Opcenter Connect FN for Teamcenter - Configuration Guide
# Contents

**Preface**  5

**Introduction**  1-1

**FN4T and Opcenter EX DS Interaction**
Integration Story  2-1

**Basic Configuration**
Merge T4S Mapping  3-1
Add T4x Job Agent  3-1
Mapping Templates and Preferences  3-1
Supported BOP Structures  3-2
The File fn4t_mapping_config.sd  3-4

**General Remark**  4-1

**Demo Workflow Overview**  5-1

**Equipment**
Create Opcenter EquipmentType  6-1
Maintain EquipmentType  6-2
Create EquipmentConfiguration  6-2
Maintain EquipmentConfiguration  6-4
Create EquipmentGraphConfiguration  6-5
Create EquipmentHierarchy  6-6

**150 Percent BOP Components**
Opcenter EX DS Material  7-1
  Create Opcenter EX DS Material  7-1
  Maintain Opcenter EX DS Material Fields  7-2
  Create and Associate Document to Opcenter EX DS Material  7-2
Opcenter EX DS Tool Definition  7-2
  Create Opcenter EX DS Tool Definition  7-2
  Maintain Opcenter EX DS Tool Definition Fields  7-3
  Create and Associate Document to Opcenter EX DS Tool Definition  7-3
Opcenter EX DS Process  7-3
  Create Opcenter EX DS Process  7-3
  Maintain Opcenter EX DS Process  7-4
Opcenter EX DS Operation  7-4
  Create Opcenter EX DS Operation  7-5
Preface

This documentation cannot be used as a substitute for consulting advice, because it can never consider the individual business processes and configuration. Despite our best efforts it is probable that some information about functionality and coherence may be incomplete.

Issue: December 2019

Legal notice:

All rights reserved. No part of this documentation may be copied by any means or made available to entities or persons other than employees of the licensee of the Opcenter Connect FN for Teamcenter or those that have a legitimate right to use this documentation as part of their assignment on behalf of the licensee to enable or support usage of the software for use within the boundaries of the license agreement.

© 2018-2019 Siemens Product Lifecycle Management Software Inc.

Trademark notice:

Siemens, the Siemens logo and Opcenter are registered trademarks of Siemens AG.

Camstar and Teamcenter are trademarks or registered trademarks of Siemens Product Lifecycle Management Software Inc. or its subsidiaries in the United States and in other countries.

Oracle is a registered trademark of Oracle Corporation.

SAP, R/3, SAP S/4HANA®, SAP Business Suite® and mySAP are trademarks or registered trademarks of SAP or its affiliates in Germany and other countries.

TESIS is a registered trademark of TESIS GmbH.

All other trademarks, registered trademarks or service marks belong to their respective holders.
1. Introduction

The Opcenter Connect FN for Teamcenter (FN4T) software solution is an integration software that provides data and process integration between Teamcenter® by Siemens Product Lifecycle Management Software Inc. and Opcenter Execution Discrete by Siemens AG. FN4T comes with a demo configuration assuming that Teamcenter Gateway for SAP Business Suite (or Teamcenter Gateway for SAP S/4HANA) is in place in order to close the triangle with SAP ERP. The connection between ERP and MES is assumed to be implemented by Opcenter Connect FN for SAP S/4HANA (FN4S). Note that FN4T may also be operated without Teamcenter Gateway for SAP Business Suite, FN4S and SAP ERP. In this case order and order confirmation are out of scope.

FN4T provides a wide range of automatic, interactive and workflow functions to transfer and synchronize data between Teamcenter and Opcenter EX DS.

This document details the components of Teamcenter - Opcenter EX DS integration, which are available out of the box and configurable to meet customer specific solution requirements. It also describes the prerequisites for the integration of SAP ERP using the Teamcenter Gateway for SAP Business Suite.
While Opcenter Connect FN for Teamcenter (FN4T) only covers the integration of Teamcenter with Opcenter Execution Discrete, the integration story makes most sense when an ERP system is also included. In this chapter, we assume SAP Business Suite® is used as the ERP system, and the Teamcenter Gateway for SAP Business Suite is used for the integration between Teamcenter and SAP Business Suite®. FN4T can be used with other ERP systems and integration solutions or without an ERP, though. Note that FN4T does not cover shop floor automation.

The sample scenario assumes a structure with options and variants or unit or date effectivity applied to the BOP and/or the BOM within Teamcenter. This results in a configurable structure in Teamcenter containing more components than necessary to produce a single product. This is called 150% structure. In contrast a configured structure contains exactly the components necessary to produce a single specific product (or unit) and is called a 100% structure. In the context of MES, a pure 100% structure without detailed master data of the components is called a skeleton or "As Planned BOP".

This chapter will explain the overall integration scenario, which consists of three "closed loops" (business use cases spanning two or three systems).

Before the scenarios starts, we assume that the Teamcenter engineer (or another Teamcenter user) has released the Bill Of Resources to the MES. This included transfers of (limited) object meta data as well as the structure of the resources. Since this release process and the synchronization usually happens asynchronously to the transfer of a BOP, the demo scenario provides separate workflows for this purpose.
Closed Loop 1

This closed loop handles the initial master data transfer (150% unconfigured components) and skeleton transfers (100% configured structures) and supplies Opcenter EX DS with the necessary data to execute an order.

1. The Teamcenter engineer prepares and verifies the Bill Of Process (BOP), the Bill Of Materials (BOM) and the Bill Of Resources in Teamcenter Manufacturing Process Planner (MPP). He also creates and verifies the variability of the product using classic options and variants or unit or date effectivity. This step is not covered in this manual as it is not related to integration. For further help on MPP, please consult the Teamcenter manuals. Please note that there are some restrictions and requirements the authored data must conform to in order to support the FN4T scenario. These will be explained in subsequent chapters.

2. The Teamcenter engineer (or another Teamcenter user) releases the components of the master data like e.g. Material, Process, Operation, etc. from Teamcenter to the target ERP and MES systems. Structures, relations and configurations are not relevant in this step. Instead, it only cares about the building blocks of the unconfigured structures (150%).
3. The Teamcenter engineer (or another Teamcenter user) releases one or several configured structures (100% BOM and BOP, the "skeleton") from Teamcenter to SAP. This step assumes that all necessary components of the structures have already been transferred in the previous step.

4. The SAP operator creates and releases a SAP production order to MES, using the business objects created in the previous SAP releases and additional ERP-internal processes. Note that order transfer is not part of FN4T but is implemented e.g. in FN4S. If the product configuration used in the ERP order is already known in the MES system, the MES system can immediately process and execute the order, if needed. If the MES system Opcenter EX DS does not yet know the details of the product configuration, then it automatically triggers the download of the configured production structure (100 % BOP, "skeleton") from Teamcenter via FN4T.

The sequence of the steps is not strictly required. For example, the transfers to Opcenter EX DS and SAP can easily be split into smaller pieces and executed at different points in time and by different operators based on the maturity of sub-structures. Also, transfers may be repeated, e.g. if data errors have to be corrected. The only restriction is that all components of a structure must be known in the target system before the structures can be transferred.

Now that all necessary data is available in MES, the Opcenter EX DS operator can start the execution of the WorkOrder. The execution itself is not covered in this manual as it is not part of the integration. For details see the manuals of Opcenter Execution Discrete. The non-conformance loop is also part of the execution and includes integration-relevant transfers and is covered in the next section (Closed Loop 2). Finishing the WorkOrder is covered in the last section of this chapter (Closed Loop 3).

**Closed Loop 2**

In case a non-conformance is detected during order execution in Opcenter Execution Discrete that needs to be escalated to product development (engineering, i.e. the PLM system Teamcenter), FN4T supports the automatic synchronization of a change object in Teamcenter.
1. An Opcenter EX DS operator executes the WorkOrder on the shop floor. The operator creates a Non-Conformance Notification (NCN) during WorkOrder execution. Opcenter EX DS reports the creation to FN4T and FN4T creates a problem report in Teamcenter. Teamcenter will automatically retrieve details on the NCN and populate the problem report with that data. Based on the NCN status a configurable workflow is started in Teamcenter, which might e.g. set the corresponding status or notify an engineer of the NCN.

2. In Opcenter EX DS the operator and the supervisor decide how to proceed with the NCN. The decision may result in a status change in Opcenter EX DS. Each status change of the NCN in Opcenter EX DS is reported to FN4T and again a configurable workflow template gets started, which usually only sets the according status and ends. In case the NCN needs to be escalated to engineering, a certain status is used.

3. If the NCN was escalated in the last step, the corresponding Teamcenter workflow will inform an engineer.

4. The engineer will analyze the provided data and view live data of the Opcenter EX DS WorkOrder in the EXDS Dataview in Teamcenter. If necessary, appropriate actions are started to correct engineering data (not requiring integration). When the corrections are completed, changed data must be updated in SAP and Opcenter EX DS. That essentially means to repeat some of the steps
from the previous section (closed loop 1). Finally the engineer documents recommendations to the shop floor in the problem report. Documents may also be attached.

5. Finishig the Teamcenter workflow, FN4T will transfer the updated data back to the Opcenter EX DS and update the status of the non-conformance.

6. The Opcenter EX DS operator can now verify the NCN status and additional data or documents. The NCN can now be handled further and its status may change again. The loop may repeat if shop floor decides to escalate once more (step 2).

Closed Loop 3

This closed loop covers finishing the Opcenter EX DS WorkOrder and the operation or order confirmation back to SAP Business Suite®.

1. When the Opcenter EX DS work order is completed, Opcenter EX DS will automatically confirm the production order in SAP Business Suite®. Alternatively, based on MES-internal configuration, Opcenter EX DS may confirm each operation to notify the ERP of the progress. In this case the ERP system will assume the order to be completed when all operations are confirmed. Note that confirmation is not part of FN4T, but is implemented in FN4S.
2. The SAP operator can now finalize the production order and e.g. verify or correct goods movements. This is a SAP internal process and requires no CLM integration.

3. The Teamcenter engineer can view live data of the Opcenter EX DS WorkOrder and the SAP production order and its components in the Opcenter or T4S Dataview in Teamcenter.
3. Basic Configuration

Merge T4S Mapping

Note:

This chapter is only relevant if Teamcenter Gateway for SAP Business Suite or Teamcenter Gateway for SAP S/4HANA is used in addition.

In case you are already using or plan to use Teamcenter Gateway for SAP Business Suite or Teamcenter Gateway for SAP S/4HANA (T4S or T4S4) to provide integration from PLM (Teamcenter) to ERP (SAP), you have to adapt your T4S mapping. FN4T provides a sample mapping for T4S. In the simplest use case, when there is no custom T4S mapping (only the OOTB T4S demo mapping), you can just delete any existing directory `<GS_ROOT>/var/mmap/t4s_mapping_config` and then copy the FN4T sample mapping directory `<GS_ROOT>/var/template/fn4t/mmap/t4s_mapping_config` to `<GS_ROOT>/var/mmap`. If there is a custom mapping you want to keep, you will have to merge the existing t4s_mapping_config with the FN4T sample mapping.

Add T4x Job Agent

For some external events, FN4T needs to execute jobs in the background (e.g. incoming web services from MES, long-running transfer workflows). Therefore the FN4T GS instance(s) need to be configured to have at least one Job Agent. For details on Job Agent creation, see chapter T4x Agent configuration in the Active Integration - Generic Configuration Guide. Configuring at least one Job Agent is necessary for the demo scenario to work. In case of several GS instances, at least one of the instances has to have a job agent. From a functional point of view a single one is enough to cover FN4S and FN4T functionality. The job agent needs to connect to Teamcenter and needs credentials. These have to be created upfront using the script `itk_connect2db.tcl` and made available to the job agent by calling `::ITK::setCredentialsAlias` for the alias in the mapping. The Teamcenter user name for the demo scenario should be “t4xbatch” and the user should be in the “dba” group to avoid permission issues.

Mapping Templates and Preferences

FN4T supports different Bill of Process (BOP) structures created differently in Teamcenter (see Supported BOP Structures for details) while providing a single set of mapping files and workflows. Because the semantics of a Teamcenter operation differ between the different BOP types, FN4T provides different TargetTypeNames for the 150% components (no structure). Each TargetTypeName has its own set of preferences, a separate TCL namespace and in most cases a separate mapping file. Many procedures within the specific implementations of the 150% TargetTypeNames are identical (e.g. for Operation and PlantBOPOperation) and so the procedures just forward to a common implementation. Some of these shared implementations are in the mapping file `fn4t_toolbox.sd`.

The 150% transfer (no structure) evaluates only master data of the components, but for DCDs it is necessary to evaluate structures: In Teamcenter assignments of DCDs to an operation or step have to be made in context of a structure (occurrences) while Opcenter EX DS assumes data points are related to
operations or steps independent of a structure. In order to evaluate the correct data collections FN4T evaluates the object to extract master data (e.g. for operations and steps). In a separate transfer task in the workflow it then evaluates the structure below operations and steps to transfer data collections and relate them to their parent.

Many adoptions of the OOTB behaviour of these transfers can be achieved without understanding or modifying the mapping logic by just changing the settings in the file fn4t_custom_parameter.sd. The mapping of plants to Opcenter systems, several Teamcenter type names and attribute names used by the mapping, transferred file types, the reverse mapping, settings for creation of the CC object and Teamcenter workflow templates can be modified here.

The following chapters explain the different transfers in detail. For a deeper understanding of the basic mapping implementation and hints on customization, please also see Introduction.

**Supported BOP Structures**

The OOTB FN4T demo scenario supports three Teamcenter BOP structure types. Each BOP type consists of three levels that correspond directly to the Opcenter EX DS objects Process, Operation and Step. The following table shows the possible combinations:

<table>
<thead>
<tr>
<th>Opcenter type and level</th>
<th>Classic BOP</th>
<th>Compound BOP</th>
<th>Plant BOP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Teamcenter Type</td>
<td>TargetTypeName</td>
<td>Teamcenter Type</td>
</tr>
<tr>
<td>Process (level 1)</td>
<td>MEProcess</td>
<td>Process</td>
<td>MEP</td>
</tr>
<tr>
<td>Process Attachments</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Operation (level 2)</td>
<td>MEOP</td>
<td>Operation</td>
<td>Mfg0MEOCompOP</td>
</tr>
<tr>
<td>Operation Attachments</td>
<td>MEActivity, MENCP Program or MENCToolActivity</td>
<td>implicitly handled by Operation</td>
<td>Mfg0MSTEP</td>
</tr>
<tr>
<td>Step (level 3)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Consumed material, workarea, tool/resource, equipment, characteristic representation with specification
- Time
- Files (e.g. textual work instruction, image, 3D snapshot, ...)
- DCD
- Skill

- Consumed material, workarea, tool/resource, equipment, characteristic representation with specification
- Time
- Files (e.g. textual work instruction, image, 3D snapshot, ...)
- DCD
- Skill

- Consumed material, workarea, tool/resource, equipment, characteristic representation with specification
- Time
- Files (e.g. textual work instruction, image, 3D snapshot, ...)
- DCD
- Skill
Explanations:

- In the interest of shortness, the table shows the (internal, non-localized) Teamcenter item type names. However, the target of an actual transfer and the name used in the preference definition must be the corresponding revision type.

- TargetTypeName is the name used in FN4T workflow handlers and preferences in Teamcenter and in the mappings that identifies a certain type of transfer. Note that you have to use a specific and different type in case of certain PlantBOP transfers.

- Most FN4T demo workflow templates delegate workflow execution to the T4x job engine, so the interactive user’s Teamcenter session is not blocked. In the OOTB configuration, the job engine uses the Teamcenter user "t4xbatch" to execute the tasks. In case of an error, the workflow is routed to an interactive "Do" task that will pop up in the Teamcenter Inbox of user "engineer", who is intended to fix data errors and then complete the workflow.

- The FN4T demo workflow templates CLM_Release_150PerCent StandAloneObjects_Factory1 (or ..._AllFactories) and CLM_Release_150PerCent StandAloneObjects PlantBOP can be used on all components of a complete BOP. They consist of several tasks specific to certain parts of the BOP. Each task will pick the relevant parts and ignore the irrelevant targets. This way you can select all components of an expanded BOP in Manufacturing Process Planner and simply start that single workflow template on it. In case of data errors in the workflow, do not abort and restart the whole workflow instance, but instead correct the error in the data and then complete the previously failed task or the corresponding "Do" task.

- Classic BOP and compound BOP can be mixed, i.e. on the second level of the BOP under an MEProcess you may use MEOps (with or without Activities) and MfgOMECompOPs in parallel.

- Teamcenter Activities do not have a specific TargetTypeName for the transfer. Instead they are transferred by TargetTypeName Operation together with the operation (MEOP revision) they are attached to.

- The listed attachments are partially handled by separate TargetTypeName (DCD, PlantBOPDCD, Skill), and partially together with the parent TargetTypeName (files, time, skill association). Consumptions (material, workarea) are not transferred in the 150% workflow but in the 100% transfer and are listed here for completeness only. Strikethrough attachments will not be transferred for this specific case due to data model restrictions of either Teamcenter or Opcenter.
• Characteristics Representation Element with attached Specification are maintained in Teamcenter Active Workspace. To find it on easier way later in Teamcenter RAC, the Characteristics Representation Element could be defined as a part of the Control Plan structure. You can not change any Characteristics Representation Element in Teamcenter RAC. It can be only attached, for example, to an operation or a step in your BOP Structures.

Caution:
• The FN4T demo scenario as provided in the demo mapping and the "t4clm" demo template, including Teamcenter data types, workflow templates, preferences, users, etc. is only meant as a sample configuration and must not be applied to productive environments or environments without T4SI! Instead it is necessary to analyze customers needs and processes and create a custom specific configuration depending on only the base template "sit0gateway".

• While Opcenter supports several levels of Processes, the FN4T demo supports only a single process level. To transfer a multi-level process BOP, transfer all the process "leafs" separately.

• The interpretation of an MEOP object is different depending on the structure type you are working in: For a classic or a compound BOP a Teamcenter MEOP translates to a Opcenter EX DS Operation, whereas for a PlantBOP a Teamcenter MEOP translates to a Opcenter EX DS Step. For the 150% transfert (no structure) there are different Teamcenter workflow templates that take care of this difference. This is the reason why you can't mix a PlantBOP structure with the other two types. All other actions (100% transfer) are identical.

• The top-level Process in all structure types has to have a single product structure assigned via the GRM relation "Mfg Targets" ("IMAN_METarget") and a single top-level work area with GRM relation "IMAN_MEWork area" ("IMAN_MEWorkArea"), else a later request to create a Teamcenter CollaborationContext object or a configuration request from FN4T will fail.

The File fn4t_mapping_config.sd

The file fn4t_mapping_config.sd in the directory <GS_ROOT>\var\mmap\fn4t_mapping_config is the first configuration file of the FN4T software that is read by each TCL worker thread or process. It loads the rest of the mapping files and may contain basic settings as well as variables which are supposed to be used in more than one mapping file.

The following basic settings can be configured in fn4t_mapping_config.sd:

• ::ITK::setCredentialsAlias allows to set the alias to be used to make the Teamcenter connection ("Default@Teamcenter" by default, but this can be changed). The credentials for this alias must be set using the script itk_connect2db.tcl.

• ::FN4T::CONNECTION2EA::selectActiveConnection2EA, ::T4X::CONNECTION2EA::setCredentialsAlias4UseCase and ::FN4T::CONNECTION2EA::readConnectionInfo4Session are used to setup the connection(s) to Opcenter. The credentials to be used must be set using the script exdsconnect.tcl and the hostname and port of the Opcenter host must be set as communication parameters.
channel in the Admin UI of the Gateway Service. Please note that `::T4X::CONNECTION2EA::setCredentialsAlias4UseCase` expects three arguments `ProductKey`, `Destination` and the `CredentialsAlias`. The only implemented authentication method for FN4T is "2-legged" OAuth2 (see Install Certificate in the Active Integration - Installation Guide for details).

- **Sourcing of mapping files** with `source -relax`.
  Example: `source -relax fn4t_material_mapping.sd`. This means FN4T has to read the content of the file `fn4t_material_mapping.sd`. The argument `-relax` means the file is looked up in the memory first and only if its content is not there, the file is loaded from disc (only in the same directory `<GS_ROOT>\var\mmap\fn4t_mapping_config`).
  In order to not use a mapping file, it is not enough to not "source" it in `fn4t_mapping_config.sd`. We strongly recommend keeping only those mapping files in the `mmap` directory that you really want to use!
  Be sure to have the correct file names (the files located in `<GS_ROOT>\var\mmap\fn4t_mapping_config`) in the "source" section of this file.

- In addition to the above settings, this file is also the place to define the configuration for multiple Opcenter EX DS servers (see chapter Addressing Multiple Simatic IT Servers in the Active Integration - Installation Guide) and to configure the language of internal messages directly returned by FN4T.

- The code starting with `set overlay...` is only relevant in SPLM-internal development and testing environments and will not become active in any custom environment, so it can be ignored completely.

- **Configure the language of internal messages directly returned by FN4T:**
  Example:

  ```
  set strLanguageCode "en_US" ; # default language (check /var/lang/FN4T/ for more language codes)
  ```

  Other possible languages can be found in the directory `<T4x_GS_ROOT>\var\lang\FN4T>`. It is also possible to control the language using an OS environment variable:

  ```
  set TP_LANGUAGE_CODE=de_DE
  ```
4. General Remark

The following chapters give an overview of the capabilities of the Opcenter Connect FN for Teamcenter. This part of the manual is meant to support adaption of the solution to customer needs by linking each capability to the piece of TCL code where it is implemented and can be adapted. Basic knowledge of the TCL programming language is necessary to understand or change the mapping logic.

The "Maintain" chapters describe how fields can be adapted. Note that "Maintain" in this context does not imply "Update": While the Active Integration Gateway platform supports updates of existing objects, most of the Opcenter EX DS commands used to implement the capabilities do not support updates of existing objects. Instead the result of such a call will be "success" but the data within the Opcenter EX DS database will not have changed. Note that usually a clean Teamcenter release process would only transfer released (i.e. unchangeable) objects to Opcenter EX DS. In order to semantically change master data or structures, items, processes, operations, etc. have to be revised on Teamcenter side. When transferring the new revisions, they will establish new objects on Opcenter EX DS side, hence storing the changed values correctly.
5. Demo Workflow Overview

The FN4T demo template t4clm comes with several sample workflows to transfer the BOP components from Teamcenter to the target systems. Note that the demo workflows assume T4S to be in place. The workflows can be adapted as needed (e.g. remove T4S components, adapt target system or connection names) and can be used as templates for the creation of specific workflows. Some of the workflows use other sub-workflows because they implement a repeated sequence of identical tasks. Here is an overview of the demo workflows that should be used for 150% transfers. In general all the workflows starting with "CLM" are meant to be started manually during the demo scenario, while workflows with prefix "FN4T" are internal or automatically started workflows or are components from which specific custom workflows can be built. These are the workflow templates important for the demo scenario:

- **CLM_Release_150PerCent_StandAloneObjects_Factory1**: This workflow is an example to transfer master data of a classic or compound BOP to a single SAP system and to a single Opcenter EX DS instance. The workflow can easily be extended to support a second or several Opcenter EX DS instances, see next item. The workflow handles all types of components that make up a BOP like materials (items), process, operations, etc. as described in the following chapters. It does not transfer any structure information.

- **CLM_Release_150PerCent_StandAloneObjects_AllFactories**: This workflow is an example to transfer master data to a single SAP system and to two Opcenter EX DS instances. It is an extension of the previous workflow.

- **CLM_Release_150PerCent_StandAloneObjects_PlantBOP**: This workflow is similar to the first one (transfer to a single SAP and single Opcenter EX DS system), but it covers a plant BOP instead of a classic or compound BOP. The difference manifests in different TargetTypeNames in the handler configuration, accepting different Teamcenter types and interpreting the type MEOP differently. See Supported BOP Structures for details. This workflow can also be extended to support several Opcenter EX DS systems, if needed.

The following workflows are not part of the demo scenario but cover additional use cases:

- **CLM_EquipmentType_Transfer**: Sample workflow to transfer Teamcenter work area revisions as Opcenter EX DS EquipmentType objects.

- **CLM_EquipmentHierarchy_Transfer**: This sample workflow should be started on the top-level element of a work area structure in Teamcenter (MESiteRevision or MEPlantRevision) and will create an EquipmentHierarchy in Opcenter EX DS, evaluating the multi-level BOM and using the EquipmentType objects which must already exist in Opcenter EX DS.

- **CLM_MasterPlan_Transfer**: This sample workflow allows to transfer a Teamcenter CollaborationContext object and creates a Opcenter EX DS MasterPlan, which can later be used in an order which is based on that MasterPlan. The CC object must have been transferred to ERP already, since the workflow evaluates the attached ERP form in order to create the name and ID of the MasterPlan.
6. Equipment

Create Opcenter EquipmentType

The capability to create a EquipmentType in Opcenter is based on several item revision types released from Teamcenter. The actual transfer is implemented in procedure ::FN4T::GENOBJ::MAPPING::EQUIPMENTTYPE::performTransfer which is not part of the custom mapping but can be overwritten, if needed. The custom mapping queries Opcenter for the given EquipmentType NId, retrieving the internal Id, and creates the object if the query returns no result.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>EquipmentType</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>MESiteRevision,</td>
</tr>
<tr>
<td></td>
<td>MEPlantRevision,</td>
</tr>
<tr>
<td></td>
<td>MELineRevision,</td>
</tr>
<tr>
<td></td>
<td>MEStationRevision,</td>
</tr>
<tr>
<td></td>
<td>MEWorkareaRevision,</td>
</tr>
<tr>
<td></td>
<td>MfgOMEEquipmentRevision</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter</td>
</tr>
<tr>
<td>Used API</td>
<td>ODATA Query: /EquipmentType</td>
</tr>
<tr>
<td></td>
<td>$filter=NId=&lt;item_id&gt;_&lt;item_revision_id&gt;</td>
</tr>
<tr>
<td></td>
<td>CreateEquipmentType</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/</td>
</tr>
<tr>
<td></td>
<td>fn4t_mapping_config/</td>
</tr>
<tr>
<td></td>
<td>fn4t_equipment_type_mapping.</td>
</tr>
<tr>
<td></td>
<td>sd</td>
</tr>
<tr>
<td>Custom namespace</td>
<td>::FN4T::GENOBJ::CUSTOM:</td>
</tr>
<tr>
<td></td>
<td>::MAPPING::EQUIPMENTTYPE</td>
</tr>
<tr>
<td>Preferences start with</td>
<td>FN4T_EquipmentType</td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td>::FN4T::GENOBJ::MAPPING::EQUIPMENTTYPE::performTransfer</td>
</tr>
<tr>
<td>Sample workflow in demo template</td>
<td>&quot;CLM_EquipmentType_Transfer&quot;</td>
</tr>
</tbody>
</table>

The dict MESInputDatEquipmentType contains fields mapped using preferences and calling ::T4X::TC::MAPPING::FieldMapping.
Maintain EquipmentType

The capability to maintain a EquipmentType in Opcenter is based on several item revision types released from Teamcenter. The actual transfer is implemented in procedure `::FN4T::GENOBJ::MAPPING::EQUIPMENTTYPE::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. The custom mapping queries Opcenter for the given EquipmentType NId, retrieving the internal Id, and updates the object if the Id was found.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
</tr>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API</td>
</tr>
<tr>
<td>Mapping template</td>
</tr>
<tr>
<td>Custom namespace</td>
</tr>
<tr>
<td>Preferences start with</td>
</tr>
<tr>
<td>Dictionaries expected by <code>::FN4T::GENOBJ::MAPPING::EQUIPMENTTYPE::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow in demo template</td>
</tr>
</tbody>
</table>

The dict `MESInputDatEquipmentType` contains fields mapped using preferences and calling `::T4X::TC::MAPPING::FieldMapping`.

Create EquipmentConfiguration

The capability to create a EquipmentConfiguration in Opcenter is based on several item revision types released from Teamcenter. The actual transfer is implemented in procedure `::FN4T::GENOBJ::MAPPING::EQUIPMENTHIERARCHY::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. The OOTB configuration assumes that each Teamcenter revision is transferred as EquipmentType in a simple object transfer first and then
the occurrence of the same revision is transferred to become an EquipmentConfiguration. This condition is not verified by the mapping; instead Opcenter may throw an error if the reference from an EquipmentConfiguration to an EquipmentType cannot be established. The custom mapping can extract multiple levels of the Equipment BOM in Teamcenter and will handle all matching revisions. It queries Opcenter for the given EquipmentConfiguration NId, retrieving the internal Id, and creates the object if the query returns no result. The NId of the resulting objects composes of the item Id and revision Id of the object itself and an appended path which contains a component for every ancestor from the top level to parent. The path component is the reference designator of the ancestor BOMLine if present, else the item Id and revision Id of the ancestor revision. Using the reference designator allows to separate several occurrences of the same revision for the a parent, but requires the Teamcenter author to assign unique reference designators (the system does not check the uniqueness).

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TargetTypeName</strong></td>
</tr>
<tr>
<td><strong>Teamcenter Types</strong></td>
</tr>
<tr>
<td><strong>Target system</strong></td>
</tr>
<tr>
<td><strong>Used API</strong></td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
</tr>
<tr>
<td><strong>Custom namespace</strong></td>
</tr>
<tr>
<td><strong>Preferences start with</strong></td>
</tr>
<tr>
<td><strong>Dictionaries expected by</strong></td>
</tr>
<tr>
<td><strong>Sample workflow in demo template</strong></td>
</tr>
</tbody>
</table>
The dictionary `MESInputDatEquipmentConfigurationDictList` has a list with a dict for each `EquipmentConfiguration`. Each such dict contains fields mapped using preferences and calling ::T4X::TC::MAPPING::FieldMapping.

## Maintain EquipmentConfiguration

The capability to maintain a `EquipmentConfiguration` in Opcenter is based on several item revision types released from Teamcenter. The actual transfer is implemented in procedure ::FN4T::GENOBJ::MAPPING::EQUIPMENTHIERARCHY::performTransfer which is not part of the custom mapping but can be overwritten, if needed. The custom mapping can extract multiple levels of the Equipment BOM in Teamcenter and will handle all matching revisions. It queries Opcenter for the given `EquipmentConfiguration` NId, retrieving the internal Id, and updates the object if the query returns the NId. The NId of the resulting objects composes of the item Id and revision Id of the object itself and an appended path which contains a component for every ancestor from the top level to parent. The path component is the reference designator of the ancestor BOMLine if present, else the item Id and revision Id of the ancestor revision. Using the reference designator allows to separate several occurrences of the same revision for the a parent, but requires the Teamcenter author to assign unique reference designators (the system does not check the uniqueness).

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TargetTypeName</strong></td>
</tr>
<tr>
<td><strong>Teamcenter Types</strong></td>
</tr>
<tr>
<td><strong>Target system</strong></td>
</tr>
<tr>
<td><strong>Used API</strong></td>
</tr>
<tr>
<td><strong>UpdateEquipmentConfiguration</strong></td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
</tr>
<tr>
<td><strong>Custom namespace</strong></td>
</tr>
<tr>
<td><strong>Preferences start with</strong></td>
</tr>
<tr>
<td><strong>Dictionaries expected by</strong></td>
</tr>
</tbody>
</table>

© 2019 Siemens
The dictionary `MESInputDatEquipmentConfigurationDictList` has a list with a dict for each `EquipmentConfiguration`. Each such dict contains fields mapped using preferences and calling `::T4X::TC::MAPPING::FieldMapping`.

### Create EquipmentGraphConfiguration

The capability to create a `EquipmentGraphConfiguration` in Opcenter is based on `MESiteRevision` or `MEPlantRevision` types released from Teamcenter. The actual transfer is implemented in procedure `::FN4T::GENOBJ::MAPPING::EQUIPMENTHIERARCHY::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. The OOTB configuration assumes that each Teamcenter revision is transferred as `EquipmentType` in a simple object transfer first and then the top level element is transferred to become an `EquipmentGraphConfiguration`. This condition is not verified by the mapping; instead Opcenter may throw an error if the reference from an `EquipmentGraphConfiguration` to an `EquipmentType` or an `EquipmentConfiguration` cannot be established. The custom mapping can extract multiple levels of the Equipment BOM in Teamcenter, but only the top level element will create an `EquipmentGraphConfiguration`. It queries Opcenter for the given `EquipmentGraphConfiguration` NId, retrieving the internal Id, and creates the object if the query returns no result or deletes and creates the object, if the NId was present. Note that in this case also the `EquipmentHierarchy` below the `EquipmentGraphConfiguration` gets deleted and re-created. The NId of the resulting object composes of the item Id and revision Id of the top-level revision.

### Export

<table>
<thead>
<tr>
<th>Export</th>
<th>EquipmentConfigurationDictList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample workflow in demo template</td>
<td>&quot;CLM_EquipmentHierarchy_Transfer&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TargetTypeName</th>
<th>EquipmentHierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teamcenter Types</td>
<td>MESiteRevision or MEPlantRevision.</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter</td>
</tr>
<tr>
<td>Used API</td>
<td>ODATA Query: /EquipmentGraphConfiguration $filter=NId=&lt;EquipmentGraphConfigurationNId&gt;</td>
</tr>
<tr>
<td></td>
<td>DeleteEquipmentGraphConfiguration</td>
</tr>
<tr>
<td></td>
<td>CreateEquipmentGraphConfiguration</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_equipment_hierarchy_bom_mapping.sd</td>
</tr>
</tbody>
</table>
The dictionary `MESInputDatEquipmentGraphConfiguration` is a dict representing a single `EquipmentGraphConfiguration`. The fields are mapped using preferences and calling `::T4X::TC::MAPPING::FieldMapping`.

### Create EquipmentHierarchy

The capability to create a `EquipmentHierarchy` in Opcenter is based on several item revision types released from Teamcenter. The actual transfer is implemented in procedure `::FN4T::GENOBJ::MAPPING::EQUIPMENTHIERARCHY::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. The OOTB configuration assumes that each Teamcenter revision is transferred as `EquipmentType` in a simple object transfer first and then the top level element of the BOM is transferred as `EquipmentGraphConfiguration` (representing the whole hierarchy) and occurrences of all (relevant) revisions in the BOM are transferred to become `EquipmentConfigurations` (representing the nodes in the hierarchy). Since `EquipmentConfiguration`, `EquipmentGraphConfiguration` and `EquipmentHierarchy` are all transferred during the same BOM transfer, the referential integrity is ensured. The custom mapping extracts multiple levels of the Equipment BOM in Teamcenter and will handle all matching revisions. Note that the transfer of the hierarchy is a single call with an encoded JSON payload (JSON in JSON). Since the `EquipmentGraphConfiguration` and the `EquipmentHierarchy` have a 1:1 relation, there is no need to transfer an NId of the `EquipmentHierarchy`. There is also no update since the transfer deletes the old hierarchy before creating a new one.
### Export

<table>
<thead>
<tr>
<th>Mapping template</th>
<th><code>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_equipment_hierarchy_bom_mapping.sd</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom namespace</td>
<td><code>::FN4T::GENBOM::CUSTOM::MAPPING::EQUIPMENTHIERARCHY</code></td>
</tr>
<tr>
<td>Preferences start with</td>
<td><code>FN4T_EquipmentHierarchy</code></td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td><code>::FN4T::GENBOM::CUSTOM::MAPPING::EQUIPMENTHIERARCHY::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow in demo template</td>
<td>&quot;CLM_EquipmentHierarchy_Transfer&quot;</td>
</tr>
</tbody>
</table>

The dictionary `MESInputDatEquipmentHierarchy` has a list with a single dict containing the reference to the `EquipmentGraphConfiguration` and the hierarchy dict encoded in a JSON string value. The fields are mapped using preferences and calling `::T4X::TC::MAPPING::FieldMapping`. 
7. 150 Percent BOP Components

Opcenter EX DS Material

Create Opcenter EX DS Material

Maintain Opcenter EX DS Material Fields

Create and Associate Document to Opcenter EX DS Material

Create Opcenter EX DS Material

The capability to create a Material in Opcenter EX DS is based on ItemRevision or PartRevision released from Teamcenter. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::MATERIAL::performTransfer` which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Material</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>ItemRevision, PartRevision</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>PLMCreateMaterialDefinitionList</td>
</tr>
<tr>
<td>Mapping template</td>
<td><code>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_material_mapping.sd</code></td>
</tr>
<tr>
<td>Custom namespace</td>
<td><code>::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL</code></td>
</tr>
<tr>
<td>Preferences start with</td>
<td>FN4T_Material</td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td><code>::FN4T::GENOBJ::MAPPING::MATERIAL::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;FN4T_Masterplan_Material_Export&quot; in the 150% demo workflows</td>
</tr>
</tbody>
</table>

The dictionary key `MaterialList:dicts` in `MESInputDatObject` has a list with only a single dictionary `ItemDict`. The `ItemDict` contains fields mapped using preferences and calling `::T4X::TC::MAPPING::FieldMapping`.

In case of PartRevision all of the Part Logistics Form attributes such as Lot or Serialized are mapped to the corresponding attributes in Opcenter EX DS Material. These attributes permit to create instances of the Material in Opcenter EX DS having characteristics like a Opcenter EX DS Material Tracking Unit (batchId or serial number).
Maintain Opcenter EX DS Material Fields

The maintained fields are the keys of the ItemDict. Go to the ::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL::TC_Object2MES_Object procedure in <GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_material_mapping.sd to get more details. Note that the same procedure populates the DocumentList:dicts used to create Documents and DocumentAssociationList:dicts used to associate the given DocumentList:dicts to the Material in Opcenter EX DS.

Create and Associate Document to Opcenter EX DS Material

The capability to transfer and associate a Document to Opcenter EX DS Material is based on the document linked to the ItemRevision or PartRevision. Please refer to Create and Associate Opcenter EX DS Document for details.

The performTransfer procedure for associating the document uses the PLMAssociateDocumentToMaterialDefinitionList command in Opcenter EX DS.

Opcenter EX DS Tool Definition

Create Opcenter EX DS Tool Definition

Maintain Opcenter EX DS Tool Definition Fields

Create and Associate Document to Opcenter EX DS Tool Definition

Create Opcenter EX DS Tool Definition

The capability to create a Tool Definition in Opcenter EX DS is e.g. based on Mfg0MEResourceRevision release from Teamcenter. The dictionary key ToolList:dicts has a list of single dictionaries ItemDict. Each of the ItemDict contains the fields mapping based on the T4x approach using preferences and mapping by calling ::T4X::TC::MAPPING::FieldMapping. Please also refer to Mapping Templates and Preferences for more details about how to configure preferences in Teamcenter and mapping in T4x.

The actual transfer is implemented in the procedure ::FN4T::GENOBJ::MAPPING::TOOL::performTransfer which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Tool</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>Mfg0MEResourceRevision</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>PLMCreateToolDefinitionList</td>
</tr>
<tr>
<td><strong>Export</strong></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Mapping template</td>
<td><code>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_tool_mapping.sd</code></td>
</tr>
<tr>
<td>Custom namespace</td>
<td><code>::FN4T::GENOBJ::CUSTOM::MAPPING::TOOL</code></td>
</tr>
<tr>
<td>Preferences start with</td>
<td><code>FN4T_Tool</code></td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td><code>::FN4T::GENOBJ::MAPPING::TOOL::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;FN4T_Masterplan_Tool_Export&quot; in the 150% demo workflows</td>
</tr>
</tbody>
</table>

**Maintain Opcenter EX DS Tool Definition Fields**

The maintained fields are the keys of the `ItemDict`. Go to the `::FN4T::GENOBJ::CUSTOM::MAPPING::TOOL::TC_Object2MES_Object` procedure in `<GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_tool_mapping.sd` to get more details. Note that the same procedure populates the `DocumentList:dicts` used to create Documents and `DocumentAssociationList:dicts` used to associate the given `DocumentList:dicts` to the Tool Definition in Opcenter EX DS.

**Create and Associate Document to Opcenter EX DS Tool Definition**

The capability to transfer and associate Document to Opcenter EX DS Tool Definition is based on the document linked to the corresponding object ItemRevision or PartRevision. Please refer to Create and Associate Opcenter EX DS Document for details. The specific Opcenter EX DS command used to associate Documents to Materials is "PLMAssociateDocumentToToolDefinitionList".

**Opcenter EX DS Process**

- **Create Opcenter EX DS Process**
- **Maintain Opcenter EX DS Process**

**Create Opcenter EX DS Process**

The capability to create a Process in the catalogue of Opcenter EX DS is based on MEPProcessRevision, Mfg0MEPlantBOPRevision or Mfg0MEProcLineRevision release from Teamcenter.

The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::PROCESS::performTransfer` which is not part of the custom mapping but can be overwritten, if needed.
<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TargetTypeName</strong></td>
</tr>
<tr>
<td><strong>Teamcenter Types</strong></td>
</tr>
<tr>
<td><strong>Target system</strong></td>
</tr>
<tr>
<td><strong>Used API</strong></td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
</tr>
<tr>
<td><strong>Custom namespace</strong></td>
</tr>
<tr>
<td><strong>Preferences start with</strong></td>
</tr>
<tr>
<td><strong>Dictionaries expected by</strong></td>
</tr>
<tr>
<td><strong>Sample workflow task in demo template</strong></td>
</tr>
</tbody>
</table>

The dictionary key `ProcessList:dicts` has a list of single dictionaries `ItemDict`. Each of the `ItemDict` contains the fields mapping based on the T4x approach using preferences and mapping by calling `::T4X::TC::MAPPING::FieldMapping`. Please also refer to Mapping Templates and Preferences for more details about how to configure preferences in Teamcenter and mapping in T4x.

Please note that by default, function `::FN4T::GENOBJ::CUSTOM::MAPPING::PROCESS::validateProcess` will verify the attachments of the process using the variable `::FN4T::CONFIGURATION::PlantList` which defines all mandatory attachments of a process revision. The procedure will fail the transfer if any of the attachments is missing. The task "FN4T_Masterplan_Process_Export" should be placed after the material transfer because otherwise Opcenter EX DS will complain about missing objects when the transfer tries to create the associations.

**Maintain Opcenter EX DS Process**

The maintained fields are the keys of the `ItemDict`. You can configure the fields in the `::FN4T::GENOBJ::CUSTOM::MAPPING::PROCESS::TC_Object2MES_Object` procedure in `<GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_process_mapping.sd`.

**Opcenter EX DS Operation**

Create Opcenter EX DS Operation

Maintain Opcenter EX DS Operation Fields
Create and Associate Document to Opcenter EX DS Operation

Associate Skill to Operation in Opcenter EX DS

Create Opcenter EX DS Operation

The capability to create an Operation in Opcenter EX DS is based on the MEOPRevision in the case of classic BOP structure, Mfg0MECompOPRevision in the case of compound BOP structure or Mfg0MEProcStatnRevision in the case plant BOP release from Teamcenter. The actual transfer is implemented in the procedure \::FN4T::GENOBJ::MAPPING::OPERATION::performTransfer (or the identical \::FN4T::GENOBJ::MAPPING::PLANTBOPOPERATION::performTransfer for plant BOP) which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
<th>Operation, PlantBOPOperation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Operation, PlantBOPOperation</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>MEOPRevision, Mfg0MECompOPRevision, Mfg0MEProcStatnRevision</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>PLMCreateOperationList</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/\fn4t_mapping_config/\fn4t_operation_mapping.sd</td>
</tr>
<tr>
<td>Custom namespace</td>
<td>::FN4T::GENOBJ::CUSTOM::MAPPING::OPERATION, ::FN4T::GENOBJ::CUSTOM::MAPPING::PLANTBOPOPERATION</td>
</tr>
<tr>
<td>Preferences start with</td>
<td>FN4T_Operation, FN4T_PlantBOPOperation</td>
</tr>
<tr>
<td>Dictionaries expected by ::FN4T::GENOBJ::MAPPING::OPERATION::performTransfer</td>
<td>::FN4T::GENOBJ::CUSTOM::MAPPING::PROCESS::MESISInputDatObject</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;FN4T_Masterplan_Operation_Export&quot; in the 150% demo workflows</td>
</tr>
</tbody>
</table>

The dictionary key OperationList\:d\_icts has a list of single dictionaries ItemDict. Each of the ItemDict contains the fields mapping based on the T4x approach using preferences and mapping by calling \::T4X::TC::MAPPING::FieldMapping. Please also refer to Mapping Templates and Preferences for more details about how to configure preferences in Teamcenter and mapping in T4x. The namespace \::FN4T::GENOBJ::CUSTOM::MAPPING::PLANTBOPOPERATION simply reuses all the implemented procedures of the namespace \::FN4T::GENOBJ::CUSTOM::MAPPING::OPERATION. So the implementations are identical, only the preferences differ.
Maintain Opcenter EX DS Operation Fields

The maintained fields are the keys of the ItemDict. To find this dictionary follow the :FN4T::GENOBJ::CUSTOM::MAPPING::OPERATION::TC_Object2MES_Object procedure in the <GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_operation_mapping.sd.

Create and Associate Document to Opcenter EX DS Operation

The capability to transfer and associate Document to Opcenter EX DS Operation is based on the document linked to the corresponding object, please refer to Create Opcenter EX DS Operation for supported Teamcenter objects. Read about Create and Associate Opcenter EX DS Document to learn how the generic configuration for the Document Opcenter EX DS transfer looks like. The specific Opcenter EX DS command used to associate Documents to Materials is "PLMAssociateDocumentToOperationList".

Associate Skill to Operation in Opcenter EX DS

The capability associate Skill to Operation in Opcenter EX DS is based on the MOMSkill linked to the corresponding object, please refer to Opcenter EX DS Skill for more details about Skill transfer. The procedure :FN4T::GENOBJ::CUSTOM::MAPPING::OPERATION::TC_Object2MES_Object has to fill the TCL dict :FN4T::GENOBJ::CUSTOM::MAPPING::OPERATION::MESInputDatSkillAssociations, e.g. by calling the procedure populateMESInputDatSkillAssociations in the same namespace.

The actual transfer is implemented in the procedure :FN4T::GENOBJ::MAPPING::OPERATION::performTransfer ( :FN4T::GENOBJ::MAPPING::PLANTBOperation::performTransfer for plant BOP) which is not part of the custom mapping but can be overwritten, if needed. This procedure ensures that the associated Skills have been transferred to Opcenter EX DS before.

Later if Opcenter EX DS creates WorkOrderOperation from the Operation the associated Skills will be taken as well.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Dictionaries expected</td>
</tr>
<tr>
<td>by :FN4T::GENOBJ::MAPPING::OPERATION::performTransfer</td>
</tr>
</tbody>
</table>
Opcenter EX DS Step

Create Opcenter EX DS Step

Maintain Opcenter EX DS Step Fields

Create and Associate Document to Opcenter EX DS Step

Create Opcenter EX DS Step

The capability to create a Step in Opcenter EX DS is based on Mfg0MEStepRevision in the case of compound BOP structure, MEOPRevision in case of plant BOP or MEActivity in case of classic BOP structure release from Teamcenter.

The actual transfer is implemented in the procedure ::FN4T::GENOBJ::MAPPING::STEP::performTransfer (or the identical ::FN4T::GENOBJ::MAPPING::PLANTBOPSTEP::performTransfer for plant BOP) which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>and ::FN4T::GENOBJ::MAPPING::PLANTBOPPERATION::performTransfer</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>

Opcenter EX DS Step

Create Opcenter EX DS Step

Maintain Opcenter EX DS Step Fields

Create and Associate Document to Opcenter EX DS Step

Create Opcenter EX DS Step

The capability to create a Step in Opcenter EX DS is based on Mfg0MEStepRevision in the case of compound BOP structure, MEOPRevision in case of plant BOP or MEActivity in case of classic BOP structure release from Teamcenter.

The actual transfer is implemented in the procedure ::FN4T::GENOBJ::MAPPING::STEP::performTransfer (or the identical ::FN4T::GENOBJ::MAPPING::PLANTBOPSTEP::performTransfer for plant BOP) which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>and ::FN4T::GENOBJ::MAPPING::PLANTBOPPERATION::performTransfer</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>

Opcenter EX DS Step

Create Opcenter EX DS Step

Maintain Opcenter EX DS Step Fields

Create and Associate Document to Opcenter EX DS Step

Create Opcenter EX DS Step

The capability to create a Step in Opcenter EX DS is based on Mfg0MEStepRevision in the case of compound BOP structure, MEOPRevision in case of plant BOP or MEActivity in case of classic BOP structure release from Teamcenter.

The actual transfer is implemented in the procedure ::FN4T::GENOBJ::MAPPING::STEP::performTransfer (or the identical ::FN4T::GENOBJ::MAPPING::PLANTBOPSTEP::performTransfer for plant BOP) which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>and ::FN4T::GENOBJ::MAPPING::PLANTBOPPERATION::performTransfer</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>
Export

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>ormTransfer and ::FN4T::GENOBJ::MAPPING::PLANTBOPS TEP::performTransfer</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
<tr>
<td>&quot;FN4T_Masterplan_Step_Export&quot; in the 150% demo workflows</td>
</tr>
</tbody>
</table>

<GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_step_mapping.sd operates as a wrapper and handles both TargetTypeNames "Step" and "PlantBOPStep". The namespace ::FN4T::GENOBJ::CUSTOM::MAPPING::PLANTBOPSTEP simply reuses all the implemented procedures of the namespace ::FN4T::GENOBJ::CUSTOM::MAPPING::STEP. So the implementations are identical, only the preferences differ.

The dictionary key StepList:dicts has a list of single dictionaries ItemDict. Each of the ItemDict contains the fields mapping based on the T4x approach using preferences and mapping by calling ::T4X::TC::MAPPING::FieldMapping. Please also refer to Mapping Templates and Preferences for more details about how to configure preferences in Teamcenter and mapping in T4x.

Note: There is no need to set the preferences for MEActivity in context of the "Step" transfer, because Activities are handled implicitly by the "Operation" transfer.

**Maintain Opcenter EX DS Step Fields**

The maintained fields are the keys of the ItemDict in TCL dict ::FN4T::GENOBJ::CUSTOM::MAPPING::STEP::MESInputDatObject. To find this dictionary follow the ::FN4T::GENOBJ::CUSTOM::MAPPING::STEP::TC_Object2MES_Object procedure in the <GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_step_mapping.sd.

**Create and Associate Document to Opcenter EX DS Step**

The capability to transfer and associate Document to Opcenter EX DS Step is based on the document linked to the corresponding object, please refer to Create Opcenter EX DS Step for supported Teamcenter objects. Read about Create and Associate Opcenter EX DS Document to learn how the generic configuration for the Document Opcenter EX DS transfer looks like. The specific Opcenter EX DS command used to associate Documents to Materials is "PLMAssociateDocumentToStepList".

**Caution:**

The capability to transfer and associate Document to Opcenter EX DS Step for MEActivity as a corresponding is not supported.

**Opcenter EX DS Data Collection Definition**

By default, if an object corresponding to an Operation or a Step has a Data Collection Definition (DCD) in Teamcenter, FN4T creates and associates a DCD in Opcenter EX DS.
The actual transfer is implemented in the procedure `::FN4T::GENBOM::MAPPING::DCD::performTransfer` (for plant BOP) which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TargetTypeName</strong></td>
</tr>
<tr>
<td><strong>Teamcenter Types</strong></td>
</tr>
<tr>
<td><strong>Target system</strong></td>
</tr>
<tr>
<td><strong>Used API</strong></td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
</tr>
<tr>
<td><strong>Custom namespace</strong></td>
</tr>
<tr>
<td><strong>Preferences start with</strong></td>
</tr>
<tr>
<td><strong>Dictionaries expected by</strong></td>
</tr>
<tr>
<td><strong>Sample workflow task in demo template</strong></td>
</tr>
</tbody>
</table>

The TCL namespace variable `headerIsOperation` has to be set by the mapping procedure `TC_Object2MES_Object` in the same namespace. This variable determines the Opcenter EX DS command to use later in the `performTransfer` procedure.

The generic workflow handler for both TargetTypeNames uses the workflow argument – `useView4Transfer=TRUE`.

Please note the DCD supported types are configured in the `::FN4T::CONFIGURATION::DCDList` in `<GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_custom_parameter.sd`. These parameters are used...
for the data extraction from Teamcenter. To add a new type, you need to add a new dict into the ::FN4T::CONFIGURATION::DCDList. Please do not forget to extend the preferences after all.

Caution:

In the demo configuration, the second level objects (operations) of the plant BOP structure do not support DCD attachments, because the used Teamcenter type Mfg0MProcStatnRevision does not support DCD attachments. This is why there is no DCD transfer for the plant BOP Operation.

Opcenter EX DS Skill

The demo scenario uses custom object T4CLSkill (display name: MOMSkill) as a corresponding object for the Opcenter EX DS Skill transfer. FN4T creates Skill and later associates it to the Operation during Operation transfer in the Opcenter EX DS. Note that the workflow task for Skill transfer should be one of the first tasks to be executed because the later operation transfer will try to reference existing skills.

The actual transfer is implemented in the procedure ::FN4T::GENOBJ::MAPPING::SKILL::performTransfer which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Skill</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>T4CLSkill</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>CreateSkill_BOP</td>
</tr>
<tr>
<td></td>
<td>ODATA Query: /Skill $filter=NId=&lt;SkillName&gt;</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_skill_mapping.sd</td>
</tr>
<tr>
<td>Custom namespace</td>
<td>::FN4T::GENOBJ::CUSTOM::MAPPING::SKILL</td>
</tr>
<tr>
<td>Preferences start with</td>
<td>FN4T_Skill</td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td>::FN4T::GENOBJ::MAPPING::SKILL::performTransfer</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;FN4T_Masterplan_Skill&quot; in the 150% demo workflows</td>
</tr>
</tbody>
</table>

Note that FN4T calls a generic Opcenter EX DS command in contrast to the PLM commands used for other transfers. ::FN4T::TOOLBOX::searchInSITQuery will be executed to determine whether the given skill already exists in Opcenter EX DS and skips the actual transfer if so.
Opcenter EX DS Inspection Definition
Defect Group and Defect Type

Create and Update Defect Group

Create and Update Defect Type

Create and Update Defect Group

The capability to create and update an Defect Group in Opcenter EX DS is based on an Failure Specification in Teamcenter Active Workspace. Defect Group in Opcenter EX DS does not have revisions. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::DEFECT::performTransfer`. Please note that FN4T creates a virtual root parent Defect Group by transferring root Failure Specification(s) from Teamcenter. The same Failure Specification becomes a Defect Group and a Defect Type if it has at least one child. Otherwise FN4T creates a Defect Group only.

Currently the Failure Specification can not be submitted to workflow in Teamcenter Active Workspace Client. Therefore, you should use the Teamcenter RAC to trigger transfer.

Please use the "Failure Specification..." search query in Teamcenter RAC and search for all (leave the search field "Name" empty) and select the option "true" for the search field "LATEST". Note that the Defect Group in Opcenter EX DS has a list of parents and children. So if you select a child element in Teamcenter and transfer it, only the children list of the object list become updated but not the children list of the parent object and not the parents list of the children.

<table>
<thead>
<tr>
<th>Export</th>
<th>Defect Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Defect Group</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>Failure Specification Version</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>CreateDefectGroup or</td>
</tr>
<tr>
<td></td>
<td>UpdateDefectGroup</td>
</tr>
<tr>
<td>Mapping template</td>
<td>`&lt;GS_ROOT&gt;/var/mmap/</td>
</tr>
<tr>
<td></td>
<td>fn4t_mapping_config/</td>
</tr>
<tr>
<td></td>
<td>fn4t_defect_mapping.sd`</td>
</tr>
<tr>
<td>Custom namespace</td>
<td><code>::FN4T::GENOBJ::CUSTOM::</code></td>
</tr>
<tr>
<td></td>
<td><code>MAPPING::DEFECT</code></td>
</tr>
<tr>
<td>Preferences start with</td>
<td><code>FN4T_Defect</code></td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td><code>::FN4T::GENOBJ::MAPPING::DEFECT::performTransfer</code></td>
</tr>
<tr>
<td></td>
<td><code>::FN4T::GENOBJ::CUSTOM::</code></td>
</tr>
<tr>
<td></td>
<td><code>MAPPING::DEFECT::MESIn</code></td>
</tr>
<tr>
<td></td>
<td><code>putDatDefectGroup</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;FN4T_Defect_Transfer&quot; triggers object transfer</td>
</tr>
</tbody>
</table>
If the dictionary key `Id:string` exists
in `::FN4T::GENOBJ::CUSTOM::MAPPING::DEFECT::MESInputDatDefectGroup` FN4T updates the existing Failure Specification using `UpdateDefectGroup` command.

### Create and Update Defect Type

The capability to create and update an Defect Type in Opcenter EX DS is based on an Failure Specification in Teamcenter Active Workspace. Defect Type in Opcenter EX DS does not have revisions. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::DEFECT::performTransfer`. The same Failure Specification becomes a Defect Group and a Defect Type if it has at least one child. Otherwise FN4T creates a Defect Group only.

Please use the "Failure Specification..." search query in Teamcenter RAC and search for all (leave the search field "Name" empty) and select option "true" for the search field "LATEST".

<table>
<thead>
<tr>
<th>Export</th>
<th>&quot;FN4T_Defect_Structure&quot; triggers structure transfer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Export</th>
<th>Defect Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Defect Type</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>Failure Specification Version</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>CreateDefectType or UpdateDefectType</td>
</tr>
<tr>
<td>Mapping template</td>
<td><code>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_defect_mapping.sd</code></td>
</tr>
<tr>
<td>Custom namespace</td>
<td><code>::FN4T::GENOBJ::CUSTOM::MAPPING::DEFECT</code></td>
</tr>
<tr>
<td>Preferences start with</td>
<td><code>FN4T_Defect</code></td>
</tr>
<tr>
<td>Dictionaries expected by <code>::FN4T::GENOBJ::MAPPING::DEFECT::performTransfer</code></td>
<td><code>::FN4T::GENOBJ::CUSTOM::MAPPING::DEFECT::MESInputDatDefectType</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template &quot;FN4T_Defect_Transfer&quot;</td>
<td>&quot;FN4T_Defect_Object&quot; triggers object transfer &quot;FN4T_Defect_Structure&quot; triggers structure transfer</td>
</tr>
</tbody>
</table>

If the dictionary key `Id:string` exists
in `::FN4T::GENOBJ::CUSTOM::MAPPING::DEFECT::MESInputDatDefectType` FN4T updates the existing Failure Specification using `UpdateDefectGroup` command.
Opcenter EX DS Inspection Definition and Characteristic

Create and Update Inspection Definition

Create and Update Attributive Characteristic

Create and Update Visual Characteristic

Create and Update Variable Characteristic

Create and Update Inspection Definition

The capability to create and update an Inspection Definition in Opcenter EX DS is based on Characteristics Representation Element Revision created in Teamcenter Active Workspace. Inspection Definition in Opcenter EX DS does not have revisions. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. Please note that FN4T transfers Characteristics Representation Element and Specification linked to given Characteristics Representation Element in Teamcenter at the same time.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
</tr>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API</td>
</tr>
<tr>
<td>Mapping template</td>
</tr>
<tr>
<td>Custom namespace</td>
</tr>
<tr>
<td>Preferences start with</td>
</tr>
<tr>
<td>Dictionaries expected by</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>

If the dictionary key **Id:string** exists in `::FN4T::GENOBJ::CUSTOM::MAPPING::INSPECTION::MESInputDatInspection` the
Create and Update Variable Characteristic

The capability to create and update an Variable Characteristic in Opcenter EX DS is based on an Variable Specification Version attached to a Characteristics Representation Element Revision in Teamcenter Active Workspace. Variable Characteristic in Opcenter EX DS have revisions. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. Please note that FN4T transfers Characteristics Representation Element and Specification linked to given Characteristics Representation Element in Teamcenter at the same time.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
</tr>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API</td>
</tr>
<tr>
<td>Mapping template</td>
</tr>
<tr>
<td>Custom namespace</td>
</tr>
<tr>
<td>Preferences start with</td>
</tr>
<tr>
<td>Dictionaries expected by <code>::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>

If the dictionary key `Id:string` exists in `::FN4T::GENOBJ::CUSTOM::MAPPING::INSPECTION::MESInputDatVariableCharacteristic`, FN4T updates the existing Variable Characteristic using `UpdateVariableCharacteristicSpecification` command.
Create and Update Visual Characteristic

The capability to create and update an Visual Characteristic in Opcenter EX DS is based on an Visual Specification Version attached to a Characteristics Representation Element Revision in Teamcenter Active Workspace. Visual Characteristic in Opcenter EX DS have revisions. The actual transfer is implemented in the procedure ::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer which is not part of the custom mapping but can be overwritten, if needed. Please note that FN4T transfers Characteristics Representation Element and Specification linked to given Characteristics Representation Element in Teamcenter at the same time.

Currently, FN4T supports only meta data transfer, no file upload and association to Visual Characteristic is possible.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Visual Characteristic</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>Characteristics Representation Element Revision</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>CreateVisualCharacteristicSpecification or</td>
</tr>
<tr>
<td></td>
<td>UpdateVisualCharacteristicSpecification</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/ fn4t_mapping_config/ fn4t_inspection_mapping.sd</td>
</tr>
<tr>
<td>Custom namespace</td>
<td>::FN4T::GENOBJ::CUSTOM: :MAPPING::INSPECTION</td>
</tr>
<tr>
<td>Preferences start with</td>
<td>FN4T_Inspection</td>
</tr>
<tr>
<td>Dictionaries expected</td>
<td>::FN4T::GENOBJ::CUSTOM: :MAPPING::INSPECTION: MESInputDatVisualCharacteristic</td>
</tr>
<tr>
<td>by ::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer</td>
<td></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;FN4T_Inspection&quot; in the 150% demo workflows</td>
</tr>
</tbody>
</table>

If the dictionary key Id:string exists in ::FN4T::GENOBJ::CUSTOM: :MAPPING::INSPECTION::MESInputDatVisualCharacteristic FN4T updates the existing Visual Characteristic using UpdateVariableCharacteristicSpecification command.

Create and Update Attributive Characteristic

The capability to create and update an Attributive Characteristic in Opcenter EX DS is based on an Attributive Specification Version attached to a Characteristics Representation Element Revision in
Teamcenter Active Workspace. Attributive Characteristic in Opcenter EX DS have revisions. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer` which is not part of the custom mapping but can be overwritten, if needed. Please note that FN4T transfers Characteristics Representation Element and Specification linked to given Characteristics Representation Element in Teamcenter at the same time.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
</tr>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API</td>
</tr>
<tr>
<td>Mapping template</td>
</tr>
<tr>
<td>Custom namespace</td>
</tr>
<tr>
<td>Preferences start with</td>
</tr>
<tr>
<td>Dictionaries expected by <code>::FN4T::GENOBJ::MAPPING::INSPECTION::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>

If the dictionary key `Id:string` exists in `::FN4T::GENOBJ::CUSTOM::MAPPING::INSPECTION::MESInputDatAttributiveCharacteristic` FN4T updates the existing Attributive Characteristic using `UpdateVariableCharacteristicSpecification` command.

**Create and Associate Opcenter EX DS Document**

By default, if a target object of a transfer has a document in Teamcenter, FN4T creates and associates the Document in Opcenter EX DS. This is done by populating the dictionary `MESInputDatDocuments` with a key `DocumentList:dicts` and `DocumentList` as value and another dict `MESInputDatDocumentAssociations` with a key `DocumentAssociationList:dicts` and `DocumentAssociationsList`. Both dicts reside in the custom namespace of the actual `TargetTypeName`. Please note that `DataSetType`, `MIMEType`, `Reference` and Opcenter EX DS type are configured centrally in the `::FN4T::CONFIGURATION::FileHandlingDatasetMap` in the `<GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_custom_parameter.sd`. These parameters are used.
for the data extraction in Teamcenter. To extract a new data type from Teamcenter, you need to add a new key named exactly as the object_type, e.g. for a .pdf file it must be PDF or for .docx - MSWordX. Please do not forget to extend the preferences afterward.

To populate the dictionaries the procedure `::FN4T::TOOLBOX::populateDocumentAndDocumentAssociationDict` can be used in the mapping procedure. The `performTransfer` procedure of the used TargetTypeName always uses the same command `PLMCreateDocumentList` for the document transfer but different command for associating the documents.

<table>
<thead>
<tr>
<th>Target System</th>
<th>Used API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcenter EX DS</td>
<td>PLMCreateDocumentList, PLMAssociateDocumentToMaterialDefinitionList</td>
</tr>
<tr>
<td></td>
<td>PLMAssociateDocumentToOperationList, PLMAssociateDocumentToStepList</td>
</tr>
<tr>
<td></td>
<td>PLMAssociateDocumentToToolDefinitionList</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
</tr>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Mapping template</td>
</tr>
<tr>
<td>Custom namespace</td>
</tr>
<tr>
<td>Preferences start with</td>
</tr>
<tr>
<td>Dictionaries expected by <code>::FN4T::GENOBJ::*:MATERIAL::performTransfer</code></td>
</tr>
<tr>
<td><strong>Export</strong></td>
</tr>
<tr>
<td>------------------------------------</td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
</tr>
</tbody>
</table>
8. 100 Percent Components

Prepare 100 Percent BOM and BOP

Note:
This capability is *not* primarily related to the integration or to Closed Loop Manufacturing as it does not include any communication with an external system, it is just a Teamcenter-internal functionality. Manual creation of 100% CCObjects is possible and valid, but error prone. Therefore the demo scenario provides a helper workflow and a web service to automate the creation of the CCObject. However the creation can be replaced by any other process as needed without influencing the overall CLM scenario.

Teamcenter Workflow to Create CCObject

The capability to create the 100% CCObject that represents the Opcenter EX DS AsPlannedBoP is supported by a Teamcenter workflow. The demo scenario covers two possibilities to configure 100 percent structures by applying:

- revision rule using unit effectivity and/or
- variant rule using variant conditions.

Optionally, before you start the workflow, copy the Item or the ItemRevision of the BOM header and, if applicable a variant rule to the Teamcenter clipboard. The Item or ItemRevision will become the end item for the configuration and the variant rule will be used to configure options and variants. You may also add the end item later when the workflow is in your Teamcenter Inbox.

Select the top level process revision of your BOP for example the MEProcessRevision in Manufacturing Process Planner (MPP) and start the Teamcenter workflow dialog (<Ctrl><P> or "File/New/Workflow Process...`). Your process revision will appear in the "Targets" folder (see "2") of the dialog. Select the workflow template `CLM_CreateCC_4_100PerCent_Structures` (see "1" in the below image). In the next step paste the end item (or revision) from the Teamcenter clipboard into References folder ("3").
If you use classic variant rule to configure structures, add a variant rule in addition to the "References" folder as shown in following figure.

Now start the workflow by clicking "OK". As a result, you will find the Do task in My Worklist to perform. If you select the task, you will see a form in the "Viewer". In the next step you have to fill out the form as follows:
Please note that the value of the field "BOM Alternate" (field "1") will be used to create BOM Alternative, Routing Group Counter and Product Version in SAP as well. Furthermore, it will be used as the name of the Opcenter EX DS AsPlannedBoP. The field Sample Unit ("3") is used to configure the Teamcenter revision rule in the CCObject only and has no corresponding field in SAP or Opcenter. The Unit Range ("4") will not be used for configuration in Teamcenter but will be stored as a string in the Production Version description in SAP.

The same workflow can be used to create the CCObject using classic options and variants. The form still needs the BOM Alternate, CC Object Name and Unit Range as input. In addition to the end item the variant rule has to be attached as reference to the workflow.

Note that it is sufficient to add the revision rule and the end item (or revision) in this step instead of the workflow start dialog.

Finally check the "Complete" radio button ("5") and click "Save" (6”). This will create the CCObject with the name as provided in field "2".
FN4T script to create an Work Order Header in Opcenter EX DS

The FN4T demo scenario provides the script `<GS_ROOT>/var/test/tools/fn4t_create_work_order_header.tcl` to emulate the ERP order creation based on Teamcenter identifiers. This script essentially offers a UI to enter several order data and then calls the Opcenter EX DS web service to create the order. Opcenter EX DS will then request the 100% skeleton from FN4T. Of course all the 150% data have to be present in Opcenter EX DS and the 100% CC object must be present in Teamcenter for this to work. The script UI directly resembles the input parameters of the Opcenter EX DS service:

Enter one of the configured Opcenter connections in the field "Name of the FN4T connection to Opcenter". Choose "false" for the field "Encode value in HEX16" if you want to enter the Teamcenter UIDs in the format commonly used in Teamcenter Gateway products (tag format). Choose "true" if you want to enter the UIDs in the format displayed by Teamcenter, e.g. in the "Print Object" view or when you copy an object to the operating system clipboard. If you click on the "Run Script" button (green triangle in upper right corner of the input area), FN4T will issue the service call and print the result. In case everything went okay, you'll see a similar output (wrapped for readability):

```
START CONSOLE BUFFER ID: 157426506132924
Opcenter Execution Discrete Manufacturing connection status:
LOGON_OK_PARTIAL
Logon successful.

Transfer parameters:
NId:string --> 123456
Name:string --> MyOrder
Plant:string --> Factory1
RoutingID:string --> 566756783236515A345862695F41
PBOPIdentID:string --> 41426178474C5936345862695F41
TargetProductID:string --> 314164783236515A345862695F41
```
The script outputs all parameters as they are transferred. The line “Service call status: OK” indicates success. In any error case you'll also see some diagnostic message, but you may need to analyze the log files in addition.

Note that the script code may also serve as a blueprint to build a non-SAP ERP integration to Opcenter for order creation e.g. using Teamcenter Gateway for Enterprise Applications.

### Web Service to Create CCOBJECT and Trigger Download 100% Skeleton

As a alternative to the workflow and the order creation script a specific FN4T web service can be called to create a CCOBJECT and trigger the download of the 100% skeleton. This may be helpful especially in the FN4T stand-alone use case, i.e. when T4S, FN4S and SAP are not available and an ERP system has to be emulated.

**Caution:**

This web service is delivered on an as-is basis. It may change in the future, breaking compatibility with this version and in its current version it has some limitations: The JSON payload contains a string attribute with an embedded JSON that has to be escaped in a JSON-conform way. The web service is not capable to attach a Teamcenter variant rule, so no options and variants can be used. This FN4T web service as such does not create any order- or AsPlannedBoP-related objects in Opcenter EX DS!
The endpoint has to be configured in Opcenter EX DS (or any other system) as follows:

http://{{FN4TGSInstance}}:11301/pxml/updateWorkOrderCC

Request-Example:

```json
{
    "OrderID": "OrderForMultiSITTest",
    "RoutingUID": "68665674374D74306F24244E4644",
    "TargetProductUID": "7865174374D74306F24244E4644",
    "Enterprise": "Factory1",
    "ConfigurationParameter": "{"EffectivityUnit": "1", "EffectivityParams": [{"Name": "ID", "Value": "50076677"}]}
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderID</td>
<td>String</td>
<td>CCOBJECT name</td>
</tr>
<tr>
<td>RoutingUID</td>
<td>String</td>
<td>BOP header (process) tag</td>
</tr>
<tr>
<td>TargetProductUID</td>
<td>String</td>
<td>BOM header tag</td>
</tr>
<tr>
<td>Enterprise</td>
<td>String</td>
<td>Enterprise from Opcenter EX DS</td>
</tr>
<tr>
<td>ConfigurationParameter</td>
<td>String</td>
<td>Parameters in JSON (escaped string)</td>
</tr>
</tbody>
</table>

Success-Response:

```json
HTTP/1.1 200 OK
{
    "status": "OK",
    "message": "Import job (ID 596d2a2f5bed28f3926261e8) was created"
}
```

Error-Response:

```json
HTTP/1.1 500
{
    "status": "ERROR",
    "message": "The job cannot be created."
}
```

**Transfer 100 Percent BOM and BOP to SAP**

The capability to transfer the 100% structures to SAP is supported by the `CLM_Release_100PerCent_Structures_SAPOnly` workflow. The workflow uses standard T4S functionality only and is therefore not described in detail here.
First, search and identify the CCObject containing the 100% configuration in Teamcenter. The demo template contains the saved query `FN4TFindCCObjectByName`, so that you can easily find the 100% CCObject (e.g., the one created in the previous step).

At the end of the transfer, FN4T will store the mapping between SAP and Teamcenter by using the SAP concatenated object's IDs as a key and the Teamcenter CCObject UID hex encoded as a value for the current transfer.

```plaintext
set IDMappingKey [::FN4S::IDMAPPING::buildConfigurationIDKey $Info(InternalMaterialNumber) $Info(PlantId) $GroupId $GroupCounter $Info(ProductionVersion)]
set IDMappingValue [tpco_scanHEX16 $Info(CCObjectTag)]
```

This is necessary to provide the information originating from Teamcenter for later processing it during order download and confirmation, as implemented e.g. in the Opcenter Connect FN for SAP S/4HANA.

## Transfer 100 Percent BOP to Opcenter EX DS

The transfer of the 100 Percent BOP to Opcenter EX DS usually gets triggered by Opcenter EX DS by calling the web API published on the FN4T GS instance. Please note that event subscription should be configured in Opcenter EX DS additionally. The endpoint has to be configured in Opcenter EX DS configuration keys in the section "FN4T Integration" as follows:

```plaintext
http://{{FN4TGSInstance}}:11301/pxml/triggerSkeletonTransfer
```

**Request-Example:**

```plaintext
{
    "PBOPIdentID": "7776274655057304a4569394d42",
    "Enterprise": "Factory1"
}
```

**Field | Type | Description**

| PBOPIdentID | String | CCObject unique Id from Teamcenter |
| Enterprise  | String  | Enterprise from Opcenter EX DS |

**Success-Response:**

```
HTTP/1.1 200 OK
{
    "JobID": "7776274655057304a4569394d4241414141414141414141414141414141",
    "status": "OK",
    "message": "Skeleton export workflow created"
}
```

**Error-Response:**
HTTP/1.1 500
{
    "status": "ERROR",
    "message": "Skeleton export workflow creation failed!"
}

Please read more about the Opcenter EX DS AsPlannedBoP.

The transfer can also be triggered by manually starting the workflow "CLM_Release_100PerCent_Structures_SITOnly" on a configured 100% CCObject as created using the procedure described in Prepare 100 Percent BOM and BOP.

**Opcenter EX DS AsPlannedBoP**

The Opcenter EX DS AsPlannedBoP is represented in Teamcenter as a configured structure saved as CCObject. It also contains information about the dependencies between Opcenter EX DS operations and is an essential input for the creation of a Work Order. The Work Order Header based on the production order from the ERP system will be merged with the process and operations to from the Work Order. The Work Order in turn references the Opcenter EX DS AsPlannedBoP, which will be validated.

If the AsPlannedBoP already exists in Opcenter EX DS and is up-to-date, the Work Order can be executed as is and is visible for the operator. Otherwise Opcenter EX DS automatically triggers an event to download the AsPlannedBoP from Teamcenter (see Transfer 100 Percent BOP).

Creating the Opcenter EX DS AsPlannedBoP requires a sequence of API calls that link and associate the previously created catalogue (150%) objects. Finally, if everything worked well, FN4T issues a "PLMSubmit" command. If any error occurred (either on FN4T side or reported as a result of a web service call to Opcenter EX DS), FN4T will try to issue a "PLMRollback" command to revert all changes.

This mapping uses PLMXML to extract the necessary information in procedure PLMXML_Data2EA_Object and transforms it into a set of dictionaries in the same namespace which will then be send to Opcenter EX DS commands in ::FN4T::PLMXML::OBJECT::MAPPING::ROUTING::performEATransfer. This mapping is usually called from the workflow CLM_Release_100PerCent_Structures_SITOnly after having created a Teamcenter AI object containing the PLMXML file content. Since this transfer is based on PLMXML, no preferences are necessary. For the data in the demo scenario, the custom transfer mode "FN4T_WO_EXPORT" is necessary (see workflow task "Create AI Object" in the 100% workflows targeting Opcenter). For productive use, a tailored and optimized transfer mode should be used instead.

The actual transfer is implemented in the procedure ::FN4T::PLMXML::OBJECT::MAPPING::ROUTING::performEATransfer which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
</tr>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Export</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td><strong>Target system</strong></td>
</tr>
<tr>
<td><strong>Used API</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
</tr>
<tr>
<td><strong>Custom namespace</strong></td>
</tr>
<tr>
<td><strong>Dictionaries/Variables expected by</strong></td>
</tr>
</tbody>
</table>
Note that the implementation of the transfer is also used to implement the similar MasterPlan transfer (see chapter Master Plan). The distinction is based on the namespace-variable `::FN4T::PLMXML::CUSTOM::MAPPING::ROUTING::BoPType` which can have values "AsPlannedBoP" (this transfer, default value) or "MasterPlan" (MasterPlan transfer, see Master Plan). The variable can be set by the Teamcenter workflow argument "-BoPType" for the handler `T4X-transfer_GENERICOBJECT4TARGETTYPE`. 
9. Master Plan

Create MasterPlan

The capability to create a MasterPlan in Opcenter EX DS is based on an unconfigured BoP in Teamcenter. The task of configuring the BoP (i.e. dropping operations that are not needed for actual consumed materials) happens at a later point in time (usually at order arrival) inside Opcenter EX DS and hence is not related to FN4T. The specification of the source BoP is similar to the AsPlannedBoP (see Supported BOP Structures and Opcenter EX DS AsPlannedBoP). Indeed the same mapping and TargetTypeName handles both types of transfer since they only differ in very few details. The actual transfer is implemented in the procedure ::FN4T::PLMXML::OBJECT::MAPPING::ROUTING::performEATransfer which is not part of the custom mapping but can be overwritten, if needed. There is no update for the MasterPlan since the transfer first queries Opcenter EX DS for the given NId and deletes the object if it exists by calling "PLMRollbackMasterPlan" before transferring the new one.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>Routing</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>MECollaborationContext</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>PLMCreateMasterPlan</td>
</tr>
<tr>
<td></td>
<td>PLMCreateStepMaterialSpecificationFullList</td>
</tr>
<tr>
<td></td>
<td>PLMCreateStepToolSpecificationFullList</td>
</tr>
<tr>
<td></td>
<td>PLMLinkOperationToStepFullList</td>
</tr>
<tr>
<td></td>
<td>PLMCreateStepDependencyList</td>
</tr>
<tr>
<td></td>
<td>PLMCreateOperationEquipmentSpecLocationFullList</td>
</tr>
<tr>
<td></td>
<td>PLMCreateOperationMaterialSpecificationFullList</td>
</tr>
<tr>
<td></td>
<td>PLMCreateOperationToolSpecificationFullList</td>
</tr>
<tr>
<td></td>
<td>PLMLinkProcessToOperationFullList</td>
</tr>
<tr>
<td></td>
<td>PLMCreateOpDependencyList</td>
</tr>
<tr>
<td></td>
<td>LinkCharacteristicRepresentationToOperation</td>
</tr>
<tr>
<td></td>
<td>LinkCharacteristicRepresentationToStep</td>
</tr>
<tr>
<td><strong>Export</strong></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>PLMSubmitMasterPlan</td>
<td></td>
</tr>
<tr>
<td>PLMRollbackMasterPlan</td>
<td></td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
<td><code>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_plxml_routing_mapping.sd</code></td>
</tr>
<tr>
<td><strong>Custom namespace</strong></td>
<td><code>::FN4T::PLMXML::CUSTOM::MAPPING::ROUTING</code></td>
</tr>
<tr>
<td><strong>Dictionaries/Variables expected by<code>::FN4T::PLMXML::OBJECT::MAPPING::ROUTING::performEATransfer</code></strong></td>
<td><code>BoPType, CorrelationId, MESInputDatMasterPlan, MESInputDatOperationsEquipmentSpecLocations, MESInputDatOperationsMaterialSpecifications, MESInputDatOperationToolsSpecifications, MESInputDatOperations, MESInputDatOperationsDependency, MESInputDatOperationsInspectionDefinitions, MESInputDatStepsEquipmentSpecLocations, MESInputDatStepsMaterialSpecifications, MESInputDatStepToolSpecifications, MESInputDatSteps, MESInputDatStepsDependency, MESInputDatStepsInspectionDefinitions, all in namespace </code>::FN4T::PLMXML::CUSTOM::MAPPING::ROUTING`</td>
</tr>
<tr>
<td><strong>Sample workflow template in demo template</strong></td>
<td>&quot;CLM_MasterPlan_Transfer&quot;</td>
</tr>
</tbody>
</table>

Note that the implementation of the transfer is also used to implement the similar AsPlannedBoP transfer (see chapter [Opcenter EX DS AsPlannedBoP](#)). The distinction is based on the namespace-variable `::FN4T::PLMXML::CUSTOM::MAPPING::ROUTING::BoPType` which can have values "AsPlannedBoP" (default value) or "MasterPlan" (this transfer). The variable can be set by the Teamcenter workflow argument "-BoPType" for the handler `T4X-transfer-GenericObject4TargetType`. 
10. UADM and ERP interaction

Create WorkOrderHeader in Opcenter EX DS

Caution:

This chapter is valid only for Teamcenter, ERP and Opcenter EX DS integration scenarios. It does not apply for ERP and Opcenter EX DS only (FN4S-only) scenarios. For FN4T stand-alone scenarios, please also see chapter Prepare 100 Percent BOM and BOP.

The capability to create a WorkOrderHeader in Opcenter EX DS based on the production order from the ERP system. According to the demo scenario, the following steps are required:

• Configure T4S or T4S4 to maintain the Z-Table and implement the creation of the PROD_ORDER entry (stands for Production Order) in Z-Table. This can be done for example on release or save Production Order user exits.

• Start trigger script called "FN4S Trigger Z-Table import job (Scheduler)" with parameter "Object Type" set to either "Work Order" or "Work Order based on MasterPlan" in the FN4T Gateway Service Admin UI to check the Z-Table for the created Production Order. Learn more about the Script-based (scheduled) Triggers in the Active Integration - Generic Configuration Guide.

• Create a production order in SAP.

Caution:

To create a Production Order in SAP use only MaterialMaster and Plant transferred from Teamcenter to SAP and Opcenter EX DS and Production Version created by FN4T from Teamcenter.

For the implementation mapping details you can take a closer look at <GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_work_order_header_steps.sd. Please note that the mapping contains so called steps. The list of steps ProdOrderStepsCustom can be extended.

Moreover remember, that the enrichWithTCData step requires the data synchronization during 100 Percent transfer to SAP, in particular by invoking ::FN4S::IDMAPPING::buildConfigurationIDKey or ::FN4S::IDMAPPING::readMapping to get the linkage between Teamcenter configuration parameters and SAP.

Important: The enrichmentList must be executed before mapProdOrder2WorkOrderHeader step, i.e. prepare the data for the transferring.

And finally the execution of the callSITCreateWorkOrderHeader step to post the payload with a command named PLMCreateWorkOrderHeader.
Transfer

<table>
<thead>
<tr>
<th>Source system</th>
<th>SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used API</td>
<td>BAPI_PRODORD_GET_DETAIL</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>PLMCreateWorkOrderHeader</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/ fn4t_mapping_config/ fn4t_work_order_header_steps. sd</td>
</tr>
</tbody>
</table>

Create Confirmation in ERP

Create Confirmation in ERP is the capability to send a confirmation about completion from the Opcenter EX DS Operator Landing Page to ERP system.

Caution:
This chapter is valid only for Teamcenter, ERP and Opcenter EX DS integration scenarios. It does not apply for Teamcenter and Opcenter EX DS only integration scenarios. The following features are shipped with FN4S and not with FN4T.

By default, you can decide either to send the confirmation by each of the Work Order Operations or only by completion of the whole Work Order in Opcenter EX DS. In case, if a component in SAP has a backflush flag, FN4S does not transfer any data from Opcenter EX DS and uses only proposal data from SAP.

By default, the demo mapping fn4s_operation_conf_config.sd in prepareSAPOperationConfirmationParameters is preconfigured so that the service sends no separate good issue transaction in SAP by invoking BAPI_GOODSMVT_CREATE. It is applicable only for SAP production order consumptions without backflush, so that the goods movements transfer happens separately. In this case, you can send the serial numbers stored in the Work Order Operation instead of automatic assignment in SAP. Please uncomment the following lines to enable serial number transfer from Opcenter EX DS:

```
# dict set inputDict SAPDictGM
$EXDSMaterialTrackingValue # dict set inputDict SAPDictGM
GoodsMvt:GOODSMVT_SERIALNUMBER:UII:$index ""
```

If you use components with a serial number without backflush in SAP, you need to change the skipSeparateGoodsMovements to true, and the dictionary key ProdOrderConf:GOODSMOVEMENTS:SERIALNO_AUTO_NUMBERASSIGNMENT:$index has to be changed to "X". This means again that SAP assigns serial number automatically, regardless from inputs in Opcenter EX DS. In contrast to a serial number, by using components with a batchId FN4S can
transfer the `batchId` value from Opcenter EX DS, as shown in the following example (backflash is not set in SAP).

```python
if {$EXDSMaterialTrackingType eq "BatchId"} {
    dict set res ProdOrderConf:GOODSMOVEMENTS:BATCH:$index $EXDSMaterialTrackingValue
} else {
    #You can use auto assignment in SAP by setting to "X"
    dict set res ProdOrderConf:GOODSMOVEMENTS:SERIALNO_AUTO_NUMBERASSIGNMENT:$index ""
}
```

Note that Opcenter EX DS supports either serial number or batch number assignment, but not both at the same time.

- If the execution of the Opcenter EX DS Work Order Operation is completed, Opcenter EX DS sends a signal to trigger confirmation for the corresponding operation in the production order. Add the URL as followed (please adapt the `<GSInstance>`):
  ```
  ```

```json
Request-Example:
{
  "orderId": "000100004298", "Enterprise": "Factory1"
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orderId</td>
<td>String</td>
<td>Production order unique ID from SAP with leading zeroes will be used for SAP BAPI call.</td>
</tr>
<tr>
<td>Enterprise</td>
<td>String</td>
<td>Enterprise from Opcenter EX DS.</td>
</tr>
</tbody>
</table>

Success-Response:

```json
HTTP/1.1 200 OK
{
  "status": "QUEUED",
  "message": "9bba11525aec537240317d57"
}
```

Error-Response:

```json
HTTP/1.1 500 Internal Server Error
{
  "status": "ERROR",
  "message": "The job cannot be created"
}
```

The Web API above calls the following BAPI's:
<table>
<thead>
<tr>
<th>Target system</th>
<th>Used API</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP</td>
<td>BAPI_PRODORD_GET_DETAIL</td>
</tr>
<tr>
<td></td>
<td>BAPI_PRODORDCONF_GET_HDR_PROP</td>
</tr>
<tr>
<td></td>
<td>BAPI_PRODORDCONF_CREATE_HDR</td>
</tr>
<tr>
<td></td>
<td>BAPI_GOODSMVT_CREATE</td>
</tr>
</tbody>
</table>

OData query to retrieve the data for the given Work Order and Work Order Operation:

```
Target system | Used API |
---------------|----------|
Opcenter EX DS | WorkOrder?$expand=ProductionType($select=NId),WorkOrderOperations($filter=Sequence%20eq%20'0010')&$filter=NId%20eq%20'10001234' |
```

OData query to retrieve the ActualConsumedMaterials data for the given WorkOperationNId:

```
Target system | Used API |
---------------|----------|
Opcenter EX DS | WorkOrderOperation?$expand=ActualConsumedMaterials($expand=ToBeConsumedMaterial($select=NId),DM_MaterialTrackingUnit($expand=MaterialTrackingUnit))&$filter=NId%20eq%20'SITNId' |
```

- If the execution of the Opcenter EX DS Work Order is completed, Opcenter EX DS sends a signal to trigger confirmation for the entire production order in ERP system by calling the web API in FN4S: http://<GSInstance>:11301/pxml/createProdOrderConfAsync.

Request-Example for SAP and Opcenter EX DS integration scenarios only:

```
{
   "confirmationId": "000000014298",
   "Enterprise": "Factory1"
}
```

Request-Example for Teamcenter, SAP and Opcenter EX DS integration scenarios:

```
{
   "orderId": "100004298",
   "operationId": "SITUIdForWOP",
   "Enterprise": "Factory1"
}
```
**Field** | **Type** | **Description**
---|---|---
orderId | String | Production order unique ID from SAP.
operationId | String | Occurrence ID tag from Teamcenter hex-encoded, FN4T and FN4S integration scenario only.
confirmationId | String | Confirmation ID from SAP for operation, FN4S standalone integration scenario only.
Enterprise | String | Enterprise from Opcenter EX DS.

Success-Response:

```
HTTP/1.1 200 OK
{
  "status": "QUEUED",
  "message": "9bba11525aec537240317d67"
}
```

Error-Response:

```
HTTP/1.1 500 Internal Server Error
{
  "status": "ERROR",
  "message": "The job cannot be created"
}
```

The web API must be configured in Opcenter EX DS configuration keys in the section "FN4T Integration" accordingly.
The payload for this API consists only of orderId and Enterprise for the completed Opcenter EX DS Work Order. For the Work Order Operation operationId is the additional attribute in the payload. The data for the Work Order or Work Order Operation will be retrieved from Opcenter EX DS in the step getWorkOrderExecutionData via an OData query. For more details please take a look at `<GS_ROOT>/var/mmap/fn4s_mapping_config/fn4s_operation_conf_config.sd` for operation confirmation and `<GS_ROOT>/var/mmap/fn4s_mapping_config/fn4s_production_order_conf_config.sd` for production order.

Please pay attention that both interfaces are implemented as pipelines based on the so-called step engine framework. You can find an example in `<GS_ROOT>/var/mmap/fn4s_mapping_config/fn4s_production_order_conf_config.sd` how to add a step in the existing pipeline for the confirmation for the production order header.

If the backflash flag is enabled for the component in SAP, FN4S sends the confirmation using proposal data from SAP otherwise FN4S retrieves the ActualConsumedMaterials from Opcenter EX DS and sends it to SAP instead of proposal for values.

The Web API above calls the following BAPI's:

<table>
<thead>
<tr>
<th>Target system</th>
<th>Used API</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP</td>
<td>BAPI_PRODORD_GET_DETAIL</td>
</tr>
<tr>
<td></td>
<td>BAPI_PRODORDCONF_GET_TT_PROP</td>
</tr>
<tr>
<td>Target system</td>
<td>Used API</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>BAPI_PRODORDCONF_CREATE_TT</td>
</tr>
<tr>
<td></td>
<td>BAPI_GOODSMVT_CREATE</td>
</tr>
</tbody>
</table>

- **OData query to retrieve the ActualConsumedMaterials data for the given WorkOperationNId:**

The
11. Opcenter EX DS Non-Conformance Notification

Import Non-Conformance Notification

The capability to transfer the Opcenter EX DS Non-Conformance Notification (NCN) to Teamcenter is based on the Problem Report (PR). According to the demo scenario (closed loop 2), the operator can rise an Opcenter EX DS NCN during the execution of the operation in shop floor, might attach documents and save it. On this event and every subsequent status change, Opcenter EX DS 3.1 and newer sends the NCN to Teamcenter by posting on the preconfigured endpoint in FN4T:


Request-Example:

```
{
    "NonConformanceID": "NCN-123",
    "Enterprise": "Factory1"
}
```

Note that Opcenter EX DS 3.0 only sends a notification when change in the status to NOTIFYING-ENGINEERING-ISSUE and not any other status.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NonConformanceID</td>
<td>String</td>
<td>NCN unique Id from Opcenter EX DS.</td>
</tr>
<tr>
<td>Enterprise</td>
<td>String</td>
<td>Enterprise from Opcenter EX DS.</td>
</tr>
</tbody>
</table>

Success-Response:

HTTP/1.1 200 OK

```
{
    "JobID": "0e9bf1125dd5701d1129ab49",
    "NonConformanceID": "NCN-123",
    "status": "OK",
    "message": "Import job (ID 0e9bf1125dd5701d1129ab49) was created"
}
```

FN4T creates the import job to create a PR. Since the execution of the import is deferred to an AIG job, the Teamcenter ID of the created Problem Report is unknown at the time the response is sent back.

Error-Response:
HTTP/1.1 500 Internal Server Error
{
  "status": "ERROR",
  "message": "The job cannot be created"
}

FN4T then retrieves meta data and attached files from Opcenter EX DS and creates or updates the Problem Report.

The demo scenario uses the custom type "Non Conformance" derived from Problem Report, which is able to store the Opcenter EX DS NId in addition to the OOTB type. Please note that you have the possibility to configure a different corresponding object like Issue Report (IR) in <GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_custom_parameter.sd by changing as followed:

```plaintext
variable IssueReportType IssueReport
variable IssueReportRevision A
variable ReferenceFolder CMReferences
```

You can change the default IssueReportRevision or the corresponding ReferenceFolder as well.

Explore in more detail <GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_issuereport_import_mapping.sd. The current mapping follows the pattern of an object import and executes the following steps:

- Identify existing object or create new object.
- Get data for Opcenter EX DS NCN by calling OData query.
- Download files and Defects (if needed).
- Associate objects to the preconfigured reference folder.
- If the incoming status is NOTIFYING-ENGINEERING-ISSUE (which only happens with Opcenter EX DS 3.0), then FN4T sends a notification to Opcenter EX DS using the command PLMCreateIssueReportResult. In case if the job executed correctly, Opcenter EX DS NCN status is updated to NOTIFIED-ENGINEERING-ISSUE.
- Finally FN4T determines a Teamcenter workflow template based on the NCN status and starts a workflow instance. This workflow usually sets the Teamcenter status to an equivalence of the NCN status. The demo scenario starts a workflow that additionally includes an engineering review and a transfer of changes back to Opcenter EX DS in case the NCN status is NOTIFIED-ENGINEERING-ISSUE (the notification with this status implicitly indicates that the "new" state machine is in place and the system is Opcenter EX DS 3.1 or newer). This workflow finally sets the status back to OPEN.

The fields to import are configured in the GenObjMapping2TC_Object and updateTC_Object functions in the mapping file <GS_ROOT>/var/mmap/fn4t_mapping_config/
fn4t_issuereport_import_mapping.sd. Please note that every object exposed from the OData query from Opcenter EX DS has to exist in Teamcenter.

**OData query to retrieve details for Non-Conformance Notification (NCN)**

<table>
<thead>
<tr>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teamcenter Types</td>
</tr>
<tr>
<td>Target system</td>
</tr>
<tr>
<td>Used API (Existence, Details&amp;Defects, Attachments)</td>
</tr>
<tr>
<td>Mapping template</td>
</tr>
</tbody>
</table>

The function `getObjectAndAttach` seeks for the object and attaches to the just created PR. The download of the files attached to the Opcenter EX DS NCN will be started at the end of the procedure.

**Maintain Non-Conformance Notification**

The capability to maintain an Non-Conformance Notification (NCN) in Opcenter EX DS is based on ProblemReportRevision released from Teamcenter. The actual transfer is implemented in the procedure `::FN4T::GENOBJ::MAPPING::NONCONFORMANCE::performTransfer` which is not part of the custom mapping but can be overwritten, if needed.

<table>
<thead>
<tr>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName                                                                             NonConformance</td>
</tr>
<tr>
<td>Teamcenter Types                                                                           ProblemReportRevision, NonConformanceRevision in the demo scenario</td>
</tr>
<tr>
<td>Target system                                                                              Opcenter EX DS</td>
</tr>
<tr>
<td>Used API (for attributes, documents, status)                                              UADMUpdateNonConformance, CreateNonConformanceAttach</td>
</tr>
</tbody>
</table>
### Export

<table>
<thead>
<tr>
<th>Mapping template</th>
<th><code>&lt;GS_ROOT&gt;/var/mmap/fn4t_mapping_config/fn4t_nonconformance_mapping.sd</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom namespace</td>
<td><code>::FN4T::GENOBJ::CUSTOM::MAPPING::NONCONFORMANCE</code></td>
</tr>
<tr>
<td>Preferences start with</td>
<td><code>FN4T_NonConformance</code></td>
</tr>
<tr>
<td>Dictionaries expected by</td>
<td><code>::FN4T::GENOBJ::MAPPING::NONCONFORMANCE::performTransfer</code></td>
</tr>
<tr>
<td>Sample workflow task in demo template</td>
<td>&quot;UpdateNonConformance&quot; in template &quot;FN4T_NC_NOTIFIED_ENGINEERING_ISSUE&quot;</td>
</tr>
</tbody>
</table>

The dictionary **MESInputDatObject** has a list of attributes to set (Notes and Severity). **MESInputDatDocuments** and **MESInputDatDocumentAssociations** contain the documents to transfer (based on the contents of the solution folder of the Problem Report) and **MESInputDatResult** contains the UID of the NCN and the new status to set.

---

**Note:**

In the demo scenario, defect associations and the files in the reference folder are only updated in Teamcenter on **NCN import**, while files in the solutions folder are only transferred to Opcenter (**NCN export**). Status and the fields Notes and Severity are updated in **both directions**.

The NCN loop (bi-directional update consisting of import and export) is only fully supported by Opcenter EX DS 3.1. Opcenter EX DS 3.0 does not support status updates and does not trigger the import on every status change, but only for **NOTIFYING-ENGINEERING-ISSUE**.
12. Dataview

Dataview Configuration

The FN4T Dataview extension for the Teamcenter RAC needs no configuration once the demo template is installed. However, if you want to change the mapping or add a new field, please learn first about how to configure the Dataview in the Active Integration - Generic Configuration Guide.

<table>
<thead>
<tr>
<th>Export</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetTypeName</td>
<td>AffectedWorkOrder</td>
</tr>
<tr>
<td>Teamcenter Types</td>
<td>MECollaborationContext,</td>
</tr>
<tr>
<td></td>
<td>ProblemReportRevision</td>
</tr>
<tr>
<td>Target system</td>
<td>Opcenter EX DS</td>
</tr>
<tr>
<td>Used API</td>
<td>WorkOrder?$filter=PBOPIdentID %20eq%20'CCUID'&gt;'</td>
</tr>
<tr>
<td>Mapping template</td>
<td>&lt;GS_ROOT&gt;/var/mmap/ fn4t_mapping_config/ fn4t_prop_mapping.sd</td>
</tr>
<tr>
<td>Custom namespace</td>
<td>::FN4T::PROP::CUSTOM::MAPPING</td>
</tr>
<tr>
<td>Preferences start with</td>
<td>FN4T_AffectedWorkOrder</td>
</tr>
</tbody>
</table>

The FN4T Dataview lists only work orders created from an AsPlannedBoP, not from a MasterPlan.
13. Levels of Configuration

Introduction

This and the following chapters are meant to support the configurator in implementing specific customer requirements. They give an overview of how Opcenter Connect FN for Teamcenter can be modified and extended. Depending on the complexity the presented scenarios require more or less modifications. None of the adaptions described here is necessary to operate the demo scenario.

FN4T’s transfers are implemented following approaches: While most of the 150% transfers and the 100% PLMXML routing transfer follow the traditional Teamcenter Gateway template export and import approach, only the web service layers on top of these transfers are implemented using the step engine. All three approaches (import, export, step engine) will be explained here in general.

FN4T Teamcenter Gateway template export approach

Please refer to Generic Preference Concept and Generic Mapping Concept in the Active Integration - Generic Configuration Guide to read more about how to configure preferences in Teamcenter and mapping in T4x.

All 150% and 100% export transfers (Teamcenter to Opcenter) are based on the so-called generic transfer handler. In a Teamcenter workflow only a single handler is used: T4X-transfer-GenericObject4TargetType. The handler can be configured to support different target types in Opcenter and can be configured to support simple objects, structures (BOMs) or PLMXML (and it can even be used for different Teamcenter Gateway products and has some more options, see FN4T API Reference). Using this handler, the actual type of transfer is determined by the parameter “TargetTypeName”. This parameter determines the preferences which describe the Teamcenter data extraction as well as the TCL namespace to call for actually executing the specific mapping logic. The TargetTypeName always addresses two namespaces: the standard namespace :<Product>::<ObjectType>::MAPPING::<TargetTypeName> and the custom namespace :<Product>::<ObjectType>::CUSTOM::MAPPING::<TargetTypeName>. <Product> is always "FN4T" in this context, <ObjectType> is one of "GENOBJ" (for simple, unstructured object transfers), "GENBOM" (for structured objects, BOMs) or "PLMXML" (for transfers based on PLMXL, for PLMXML the standard namespace carries an additional "OBJECT::" namespace component). While the standard namespace’s procedures are packaged in fn4t.rfdt and cannot be modified, the custom namespace procedures are available as source files and can be adapted. Moreover each procedure in the standard namespace can be overwritten by implementing a procedure with the same name in the custom namespace. This allows for easy and upgrade-safe customization.

A transfer in FN4T (also called transaction) handles a single Teamcenter object. For each transaction FN4T calls the procedures of the TargetTypeName in the following sequence:

1. Data Retrieval (getTcData if called from TCL, implicitly handled in C code within the tcserver process if called from a Teamcenter workflow). The behaviour of the retrieval can be configured in a wide range using Teamcenter preferences.
2. Mapping (**TC_Object2MES_Object** for object and BOM transfers, **PLMXML_Data2EA_Object** for PLMXML transfers). This procedure has to evaluate the data retrieved in the previous step (or provided by the PLMXML created in a previous workflow task) and map it to the format expected by the target system and command. In FN4T usually the procedure creates one TCL dictionary (dict) per Opcenter command that will be called for this transfer. Each dict contains data in a format that directly corresponds to the input of the target command. Each data field in the dict carries a type specifier separated by colon (using TCL safejson). This allows an easy and generic transformation of the dictionaries to the target JSON format and so the payload can later easily be created. The mapping procedure has to be defined in the custom namespace, no transfer has an implementation in the standard namespace.

3. Position mapping (**TC_Object2MES_ObjectPosition**, only available for structured transfers!) This procedure gets a single position index as input that allows to retrieve information about a single, specific position of the BOM. The framework calls this procedure for all direct children of the BOM header. This can be extended to multiple levels, see the **FN4T API Reference** of the handler. The procedure stores the information about the position in an intermediate format (TCL dicts in case of FN4T), so the next step can easily consume it.

4. Actual Transfer (**performTransfer** for object and BOM transfers, **performEATransfer** for PLMXML). This proc uses the dictionaries created in the last step, transforms it to the target format (e.g. JSON) and transfers it to the target system (e.g. per web service). For all the TargetTypeNames in the demo scenario the standard namespace defines an implementation for this proc. If needed you can overwrite this by defining the same proc in the custom namespace.

5. Retrieve target system information (**getObjectInfo**). This procedure allows to retrieve information about the newly created or updated object (e.g. an internal, automatically generated ID, see last step) from the target system in order to include them in the next step. This is rarely used and so the standard namespaces of all the demo TargetTypeNames define such a procedure, but the implementation actually does nothing but return "OK".

6. Reverse Mapping (**MES_Object2TC_Object** for object and BOM transfer, **EA_Object2PLMXML_Data** for PLMXML transfers). This procedure uses the information from the actual transfer step and the target object information and can modify the Teamcenter object. It could e.g. store the internal ID from the target system in a form field or some date and time or transfer status in Teamcenter form attributes. Most demo TargetTypeNames implement an empty reverse mapping, i.e. nothing gets written back to Teamcenter. Note that the obligation of this procedure is only to prepare the data to write back to Teamcenter; the framework finally takes care of the actual storage in Teamcenter.

The demo scenario defines the following TargetTypeNames for simple object transfers:

- Material
- Process
- Operation (PlantBOPOperation)
• Step (PlantBOPStep)
• Skill
• Tool
• EquipmentType
• Defect
• Inspection
• NonConformance
• CreateCC (this transfer type is implemented completely in the target namespace as it is considered a demo helper only.)

The demo scenario defines the following TargetTypeNames for BOM transfers:

• DCD
• EquipmentHierarchy

The demo scenario defines the following TargetTypeName for PLMXML transfer:

• Routing

**FN4T Teamcenter Gateway template import approach**

This approach works exactly as described in the [Active Integration - Generic Configuration Guide](#). The following capabilities are implemented using this approach:

• **Opcenter EX DS Non-Conformance Notification**

• **Prepare 100 Percent BOM and BOP** (for "Workflow to Create CCObject" and "Web Service to Create CCObject")

**Step engine based approach**

The step engine abstracts the Teamcenter Gateway template approach and assumes that a transfer (export, import, etc) consists of a linear sequence of implementation steps which get executed one after the other and pass along some context (typically TCL dictionaries). In case of an error the whole execution is stopped. There are more elaborated features of the step engine and a layer that leverages this abstract concept to Teamcenter Gateway concepts, e.g. import and export, executing a step engine from a Teamcenter workflow handler, logging, synchronous and asynchronous web services, etc. The step engine is used to implement the transfer of work order header and non-conformance (because
these transfers are Teamcenter agnostic and hence no Teamcenter Gateway approaches apply) and to add the web service layer on top of the Teamcenter Gateway template export approach and the Teamcenter Gateway template import approach.

The following capabilities make use of the step engine: Create WorkOrderHeader in Opcenter EX DS, Create Confirmation in ERP, Transfer 100 Percent BOP to Opcenter EX DS, Prepare 100 Percent BOM and BOP (for "Web Service to Create CCObject").

**Changes of the Attribute Mapping**

The demo scenario delivered with FN4T is restricted to the Teamcenter standard data model. Customer data models usually contain more attributes on Teamcenter and Opcenter side or require a different handling of certain attributes. Many simple type changes and attribute mappings can be changed in the file *fn4t_custom_parameter.sd*. If you need to go beyond these changes, you need to look at the specific mapping procedures. First you need to determine whether the object or attribute is transferred in the 150%, 100% transfers or in one of the imports. Then identify the TargetTypeName, e.g. Material or name of the import. The TargetTypeName usually corresponds to the type name of the transferred object in the target system (Opcenter). In the mapping directory look for the mapping file containing the TCL namespace `::FN4T::GENOBJ::CUSTOM::MAPPING::<TargetTypeNameInUppercase>`. For the sample "Material" TargetTypeName the namespace is `::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL` and the file containing this namespace is `fn4t_material_mapping.sd`. Next in this file look for the TCL procedure implementing the mapping, usually the name is `TC_Object2MES_Object`. Within this procedure you will find a direct mapping from Teamcenter attributes to Opcenter attributes, e.g. the following line which maps the Teamcenter item revision name (object_name) to the Opcenter description:

```tcl
dict set ItemDict Description:string 
    [::T4X::TC::MAPPING::FieldMapping $ItemRev object_name]
```

Note that the Opcenter attribute name in general has to be postfixed with a type in order for the framework to create the correct JSON representation of the value. Also note that this kind of configuration is limited by the amount of available attributes in your Opcenter instance. Consult your Opcenter administrator for details.

If we wanted to change this mapping, we could e.g. use the fixed string "My description: " plus the Teamcenter description (object_desc) instead of the name. So the line would change to:

```tcl
dict set ItemDict Description:string 
    "My description: [:T4X::TC::MAPPING::FieldMapping $ItemRev object_desc]"
```

After saving this change and redeploying the mapping (see Manage a development environment) a newly initiated Material transfer will create a Opcenter EX DS Material with the new description. Note however that the used Opcenter EX DS command will not update existing objects. You have to transfer a new item or revision to observe the change.
The OOTB configuration of FN4T extracts all custom attributes from the direct targets of a transfer in the Teamcenter database (only some attributes are black-listed). This may cause problems if types are customized. It is required to change from the blacklist approach suitable for demos to the whitelist approach described in Blacklist/Whitelist for productive environments!

If you need to include attributes that reside on objects attached to the target of the transfer in Teamcenter, this can be achieved in many cases by adding transfer preferences (see Preferences). In this case also the path to the attached object has to be added to the TCL mapping. Let's assume we wanted to use the master revision form attribute "item_comment" instead of the OOTB mapping, we would need to add the following line to the preference FN4T_MaterialMapping4ItemRevision:

```
IMAN_master_form_rev:ItemRevision Master
```

or in XML:

```
<value>IMAN_master_form_rev:ItemRevision Master</value>
```

This will add the attributes of the revision master form to the extraction. In the TCL mapping we have to change the description line to:

```
dict set ItemDict Description:string [::T4X::TC::MAPPING::FieldMapping "$ItemRev:IMAN_master_form_rev:ItemRevision Master" item_comment]
```

This will use the attribute of newly included object to the Opcenter EX DS Material. The same approach can be used to navigate to other objects related via Teamcenter GRM or linked by reference attributes.

In principle also additional Opcenter EX DS attributes can be filled by the mapping code, but usually this would require a custom command on Opcenter EX DS side and will not work with OOTB Opcenter EX DS. Nevertheless, let's assume we wanted to populate an imaginary "AdditionalInfo" attribute on Opcenter EX DS side with the value of the Teamcenter revision description, we would simply have to add the following mapping line:

```
dict set ItemDict AdditionalInfo:string "$[::T4X::TC::MAPPING::FieldMapping $ItemRev object_desc]"
```

The generic transfer procedure will take care of the new attribute, transform it to JSON and include it in the web service call.

### Support of Custom Teamcenter Types

Many simple type changes and attribute mappings can be changed in the file fn4t_custom_parameter.sd. If however you need to change to a Teamcenter type that cannot be configured this way, you need to change the specific mapping. First identify the TargetTypeName, mapping file and mapping procedure as described in the last chapter.

Let's assume we want to change the item type of the Material transfer from the OOTB type "ItemRevision" and "PartRevision" to the imaginary "CT4CustomRevision" type (which should derive from "ItemRevision" for simplicity). First we need to add the new type to the type list preference:

```
Support of Custom Teamcenter Types

Many simple type changes and attribute mappings can be changed in the file fn4t_custom_parameter.sd. If however you need to change to a Teamcenter type that cannot be configured this way, you need to change the specific mapping. First identify the TargetTypeName, mapping file and mapping procedure as described in the last chapter.

Let's assume we want to change the item type of the Material transfer from the OOTB type "ItemRevision" and "PartRevision" to the imaginary "CT4CustomRevision" type (which should derive from "ItemRevision" for simplicity). First we need to add the new type to the type list preference:
We removed the existing entries, because we no longer want to support the OOTB types for the sake of simplicity. Next we also need to remove the two existing preferences FN4T_MaterialMapping4ItemRevision and FN4T_MaterialMapping4PartRevision and introduce a new preference instead:

```
<preference name="FN4T_MaterialMapping4CTCustomRevision" type="String" array="true" disabled="false" protectionScope="Site" envEnabled="false">
  <preference_description>Describes what to read for the Custom Revision in context of the FN4T Material Transfer</preference_description>
  <context name="Teamcenter">
    <value>items_tag:Item</value>
    <value>#__getAllProperties__#:Properties</value>
    <value>IMAN_Rendering:DirectModel:ref_list/JTPART:ImanFile</value>
    <value>IMAN_specification:JPEG:ref_list/JPEG_Reference:ImanFile</value>
  </context>
</preference>
```

Basically we will extract the same navigation paths as for ItemRevision. Note that you have to change from blacklist to whitelist for productive environments!

Of course it is also possible to add custom types while keeping the OOTB configuration.

In our example case the mapping code in procedure ::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL::TC_Object2MES_Object is generic enough to support the new type, so we do not need to change it unless we wanted to extract and transfer additional (custom) attributes (see last chapter for instructions).

**Overwrite Opcenter EX DS command**

Opcenter EX DS provides web interfaces to call API's via HTTPS. To create Material(s), for example, the URL for posting looks like:

https://<opcenterhost>/sit-svc/Application/AppU4DM/odata/PLMCreateMaterialDefinitionList

The "Application/AppU4DM/odata" part of the URL above is the so-called "Application" and "PLMCreateMaterialDefinitionList" is the command.
If you need to call different command instead of called in the out of the box solution, overwrite the default command by setting the following global variable to overwrite the default application and the command:

```plaintext
set ::EXDS::CommandName(/Application/AppU4DM/odata/PLMCreateMaterialDefinitionList) [list "/Application/AppU4DMCust/odata/" "CreateMaterialDefinition"]
```

In this case the service calls your custom application called "AppU4DMCust" and at the same time the custom command CreateMaterialDefinition instead of "AppU4DM" and PLMCreateMaterialDefinitionList accordingly.

### Change Transfer Behavior

In case you need to change the behavior of the OOTB performTransfer procedure (e.g. for the Material transfer: ::FN4T::GENOBJ::MAPPING::MATERIAL::performTransfer) of a template based transfer, you can define a custom performTransfer method in the CUSTOM namespace:

```plaintext
namespace eval ::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL {

  ...

  # # Method: ::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL::performTransfer
  # # \brief Example template procedure for a custom performTransfer procedure
  # # \details Uses at least the same dict variables as the standard transfer.
  # # \param TransactionId TC session id
  # # \param TargetTypeName target type name
  # # \param CorrelationId to be used in all web service calls for this transaction
  # # \return OK or ERROR
  #
  proc performTransfer { TransactionId TargetTypeName CorrelationId } {
    variable MESInputDatObject
    variable MESInputDatDocuments
    variable MESInputDatDocumentAssociations
    set Status OK
    if {[lindex $Status 0] eq "OK"} {
```
The method may or may not call the original performTransfer if needed and call other custom functionality as needed (see “TODO” comments).

Just as an example of a simple transfer for your reference: Here is a standard implementation of the Material transfer from the non-CUSTOM namespace. This code is usually contained in an rfdt file and not visible. Please note that this code is subject to changes without notice!

The procedure simply calls an Opcenter EX DS command per each of the dicts provided by the mapping:

```bash
namespace eval ::FN4T::GENOBJ::MAPPING::MATERIAL {

    # Method: ::FN4T::GENOBJ::MAPPING::MATERIAL::performTransfer
    # \brief Standard procedure to perform the actual material transfer to the MES system.
    # \details Uses the dict variables MESInputDatObject, MESInputDatDocuments and MESInputDatDocumentAssociations of the namespace ::FN4T::GENOBJ::MAPPING::MATERIAL and calls the MES transfer for each. This will create a EXDS material, zero to n documents and accordingly zero to n associations.
    # \param TransactionId TC session id
    # \param TargetTypeName target type name
    # \param CorrelationId to be used in all web service calls for this transaction
    # \return OK or ERROR
```
# proc performTransfer { TransactionId TargetTypeName CorrelationId } {

    set ::errorCode "NONE"
    set ::errorInfo ""
    set Status "OK"

    tpwrite -logchannel [::PL4X::CORE::getSessionLogChannel] -mtype
    INTERN "::FN4T::GENOBJ::MAPPING::MATERIAL::performTransfer starts for
    $TransactionId $TargetTypeName $CorrelationId"

    # create material
    if {[lindex $Status 0] eq "OK"} {
        set Status [::EXDS::SERVICES::callCommand4Dict
        $::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL::MESInputDatObject
        {::EXDS::SERVICES::PrefixUADM} PLMCreateMaterialDefinitionList
        $CorrelationId]
    }

    # create documents
    if {[lindex $Status 0] eq "OK"} {
        set Status [::EXDS::SERVICES::callCommand4Dict
        $::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL::MESInputDatDocuments
        {::EXDS::SERVICES::PrefixUADM} PLMCreateDocumentList $CorrelationId]
    }

    # associate documents to materials if there are documents
    if {[lindex $Status 0] eq "OK"} {
        set Status [::EXDS::SERVICES::callCommand4Dict
        $::FN4T::GENOBJ::CUSTOM::MAPPING::MATERIAL::MESInputDatDocumentAssociations
        {::EXDS::SERVICES::PrefixUADM} PLMAssociateDocumentToMaterialDefinitionList
        $CorrelationId]
    }

    tpwrite -logchannel [::PL4X::CORE::getSessionLogChannel] -mtype
    INTERN "::FN4T::GENOBJ::MAPPING::MATERIAL::performTransfer result:
    $Status"

    # the calling interface accepts only a sinlge OK or ERROR
    if {[lindex $Status 0] eq "OK"} {
        set Status OK
    } else {
        set Status ERROR
    }

    return $Status
}
}
Introduce New Transfer

This is the most complex modification. It allows to implement a transfer that has not been part of the demo scenario and will address a new Opcenter command. It requires to first define a new TargetTypeName describing the intended Opcenter target object, a logical mapping of Teamcenter attributes to Opcenter with special focus on identifying attributes and to identify a Opcenter command.

Next a set of T4x preferences needs to be created. For a first version the following minimum set is required: FN4T_<TargetTypeName>**TypeList** (listing the valid Teamcenter revision types), FN4T_<TargetTypeName>**Mapping4<TCRevisionType>** (listing the navigation paths for the type) with minimum entries (Please replace <TargetTypeName> and <TCRevisionType> with the corresponding values!):

```xml
<preference name="FN4T_<TargetTypeName>Mapping4<TCRevisionType>"
type="String" array="true" disabled="false" protectionScope="Site"
envEnabled="false">
    <preference_description>Describes what to read for the Revision in context of the FN4T Transfer</preference_description>
    <context name="Teamcenter">
        <value>items_tag:Item</value>
        <value>#__getAllProperties__#:Properties</value>
    </context>
</preference>
```

Now create a new mapping file in `<GS_ROOT>/var/mmap/fn4t_mapping_config` and source the new file in `<GS_ROOT>/var/mmap/fn4t_mapping_config/fn4t_mapping_config.sd`. The new file has to contain all the procedures listed in division "FN4T Teamcenter Gateway template export approach" of Introduction except for `getTcData`. The necessary signatures (positional parameters) of the procedures can be found in the FN4T API Reference for one of the OOTB namespaces, e.g. Material. For the implementation of the mapping procedure you can use the Material mapping as a template and has to provide a TCL dict conforming to the target command's signature. The reverse mapping and `getObjectInfo` procedures can be left empty, i.e. these should just return `OK` in the simplest case. The `performTransfer` procedure has to be implemented in the custom namespace and should take the TCL dict prepared by the mapping and transform and transfer it to the identified Opcenter command using `::EXDS::callCommand4Dict` (see FN4T API Reference for details). The FN4T framework will take care to address the correct Opcenter system.

In order to test the new mapping (but not the transfer) you can use the script `Mapping/fn4t_test_object_mapping.tcl`. For using the new transfer in a workflow create a new task in some test- or release workflow with the handler `T4X-transfer-GenericObject4TargetType` with parameter `-TargetTypeName` set to your new TargetTypeName. As an example or template, look at the Material task in the 150% workflows of the FN4T demo template.
14. Troubleshooting

CLM Services are not triggered

Issue

Certain events in Opcenter cause a service to be called on FN4T. What if these service calls are not observable in FN4T?

Solution Steps

Step 1: In order to verify if the services were called at all in FN4T, open Basic Gateway Service Admin UI and open "Log Files/System". Find a log file named "tpapps64_http.log" for the Gateway Service configured on Opcenter side. In case the log file contains entries like the following the service was successfully called:

```
08/04/19 08:30:25.314732 tpapps ::ffff:10.134.120.33 - - [08/Apr/2019:08:30:25] "POST /pxml/createProdOrderOperationConfAsync HTTP/1.1" 401 260 "-" "-"
08/04/19 08:30:25.403780 tpapps ::ffff:10.134.120.33 - - [08/Apr/2019:08:30:25] "POST /pxml/createProdOrderOperationConfAsync HTTP/1.1" 200 313 "-" "-"
```

Note that typically the first call is answered with an HTTP 401 response (authentication denied) while the second succeeds. This is due to the commonly used protocol of web service clients to first call a web service without credentials and only if the server responds with 401 the client will resend the request with credentials.

Instead of "createProdOrderOperationConfAsync" also other services like "createProdOrderConfAsync", "triggerSkeletonTransfer", "updateIssueReport", "updateWorkOrderCC" might be candidates.

Step 2: If the service was actually called, but the response code (the number after the "HTTP/1.1") for the second entry was "404" (not found) then this service has not been exposed by your Gateway Service. This may be due to a syntax error in your TCL mapping, a wrong port number addressed by Opcenter or the Gateway Service server instance does not expose this service. Check your configuration!

Step 3: If the service was actually called, next verify if the service processing was successful. Check your transaction logs and system logs and the job list in Basic Gateway Service Admin UI. In any of these locations you should be able to track an error.

Step 4: In case the service was not called, verify user, password and URLs in Opcenter in the menu "System Configuration/Configuration Keys/FN4T Integration". Note that the host name and port and even the extension component of the URL may be rewritten by some intermediate network component (NAT gateway, load balancer, firewall, proxy, ...)! Make sure the Opcenter server can access the configured URLs: you can enter the URL in a browser opened on the
Opcenter host. The URLs should cause the browser to ask for the FN4T credentials and then display an error message with a text like: `{"status": "ERROR", "message": "HTTP POST method expected!"}`. This error message is intended as the services usually expect a payload to be passed which the browser does not provide. However this reaction shows that the URL is correct. Any other reaction (missing credentials dialog, a 404 response or no response at all, any other error message) probably indicates a connectivity issue (firewall, DNS, …). Consult your system administrator for help in establishing connectivity between the Opcenter and the Gateway Service host.

Step 5: In case the service was not called and you verified that the Opcenter configuration is correct and the connectivity between the Opcenter and the Gateway Service host is established, the reason for the failure may be a missing signal rule within Opcenter: The Signal/Event Rule Management Table is empty or needs to reload. Check with your Opcenter administrator. Here is a short overview of what to do:

- In Opcenter EX DS, look for "Host Management", then select your host and click on "Signal Manager".
- In case there is something on this page: Click the items one by one and remove all of them
- As soon as there is no more entry present on this page: On the right side click on the "Import" command.
- Browse for a folder like `C:\Program Files\Siemens\SimaticIT\Unified\VApps\UADM\Signals`.
- Select and import the available json files listed in the folder one by one. Select the option "Overwrite".
- From the "Signal Manager" page select the imported signals one by one and approve.
- Select all of the approved signals and click on the deploy icon.
• You will see the warning icon being displayed for all of the deployed signals. However the warning icon will disappear automatically soon.
A. Glossary

A

**ABAP**
ABAP is a proprietary programming language of the SAP AG.

**Admin**
is the term used in this document for people who install and configure Teamcenter and its components. This is in contrast to the "user" role.

**Admin UI**
Web based administrative user interface of the GS and BGS.

**AIG**
The entire Active Integration Gateway product family.

**AIG_ROOT**
Please see **GS_ROOT** and **BGS_ROOT**. This term is used if something is true for both the GS and BGS.

**AI-Object**
Application-Interface Object

**API**
Application Programming Interface.

**Apps**
See "GS".

**AppServer**
Application Server.

B

**BAPI**
The Business Application Programming Interface allows external programs to access objects and business processes in SAP.

**BGS**
Basic Gateway Service.
**BGS_ROOT**
The installation directory of the Basic Gateway Service (e.g. C:\Siemens\BGS).

**BMIDE**
Teamcenter Business Modeler IDE (Integrated Development Environment)

**BOM**
A Bill Of Materials is a list of the parts or components and their quantities that are required to build a product.

**BOM Header**
A BOM Header is the top item of a BOM. BOMs can have multiple levels, so this often means the top item of the actual level.

**BOP**
The Bill Of Process describes a manufacturing process and lists the operations and steps with all their instructions, consumed materials, resources, work places and machines.

**C**

**CCObject**
Collaboration Context Object

**CEP**
Camstar Enterprise Platform

**Change Master**
The Engineering Change Master (ECM) contains the metadata to a change number.

**Characteristic**
An characteristic is an attribute of a SAP class.

**CIO**
Camstar Interoperability

**D**

**Data Carrier**
Please see Vault.
Dataview
The Dataview is an extension to the Teamcenter RAC and is deployed as part of the TEM installation process of the Teamcenter Gateway. The Dataview is used to display the real-time data of external applications, associated with Teamcenter objects.

Dataview mark-up
is the language understood by the Dataview. The Dataview receives messages written in this language from the T4x server. Such messages can be formatted as XML or JSON. Normally users do not see such messages. They may however appear in log files or error messages. The so called prop mapping (e.g. t4s_prop_mapping_template.sd) contains TCL commands that compose messages in the Dataview mark-up.

DCD
Data Collection Definition

DIR
DIR is the abbreviation for a SAP Document Info Record.

Document Key
A Document Info Record is identified by the combination of Document Type, Document Number, Document Part and Document Version.

Document Structure
A Document Structure is like a Bill Of Materials for Documents.

E

EA
stands for Enterprise Application, any software or set of computer programs used by business users to perform various business functions in context of current integration's portfolio with Teamcenter.

ECN
The Engineering Change Notice can also be called an Engineering Change Note, Engineering Change Order (ECO), or just an Engineering Change (EC).

EPM
Enterprise Process Modeling

EWI
Electronic Work Instructions
A. Glossary

F

File Stream
Method of transfer to send an original to SAP.

FN4S
Closed Loop Manufacturing for SAP S/4HANA®

G

Gateway Menu
An additional menu item of the Teamcenter Gateway software available in the Teamcenter RAC.

GRM
The Generic Relationship Management provides a general way in which two objects can be associated via a relationship.

GS
Gateway Service, manages the communication between Enterprise Applications.

GS_ROOT
The installation directory of the Gateway Service (e.g. C:\Siemens\GS).

GUI
Graphical user interface.

GUID
Globally Unique Identifier

I

IDGEN
The IDGEN is a mechanism to get an external ID from the ERP system when assigning a Teamcenter ID.

Inspection Plan
Contains characteristics to be inspected in an operation and equipment to be used.

iPPE
Integrated Product and Process Engineering is a module that can be used to manage products with many variants.
**ITK**
The Integration Toolkit (ITK) is a set of software tools provided by Siemens PLM Software that you can use to integrate third-party or user-developed applications with Teamcenter.

**J**

**JCO**
The Java Connector is an interface to . In the context of it is now mostly replaced by the Netweaver RFC interface.

**JDBC**
Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database.

**Job**
Teamcenter Gateway features asynchronous transfer. This data transfer is managed via a Job.

**Job Pool**
The Job Pool contains all finished and unprocessed Jobs. It is managed by the BGS.

**Job Server**
The Job Server on the Basic Gateway Service (BGS) manages the Job and distribution them to the Job Agent for processing.

**JSON**
JavaScript Object Notation is a lightweight data-interchange format\(^1\).

**K**

**KPro**
Kpro stands for Knowledge Provider. See also Data Carrier.

**L**

**LOV**
List of Values

\(^1\) [JSON.org](https://www.json.org)
**M**

**Mapping**
The mapping is part of the T4x configuration. It contains the code that controls the behavior of the data transfer between Teamcenter and the ERP system.

**MFK**
Multi-key functionality in Teamcenter.

**MM**
MM is the abbreviation for a SAP Material Master.

**MOM**
Manufacturing Operations Management

**N**

**NCN**
Non-Conformance Notification

**NetWeaver RFC SDK**
The NetWeaver RFC SDK contains libraries for 3rd party applications to connect to. It can be obtained from the SAP ONE Support Launchpad.

**O**

**Object Key**
The Object Key is a string that contains the ID of an Enterprise Application object. If the identifier is a combination of multiple keys, then the Object Key is a combination of those keys in a defined order and format.

**Object Link**
A relation between SAP objects like Material Master and Document Info Record.

**Object Management Record**
Belongs to a SAP Change Number and Documents changes of one particular SAP object like a Material Master.

**OOTB**
Out of the box
A representation of a file in SAP.

**OSS Note**
The OSS Note is an online patch service for SAP. The patch can be identified by the OSS Notes number.

**PIR**
PIR is an abbreviation for a SAP Purchase Info Record.

**Portal Transaction**
This means that a transfer to SAP that is not triggered by a workflow handler but via the Gateway Menu.

**RAC**
stands for Rich Application Client also referred to as rich client or portal.

**Revision Level**
Used to show changes with reference to a change to a SAP Material Master or Document Info Record.

**RFC**
Remote Function Call (SAP)

**SAP**
SAP S/4HANA® / SAP Business Suite®

**SAP GUI**
This is the application for the SAP Business Suite® and SAP S/4HANA®.

**SAP Logon**
This is the application that a user needs to start the SAP GUI for a particular system. It may also refer to the process of logging in to SAP in Teamcenter via .

**SAP Portal iView URL**
Can be used to show sap content in a browser window.

**Session Log**
Shows one log file for each Teamcenter session. Written if T4x transactions are executed.
SSL
Secure Sockets Layer.

T

T4O_ROOT
Please see GS_ROOT

T4S 4-Tier Client (SAP Lite)
The 4-Tier Client or SAP Lite is a stripped down GS. It’s only purpose is to open the SAP GUI on a Teamcenter 4-Tier Client.

T4x
The entire Teamcenter Gateway product family.

TAO
The ACE ORB is a open-source and standards-compliant real-time C++ implementation of CORBA based upon the Adaptive Communication Environment (ACE).

TargetTypeName
This is the T4x internal name for the transaction type. E.g. MaterialMaster or DocumentInfoRecord.

TC
Teamcenter

TCL
is a high-level, general-purpose, interpreted, dynamic programming language.

TCPCM
Teamcenter Product Cost Management

TCPCM4S
Teamcenter Product Cost Management Gateway for SAP S/4HANA

TEM
Teamcenter Environment Manager

Transaction Code
A Transaction Code is a quick access code for a Transaction in the SAP GUI:
### Transaction Log
The Transaction Log is a T4x logfile on the BGS. It contains log information for a specific T4x transaction.

### Transfer Window
The Transfer Window triggers transactions via the Gateway Menu.

### Transport Package
A file that contains functions that can be imported to SAP.

### U

#### UOM
UOM stands for Unit of Measure.

#### URI
Unified Resource Identifier: a generalized from of a resource locator (URL) and resource name (URN), which just identifies a resource, but is not necessarily sufficient to locate (find) the resource. URIs are often used to identify configurations in Java and other languages. See [https://en.wikipedia.org/wiki/Uniform_Resource_Identifier](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier) for more details.

#### URL
Unified Resource Locator: a string with a certain format, allowing to load a resource from a network. URLs are a specific form or URNs.

### User Exit (SAP)
A User Exit is a code for a program that is called if an object like an MaterialMaster has been changed or updated. In the context of T4S it is often used to initiate the process to trigger a transfer from SAP to Teamcenter.

### User Log
The User Log is a T4x logfile on the BGS. If you define a customized logchannel, the information is written into a User Log of that name.

### V

#### Value Set
A Value Set is the SAP term for a list of selectable values for a characteristic.
Vault
The Vault is a server where a SAP DocumentInfoRecord original is stored. A synonym is also Data Carrier.

W

WBS
WBS is an abbreviation for a SAP Work Breakdown Structure.

X

XML
Extensible Markup Language is designed to store and transport data in a format that is both human- and machine-readable.

XRT
stands for XML Rendering Template, also known as XML Rendering Stylesheet. These are XML documents stored in datasets that define how parts of the Teamcenter user interface are rendered. They are used for the Rich Client as well as the Active Workspace.

Z

ZPTC
This is the short name for a Z-Table with the name /TESISPLM/ZPTC, used to trigger a transfer from SAP.

Z-Table
"Z" is a well-known prefix name for custom tables in the SAP world. A special table used with is the table /TESISPLM/ZPTC.
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 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.

© 2019 Siemens. Siemens, the Siemens logo and SIMATIC IT are registered trademarks of Siemens AG. Camstar, D-Cubed, Femap, Fibersim, Geolus, I-deas, JT, NX, Omneo, Parasolid, Solid Edge, Syncrofit, Teamcenter and Tecnomatix are trademarks or registered trademarks of Siemens Industry Software Inc. or its subsidiaries in the United States and in other countries. All other trademarks, registered trademarks or service marks belong to their respective holders.