S4TM6

Technical Deep Dive for Transportation Management in SAP S/4HANA

PARTICIPANT HANDBOOK
INSTRUCTOR-LED TRAINING

Course Version: 14
Course Duration: 5 Day(s)
Material Number: 50155633

SAP Copyrights, Trademarks and Disclaimers

Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation	
Demonstration	
Procedure	
Warning or Caution	
Hint	
Related or Additional Information	
Facilitated Discussion	
User interface control	*Example text*
Window title	*Example text*

Contents

Course Overview

TARGET AUDIENCE

This course is intended for the following audiences:

- Application Consultant
- Development Consultant
- Technology Consultant
- Developer
- Enterprise Architect
- Help Desk/CoE Support
- Solution Architect
- System Architect

UNIT 1 Architecture Overview

Lesson 1

UNIT OBJECTIVES

- Explain the SAP TM Software Component Architecture
- Describe the TM system landscape and integration options
- Define the Process Driven Solution Design
- Describe the difference between business documents and technical Business Objects
- Explain the Business Object Model of BO Transportation Request
- Describe the User Interface architecture

Unit 1
Lesson 1

Understanding the Architecture

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the SAP TM Software Component Architecture
- Describe the TM system landscape and integration options
- Define the Process Driven Solution Design
- Describe the difference between business documents and technical Business Objects
- Explain the Business Object Model of BO Transportation Request
- Describe the User Interface architecture

SAP TM Software Component Architecture

To go deeper in the technical background of SAP Transportation Management (TM), it is necessary to have a good overview of the architecture as well as the business object model.

The following figure shows the SAP Transportation Management (TM) architecture.

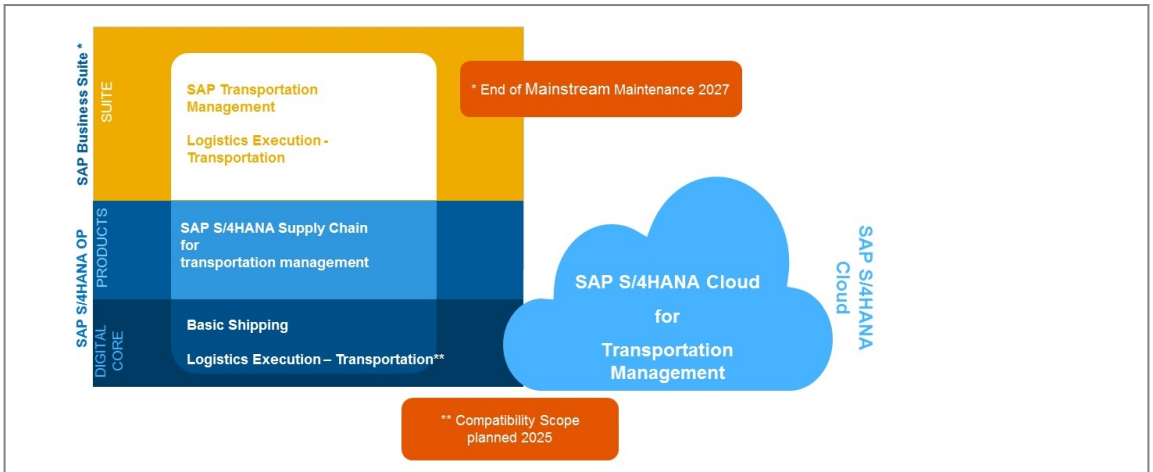


Figure 1: Transportation Management Architecture

The following figure shows you the possible SAP TM deployment options.

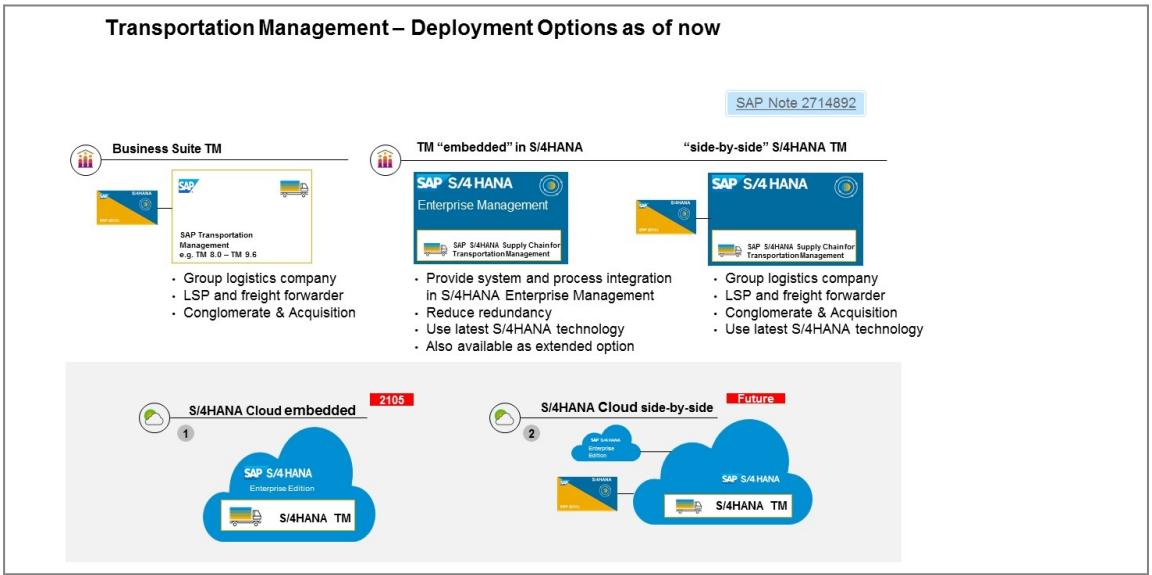
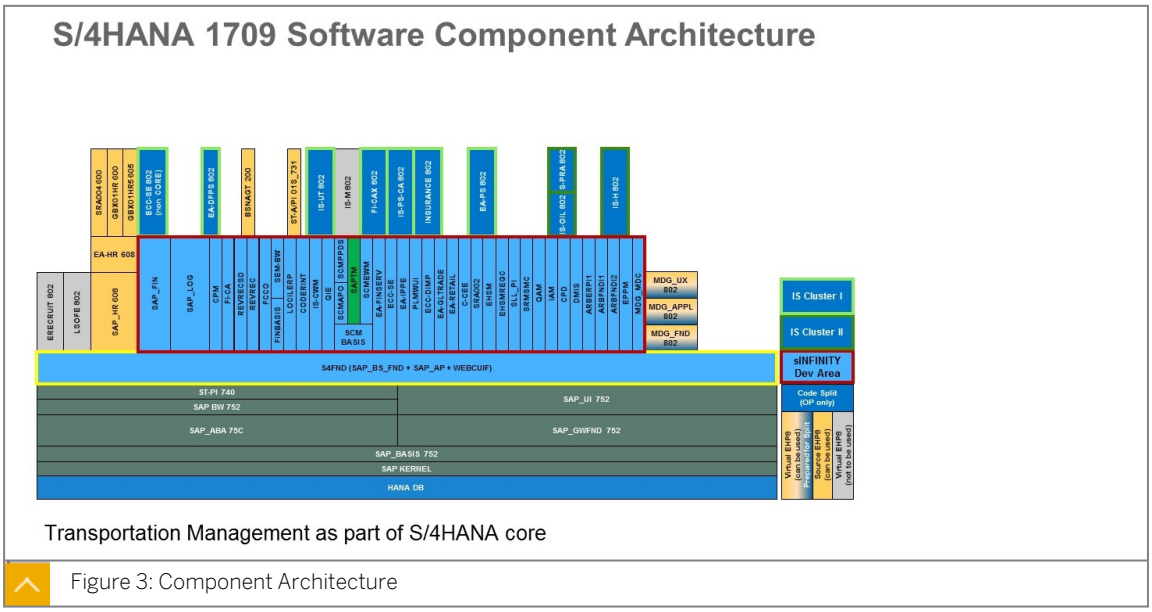


Figure 2: Deployment Options

The following figure shows you SAP TM as part of the S/4HANA core.



Transportation Management as part of S/4HANA core

Figure 3: Component Architecture

SAP TM System Landscape and Integration Options

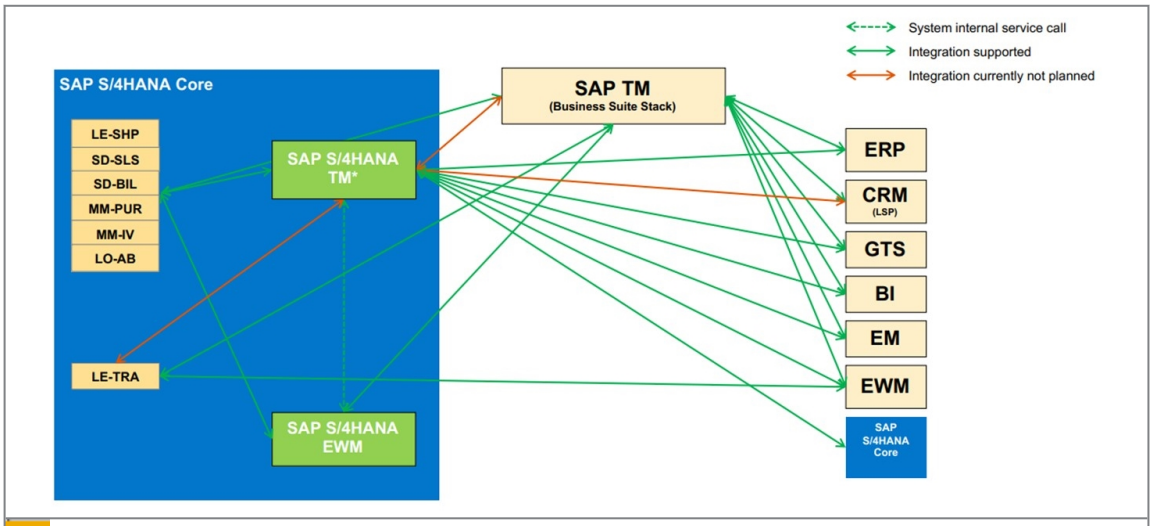


Figure 4: Integration Scenarios

Process Integration

- Service integration is supported as a point-to-point communication using Web-Service Reliable Messaging (WSRM)
- Optionally SAP PI can be used (mandatory for SAP ECC < ECC 6.00 EHP5)

Tool Integration

- External GIS Systems are connected via enterprise services
- Alternatively Geo Coding might be connected via IGS (Internet Graphics Server, part of Netweaver)
- SAP EM is always connected to TM via RFC at this time

The following figure gives a detail overview of the TM architectue.

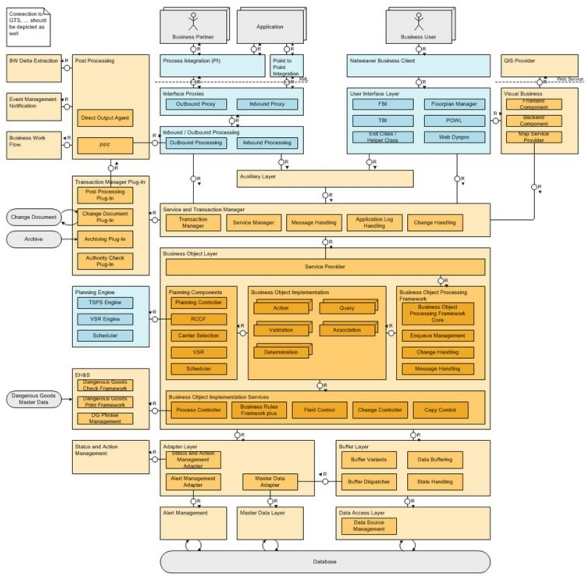


Figure 5: TM Architecture

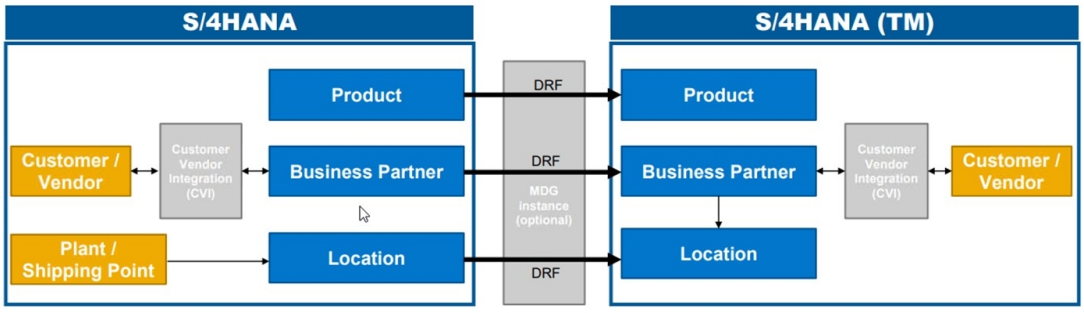


Figure 6: S/4HANA TM Master Data Integration

- Business Partner is a leading entity in S/4HANA:
  - CVI is used to replicate to customer/vendor master and vice versa (for compatibility reasons; S/4HANA default behavior)
  - Locations can be created out of Business Partner address (enhancement of S/4HANA TM 1709
  - Web Service is used for Product, Business Partner and Location

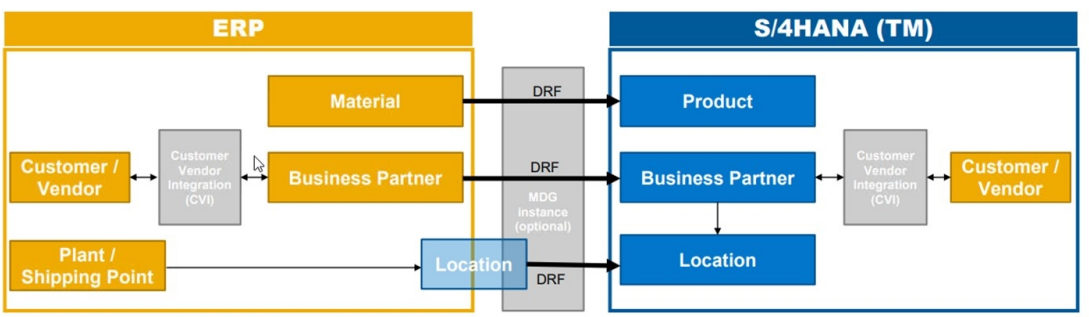


Figure 7: S/4HANA TM Master Data Integration 2

- Business Partner is a leading entity in S/4HANA:
  - CVI is used in ERP to replicate customer/vendor master to Business Partner and vice versa
  - Locations can be created out of Business Partner address (enhancement of S/4HANA TM 1709)
  - Web Service is used for Business Partner, Location and IDOC for Material
  - Plant/Shipping Point is mapped to location web service

Process Driven Solution Design

Object Design – focus on business:

- Integrated business object model allows dynamic, interactive processing of requirements and orders
- Business objects with *local* business logic and business *interfaces* to other objects
- Consistent re-work of requirements and planning and execution by design – *appreciate change*

Separation of concerns – be explicit:

- Transportation Requirements to reflect demand signals and customer needs
- Invent logical link to reflect single consignments as Freight Units
- Transportation Orders to reflect planning and business results
- Settlement documents to reflect financial statements and needs

Feature follows Process – process any time:

- Integrated at any time – connected features instead of sequential functionality
- Interaction a2a and b2b at same dynamic level supported – openness and sustainable

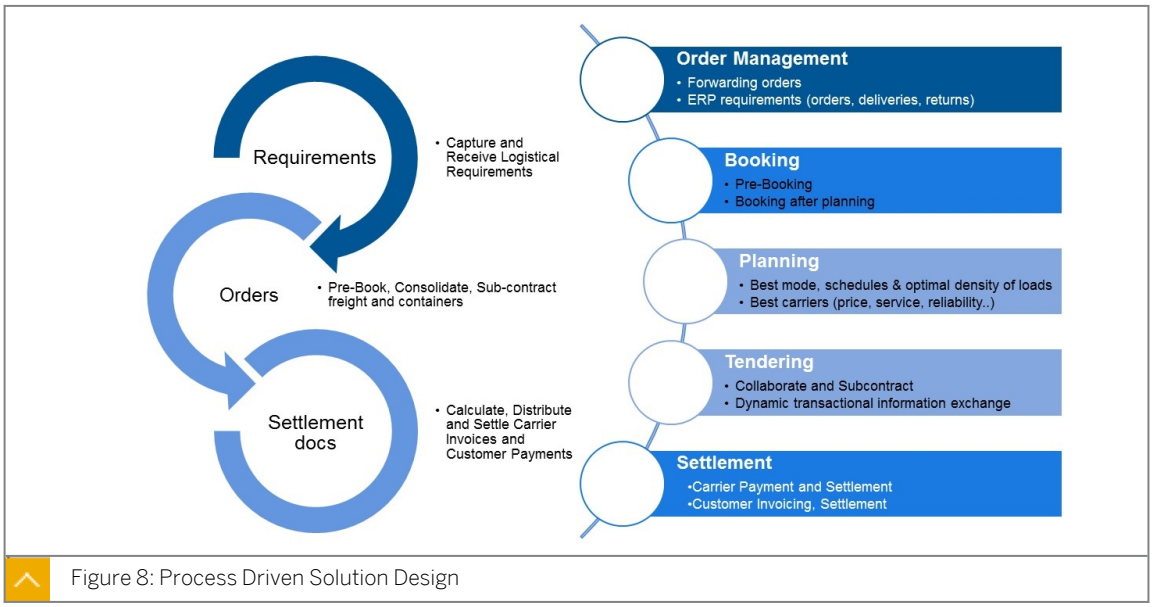


Figure 8: Process Driven Solution Design

Simplified Business Object Model

Transportation Requirements (TRQ):

- Forwarding Orders
- Order based requirements
- Delivery based requirements

Transportation Orders (TOR):

- Freight Units
- Freight Orders
- Freight Bookings

Goals

- Optimize Performance
- Reduce Memory Consumption
- No redundant storage of data
- No redundant implementation of business logic
- Reduce Maintenance Effort
- Reduce Enhancement Effort

SAP TM Business Object Model

- Transportation Request (TRQ)
  - Technical Object used to present Forwarding Order, Forwarding Order Quotation.
- Transportation Order (TOR)
  - Technical Object used to represent Freight Unit, Package Units, Container Units, Freight Order, Freight Booking.
- TRQ and TOR are the main Business Objects in TM.
- This design approach is based on the most simple processes of the transportation management solution where a transportation demand is directly forwarded to execution.
  - Therefore the corresponding transportation functionality must be available on these two Business Objects.
- All other (more complex) process variants are extensions of this simple basic process and are handled by additional instances of the two mentioned basic Business Objects.
- Further Business Objects in TM:
  - Forwarding Settlement (FWSD) – also called CFIR (Customer Freight Invoice Request).
  - Freight Settlement (FSD) – also called SFIR (Supplier Freight Invoice Request).
  - Master Data Objects to represent entities like Business Partner, Location, Schedule, etc.
  - Planning Transient Business Object (TBO) used in the context of planning functionality.

- Transportation Charges - Dependent Object to be reused in all objects subject to charge calculation.

Business Documents vs Business Objects

The technical Business Object Transportation Request defines the data structure and business logic to reflect transportation demands for transportation planning arising from different sources.

However the technical Business Object is not visible to the business user. Instead Business Documents are derived that share the same underlying data model.

The following Business Documents are derived from the technical object Transportation Request:

- Forwarding Order
- Forwarding Quotation (Request for Quotation)
- Order-Based Transportation Requirement
- Delivery-Based Transportation Requirement

This is defined and controlled by the TRQ Category The logistics integration in an embedded scenario (S/4HANA) is handled slightly different to dedicated TRQ category but no Business Object instance.

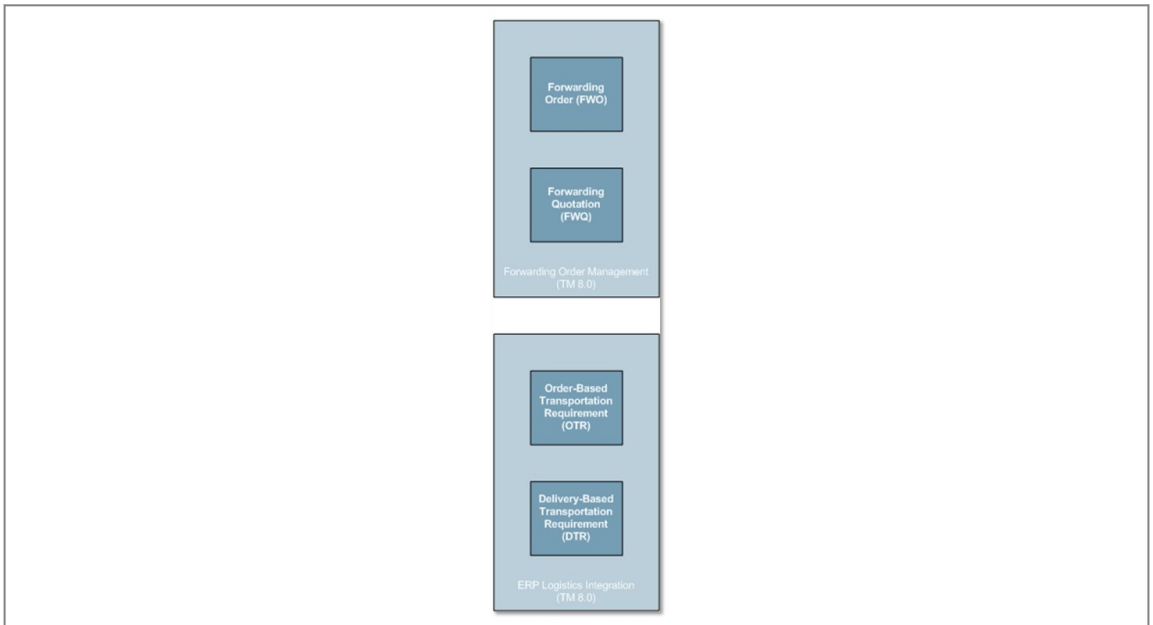


Figure 9: Business Document Vs. Business Object

Business Object Model

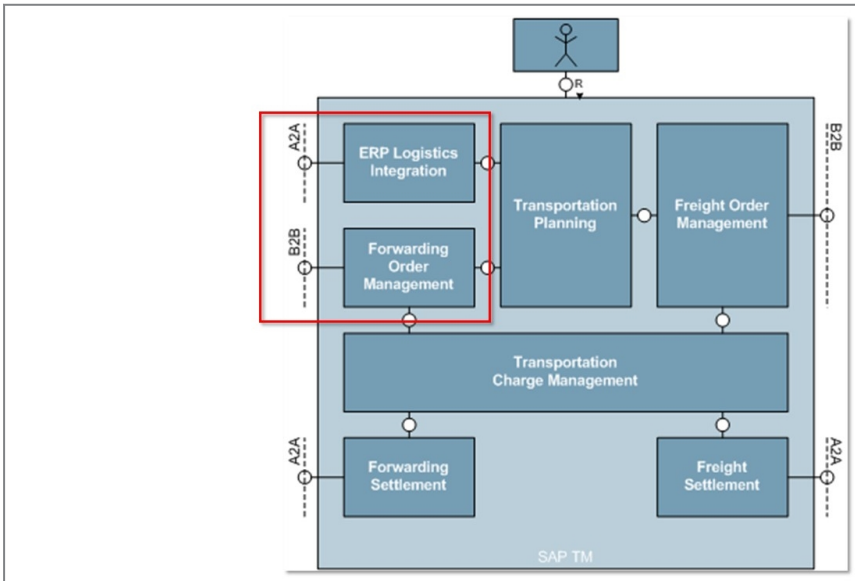


Figure 10: Transport Management Overview

The following lists shows you example usage of Technical Business Object Transportation Request in SAP TM.

Forwarding Order Management

- Comprises of functionality such as creation, update or confirmation of forwarding orders
- Forwarding Orders can be received electronically via EDI or can be entered through UI
- Tight integration with neighboring building blocks such as Transportation Planning, Transportation Charge Management and Forwarding Settlement

ERP Logistics Integration

- Comprises of functionality to receive transportation requirements arising from ERP Orders and ERP Deliveries (of an external ERP system)
- Functionality to create ERP Deliveries (in an external ERP system)
- Tight integration with Transportation Planning

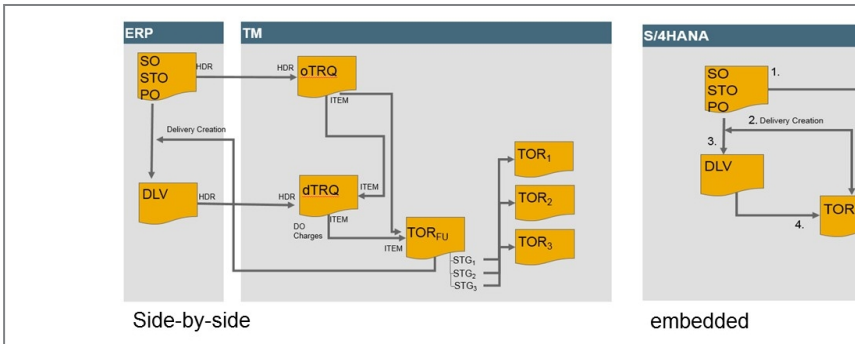


Figure 11: Side-by-Side Vs Embedded

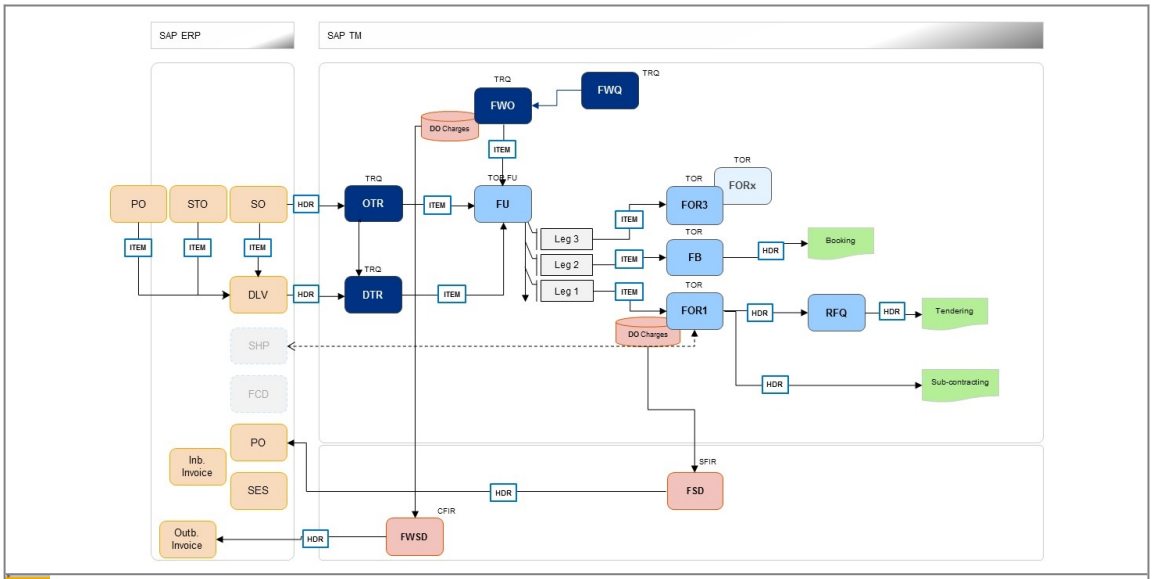


Figure 12: Business Object Model Holistic View

Business object model is designed in an inside-out approach, main driver are the standard TM business processes.

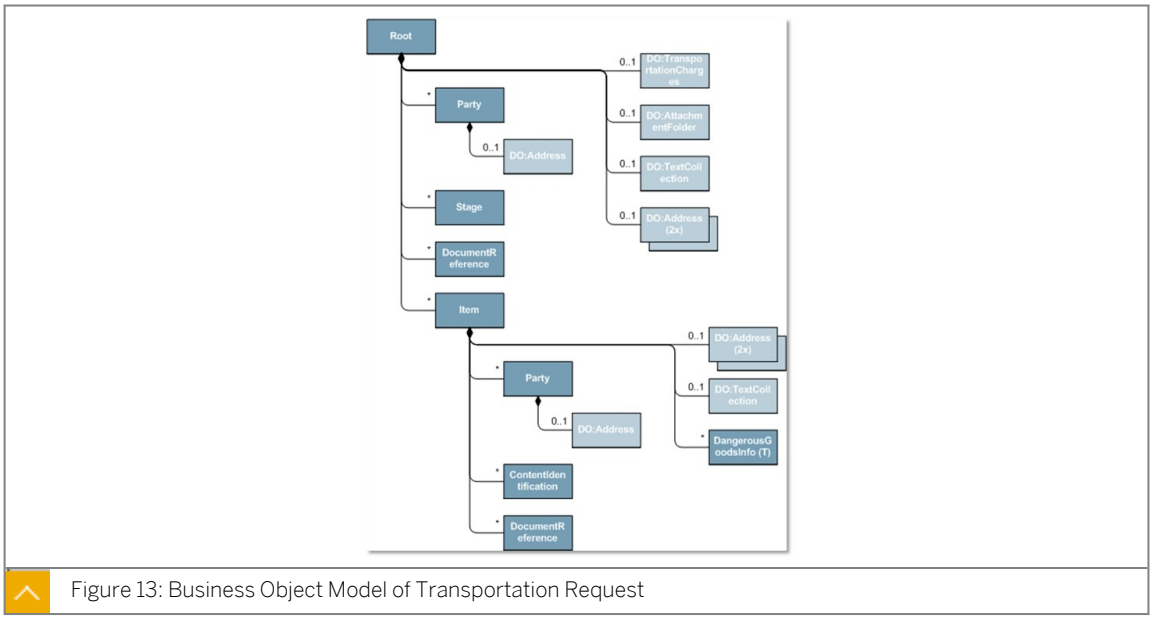


Figure 13: Business Object Model of Transportation Request

User Interface Architecture

SAP TM User Interface

- The UIs are based on intensive feedback from customers.
  - Usability tests were done in cooperation with customers and partners.
  - A UI First approach was followed. The UI is process oriented, following the SAP UI guidelines.
- Web Dynpro ABAP in combination with Floorplan Manager (FPM) is used to build the UIs.

- Increased development efficiency for Floorplan UIs.
  - Enables configurable UIs based on the processes as well as customers and users needs.
  - No programming is required for enhancements.
- Increased adaptation capabilities for customers and partners.
  - Reduced technical barrier for UI adjustments.
  - Adaptation by configuration instead of modifying Web Dynpro ABAP coding.
- Increased consistency across all application areas by reusing generic UI building blocks.

- The connection between the FPM UIs and the application backend is realized via the Floorplan Manager BOPF Integration (FBI) layer.
  - The BOPF Framework is used to model, implement and execute the business objects of TM, i.e. the backend functionality with all the business logic for any process step.
  - FBI provides Exit- and Conversion Classes that also allow to influence the UI behavior by coding.
- Data-sensitive UIs are possible now:
  - Variation of the UI depending on BO data.
  - For example:
    - TRQ Type 001: Express Packaged Goods,
    - TRQ Type 002: Full Container Load.
- Flexibility:
  - SAP TM standard delivers a UI proposal (general purpose).
  - Customers and partners can adapt it.
  - The adaptation does not overwrite the TM standard UI.

LESSON SUMMARY
You should now be able to:

- Explain the SAP TM Software Component Architecture
- Describe the TM system landscape and integration options
- Define the Process Driven Solution Design
- Describe the difference between business documents and technical Business Objects
- Explain the Business Object Model of BO Transportation Request
- Describe the User Interface architecture

Learning Assessment

1. Which of the following are valid deployment options for SAP TM?

   Choose the correct answers.

   - ☐ A SAP ERP to SAP TM ("side-by-side").
   - ☐ B SAP ERP to SAP S/4HANA TM ("side-by-side").
   - ☐ C SAP S/4HANA to SAP S/4HANA TM ("side-by-side").
   - ☐ D SAP S/4HANA to SAP TM ("side-by-side").
   - ☐ E SAP S/4HANA ("embedded").

2. How do you install S/4HANA TM in a S/4HANA system?

   Choose the correct answers.

   - ☐ A You install it using transaction `SAINT`.
   - ☐ B You can implement a note via transaction `SNOTE`.
   - ☐ C You cannot install it separately because it is already part of S/4HANA core.

3. Which integration options are available for S/4HANA TM?

   Choose the correct answers.

   - ☐ A Integration with SAP EM.
   - ☐ B Integration with SAP CRM.
   - ☐ C Integration with SAP EWM.
   - ☐ D Integration with SAP GTS.

4. Which business documents are based on the technical business object */SCMTMS/TOR*?

   Choose the correct answers.

   - ☐ A Freight Unit.
   - ☐ B Container Unit.
   - ☐ C Forwarding Order.
   - ☐ D Freight Settlement Document.

Learning Assessment - Answers

1. Which of the following are valid deployment options for SAP TM?

   Choose the correct answers.

   - [X] A SAP ERP to SAP TM ("side-by-side").
   - [X] B SAP ERP to SAP S/4HANA TM ("side-by-side").
   - [X] C SAP S/4HANA to SAP S/4HANA TM ("side-by-side").
   - [X] D SAP S/4HANA to SAP TM ("side-by-side").
   - [X] E SAP S/4HANA ("embedded").

2. How do you install S/4HANA TM in a S/4HANA system?

   Choose the correct answers.

   - [ ] A You install it using transaction `SAINT`.
   - [ ] B You can implement a note via transaction `SNOTE`.
   - [X] C You cannot install it separately because it is already part of S/4HANA core.

3. Which integration options are available for S/4HANA TM?

   Choose the correct answers.

   - [X] A Integration with SAP EM.
   - [ ] B Integration with SAP CRM.
   - [X] C Integration with SAP EWM.
   - [X] D Integration with SAP GTS.

4. Which business documents are based on the technical business object */SCMTMS/TOR*?

   Choose the correct answers.

   - [X] A Freight Unit.
   - [X] B Container Unit.
   - [ ] C Forwarding Order.
   - [ ] D Freight Settlement Document.

UNIT 2

Business Object Processing Framework (BOPF)

Lesson 1

UNIT OBJECTIVES

- Explain and apply the concepts of BOPF
- Define the BOPF model
- Explain the BO node structure
- Explain the different BO entities

Unit 2
Lesson 1

Understanding the Business Object Processing Framework (BOPF)

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain and apply the concepts of BOPF
- Define the BOPF model
- Explain the BO node structure
- Explain the different BO entities

Business Object Processing Framework

The Business Object Processing Framework (BOPF) controls the application business logic as well as the data retrieval of the buffer and persistency layer. Therefore, it is a very important component of TM. BOPF consists of different concepts and models to implement Business Objects.

What is the Business Object Processing Framework (BOPF)?

- BOPF is a framework for building whole applications
- UI independent
- No programming language, uses ABAP OO
- Flexible, but with a new way to develop applications

Why use the Business Object Processing Framework? Problems in Software Development.

- How to realize and solve real business world problem in ABAP - Business Object.
- Data access - Create, retrieve, update and delete DB entries.
- Data buffering - Buffer synchronization with DB.
- Locking mechanisms - Enqueue: shared, optimistic, exclusive.
- Message handling.
- Authorization and property handling - Mandatory, read-only, enabled etc.
- Consistent programming model over all application objects - Object oriented approach (ex: Reuse).
- Enhancement Concept for Partners and Customers, Rapid Prototyping BUI Integration, Service and Support.

Who uses BOPF?

- SAP Standard Development:
  - Uses BOPF
  - Implements BO structure
  - Implements business logic
- SAP Custom Development Partners and Customers:
  - Extend BO models with custom fields
  - Extend business logic with custom logic
  - Extend UI and UI behavior
  - Create own BO models
- SAP Service and Support

Where is BOPF used?

- The Business Object Processing Framework is part of the SAP Basis, located in `SAP_BASIS`. The following list shows the BOPF Transactions:
- Business Object Configurator (SAP Internal): `BOBF (/n/BOBF/CONF_UI)`
- Business Object Configurator (partners/customers): `BOBX`
- BOPF Test UI: `BOBT (/n/BOBF/TEST_UI)`

Business Objects and the Business Object Processing Framework

- Business Objects are the basis of the SAP TM application. Each Business Object represents a type of a uniquely identifiable business entity, described by a structural model, an internal process model. The business processes provided with SAP TM operate on these Business Objects.
- BOPF controls the application business logic as well as the data retrieval of the buffer and persistency layer. The main design principles are a clear separation of the business logic and the buffering of data as well as a clear structuring of the business logic into small parts with a clear separation of changing and checking business logic. The BOPF approach for implementing business objects breaks down business logic into the following entities:
  - Actions
  - Determinations
  - Validations
  - Queries Associations

BOPF Model

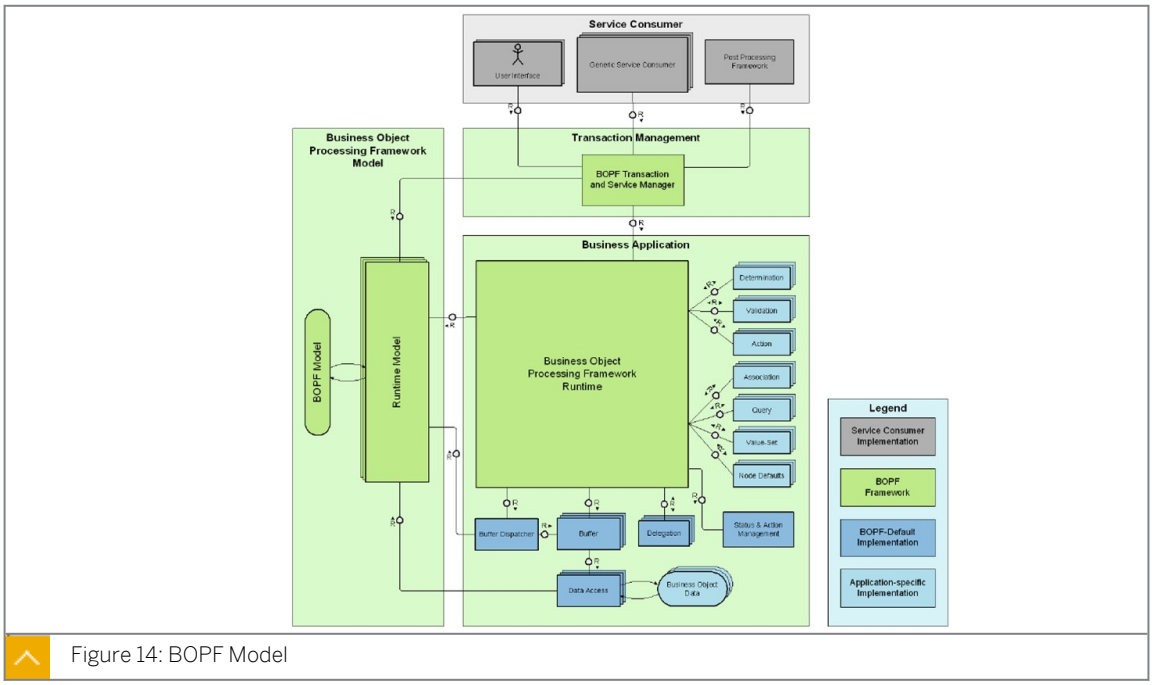


Figure 14: BOPF Model

Business Application is modeled within BOPF:

- Business logic implemented via Validations, Determinations and Actions
- BO internal and Cross BO Associations
- Dependent Objects being parts of Host BOs
- BOPF addressing Buffer Dispatcher handling Node Buffers addressing DAC accessing Database

BOPF configuration existing in Shared Memory as optimized runtime object. Transaction and Service Manager handle Application Transaction. UI / Consumer accessing Application via Service Manager.

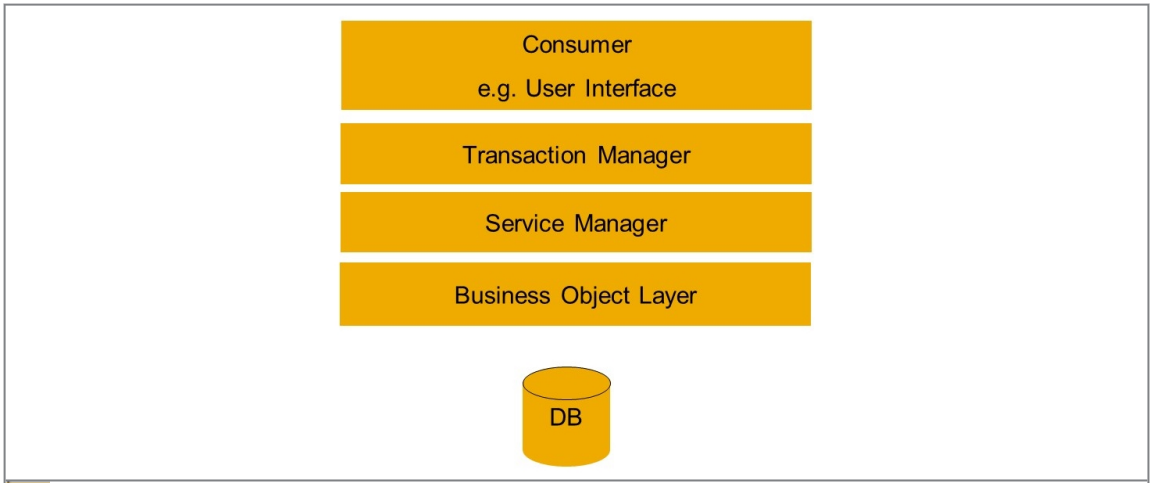


Figure 15: Layers

- Transaction Manager handles Service Manager instances
- Service Manager handles Business Object Layer instances
- Business Object Layer:
  - Handles Locking, Loading, Buffer, DAC etc.
  - Executes Validations, Determinations and Actions

BO Node Structure

The following figure shows the node structure at both Design Time and Runtime.

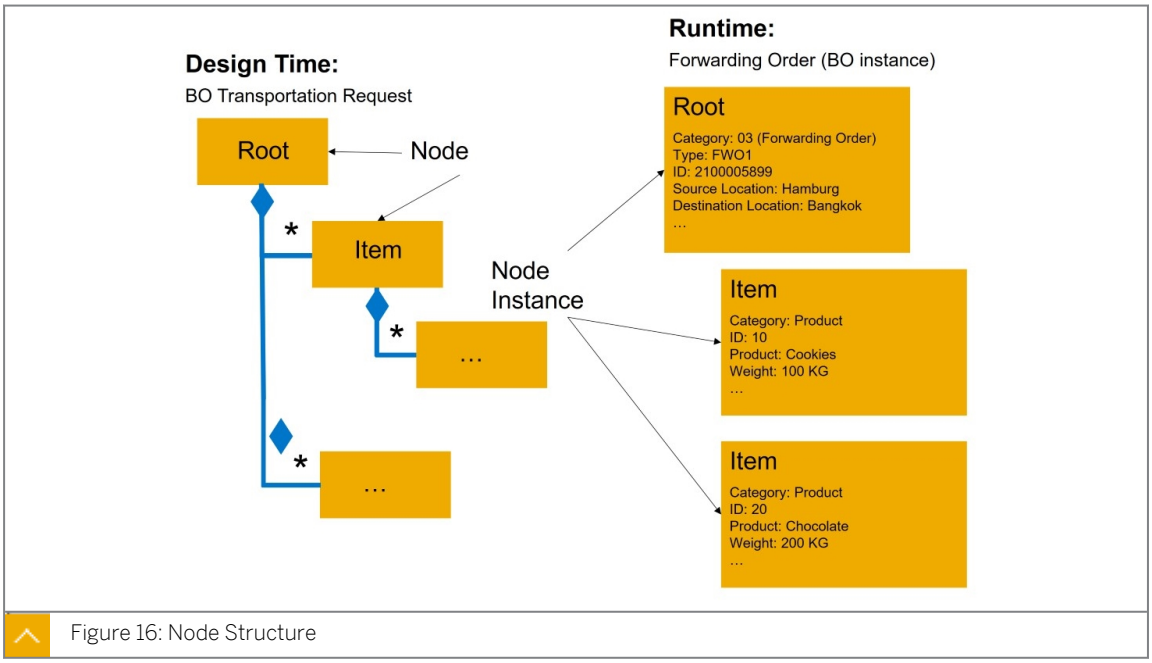


Figure 16: Node Structure

- **Design Time**. A Node is a semantically related set of attributes of a business object. Nodes can be used to define and structure your business object. The attributes of a business object node are defined by dictionary data types (structures). Nodes can be hierarchically defined and related. Each business object has only one Root Node. Nodes are defined via compositions in a tree, but nodes can also be related in an arbitrary structure via associations that can be separate from the tree structure.
- **Run Time**. Several node instances of a node can be created during runtime. Every node instance is identified by a unique (technical) GUID (node instance key).

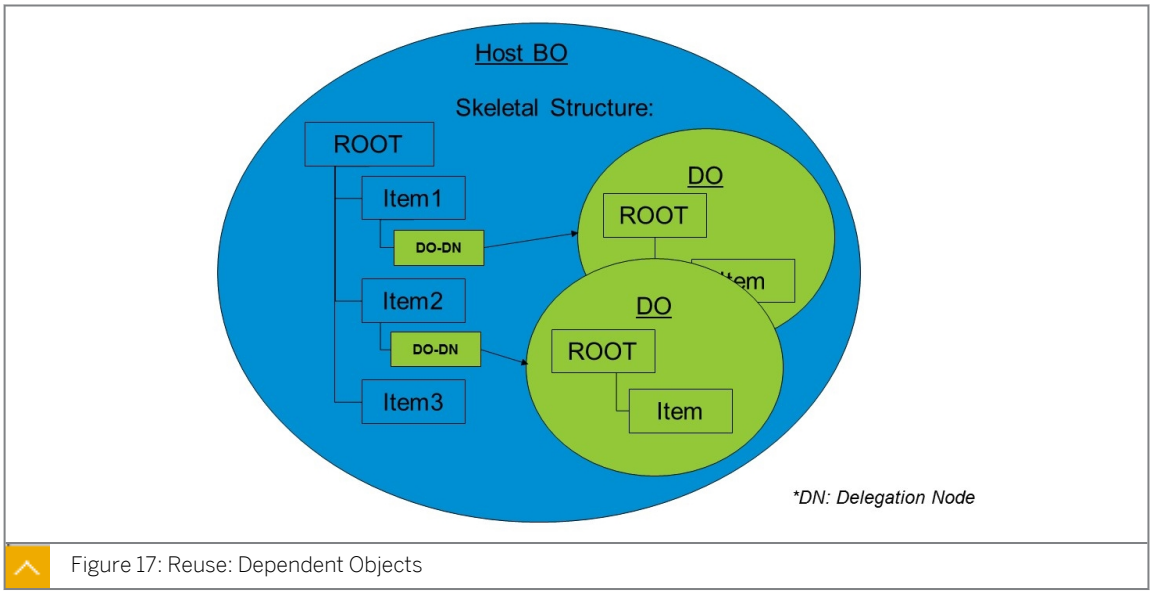


Figure 17: Reuse: Dependent Objects

BO Entities

Node Level

All entities are defined at node level.

Action

Actions are business logic entities assigned to a business object node called by consumer.

Determination

A Determination contains business object specific logic to perform side effects, which are executed if a business object internal event occurs.

Validation

Validations are business logic entities that are triggered in certain situations to check different aspects of a given set of node instances.

Association

Relationship between two nodes in a or between two Business Objects.

Query

A query is a business object entity and is always assigned to a certain node, whose instances shall be returned.

Alternative Key

Alternative keys are combinations of node attributes that identify an instance of a node and have some kind of business semantics (in difference to the technical node instance key).

Actions

- An **action** is an element of a business object node that describes an operation performed on that node.
- An action can be used to allow the external triggering of business logic by:
  - A service consumer.

  - Another business object.
  - Internally (for example, by another action or by a determination).
- Actions can modify node instances of the own as well as node instances of other business objects.
- Actions may have input parameters that are provided by the consumer.
- Actions return messages, failed keys (keys of instances for which the action has not been executed). In addition, the changes done by the action are returned (e.g. create, delete, update of certain node instances).
- Actions can have exporting parameters (node or a defined data type).

Determination

- A **determination** contains business object specific logic to perform side effects. For example, determinations can react on the creation, update, deletion or the loading of a node instance by deriving new values for fields.
- Determinations are executed by the framework (cannot be called explicitly).
- Request nodes that trigger the determination need to be configured Determinations can be configured to be executed at different times (after modify, before save, ...)
- Determination Types:
  - Transient Determination: Modifies only transient fields or nodes (e.g. buffer), thus no locking is necessary.
  - Persistent Determination: Modifies transient and persistent fields or nodes (e.g. database), thus locking is necessary.
- Determinations do not perform consistency checks – therefore use validations. Determinations are never executed in a nested manner to avoid endless loops (exception: after validation can trigger another determination cycle).

Validation

- **Validations** are business logic entities that are triggered in certain situations to check different aspects of a given set of node instances.
- Validations are executed by the framework (cannot be called explicitly).
- Validations never modify any data.
- Validations return messages and the keys of failed node instances.
- There are two categories of validations:
  - Consistency validations: Check the consistency of a node instance.
  - Action validations: Check, if necessary preconditions for executing an action on a node are fulfilled.

Consistency Validations

- **Consistency validations** check the consistency of node instances.

- Consistency validations only return error messages and the keys of the failed instances (i.e. instances that do not match the consistency criteria).
- Failed consistency validations can prevent saving if configured.
- Consistency validations can be configured to be executed on the following times:
  - If a consumer executes a check and the request node of the consistency validation is in the check scope (e.g. check or check & determine).
  - If there is a change on the request node of the consistency validation (e.g. create, update, delete).
  - If a consistency group, which contains this consistency validation, is executed.

Action Validations

- **Action validations** check the possibility to execute a certain action on a set of node instances.
- Action validations return only messages and the keys of failed instances (i.e. instances for which the action may not be executed).
- Action validations are triggered by the attempt to execute an action. Therefore it is not necessary to configure request nodes.
- The BOPF framework actions (e.g. create, delete, and update) can also be subjected to an action validation (e.g. action validation can prevent creation of instances with incomplete data).

Association

- An **association** is a direct, unidirectional relationship between two business object nodes.
- Associations can be used to relate two nodes in a well-defined direction. The association can be used to navigate from one node (source node) to the related node (target node).
- The associated nodes can be nodes within one business object or in different business objects (cross business object association).
- Associations can have parameters to filter the result of the related nodes.
- They can only be defined between two nodes and in one defined direction.
- Moreover, they have a defined cardinality which gives information about the existence of an association and the number of associated nodes.

BOPF offers the following two different types of associations in the internal model:

- Composite Associations: Along compositions between parent and child node (e.g. from Root Node to Item Node):
  - Regular Composition Association (node to subnode, `TO_ROOT`, `TO_PARENT`)
  - Specialization (not equal to the specialization from object orientation)
  - Association to delegated node (Dependent Object)
  - Framework Associations (Link to Framework Nodes, e.g. Property Nodes)

- General Associations: Linked across the composition tree:
  - Foreign Key Association / Reverse Foreign Key Association
  - Reverse Specialization
  - Cross-BO Association
  - Regular Association (implementation specific)

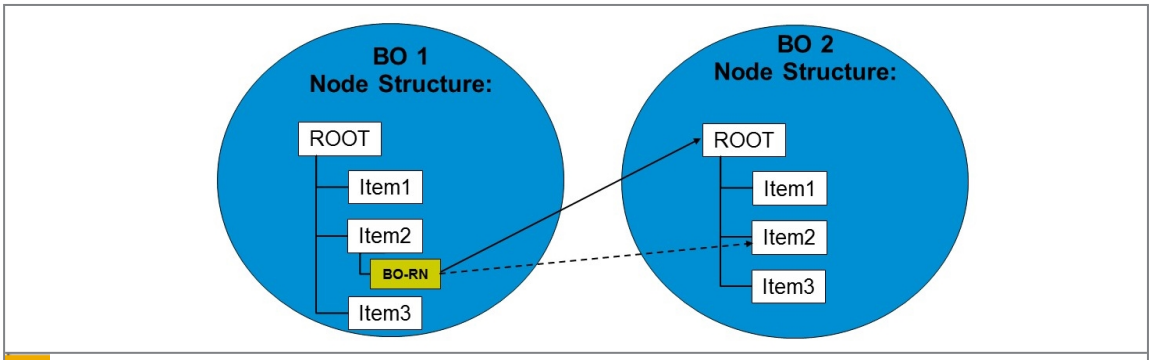


Figure 18: Cross BO Associations

Query

- A **query** is a business object entity and is always assigned to a certain node, whose instances shall be returned.
- The consumer is able to query for either only the keys or for the data.
- The query offers selection criteria.
- Queries never modify any node instance data.
- There are three types of queries:
  - Node Attribute Query (implemented by the framework; can returns the data of the query node).
  - Custom Query (implementation is application-specific; can returns the data of the query node).
  - Generic Result Query (implementation is application-specific; can return any data).

Alternative Key

- **Alternative keys** are combinations of node attributes that identify an instance of a node and have some kind of business semantics (in difference to the technical node instance key).
- Alternative keys can be either defined as unique or as non-unique.
- Both queries and alternative keys are usually used as entry point for the consumer in order to get keys of node instances. In contrast to queries, alternative key resolution can be also done on the transactional (buffer) image.

- For example, if a customer invoice instance has been created during the current transaction, it is not possible to get the key of this instance using a query but only via alternative key.
- The core service `convert_alternative_key` allows to resolve alternative key values.
- For example, it is possible to resolve the alternative key value *342* of alternative key `INVOICE_NUMBER` to the node instance key `KEY`. The result is *0050562501281DDF97848577AF34366C* and can be used in order to retrieve this invoice node instance.

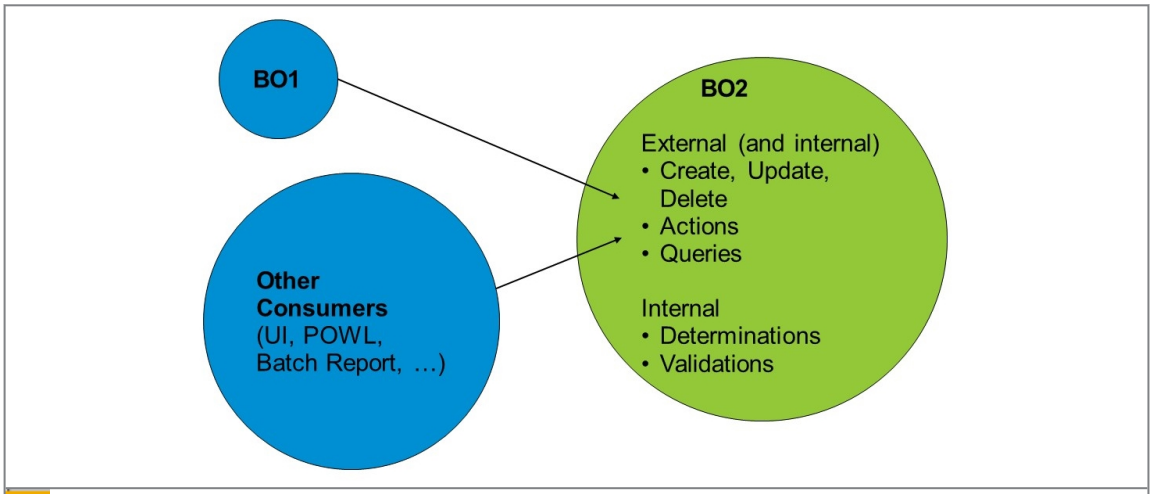


Figure 19: Business Logic

LESSON SUMMARY

You should now be able to:

- Explain and apply the concepts of BOPF
- Define the BOPF model
- Explain the BO node structure
- Explain the different BO entities

Learning Assessment

1. What is the Business Object Processing Framework?

   Choose the correct answers.

   - ☐ **A** Programming Language.
   - ☐ **B** A framework for building whole applications.
   - ☐ **C** UI Independent.
   - ☐ **D** Database.

2. Which capabilities does the Business Object Processing Framework offer?

   Choose the correct answers.

   - ☐ **A** Data access.
   - ☐ **B** Data buffering.
   - ☐ **C** Locking mechanisms.
   - ☐ **D** Authorization and property handling.
   - ☐ **E** Enhancement concept for customers and partners.

3. Which BOPF entities can modify a business object instance?

   Choose the correct answers.

   - ☐ **A** Validation
   - ☐ **B** Query
   - ☐ **C** Action
   - ☐ **D** Determination

4. How to avoid saving in case of errors in the document?

   Choose the correct answers.

   - ☐ A Action validation for save-action.
   - ☐ B Error message in an after modify determination.
   - ☐ C Failed keys from an action.
   - ☐ D Consistency Validation together with consistency groups.

5. How to call BOPF external entities like actions or retrieve / retrieve by association?

   Choose the correct answers.

   - ☐ A In a loop to handle the results easier.
   - ☐ B Only with one key in each call.
   - ☐ C Mass-enabled, as many keys as possible, not in a loop.

6. Which transaction should be used to build / enhance business objects as a customer / partner?

   Choose the correct answers.

   - ☐ A `/BOBF/CONF_UI` (or BOBF)
   - ☐ B `BOBX`
   - ☐ C `/BOBF/CUST_UI`
   - ☐ D `/BOBF/TEST_UI` (or BOBT)

Learning Assessment - Answers

1. What is the Business Object Processing Framework?

   Choose the correct answers.

   - [ ] A Programming Language.
   - [x] B A framework for building whole applications.
   - [x] C UI Independent.
   - [ ] D Database.

2. Which capabilities does the Business Object Processing Framework offer?

   Choose the correct answers.

   - [x] A Data access.
   - [x] B Data buffering.
   - [x] C Locking mechanisms.
   - [x] D Authorization and property handling.
   - [x] E Enhancement concept for customers and partners.

3. Which BOPF entities can modify a business object instance?

   Choose the correct answers.

   - [ ] A Validation
   - [ ] B Query
   - [x] C Action
   - [x] D Determination

4. How to avoid saving in case of errors in the document?

   Choose the correct answers.

   - [X] A Action validation for save-action.
   - [ ] B Error message in an after modify determination.
   - [ ] C Failed keys from an action.
   - [X] D Consistency Validation together with consistency groups.

5. How to call BOPF external entities like actions or retrieve / retrieve by association?

   Choose the correct answers.

   - [ ] A In a loop to handle the results easier.
   - [ ] B Only with one key in each call.
   - [X] C Mass-enabled, as many keys as possible, not in a loop.

6. Which transaction should be used to build / enhance business objects as a customer / partner?

   Choose the correct answers.

   - [ ] A `/BOBF/CONF_UI` (or BOBF)
   - [X] B `BOBX`
   - [ ] C `/BOBF/CUST_UI`
   - [ ] D `/BOBF/TEST_UI` (or BOBT)

UNIT 3

Floor Plan Manager (FPM) & FPM-BOPF Integration (FBI)

UNIT OBJECTIVES

- Explain the FPM and what it is used for
- Explain the basic entities of a FPM-based UI like GUIBBs, Feeder Classes and Wire Model
- Understand FBI and what it is used for
- Understand the basic steps how a FPM UI interacts with a BOPF application via FBI
- Explain the basic entities of an FBI view and their usage

Unit 3
Lesson 1

Understanding and Working with the Floor Plan Manager (FPM)

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain the FPM and what it is used for
- Explain the basic entities of a FPM-based UI like GUIBBs, Feeder Classes and Wire Model

Floor Plan Manager

The TM UI is the surface on which the user works. It is very important that this surface is clear, intuitive, and hereby fast usable. Such an UI is created by FPM, BOPF, and FBI as a framework in TM.

What is Floor Plan Manager (FPM)?

- Since SAP Transportation Management 8.0 the Floor Plan Manager (FPM) is used to realize TM User Interfaces.
- FPM is a Web Dynpro ABAP application that provides a framework for developing new Web Dynpro ABAP application interfaces consistent with the SAP UI guidelines.
- FPM allows a modification-free composition of discrete User Interface Building Blocks (UIBBs) which are compliant with the mentioned guidelines.

Why FPM?

- Rapid development and implementation of running User Interfaces that are compliant with UI guidelines.
- Flexibility to adjust the UI without the need for coding to configuration:
  - In the Transportation Management Business it is hard to define one unified UI that fulfills all customer requirements, UIs, and the process steps realized with it look different from customer to customer.
  - Enables process-specific configurations of a UI and allows building UIs independent of the representation of content in the backend application.
- Reuse of UI parts in different application areas.
- Personalization is supported.

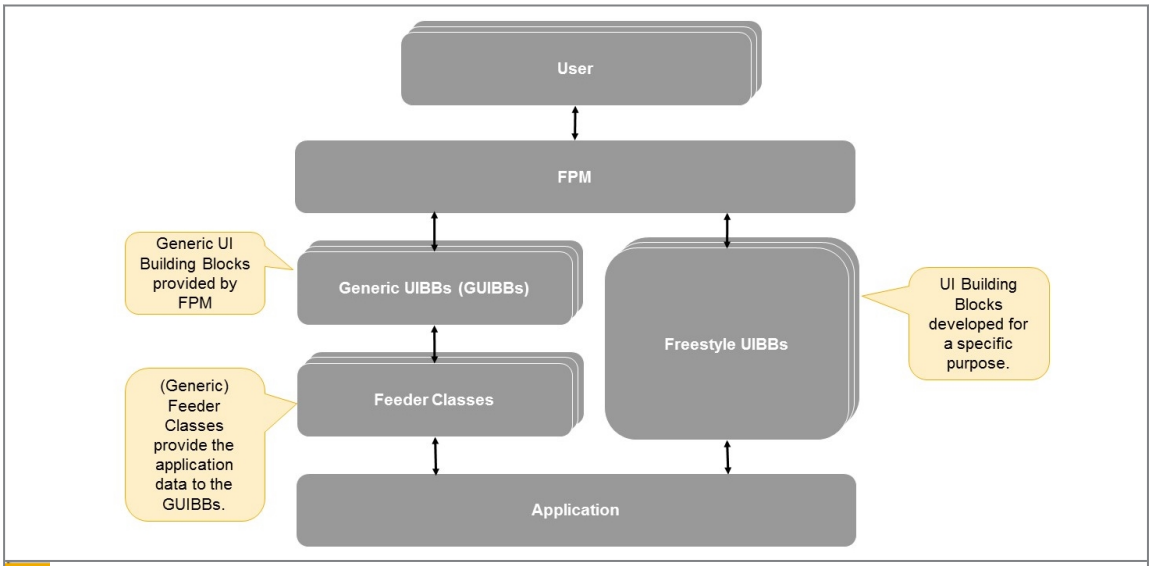


Figure 20: FPM Architecture

Basic Entities of an FPM-based UI

Building a FPM UI with Generic UI Building Blocks (GUIBBs)

- The Web Dynpro ABAP Floorplan Manager (FPM) is a framework which composes application specific views (UIBBs) to an application.
  - This allows a homogeneous high-level application structuring and interaction behavior.
- Instead of building the User Interface as an individual Web Dynpro Application, FPM centrally provides predefined UIBBs, so called Generic UI Building Blocks (GUIBBs) that can be reused to create UIBBs.
- GUIBBs used in the TM User Interface are:
  - Overview Pages (*FPM_OVP_COMPONENT*): Defines the general layout of the screens. It displays a title bar, a tool bar as well as one or more UIBBs.
  - Form GUIBB (*FPM_FORM_UIBB_GL2*): A flat collection of input elements which displays the content of a (flat) structure, must use a form-compliant Feeder Class.
  - List GUIBB (*FPM_LIST_UIBB or FPM_LIST_UIBB_ATS*): Displays the content of an (internal) table, must use a list-compliant Feeder Class.
  - Tree GUIBB (*FPM_TREE_UIBB*): Displays the content of an (internal) table in a hierarchically way, must use a tree-compliant Feeder Class.
  - Tabbed GUIBB (*FPM_TABBED_UIBB*): Used to display a tab strip including further UIBBs with an optional master UIBB on top of it, does not require a Feeder Class (sometimes misused for layout purposes which it was not designed for).
  - Composite UIBB (*FPM_COMPOSITE_UIBB*): Used for layout purposes and for putting together more complex UIBBs based on other UIBBs.

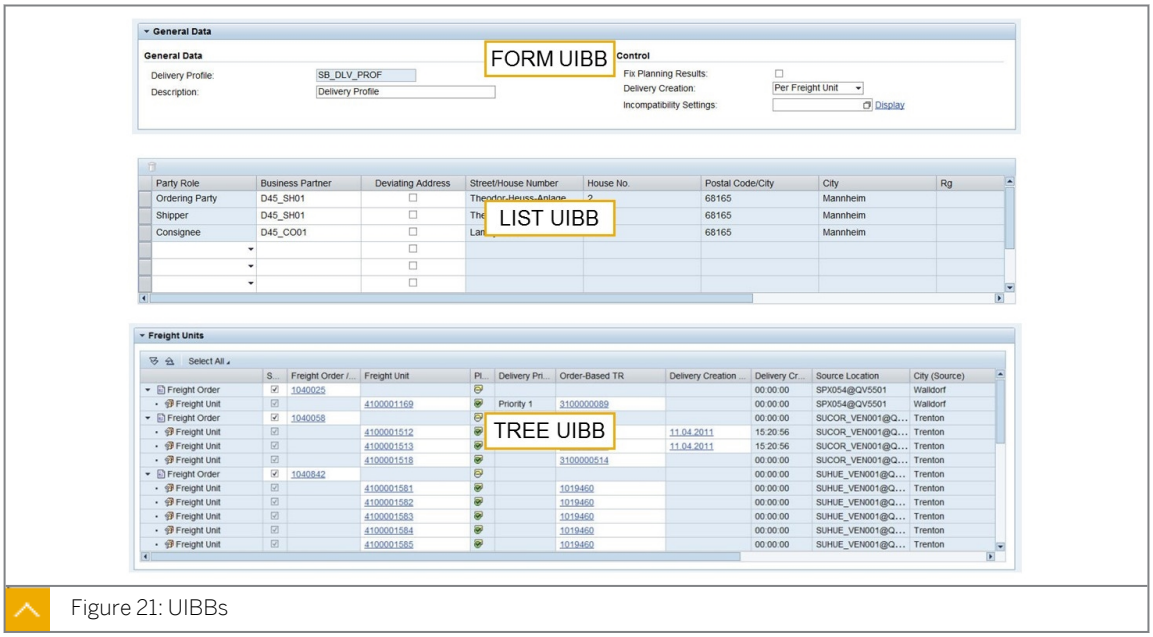


Figure 21: UIBBs

GUIBB Configuration

- **GUIBBs** are design templates for which, at design time, the application defines the data to be displayed along with a configuration.
- The composition (configuration) of those building blocks takes place in a design time application (in this case a Web Application) where all the necessary field attribute, positioning and layout properties are assigned or composed.
- The application only provides the data and a layout configuration to the GUIBBs.
- The rendering is handled by the framework itself.
- GUIBBs provide a comprehensive way of creating or changing User Interface Compositions, without the necessity to change the underlying application code base and thereby offering a concept for modification free customer UI enhancements.
- The concrete display of the data on the user interface is not determined and generated by the GUIBB until runtime. This is done automatically using the configuration provided to the GUIBB.

Feeder Class

- Necessary or mandatory application specific information will be supplied by the application itself via a **Feeder Class** implementation.
- Feeder Classes are based on a predefined interface definition providing all necessary methods and corresponding signatures for standardizing the communication between the application and the GUIBB.
- With these Feeder Classes the application:
  - Provides a field catalogue to the GUIBB design and runtime.
  - Provides the data at runtime.

  - Accepts UI changes at runtime by calling application middleware.
  - Handles user interactions (events) at runtime by calling application middleware.
  - Provides field control data to control visibility and changeability of UI elements.
- The UI Administrator or Designer can:
  - Create UI layouts as a Web Dynpro Configuration for the standard GUIBBs.
  - Put together such discrete GUIBB configurations in an application configuration.

Wire Model

- The runtime interdependencies between UIBBs are defined in a **Wire Model** by configuration entities called Wires which are based on reusable Connector Classes implementing the dependency semantics. The Wire Model is defined on the level of the Floorplan Configuration.
- A wire determines the data content of a target UIBB depending on user interaction changing the Outport of the source UIBB. Outports can be of type lead selection, selection or collection. For example, changing the lead selection in a list of Forwarding Order Items may change the data content of another list displaying the associated Item Details.
- The primary use cases for the wire model are object models with generic access interfaces like BOPF, providing standard connector classes. If the Floorplan contains composite components (tabbed components), the model UIBBs contained in the tabbed components can also be wired.

Interfaces

- With the FPM approach, it is possible to enhance application user interfaces and fit them to your business needs, based on configuration/customizing instead of modifications.
  - Besides the GUIBBs, FPM still allows the implementation and usage of freestyle UIBBs that can be realized individually to serve specific purposes that cannot be handled via GUIBBs.
- At runtime, user interactions are handled by FPM events that pass an FPM phase model (Event Loop).
- Within the FPM event loop specific methods are called that are based on a predefined interface definition and corresponding signatures in order to standardize the communication between the application and the GUIBB.
- A Feeder Class implements such a predefined interface for a specific GUIBB, for example, the interface *IF_FPM_GUIBB_FORM* for *Form* components.

The following list shows you the important Interface Methods (FPM phase model – Event Loop):

- *INITIALIZE*: Called at runtime when the form is created. It is the first feeder method which is called from FPM.
- *GET_DEFINITION*: Allows the feeder to provide all necessary information for configuring a form: the list of available fields and their properties and the list of actions (FPM events).

- *FLUSH*: The first feeder method which is called during an event loop. Whenever an FPM event is triggered (this includes all round trips caused by the form itself) this method is called. Use it to forward changed data from the form to other components in the same application.
- *PROCESS_EVENT*: Called within the FPM event loop. The *FPM PROCESS_EVENT* is forwarded to the feeder class. Here the event processing can take place and this is where the event can be canceled or deferred.
- *GET_DATA*: Called within the FPM event loop. The *FPM PROCESS_BEFORE_OUTPUT* event is forwarded to the feeder class. Here you specify the form data after the event has been processed.

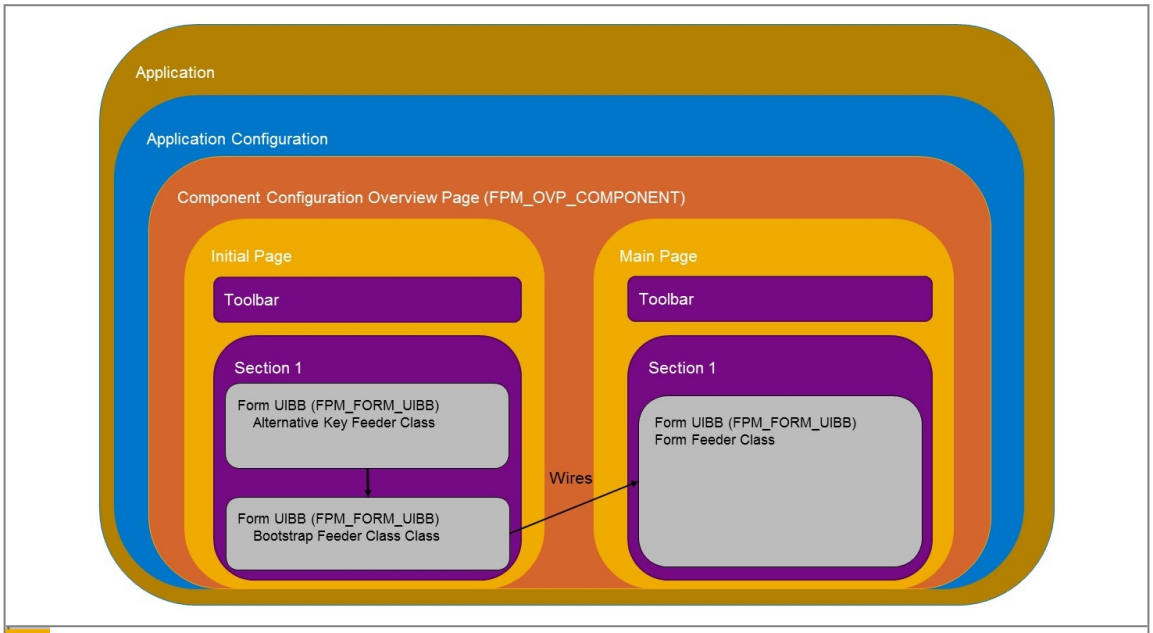


Figure 22: Application Hierarchy

Application Hierarchy

- Application Configuration - a container for Component Configuration
  - Overview Page Component Configuration (FPM_OVP_COMPONENT)
    - Pages
      - Toolbar
      - Sections
        - UIBBs
    - Wires
- UIBBs are components with configurations
  - Form UIBB: FPM_FORM_UIBB, FORM_WINDOW
  - List UIBB: FPM_LIST_UIBB, LIST_WINDOW
  - Tree UIBB: FPM_TREE_UIBB, TREE_WINDOW

- Each UIBB needs to be configured, which is a Component Configuration in itself

The following figure shows the Overview Page, it contains: Identification Region, Main Toolbar, Content Blocks, and Wiring of Content Blocks. Other Content Areas like Initial Screen, Pop-ups, and Edit Pages.

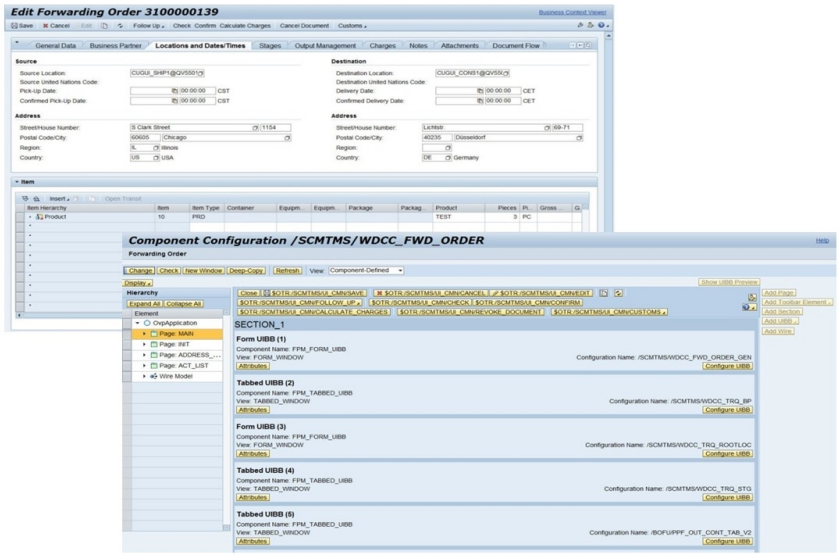


Figure 23: Overview Page

LESSON SUMMARY

You should now be able to:

- Explain the FPM and what it is used for
- Explain the basic entities of a FPM-based UI like GUIBBs, Feeder Classes and Wire Model

Unit 3
Lesson 2

Understanding and Working with the FPM-BOPF Integration (FBI)

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Understand FBI and what it is used for
- Understand the basic steps how a FPM UI interacts with a BOPF application via FBI
- Explain the basic entities of an FBI view and their usage

Floor Plan Manager BOPF Integration (FBI)

What is FBI?

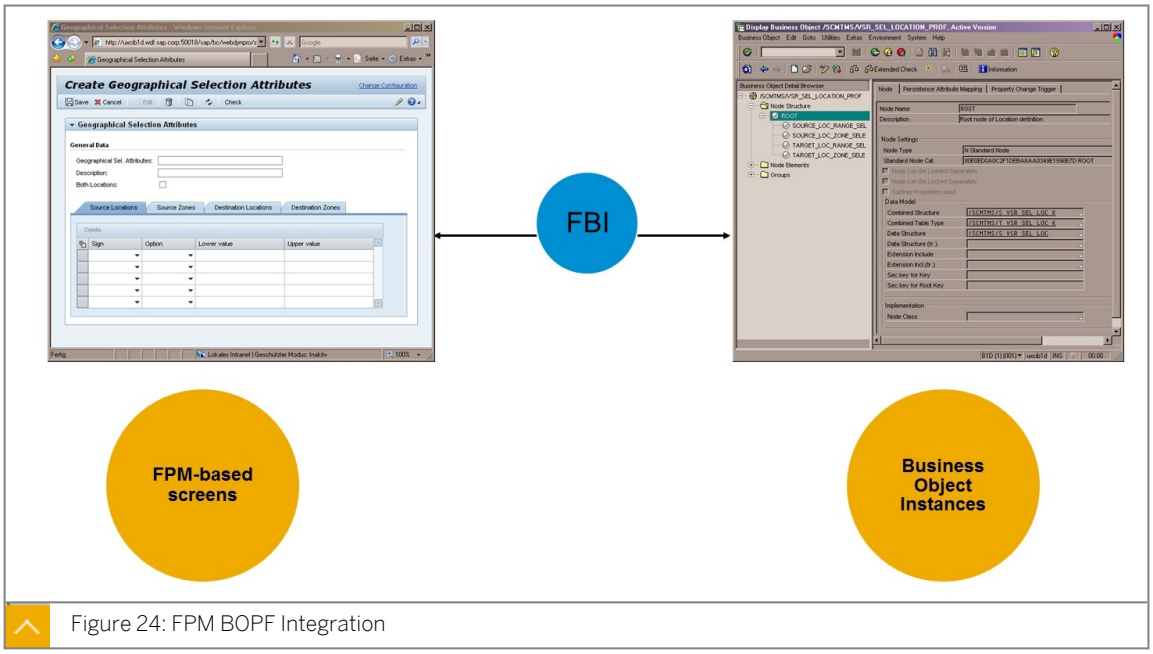


Figure 24: FPM BOPF Integration

Floor Plan Manager BOPF Integration (FBI)

- The Floor Plan Manager BOPF Integration (FBI) is used to integrate FPM with the BOPF-based Business Objects.
- FBI provides generic FPM application Feeder Classes together with the relevant application configuration that allows consuming services of Business Objects modeled in BOPF. These BOPF services can be used seamlessly in a modification-free UI environment.

Why FBI?

- Rapid development and implementation of running User Interfaces by reusing generic Feeder class implementations provided by FBI.
- FBI provides the following functionalities that support the communication and corporation between FPM applications and BOPF-based Business Objects.
  - Editing data of BO node instances in the standard GUIBBs FORM and LIST.
  - Accepting action parameter values and invoking corresponding actions on BO node instances.
  - Overview Search (OVS) based on BO node queries.
  - Input of external IDs on initial screens and subsequent conversion of these external IDs into internal (technical) IDs (Alternative Key Conversion).
  - UI-specific services are supported: Navigation to multiple targets - Calling dialog boxes and editing application data in these dialog boxes - Support of UI-specific non-BOPF actions.

FPM UI Interaction with a BOPF Application via FBI

FBI Feeder Classes

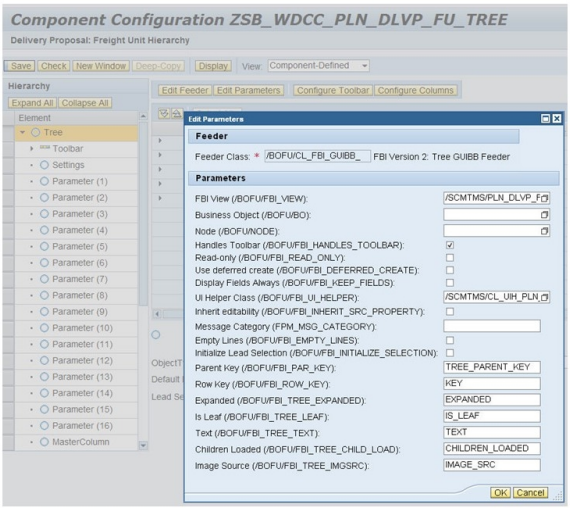


Figure 25: FBI Feeder Classes

Each FPM Component Configuration needs to be assigned a Feeder Class to define the communication with the back-end. FBI provides specific Feeder Classes to communicate with BOPF.

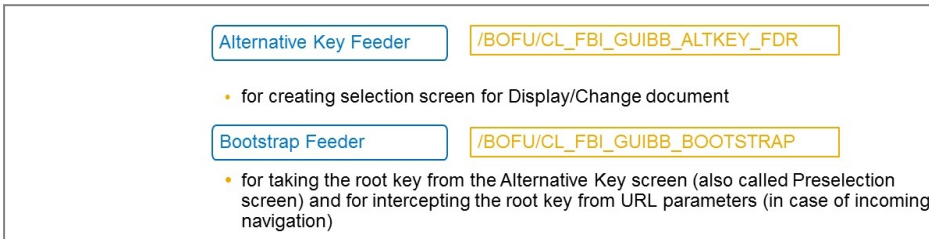


Figure 26: Initial Page

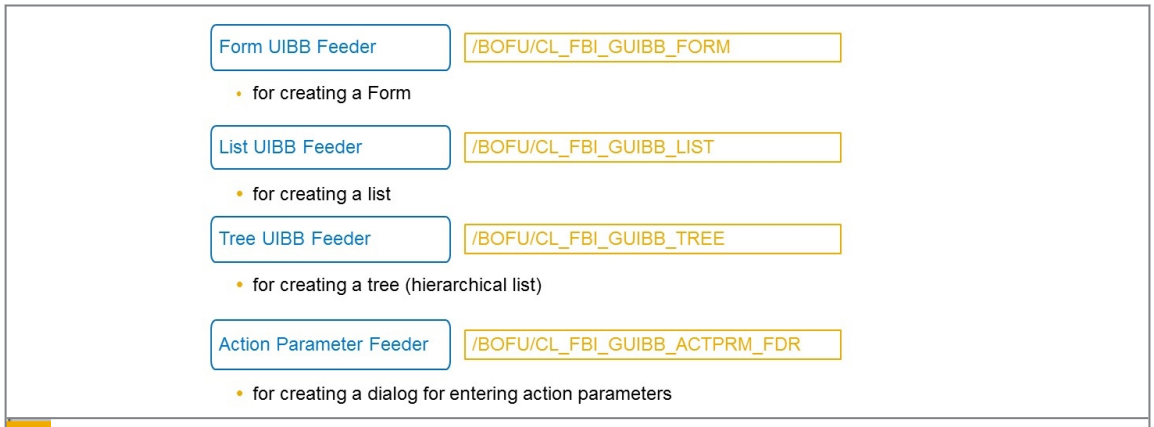


Figure 27: Main Page

UI Structure

To define the fields that will be available to be displayed on the screen for a given BO Node. For example UI Structure for *TRQ → ROOT* .

UI Structure needs a Key of type */bobf/conf_key* (for internal processing) along with the necessary fields.

Example of a Typical UI Structure for TRQ Header Data:

- Key (Mandatory field)
- Fields available for display:
  - TRQ ID (Will be available to be displayed on screen)
  - TRQ Category (Will be available to be displayed on screen)
  - Sales Organization

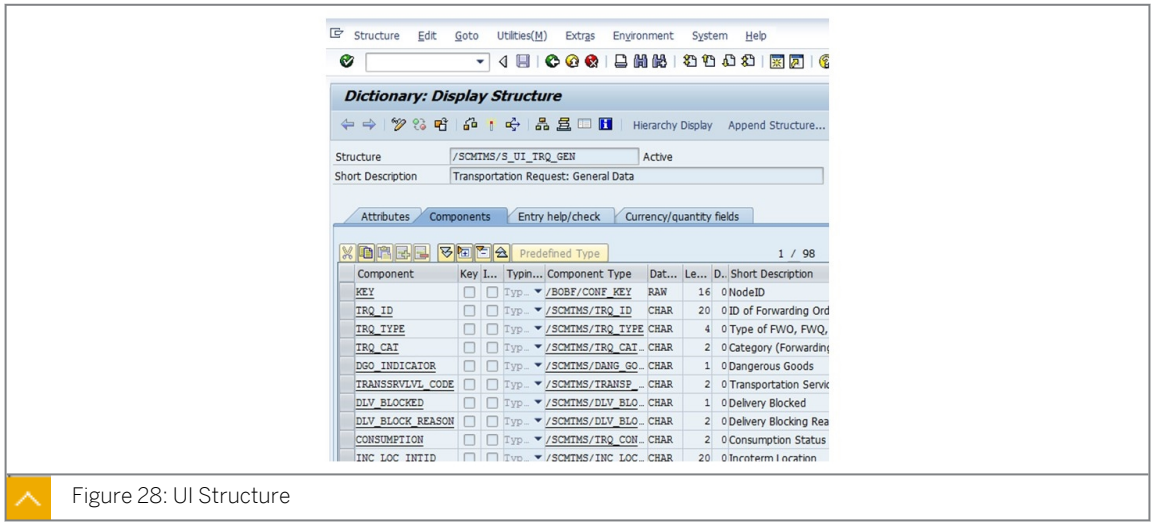


Figure 28: UI Structure

FBI View Entities

FBI Views (Design Time)

The following list shows you the most important parameter of a generic FBI feeder:

- Specifies with which BO node the data exchange should occur.

- Specifies the UI Structure.
- Specifies how the data is to be converted/mapped between BO and UI.
  - View Mapper class (Conversion class).
- Specifies how the data is to be adjusted during roundtrips.
  - Exit Interface class (Viewexit class).
- Specifies Field Descriptions and Field Groups.
- Actions that are not related to the BO are also defined in the FBI View.
- Related Views can be used to include data from multiple BO nodes into a single UI structure.
- FBI views are stored as a configuration of Web Dynpro Component */BOFU/FBI_VIEW*.

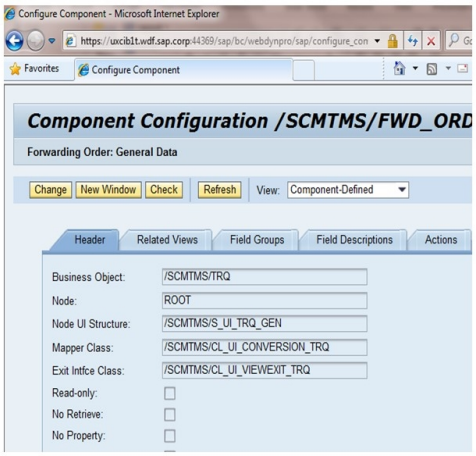


Figure 29: FBI Feeder

Header:

- Contains the mandatory part: Business Object and Node.
- Optional UI Structure (if not specified, then node structure is used).
- Optional Mapper Class (if not specified, MOVE-CORRESPONDING is used).
- Optional Exit Interface Implementation Class (details later).
- Additional settings, like Read-Only, etc.

Related Views (optional):

- Allows the definition of a chain of Views to read data from more nodes (for example, when information coming from several nodes shall be combined into on flat UI structure to be displayed in a list).
- Each related view is included with a mandatory suffix.

Field Descriptions (optional):

- Can be used to specify additional properties for structure attributes.
- These settings are passed to the FPM field catalogue.

- For example, Sorting Allowed, Allow Filter, Domain Fixed Values, Fixed Values, F4-Values from Code Value List etc.

Actions (optional):

- Allows definition of new Event IDs.
- For the new Event IDs, as well as for the existing ones (Standard FBI and BO Actions, which are taken into account automatically), you can specify additional settings:
  - Set specific Name and Tooltip (via OTR aliases) – they will be passed to the FPM action catalogue.
  - Specify another Event ID, whose enable/disable properties are to be inherited.
  - Specify whether the action is allowed to be triggered in read only mode.
  - Specify whether the action is allowed to be executed only when a record is selected.
  - Specify navigation target (in this case, FBI calls Navigation Class instead of Standard handling).

FBI Views (Runtime)

- FBI Views (Runtime):
  - The instances of an FBI View hold the keys of the displayed instances (coming from the wires that UIBBs are connected with).
  - An instance, for example prepares modifications, executes actions and posts change notification to the controller.
  - It reacts to the FBI-specific SYNCUP Event. That is, it evaluates change notifications and determines which of the keys must be refreshed.
  - Moreover it reads the data from node buffers or from the BO layer in case of modified keys.
  - Where required it also calls conversion classes for the modified records (e.g. to convert a document ID into its corresponding technical key).
  - A view instance calls available Exit methods at the appropriate places.

FBI Controller

- FBI Controller:
  - The FBI controller is responsible to do the orchestration between FBI View instances and the BO layer.
  - It does not contain any application logic but only provides the technical framework for the orchestration.
  - It provides a Modification Buffer for changes done on the UI that are then forwarded from there to the BO layer, i.e. it centralizes the BO Layer responses.
  - It also collects the change notifications coming from the BO layer that then need to trigger updates on the UI.

- Further buffers hold the information on the nodes read from the BO layer and the properties of nodes and their attributes.
- The properties determine e.g. whether an attribute is a mandatory field or is ready for input. Moreover, these buffers help to avoid redundant BOPF service calls.

Conversion Classes

- When data is send from the BO layer to the UI, the **conversion class** is called to convert technical attributes into their clear text representation.
- The same conversion class is also called when data is send from the UI back to the BO layer, i.e. it converts clear text information in to its technical representation.
- Conversions are done immediately after retrieval of data and shortly before sending modifications to the buffer. A conversion class is specified in the FBI View definition.
- Example Conversion: Mapping rule for Date-Time Conversion:
  - The conversion rule maps a field of type *TIMESTAMP* into its Date, Time and Time Zone part.
  - A BO node attribute *FIELD* of type *TIMESTAMP* is automatically converted with this rule if the UI structure contains the attributes *FIELD_D (Date)*, *FIELD_T (Time)* and *FIELD_TZ (Time Zone)*.
  - The conversion from the three attributes back into a timestamp is also done when required - bidirectional mapping.

Conversions are done immediately after retrieval of data and shortly before sending modifications to the buffer. A conversion class is specified in the FBI View definition. Implementations of Conversion Classes do always inherit from TM super class */SCMTMS/CL_UI_CONVERSION*. Each redefinition of the super class must define its own mapping table in method *BUILD_MAP_TABLE*. The super class already contains a few generic (bidirectional) mapping rules which are based on field naming conversions:

- **Mapping rule for Date-Time Conversion**: The conversion rule maps a field of type *TIMESTAMP* into its Date, Time and Time Zone part. A BO node attribute *FIELD* of type *TIMESTAMP* is automatically converted with this rule if the UI structure contains the attributes *FIELD_D (Date)*, *FIELD_T (Time)* and *FIELD_TZ (Time Zone)*.
- **Mapping rule for Date-Time Conversion into String**: The conversion rule maps a field of type TIMESTAMP into a String. A BO node attribute *FIELD* of type *TIMESTAMP* is automatically converted with this rule if the UI structure contains the attribute *FIELD_TTT* (Formatted Date).
- **Mapping Rule for Alternative Key Conversion**: The conversion rule maps a BO node foreign instance key into its corresponding foreign readable ID. For this, it uses the BO key, the BO node name and the alternative key for this node defined in the node Meta Data.
- **Mapping Rule for Code List Conversion**: The conversion rule maps a BO code into a readable UI code value. A BO attribute *FIELD* with its code value X is converted into its readable UI code value if the UI structure contains the attribute *FIELD_TXT*.

Exit Classes

- The generic feeder classes provided by FBI usually take care of all communication aspects between the corresponding GUIBB and the application.
- Nevertheless there might be use cases that require a more specific implementation.
- For this, an Exit Class can be specified in the FBI View definition.
- Exit Classes provide many extension options and are the recommended means for adapting the standard FBI processing to customers and partner's needs.
- An Exit Class implements a predefined FBI Exit Interface.
- The following is an example Method of an Exit Class: *ADAPT_EVENT*. Intercept and process any event that arrives in the underlying FBI view. If custom event IDs were added to the FBI View, this is the place to implement the action handling for them.

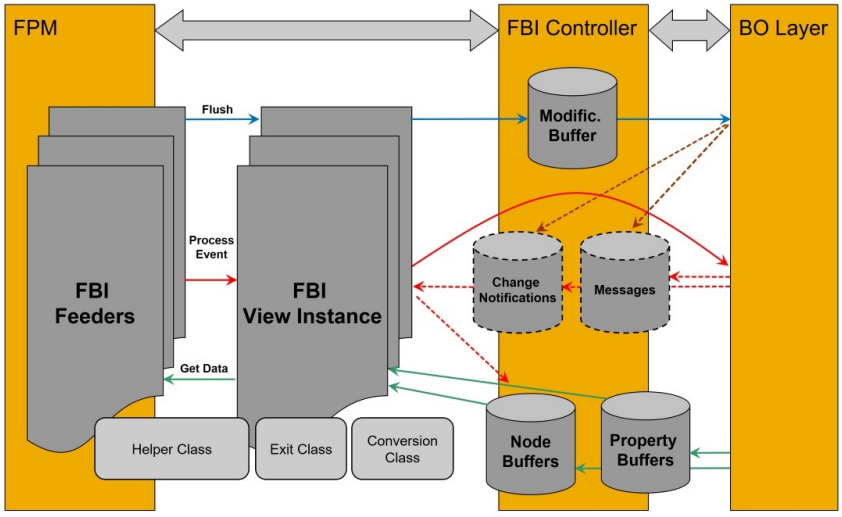


Figure 30: FBI Architecture

At runtime, user interactions are handled by FPM events that pass an FPM phase model (Event Loop). Within the FPM event loop specific methods are called that are based on a predefined interface definition and corresponding signatures in order to standardize the communication between the application and the GUIBB. A Feeder Class implements such a predefined interface for a specific GUIBB, e.g. the interface IF_FPM_GUIBB_FORM for Form components. Important methods are:

- *INITIALIZE*: Called at runtime when the form is created. It is the first feeder method which is called from FPM.

  GET_DEFINITION: Allows the feeder to provide all necessary information for configuring a form: the list of available fields and their properties and the list of actions (FPM events).

- *FLUSH*: The first feeder method which is called during an event loop. Whenever an FPM event is triggered (this includes all round trips caused by the form itself) this method is called. Use it to forward changed data from the form to other components in the same application.
- *PROCESS_EVENT*: Called within the FPM event loop. The FPM PROCESS_EVENT is forwarded to the feeder class. Here the event processing can take place and this is where the event can be canceled or deferred.

- *GET_DATA*: Called within the FPM event loop. The FPM PROCESS_BEFORE_OUTPUT event is forwarded to the feeder class. Here you specify the form data after the event has been processed.

Note that FBI has its own Buffer for Node Information, Field Properties, etc. to gain performance, i.e. it does not have to read this information from BOPF again every time it needs to access it.

Transient BOPF Integration

Transient UI – BOPF Integration (TBI) is a TM-specific concept for implementation of more complex UI Building Blocks, which cannot be covered by the generic Floorplan Manager BOPF Integration (FBI).

TBI can handle the following field-specific data sources:

- The *forwarding order* hierarchical level - from TRQ Root.
- The freight unit level - from TOR (FU) Root.
- The freight document level - from TOR (FO/BO) Root and TOR Stop.

Hierarchy	Document Type	Source Location	Source Location Address	Departure Date	Departure Time	Departure Timezone	Dest. Location	Dest. Location Address	Arrival Date	Arrival Time	Arrival Timezone
Forwarding Order 100111	FWO	W-001	Warehouse Walldorf	01.07.2015	10:00:00	CET	C-666	Jane Doe / Boston	15.07.2015	16:00:00	EST
Freight Unit 501234	0001										
Truck FO 201234	1001	W-001	Warehouse Walldorf	02.07..2015	08:00:00	CET	DEHAM	Port Hamburg	02.07.2015	17:50:00	CET
Ocean Booking 15	BSEA	DEHAM	Port Hamburg	03.07.2015	15:00:00	CET	USNEK	Port Newark	10.07.2015	16:00:00	EST
US-Truck FO 301234	1002	USNEK	Port Newark	11.07.2015	08:00:00	EST	C-666	Jane Doe / Boston	11.07.2015	12:30:00	EST
Freight Unit 505678	0001										
Truck FO 205678	1001	W-001	Warehouse Walldorf	02.07..2015	16:00:00	CET	DEHAM	Port Hamburg	03.07.2015	08:00:00	CET
Ocean Booking 15	BSEA	DEHAM	Port Hamburg	03.07.2015	15:00:00	CET	USNEK	Port Newark	10.07.2015	16:00:00	EST
US-Truck FO 305678	1002	USNEK	Port Newark	11.07.2015	07:00:00	EST	C-666	Jane Doe / Boston	11.07.2015	11:30:00	EST

TBI provides a set of feeder classes and interfaces for the application-specific logic.

Note:
SAP note 2548763 - TM: Developer's cookbook for Transient BOPF Integration (TBI).

LESSON SUMMARY

You should now be able to:

- Understand FBI and what it is used for
- Understand the basic steps how a FPM UI interacts with a BOPF application via FBI
- Explain the basic entities of an FBI view and their usage

Learning Assessment

1. Which GUIBBs are provided by FPM?

   Choose the correct answers.

   - ☐ **A** Tree
   - ☐ **B** List
   - ☐ **C** Form
   - ☐ **D** Composite

2. Which functionalities / concepts are provided by FBI?

   Choose the correct answers.

   - ☐ **A** BOPF-specific feeder class implementations
   - ☐ **B** Views
   - ☐ **C** Connector classes
   - ☐ **D** Events

3. Is FBI a TM-specific framework?

   Choose the correct answer.

   - ☐ **A** Yes, FBI is only used by TM.
   - ☐ **B** No, FBI is also used by other applications.

4. Which suffixes support an automatic conversion?

   Choose the correct answers.

   - ☐ **A** "_LOC" for converting a location ID into its address.
   - ☐ **B** "_D" for converting a timestamp into a date.
   - ☐ **C** "_TXT" for converting a code value to its description.
   - ☐ **D** "_USR" for converting a username into its forename and surname.

Learning Assessment - Answers

1. Which GUIBBs are provided by FPM?

   Choose the correct answers.

   - [X] A Tree
   - [X] B List
   - [X] C Form
   - [X] D Composite

2. Which functionalities / concepts are provided by FBI?

   Choose the correct answers.

   - [X] A BOPF-specific feeder class implementations
   - [X] B Views
   - [X] C Connector classes
   - [ ] D Events

3. Is FBI a TM-specific framework?

   Choose the correct answer.

   - [ ] A Yes, FBI is only used by TM.
   - [X] B No, FBI is also used by other applications.

4. Which suffixes support an automatic conversion?

   Choose the correct answers.

   - [ ] A "_LOC" for converting a location ID into its address.
   - [X] B "_D" for converting a timestamp into a date.
   - [X] C "_TXT" for converting a code value to its description.
   - [ ] D "_USR" for converting a username into its forename and surname.

UNIT 4 Personal Object Work List (POWL)

Lesson 1

UNIT OBJECTIVES

- Understand and work with Personal Object Work List (POWL)

Unit 4
Lesson 1

Understanding and Working with Personal Object Work List (POWL)

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Understand and work with Personal Object Work List (POWL)

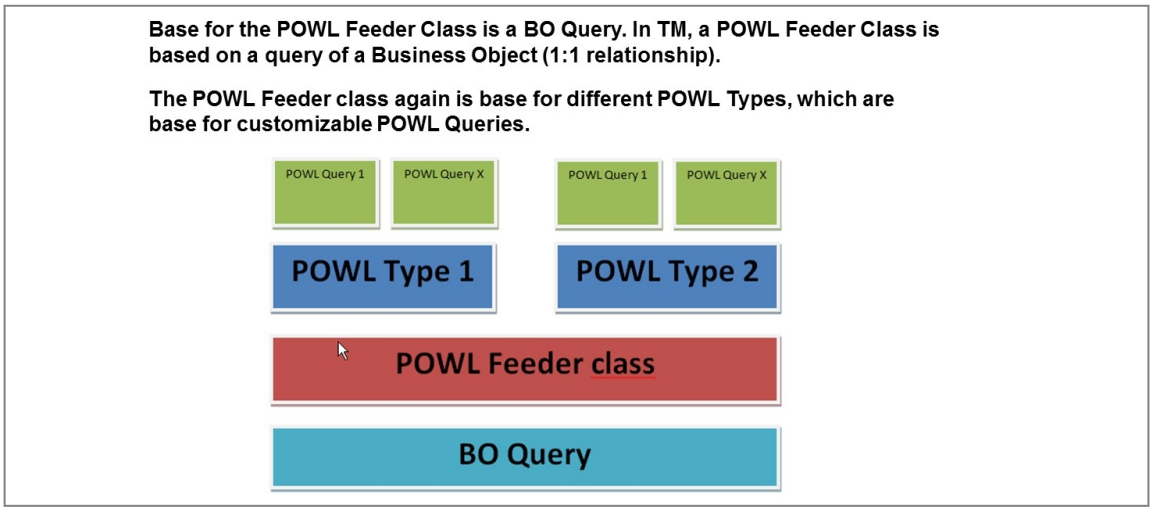
Personal Object Work List

Personal Object Work List (POWL) is a tool that provides central personalized access to objects in the lists. It gives users a portal-based interface that they can use to define (and optionally store) queries in a similar way as they are used to from SAP GUI selection screen variants.

POWL:

- List of specific Business Objects
- Homogenous User Interface
- Elementary enabler for Object-Based Navigation (OBN)
- Personalizable by end users
- Simplifies business processes
- Web Dynpro for ABAP based framework
- Developers need pure ABAP knowledge

The POWL Feeder Class



The main access point for a POWL is the POWL Feeder Class. It contains the definition of the POWL's selection criteria, the field catalog (i.e. the result structure) and the actions that can be executed from the POWL Toolbar for a selected set of object instances from the POWL result list. In TM a POWL Feeder Class is usually based on a query of a Business Object.

The relationship between a Business Object query (a Generic Result Query) and a POWL Feeder class is always one to one. So whenever there is no POWL Feeder Class making use of an existing Generic Result Query of a BO you will need a new POWL Feeder Class. You should also keep this in mind when creating BO Queries for POWL usage. If you have two completely different requirements leading to different BO Queries, of course two different POWL Feeder Classes are required.

If you have different business categories or usages for POWLs which share many common parts form a technical perspective, it makes sense to design one (technical) BO query which has one common technical POWL Feeder Class in the end. Based on this feeder class, you can separate your different usages or categories with corresponding POWL types.

An example is the TOR Feeder class */SCMTMS/CL_UI_POW_FD_TOR*, with separate POWL types for each BO Category. This example POWL Feeder Class provides selection criteria, result attributes and actions depending on the TOR Category, e.g. TO for Freight Order or FU for Freight Units.

How is a POWL implemented?

- Each POWL is associated with a so-called **POWL Feeder Class** that contains the coding for providing the data and the invocation of actions configured to be available on the toolbar of a POWL. Example Class: */SCMTMS/CL_UI_POW_FD_TRQ*.
- The *CONSTRUCTOR* method of this class contains the definition of various POWL elements:
  - Name and node of the BO that provides the data.
  - The Generic Result Query (BOPF Query) that is used to query the data.
  - The output structure, i.e. the list of fields that will be displayed in the result table of a POWL.
  - The select structure, i.e. the list of selection criteria that is available to define and execute POWL Query executions.
  - Action Class name: The assigned Action Class contains the coding to execute the functions and logic associated with the POWL toolbar buttons. Example class: */SCMTMS/CL_UI_ACTION_TRQ*.
  - You can implement your very own POWLs by implementing a new POWL Feeder Class (should inherit from class */SCMTMS/CL_UI_POW_FD_BASE*) or you enhance existing POWLs
- The underlying Generic Result Query can be your own implemented or an existing one. In the latter case you can add additional selection criteria and result attributes by the introduced Query Enhancement concept.
- In case you need to enhance existing Feeder and Action Classes you need to work with implicit enhancements, there is no explicit enhancement concept.
- New POWLs require further customizing.

POWL Type

A POWL Type is always required and is based on a feeder class. If you need adjustments which require coding changes in the feeder class, a different POWL Type is needed.

Changes inside feeder class coding (new POWL type is needed):

- Display/Hide columns in result table
- Rename column headers (e.g. Airport vs. Port)
- Enable/Disable selection criteria
- Rename labels for selection criteria (e.g. Airport vs. Port)
- Change the action toolbar (different buttons)
- Default selection parameters (e.g. Mode of Transport = Air)

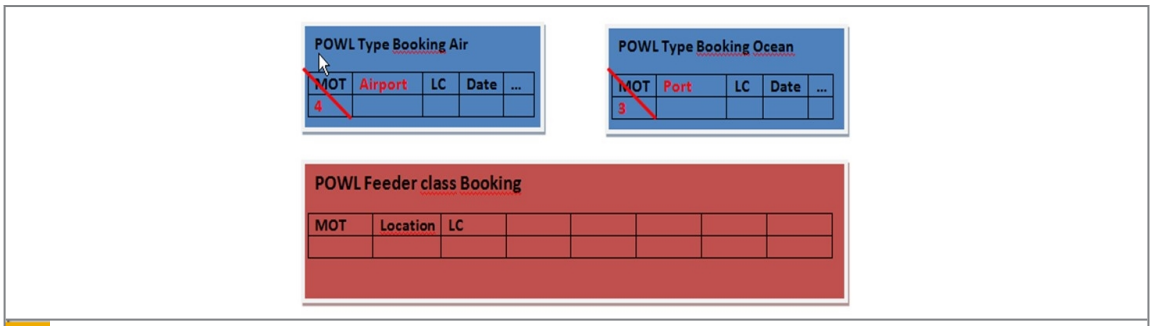


Figure 31: POWL Type

The figure is an example of Freight Booking POWLs. Here for each mode of transport exists an extra POWL Type. So it is possible to call one time a technical field *location* one time *harbor* and another time *airport*. From a technical point of view it is the same BO Query and the same attribute. An example for separated selection ranges are also the Bookings. Basically the BO Query can select all Bookings independent of the mode of transport. It is possible now to set a default parameter depending on a POWL Type (one time Air Bookings one time Ocean Bookings). This default parameter is hard coded and later not visible for the POWL admin and user. So it is not possible to create a query based on a POWL Type for Air Booking which selects Ocean Bookings. The POWL Type is the base for different queries and offers a kind of super set with selection attributes and fields for the result list and a fixed defined toolbar.

POWL Query

A POWL Query is customizing and based on a POWL Type. POWL offers the possibility to create different queries based on an existing POWL Type. A POWL Query can differ in the following elements:

- Selection Criteria (+Quick Criteria Maintenance)
- Values of Selection Criteria (+Quick Criteria Maintenance)
- Visible fields in result list
- Order of result list fields,
- Sort options and all other ALV features
- Refresh behavior (only manual refresh, every list visit, etc...)

In case of the freight booking POWL a POWL Query might be enough if you would like to see a list of all Air Booking with a Life Cycle Status *New*. Furthermore the life cycle shall be hidden

for the customer. Another example for the query would be a list with all Air Bookings which was created in the last 30 days.

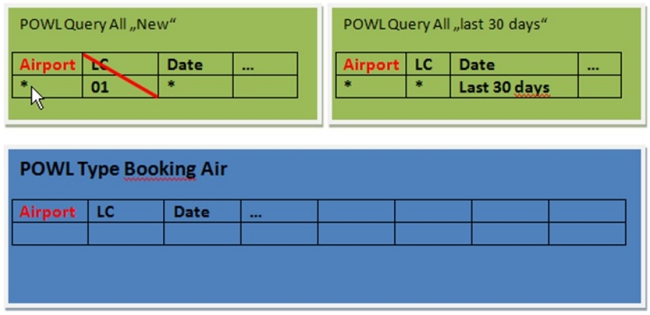


Figure 32: POWL Query

Define New Query or Change Query

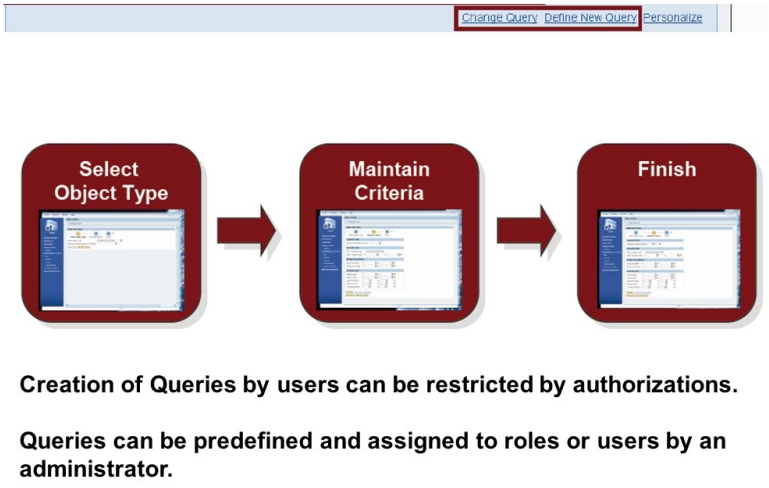


Figure 33: Define or Change a Query

The following steps show you how to Create a POWL:

1. Define BO Query.
2. Define Selection Criteria (Data Dictionary Structure).
3. Define Result Structure for result table columns (Data Dictionary Structure).
4. Feeder Class to register the BO/Node, Selection criteria, Result Structure, BO Query, Action Class (ABAP OO Class).
5. Create POWL Type – name to be assigned to the Feeder Class (Customizing).
6. Assign POWL type to a User Role (Customizing).
7. Assign POWL type to User (Customizing, optional).
8. Define new Query using the POWL Type.
9. Create POWL Action Handler Class, register in Feeder Class, define Action Buttons (ABAP OO Class, optional).

Enhancing BOPF Queries

- The **BOBF Enhancement Workbench** does not support enhancing existing standard BO queries.
  - It is for example not possible to assign a query to the ROOT node, which returns instances or the keys of the related ITEM node.
  - The only way to extend queries in the standard BOPF environment is to add further query node attributes to the query structure.

The following list shows you how to enhance standard BOPF Queries:

- All queries of SAP TM are derived from a super class: Class */SCMTMS/CL_Q_SUPERCLASS*.
- This super class contains an enhancement mechanism which makes SAP TM queries extensible in a unified way. The mechanism is based on the query enhancement table ("QET") */SCMTMS/C_QENH*.
- The query super class provides an API for creating an optimized database *SELECT* statement to execute the query and returning the requested result table. At runtime the super class is creating an optimized database *SELECT* statement from the content of the query structure and the query enhancement table. –
- The meta data for the query enhancement can be maintained via report */SCMTMS/MAINT_QUERY_ENH*.
- See Enhancement Guide Document for a detailed example of Enhancing BOPF Queries.

Enhancing Personal Object Work Lists (POWLs)

- POWLs are used throughout the whole SAP TM Application to access documents and the related data, not only for displaying but also for processing documents.
- Customers require the POWLs to also contain their custom fields to be included in the search criteria list of a POWL as well as in the result list.
- The following list shows you how to enhance POWLs:
  - Each POWL gets its data from a BOPF Generic Result Query.
  - The query used for a POWL must be identified and can be enhanced with the introduced enhancement concept for queries.
  - The other steps require coding enhancements to bring the additional fields into the list of search criteria and result displayed by the POWL.
  - Many customers have decided to implement their completely own POWLs with the POWL Framework instead of adjusting the standard POWLs.
  - In case of a newly created POWL, also customizing is required.

Note:
See Enhancement Guide Document for a detailed example on how to enhance existing POWLs and how to create a new POWL from scratch.

LESSON SUMMARY

You should now be able to:

- Understand and work with Personal Object Work List (POWL)

Learning Assessment

1. Which POWL entity can you use in your POWL feeder class to control your implementation?

   Choose the correct answers.

   - ☐ A POWL type
   - ☐ B POWL category
   - ☐ C POWL query

2. Which information must be at least provided if you create your own POWL with your own feeder based on the base class */SCMTMS/CL_UI_POW_FD_BASE*?

   Choose the correct answers.

   - ☐ A BO name
   - ☐ B BO query key
   - ☐ C POWL output structure
   - ☐ D List of actions

3. TM standard queries can be enhanced by:

   Choose the correct answers.

   - ☐ A Customer-specific enhancement coding
   - ☐ B BAdI implementation
   - ☐ C Customizing

Learning Assessment - Answers

1. Which POWL entity can you use in your POWL feeder class to control your implementation?

   Choose the correct answers.

   - [X] A POWL type
   - [ ] B POWL category
   - [ ] C POWL query

2. Which information must be at least provided if you create your own POWL with your own feeder based on the base class */SCMTMS/CL_UI_POW_FD_BASE*?

   Choose the correct answers.

   - [X] A BO name
   - [X] B BO query key
   - [X] C POWL output structure
   - [ ] D List of actions

3. TM standard queries can be enhanced by:

   Choose the correct answers.

   - [X] A Customer-specific enhancement coding
   - [X] B BAdI implementation
   - [X] C Customizing

UNIT 5 Business Objects Enhancements

Lesson 1

UNIT OBJECTIVES

- Explain and apply BOBF Enhancement Features
- Explain the basic steps how to enhance BOPF BOs

Unit 5
Lesson 1

Enhancing BOPF Business Objects

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain and apply BOBF Enhancement Features
- Explain the basic steps how to enhance BOPF BOs

BO Enhancements

In a SAP TM implementation, it might be necessary to add custom fields and custom functionalities to the TM system. Since business documents and business logic are implemented in BOPF Business Objects, BOBF Enhancement features are used to do so.

The BOPF Business Object Configurator for enhancing standard BOs and creating custom BOs can be executed using transaction `BOBX`.

Features and functions:

- SAP Standard Development release BOs, Nodes and Actions to be extensible.
- Existing Standard BO Nodes are enhanced with persistent or transient fields via DDIC.
  - Enhancement Include available on all released nodes to add append structure.
- New Subnodes can be added to the BO Model.
- New Determinations can be added to BO Nodes.
  - Restriction: They always run only after all standard Determinations have been executed. New Actions can be added to BO Nodes.
- Pre- and Post Action Enhancements can be defined.
- New Action & Consistency Validations can be defined.
  - Restriction: They always run only after all standard Validations have been executed.
- Add new Queries (not extending existing ones -> own TM concept used instead)
- Own partner/customer specific Business Objects can be created.

LESSON SUMMARY
You should now be able to:

- Explain and apply BOBF Enhancement Features
- Explain the basic steps how to enhance BOPF BOs

Learning Assessment

1. How can you enhance business objects?

   Choose the correct answer.

   - ☐ **A** Business objects cannot be enhanced at all.
   - ☐ **B** Business objects can be enhanced using transaction `BOBX`.
   - ☐ **C** Business objects can be enhanced using transaction */BOBF/CONF_UI*.

2. Is it possible to enhance all business objects?

   Choose the correct answer.

   - ☐ **A** Yes, you can enhance all business objects.
   - ☐ **B** No, you can only enhance business objects which are marked as “can be enhanced”.
   - ☐ **C** No, you can only enhance business object of category “standard business object”.

3. It is possible to define a customer-specific determination so that the determination is executed before the standard determinations?

   Choose the correct answer.

   - ☐ **A** Yes, you can simply define the standard determination as “dependent determination” for your customer-specific determination.
   - ☐ **B** Yes, but only for ROOT determinations.
   - ☐ **C** Yes, but only for after-modify determination.
   - ☐ **D** No, it’s not possible. Customer-specific determinations are always executed after the standard determinations.

4. Is it possible to create multiple “business object enhancements” for the same BO?

   Choose the correct answer.

   ☐ A Yes, but only the latest added enhancement is active.

   ☐ B No, only one active enhancement is allowed per BO.

   ☐ C Yes, this is possible. The individual elements of each “business object enhancements” are merged.

5. How do you reference / consume customer-specific enhancements in your coding?

   Choose the correct answer.

   ☐ A The element keys / names are available in a “business object enhancement”-specific constants interface.

   ☐ B The element keys / names are available in constants interface of the standard BO.

   ☐ C The element keys / names must be hard-coded, they are not available in any constant interface.

Learning Assessment - Answers

1. How can you enhance business objects?

   Choose the correct answer.

   - [ ] A Business objects cannot be enhanced at all.
   - [x] B Business objects can be enhanced using transaction `BOBX`.
   - [ ] C Business objects can be enhanced using transaction */BOBF/CONF_UI*.

2. Is it possible to enhance all business objects?

   Choose the correct answer.

   - [ ] A Yes, you can enhance all business objects.
   - [x] B No, you can only enhance business objects which are marked as “can be enhanced”.
   - [ ] C No, you can only enhance business object of category “standard business object”.

3. It is possible to define a customer-specific determination so that the determination is executed before the standard determinations?

   Choose the correct answer.

   - [ ] A Yes, you can simply define the standard determination as “dependent determination” for your customer-specific determination.
   - [ ] B Yes, but only for ROOT determinations.
   - [ ] C Yes, but only for after-modify determination.
   - [x] D No, it’s not possible. Customer-specific determinations are always executed after the standard determinations.

4. Is it possible to create multiple “business object enhancements” for the same BO?

   Choose the correct answer.

   - [ ] A Yes, but only the latest added enhancement is active.
   - [ ] B No, only one active enhancement is allowed per BO.
   - [X] C Yes, this is possible. The individual elements of each “business object enhancements” are merged.

5. How do you reference / consume customer-specific enhancements in your coding?

   Choose the correct answer.

   - [X] A The element keys / names are available in a “business object enhancement”-specific constants interface.
   - [ ] B The element keys / names are available in constants interface of the standard BO.
   - [ ] C The element keys / names must be hard-coded, they are not available in any constant interface.

UNIT 6 UI Enhancements

Lesson 1

UNIT OBJECTIVES

- Explain and apply FPM/FBI Enhancement Features
- Explain the basic steps how to enhance FPM/FBI UIs

Unit 6
Lesson 1

Enhancing FPM/FBI User Interfaces

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain and apply FPM/FBI Enhancement Features
- Explain the basic steps how to enhance FPM/FBI UIs

FPM/FBI Enhancement Features

In a SAP TM implementation, it might be necessary to add custom fields and custom functionalities to the TM system. To provide these fields and functionalities on the UI, it is necessary to make use of FPM/FBI enhancement capabilities.

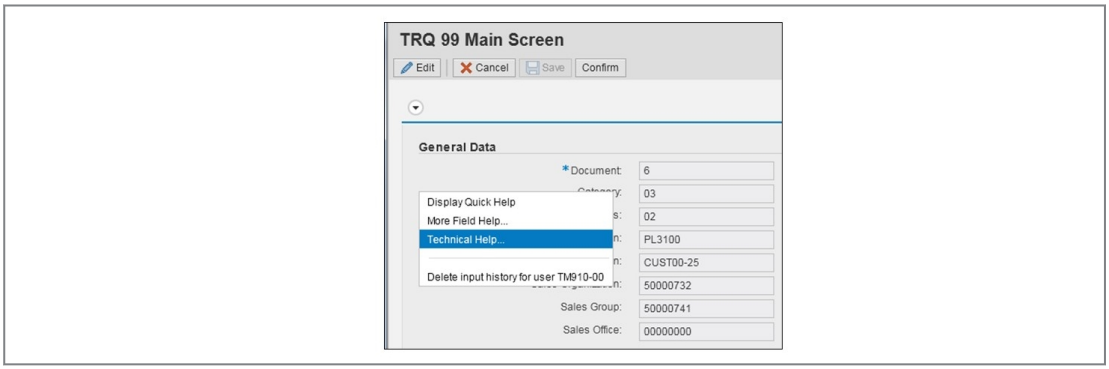
Component Customizing

- TM user interfaces are built with the help of Floor Plan Manager (FPM) and the Floor Plan Manager BOBF Integration (FBI). These two frameworks enable enhancing a user interface via configuration rather than having to implement additional code.
- Basic enhancements ideally can be done without any coding.
- For more complex user interfaces and enhancements, coding might be required. For example, implementation of the *Exit Class Methods* mentioned in the last section.
- Each Component Configuration can be enhanced by partners and customers by creating a **Component Customizing**.
- The standard component configuration will remain untouched.
- Component Customizing can be created. For example, in a development system and get transported to a test or production system.
- Component Customizing can also be deleted again. Afterwards, the original standard component configuration is in place again to define the corresponding user interface.

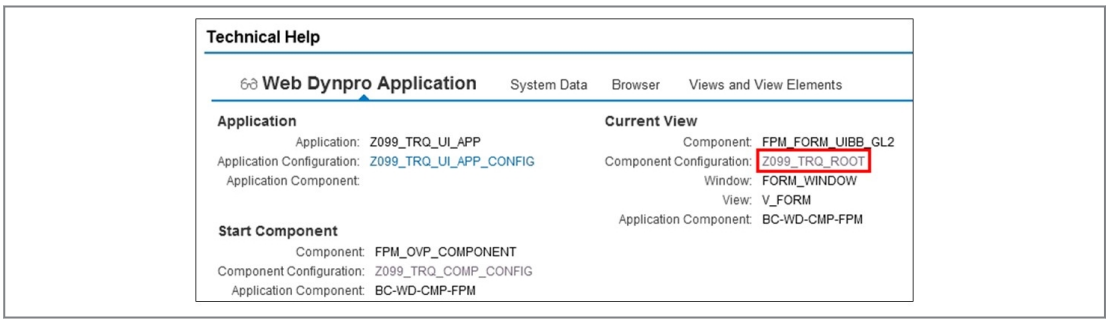
Enhance FPM/FBI UIs

To Create Component Customizing

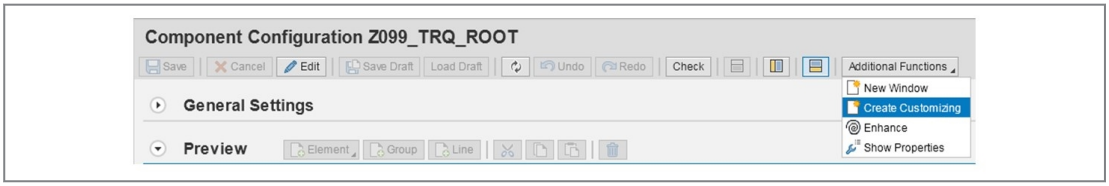
1. To create a Configuration Customizing, just open the *Component Configuration*. You have two options, open the Technical Help from the context menu in the UI itself.



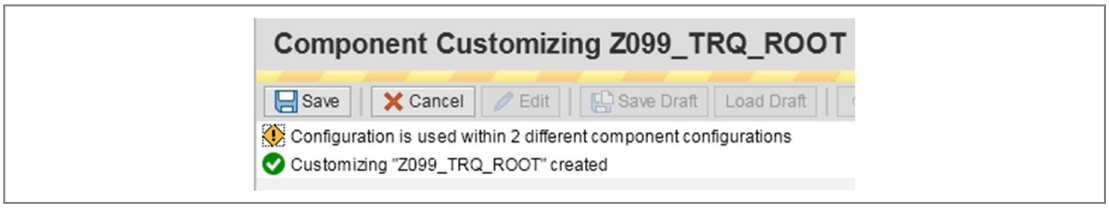
2. From the popup, open the *Component Configuration*.



3. From the Component Configuration menu, choose *Create Customizing*.



4. This will take you to the *Customizing Mode*.



5. Proceed with the configuration as usual (for example, you can add fields).

LESSON SUMMARY

You should now be able to:

- Explain and apply FPM/FBI Enhancement Features
- Explain the basic steps how to enhance FPM/FBI UIs

Learning Assessment

1. How can you enhance an FPM-based UI?

   Choose the correct answer.

   - ☐ A Customers / partners cannot enhance the standard UIs.
   - ☐ B Standard UIs can be enhanced only via modification.
   - ☐ C Standard UIs can be enhanced by customizing.

2. To enhance an FPM-UI do you always need a workbench transport request?

   Choose the correct answer.

   - ☐ A Yes, because you always have to enhance the UI structure.
   - ☐ B No, a customizing transport request can be sufficient.
   - ☐ C Yes, because you always need to write coding.
   - ☐ D Yes, because you must regenerate the BO constants interface.

3. Can you add customer-specific UIBBs to a standard OVP?

   Choose the correct answer.

   - ☐ A No, because the OVP cannot be enhanced.
   - ☐ B No, because you cannot add new wires to the standard OVP.
   - ☐ C Yes, you can also enhance an OVP and create wires.

4. Why must you be careful when enhancing standard FPM-UIs via customizing?

   Choose the correct answer.

   - ☐ A Because UI enhancements are client-independent and therefore active in all clients.
   - ☐ B Because UI enhancements are user-independent and therefore active for all users.
   - ☐ C Because UI enhancements cannot be undone.
   - ☐ D Because UI enhancements are lost when upgrading the system.

Learning Assessment - Answers

1. How can you enhance an FPM-based UI?

   Choose the correct answer.

   - [ ] A Customers / partners cannot enhance the standard UIs.
   - [ ] B Standard UIs can be enhanced only via modification.
   - [x] C Standard UIs can be enhanced by customizing.

2. To enhance an FPM-UI do you always need a workbench transport request?

   Choose the correct answer.

   - [ ] A Yes, because you always have to enhance the UI structure.
   - [x] B No, a customizing transport request can be sufficient.
   - [ ] C Yes, because you always need to write coding.
   - [ ] D Yes, because you must regenerate the BO constants interface.

3. Can you add customer-specific UIBBs to a standard OVP?

   Choose the correct answer.

   - [ ] A No, because the OVP cannot be enhanced.
   - [ ] B No, because you cannot add new wires to the standard OVP.
   - [x] C Yes, you can also enhance an OVP and create wires.

4. Why must you be careful when enhancing standard FPM-UIs via customizing?

   Choose the correct answer.

   - [ ] A Because UI enhancements are client-independent and therefore active in all clients.
   - [X] B Because UI enhancements are user-independent and therefore active for all users.
   - [ ] C Because UI enhancements cannot be undone.
   - [ ] D Because UI enhancements are lost when upgrading the system.

UNIT 7 Business Add-Ins & Implicit Enhancements

Lesson 1

UNIT OBJECTIVES

- Understand what BAdIs are and what they are used for
- Explain how to find BAdIs in TM for a specific purpose
- Understand how to use BAdIs to influence the TM business logic
- Explain the basic steps how to create a BAdI-Implementation
- Understand what Implicit Enhancements are and what they can be used for
- Explain the critical aspects of Implicit Enhancements
- Understand how Implicit Enhancements influence the TM business logic
- Explain the basic steps how to create an Implicit Enhancement

Unit 7
Lesson 1

Implementing Business Add-Ins (BAdIs)

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understand what BAdIs are and what they are used for
- Explain how to find BAdIs in TM for a specific purpose
- Understand how to use BAdIs to influence the TM business logic
- Explain the basic steps how to create a BAdI-Implementation
- Understand what Implicit Enhancements are and what they can be used for
- Explain the critical aspects of Implicit Enhancements
- Understand how Implicit Enhancements influence the TM business logic
- Explain the basic steps how to create an Implicit Enhancement

Business Add-In (BAdI)

Business Add-Ins in SAP Transportation Management

During a SAP TM implementation, it might required to add custom functionalities to the TM system. To meet these requirements, BOPF enhancements might not always be feasible. BAdIs and Implicit Enhancements are alternative enhancement techniques.

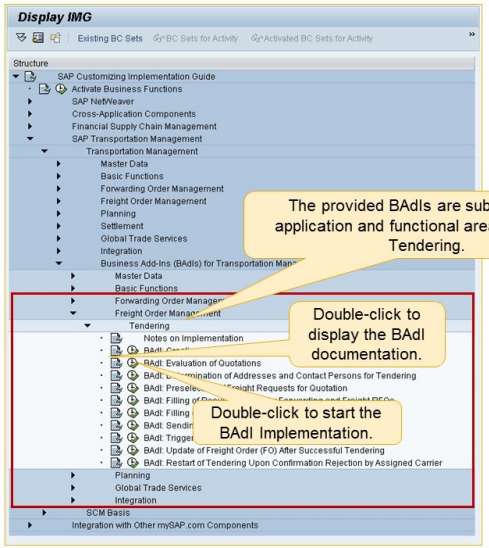
- What is BADi?
  - The Business Add-In (BAdI) concept is SAP's object-oriented plug-in concept for ABAP.
  - BAdIs are used to plug in custom behavior either in an additive way or by replacing the standard behavior.
- Why BAdIs?
  - BAdIs are a mechanism for planned extensibility.
  - Planned means that the developer of the standard software already anticipates that others may want to change or enhance the standard behavior at certain points in the application.
  - BAdI Interfaces represent a *contract* between SAP and customers/partners. That is, they have to be kept stable for at least two years and are not allowed to be changed in an incompatible way.
  - BAdIs represent a modification free concept to enhance business logic at predefined points in a process.

- The creation of a BAdI is motivated by a business reason.
- It allows customers and partners to realize a required business behavior at a defined point of a process in the application.
- As input and output, a BAdI's interface should comprise only the attributes/structures which are relevant for its defined business purpose.
- There are usually no standard application logic shipped via BAdIs and their implementation. Example implementations can be provided to showcase the BAdIs usage.

Find Business Add-In (BAdI)

How to Find BAdIs in SAP TM

- In SAP Transportation Management, more than 150 BAdIs are available. The provided BAdIs represent fixed extension points that can be used to realize/change different behavioral aspects of the standard application.
- Via Transaction `SPRO` (Customizing), the following available BAdIs can be found:
  - *SPRO → SAP Transportation Management → Transportation Management → Business Add-Ins (BAdIs) for Transportation Management → (subarea of interest)*.
  - From here, the BAdI documentation and the example implementation (if available) can be reviewed.
  - An implementation can be triggered.
- Some of the available BAdIs come with an example implementation to give customers, partners, and developers a draft idea on how to make use of the BAdI.



- Transaction SE18 allows to search for available BAdIs: search for BAdIs starting with /SCMTMS/*.
- Transaction `SE18` can be used to navigate to the example implementation of a selected BAdI (if available).

- Enter the BAdI name and choose *Display*.
- Navigate to the Implementations of the BAdI.

- Transaction `SE19` allows to search for available BAdIs implementations.

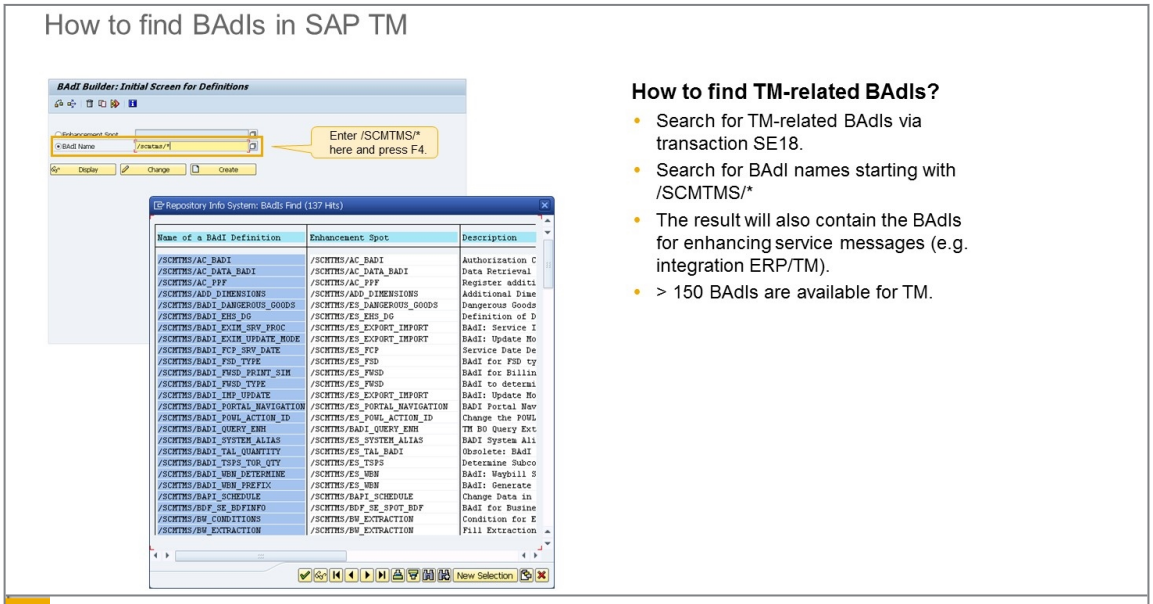


Figure 34: Find BAdIs

Business Add-In (BAdI) Work Modes

For some of the BAdIs provided with SAP TM it is possible to define a **BAdI work mode** that controls whether a BAdI method is executed at runtime and how it interacts with the SAP TM standard business logic.

The customer or partner can decide whether a BAdI method bypasses or enhances the standard business logic. The work modes can be set independently for each method within a BAdI. For example, a customer wants to execute method A of a BAdI, but not method B.

There are three BAdI work modes. The following figures shows all three modes.

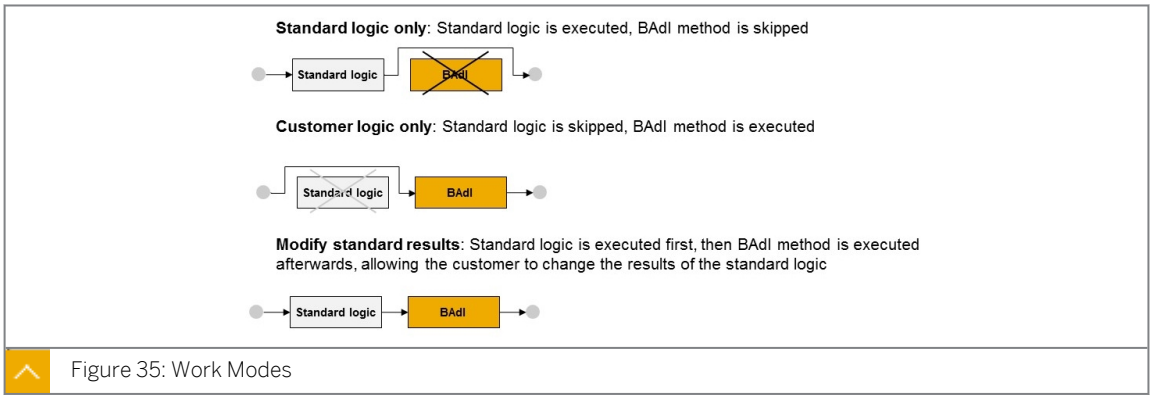


Figure 35: Work Modes

Business Add-In (BAdI) Implementation

Case 1: BAdI interface contains only a single functional method (e.g. FILL_RESULT_STRUCTURE)

```
DATA:
  lo_badi          TYPE REF TO /scmtms/tend_g_req_resp_result,
  lv_badi_work_mode  TYPE /scmtms/badi_work_mode.

* Get BAdI and BAdI work mode
 GET BADI lo_badi.

 lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_standard_logic_only.

 IF lo_badi IS BOUND.
  CALL BADI lo_badi->/scmtms/if_common_badi~set_badi_work_mode
    CHANGING
     cv_work_mode = lv_badi_work_mode.
 ENDIF.

* ---------------------------------------------------------------------
* Execute standard logic first
* ---------------------------------------------------------------------
 IF lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_standard_logic_only OR
    lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_modify_stand_results.

*  [Standard logic]

 ENDIF.

* ---------------------------------------------------------------------
* Execute customer logic (BAdI)
* ---------------------------------------------------------------------
 IF lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_customer_logic_only OR
    lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_modify_stand_results.

  CALL BADI lo_badi->fill_result_structure
    EXPORTING
     [...]
    CHANGING
     [...]

 ENDIF.
```

← Pre-set the BAdI work mode

← Customer sets the work mode by implementing this method. Use changing parameter CV_WORK_MODE in this case!

← Implement standard logic first

← Call BAdI method after standard logic

Figure 36: Case 1

Case 2: BAdI interface contains multiple functional methods (e.g. ADD_PREFERRED_TSP and PROPOSE_PRICE_LIMIT)

```
DATA:
  lo_badi          TYPE REF TO /scmtms/tend_a_crt_proposal,
  lt_badi_work_mode  TYPE /scmtms/t_badi_work_mode.

 GET BADI lo_badi.

 IF lo_badi IS BOUND.
  CALL METHOD /scmtms/cl_common_badi_tools=>fill_badi_work_mode_table
    EXPORTING
     io_badi_instance = lo_badi
    IMPORTING
     et_badi_work_mode = lt_badi_work_mode.

  CALL BADI lo_badi->/scmtms/if_common_badi~set_badi_work_mode
    CHANGING
     ct_work_mode = lt_badi_work_mode.
 ENDIF.

 READ TABLE lt_badi_work_mode ASSIGNING <ls_badi_work_mode>
      WITH KEY intf_method_name = 'ADD_PREFERRED_TSP'.
 IF sy-subrc = 0.
  lv_badi_work_mode = <ls_badi_work_mode>-badi_work_mode.
 ELSE.
  lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_standard_logic_only.
 ENDIF.

 IF lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_standard_logic_only OR
    lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_modify_stand_results.
*  [Standard logic]
 ENDIF.

 IF lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_customer_logic_only OR
    lv_badi_work_mode = /scmtms/if_common_badi=>gc_mode_modify_stand_results.

  CALL BADI lo_badi->add_preferred_tsp
    [...]

 ENDIF.
```

Example

INTF_METHOD_NAME	BADI_WORK_MODE
ADD_PREFERRED_TSP	MODIFY_STANDARD_RESULTS
PROPOSE_PRICE_LIMIT	STANDARD_LOGIC_ONLY

← Prepare the work mode table

← Customer sets the work mode for each functional BAdI method. Pass the work mode table LT_BADI_WORK_MODE to the method.

← Get the work mode for a specific BAdI method, e.g. ADD_PREFERRED_TSP

← Implement standard logic first

← Call BAdI method after standard logic

Figure 37: Case 2

Implicit Enhancement

- **What is Implicit Enhancement?** For ABAP programs, a number of implicit enhancement options exist, for example:
  - At the end of an include
  - At the end of a structure definition (types, data, constants, statics)
  - At the start and at the end of a method or function module
  - At the start and at the end of a method or function module
- Implicit Enhancements can be created via transaction `SE80`. That is, they are fully integrated into the ABAP Workbench.

- **Why Implicit Enhancements?** The mentioned options provide very powerful means to alter standard code.
- In some cases, there are no other ways to enhance, for example when adding a type or data definition.
- Can be used to realize coding adjustments in case there is no other enhancement technology available for the given situation (e.g. BAdIs, Conditions, etc.).

- Implicit Enhancements are build into the ABAP language and are available without further declarations.
- Enhancement options are:
  - New data fields can be added to existing structures in the definition of data, types or constants.
  - Arbitrary code can be injected at specific locations, such as start or end of a procedure (form subroutine, function module or method).
  - Function Group Enhancements allow adding parameters to the interface of function modules.
  - ABAP objects classes or interfaces can be enhanced:
    - By adding new fields or methods.
    - Methods can also be enhanced with parameters, and pre or post methods that are executed before or after the original method.
    - It is also possible to enhance an existing class by replacing an existing method implementation with a new one (overwrite).

Critical Aspects of Implicit Enhancements

- SAP strongly recommends to use BAdIs wherever possible, since they provide a defined interface.
- Implicit Enhancements should be used with care! The following aspects should be kept in mind when making use of this enhancement technique.
- Detailed knowledge on the application code is required for identifying the objects to be enhanced for a specific purpose.
- In case of methods that are not part of a stable interface, the signature can potentially change.
- This can lead to problems in case a pre- or post-method implementation relies on parameters from the methods signature, especially when parameters might have been removed.
- Enhancement SPAU might become necessary after updates to analyze conflict situations related to your Enhancement Implementations.
- In case of overwriting methods by copying the code of a standard method and adjusting it within an overwrite method implementation, you will not get the changes / corrections for the standard portion of your implementation.

TM Business Logic Influence

1. The following steps show you how to Implicit Enhancement Points. Navigate to the class or method that will be extended by the implementation of its Implicit Enhancement Points and display the class.
   a. Transaction `SE24` can be used, in case the class is already known. Transaction `SE80` can be used to navigate to the class. `BOBX` can be used to navigate to the implementing class of BO node elements like Determinations or Validations of the BO to be enhanced.
2. Double-click the method that will be extended to display its current implementation.
3. Click the *create / change* button for Enhancement Implementation in the toolbar of the Class Builder.
4. Display the Implicit Enhancement Points of the method, follow the menu path *Edit → Enhancement Options → Show Implicit Enhancement Options*.
5. Position the cursor on the Pre- (or the Post-Exit).
6. Right-click the context menu. Follow the menu path *Enhancement Implementation → Create Implementation*. Alternatively, choose *Edit → Enhancement Options → Create Enhancement*. Again: The cursor must be placed on the Pre- or Post-Exit at this time.
7. On the following popup, choose the Type of Enhancement, Declaration, and Code.
8. On the next popup, enter the name of the *Enhancement Implementation* and a short description and choose *Enter*. Ensure that the extension code is free of syntax errors. Ensure that you do not place coding here that causes inconsistencies or other critical situations. Use with care.
9. Place your extension declaration or coding into the Enhancement Implementation section that has opened up.
10. Save and activate the Enhancement Implementation.
11. Test the Enhancement Implementation carefully to make sure that the changed behavior does not cause inconsistencies or other critical situations.

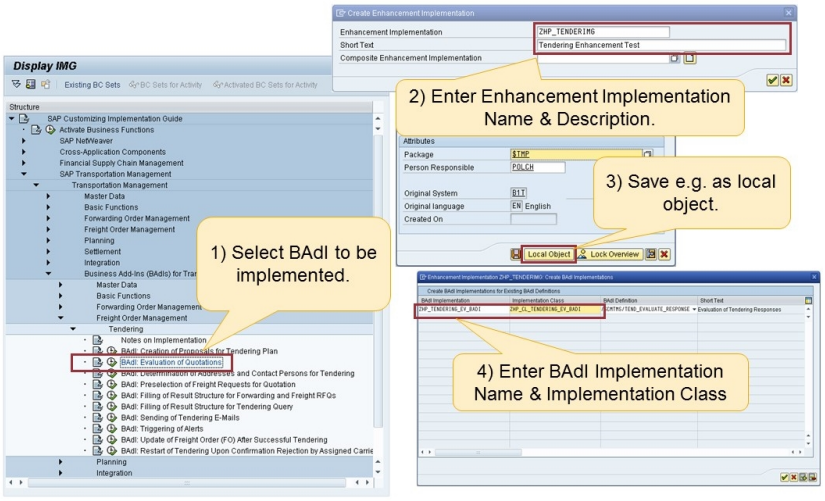


Figure 38: Tools & Techniques to extend Business Logic 1

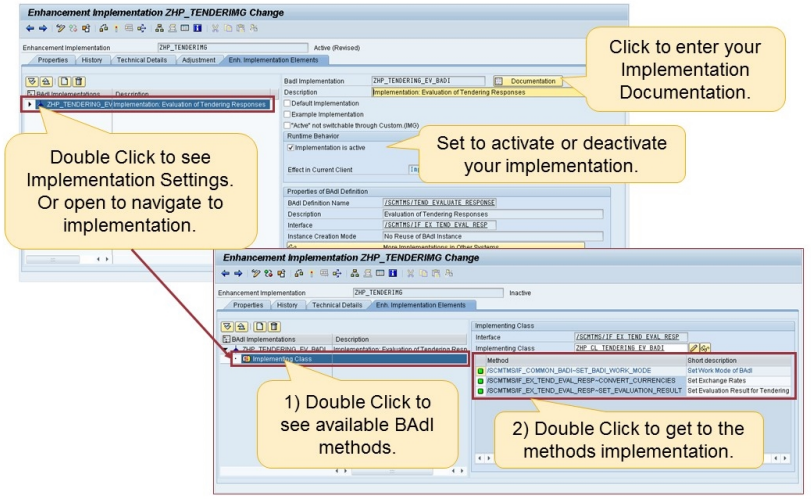


Figure 39: Tools & Techniques to extend Business Logic 2

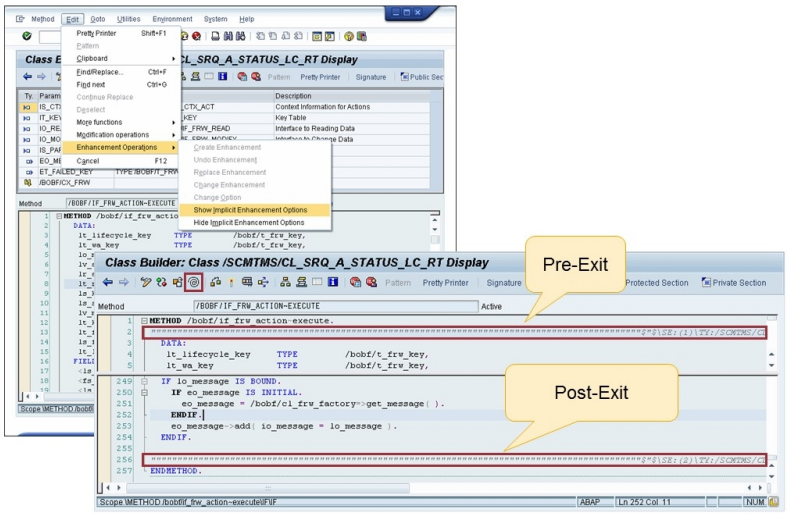


Figure 40: Tools & Techniques to Extend Business Logic 3

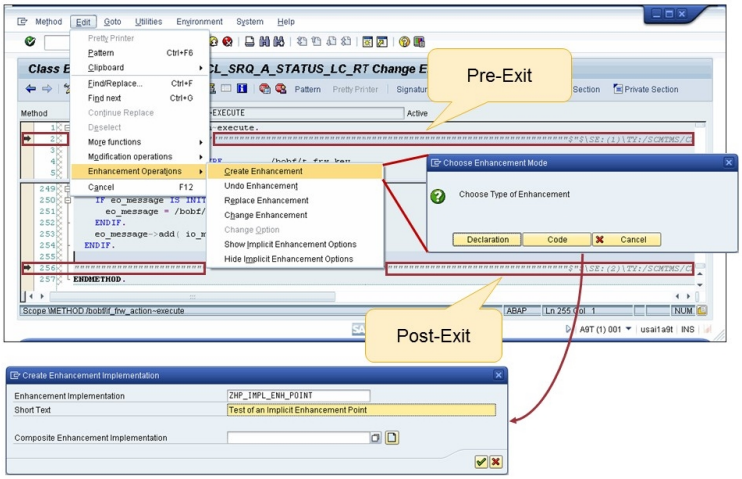


Figure 41: Tools & Techniques to Extend Business Logic 4

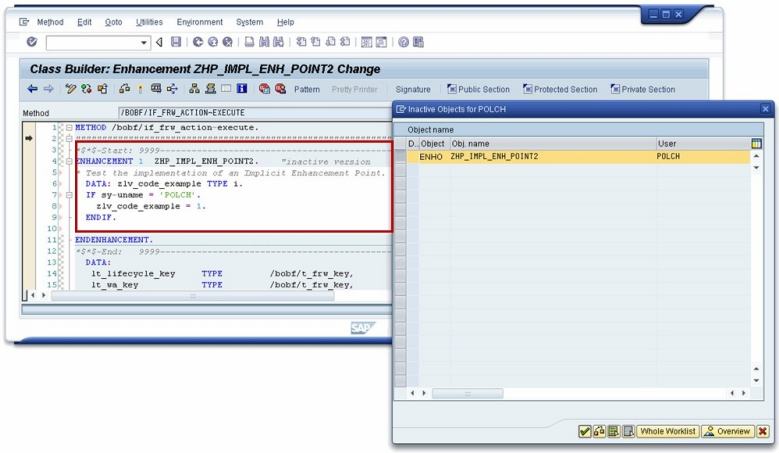


Figure 42: Tools & Techniques to Extend Business Logic 5

LESSON SUMMARY

You should now be able to:

- Understand what BAdIs are and what they are used for
- Explain how to find BAdIs in TM for a specific purpose
- Understand how to use BAdIs to influence the TM business logic
- Explain the basic steps how to create a BAdI-Implementation
- Understand what Implicit Enhancements are and what they can be used for
- Explain the critical aspects of Implicit Enhancements
- Understand how Implicit Enhancements influence the TM business logic
- Explain the basic steps how to create an Implicit Enhancement

Learning Assessment

1. Which BAdI work modes exist?

   Choose the correct answers.

   - ☐ A Standard logic only
   - ☐ B Customer logic only
   - ☐ C Modify standard result

2. What are the properties of a BAdI?

   Choose the correct answers.

   - ☐ A It has a defined interface which acts as a kind of *contract*.
   - ☐ B It can only be implemented by customers / partners.
   - ☐ C Once created, a BAdI is always active.
   - ☐ D A BAdI cannot be changed in an incompatible way.

3. At which spots can an implicit enhancement be created?

   Choose the correct answers.

   - ☐ A At the beginning of a method.
   - ☐ B At the end of a method.
   - ☐ C Anywhere within a method.
   - ☐ D At the beginning of a function module.

4. Which statements are correct for implicit enhancements?

   Choose the correct answers.

   - ☐ A They can cause syntax errors when implementing a note.
   - ☐ B They can prevent the execution of standard code.
   - ☐ C They have access to local variables in a method.
   - ☐ D They can call private methods of a class.

Learning Assessment - Answers

1. Which BAdI work modes exist?

   Choose the correct answers.

   - [X] A Standard logic only
   - [X] B Customer logic only
   - [X] C Modify standard result

2. What are the properties of a BAdI?

   Choose the correct answers.

   - [X] A It has a defined interface which acts as a kind of *contract*.
   - [ ] B It can only be implemented by customers / partners.
   - [ ] C Once created, a BAdI is always active.
   - [X] D A BAdI cannot be changed in an incompatible way.

3. At which spots can an implicit enhancement be created?

   Choose the correct answers.

   - [X] A At the beginning of a method.
   - [X] B At the end of a method.
   - [ ] C Anywhere within a method.
   - [X] D At the beginning of a function module.

4. Which statements are correct for implicit enhancements?

   Choose the correct answers.

   [X] A They can cause syntax errors when implementing a note.

   [X] B They can prevent the execution of standard code.

   [X] C They have access to local variables in a method.

   [X] D They can call private methods of a class.

UNIT 8 Process Control Framework

Lesson 1

UNIT OBJECTIVES

- Explain the basics of Process Control Framework
- Define the components of the PCF
- Customize and use the PCF

Unit 8
Lesson 1

Customizing and Using the Process Control Framework (PCF)

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain the basics of Process Control Framework
- Define the components of the PCF
- Customize and use the PCF

Process Control Framework

Process Control Framework supports the flexible definition of application processes as a sequence of methods (strategies). This flexibility does not only allow SAP to define various application processes, but also SAP customers to extend these processes with own implementations in order to meet their specific needs. For example, it is possible to implement the functionality that the system should execute Charge Calculation during saving a freight order.

- What is the Process Controller Framework (PCF)?
  - The Process Controller Framework (PCF) allows the flexible definition of application processes. This is accomplished by defining a process as a sequence of methods which represent the single process steps. Such a sequence of methods is called a Strategy.
  - Using the PCF means the definition of an application process as a sequence of methods (a Strategy) in customizing by SAP, partners, and customers.
- Why Process Controller Strategies?
  - Pieces of functionality can be packed into a method, which can then be included and used in a strategy.
  - Partners and customers can define their own methods and combine them either to completely new strategies or they can include their own methods in SAP standard strategies. That is, they also can enhance them with their customer specific functionality.
  - Allows modification free enhancements of business logic.

- **Process Controller Overview.** The Process Controller allows you to define a sequence of process steps to be executed.
  - The process steps are represented by methods.

- A Strategy is a sequence of methods from a pool of available methods.
- Customers and partners can define their own methods and sequences. Standard Strategies can be enhanced with customer specific methods and customer specific strategies can be defined.

- For Developers: The Process Controller should be considered in the following situations:
  - If multiple process steps are to be executed in a given sequence.
  - If the process to be realized can vary in the sequence of steps.
  - If a standard process step sequence shall contain additional, customer specific steps.

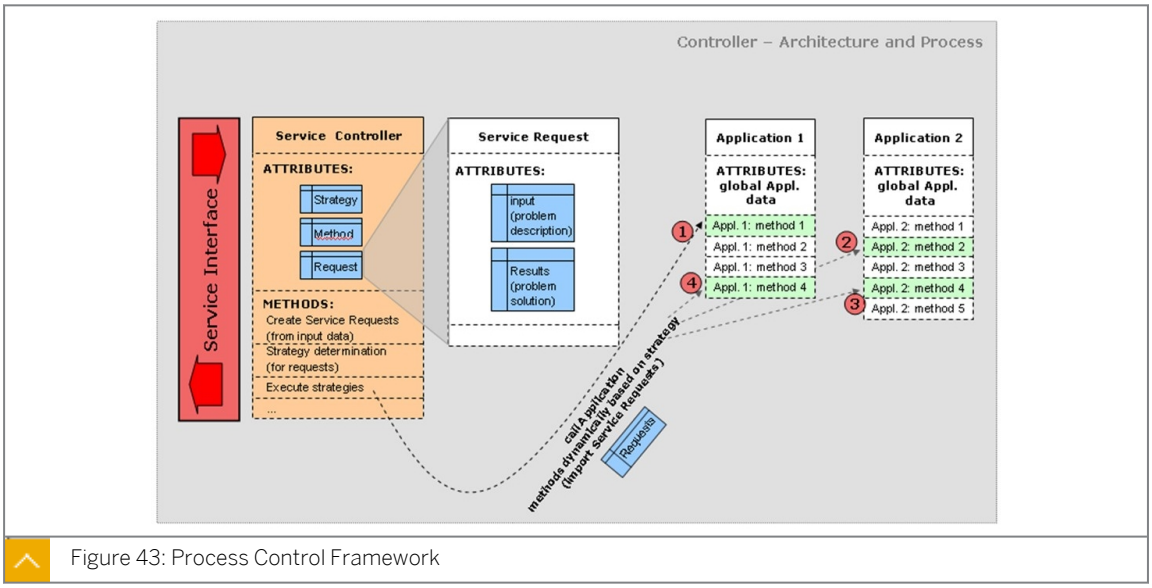


Figure 43: Process Control Framework

- PCF supports the flexible definition of application processes. This flexibility does not only allow SAP to define various application processes, but also SAP customers to extend these processes with own implementations in order to meet their specific needs.
- Using the PCF:
  - The definition of application processes as sequence of methods (strategies) using the customizing activities (by developers of SAP, partners or customers).
  - For each single request to the application process, the execution of the strategy which has been assigned to this request by the calling application.
  - In TM, a Process Controller Strategy is typically assigned in Customizing or in Profiles (or determined by a condition).

Process Control Framework Components

PCF Components

Service

A service is clustering strategies and methods working on the same process and in the same application area. It is used to clearly separate the maintenance.

Method

All methods share the same interface of an object for providing parameters and a list of mutually independent requests. Some methods comprise functionality themselves, whereas others mainly encapsulate existing functionality (like EH&S check within FUB). There is a method pool for each process, and each method pool contains all methods which the application process requires in order to define its strategies. A method contains the coding (ABAP / ABAP OO).

Strategy

A strategy serializes a selection of methods from the method pool. It brings them in the sequence in which they shall be executed by the Controller.

Parameter

Parameters influence the behavior of the respective methods, but do not change the sequence in which the methods are executed. The Controller passes them on to each method through its interface. A parameter may not only have different values in different methods, but also in the same method as part of different strategies, or even in the same method of the same strategy as part of different requests.

Request

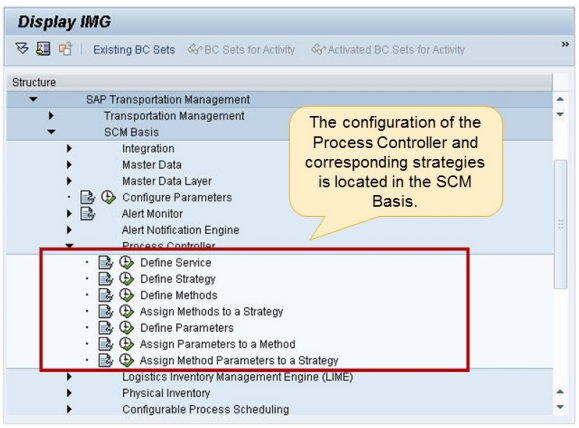
The requests represent the various business demands (for example, determination of a best route) on an application process. The Controller processes the requests by passing them on to the strategy methods through their interfaces. The requests may further allow storing possible solutions.

Application Class

The methods in the pool of the Controller are provided by so-called application classes. Method execution by the Controller works (dynamically) since the definition of a new method includes the name of the providing application class. The results, which a method produces, are either passed on to all succeeding methods with the corresponding requests or, if the calling application, but not succeeding methods take advantage of these results, then stored externally.

Process Control Framework Customization

- The following list shows you Customizing activities:
  - Define Service
  - Define Strategy
  - Define Methods
  - Assign Methods to Strategy
  - Define Parameters
  - Assign Parameters to Method
  - Assign Method Parameter to Strategy



Process Controller

- In SAP TM 9.x the customizing can be found under the following path:
  - *SAP Transportation Management → SCM Basis → Process Controller*.
- In S/4HANA the customizing can be found under the following path:
  - *Transportation Management → Basic Functions → Process Controller*.

Figure 44: Process Controller

Classes

- Controller: `/SCTM/CL_CONTROLLER`. The Controller class is the generic process controller. It handles the request data, strategy determination, gets the application object with the concrete methods and executes the strategies.
- Process Control Request: `/SCTM/CL_REQUEST`. The request class is the generic process request. It is the container for all process data (including input, output and options). A table of generic requests is passed from one PCF method to another.

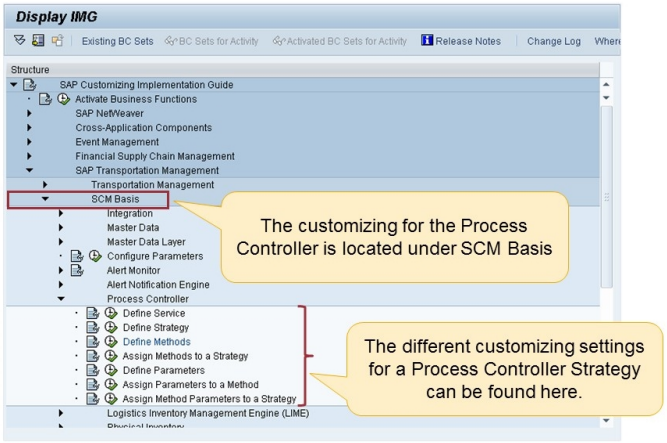


Figure 45: Tools & Techniques to Extend Business Logic Process Controller 1

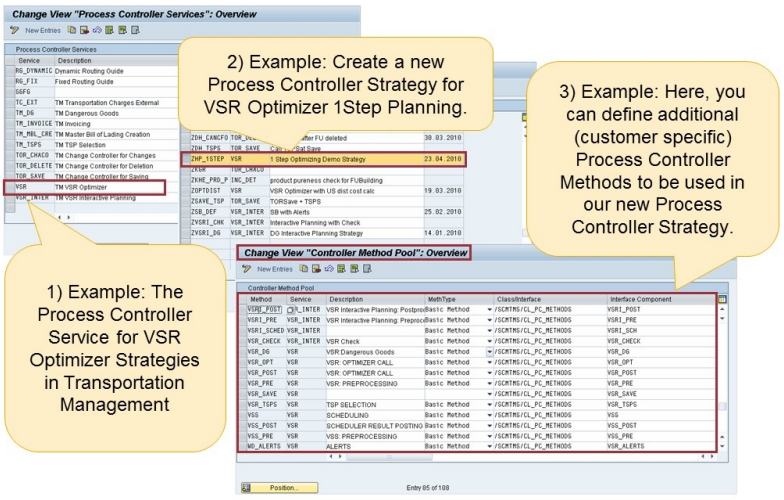


Figure 46: Tools & Techniques to Extend Business Logic Process Controller 2

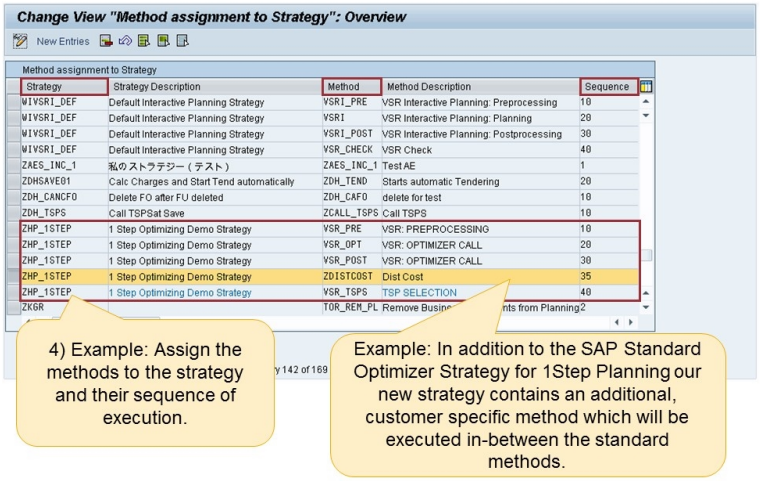


Figure 47: Tools & Techniques to Extend Business Logic Process Controller 3

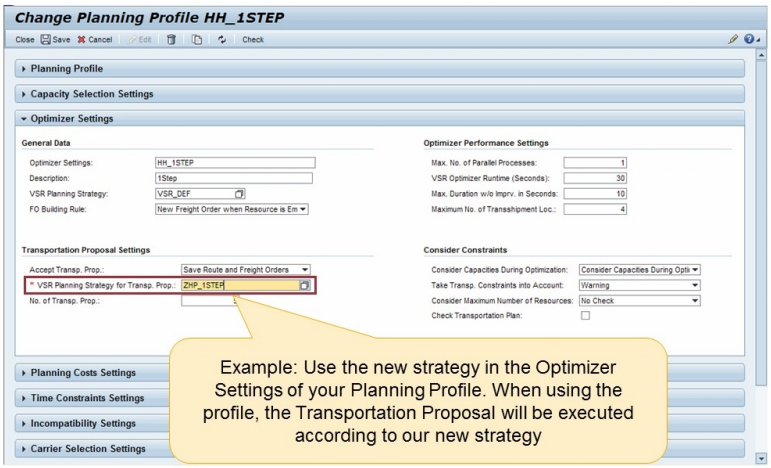


Figure 48: Tools & Techniques to Extend Business Logic Process Controller 4

LESSON SUMMARY

You should now be able to:

- Explain the basics of Process Control Framework
- Define the components of the PCF
- Customize and use the PCF

Learning Assessment

1. Process controller (PC) is only used by customers / partner for enhancement purposes?

   Choose the correct answer.

   - ☐ **A** Yes, because no PC customizing is delivered by standard.
   - ☐ **B** Yes, because standard functionalities are not implemented using the PC framework.
   - ☐ **C** No, also standard processes are implemented using the PC framework.
   - ☐ **D** No, but only PC service and strategy customizing is delivered in standard. Coding must be implemented by the customer.

2. What are process controller entities?

   Choose the correct answers.

   - ☐ **A** PC service customizing
   - ☐ **B** PC strategy customizing
   - ☐ **C** PC strategy method assignment customizing
   - ☐ **D** ABAP coding

3. Which statement is correct? (multiple answers possible)

   Choose the correct answers.

   - ☐ **A** There is a defined interface which must be implemented by ABAP classes / method which shall serve as PC method.
   - ☐ **B** There is a defined signature which must be maintained for an ABAP method which shall serve as PC method.
   - ☐ **C** The process controller framework can only be used in a BOPF action implementation.
   - ☐ **D** You can assign the same PC method multiple times to a PC strategy.

Learning Assessment - Answers

1. Process controller (PC) is only used by customers / partner for enhancement purposes?

   Choose the correct answer.

   - [ ] A Yes, because no PC customizing is delivered by standard.
   - [ ] B Yes, because standard functionalities are not implemented using the PC framework.
   - [x] C No, also standard processes are implemented using the PC framework.
   - [ ] D No, but only PC service and strategy customizing is delivered in standard. Coding must be implemented by the customer.

2. What are process controller entities?

   Choose the correct answers.

   - [x] A PC service customizing
   - [x] B PC strategy customizing
   - [x] C PC strategy method assignment customizing
   - [ ] D ABAP coding

3. Which statement is correct? (multiple answers possible)

   Choose the correct answers.

   - [ ] A There is a defined interface which must be implemented by ABAP classes / method which shall serve as PC method.
   - [x] B There is a defined signature which must be maintained for an ABAP method which shall serve as PC method.
   - [ ] C The process controller framework can only be used in a BOPF action implementation.
   - [x] D You can assign the same PC method multiple times to a PC strategy.

UNIT 9 Conditions

Lesson 1

UNIT OBJECTIVES

- Understand what conditions are and what they are used for
- Understand how to use a condition to influence the TM business logic
- Explain how to define a condition in SAP TM
- Explain the basic customizing settings required to define conditions in SAP TM
- Define Condition Types
- Create Data Access Definitions
- Explain the usage of BRF+ in TM Conditions

Unit 9
Lesson 1

Setting up and Using Conditions

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understand what conditions are and what they are used for
- Understand how to use a condition to influence the TM business logic
- Explain how to define a condition in SAP TM
- Explain the basic customizing settings required to define conditions in SAP TM
- Define Condition Types
- Create Data Access Definitions
- Explain the usage of BRF+ in TM Conditions

Conditions

Conditions are used to determine dependent values, for example, in the area of filtering Freight Units, determining FUBR, Change Controller, etc. Data Access Definition defines which values the system is to adopt directly or use as input values for a decision table, and where it is to determine this data.

- Conditions can be considered in a multitude of use cases. for example:
  - Decisions
  - Data Validation and Error Detecting
  - Classification and Derivation
  - Matching
  - Calculation
- Conditions can be defined, adjusted and simulated without having to write code.
  - High degree of flexibility
  - Modification free adjustments can be done also by partners and customers.
- For Developers: The class `/SCMTMS/CL_COND_OL` provides reusable methods to execute and handle input & output data of Rules / Conditions.
  - Minimum coding effort the get rules, conditions running.
- Creating and changing Conditions is integrated into the TM UI, including a functionality for simulation.

- **Usage of BRF+ based Conditions in TM**. TM makes use of the Reuse Component BRF+ to realize condition-based business logic (Business Rules Framework).
- Conditions use a set of input values which can, for example, come from attributes of the TM Business Objects.
- These input values are then mapped onto corresponding output values. The output of a condition can be a simple Boolean which indicates to select a business object instance (yes/no), it can be a simple value like an ID of a business partner or it can be a set of output values which are determined by the condition based on the input values.
- Conditions are called at explicit points in time / coding places during execution of process steps.
- In TM, a condition usually is assigned in customizing (for example, to a document type) or in Profiles.

Why Conditions & BRF+?

- BRF+ is an available reuse component that already provides the required framework to define, configure and execute condition-based business logic.
- It enables modification free, easy to use and highly flexible adjustments and enhancements of business logic / system behavior.
- BRF+ UI for condition configuration and maintenance is reused in the related TM UI.

- Conditions are defined and executed with BRF+ with the following caveats:
  - As per SAP TM 9.0 conditions can be defined via a specific UI integrated into the application that reuses and integrates certain parts of the BRF+ UI. In NWBC (or the Easy Access Menu) path: *Application Administration → General Settings → Conditions*.
  - Via a POWL you can search for existing conditions. Moreover UIs for creating, editing and displaying conditions are available.
  - The BRF+ UI for defining decision tables is reused within this SAP TM specific UI. Also simulation of conditions is reused.
  - No need to use the BRF+ UI directly to create and configure conditions (but it is possible).

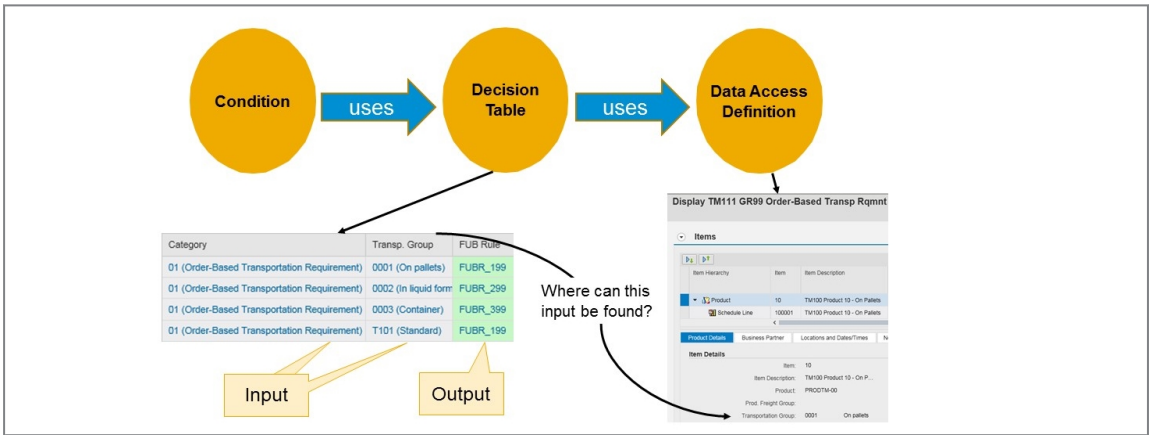


Figure 49: Architecture

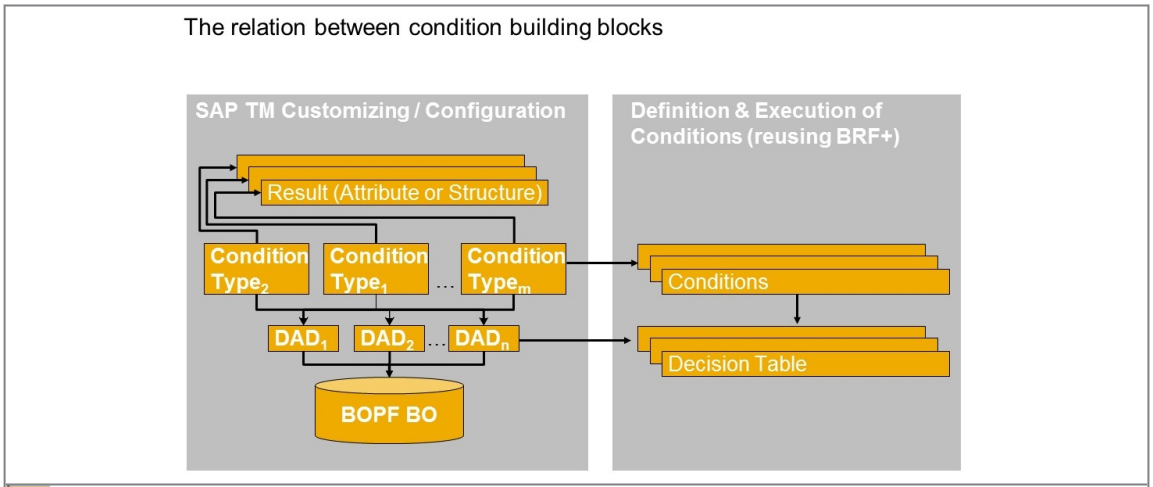


Figure 50: Conditions and Building Blocks

Customizing (SPRO)

The TM-related customizing for conditions can be found via transaction SPRO under the following path: *SAP Transportation Management → Transportation Management → Basic Functions → Conditions*. A condition type defines possible input values as well as the output of conditions of that type.

Data access definitions define how the content of the input values is read during runtime. For example, this can be a generic access to a defined BO, BO node and BO node attribute (can be configured in the data access definition), or it can be a specific class method that will provide the data (implementation required).

A condition type gets assigned one or more such data access definitions. When creating a condition of this type, the input can be build up from these assigned data access definitions.

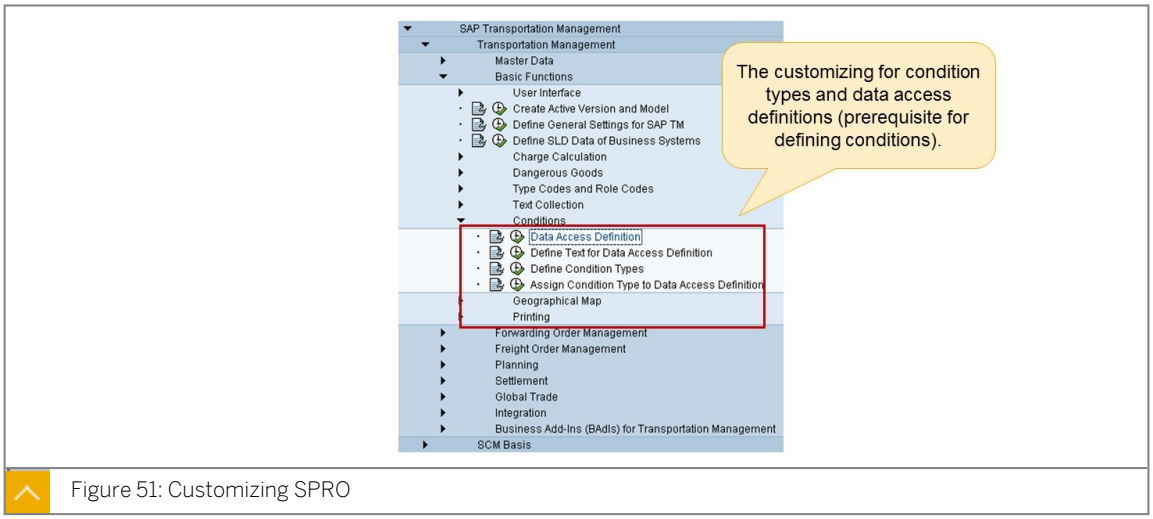
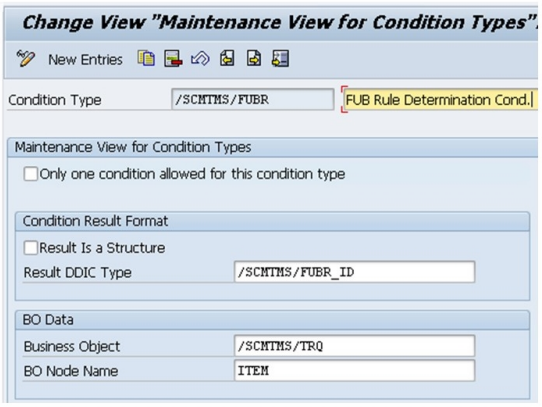


Figure 51: Customizing SPRO



- Result DDIC Type: Defines the DDIC element for the result for the condition
  - Result can be a single value or a structure
- Business Object and BO Node Name define the base for this condition type

Figure 52: Condition Type

Data Access Definition

Data Access Definitions define how input data is read, there are three possibilities:

Dynamic BO Data Access

Read input directly from the BO. Define the BO name, node and field and optional filter fields and values. Only nodes within the entry BO can be used.

Data Crawler

A Data Crawler Profile can consist of several steps to navigate along associations and read the input from the target node. Navigation along Cross-BO associations to read input from different BOs is possible.

Determination Class

Use a determination class to determine the input. The class must implement the interface `/SCMTMS/IF_COND_DETERM_CLASS`, method `EXTRACT_DATA_MASS`. Optionally a parameter can be defined and passed into the method to distinguish between different scenarios within the class. Coding can be used to determine data from any source.

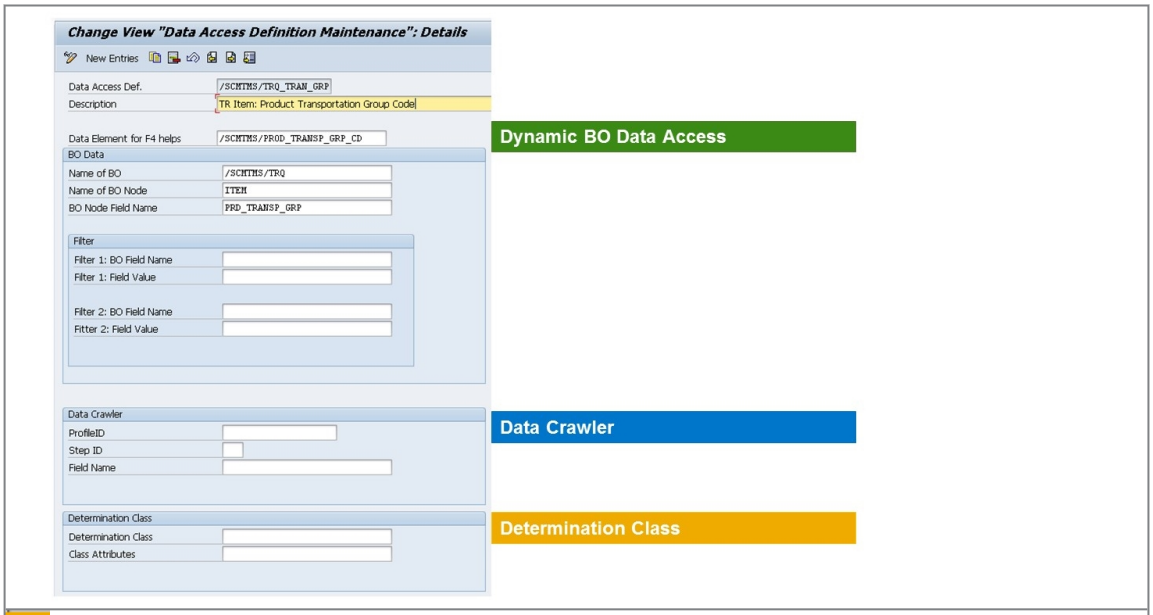


Figure 53: Conditions Definitions

Assigning Data Access Definitions to a Condition Type:

- One or more Data Access Definitions can be assigned to a Condition Type.
- When creating a condition of this type, these Data Access Definitions will be provided by the value help for the input of the condition.
- If a Data Access Definitions is defined as Default DAD in the assignment, it will automatically be added once a condition of this condition type is created.

Condition Types

Assigning Data Access Definitions to a Condition Type

- One or more Data Access Definitions can be assigned to a Condition Type.
- When creating a condition of this type, these Data Access Definitions will be provided by the value help for the input of the condition.
- If a Data Access Definitions is defined as Default DAD in the assignment, it will automatically be added once a condition of this condition type is created.

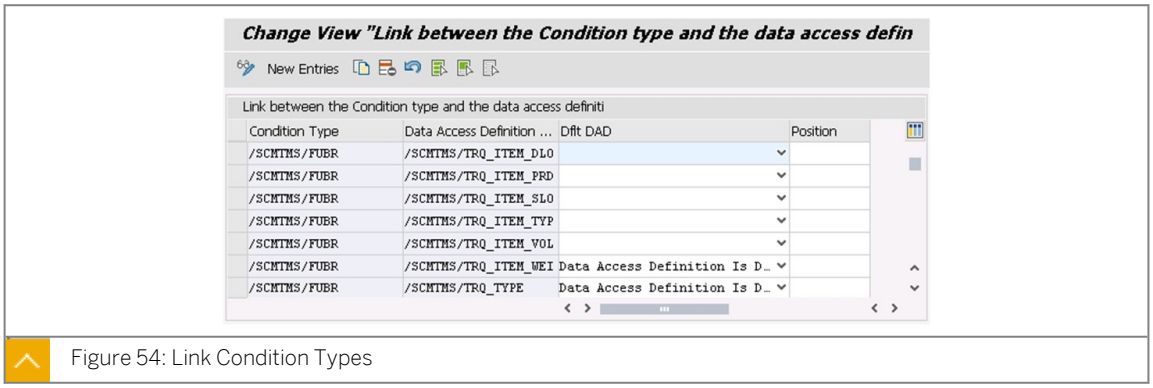


Figure 54: Link Condition Types

Condition Definitions

A condition is based on a condition type. The Origin of the Condition defines how the output result is determined from the input. Again, there are the following three alternatives:

Direct Business Object Access

The input is directly returned (input = output). This feature is used to flexibly read BO data using conditions, e.g. to set up incompatibilities.

Condition Based on BRFPlus Decision Tables

The input is used to evaluate the entries of a Decision Table and to return either the output assigned to the first or all entries of the Decision Table that match the input.

Condition Based on BRFPlus Expression

For more advanced scenarios, BRF+ can be used to determine the result based on logical expressions or formulas.

When creating a new condition of a given condition type:

- The DADs to be used can be defined (i.e. a subset of DADs assigned to the condition type is used for e.g. the decision table).
- The UI for maintaining decision tables is the reused UI of BRF+.
- The condition can be also simulated directly out of the SAP TM UI (again reusing the corresponding BRF+ UI).
- All features of BRF+ for defining conditions are also available on this SAP TM UI.

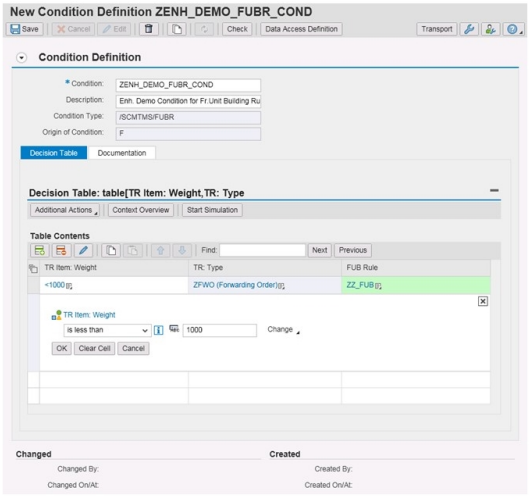


Figure 55: New Condition Definition

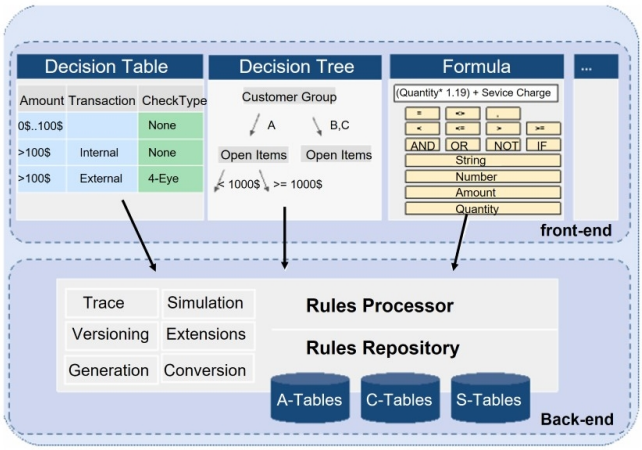


Figure 56: BRF+ Overview

The following series of figures show you the tools and techniques to extend Business Logic Rules and Conditions.

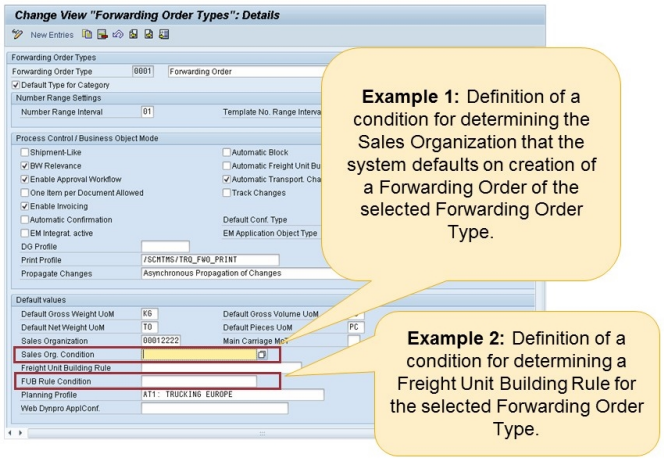


Figure 57: Tools and Techniques to Extend Business Logic Rules / Conditions 1

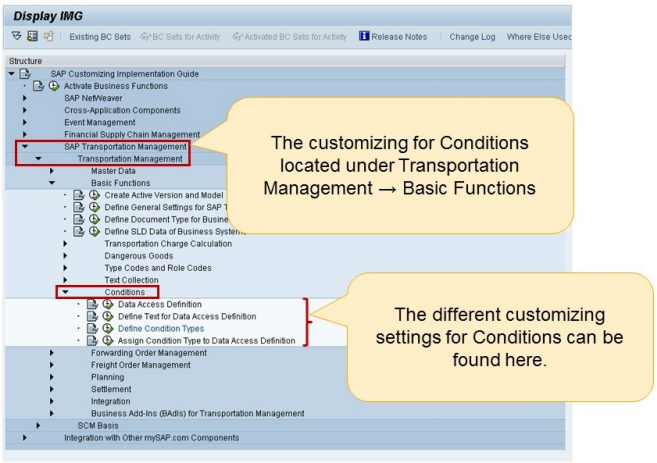


Figure 58: Tools and Techniques to Extend Business Logic Rules / Conditions 2

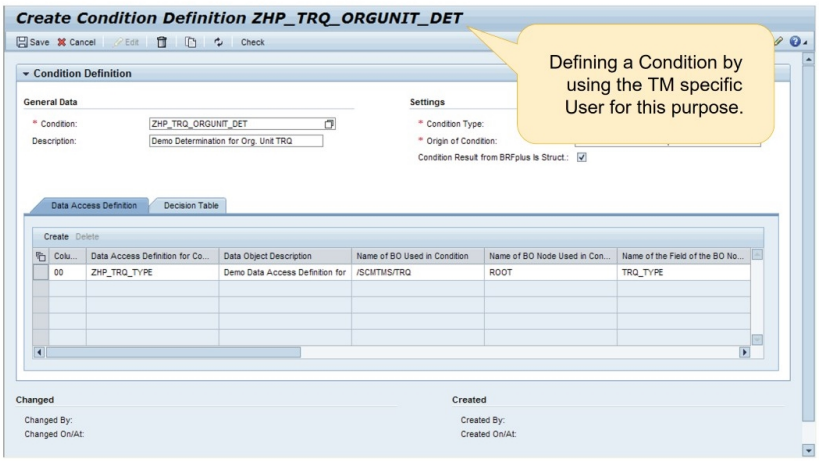


Figure 59: Tools and Techniques to Extend Business Logic Rules / Conditions 3

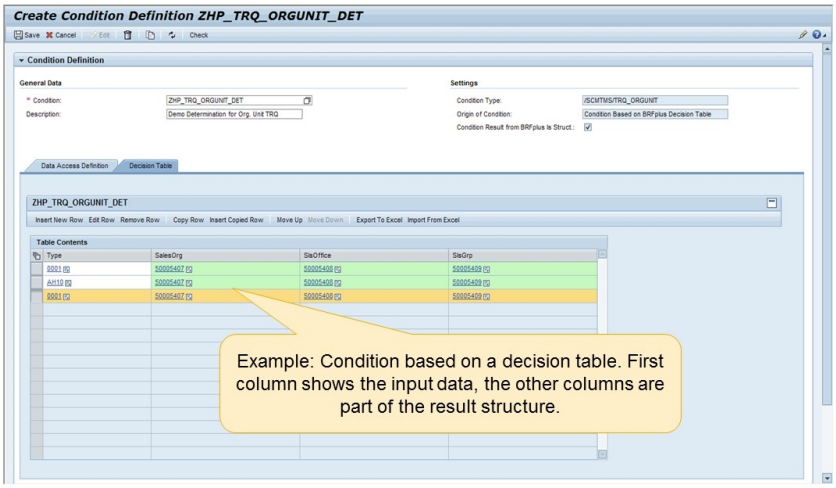


Figure 60: Tools and Techniques to Extend Business Logic Rules / Conditions 4

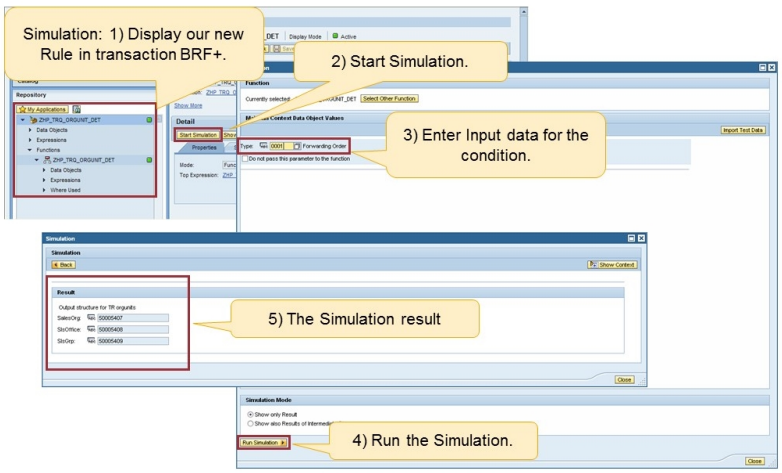


Figure 61: Tools and Techniques to Extend Business Logic Rules / Conditions 5

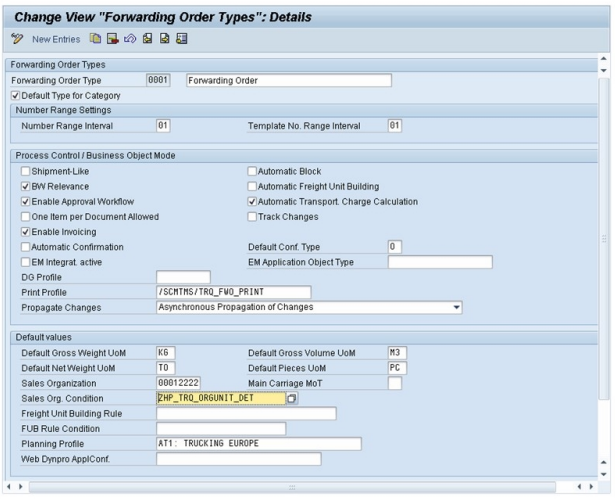


Figure 62: Tools and Techniques to Extend Business Logic Rules / Conditions 6

LESSON SUMMARY

You should now be able to:

- Understand what conditions are and what they are used for
- Understand how to use a condition to influence the TM business logic
- Explain how to define a condition in SAP TM
- Explain the basic customizing settings required to define conditions in SAP TM
- Define Condition Types
- Create Data Access Definitions
- Explain the usage of BRF+ in TM Conditions

Learning Assessment

1. Which options do you have to provide input data for a condition via a data access definition?

   Choose the correct answers.

   - ☐ **A** Dynamic BO Data Access
   - ☐ **B** Data Crawler
   - ☐ **C** Determination Class
   - ☐ **D** CDS view

2. How can you determine the output via a condition?

   Choose the correct answers.

   - ☐ **A** Direct Business Object Access
   - ☐ **B** ABAP coding
   - ☐ **C** BRFPlus Decision Tables
   - ☐ **D** BRFPlus Expression

3. What is defined in the condition type?

   Choose the correct answers.

   - ☐ **A** The entry BO and entry node
   - ☐ **B** The result data type
   - ☐ **C** How the output is determined
   - ☐ **D** If there can be multiple conditions or only one for this type

Learning Assessment - Answers

1. Which options do you have to provide input data for a condition via a data access definition?

   Choose the correct answers.

   - [X] A Dynamic BO Data Access
   - [X] B Data Crawler
   - [X] C Determination Class
   - [ ] D CDS view

2. How can you determine the output via a condition?

   Choose the correct answers.

   - [X] A Direct Business Object Access
   - [X] B ABAP coding
   - [X] C BRFPlus Decision Tables
   - [X] D BRFPlus Expression

3. What is defined in the condition type?

   Choose the correct answers.

   - [X] A The entry BO and entry node
   - [X] B The result data type
   - [ ] C How the output is determined
   - [X] D If there can be multiple conditions or only one for this type

UNIT 10 Post Processing Framework

Lesson 1

UNIT OBJECTIVES

- Explain the purpose of the Post Processing Framework
- Configure the Post Processing Framework for SAP TM

Unit 10
Lesson 1

Configuring and using the Post Processing Framework (PPF)

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the purpose of the Post Processing Framework
- Configure the Post Processing Framework for SAP TM

Post Processing Framework

In SAP TM, you would like to configure output like printing of documents or Web Service communication with internal or external systems using SAP TM Output Management. Technically this is achieved using the Post Processing Framework.

Output Management in SAP TM

For external communication, your business can use standard printing, fax, and e-mail with a document preview option. For internal communication, alert management options are available, and workflow can be used to pass requirements from one business process step to another. Communication with B2B systems is also available in situations such as tendering freight orders when a bidding or purchasing activity is present.

BI data uploads can be executed to provide management reporting on various documents like freight units, freight orders, and freight bookings. Various order events can trigger communication to event management systems to make cross-system activities transparent. The Output Management Adapter uses the Post Processing Framework (PPF) to generate and process outputs. Output can be triggered by the PPF either manually from the UI or automatically in the back end depending on the document-specific schedule conditions.

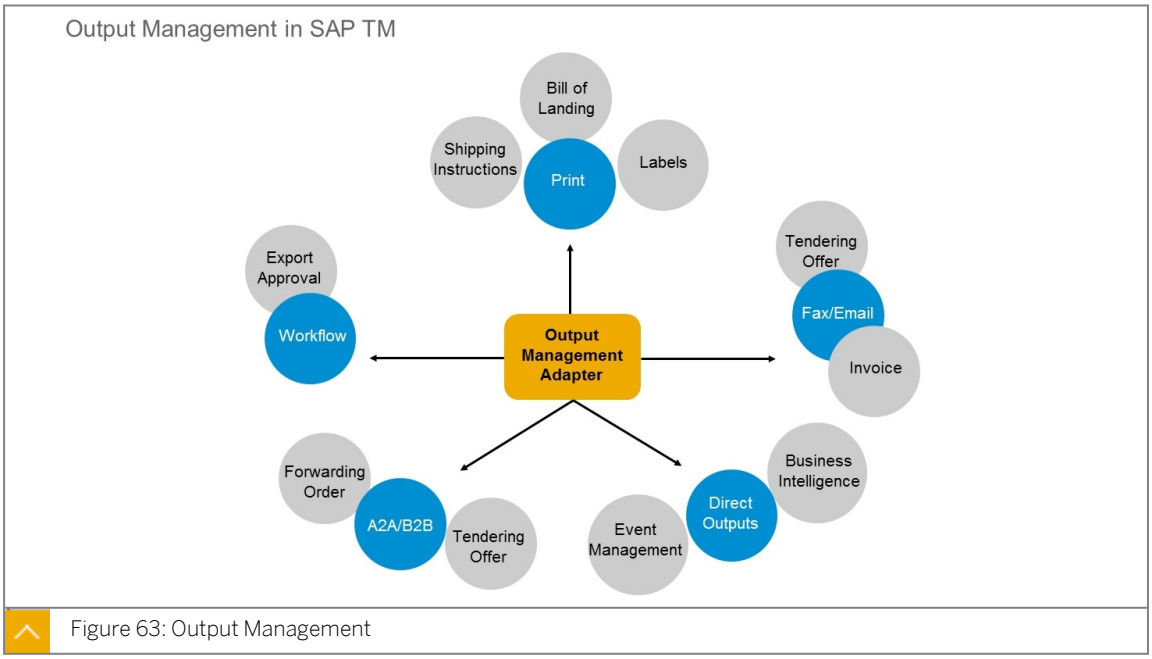


Figure 63: Output Management

Overview

The Post Processing Framework (PPF) provides SAP applications with a uniform interface for the condition-dependent generation of actions (for example, printing delivery notes, faxing order confirmations, or triggering approval procedures). The actions are generated if specific conditions are fulfilled for an application document, for example a specific status is set (approval by some person) or a specific date has been reached (two weeks before end of contract). The actions are then processed either directly, in a scheduled report later, or can be deferred via queued Remote Function Call (qRFC).

The PPF is part of the SAP NetWeaver Application Server and can be used by applications. It is the successor to message control and offers a wider range of functions, simpler connection to the applications, and greater flexibility. The PPF provides tools for scheduling, starting, and monitoring actions. Actions can be determined, generated, and processed either automatically or with user interaction. With the PPF you can evaluate modifiable conditions for determining actions. The application can be used to select the determination technology with which the conditions are evaluated or use its own determination technology.

Processing types are provided by the PPF to execute various actions, as follows:

- Printing, sending e-mails, and faxing with Smart Forms.
- Starting a workflow.
- Starting a Business Add-In.

The processing types can be adopted by the application, adapted to meet the requirements of the application, or replaced with the application's own processing types. The PPF also provides tools for the administration of actions. It has an action overview with status display, determination logs, and processing logs. The overview can be included in the application as a subscreen.

The PPF triggers the main outbound communication and integration steps such as PI outbound messaging, integration to SAP Event Management, BI, and workflow. In addition, printing is triggered via PPF and handled by Adobe Document Services (ADS).

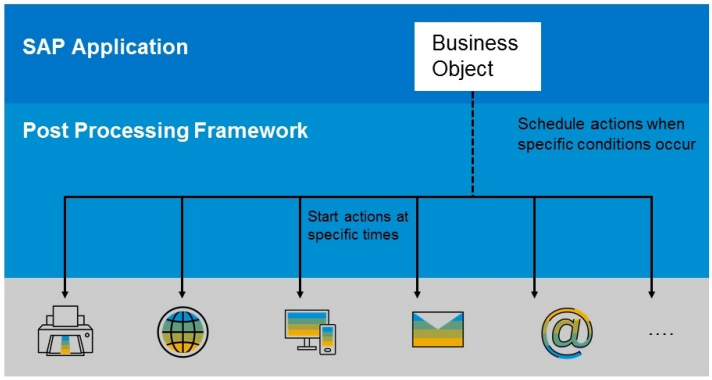


Figure 64: Overview

Architecture

Output management automates the output of business documents such as printouts, in response to certain business events. This output management adapter helps you to integrate output management functionality with application business objects (BOs) that are implemented in a business object processing framework (BOPF).

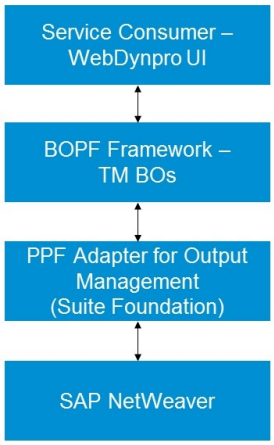


Figure 65: Architecture

Configuration

Depending on how you configure PPF, it generates output actions for particular application data records. The system then processes these actions and sends the output data.

An action definition is metadata or a skeleton of a business task and has to be maintained with its individual configuration. While defining an action, it is possible to specify the type of action (external, workflow, method call, or alert).

The processing details are taken from the action definition and can be modified. For example, the recipients (email/fax) for external communication can be set.

A valid printer has to be set with the PPF conditions for actions which result in printing.

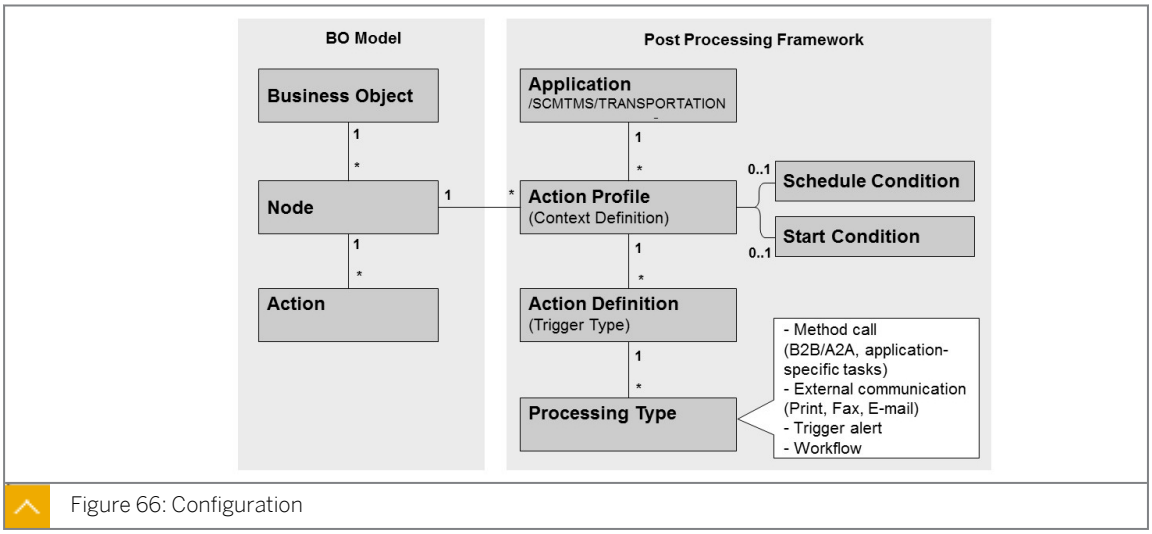


Figure 66: Configuration

Some important settings for an Action Definition are:

- Processing Time:
  - Processing when saving document – processing can be started once the document has been posted.
  - Processing using selection report – actions are initially scheduled only and processing is not started until later using a selection report or by a user in the document.
  - Immediate processing – processing can be started immediately when an action is scheduled.
- Schedule Automatically – If you select this indicator, the actions are automatically activated for processing as soon as the schedule conditions are met. If the indicator is not selected, the person responsible must manually activate the actions in the application document for processing.
- Changeable in Dialog – If you do not select this indicator, actions within this action definition cannot be changed manually after automatic determination.
- Executable in Dialog – If you do not select this indicator, actions within this action definition can only be executed automatically, either immediately after scheduling or later using a selection report.
- Partner-Dependent – Partners are determined for partner-dependent actions. The specified partner function is compared to the function of the document partner and an action is generated only if the partner functions match. If you select the Partner-Dependent indicator, you must enter a partner function.
- Partner Function – The rights and responsibilities of each partner in a business transaction. The relevant partner roles for an application can be communicated to the PPF by implementing the BAdI *GET_PARTN_ROLES_PPF*.

A PPF action profile aggregates the number of PPF action definitions by logically grouping them together. Multiple PPF action profiles can be created for a given application data table (or business object node in BOPF) depending on the actual output requirements.

For example, SAP Transportation Management contains one technical business object that serves multiple business uses. The TRQ category determines whether the business object is a request object or quotation object. The output requirements for a request object may differ

from those of a quotation object. From a logical perspective, it may therefore be necessary to set up multiple PPF action profiles for each TRQ category. The requirements for output setup may also vary in the different BO nodes.

The PPF application is the application that uses the Post Processing Framework (PPF) to generate actions.

To trigger PPF actions, start and schedule conditions need to be configured according to business needs/rules, as follows:

The schedule condition decides whether an action should be scheduled for processing. An action is therefore only generated if the schedule condition is met.

The filter value is specified in the Schedule condition field. The standard value */BOFU/EVAL_SCHEDULE_CONDITION* ensures that the method in the agent class is invoked.

The start condition is checked before the action is executed. The action is only executed when the start condition has been fulfilled.

The filter value is specified in the Start condition field. The standard value */BOFU/EVAL_START_CONDITION* ensures that the method in the agent class is invoked.

Action Profile Determination

In some cases, multiple action profiles are valid for a use case. In a transportation scenario, for example, you can assign multiple action profiles to a document type. However, the relevant profiles must be communicated to the output management adapter. You can select the action profiles based on the business scenario. All of the business logic and criteria for selecting the relevant action profiles can be implemented in method GET_PROFILES. Each of the interface methods can use the ABAP implementation as well as the variant for BRFplus. Implement the appropriate variant depending on the usage, that is, redefine the method with either the suffix ABAP or FDT depending on your scenario.

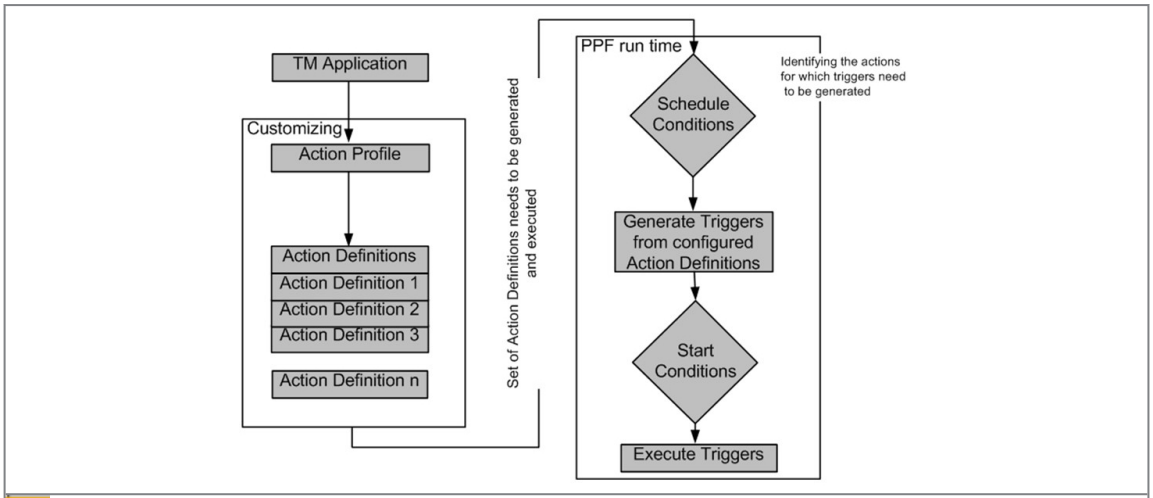


Figure 67: Integration of Customizing and Runtime

Agent Class

Any BOPF application implementing output management should create a class for handling the actions according to business needs. This class should be created with */BOFU/CL_PPF_SERV_FOR_BO* as the super class.

The agent classes can be assigned to the BO node using the O/P adapter settings.

All the relevant action-level outputs are hard wired to the corresponding methods in the agent class that is specified in the Output Management Adapter customizing. This enables easy implementation at the customer end.

The agent class provides the PPF with the application data relevant for executing outputs. Standard class */BOFU/CL_PPF_SERV_FOR_BO* must be extended by the application and the relevant methods redefined. The methods enable the logic to be implemented by either ABAP or BRFplus.

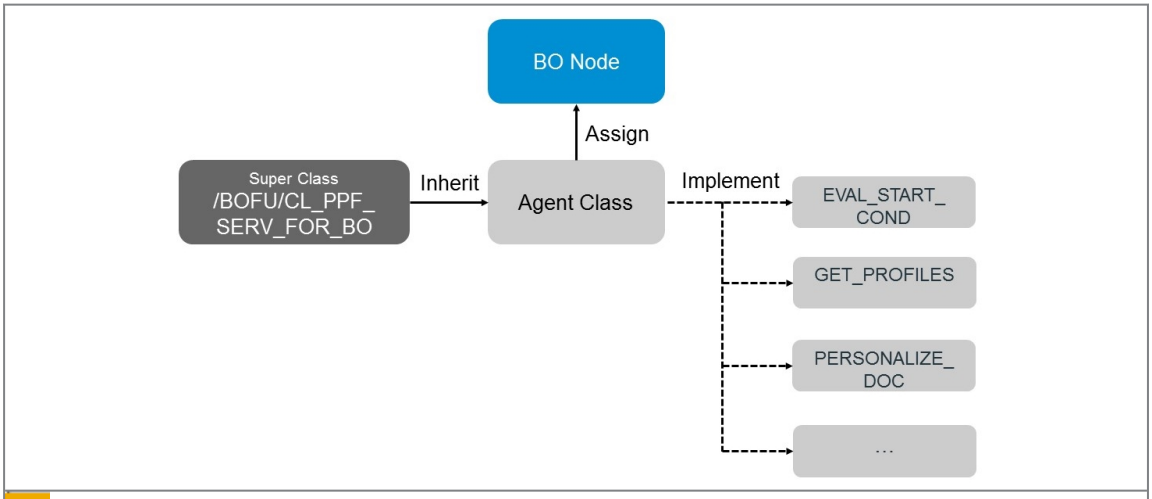


Figure 68: Agent Class

Customizing

When associating an agent class to a BO, it is possible to assign an action profile; assign actions to the action profile, and specify if the action is critical or uncritical.

Critical outputs must be completed within the logical unit of work (LUW), such as business-to-business (B2B) method calls or workflows.

Non-critical outputs fall into the category of communication methods, such as print, e-mail, and fax.

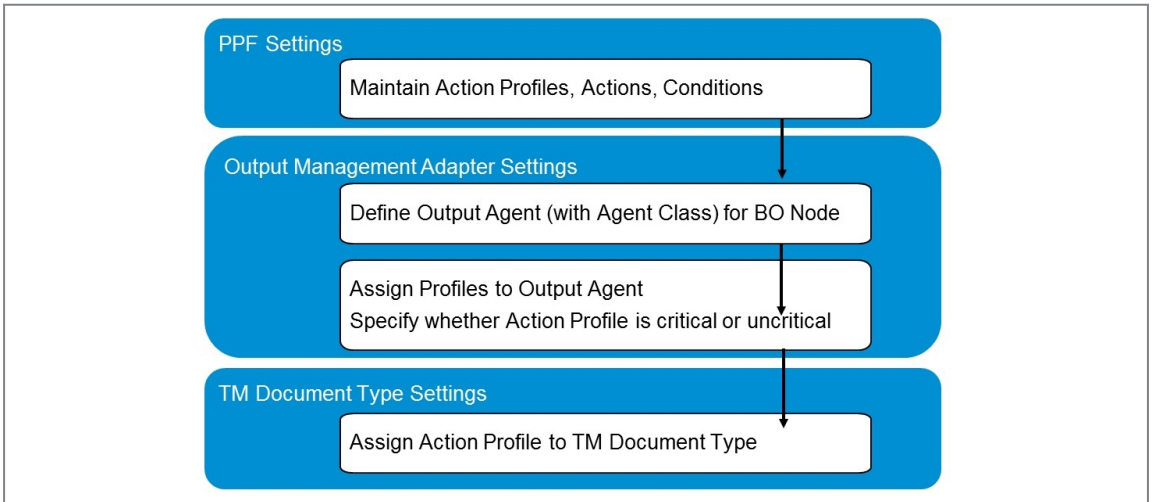


Figure 69: Customizing

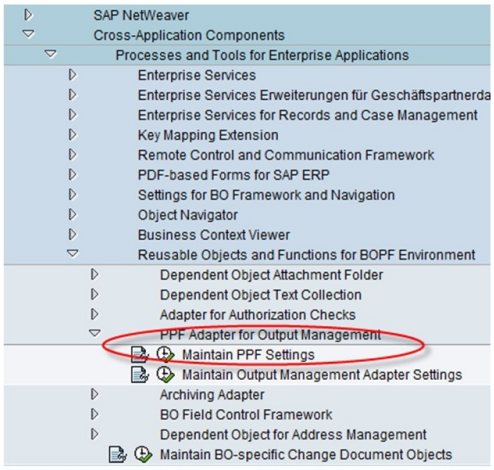


Figure 70: Post Processing Framework BPPF Settings

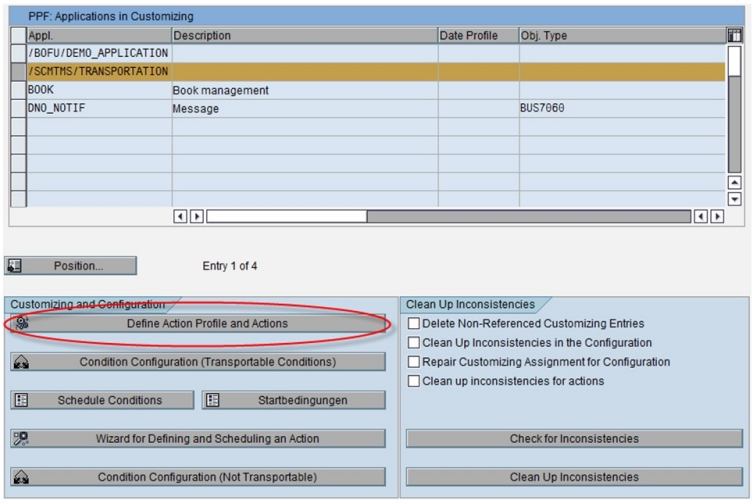


Figure 71: Post Processing Framework Define Action Profiles and Actions

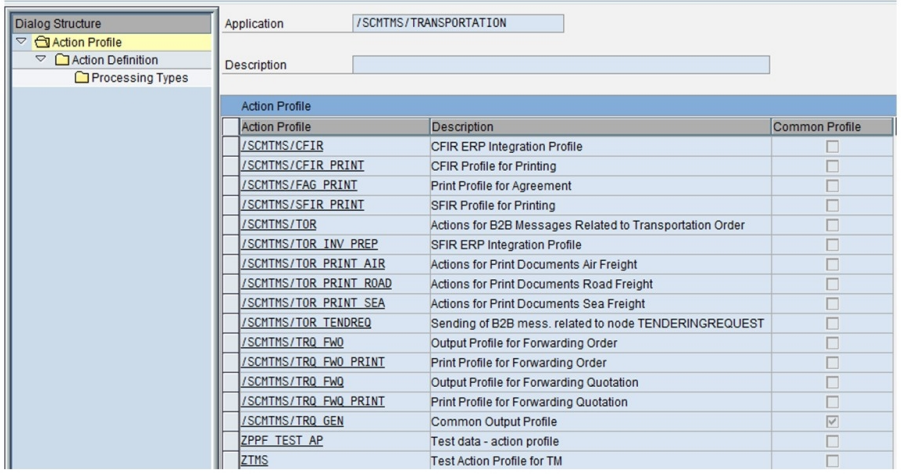


Figure 72: Post Processing Framework Action Profile

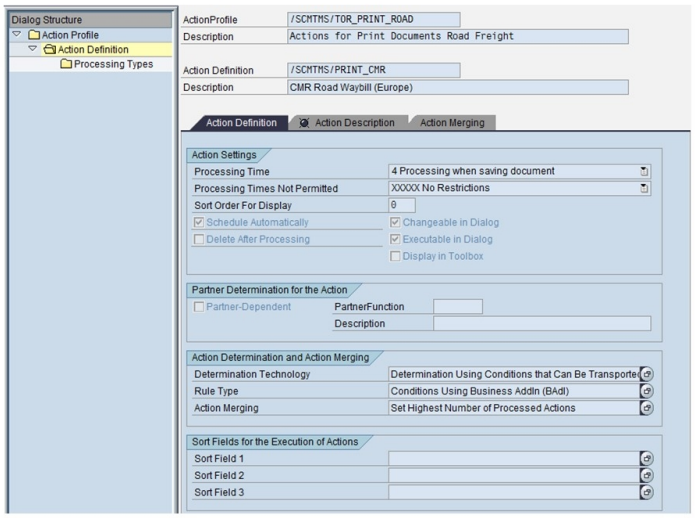


Figure 73: Post Processing Framework Define Action Definition

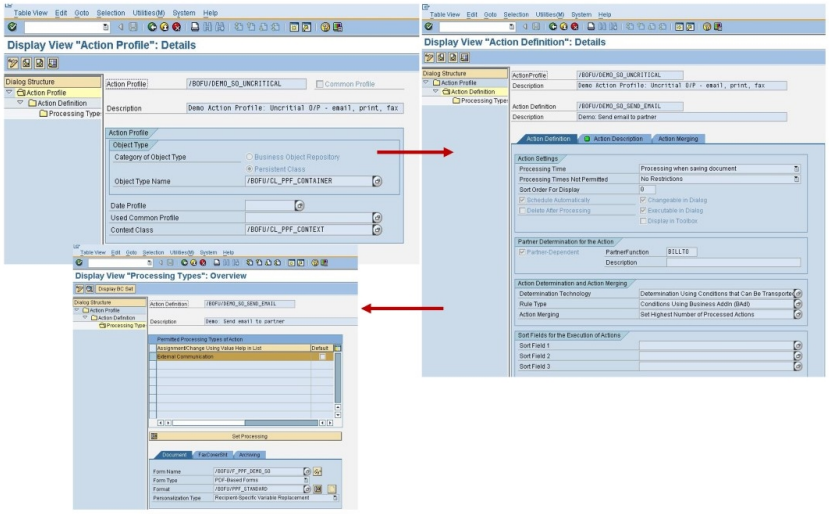


Figure 74: Define Action Profile, Action Definition and Processing Type

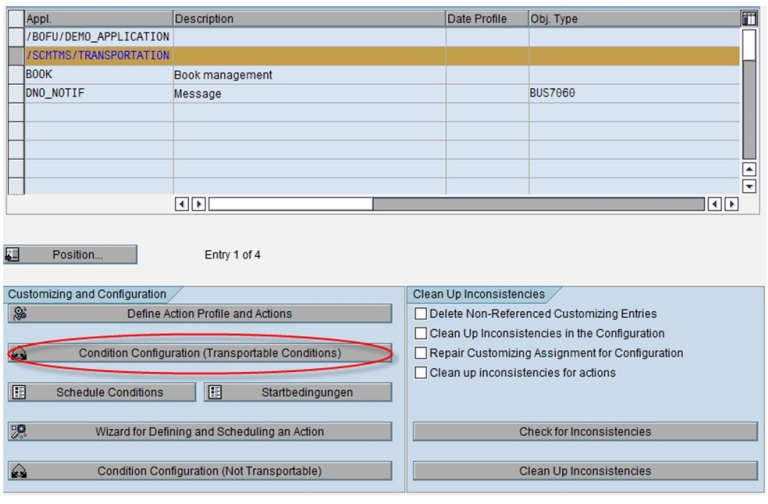


Figure 75: Condition Configuration

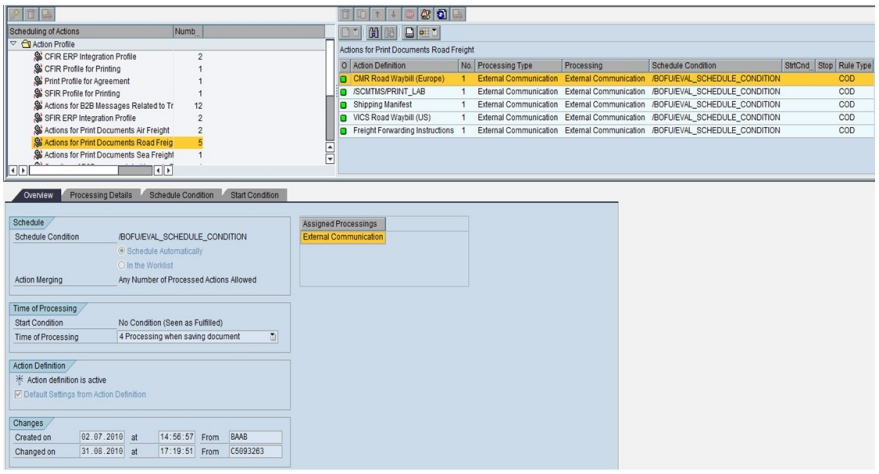


Figure 76: Condition Configuration: Overview

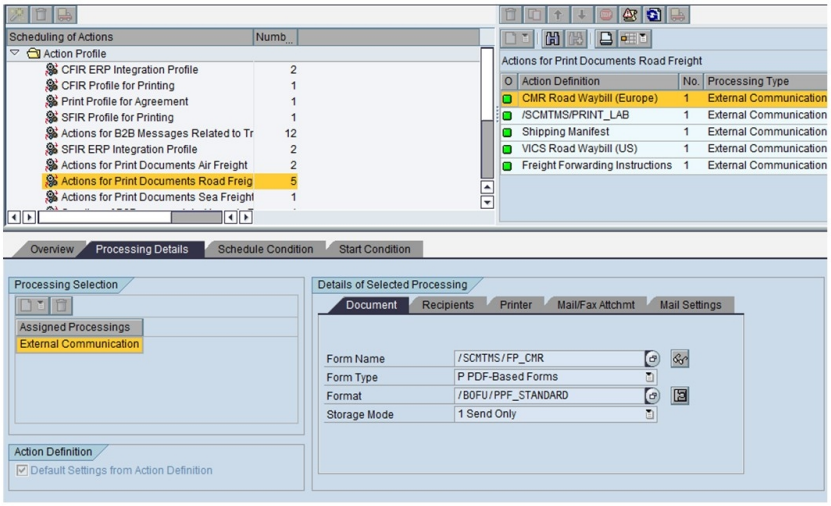


Figure 77: Condition Configuration: Processing Details 1

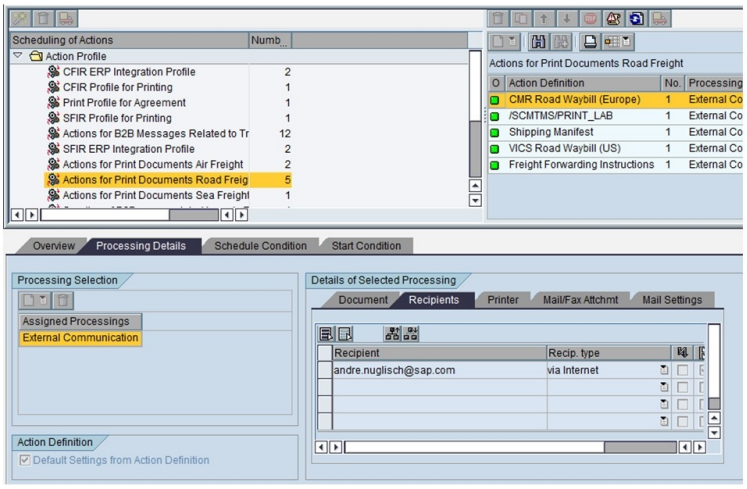


Figure 78: Condition Configuration: Processing Details 2

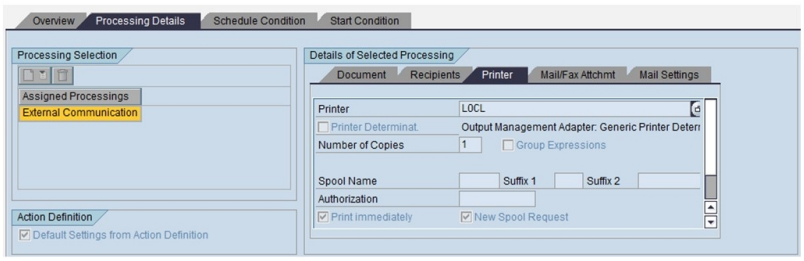


Figure 79: Condition Configuration: Processing Details 3

If the *Printer Determinat* indicator is selected, the printer is determined in the following order:

- Implementation of BAdI *PRINTER_DETERM_PPF*
- Printer setting in the condition configuration
- Printer setting in the user profile

A standard implementation is available with the filter value */SCMTMS/** for the BAdI *PRINTER_DETERM_PPF*, and so the *Printer Determinat*. Indicator is selected automatically. For customer-specific BAdI implementations, make sure that the indicator is selected for the action definitions.

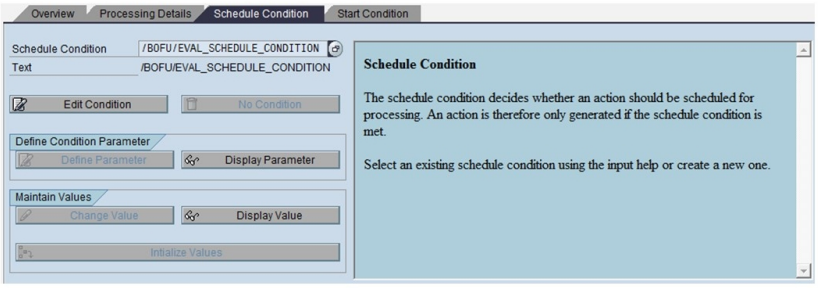


Figure 80: Condition Configuration: Schedule Conditions

The schedule condition determines whether an action is to be scheduled for processing. An action is only generated if the schedule condition is met. The filter value is specified in the Schedule Condition field. The standard value */BOFU/EVAL_SCHEDULE_CONDITION* ensures that the method in the agent class is invoked.

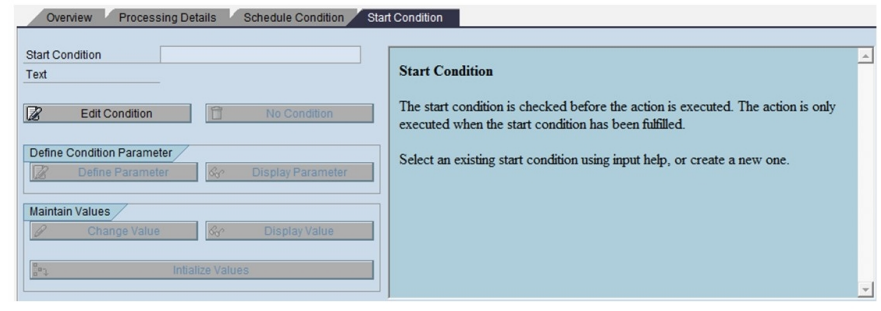


Figure 81: Condition Configuration: Schedule Conditions

The start condition is checked before an action is executed. The action is only executed if the start condition has been fulfilled. The filter value is specified in the Start Condition field. The standard value */BOFU/EVAL_START_CONDITION* ensures that the method in the agent class is invoked.

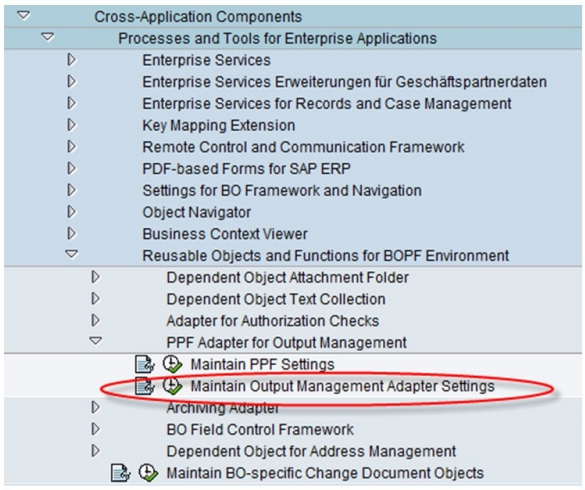


Figure 82: Output Management Adapter Settings

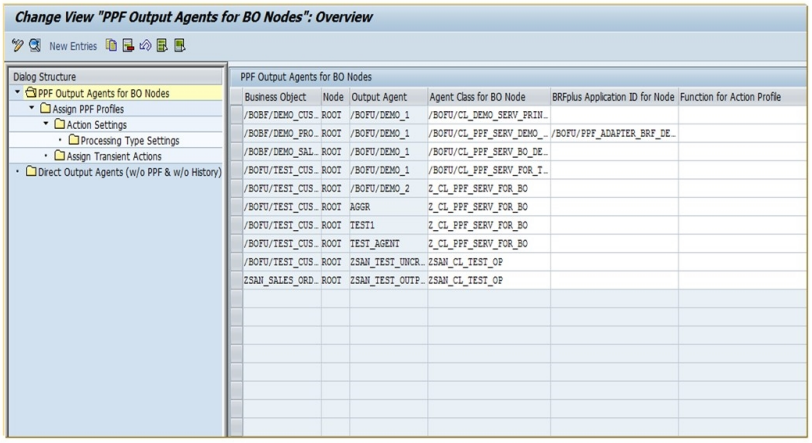


Figure 83: Output Management Configuration and Setup

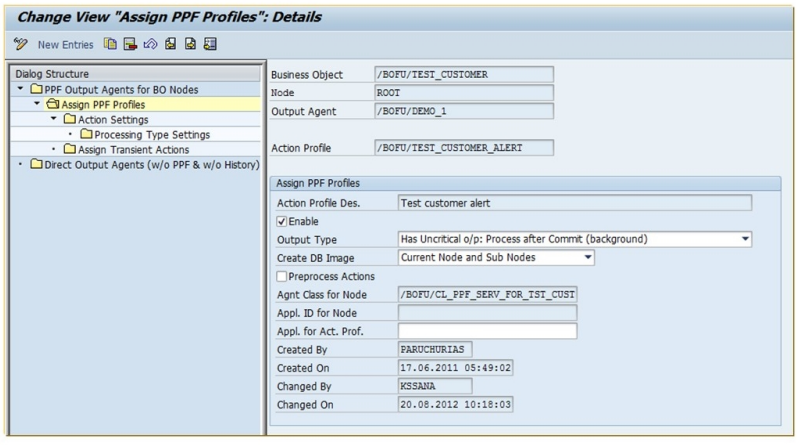


Figure 84: Output Management Configuration and Setup Assign Action Profile to BO Node

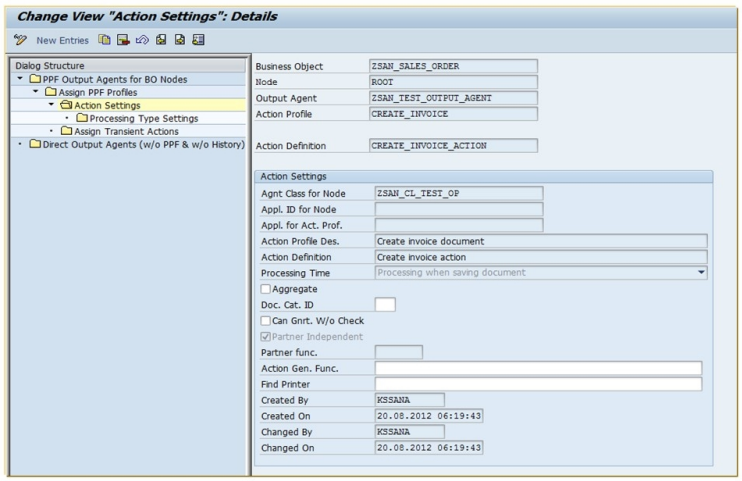


Figure 85: Output Management Configuration and Setup Action Settings

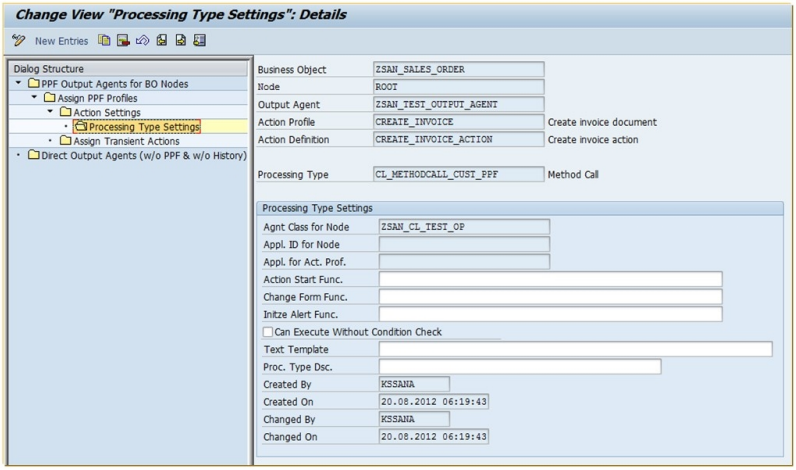


Figure 86: Output Management Configuration and Setup Processing Type Settings

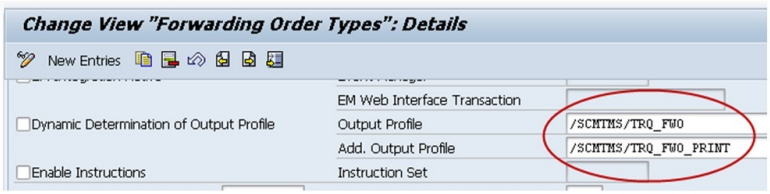


Figure 87: Assignment of Action Profile to TM Document Type

LESSON SUMMARY

You should now be able to:

- Explain the purpose of the Post Processing Framework
- Configure the Post Processing Framework for SAP TM

Learning Assessment

1. Which framework is used to consume the PPF from a TM business object?

   Choose the correct answer.

   - ☐ **A** Output management framework
   - ☐ **B** TM Conditions
   - ☐ **C** FPM / FBI
   - ☐ **D** BOPF

2. What is the purpose of an output agent class?

   Choose the correct answers.

   - ☐ **A** Check if an output should be scheduled.
   - ☐ **B** Determine the relevant PPF profiles.
   - ☐ **C** Prepare / trigger the actual output.
   - ☐ **D** Check if an output should be executed.

3. Is it possible to access database image information of the triggering transaction during an uncritical output?

   Choose the correct answer.

   - ☐ **A** Yes, this is always possible.
   - ☐ **B** Yes, but only if the customizing is setup properly.
   - ☐ **C** No, uncritical output is executed asynchronously and therefore access to the database image of the triggering transaction is not possible anymore.

4. Which types of outputs are typically generated by the PPF?

   Choose the correct answers.

   - ☐ A Printouts
   - ☐ B Emails
   - ☐ C Webservice communication
   - ☐ D FAX

Learning Assessment - Answers

1. Which framework is used to consume the PPF from a TM business object?

   Choose the correct answer.

   - [X] A Output management framework
   - [ ] B TM Conditions
   - [ ] C FPM / FBI
   - [ ] D BOPF

2. What is the purpose of an output agent class?

   Choose the correct answers.

   - [X] A Check if an output should be scheduled.
   - [X] B Determine the relevant PPF profiles.
   - [X] C Prepare / trigger the actual output.
   - [X] D Check if an output should be executed.

3. Is it possible to access database image information of the triggering transaction during an uncritical output?

   Choose the correct answer.

   - [ ] A Yes, this is always possible.
   - [X] B Yes, but only if the customizing is setup properly.
   - [ ] C No, uncritical output is executed asynchronously and therefore access to the database image of the triggering transaction is not possible anymore.

4. Which types of outputs are typically generated by the PPF?

   Choose the correct answers.

   [X] A Printouts

   [X] B Emails

   [X] C Webservice communication

   [X] D FAX

UNIT 11 Print Forms

Lesson 1

UNIT OBJECTIVES

- Understand what Print Forms are and where to find them in SAP TM
- Explain the basic steps for Print Form Enhancements

Unit 11
Lesson 1

Enhancing Print Forms

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understand what Print Forms are and where to find them in SAP TM
- Explain the basic steps for Print Form Enhancements

Print Forms

- What are Print Forms?
  - SAP TM provides a number of standard print forms that can be printed out. For example, a Pro Forma Invoice Document or different kinds of Bills of Lading.
  - The provided Print Forms take the data of a Business Object (e.g. a Forwarding Order) and places it onto a PDF-based form where a specific layout is defined to represent this data on paper.
- Why enhancing Print Forms?
  - The provided standard forms may not contain all information required by customers or the layout might not match the requirements.
  - For example, there are extension fields (customer specific) on Business Object level that shall be also printed with a form or the layout needs to be adjusted. For example changing the sequence of fields or adding a company logo.
- How is it done?
  - The available forms can be found in transaction SFP (Form Builder) by using the F4-Help in field Form on the initial screen. Search for forms that start with */SCMTMS/**.

> Note:
> See Enhancement Guide Document for a detailed example and information on how to enhance existing as well as to create new Print Forms.

Print Form Enhancements

- How to enhance or create a Print Form and what to implement?
  - You should always copy a standard Print Form before enhancing it or you create your very own new *Print Form → transaction SFP*.

- On the *Properties* Tab of SFP you can identify the related technical interface for your Print Form which provides the Print Structure that contains all data and attributes that can be used on the form.
- The Print Structure can be enhanced via an Append-Structure (`SE11`). These enhancements can be added to the Context of the Print Form and will be available to be placed in the Form's layout.

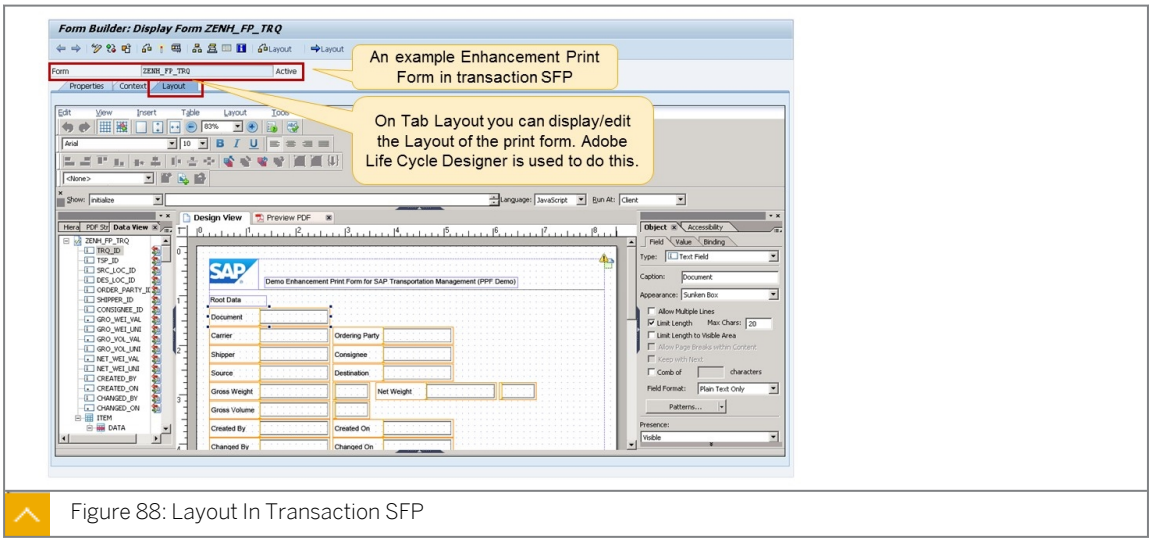


Figure 88: Layout In Transaction SFP

What to Implement?

- Each Print Form is associated with a Printing Class that contains the coding for providing the actual data to a Form (Example class: */SCMTMS/CL_PRINTOUT_FWO*).
  - The most important method is *FILL_PRINTSTRUCTURE*.
  - It contains the code to read and map data to the Print Structure.
- A BO Service class will contain coding to communicate with the Post Processing Framework (PPF)/Output Management. It is assigned to the Form in the related PPF/ Output Management Configuration (Example class: */SCMTMS/ CL_TRQ_ROOT_PPF_SERV*).
  - Method *PERSONALIZE_DOC_BY_ABAP* contains the call of the relevant methods from the Printing Class, i.e. it represents the *connection* between the Print Form-specific implementation and the configuration that is required for the usage of the Print Form.

What to Configure?

- With a new or enhanced Print Form and the required backend coding in place you can now configure the required Post Processing Framework (PPF) and Output Management Adapter Settings to get the form displayed in the print preview or printed out on paper.
- PPF Settings: In transaction `SPRO` follow the path *Cross-Application Components → Reusable Objects and Functions for BOPF Environment → PPF Adapter for Output Management → Maintain PPF Settings*.
  - Action Profiles and Actions need to be configured.
  - Condition Configuration needs to be set up.

- Output Management Adapter Settings: In transaction SPRO follow the path *Cross-Application Components → Reusable Objects and Functions for BOPF Environment → PPF Adapter for Output Management → Maintain Output Management Adapter Settings*.
  - Determines output for a given business object (BO) node.
  - Maintain output management settings for the business object (BO) node, using PPF output agents for the BO node.
  - Assign PPF profiles to the given BO node.
  - Maintain action settings and processing types.

> Note:
> See SAP TM Enhancement Guide Document for a fully detailed configuration example.

In the Document Type Customizing you can then finally assign the Output Profile that will be used for documents of this type to be displayed in the print preview, printing it on paper or sending it, for example, via Email.

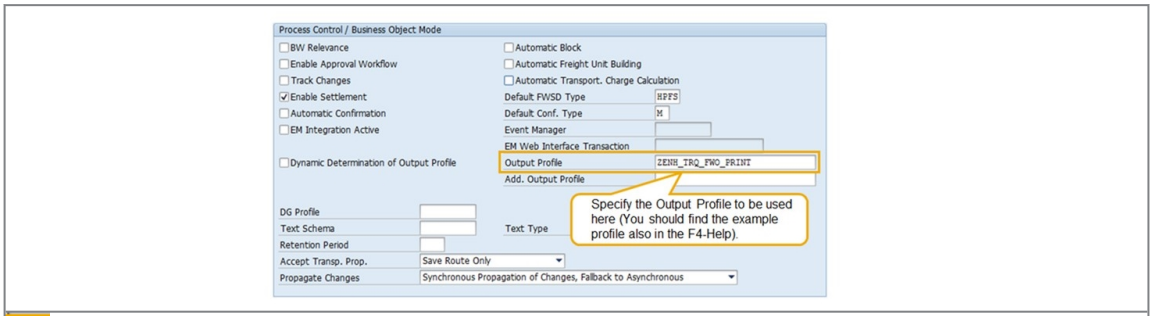


Figure 89: Configuration Aspects

The following figure is an example of displaying a FWO document preview.

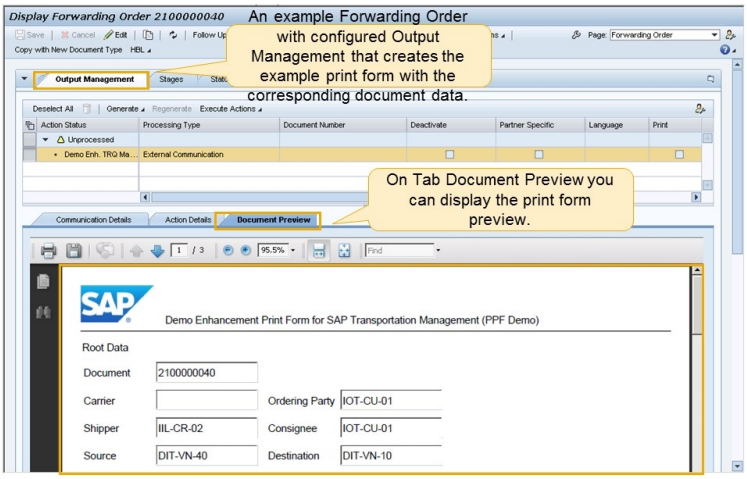


Figure 90: FWO Document Preview

LESSON SUMMARY

You should now be able to:

- Understand what Print Forms are and where to find them in SAP TM
- Explain the basic steps for Print Form Enhancements

Learning Assessment

1. How should you adapt standard print forms to your needs?

   Choose the correct answer.

   - ☐ A You should modify the standard form.
   - ☐ B You should create the form from scratch.
   - ☐ C You should copy the standard form and change the copied version.

2. How do you provide the data for the print form?

   Choose the correct answer.

   - ☐ A Via specific printout classes.
   - ☐ B Via the data crawler.
   - ☐ C Via database views.
   - ☐ D Via BO query.

Learning Assessment - Answers

1. How should you adapt standard print forms to your needs?

   Choose the correct answer.

   - [ ] A You should modify the standard form.
   - [ ] B You should create the form from scratch.
   - [x] C You should copy the standard form and change the copied version.

2. How do you provide the data for the print form?

   Choose the correct answer.

   - [x] A Via specific printout classes.
   - [ ] B Via the data crawler.
   - [ ] C Via database views.
   - [ ] D Via BO query.

UNIT 12 Integration

UNIT OBJECTIVES

- Understand the enterprise services that TM provides
- Enterprise Services in TM
- Understanding the integration scenarios TM provides
- Explain how to monitor messages in TM
- Understand how to enhance a TM web service
- Enhance SD Output determination
- Enhance the inbound agent in TM

Unit 12
Lesson 1

Understanding the Enterprise Services in TM

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understand the enterprise services that TM provides
- Enterprise Services in TM

Enterprise Services in TM

Overview

- TM provides Enterprise Service in the following areas:
  - Transportation Order Processing
  - Transportation Request Processing
  - Customer Freight Invoice Request Processing
  - Supplier Freight Invoice Request Processing
  - Other Web Services

Transport Order Processing

- Related to business object */SCMTMS/TOR*, services are provided to:
  - Order transportation services from carriers.
  - Exchange transportation-relevant data and create, change, or cancel freight orders, freight bookings, or freight requests for quotation.
  - Assign transportation demands to transportation capacities.
  - Create delivery proposals for further processing in the corresponding logistics execution system (SAP ERP).
  - Request the execution of transportation orders (Shipment creation in SAP ERP).
  - Request the creation, change, or cancellation of business transaction documents that are prerequisites for transportation order invoicing.
  - Request the creation or cancellation of business transaction documents that are prerequisites for an export declaration.
  - Request the processing of a trade compliance check.
  - Request the creation, change, or cancellation of loading appointments for transportation orders of the category *freight order* or *freight booking*.

Transportation Request Processing

- Related to business object */SCMTMS/TRQ*, services are provided to:
  - Processing of requests from a customer / ordering party for the provision of transportation services.
  - Receive requests for quotation from a customer and submit a forwarding quotation.
  - Create, change, or cancel a forwarding order based on an order received from a customer.
  - Query the creditworthiness of business partners.
  - Simulate, create, change, or cancel transportation requirements, based on order or delivery data received from a logistics execution system (SAP ERP).
  - Request the update or split of outbound deliveries.
  - Request the processing of a trade compliance check.

Freight Invoice Request Processing

- Customer Freight Invoice Request Processing.
- Related to Forwarding Settlement Documents, services are provided to:
  - Request invoicing for transportation services performed for a customer.
  - Request sent to billing to create one or more customer freight invoices.

- Supplier Freight Invoice Request Processing.
- Related to Freight Settlement Documents, services are provided to:
  - Request invoice verification for invoices submitted by a supplier (carrier) for ordered transportation services.
  - Request sent to invoice verification advising that a freight invoice is expected or is to be created through evaluated receipt settlement.

Other Web Services

- Other web services are provided to:
  - Provide a logistics execution system with information about TM documents and their statuses that relate to a specific business document in the logistics execution system (SAP ERP cross-system document flow).
  - Request the URL to display TM documents in SAP ERP while using invoice verification or agency business processing.

LESSON SUMMARY

You should now be able to:

- Understand the enterprise services that TM provides
- Enterprise Services in TM

Unit 12
Lesson 2

Understanding the Integration Scenarios in TM

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understanding the integration scenarios TM provides
- Explain how to monitor messages in TM

Integration Scenarios

- The following integration scenarios are available in the Enterprise Services Repository (ESR) in software component SAPTM IC:
  - TM_ERPInvoiceIntegration
  - TM_ERPOrderIntegration
  - TM_ERPSalesOrderScheduling
  - TM_ERPShipmentIntegration_Out
  - TM_ERPShipmentIntegration_In
  - TM_GTSExportDeclarationIntegration
  - TM_GTSTradeComplianceCheckIntegration
  - TM_FINCreditManagementIntegration

Invoice Integration

- TM_ERPInvoiceIntegration:
  - Transfer settlement documents from SAP TM to SAP ERP to create billing documents or accruals
  - Forwarding Settlement
  - Freight Settlement
  - Invoice Notification
  - Invoice Simulation

Order & Delivery Integration

- TM_ERPOrderIntegration:

- Transfer orders and deliveries from SAP ERP to SAP TM for transportation planning
- Propose deliveries in SAP TM that can be created in SAP ERP
- Send SAP ERP Order to SAP TM
- Send Delivery Proposal to SAP ERP
- Send SAP ERP Delivery to SAP TMSplit Outbound Delivery
- Update Outbound Delivery

- TM_ERPSalesOrderScheduling:
  - Carry out transportation planning synchronously in SAP TM during sales order processing in SAP ERP to take into account the transportation situation when calculating feasible dates for sales order items.

Shipment Integration

- TM_ERPShipmentIntegration_Out:
  - Perform transportation execution in SAP ERP based on freight orders or freight bookings after having carried out transportation planning for delivery-based transportation requirements in TM
  - Create Shipment in SAP ERP
  - Send Cancellation to SAP ERP
  - Notify of Status Change
  - Notify of Changed Delivery Assignment
  - Notify of Changed Packaging and Item Assignment

- TM_ERPShipmentIntegration_In:
  - Send shipments created in SAP ERP to SAP TM to create freight orders or freight bookings and carry out tendering in SAP TM
  - Send Shipment
  - Notify of Tendering Result

Other Integrations

- TM_GTSExportDeclarationIntegration
  - Integrate SAP TM with SAP Global Trade Services (GTS) to create an export declaration and handle the required customs processes.
- TM_GTSTradeComplianceCheckIntegration
  - Integrate SAP TM with SAP Global Trade Services (GTS) to perform a trade compliance check in SAP GTS for SAP TM business document.
- TM_FINCreditManagementIntegration

- Perform a credit limit check on a business partner in SAP TM based on credit information provided by SAP Credit Management.
- Creditworthiness Query (Synchronous Communication)
- Credit Commitment Notification (Asynchronous Communication)

Other Content

- The following interface mappings and message mappings are available in the Enterprise Services Repository (ERP) in software component SAPTM IC.
  - In the integration scenarios *TM_ERPShipmentIntegration_Out* and *TM_ERPShipmentIntegration_In*, interface mappings are used to map messages from an SAP TM system to IDocs in an SAP ERP system, and vice versa.
- A complete documentation can be found on http://help.sap.com/tm.
  - *SAP Transportation Management 9.0 → Application Help → SAP Library → SAP Transportation Management (SAP TM) → Enterprise Services and ESR Content*.

Enterprise Repository Browser (SPROXY)

Transaction SPROXY can be used in the TM system to review the ESR content.

Service interfaces, message types, and data types can be found in the following namespaces:

- http://sap.com/xi/TMS/Global
- http://sap.com/xi/TMS/
- http://sap.com/xi/TMS/UX

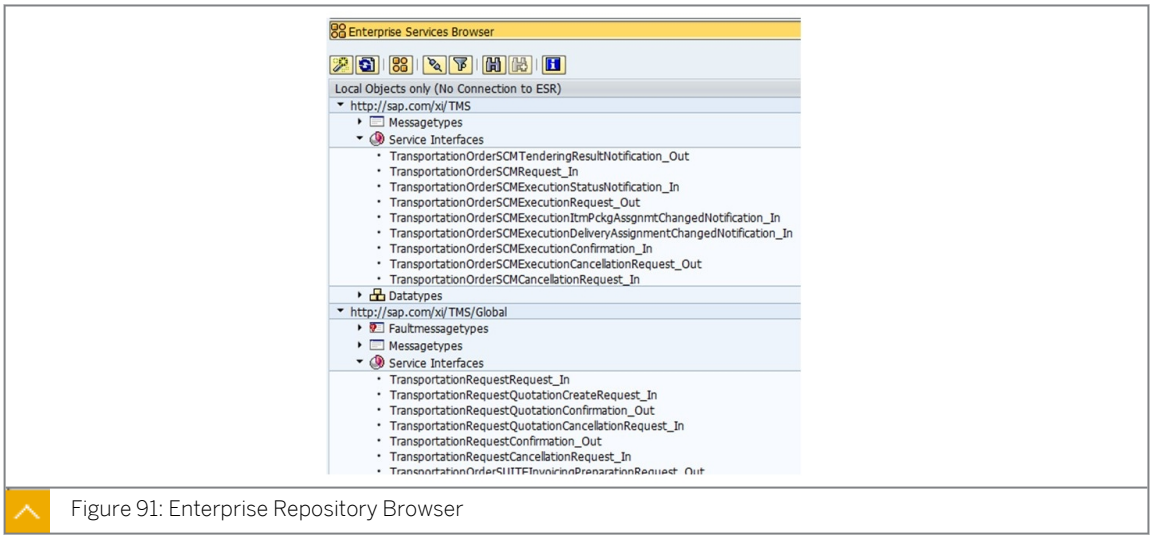


Figure 91: Enterprise Repository Browser

SPROXY can also be used to test and debug message processing:

- Message processing during implementation phase is difficult to debug, because it is done by a remote user
- For testing purposes, this can also be done in SPROXY

- Set breakpoints.
- Download a message from *SXI_MONITOR*.
- In `SPROXY`, open the service interface definition.
- Press the *Test* button (F8).
- Check *Debug* method.
- Load the downloaded message.
- Optional: Edit the message.
- Execute.

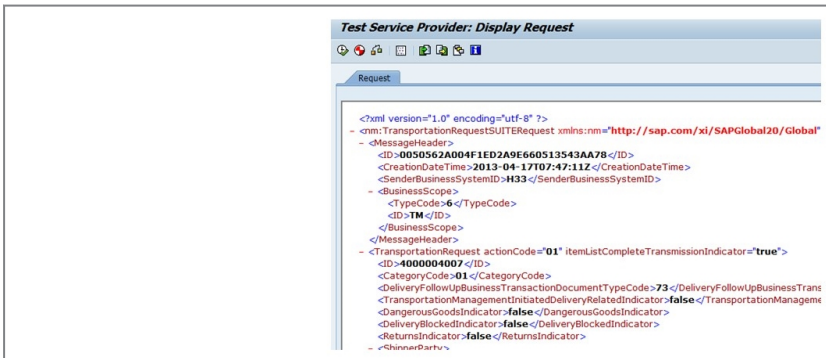


Figure 92: Test and Debug

Monitor Message in TM

XI Message Monitor (Transaction SXI_MONITOR)

First level of message monitoring.

Technical status of message can be checked:

- Processed successfully.
- Scheduled: Message is still queued. Select and goto *QRFC Monitor* via menu or transaction `SMQ2`.
- System error: In case of dumps. Dump can be found in transaction `ST22`. Messages can be reprocessed manually from here or via report *RSXMB_RESTART_MESSAGES*.
- Application error: Message has been forwarded to second level monitor. Only if it has been activated. Don't reprocess from here.

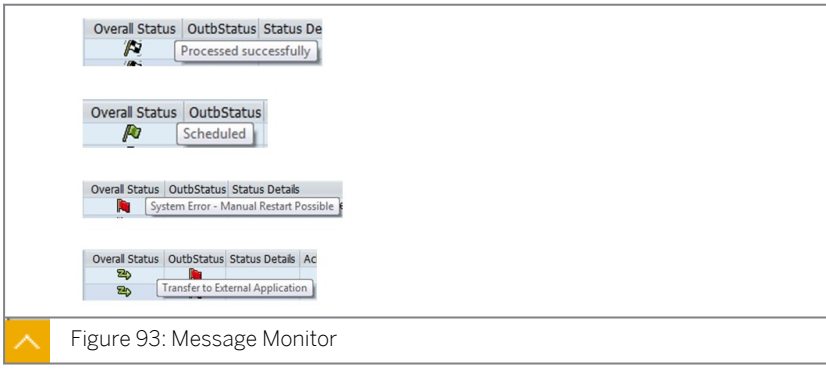


Figure 93: Message Monitor

Second level of message monitoring:

- Needs to be activated in SPRO: *Cross-Application Components → Processes and Tools for Enterprise Applications → Enterprise Services → Activate Error and Conflict Handler → .*
- Messages that were not processed due to an application error are forwarded here
- In case error has been resolved within TM, message can be reprocessed from here
  - For example, missing master data, locked documents
  - Reprocessing can also be scheduled using transaction /SAPPO/RESUBMIT
- In case error has been resolved with message update / new message, old message can be confirmed / discarded:
  - For example, wrong data used in ERP, ERP document was updated.
  - In this case, the new message will be parked in the queue (status *Scheduled* in XI message monitor) until the old one is confirmed/discarded. Afterwards the new message will be processed automatically.

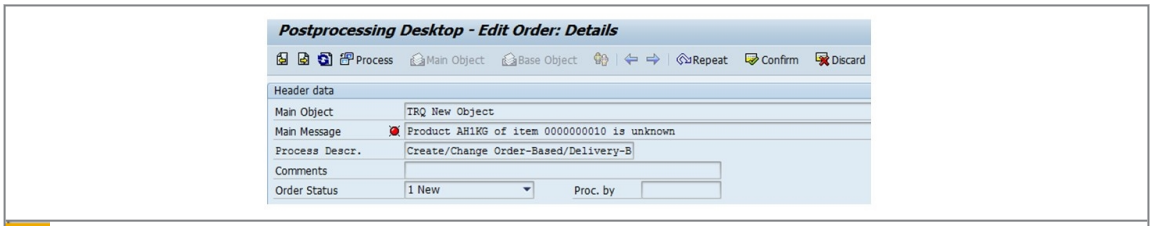


Figure 94: Second Level of Message Monitoring

Forward Error Handling

- To enable monitoring in the *Postprocessing Desktop*, Forward Error Handling is used.
  - Forward Error Handling (FEH) is a concept that enables errors, detected during the execution of asynchronous services to be processed on the provider side.
  - This concept is implemented using the Error and Conflict Handler (ECH).
  - The resolution process depends on the type of error. An error may be resolved automatically, resolved manually, or rejected and delegated back to the consumer.
  - The ECH Framework classifies errors into error categories. You can define a resolution strategy individually for each error category.
  - More documentation can be found on http://help.sap.com/tm. *SAP Transportation Management 9.0 → Application Help → SAP Library → SAP Transportation Management (SAP TM) → Enterprise Services and ESR Content → Forward Error Handling*.

LESSON SUMMARY

You should now be able to:

- Understanding the integration scenarios TM provides
- Explain how to monitor messages in TM

Unit 12
Lesson 3

Enhancing Web Services

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Understand how to enhance a TM web service
- Enhance SD Output determination
- Enhance the inbound agent in TM

TM Web Service Enhancements

Service Enhancement using SPROXY

- Define a new namespace in */nSPXNGENAPPL*.
- Open the corresponding service in `SPROXY` and navigate to the data type you would like to enhance.
- Create the enhancement.
- Save and regenerate the proxy.
- Implement the business logic in the corresponding BAdI.
- Use `SPROXY` to create new services.

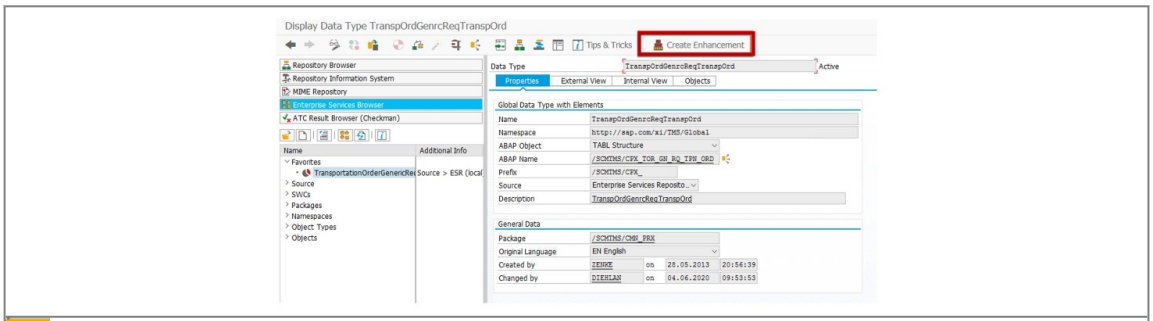


Figure 95: Create Enhancement

Service Enhancement using ESR

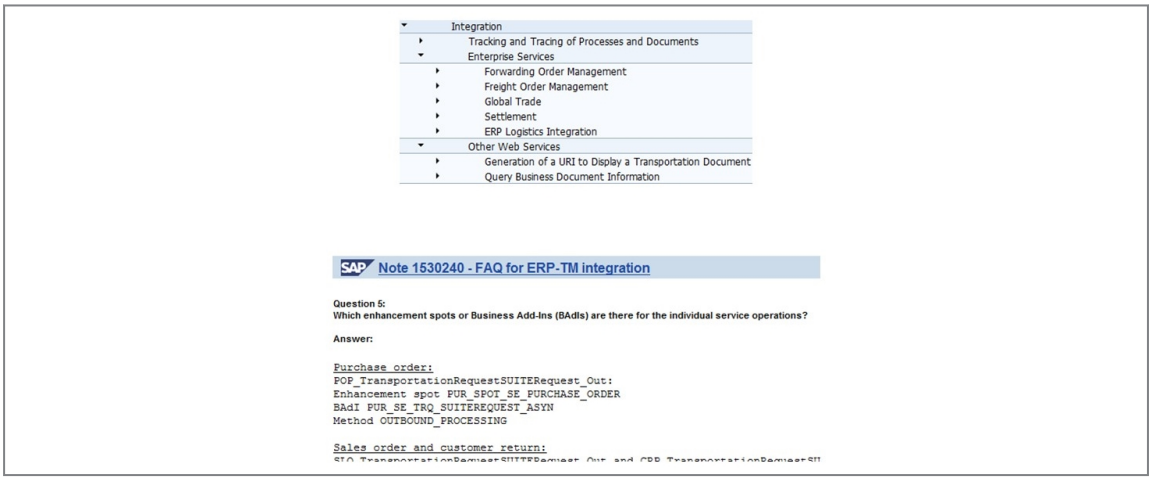
- Enterprise Services can be enhanced to transfer additional fields:
  - Standard fields which are not transferred in standard service
  - Custom fields
- The following steps are necessary to enhance the service (for example, Send an additional field from ERP to TM):

- Development in System Landscape Directory (SLD):
  - Create a Product and Software Component
  - Define Dependencies Between an EnSWCV and an Underlying SWCV
- Development in Enterprise Service Repository (ESR)
  - Import an EnSWCV into ESR
  - Create a Namespace in EnSWCV
  - Create a Data Type in the SWC
  - Create an Data Type Enhancement for TM and for ERP in the SWC
- Development in the Backend System (ERP, TM)
  - Generate the Enhancement Proxy Structure in TM and in ERP
  - Enhance the mapping in ERP (BAdI)
  - Enhance the BO and UI in TM (BOPF and FBI)
  - Enhance the mapping in TM (BAdI)

BAdIs are available to enhance message mappings

In TM, these BAdIs can be found in the Customizing Implementation Guide (`SPRO`): *SAP TM → TM Business Add-Ins (BAdIs) for TM → Integration*.

The ERP BAdIs are listed in note 1530240: FAQ for ERP-TM integration (ERP part).



The following figure shows you an example of an enhanced service.

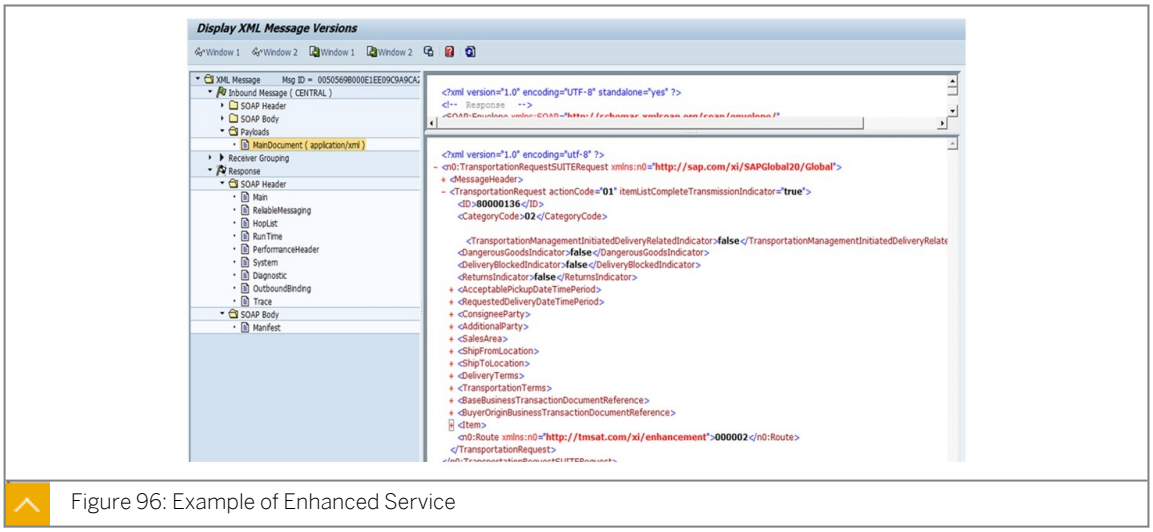


Figure 96: Example of Enhanced Service

Additional field from the ERP

Outbound proxy information to be transferred from ERP via PI to TM:

- Server Side enhancement on PI
- Enhancing Data Types in Enterprise Service Repository
- Enhancing Proxy structures on ECC and TM side

10 Steps to Enhance Web Services:

- System Landscape Directory 2 steps, Wizard supported
- Enterprise Service Repository 6 steps, Wizard supported
- Back end enhancement in TM, ECC 2 steps

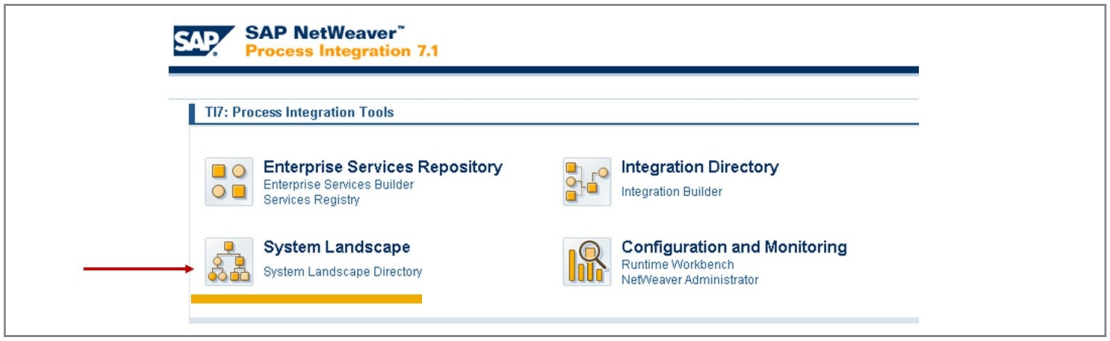
Afterwards enhanced fields need to be mapped:

- ERP BAdI to fill custom fields in message
- TM BAdI to read custom fields from message and copy to BO

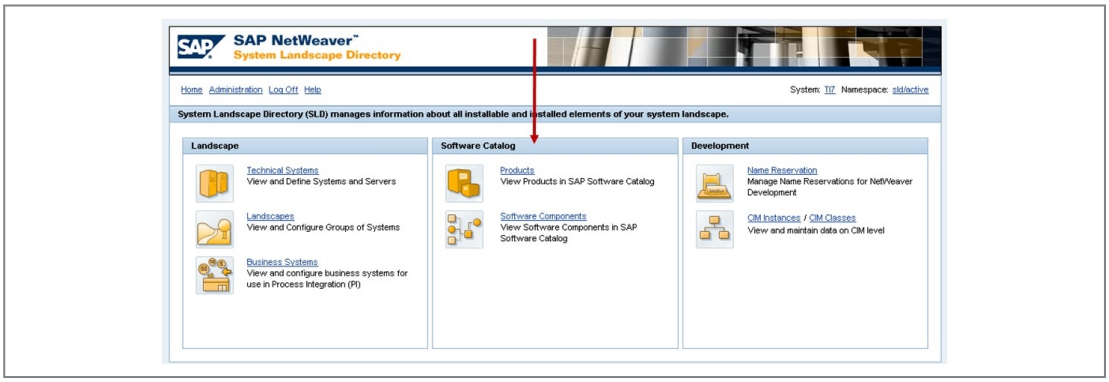
The following section shows you how to develop in the system landscape directory.

The following steps show you how to creating a product and software component:

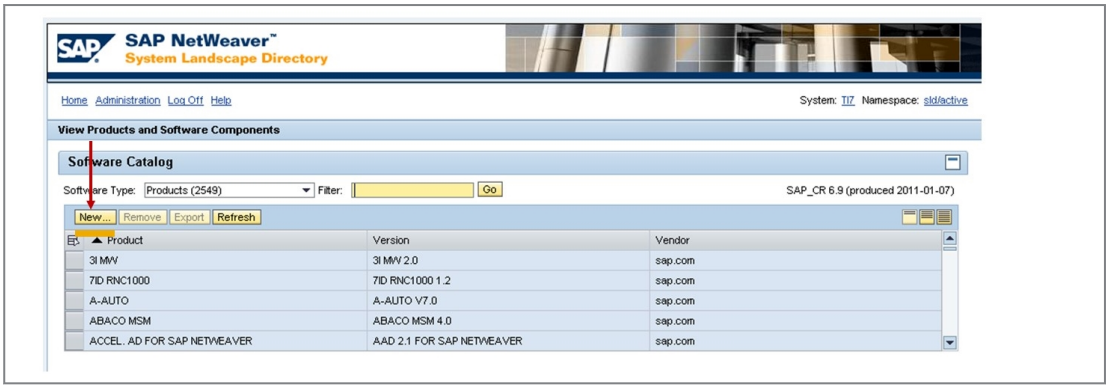
1. On the tools entry page choose *System Landscape Directory*.



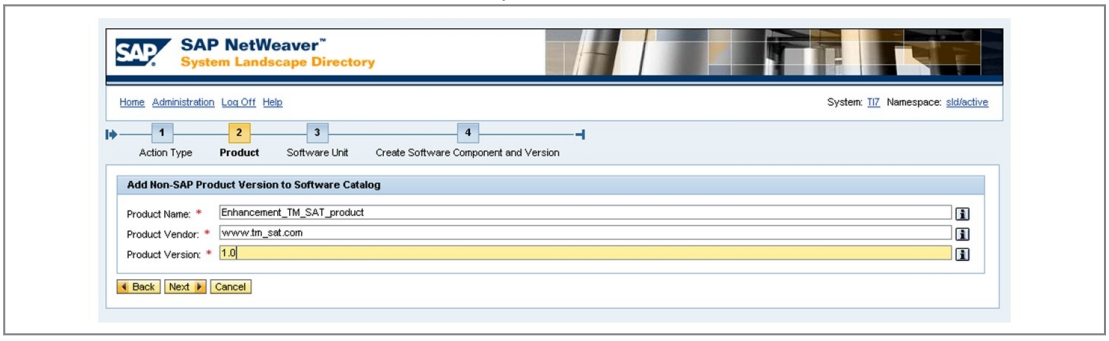
2. Under *Software Catalog* choose *Products*.



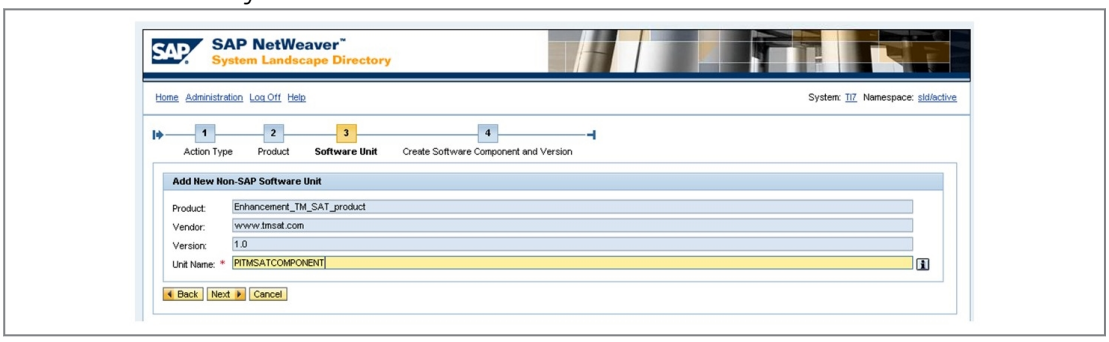
3. Choose *New Product Version*.



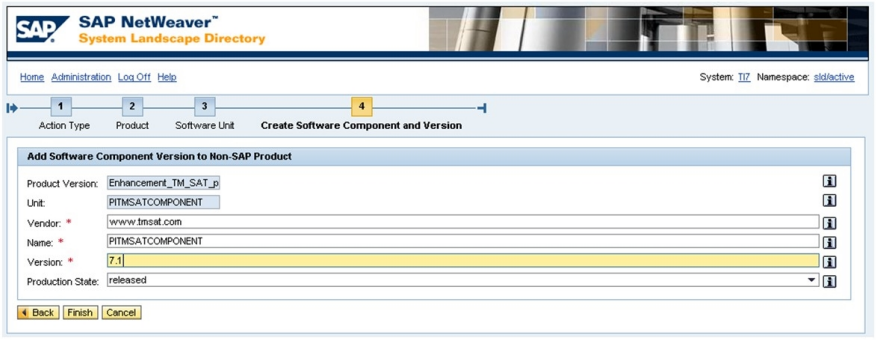
4. Enter *Name*, *Vendor*, and *Version* for the product and choose *Next*.



5. Enter the Name for your new software unit and choose *Next*.



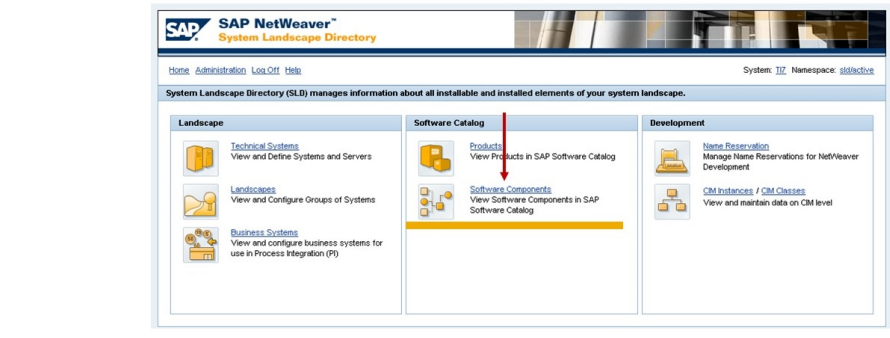
6. Enter *Name* and *Version* and choose *Finish*.



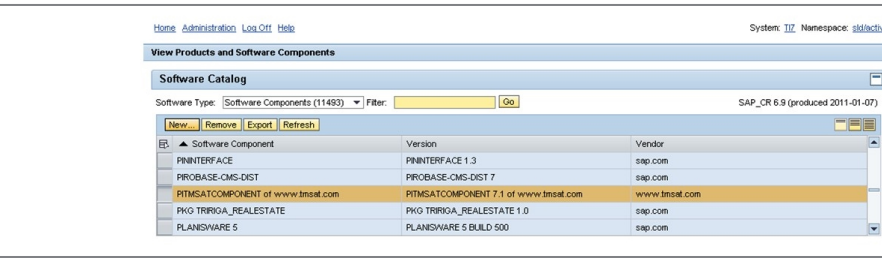
Note: To import the EnSWCV into the ESR immediately after its creation in the SLD, use the production state Released.

The following steps show you how to define dependencies between an EnSWCV and an Underlying SWCV.

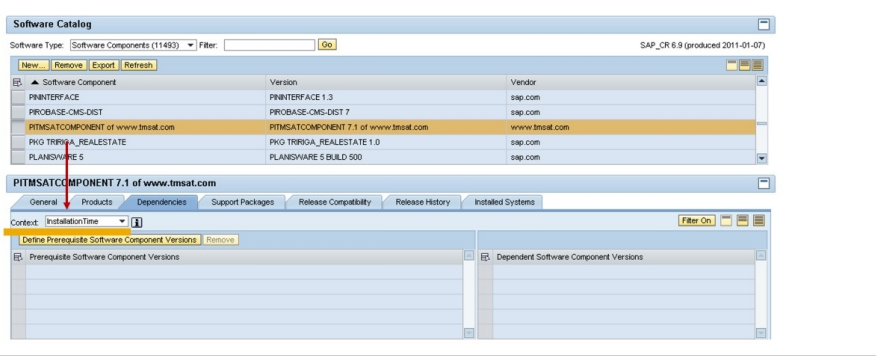
1. On the tools entry page choose *System Landscape Directory* → *Software Catalog* → *Software Components*.



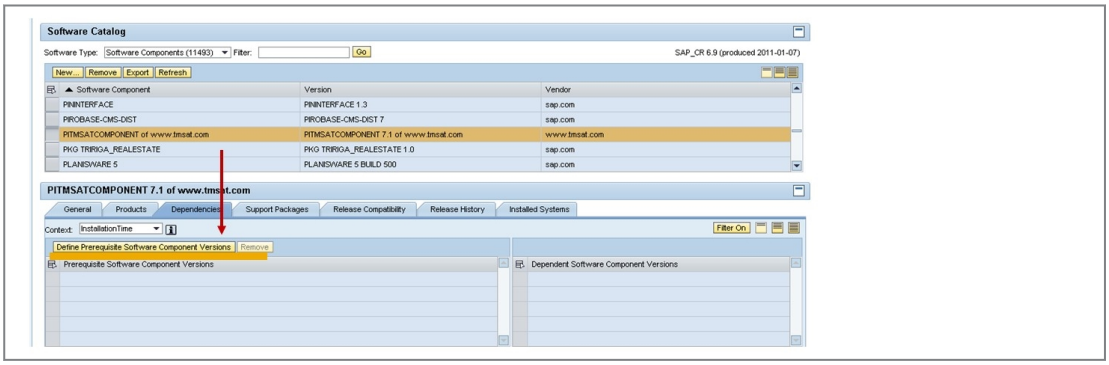
2. Choose the *EnSWCV* that you created.



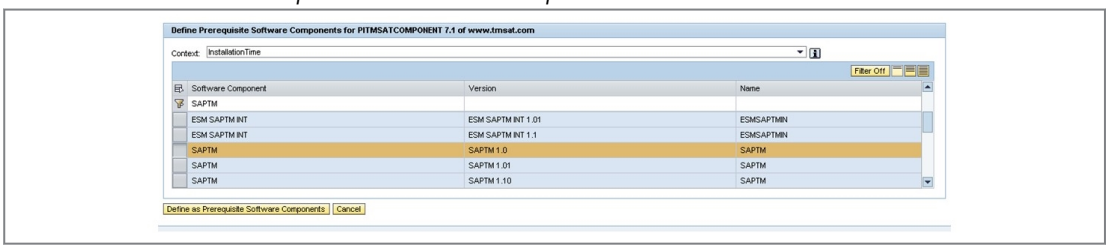
3. On the *Dependencies* tab, choose *InstallationTime* in the dropdown menu for *Context*.



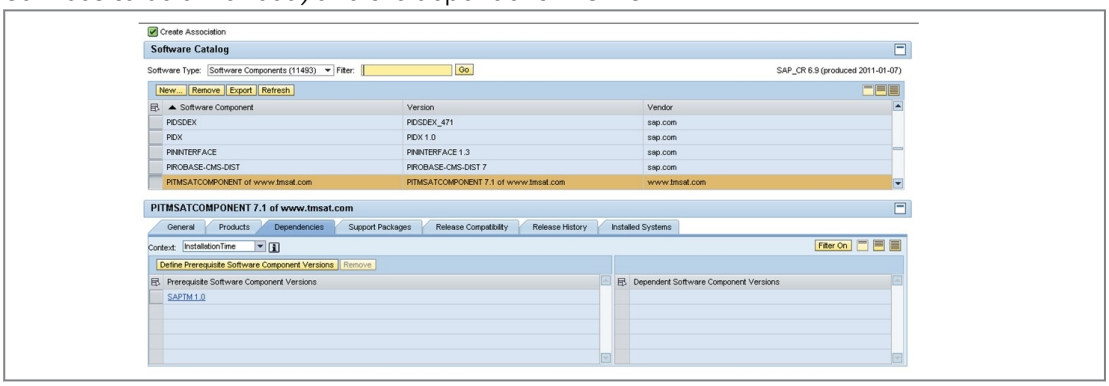
4. Choose *Define Prerequisite Software Component Versions*.



5. Filter and find the Software Component you want to define the dependency. (SAPTM 1.0) Choose *Define as Prerequisite Software Components*.

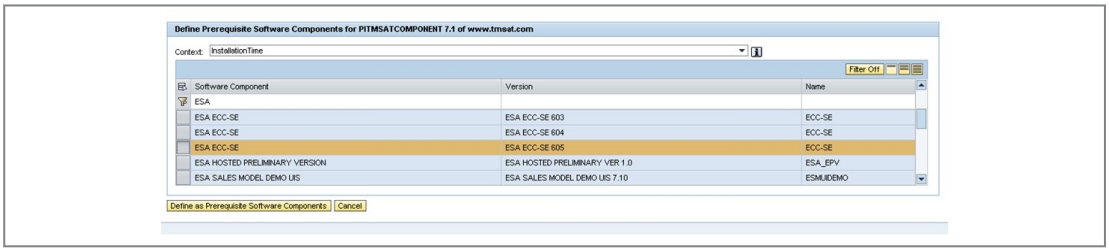


6. The system displays the dependencies between the prerequisite SWCV (that contains the services to be enhanced) and the dependent EnSWCV.

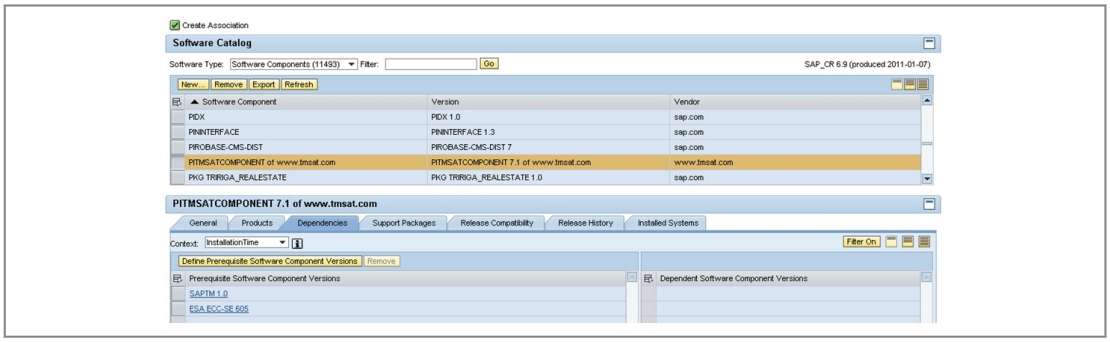


7. Create another dependency for ECC with the same steps used for TM.

8. Choose the *ESA ECC-SE 605*.

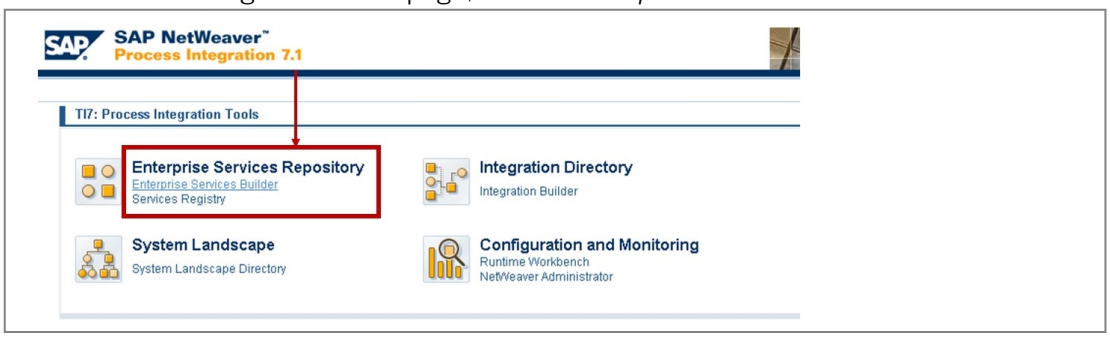


9. The following windows displays.

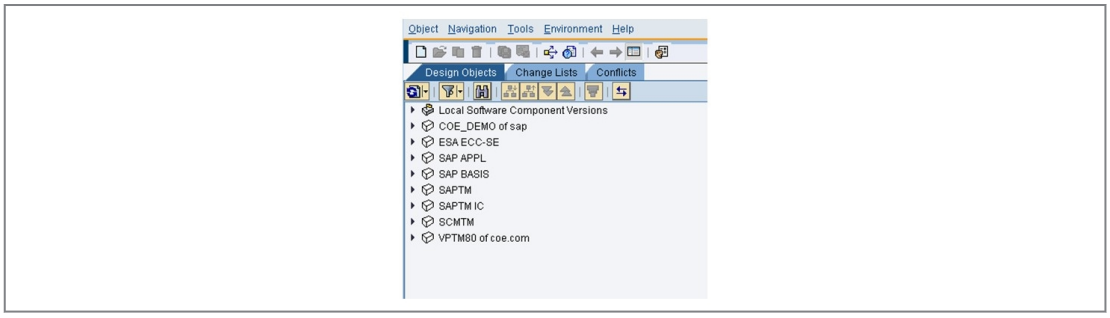


The following steps show you how to import an EnSWCV into ESR.

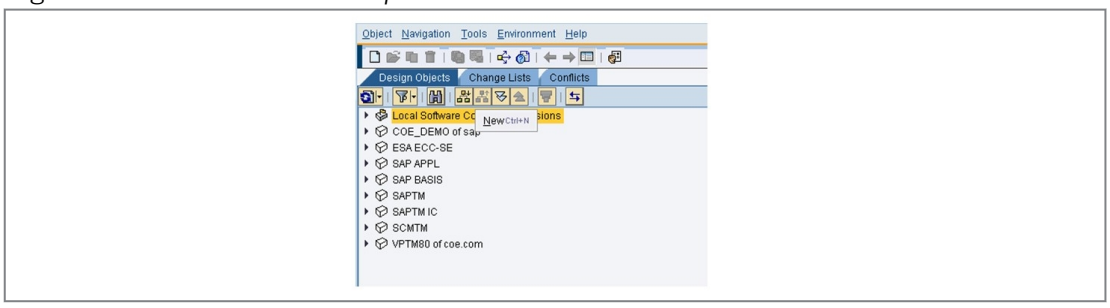
1. On the *Process Integration Tools* page, choose *Enterprise Services Builder*.



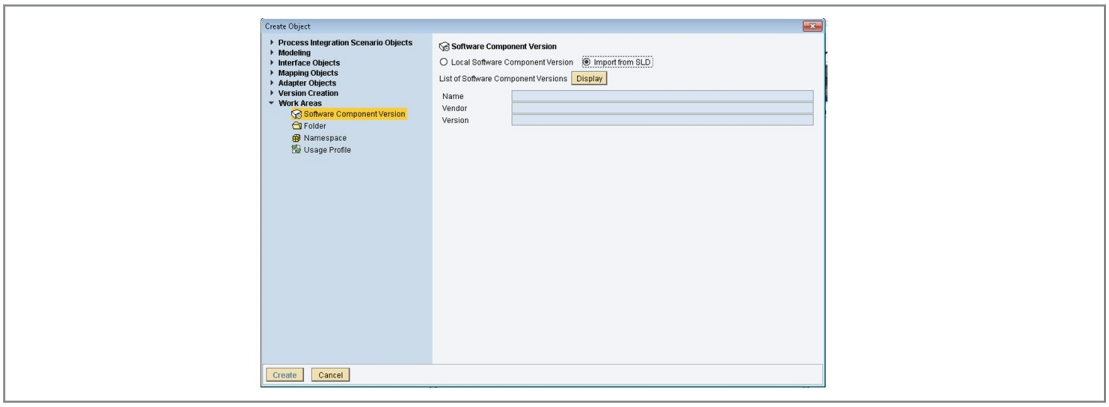
2. Choose the *Design Objects* tab.



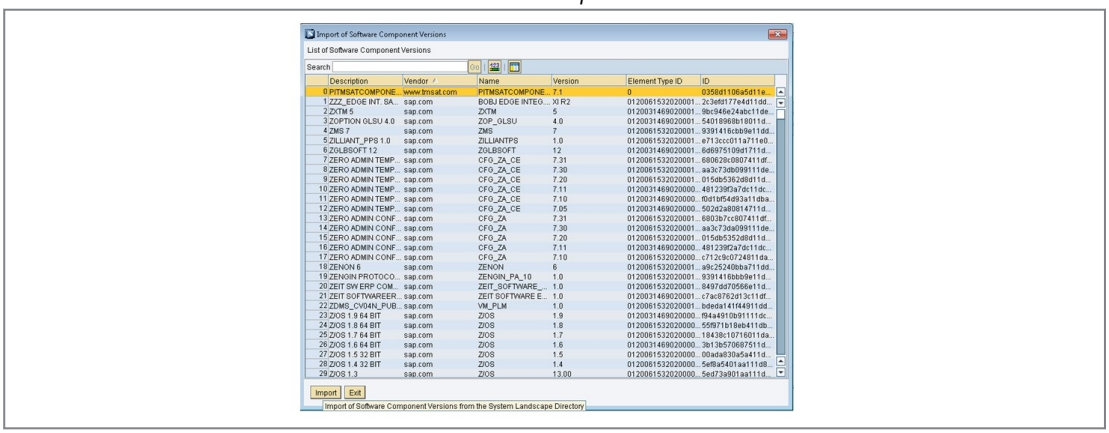
3. Right-click Local *Software Component Versions* and choose *New*.



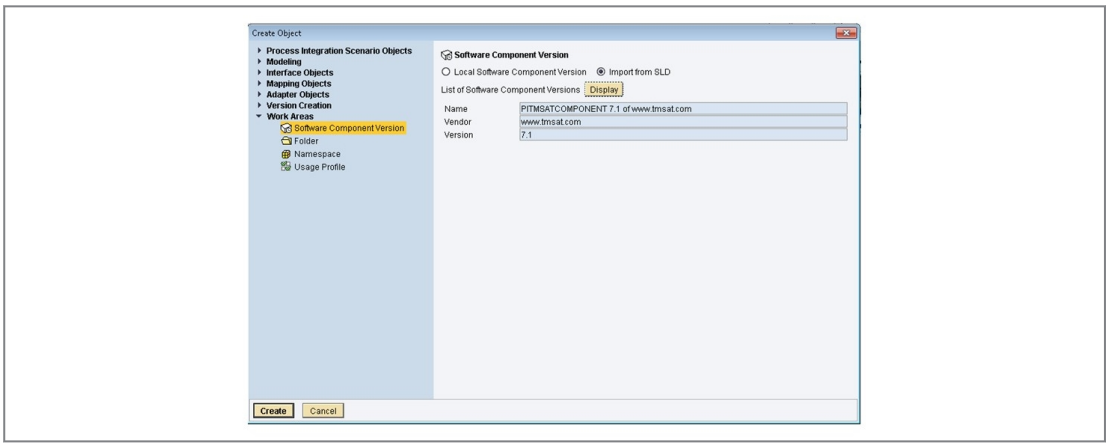
4. On the screen, choose *Import from SLD* and choose *Display*.



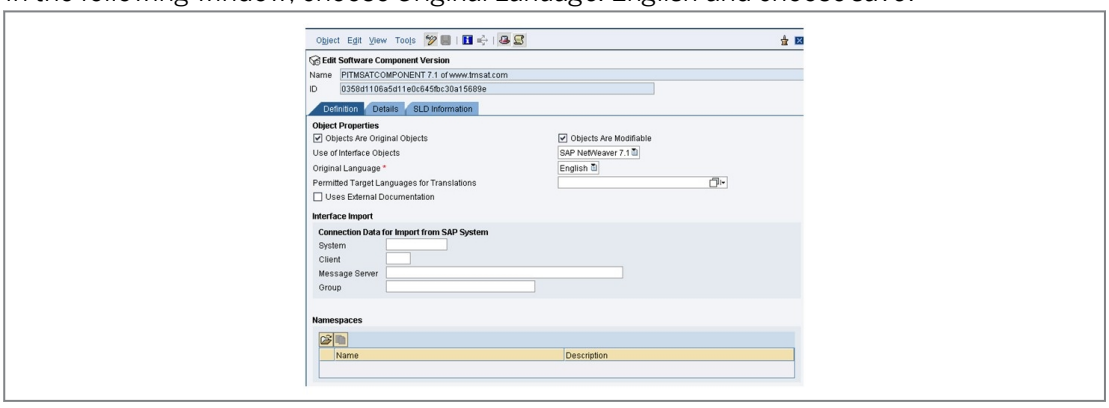
5. Select the EnSWCV from the list and choose *Import*.



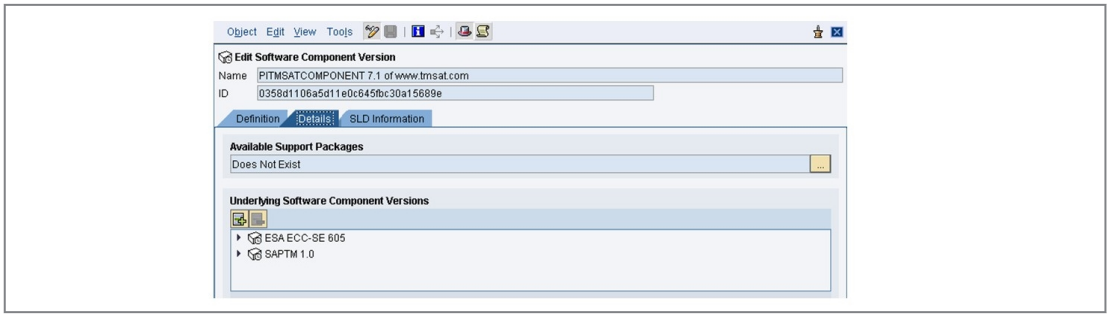
6. Choose *Create*.



7. In the following window, choose *Original Lanuage: English* and choose *save*.

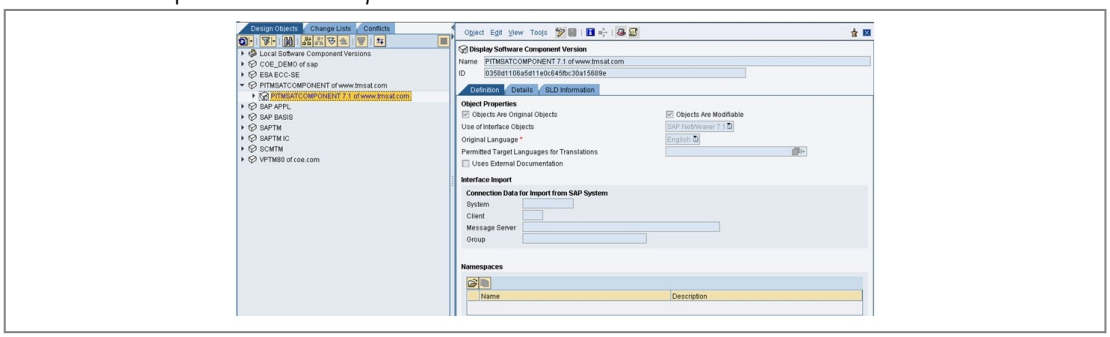


8. You see your EnSWCV in the object tree. You also see the folder Basis Objects where all namespaces and their objects from the underlying SWCV are located.

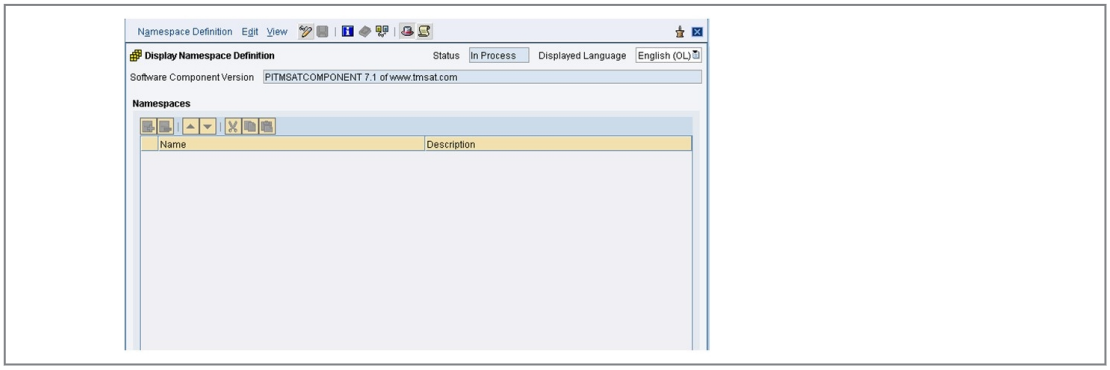


You must create a new namespace in the ESR to identify the objects in the SWCV before you develop an enhancement structure in the EnSWCV. To do so, select a unique name such as the URL address of the company, together with some additional information that which identifies the application. The following steps show you how to create a Namespace in EnSWCV.

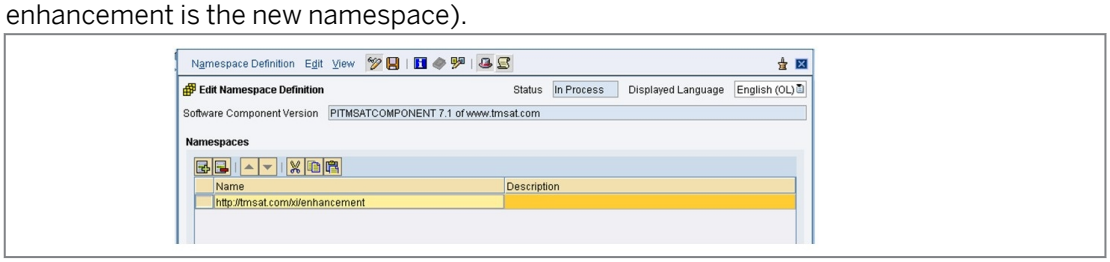
1. Under Namespaces choose *Open*.



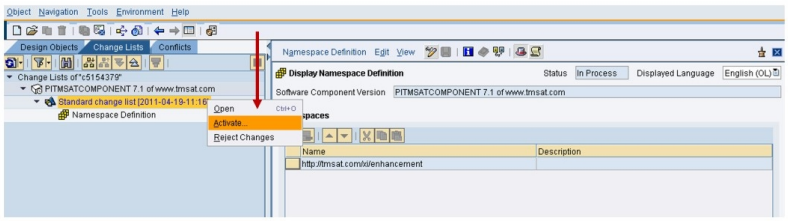
2. Choose the *Switch Between Display* and Edit Modes push button to go to the change mode in the Namespace Definition editor.



3. Insert the new namespace and save it (in the example, http://tmsat.com/xi/enhancement is the new namespace).



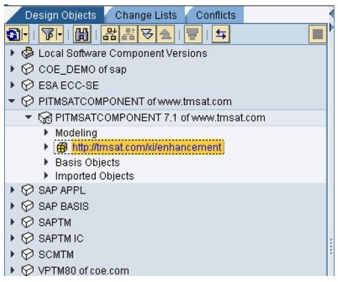
4. In the *Change Lists* tab, right click the *Standard change list* and choose *Activate*.



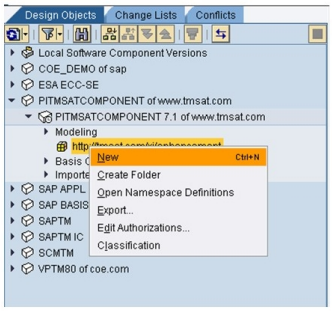
You can now use the namespace to create enhancements for your Enterprise Service.

Data types that you can use as a basis for future enhancements can be found in the namespaces in Basis Objects and in the local namespace. You can enhance complex data types using all of the namespaces from Basis Objects. The following steps show you how to create an Enhancement Data Type for TM in the SWC.

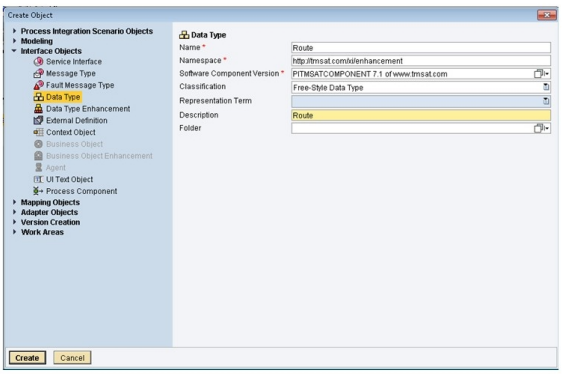
1. Navigate in the EnSWCV to the desired namespace.



2. On the *Design Objects* tab, select Namespace, right-click and choose the *New*.

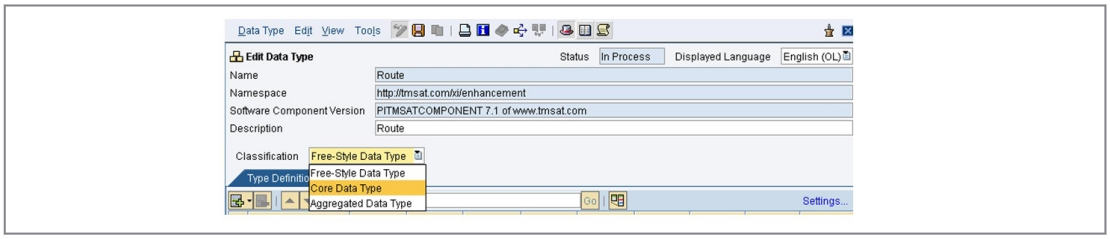


3. Select *Data Type*. Enter Name and Description of the data type and choose *Create*

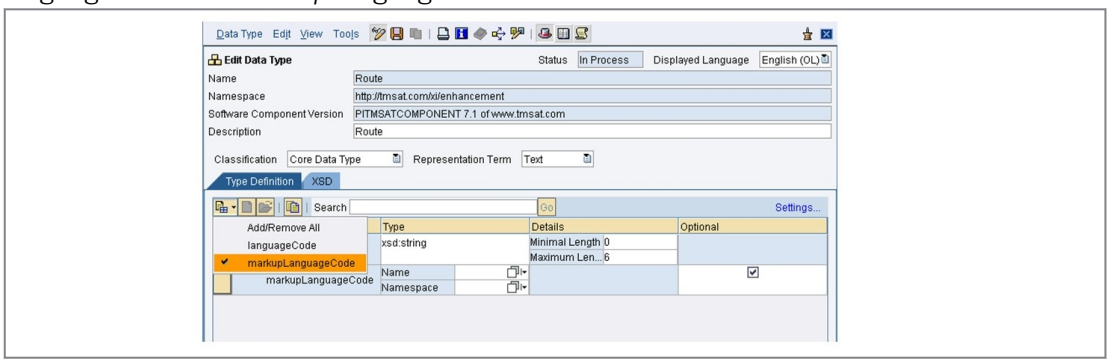


.

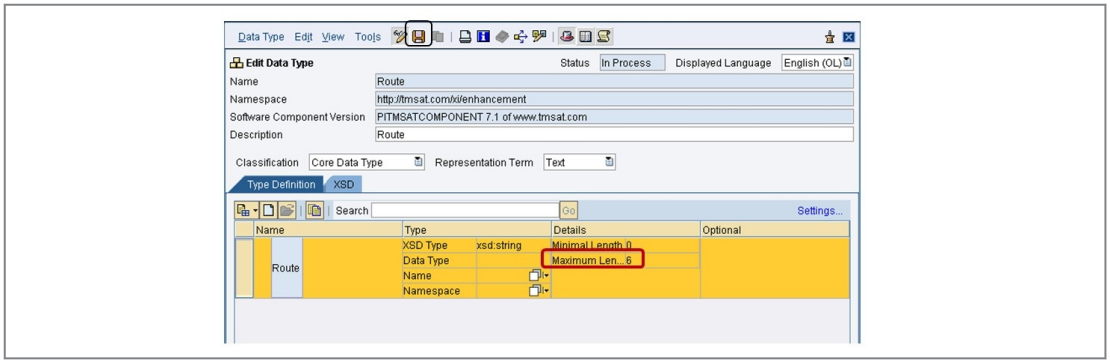
4. From the *Classification* dropdown, select *Core Data Type*.



5. Under the *Type Definition* tab select the Select/Unselect button and unselect *languageCode* and *markupLanguageCode*.

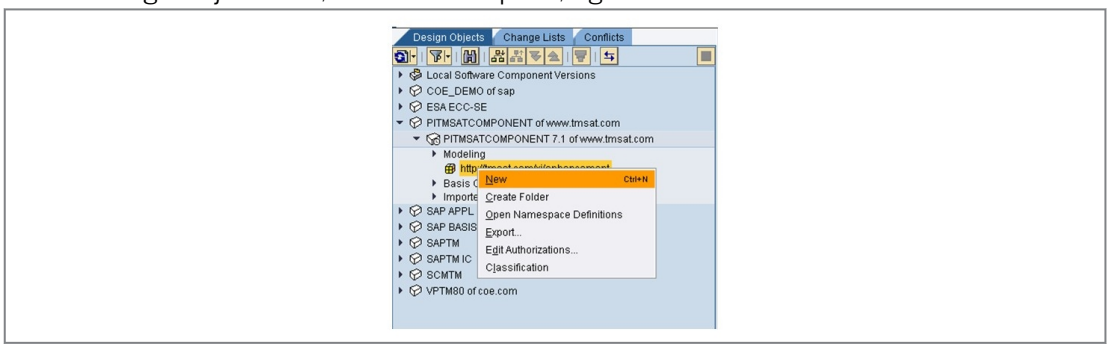


6. Enter Maximum Length (In the example 6 is used for Route) and choose *Save*.

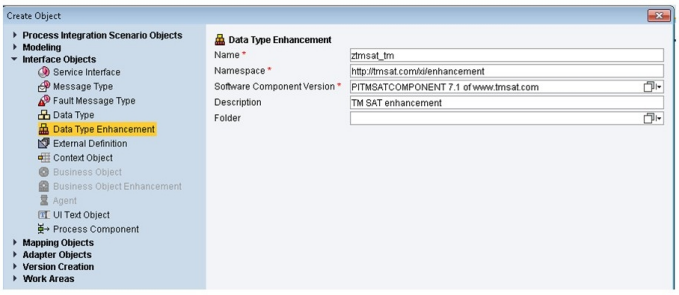


The following steps show you how to create an Enhancement Data Type for ECC in the SWC.

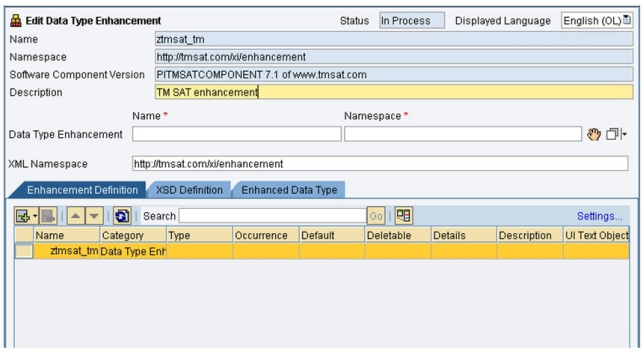
1. On the *Design Objects* tab, select Namespace, right-click and choose the *New*.



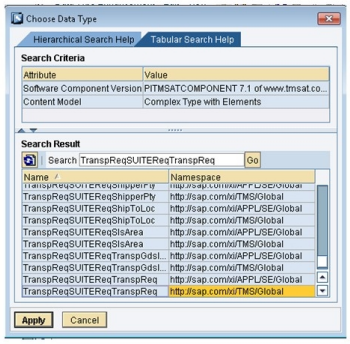
2. Select *Data Type Enhancement*. Enter Name and Description of the data type enhancement and choose *Create*. (Example name **`ztmsat_tm`** for the TM).



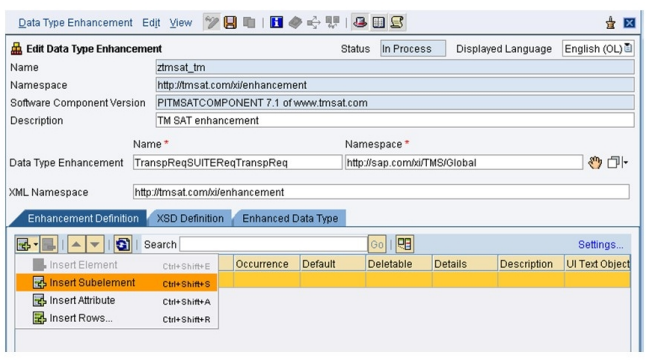
3. On the *Data Type Enhancement* tab, choose the data type to be enhanced.



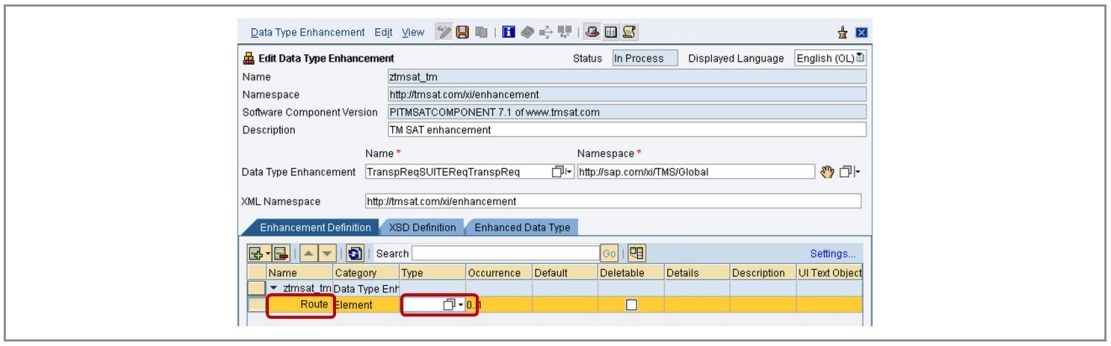
4. Select the relevant data type from the list that appears. It contains the data types that are located in the namespaces in the underlying SWCV. Choose *Apply*.



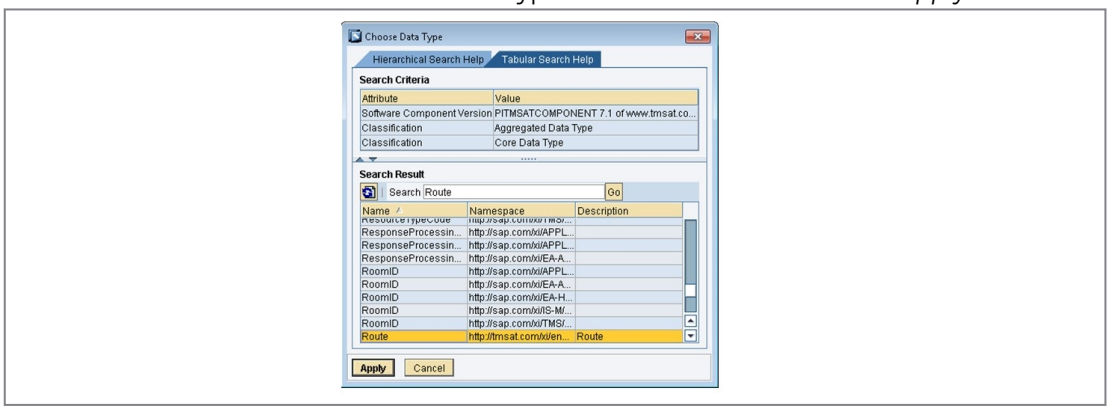
5. Choose the *Enhancement Definition* tab and create the structure of the enhancement data type. Choose *Insert Subelement*.



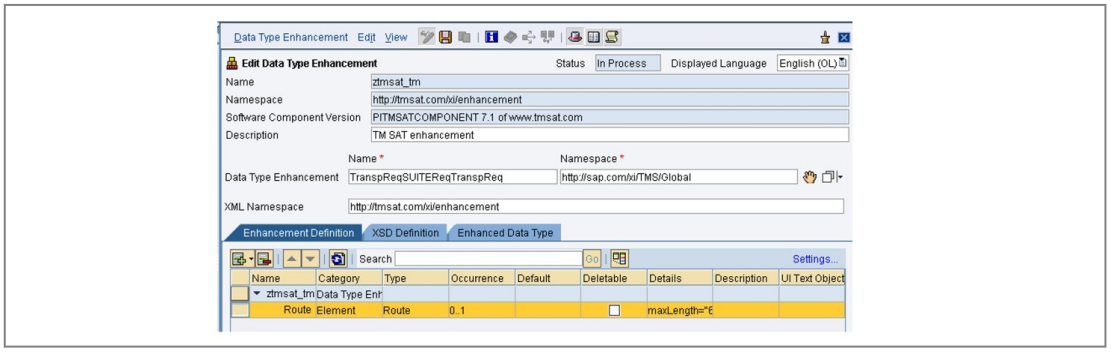
6. Insert Name (In the example Route has been used) and than choose *Type*.



7. In the selection window choose the Data Type created before and choose *Apply*.

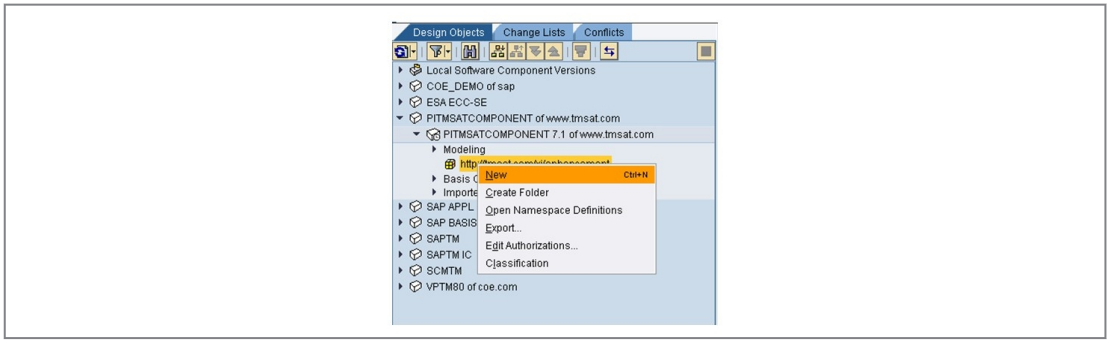


8. Choose *Save* .

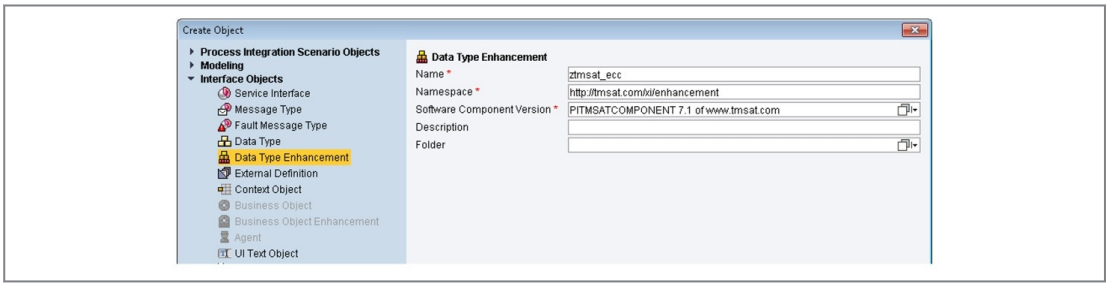


Creating an Enhancement Data Type for ECC in the SWC.

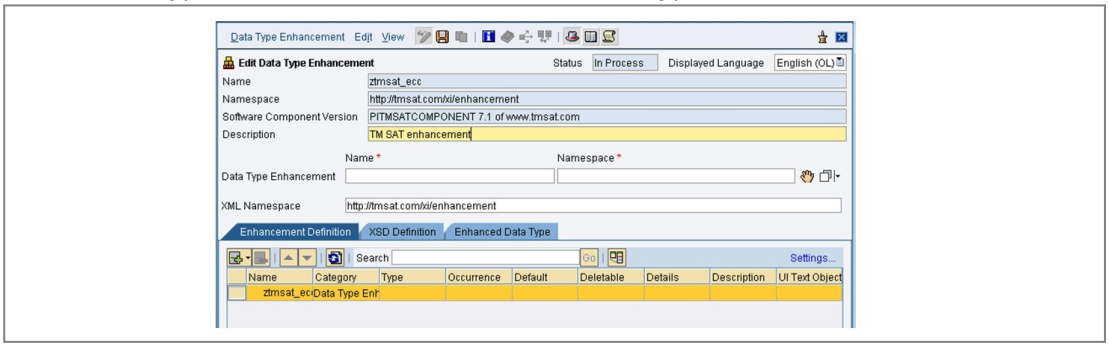
1. On the *Design Objects* tab, select *Namespace*, right-click and choose the *New*.



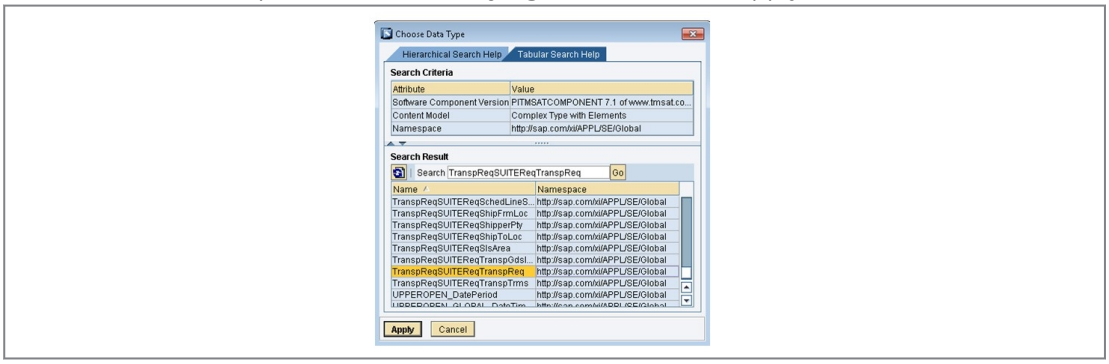
2. Select *Data Type Enhancement*. Enter Name and Description of the data type enhancement and choose *Create*. (Example choose name **`ztmsat_ecc`** for the ECC).



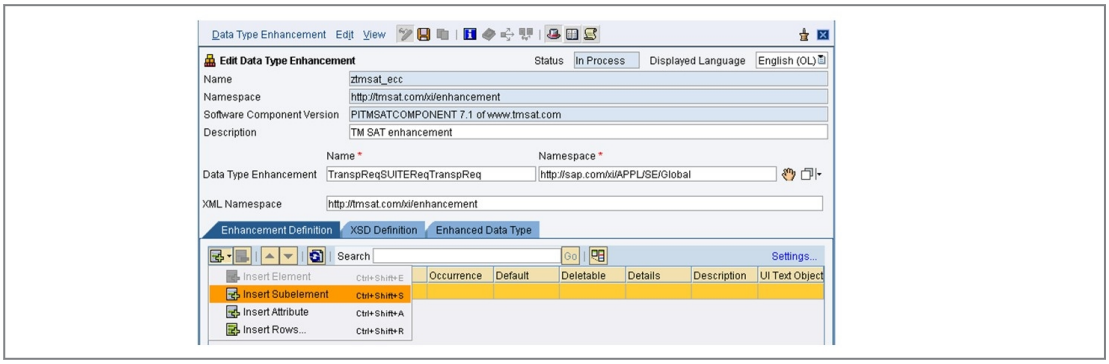
3. On the *Data Type Enhancement* tab, choose the data type to be enhanced.



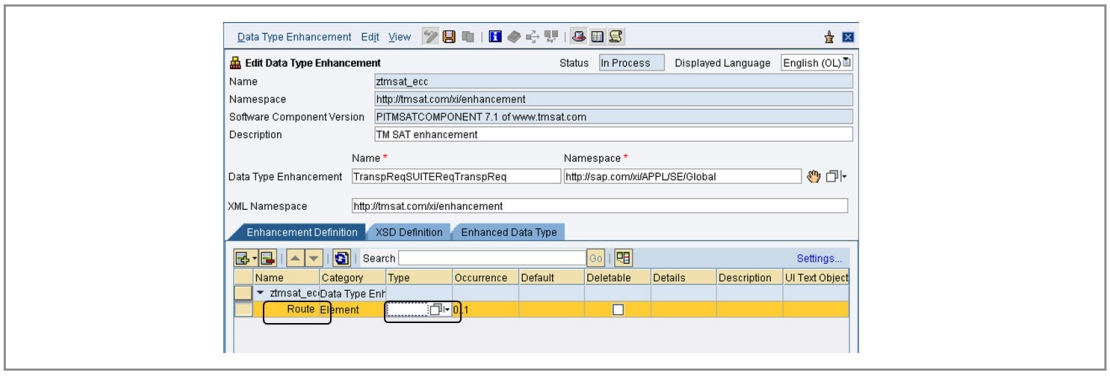
4. Select the relevant data type from the list that appears. It contains the data types that are located in the namespaces in the underlying SWCV. Choose *Apply*.



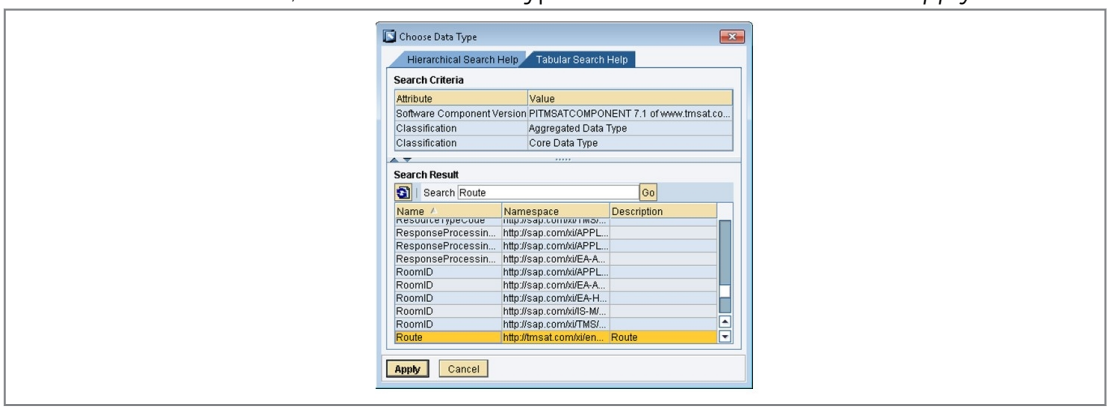
5. Choose the *Enhancement Definition* tab and create the structure of the enhancement data type. Choose *Insert Subelement*.



6. Insert Name (In the example Route has been used) and than choose *Type*.



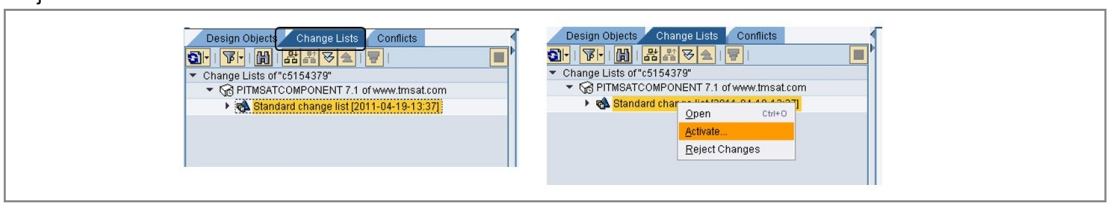
7. In the selection window, choose the Data Type created before and choose *Apply*.



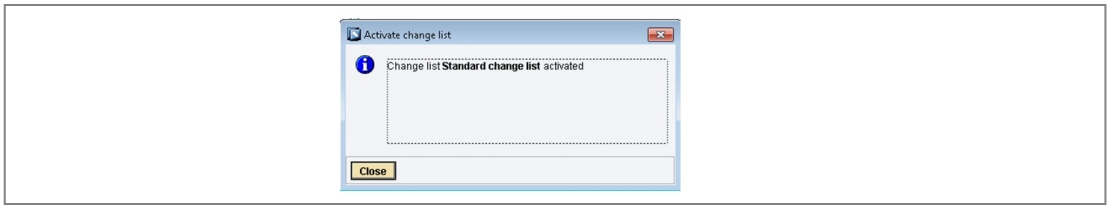
8. Choose *Save* .

Activate all objects.

1. Choose *Change Lists* tab and than choose *Standard change list* and activate all created objects.



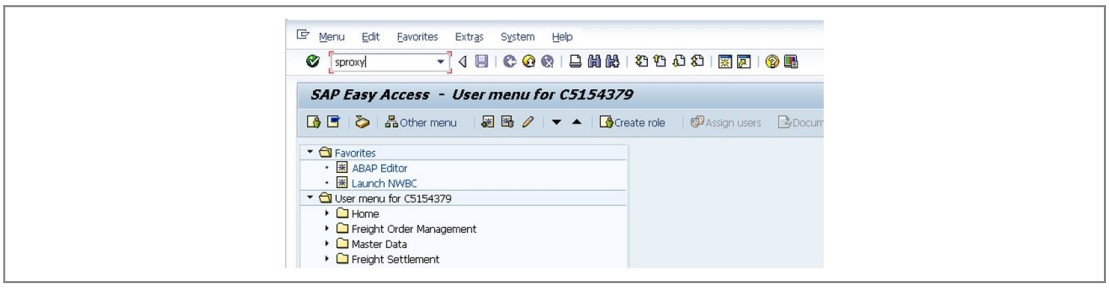
2. The system displays a confirmation message after the objects have been successfully activated.



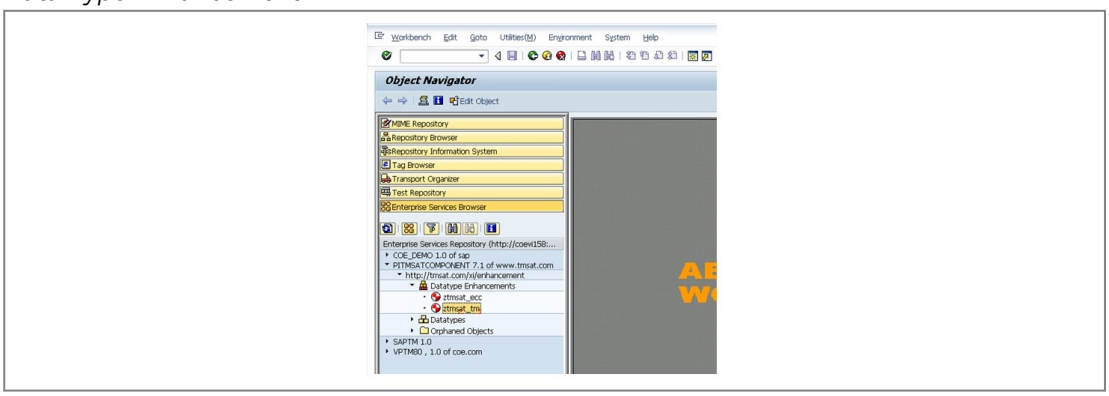
You need to carry out certain steps in the back-end system before you can use the enhanced interface objects from SAP NetWeaver SAP NetWeaver PI. In the back-end system you generate a proxy structure for the enhancement. The ABAP system generates an append structure for the service node that is to be enhanced. The system automatically adds the fields from the enhancement to the Web Services Description Language (WSDL) as optional fields.

Generating an Enhancement Proxy Structure for TM.

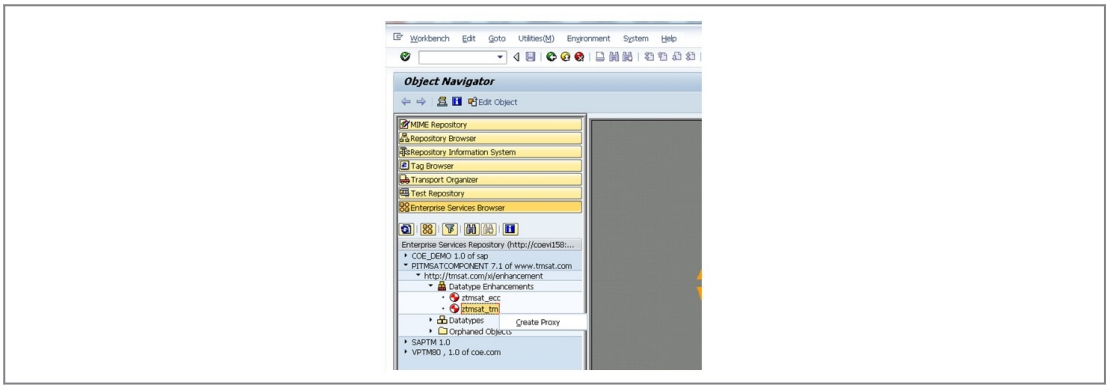
1. On the SAP Easy Access screen, enter the transaction `SPROXY`.



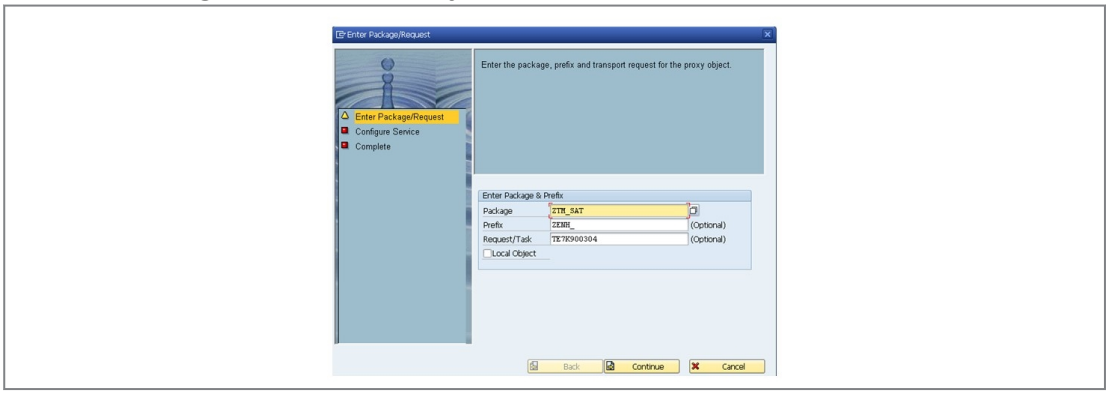
2. In the EnSWCV, navigate to the namespace that you created in ESR and choose the folder *Data Type Enhancement*.



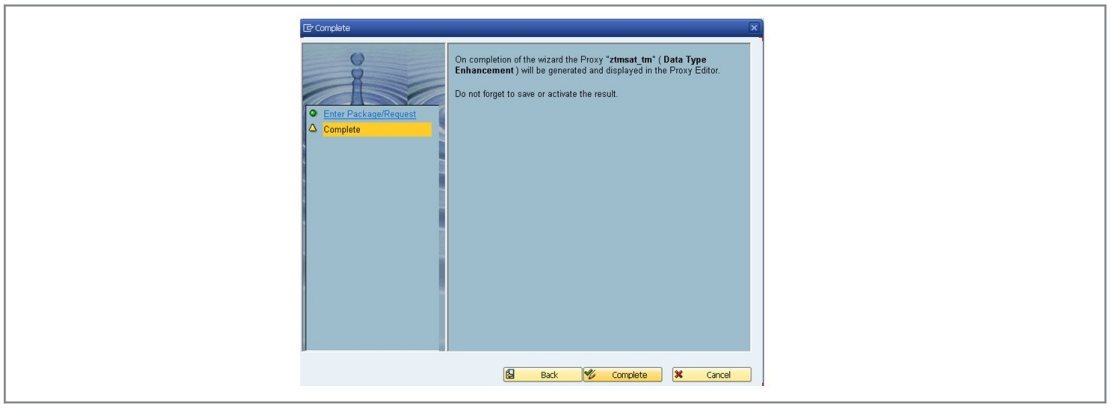
3. Double-click the enhancement name and choose *Create Proxy*. Choose *Yes* on the dialog box.



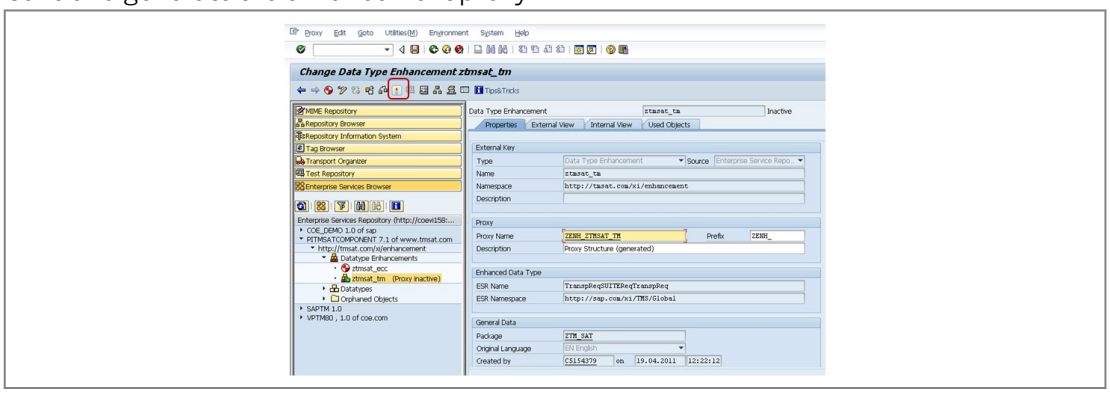
4. Enter the Package and the Prefix for your enhancement and choose *Continue*.



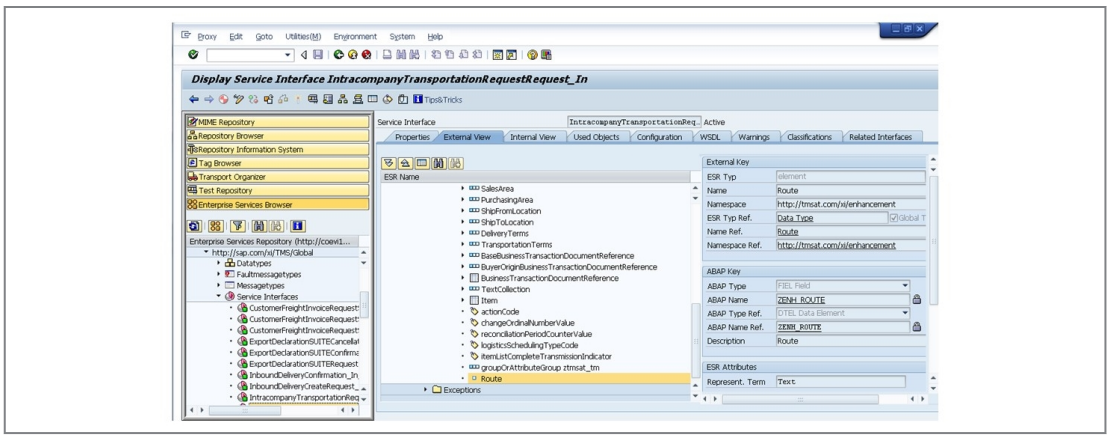
5. Choose *Complete*.



6. Save and generate the enhancement proxy.

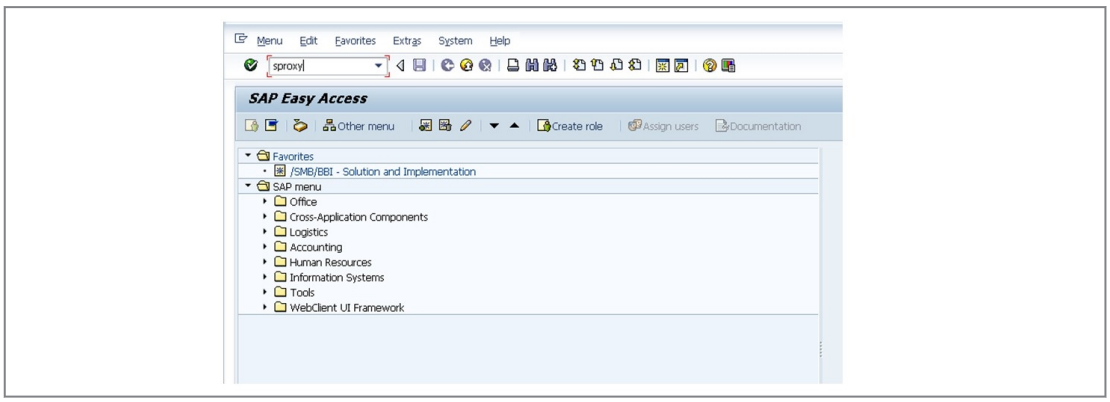


7. You can now see the enhancement in the Service Interface.

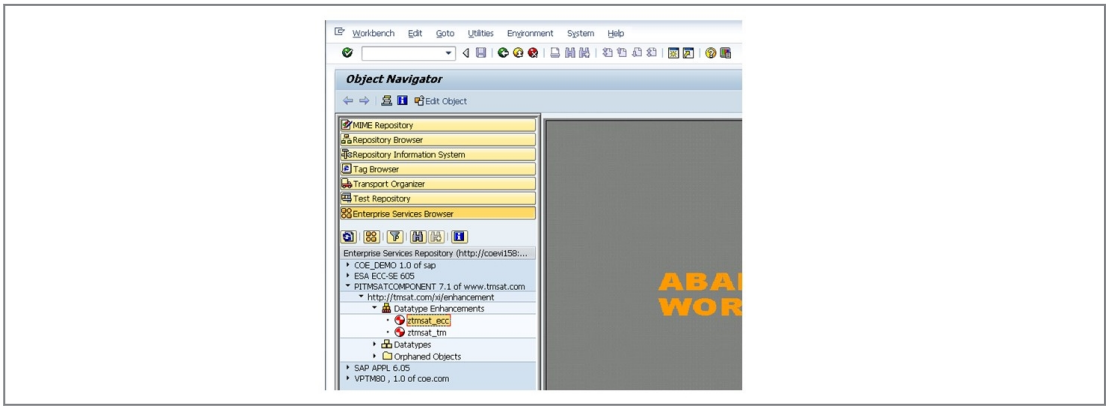


Generating an Enhancement Proxy Structure for ECC.

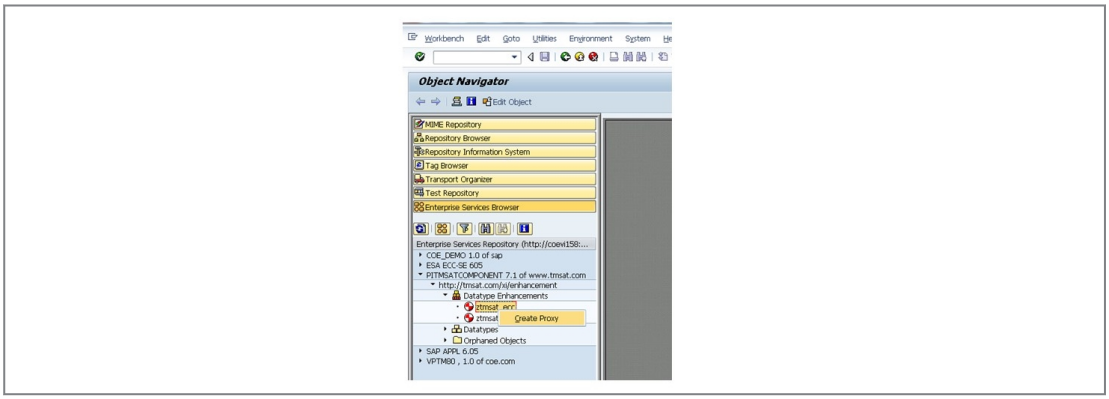
1. On the SAP Easy Access screen, enter the transaction `SPROXY`.



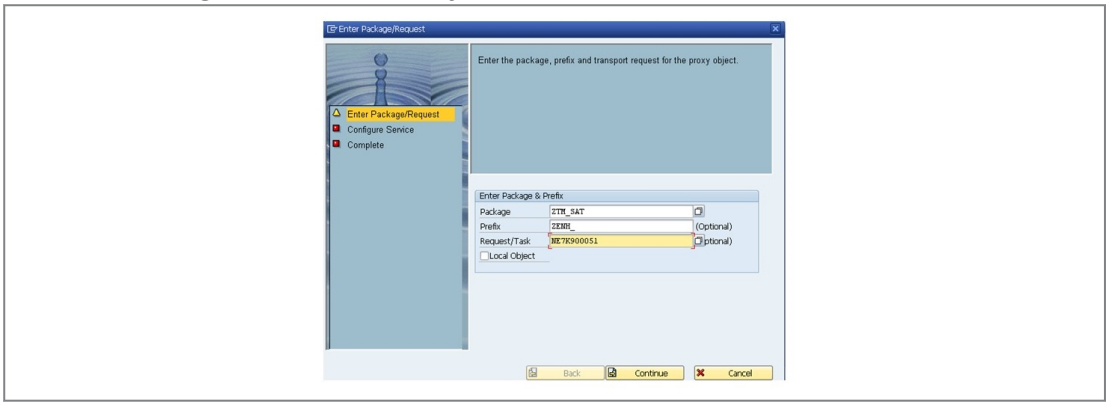
2. In the EnSWCV, navigate to the namespace that you created in ESR and choose the folder *Data Type Enhancement*.



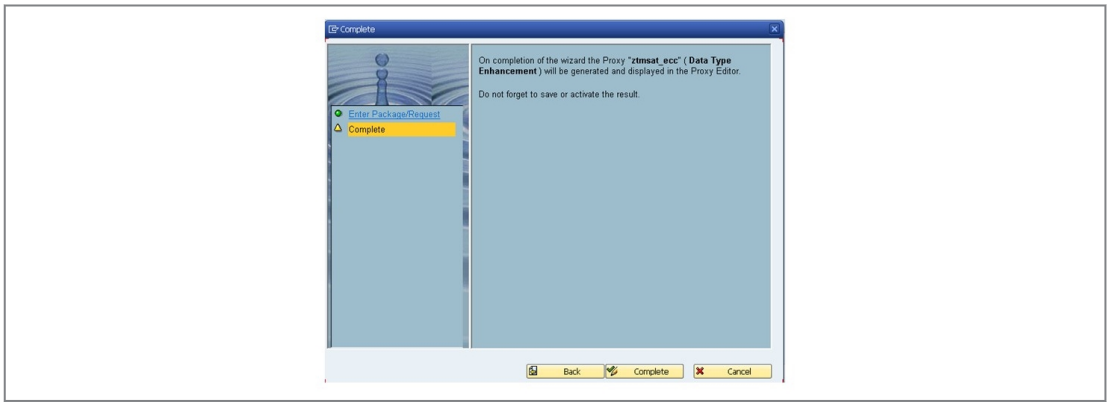
3. Double-click the enhancement name and choose *Create Proxy*. Choose *Yes* on the dialog box.



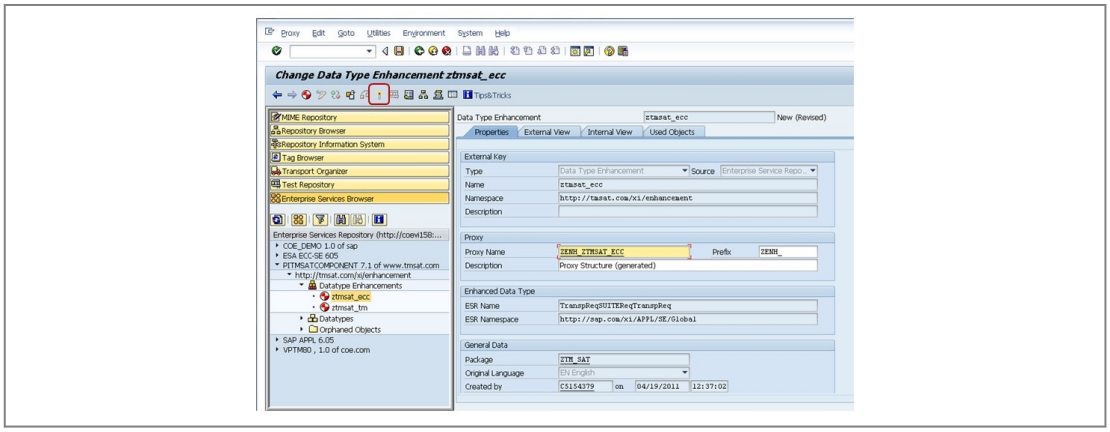
4. Enter the Package and the Prefix for your enhancement and choose *Continue*.



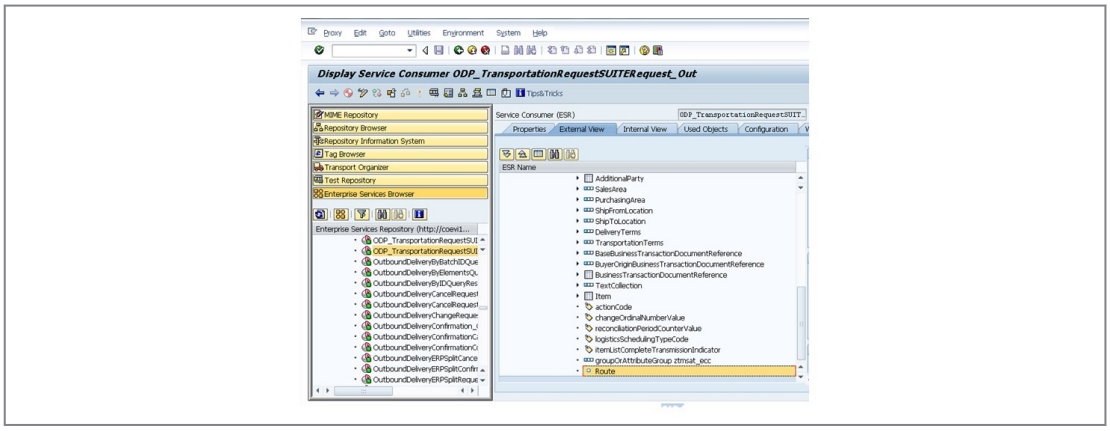
5. Choose *Complete*.



6. Save and generate the enhancement proxy.

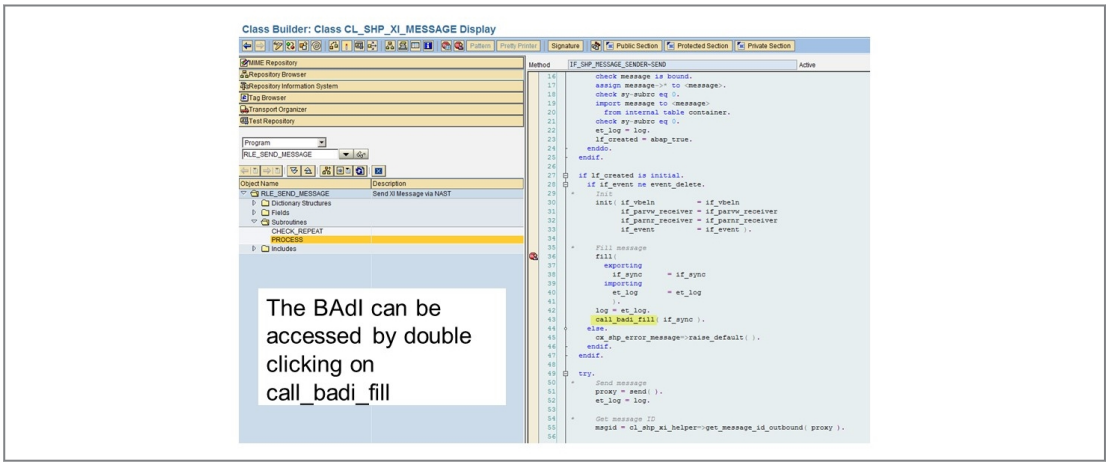


7. You can now see the enhancement in the Service Interface.

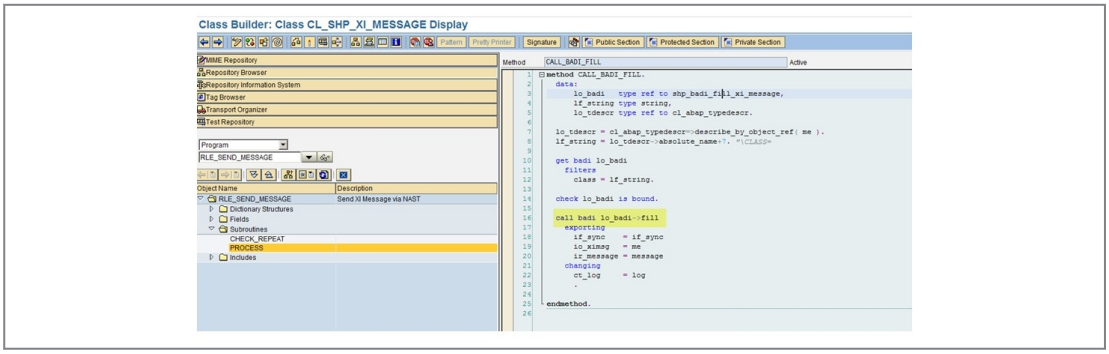


Implement BAdIs to map enhanced fields

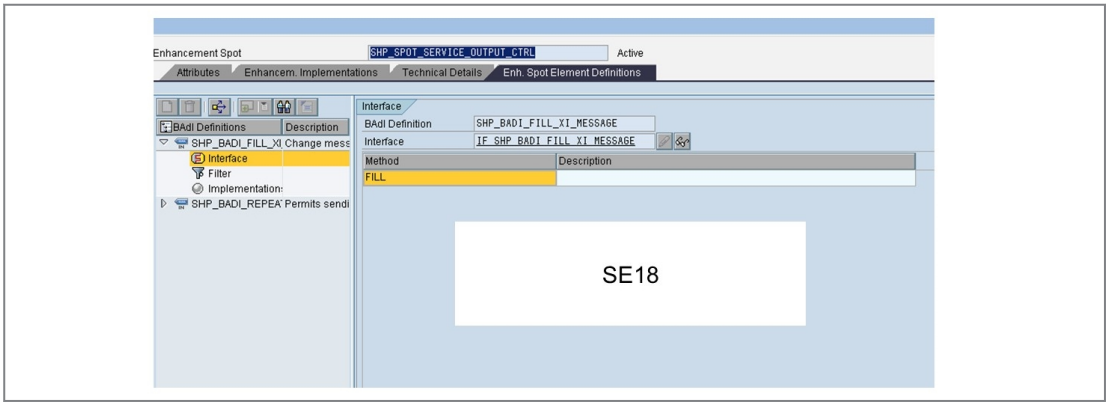
1. BAdI *IF_SHP_BADI_FILL_XI_MESSAGE (1)*.



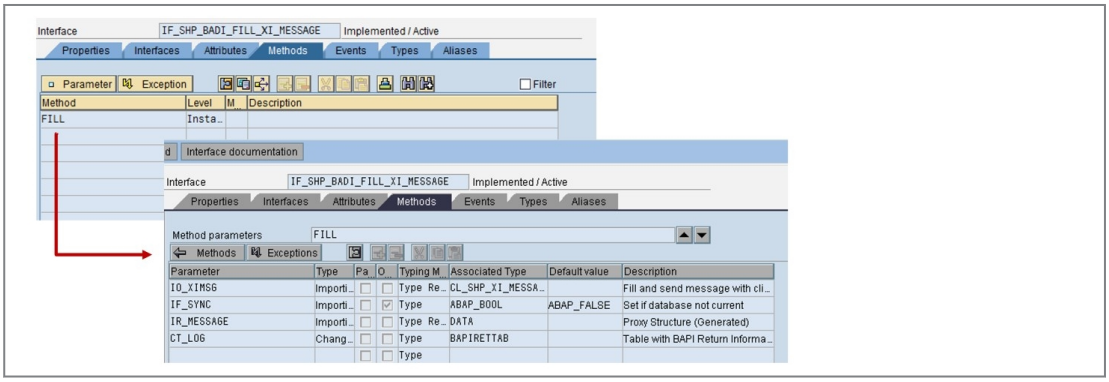
2. Double-click on *lo_badi → fill*.



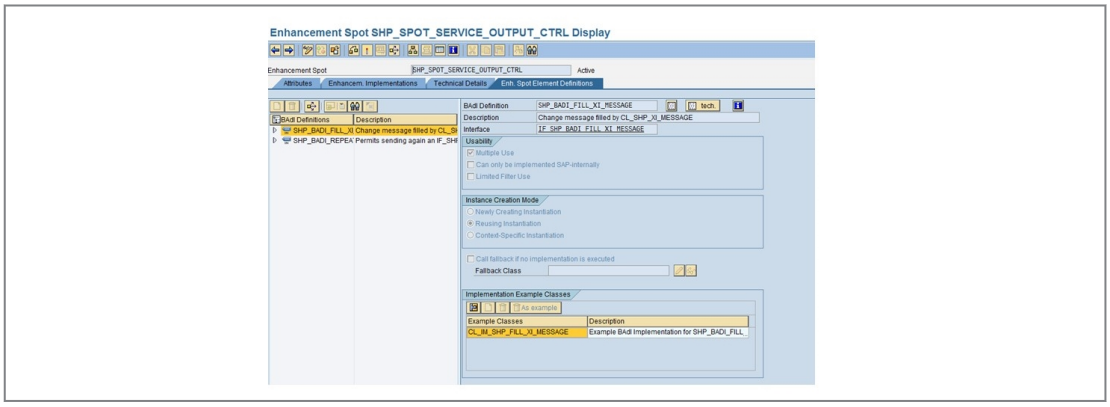
3. In `SE18` choose implementation example class.



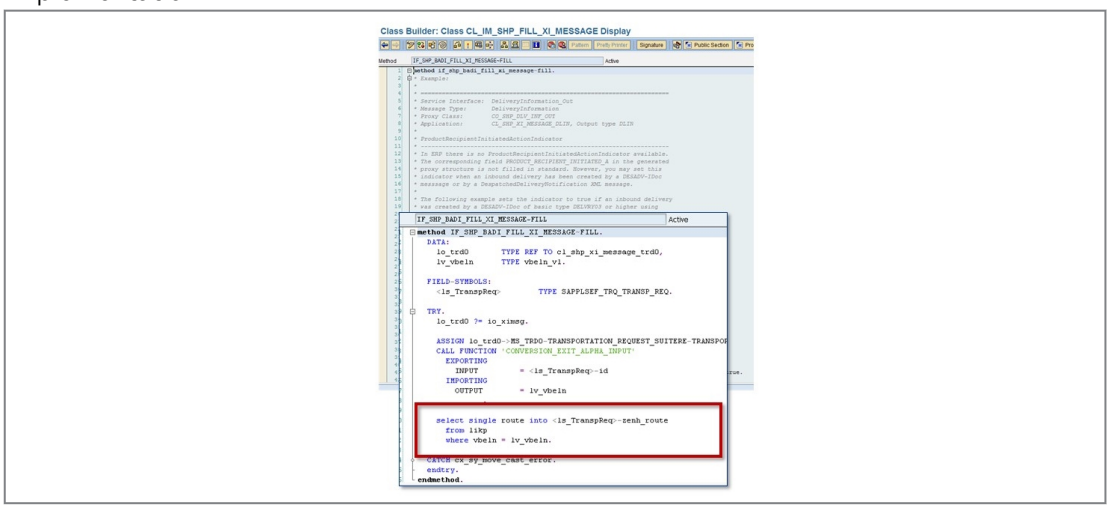
4. BAdI *IF_SHP_BADI_FILL_XI_MESSAGE (4)*.



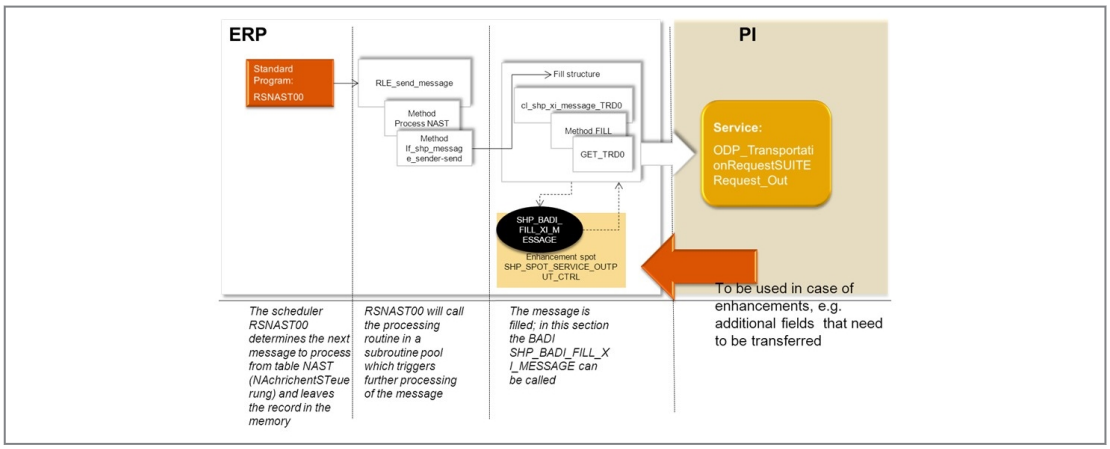
5. In `SE18` choose implementation example class.



6. Make the necessary coding for filling the additional field and activate the BAdI implementation.

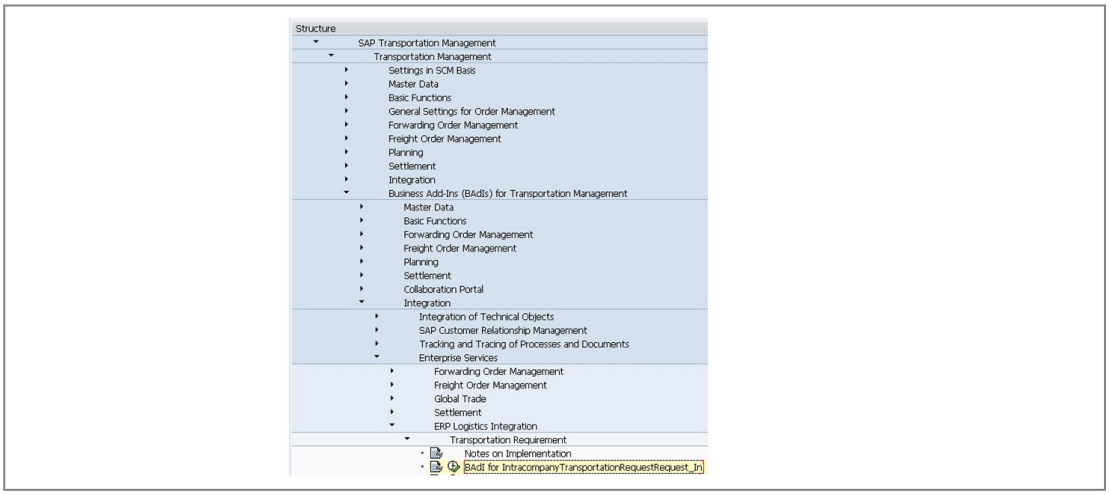


7. A program (usually `RSNAST00`) polls the table NAST for unprocessed messages and activates the program that has been dedicated to process that message: in this case '*RLE_SEND_MESSAGE*'.

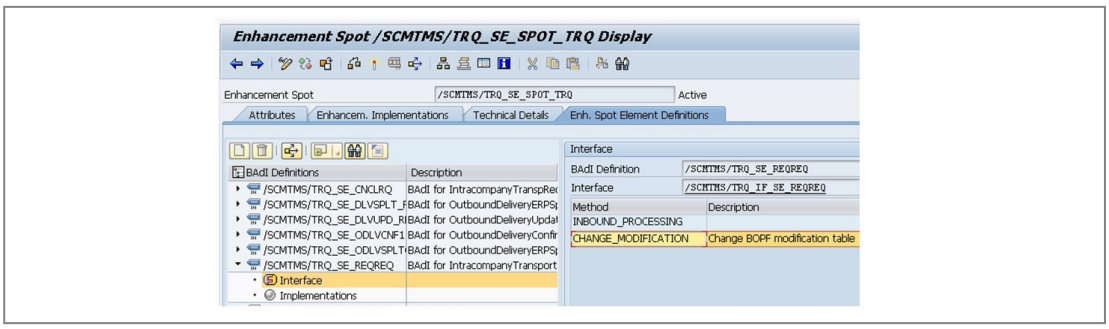


Create implementation from `SPRO`.

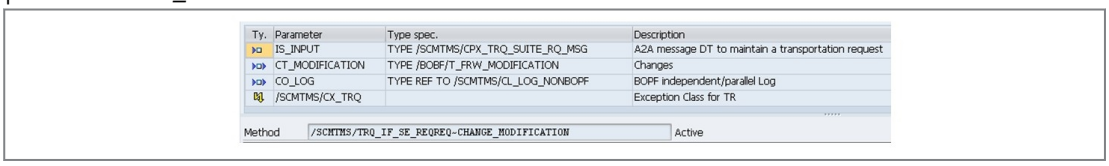
1. BAdI: */SCMTMS/TRQ_SE_REQREQ*.



2. Method: *CHANGE_MODIFICATION*.



3. From method interface, read custom fields from message in importing parameter *IS_INPUT* and change BOPF modification table to map custom fields to BO in changing parameter *CT_MODIFICATION*.



LESSON SUMMARY

You should now be able to:

- Understand how to enhance a TM web service
- Enhance SD Output determination
- Enhance the inbound agent in TM

Learning Assessment

1. Which communication technique is used mostly in TM for B2B communication?

   Choose the correct answer.

   - ☐ **A** Webservices / SOAP services.
   - ☐ **B** Native RFC calls.
   - ☐ **C** IDOCs.
   - ☐ **D** ABAP Push channels.

2. How can you monitor the webservice based communication?

   Choose the correct answer.

   - ☐ **A** Via transaction `BD87`.
   - ☐ **B** Via transaction `BOBT`.
   - ☐ **C** Via transaction `SXMB_MONI` or `SRT_MONI`.
   - ☐ **D** In the application log.

3. Which statement is correct? (multiple answers possible)

   Choose the correct answers.

   - ☐ **A** You can enhance a webservice in transaction `SPROXY`.
   - ☐ **B** You cannot enhance a standard webservice. You must always create a copy.
   - ☐ **C** You can enhance a webservice using ESR.
   - ☐ **D** You always need ESR to enhance a webservice. You cannot enhance it in the backend system only.

Learning Assessment - Answers

1. Which communication technique is used mostly in TM for B2B communication?

   Choose the correct answer.

   - [X] A Webservices / SOAP services.
   - [ ] B Native RFC calls.
   - [ ] C IDOCs.
   - [ ] D ABAP Push channels.

2. How can you monitor the webservice based communication?

   Choose the correct answer.

   - [ ] A Via transaction `BD87`.
   - [ ] B Via transaction `BOBT`.
   - [X] C Via transaction `SXMB_MONI` or `SRT_MONI`.
   - [ ] D In the application log.

3. Which statement is correct? (multiple answers possible)

   Choose the correct answers.

   - [X] A You can enhance a webservice in transaction `SPROXY`.
   - [ ] B You cannot enhance a standard webservice. You must always create a copy.
   - [X] C You can enhance a webservice using ESR.
   - [ ] D You always need ESR to enhance a webservice. You cannot enhance it in the backend system only.