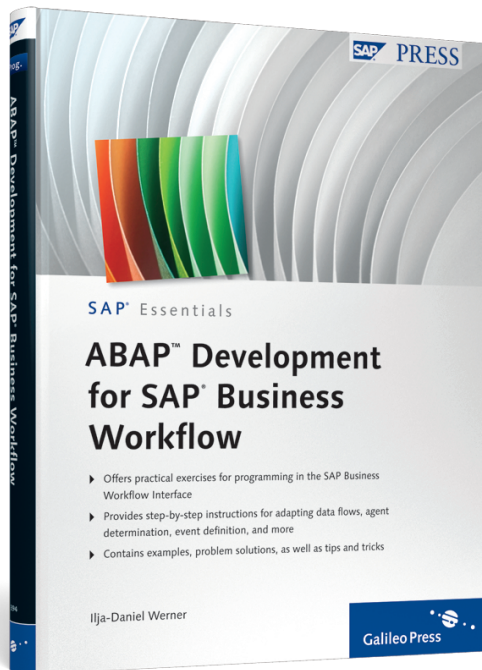


Ilja-Daniel Werner

ABAP™ Development for SAP® Business Workflow



Galileo Press 

Bonn • Boston

Contents at a Glance

1	Introduction	11
2	Getting Started	15
3	Compiling a Workflow Development Environment	43
4	Methods, Work Items, and Events	53
5	Intervening in the Agent Determination	67
6	Containers, Binding, and Conditions in the Workflow	97
7	Sample Project—Designing an ABAP Objects Class for the Workflow	123
8	Sample Project—Designing a BOR Object for the Workflow	147

Contents

1	Introduction	11
2	Getting Started	15
2.1	Customizing the Workflow Engine	15
2.1.1	Implementing Customizing (Transaction SWU3)	15
2.1.2	System User WF-BATCH	17
2.1.3	Logical RFC Destination WORKFLOW_LOCAL_xxx	18
2.1.4	Checking the Customizing (Transactions SWU3 and SWUI_VERIFY)	19
2.2	Starting Workflows and Monitoring the Workflow Events	23
2.2.1	Monitoring Events (Transactions SWELS and SWEL)	24
2.2.2	Considering Workflow Definitions (Transaction SWDS) ...	27
2.3	Maintaining a Minimal Organizational Structure	32
2.3.1	Creating Organizational Units (Transaction PPOCW)	33
2.3.2	Editing Organizational Units (Transaction PPOMW)	35
2.3.3	Assigning SAP Users to Positions	38
2.3.4	Testing the Agent Determination (Transaction PFAC) ...	40
3	Compiling a Workflow Development Environment	43
3.1	Relevant Transactions	43
3.2	Workflow Development Process with Standard SAP Functions	44
3.3	Events	45
3.3.1	Event Type Linkage	46
3.3.2	Event Instance Linkage	47
3.4	Where-Used Lists in SAP Business Workflow	48
3.4.1	From Object to Standard Task to Workflow Template	48
3.4.2	From BOR Object to Standard Task (Classic)	49
3.4.3	From ABAP Objects Class to Standard Task (Classic)	50
3.4.4	From Standard Task to Workflow Template	51

4	Methods, Work Items, and Events	53
4.1	Types of Methods in the Workflow	53
4.2	Types and Statuses of Work Items	56
4.3	Events and Their Delivery	59
4.3.1	Check and Receiver Type Function Modules	61
4.3.2	Event Queue	64
5	Intervening in the Agent Determination	67
5.1	Creating a Workflow	67
5.2	Determining and Selecting the Agent Determination Dynamically	68
5.3	Testing the Workflow Template	72
5.4	Modeling the Agent Determination Using Task Groups	75
5.5	Agent Determination with Responsibilities	81
5.5.1	Creating Rules Based on Responsibilities	83
5.5.2	Integrating a Rule with the Workflow Template	86
5.6	Programming the Agent Determination	88
5.6.1	Creating Classic Function Modules for Agent Determination	89
5.6.2	Creating an ABAP Class for Agent Determination	94
6	Containers, Binding, and Conditions in the Workflow	97
6.1	Preparation	97
6.2	Container—Location of the Data Used by a Workflow	98
6.2.1	Event Container	100
6.2.2	Workflow Container	100
6.2.3	Rule Container	101
6.2.4	Task Container	101
6.2.5	Method Container	102
6.3	ABAP-Coding with Containers (Macros)	102
6.4	ABAP Objects Classes for Handling Containers, Bindings, and Conditions	105
6.5	Coding Example for Containers, Bindings, and Conditions	107
6.5.1	Creating Container Elements	107
6.5.2	Creating Containers	109

6.5.3	Converting Containers	111
6.5.4	Creating Workflow Conditions	112
6.5.5	Defining the Binding	115
6.6	Advanced Functionality in the Binding	118
6.7	Programmed Binding	119

7 Sample Project—Designing an ABAP Objects Class for the Workflow 123

7.1	Initial Situation	123
7.2	Specific Features in the Workflow Environment	125
7.3	Creating a Class, Integrating IF_WORKFLOW, and Defining Key Attributes	126
7.4	Managing and Creating Instances	128
7.5	The Little Persistence in Between	133
7.6	Troubleshooting with Exception Classes	138
7.7	Creating Workflow Events from ABAP Objects Classes	140
7.8	BOR Objects as Attributes in ABAP Objects Classes	143

8 Sample Project—Designing a BOR Object for the Workflow ... 147

8.1	Initial Situation	147
8.2	Creating a New BOR Object	148
8.3	Creating Persistence for BOR Attributes	150
8.4	Creating Key and Other Attributes	151
8.5	Creating BOR Methods	159
8.5.1	Additional BOR Interfaces	161
8.5.2	Redefining the "CREATE" Method	161
8.5.3	Method Containers for Parameters	162
8.5.4	Redefining the "DELETE" Method	164
8.6	Exceptions and Errors	166
8.7	BOR Events	167
8.8	BOR Release Statuses	169
8.9	Default BOR Specifications	170
8.10	Inheritance and Delegation in BOR	172

Appendices	175
A Step Types and Sample Workflows	177
A.1 Step Types	177
A.2 Other Workflow Technologies	181
B Important Transactions	183
C The Author	187
Index	189

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.

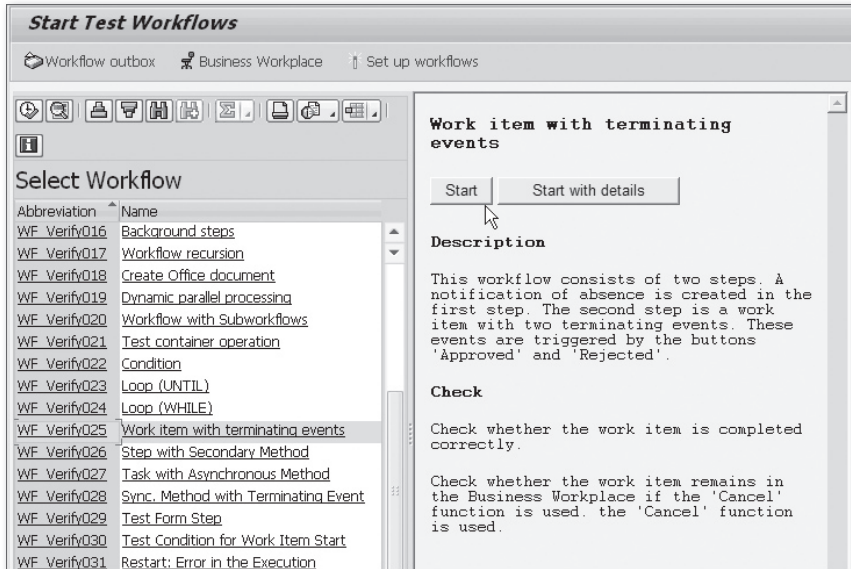


Figure 2.11 Transaction SWUI_VERIFY—Many Test Options for the Workflow Engine

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).

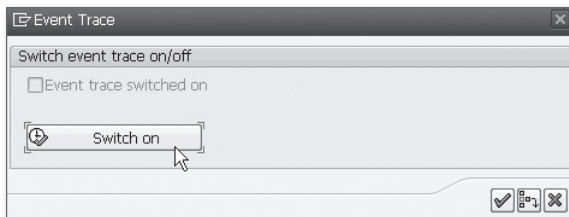


Figure 2.12 Transaction SWELS—Switching on the Event Trace

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

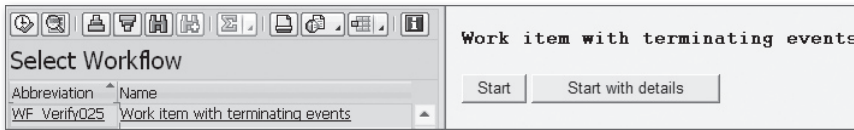


Figure 2.13 Workflow using a BOR Object for Notification of Absence Demonstrating Terminating Events

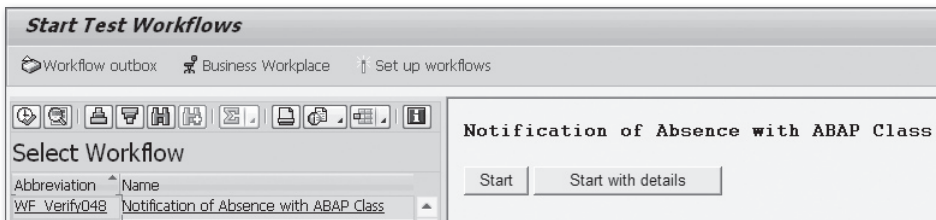


Figure 2.14 Workflow using an ABAP Class for Notification of Absence

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.

- 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).

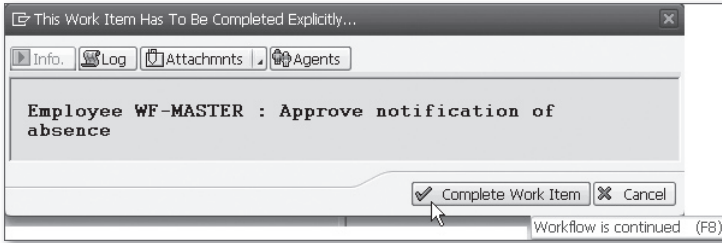


Figure 2.15 Explicit Completion of a Work Item

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).

Display Event Trace				
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).

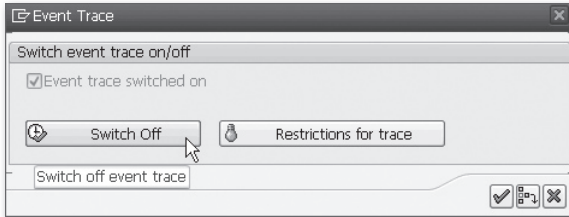


Figure 2.17 Transaction SWELS—Switching Off the Event Traces

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).

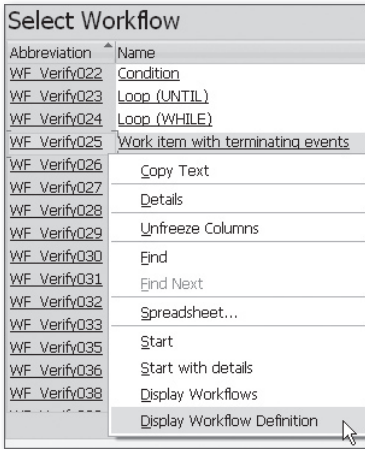


Figure 2.18 Transaction SWUI_VERIFY—View a Workflow Template Via the Context Menu

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 (see Figure 2.19).

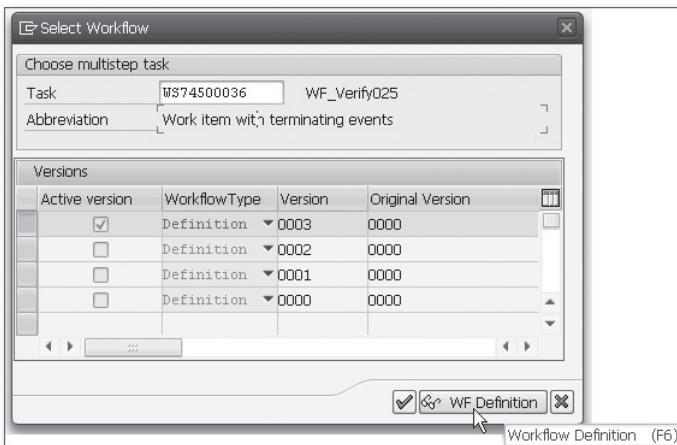


Figure 2.19 Transaction SWDS—Selection Screen

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.

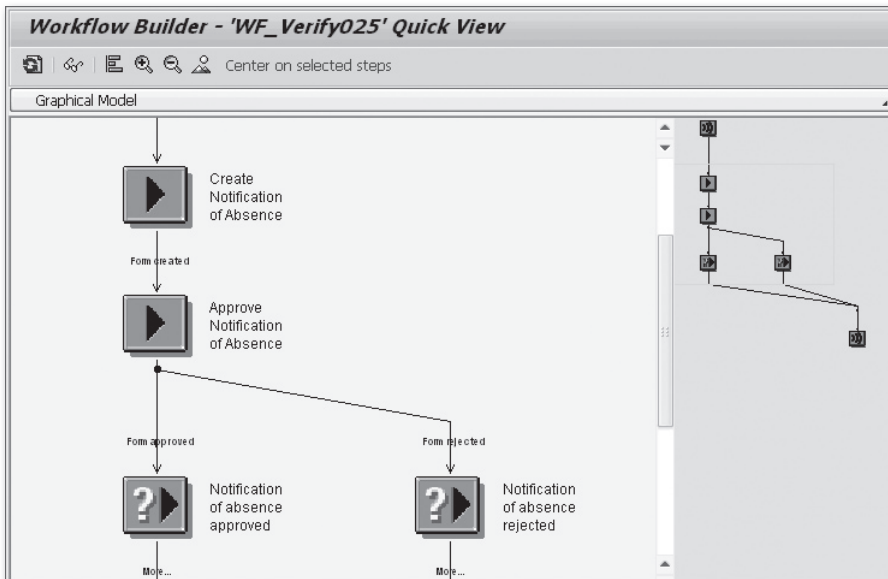


Figure 2.20 Graphical Display of the Workflow Definition

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).

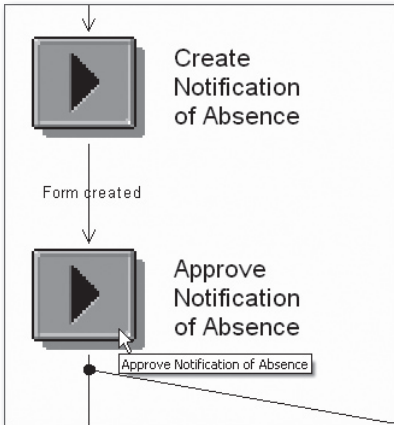


Figure 2.21 Navigating to an Individual Workflow Step

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.

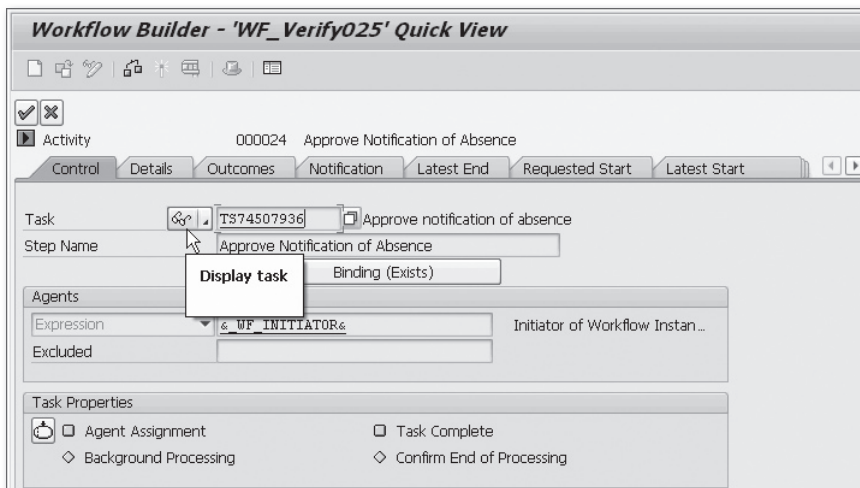


Figure 2.22 Workflow Step "Approve Notification of Absence"

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).

Standard Task: Display

Standard task: 74507936 TS_Verify009
 Name: Approve notification of absence
 Package: SWH Applicatn Component: BC-BMT-WFM

Basic data | Description | Container | Triggering events | Terminating events

Name

Abbr.: TS_Verify009
 Name: Approve notification of absence
 Release status: Released

Work Item Text

Work item text: Workflow verification: Approve notification of absence & _WI_OBJECT_ID...

Object method

Object Category: BOR Object Type
 Object Type: FORMABSENC Notif. of Absence
 Method: APPROVEASYNCHRON Approve
 Synchronous object method
 Object method with dialog

Execution

Background processing Executable with SAPforms
 Confirm end of processing

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).

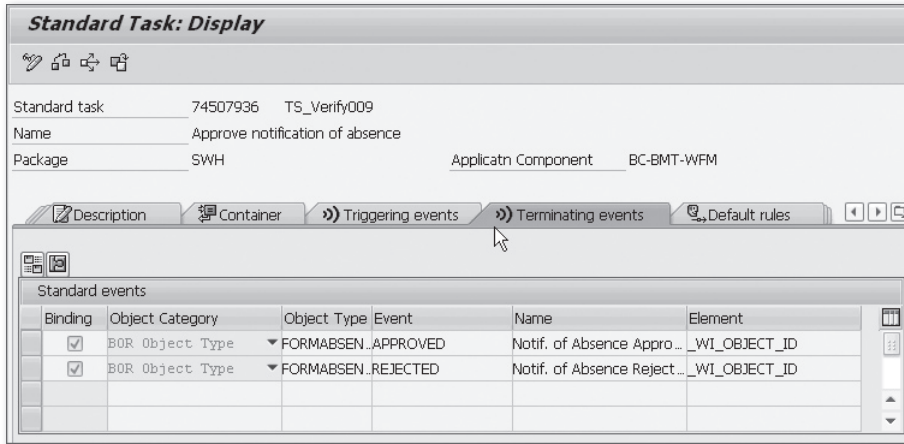


Figure 2.24 List of Events that Can Terminate the Standard Task

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

A

- ABAP Objects class, 32
 - CL_ABAP_ZIP*, 123
 - CL_SWF_BND_BINDING*, 106
 - CL_SWF_CNT_CONTAINER*, 105
 - CL_SWF_CNT_CONVERSION_SERVICE*, 106
 - CL_SWF_CNT_ELEMENT*, 106
 - CL_SWF_CNT_FACTORY*, 102
 - CL_SWF_EXP_FACTORY*, 106
 - CL_SWF_RLS_CONDITION*, 106
 - CL_XML_DOCUMENT*, 124, 148
 - persistent class*, 124
 - use in workflow*, 13, 123
- ABAP Objects interface
 - IF_SERIALIZABLE_OBJECT*, 126, 133, 135
 - IF_WORKFLOW*, 126, 127
- Advance
 - in dialog*, 25
- After method, 55
- Agent determination, 67, 68
 - dynamic*, 68, 81
 - function to be executed asynchronously*, 94
 - test (Transaction PFAC)*, 40, 42
 - via ABAP Objects class*, 94
 - via function module*, 89
 - with responsibility*, 81

B

- Before method, 55
- Before, secondary, and after method, 181
- Binding, 98
 - binding debugger*, 116
 - define*, 115
 - for workflow rule*, 87
 - programmed*, 119
- Binding editor, 80, 125
 - syntax*, 118
- BOR event, 167
- BOR interface

- IFAUTH*, 150
- IFCREATE*, 150, 161
- IFDELETE*, 150, 161
- IFFIND*, 150
- IFSAP*, 149, 159
- BOR key, 148
 - SIBFLPORB*, 148
- BOR macro
 - EXIT_CANCELLED*, 166
 - EXIT_NOT_IMPLEMENTED*, 164
 - EXIT_OBJECT_NOT_FOUND*, 160, 166
 - EXIT_RETURN*, 166
 - SWC_CREATE_OBJECT*, 159
 - SWC_GET_ELEMENT*, 163
 - SWC_REFRESH_OBJECT*, 54, 157
 - SWC_SET_ELEMENT*, 158, 163
 - SWC_SET_TABLE*, 163
- BOR method
 - Create*, 162
 - Display*, 149
 - ExistenceCheck*, 149, 158
- BOR object, 32
 - as attribute*, 143
 - database attribute*, 148, 152
 - default attribute*, 171
 - default method*, 171
 - delegation*, 172
 - GET_PROPERTY*, 158
 - inheritance*, 172
 - key field*, 151
 - reference to other BOR object*, 150
 - release status*, 169
 - self reference*, 159
 - SUBTYPE*, 172
 - SUPERTYPE*, 172
 - use in the workflow*, 147
 - use in workflow*, 13
 - variable OBJECT-KEY-*<key field>**, 157
 - variable OBJECT-*<table>**, 157
 - variable OBJECT-*<VirtualAttribute>**, 157
 - virtual attribute*, 153
 - XML_DOC*, 148, 153
- BOR release status, 169

change, 170
implemented, 170
modeled, 169
obsolete, 169
released, 169

C

Container, 97
ABAP Objects class for programming,
 105
convert, 111
create, 109
create element, 107
event container, 100
method container, 102
rule container, 101
structure SWCONT, 92
task container, 101
workflow container, 100
 Container element, 97
create, 107
initial value, 99
mandatory, 99
property, 99
_RULE_RESULT, 101
use for agent determination, 70
_WF_INITIATOR, 100
 Customizing
check, 19
start verification workflow, 19
Transaction SWU3, 15
Workflow Engine, 15

D

Database attribute
create, 152
 Deadline monitoring, 181
 Decision task, 20
 Default attribute, 171
 Default method, 171
 Delegation
link, 172
of a BOR object, 172

Dialog step
with end confirmation, 68
 Dialog work item
advance, 25

E

Error
temporary, 139
 Event, 59
check function module, 60
create (SWE_EVENT_CREATE), 167
create (SWE_EVENT_CREATE_FOR_UPD_
TASK), 167
event handler, 140
receiver function module, 60
receiver type function module, 61
start event, 60
within an ABAP Objects class, 140
 Event queue, 64
 Event queue browser, 64
 Events
terminate, 31
 Event trace, 24
activate/deactivate (Transaction SWEL(S)),
 27
Transaction SWEL, 24
Transaction SWELS, 24
 Event type coupling
flag triggering object does not exist, 168
 Exception
message class, 166
 Exception class, 138
CX_BO_ACTION_CANCELLED, 139
CX_BO_APPLICATION, 138
CX_BO_ERROR, 125, 138
CX_BO_INSTANCE_NOT_FOUND, 139
CX_BO_TEMPORARY, 125, 139

F

Forward, 78
 Function module
SAP_WAPI_START_WORKFLOW, 100
SWE_EVENT_CREATE, 167

SWE_EVENT_CREATE_FOR_UPD_TASK,
167
SWR_START_WORKFLOW, 100

G

General forwarding, 78
GUID, 124
GUID_CONVERT, 135
GUID_CREATE, 135

I

Instance management, 124, 126, 128
Interface IF_WORKFLOW
BI_OBJECT~DEFAULT_ATTRIBUTE_
VALUE, 128
BI_OBJECT~EXECUTE_DEFAULT_
METHOD, 128
BI_OBJECT~RELEASE, 128, 133
BI_PERSISTENT~FIND_BY_LPOR, 127, 130
BI_PERSISTENT~LPOR, 128
BI_PERSISTENT~REFRESH, 128, 131

K

Key structure
SIBFLPOR, 128

L

Lifecycle
instance management for ABAP Objects
class, 124
of ABAP Objects class and BOR object, 124
Line supervisor, 35
Link
delegation, 172

M

Main method, 55

MD5 hash
as ID for key structure, 124
Memory aid, 27, 32, 42
BOR editor, 159
important background jobs, 23
main and secondary methods, 55
transactions in the workflow environment,
43
Message class
for error message in the workflow
environment, 166
Method
asynchronous, 53
before, after, and secondary, 55
synchronous, 53
Method container, 162
RESULT element, 163

O

Organizational structure, 32
Organizational unit
create (Transaction PPOCW), 33
edit (Transaction PPOMW), 35
position, 36
supervisor, 35

P

ParForEach, 181
Persistence, 133
Persistence layer, 126
Persistent object reference (POR), 128
Position
SAP user, 38
Positions, 36

R

RFC destination
WORKFLOW_LOCAL_xxx, 18
Rule
based on function module, 89
based on responsibility, 83

in the workflow template, 86
 rule container, 101
 Simulate, 85
 structure SWHACTOR, 92

S

SAP Business Workplace (SBWP), 19
 SAP Note
 888279 (*Regulating the Workflow Load*),
 18
 935047 (*Creating GUIDs*), 135
 1251255 (*Authorizations for System User*
 WF-BATCH, 18
 1334035 (*Problems when Executing*
 Secondary Methods), 55
 SAP_WAPI, 58
 SAP Workflow-API, 100
 SWRC function group, 59
 SWRI function group, 58
 SWRR function group, 58
 Secondary method, 55
 Standard task, 29
 Start event, 60
 Step type
 activity, 177
 ad hoc anchor, 181
 block, 180
 condition, 178
 container operation, 178
 document from template, 179
 event creator, 178
 fork, 179
 form, 180
 local workflow, 180
 loop (UNTIL/WHILE), 179
 multiple condition, 178
 process control, 179
 send mail, 180
 subworkflow, 177
 undefined step, 178
 user decision, 178
 wait step (wait for event), 179
 web activity, 180
 Structure
 resolve by user, 42

SIBFLPOR, 123, 128
 SIBFLPORB, 148
 Subworkflow, 101
 Supervisor
 head of own organizational unit, 37

T

Task group, 75
 Temporary error
 background work item, 139
 dialog work item, 139
 Transaction
 PFAC (*Maintain Workflow Rule*), 83
 PFAC (*Test/Maintain Workflow Rule*, 67
 PFAC (*Test/Maintain Workflow Rule*), 40
 PFTC (*Maintain Workflow Tasks*), 82
 PPOCW (*Organization and Staffing*
 (Workflow) Create), 33
 PPOMW (*Edit Organizational Units*), 35
 SBWP (SAP Business Workplace), 19
 SE80 (*Development Environment ABAP*),
 127
 SE91 (*Message Maintenance*), 166
 ST22 (*ABAP Runtime Error*), 183
 SWDM (*Business Workflow Explorer*), 48
 SWDS (*Workflow Builder*), 28, 29
 SWE5 (*Consistency Check for Event*
 Linkages), 183
 SWEL (*Display Event Trace*), 26
 SWEL(S) (*Event Trace*), 24
 SWEQADM (*Administration of the Event*
 Queue), 64
 SWEQADM (*Event Queue Administration*),
 183
 SWEQBROWSER (*Event Queue Browser*),
 64
 SWETYPV (*Event Type Linkage*), 60, 62
 SWF_GMP (*Workflow Administrator*
 Overview), 183
 SWI2_ADM1 (*Work Items Without Agent*),
 183
 SWI2_ADM1 (*Work Items Without Agents*),
 95
 SWI2_ADM2 (*Work Items with Deleted*
 Agents), 183

- SWI11 (*Where-Used List for Tasks*), 51
 SWIA (*Process Work Item as Administrator*), 183
 SWO1 (*Business Object Builder*), 49, 148, 151, 161, 173
 SWU3 (*Automatic Workflow Customizing*), 15, 183
 SWUI_VERIFY (*Start Test Workflows*), 23
 SWUI_VERIFY (*Test Workflows*), 27, 67
 SWU_OBUF (*Refresh Organizational Unit*), 40
 SWUS (*Workflow Test Environment*), 72
 SWWA (*Maintain Work Item Deadline Monitoring*), 183
 SWWB (*Re-Schedule Work Item Deadline Monitoring*), 183
 SWWD (*Configure Work Item Error Monitoring*), 183
 SWWL (*Delete Work Items*), 183
 Trigger time, 45
- ## U
-
- User WF-BATCH
 background step, 54
 SAP_ALL, 17
 SAP_BC_BMT_WFM_SERV_USER, 18
 system user, 17
- ## V
-
- Virtual attribute
 create, 153
 create with data type reference, 154
- ## W
-
- Where-used list, 48
 ABAP Objects class in standard task, 50
 BOR-object in standard task, 49
 standard tasks in workflow templates, 51
 Workflow
 copy task, 77
 default attribute, 125
 default method, 125
 standard task, 29
 test environment, 72
 Workflow Builder, 29, 68
 wizard, 69
 Workflow condition
 create, 112
 Workflow definition
 display, 27
 quick view, 28
 Workflow Engine
 background job, 23
 binding, 97
 container, 97
 customizing, 15
 log, 21
 method, 53
 RFC destination WORKFLOW_LOCAL_XXX, 18
 runtime environment, 16
 self-test, 23
 SWWDHEX (*background job*), 54
 WF-BATCH (*system user*), 17
 XML persistence, 98
 Workflow event
 create in ABAP Objects class, 140, 142
 trigger time, 45
 Workflow initiator, 23
 container element _WF_INITIATOR, 69
 Workflow key structure
 SIBFLPOR, 123
 Workflow log, 21
 graphical, 21
 with technical details, 74
 Workflow macro
 include <CNTN01>, 102
 include <CNTN02> <CNTN03>, 102
 include <SWFCNTN01>, 102
 Workflow method, 53
 Workflow rule
 simulate rule resolution, 85
 Workflow step
 background step, 54
 dialog step, 54
 Workflow tasks
 decision task, 20

Index

Workflow template
 assign task group, 75
 test, 72

Work item, 56
 background, 56
 background work item, 18
 behavior, 25
 detailed view, 21
 dialog, 56
 dialog work item, 18
 explicit completion, 25
 forward, 78
 inbox, 19
 missed deadline, 56

outbox, 21
 SAP Business Workplace, 19
 (sub)workflow, 56
 wait step, 56
Work item manager, 54

X

XSLT-Transformation
 identity mapping (ID), 135