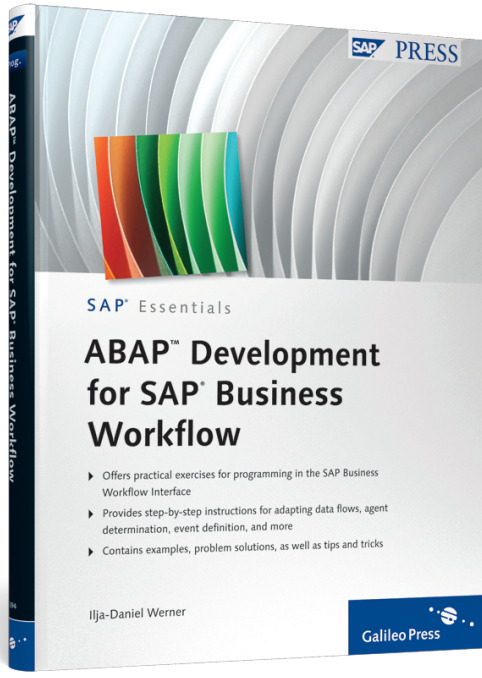
Ilja-Daniel Werner

ABAP™ Development for SAP® Business Workflow



Galileo Press

Bonn • Boston

Contents at a Glance

Contents

1 Introduction

First, I'd like to welcome you to the group of SAP Business Workflow programmers. This book evolved from the experience I've gained in my own workflow projects, as I'm originally an ABAP and Java programmer. As soon as you tackle a workflow project, "workflow-focused humanoids" suddenly bounce around the department using very strange vocabulary and have very unusual requirements for transportation dates, parameter design, quality assurance, and test scenarios. But they don't reveal exactly why these things are so important to them. After a fairly long time, these workflow people disappear, and it's now up to you to handle the ABAP part of the commissioned workflows. You might think this is not very difficult. But as soon as you change two or three totally harmless things, nothing works any longer. And that's when the workflow people appear again.

The special challenge in workflow programming is that the workflow-related ABAP development requires a somewhat different programming approach and you, as the developer, must understand the process knowledge. So you not only require special programming skills, but a very special process knowledge as well.

Motivation and Target Group

In this context, the modeling of new workflows is the most obvious "new territory," but it's not as difficult as it may seem to get used to the Workflow Builder and the various step types in the workflow. Literature, SAP online help, and of course SAP training (see Appendix A) are available for this purpose. Because I hold some of these training sessions myself, I'm aware of the gaps that exist between day-to-day ABAP programming and the training exercises. The goal of this SAP Essentials book is to attend to those gaps.

The difference between working with the Workflow Engine and using the classic application development is that different people execute their code in different roles. You must have multiple test users created in more complex workflows and examine your coding accordingly. Also, you must always ensure persistence, unique keys, and central logging—aspects whose importance is not as high as in "regular" ABAP programming.

This book is aimed at those who contribute to the workflow development and want to design their ABAP world to be maintenance-friendly and still "workflow-conscious." Of course, workflow modelers who want to take a close look at the ABAP side of workflow programming can find possible "starting points."

Content and Structure

In this book, you will get to know the Workflow Engine from the ABAP development perspective.

Initially, **Chapter 2**, Getting Started, deals with the workflow environment and how you can customize it. You learn how to find and examine sample workflows. Within the scope of this workshop, you can quickly determine the SAP Basis administrators in your team that you should see on a regular basis in future. Moreover, you are introduced to the maintenance of organizational structures that you require for agent determination.

Then **Chapter 3**, Compiling a Workflow Development Environment, presents the workflow development environment to give you a feel for the method of developing in the workflow environment. You learn how to navigate within the workflow components and keep track of your developed elements using a where-used list.

Chapter 4, Methods, Work Items, and Events, addresses and details the terms "work item," "methods," and "events" and the concepts behind these terms.

Chapter 5, Intervening in the Agent Determination, deals with the options that are available to influence the agent determination via the administration of responsibilities and programmed agent determination. As nice as a process may be modeled and as versatile as this model may be, it's also important that the right people are involved in a process. This programming activity is often underestimated in the workflow environment.

Even if a workflow is established in your operation, the people who are involved in this workflow presumably change more frequently than the workflow design itself. It's unacceptable that you or the workflow expert must be called whenever an employee is hired or transferred to a new department. The most powerful tool for this task is the organizational management borrowed from HR for the Workflow Engine. This means that as an ABAP programmer, you will still need to deal with this HR-related philosophy to some extent.

In Chapter 6, Containers, Binding, and Conditions in the Workflow, you learn how you can use ABAP programming to add additional features to the standard objects in the SAP Business Workflow. We give you detailed descriptions of the options that the workflow containers provide as a general data structure. Exercise examples describe the intervention in workflow processes (condition editor), binding, or event creation.

The last two chapters—**Chapter 7**, Sample Project—Designing an ABAP Objects Class for the Workflow, and **Chapter 8**, Sample Project—Designing a BOR Object for the Workflow—illustrate the development of concrete workflow objects. Based on identical core functionality, these chapters compile a workflow-enabled ABAP Objects class and a BOR object, respectively. The chapter discusses the specific features of BOR objects, such as enhancement, inheritance, and delegation, using a concrete example.

The appendices provide additional information: **Appendix A** lists all possible workflow steps and refers to sample workflows from the SAP standard that show their use. **Appendix B** summarizes the most important transactions for workflow programming in ABAP.

The entire presentation includes numerous figures and step-by-step instructions to help you implement your workflow requirements completely and deal with typical obstacles.

After this first successful workflow, you should concern yourself with another verification workflow, which is available via Transaction SWUI_VERIFY (Start Test Workflows). As an ABAP developer, you can surely find some interesting examples in this list. Execute them and frequently check the workflow log if possible. This way you can get an impression of how your Workflow Engine ticks.

The customizing of the Workflow Engine is definitely interesting for ABAP programmers.

Memory Aid: Particularly Important Background Jobs

Familiarize yourself with the classic ABAP techniques for asynchronous function calls via tRFC and qRFC:

- `SWWCOND`: Work item rule monitoring
- `SWWDHEX`: Work item deadline monitoring
- `SWWERRE`: Work item error monitoring
- `SWWCLEAR`: Clearing tasks in the workflow system

2.2 Starting Workflows and Monitoring the Workflow Events

After you've completed the preparations for the Workflow Engine operation, you can take a closer look at a demo workflow. Let's use the already known Transaction SWUI_VERIFY as the starting point (see Figure 2.11). The workflow template available there tests the Workflow Engine and dispenses the mapping of an organizational structure to the greatest extent.

So in the following business process example, you as the *workflow initiator* initially receive all work items. This is the user who manually started a workflow template or in whose user context the starting event was executed for the workflow template.

The procedure according to which the Workflow Engine determines the users to which a work item is sent is referred to as *agent determination.* For now, let's only discuss the workflow initiator, as this is always the SAP user who executes the test transaction or has generated the relevant event.

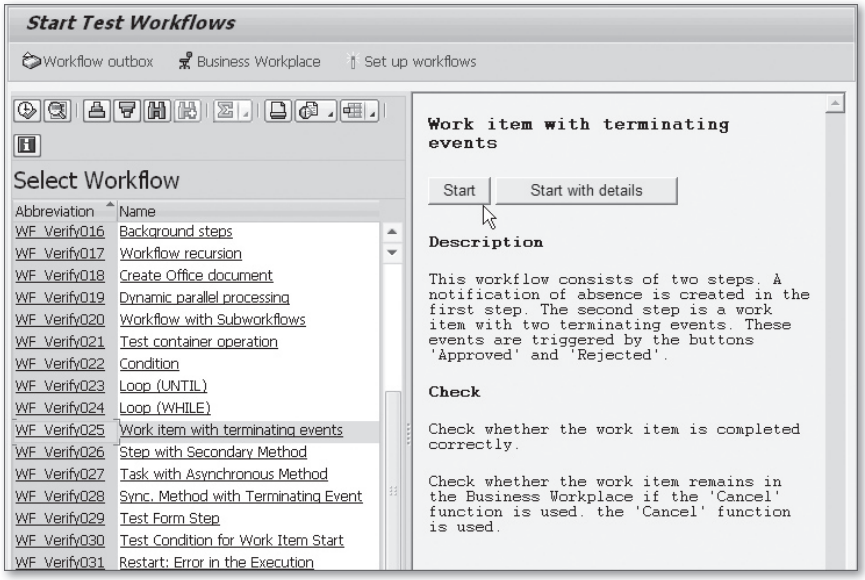


Figure 2.11 Transaction SWUI_VERIFY—Many Test Options for the Workflow Engine

2.2.1 Monitoring Events (Transactions SWELS and SWEL)

A business process may take a long time. During a business process, you can stop and restart the SAP system. To prevent the Workflow Engine from being caught in innumerable background loops, SAP has put great emphasis on persistence and asynchronous processing.

To design such an asynchronous processing robustly, special importance is attached to the delivery of events—nothing is supposed to "get lost."

1. Use the event trace to monitor which events are created in the system and how they are delivered to the Workflow Engine. Switch on the general event trace in Transaction SWELS (Switch Event Trace On/Off) (see Figure 2.12).

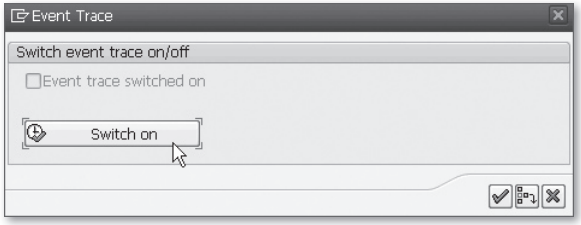


Figure 2.12 Transaction SWELS—Switching on the Event Trace

2. Return to Transaction SWUI_VERIFY and successively start the workflow templates `WF_Verify025` (see Figure 2.13) and `WF_Verify048` (see Figure 2.14).

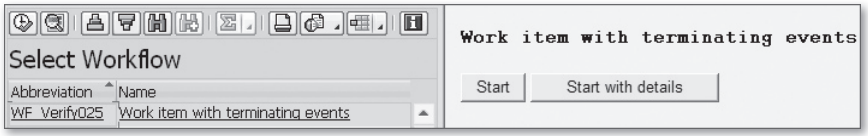


Figure 2.13 Workflow using a BOR Object for Notification of Absence Demonstrating Terminating Events

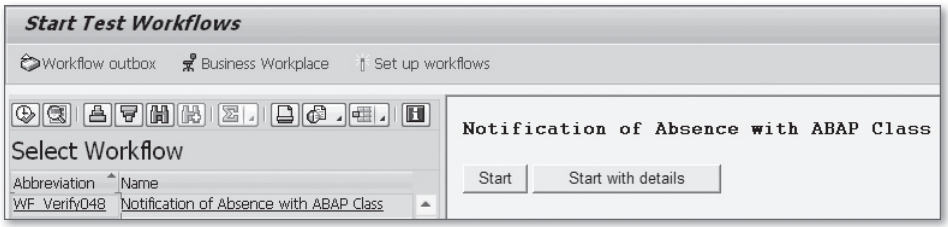


Figure 2.14 Workflow using an ABAP Class for Notification of Absence

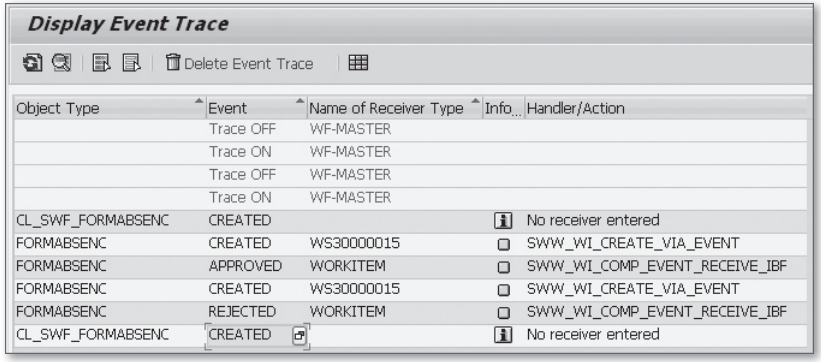
Both workflows record the notification of absence of an employee in a simple form and submit this form for inspection or approval to another employee.

3. Follow the instructions for the implementation of the test and sample workflows, which are displayed on the right-hand side of the screen, as accurately as possible. During this test, you will come across several possible behaviors of a work item:
   - At one point, you must go to the SAP Business Workplace.
   - At another point, the Workflow Engine analyzes whether the work item is intended for the user whose SAP GUI instance is currently active. If this is true, the form is displayed automatically. This procedure is referred to as *advance in dialog* and is frequently used in test, training, and demo workflows. In more complex workflows that model comprehensive business processes, this procedure is rarely used.
   - At yet another point, you are prompted to complete the work item. This *explicit completion* can make sense if multiple comprehensive user actions are expected in a workflow step and the system cannot exactly determine based on events whether the prerequisites for continuing the workflows are

met. Here, the acting user must confirm explicitly that the workflow step is complete (see Figure 2.15).

Figure 2.15 Explicit Completion of a Work Item

4. After you've passed the test workflow, go to Transaction SWEL (Display Event Trace) and select the events for object type `*FORM*`. You should now see the entries for `FORMABSENC` (a BOR object) and `CL_SWF_FORMABSENC` (an ABAP Objects class or ABAP OO class) (see Figure 2.16).



Object Type	Event	Name of Receiver Type	Info...	Handler/Action
	Trace OFF	WF-MASTER		
	Trace ON	WF-MASTER		
	Trace OFF	WF-MASTER		
	Trace ON	WF-MASTER		
CL_SWF_FORMABSENC	CREATED			No receiver entered
FORMABSENC	CREATED	WS30000015		SWW_WI_CREATE_VIA_EVENT
FORMABSENC	APPROVED	WORKITEM		SWW_WI_COMP_EVENT_RECEIVE_IBF
FORMABSENC	CREATED	WS30000015		SWW_WI_CREATE_VIA_EVENT
FORMABSENC	REJECTED	WORKITEM		SWW_WI_COMP_EVENT_RECEIVE_IBF
CL_SWF_FORMABSENC	CREATED			No receiver entered

Figure 2.16 Events in the Event Trace—BOR Objects and ABAP Classes as Object Type

Workflow-relevant events can be triggered both by ABAP classes (identified by `CL_` in the name) and by BOR objects. For now, you only need to remember that these two object types exist.

You now know one of the most important analysis tools of the workflow development. If a workflow is not started, the triggering event is usually not created either.

Now switch off the event trace in Transaction SWELS again so the system doesn't write too much unnecessary information to the tracing tables (see Figure 2.17).

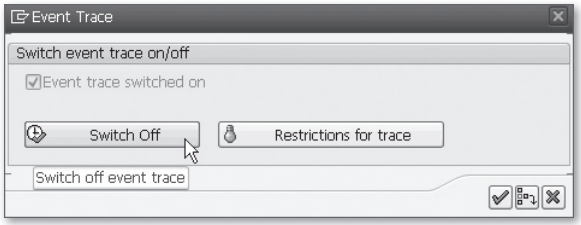


Figure 2.17 Transaction SWELS—Switching Off the Event Traces

The delivery of events can also depend on other criteria. For example, if an event is supposed to start a workflow only under specific conditions or if different workflows are to be started depending on the parameters of an event, special ABAP function modules are used.

Memory Aid: General Information on Workflows and Events

Remember the following:

- Objects the Workflow Engine uses—whether ABAP classes or BOR objects—are all programming-intensive elements.
- Events can be created directly using ABAP code.
- The delivery of events and the selection of the correct workflow can be influenced via ABAP.

2.2.2 Considering Workflow Definitions (Transaction SWDS)

In the previous section, we introduced an initial impression of the interaction of events and workflows. But how do you obtain an overview of the actual processes within a workflow?

The context menu of Transaction SWUI_VERIFY provides the option to go to the graphical presentation of the workflow template.

1. For this purpose, right-click a selected workflow template to open the context menu, and then select Display Workflow Definition (see Figure 2.18).

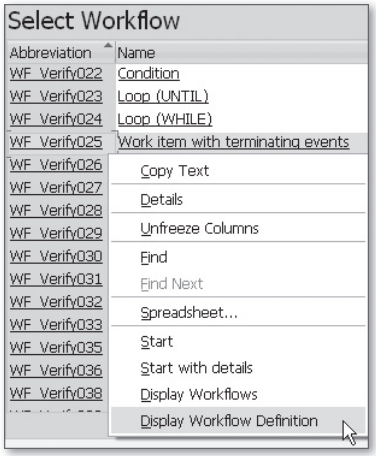


Figure 2.18 Transaction SWUI_VERIFY—View a Workflow Template Via the Context Menu

The function for the workflow display in Transaction SWDS (Workflow Builder) has the same effect.

2. In the SELECT WORKFLOW dialog, for simplified selection enter the abbreviation of the workflow template under TASK, and then press Enter (see Figure 2.19).

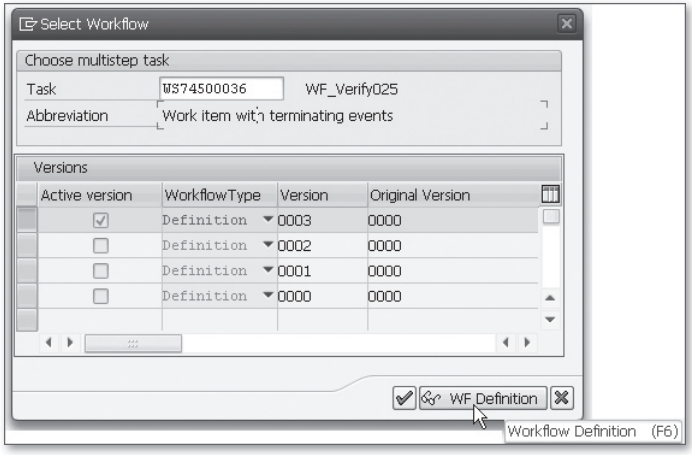


Figure 2.19 Transaction SWDS—Selection Screen

3. The system takes you to the quick view of the workflow definition of an existing workflow template (see Figure 2.20). You are already familiar with a similar view from the graphical workflow log.

However, the selection screen of Transaction SWDS (see Figure 2.19) already reveals a specific feature of the workflow development: Various versions of a workflow template can exist in the system, but only one of them is active. The idea behind it is that an already started workflow is supposed to run with the version of the workflow template with which it was started. If a new version of the template is created and activated while the business process is still running, this "update" is not supposed to lead to incompatibilities. New workflows are started with the new version of the workflow template. Old workflows continue with the old version until they are completed.

All blue boxes in the graphical presentation of a workflow template (see Figure 2.20) represent individual workflow steps.

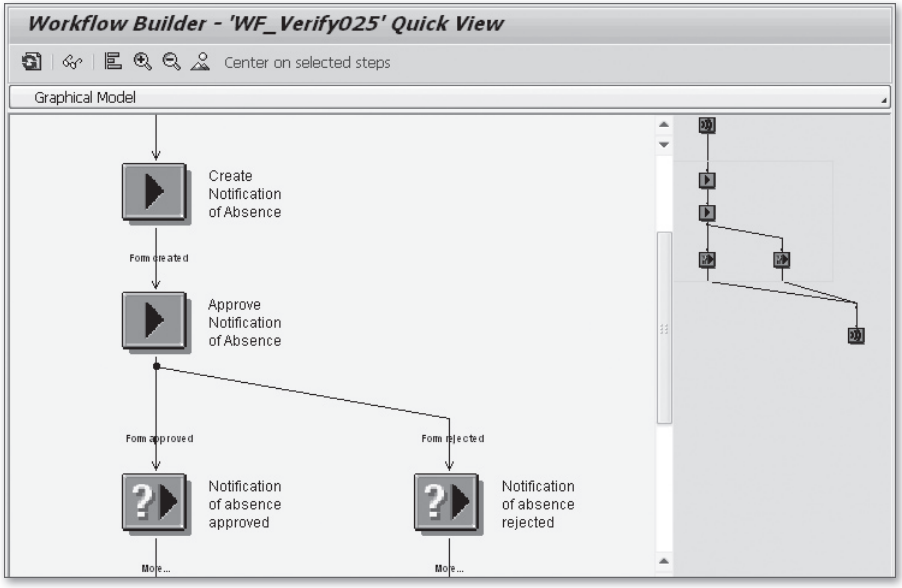


Figure 2.20 Graphical Display of the Workflow Definition

Here you can immediately recognize the philosophy of the Workflow Engine: An individual, reusable workflow step can be defined as a *standard task* and used in various workflow templates. In this example, various business processes can result in a notification of absence being entered. This could be required both in a vacation workflow and in a business trip or training workflow.

If you want to obtain further information on a workflow step, you can double-click it to navigate to the definition. Select the Approve Notification of Absence step as an example (see Figure 2.21).

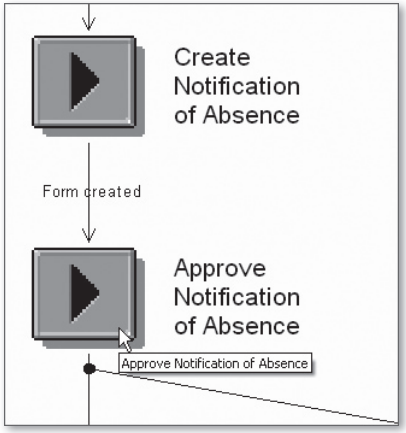


Figure 2.21 Navigating to an Individual Workflow Step

Besides other information, Figure 2.22 shows that in this workflow template the workflow step Approve Notification of Absence has the internal ID 24 and is implemented by standard task TS74507936.

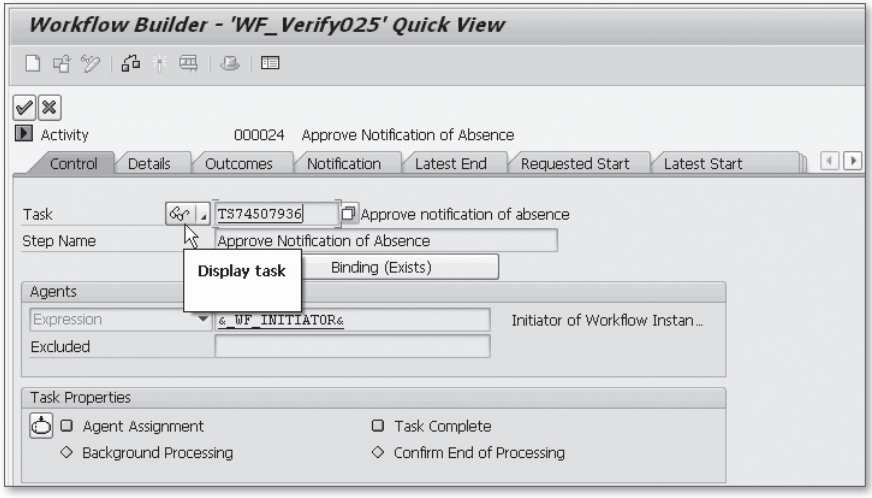


Figure 2.22 Workflow Step "Approve Notification of Absence"

On the CONTROL tab, click on the glasses icon next to the task number in order to navigate to the standard task's details.

On the BASIC DATA tab, you can see that the ABAP functionality for this step is provided by the APPROVEASYNCHRON method of the FORMABSENC BOR object (see Figure 2.23).

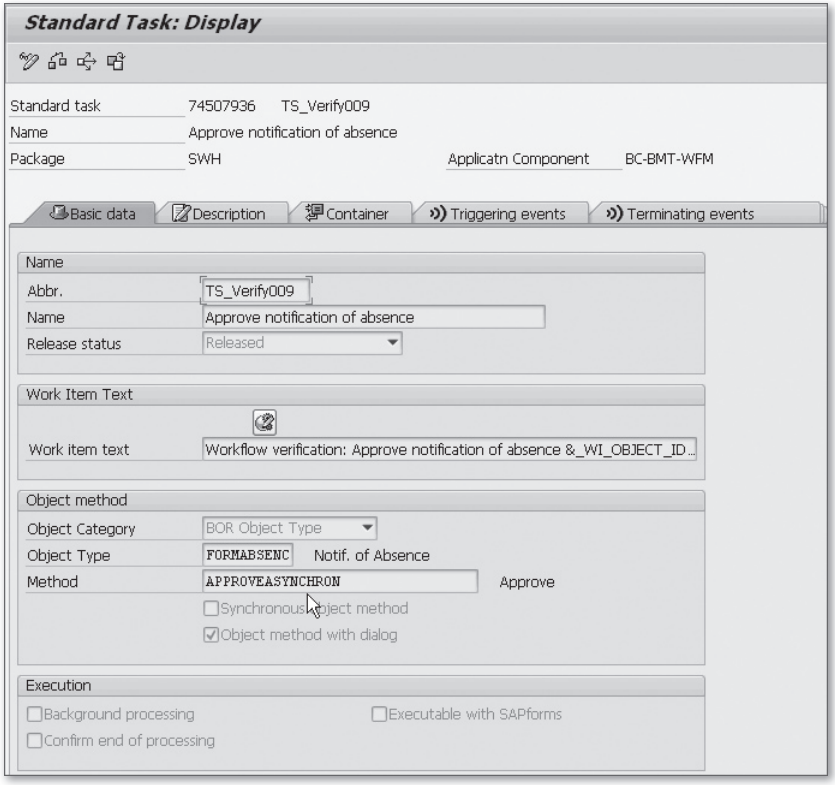


Figure 2.23 Basic Data of the Standard Task for Approving a Notification of Absence

According to the TERMINATING EVENTS tab, this standard task is terminated by the standard events FORMABSENC-APPROVED and FORMABSENC-REJECTED. You've seen at least one of these events in the event trace (see Figure 2.24).

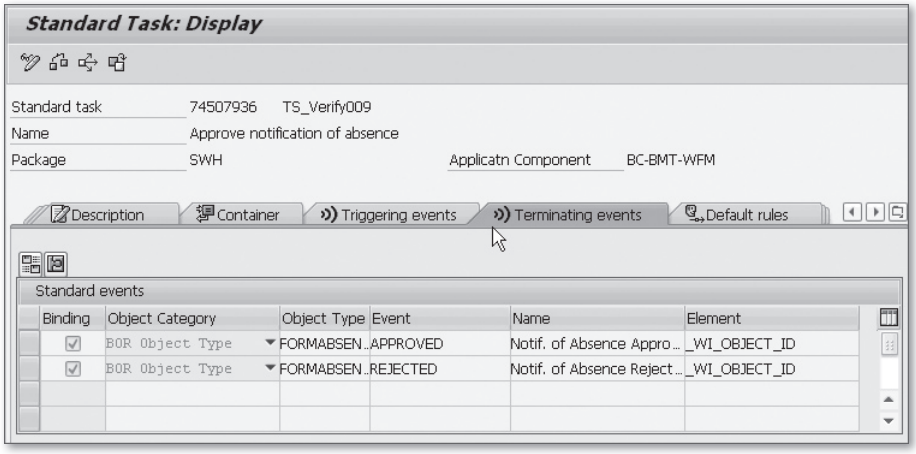


Figure 2.24 List of Events that Can Terminate the Standard Task

These examples illustrate that only the process flow itself is modeled in a workflow template. The actual SAP functionality is bundled by BOR objects or ABAP classes.

Memory Aid: Binding

Remember the following:

- You can influence how the data is transported from the workflow to the BOR objects and ABAP classes in the ABAP programming.
- The variables displayed in the work item text are based on BOR attributes. You can program them in ABAP in such a way that they are not calculated until they are queried.

2.3 Maintaining a Minimal Organizational Structure

Genuine business processes comprise a myriad of users and departments. Let's use a notification of absence as an example: There's the person placing the request, his departmental supervisor, and possibly the HR department (human resources).

Therefore, you cannot avoid summarizing and managing all of these participants of a workflow in some way or another.

Index

C

D

E

F

S

T

U

V

W

X