Enhancement Guide
SAP Yard Logistics
Document Version: 1.0 – 2016-11-25

CUSTOMER

SAP Yard Logistics 2.0

Enhancement Guide

Typographic Conventions

Type Style	Description
Example	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.
Example	Emphasized words or expressions.
EXAMPLE	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
Example	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE	Keys on the keyboard, for example, F2 or ENTER.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

Document History

Version	Status	Date	Change
1.0	Released to Customer	2016-11-25	Final

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

Table of Contents

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

1 Introduction

1.1 About this Document

This document serves as a reference for all enhancement technologies that can be used in SAP Yard Logistics. The enhancements include Technical Enhancements, Business Logic Enhancements, User Interface Enhancements, and UI5 Fiori Enhancements. Note that major parts of this document overlap with already existing initiatives. The SAP TM Enhancement Guide served as a template for this document as the enhancement strategies have been mostly developed within the TM domain. Some parts have been extracted from there and put in into this document for basic illustration. We recommend that you also go through the original document to get further details.
There are also many documents published in the SAP Community Network (SCN), which is also a good source of information for many enhancement topics.

1.2 Terminology

Abbreviation	Term
BAdI	Business Add In (also: Business Application Development Interface)
BOPF	Business Object Processing Framework
BO	Business Object
FBI	FPM - BOPF Integration
FPM	Floorplan Manager
GUIBB	Generic User Interface Building Block
PPF	Post Processing Framework
TM	Transportation Management
UI	User Interface
UIBB	User Interface Building Block
WDCC	Web Dynpro Component Configuration
YL	Yard Logistics

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

1.3 References

[TMENH]	SAP TM Enhancement Guide http://scn.sap.com/blogs/SAP_TM_Enhancements
[BOPFINTRO]	Introduction to Business Object Processing Framework (BOPF) http://scn.sap.com/docs/DOC-45425
[SCNBOPF]	SAP SCN BOPF Starting Point http://scn.sap.com/docs/DOC-61222
[BOPFNAV3]	Navigating the BOPF, Part 3: Application Programming Interface http://scn.sap.com/community/abap/blog/2013/01/16/navigating-the-bopf-part-3--working-with-the-bopf-api
[BOPFNAV5]	Navigating the BOPF, Part 5: Enhancement Techniques http://scn.sap.com/community/abap/blog/2013/02/22/navigating-the-bopf-part-5--enhancement-techniques
[SCNPPF]	Post Processing Framework (PPF) http://scn.sap.com/docs/DOC-7944
[SCNFPM]	Floorplan Manager http://scn.sap.com/docs/DOC-8068
[SCNFBI]	FPM BOPF Integration http://scn.sap.com/community/abap/bopf/blog/2014/03/19/fbi-views-for-web-dynpro-applications
[SCNFPMENH]	How to Create, Enhance, and Adapt Floorplan Manager Applications on SAP NetWeaver 7.40 - Part 1 http://scn.sap.com/docs/DOC-65883
[HELPGTWENH]	Gateway Enhancements https://help.sap.com/saphelp_gateway20sp08/helpdata/en/fd/d29d56d16244f79c8dbea781d5480e/content.htm
[SCNFIORIEXT]	Extensibility of SAP Fiori Apps http://scn.sap.com/docs/DOC-52539 http://scn.sap.com/community/developer-center/front-end/blog/2015/03/03/extend-a-fiori-application-with-sap-web-ide-part-1
[BAPITRCOMMIT]	BAPI Transaction model (with Commit) http://help.sap.com/saphelp_nw74/helpdata/en/4d/5b2474af960b4ae10000000a42189e/content.htm?frameset=/en/4d/5b102ba1483d8fe10000000a42189e/frameset.htm&current_toc=/en/c2/0da27f769e4c7d99f119110f6a24f4/plain.htm&node_id=37

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

2 Technical Enhancements

SAP Yard Logistics is based on a set of Frameworks that help to realize different aspects of the application. The business objects are modeled and implemented with the Business Object Processing Framework (BOPF). The expert user interface is based on ABAP Web Dynpro and is realized with the Floorplan Manager (FPM), which supports modeling, implementing and configuring the user interfaces. The Floorplan Manager BOPF Integration (FBI) is used to connect the backend with the user interface. It provides the connection between the business objects in the backend with the corresponding user interface realized with the FPM.

To utilize the enhancement capabilities, some general knowledge on these frameworks is required. Besides these frameworks, general knowledge on the following implementation and configuration technologies are a prerequisite for creating enhancements:

- BAdIs (Implementation)
- Implicit Enhancements (Implementation)
- BOPF Enhancement Workbench (Configuration / Implementation, part of the BOPF Framework)

The mentioned frameworks and technologies are described in the following sections to provide a very basic insight on how they are involved in the SAP YL solution and how they are used for creating enhancements.

More Information

You find this documentation on the SAP Library for SAP NetWeaver 7.4. Go to SAP Help portal at section http://help.sap.com and choose *Technology Platform* → *SAP NetWeaver* → *SAP NetWeaver 7.4* → Developer's Guides. Go to *Development Information, Custom Development (UI)* section *or to Custom Application Development (ABAP)* section.

2.1 Business Object Processing Framework (BOPF)

Business objects are the basis of the SAP YL solution. Each business object represents a type of a uniquely identifiable business entity, described by a structural model, an internal process model as well as one or more service interfaces. The business processes provided with SAP YL operate on these business objects. Examples for YL business objects are the Yard Order or the Yard Task.

For a general introduction to BOPF, see [BOPFINTRO] or [SCNBOPF].

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 concepts:

- Actions
- Determinations
- Validations
- Queries

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

The reason for this breakdown is to avoid the mixing of the four types of functionality into one single entity. This improves the potential for reusing implementations and simplifies maintenance by reducing the complexity and dependencies, thereby reducing the development effort.

The architecture of BOPF comprises of the following principal areas:

- *Business Application*, which is the heart of the application and provides an interface between the business data, the business logic, and the end user
- *BOPF Model*, where the runtime configuration parameters for each of the implemented business objects are located

The *Business Application* includes specific entities that support the configuration and runtime operation of each business object, and offers access to the business object's data via buffer classes and data access classes. Furthermore, the *Business Application* includes specific determinations, validations, actions and associations that dictate the specific behavior for each implemented business object.

You can access the business objects only via a defined API (Service Manager). Changing and checking business logic of a BOPF business object is clearly separated. There is no mixture of methods that change the business object with methods that check the consistency of business objects.

Moreover, business logic and data buffering are clearly separated. The business logic is built on top of the business object and the buffer to behave independent of the way in which data is buffered and where data is buffered. BOPF allows replacing buffer and data access classes for business objects. Both do not contain the business Logic.

Data buffer and persistency are also clearly separated from each other as well as from the business logic. This allows establish individual buffer and persistency implementations, that is, both are exchangeable, for example, to achieve specific performance requirements.

A *Business Object* is a representation of a type of uniquely identifiable business entities described by a structural model and an internal process model. Implemented business processes operate on business objects. Most important for the context of this document is that you can enhance a *Business Object* and its characteristics, as well as its configuration settings.

A *BOPF Business Object* model consists of the following entities:

Nodes

A Node is a semantically related set of attributes of a business object. You can use the nodes to define and structure your business object. The attributes of a business object node are defined by dictionary data types. 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.

Business Object Representation nodes are placeholders for other business objects and the associations to these business objects. They are only for visualization of the association to other business objects.

Associations

An Association is a direct, unidirectional, binary relationship between two business object nodes. You can use associations to relate two nodes in a well-defined direction. You can use the association 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. You can only define them 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.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

Actions

An action is an element of a business object node that describes an operation performed on that node. You can use an action to allow the external triggering of business logic (in contrast to a determination). When you perform the action, you must specify the key for the instances on which it is to be performed (if it is not a static action) and any input parameters that the action requires.

You can only perform an action with the number of instances that is configured in the cardinality of the action. It is performed for all instances if an error in the action validation has not occurred. If errors occur, then the behavior depends on the action settings.

Determinations

A Determination is an element of a business object node that describes internal changing business logic on the business object. You can use it to trigger business logic based on internal changes (in contrast to an action). There are two types of determinations - Transient and Persistent. This categorization indicates whether a determination alters persistent or only transient data. You can use a determination to mostly compute data that can be derived from the values of other attributes.

For example:

- Products (for example, item amount = quantity × list price) and ratios
- Totals of items (for example, invoice amount = Σ item amounts)
- Statuses

The determined attribute and the determining attributes can belong to the same node (example 1) or to different nodes (example 2). There are also values that do not depend on any other value but still have to be determined automatically upon creation or modification of a node instance, for example, IDs, UUIDs, and GUIDs.

For each determination, it is necessary to specify which changes (such as create, update, delete or load) on which nodes will trigger the determination at a specific time. A determination is called at different points in time (determination time), depending on the model.

Validations

A validation is an element of a business object node that describes some internal checking business logic on the business object. You can use validations to check if an action is allowed. You can assign action validations to object-specific actions and to the framework actions create, update, delete, and save. You can use them to check if an action can be carried out. An action validation is carried out when an action is called before it is performed. If some validations fail, the action is not performed for the instances where the validation failed. Depending on the action settings, the action is also not performed.

You can use a validation to check the consistency of a business object. You can use consistency validations to check the consistency of a business object. You can assign them to the framework actions check of each node. Consistency validations are carried out when this action is called or automatically after a change is made if they are triggered via trigger nodes based on the changes. It is only triggered if some of the trigger nodes are assigned and instances of these trigger nodes are changed.

Queries

Queries represent a defined set of attributes, such as search parameters, that return the queried IDs of the business object node instances. A query allows you to perform searches on a business object. They provide the initial point of access to business objects. Each query has an associated parameter structure. The result of the query is a set of all the record IDs in a business object that match the query criteria.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

2.2 BOPF Modelling Tool

The models of the SAP Yard Logistics business objects can be displayed with the BOPF Modeling Tool. You can start it via transaction `/BOBF/CONF_UI`. It allows browsing through the list of the business objects of the application. From here, you can navigate to the details of each business object to display its node structure and hierarchy, the configuration, the DDIC structures for each node, the node elements, for example, Associations, Actions, Determinations, Validations and Queries. Moreover, it allows navigating to the implementing ABAP classes of the business object.

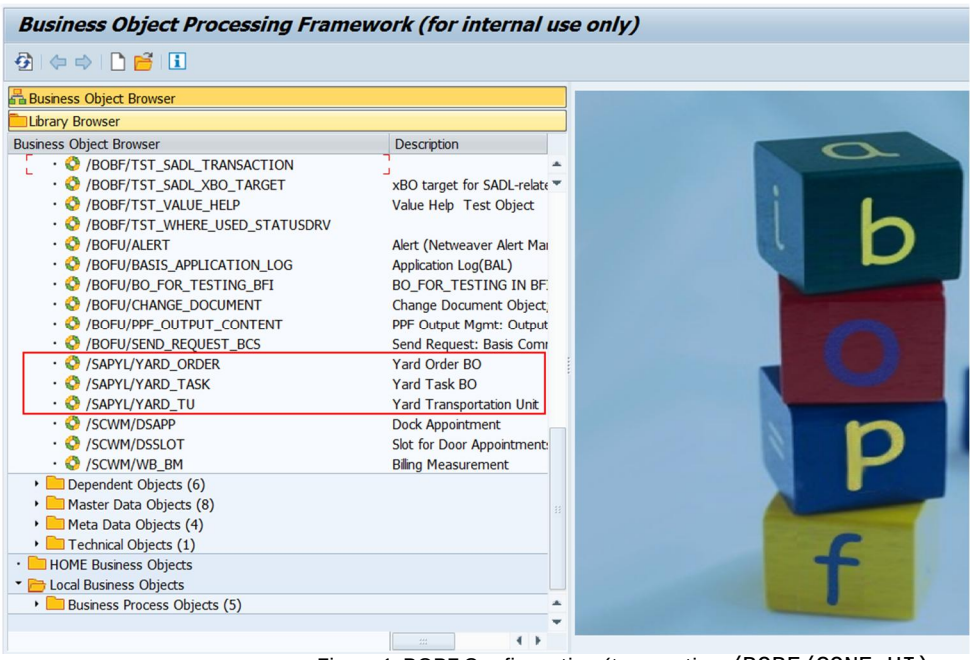

Figure 1: BOPF Configuration (transaction `/BOBF/CONF_UI`)

On the initial screen, you can browse through the available YL business objects as well as four other object categories, which are partially reused. These are as follows:

Dependent Objects:

Used in SAP YL for reusable parts of business objects that are not objects on their own, that is, they only exist in the context of a business object (the hosting object). For example, attachment folder and text collection.

Master Data Objects:

Most master data BOs call the SCM Basis Master Data Layer (MDL) via an adapter in a read-only way. You can maintain the content of these master data objects via the standard transactions in SCM Basis. The master data distribution between SAP ERP and SAP YL follows the standard SCM middleware architecture of the SCM Core Interface (CIF). For example, Location, Business Partner, Material, and so on.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

In the *Business Object Detail Browser*, you can navigate through the node hierarchy of the business object and display the node details. Besides other information, the node details show the data model of the node.

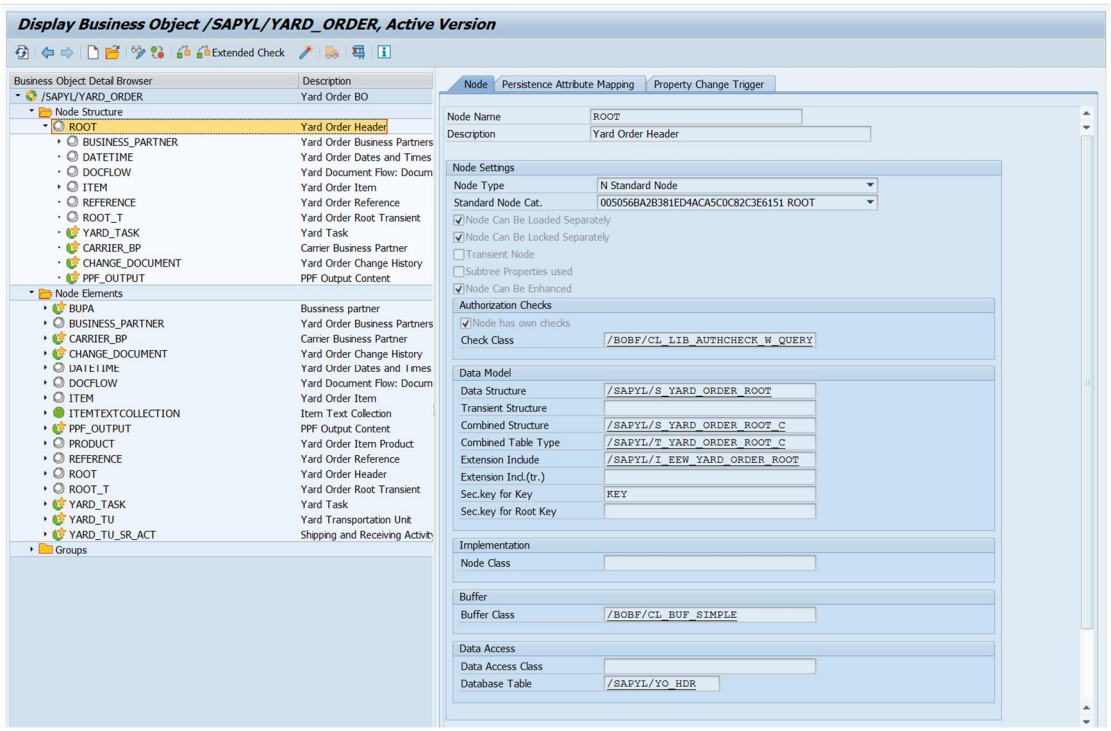

Figure 2: The Business Object Browser

Combined Structure and Table Type:

This dictionary structure includes the data structure of a node. In addition, it includes a fixed BOPF DDIC structure, which contains the node instance key (KEY), the key of the direct parent node instance (PARENT_KEY) as well as the key of the related business object instance (ROOT_KEY). The Combined Table Type has the Combined Structure as its line type.

Data Structure:

This DDIC structure contains the attributes of the node, representing the node data.

Data Structure (tr.):

The data structure contains the transient attributes of a node, that is, attributes that is not persisted but are only filled and used during runtime.

Extension Include and Extension Include (tr.):

Extension Include is important for field extensions on a node. With this include, all extension fields are added (via Append Structures), which are to be persisted. Extension fields, which are only relevant at runtime and not relevant to be persisted, are placed in the corresponding transient Extension Include.

Database Table:

Shows the database table where the persistent node information gets stored.

When expanding the Node Elements, you can further navigate to a node and the elements assigned to it (for example, Associations, Determinations, Validations, Actions and Queries as described in the previous sections).

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

Moreover, the details for each of these elements can be displayed from here. For example, the details of an Action include a link to the implementing class of this Action, and if the Action has parameters, the corresponding parameter structure.

The details of the node elements like Actions, Validations, Determinations, and so on, are the starting point to identify places in the coding where a specific functionality of interest is implemented. Within the implementing classes of the node elements, further classes and their methods are used to realize its functionality.

2.3 BOPF Implementation Basics

You can find a comprehensive representation of BOPF implementation aspects at the following places:

- TM Enhancement Guide [TMENH], Section 3.2
- Navigating the BOPF, Part 3: [BOPFNAV3]

2.4 BOPF Enhancement Workbench

Typically, you implement a business object based on a specific business semantics that is defined via the core requirements as defined by the product owner team. When the core functionality does not cover the requirements of a customer, you have to enhance the business object has. Fortunately, the BOPF framework comes with a set of enhancement techniques that you can use to add the functionality. The tool that allows the enhancement is called the BOPF Enhancement Workbench.

A general introduction to the enhancement features of BOPF can be found in [BOPFNAV5]. The SAP TM Enhancement Guide [TMENH] shows the enhancement features in the context of SAP TM, which has a similar architecture as SAP YL.

The BOPF Enhancement Workbench is available to enhance the standard YL BOBF Business Objects. You can use it to create, change or delete enhancements of the standard YL BOPF Business Objects. You can again enhance such enhancements with the same tool, that is, nested enhancements are also possible. The BOPF Enhancement Workbench supports the following enhancements:

Create, change or delete additional:

- Sub nodes
- Actions and action enhancements
- Determinations
- Consistency and action validations
- Queries
- Customer/Partner-specific Business Objects

The BOPF Enhancement Workbench now allows creating new customer/partner-specific business objects. Customers and partners can create their very own business objects that they can freely implement according to their requirements. SAP development must have declared Standard YL BOPF business objects as extensible. Only such business objects can be enhanced. The same applies to a business object entities like nodes, actions, and so on. They can only be enhanced if SAP development has declared them to be extensible.

The BOPF Enhancement Workbench is started with transaction `/BOBF/CUST_UI`.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

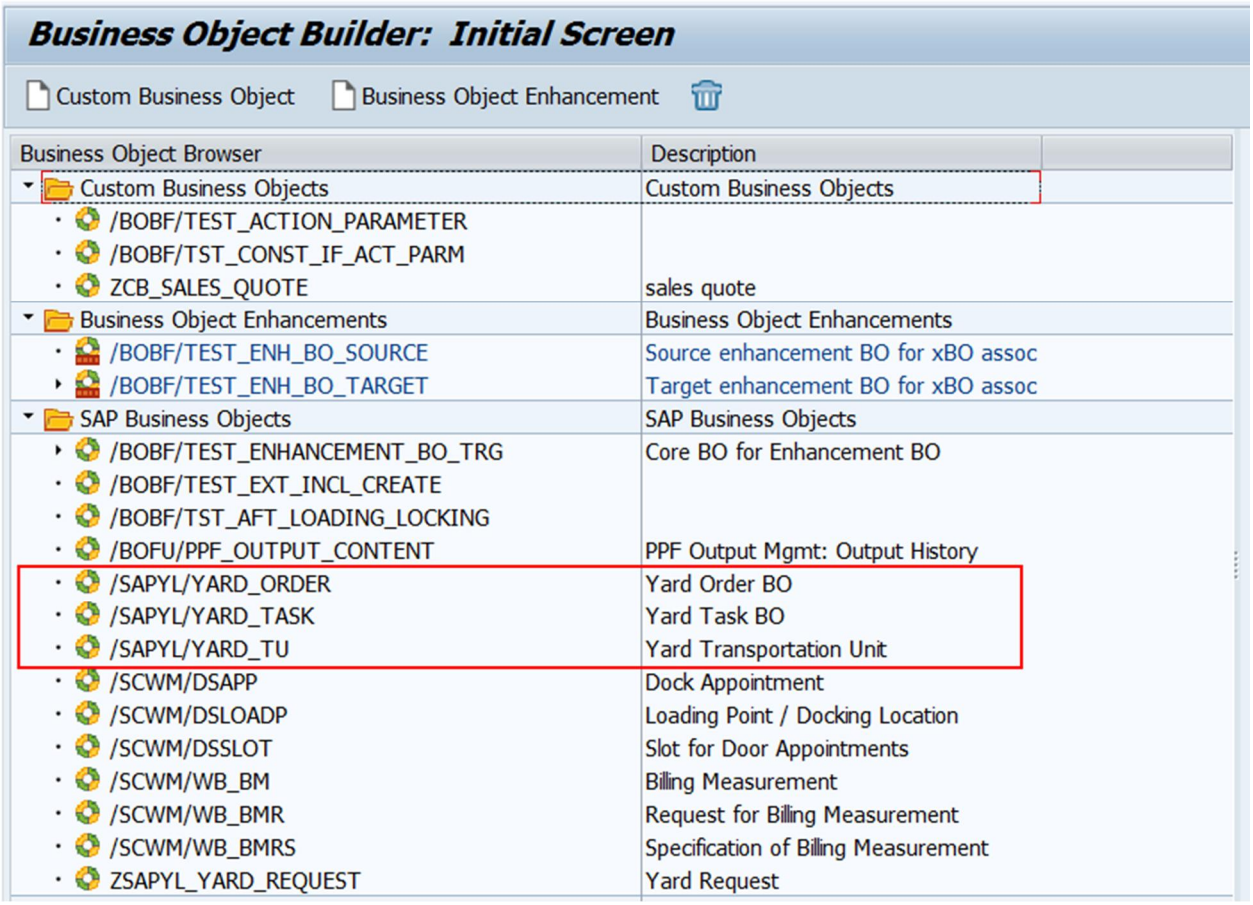

Figure 3: Enhancement Workbench

On the initial screen, you can see the business objects that are allowed to be enhanced in general (see Business Objects in the figure above). Whenever you want to enhance one of the listed Standard Business Objects, the first step is to create a so-called enhancement object for this business object. For the enhancement object, the original business object represents the so-called super business object or base object.

Important to know is that this enhancement object neither replaces nor represents a copy of the standard business object. Instead, it serves as a container for all enhancements that you add to the business object via the enhancement workbench. At runtime, the standard business object functionality is still being executed with the enhancements in addition.

Some general remarks that are valid for creating enhancements via the wizards provided by the BOPF Enhancement Workbench:

Enhancement Name:

The enhancement name should start with the namespace or prefix of the open enhancement. This ensures there is a clear separation between the entities of different enhancements, for example, two or more different implementation partners want to separate their enhancements. The system automatically enters the value in the field for the enhancement name. You should add a meaningful enhancement name.

When completing the wizard, further required objects are generated automatically. The system adds the new enhancement to the enhancement object. It displays the new enhancement in the Entity Browser when you select the corresponding assigned node or action. As mentioned, in case an implementing class is required for the

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

enhancement, it is generated and must be implemented manually afterwards. The constants interface of the enhancement object is regenerated and contains a unique constant identifying the new enhancement. In the enhancement constants interface, you will only find constants for your enhancements. The constants for standard entities are located in the standard BO constants interface.

You can also delete every enhancement created with the BOPF Enhancement Workbench. For each create wizard, a corresponding delete wizard is available that guides you through the relevant steps and checks the preconditions to be fulfilled for a deletion.

Implementing Classes:

Some types of enhancements require providing an implementing class. This class contains the business logic of the enhancement. In the wizard, you define a class name and the system creates it automatically with the right interface assigned for the corresponding entity that the system creates after finishing the wizard. You must implement the enhancement manually.

The implementing class name should meet naming conventions. The wizard automatically suggests a class name that corresponds to the BOPF naming conventions. You can also provide already existing classes as implementing classes. They need to implement the corresponding BOPF Interface, for example, `/BOBF/IF_FRW_ACTION` for actions or `/BOBF/IF_FRW_VALIDATION` for validations, and so on). The system does not overwrite the implementing class if it already exists when finalizing the wizard.

Enhancement Constants Interface:

Note that each enhancement object has its own constants interface. Enhancement node elements are not added to the standard constants interface but to the corresponding enhancement object constants interface. You must use the latter to reference your enhancement node elements in the coding.

Enhancement Scenarios:

Enhancements to a business object can concern many different topics:

- Field Extensions

  Enhancements can be as simple as adding additional persistent or transient attributes to existing nodes. This is supported via append structures to the persistent and transient include structures of the BO.

- Create Additional Sub Nodes

  Enhancement can also comprise of adding additional nodes in the BO definitions. Such nodes can be persistent or transient. Thereby, you can add completely new features to existing BOs.

- Create Additional BOPF Artifacts

  You can enhance an existing BO by new basic BO artifacts like actions, determination, validations, associations, and queries. Typically, field extensions or new sub nodes are filled / created by additional determinations. You typically enhance a BO status and consistency by additional validations. Here, the execution sequence of added artifacts with respect to already existing standard artifacts plays a role. The definition of the execution sequence is part of enhancement process.

- Enhancing Existing BOPF Artifacts

  To change the basic behavior of a BO, you can enhance the existing standard artifacts via pre- and post-enhancements. You have to define such enhancements carefully.

You can find many concrete examples regarding the above enhancement scenarios in section 3.3 of the TM Enhancement Guide [TMENH].

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

Summary

BO enhancements work best if you can clearly define the semantics of the enhancements by new additional features of the BO. Such features can be field extensions but also new additional sub nodes. You can use the added BOPF artifacts to initialize/fill the content of the additional features. More complex are enhancements where the existing BO logic is actually altered by changing existing BOPF artifacts via pre- and post-enhancements. All enhancements are placed in a business object enhancement. The best practice is to arrange the required ABAP source code in dedicated packages and namespaces.

2.5 Implicit Enhancements (ABAP)

For ABAP programs, a number of so-called 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
- Replacing method implementations by overwrite-methods

These 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. In other cases, SAP strongly recommends to use BAdIs instead, since they provide a defined interface. Nevertheless, some of the mentioned options are described here to be used as a means to create enhancements. You should use the implicit enhancements with care. Keep in mind the following aspects when making use of this enhancement technique:

- Detailed knowledge on the application code is required for identifying the objects to that you want to enhance 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 a 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.

ABAP objects, classes or interfaces, can be enhanced by pre-, post- and overwrite methods that are executed before or after the original method implementation or in case of overwrite methods, replace the complete original implementation at runtime.

To stress again, you should make sure to handle the depicted implicit enhancements (pre-, post- and overwrite-methods) with care. Keep in mind the potential problems and consequences mentioned at the beginning of this section.

Moreover, it is highly recommended to implement such enhancement code in your own local class methods and just place the call of these local class methods into the Enhancement Implementation. This provides more transparency for customers and partners, as well as, for SAP in case of problem analysis, support, and so on. This also provides a better overview for customers and partners over the coding that they have added with the described techniques.

You are provided with several concrete examples regarding these enhancement topics in the TM enhancement guide [TMENH], They can also be applied to SAP Yard Logistics.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

3 Enhancements of Business Logic

3.1 Business Add-Ins

The Business Add-In (BAdI) concept is SAP's object-oriented plug-in concept for ABAP. 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. BAdIs are used to plug in custom behavior either in an additive way or by replacing the standard behavior.

All BAdIs can be found in the IMG (transaction SPR0) under the following path:

SAP Customizing Implementation Guide → *Yard Logistics* → *Business Add-Ins (BAdIs) for Yard Logistics*.

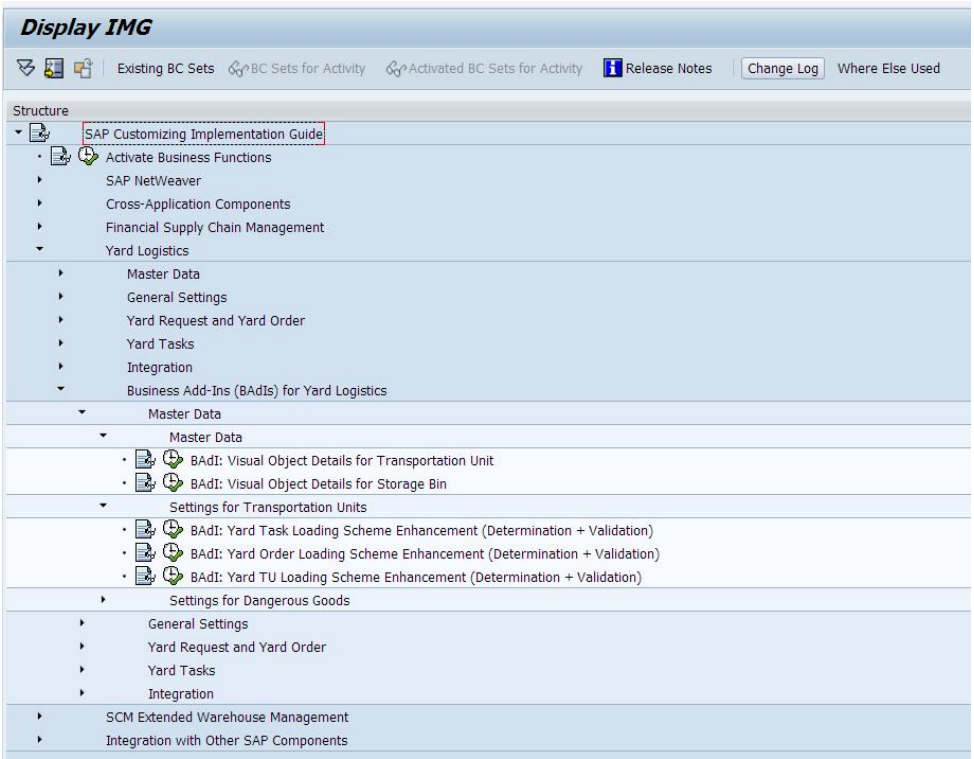

Figure 4: Available BAdis for SAP Yard Logistics (excerpt)

You can start a BAdI implementation directly from the IMG. As an alternative, you can use transaction SE19 to either edit existing enhancement implementations or create new ones. Besides the initial screen, the other steps to implement a BAdI with SE19 are the same as the procedure to start from the IMG.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

3.2 Post-Processing Framework

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 system generates the actions if specific conditions occur for an application document. The system then processes them either directly or later.

The PPF is part of the SAP Web Application Server and the applications can use them. It is the successor to Message Control and offers a wider functional scope, more simple connection to the applications, and greater flexibility.

The PPF provides tools for the scheduling, starting, and monitoring of actions. Determination, generation, and processing of actions can either take place automatically or with user interaction.

The PPF provides an evaluation of modifiable conditions for action determination. The application can set the determination technology, with which the conditions are evaluated, according to its requirements, or use its own determination technology.

The PPF provides processing types, which take over the execution of various actions:

- Printing, sending e-mails, and faxing with Smart Forms
- Starting a workflow
- Starting a Business Add-in for follow up processing

The application can adopt the processing types, which is adapted to meet the requirements of the application, or replaced with the application's own processing types.

A starting point for further information is [SCNPPF].

Within SAP YL, we use PPF definitions to call outbound web services.

Action Profile	Description
/SAPYL/EXT_COMM_YO	Profile for External communication at YO
/SAPYL/EXT_COMM	Profile for External communication
/SAPYL/TU	Yard Transportation Unit
/SAPYL/TU_WHSE_VISIT	Yard Transportation Unit. Warehouse Visit
/SAPYL/YO	Yard Order
/SAPYL/YO_PRINT	Yard Order Printing
/SAPYL/YR_PRINT	Yard Request Printing
/SAPYL/YT	Yard Task
/SAPYL/YT_PRINT	Yard Task Printing

Figure 5: PPF Action profiles for SAP Yard Logistics

The system calls them from PPF actions of YL business objects, `/SAPYL/YARD_TU` (Yard Transportation Unit) and `/SAPYL/YARD_TASK` (Yard Task). To process the PPF actions, you are provided with the following agent classes:

`/SAPYL/CL_YO_PPF_EXT_COM_AGENT` External communication agent

`/SAPYL/CL_YO_PPF_PRINT_AGENT` Yard Order Printing Agent Class

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

`/SAPYL/CL_YO_PPF_SERV_AGENT` Yard Order PPF Service Agent

`/SAPYL/CL_YR_PPF_PRINT_AGENT` Yard Request Printing Agent Class

`/SAPYL/CL_YTU_PPF_SERV_AGENT` Yard Task PPF Agent Class

`/SAPYL/CL_YT_PPF_EXT_COM_AGENT` External communication agent

`/SAPYL/CL_YT_PPF_PRINT_AGENT` Yard Task Printing Agent Class

`/SAPYL/CL_YT_PPF_SERV_AGENT` Yard Task PPF Agent Class

`/SAPYL/CL_WH_VIS_PPF_SRV_AGENT` YTU Warehouse Visit PPF Agent Class

3.2.1 Output Management

You can use output management to print, fax, and email documents such as yard order, yard request or yard task information. You can also execute application-to-application or business-to-business actions. You can use output management features in the following business documents:

- Yard Requests
- Yard Orders
- Yard Tasks

The following are the main output management features:

- Display an action such as a settlement document
- Choose the output method such as print, fax, and email
- Preview a document
- Enter an attachment
- Enter a note
- Enter a recipient
- Execute the action, such as printing and faxing a yard document

You can also regenerate and execute an action that you have already executed. For example, you can reprint a business document that you have already printed. When you regenerate an action, you can change the attributes, for example, the recipients.

The system also displays the status of an action after execution. If a particular action was not processed successfully, the system displays the cause of the error.

For printing yard specific documents, we use the Adobe Document Services with the following sample PDF forms:

- `/SAPYL/F_YO_PDF` - SAP YL Yard Order Printing Document to PDF
- `/SAPYL/F_YR_PDF` SAP YL Yard Request Printing Document to PDF
- `/SAPYL/F_YT_PDF` SAP YL Yard Task Printing Document to PDF

You can adapt a standard print form to your requirements, or create a new print form.

If you are creating a new print form, you have created and activated a form using the Form Builder (transaction SFP). Then execute the following procedure:

- Create a custom ABAP class for the print form (see Creation of an ABAP Class for a Print Form).
- Create a custom ABAP class for the Post Processing Framework (PPF) service (see Creation of an ABAP Class for the PPF Service).
- Define an action profile and an action in the Post Processing Framework (PPF).

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

- An action controls the point at which a document is to be printed (for example, when a user saves a document). When you define an action, you have to specify the name of the print document. You also specify that the document is a PDF document, and you enter a standard BAdI. Alternatively, you can use an existing action profile and action.
- Create the required schedule conditions in Customizing for Cross-Application Components under *Processes and Tools for Enterprise Applications* → *Reusable Objects and Functions for BOPF Environment* → *PPF Adapter for Output Management* → *Maintain PPF Settings*.
- Assign the ABAP class for the PPF service created in step 2 to the relevant business object (BO) nodes.
- The new PPF service class must be assigned to the relevant business object node as an output agent. You do this in Customizing for *Cross-Application Components* under *Processes and Tools for Enterprise Applications* → *Reusable Objects and Functions for BOPF Environment* → *PPF Adapter for Output Management* → Maintain Output Management Adapter Settings.

More Information

SAP Library for SAP Supply Chain Management on SAP Help Portal at http://help.sap.com/scm. Under SAP SCM Server application help, open SAP Library and choose Processes and Tools for Enterprise Applications Reusable Objects and Functions for the BOPF Environment (CA-EPT-BRC)

http://help.sap.com, then "Function oriented view" > "UI Technologies in SAP Netweaver", then "SAP Interactive Forms by Adobe", see link:

http://help.sap.com/saphelp_nw74/helpdata/en/a9/b128543eaa4a508b5120b695e29391/content.htm?frameset=/en/47/ab79e202e84be69174b5683fd02f9c/frameset.htm&current_toc=/en/47/ab79e202e84be69174b5683fd02f9c/plain.htm&node_id=32&show_children=false

3.2.2 External Driver Communication

As soon as a Yard Task is activated, an SMS is sent to the phone number, informing the driver about this task. The SMS contains the destination information in the yard. The same should be possible when a task is activated automatically as a follow-on task or manually using the yard task maintenance. Also in this situation, the SMS is sent to the driver giving him the information about the next location in the yard.

The user is able to:

- Change the message text
- Confirm the sending of the message - sends the message
- Cancel the sending of the message - no message is sent but still the task is activated

The determined text is shown to the user in a popup. The language for the text is the driver language maintained in the yard order. If no language is maintained, the user language is used. A new setting in the user default settings "Show Driver Communication" (checkbox) for YO and YT is provided. Only if this flag is maintained, the popup with the text that is sent is shown.

In order to show the popup, a new checkbox is implemented in the *Overview Yard Task* or *Overview Yard Order* (see next figure).

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

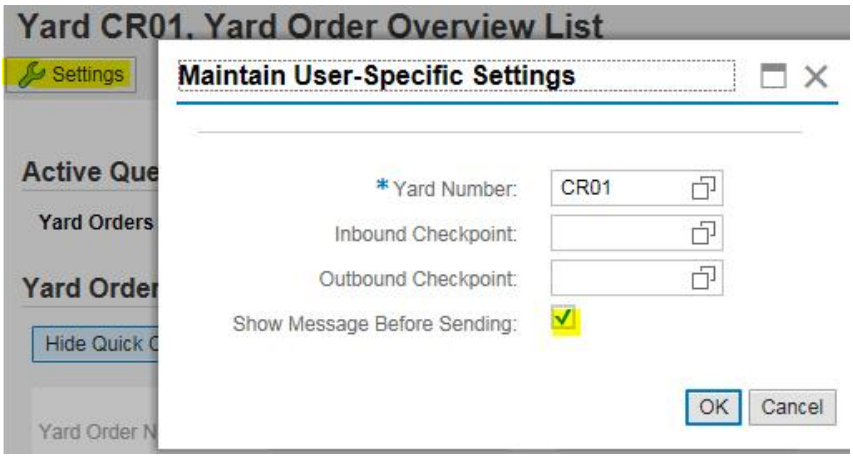

Figure 6: Popup for Driver Communication

The text used for the SMS is composed out of predefined texts and variables (placeholders) from the Yard Task.

The text name must be maintained in the Application Menu:

Application Menu Path: Settings -> Driver Communication -> Define Text for Driver Communication

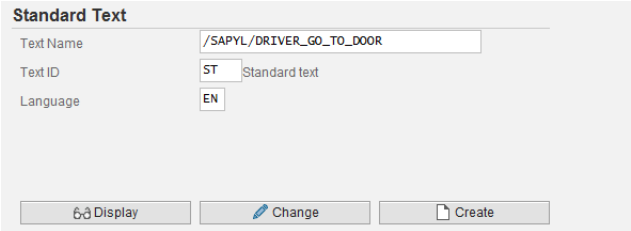

Figure 7: Text Definition

The figure above shows an example for the text maintenance using transaction `SO10`. Within the text, placeholders for runtime information from the yard documents is allowed (see next figure).

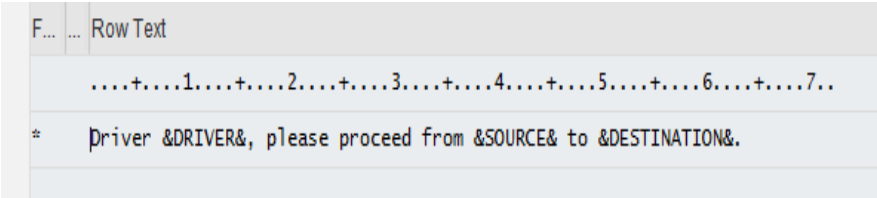

Figure 8: Text with Placeholders (example)

These placeholders have to be between the ampersands like &NAME&.

The placeholder variables can be defined in the Application under the menu path: *Settings -> Driver Communication -> Assign Text Messages to Activity Type*

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

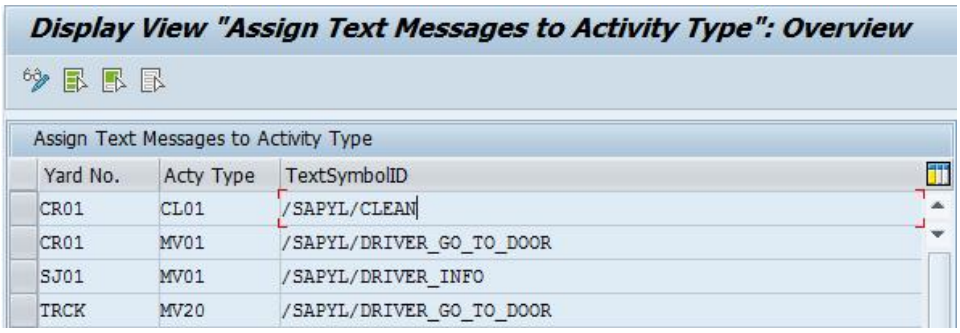

Figure 9: Assign Text Messages to Activity Types

Now that we aligned our Yard Number and Activity Type to our Text Name we shall also add logic to the Placeholders && from above.

This can be set in the application menu following the path: *Settings -> Driver Communication -> Maintain Text Symbol for Driver Communication*

The following figure shows some samples.

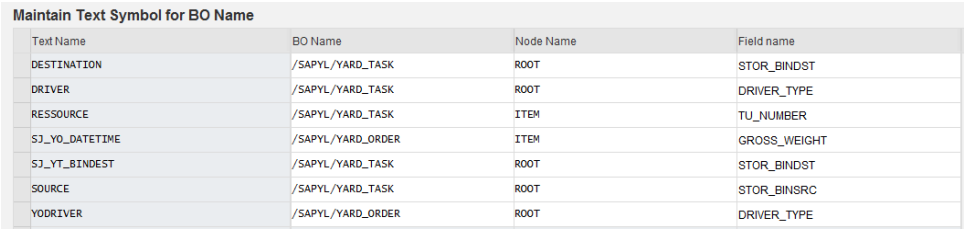

Figure 10: Samples for Text Symbols

Once all conditions are met and the ‘Start Processing’ or ‘Activate’ starts, the following popup appears where you have the possibility to change the message (see next figure).

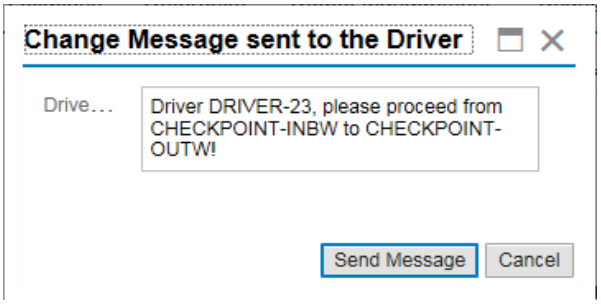

Figure 11: Sample Message for Driver

In order to create the popup, a new dialog box was created in the Component Configuration `/SAPYL/WDCC_YO` with the Page ID ‘`ACTIVATE_POPUP`’ and UIBB Form ‘`/SAPYL/WDCC_DRIVER_POPUP`’.

Feeder Class: /SAPYL/CL_DRIVER_POPUP_FEEDER

This feeder class contains all logic to create the message and save it on the Root of the Yard Task, so that in the later steps it is ready to be sent to the driver.

Once the message is saved on the Yard Task, you should still be able to send the message to the Driver.

For this, a PPF action is implemented:

- Action Profile: /SAPYL/EXT_COMM

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

- Action Definition: /SAPYL/DRIVER_COMM

This action will be processed whenever a yard document is saved and it is automatically actioned only once (the next figure shows the PPF definition).

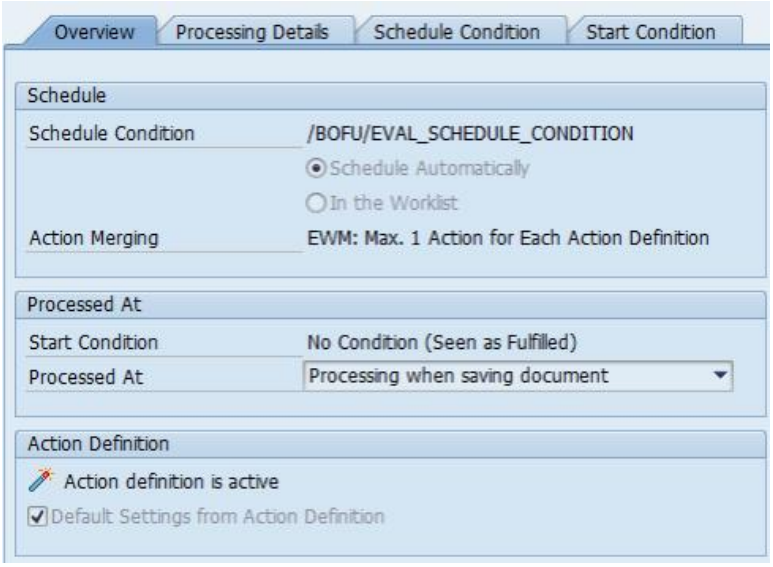

Figure 12: PPF definition for Driver's Communication

In addition, the Output Management Adapter Settings are maintained as seen in the next figure:

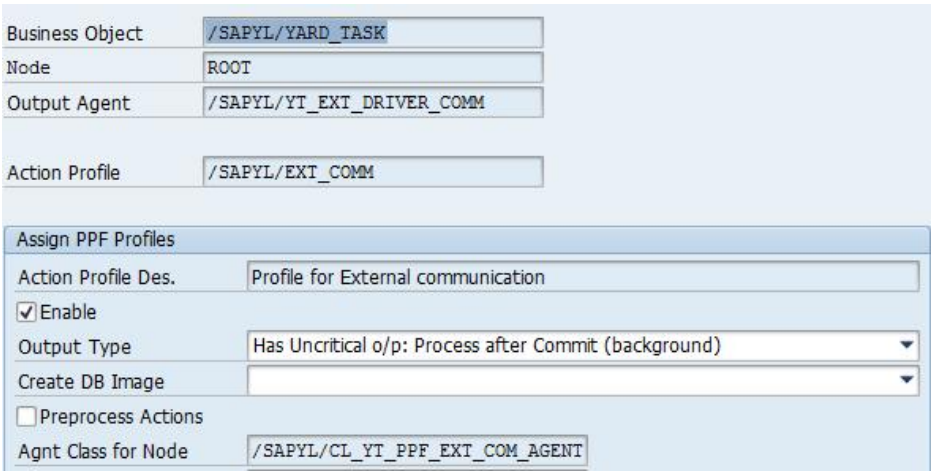

Figure 13: Output Adapter Settings

For this PPF, the Agent Class /SAPYL/CL_YT_PPF_EXT_COM_AGENT is used.

The two most important methods are:

- DO_YT_SCHEDULE_ACTION_BY_ABAP – In this method verifies if the status of the Yard Task is "Active", if there is an existing customizing entry for the message and if the phone number is maintained. If all of the above conditions are satisfied, then the PPF is triggered.
- EXT_DRIVER_COMMUNICATION –. This method saves all the information regarding the Message in an Alert Container and it sends it to the maintained Phone number via an SMS. (Alert Configuration is needed as a prerequisite.)

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

How to configuring alerts and its related containers

Go to Transaction *ALRTCATDEF* – Create an own Alert Category Package: (for example: ZSAPYL Alert Categories)

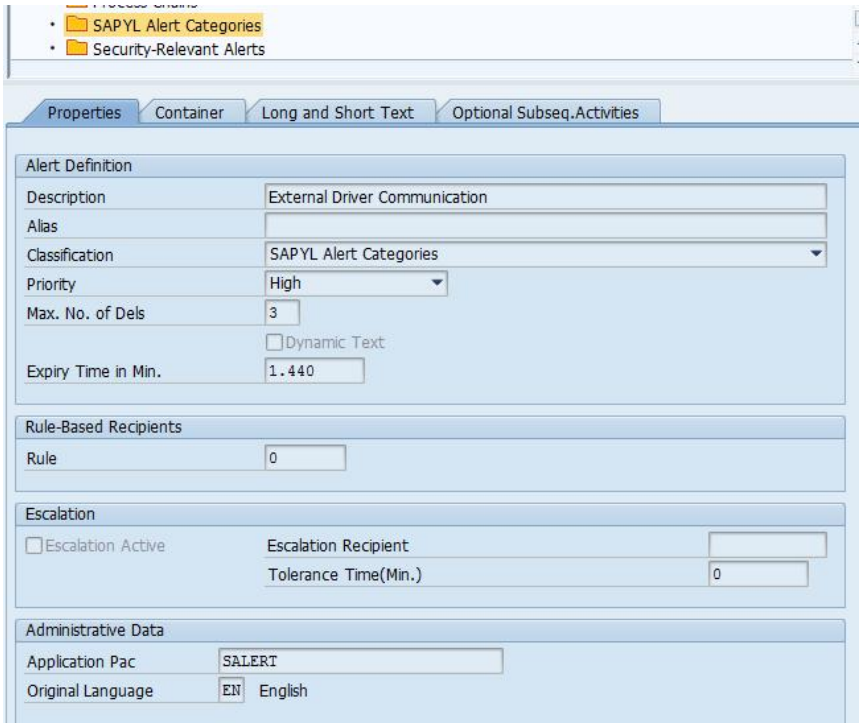

Figure 14: Alert Category

On the right side of the screen, you have the possibility to add an alert category:

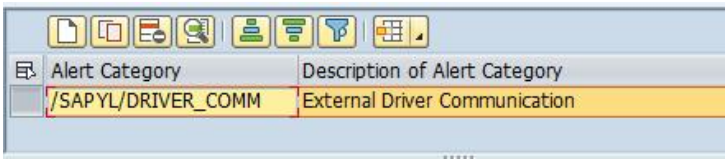

Figure 15: Alert Category (followed)

In the Container tab, create a container element (that is, it should be identical to a data element).

For the Driver Communication in SAP Yard Logistics, the container element *SAPYL_DYNAMIC_MESSAGE_MVT* is used:

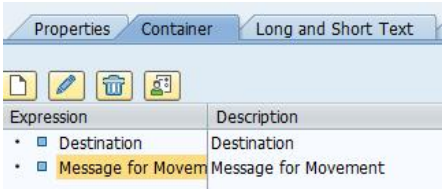

Figure 16: Container Values

In the Long and Short Text tab, you have to define the specific container. This container element that contains the value of the message is customized and created in the PPF mentioned above.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

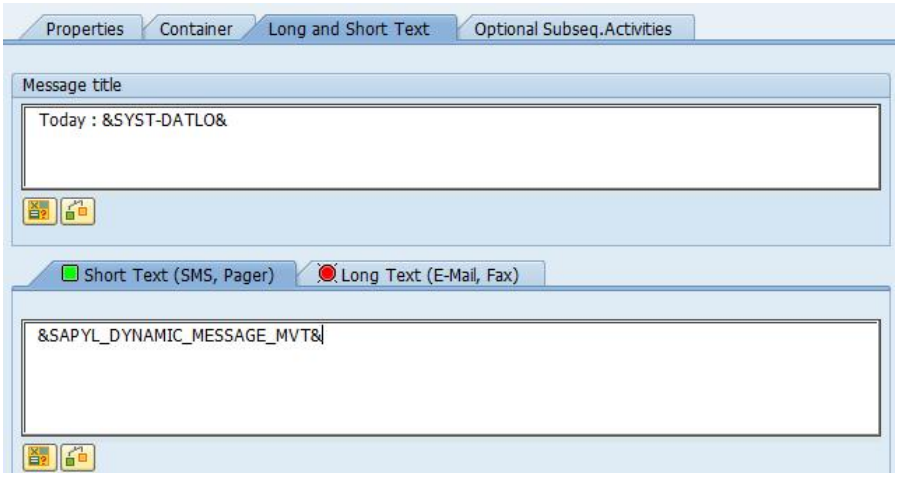

Figure 17: Container Description

As a title, the system date, and as a short text, we use the container element defined before in the sample setup for SAP Yard Logistics.

To change the alert category or the container element, you have to implement the BAdI */SAPYL/DRV_MSG* and the method *OVERWRITE_ALERT*.

In the sample implementation, the fields are:

- Alert Category */SAPYL/DRIVER_COMM*
- Container Element *SAPYL_DYNAMIC_MESSAGE_MVT*

3.2.3 Warehouse Visits

In order to support integration of SAP Yard Logistics with several external EWM systems, a new so-called "Warehouse Visit" is introduced. A warehouse visit represents a combination of EWM business system and EWM warehouse. They should be maintained for each Yard Order Item that is processed by an external EWM system. At that level, as many warehouse visits are defined as EWM warehouses need to be visited by a Yard Transportation Unit.

The integration with these external EWM systems is done by means of web services. All web service calls are wrapped into the BADI */SAPYL/SUBSYST_CALL* in enhancement spot */SAPYL/ES_WEB_SERVICES*.

The interface `/SAPYL/IF_SUBSYST_CALL` contains methods for all used outbound web services:

- CHECK_IN (TU Check-in)
- CHECK_OUT (TU Check-out)
- DOOR_DETERMINATION (TU Door Determination)
- DOOR_ARR_DEP (TU Arrival / Departure at door)
- REVERSE CHECK-IN
- REVERSE CHECK_OUT

This BAdI is triggered within corresponding qRFC modules of function group /SAPYL/EWM_RFC (RFC for EWM communication).

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

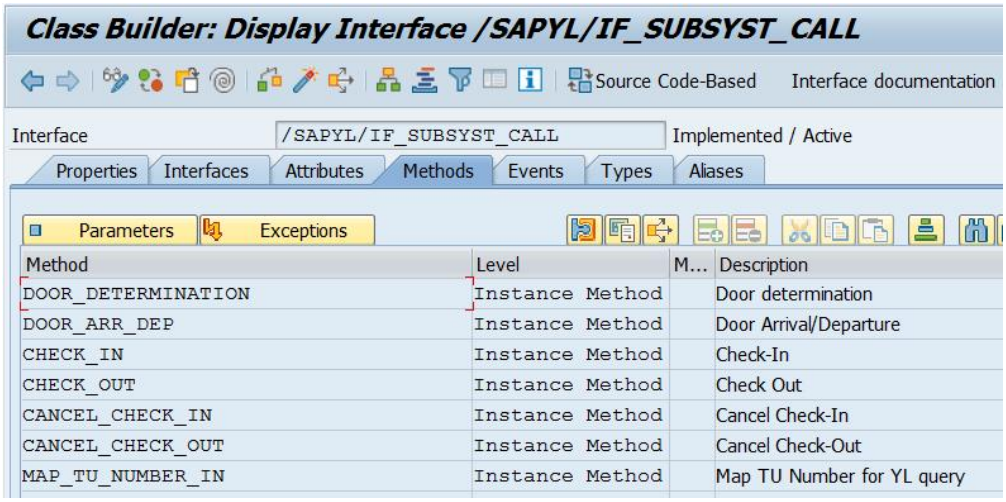

Figure 18: Interface of the BADI for EWM communication

The next figure shows the BAdI definition.

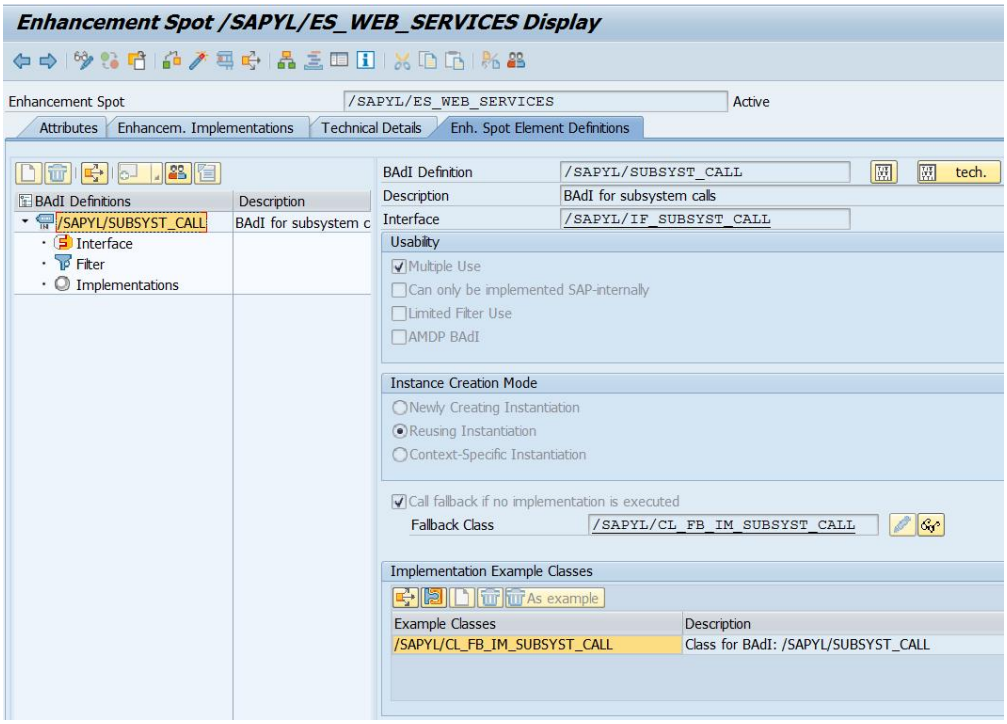

Figure 19: BADI for EWM Communication

The interface of the BAdI provides a possibility to retrieve additional data for Transportation Units, corresponding Yard Orders, Yard Tasks etc. By using a default implementation, the EWM data is retrieved either by TU external number or by freight order number (in case of integration with SAP TM). In addition, it provides data for required warehouse visit (SAP EWM system and EWM warehouse).

In case of integration with multiple warehouses within one business system, it is assumed that the Transportation Unit's external number is unique per EWM warehouse and EWM system. Depending on the number of warehouses per Yard Order Item that should be visited within one EWM system, Transportation Units should be created in EWM following this assumption. In order to support this functionality, method MAP_TU_NUMBER_OUT of the BAdI

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

is foreseen for communication between SAP EWM and SAP YL. By default, the mapping TU number plus EWM warehouse is implemented.

Example

Assume that in SAP YL you have Transportation Unit number TRUCK_001.

Assume also, that a warehouse visit is created for the same EWM system SYS_001 for warehouses WSE_01 and WHSE_02.

Following the rule above in the BAdI, the TU numbers are mapped as follows: TRUCK_001_WHSE_01 and TRUCK_001_WHSE_02.

Communication from SAP EWM to SAP YL

The communication from SAP EWM to SAP YL is done for loading/unloading notification by using the PPF action, which is triggered by the corresponding action of EWM Transportation Unit. It has been setup as:

- Application /SCWM/SHP_RCV
- Action Profile /SCWM/TU
- Action Definition /SAPYL/YL/LOAD_UNLOAD_NOTIF

The existing default implementation can be overwritten by using a new PPF action.

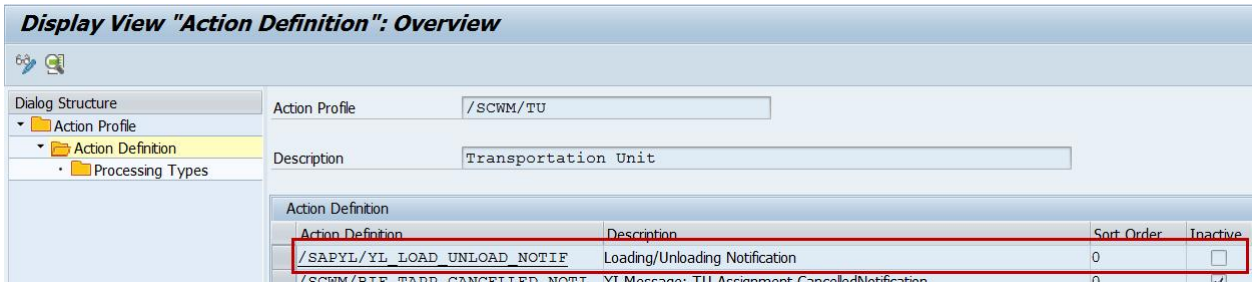

Figure 20: PPF Action Definition for SAP EWM to SAP YL communication

More information

For more information on the integration between SAP EWM and SAP Yard Logistics release 2, consult SAP Note 2392337 *Implementation information about the integration of SAP YL and SAP EWM.*

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

4 User Interface Enhancements

This chapter provides the basics on how to enhance the existing user interface. The user interface is built with the help of the Floorplan Manager (FPM) and the Floorplan Manager BOBF Integration (FBI). These two frameworks enable enhancing a user interface via configuration rather than having to implement additional code.

This document cannot cover a complete description of FPM and FBI. It, therefore, concentrates on the very basic things that customers and partners need for creating basic and common enhancements of the UI by adjusting the standard configurations of the YL user interface. The examples used here are based on the Yard Order UI but the principles and techniques are valid for any other user interface also. For enhancements of more complex user interfaces, it is recommended to build up a more detailed FPM and FBI knowledge.

4.1 Floorplan Manager (FPM)

The FPM is based on 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) that are compliant with the mentioned guidelines.

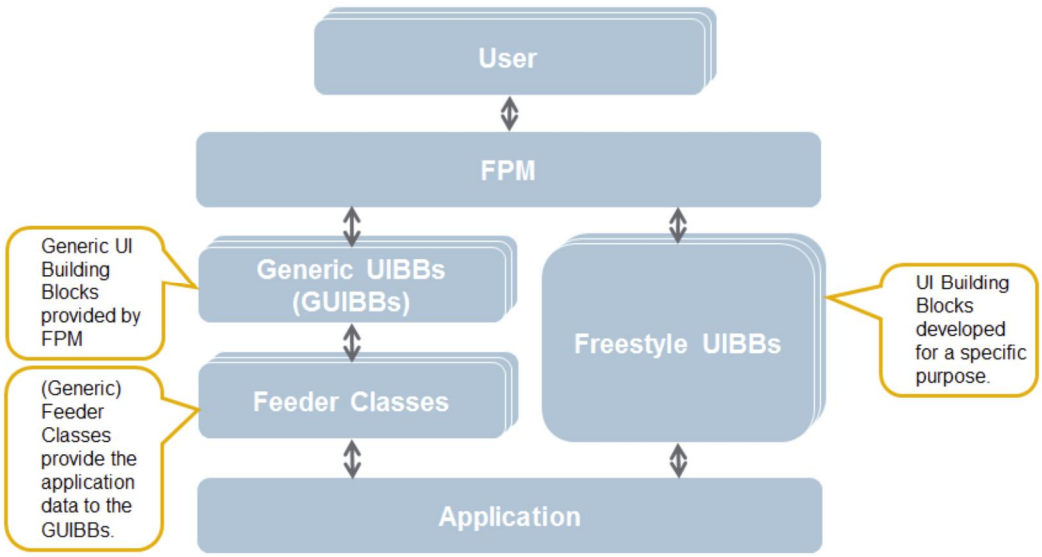

Figure 21: FPM Architecture

For more information about which GUIBBS exist, how feeder classes are generated, about the detailed FPM event loop, and how the UIBBs are talking with each other via wiring, consult the existing sources, for example, [SCNFPM].

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

4.2 FPM - BOPF - Integration (FBI)

The Floorplan Manager BOPF Integration (FBI) is used in SAP Yard Logistics to integrate FPM with the BOPF-based business objects. The FBI provides generic FPM application feeder classes together with the relevant application configuration that allows consuming services of Business Objects modeled in BOPF. You can use these BOPF services seamlessly in a modification-free UI environment.

The FBI provides the following functionalities that support the communication and cooperation between FPM applications and BOPF-bases 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)

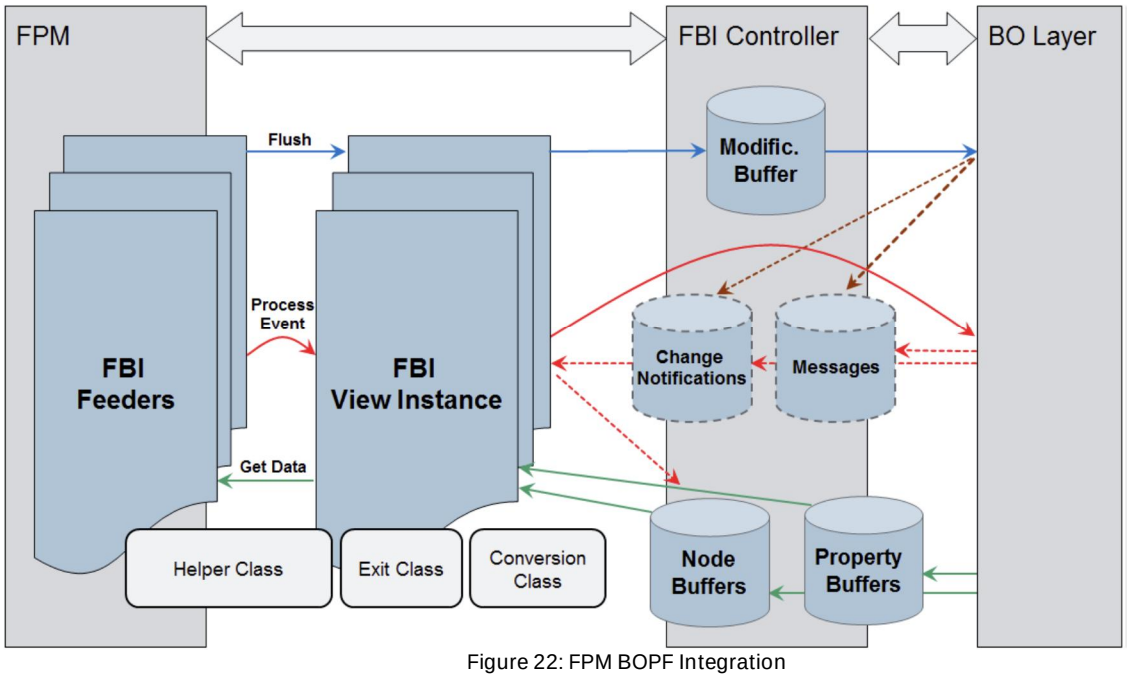

Figure 22: FPM BOPF Integration

You can obtain a general overview of the required concepts of FBI in [SCNFBI].

4.3 Adapting FPM Applications

Some general remarks for creating user interface enhancements:

- You can execute the basic enhancements ideally without any coding. For more complex user interfaces and enhancements, coding might be required, for example, implementation of exit class methods.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

- SAP YL user interfaces are built-up from the so-called User Interface Building Blocks (UIBBs). Each of these building blocks have a configuration that can be adapted by partners and customers.
- The standard configurations remain untouched. You can create the Configuration Enhancements, for example, in a development system and transport them to a test or a production system. The client of the system where you do the UI enhancements must be set up in a way that it allows development and transporting configurations.
- You can also delete the enhancements again. After deleting, for example, an Enhancement of a configuration, the original standard configuration is in place again for processing the corresponding user interface.
- Reconfiguration of UIs based on FPM is simple. You just need to start the required user interface and use the technical help link (right-click into the UI pane) from where you can then navigate further to the different configurations that make up the UI.

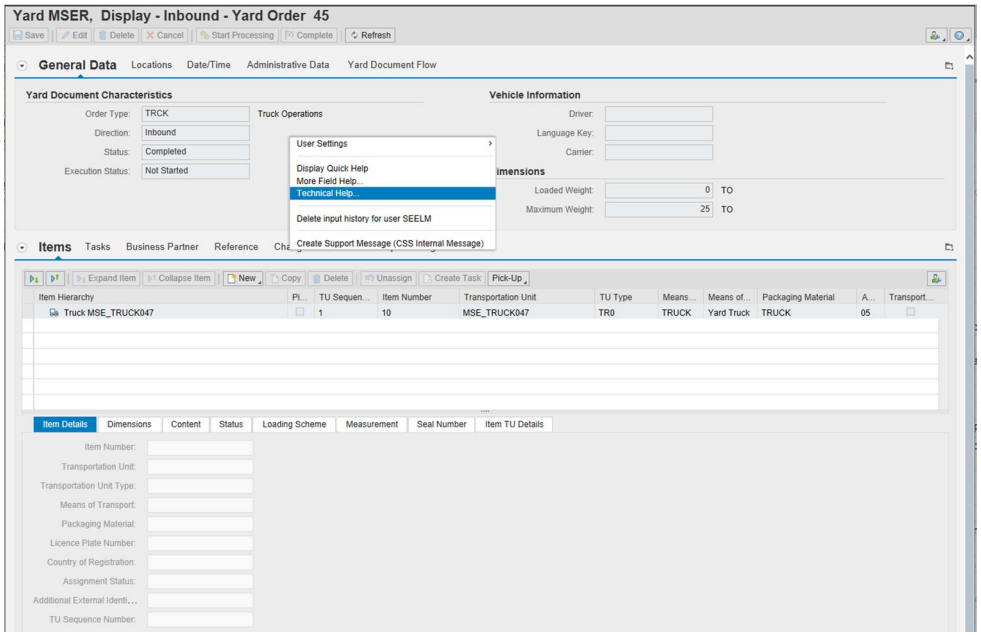

Figure 23: Technical Help

The technical help shows you all required information for configuration, customizing and personalization.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

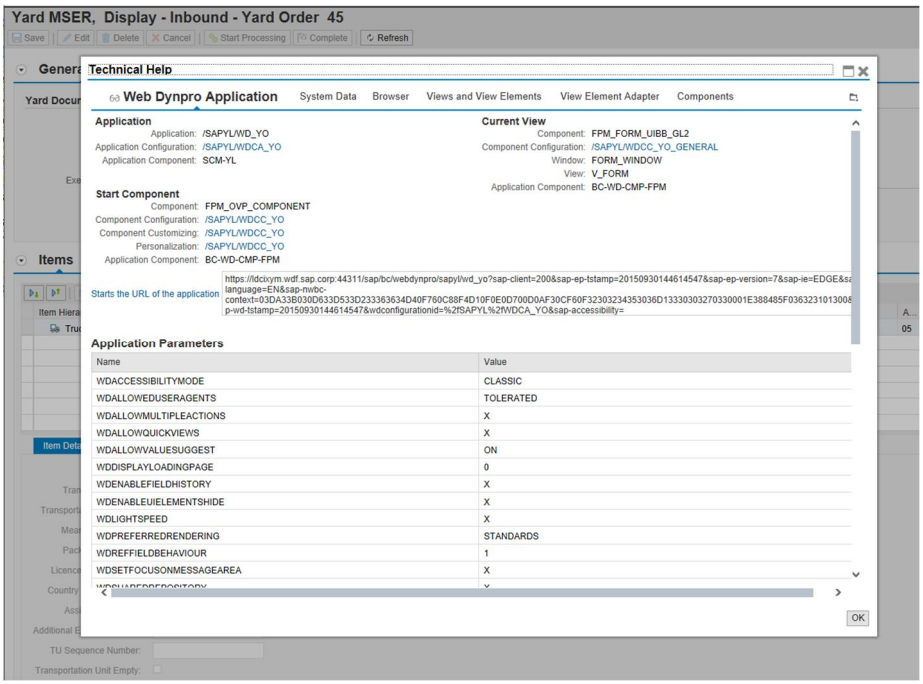

Figure 24: Configuration, Customizing and Personalization

On the tab *Web Dynpro Application* of the *Technical Help* screen, you can find the application configuration in the section *Application*. Click the link to navigate to the application configuration where you can display (and adjust) general application parameters.

In section *Start Component* of the same screen, you can find the leading Web Dynpro Component Configuration (WDCC). The links are listed here.

- Component Configuration

  It represents the starting configuration of the corresponding application and contains all sub-elements (UIBBs, Views, and so on) with their related configurations. When you display the component configuration, you can navigate further to all sub components of this application.

- Component Customizing

  In the component configuration, you can only see the standard content. Only in the component customizing, you can later see the enhancements that you did for a standard configuration. Moreover, only here you can add enhancements to existing configurations.
  Make sure that in transaction SICF, you activate the corresponding service. You can check this under the path: *sap* → *bc* → *webdynpro* → *sap* → *customize:component*. In case the service is not active, let your system administrator activate it. Otherwise, you cannot create any UI enhancements.

- Personalization

  When following this link, you get to the personalization settings for the specific application, that is, here you can see all personalization settings of each authorized user as well as the general personalization settings for the application that are valid for all authorized users.

In section *Current View*, you can find a link to the *Component Configuration* and *Component Customizing* that you have marked with the mouse pointer to start the technical help. That is, it allows navigating to the configuration and customizing of the UI component that currently has the focus.

To create enhancements, you need to start the corresponding user interface from the SAP user menu or from within NWBC. Within the user interface, you can then use the mentioned technical help (as per NW 7.31).

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

A UIBB might in turn include other UIBBs. So when entering the configuration of a UIBB in the configuration editor, you may have to navigate to further configurations in the Configuration Editor to get to the specific UIBB's configuration that gets enhancements.

For a general guide on how to adapt existing FPM application like those shipped within SAP Yard Logistics, see [SCNFPMENH].

There are also several explicit examples of FPM UI enhancements in the TM Enhancement Guide [TMENH] in section 5.4 like:

- Field Extensions
- Adding a new action to a toolbar
- Adding a new tab from a new BO sub node

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

5 External Business Function Interfaces

In release 2 of SAP Yard Logistics, we introduce external business function interfaces as standard SAP interfaces. They are remote function call enabled and can be compared with BAPIs (Business Application Programming Interfaces).

These new interfaces play an important role in the technical integration and in the exchange of business data between SAP Yard Logistics and other SAP or non-SAP components. These RFC-enabled business function interfaces enable you to integrate these components and are therefore an important part of developing integration scenarios where multiple components are connected to each other, either on a local network or on the Internet.

 Caution

The external business function interfaces described in this chapter only support the transaction model with a COMMIT WORK when they are called. For more details see [BAPITRCOMMIT].

5.1 Functions for Yard Orders and Yard Requests

Each function module has two BAdI calls used to extend the processing logic. The BAdIs are called before and after the processing block (based on the BOPF processing logic blocks). You can find out the calling point by checking the `iv_before_selection` importing parameter. If it has the value `abap_true`, you are before the processing block. If it has the value `abap_false`, the calling point is after the main processing block.

You have two extension tables, EXTENSION_IN and EXTENSION_OUT for each function module. The extension parameters are based on the BAPIPAREX structure.

The format of the data records in this table is determined by the BAPIPAREX structure. This structure contains multiple data record fields (VALUEPART1, VALUEPART2 ...) and one field for the name of the help structure (STRUCTURE). Since the data for each data record is written to the corresponding data record fields in parts one after the other, a help structure is required for interpreting the data.

You find in the next chapters the BAdI for each function module of the business function interface.

5.1.1 Function Module /SAPYL/BAPI_YO_CREATE

This function module is used to create yard request or a yard order with items, products, date and time, reference, item reference, dock appointment scheduling on item level, warehouse visit on item level, threshold on item level and business partner. The document category can be used to differentiate between yard order or yard request creation (see values for `/sapyl/if_constants=>gcs_document_category`).

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YO_CREATE.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

5.1.2 Function Module /SAPYL/BAPI_YO_UPDATE

This function module is used to update yard request or yard order with items, products, date and time, reference, item reference, dock appointment scheduling on item level, warehouse visits on item level, threshold on item level, checks and business partner.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YO_UPDATE.

5.1.3 Function Module /SAPYL/BAPI_YO_GET_LIST

This function module retrieves the yard request or yard order headers based on the selection criteria provided.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which could be processed by implementing the BAdI /SAPYL/BAPI_YO_GET_LIST.

5.1.4 Function Module /SAPYL/BAPI_YO_GET_DETAIL

This function module retrieves the yard request or yard order details based on the yard number, yard document category and yard document number.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YO_GET_DETAIL.

5.1.5 Function Module /SAPYL/BAPI_YO_EXECUTE

This function module executes the yard request or yard order action for the given document.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which could be processed by implementing the BAdI /SAPYL/BAPI_YO_EXECUTE.

5.2 Functions for Yard Tasks

5.2.1 Function Module /SAPYL/BAPI_YT_CREATE

This function module is used to create yard tasks with items.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YT_CREATE.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

5.2.2 Function Module /SAPYL/BAPI_YT_UPDATE

This function module updates the yard task with its items.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YT_UPDATE.

5.2.3 Function Module /SAPYL/BAPI_YT_GET_LIST

This function module retrieves the yard task headers based on the selection criteria provided.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YT_GET_LIST.

5.2.4 Function Module /SAPYL/BAPI_YT_GET_DETAIL

This function module retrieves the yard task details based on the yard number and yard task number.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YT_GET_DETAIL.

5.2.5 Function Module /SAPYL/BAPI_YT_EXECUTE

This function module executes the selected yard task action for the given yard task.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which could be processed by implementing the BAdI /SAPYL/BAPI_YT_EXECUTE.

5.3 Functions for Yard TUs

5.3.1 Function Module /SAPYL/BAPI_YTU_UPDATE

This function module updates the weight, loading scheme, license plate number, country, products and thresholds of the transportation unit.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YO_UPDATE.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

5.3.2 Function Module /SAPYL/BAPI_YTU_GET_LIST

This function module retrieves the yard transportation unit headers based on the selection criteria provided.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YTU_GET_LIST.

5.3.3 Function Module /SAPYL/BAPI_YTU_GET_DETAIL

This function module retrieves the yard transportation unit details based on the external yard transportation unit number.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YTU_GET_DETAIL.

5.3.4 Function Module /SAPYL/BAPI_YTU_EXECUTE

This function module executes the selected yard transportation unit action for the given yard transportation unit.

The tables EXTENSION_IN and EXTENSION_OUT should contain the customer-specific data, which can be processed by implementing the BAdI /SAPYL/BAPI_YTU_EXECUTE.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

6 Usage and Extensions of Internet of Things

SAP Yard Logistic provides the default integration with SAP HANA Cloud Platform (HCP) Internet of Things service (IoT). This linkage does not need any SAP Yard Logistics-specific installation on HCP but only the IoT and the persistency layer service needs to be activated. The following figure depicts an overview on the architecture.

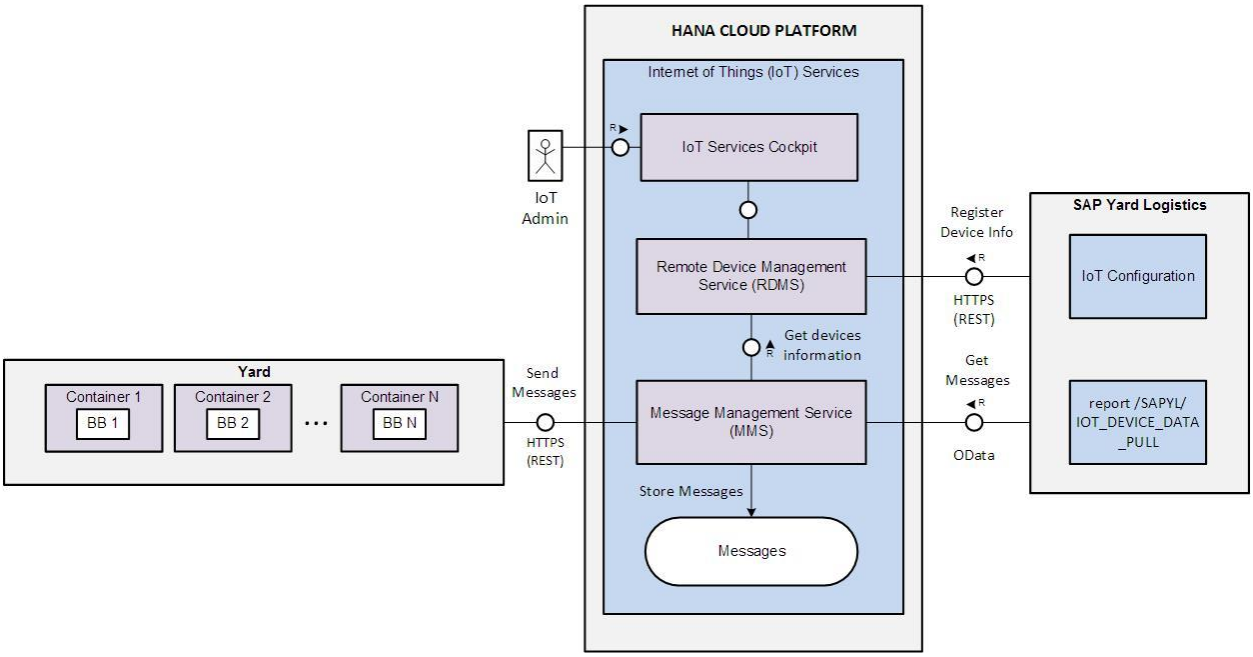

Figure 25: IOT Linkage in SAP Yard Logistics

In SAP Yard Logistics, the following functionality is supported:

- Definition of new message types and its registration in HCP
- Definition of new device types and its registration in HCP
- Definition and registration of a new device
- Configuration of mapping between the device's message data and Yard TU measurements
- Assignment of a device to Yard TU
- Sending the device message to HCP using the simulation report

Pulling the device data from HCP with updates of measurement details of a corresponding Yard Transportation Unit.

6.1 Configuration Tables

The information about connection endpoints to IoT services is stored in the database table /SAPYL/IOTCONN. This table keeps the data about RFC destinations to Device Management Service (DMS) and Message Management Service (MMS) for each logical system. The maintenance view is available for this table (transaction

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

/SAPYL/IOT_SETTINGS). The method GET_CONNECTION_SETTINGS of utility class /SAPYL/CL_IOTSERVICE provides the access to the content of this table.

As the default integration scenario assumes that SAP Yard Logistics initiates the communication to Device Management taking the role of primary storage and maintenance system for IoT device settings, the following database tables are provided to keep the configuration data.

Database Table	Description
/SAPYL/IOTDEV	Contains the list of all available devices
/SAPYL/IOTDEVTU	Provides the assignment of devices to transportation units
/SAPYL/IOTDTYPE	Contains the list of device types
/SAPYL/IOTDTYPEM	Assignment of message types to device types
/SAPYL/IOTMSG	Messages types defined in the system
/SAPYL/IOTMSGFLD	List of fields for messages types
/SAPYL/IOTMSGMSR	Mapping table between message types and TU measurement types

Figure 26: Tables used for IoT Linkage

6.2 Communication to Remote Device Management Service

IoT Device Management Service (DMS) provides the REST API that allows receiving or sending the data related to device configuration. The API is currently restricted to the use of the following technologies:

- HTTPS / REST
- JSON documents
- Supported HTTPS calls GET, POST, DELETE

The class /SAPYL/CL_IOTSERVICE_DMS implements the access functions to the DMS API. The implementation for communication and data exchange process is based on using the instance of REST HTTP client provided by class CL_REST_HTTP_CLIENT. The HTTP-client instance is initialized via constructor of class /SAPYL/CL_IOTSERVICE_DMS based on the configured RFC destination for Device Management Service taken from the application table /SAPYL/IOTCONN. Once the instance of HTTP-client is initialized, it can be used multiple times by the wrapper method EXECUTE_REST_REQUEST.

There are three HTTP methods, which are used by DMS class during the communication process:

- GET - requests information from DMS service about existing objects
- POST - provides data which must be created and stored within DMS database
- DELETE - requests for deletion of existing data in DMS database.

The JSON content is used during communication. In case of "read" methods (HTTP GET-request) the JSON is received from DMS service. The conversion of JSON content to ABAP structures is done by means of XSL transformations. The following XSL transformations are available:

XSL Transformation	Description
/SAPYL/ST_JSON_DEVICES	JSON to Devices List Transformation

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

XSL Transformation	Description
/SAPYL/ST_JSON_DEVICE_TYPES	JSON to Devices Types Transformation
/SAPYL/ST_JSON_MESSAGE_TYPES	JSON to Message Types Transformation

Figure 27: JSON Transformations for IoT

The methods of service class /SAPYL/CL_IOTSERVICE_DMS are mainly calls from the maintenance views of the corresponding tables. The sections below covers the methods of class /SAPYL/CL_IOTSERVICE_DMS in more details.

6.2.1 Device Types

A device type specifies a group of devices that share the same specification. In Internet of Things services, this specification is interpreted as the capability of sending, receiving, and processing certain types of messages.

Method REGISTER_DEVICE_TYPE is used to register a new device type. The device type name is required as the input parameter. In case of successful execution of device registration process, the device type ID is provided as the export parameter. The device type ID must be stored in YL database table by the caller, so it can be later on user to refer to this device during the follow-up calls to DMS.

Method UNREGISTER_DEVICE_TYPE is used to delete the device types from IoT service's database.

6.2.2 Message Types

Before a device can send messages to the Internet of Things services a device type has to be defined, and the supported messages of the device type need to be specified and assigned to the device type.

Method REGISTER_MESSAGE_TYPE is used to register a new message type. The message type name is required as the input parameter. This method also reads from application table /SAPYL/IOTMSGFLD the list of all fields related to the message type. In case of successful execution of message type registration process, the message type ID is provided as the export parameter.

Method UNREGISTER_MESSAGE_TYPE deleted the existing message type from IoT service.

6.2.3 Devices

A device represents a physical object that is able to send or receive messages.

Method REGISTER_DEVICE is used to register a new device. A device can be registered using different authentication methods. In addition to Basic Authentication, a device may be added using either an OAuth token. When registering a device, the IoT service provides the device ID as well as the device token. Both are returned by the method REGISTER_DEVICE.

Method UNREGISTER_DEVICE deletes the device from IoT service.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

6.3 Communication to the Message Management Service

The Message Management Service (MMS) provides interfaces, which can be used by devices to send messages. Currently the MMS supports HTTP(S), WebSocket and MQTT over WebSocket.

Yard Logistics uses HTTP access to MMS. For that purpose the gateway service /SAPYL/IOTSERVICES_MMS_SRV was generated based on OData service provided by MMS. The generated service acts as a proxy connecting Yard Logistics to IoT MMS. The generated runtime artifacts include two classes:

- /SAPYL/CL_IOTSERVICES_MMS_MPC – metadata processing class, derives from /IWBEP/CL_OCI_MP with redefinition the method GET_OCI_NAMESPACE – sets the remote service namespace “v1/api/http/app.svc”
- /SAPYL/CL_IOTSERVICES_MMS_DPC – data processing class which fully derives from default class /IWBEP/CL_OCI_DP

In addition to the generated gateway, service the wrapper service class /SAPYL/CL_IOTSERVICE_MMS provides the access to MMS and is used by data pull report. The following methods are defined:

- GET_DATA – calls MMS service to get all messages for the given device (import parameter IT_DEVICE_NAME) received by IoT service within the defined period in seconds (import parameter IV_LAST_SECONDS)
- SEND_MESSAGE – Sends the message MMS on behalf on specified device name. The method usage is to message test report in order to simulate the devices sending messages.
- EXECUTE_REQUEST – internal method to execute the HTTP request to oData service via the gateway framework. The received XML is mapped and converted into ABAP structures.

The data retrieval process from IoT MMS is based on pull principle and initiated by Yard Logistic. The report /SAPYL/IOT_DEVICE_DATA_PULL is executed to receive the message data for all active transportation units in the yard and map this data to TU measurements. This report can either be scheduled for automatic executed or run manually.

6.4 Configuration of Connection Settings to IoT Service

Prior to start using the IoT services, the connection endpoints to SAP HANA Internet of Things services must be configured. This includes:

Setting up RFC destinations (transaction SM59) to HCP Device Management Service (DMS) and Message Management Service (MMS). The example of the settings is shown on the screenshot below:

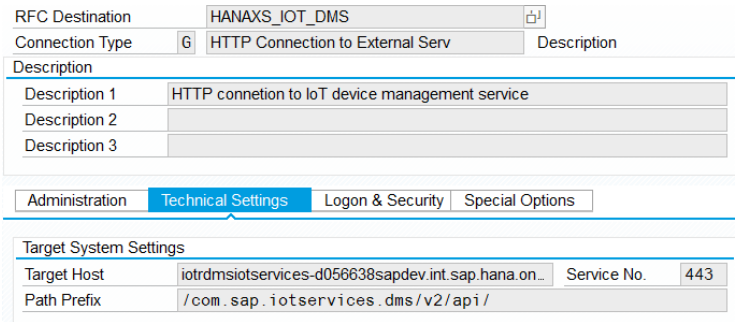

Figure 28: HTTP Connection to DMS (sample)

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

Setting up the system alias and activation of the SAP NetWeaver Gateway service /SAPYL/IOTSERVICES_MMS_SRV for MMS (transaction code /IWFND/MAINT_SERVICE):

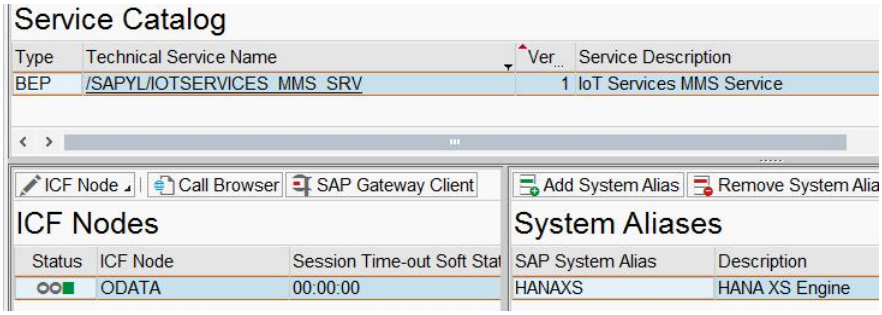

Figure 29: Setup of the Gateway Service for MMS

Assignment of DMS RFC destination and MMS alias to the logical system (IMG: Yard Logistics à Integration à Internet of Things à Maintain Connection Settings for Internet of Things):

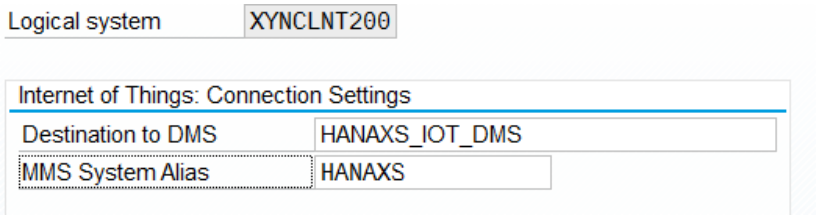

Figure 30: Logical System and DMS link (sample)

The initialization of the HANA HCP account and the configuration of the IoT Service are also required but not described here.

More Information

For more information see the SAP Online Help for SAP HANA Cloud Platform Internet of Things Services, available at https://help.hana.ondemand.com/iot/frameset.htm?ad829c660e584c329200022332f04d00.html

6.5 Enhancements for the IoT Services

The major enhancement is to adjust the content received from HCP IoT services before this content is mapped to TU measurements. For this purpose the BAdI /SAPYL/IOT_MESSAGE_INBOUND is provided within the enhancement spot /SAPYL/ES_IOT_INTEGRATION. This BAdI is called from MMS integration service class /SAPYL/CL_IOTSERVICE_MMS after the data has been received from IoT web-service.

The interface of the BAdI provides two methods:

- CHANGE_RAW_CONTENT – used to change the raw oData content which have been received from the oData service
- CHANGE_MESSAGE – is called after the raw oData content has been converted and mapped to ABAP structure /SAPYL/S_IOT_MESSAGE, which is passed to the method as changing parameter CS_MESSAGE.

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

By implementing this method, it is possible to make the final adjustments for the inbound message before it will be processed by the mapping logic that will update the measurements of the target transportation unit.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

7 Usage and Extensions of EH&S

For extension of EH&S functionality for Yard Logistics the following BAdIs can be used:

- /SAPYL/BADI_DG_DETERMINE_TU - BAdI for Dangerous Goods Data Redetermination for Transportation Unit
- /SAPYL/DG_CHECK - BAdI for Dangerous Goods Checks (with a sample for a separation check)
- /SAPYL/DG_CHECK_CICO - BAdI for Dangerous Goods Checks during Check-In/Check-Out (with a sample)

7.1.1 Dangerous Goods checks during location determination

During destination location determination for Yard Task several checks for dangerous good run:

- check on allowed hazardous materials
- check on maximum storage quantity
- check maximum stacking height for containers

All these checks run within the BAdI for location determination /SAPYL/LOCATION_DETERM. This BAdI is triggered with the set of prepared products, which are collected in the BAdI /SAPYL/HAZARD_RATING_DETERM. Both BAdIs are triggered within the determination DET_DEST_LOCATION of BO Yard Task /SAPYL/YARD_TASK, node ROOT.

7.1.2 Visualization EH&S information in YR/YO item and TU content

During content maintenance in Yard Order and Transportation Unit, dangerous goods mater data is automatically retrieved based on product and dangerous good regulation.

A possibility to overwrite the product information is foreseen for both BOs.

- For BO /SAPYL/YARD_ORDER a BAdI /SAPYL/BADI_DG_DETERMINE_YO is used. The interface provides data of Yard Order Root, Yard Order Item and products. BAdI is triggered in the determination DET_PRODUCT_GET_PRODUCT_T of BO /SAPYL/YARD_ORDER, node PRODUCT.
- For BO /SAPYL/YARD_TU a BAdI /SAPYL/BADI_DG_DETERMINE_TU is used. The interface provides data of the nodes ROOT, SR_ACT and products. BAdI is triggered in the determination DET_PRODUCT_GET_PRODUCT_T of BO /SAPYL/YARD_TU, node PRODUCT.

7.1.3 Dangerous Goods check during Yard Order Check-In

During check-in/check-out posting checks should happen that control whether the entrance/leaving the yard is allowed or not. To support this functionality a BAdI /SAPYL/DG_CHECK_CICO is used. Within this BAdI the DG

CUSTOMER
© 2016 SAP SE or an SAP affiliate company. All rights reserved.

specific checks have to be executed based on all attributes of a YR/YO item. The BAdI runs within the BOPF actions ACT_ROOT_CHECKIN and ACT_ROOT_CHECKOUT of the BO /SAPYL/YARD_ORDER, node ROOT.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

8 UI5 Fiori Enhancements

8.1 Gateway Extensions

For information on how to enhance gateway services, see [HELPGTWENH].

8.2 UI5 Fiori Extension

For extending Fiori (SAPUI5) application, you can use the concept of extension points. The main idea of this approach is to have some special tags in the code (called extension points), where new Views or Fragments can be inserted.

Apart from specifying new Views/Fragments/Controllers, you need to make some changes in the component file, which contain basic information of Views/Controllers available in the application. For that, you need to extend Component.js file.

You can find additional information about the used concepts in [SCNFIORIEXT].

8.2.1 Yard Task Application Extension

Currently, in Yard Task application you can extend the following functionalities:

- Form fields that are defined in TasksTab.view.xml
- New tabs can be added to Details.view.xml

5.2.1.1 Extended Component

Extended component can have description of Views (Fragments) and Controllers, which is used in extended application. The main idea is to create new Component.js file, which can be used as a starting point for the application. This new Component.js should inherit from the standard app's Component.js.

Additional Information: Example: Component Configuration

In the example below, initial configuration for form fields/ tabs extension is placed:

```
jQuery.sap.declare("com.sap.cd.yl.ui.manage.tasks.customer.Component");
jQuery.sap.require("com.sap.cd.yl.ui.manage.tasks.Component");

com.sap.cd.yl.ui.manage.tasks.Component.extend("com.sap.cd.yl.ui.manage.tasks.customer
.Component", {
    metadata : {
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
        "customizing": {
            "sap.ui.viewExtensions": {
                "com.sap.cd.yl.ui.manage.tasks.view.TasksTab": {
                    "CustomFieldExt": {
                        "className": "sap.ui.core.Fragment",
                        "fragmentName":
"com.sap.cd.yl.ui.manage.tasks.fragment.CustomField",
                        "type": "XML"
                    }
                },
                "com.sap.cd.yl.ui.manage.tasks.view.Detail": {
                    "CustomTabExt": {
                        "className": "sap.ui.core.Fragment",
                        "fragmentName":
"com.sap.cd.yl.ui.manage.tasks.fragment.CustomTab",
                        "type": "XML"
                    }
                }
            }
        }
    }
});
```

For the described extended Component, there is an assumption used that Component is placed in a folder named *customer* and this module is visible for SAPUI5. The method ‘registerModulePath(...)’ is used for local testing.

In Fiori Launchpad, you need to specify an appropriate component path in navigation targets configuration. See Customizing Navigation Targets.

5.2.1.2 Add New Fields to the Form

Extension point for new form fields is specified In TasksTab.view.xml:

```
<core:ExtensionPoint name="CustomFieldExt"/>
```

Configuration of this extension point is mentioned in the section **Error! Reference source not found.**.

In the configuration, it is specified that fragment should use specific fragment.

In the example below, the fragment is shown (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
    <Label text="New Field" />
    <Text text="{fieldNameInEntity}"/>
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
</core:FragmentDefinition>
```

As shown in this example, you can also use some custom formatter. As those fields are created for displaying the data, this will be better if only Label and Text controls will be added.

`fieldNameInEntity` is the name of a new property added to `YardTaskSet` entity

To see how you can enhance the translation texts, refer to Handling Localized Texts for Extended Applications

For further information, see View Extension.

5.2.1.3 Add New Tab to Details View

In Details.view.xml after last IconTabFilter control new extension point is specified:

```
<core:ExtensionPoint name="CustomTabExt"/>
```

The configuration of this extension point was mentioned in section on **Error! Reference source not found.**. In the configuration, it is specified that the fragment should use specific fragment. The content of the fragment is as follows:

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:l="sap.ui.layout"
        xmlns:mvc="sap.ui.core.mvc"
        xmlns:core="sap.ui.core">
    <IconTabFilter
            id="customTabFilter"
            text="Custom Tab"
            icon="sap-icon://expand"
            key="customTab">
        <content>
            <mvc:XMLView id="CustomTabContent"
viewName="com.sap.cd.yl.ui.manage.tasks.view.CustomTabContent"/>
        </content>
    </IconTabFilter>
</core:FragmentDefinition>
```

In this fragment, a separate XMLView is defined. Content of that view is as follows:

```
<mvc:View
        controllerName="com.sap.cd.yl.ui.manage.tasks.view.CustomTabContent"
        xmlns:mvc="sap.ui.core.mvc"
        xmlns:l="sap.ui.layout"
        xmlns="sap.m">
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
    <l:Grid defaultSpan="L12 M12 S12"
            width="auto">
        <l:content>
            <VBox class="sapUiSmallMargin">
                <Text text="Lorem ipsum dolor st amet, consetetur sadipscing elitr,
                        sed diam nonumy eirmod tempor invidunt ut labore et dolore" />
            </VBox>
            <Button press="press" text="Press"/>
        </l:content>
    </l:Grid>
</mvc:View>
```

You need to also create an appropriate controller (CustomTabContent) if some events need to be handled for the view.

```
sap.ui.controller("com.sap.cd.yl.ui.manage.tasks.view.CustomTabContent", {
       onInit: function() {
       },
       press: function(){
       },
       onExit: function() {
       }
});
```

8.2.2 Self Check-In Application Extensions

Currently, in Self Check-In application you can extend the following functionalities:

- New steps can be added to Wizard View
- Replace a step in Wizard View
- Replace or add content in a step fragment
- Add a new form in Summary View
- Replace or add content in a form in Summary View
- New UI controls can be added to the header bar in Wizard and Summary Views
- New UI controls can be added to the footer bar in Summary View

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

8.2.2.1 Extended Component

Extended component can have description of Views (Fragments) and Controllers, which is used in extended application. The main idea is to create new Component.js file, which can be used as a starting point for the application. This new Component.js should inherit from the standard app's Component.js.

Additional Information: Example: Component Configuration

In the example below, initial configuration for form fields/ tabs extension is placed:

```
jQuery.sap.declare("com.sap.cd.yl.ui.selfcheckin.customer.Component");
jQuery.sap.require("com.sap.cd.yl.ui.selfcheckin.Component");

com.sap.cd.yl.ui.selfcheckin.Component.extend("com.sap.cd.yl.ui.selfcheckin.customer.C
omponent", {
    metadata : {
        "customizing": {
            "sap.ui.viewExtensions": {
                "com.sap.cd.yl.ui.selfcheckin.view.Wizard": {
                    "customFirstWizardStep": {
                        "className": "sap.ui.core.Fragment",
                        "fragmentName":
"com.sap.cd.yl.ui.selfcheckin.view.fragments.CustomFirstStep",
                        "type": "XML"
                    }
                },
                "com.sap.cd.yl.ui.selfcheckin.view.fragments.AdditionalInfo": {
                    "customAdditionalInfoFields": {
                        "className": "sap.ui.core.Fragment",
                        "fragmentName":
"com.sap.cd.yl.ui.selfcheckin.view.fragments.CustomInfo",
                        "type": "XML"
                    }
                }
            }
        }
    }
});
```

For the described extended Component, there is an assumption used that Component is placed in a folder named *customer* and this module is visible for SAPUI5. In Fiori Launchpad, you need to specify an appropriate component path in navigation targets configuration. See Customizing Navigation Targets.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

8.2.2.2 Add new steps to Wizard View

Extension points for new steps are specified In Wizard.view.xml:

```
<core:ExtensionPoint name="customFirstWizardStep"/>
```

Configuration of this extension point is mentioned in the section **Error! Reference source not found.**.

In the example below, the fragment is shown (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
    <WizardStep title="{i18n>newStepTitle}">
      <content>
            ...
      </content>
    </WizardStep>
</core:FragmentDefinition>
```

A new step can be added in any place within Wizard.view.xml. Content needs to be specified according to enhancement requirements.
In the example above you can find that title is connected to Resource Model (i18n). To see how you can enhance the translation texts, refer to Handling Localized Texts for Extended Applications

For further information, see View Extension.

It is required to create a new controller (CustomController) in order to handle view/fragment events and data binding.

```
sap.ui.controller("com.sap.cd.yl.ui.selfcheckin.controller.CustomController", {
      onInit: function() {
      },
      onExit: function() {
      }
});
```

The following customizing merges custom and standard controllers:

```
customizing: {
  "sap.ui.controllerExtensions": {
    "com.sap.cd.yl.ui.selfcheckin.controller .Wizard": {
      controllerName: " com.sap.cd.yl.ui.selfcheckin.controller.CustomController "
    }
  },
```

For additional information, see Controller Extension

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

8.2.2.3 Replace a step in Wizard View

In order to replace a step In Wizard.view.xml, extension points with default content are specified:

```
<core:ExtensionPoint name="customInfoWizardStep">
	<WizardStep id="selfCheckInAdditional" title="{i18n>additionalInfoTitle}">
		<content>
			...
		</content>
	</WizardStep>
</core:ExtensionPoint>
```

In the example below, the fragment is shown (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
	xmlns="sap.m"
	xmlns:core="sap.ui.core">
    <WizardStep title="{i18n>newAdditionalInfoStepTitle}">
	<content>
		...
	</content>
    </WizardStep>
</core:FragmentDefinition>
```

For more information regarding extension points with default content see View Extension.

8.2.2.4 Replace or add content in a step fragment

Each step is represented by a separate fragment. At this moment, three are available: Identification.fragment.xml, AdditionalInfo.fragment.xml and Questionnaire.fragment.xml.

It is possible to add new UI elements to each fragment or completely redesign existing functionality with new content.

For instance, extension point in AdditionalInfo.fragment.xml with default content:

```
<core:ExtensionPoint name="customAdditionalInfoFields">
	<Label id="idDriverName" text="{i18n>driverNameLabel}" required="true"/>
	<SearchField id="idDriverNameSearch" placeholder="{i18n>driverNamePlaceholder}"
liveChange="validateAdditionalInfo" showSearchButton="false"
	value="{ui>/AdditionalInfo/DriverType}"/>
	<Label id="idCarrier" text="{i18n>carrierLabel}" required="true"/>
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
        <SearchField id="idCarrierSearch" placeholder="{i18n>carrierPlaceholder}"
liveChange="validateAdditionalInfo" showSearchButton="false"
        value="{ui>/AdditionalInfo/Carrier}"/>
        ....
</core:ExtensionPoint>
```

Configuration of this extension point is mentioned in the section **Error! Reference source not found.**.

In the example below, fields are placed in different order with a new additional field (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
    xmlns="sap.m"
        xmlns:form="sap.ui.layout.form"
    xmlns:core="sap.ui.core">
        <Label text="New Field" />
        <Text text="{ui>/fieldNameInModel}"/>
        <Label id="idCarrier" text="{i18n>carrierLabel}" required="true"/>
        <SearchField id="idCarrierSearch" placeholder="{i18n>carrierPlaceholder}"
liveChange="validateAdditionalInfo" showSearchButton="false"
        value="{ui>/AdditionalInfo/Carrier}"/>
        <Label id="idDriverName" text="{i18n>driverNameLabel}" required="true"/>
        <SearchField id="idDriverNameSearch" placeholder="{i18n>driverNamePlaceholder}"
liveChange="validateAdditionalInfo" showSearchButton="false"
        value="{ui>/AdditionalInfo/DriverType}"/>
        ....
</core:FragmentDefinition>
```

`fieldNameInModel` is the name of a new property added to local JSON model and contains a value of added field.

For further information, see View Extension.

8.2.2.5 Add a new form in Summary View

Extension points for new forms are specified In Summary.view.xml:

```
<core:ExtensionPoint name="customFirstSummaryExt"/>
```

In the example below, the fragment is shown (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:form="sap.ui.layout.form"
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
        xmlns:core="sap.ui.core">
    <form:SimpleForm id="newId" layout=" ResponsiveGridLayout">
      <form:content>
            ...
      </form:content>
    </form:SimpleForm>
</core:FragmentDefinition>
```

A new form can be added in any place within Summary.view.xml. Content needs to be specified according to enhancement requirements.

For further information, see View Extension.

8.2.2.6 Replace or add content in a form in Summary View

In order to replace content of a form In Summary.view.xml, extension points with default content are specified:

```
<form:content>
        <core:ExtensionPoint name="customSummaryIdentification">
                <Label id="idSummaryIdent" text="{ui>/identificationLabel}"/>
                <Text text="{ui>/selectedNumber}"/>
        </core:ExtensionPoint>
</form:content>
```

In the example below, the fragment is shown (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
       <Label text="New Field" />
      <Text text="{ui>/fieldNameInModel}"/>
</core:FragmentDefinition>
```

As those fields are responsible only for displaying the data, this will be better if only Label and Text controls will be added. `fieldNameInModel` is the name of a new property added to local JSON model.

To add additional fields and leave already implemented content, it is needed to copy existing fields and adjust them according to new requirements. For instance, the example of this implementation:

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
        <Label id="idSummaryIdent" text="{ui>/identificationLabel}"/>
        <Text text="{ui>/selectedNumber}"/>
```


© 2016 SAP SE or an SAP affiliate company. All rights reserved.

```
        <Label text="New Field" />
        <Text text="{ui>/fieldNameInModel}"/>
</core:FragmentDefinition>
```

For further information, see View Extension.

8.2.2.7 Add new UI controls to the header bar in Wizard and Summary Views

Extension points for new UI controls to be added to the header bar are specified In Wizard.view.xml and Summary.view.xml:

<core:ExtensionPoint name="customWizardHeaderExt"/>

<core:ExtensionPoint name="customSummaryHeaderExt"/>

For instance, new button can be added with the following structure (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
        <Button text="New Header Button" press="newHeaderPressEventHandler"/>
</core:FragmentDefinition>
```

For further information, see View Extension.

8.2.2.8 Add new UI controls to the footer bar in Summary View

Extension points for new UI controls to be added to the footer bar are specified only In Summary.view.xml:

<core:ExtensionPoint name="customSummaryFooterExt"/>

In the example below, the fragment is shown (this fragment needs to be created corresponding to the path defined in extended Component.js):

```
<core:FragmentDefinition
        xmlns="sap.m"
        xmlns:core="sap.ui.core">
        <Button text="New Footer Button" press="newFooterPressEventHandler"/>
</core:FragmentDefinition>
```

For further information, see View Extension.

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

www.sap.com/contactsap

© 2016 SAP SE or an SAP affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.
National product specifications may vary.
These materials are provided by SAP SE and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE in Germany and other countries. Please see www.sap.com/corporate-en/legal/copyright/index.epx#trademark for additional trademark information and notices.