

Reading Sample

This sample chapter discusses how to trigger events in SAP S/4HANA and gives details about event creators, receivers, and event linkage. It walks through how to implement a start condition in the workflow as well as how to use terminating events and instance linkage. The chapter concludes by providing information about check function modules and receiver function modules.

-  **“Defining and Triggering Events”**
-  **Contents**
-  **Index**
-  **The Authors**

Dutta, Ghosh, Goon, Jana, Mukherjee, Rao, Rao, Sane, Veshala, Viswanathan

Workflow for SAP S/4HANA

686 pages | 11/2023 | \$89.95 | ISBN 978-1-4932-2428-9

 www.sap-press.com/5697

Chapter 5

Defining and Triggering Events

This chapter talks about different triggering mechanisms for workflow events. We look at each event-triggering technique in detail with the help of some practical examples. Then, we delve deeper into event creators, receivers, event linkage, and how to attach start conditions to event linkages. We also look at terminating events and instance linkages and finally close the chapter by explaining how to use check function modules and receiver function modules in event triggering.

An event describes the change in status of an object in the SAP system, for example, when a sales order is created or changed or when a purchase order is released. In the context of this book, we'll be referring to workflow events only.

To use an event in a workflow, it must first be defined in a business object type in the business object repository (BOR) or in an ABAP object-oriented (OO) class (which implements the `IF_WORKFLOW` interface). The event carries information for the business object (or ABAP class object) instance along with optional parameters (if relevant), which may be used by the receiver (workflow, single-step task, or function module) for further processing steps.

You need the following details to trigger an event in the SAP system:

- Business object type or ABAP OO class name in which the event is defined
- Event name
- Key field(s) of the BOR or class required to create the object instance
- Event creator name (by default, this is the current system user)
- Optional event parameters if defined

When you create an event in the SAP system, an instance of the BOR object type or ABAP OO class is automatically created via the key field information passed to the event. This object instance is passed to the receiver application. If the receiver is a workflow template or a single-step task, then the event object instance is passed via event container element `_EVT_OBJECT` to the workflow container or task container. Additional parameters (if any) on the event must be explicitly bound to the receiver container via the event container element with the same name as the parameter.

Events must be triggered explicitly by the SAP application. There are various techniques available in SAP to trigger events, some of which involve configuration steps

and others only pure ABAP code. Whenever an event is triggered in SAP, you can see it from the event trace Transaction SWEL (provided the event trace is switched **ON** via Transaction SWELS). We'll discuss the event trace further in Chapter 8.

In the next section, we'll discuss the various techniques available in SAP to trigger events, followed by a section on the concepts of event creators, receivers, and event linkage. You'll learn how to implement a start condition in the workflow so that you can filter and trigger your workflow or single-step task only under the exact conditions you prefer. We'll introduce you to the concept of terminating events and instance linkage in the next section. Finally, we'll talk about check function modules and receiver function modules that may be configured against an event linkage.

5.1 Event Triggering Techniques

Following are the common techniques used for triggering events in SAP (discussed in detail in this section):

- Event trigger via change documents: Pure configuration, no ABAP coding
- Event trigger via status management: Pure configuration, no ABAP coding
- Event trigger via message control: Pure configuration, no ABAP coding
- Event trigger via ABAP code in a user exit, business add-in (BAdI), or other ABAP programs

5.1.1 Event Trigger via Change Documents

Most standard SAP applications (master data or transaction data) use the concept of change documents to track the creation, updates, and deletion of the entire object or some attributes within the object. Change document objects can be viewed with Transaction SCDO. Figure 5.1 shows an example of the change document object VERKBELEG, which is used to track changes in sales document transactions.

The change document object lists the underlying tables that are tracked for create/change/delete actions on the corresponding application transactions or programmatically via Business Application Programming Interfaces (BAPIs).

The first step to configure an event trigger via change documents is to identify the correct change document object and the corresponding table (if you're interested in specific field updates).

Next, you go to Transaction SWED to verify whether the relevant change document object tracks the change action that is relevant for your requirement. For example, for change document object VERKBELEG, you can see in Figure 5.2 that all three actions—create, change, and delete—are tracked.

Table	Int. Table	Delete Doc...	Log Initial Values	Insert Doc...	Log Initial Values	Referencing Table	Old Field String
VBAK	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		YVBAK
VBAP	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	WVBAP	
VBEP	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	WVBEP	
VBKD	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
VBLB	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
VBPA	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	WVBPA	
VBPA2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
VBPA3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
VBSN	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	WVBSN	
VEDA	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Figure 5.1 A Change Document Object in SAP

Change Document Object	Leading table in change document	Change document key with structure	Action: Create	Action: Change	Action: Delete	Action: Create Instance
QUOTEN	URC	MARC	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
QVDM	QVDM	QVDM	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RECHNUNG	RSEG	RSEG	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
RELOBJECTS	RODIR	RODIR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
RKAUFTRAG	COAS	AUFK	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SACH	SKA1	SKA1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SAMS	SKM1	SKM1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SD_CONTACT	VBKA	VBKA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
SLEI_CD_TST	SLEI_CD_TST	SLEI_CD_TST	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
STUE	STZU	STZU	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
SWE_CD_TST	SWE_CD_TST	SWE_CD_TST	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
TRIAL	RMXTT_TRIAL_HD	RMXTT_TRIAL_HD	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
VASMD	ASMD	ASMD	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
VERKBELEG	VBAK	VBAK	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
VINET	INET	INET	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
VWGATTUNG	VWPANLA	VWPANLA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
WBASISWG	T023	T023	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
WRF_MATGRP_OBJ	WRF_MATGRP_HIER	WRF_MATGRP_HIER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 5.2 Viewing the Change Actions Maintained against a Change Document Object in Transaction SWED

Note

This is just a verification step as most common standard SAP change document objects will track all the actions. However, if you've created your own custom application with a custom change document object, then this step becomes a prerequisite before you can proceed to the next step to configure your workflow events against the change document object.

Expert Tip

In Transaction SWED, there is an option to configure a function module against a change document object. This function module is required when the key field of the primary table in the change document object differs from the key field of the business object used to trigger the event in Transaction SWEC. For example, for the change document object PROJ, primary table PROJ has a key field of PSPNR (internal project number). However, the business object for project BUS2001 has two key fields, external project number (PSPID) and internal project number (PSPNR). So, a conversion needs to happen from the change document key to the business object key via the Transaction SWED function module. Refer to function module CJPN_BUS_2001_2054_OBJECT_KEY for an example of the coding involved.

The next step is to configure your workflow event for the change document object and also configure the specific change action in Transaction SWEC. Once you click on the **New Entries** button, enter the change document object (which you selected from Transaction SCDO) and the business object type and event which you want to trigger. Select the **On Create**, **On Change**, or **On Delete** radio button per your requirement. Save your entries.

You can also maintain field restrictions to track changes in specific fields of the tables listed under the change document object. This additional field restriction is relevant for change actions only. Here you simply need to select the field you want to track and enter a change condition (optional) using the old value and new value variables of this field. Figure 5.3 shows the change document configuration of a sample business object/event against the object VERKBELEG.

Change Doc. Object	ObjectCateg...	Business Obj. Type	Event	On Create	On Change	On De...
PPESGENN	BO BOR ..	BUS1197001	CHANGE	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
QINF	BO BOR ..	BUS2101		<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
RKAUFTRAG	BO BOR ..	RG_BUS2075	CREATED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SACH	BO BOR ..	BUS3006		<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
SD_CONTACT	BO BOR ..	BUS1037	CHANGED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SD_CONTACT	BO BOR ..	BUS1037	DELETED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SD_CONTACT	BO BOR ..	BUS1037	CREATED	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
SLEI_CD_TST	CL ABAP..	CL_SLEI_TST_UNIT_RUN_FACADE	EVT_CHDOC_CREATED	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
SLEI_CD_TST	CL ABAP..	CL_SLEI_TST_UNIT_RUN_FACADE	EVT_CHDOC_CHANGED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SLEI_CD_TST	CL ABAP..	CL_SLEI_TST_UNIT_RUN_FACADE	EVT_CHDOC_DELETED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SWE_CD_TST	BO BOR ..	SWE_CD_TST	CHANGED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SWE_CD_TST	BO BOR ..	SWE_CD_TST	CREATED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
SWE_CD_TST	BO BOR ..	SWE_CD_TST	DELETED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
VERKBELEG	BO BOR ..	FREBUS2032	CREATEDFRE	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
VERKBELEG	BO BOR ..	FREBUS2032	CHANGEDFRE	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
WRF_MATGRP_OBJ	BO BOR ..	BUS1235	CREATED	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
WRF_MATGRP_OBJ	BO BOR ..	BUS1235	CHANGED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 5.3 Configuring BOR/Class Events against a Change Document Object in Transaction SWEC

Once you've linked your business object (or ABAP class) and event to the change document object/action, then your event triggering configuration is complete. Now you may proceed to test your application by creating/updating/deleting an object and checking if the event was triggered from the event trace (Transaction SWEL).

Let's now look at an example in which an event is triggered when a credit memo request is blocked for billing in SAP. We've identified the standard business object for the credit memo request application as BUS2094 and verified that no standard event is available that is triggered under the same conditions. Here, we've created a subtype of BOR BUS2094, delegated the supertype to the subtype, and defined our custom event ZWEBillingBlockCreated in the subtype (refer to Chapter 3 for details on subtype creation and delegation). Figure 5.4 shows the view of the subtype after the addition of the custom event.

Event	Description
ZBUS2094.assigned	WF_COMMIT called
ZBUS2094.CREATED	generated
ZBUS2094.CHANGED	Changed
ZBUS2094.DELETED	Deleted
ZBUS2094.INDIRECT_STATUS_CHANGED	Indirect Status Changed
ZBUS2094.ZWEBillingBlockCreated	Billing Block Created

Figure 5.4 Subtype with Custom Event Added from Transaction SWO1

Now follow these steps to proceed with the event configuration:

1. Identify the table and field name for which the new custom event should be raised, and identify the specific value that needs to be checked (if relevant). In this case, the field for the billing block on credit memo request transaction is VBAK-FAKSK.
2. Identify the change document object, which includes table VBAK from Transaction SCDO. In this example, the object name is VERKBELEG.
3. Proceed to Transaction SWEC to enter the configuration steps as mentioned in the following steps. (Note: Because you're dealing with a standard transaction and a standard change document object, you don't need to configure the change actions registered in Transaction SWED.)
4. Under the **Events for Change Document** node, click on the **New Entries** button, and enter the details as shown in Figure 5.5: enter **Change Doc. Object** (change document object) as "VERKBELEG", choose **BO BOR Object Type** from the **Object Category** dropdown, enter **Object Type** as "BUS2094", and enter **Event** as "ZWEBILLINGBLOCK-CREATED".

5. Choose the **On Change** radio button under **Trigger Event**.

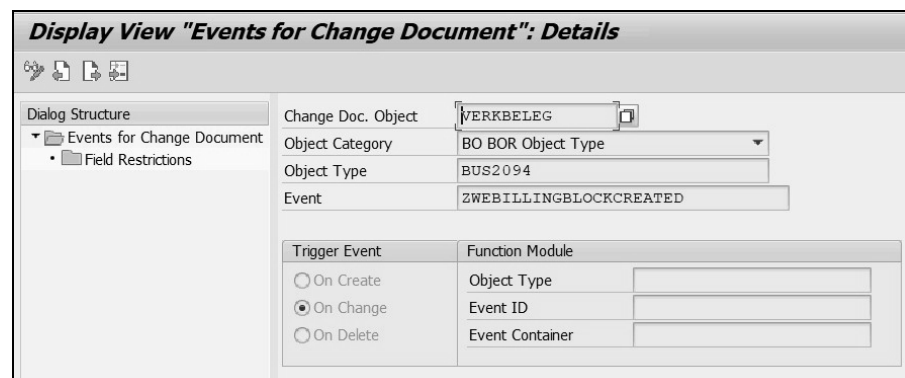


Figure 5.5 Custom Event Configured for the Change Document Object in Transaction SWEC

6. Navigate to the **Field Restrictions** child node, and enter the field change details for triggering the event. Here you have two options to enter the field restriction:

- If your condition is a simple comparison of one or more values, then enter it directly as shown in Figure 5.6 with the **Table** name, **Field Name**, **Old Value**, and **New Value** fields.

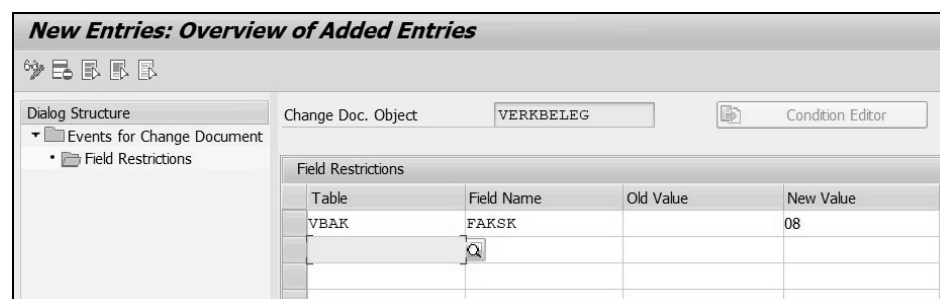


Figure 5.6 Maintaining Field Restrictions for the Event Trigger via Change Documents in Transaction SWEC

- If your condition involves a more complex expression, then use the **Condition Editor** button to navigate to the screen shown in Figure 5.7, and enter the condition as required. You can make use of a wide range of expression operators, such as **>**, **>=**, **<**, **<=**, **EX**, **NX**, **CE**, **NE**, and so on, along with logical operators, such as **And**, **Or**, and **Not**, to frame your condition. You can also use parentheses in your expression as appropriate and use multiple table fields together in a single expression.

7. Go back and save your changes. You will be prompted for a transport request.

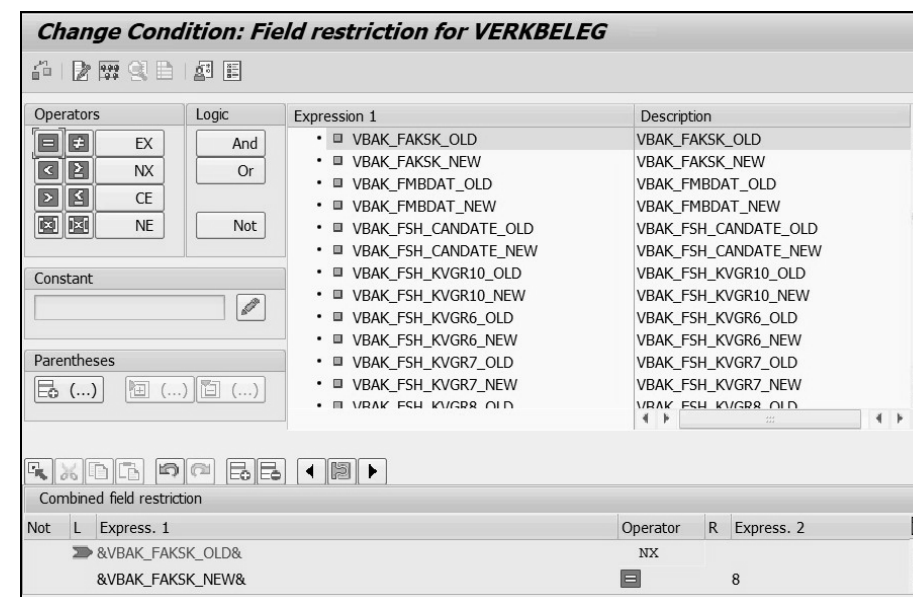


Figure 5.7 Maintaining Field Restrictions with the Condition Editor in Transaction SWEC

Next, it's time to test your changes. As shown in Figure 5.8, create/change a credit memo request in SAP S/4HANA from Transaction VA01/VA02, and enter the **Billing Block** as "08" before saving the document. Write down the document number.

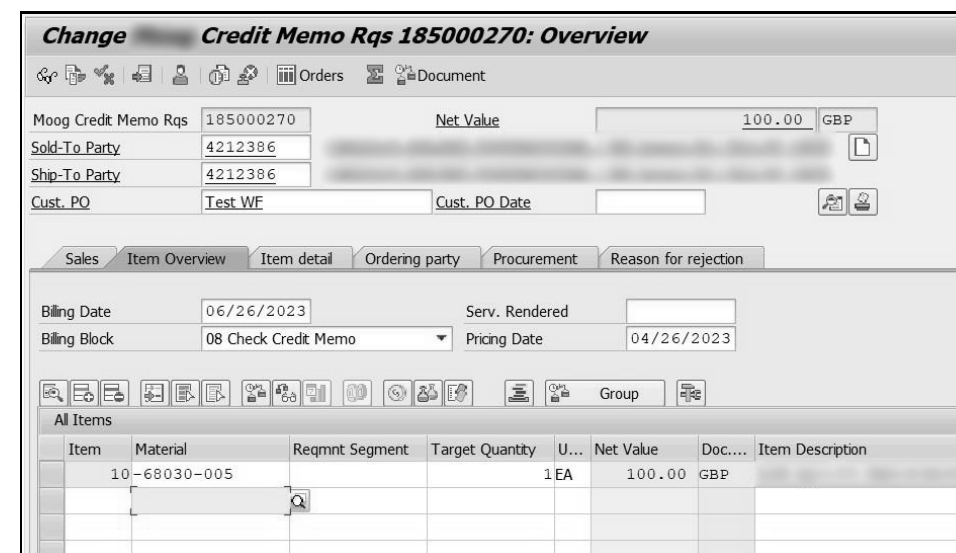


Figure 5.8 Changing a Credit Memo Request in SAP S/4HANA with the Billing Block Set

Now, check the event trace from Transaction SWEL (make sure the event trace is switched **ON** via Transaction SWELS first), as shown in Figure 5.9. Enter your business

object type (“BUS2094”) in the ‘Creator’ object type field to filter out the trace entries, and make sure the date and time are selected appropriately in the **Created From Date/Time** field.

Display Event Trace

Event Data

Event ID [] to []

'Creator' object type BUS2094 to []

'Creator' object instance [] to []

Event [] to []

Program creating event [] to []

Creator (User) [] to []

Created From Date/Time 06/20/2023 / 12:26:15

Created Until Date/Time [] / 00:00:00

Receiver Data

Receiver Type [] to []

Receiver Instance [] to []

Receiver FM [] to []

Receiver Type FM [] to []

Check FM [] to []

Figure 5.9 Checking the Event Trace from Transaction SWEL

Then execute and check the trace as shown in Figure 5.10. If the entry exists with your business object type and event, the object key matches the document number that you saved (in this case, the credit memo request in Transaction VAO1/VAO2) on the details screen (see Figure 5.11), then your test result is successful.

Event trace reveals that your custom event has been successfully triggered (see Figure 5.10), and the credit memo request number in the object key (see Figure 5.11) matches the document you’ve posted.

Display Event Trace

Delete Event Trace

Object Type	Event	Current Date	Time	Name of Receiver Type	Infor...	Handler/Action
BUS2094	ZWEBILLINGBLOCKCREATED	05/05/2023	13:18:12			No receiver entered

Figure 5.10 Event Trace Showing Event Has Triggered

On clicking the **Details** button (or double-clicking on the event trace line), you can see the instance information shown in Figure 5.11 with object type, object key, event, and event creator.

Display Event Trace

Event Container

Event Data

Event Instance ID 001DD8032C781EEDBAEE264A479CCA0A

Object Type BUS2094

Object Key 0185000270

Event ZWEBILLINGBLOCKCREATED

Event Creator US AMUKHERJ Arindam Mukherjee

Creation Time 05/05/2023 13:18:12 UTC-5

Receiver Data

Receiver Type []

Object Key []

Receiver FM []

RFC Destination []

Check FM []

Receiver Type FM []

Figure 5.11 Event Trace Details Showing Event Creator Data

Note

The **Receiver Data** section is empty here as we haven’t yet configured any receiver (workflow template or single-step task) for this event. This section will be dealt with in detail in Section 5.2 when we talk about receivers and event linkage.

5.1.2 Event Trigger via Status Management

Sometimes, you may want to trigger your workflow event via status management. Status management with respect to a workflow includes both system statuses and user statuses. System statuses are specific standard processing statuses of a document. For example, when you release a production order in SAP, the system status I0002 is set for that order. On the other hand, user statuses are usually customer defined and are included under a status profile, which is either assigned to the order type (header level) or to an item category (item level). Detailed configuration steps required to create a status profile and assignment to an SAP transaction is beyond the scope of this book. However, note the following points when dealing with status management related to workflow events:

- Both system statuses (IXXXX) and user statuses (EXXXX) may be used to trigger workflow events.
- An event may be triggered when a specific system/user status becomes active or when an **Active** status is set to **Inactive**.

- System or user status entries for a particular document are logged in table JEST. The key field of this table is object number (OBJNR), which can be retrieved from the application table for which the status is logged, for example, table AUFK for production order header, table VBAK for sales order header, table VBAP for sales order item, and so on. Other important tables related to status include table JSTO (status object information) and table JCDS (change documents for status).
- Status profile may be maintained from Transaction BSO2.
- Workflow events may be configured for system statuses in Transaction BSVX. Figure 5.12 shows an example. These entries are usually provided by standard SAP. Here, you maintain the business object type and the event name. Under the **Status restrictions** node (see Figure 5.13), you can maintain one or more system statuses for which the event will be triggered.

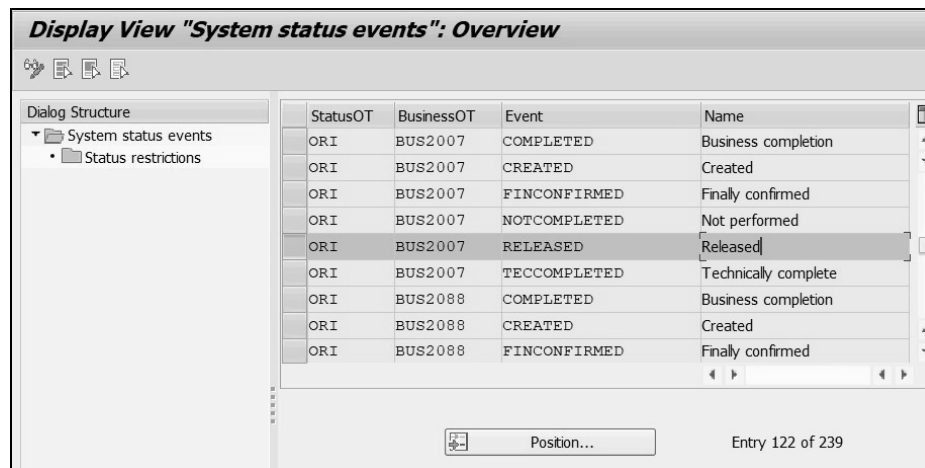


Figure 5.12 Event Configuration with System Status in Transaction BSVX

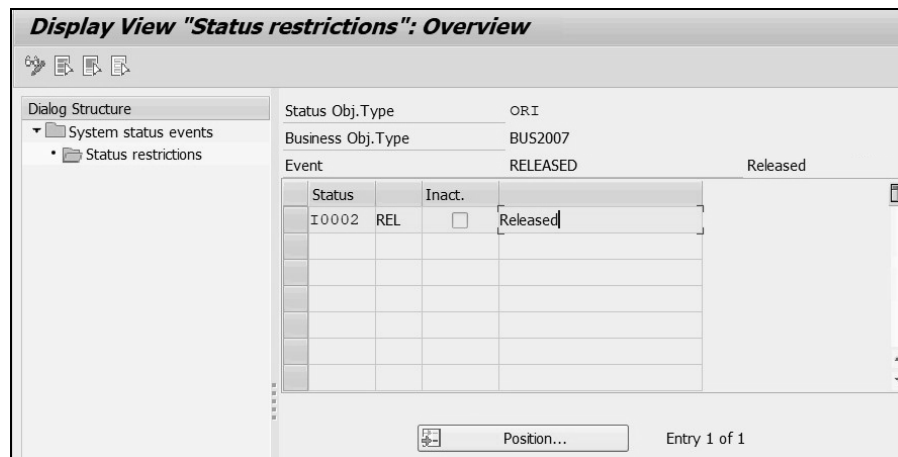


Figure 5.13 System Status Restrictions for Event Configuration

- Workflow events may be configured for system/user statuses in Transaction BSVZ (see Figure 5.14 for an example). These entries are always maintained by the customer and not provided by SAP. Here, we've configured the event ZWECHANGED of business object BUS2007 and a custom status profile for status object type ORI (maintenance order).

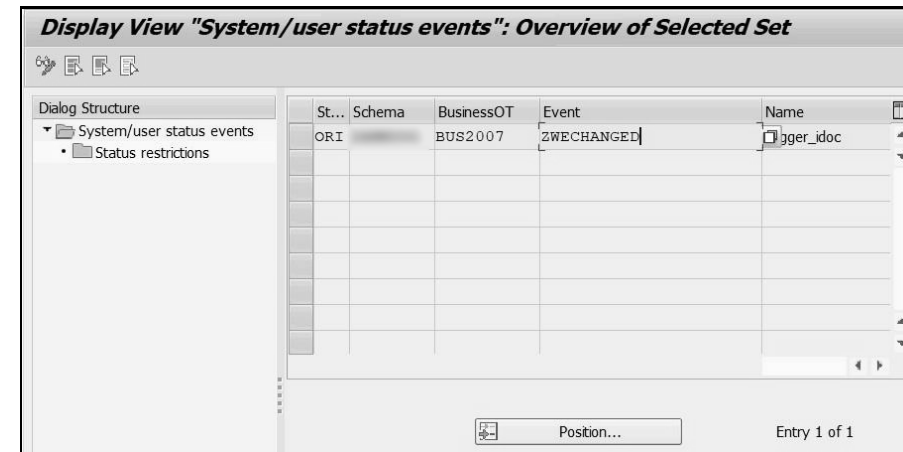


Figure 5.14 Event Configuration with User Status Profile in Transaction BSVZ

- Under **Status restrictions**, shown in Figure 5.15, we've maintained the user status QTCR (E0003) with the **Inact.** (inactive) checkbox unselected. This means that the event ZWECHANGED of business object type BUS2007 will be triggered when the user status QTCR is set for a particular service order at the header level.

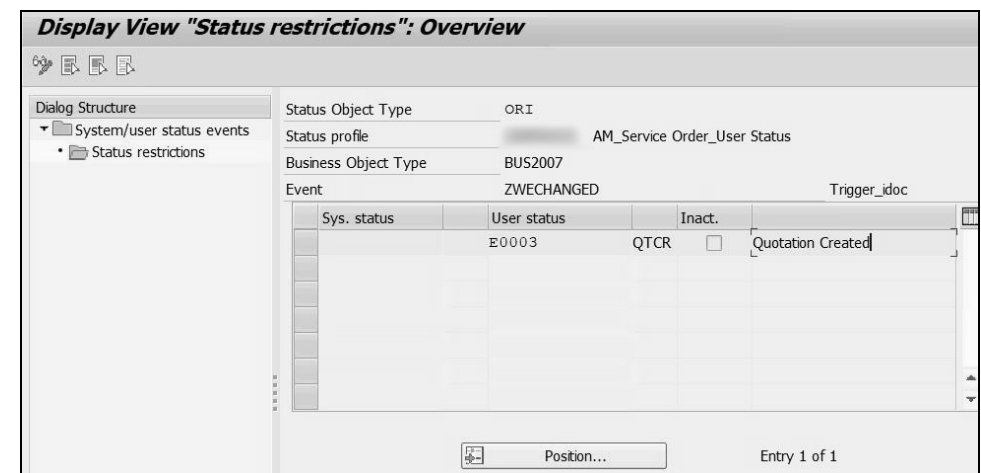


Figure 5.15 User Status Restrictions for Event Configuration

5.1.3 Event Trigger via Message Control

This is one of the less commonly used techniques for triggering events via configuration. In this technique, you use an output type based on traditional table `NAST` to trigger the event. Steps to configure the output type in Transaction NACE are very much like print, email or Application Link Enabling (ALE) outputs. The only difference is the transmission medium, which in this case, should be **9 Events (SAP Business Workflow)**. Detailed configuration steps for output type configuration in Transaction NACE is beyond the scope of this book, but Figure 5.16 lists the important steps involved in this configuration.

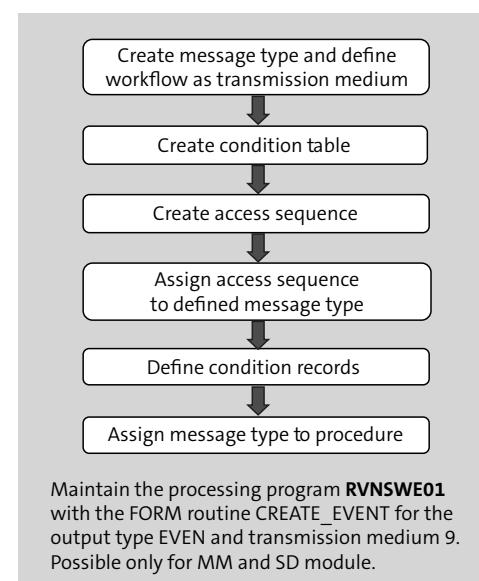


Figure 5.16 Steps for Event Configuration via Table NAST Message Control Configuration

Following are some important points about the restrictions applicable for this event triggering technique:

- This technique only applies to those standard applications that can use output types based on table `NAST`. With the introduction of new Business Rules Framework plus (BRFplus) output types in SAP S/4HANA, the list of standard applications may be further reduced because the output determination technique doesn't support workflow channels in SAP S/4HANA yet.
- Message control can only raise business object type events. Class-based events aren't supported with this technique.
- The business object type and event name are maintained in condition records, so business users have the flexibility of activating and deactivating the event trigger separately in each system. They also have the flexibility to raise different events according to different condition criteria.

Let's now look at an example of triggering a custom event in an inbound delivery transaction via message control. For this example, you'll trigger an event called *quality inspection request* after posting the goods receipt against an inbound delivery. This event may further be used to trigger a workflow that would manage the quality inspection approval of the goods received from a vendor. The standard business object for inbound delivery transaction has been identified as `BUS2015`. We've created subtype `ZBUS2015` of the standard `BOR` type, delegated supertype `BUS2015` to the subtype, and defined custom event `ZWEQualityInspection` in the subtype. (Refer to Chapter 3 for details on subtype creation and delegation.) Figure 5.17 shows the view of subtype `ZBUS2015` after adding the custom event from Transaction `SWO1`.

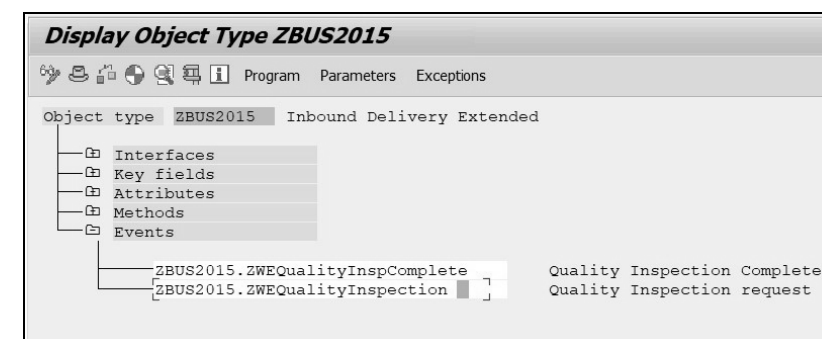


Figure 5.17 BOR Subtype Definition with Custom Events

Next, you'll see the steps to configure your custom output type with transmission medium **9 Events (SAP Business Workflow)** and use it for triggering your quality inspection event after goods receipt posting for the inbound delivery. Note that although Transaction NACE output type configuration details aren't in the scope of this book, we've listed some high-level steps, mostly related to workflow event configuration for the sake of understanding the concepts, as follows:

1. Define your custom output type in the **Application E1** (inbound delivery) in Transaction NACE. Once you select the application, click on the **New Entries** button, and enter the output type name and short description. Figure 5.18 shows the custom output type definition in Transaction NACE.
2. Click on **Processing routines** in the **Dialog Structure**, and then maintain Transmission medium as **9 Events (SAP Business Workflow)**. Maintain the **Program** as "RVNSWE01" and **Form Routine** as "CREATE_EVENT", as shown in Figure 5.19.
3. Maintain the access sequence for the output type in Transaction NACE in the **Access Sequences** screen per your business requirement. In this case, we've assigned an access sequence based on shipping point/delivery type. This step depends on your business requirement. You can use a standard access sequence if it meets your requirement or create a custom one according to your needs. We won't go into

details about access sequence and access creation here as this is an elaborate topic by itself and not related to workflows.

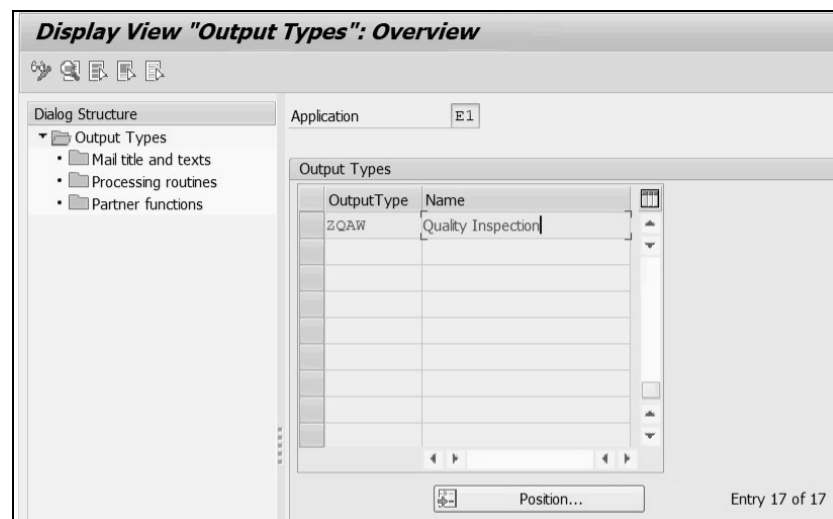


Figure 5.18 Custom Output Type Definition in Application E1 in Transaction NACE

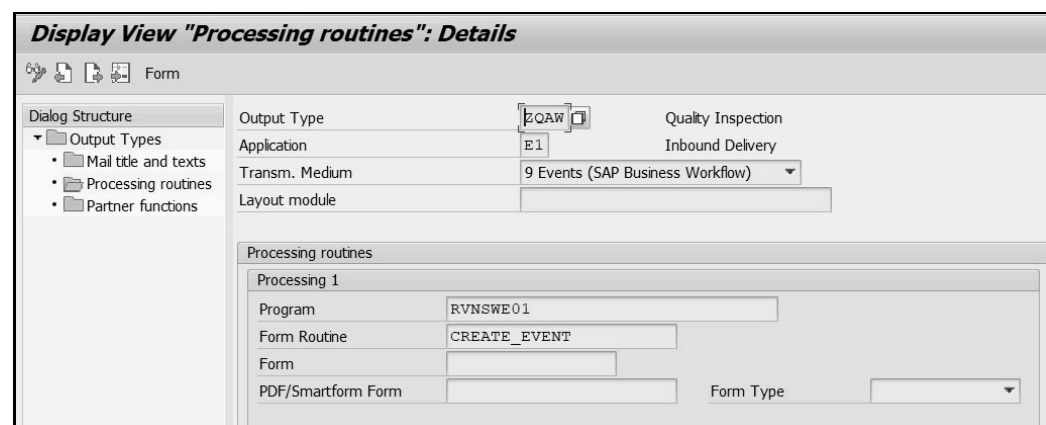


Figure 5.19 Maintenance of Processing Routine for Transmission Medium 9 in Transaction NACE

- Assign the output type to an output determination procedure by clicking on the **Procedures** button in Transaction NACE and then adding the output type under a suitable procedure (see Figure 5.20) Make sure this procedure is assigned to the relevant delivery type(s) in Transaction SPRO Customizing under **Logistics Execution • Shipping • Deliveries • Define Delivery Types**. Select your delivery type, and click the **Details** button. Then, enter the procedure name in the **OutputDet.Proc.** field. In

Figure 5.20, we've also maintained a requirement routine in the output procedure against our custom output type to restrict the output trigger after post goods receipt only.

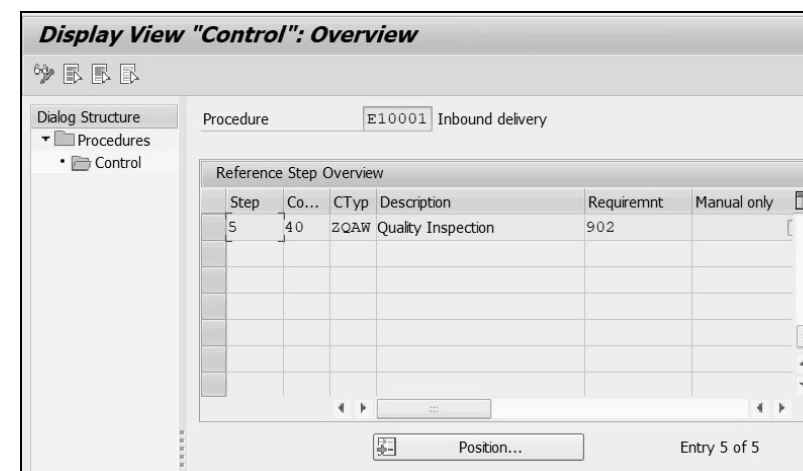


Figure 5.20 Addition of Custom Output Type to the Output Procedure along with Requirement Routine

- Maintain the condition record for the output type in your test client, as shown in Figure 5.21. You can navigate to this screen by clicking on the condition records button in Transaction NACE; then choose your output type and enter the appropriate criteria per your access sequence definition. In the condition record, you must enter the transmission medium as "9".

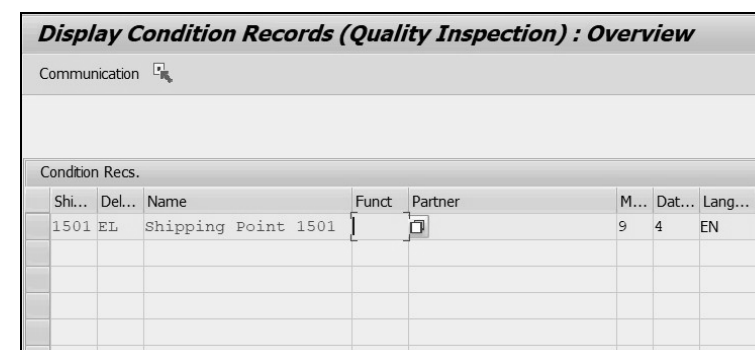


Figure 5.21 Condition Record Maintenance for Custom Output Type

- Maintain the object type and event under the **Communication** tab, as shown in Figure 5.22. To navigate here, click on the **Communication** button on the application toolbar of the condition record screen shown previously in Figure 5.21.

Display Condition Records (Quality Inspection) : Communication

Variable Key

Shipping Point	Delivery Type	Description
1501	EL	Shipping Point 1501

Object type: BUS2015

Event: ZWEQUALITYINSPECTION

Figure 5.22 Enter the Communication Data for the Condition Record with BOR Type and Event

Now it's time to test your changes. Create an inbound delivery based on a purchase order item in your test system. Perform post goods receipt for the delivery (via inventory management or embedded extended warehouse management (EWM), depending on your storage location settings). After post goods receipt, the output type ZQAW will be automatically triggered, which will fire our custom BOR event. Figure 5.23 shows the custom output type ZQAW triggered from inbound delivery after post goods receipt. The transmission medium shows **9 Events (SAP Business Workflow)**, as required by our scenario.

Inbound Delivery: Output

Communication method Processing log Further data

Delivery: 0180001175

Output

Sta...	Outpu...	Description	Medium	Fun...	Partner
	ZQAW	Quality Inspection	9 Events (SAP Business Workflow)		

Figure 5.23 Workflow Event Output Generated after Post Goods Receipt of Inbound Delivery

You can check the event trace (Transaction SWEL) for tracking the event (same steps as detailed in the test scenario for the event trigger with change documents). In this case, you filter the trace with your object type BUS2015. Figure 5.24 shows the event trace with our custom event ZWEQUALITYINSPECTION triggered for object type BUS2015.

Display Event Trace

Delete Event Trace

Object Type	Event	Current Date	Time	Name of Receiver Type	Infor...	Handler/Action
BUS2015	ZWEQUALITYINSPECTION	05/15/2023	02:22:31			No receiver entered

Figure 5.24 Event Trace Showing Event Triggered for the Output Type

On clicking the **Details** button (or double-clicking on the event trace line), you can see the instance information, as shown in Figure 5.25, with **Object Type**, **Object Key**, **Event** name, and **Event Creator** details. The **Object Key** number is the key field of the business object type BUS2015, that is, the inbound delivery number.

Display Event Trace

Event Container

Event Data

Event Instance ID: 001DD8032C781EDDBCDE2823A0452853

Object Type: BUS2015

Object Key: 0180001175

Event: ZWEQUALITYINSPECTION

Event Creator: US LCUSER LCUSER LCUSER

Creation Time: 05/15/2023 02:22:31 UTC-5

Receiver Data

Receiver Type

Object Key

Receiver FM

RFC Destination

Check FM

Receiver Type FM

Figure 5.25 Event Trace Details Showing Event Creator Instance Information

Note

The **Receiver Data** section of the screen is empty here because we haven't yet configured any receiver (workflow template or single-step task) for this event. This section will be discussed in more detail in Section 5.2 when we talk about receivers and event linkage.

5.1.4 Event Trigger via ABAP Code in User Exits, Business Add-Ins, and Custom Programs

Finally, you always have the option to trigger a workflow event via ABAP code. For this, you need a BAdI, user exit, function exit, or some other form of enhancement in a standard SAP application program. If you've developed your own custom program, then you can call the workflow event API from any appropriate point in that program (preferably a save or update module). An explicit COMMIT WORK is required to register the event in the system, which may be part of your custom program, or for an enhancement, it should be part of the standard Transaction LUW processing.

Some of the common function modules or APIs that can be used to raise a workflow event are as follows:

- Function module SWE_EVENT_CREATE (used for BOR events)
- Workflow API SAP_WAPI_CREATE_EVENT (used for BOR events)
- Update function module SWE_EVENT_CREATE_IN_UPD_TASK (used for BOR events)
- Method RAISE of class CL_SWF_EVT_EVENT (used for both BOR events and ABAP class events)
- Workflow API SAP_WAPI_CREATE_EVENT_EXTENDED (used for both BOR events and ABAP class events).

All these APIs perform the same thing in the SAP system, that is, raise a workflow event with required instance data and additional parameters. The event API interface parameters are as follows:

- **Object type**
This is the name of the BOR type, for example, BUS2032.
- **Object key**
This is the concatenated key of the business object instance. For example, for BUS2032, it's the sales order number, whereas for BUS2009, it's the purchase requisition number and the item number.
- **Event**
This is the name of the BOR event that you want to trigger. This event must be defined as part of a BOR definition.
- **User**
The user ID should create the event (available as _EVT_CREATOR in the event container). By default, this is the system user calling the workflow API.
- **Event container**
This container may be used to pass any event parameters. For example, if you have sales orders being created via multiple interfaces and directly within SAP as well, then you may want to add a parameter to identify the system or the application that triggered the sales order. This will be an event parameter and can be passed via the event container table parameter of the workflow API.
- **Event ID and return code**
The output of the workflow API is the event ID number (internal) and a return code indicating success or error. In case of error, additional parameter(s) for messages provide the details.

For class-based events, the method parameter requires the business class name and event in place of the BOR type name and event. The rest of the parameters are like the event function module.

Now let's look at a couple of source code examples to learn how to trigger workflow events via APIs. In Listing 5.1, we'll raise event ZWEQualityInspComplete for BOR type BUS2015, which will be triggered at the end of the quality inspection review process discussed in our case study in Section 5.1.3. Event ZWEQualityInspComplete has event parameter ZWCDecisionCode, which will pass either APPROVED or REJECTED values through the event. In Listing 5.2, we'll raise an event for a custom workflow business class to release a sales and distribution invoice to accounting.

Note

For both of these examples, the enhancement or the custom program from where the event should be triggered depends on your application design and business requirements.

In Listing 5.1, we'll raise an event for the business object type, along with an additional event parameter. The inbound delivery number, which is the key field of object type BUS2015, is passed as an input to the program as P_INBDEL, along with the decision code that is passed via parameter P_ACTION. Workflow API SAP_WAPI_CREATE_EVENT triggers event ZWEQualityInspComplete for the entered delivery number and maps the action code to the event container.

```
INCLUDE: <cntn01>. " Include for Container Macros

***Selection screen
PARAMETERS: p_inbdel TYPE likp-vbeln OBLIGATORY, " Delivery
             p_action TYPE char10 OBLIGATORY.    "Approved or Rejected

***Data declarations
DATA: v_object_key   TYPE swo_typeid, " Object key
      v_subrc        TYPE sysubrc,    " Return Code
      s_evt_cont     TYPE swr_cont,    " Container (name-value pairs)
      t_evt_container TYPE swrtcont.

***Constant declarations
CONSTANTS: c_object_type TYPE swo_objtyp VALUE 'BUS2015', " Type
           c_event       TYPE swo_
event VALUE 'ZWEQualityInspComplete', " Event ,
           c_evt_param   TYPE swc_
elem VALUE 'ZWCDecisionCode'.    " Element

*Fill object key for event
DATA(v_delivery) = CONV vbeln_vl( |{ p_inbdel ALPHA = IN }| ).
v_object_key = v_delivery.
```

```

*Fill event container with decision code
s_evt_cont-element = c_evt_param.
s_evt_cont-value = p_action.
APPEND s_evt_cont TO t_evt_container.
CLEAR s_evt_cont.

CALL FUNCTION 'SAP_WAPI_CREATE_EVENT'
  EXPORTING
    object_type      = c_object_type "BUS2015
    object_key       = v_object_key
    event            = c_event
    commit_work      = abap_true
    event_language   = sy-langu
    language         = sy-langu
    user            = sy-uname
  IMPORTING
    return_code      = v_subrc
  TABLES
    input_container = t_evt_container.

```

Listing 5.1 Example Source Code to Trigger a BOR Event with Parameters via the Workflow API

In Listing 5.2, we'll raise an event for an ABAP class, along with an additional event parameter. The billing document number, which is the key attribute for ABAP class ZCLOTCH_INV_RELEASE_ACCOUNTING is passed as input to the program. The additional event parameter for auto release is set to true (X). The method RAISE of class CL_SWF_EVT_EVENT raises the event APPROVE_INV_WORKFLOW for the entered billing document number and maps the parameter for auto release to the event container.

```
PARAMETERS: p_vbeln TYPE vbrk-vbeln. "SD Billing document number
```

```

***Data declarations
DATA : lr_event_parameters TYPE REF TO if_swf_ifs_parameter_container, "
Container for Transfer of Parameters
      lr_catch             TYPE REF TO cx_root, " Abstract Superclass for All
Global Exceptions
      lv_msg               TYPE string,
      lv_objkey            TYPE char32, " Objkey of type CHAR32
      lv_id                TYPE char01, " Id of type CHAR01
      lv_param_name        TYPE swfdname. " Element ID (32 Characters,
Unique, Not Language-Dependent)

```

```

***Constant declarations
CONSTANTS : lc_objtype TYPE sibftypeid
            VALUE 'ZCLOTCH_INV_RELEASE_ACCOUNTING', " Type
            lc_event    TYPE sibfevent
            VALUE 'APPROVE_INV_WORKFLOW', " Event
            lc_catid    TYPE sibfcetid
            VALUE 'CL'. " Category of Objects in Persistent Object References

cl_swf_evt_event=>get_event_container(
  EXPORTING
    im_objcateg = cl_swf_evt_event=>mc_objcateg_cl
    im_objtype  = lc_objtype
    im_event    = lc_event
  RECEIVING
    re_reference = lr_event_parameters ).

* set up the name/value pair to be added to the container
lv_param_name = 'AUTO_RELEASE'. "parameter name of the event
lv_id         = abap_true.

* Add the name/value pair to the event container
TRY.
  lr_event_parameters->set(
    EXPORTING
      name = lv_param_name
      value = lv_id ).

CATCH cx_swf_cnt_cont_access_denied INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
CATCH cx_swf_cnt_elem_access_denied INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
CATCH cx_swf_cnt_elem_not_found INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
CATCH cx_swf_cnt_elem_type_conflict INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
CATCH cx_swf_cnt_unit_type_conflict INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
CATCH cx_swf_cnt_elem_def_invalid INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
CATCH cx_swf_cnt_container INTO lr_catch.
  lv_msg = lr_catch->get_text( ).
ENDTRY.

DATA(lv_vbeln) = CONV vbeln_vf( |{ p_vbeln ALPHA = IN }| ).

```



```

*      assigning the billing no to object key
lv_objkey = lv_vbeln.
*      Raise event to trigger the workflow
TRY.
  cl_swf_evt_event=>raise(
    EXPORTING
      im_objcateg      = cl_swf_evt_event=>mc_objcateg_cl
      im_objtype       = lc_objtype
      im_event         = lc_event
      im_objkey        = lv_objkey
      im_event_container = lr_event_parameters ).

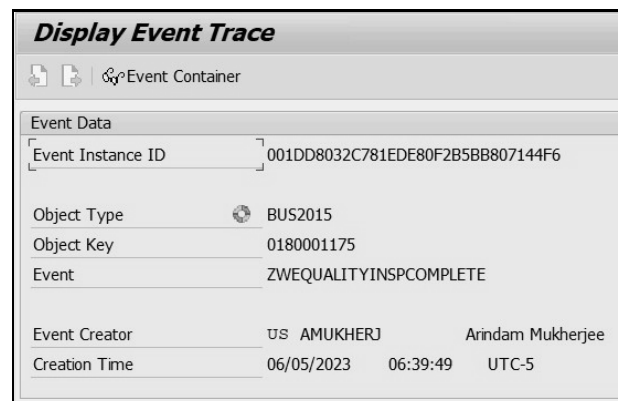
  CATCH cx_swf_evt_invalid_objtype INTO lr_catch.
    lv_msg = lr_catch->get_text( ).
  CATCH cx_swf_evt_invalid_event INTO lr_catch.
    lv_msg = lr_catch->get_text( ).
ENDTRY.

```

Listing 5.2 Example Source Code to Trigger a Class-Based Event with Parameters via a Method Call

5.2 Event Creators, Receivers, and Event Linkage

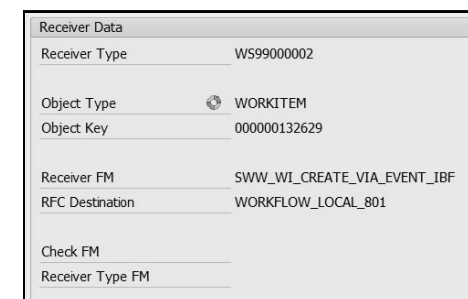
An event creator is the object that raises an event. In the previous section, we discussed the various options for raising an event from an application. Technically, the event creator is the user who raised the event, with any of the previously mentioned event publishing mechanisms. The name of the event creator is visible in the event trace (Transaction SWEL), and normally when the event is used to trigger a workflow, the event creator is bound to the workflow container element `_Wf_Initiator` (workflow initiator). Figure 5.26 shows the event creator details in the event trace Transaction SWEL.



Display Event Trace			
Event Data			
Event Instance ID	001DD8032C781EDE80F2B5BB807144F6		
Object Type	BUS2015		
Object Key	0180001175		
Event	ZWEQUALITYINSPCOMPLETE		
Event Creator	US AMUKHERJ	Arindam Mukherjee	
Creation Time	06/05/2023	06:39:49	UTC-5

Figure 5.26 Event Trace Showing Event Creator Information

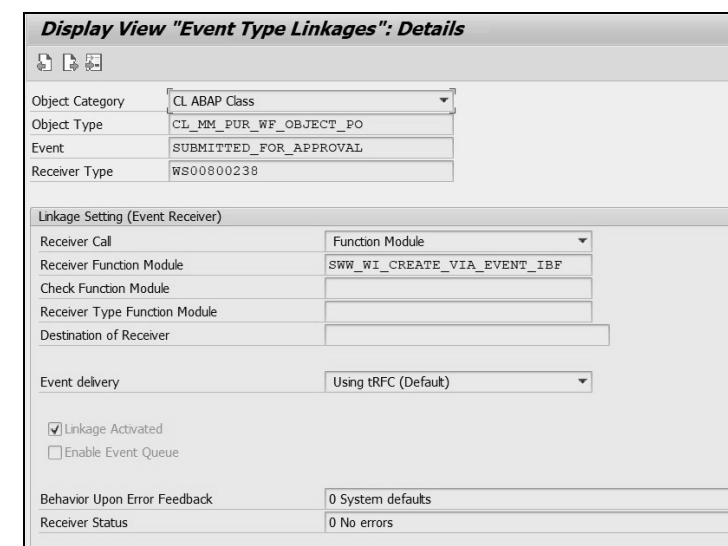
An event receiver is the object that receives the event and performs subsequent processing. Technically, the event receiver could be a workflow template (multistep task), a single-step task, a function module, or a handler class method. In the event trace Transaction SWEL, you can see the event receiver details (if any) under the receiver data. In Figure 5.27, you can see the event receiver details in the event trace in Transaction SWEL. Here, the receiver is a workflow template (multistep task) and the receiver **Object Key** shows the work item ID of the workflow triggered from the event.



Receiver Data	
Receiver Type	WS99000002
Object Type	WORKITEM
Object Key	000000132629
Receiver FM	SWW_WI_CREATE_VIA_EVENT_IBF
RFC Destination	WORKFLOW_LOCAL_801
Check FM	
Receiver Type FM	

Figure 5.27 Event Trace Showing Event Receiver Information

Event linkage is the link between an event creator (semantically, the BOR object type/class and event) and a receiver (workflow template/task/function module/method call). Event linkages can be created in Transaction SWE2 or Transaction SWETYPV. This linkage is also implicitly created when you enter a triggering event on a workflow template or single-step task from Transaction PFTC. Figure 5.28 shows a sample event linkage entry from Transaction SWE2 with various details. We'll look at the components on the screen to understand the significance of that entry in the event linkage.



Display View "Event Type Linkages": Details	
Object Category	CL ABAP Class
Object Type	CL_MM_PUR_WF_OBJECT_PO
Event	SUBMITTED_FOR_APPROVAL
Receiver Type	WS00800238
Linkage Setting (Event Receiver)	
Receiver Call	Function Module
Receiver Function Module	SWW_WI_CREATE_VIA_EVENT_IBF
Check Function Module	
Receiver Type Function Module	
Destination of Receiver	
Event delivery	Using tRFC (Default)
<input checked="" type="checkbox"/> Linkage Activated	
<input type="checkbox"/> Enable Event Queue	
Behavior Upon Error Feedback	0 System defaults
Receiver Status	0 No errors

Figure 5.28 Event Linkage from Transaction SWE2 or Transaction SWETYPV

The following details are visible in an event linkage entry under the **Linkage Setting (Event Receiver)** section of the screen, as shown in Figure 5.28:

■ Receiver Call

This dropdown field lets you choose between a function module or method call as the event handler and acts as a trigger for the receiver type. If you choose **Function Module** in this option, then you must enter a receiver function module in the **Receiver Function Module** field that appears. This function module must use the interface mentioned in the **Receiver Function Module** bullet point later in this list. For single-step or multistep tasks as receiver, the function module is the default receiver call, and `SWW_WI_CREATE_VIA_EVENT_IBF` is the default function module.

If you choose a **Method** call in this field, then you must enter a class that implements interface `BI_EVENT_HANDLER_STATIC`. The **Method Name** field is defaulted as `ON_EVENT`, which must implement the logic to handle the event. Figure 5.29 shows an example of an event linkage with a method call as the receiver call.

Display View "Event Type Linkages": Details	
Object Category	CL ABAP Class
Object Type	CL_CRMS4_SERVICE_CONTRACT_WF
Event	ITEMS_CREATED
Receiver Type	BEH_BSP
Linkage Setting (Event Receiver)	
Receiver Call	M Method
Class Name	CL_CRMS4_BSP_BEH_ITEM_RENEWAL
Interface Name	BI_EVENT_HANDLER_STATIC
Method Name	ON_EVENT
Check Function Module	CRMS4_BSP_BEH_CHK_RENEWAL
Receiver Type Function Module	
Destination of Receiver	
Event delivery	Using tRFC (Default)
<input type="checkbox"/> Linkage Activated	
<input type="checkbox"/> Enable Event Queue	
Behavior Upon Error Feedback	0 System defaults
Receiver Status	0 No errors

Figure 5.29 Sample Event Linkage Showing the Receiver Method Call

■ Receiver Type

For a multistep task or single-step task, you'll see the workflow or task ID in this field starting with a prefix of WS or TS, respectively. This is the workflow template or the task that has been triggered by the event. The event must be maintained in the **Triggering Events** tab of the workflow template or task, and the event linkage must be

active (green). If the receiver is a function module or a class method, then this field acts as a dummy entity. No object exists in the system with this ID.

■ Object Type/Object Key

These fields appear in the event trace in the **Receiver Data** area of the screen (refer to Figure 5.28). If the receiver type is a workflow or task, then **Object Type** is **WORKITEM**, which is the runtime object type for a workflow or task. The **Object Key** then represents the work item ID of the triggered workflow instance (or single-step task). If the receiver is a function module or class method, the **Object Type** and **Object Key** fields are both empty.

■ Receiver Function Module

This field appears when you choose **Function Module** from the **Receiver Call** type dropdown. When the receiver type is a workflow or a single-step task, this function module is defaulted as `SWW_WI_CREATE_VIA_EVENT_IBF`. If you choose some other receiver type, then you can choose your own function module or class method based on the **Receiver Call** dropdown. You can choose the receiver as function module or method when you simply want to update some transaction data via a BAPI or other API once the event is raised from an application. For example, you may want to create an outbound delivery automatically when a sales order is saved. The receiver function module must implement the same interface as template function module `SWE_TEMPLATE_REC_FB` (BOR-based objects only) or template function module `SWE_TEMPLATE_REC_FB_2` (both BOR- and class-based objects).

■ Check Function Module

A check function module is a source code triggered synchronously by the event creator if an active event linkage is found and before triggering the event receiver. The check function module may be used to determine if the event receiver should be triggered or not based on the event instance data. For example, you've developed a workflow for approval of purchase orders, but you only want to trigger the workflow for specific purchase order types. The check function module may be attached to the event linkage to check the purchase order type (EKKO-BSART) before you decide whether to trigger the workflow (receiver) for a given purchase order. The check function module must implement the same interface as template function module `SWE_TEMPLATE_CHECK_FB` (BOR-based objects only) or template function module `SWE_TEMPLATE_CHECK_FB_2` (both BOR- and class-based objects).

■ Receiver Type Function Module

This is used when you have multiple receivers linked to the same event (event linkages), and you only want to trigger one receiver at runtime based on the object instance data. For example, you've developed three different workflow templates for the approval of three types of purchase orders, such as, direct purchases, indirect purchases, and stock transfer orders. All three workflows have the same triggering event. Now when the purchase order Created event is raised, you need to check the

purchase order document type and decide which workflow to trigger. This requirement may be achieved via receiver type function module. The output of the function module is the receiver type (workflow or single-step task or function module/method). The receiver type function module must implement the same interface as template function module `SWE_TEMPLATE_RECTYPE_FB` (BOR-based objects only) or template function module `SWE_TEMPLATE_RECTYPE_FB_2` (both BOR- and class-based objects).

■ Destination of Receiver

In this field, you can enter the logical Remote Function Call (RFC) destination of the receiver, if the receiver is in a different system from the event creator.

■ Event delivery

Event delivery from creator to receiver can happen via transactional RFC (tRFC; default) or via queued RFC (qRFC). This setting goes together with the **Enable Event Queue** checkbox in the same screen. We'll study more about event delivery and event queue administration in Chapter 8, Section 8.7.

■ Linkage Activated

This indicates that the event linkage is active. Receiver determination and triggering only happens when the event linkage is active. This flag can also be updated from Transaction PFTC when you activate the triggering event on a workflow template or a task.

■ Behavior Upon Error Feedback

This setting determines how the system should react if an error occurs while delivering an event to the receiver. Default setting is **0 System defaults**, which means that the global setting from event administration (Transaction SWEAD) is used. Other options for this field include the following:

- **1 Deactivation of Linkage:** If an error occurs, then the event linkage will be automatically deactivated.
- **2 Mark linkage as having errors:** If an error occurs, then the event linkage is marked as `Errors`. This setting influences the next field on the event linkage, that is, **Receiver Status**.
- **3 Do not change linkage:** There is no impact on the event linkage with this setting even if an error occurs.

Normally, you'll maintain the setting **0 System defaults** in each individual event linkage and control the error feedback behavior via the global setting in Transaction SWEAD. More details on event administration will be covered in Chapter 8.

■ Receiver Status

Based on the error feedback setting in the previous field (or global setting), the **Receiver Status** field may show as **0 No errors** or **1 Errors** after an error occurs while trying to trigger an event receiver.

5.3 Start Conditions in Workflows

In the previous section, you learned about check function modules in event linkage, which can be used to evaluate any condition before deciding to trigger the receiver for a particular object instance. For example, you might want to trigger a workflow for the release of purchase orders, but you want to restrict the approval for specific purchase order types only. This condition may be evaluated in a check function module. If you raise the exception `NO_RECTYPE` in this check function module, then the workflow is not triggered by the event.

An alternative and more recommended option compared to the check function module is **Start Conditions in workflows**. This is more of a configuration approach, provided that the fields or the variables you want to use for the condition are already available as an attribute in the BOR type or the ABAP class, which defines the event. If not, then you first need to create a custom attribute in the business object or ABAP class that can be used for configuring the start condition in Transaction `SWB_COND` (or via the **Start Events** tab under the header details of a workflow definition in Transaction `SWDD`). The event linkage should exist before you can create a start condition for the same. The start condition itself is a Customizing object and can be transported to other systems or clients in a Customizing transport request.

In Section 5.1.3, we looked at an example of how to trigger custom event `ZWEQualityInspection` from the BOR type `BUS2015` (delegated to subtype `ZBUS2015`). Now, let's suppose that we've created an event linkage of this event with a custom workflow, but we only want to trigger the workflow for certain delivery types. There are many ways to apply this condition filter, but, in this example, we'll explore the approach of start conditions using Transaction `SWB_COND`:

1. Ensure that an event linkage exists in Transaction `SWE2` or Transaction `SWETYPV` for the concerned event and workflow/task because that is a prerequisite to creating a start condition. Figure 5.30 shows an event linkage entry from Transaction `SWE2` without a start condition. Note that the **Check Function Module** field is empty in this case.
2. Ensure that the field(s) to be used in the start condition exist as attribute(s) in the BOR type of the event. In this example, you need the `Delivery` type as an attribute in business object type `BUS2015`. Because this attribute doesn't exist in the standard, you must create a custom attribute in the delegated subtype `ZBUS2015`, as shown in Figure 5.31. (Refer to Chapter 3, Section 3.1.2, for details on how to create a database attribute in a business object type.) We've created custom database attribute `ZWADeliveryType` for this purpose.

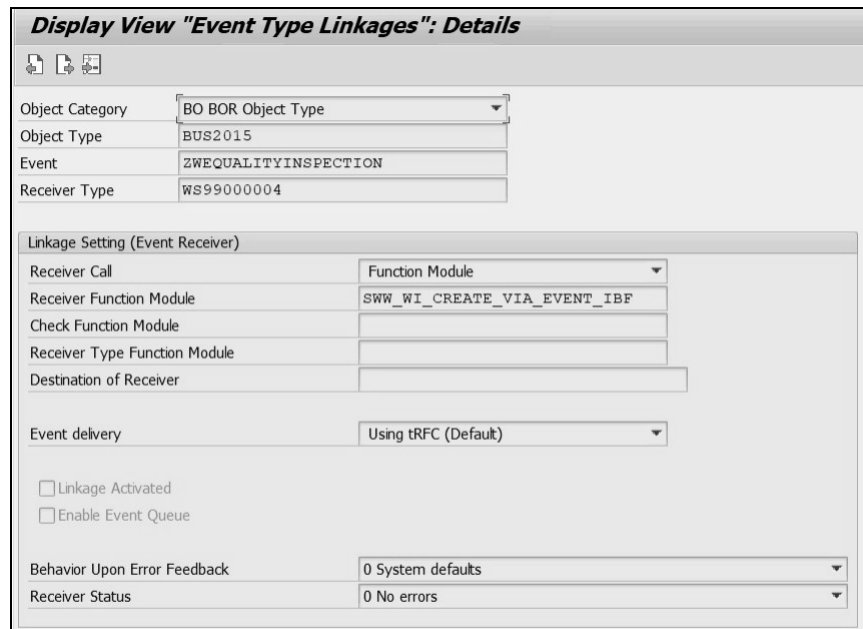


Figure 5.30 Event Linkage before Assigning a Start Condition

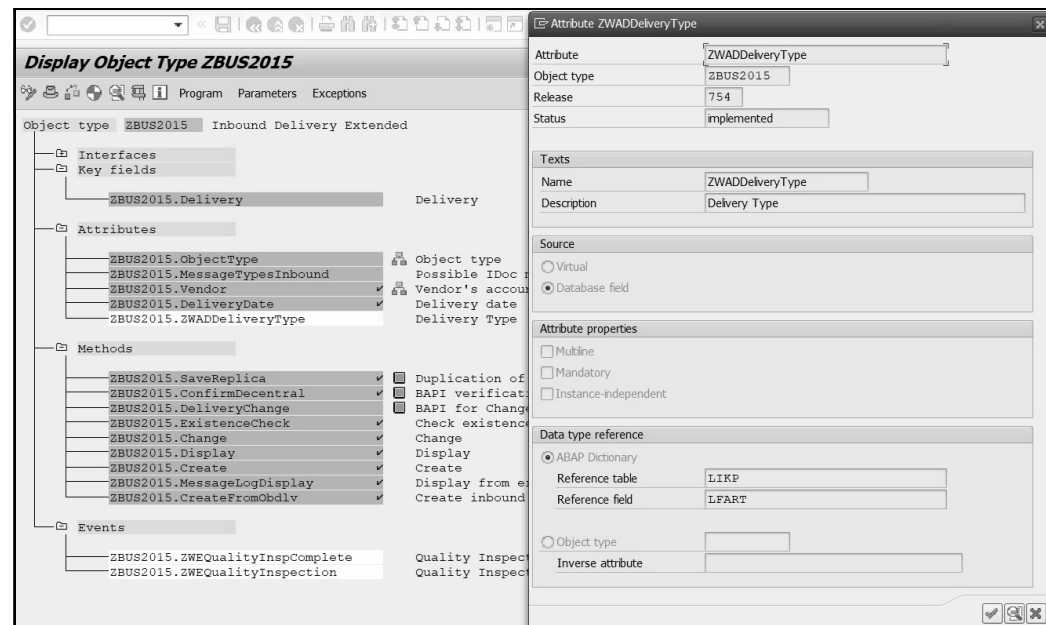


Figure 5.31 Creation of a Custom Attribute in the BOR Subtype Definition

3. Now you can create the start condition using Delivery type attribute for the event linkage in Transaction SWB_COND. Figure 5.32 shows a view of the event linkage entry in Transaction SWB_COND before creating a start condition for the same. (You

can search for the event linkage based on the business object type/ABAP class, event name, receiver workflow, or task ID.)

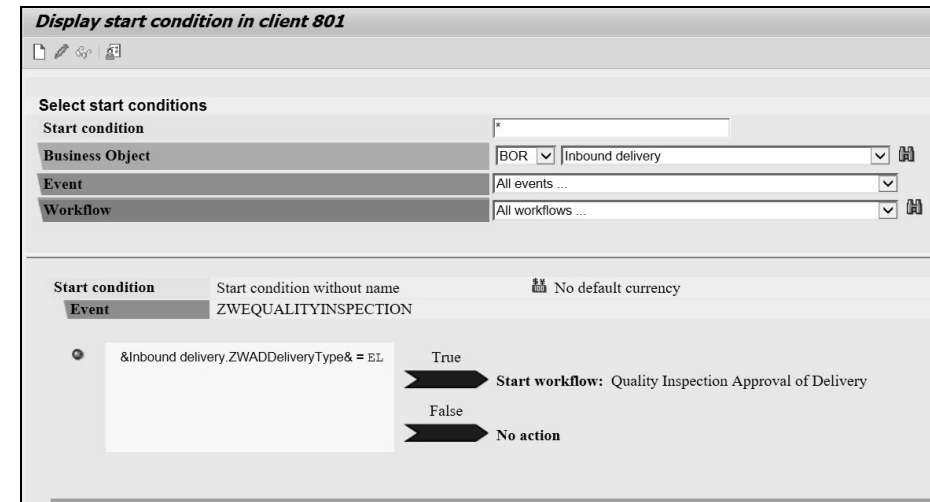


Figure 5.32 Creation of Start Condition for Event Linkage from Transaction SWB_COND

4. Click on the **Create** button on the toolbar, and then click on the selected event linkage. A popup screen will appear with the condition editor that allows you to create flexible start conditions. In the screen shown in Figure 5.33, enter the start condition per your requirement. (In this case, we're checking for delivery type = EL).

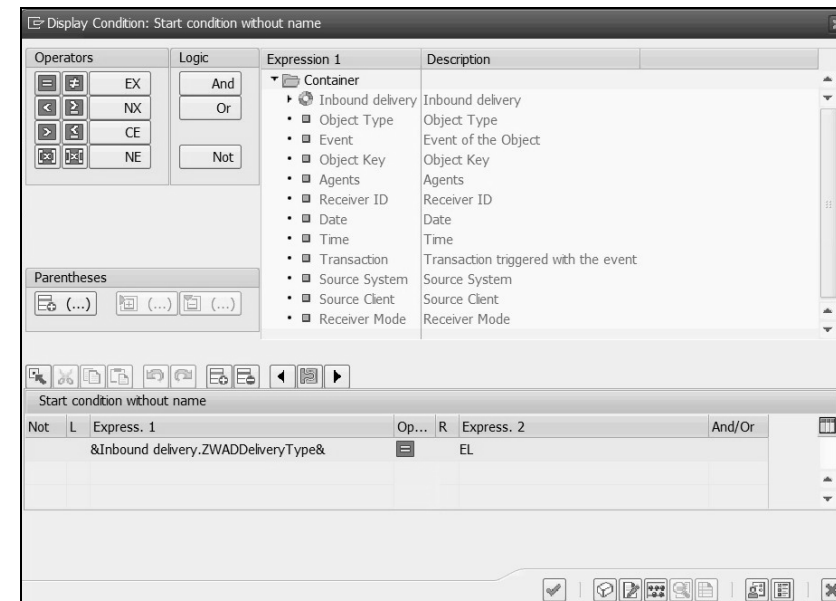


Figure 5.33 Maintaining the Start Condition Using the Condition Editor in Transaction SWB_COND

5. Confirm the condition popup screen once editing is complete.
6. In Figure 5.34, activate the condition by clicking on the red-light icon until it turns green. Save the condition and capture it in a Customizing transport for moving to other systems/clients.

Figure 5.34 Activate the Start Condition (Indicated by the Green Icon to the Left)

Now it's time to test the start condition. When you trigger event ZWEQualityInspection for a delivery with type EL, the start condition will evaluate to True, which may be viewed from the event trace in Transaction SWEL. Figure 5.35 shows the event trace with a successful receiver trigger after the start condition evaluation.

Object Type	Event	Current Date	Time	Name of Receiver Type	Infor...	Handler/Action
ZBUS2015	ZWEQUALITYINSPECTION	06/19/2023	06:26:11	WS99000004		SWW_WI_CREATE_VIA_EVENT_IBF

Figure 5.35 Event Trace Showing Event Triggered after Evaluation of Start Condition

Now if you trigger the event for a delivery with any other type, for example, DIG, then the start condition will evaluate to False, and the receiver workflow won't be triggered from the event trace, as shown in Figure 5.36.

Event Data	
Event Instance ID	001DD8032C781EEE83D2A171EB0E91AD
Object Type	ZBUS2015
Object Key	0180001179
Event	ZWEQUALITYINSPECTION
Event Creator	US AMUKHERJ Arindam Mukherjee
Creation Time	06/19/2023 06:33:00 UTC-5
Receiver Data	
Receiver Type	WS99000004
Object Key	
Receiver FM	SWW_WI_CREATE_VIA_EVENT_IBF
RFC Destination	WORKFLOW_LOCAL_801
Check FM	SWB_2_CHECK_FB_START_COND_EVAL
Receiver Type FM	
Message	Start condition returns 'FALSE' for object [BO,BUS2015,0180001179]

Figure 5.36 Event Trace Showing Exception Raised due to Start Condition Being Evaluated to False

5.4 Terminating Events and Instance Linkage

Up to this point, we've discussed triggering events and their linkage to workflows (multistep tasks) or single-step tasks. These events are maintained in the **Triggering events** tab of a workflow or single-step task. Tasks can also have terminating events that are used to signal the end of processing for a dialog work item. Usually, terminating events are attached to asynchronous dialog tasks. This means that the underlying BOR method is declared as asynchronous (**Synchronous object method** checkbox is unchecked in the method definition in Transaction SWO1). For ABAP classes, however, this synchronous/asynchronous attribute doesn't exist at the method level. It's flagged at the task level only from Transaction PFTC.

In Figure 5.37, the task for releasing a purchase order is marked as asynchronous.

On the **Terminating events** tab, events **RELEASED**, **RESET**, and **SIGNIFICANTLYCHANGED** are maintained as terminating events for this task, as shown in Figure 5.38. Terminating events are entered by selecting the object type container element, from where the BOR object type or the ABAP class name is derived. Then, you must select the event name from the BOR object type or ABAP class. Binding can be maintained between the event container and task container, similar to triggering events.

Standard Task: Display

Standard task: 20000166 mm_po_rel
 Name: Release of purchase order
 Package: ME Applic. Component: MM-PUR

Abbr.: mm_po_rel
 Name: Release of purchase order
 Release status: Not defined

Work Item Text: Please release purchase order & _WI_Object_Id.PurchaseOrder&

Object method: BO BOR Object Type
 Object Type: BUS2012 Purchase Order
 Method: SINGLERELEASE Individual Release
 Synchronous object method
 Object method with dialog

Figure 5.37 Example of an Asynchronous Standard Task

Standard Task: Display

Standard task: 20000166 mm_po_rel
 Name: Release of purchase order
 Package: ME Applic. Component: MM-PUR

Binding	Object Category	Object Type	Event	Name	Element
<input checked="" type="checkbox"/>	BO BOR Object Type	BUS2012	RELEASED	Purchase Order Release PO	_WI_OBJECT_ID
<input checked="" type="checkbox"/>	BO BOR Object Type	BUS2012	RESET	Purchase Order Not Used	_WI_OBJECT_ID
<input checked="" type="checkbox"/>	BO BOR Object Type	BUS2012	SIGNIFICANTLYCHANGED	Purchase Order Changed Signi...	_WI_OBJECT_ID

Figure 5.38 Terminating Events in an Asynchronous Dialog Task

Whenever the system triggers any one of the preceding events for a purchase order; any **Ready** or **In Process** work items for task TS20000166 will be set to **Completed** status. Terminating events may be raised using the same kind of mechanisms as a triggering event, namely change documents, status management, message control, or ABAP code in enhancements. Binding may be done from event to task container to receive the updated object instance into the task/workflow.

Moving on to our next topic, instance linkages are automatically created by the workflow runtime system, whenever a work item is created for an asynchronous task that has terminating events attached to it. Unlike event linkage for triggering events, instance linkages aren't created manually. They may be viewed or updated manually (as an administrator for exception situations) via Transaction SWE3 or Transaction SWEINST. Each instance linkage consists of a header record with the BOR/class, event, and receiver type, along with the type-linkage active and event queue active indicators. Under the header entry, you'll see the object details data, with the object key fields and the receiver key (work item ID if task is the receiver type). Figure 5.39 shows the view of instance linkages created through workflow runtime data in Transaction SWE3 or Transaction SWEINST.

Display View "Instance Linkages": Overview

Object Category	Obj. Type	Event	Receiver Type	Type linkage...	Enable event...	Status
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	WITHDRAWN_FROM_APPR..	WORKFLOW_HANDLER_CA..	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	CANCEL_WORKFLOW	WORKFLOW_HANDLER_CA..	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RESTART_WORKFLOW	WORKFLOW_RESTART_FL..	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	CL_MM_PUR_WF_OBJECT..	CANCELLED	WORKFLOW_HANDLER_CA..	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors
CL ABAP Class	ZCLOT_INV_RELEASE..	CANCELLED_INV	EVENTITEM	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0 No errors

Figure 5.39 Instance Linkages from Transaction SWE3 or Transaction SWEINST

On selecting any row from the table and clicking on the **Object Data** node, you can view the object instances (list of documents) for which the instance linkage header entry was created (see Figure 5.40).

Display View "Object Data": Overview

Category	Obj. Type	Event	Receiver Type	Object Key	Receiver Key
CL ABAP (CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000034	000000122849
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000000	000000128463
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000287	000000128559
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000289	000000128576
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000314	000000130839
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000316	000000131082
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000317	000000131217
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000320	000000131496
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000321	000000131499
CL ABAP..	CL_MM_PUR_WF_OBJECT..	RELEASED	WORKITEM	8600000329	000000131710

Figure 5.40 Object Instances for an Instance Linkage from Transaction SWE3 or Transaction SWEINST

Instance linkages work automatically in the background based on design-time definitions entered in the task. Usually, no manual intervention is required. You can use the instance linkage transaction for analysis to check which objects are currently awaiting a terminating event. Once the terminating event is raised for an object instance, the corresponding entry is deleted from this table.

5.5 Check Function Module and Receiver Function Module for Events

In Section 5.2, you learned the definition and purpose of check function modules and receiver function modules, which are maintained in the event linkage entry. Both function modules may be entered automatically by the system or manually by the developer.

When you configure a start condition for an event linkage via Transaction SWB_COND (or directly via the **Start Events** tab under the header details of a workflow definition in Transaction SWDD), then default check function module SWB_2_CHECK_FB_START_COND_EVAL is inserted in the event linkage entry. If no start condition is maintained in Transaction SWB_COND, then adding a check function module is a manual task.

Similarly, when you maintain a workflow or a single-step task as the receiver for an event, then default receiver function module SWW_WI_CREATE_VIA_EVENT_IBF is automatically added to the event linkage entry. For nonworkflow receivers, you must manually enter a receiver function module or a class and method name as the event handler object.

In Section 5.3, we used the start condition approach to add a filter based on the delivery type on the quality inspection approval workflow event linkage. In this case, we explored an alternate approach using the check function module. The advantage of this approach is that you don't need the condition field(s) to be created as attributes in the leading BOR type or ABAP class. The disadvantage is that you need to write some code for formulating the start condition. In a real business scenario, you must carefully consider both approaches and decide which one is the cleaner and more efficient option for you. Listing 5.3 contains sample source code of a custom check function module. Note the use of exception NO_RECTYPE in determining the result of the check.

```
FUNCTION z_delivery_qinsp_wf_check.
*"-----
**"Local Interface:
*" IMPORTING
*"   VALUE(OBJTYPE) LIKE SWTYPECOU-OBJTYPE
*"   VALUE(OBJKEY) LIKE SWEINSTCOU-OBJKEY
*"   VALUE(EVENT) LIKE SWTYPECOU-EVENT
*"   VALUE(RECTYPE) LIKE SWTYPECOU-RECTYPE
*" TABLES
```

```
**"   EVENT_CONTAINER STRUCTURE SWCONT
**" EXCEPTIONS
**"   NO_RECTYPE
**"-----

*--Local constant declarations
  CONSTANTS: lc_inb_delivery TYPE lfart VALUE 'EL'. " Delivery Type

  IF objtype = 'BUS2015' AND
     event = 'ZWEQUALITYINSPECTION'.

     DATA(lv_vbeln) = CONV vbeln_vl( objkey ).

***Fetch delivery type from delivery header table
  SELECT SINGLE FROM likp " SD Document: Delivery Header Data
    FIELDS lfart
    WHERE vbeln = @lv_vbeln
    INTO @DATA(lv_lfart).
  IF sy-subrc = 0.
    IF lv_lfart <> lc_inb_delivery. "EL
      MESSAGE 'Invalid delivery type for Quality Inspection WF' TYPE 'E'
RAISING no_rectype.
    ENDIF. " IF lv_lfart <> lc_inb_delivery
  ENDIF. " IF sy-subrc = 0

  ENDIF. " IF objtype = 'BUS2015' AND

ENDFUNCTION.
```

Listing 5.3 Sample Code for a Check Function Module in Event Linkage

Receiver function modules (or method calls) for nonworkflow receiver types may be used for a variety of purposes in real business scenarios, for example, if you want to trigger an interface after a sales order is saved. You would ideally want to send the sales order number or the updated data from the order in the interface, so you first look for an exit or BAdI in the update task, or you may decide to go for a custom receiver function module configured in an event linkage. Because receivers are triggered via an event that is raised after the standard save of the transaction, you can query all database tables related to the transaction in the receiver function module. It also allows you the flexibility of configuration to activate or deactivate the interface or maintain start conditions per requirement. Another common scenario for using custom receiver function modules is to create a follow-on document from a transaction on save, for example, if you want to create the delivery for a sales order automatically after saving.

Receiver function modules must implement the same interface as template function module `SWE_TEMPLATE_REC_FB` (BOR-based objects only) or function module `SWE_TEMPLATE_REC_FB_2` (both BOR- and class-based objects). For receiver method calls, you must create a custom class using interface `BI_EVENT_HANDLER_STATIC`. Method `ON_EVENT` of this interface must be implemented for the handler logic.

5.6 Summary

In this chapter, we started by explaining the concept of events with respect to workflows. Then we discussed the different event triggering techniques in detail, some involving configuration and others involving ABAP code. Then, we talked about event creators and event receivers, followed by describing each element of an event linkage entry. We also talked about event instance linkages. Finally, we looked at some examples to understand the concept of start conditions, check function modules, and receiver function modules with respect to SAP Business Workflow.

Contents

Preface	19
1 Getting Started	23
1.1 Workflows in SAP S/4HANA	23
1.1.1 Evolution of Workflows from SAP ERP to SAP S/4HANA	24
1.1.2 Workflow Options	27
1.2 Workflows in SAP Business Technology Platform	32
1.3 Workflow Development Tools	34
1.3.1 Tools for Classical Workflow	34
1.3.2 Tools for Flexible Workflow	38
1.3.3 BRFPplus Development Tools	38
1.3.4 Tools for SAP Business Technology Platform	43
1.4 Identity and Access Management	45
1.4.1 Roles and Authorizations in SAP S/4HANA	45
1.4.2 Roles and Authorizations in SAP Business Technology Platform	47
1.5 Summary	48

Part I SAP Business Workflow for SAP S/4HANA

2 Introduction to Classical Workflows	51
2.1 Evolution of Classical Workflows	51
2.2 Standard Workflows	53
2.2.1 Searching for Standard Workflows	53
2.2.2 Commonly Used Standard Workflows	55
2.3 Configuring the SAP Business Workflow System	56
2.3.1 Maintain Runtime Environment	57
2.3.2 Maintain Definition Environment	64
2.3.3 Maintain Additional Settings and Services	66
2.3.4 Classify Tasks as General	68
2.4 Activating and Deactivating Standard Workflows	70

2.5	Configuring Agents for Standard Workflows	72
2.6	When to Develop a Custom Workflow	73
2.7	Summary	75
3	Building Methods and Tasks	77
3.1	Business Object Repository Approach	78
3.1.1	Business Object Type Definition	79
3.1.2	Defining a Custom Business Object Repository Object Type	88
3.1.3	Creating Key Fields and Attributes	91
3.1.4	Creating Methods and Defining Properties, Parameters, and Exceptions	97
3.1.5	Creating Business Object Repository Events	105
3.1.6	Testing a Business Object Repository Object Type	106
3.1.7	Creating a Subtype of a Standard Business Object Repository Object Type	108
3.1.8	Delegation	110
3.1.9	Business Object Repository Programming	111
3.2	ABAP Class Approach	116
3.2.1	Creating a Workflow Class	116
3.2.2	Defining Key Attributes and Non-Key Attributes	118
3.2.3	Creating Methods and Defining Attributes, Parameters, and Exceptions	120
3.2.4	Adding New Methods for Dialog and Background Tasks	123
3.2.5	Method Exceptions	124
3.2.6	Creating Events	128
3.2.7	Testing an ABAP Workflow Class	129
3.3	Task Definition	130
3.3.1	Defining Standard Tasks and Task Settings	131
3.3.2	Work Item Text and Description	135
3.4	Summary	137
4	Defining Workflows and Adding Steps	139
4.1	Workflow Builder	139
4.2	Common Workflow Steps	144

4.3	Adding Tasks	154
4.3.1	Adding an Activity Step	155
4.3.2	Adding a Send Email Step	157
4.3.3	Adding a User Decision Step	158
4.4	Containers and Bindings	160
4.4.1	Types of Containers	160
4.4.2	Binding Definition and Binding Operators	165
4.4.3	Custom Transformations in Binding	166
4.5	Multiline Elements and Dynamic Parallel Processing	167
4.6	Deadline Definition	169
4.7	Summary	176
5	Defining and Triggering Events	177
5.1	Event Triggering Techniques	178
5.1.1	Event Trigger via Change Documents	178
5.1.2	Event Trigger via Status Management	185
5.1.3	Event Trigger via Message Control	188
5.1.4	Event Trigger via ABAP Code in User Exits, Business Add-Ins, and Custom Programs	193
5.2	Event Creators, Receivers, and Event Linkage	198
5.3	Start Conditions in Workflows	203
5.4	Terminating Events and Instance Linkage	207
5.5	Check Function Module and Receiver Function Module for Events	210
5.6	Summary	212
6	Agent Determination	213
6.1	Different Types of Agents in Workflow	214
6.1.1	Possible Agents and Responsible Agents	214
6.1.2	Excluded Agents	217
6.1.3	Actual Agents	218
6.1.4	Deadline Agents	219
6.1.5	Notification Agents	219

6.2	Agent Determination Rules	220
6.2.1	Rule Definition	221
6.2.2	Rule Container	222
6.2.3	Binding to the Rule Container	223
6.2.4	Rule Types	224
6.3	Possible Agents in Tasks and Default Rules	234
6.4	Organizational Structure Definition and Linking to Workflow Agents	237
6.5	Summary	239
7	Email Notifications and Other Runtime Jobs	241
7.1	Prerequisites for Setting Up Email Notifications in Workflow	241
7.1.1	Setting Up Email IDs for SAP Users	242
7.1.2	Setting up SAPconnect	242
7.2	Classical Work Item Email Notifications via Program RSWUWFML2	244
7.3	Extended Notification Configuration with Program SWNCONFIG	252
7.3.1	Overview of the Notification Process	252
7.3.2	Detailed Customizing	254
7.4	Adding Inbox and Work Item URL Links to Workflow Work Item Notifications	262
7.5	Additional Workflow Runtime Jobs	264
7.6	Summary	265
8	Workflow Administration, Monitoring, and Troubleshooting	267
8.1	Workflow Log	268
8.2	Workflow Administration	274
8.3	Workflow Error Diagnosis and Resolution	279
8.4	Workflow Inbox and Features	283
8.5	Substitution and Automatic Forwarding	289
8.5.1	Substitutions in SAP GUI	290
8.5.2	Substitutions in SAP Fiori	297

8.6	Display Dynamic Labels for Tasks to Display in Business Workplace	299
8.7	Event Trace and Event Queue Administration	301
8.8	Process Incoming Documents with ArchiveLink	305
8.9	Summary	310
9	Application Link Enabling and Reporting	313
9.1	Error Handling during IDoc Processing	314
9.1.1	Business Requirements for Inbound IDoc Error Handling	314
9.1.2	Handling the Inbound IDoc Error	315
9.1.3	Notification of Successful Posting	320
9.1.4	Testing Procedure	323
9.1.5	Setting Up an Inbound IDoc Process via a Workflow	326
9.2	Active Monitoring	327
9.3	Common Workflow Data Tables	329
9.4	Common Workflow Application Programming Interfaces	330
9.5	Workflow Reporting	331
9.6	Implementing Program Exits to Capture Data from Workflow Steps	335
9.7	Summary	339
10	BRFplus	341
10.1	Introduction to BRFplus	341
10.1.1	Business Rules Management Systems	342
10.1.2	Rule Modeling	342
10.1.3	Rule Execution Engine	346
10.2	Integrating BRFplus Applications in SAP Business Workflow	347
10.2.1	BRFplus Application Overview	348
10.2.2	Attach a BRFplus Function in a Business Workflow	348
10.2.3	Executing the Workflow	352
10.3	Summary	356

11 Integrating Workflows with User Interface Applications and External Applications	357
11.1 My Inbox App Overview	357
11.2 Workflow Task Integration with User Interface Applications	359
11.2.1 Set Up a Scenario-Specific My Inbox Tile	359
11.2.2 Create and Maintain User Attributes: Adding Additional Attributes for a Task	365
11.2.3 Launching SAPUI5 Applications from Workflow Tasks	368
11.2.4 Launching Web Dynpro Applications from Workflow Tasks	371
11.3 Email Templates in SAP S/4HANA	373
11.4 Integrating External Applications with SAP Business Workflow	373
11.5 Summary	374
12 Migrating SAP ERP Workflows to SAP S/4HANA	375
12.1 Migration Options from SAP ERP to SAP S/4HANA	376
12.2 Conversion Projects (Brownfield)	378
12.2.1 Handling System Upgrades for Standard and Custom Workflows	378
12.2.2 Migrating Your SAP ERP Workflows to SAP S/4HANA	381
12.3 New Implementation Projects (Greenfield) and Selective Data Transition	384
12.3.1 Selective Data Transition Overview	384
12.3.2 Managing the Technical Migration of In-Process Workflows	386
12.3.3 Migrating User Data and SAP Office Settings from SAP ERP to SAP S/4HANA	387
12.4 Summary	389
13 Workflow in SAP Master Data Governance	391
13.1 Introduction to SAP Master Data Governance	392
13.1.1 Data Models in Central Governance	393
13.1.2 Change Request in Central Governance	394
13.1.3 Business Object BUS2250	397
13.1.4 Standard Dialog Tasks Used in Workflow Templates	399
13.1.5 Standard Dialog Tasks	401

13.1.6 Standard Background Tasks	402
13.1.7 Agent Determination	403
13.1.8 Workflow Container Used by Workflow Templates	405
13.1.9 Workflow Log for Change Requests	406
13.1.10 Rule-Based Workflow Template	407
13.1.11 SAP Master Data Governance, Consolidation and Mass Processing	418
13.2 Business Partner Workflows	419
13.2.1 Business Partner Data Model and Approach	419
13.2.2 Change Requests for Business Partner Master Data	421
13.2.3 Workflow Templates Used in Business Partner Change Requests	422
13.3 Finance Workflows	431
13.3.1 Finance Data Model	432
13.3.2 Change Request Types in the Finance Data Model	433
13.3.3 Workflow Templates Used in Finance Change Requests	433
13.4 Material Workflows	436
13.5 Summary	437
Part II Flexible Workflow in SAP S/4HANA	
14 Introduction to Flexible Workflow	441
14.1 Authorizations and SAP Fiori Applications Required for Development	442
14.2 Flexible Workflow Scenarios	443
14.2.1 Standard Flexible Scenarios	444
14.2.2 Custom Flexible Scenarios	444
14.2.3 Comparing Flexible and Classical Workflows	444
14.2.4 Choosing Between Classical and Flexible Workflows	445
14.3 Migrating to Flexible Workflows	446
14.4 Setting Up a Standard Flexible Workflow Scenario	447
14.4.1 Finding Standard Workflows on SAP Help	447
14.4.2 Finding Workflows in Scenario Editor and the Manage Workflows App	448
14.4.3 Activating the Scenario	449
14.4.4 Setting Up a Standard Scenario Using the Manage Workflows App	454
14.5 Extending the Standard Flexible Scenario	460
14.6 Summary	461

15 Custom Scenario Development	463
15.1 Workflow Class Development	463
15.1.1 Use Case for Walkthrough of Custom Scenario	464
15.1.2 Classes	465
15.1.3 Interfaces	466
15.1.4 Attributes	466
15.1.5 Events	467
15.1.6 Standard Methods	468
15.2 Business Objects	470
15.2.1 Maintain Business Object Type (V_BO_TYPE)	471
15.2.2 Maintain Object Node Type (SBO_V_NODETYPE)	472
15.2.3 Maintain Core Data Services View (V_SBO_NODE_CDS)	472
15.2.4 Maintain Object Representation	473
15.3 Scenario Development	473
15.3.1 Context Element	474
15.3.2 Process Data	475
15.3.3 Control	477
15.3.4 Activities	478
15.3.5 Conditions	481
15.3.6 Agent Rules	483
15.3.7 Value Helps	485
15.3.8 Email Templates	487
15.4 Create a Workflow Template Using the Manage Workflows App	490
15.5 Initiating the Custom Flexible Workflow	497
15.6 My Inbox Integration	498
15.6.1 Define Step Names and Decision Options	498
15.6.2 Define Visualization Metadata for My Inbox	499
15.6.3 My Inbox for Custom Scenario Walkthrough	499
15.7 Troubleshooting	501
15.8 Summary	504
16 SAP Fiori Applications for Flexible Workflow	505
16.1 Adaptation Transport Organizer Setup	505
16.2 Maintain Email Templates App	507
16.3 Notification Features	510

16.4 Manage Teams and Responsibilities App	514
16.5 Transporting Extensions	515
16.5.1 Configure Software Packages	516
16.5.2 Register Extensions for Transport	518
16.5.3 Transporting Workflow Scenario Content	519
16.6 Summary	520
Part III Workflows with SAP Business Technology Platform	
17 Introduction to SAP Build Process Automation	523
17.1 Overview	523
17.2 Typical Use Cases for SAP S/4HANA	524
17.3 System and Service Requirements	525
17.4 Setting Up the Required Services	526
17.5 Working with the Workflow Cockpit	532
17.6 Security	535
17.6.1 Process Automation Admin	535
17.6.2 Process Automation Developer	535
17.6.3 Process Automation Participants	536
17.7 Troubleshooting	536
17.8 Summary	537
18 Process Development	539
18.1 Workflow Design Techniques	540
18.2 Creating a Workflow Using SAP Business Application Studio	541
18.2.1 Create a Workflow Module	542
18.2.2 Tasks in Workflows	546
18.2.3 Using Gateways	559
18.2.4 Events	562
18.3 Creating a Workflow Using Process Builder	567
18.3.1 Using Process Builder	568
18.3.2 Using Triggers	574
18.3.3 Using Forms	574

18.3.4	Using Approval Forms	576
18.3.5	Using Automation Tasks	577
18.3.6	Using Decisions	578
18.3.7	Using Subprocesses	584
18.3.8	Using Actions	585
18.3.9	Using Mail	590
18.3.10	Using Controls	592
18.4	Building and Deploying the Project	595
18.4.1	Release	595
18.4.2	Deploy	596
18.4.3	Run	597
18.5	Destination Configuration with Authentication	598
18.5.1	Destination Setup	599
18.5.2	Authentication	602
18.6	Transport Management	602
18.7	Using APIs for SAP Build Process Automation	605
18.7.1	Application Programming Interfaces for Workflow	605
18.7.2	Application Programming Interfaces for Decisions	606
18.7.3	Application Programming Interfaces for My Inbox	607
18.8	Design a Process/Workflow for the Use Case	607
18.8.1	Use Case and Solution	607
18.8.2	Design Using Process Builder	609
18.8.3	Design Using SAP Business Application Studio	626
18.9	Summary	634
19	Process Visibility	635
19.1	Configuring Process Visibility	636
19.1.1	Roles	637
19.1.2	Process Preparation	638
19.1.3	Create Visibility Scenario	640
19.1.4	Configure Visibility Scenario	641
19.1.5	Add Process to the Visibility Scenario	641
19.1.6	Configure Phases	643
19.1.7	Configure Status	644
19.1.8	Configure Performance Indicators	647
19.1.9	Release the Project	648
19.1.10	Deploy the Project	650

19.2	Testing the Process Visibility Scenario	652
19.3	Process Monitoring and Real-Time Insights	653
19.4	Add Workflow Actions to the Dashboard	654
19.5	Using Application Programming Interfaces for Process Visibility	660
19.6	Summary	662
20	Task Processing with My Inbox	663
20.1	My Inbox for SAP Business Technology Platform	664
20.1.1	Standard App in SAP Build Process Automation	664
20.1.2	Configure My Inbox in SAP Build Work Zone	665
20.2	SAP Task Center	670
20.3	Summary	673
The Authors	675
Index	679

Index

_Attach_Objects	134
_EVT_OBJECT	177
_Wi_Actual_Agent	134, 218
_Wi_Object_ID	134

A

ABAP CDS views	470
ABAP classes	116
<i>test</i>	129
ABAP Test Cockpit	382
Actions	585, 654
<i>configure</i>	588
Active area	394
Active monitoring	327
Activities	144, 478
ACTOR_TAB	228
Actual agents	218
Ad hoc anchor	148
Adaptation transport organizer (ATO)	505
Administration	274
<i>documentation</i>	276
<i>key activities</i>	277
Agent assignment error	280
Agent determination	28, 213, 403, 483, 501
<i>rule-based workflow</i>	405
<i>rules</i>	220
<i>standard workflow</i>	404
<i>with function module</i>	221
<i>with organization data</i>	221
<i>with responsibilities</i>	221
Agent rules	483, 495, 502
Agent types	235
Agents	72, 155, 274, 504
API call	99
APIs	605, 661
<i>for decisions</i>	606
<i>for workflows</i>	330, 605
<i>My Inbox</i>	607
Application errors	103
Application Link Enabling (ALE)	188, 313
AppRouter	668
Approval forms	576, 617, 620
ArchiveLink	305, 308
<i>configuration</i>	309
Attributes	82, 466
<i>create</i>	91
<i>define key ones</i>	118

Authentication	602
Automatic forwarding	289
Automation tasks	577

B

Background activity	479
Background tasks	402
BAdI	482, 483, 497
BI_EVENT_HANDLER_STATIC	200
BI_OBJECT	116
BI_PERSISTENT	116
Binding editor	132
Bindings	160, 223
<i>custom transformations</i>	166
<i>definition</i>	165
Boundary escalation event	566
Boundary timer event	566
Branches	593, 620
BRFplus	24, 38, 341, 411, 430
<i>APIs</i>	346
<i>application</i>	344
<i>application overview</i>	348
<i>attach function</i>	348
<i>building blocks</i>	344
<i>create new task</i>	349
<i>data objects</i>	344
<i>decision tables</i>	345
<i>execute workflow</i>	352
<i>expressions</i>	345
<i>functions</i>	345
<i>layout</i>	343
<i>overview</i>	341
<i>rule execution engine</i>	346
<i>rule modeling</i>	342
<i>test workflow</i>	353
<i>workbench</i>	342, 351
Brownfield project	376
Buffering	281
Business Application Programming Interface (BAPI)	79
Business object repository (BOR)	28, 77, 177
<i>builder</i>	36
<i>events</i>	105
<i>object types</i>	88
<i>program</i>	86
<i>programming</i>	97, 111
<i>test object types</i>	106

- Business object type definition 79
 - Business objects 316, 470, 504
 - BUS2001 81
 - BUS2032 79
 - BUS2054 81
 - BUS2250 397
 - Business partners 419
 - change requests* 421
 - data model* 419
 - workflow templates* 422
 - Business rules 34
 - Business rules management
 - system (BRMS) 341
 - Business Workplace 284, 357
 - attachments* 287
 - inbox* 285
 - outbox* 287
 - toolbar* 288
 - work items* 287
- C**
- Callback classes 465, 466, 469, 484
 - Central governance 393
 - Change documents 178
 - Change request actions 396
 - Change request step types 396
 - Change request steps 394
 - Change request types 394
 - Check function module 201, 203, 210
 - CL_SWF_EVT_EVENT 194
 - CL_SWF_FLEX_IFS_DEF_APPL_
 - BASE 465, 477
 - CL_SWF_FLEX_IFS_RUN_APPL_
 - BASE 465, 477
 - CL_SWF_IFS_WF_CONSTRUCTOR 335
 - CL_SWF_IFS_WF_DESTRUCTOR 335
 - CL_SWH_WORKITEM_EXIT 335
 - Classic technical view 268
 - Classic user view 268
 - Classical workflows 23, 25, 27, 382
 - activate/deactivate* 70
 - commonly-used* 55
 - evolution* 51
 - search* 53, 54
 - standard* 53
 - tools* 34
 - trigger* 28
 - Classifications 293
 - Condition controls 592
 - Condition editor 182, 319
 - Condition records 191
 - Conditions 137, 145, 157, 481, 486, 583, 619, 622, 631
 - Configure Software Packages app 516
 - CONSTRUCTOR 118
 - Container element binding 150
 - Containers 146, 160
 - types* 160
 - Context elements 474
 - Controls 477, 592
 - CREATE_EVENT 189
 - Create, read, update, and delete (CRUD) 394
 - Custom scenarios 463
 - Custom workflows 73
 - CX_BO_ERROR 124
 - CX_BO_TEMPORARY 124
- D**
- Data elements 617
 - Data models 393
 - Data tables 329
 - Database attributes 92
 - Deadline agents 219
 - Deadline monitoring 512
 - Deadlines 157
 - definition* 169
 - types* 172
 - Decision tables 411, 581, 616
 - nonuser agent* 413
 - single-value* 411
 - user agent* 412
 - Decisions 578, 612
 - Default rules 135, 236
 - Delegation 110
 - Design techniques 540
 - Destinations 598, 671
 - configure* 551
 - setup* 599
 - variable* 601
 - Dialog tasks 399, 401, 479
 - Document from template 145
 - Document templates 143
 - Dynamic column 300
 - Dynamic labels 299
 - Dynamic parallel processing 167
- E**
- Email notifications 241
 - email IDs* 242
 - prerequisites* 241
 - program RSWUWFML2* 244

- Email notifications (Cont.)
 - set up SAPconnect* 242
 - URL links* 262
 - Email tasks 558
 - Email templates 373, 487, 489, 496, 503, 511
 - End event 565
 - Entitlements 526
 - Error diagnosis 279
 - Error handling 314
 - Event linkages 282, 318
 - Event queue 301
 - Event traces 301
 - Event type linkages 397
 - Event-based triggers 610
 - Events 36, 85, 146, 467, 497–499, 562
 - configuration* 181
 - create* 128
 - creator* 198
 - define* 177
 - delivery* 202
 - linkage* 199
 - parameters* 106, 468
 - receiver* 199
 - terminating* 207
 - trigger* 177
 - trigger via ABAP code* 193
 - trigger via message control* 188
 - trigger via status management* 185
 - triggering techniques* 178
 - Exceptions 102, 124
 - Excluded agents 217
 - Exclusive gateway 560
 - Execution views 274
 - EXIT_CANCELLED 115
 - EXIT_NOT_IMPLEMENTED 115
 - EXIT_OBJECT_NOT_FOUND 115
 - EXIT_PARAMETER_NOT_FOUND 115
 - EXIT_RETURN 115
 - Expressions 215
- F**
- Field restrictions 180
 - Flexible blocks 474, 478
 - Flexible workflows 23, 26, 29, 382, 441
 - activate* 450
 - activate scenario* 449
 - authorizations* 442
 - custom* 444
 - customization* 450
 - deactivate event type linkage* 454
 - define steps and decisions* 451
 - Flexible workflows (Cont.)
 - extending* 460
 - migration* 446
 - SAP Help* 447
 - scenarios* 443
 - set up* 447
 - standard* 444
 - tools* 38
 - visualization metadata* 452
 - vs classical workflows* 444
 - when to use* 445
 - Fork 148
 - Forms 145, 555, 574
 - add decision* 556
 - Forward work item 277
 - Forwarding 234
- G**
- Gateways 559
 - get_task_container() 469
 - get_workflow_container() 469
 - Graphical model 143
 - Greenfield implementation 376
- I**
- I_WorkflowTask 487
 - I_WorkflowTaskApplObject 487
 - Identity and access management 45
 - Identity management 530
 - IDOCAPPL 316
 - IDocs 313
 - handle error* 314, 315
 - monitoring* 327
 - processing* 314
 - sales order inbound* 314
 - testing* 323
 - IF_SWF_FLEX_IFS_DEF_APPL 466, 478
 - IF_SWF_FLEX_IFS_RUN_APPL 466, 477
 - IF_SWF_FLEX_IFS_RUN_APPL_STEP 477
 - IF_SWF_IFS_WF_CONSTRUCTOR 335
 - IF_SWF_IFS_WF_DESTRUCTOR 335
 - IF_SWF_IFS_WORKITEM_EXIT 335
 - IF_T100_DYN_MSG 125
 - IF_T100_MESSAGE 125
 - IF_WAPI_WORKITEM_CONTEXT 337
 - IF_WORKFLOW 116, 465, 466, 468
 - IFEXIST 81
 - IFSAP 81
 - IFSTATUS 83
 - IM_EVENT_NAME 337

Import parameters 101
 Incoming documents 305
 Instance linkages 209
 Integration 357
 Interface methods 123
 Interfaces 81, 466
 Intermediate escalation events 565
 Intermediate message events 563
 Intermediate timer events 564

J

Java Unified Expression Language (JUEL) 553
 Jobs 315

K

Key fields 81
 create 91

L

Latest end 157
 Latest start 157
 Leading objects 465, 475, 476
 Local persistent object reference 118
 Local workflow 149
 Loop 148

M

Macros 112
 Mail tasks 590
 Maintain Email Templates app 487, 489,
 496, 508
 custom template 509
 Manage My Substitutes app 667
 Manage Process and Workflows app 633
 Manage Teams and Responsibilities app 514
 Manage Workflow Scenarios app 38, 503,
 508, 519
 Manage Workflows app 38, 443, 448, 478,
 481–483, 485, 491, 519
 deadlines 458
 dialog activity 456
 exceptions 458
 new template 455
 set up standard scenario 454
 step conditions 457
 Material workflows 436
 Message control 188

Methods 77, 83, 468, 469
 attributes 99
 create 97, 120
 definition 84
 Migrating workflows 375, 381
 archive objects 383
 custom code 382
 new features 381
 options 376
 system changes 383
 technical migration 386
 Modeled 171
 Multiline workflow container 168
 Multiple condition 146
 My Inbox 24, 297, 443, 451, 487, 498–500,
 625, 634
 additional task attributes 365
 configure in SAP Build Work Zone 665
 features 359
 integrate external applications 368
 navigation 668
 overview 357
 SAP BTP 664
 scenario-specific 359
 tile 666
 variants 358
 My Inbox – All Items app 359
 target mapping 364

N

Near-zero downtime 385
 Notification agents 219
 Notifications 510

O

Object type definition 93
 Object type status 90
 Object types 191, 316
 Object-oriented programming (OOP) 77
 OData 587
 Optical character recognition (OCR) 306
 Organization management 36
 Organization structure 237
 Organization units 315

P

Parallel gateway 559, 629
 Partner profiles 315

Persistence object references 466
 Person 315
 Phases 643
 Position 315
 Possible agents 214, 234, 235
 Pre-migration activities 375
 Process and Workflow Definitions app 45
 Process and Workflow Instances app 44, 633
 Process builder 44, 567, 609, 638
 artifacts 570
 project name 570
 repository 570
 steps 572
 use 568
 Process codes 315
 Process control 147, 148
 Process data 475
 Process monitoring 653
 Process patterns 409
 Process visibility 635
 add process 641
 configure 636
 configure phases 643
 configure scenario 641
 configure status 644
 create scenario 640
 dashboard 654
 deploy project 650
 performance indicators 647
 process preparation 638
 release project 648
 roles 637
 scenarios 34
 substatus 645
 test scenario 652
 version control 649
 Program exits 157, 335
 Program RESIDOCA 328
 Program RHWEGID00 231
 Program RSWNNOTIFDEL 265
 Program RSWUWFML2 244
 data for individual run 250
 executable attachments 247
 exits 249
 granularity 247
 instance data 246
 log 250
 message text 248
 selection of work items and recipients 251
 shortcuts 249
 variants 251
 Program SWN_SELSEN 264

Program SWNCONFIG 252
 business scenario 254
 category 255
 customizing 254
 delivery 254
 delivery schedule 260
 delta filter 257
 filter pair 256
 full filter 256
 notification process 252
 selection 253
 selection schedules 258
 subscription 260

Q

Quality inspection request 189

R

Receiver call 200
 Receiver determination 202
 Receiver function module 201, 210
 Receiver type 200
 Register Extensions for Transport app 518
 Remote Function Calls (RFCs) 43
 Reporting 331
 Requested end 157
 Requested start 157
 Requirement routine 191
 Responsibility 224, 225
 Responsible agents 214
 RH_GET_ACTORS 220
 Roles 484, 485, 494, 495, 501
 Rule-based workflows 407
 design 414
 extend 417
 Rules 36, 160, 319, 580, 613
 containers 222
 types 224
 Run time class 489
 Runtime jobs 241
 RVNSWE01 189

S

SAP BTP cockpit 526, 599
 SAP Build Process Automation 25–27, 32,
 374, 523, 567
 booster 528
 deploy project 596
 lobby 532, 567

SAP Build Process Automation (Cont.)

- monitor* 532
- My Inbox* 664
- overview* 523
- project name* 534
- release project* 595
- run project* 597
- security* 535
- set up required services* 526
- start project* 533
- store* 532
- system requirements* 525
- troubleshooting* 536
- with SAP S/4HANA* 525
- workflow editor* 534

SAP Build Work Zone 665

SAP Business Accelerator Hub 374

SAP Business Application Studio ... 43, 541, 626

- namespace* 544
- tasks* 546
- workflow module* 542

SAP Business Process Automation 44

SAP Business Technology

- Platform (SAP BTP) 23
- My Inbox* 664
- roles and authorizations* 47
- workflows* 32

SAP Business Workflow

- check HR table entries* 66
- check number ranges* 65
- classify decision task as general* 60
- classify tasks as general* 68
- configure* 56
- configure RFC destination* 58
- email notifications* 245
- integrating external applications* 373
- integrating with BRFplus* 347
- maintain active plan version* 60
- maintain additional settings and services* 66
- maintain administrator* 59
- maintain definition environment* 64
- maintain runtime environment* 57
- maintain time units* 61
- SAP ERP vs SAP S/4HANA* 24
- schedule background job* 61

SAP Cloud Transport Management 602

SAP Content Agent 603

SAP Event Mesh 562

SAP Fiori 376, 505

SAP Fiori launchpad 297, 362

- dynamic tile* 363

SAP GUI 376

SAP inbox 284

SAP Integration Suite 374

SAP Master Data Governance (SAP MDG) .. 391

- consolidation and mass processing* 418
- overview* 392

SAP MDG, Financials 431

- change request types* 433
- data models* 432
- workflow templates* 433

SAP Notes 378

SAP S/4HANA 463

SAP S/4HANA Cloud, private edition 30

SAP S/4HANA Cloud, public edition 30

SAP S/4HANA migration 375

SAP Solution Manager 603

SAP Task Center 24, 670

- booster* 671
- product support* 670
- restrictions* 673

SAP Workflow Management 32, 523, 562

SAP_WAPI_CREATE_EVENT 194

SAP_WAPI_CREATE_EVENT_EXTENDED ... 194

SAP_WFRT 383

SAP_WORKFLOW_SYSTEM 103

SAP_WORKFLOW_SYSTEM_

- TEMPORARY 103

SAPconnect 243

SAPUI5 557

SAPUI5 applications 368

Scenario context 474

Scenario editor 448

Scenario ID 508

Script tasks 546, 627

Secondary priority 227

Security roles 45

Selected agents 217

Selective data transition 376, 384

- overview* 384

Send mail 144, 157, 321

Sentiment 655

Sequence flow 561

Service tasks 549, 629

SIBFLPOR 466

Simulations 41

Staging area 394

Start conditions 203, 320

- test* 206

Start event 476, 563, 627

Status management 185

Steps 144, 478

- add* 155

Subprocesses 584

Substitution 289, 290

- deactivation* 292
- details* 292
- profile* 295
- SAP Fiori* 297
- SAP GUI* 290
- work item-specific* 293

Substitutions 667

Subtypes 87

- create* 108

Subworkflows 151, 152

Supertypes 87

SWB_2_CHECK_FB_START_COND_EVAL ... 210

SWC_CALL_METHOD 115

SWC_CONTAINER 115

SWC_CONTAINER_CREATE 115

SWC_CREATE_OBJECT 114

SWC_GET_PROPERTY 112

SWC_GET_TABLE_PROPERTY 112

SWC_SET_ELEMENT 113

SWC_SET_TABLE 113

SWCONT 111

SWE_EVENT_CREATE 194

SWE_EVENT_CREATE_IN_UPD_TASK 194

SWE_TEMPLATE_CHECK_FB 201

SWE_TEMPLATE_CHECK_FB_2 201

SWE_TEMPLATE_REC_FB 201

SWE_TEMPLATE_REC_FB_2 201

SWE_TEMPLATE_RECTYPE_FB 202

SWE_TEMPLATE_RECTYPE_FB_2 202

SWEAD 202

SWF_CRT_NOTIFY_RECIPIENTS 487

SWF_PROCESS_WORKFLOW_

- CONDITION 482

SWF_WORKFLOW_CONDITION_DEF 461

SWF_WORKFLOW_CONDITION_EVAL 461

SWHACTOR 228

SWW_WI_CREATE_VIA_EVENT_IBF .. 201, 210

System conversion 377

System errors 103

T

Task groups 36

Tasks 36, 77, 153, 546

- add* 154
- container* 134
- define standard* 131
- definition* 130
- description* 135
- settings* 131

Tasks (Cont.)

- visualizations* 369

Technical logs 501–503

Technical view 268

Temporary errors 103

Terminate end events 565

Terminating events 134

Testing 36

Traces 41, 184

Transaction

- BD67* 321, 322
- BRF+* 342
- BSO2* 186
- BSVW* 381
- BSVX* 186
- BSVZ* 187
- FB50* 276
- FDT_HELPERS* 40
- MDGIMG* 393
- NACE* 188, 189
- OOCU_RESP* 225
- PFAC* 134, 221
- PFAC_CHG* 220
- PFAC_DIS* 220
- PFCG* 361
- PFOM* 233
- PFTC* 54, 130, 149, 163
- PPOSW* 36
- RMPS_SET_SUBSTITUTE* 290
- S_ATO_SETUP* 505
- SBWP* 357
- SCDO* 178
- SE16N* 53
- SJOBREPO* 383
- SLG3* 278
- SOST* 325
- SU01* 242
- SWB_COND* 203, 204, 319, 381
- SWDC_RUNTIME* 60
- SWDD* 35, 139, 223, 348
- SWDD_SCENARIO* 38, 442, 448, 474, 511
- SWDM* 53
- SWE2* 70, 199, 381, 418
- SWE3* 209
- SWEC* 180, 381
- SWED* 178
- SWEINST* 209
- SWEL* 178, 192
- SWELS* 178, 282
- SWEQADM* 302
- SWETYPV* 199, 301, 381, 477
- SWF_USER_ATTR* 365

Transaction (Cont.)

SWFVISU	368, 499
SWFVMD1	368
SWI2_ADM1	278
SWI2_DEAD	333
SWI2_DIAG	278, 501
SWI2_DURA	333
SWI5	332
SWI6	37, 269
SWIA	273
SWNCONFIG	253, 263
SWO1	78, 80, 105, 163, 232, 316
SWO3	89
SWPR	279
SWU_OBUF	282
SWU3	56, 61, 132, 380
SWUS	36
VAO1/VAO2	183
WE19	323
WE20	315
WE42	326
Transport management	602
Transport workflow definitions	519
Transporting extensions	515
Transports	42
Trigger workflows	658
Triggering events	134, 150
Triggers	574, 610
Triggers app	45

U

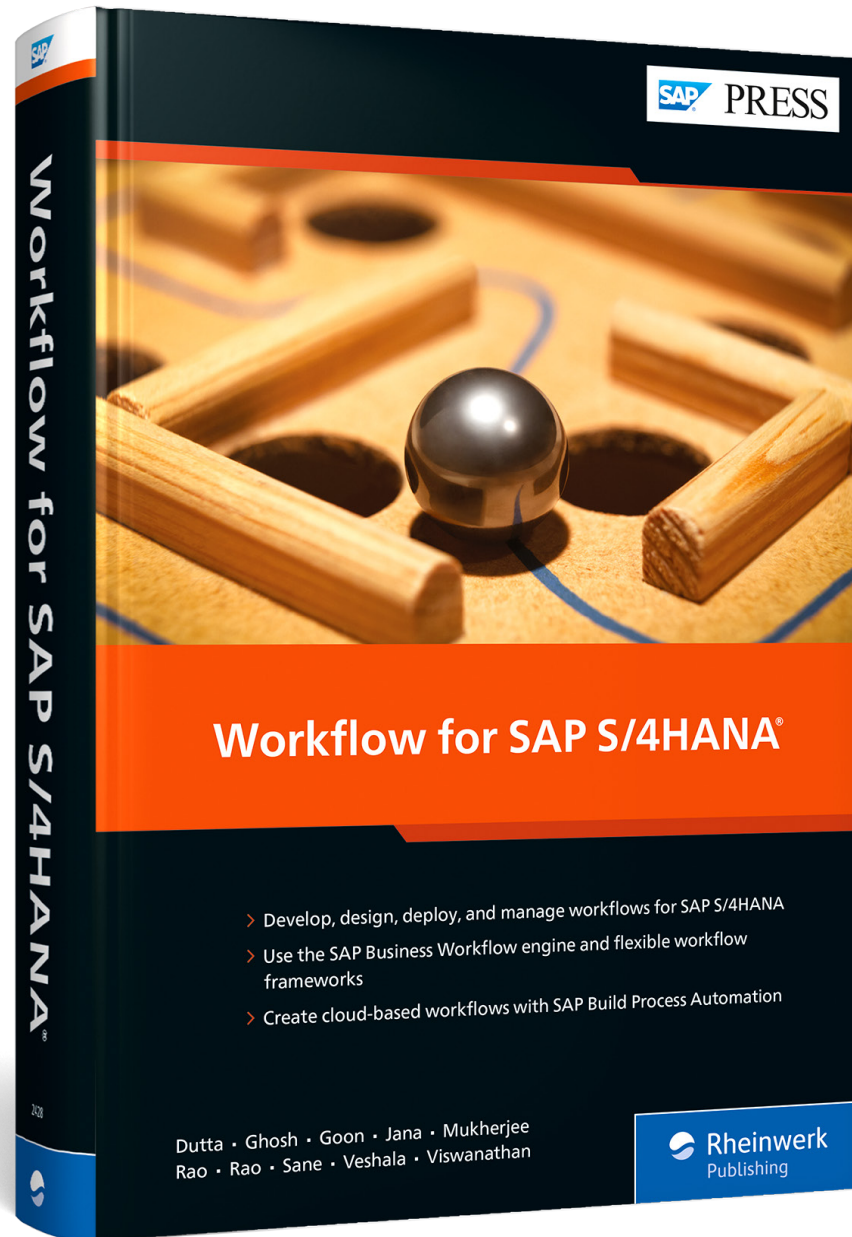
Universal worklist	24, 368
Upgrades	378
<i>workflow testing</i>	380
<i>workflow versions</i>	379
User authentication	530
User decision	145, 478, 479
User interface (UI)	359, 382
User master data migration	387
User tasks	552
User view	268

V

Value help	485, 486
Virtual attributes	92
<i>create</i>	95
<i>definition</i>	111

W

Wait controls	594
Web activities	144
Web Dynpro applications	371
WF chronicle	269
WF-BATCH	383
Work centers	315
Work item text	131
Work items	287
Workflow administrator	46
Workflow Builder	34, 139, 140
<i>information area</i>	140
<i>navigation</i>	141
<i>step types</i>	141
Workflow classes	116, 463
Workflow containers	142, 405
Workflow Designer	549
Workflow developer	45
Workflow events	476
Workflow graphical area	141
Workflow logs	268, 271, 406
Workflow object view	271
Workflow scenarios	473, 475, 481, 490, 491, 493, 504
Workflow start events	476
Workflow steps	481
Workflow tasks	478
Workflow templates	397
WS54300003	427
WS54300007, WS60800059, and WS60800068	429
WS54400001 and WS54300005	424
WS60800086	408, 423
WS60800095 and WS72100006	423
WS75700027	435
WS75700040	433
Workflow wizard	143



The Authors

Sabyasachi Dutta, Nilay Ghosh, Kousik Goon, Sandip Jana, Arindam Mukherjee, Srinivas Rao, Yogeendar Rao, Yogesh Sane, Naveen Veshala, and Kiran Viswanathan are a team of expert consultants, architects, and technical leaders at IBM Consulting.



Dutta, Ghosh, Goon, Jana, Mukherjee, Rao, Rao, Sane, Veshala, Viswanathan

Workflow for SAP S/4HANA

686 pages | 11/2023 | \$89.95 | ISBN 978-1-4932-2428-9

 www.sap-press.com/5697

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.