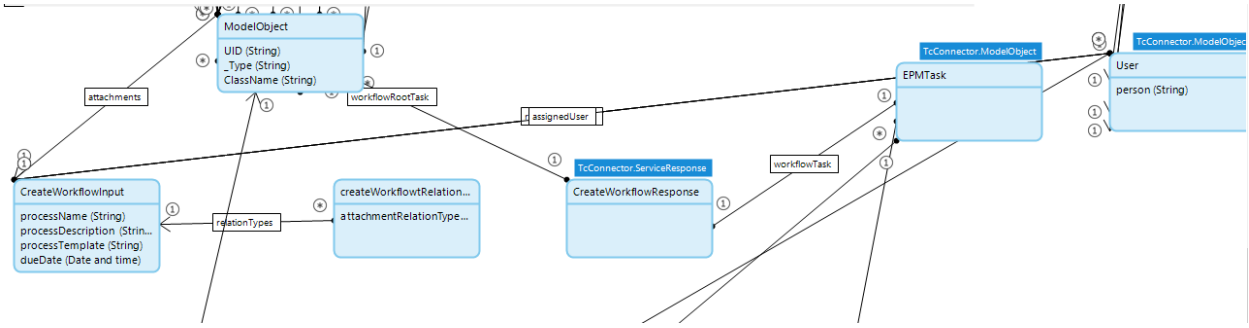
Request:

```
{
  input:
  {
    processName: "String",
    processDescription: "String",
    processTemplate: "String",
    workflowOwner: "JModelObject",
    responsibleParty: "JModelObject",
    assignedUserList: "JModelObject[]",
    dueDate: "Date",
    attachments: "JModelObject[]",
    attachmentRelationTypes: "String[]",
  }
}
```

Response:

```
{
  workflowTask: "JModelObject",
  attributes:
  {
    SampleStringKey: "String"
  },
  serviceData: "JServiceData"
}
```

To create the operation mapping, refer the request and response entities and substitute the corresponding entity attributes and associations for the key or the structure. In this example we have created the entities structure as follows:



Request Entities:

1. **CreateWorkflowInput** – Input entity.
2. **createWorkflowtRelationTypes** – Entity representing attachmentRelationTypes key in the request.

3. **relationTypes** – One to Many Association between CreateWorkflowInput and createWorkflowtRelationTypes entities.
4. **attachments** - One to Many Association between CreateWorkflowInput and ModelObject entities representing attachments key in the request.
5. **assignedUser** - One to Many Association between CreateWorkflowInput and User entities representing assignedUserList key in the request.
6. **responsibleParty** - One to One Association between CreateWorkflowInput and User entities representing responsibleParty key in the request.
7. **workflowOwner** - One to One Association between CreateWorkflowInput and User entities representing workflowOwner key in the request.
8. **processName** – String attribute on CreateWorkflowInput entity representing processName key in request.
9. **processDescription** – String attribute on CreateWorkflowInput entity representing processDescription key in request.
10. **processTemplate** – String attribute on CreateWorkflowInput entity representing processTemplate key in request.
11. **dueDate** – Date and time attribute on CreateWorkflowInput entity representing dueDate key in request.
12. **attachmentRelationTypes** – String attribute on createWorkflowtRelationTypes entity representing the attachmentRelationTypes key in request, which is array of strings, so we add the instruction AttributeAsArray after the path to attribute. This will take care to give the input to this key in form of array of strings.

Response Entities:

1. **CreateWorkflowResponse** – Response Entity derived from ServiceResponse.
2. **workflowTask** – One to Many association between CreateWorkflowResponse and EPMTask entities representing workflowTask structure of the response.

Following is the operation mapping file for createWorkflow SOA call:

```
{
    "ServiceOperation": "Workflow-2014-10-Workflow\/createWorkflow",
    "InputType":        "TcConnector.CreateWorkflowInput",
    "ResponseType":     "TcConnector.CreateWorkflowResponse",
    "ObjectMapping":    "",
    "OperationInput":
    {
        "input":
                {
                    "processName": "$Input\/processName",
                    "processDescription": "$Input\/processDescription",
                    "processTemplate":"$Input\/processTemplate",
                    "workflowOwner": "$Input\/TcConnector.workflowOwner",
                    "responsibleParty": "$Input\/TcConnector.responsibleParty",
                    "assignedUserList": ["$Input\/TcConnector.assignedUser"],
                    "dueDate": "$Input\/dueDate",
                    "attachments": ["$Input\/TcConnector.attachments"],
                    "attachmentRelationTypes": "$Input\/TcConnector.relationTypes\/attachmentRelationTypes;AttributeAsArray"
                }
    },
    "OperationResponse":
    {
        "workflowTask": "$Response\/TcConnector.workflowTask"
    }
}
```

Entity Mapping

Mendix Domain Model entities are mapped to Teamcenter service operation data structures based on naming conventions. The Teamcenter structure element names (keys in JSON document) map one-to-one to the entity member (attributes or associations) names, with the following caveats:

- The entity member name is prefixed with an underscore '_', for example '*_type*'. In this case the '_' is ignored, thus matching the Teamcenter name of '*type*'.
- The entity member name is suffixed with '__XXX', for example, '*phone__Home*'. In this case the '*__Home*' is ignored, thus matching the Teamcenter name of '*phone*'.

When traversing entities across associations, the entity names are not used. Only the associations between entities must match or be mappable.

ModelObject Mapping

The ModelObject entity is the Mendix Domain Model representation of a Teamcenter business object. Any entity that extends from ModelObject is considered to be a ModelObject. The mapping of ModelObject entities follows the general mapping of entity mapping (see previous section), with the following additions:

- Member names defined directly on the ModelObject entity (UID, _Type...) are ignored.
- All other entity member names are mapped one-to-one with the business object type property name.

- Entity Attribute types must match the type of the business object property type. Attributes of type *String* are the display value or localized value of the business object property, while attributes of other types (Boolean, Decimal, and Integer, Localized Date and time), are the database value of the business object property. To map a database value of a business object String property, the attribute name must be suffixed with '__DB' that is, 'description__DB'.

Business object mapping

This refers to the mapping of Teamcenter business object type names to Mendix entity names. This mapping is applied to the business objects that are returned from the service operation. The syntax for this mapping is a semicolon-separated list of Teamcenter or Mendix names.

```
BOMLine=TcConnector.BOMLine;ItemRevision=TcConnector.ItemRevision
```

For any business object returned from the service operation that is not in this mapping (that is, EngChange_Revision), the nearest mapped parent (ItemRevision) is instantiated.

Error Handling

The CallTeamcenterService Java action handles all non-service errors that occur during the processing of the service request. This includes networking errors (HTTP errors connecting to the Teamcenter server), session time-out, and parsing errors. These errors are displayed as an exception, and the calling microflow must set an error handler on the Java action and then create a flow from the Java action that is set as the error handler. This flow must display the $latestError/Message and then exit the microflow.

Using the Java Method

The callTeamcenterService Java method (tcconnector.foundation.TcConnection.callTeamcenterService) is the entry point for calling Teamcenter service operations from the Java code. This Java method differs from the CallTeamcenterService Java action in that the input and output data are JSONObject versus Mendix entities. The developer is responsible to creating the JSONObject input that conforms to the Teamcenter service operation definition and parsing the returned JSONObject. This Java method takes 3 arguments:

- Service Name: The name of the Teamcenter service operation
- Input Argument: The JSONObject containing the input data for the service operation.
- Policy: The Object Property Policy defining which values should be returned.

A JSONObject is returned from this method with the contents of the service operation response. All business object references (UIDs) throughout the response structure are replaced with the full

ModelObject instance from the ServiceData. JSONObject instances that represent a ModelObject or ServiceData can be cast to JModelObject or JSerivceData:

```
JModelObject bomLine = (JModelObject)output.getJSONObject("bomline");
JServiceData sd = (JServiceData)response.getJSONObject("serviceData");
```

Both the JModelObject and JServiceData class have methods to conveniently access data on this structure and convert to Mendix entities.

The JModelObject Class

The JModelObject class (tcconector.foundation.JModelObject) extends the JSONObject class to represent a single Teamcenter business object (ModelObject). This class has convenience methods to access property values and other elements defined on the ModelObject type. Conversation of a JModelObject to an entity uses ModelObject Mapping.

The JServiceData Class

The JServiceData class (tcconector.foundation.JServiceData) extends the JSONObject class to represent the ServiceData structure common to most Teamcenter service operations. This class has convenience methods to access data in the ServiceData. Conversion of JServiceData to an entity uses ModelObject Mapping.

The JPolicy Class

The JPolicy class (tcconector.foundation.JPolicy) extends the JSONObject class to represent an Object Property Policy. The Object Property Policy defines what properties should be returned from the service operation call for the given set of business object types. The JPolicy is constructed from Business Object Mapping, which defines a policy that includes all the entity member names.

Teamcenter Connector services

Call Teamcenter Service

Calls any Teamcenter Service from a microflow. To call a Teamcenter Service, you must create a custom mapping between the Teamcenter Service Operation and the Mendix Domain Model Entities.

Parameters

Name	Type	Description
ServiceName	String	Fully qualified name of the service operation, for example, Core-2011-06-Session/login.
InputArgument	Object	The input Entity for the service operation.
ResponseObject	Object	The returned object. If null an instance of ServiceResponse will be created for the return.
OperationMapping	String	Path for the operation mapping definition. This path is, relative to resources/OperationMappings. The InputArgument and Response are mapped to the service operation input and output based on the definition in the mapping file.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, BOMLine=TcConnector.BOMLine;ItemRevision=TcConnector.ItemRevision.

Returns

Service response of type TcConnector.ServiceResponse. The returned list of model objects can be retrieved using appropriate association. Partial errors can be retrieved using TcConnector.ResponseData and TcConnector.PartialErrors.

Create BOM Windows

Creates a BOM Window and sets the input item revision as the top line.

SOA URL

Cad-2007-01-StructureManagement/createBOMWindows

Parameters

Name	Type	Description
InputData	Object	Input for creating the BOM Window.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

An entity of type TcConnector.CreateRelationResponse that contains the created relations.

Partial errors can be retrieved using association TcConnector.PartialErrors

CreateObject

Creates any Teamcenter business object. This action also creates secondary (compound) objects if the CompoundCreateInput for the secondary object is represented in the recursive CompoundCreateInput object. For example, Item is primary object that also creates Item Master and ItemRevision. ItemRevision in turn creates ItemRevision Master. The input for all these levels is passed in through the recursive CompoundCreateInput object.

SOA URL

Core-2015-07-DataManagement/createRelateAndSubmitObjects2

Parameters

Name	**Type**	Description
createInput	Object	Type of CreateInput entity that represents the information required for creating a business object.
businessObjectMapping	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

An object of type TcConnector.ServiceResponse.

Created objects can be retrieved using associations TcConnector.ServiceResponse, TcConnector.ResponseData, and TcConnector.Created. Partial errors can be retrieved using association TcConnector.ResponseData and TcConnector.PartialErrors.

Create Relation

Creates the specified relation between the input primary and secondary objects. If the primary object has a relation property by the specified relation name, then the secondary object is associated with the primary object through the relation property.
Ensure that if the relation has mandatory properties, ensure that they are added to the entities. The Teamcenter Connector does not enforce the mandatory properties.

SOA URL

Core-2006-03-DataManagement/createRelations

Parameters

Name	Type	Description
InputData	Object	Input argument for invoking createRelations service.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and Entity names, for example, ImanRelation=TcConnector.ImanRelation.

Returns

An entity of type TcConnector.CreateRelationResponse that contains the created relations.

Partial errors can be retrieved using association TcConnector.PartialErrors

Create Workflow

Creates a Teamcenter Workflow object.

SOA URL

Workflow-2014-10-Workflow/createWorkflow

Parameters

Name	Type	Description
InputData	Object	Input argument for creating Teamcenter workflow object.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and Entity names, for example, ImanRelation=TcConnector.ImanRelation.

Returns

An entity of type TcConnector.CreateWorkflowResponse that contains the created workflow objects.

Partial errors can be retrieved using association TcConnector.PartialErrors

Download File

Downloads all the files associated with the Dataset that is passed as an input parameter.

SOA URL

Core-2006-03-DataManagement/getProperties

Core-2006-03-FileManagement/getFileReadTickets

Parameters

Name	Type	Description
DatasetParameter	Object	Dataset object that has the UID of the Teamcenter dataset whose files are to be downloaded.

Returns

A Boolean type that returns True or False to represent success and failure of the download.

The Dataset.Documents association of the input object will be updated to point to any downloaded files.

Expand One Level

Expands the first level children of given parent BOMLine.

SOA URL

Cad-2007-01-StructureManagement/expandPSOneLevel

Parameters

Name	Type	Description
inputEntity	Object	The input for BOM Line which is being expanded one level.
businessObjectMapping	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

An entity of type TcConnector.ExpandPSOneLevelResponse.

Expanded child BOMLines can be retrieved using association TcConnector.expandPSOneLevelResponseBOMLines. Partial errors can be retrieved using association TcConnector.ResponseData and TcConnector.PartialErrors.

Find Saved Queries

Sends the request to and receive the response from the data provider. It routes the search request to a specific provider specified as providerName in the searchInput, assuming the searchCriteria for the provider is represented in the searchCriteriaInput object. For example, Awp0SavedQuerySearchProvider is the provider that is used for general search. The input criteria for GeneralSearch is passed through the searchCriteriaInput object.

SOA URL

Query-2010-04-SavedQuery/findSavedQueries

Parameters

Name	Type	Description
InputData	Object	This is a type of SearchInput entity that represents the information required for Searching the business object.
BusinessObject Mappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

An object of type TcConnector.FindSavedQueryResponse.

Find User Tasks

Finds the Task inbox of the logged-in user. The Task inbox contains **Tasks to Perform** and **Tasks to Track**.

SOA URL

Core-2007-01-Session/getTCSessionInfo

Core-2006-03-DataManagement/getProperties

Parameters

Name	Type	Description
InputData	Object	This is a type of SearchInput entity that represents the information required for Searching the business object.
BusinessObject Mappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

Task Inboxes can be retrieved using the association TcConnector.TaskInbox_FindUsersTasksResponse. The tasks_to_perform and tasks_to_track entities are the associations between EPMTask and TaskInbox. Partial errors can be retrieved using the association TcConnector.ResponseData or TcConnector.PartialErrors.

Get All Tasks

Returns a list of workflow tasks for the specified Teamcenter business object type.

SOA URL

Workflow-2008-06-Workflow/getAllTasks

Parameters

Name	Type	Description
InputData	Object	Input argument for getting the workflow tasks.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and Entity names, for example, ImanRelation=TcConnector.ImanRelation.

Returns

An entity of type TcConnector. GetAllTasksResponse that contains the workflow tasks.

Get Dataset File Types

Returns a list of Named References applicable for given the Dataset type. This Named Reference is required when creating a dataset.

SOA URL

Core-2007-06-DataManagement/getDatasetTypeInfo

Parameters

Name	Type	Description
dataset_type	String	Dataset Type for which list of NamedReference names is to be returned.

Returns

List of type TcConnector.Pair. This is a list of valid Named References for the input Dataset type.

Get Dataset Types

Returns list of available Dataset types. This Dataset type is required while creating a dataset.

SOA URL

Core-2010-04-DataManagement/getAvailableTypesWithDisplayNames

Parameters

None.

Returns

List of type TcConnector.Pair. This is a list of valid Teamcenter Dataset types.

Get Revision Rules

Gets all the persistent revision rules in the database.

SOA URL

Cad-2007-01-StructureManagement/getRevisionRules

Parameters

Name	Type	Description
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

GetRevisionRulesResponse which contains RevisionRuleInfo. RevisionRuleInfo has the revision rule. Partial errors can be retrieved using the association TcConnector.ResponseData or TcConnector.PartialErrors.

Get Properties

Receives properties of the specified model objects.

SOA URL

Core-2006-03-DataManagement/getProperties

Parameters

Name	Type	Description
ListOfModelObjects	Object	List of Mendix objects whose properties are to be fetched. All the properties available on the input object will be retrieved.
BusinessMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

An object of type TcConnector.ServiceResponse.

Get Teamcenter Session Information

Retrieves information about the Teamcenter Server session.

SOA URL

Core-2007-01-Session/getTCSessionInfo

Parameters

None

Returns

An object of type TcConnector.TcServerInfo.

Get Workflow Templates

Gets a list of workflow templates given in the list of target workspace objects and the **All** or **Assigned** criteria.

SOA URL

Workflow-2008-06-Workflow/getWorkflowTemplates

Parameters

Name	Type	Description
InputData	Object	The input data for the service operation.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

An entity of type GetWorkflowTemplateResponse. The workflow templates can be retrieved using the association TcConnector.Workflow templates. . Partial errors can be retrieved using TcConnector.ResponseData and TcConnector.PartialErrors.

Login

Authenticates user credentials and initializes a Teamcenter session for the Mendix client. This operation will throw an InvalidCredentialsException if the username, password or group is not valid.

When the client application is deployed to a 4Tier environment (communication through HTTP or TCCS) the login operation also contributes to the assignment of a Teamcenter server instance to the client session. The Teamcenter architecture varies from other client server architectures in that there is a dedicated instance of the Teamcenter server per client application. However, there are use cases where it is desirable for a single user to have multiple desktop applications running and each sharing a single instance of a Teamcenter server. This is controlled through the following elements:

hostPath	From the Connection class constructor, this specifies the address (URI) the Teamcenter server is hosted on.
Username	From this login operation, this specifies the user's Teamcenter user name.
sessionDiscriminator	From this login operation, this identifies the client session.

The hostPath argument determines the server machine that the client connects to. Once there, the pool manager on that host uses the username and sessionDiscriminator arguments of the login request to determine which Teamcenter server instance to assign the client to. If the pool manager has an existing Teamcenter server instance with the username/sessionDiscriminator key, the client is assigned to that existing instance of the Teamcenter server, and therefore sharing the server with another client; otherwise, a new instance of the Teamcenter server is used. There are a few general scenarios for the sessionDiscriminator argument:

Blank	If the user jdoe logs on to Teamcenter using two or more client applications using a blank sessionDiscriminator argument (for example, jdoe/), all of those clients are assigned to the same Teamcenter server instance. These client applications can be running on the same or different client hosts.
Constant	If the user **jdoe** logs on to Teamcenter using two or more client applications using a constant or fixed sessionDiscriminator argument (for example, jdoe/MyApp), those clients are assigned to the same Teamcenter server instance. This is similar to the blank sessionDiscriminator argument; the difference is that only multiple instances of the client application using myApp started by jdoe share the same Teamcenter server instance.

Unique	If the user **jdoe** logs on using a unique random-generated string (for example, jdoe/akdk938lakc), the client application will be assigned to a dedicated instance of the Teamcenter server.

The scenario you use depends on how your client application is used in the integrated environment. The most common case is the unique sessionDiscriminator value.

SOA URL

Core-2011-06-Session/login

Parameters

Name	Type	Description
UserCredentials	Object	Credentials to use for logging on to Teamcenter.

Returns

Basic information about the server and Partial Errors are returned when the authentication is successful but requested role or locale is not supported:

214102: The login is accepted, however the requested role is not valid, and the default role will be used.

214109: The login is accepted, however the login group was empty so default role will be used.

128003: The logon is accepted. However, data entry should be done using certain locales, as specified by the TC_language_data_entry preference. The details of the data entry are returned in the error message.

128004: The logon is accepted. However, data entry should only contain characters that belong to the encoding of the database instance. The information is in the error message.

Throws:

InvalidCredentialsException - When the credentials supplied are invalid or the requested locale is not allowed.

515143: The logon was refused due to invalid username or password.

515144: The logon was refused due to invalid username or password.

515142: The logon was refused due to an invalid group.

128001: The logon was refused due to conflict with the encoding of the database instance.

128002: The logon was refused due to missing localization.

Logout

Retrieves the Teamcenter session for the user and attempts to log them out of Teamcenter.

Once logged out the cookies associated with the session will be deleted and the Teamcenter Host Address within the session will be set to an empty string.

SOA URL

Core-2006-03-Session/logout

Parameters

None.

Returns

A Boolean type.

Perform Action

Performs the specified workflow action.

SOA URL

Workflow-2012-10-Workflow/performAction2

Parameters

Name	**Type**	Description
InputData	Object	Input for performing the workflow action.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, 'Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision'

Returns

An object of type TcConnector.ServiceResponse.

Created objects can be retrieved using associations TcConnector.ServiceResponse, TcConnector.ResponseData, and TcConnector.Created. Partial errors can be retrieved using association TcConnector.ResponseData and TcConnector.PartialErrors.

Perform General Search

Sends the request to and receive the response from the data provider Awp0SavedQuerySearchProvider for query **General...** The input criteria for GeneralSearch query is passed through the generalQuerySearchInput object that is extended from .SearchInput Object.

> **Note:**
>
> This service works only if the Teamcenter environment has an Active Workspace installation.

SOA URL

Query-2010-04-SavedQuery/findSavedQueries

Query-2014-11-Finder/performSearch

Parameters

Name	Type	Description
generalQuerySearchInput	Object	This is a type of SearchInput entity that represents the information required for Sea the business object.
businessObjectMapping	String	A semicolon-separated list of Teamcenter business object names and entity names example, Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision;WorkspaceObject nnector.WorkspaceObject.

Returns

An entity of type TcConnector.SearchResponse. Search Results can be retrieved using association TcConnector.searchResultsList. Partial errors can be retrieved using association TcConnector.ResponseData or TcConnector.PartialErrors.

Perform Item Simple Search

Sends the request to and receives the response from the data provider Awp0SavedQuerySearchProvider for the query **Item - simple**. The input criteria for GeneralSearch query is passed through the generalQuerySearchInput object which is extended from .SearchInput object.

> **Note:**
>
> This service works only if the Teamcenter environment has an Active Workspace installation.

SOA URL

Query-2010-04-SavedQuery/findSavedQueries

Query-2014-11-Finder/performSearch

Parameters

Name	Type	Description
itemSimpleQuerySearchInput	Object	This is a type of SearchInput entity which represents the information required for searching the business object.
businessObjectMapping	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

An entity of type TcConnector.SearchResponse. Search Results can be retrieved using association TcConnector.searchResultsList. Partial errors can be retrieved using association TcConnector.ResponseData or TcConnector.PartialErrors.

Perform Search

Sends the request to and receives the response from the data provider. It routes search request to a specific provider specified as providerName in the searchInput, assuming the searchCriteria for the provider is represented in the searchCriteriaInput object. For example, Awp0SavedQuerySearchProvider is provider that is used for general search. The input criteria for GeneralSearch is passed through the searchCriteriaInput object.

SOA URL

Query-2014-11-Finder/performSearch

Parameters

Name	Type	Description
InputData	Object	This is a type of SearchInput entity which represents the information required for searching the business object.
BusinessObjectMappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

An entity of type TcConnector.SearchResponse. Search Results can be retrieved using association TcConnector.searchResultsList. Partial errors can be retrieved using association TcConnector.ResponseData or TcConnector.PartialErrors.

Revise Object

Revises business objects. This operation revises the given objects and copies or creates new objects using the data for the property values and deep copy data, assuming the reviseInput for the object is provided. The input for revise object is passed through reviseInput entity.

SOA URL

Core-2015-07-DataManagement/getDeepCopyData

Core-2013-05-DataManagement/reviseObjects

Parameters

Name	Type	Description
objectToRevise	Object	The target business object being revised.
reviseInput	Object	This is a type of ReviseInput entity that represents the information required for revising the business object.
businessObjectMapping	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

An entity of type TcConnector.ReviseObjectsResponse. Revised objects can be retrieved using the association TcConnector.revise_output and the revise tree can be retrieved using TcConnector.reviseTrees. Partial errors can be retrieved using the association TcConnector.ResponseData or TcConnector.PartialErrors.

Saved Query Search

Searches for saved queries.

SOA URL

Query-2010-04-SavedQuery/findSavedQueries

Query-2008-06-SavedQuery/executeSavedQueries

Parameters

Name	Type	Description
QueryName	String	The name of the query for which search is to be performed.
InputData	Object	This is a type of SearchCriteria entity that represents the information required for searching the business object.
BusinessObject Mappings	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

An entity of type TcConnector.ServiceResponse. Search Results can be retrieved using association TcConnector.ResponseData or TcConnector.plain. Partial errors can be retrieved using association TcConnector.ResponseData or TcConnector.PartialErrors.

Update Properties

Updates Teamcenter objects corresponding to the input model object entities.

SOA URL

Core-2010-09-DataManagement/setProperties

Parameters

Name	Type	Description
modelObjects	List	A list of model object entities with updated values whose corresponding Teamcenter objects are to be updated on the Teamcenter site.
businessObjectMapping	String	A semicolon-separated list of Teamcenter business object names and entity names, for example, Item=TcConnector.Item;ItemRevision=TcConnector.ItemRevision;WorkspaceObject=TcConnector.WorkspaceObject.

Returns

An entity of type TcConnector.ServiceResponse. Partial errors can be retrieved using association TcConnector.ResponseData or TcConnector.PartialErrors. The list of modified model objects can be retrieved using the TcConnector.ResponseData or TcConnector.Updated association.

Upload File

Uploads one or more files to Teamcenter using the Teamcenter FMS service.

SOA URL

Core-2010-04-DataManagement/createDatasets

Core-2006-03-FileManagement/commitDatasetFiles

Parameters

Name	Type	Description
DatasetParameter	Object	Placeholder to maintain the files to be uploaded to Teamcenter. Dataset.Documents association should hold files to be uploaded.
NamedReference	String	Reference name to be used to associate uploaded files to the Dataset.

Returns

A Boolean type that returns True or False in case of success and failure, respectively. In addition to this, the UID property is updated on the Dataset object, which is passed as an input to action.

Siemens Industry Software

Headquarters

Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe

Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific

Suites 4301-4302, 43/F
AIA Kowloon Tower, Landmark East
100 How Ming Street
Kwun Tong, Kowloon
Hong Kong
+852 2230 3308

About Siemens PLM Software

Siemens PLM Software, a business unit of the Siemens Industry Automation Division, is a leading global provider of product lifecycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide. Headquartered in Plano, Texas, Siemens PLM Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens PLM Software products and services, visit www.siemens.com/plm.