**Complete Guide for Events in Workflows in SAP ECC 6.0**

**Applies to:** SAP ECC 6.0 and upwards

**Summary**
This tutorial covers the basics of events and their properties. It also covers adding events to Business Objects and using them in workflows. The commons methods used to trigger events in workflows have also been covered under this topic.

**Authors:** Yudhajit Mitra, Sunil Ojha and Abhijit Daptary.
**Company:** Capgemini Consulting India Pvt. Ltd.
**Created on:** 3 January 2007

**Authors Bio**

**Yudhajit Mitra** is an ABAP/Workflow consultant. He is currently working with Capgemini Consulting India Pvt. Ltd.

**Sunil Ojha** is a XI certified workflow consultant. His current engagement is with Capgemini Consulting India Pvt. Ltd.

**Abhijit Daptary** is presently working in Capgemini Consulting India Pvt. Ltd as a Consultant. He has experiences in ABAP, Workflow and XI. He has worked in different fields of and has handled various scenarios on Workflow and XI.
Table of Contents

Events .................................................................................................................................................................3
  Event parameters..................................................................................................................................................3
  Creating an Event ................................................................................................................................................3
  Using Events in Workflows................................................................................................................................6

Methods of Event Creation ....................................................................................................................................7
  Function Module ................................................................................................................................................8
  Status Changes .................................................................................................................................................10
  LIS Exceptions .................................................................................................................................................20
  Message Control .............................................................................................................................................20
  ABAP Classes ................................................................................................................................................20
  HR Master Data ..............................................................................................................................................22

Wizards for Event creation.................................................................................................................................23

Event trace........................................................................................................................................................24

Related Content:...............................................................................................................................................26

Disclaimer and Liability Notice..........................................................................................................................27
Events

Events are part of business object and are triggered by a possible change in the state of the object. Workflows are usually started using events.
Events must be triggered explicitly. The application itself can be the triggering agent.

Event parameters

Event parameters are defined as elements of the event container in the Business Object Builder and are passed to the event receivers using a binding. The following event parameters are held in the event container automatically:

- The reference to the object whose change of state is described by the event
- The user name of the party who triggered the event creation
- There are two different types of data type reference for event parameters:
  - ABAP Dictionary reference
  - Object type reference

Event Container

The event container contains the event parameters as container elements. If you define triggering events or terminating events for a task, you can specify the binding from the event container to the task container. If you define triggering events for a workflow, you can specify the binding from the event container to the workflow container.

The event container contains workflow system elements as standard. You can add more container elements to the event container within object type definition in the Business Object Builder.

Creating an Event

Alternately you can also go to the transaction SWO1.
Select any custom object and press change.
Create a new event by placing the cursor on EVENTS and pressing Create.

Provide an Event ID, name and description.

NOTE: If the checkbox “Triggering object does not exist” is set, the object no longer exists after the event is triggered and so runtime objects can no longer be generated for this object. This is required for certain events (DELETED, for example), where an attempt is made to regenerate the instance triggered by the event after the persistent object has already been deleted.

To add event parameters, position the cursor on the event ID and select PARAMETERS.
To add new event parameters, press **Create**.

When you create an event parameter whose data type reference can be specified by an ABAP Dictionary table field, you should always have the system propose the relevant table fields. Then answer YES to the question **Create with ABAP Dictionary field proposals?** If you want to create an event parameter with an object type reference, you cannot use field proposals, answer the question with NO.

Provide a parameter ID, name and description. Provide a dictionary reference or an object reference and press **to continue.**

**Note:** Selecting the multiline Indicator means that the attribute, the parameter or the container element contains a (multiline) table. For a multiline attribute, a multiline container element or a multiline parameter, a common data type reference is defined, with the result that every "table line" is of the same data type.

After adding an event parameter, return to the previous screen and change the release status of the event. This can be done from **Edit** → **Change Release Status** → **Object Type Component** to **Implemented**. After changing status “To Implemented”, change its status to “Released” by the same process. This indicates that the added event has been released for use by the customers.
Using Events in Workflows

Choose Tools → Business Workflow → Development → Workflow Builder

→ Business Object Builder to add the event as a “Triggering Event” for a custom workflow.

Alternately, you can use transaction SWDD.

In the Workflow Builder, go to BASIC DATA to add the triggering events.

Click on the Start Events tab and add the event MOD_ENTRY as a start event. Provide the Object Type Category as BOR and the object type as that used earlier. Define the binding between the event and the workflow container and activate it. Provide the necessary customizing request number and package on prompt.
**Methods of Event Creation**

There are various ways in which events can be created:

**Function module**  
Event creation by calling function module SWE_EVENT_CREATE or SAP_WAPI_CREATE_EVENT.

**Change documents**  
Event creation when change documents are written.

**General status management**  
Event creation upon status changes.

**Business Transaction Events**  
Logistics Information System (LIS)

**Event creation when an exception situation occurs (LIS exception).**

**Message Control**  
Event creation as message type.

**ABAP classes**  
Event creation by calling the method RAISE of the ABAP class CL_SWF_EVT_EVENT or by calling the method RAISE of an instance of type IF_SWF_EVT_EVENT.

**HR master data**  
Event creation when HR master data changes.
Function Module

You can create events from application or system programs by calling one of the relevant function modules SWE_EVENT_CREATE or SAP_WAPI_CREATE_EVENT.

The function module SWE_EVENT_CREATE has the following interface:

<table>
<thead>
<tr>
<th>OBJTYPE</th>
<th>SWETYPECOU-OBJTYPE</th>
<th>Type of the triggering object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJKEY</td>
<td>SWEINSTCOU-OBJKEY</td>
<td>Concatenated, object type-specific key of the triggering object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The reference to the triggering object is created internally from this information and written to the event container under the element ID _Evt_Object.</td>
</tr>
<tr>
<td>EVENT</td>
<td>SWETYPECOU-EVENT</td>
<td>ID of the event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The event must be defined for the triggering object type.</td>
</tr>
</tbody>
</table>

Export parameters

<table>
<thead>
<tr>
<th>EVENT_ID</th>
<th>SWEDUMEVID-EVTID</th>
<th>The event number has a value other than zero if the event manager could establish one or more receiver function modules.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>It is not possible to state whether the receivers were called successfully.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If no receiver could be established, zero is returned as the event number.</td>
</tr>
</tbody>
</table>

Table parameters

<table>
<thead>
<tr>
<th>EVENT_CONTAINER</th>
<th>SWCONT</th>
<th>Persistent event container of the event.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>You only pass the event container if you have defined event parameters in addition to the predefined elements. The container you pass to the function module is filled with the relevant data in the event-creating application and then contains only the event parameters you defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the function module is executed, the predefined elements (object reference, creation</td>
</tr>
</tbody>
</table>
A typical use of the function module for the created event ADD_ENTRY for object type ZYMBOR can be shown as:

CALL FUNCTION 'SWE_EVENT_CREATE'
EXPORTING
  OBJTYPE = 'ZYMBOR'
  OBJKEY = OBJKY
  EVENT = 'ADD_ENTRY'
* CREATOR = ''
* TAKE_WORKITEM_REQUESTER = ''
* START_WITH_DELAY = ''
* START_RECFB_SYNCHRON = ''
* NO_COMMIT_FOR_QUEUE = ''
* DEBUG_FLAG = ''
* NO_LOGGING = ''
* IDENT =
* IMPORTING
* EVENT_ID =
TABLES
  EVENT_CONTAINER = i_cont
EXCEPTIONS
  OBJTYPE_NOT_FOUND = 1
  OTHERS = 2
.
IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
  WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.

COMMIT WORK.

Here, the key field for the object type is passed as OBJKY and the event container is populated in I_CONT.

**Note:** Since the asynchronous RFC for calling the receiver function module is not triggered until after the next COMMIT WORK, you must write the command COMMIT WORK in your application after the function module for creating an event is called in order for the events to actually be created.
The **database commit** performed automatically with a screen change does not trigger the asynchronous RFC.

Although both the function modules SAP_WAPI_CREATE_EVENT and SWE_EVENT_CREATE have similar signatures, it is recommend that you use SAP_WAPI_CREATE_EVENT instead of the old function module SWE_EVENT_CREATE. For the former, filling the event container is pretty straightforward. There is no need to use the macro SWC_SET_ELEMENT, because the container for SAP_WAPI_CREATE_EVENT is a simple key (element) & value list.

For the following special cases, there are other function modules, which actually use the functionality of the above function module internally:

**SWE_EVENT_CREATE_IN_UPD_TASK**

This function module makes it possible to create events in an update task. In contrast to the function module SWE_EVENT_CREATE, it can be called with the addition of IN UPDATE TASK.

**SWE_EVENT_CREATE_FOR_UPD_TASK**

The event is created in the update task. (The function module is not called with the addition of IN UPDATE TASK.)

If a task in which an object is created is to use the creation of this event as a terminating event, the workflow system uses the *workflow requester*. This means that the event *Object created* can be used as a terminating event, although the object does not yet exist when the task is started. To enable the workflow system to use the *workflow requester* without errors, the function module should be called as the last function module before COMMIT WORK.

**Status Changes**

Event creation can be connected to changes in system status or user status without modifying an existing application.

A status is an indicator that can have only the states *set (active)* or *not set (not active)*. Statuses that can have more than these two states are not supported.

These statuses can be set by the system (*system status*) and by the user (*user status*).

System statuses are indicators that are only set by the system. Their purpose is to document the current state of an object from the system viewpoint.

User statuses are indicators defined by the user, which document the processing state of the object from the viewpoint of the user.

User statuses are only defined within a status profile. A status profile should be regarded as a group of user statuses.

To create an event when a status change occurs, the following settings should be made:
The event that you want to create is defined as a component of an object type. Moreover, the key fields of this object type must be compatible, i.e. same domain, with the key fields of the status object type whose status change is to cause the creation of an event. So, create a custom business object by taking the relevant standard object as Supertype. Then, add the relevant event which is to be triggered to it. For ex- the business object type BUS2005 has been taken as the supertype for a custom object type ZYBUS2005 with an additional event MANC.

Choose Tools → Business Workflow → Development → Definition Tools → Events → Event Creation → Status Management. Alternately, you can go to transaction BSVW.

In the next dialog box, decide whether you want to implement the event creation by connecting to a system status (system settings) or a user status (customer settings). The basic procedure described next is not affected by this decision.

Add a new entry for the assignment. Enter a status object type in the field StatusOT and the object type whose status change is to be indicated by the event created in the field BusinessOT. Enter the ID of the event to be created in the field Event. For ex-As an event is to be triggered for a change in the status of a production order, then status object type should be chosen as ORH and the object type has been created with BUS2005 (production order) as supertype (ZYBUS2005).
This is the event that has been added to the custom business object.

Press SAVE. Then select the row and click on “Status Restrictions”. Click on new entries and add the statuses with which the event needs to be linked. For ex- If the statuses REL and MANC are to be linked, then these are added to system statuses. Afterwards, save the entries.

Thus, if any production orders are created using transaction CO03 with statuses REL or MANC, the event MANC would be triggered. Thus this event can now be used as the triggering event of a workflow or a task as usual.

**Change Documents**

Many business objects are changed frequently, and these changes are logged into change documents to make the changes traceable. Change documents can be created for changes already made as well as for planned changes. The creation of an event can be connected to the writing of these change documents, thus connecting them to triggering of workflows.

To understand how change documents are created in the system, please refer to the documentation Change Documents.

To maintain assignment between a particular change document and an event, go to transaction SWEC. Alternately, you can go to Tools → Business Workflow → Development → Definition Tools → Events → Event Creation → Change Documents → Linkage.
Click on New Entries and add a new record linking the change document to the required event. It is to be remembered that the business object type to which the event belongs and the change documents must have the same key fields. For ex- The business object created earlier (ZYBUS2005) can be used to link the event MANC to the change document ORDER. Thus any changes in a production order, which will be reflected in the change document, will also trigger the event MANC. Also, select whether the event is to be created upon change, create or delete actions. These create, change and delete actions will always be associated with the main table.

Note: In case you may need to create the event for an object type whose key structure is different from the key structure of the change document object, you must first enter a structure for the object type key and a function module in the workflow-relevant settings for change documents. The function module "translates" the key of the change document into the key of the business object. For further information, refer to Maintenance of Workflow-Relevant Settings for Change Documents.
In case the event needs to be triggered only when the change refers to specific fields or fulfills specific conditions, select the condition record and call the view field restrictions.

If the change should refer to specific fields, then those should be added to the lower part of the screen. If certain conditions are to be processed, then those can be added via the condition editor in the upper half of the screen. An existing condition may already be displayed here. For more information on the condition editor, go to Condition editor.

Finally, before a change document object can be used for event creation, certain settings must be made. This information is required, for example, for correct assignment between change document object and object type. To maintain the workflow-relevant settings of change documents, choose Tools → Business Workflow → Development → Definition tools → Events → Event creation → Change documents → Define workflow properties. Alternately, you could go to transaction SWED.

For all change document objects supplied by SAP, the settings required are made by default. However, if you create your own change document object and want to prepare it for the purposes of flexible event creation, then the following fields must be maintained in the view maintenance.
**Main table in change document:** A change document usually refers to several tables. Changes to one of these tables (the "main table" of the change document object) determine the character of the change. For example, the main table for the change document ORDER is AFKO.

**Change document key with structure:** The change document is written with the (key) fields of the table or structure specified in the field Change document key with structure.

**Indicator Action: Create, change, delete:** Change documents are often written when table entries are changed, and less often when they are created or deleted. To construct a full reference, the actions for which a change document is written are specified.

**Indicator Action: Create instance:** Events created from the relevant change document when an object is created can only be used as terminating events if this indicator is set.

**Structure:** This structure must be specified if the key of the event-triggering object cannot be derived from the key of the change document object, but is instead determined with the function module (see below).

**Function module:** Function module called by the system with the import parameters key of change document and object type, and which returns the key of the event-triggering object.

This function module is provided by the application for each change document object, if applicable. Its interface corresponds to the function module SWE_TEMPLATE_OBJECT_KEY_SET, which can be used as a template.

**Business Transaction Events:**

G/L accounting and accounts receivable/payable accounting provide publish and subscribe interfaces, which inform other application components or external interested parties (partners, customers) that particular events (such as document entered) have taken place in financial accounting. The data this creates can be used by the interested parties for their own subsequent processing. But no data can be returned to G/L accounting or accounts receivable/payable accounting.

SAP Business Workflow uses this interface to create events of object types defined in the Business Object Repository from its events. These events can be used as triggering events of a workflow.

Let us look at a simple scenario of configuring Business Transaction Events:

Execute and go to Environment → Infosystem (Processes).

Execute the transaction with the default values.
Find the correct interface for updating the document.

Place the cursor on the key 00001120 and click on Sample Function Module. This navigates you to SE37 (Function Builder). Copy the function module (SAMPLE_PROCESS_00001120) for coding.
Inside the created function module ZSAMPLEPROCESS_00001120, you can call the function module SWE_EVENT_CREATE to trigger any event which has been added as the triggering element for a workflow or a task.

Return to the main screen for Business Transaction Events. In order to assign the function module to an event, create a product as shown.

Add a New entry as shown. Ensure that the Active check box is checked, otherwise the BTE won’t trigger. Save and return to the main menu.

Now, assign the function module created earlier to the event as shown below.
Add a new entry as shown for 00001120 and save it.

Now, that the configuration steps are completed, if a new document is created through transaction FB01, the triggering event for the workflow will be created along with it.

Some of the other methods of triggering events are:
LIS Exceptions

The processing following an exceptional situation defined as an *exception* recognized by the Early Warning System can be used to trigger an event entered as a triggering event of a workflow.

The Early Warning System uses the key figures of the Logistics Information System (LIS). The Early Warning System enables you to select and check weak points within Logistics. The Early Warning System enables you to search for exceptional situations, thus helping you to recognize and rectify potential problems at an early stage.

You define the exceptional situation as *Exception* within the Early Warning System. An *exception* consists of the specification of characteristics and/or characteristic values (such as vendor, material) and conditions. Conditions can be created as threshold values (such as materials/vendors with an order value greater than €5000), as trends (such as a positive trend in the order value) or as planned/actual comparisons.

For more information on the LIS and the Early Warning System choose Logistics ® Logistics (General) ® Logistics Information System ® Early Warning System: Overview.

A particular LIS Exception may be assigned to a particular event by the following ways:

You make entries in a *control table* to define which *LIS exception* is assigned to which object type/event.

You may also specify a function module with which operations can be performed on the event container. The system calls this function module before scanning the *active type linkages* for possible receivers.

You can use the wizard to create events, which can then be used within the definition of workflows. These can be linked to LIS Exceptions directly in the wizard itself. This will be explained in greater detail later on.

Message Control

One of the options for customer-specific creation of events is fully integrated in Message Control. For this, you must create a separate *output type* and declare 9 (workflow event) as *transmission medium* in NACE. For further information, refer to the documentation Message Control. However, this function is not available in a decoupled system.

ABAP Classes

Sometimes, it may be necessary to trigger the event in your program by using an ABAP class method. In such a scenario, the events must be triggered by the ABAP class CL_SWF_EVT_EVENT. If an event is triggered by the following *ABAP objects* command, this has no significance for the event manager:

```
RAISE EVENT <evt> EXPORTING... <ei> = <f i>...
```

However, in the *event handler method* the event can be triggered for the event manager by using the ABAP class CL_SWF_EVT_EVENT. See also: ABAP Objects -> Triggering and Handling Events.
There are implementation examples in the CL_SWF_FORMABSENCE demo class. To trigger the event, use a static method (method: DELETE of the demo class) or an instance method (method: UPDATE of the demo class).

The following parameters must be provided, regardless of whether the event is triggered by a static method or an instance method (see also: demo class):

<table>
<thead>
<tr>
<th>Import parameters</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>im_objcateg</td>
<td>SWF_CLSTYP</td>
<td>Workflow object type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· MC_OBJCATEG_CL (constant of the CL_SWF_EVT_EVENT class) for an ABAP class</td>
</tr>
<tr>
<td></td>
<td></td>
<td>· MC_OBJCATEG_BOR (constant of the CL_SWF_EVT_EVENT class) for a BOR object</td>
</tr>
<tr>
<td>im_objtype</td>
<td>C</td>
<td>ID of the object type, ID of the ABAP class</td>
</tr>
<tr>
<td>im_event</td>
<td>C</td>
<td>ID of the event</td>
</tr>
<tr>
<td>im_objkey</td>
<td>C</td>
<td>Object key</td>
</tr>
</tbody>
</table>

To set the event parameters, use the IF_SWF_IFS_PARAMETER_CONTAINER interface. Use the SET method of this interface to set the relevant parameters.

Use the TRY ... ENDTRY instruction to catch the exceptions cx_swf_evt_execption (error in event creation) and cx_swf_cnt_container (error in setting a parameter).

Finally, Since the asynchronous RFC for calling the receiver function module is not triggered until after the next COMMIT WORK, you must initiate the command COMMIT WORK in your application after the method for creating an event is called in order for the events to actually be created.
HR Master Data

To create events when HR master records change, carry out the following IMG activities:

- Activate an event linkage
- Define event types for customer-specific business objects
- Redefine event types for SAP business objects

These IMG activities can be found in the SAP Reference IMG under Personnel Management → Global Settings in Personnel Management → Business Workflow Events.
Wizards for Event creation

You can use these wizards to create events, which can then be used within the definition of workflows.

The wizards do not cover all aspects of event creation. For an alternative way to create events, choose Tools → Business Workflow → Development → Definition Tools → Events → Event Creation.

Wizards are available for the following:

Change documents
Logistics Information System
Business Transaction Events

To execute a wizard, choose Tools → Business Workflow → Development → Definition Tools → Events → Event Creation → Set Up with Wizard as shown below.

Then select the wizard that you want to execute.
Event trace

All events created correctly are logged in the event trace irrespective of whether potential receivers exist. However, the event trace is only written if it has been activated.

If an event is not created, you must first activate the event trace then create the event again to find out why.

To activate/deactivate the event trace, choose:

Tools → Business Workflow → Development → Administration → Event Manager → Event Trace → Switch Event Trace On/Off. Alternately, you can go to transaction SWEL.

To display the event trace, choose:

Tools → Business Workflow → Development → Administration → Event Manager → Event Trace → Display Event Trace. Alternately, you can go to transaction SWEL and execute.
To delete the event trace, choose:

Tools → Business Workflow → Development → Administration → Event Manager → Event Trace → Delete Event Trace. Alternately, you can go to transaction RSWELOGD.
Related Content:

*SAP Business Workflows*

*Business Workflows*

*Business transaction events in FI document posting*
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.