PUBLIC

Extensibility Guide for JIT-EWM-Integration

Embedded EWM in S/4HANA 2020 OP

September 2020

Version 1.0

THE BEST RUN SAP

DISCLAIMER

This document provides some examples of how warehouse relevant data can be accessed. The description of functions and methods in this document does not mean that these functions/methods have been released by SAP or that SAP guarantees that they will be kept stable. SAP may change or remove them without any further notice even if this is unlikely to happen. Also, do not use any other methods or parameters from the classes mentioned. When using other methods in a wrong way this may lead to data inconsistencies. Only use the methods mentioned in this document. Any use is at your own risk.

Any coding included in this document is only meant as example and is not intended to be used in a productive environment. The code is only used to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the code given herein, and SAP shall not be liable for errors or damages caused by the usage of the code, except if such damages were caused by SAP intentionally or grossly negligent.

TABLE OF CONTENTS

1 INTRODUCTION

This guide provides a rough overview of the extension possibilities for the Just-in-Time (JIT) integration into Extended Warehouse Management (EWM) and can be used for custom implementation projects. While aiming for JIT-specific extensions this guide also describes existing warehouse request processing and wave management which can still be used for the new JIT-EWM integration. Technical details are provided on implementation level so this guide can be used by developers to understand the most basic extensibility options.

The guide is divided into three major parts: The first part (Introduction) provides a high-level overview of the JIT process from a business point of view and the architectural fundamentals. You can find end-to-end extensibility scenarios in the second part (Integration Scenarios) which show the fundamentals of extending the JIT-EWM integration with customer specific attributes and processes. The last part (Building Blocks for JIT Processing) contains detailed information about the JIT specific extensibility options such as newly developed BAdIs or customizing.

1.1 Business Overview

JIT calls represent demand for a specific product of a certain quantity at a defined time and place in production processes. These demands can be fulfilled either by external or internal replenishment. The JIT-EWM integration is part of the internal replenishment process and connects an EWM managed warehouse to production lines so stock can be supplied as required directly from a warehouse. Goods need to be transferred from regular storage areas to production supply areas configured by control cycles so they can be combined and built into a final product. The supply process can be depicted as following:

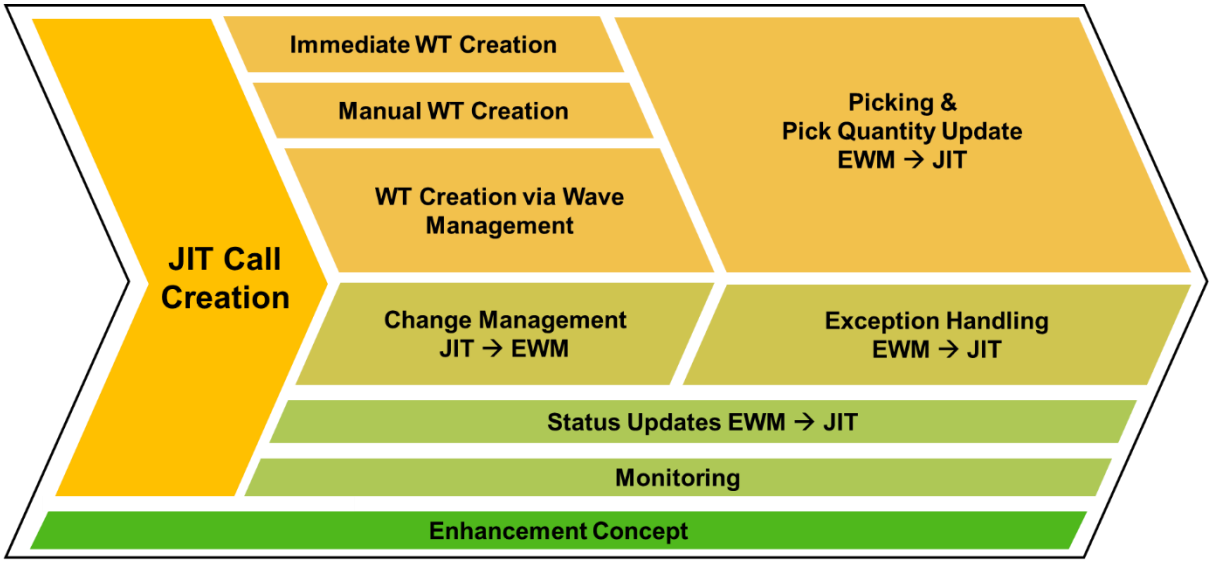

Figure 1: JIT Supply to Production - Process Overview

Once a JIT call has been created, its demand needs to be reflected somehow in the warehouse. Therefor a warehouse request is created which contains all relevant information for warehouse processing. Based on this warehouse request, warehouse tasks can be created either automatically (immediate), manually or via wave management. As long as no warehouse tasks were confirmed, a JIT call can be changed from the JIT component e.g. by updating the requested quantity or demand time. Quantity and status updates are sent from EWM to JIT if the created warehouse request or tasks change in some way (e.g. when picking is confirmed). The whole process can be monitored using the warehouse monitor (transaction `/SCWM/MON`).

The enhancement concept ensures that the standard process can be adopted to customer needs using different settings, customizing options and Business Add-In (BAdI) implementations. This guide provides an overview on how to adapt the processes accordingly.

1.2 Architectural Overview

This section provides a short overview of the underlying architecture for the JIT-EWM integration and is focused on the EWM side of the implementation. Besides the general architecture also an overview of the extensibility options is depicted. The architecture aims at the following points:

- Being able to **reuse existing EWM functionality** so (custom) implementations can be done quickly without the need of developing important features twice
- Keeping the implementation performant and being able to **process JIT calls near real-time** to support large scale deployments while not bottlenecking the production supply
- Being able to **customize the standard process** in detail to meet a wide variety of custom requirements

1.2.1 Fundamentals

Every JIT call is represented as a warehouse request of the light-weight document type Internal Stock Transfer (WMR) in EWM. The warehouse request stores all relevant data for the warehouse processing such as the JIT call priority, requested quantity, demand time and production supply area. Changes of the JIT call (e.g. requested quantity) are reflected on the warehouse request as well as changes of the warehouse request (e.g. status updates during picking) are reflected on the JIT call. These updates are processed near real-time to keep both objects in synch and allow transparent monitoring from both sides. The warehouse request is also used to create the warehouse tasks which are needed to fulfill the stock requirements.

Using the stock transfer document as JIT call representation includes the re-use of its extensibility and flexibility elements:

- **Warehouse Request Processing** with its existing customizing and BAdIs can be reused and ensures compatibility with standard procedures in EWM.
- **Warehouse Task Management** with its existing customizing and BAdIs can be reused and ensures compatibility with standard procedures in EWM. It also allows for deeper integrations like wave management or the distribution equipment process.
- **Programmatic Access** is enabled by default via common APIs like the deliveries service provider or business object management.

EWM relevant JIT calls and warehouse requests are linked with a 1:1 cardinality. A JIT call consists of three levels: header, component group and component. The warehouse request has only two hierarchy levels: header and item. However, the cardinality between JIT call header and component group is 1:1. Hence, a warehouse request header is able to reflect data from a JIT call header and component group. The cardinality between components of a JIT call and warehouse request items is 1:1.

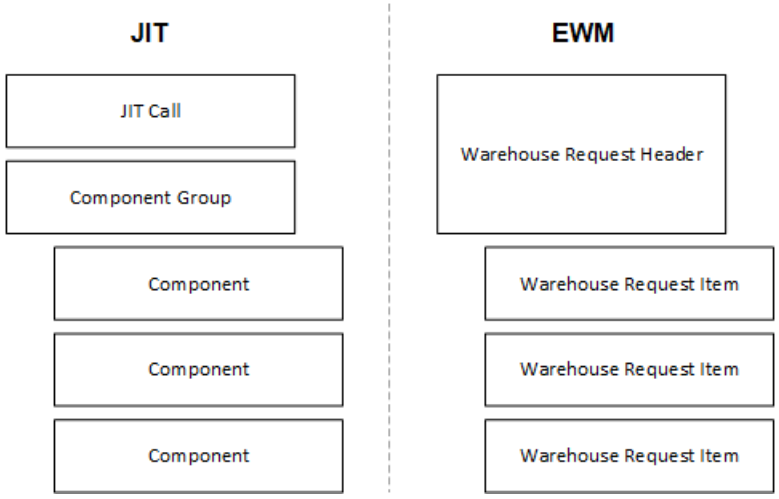

Figure 2: JIT Call Representation in EWM

1.2.2 Extensibility Options

Because a JIT call is represented by a warehouse request in EWM, all existing extensibility options for warehouse requests can be also used for the warehouse part of the JIT process. Besides those already

existing methods there are also new options available specific to the JIT-EWM integration. These allow to control aspects which are only relevant for the production supply processes and are usually used as an adapter into the existing warehouse logic. One example of this would be the JIT specific warehouse process type (WPT) determination which allows to determine the WPT based on parameters which are relevant for production supply like the production supply area or JIT action control. Some parts of the following warehouse logic can be setup based on this WPT and allow to react accordingly.

The following diagram depicts the JIT call creation on a high level from a technical point of view with the most basic extensibility options. There are more extensibility options available, but the mentioned ones are JIT specific or relevant for most implementation projects.

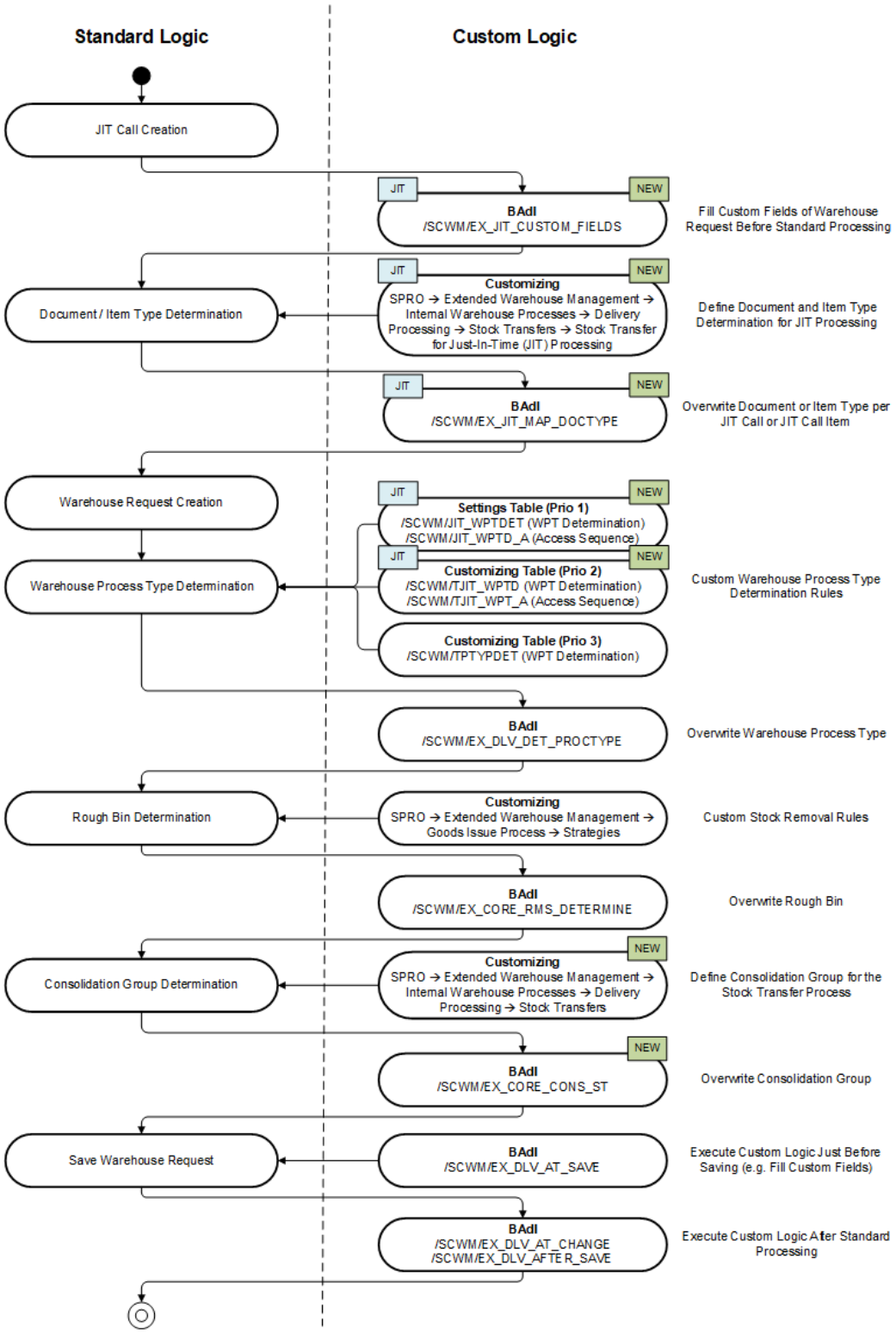

Figure 3: Warehouse Request Processing for JIT Call Creation – Overview

After a JIT call has been created in the JIT component, it will trigger the warehouse request creation on EWM side. First, the BAdI `/SCWM/EX_JIT_CUSTOM_FIELDS` is called to be able to fill custom fields of the warehouse request or execute custom logic directly before warehouse request creation. The next step is the warehouse request document and item type determination based on warehouse-dependent customizing settings. This can be overwritten using BAdI `/SCWM/EX_JIT_MAP_DOCTYPE`.

Once the warehouse request is created, the WPT determination is run. The WPT is a control parameter for the EWM processes. Its determination in JIT processes is also possible using JIT specific attributes overruling the standard logic which remains as fallback. It is recommended to use distinct WPTs for JIT processing. The determined WPT can be overwritten using BAdI `/SCWM/EX_DLV_DET_PROCTYPE`.

The next step in processing is rough bin determination if it is activated in the WPT definition. The result of the rough bin determination can also be changed by implementing BAdI `/SCWM/EX_CORE_RMS_DETERMINE`. Furthermore, the consolidation group is determined based on its selected standard determination criteria WPT and priority, but it can also be overwritten using BAdI `/SCWM/EX_CORE_CONS_ST`.

Once all determinations were run and the warehouse request is ready to be saved, the BAdI `/SCWM/EX_DLV_AT_SAVE` is called right before the save. This allows to make last changes to the warehouse request, such as custom fields or external components, before it is saved. Once the warehouse request was finally saved, the BAdI `/SCWM/EX_DLV_AFTER_SAVE` is called and can be used to cleanup any buffered data or execute further custom logic.

The execution order is similar in the case of JIT call updates and cancellations. In general it is a good idea to take a look at the enhancement options of deliveries which are described in SAP Note 1414179 (third attachment). This SAP note describes the delivery enhancements on a more fine granular level and is not specific to the JIT process but can be used for every development with relation to the delivery documents.

Not only the warehouse request processing can be extended but also the warehouse task processing. Extending the warehouse tasks does heavily depend on what processes you are using in the warehouse. The following diagram depicts the typical process of creating a warehouse task which is assigned to a wave. Like it is done with the warehouse requests, also the warehouse tasks for JIT processes can be extended like any other warehouse task. These are not specific to JIT and should be differentiated by their WPT and document type.

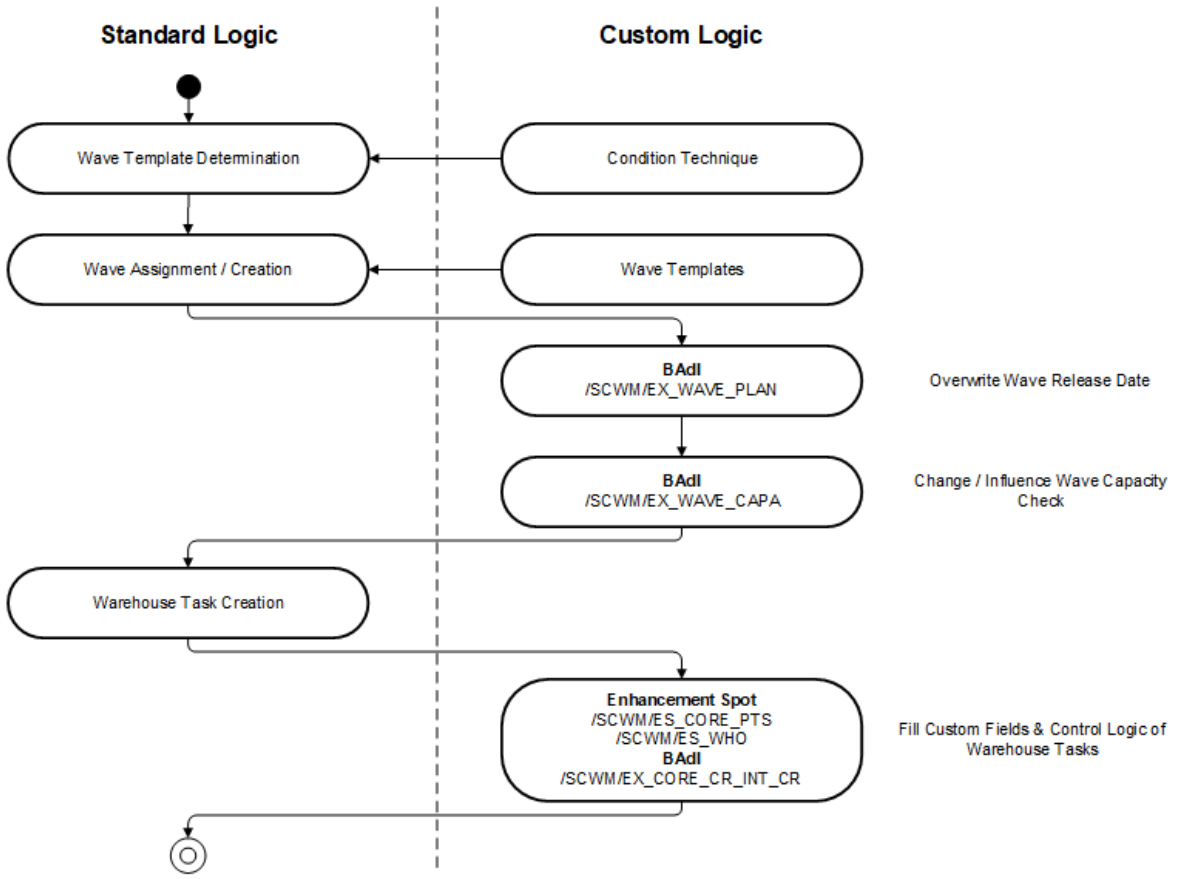

Figure 4: Warehouse Task Processing – Overview

2 INTEGRATION SCENARIOS

The following section describes common scenarios which are likely to be relevant for an implementation project. The technical details are focused on the EWM part of a custom implementation. There might be additions which are not described in detail and which need to be done on JIT side, e.g. BAdI implementations for custom fields on JIT. Besides the process-oriented descriptions in this chapter you can also take a closer look at some BAdIs in the next chapter (Building Blocks for JIT Processing) or directly in the system. There are also example implementations in the system available which show working implementations on a more technical level. When implementing BAdIs please make sure to follow the EWM transaction control which is described in SAP Note 1451135 or your implementations might lead to data inconsistencies.

2.1 Transfer Custom Fields from JIT Call to Warehouse Task

The documents in the standard implementation contain only fields which are relevant for standard processes. There are cases where this might not be sufficient and documents like the warehouse request need to be extended with custom fields in an implementation project. These custom fields might be added on the header as well as the item level depending on the use case and where applicable. This integration scenario shows you how you can transfer custom fields from a JIT call over the warehouse request to warehouse tasks.

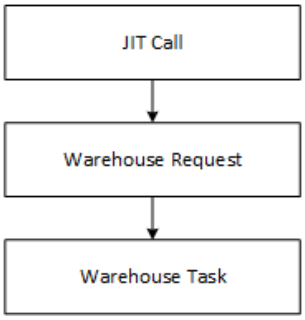

Figure 5: Hierarchy (JIT → EWM)

Before extending any structures, you should be aware of what you want to achieve with the custom fields, and in which processes these are relevant. Custom fields don't need to be persisted on every layer and can also be read dynamically in BAdI implementations on lower levels if they are persisted on a higher level. Take the following examples into consideration:

Relevance in	Persisted in
JIT Processing	JIT Call Header / Component Group / Component
Warehouse Request Processing or Warehouse Monitor	Warehouse Request
Warehouse Task Processing	Warehouse Request and Warehouse Task

You can also combine the mentioned examples and extend multiple structures by the same fields as needed. When persisting the same custom fields on multiple layers it is a good practice to use the same append structure in all include structures to keep the structures organized and to keep maintenance simple. In this case you also need to make sure to reflect updates of the custom fields on higher levels onto lower levels and vica versa if required. Depending on your use case you might extend the following include structures:

Document	Component	Structure
JIT Call	JIT	NJIT_CALL_S2P_INCL_EEW_PS
Component Group	JIT	NJIT_CALL_S2P_CGRP_INCL_EEW_PS
Component	JIT	NJIT_CALL_S2P_CMAT_INCL_EEW_PS
Warehouse Request Header	EWM	/SCDL/INCL_EEW_DLV_HEAD_STR
Warehouse Request Item	EWM	/SCDL/INCL_EEW_DLV_ITEM_STR
Warehouse Task	EWM	/SCWM/INCL_EEW_S_ORDIM

Make sure that all changes have been activated successfully because the include structures are directly used on database level. When persisting custom fields on multiple layers the following BAdIs might help you to transfer the custom fields between the different layers:

Business Add-In	Method	Functionality	Call Sequence
/SCWM/EX_JIT_CUSTOM_FIELDS	CREATE_EEW	Transfer custom fields from JIT to warehouse request	Called between the JIT call creation and the warehouse request creation
/SCWM/EX_JIT_CUSTOM_FIELDS	UPDATE_EEW	Transfer custom fields from JIT to warehouse request	Called between JIT call update and the warehouse request update
/SCWM/EX_CORE_CR_INT_CR, /SCWM/EX_DLV_TOWHR_PTO_CREA	INSERT, GET_ADDITIONAL_WHR_DATA	Transfer custom fields from warehouse request to warehouse task	Called during warehouse task creation

The BAdI /SCWM/EX_JIT_CUSTOM_FIELDS already contains the JIT custom fields in its interface during JIT call creation and JIT call updates. To transfer custom fields from the warehouse request to its warehouse tasks you can read the warehouse request with its custom fields in BAdI /SCWM/EX_CORE_CR_INT_CR via the maintained reference document as described in section “Access Pattern” in the next chapter.

2.2 Display Custom Fields in the Warehouse Monitor

Besides adding custom fields to the database structures and to work with them in the backend (Transfer Custom Fields from JIT Call to Warehouse Task), it might also be useful to display custom field values in the warehouse monitor so users can work with these values. The warehouse monitor framework allows you to simply add new fields by extending the following structures with append structures:

Document	Structure
Warehouse Request Header for JIT Calls	/SCWM/INCL_EEW_S_JIT_HEADER
Warehouse Request Item for JIT Calls	/SCWM/INCL_EEW_S_JIT_ITEM

Once these structures are extended you can implement the methods FILL_EEW_JIT_HEADER_MON and FILL_EEW_JIT_ITEM_MON of BAdI /SCWM/EX_JIT_CUSTOM_FIELDS to fill the monitor structures accordingly. The custom field values can be retrieved from the importing parameter IT_JIT_CALL_DATA. You can also use the same methods to show values directly from the JIT call in the warehouse monitor which are not persisted in the warehouse request but just on the JIT call header / component group / component. Custom fields can only be shown in the list view but not in the form view using the above method. To extend also the form view with custom fields or use other features of the warehouse monitor you can refer to the Enhancement Guide for the Warehouse Management Monitor.

2.3 Execute Custom Logic at JIT Call Creation / Update / Cancellation

Depending on the time when you would like to make changes to an object and what you would like to change, there are different BAdIs in place which can be used to execute custom logic. There is one golden rule for every implementation: When you implement any BAdI make sure to not execute any commit or rollback statements to not interrupt the standard transaction handling. Otherwise this might lead to data inconsistencies or even data loss in the worst case. Also use the BAdIs only in a way they are intended to be used. This is described in the respective BAdI documentation which you can find in the system.

If your custom logic does not depend on the warehouse request, you can use the methods CREATE_EEW and UPDATE_EEW of BAdI /SCWM/EX_JIT_CUSTOM_FIELDS. These methods are called right after the JIT call creation, updates or cancellation even before the warehouse request was processed. This process is also described in the detail section for this BAdI in the last chapter (Business Add-In: Use Custom Fields in Warehouse Requests for JIT Processing).

If your custom logic does depend on the processed warehouse request or the warehouse request in general, there are better options. One major advantage of reusing the stock transfer in EWM is that you can use any existing enhancement possibility like for every other warehouse request type. There is for example the enhancement spot /SCWM/ES_DLV_DET which offers multiple BAdIs for different purposes. You can use e.g. BAdI /SCWM/EX_DLV_DET_AT_SAVE to execute custom code after all determinations and validations were run and before the warehouse request is saved. Changes done to the warehouse request in this stage are also saved afterwards. SAP Note 1414179 (especially the third attachment for delivery enhancements) provides more information on the general enhancement options for warehouse requests and does also explain the BAdI sequence in more detail as this is valid for all warehouse request documents.

If your custom logic does not depend on the warehouse request but on warehouse tasks you might be interested in the enhancement spots `/SCWM/ES_CORE_CO` or `/SCWM/ES_CORE_CR`. When using wave management, the enhancement spot `/SCWM/ES_WAVE` offers additional BAdIs. Due to the reuse of the stock transfer for the JIT-integration you can also use all existing BAdIs for the warehouse task processing.

3 BUILDING BLOCKS FOR JIT PROCESSING

The following aspects allow custom developments to interact with the JIT-EWM integration in a standardized way. When interacting with the system it is important to follow the described procedures and use the extensibility options as intended. Custom implementations can lead to data inconsistencies if they are implemented the wrong way or deviate from the intended way. The provided APIs are not released officially by SAP. The mentioned interfaces can change at any time without prior notice from SAP even if this is unlikely to happen. Use them at your own risk.

3.1 Access Pattern

It might be required to read business documents in custom developments if these are not provided in e.g. the BAdI interface. Therefor you can use the following interfaces as shown for reading access only. Please do not use any other methods of mentioned classes if you don't know exactly what you are doing as these are powerful tools.

3.1.1 Read Warehouse Request Details

A warehouse request (also known as delivery) can be accessed on different levels depending on the use case and phase of processing. It is important to use the correct access layers to not produce any data inconsistencies and write performant implementations. This section uses the Business Object Management (BOM) access layer for reading access. There are also other options like the service provider available which are not described in this guide. For further details please consult the third attachment for delivery enhancements in SAP Note 1414179.

To read the current state of a warehouse request including all buffered changes which were not written to the database yet, you can use the following example. Make sure to replace the `DOCID` and `ITEMID` with the correct values, e.g. by looping over the imported key tables:

```
DATA(lo_bom)  = /scdl/cl_bo_management=>get_instance( ).
DATA(lo_bo)   = lo_bom->get_bo_by_id( iv_docid = DOCID ).
DATA(lo_head) = lo_bo->get_header( ).
DATA(lo_item) = lo_bo->get_item( iv_itemid = ITEMID ).
```

Note: Do not save the references `lo_bo`, `lo_head` or `lo_item` in internal tables for later use, this may lead to data inconsistencies! If you want to access the same delivery at a later point in time you must request a new instance of these objects.

In most BAdI implementations it is useful to compare the current state with the database state to detect changes in a warehouse request. Therefor you can simply specify the requested object state:

```
DATA(lo_head_db) = lo_bo->get_header(
                   iv_objectstate = /scdl/if_dl_object_c=>sc_object_state_db ).
DATA(lo_item_db) = lo_bo->get_item( iv_itemid = ITEMID
                   iv_objectstate = /scdl/if_dl_object_c=>sc_object_state_db ).
```

The objects `lo_head` and `lo_item` provide further methods to read detailed information about the warehouse request. To check whether the warehouse request is relevant for JIT processing you can execute the following check. After the check has been executed you can simply access the referenced document for further processing:

```
READ TABLE lo_item->get_refdoc( iv_refdoccat = /scdl/if_dl_c=>sc_doccat_jit )
     INTO DATA(ls_refdoc) INDEX 1.
CHECK sy-subrc = 0.
DATA(lv_jit_call_num) = ls_refdoc-refdocno.
DATA(lv_comp_mat_num) = ls_refdoc-refitemno.
```

3.1.2 Read JIT Call Details

There are multiple ways to read JIT call details depending on the data you have available and the data you need to retrieve. These methods are provided in interface `/SCWM/IF_JIT`. To create an instance of this interface you can use the following example:

```
DATA(lo_jit) = CAST /scwm/if_jit( /scdl/cl_af_management=>get_instance( )
  ->get_service( /scwm/if_jit=>sc_me_as_service ) ).
```

The following methods usually read the warehouse request as well as the corresponding JIT data. The goal of these methods is to return the latest state of an object, so they read by default from the EWM buffer. However, the retrieved JIT data is usually not returned from the buffer and only in its database state available. The method `READ_JIT_CALL_BY_JIT_KEY` does therefore contain the parameter `IV_READ_ONLY_JIT_BUFFER` which will result in not returning the warehouse request data at all but with returning the latest JIT data from the transactional buffer. When using this option, you must specify the full JIT key in parameter `IT_JIT_KEY` (JIT call number + component number), supplying `IT_CALL_NUM` is not sufficient in this case. To retrieve the full JIT key, you can take a look at the previous access pattern (Read Warehouse Request Details).

Method	Key	Read Buffer	Warehouse Request	JIT Data
READ_JIT_CALL_BY_DOCID	Warehouse Request	NA	Buffered	Non-Buffered
READ_JIT_CALL_BY_TANUM	Warehouse Task	NA	Buffered	Non-Buffered
READ_JIT_CALL_BY_JIT_KEY	JIT Call (+ Component)		Buffered	Non-Buffered
READ_JIT_CALL_BY_JIT_KEY	JIT Call + Component	X	NA	Buffered

3.1.3 Read Warehouse Order, Warehouse Task, Stock etc.

In order to collect information relevant for a process, access to some of its constituents is required. An (incomplete) list of sources is:

- Warehouse tasks can be accessed using interface `/SCWM/IF_API_WHSE_TASK` and warehouse orders using interface `/SCWM/IF_API_WHSE_ORDER`. Both APIs can be instanced via `/SCWM/CL_API_FACTORY=>GET_SERVICE( IMPORTING EO_API = LO_API )`. If the provided functionality of this API is not enough you may also have a look the function modules in function group `/SCWM/WT_MAN`.
- The assignment of warehouse tasks to warehouse requests is provided by function module `/SCWM/QUERY_LINK_WHRWT`
- Stock (and handling units) can be read using interface `/SCWM/IF_API_STOCK` that can be instanced via `/SCWM/CL_API_FACTORY=>GET_SERVICE( IMPORTING EO_API = LO_API_STOCK )`
- PSA information can be read in interface `/SCWM/IF_PSA` and instanced via `LO_PSA ?= /SCDL/CL_AF_MANAGEMENT=>GET_INSTANCE( )->GET_SERVICE( /SCWM/IF_PSA=>SC_ME_AS_SERVICE )`

3.2 Warehouse Request Processing

3.2.1 Custom Fields

Business Add-In: Use Custom Fields in Warehouse Requests for JIT Processing

The BAdI `/SCWM/EX_JIT_CUSTOM_FIELDS` allows you to fill and update custom fields of stock transfer documents which are relevant for JIT processing. It can also be used to display custom field values in the warehouse monitor nodes “Stock Transfer for JIT Call” and “Stock Transfer Items for JIT Call”.

Method `CREATE_EEW` is called after a JIT call has been created in the buffer and before the actual warehouse request is created so there is no data persisted so far. The context is provided by the importing parameter `IT_JIT_CALL` which contains all JIT calls which will be created with this action. The custom fields can be set in changing parameter `CT_EEW` which is prefilled with the JIT key structure. Make sure to extend the include structures accordingly to use custom fields as described previously (Transfer Custom Fields from JIT Call to Warehouse Task). You can also use this method to execute custom logic which is not related to custom fields if you make sure to not trigger any commit or rollback operations.

Method `UPDATE_EEW` is like `CREATE_EEW` but is called in case of JIT call updates or cancellations before the corresponding warehouse requests are updated so the changes are not persisted at this point in time. The importing parameter `IT_JIT_CALL` does contain the JIT key attributes along with the changed attributes. Attributes which were not changed are not set in this structure. You can read them manually as described in the access pattern (Read JIT Call Details). Custom fields can be changed in the changing parameter `CT_EEW`

which is prefilled with the JIT key structure and the current custom field values of the corresponding warehouse requests.

Methods `FILL_EEW_JIT_HEADER_MON` and `FILL_EEW_JIT_ITEM_MON` are called before the corresponding warehouse monitor nodes are displayed. These methods allow you to add custom fields to the warehouse monitor. You can takeover custom fields from the importing parameter `IT_JIT_CALL_DATA` which is filled with the warehouse request and JIT call data, but you can also read or calculate any other value as required. These methods should not be used to change any persistent data. The process is also described with technical details in the integration scenario “Display Custom Fields in the Warehouse Monitor”.

3.2.2 Consolidation Group

Customizing: Define Consolidation Group for the Stock Transfer Process

You can use consolidation groups to determine which delivery items can or cannot be processed together, e.g. in warehouse order creation. The consolidation group is not limited to JIT Calls, but also available for other internal Stock Transfer documents. Prerequisites:

- You must have defined at least one internal number range interval for consolidation groups for your warehouse (*SPRO → Extended Warehouse Management → Master Data → Define Number Ranges → Define Number Range Intervals for Consolidation Group*)
- You must have defined an internal number range interval from the above customizing as the interval for “Internal Consolidation Group for Stock Transfer” in your warehouse (*SPRO → Extended Warehouse Management → Goods Issue Process → Assign Number Range Intervals to Consolidation Groups*)

In customizing activity *Define Consolidation Group for the Stock Transfer Process* you can select the criteria that will be used to determine the consolidation group based on the warehouse process type and priority.

If you select warehouse process type only, all stock transfer items with the same WPT will be assigned the same consolidation group. If you select priority only, all stock transfer items with the same priority will be assigned the same consolidation group. If you select both WPT and priority, all stock transfer items that have both the same priority and WPT will be assigned to the same consolidation group.

The consolidation groups can be displayed and manually created in transaction `/SCWM/DSGR_ST`. Furthermore, you can manually change the consolidation group of a stock transfer item in transaction `/SCWM/IM_ST`.

Note that if this customizing is not defined, the system may use the consolidation groups of the destination bin’s activity area within the warehouse task.

The consolidation group will be determined during stock transfer creation (including JIT call creation) and in case of updates of the WPT or priority. It can be overwritten by the below BAdI. It is possible to use the BAdI without maintaining the customizing mentioned in this section.

You can later make use of the consolidation group in warehouse order creation rules: Use creation category “Consolidation Group” in *SPRO → Extended Warehouse Management → Cross-Process Settings → Warehouse Order → Define Creation Rule for Warehouse Orders.*

Business Add-In: Define Consolidation Group for Stock Transfer

The BAdI `/SCWM/EX_CORE_CONS_ST` allows you to overwrite the consolidation group using custom logic. The BAdI will be called during the stock transfer item creation and when the WPT of the item or the priority of the stock transfer changes.

The BAdI receives the following importing parameters:

- `IS_PARAM` of type `/SCWM/S_DSTGRP_IMPORT_ST`, which contains the warehouse number, WPT, priority, document category, document ID and item ID. With the document category, document ID and item ID the JIT call details can be read using the access patterns described above.
- `IS_TDSTGRP` of type `/SCWM/TDSTGRP_ST` contains the customizing settings from *Define Consolidation Group for the Stock Transfer Process* for the current warehouse.
- `IS_TDSTGRPNO` of type `/SCWM/TDSTGRPNO` contains the number range interval for the stock transfer consolidation groups of the current warehouse.

With the changing parameter `CV_DSTGRP` the consolidation group can be overwritten.

3.2.3 Document and Item Type

Customizing: Define Document and Item Type Determination for JIT Processing

This customizing allows you to define document and item types for the stock transfer process in JIT processing. These settings are used by the system to determine the document types and item types automatically. The customizing is specific to the warehouse.

Example: You want to use the delivered document type `SJIT` (Stock Transfer: JIT Call) and the delivered item type `SDJT` (Stock Transfer Item: JIT Call) for your processes. You assign these values to your warehouse number. The document type `SJIT` and the item type `SDJT` are automatically determined during creation of the stock transfer for JIT Call.

Business Add-In: Define Document and Item Type at Warehouse Request Creation for JIT Processing

The BAdI `/SCWM/EX_JIT_MAP_DOCTYPE` allows you to overwrite the document type for a JIT call and the item type for each of the items of the JIT call. It is called once per JIT call which is created. In the single BAdI method `DETERMINE_DOCTYPE_ITEMTYPE` you have access to the JIT call number and header custom fields as well as for each item the item data including custom fields, if you have filled them in the previous BAdI `/SCWM/EX_JIT_CUSTOM_FIELDS` in method `CREATE_EEW`.

The document type can be changed by overwriting the value of the field `CV_DOCTYPE`. You can change each item type assigned to a component material number in `CT_ITEMTYPE`. If you clear these values or delete an entry from `CT_ITEMTYPE`, this will be ignored, and the original type defined in customizing will be used.

Furthermore, the result of the BAdI must comply with the following rules, otherwise the warehouse request creation will fail:

- The document type must be defined for the Stock Transfer Process (*SPRO → Extended Warehouse Management → Internal Warehouse Processes → Delivery Processing → Stock Transfers → Define Document Types for the Stock Transfer Process*)
- The item type must be defined for the Stock Transfer Process (*SPRO → Extended Warehouse Management → Internal Warehouse Processes → Delivery Processing → Stock Transfers → Define Item Types for the Stock Transfer Process*)
- The item type must be allowed in combination with its document type (*SPRO → Extended Warehouse Management → Internal Warehouse Processes → Delivery Processing → Stock Transfers → Define Allowed Item Types for the Stock Transfer Process*)

3.2.4 Warehouse Process Type Determination

Customizing: Determine Warehouse Process Type for JIT

This customizing allows you to define the settings for the automatic determination of the warehouse process type for JIT processing. You define warehouse process type determination rules specifically for each warehouse number. This customizing is also accessible using transaction code `/SCWM/TJIT_WPTD`.

When EWM creates a warehouse request for JIT processing, it automatically determines the warehouse process type based on the determination rules and an access sequence. The access sequence for the determination rules in this customizing activity can be defined in customizing activity *Access Sequence for JIT Warehouse Process Type Det* (transaction `/SCWM/TJIT_WPT_A`).

The warehouse process type determination that is defined in this customizing activity can be overruled by the warehouse process type determination rules maintained in the transaction *Determine Warehouse Process Type for JIT* (`/SCWM/JIT_WPTDET`). Using this transaction, the production supply area is also considered during the determination of the warehouse process type for JIT. The access sequence for the determination rules in this transaction can be defined in the transaction *Access Sequence for JIT Warehouse Process Type Det.* (`/SCWM/JIT_WPTD_A`).

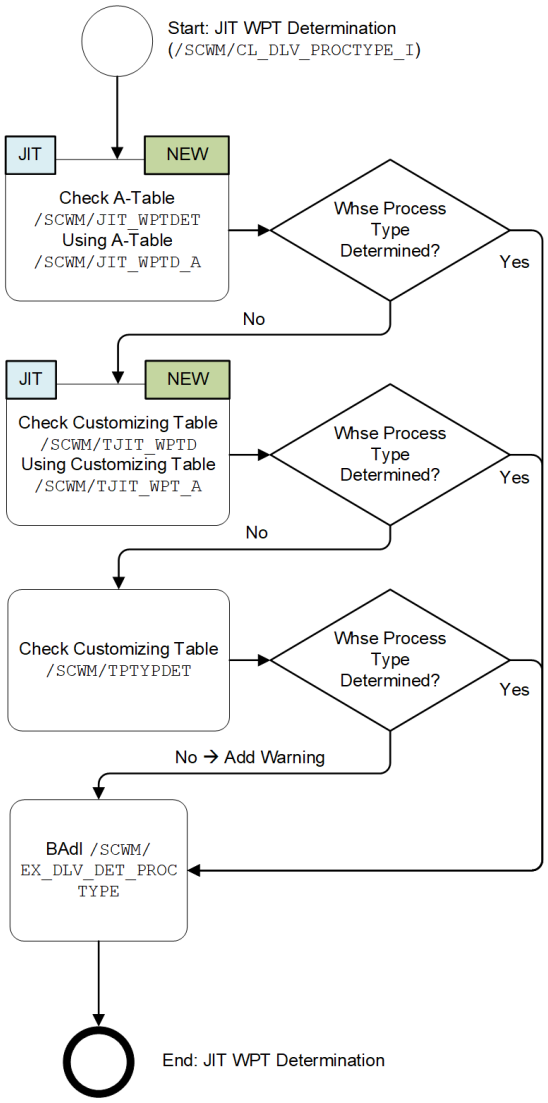

Figure 6: Warehouse Process Type Determination

Example:

In the customizing activity, you defined the following determination rule for the warehouse process type *REP1*:

- Warehouse Number: *0001*
- Replenishment Strategy: *REP1*
- Warehouse Process Type: *REP1*

You defined the following determination rule for the warehouse process type *REP2* for JIT in transaction `/SCWM/JIT_WPTDET`:

- Warehouse Number: *0001*
- Supply Area: *PSA-001*
- Replenishment Strategy: *REP1*
- Warehouse Process Type: *REP2*

Example 1: You create a JIT call for warehouse number *0001*, supply area *PSA-001*, and replenishment strategy *REP1*. The corresponding stock transfer item for JIT is created with warehouse process type *REP2*.

Example 2: You create a JIT call for warehouse number *0001*, supply area *PSA-002*, and replenishment strategy *REP1*. The corresponding stock transfer item for JIT is created with warehouse process type *REP1*.

Customizing: Access Sequence for JIT Warehouse Process Type Det.

This customizing allows you to define the access sequence for the warehouse process type determination for JIT in customizing activity *Determine Warehouse Process Type for JIT* (`/SCWM/TJIT_WPTD`) and is also accessible using transaction code `/SCWM/TJIT_WPT_A`.

Example:

During determination of the warehouse process type for JIT, the system starts with the first access sequence for warehouse process type determination and the warehouse number of the stock transfer for JIT call document:

Access sequence 1 - Access via *Replenishment Strategy*, *Action Control*

The system looks through the determination rules for the warehouse process type for JIT in order to find a match between the values of the attributes *Replenishment Strategy* and *Action Control* in the stock transfer for JIT call and in the determination rules. If there is a match, the system returns the warehouse process type for that determination rule. If there is no match the system continues with the next access sequence, and so on.

Access sequence 2 - Access via *Delivery Priority*, *Missing Material*

Access sequence 3 - Access via *Missing Material*

In summary you can use the following tables in transaction `SM30` to configure the warehouse process type determination for JIT processing. The rule priority is descending from left to right so the regular warehouse process type determination has the lowest priority and is only used when both other layers cannot determine a warehouse process type.

	With PSA for JIT	Without PSA for JIT	Regular WPT Det.
WPT Determination Rules	/SCWM/JIT_WPTDET	/SCWM/TJIT_WPTD	/SCWM/TPTYPDET
Access Sequence	/SCWM/JIT_WPTD_A	/SCWM/TJIT_WPT_A	-

3.3 Warehouse Order Processing

3.3.1 Wave Management

Business Add-In: Capacity Check for Waves

The BAdI `/SCWM/EX_WAVE_CAPA` allows you to influence and change the capacity check during wave assignment of delivery items. The BAdI is called for waves that have a wave capacity profile in the following situations:

- Wave creation
- Assignment or re-assignment of items to existing waves
- Automatic wave assignment (pre-check)
- Changes to wave attributes

In these situations, the EWM standard logic performs a capacity check for the target wave according to the customizing settings for the wave capacity profile. Afterwards, in your implementation of BAdI `/SCWM/EX_WAVE_CAPA`, you can change the result of the standard capacity check by implementing your own logic.

The context information is provided in importing parameters `IS_WAVEHDR` (target wave) and `IT_WAVEITM` (existing wave items in target wave). The items which are currently assigned to the target wave are provided in table `IT_WHRITM`. Parameter `IV_ASSIGN_MULT` is set only for the wave capacity check during automatic wave assignment.

By changing the `CV_ERROR` parameter, you can either allow further items to be assigned to the target wave, even though the capacity is exceeded, or you can reject the assignment, even though the capacity is sufficient. In addition, you can add log messages to the wave assignment log with parameter `CT_BAPIRET`.

As an example, you may want to add items for warehouse requests with high priority to target waves, which already reached the capacity limit given by the wave capacity profile. However, this should only happen when the wave assignment is done automatically. The capacity limit (e.g. 100 items per wave) should be allowed to be exceeded, but only up to a certain limit (e.g. 110 items maximum).

In your implementation, you would need to check for the `IV_ASSIGN_MULT` parameter being set. Further, the implementation only needs to run, when the capacity gets exceeded, so also check if `CV_ERROR` is set.

Then retrieve the warehouse request information by the document and item IDs in table `IT_WHRITM`. If the delivery has high priority, check if the maximum number of items (110) is reached. In `IS_WAVEHDR-NOITM`, you can see the number of items that are currently assigned to the wave. Then, if the items can be added, clear the `CV_ERROR` parameter. In addition, you may add a log message to `CT_BAPIRET` which explains about the change you made.

3.3.2 Warehouse Order / Task

In warehouse order processing, the following BAdIs can be facilitated for customer specific needs in the warehouse part of the JIT process:

- BAdI `/SCWM/EX_WHO_EEW_CHANGE` can be used to update custom fields of the warehouse order. These fields must be added to the enhancement structure `/SCWM/INCL_EEW_S_WHO` beforehand.
- BAdI `/SCWM/EX_WHO_DSTGRP` can be used to overwrite the consolidation group when creating warehouse orders. It is also a place where custom fields of the warehouse task can be filled or changed. These fields must be added to the enhancement structure `/SCWM/INCL_EEW_S_ORDIM` beforehand.
- BAdI `/SCWM/EX_CORE_CO_POST` is called when posting confirmed warehouse tasks. This BAdI allows you to access related warehouse tasks when posting. Example implementations are documented in the system in the 'Enhancement Spot Editor' (transaction `SE18`).
- BAdI `/SCWM/EX_CORE_LSC_LAYOUT` allows you to influence layout-oriented storage control. You can change the determined destination data or prevent the reassignment of the warehouse task to an intermediate point.
- BAdI `/SCWM/EX_CORE_PSC_PROCESS` allows you to influence process-oriented storage control. You can use this BAdI to change the determined destination data or to skip the intended process step.

3.3.3 Handling Unit

In warehouse order processing it might also be required to read and modify handling units. You can use especially the following BAdIs in the JIT process:

- BAdI `/SCWM/EX_CORE_CO_HU_SAVE` can be used to update customer-specific data in the handling units and is called when posting warehouse tasks.
- In transaction `/SCWM/RACK_GEN`, handling units and sub handling units are created representing a rack for distribution equipment. The BAdI `/SCWM/EX_CORE_HU_CREATE` allows flexibility in the validations for the HU hierarchy used.
- BAdI `/SCWM/EX_CORE_HU_EXT_HUID` allows you to generate external handling unit numbers from a number range with prefixes.

www.sap.com/contactsap

THE BEST RUN SAP®