SAP PRESS

Workflow for SAP S/4HANA®

- Develop, design, deploy, and manage workflows for SAP S/4HANA
- Use the SAP Business Workflow engine and flexible workflow frameworks
- Create cloud-based workflows with SAP Build Process Automation

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

Rheinwerk Publishing

Sabyasachi Dutta, Nilay Ghosh, Kousik Goon, Sandip Jana, Arindam Mukherjee, Srinivas Rao, Yogeendar Rao, Yogesh Sane, Naveen Veshala, Kiran Viswanathan

Workflow for SAP S/4HANA®

Rheinwerk
Bonn • Boston

Imprint

This e-book is a publication many contributed to, specifically:

Editor Meagan White
Acquisitions Editor Hareem Shafi
Copyeditor Julie McNamee
Cover Design Graham Geary
iStockphoto: 155443175/© ryasick
Production E-Book Hannah Lane
Typesetting E-Book Satz-Pro, Germany

We hope that you liked this e-book. Please share your feedback with us and read the Service Pages to find out how to contact us.

The Library of Congress Cataloging-in-Publication Control Number for the printed edition is as follows: 2023948041

ISBN 978-1-4932-2428-9 (print)
ISBN 978-1-4932-2429-6 (e-book)
ISBN 978-1-4932-2430-2 (print and e-book)

1st edition 2024

Notes on Usage

This e-book is **protected by copyright**. By purchasing this e-book, you have agreed to accept and adhere to the copyrights. You are entitled to use this e-book for personal purposes. You may print and copy it, too, but also only for personal use. Sharing an electronic or printed copy with others, however, is not permitted, neither as a whole nor in parts. Of course, making them available on the internet or in a company network is illegal as well.

For detailed and legally binding usage conditions, please refer to the section Legal Notes.

This e-book copy contains a **digital watermark**, a signature that indicates which person may use this copy:

Notes on the Screen Presentation

You are reading this e-book in a file format (EPUB or Mobi) that makes the book content adaptable to the display options of your reading device and to your personal needs. That’s a great thing; but unfortunately not every device displays the content in the same way and the rendering of features such as pictures and tables or hyphenation can lead to difficulties. This e-book was optimized for the presentation on as many common reading devices as possible.

If you want to zoom in on a figure (especially in iBooks on the iPad), tap the respective figure once. By tapping once again, you return to the previous screen. You can find more recommendations on the customization of the screen layout on the Service Pages.

Table of Contents

Part I SAP Business Workflow for SAP S/4HANA

1 Getting Started

2 Introduction to Classical Workflows

3 Building Methods and Tasks

6 Agent Determination

7 Email Notifications and Other Runtime Jobs

8 Workflow Administration, Monitoring, and Troubleshooting

9 Application Link Enabling and Reporting

10 BRFplus

11 Integrating Workflows with User Interface Applications and External Applications

12 Migrating SAP ERP Workflows to SAP S/4HANA

13 Workflow in SAP Master Data Governance

13.1 Introduction to SAP Master Data Governance

13.1.1 Data Models in Central Governance

13.1.2 Change Request in Central Governance

13.1.3 Business Object BUS2250

13.1.4 Standard Dialog Tasks Used in Workflow Templates

13.1.5 Standard Dialog Tasks

13.1.6 Standard Background Tasks

13.1.7 Agent Determination

13.1.8 Workflow Container Used by Workflow Templates

13.1.9 Workflow Log for Change Requests

13.1.10 Rule-Based Workflow Template

13.1.11 SAP Master Data Governance, Consolidation and Mass Processing

13.2 Business Partner Workflows

13.2.1 Business Partner Data Model and Approach

13.2.2 Change Requests for Business Partner Master Data

13.2.3 Workflow Templates Used in Business Partner Change Requests

13.3 Finance Workflows

Part II Flexible Workflow in SAP S/4HANA

14 Introduction to Flexible Workflow

15 Custom Scenario Development

16 SAP Fiori Applications for Flexible Workflow

Part III Workflows with SAP Business Technology Platform

17 Introduction to SAP Build Process Automation

18 Process Development

20 Task Processing with My Inbox

A The Authors

Index

Preface

This preface will introduce you to the concept of workflow, its usage in the SAP world, and the various options available in SAP S/4HANA and SAP Business Technology Platform (SAP BTP). As it happens for any good book, it first talks about who should read and benefit from this book. Then, it gives a glimpse into the various workflow technologies available in SAP in the objective section. Finally, it ends with how to read this book to get the most out of it.

Target Audience

This book is for the professionals and others who are involved in the design, implementation, and administration of SAP workflows and process automation. The book is useful for a variety of roles inside organizations that already use SAP systems or intend to do so in the future. The target audience may include the following:

- **SAP consultants**
  SAP consultants specializing in workflow and process automation, including SAP workflow consultants, ABAP developers, and SAP technical consultants, seeking in-depth knowledge of SAP workflows and automation concepts

- **Business process owners**
  Individuals responsible for defining, optimizing, and managing business processes within their organizations and who wish to leverage SAP Business Workflow and/or SAP Build Process Automation for process improvement
- **SAP administrators**
  SAP system administrators and Basis consultants who are involved in setting up and configuring SAP workflows and automation solutions using SAP Workflow Management and/or SAP Build Process Automation
- **Workflow designers and developers**
  Workflow designers and developers who want to understand the intricacies of designing efficient and effective workflows using SAP tools, including the ones available as part of SAP BTP
- **SAP project managers**
  Project managers leading SAP implementation projects, especially those involving workflow automation, to gain a holistic understanding of the concepts and best practices
- **Business analysts**
  Business analysts and process analysts who work closely with business stakeholders to gather requirements and translate them into workflow automation solutions
- **SAP end users**
  SAP end users involved in approving or participating in workflow processes, who would benefit from understanding the workflows they interact with and the automation concepts

- **IT managers and decision-makers**
  IT managers and decision-makers responsible for strategizing and overseeing the adoption of SAP Business Workflow, flexible workflow, and SAP Build Process Automation in their organizations
- **SAP developers and architects**
  SAP developers and solution architects looking to integrate workflow automation into broader SAP-based solutions

Objective of This Book

The objective of the book is to give you a thorough and useful manual that equips you with the information and abilities required to design, execute, and optimize successful and effective workflow solutions inside the SAP environment. The book aims to accomplish the following objectives:

- **Understanding workflow concepts in SAP**
  The book should introduce you to the fundamental concepts of workflow in SAP. Then it goes deep into the various workflow options available in SAP such as classical workflow, flexible workflow, SAP Workflow Management (soon to be retired), and the latest option, SAP Build Process Automation. While going deep into each of these options, the book looks into the architecture, terminology, and components. It also helps you understand the usage of each of these options.
- **Workflow design and configuration**
  The book guides you through the process of designing and

configuring workflows in each of the preceding options. It covers workflow templates, steps, tasks, and decision logic, empowering you to create workflows tailored to your specific business processes.

- **SAP Business Workflow**
  For the technical readers among you, the book delves into the development aspects of SAP Business Workflow, covering ABAP coding, user exits, event triggers, and integrating custom functionality with workflows.
- **Flexible workflow**
  The book takes a deep dive into the concept of flexible workflow introduced in SAP S/4HANA, how is it different from SAP Business Workflow, why it’s called flexible workflow, what value it offers to business users, where it’s used, and so on.
- **SAP Build Process Automation**
  For those interested in process automation, the book explores automation techniques and best practices, using SAP Build Process Automation as a tool to streamline and optimize business processes.
- **Workflow integration with SAP applications**
  The book demonstrates how to integrate SAP Business Workflow with various SAP applications, such as SAP S/4HANA, SAP Fiori, and SAP BTP, to enable seamless end-to-end business processes. Advanced topics such as parallel processing, escalation handling, dynamic decision routing, and event-driven workflows are covered to address complex workflow scenarios.
- **Best practices and optimization**
  The book provides you with best practices to ensure

efficient workflow design, performance optimization, and tips for troubleshooting and error handling.

- **Latest trends and innovations**
  To keep you up to date, the book covers the latest trends, innovations, and future road map for each of the different variants of the workflow technology in SAP, including the cloud-based workflows and integration with emerging technologies such as machine learning.

How to Read This Book

Reading a book on workflows can be a rewarding experience if you approach it systematically and with clear objectives. Here are some steps to help you make the most out of reading the book on workflows:

- **Start with Chapter 1**
  Begin by reading the first chapter of the book, which provides an overview of the topics covered throughout the entire book. It will give you a sense of what to expect from the book.
- **Set learning objectives**
  Before diving into the book, determine your specific learning objectives. Are you looking to understand the basics of various workflow technology options in SAP, learn about advanced concepts, or explore workflow integration with other SAP applications? Having clear objectives will help you focus your reading and retain information better.

- **Skim through chapters**
  Take a quick look at the table of contents and skim through the chapters. This will give you an idea of the book’s structure and organization. Identify chapters that are most relevant to your learning objectives, and start with those.
- **Read chapters in a logical sequence**
  Read the chapters in a logical sequence to build a solid foundation. Start with the basics and gradually move to more advanced topics. Ensure you understand each concept before proceeding to the next chapter.
- **Take notes**
  Keep a notebook or use digital note-taking tools to jot down important points, key terms, and concepts you find valuable. Taking notes will aid in retention and serve as a reference for future use.
- **Practice with examples**
  Most chapters include practical examples. Work through these examples to gain hands-on experience and reinforce your understanding.
- **Review frequently**
  Periodically review the chapters you’ve read to reinforce your understanding. This will help you retain information for the long term.
- **Consult SAP documentation and other resources**
  As you progress through the book, consult SAP official documentation and additional resources such as blogs, videos, and so on to supplement your learning and gain further insights.

- **Engage in discussion and communities**
  Engage in discussions with peers, colleagues, or online communities related to workflow options in both SAP S/4HANA and SAP BTP. Sharing insights and discussing concepts with others can enhance your understanding.
- **Apply learnings in real projects**
  If you have the opportunity, try to apply what you've learned in real-world workflow projects. Practical application will solidify your knowledge and boost your confidence.
- **Ask questions**
  If you encounter any doubts or questions while reading, don't hesitate to seek answers from SAP experts, forums, or community platforms.

Remember, reading this book is just the first step in your learning journey. Practical experience and continuous learning are essential to master the various options of workflow technology and process automation concepts in SAP effectively.

Part I

SAP Business Workflow for SAP S/4HANA

This part will teach you how to create, maintain, and customize classic workflows using the SAP Business Workflow engine in SAP S/4HANA. You will also learn about migrating workflows to the new environment, designing a UI, and external system integration.

1 Getting Started

This chapter will introduce you to workflows in SAP S/4HANA. You'll get an overview of the workflow options both within SAP S/4HANA and those available via SAP Business Technology Platform (SAP BTP). The chapter also explores workflow development tools and offers a glimpse of the key roles and authorizations required in each environment before you can design workflows.

In this chapter, we'll discuss how workflow evolved from SAP ERP to SAP S/4HANA and some highlights of the new capability SAP S/4HANA brings in the workflow area. SAP Business Workflow provides a capability to automate the business process steps into an orchestrated coherent single end-to-end business process in your organization. It improves the efficiency and performance of the process and reduces the cycle time to complete the business process. SAP Business Workflow isn't just a technical capability of a system; it's built around human activities, so it's very important to consider the human factor when implementing any workflow solution. We'll also consider the similar areas of evolution in workflow context when we see how SAP Business Workflow evolved from SAP ERP to SAP S/4HANA. We'll provide an overview of the new capabilities SAP S/4HANA brought to workflows. With so many options, it's

sometimes confusing what to use when. We'll provide you with guidance on what SAP S/4HANA workflow capabilities can be used where.

1.1 Workflows in SAP S/4HANA

The classic workflow was the only option in SAP ERP to develop any business workflow. It's still available in SAP S/4HANA and supported, but there are now two new ways of workflow development: flexible workflow and workflow in SAP Business Technology Platform (SAP BTP). We'll cover all three types of workflow development in this book. We'll start with classical workflows, and then we'll discuss further on the flexible workflow and workflow on SAP BTP. In the classical workflow, we'll focus mostly on the capability used and provide more details on the flexible workflow and workflow on SAP BTP.

1.1.1 Evolution of Workflows from SAP ERP to SAP S/4HANA

There are three main aspects of any workflow development:

- Workflows can integrate different business transactions to improve the end-to-end business process experience and performance.
- Workflows manage people's actions, so the user interface (UI) is always a key attribute of any workflow development.

- Workflows are always about the business process flow. Business user involvement is needed during build and maintenance of workflows.

There are significant changes in these three aspects of workflow from SAP ERP to SAP S/4HANA, so let’s discuss how workflows have evolved in these areas:

- **Integrating the process steps across systems**
  SAP Business Workflow started with integrating the business transactions within the same application in SAP ERP. Normally, the business scenarios are limited within one system. There were simpler use cases where any specific functions are called from other applications, or another workflow can be triggered. But modern end-to-end business processes cut across different systems; they don’t stay within the boundary of SAP ERP or one application. You can now implement workflows across different SAP and non-SAP applications in SAP BTP. We’ll discuss this in more detail in Section 1.2.
- **UI to process the work items**
  The UI has changed from SAP ERP to SAP S/4HANA. SAP Fiori is the new recommended UI in SAP S/4HANA instead of SAP GUI in SAP ERP, so you have to keep in mind the new UI of SAP S/4HANA when using the classical workflow (discussed in detail in Chapter 11). The way users access any work item to process it has also evolved over time. It started with Business Workplace, which is still available in SAP S/4HANA. But considering all the new capabilities of a platform like SAP BTP and the requirement to have one common inbox for all kinds of work items, it isn’t used if SAP S/4HANA is implemented with the modern capability.

The universal worklist of SAP Enterprise Portal was used as single entry point for all work items in SAP ERP. Now SAP Task Center or the My Inbox app is used in the context of SAP S/4HANA. It can be a single user inbox where user can access all SAP work items across the SAP applications. including public cloud, on-premise applications, and SAP BTP.

- **Easily configurable workflow**
  Flexible workflow was introduced in SAP S/4HANA as a preconfigured workflow for some common business cases that business users can configure themselves. The preconfigured workflows may not always meet your business requirements. You may have to engage developers to enhance these workflows, but the preconfigured workflows will at least provide a starting point where the business or functional user can perform the initial configuration and simulate the business scenario. SAP Fiori-based agent determination and BRFplus rules (integrated with workflows), also provide more options for business users to maintain the workflow. The workflow log available in My Inbox is user friendly. Unlike in SAP ERP where workflow log can only be understood by workflow developer, it’s simpler in SAP S/4HANA, and anyone will be able to understand the workflow approval history.

In SAP ERP, the workflow was developed in a classical way. Now, in SAP S/4HANA, there are three ways to develop workflow applications:

- Classical workflow in SAP S/4HANA
- Flexible workflow in SAP S/4HANA

- Workflow in SAP Build Process Automation on SAP BTP

Let’s discuss the scenarios for when to adopt which workflow build approach when you’re developing a new workflow (see Table 1.1). We’ll also consider the availability of the technology in three deployment platforms available for SAP S/4HANA: on-premise SAP S/4HANA; SAP S/4HANA Cloud, private edition; and SAP S/4HANA Cloud, public edition.

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
Classical workflow	The classical workflow is still available and supported in SAP S/4HANA. End users process these work items using Business Workplace (Transaction SBWP) or universal worklist in SAP Enterprise Portal. In the SAP S/4HANA classical workflow, work items are compatible with the My Inbox app, and all features are available, including building new custom workflows. It’s recommended to look for a flexible workflow	Available in on-premise SAP S/4HANA and SAP S/4HANA Cloud, private edition (not available in SAP S/4HANA Cloud, public edition)

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
	first in SAP S/4HANA. If there is no flexible workflow, but the standard classical workflow is available, then use the classical workflow. Any business process enhancement of the classical workflow should be done with the SAP developer extensibility framework. In-app extensions via business add-ins (BAdIs) are available for custom conditions, custom agent rules in the responsibility management framework, and scenario-specific extensions (e.g., procurement). The following help document discusses predelivered SAP workflow scenarios: *http://s-prs.co/v569700*.	

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
Flexible workflow	Flexible workflow is available in SAP S/4HANA, and predelivered flexible workflows are available. The basic functionality and agent determination are configurable. The Manage Workflows app from SAP Fiori is used for configuring predelivered flexible workflows. The Teams and Responsibility Management app is used for agent determination. Flexible workflow is built on the same classical workflow engine. The workflow log transactions can be used to view the flexible workflow as well. But the classical Workflow Builder Transaction SWDD won’t work for flexible workflow. Transaction SWDD_SCENARIO is used to create the scenario for flexible	Available in on-premise SAP S/4HANA; SAP S/4HANA Cloud, private edition; and SAP S/4HANA Cloud, public edition

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
	workflow. Flexible workflow is easy to configure and is highly integrated with SAP Fiori apps and the new user experience. This is the preferable approach for local SAP S/4HANA workflow deployment.	
SAP Build Process Automation	SAP Build Process Automation on SAP BTP is the strategic solution for process orchestration and workflow development with process insight. SAP Build Process Automation is discussed in detail in Part III from Chapter 17 onward. Here are some of the highlights of its capabilities: • Integrates with SAP and non-SAP applications to orchestrate	Implemented on SAP BTP integrated with any on-premise and cloud applications (e.g., SAP Ariba; SAP SuccessFactors; and SAP S/4HANA Cloud, public edition)

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
	seamlessly across applications • Integrates with the SAP Intelligent Robotic Process Automation functionality and process visibility with SAP Signavio Process Insights and Qualtrics for Experience Management • Provides hundreds of prebuilt content items to jump-start any process automation development	
SAP Build Process Automation (Cont.)	• Gives you easy-to-use configuration components to develop workflow process, approval forms, task automation, and business rules for decisions	

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
	• Opens with a unified user inbox launchpad called the SAP Task Center with easy-to-consume graphical process visibility • Enables simple workflow processes to be built with low-code app development, and provides full build capability for complex developments. Here are some of the scenarios where it should be used: • Workflow for process that spans multiple applications, SAP or non-SAP. • End-to-end workflow that needs to be integrated with the capabilities of SAP Intelligent Robotic Process Automation,	

Workflow Development Approach	When to Adopt Recommended Scenario	Deployment Platform
	SAP Integration Suite, SAP Signavio Process Insights and sentiment analysis, or SAP Build Process Automation. If any workflow—whether within one system or across multiple systems—needs other automation or analytical capabilities, SAP Build Process Automation should be considered.	

Table 1.1 Workflow Development Approaches Available in SAP S/4HANA

1.1.2 Workflow Options

As mentioned in the previous section, classical workflow and flexible workflow are mainly used with SAP S/4HANA systems. The workflow on SAP BTP is used mainly to stich processes across multiple systems with some exceptions. We'll now discuss classical and flexible workflow in this section. Workflow on SAP BTP will be discussed in Section 1.2.

Classical Workflow

With the evolution of SAP ERP into SAP S/4HANA, the workflow development options have evolved significantly as well. Developers get to use the new flexible workflow option with the scenario approach (standard and custom) in a process running within the SAP S/4HANA system only, or, in case of a distributed environment scenario with processes running across integrated systems, they may choose to develop an SAP BTP workflow. However, the good old classical option is still available in SAP S/4HANA as well. This option should be explored only when your process requirements can’t be met with flexible workflow.

Classical workflows offer the maximum flexibility in terms of design and development due to the huge inventory of tools and options available to the developer. However, they also usually involve significant development efforts and incur more costs to the customer to develop and support. They also require a lot of GUI-based transactions to develop, configure, and monitor, in contrast to flexible workflow, which uses SAP Fiori apps to perform most development tasks. While designing a classical workflow, you need to consider the following key elements:

- **How to trigger the classical workflow**
  Usually, you’ll use the business object repository (BOR) or class-based events to trigger a workflow. Therefore, the first step is usually to identify if a standard event is already raised by the application that you’re working with. If a standard event already exists, then the business object type or the ABAP class triggering this event is probably the right choice for the leading object type of

your workflow. If no such event exists in standard, then you may need to develop your custom ABAP class or BOR type and configure a custom event to be raised under the right conditions.

- **Workflow design**
  If a standard event already exists, then the next logical step is to identify if a standard workflow exists as well that suits your requirements. In most cases, the standard workflow may not suffice and even if it did, you would probably need to send your own subjects and texts on the work items and notifications. Therefore, you would probably need to copy the standard workflow and customize it. If no standard workflow exists, then you would need to develop a custom one using the Workflow Builder. Start with a high-level design of the process just like a flow diagram, and then expand on each building block with more details. Any custom business logic required in your workflow may be developed using BOR methods or ABAP class methods using the leading object type chosen in the previous step. You may sometimes need to extend a standard BOR type using a subtype to add your custom logic in a workflow.
- **Agent determination**
  This is usually a complex topic as the agent determination technique for the same workflow may vary from one customer to another. First you should check for any standard options available for the application such as a configurable table, rule, or responsibility. If you go for the custom approach, then you can develop your own rule using the classical responsibility approach, integrate with a BRFplus application, or use the HR organization

structure if possible. If none of these options work, then you can develop your own custom table(s) and a method or a rule to read from the same.

- **Approvals and notification requirements**
  Consider the requirements of the workflow users in terms of work item approvals and notifications. For work items, the common access point is the My Inbox app. What kind of frontend UIs are required for your workflow? The classic user decision task is still the most-used frontend for approvals. But if the users need to launch an SAP Fiori app through My Inbox, then you may need to develop a task with SAP Fiori integration. Similarly, for notifications, you send emails to the user's email address maintained in the user master. You may need to think about extended notifications for work items and approval, rejection, and deadline notifications per the business requirements.

Figure 1.1 illustrates a simple process flow design using the classical workflow approach.

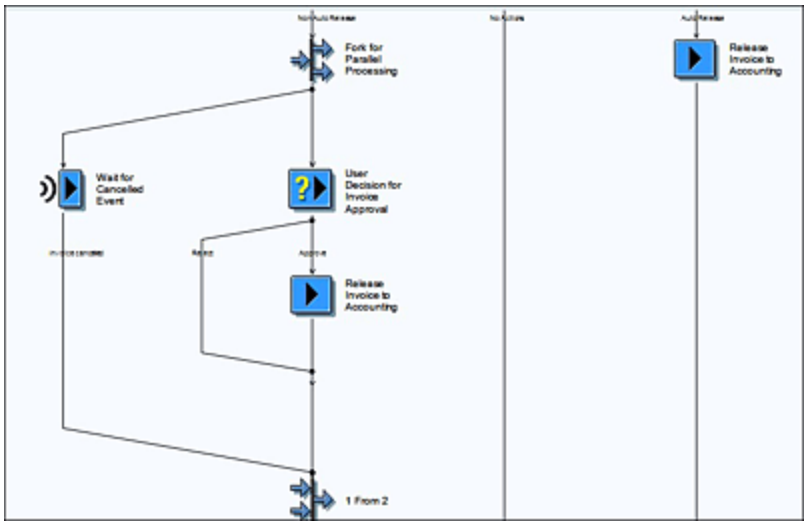


Figure 1.1 Sample Workflow Definition Using the Classical Workflow Approach

Flexible Workflow

Developing workflows by using the flexible workflow framework is the recommended option in SAP S/4HANA systems. SAP introduced a new way of creating workflows in SAP S/4HANA systems with the no-code/low-code option. Flexible workflow is a framework introduced in SAP S/4HANA as part of the SAP S/4HANA workflow engine. It allows developers to create simple workflow scenarios and allows functional/business process specialists to model the process flow very easily using SAP Fiori apps.

It's built on top of the existing classical workflow framework. By using this framework, you can make use of standard flexible workflow scenario templates and easily tailor those per your custom requirements. These workflow templates are closely integrated with email templates to send notifications to agents/approvers.

SAP tightly integrated the flexible workflows approval process with the My Inbox app. All work items will be routed to the My Inbox app, so approvers can check their work items in their My Inbox app and take appropriate actions based on the workflow setup. You can learn all the details about flexible workflows in Chapter 14, Chapter 15, and Chapter 16.

Let's discuss some features of these local SAP S/4HANA instance workflows by considering on-premise SAP S/4HANA; SAP S/4HANA Cloud, private edition; and SAP S/4HANA Cloud, public edition. On-premise SAP S/4HANA and SAP S/4HANA Cloud, private edition, provide similar features. So, in Table 1.2, let's compare how some workflow

build features will work in SAP S/4HANA Cloud, private edition, versus SAP S/4HANA Cloud, public edition.

Feature	SAP S/4HANA Cloud, Private Edition	SAP S/4HANA Cloud, Public Edition

Feature	SAP S/4HANA Cloud, Private Edition	SAP S/4HANA Cloud, Public Edition
User inbox to process work items	Both Business Workplace and the My Inbox app are available, but Business Workplace isn’t recommended. The My Inbox app should be used for all kinds of work item processing and notification management. My Inbox supports work items created from both the classical workflow and flexible workflow. SAP Task Center also can be considered as a unified user inbox. Standard substitution features are available. You can also see the workflow log from the inbox or outbox. The workflow log view is much simpler and more user friendly, but the graphical log isn’t available.	My Inbox and SAP Task Center should be used in SAP S/4HANA Cloud, public edition. All features are the same as in SAP S/4HANA Cloud, private edition.

Feature	SAP S/4HANA Cloud, Private Edition	SAP S/4HANA Cloud, Public Edition
Workflow development approach	Flexible workflow is recommended. Classical workflow is also supported. If any business requirement can’t be fulfilled by flexible workflow, then classical workflow should be considered. Workflow developed in the classical way can’t be displayed in the Workflow Builder for flexible workflow. Flexible workflow can’t be displayed in classical workflow tools. The Manage Workflow app is used for flexible workflow configuration, and Transaction SWDD_SCENARIO is used for scenario building.	Classical workflow isn’t available for customer implementation. Flexible workflow is used instead. Each application is configured separately with their scenario IDs.

Feature	SAP S/4HANA Cloud, Private Edition	SAP S/4HANA Cloud, Public Edition
Agent determination	Classical organizational unit–based agent determination is still available, but it’s not recommended. The Terms and Responsibilities app is available where you can define the responsible persons for a specific company code, plant, and so on. You can also use the agent rule and write custom code to determine the agent. The Terms and Responsibilities app is supported for both classical and flexible workflows.	Classical organizational unit–based agent determination isn’t available. The Terms and Responsibilities app should be used for agent determination.

Feature	SAP S/4HANA Cloud, Private Edition	SAP S/4HANA Cloud, Public Edition
Workflow development capability	Flexible workflow doesn’t have parallel processing capability compared to classical workflow. Exceptions are handled using a modeled outcome and flow in classical workflow. It’s handled using code in the Manage Workflows app for flexible workflow. All the tasks created in flexible workflow are automatically set as general tasks.	Custom scenarios should be developed using SAP Build Process Automation in SAP BTP. The standard scenario supports all flexible workflow capabilities.

Table 1.2 Comparison of Different Workflow Capabilities Available in SAP S/4HANA Cloud, Private Edition, and SAP S/4HANA Cloud, Public Edition

1.2 Workflows in SAP Business Technology Platform

Organizations can build, automate, and manage workflows in a cloud environment with the help of SAP Workflow Management, a cloud-native service offered by SAP. It offers a low-code/no-code approach, allowing both business users and developers to create, alter, and automate workflows without having substantial coding experience.

While SAP Workflow Management is a powerful tool for automating and managing workflows in a cloud environment, like any technology, it has some limitations as well such as handling complex workflows, complex integrations, performance, handling high data volumes, offline support, automation, and so on. In addition, it was found that the learning curve is high in practical cases. SAP has announced the retirement of SAP Workflow Management and has introduced a more modern, comprehensive, and “citizen friendly” tool called SAP Build Process Automation. The key differences between SAP Workflow Management and SAP Build Process Automation are summarized in Table 1.3.

SAP Workflow Management	SAP Build Process Automation
Requires programming knowledge and therefore more of a pro-code solution	Truly a no-code solution that uses visual programming friendly for citizen developers

SAP Workflow Management	SAP Build Process Automation
Needs experience in the pro-code tool SAP Business Application Studio to design, develop, and deploy	Easy to design, develop, and deploy through the integrated build lobby
SAPUI5-based start and task UI	Simple forms easily created using the no-code forms with SAP Build Apps and SAPUI5 used for more complex forms
Uses untyped interfaces	Uses typed interfaces and data types
Local My Inbox at the subaccount level	SAP Task Center: one inbox for all workflow tasks
Different versions for artifacts in the same project	One version for all artifacts in a project
Requires different monitoring applications for different artifacts used in the solution	Integrated monitoring of all project artifacts
Needs separate license of SAP Intelligent Robotic Process Automation to handle automation	Automation capability integrated and handled through automation artifact

SAP Workflow Management	SAP Build Process Automation
Requires additional SAP Build Work Zone standard license	SAP Build Work Zone included for SAP Build Process Automation usage

Table 1.3 Comparison between SAP Workflow Management and SAP Build Process Automation Capabilities

Because SAP Workflow Management capability has been replaced with SAP Build Process Automation, the direction forward for SAP Workflow Management customers is to build new workflows in SAP Build Process Automation. Existing workflows built in SAP Business Application Studio can be consumed in SAP Build Process Automation. SAP has provided some migration options. However, there are some constraints in these migration options. While the SAP Workflow Management solution was available in SAP's own data centers (for Neo) and various hyperscalers (for Cloud Foundry), SAP Build Process Automation is available only in limited Cloud Foundry data centers.

Let's take a quick look at the process of moving to SAP Build Process Automation, as follows:

- **Workflows**
  Though the ideal solution would have been to get the existing workflows developed using SAP Workflow Management converted easily, unfortunately it isn't possible. Due to a different internal architecture, existing workflow models won't be converted automatically into the process builder in SAP Build Process Automation; rather, the new workflows should be built in SAP Build

Process Automation. Workflows built in SAP Business Application Studio, however, can be modified in SAP Business Application Studio and deployed to SAP Build Process Automation.

- **Business rules**
  Business rules projects from an SAP Workflow Management subaccount have to be imported into an SAP Build Process Automation subaccount and need to be redeployed as rule services in an SAP Build Process Automation subaccount. The user and role assignments within the SAP Build Process Automation subaccount will also be needed for further modification, deployment, and management of these rules. The business rules directly deployed to SAP S/4HANA need to be recreated as projects in the SAP Build Process Automation subaccount before they can be used.
- **Process visibility scenarios**
  The visibility scenarios from the SAP Workflow Management service need to be imported into a business process project and then be redeployed. Once imported successfully, they can be adjusted just like the visibility scenarios created as part of the business process project in SAP Build Process Automation.

Chapter 18 is dedicated to both SAP Workflow Management and SAP Build Process Automation where you'll see the components of both solutions in more detail.

1.3 Workflow Development Tools

We'll now briefly talk about the workflow development tools used for the three types of workflow discussed in the earlier sections: classical workflow, flexible workflow, and SAP Build Process Automation. We'll discuss the toolset used in these types of workflows, as well as the BRFPlus tools here. BRFplus is heavily used in the decision matrix and agent determination of workflows.

1.3.1 Tools for Classical Workflow

Classical workflow uses several development tools, which remain mostly unchanged with on-premise SAP S/4HANA, SAP S/4HANA Cloud, and SAP S/4HANA Cloud, private edition. In SAP S/4HANA Cloud, public edition, the classical workflow isn't available as a development option, so our discussion in this book will always refer to on-premise SAP S/4HANA and SAP S/4HANA Cloud, private edition, in the context of the classical workflow. Following are some of the commonly used definition time and runtime tools in classical workflow:

- **Workflow Builder**
  This is the tool used to create a workflow definition. A workflow definition describes the workflow process and consists of the following:
  - Basic data
  - Information about triggering events

- Initial values
- Containers
- Bindings

Workflow Builders may be accessed directly by using Transaction SWDD or using Transaction PFTC (select a multistep task or workflow template ID), followed by selecting the **Workflow Builder** button on the **Basic Data** tab. You can also access Workflow Builder from menu path **Tools** • **Business Workflow** • **Development** • **Definition Tools** • **Workflow Builder**. Details of designing a workflow via the Workflow Builder can be found in Chapter 4. Figure 1.2 shows a view of the Workflow Builder in SAP with different development and navigation options:

❶ **Information Area**: Displays the current workflow, version, and status information.

❷ **Navigation Area**: Allows you to navigate between different steps in the workflow.

❸ **My Workflow and Tasks** toolbox dropdown with multiple options: Allows you to switch between various options such as **Workflow Containers**, **My Workflow and Tasks**, **Step Types That Can Be Inserted**, and so on.

❹ **Message** area with navigation objects: Displays warning and error messages after the syntax check and allows the user to navigate to the step with the issued message.

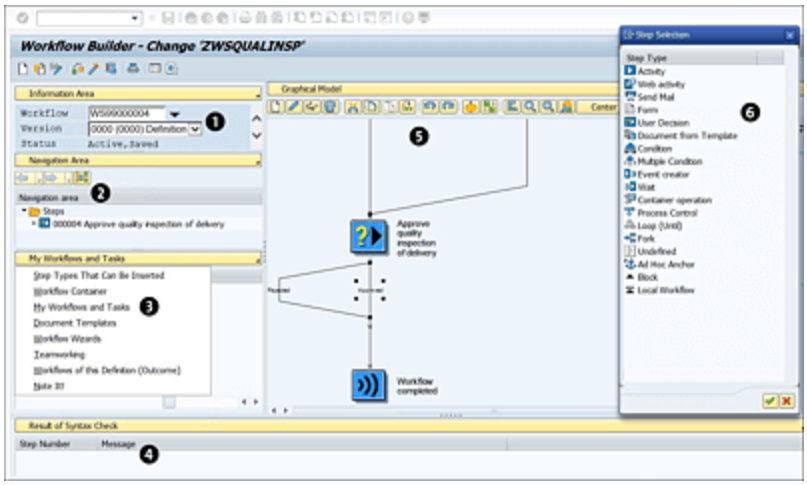


Figure 1.2 View of Workflow Builder with Different Development and Navigation Options

❺ **Graphical Model**: Displays the graphical workflow.

❻ **Step Selection**: Displays the step types that can be created in the workflow.

- **Business object builder and class builder**
  These two tools are used to develop the business object layer of the workflow definition. The business object builder can be accessed via Transaction SWO1 and the class builder via Transaction SE24. Both these transactions enable the developer to build the methods behind the tasks and activity steps of a workflow. Details of the business object repository (BOR) builder and class builder in the context of workflow development may be found in Chapter 3.

- **Tasks and task groups**
  This tool may be accessed via Transaction PFTC and is used to define the various attributes of a task definition such as the basic data, which includes the object method details and task attributes along with binding, task long text, container definition, triggering events, terminating events, and default rules. The same transaction may be used to access a workflow definition as well. Here, the Basic Data tab captures the workflow title text along with a link to the Workflow Builder. Additional tabs include details on long text, container definition and triggering events. Further details on this tool can be found in Chapter 3, Section 3.3.
- **Rules**
  Rules are one of the primary mechanisms of agent determination in workflows. The rule builder is a tool that may be accessed via Transaction PFAC. It provides the option to define a rule using various options such as responsibilities, organization structure, function module, and so on. Depending on the selected category you can enter an appropriate object, define container elements to be used in the rule, and add long texts.
- **Organization management**
  If you want to integrate the HR organization structure with your agent determination process, then you need to maintain the organization structure for workflow. This can be done through the organization plan maintenance tools via Transaction PPOSW and its variants.
- **Events**
  Similarly, there is a list of tools for maintaining the event

creation and event linkage for a workflow. This includes event creation via change documents, status management, Transaction NACE output control, and HR master data changes. These event-related tools can be accessed via menu path **Tools** • **Business Workflow** • **Development** • **Definition Tools** • **Events**.

- **Test workflow**
  Transaction SWUS can be used as a testing tool for workflows. In this transaction, you can trigger a workflow manually without the use of events after populating the required container elements. Figure 1.3 shows an example of single testing a workflow via Transaction SWUS.

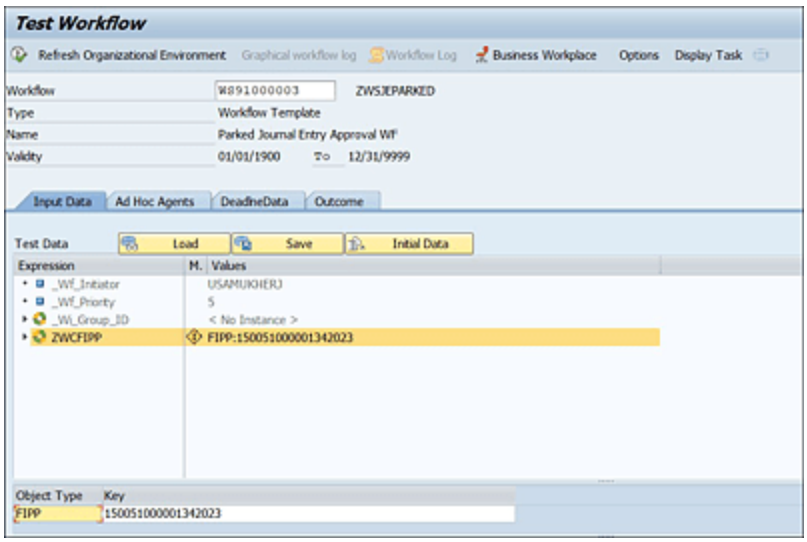


Figure 1.3 Example of Workflow Single Testing via Transaction SWUS

- **Workflows for object**
  Another commonly used runtime monitoring tool for workflows is Transaction SWI6, which lets you find workflows based on the leading object instance. Here, you must enter the BOR or ABAP class **Object Type** and the **Key** field(s). Additional filters include the workflow status group in the **Selection variant** field (**Active**, **All**

Instances, and **Completed**), **Task** ID, and **Selection Period**. This tool is pretty useful when you don't have an easy way to directly find the workflow log for an application object. Figure 1.4 illustrates sample selection criteria, and Figure 1.5 shows the output of the query from Transaction SWI6.

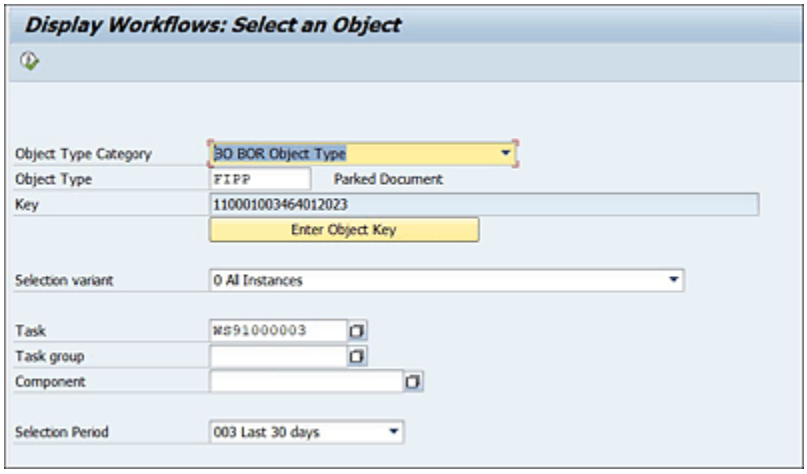


Figure 1.4 Sample Selection Criteria for Parked Journal Entry Approval Workflow Search Based on Document Key

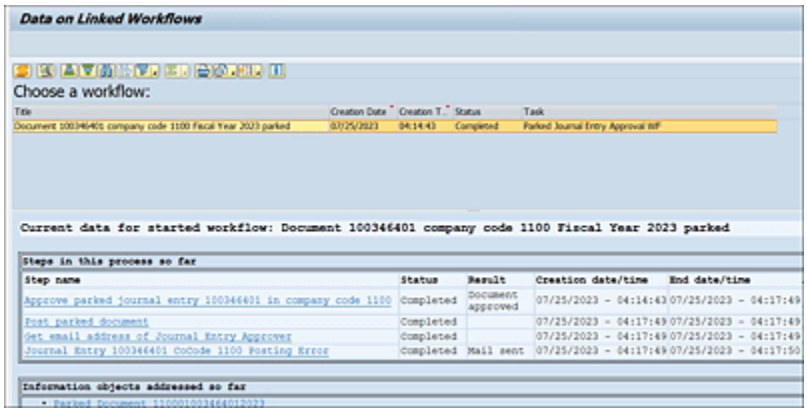


Figure 1.5 Sample Output of Transaction SWI6 with Details of Workflow Triggered for a Parked Journal Entry Document

1.3.2 Tools for Flexible Workflow

In addition to classical workflow transactions, SAP provided new tools for flexible workflow, which are listed in Table 1.4.

Tool	Details
Transaction SWDD_SCENARIO	This is a new transaction in SAP S/4HANA used to check all standard scenarios and create new flexible workflow scenarios.
Manage Workflows app	Business process consultants/functional consultants use this SAP Fiori app to manage workflows and create templates for required scenarios as and when required.
Manage Workflow Scenarios app	This is another important app that is used to check scenario definitions and import/export scenarios.

Table 1.4 Flexible Workflow Toolset

You can find all the details about these tools in Chapter 14 and Chapter 16.

1.3.3 BRFplus Development Tools

BRFplus isn't a direct workflow building tool, but it's now heavily used for any decision matrix and agent determination. Therefore, we've included the BRFplus toolset as a workflow supporting component development tools.

The development tools of the BRFplus are under the dedicated dropdown option in the BRFplus workbench, which you can see in Figure 1.6. These tools are available in the expert mode of the layout, which can be accessed via the **Personalization** menu. The layout of the BRFplus workbench is discussed in detail Chapter 10.

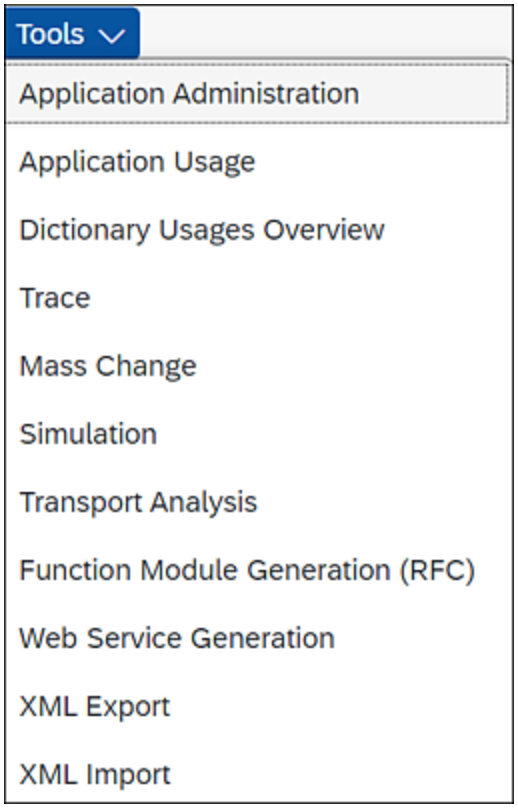


Figure 1.6 Development Tools

Let’s look at each of the development tools available:

- **Application Administration**
  This is a powerful tool to clean up the database and ensure that only the productive versions are available in the system. There are multiple operations possible with the tool on any given application. Figure 1.7 shows the possible operations.

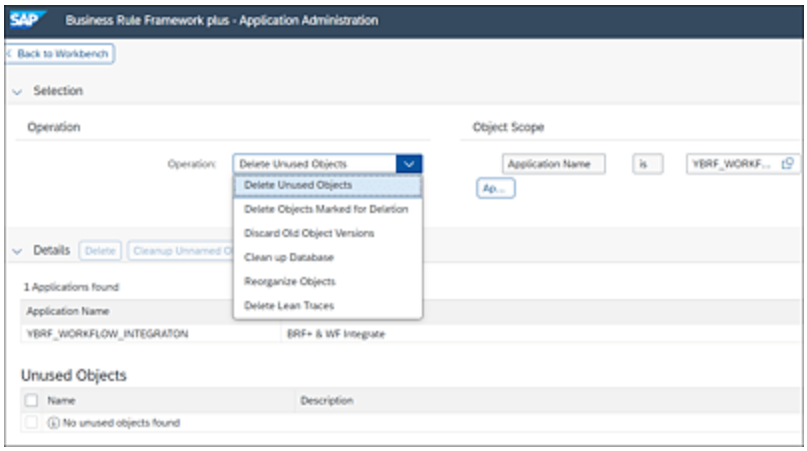


Figure 1.7 Operations Possible via the Application Administration Tool

It’s important to note that all the operations performed via this tool can also be performed via backend Transaction FDT_HELPERS. This backend transaction has many more operations than what you see in Figure 1.7, so it’s the preferred option for any administrative actions.

- **Application Usage**
  This tool gives insights into what application is being used in any given application or what application is using the given application. The tool’s search page is depicted in Figure 1.8.

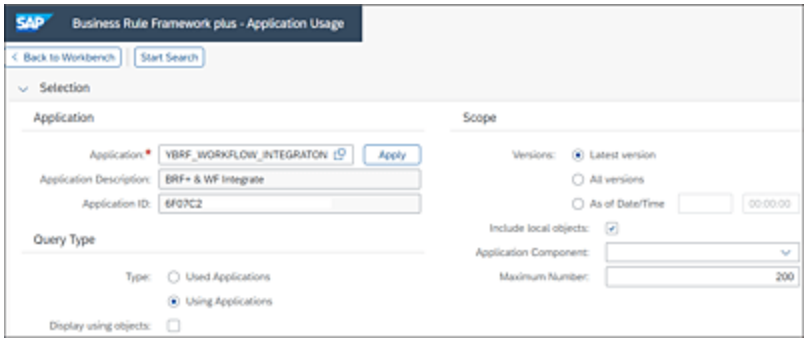


Figure 1.8 Search Screen for the Application Usage Tool

- **Dictionary Usages Overview**
  This tool helps to provide the dictionary reference of the

data object types. This is a very helpful tool for developers to ensure that the data types of the calling application and the data type used in the application are uniform, and there are no mismatches (see Figure 1.9).

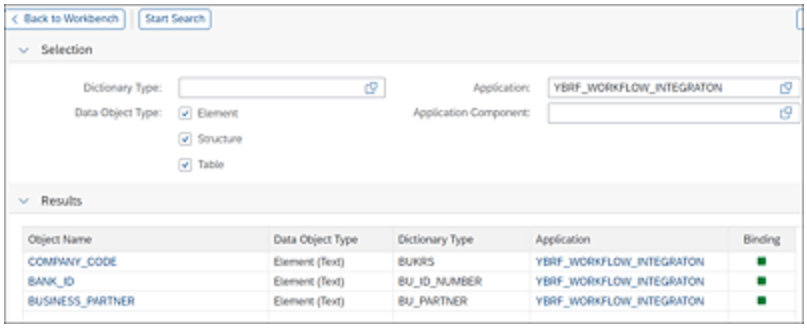


Figure 1.9 Dictionary Usages Tool

In case of any mismatch, the tool also highlights the specific data object with the red traffic light icon.

- **Trace**
  The processing logs of BRFplus application execution are very important for issue analysis. The log generation and depth of log capture is based on how the calling application invokes the BRFplus API. There are three levels of depth, as follows:
  - **No trace**
    This is when the calling application doesn’t want any traces to be captured in the system.
  - **Lean trace**
    This is already part of the generated BRFplus code and available for the processing logic, which directly results in the outcome of the overall BRFplus function execution.

- **Technical trace**
  This trace level is specifically required for detailed level analysis and debugging purposes.

- **Simulation**
  This is just like any other test environment available for testing the development. You must provide the inputs (context), and then the function—the starting point of the execution—is invoked, and all the processing logic is executed to finally produce the result.
  Note that there is simulation capability available for expressions and actions as well, as depicted in Figure 1.10 and Figure 1.11.

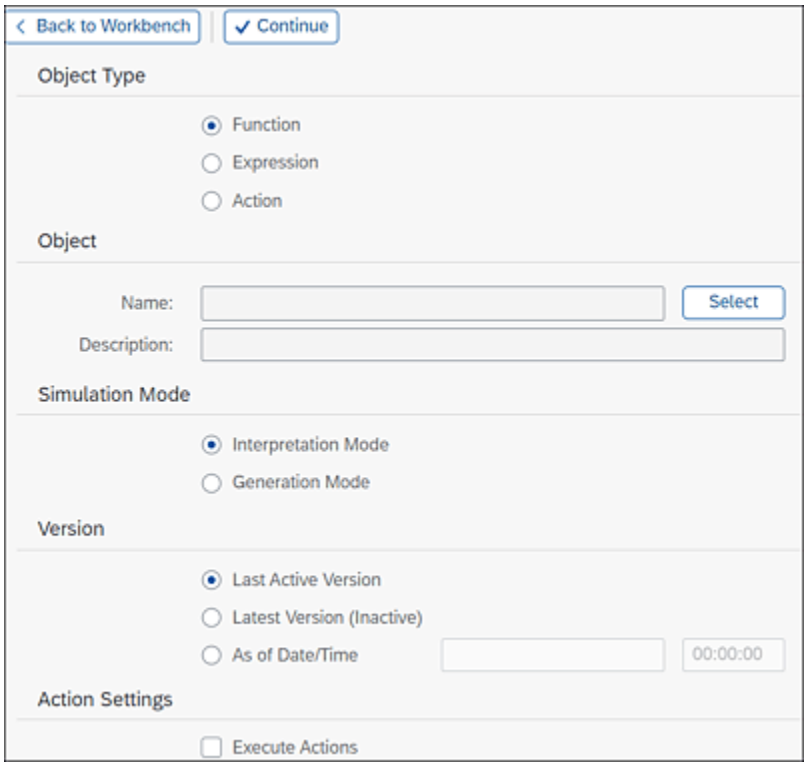


Figure 1.10 Selection Screen of Simulation Tool

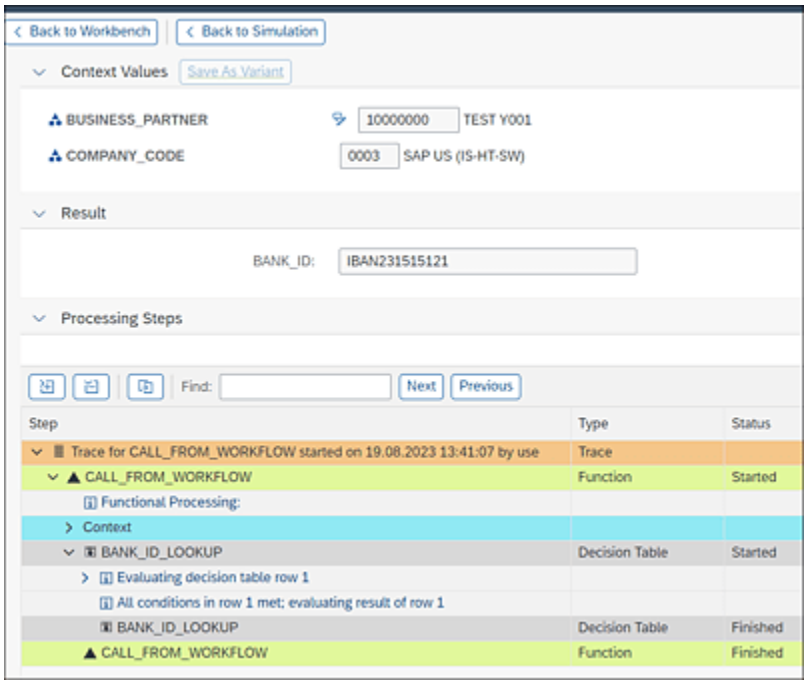


Figure 1.11 Simulation Outcome

- **Transport Analysis**
  This tool comes in handy for developers to ensure that they have smooth movement of the transports, through the complex before export and after import framework applicable to the BRFplus objects. These frameworks kick in for the validation as part of the overall transport organizer framework.
  The before export analysis of the transport can be done in the source system where the application is built. Using this tool, possible inconsistencies are highlighted in the application that will lead to transport failure. Therefore, it’s recommended to use this tool during releasing of the transport from the source system to avoid.
  The after import analysis comes in handy when the BRFplus application is transported to the target system but remains inactive. This tool highlights the causes of

unsuccessful transport imports and provides ways to reimport the inactive objects once the fix is applied.

- **Web Service Generation/Function Module Generation (RFC)**
  To make the business rules available for BRFplus consumption for a third-party or legacy application, it's important to provide a relevant endpoint. The endpoints can be made available via a web service. With a web service, the additional steps of Transaction SICF activation and Transaction SOAMANAGER configuration must be done for making the endpoint reachable.
  A similar step is available to automatically generate a Remote Function Call (RFC) function module that consumes the given BRFplus application inside it.
- **XML Export/XML Import**
  The export tool is used to convert the BRFplus application into XML data. The same XML representation can then be imported into another system using the *import XML* tool. This tool comes in handy when the applications are to be moved into different systems where no transport connection is available.

1.3.4 Tools for SAP Business Technology Platform

Workflow design and development in SAP BTP can be done in two ways: One is based on the older SAP Business Application Studio, and the other is the newest process builder that is part of the SAP Build Process Automation service. The former is more of a developer-based method

that required a bit of coding and the use of complex scenarios, and the latter is a citizen developer–based low-code/no-code tool. In addition, there are a few applications that help both the developer and the operational user with the much-needed view of monitoring the process and its steps. Let’s look at these tools in a bit more detail:

- **SAP Business Application Studio**
  SAP Business Application Studio is an integrated development environment (IDE) available as a service on SAP BTP. It’s a browser-based application development environment that encompasses the various tools and resources required to develop all kinds of applications, including workflows.
  SAP Business Application Studio is the preferred development tool for SAP BTP workflows when it comes to designing complex developer-centric workflows because of the following features:
  - Provided plug-ins and templates required to design, develop, deploy, and debug workflows in SAP BTP
  - Feature-rich code editor interface to make the life of a developer easy when it comes to developing workflows, applications in SAPUI5, or services using Node.js
  - Built-in command-line interface
  - Build and deploy tools specifically for SAP BTP
  - Ability to create individual spaces for each type of development, such as workflows, and contains its own set of runtimes and tools required to develop such types
  - Built-in integration with Git

SAP Business Application Studio needs to be subscribed to as a service in the SAP BTP subaccount, and then it will appear under **Instances and Subscriptions** in your subaccount. You can access it by clicking on the listed **SAP Business Application Studio** link under the **Subscribed Applications** section.
You'll see more of SAP Business Application Studio being used in designing and developing workflows in Part III of this book.

- **Process builder in SAP Business Process Automation**
  While we introduced SAP Business Application Studio as a tool for hard-core developers, there was a necessity to provide citizen developers with a low-code/no-code tool to rapidly design and develop workflows in SAP BTP. The answer to this quest is the process builder, which is part of the SAP Business Process Automation service in SAP BTP.
  Using the process builder browser-based cloud tool integrated to SAP Build Process Automation service, citizen developers can develop workflow processes incorporating automation scenarios, with just a few clicks of the mouse. Compared to SAP Business Application Studio, which requires writing code to achieve many tasks, process builder in SAP Build Process Automation is more intuitive and doesn't need coding at all. Most of the build part is taken care of by just dragging and dropping various controls and binding them to a data variable. With the process builder, you can create artifacts such as the following:
  - Workflows

- Automation scenarios
- Simple forms and approval forms
- Decision and rules
- Visibility scenarios

To access the process builder, you can navigate to the tool after you create a new project in the Lobby screen of the SAP Build Process Automation service by just clicking on the project name. You'll see more about how to create workflows using the process builder in Part III of this book.

You just saw the two major tools for designing and developing workflows in SAP BTP. Now let's talk about a few other key tools that help in the administration of the workflows that have been developed using these tools. From the SAP Build Process Automation **Lobby** screen, you'll see an option called **Monitor** which hosts a bunch of SAP Fiori apps that will help you with administering and monitoring your workflow processes, as follows:

- **Process and Workflow Instances**
  This app provides a view of all the workflow instances that are running, on hold, in error, completed, or canceled. It also gives the metadata details of the process such as started date, started by, instance ID, definition, and so on. The workflow logs that describe the various steps traversed so far are also provided, along with the context data. If the process errored out in some step, the log provides the details of the error in the step that failed to execute.
- **Process and Workflow Definitions**
  In this app, you can see the deployed workflows

developed using either SAP Business Application Studio or the process builder. It shows the ID and deployed version. You can perform the following actions in this app:

- **Show Instances**: After selecting a process ID, click **Show Instance** to go to the Process and Workflow Instances app while filtering the instances with the selected ID.
- **Start New Instance**: Start a new instance of the selected workflow by entering the necessary starter data payload in JavaScript Object Notation (JSON) format.
- **Download Model**: Download the process model that can be used later to deploy to another subaccount.

- **Triggers**
  In this app, you can view the triggers that have been created while designing the workflows. A trigger is basically a REST API call that delivers the initial data payload to trigger/start the workflow from an external source. In this tool, you can view the trigger with the API endpoint and the required data payload in JSON format. You can provide the information to the triggering source system where this will be configured, or you can use this to test the workflow using a REST client such as Postman. You can also add and delete the triggers from here.

1.4 Identity and Access Management

Some specific SAP security roles are required for workflow developers and administrators. The workflow can be developed in both SAP S/4HANA and SAP BTP, so we've discussed the specific roles required in both of these applications. We'll discuss first the roles needed in SAP S/4HANA for the developer, administrator, functional configurator, and end user. Then, we'll share a similar viewpoint for SAP BTP.

1.4.1 Roles and Authorizations in SAP S/4HANA

While working with SAP Business Workflow, one of the important aspects is corresponding roles and authorizations required to develop, configure, test, and execute the workflows. In this section, we'll discuss what corresponding authorizations will be given based on the work to be done related to workflows.

There are four primary types of users who are required to have these authorizations based on their roles:

- **Workflow developer**
  These are the users who are primarily responsible for workflow development. Usually, `S_DEVELOP` authorization is required to do development-related work. Additionally, following are some basic workflow development–related

transactions that workflow developers should have access to: Transaction PFTC, Transaction SWDD, Transaction SWDD_SCENARIO (for flexible workflow), Transaction SWO1, Transaction SWI2_FREQ, Transaction SWI1, Transaction SWEL, Transaction SWEC, Transaction SWE2, Transaction SBWP, and so on. Apart from other regular development-related authorizations, you need to have some OData-related authorizations to access different OData services and some specific SAP Fiori apps primarily for flexible workflow development (see details of this in SAP Note 3100365).

- **Workflow administrator**
  This is more toward Basis-related activities where initial one-time system activation and configurations related to workflow are required. Some of the activities are also related to workflow monitoring and administration. Following are the primary roles and authorizations required here:
  - `EXX_BC_SAP_ALL_RESTRICTED`: All authorization without Basis
  - `SAP_BC_BMT_WFM_ADMIN`: Administrator for SAP Business Workflow
  - `SAP_BC_BMT_WFM_DEVELOPER`: Developer for SAP Business Workflow
  - `SAP_SWFMOD_ADMIN`: Workflow modeler administrator
  - `SAP_WF_ADMINISTRATION`: SAP Business Workflow work for administrator

  Some of the specific transaction codes for which access should be given are Transaction SWNCONFIG (Extended Notification Configuration), Transaction SWU3 (Automatic

Workflow Customizing), Transaction SWWCOND_INSERT (Schedule Background Job for Work Item Deadline Monitoring), Transaction SWWCLEAR_INSERT (Schedule Background Job for Clearing Tasks), Transaction SCOT (Configuring Email), Transaction SOST (Checking Email), and so on.

- **Workflow-related functional configuration**
  This is more of a functional configuration role, but it's related to workflow. For example, to configure standard classical workflow for vendor invoices for park and post workflow, apart from workflow technical configuration, there are financial configurations required using Transaction SPRO menu path **Financial Accounting** • **Accounts Receivable and Accounts Payable** • **Business Transactions** • **Make and Check Settings for Document Parking**. Similarly, to configure the flexible workflow for purchase requisitions, you need to have authorization for the Manage Workflows for Purchase Requisitions app in SAP Fiori.
- **Workflow end users**
  These end users will be executing different tasks related to workflow from their inbox. From SAP GUI, Transaction SBWP access is given for the My Inbox app access. However, the My Inbox app access will be given if no SAP GUI access is given for any end user to execute the corresponding workflow tasks. Apart from this, the other corresponding business-related roles will be assigned based on the business area. For example, if some end user is responsible for approving the purchase requisitions tasks, then corresponding purchasing roles will be given to approve the requisitions; if some end user is

responsible for approving the invoices, then that business role will be given; and so on.

1.4.2 Roles and Authorizations in SAP Business Technology Platform

In SAP BTP, roles and authorizations are essential for ensuring proper access control, security, and governance over workflow-related activities both in SAP Workflow Management and SAP Build Process Automation. While this topic will be covered in detail in the respective chapters, here's an overview of the authorization process involved in SAP Workflow Management and SAP Build Process Automation.

There are two types of roles: global and instance level. While the global roles give permissions for all workflow definitions, instances, and tasks, roles at the instance level control the activities around a specific instance of a workflow, as follows:

- **`WorkflowManagementBusinessExpert`**
  This role collection enables discovery and importing of predelivered live process packages, and configuring process variants, decisions, and visibility scenarios within these live process packages. It also enables viewing, creating, editing, deleting (draft version), and activating visibility scenarios.

- **`WorkflowManagementAdmin`**
  This enables viewing, exportng, and importing packages; monitoring workflows; viewing workflow definitions and instances; downloading workflow models;

suspending/resuming workflow instances; retrying failed instances, and modifying/overwriting the workflow context. This also handles the authorization to view active visibility scenarios, trigger the processing of data, clear existing processed data, and view information and errors related to the processing of data. Pushing events and viewing acquired events and the related errors are also covered by this role collection besides managing and deploying rules.

- **`WorkflowManagementDeveloper`**
  This role collection enables actions such as view, create, edit, activate, and delete (draft version) live process packages, workflows, business rules, and visibility scenarios.

In addition, fine-grained authorization can be controlled in roles for individual capabilities such as workflows, business rules, and process visibility. Custom role collections can be created by adding these additional roles from capabilities such as workflows, business rules, process visibility, and workflow management. These are covered in detail in their respective chapters.

Like SAP Workflow Management, access to perform relevant activities are controlled through a wide range of roles and role collections in SAP Build Process Automation. Some of the key role collections are as follows:

- **`ProcessAutomationAdmin`**
  Manages the process configuration, permissions, and authorizations within SAP Build Process Automation.

- **`ProcessAutomationDeveloper`**
  Manages the creation, editing, and publishing of individual processes and automations within SAP Build Process Automation.
- **`ProcessAutomationParticipant`**
  Participates in active SAP Build Process Automation processes.

In addition, granular control can be enabled through roles for the individual capabilities such as SAP Intelligent Robotic Process Automation roles, SAP Workflow Management, process visibility, business rules, and workflows. Details of these roles are covered in their respective chapters.

1.5 Summary

Workflow management is a key capability that has been part of SAP ERP and has continued into SAP S/4HANA and SAP BTP. The core capability of workflow management has remained the same though the internal architecture evolved over time between these products.

Workflow management in SAP ERP facilitates the automation of business processes by assigning assignments and approvals to the proper users in accordance with specified criteria. It has tools for developing workflows, processing tasks, and triggering workflows based on events, such as workflow builder (Transaction SWDD) and Business Workplace (Transaction SBWP). SAP ERP's workflow management is primarily focused on process automation within the SAP ERP system, even if it offers necessary automation capabilities.

Workflow management in SAP S/4HANA was improved with more sophisticated features. Although SAP S/4HANA adds SAP Fiori interfaces for task processing, the workflow builder is still used to develop workflows. This enhances the user experience. SAP S/4HANA also offers a more simplified and integrated workflow management strategy across many modules and business segments. SAP S/4HANA brings the flexible workflow capability that is easily configurable and can be set up by business users or functional process consultants. It's built on top of SAP's classical workflow engine and tightly integrated with SAP Fiori and My Inbox.

This provides a low-code/no-code platform for easy workflow setup.

However, the new workflow capability didn’t stop here. SAP brought an intelligent workflow capability that can span multiple enterprise-binding applications to deliver end-to-end business processes on SAP BTP. It can consume new technology such as artificial intelligence and robotic process automation in a consumable architecture.

2 Introduction to Classical Workflows

This chapter introduces the details of classical workflows, focusing on standard SAP-delivered workflows. As classical workflows play a significant role in building workflow solutions in SAP S/4HANA, this chapter is required for understanding and using classical workflows in SAP S/4HANA.

This chapter introduces you to classical workflows—workflows that use the SAP Business Workflow engine in SAP S/4HANA. You'll learn about standard, SAP-delivered workflows, before configuring the SAP S/4HANA system to work with SAP Business Workflow. The chapter describes how to activate standard workflows and maintain agent assignments. You'll then learn to prepare the system for the SAP Business Workflow runtime. The chapter will also discuss situations when it's appropriate to use custom workflows, rather than standard workflows.

2.1 Evolution of Classical Workflows

A *classical workflow* refers to the workflow management functionality provided by SAP Business Workflow, which is introduced in very early releases of SAP's software, such as

SAP R/3. Starting from its inception to its current state, classical workflows have undergone several iterations, enhancements, and advancements. Let’s explore the key milestones in the evolution of classical workflows:

- **Introduction of SAP R/3**
  The foundation of classical workflows can be traced back to the introduction of SAP R/3 in the early 1990s. With SAP R/3, businesses gained the ability to automate and streamline their core processes, paving the way for the future development of workflows.
- **SAP Business Workflow**
  In the late 1990s, SAP introduced SAP Business Workflow, which served as the precursor to classical workflows. SAP Business Workflow provided a framework for modeling and executing business processes within the SAP system, enabling companies to digitize their manual workflows and enhance efficiency.
- **Release of SAP ERP**
  The release of SAP ERP brought further improvements to classical workflows. SAP ERP expanded the capabilities of workflows, allowing for more complex process modeling, integration with other SAP modules, and enhanced user interfaces for task management.
- **Integration with SAP NetWeaver**
  SAP NetWeaver, introduced in the early 2000s, played a crucial role in the evolution of classical workflows. SAP NetWeaver provided a platform for seamless integration between different SAP systems and applications, enabling workflows to span across various modules and systems within an organization.

- **Web-based workflows**
  As internet technologies advanced, classical workflows evolved to embrace web-based interfaces. This enabled users to access and interact with workflows through web browsers, enhancing the usability and accessibility of workflow management. As part of this, classical workflows supported SAP Enterprise Portal–based applications, and SAP Business Workflow and SAP Enterprise Portal integration was one of key aspects of this evolution journey.
- **Integration with SAP Fiori**
  With the rise of SAP Fiori, a modern and intuitive user experience for SAP applications, classical workflows received a significant boost. SAP Fiori provided a responsive, mobile-friendly interface for managing workflows, making it easier for users to participate in and monitor workflow processes on the go.
- **Cloud integration**
  As the adoption of cloud technologies increased, classical workflows expanded into the cloud environment. SAP Workflow Management, which runs on SAP Business Technology Platform (SAP BTP) offers cloud-based workflow capabilities, enabling organizations to leverage the scalability, flexibility, and cost-effectiveness of the cloud for their workflow needs.
- **Intelligent automation**
  In recent years, classical workflows have embraced intelligent automation technologies, such as artificial intelligence (AI) and machine learning. These technologies enhance the capabilities of workflows by automating

decision-making, providing predictive analytics, and enabling intelligent routing of tasks based on various factors.

Over the years, as technology and innovation progressed and with the introduction of newer SAP products and features through this journey, SAP aimed to provide a unified workflow solution that combines classical and cloud-based workflow capabilities, offering enhanced flexibility, scalability, and integration possibilities.

2.2 Standard Workflows

SAP standard workflows are preconfigured templates within the SAP system that define a series of sequential steps required to complete a specific business process. These workflows cover a wide range of business functions, such as purchasing, sales, human resources, finance, and more. They are designed based on best practices and industry standards, providing organizations with a solid foundation for automating their processes. In the following sections, we'll discuss the details of how to search for standard SAP-provided workflows and cover what the commonly used standard workflows are.

2.2.1 Searching for Standard Workflows

Now you'll see how to find standard workflows in an SAP system (this can be either SAP ERP or SAP S/4HANA) via standard Transaction SWDM (Business Workflow Explorer). Here, you can search by using filters such as application component, business object, and classes (including methods).

Figure 2.1 illustrates Transaction SWDM (for business application purchasing) where you can see the list of workflow templates available in the **Business Workflow Explorer** screen.

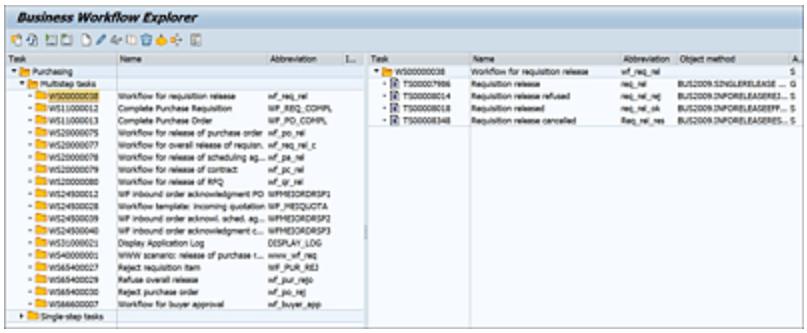


Figure 2.1 List of Workflow Templates in Business Workflow Explorer

You can search for standard tasks from table `HRS1000`, as shown in Figure 2.2. Go to Transaction SE16 or Transaction SE16N, and then use the following filter criteria: **Std.object type** as “WS”, **Language** as “EN”, **User Name** as “SAP”, and **Object abbr**. as “MM*” to find out all the standard workflow templates in the material management area.

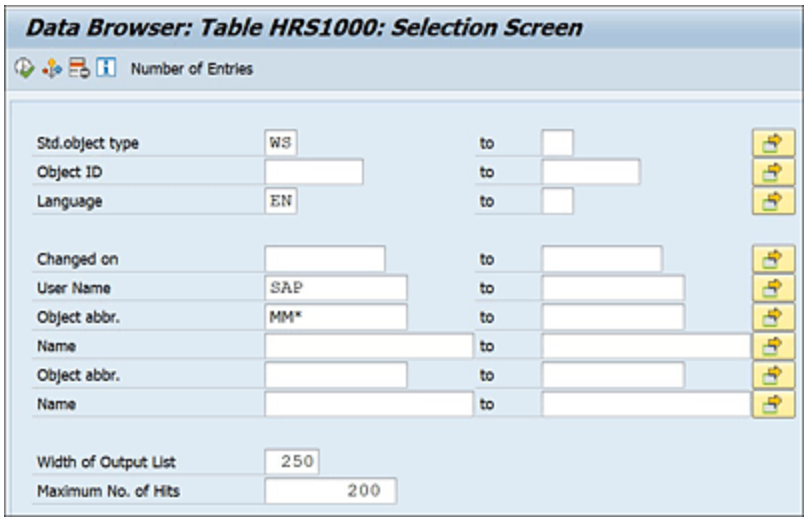


Figure 2.2 Selection Criteria to Fetch All the SAP Standard Workflow Templates in the Material Management Area

Figure 2.3 shows the result of the selections you made in Figure 2.2. It lists all SAP-provided standard workflow templates in the material management area.

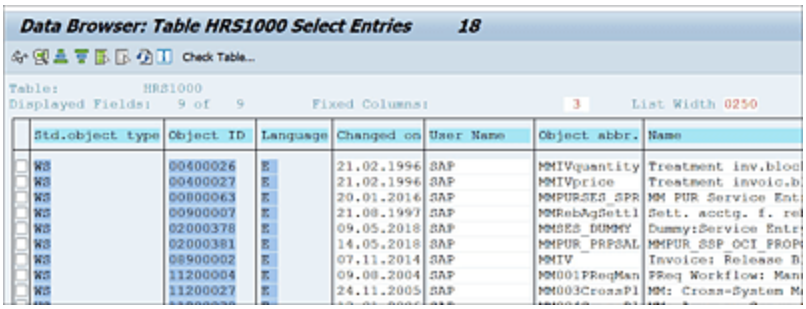


Figure 2.3 Results of the Preceding Selection Showing the Standard Workflow Template in Materials Management

Note

You can search all standard tasks the same way using the standard object type of “TS” in table `HRS1000`.

Let’s consider another way to search SAP-provided standard workflows. Say you’re looking to find out all the SAP purchasing workflows. Go to Transaction PFTC, provide **Task type** as “Workflow Template”, and click the F4 help of the **Task** field. This will open a popup where you need to enter the **Search Term** as “*PURCH*” and click the green checkmark. In another window, you’ll see the results where workflow **Object name** or object description contains the word “PURCH”, and you can thus get a tentative list of purchasing workflows. Figure 2.4 shows the list of purchasing workflow templates from Transaction PFTC.

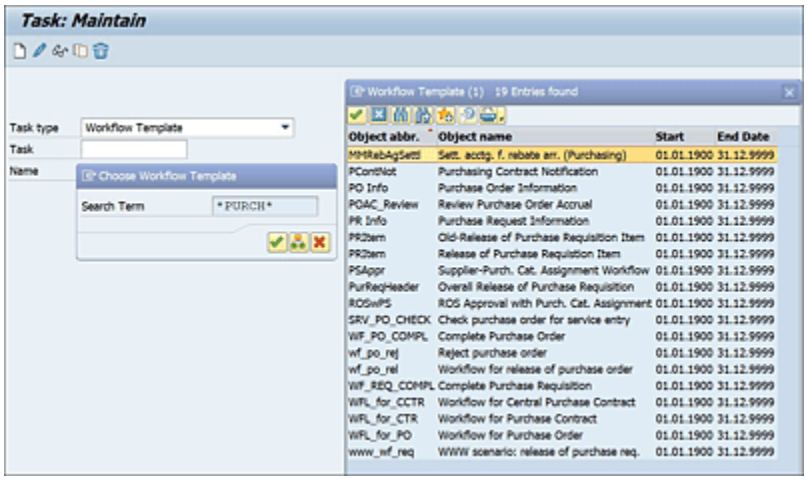


Figure 2.4 List of Purchasing Workflows from Transaction PFTC

> **Note**
>
> This will return both standard and custom workflows, so you'll need to determine which are standard workflows by checking inside the workflow template.

2.2.2 Commonly Used Standard Workflows

SAP has provided a good number of standard classical workflow templates in different functional areas such as purchasing, finance, travel management, HR, and so on. Table 2.1 lists some of the common standard classical workflows that are used in different SAP S/4HANA implementations.

Template Number	Short Text
WS20000077	Workflow for Overall Release of Purchase Requisition

Template Number	Short Text
WS02000471	Workflow for Release of Purchase Requisition Item
WS20000075	Workflow for Release of Purchase Order
WS20000078	Workflow for Release of Scheduling Agreement
WS20000079	Workflow for Release of Purchase Contract
WS00800302	Workflow for Purchasing RFQ
WS20000050	Travel Request Workflow
WS20000040	Trip Approval Workflow
WS20001004	Workflow for Release of Incoming Invoices
WS10000051	Financial Accounting Park and Post Workflow
WS12300111	SAP Employee Self-Service/Manager Self-Service General Leave Request Workflow

Table 2.1 Commonly Used Standard Classical Workflows

Note

The workflow template numbers given in the preceding list may vary based on the SAP system version.

2.3 Configuring the SAP Business Workflow System

Before you start testing any classical workflows in any SAP system, you need to configure the SAP Business Workflow system; for this task, the primary activity is **Automatic Workflow Customizing** executed through Transaction SWU3. This activity needs to be performed in each system in the landscape manually, so it should be listed as one of the cutover tasks.

Follow SAP Implementation Guide (IMG)/Transaction SPRO to perform a system-wide initial workflow setup. Follow menu path **SPRO** • **SAP NetWeaver** • **Application Server** • **Business Management** • **SAP Business Workflow** • **Maintain Standard Settings**. You'll get the **Automatic Workflow Customizing** screen as shown in Figure 2.5.

The system has provided an option to automatically set it up with a click of a button. Click the  button to automatically set up the system for the workflow. You can then focus on the items that can't be set up automatically. You can also go to individual items and change the value from whatever is set up automatically. We'll now go through some of the items that may sometimes give you error or require you to change the default setting. To execute individual items, select the line item you want to execute, and click the  button.

Figure 2.5 Automatic Workflow Customizing

In the following sections, we'll walk you through how to maintain the runtime environment, definition environment, and additional settings and services. You'll also learn how to classify tasks as general tasks.

2.3.1 Maintain Runtime Environment

The activities performed in this section must be executed so that workflows can be executed. When you go to the **Automatic Workflow Customizing** screen, only those activities that currently have no errors are executed automatically. The following activities can be performed automatically:

- **Configure RFC Destination**
- **Maintain System Administrator for Workflow**
- **Maintain Active Plan Variant**
- **Classify Decision Task as General**
- **Document Generation/Form Integration**
- **Maintain Time Units**

- **Schedule Background Job for Missed Deadline**
- **Schedule Background Job for Work Items with Errors**
- **Schedule Background Job for Condition Evaluation**
- **Schedule Background Job for Event Queue**
- **Schedule Background Job for Clearing Tasks**
- **Schedule Background Job for Deleting Expired Request (CP Workflow Integration)**

Configure Remote Function Call Destination

If we carry out this activity automatically, the logical RFC destination `WORKFLOW_LOCAL_XXX` (where XXX = client number) is created if it doesn't yet exist. The workflow runtime engine always gets executed within one client, so this logical RFC destination name provides a unique name. If this logical destination doesn't exist, the system automatically creates this logical system.

The user `WF-BATCH` is assigned to this destination. If the `WF-BATCH` user ID doesn't exist, this activity is automatically executed, and the current user is part of the `SUPER` user group, then the `WF-BATCH` user ID is created with maximum authorization of the current user who is executing it. It's also possible to manually execute this step, and a different user ID and password can be maintained.

Note

Starting with SAP S/4HANA 1709, the workflow system user WF-BATCH isn't generated automatically anymore. The workflow system user is called SAP_WFRT now instead of WF-BATCH and must be created manually (refer to SAP Note 2568271).

The user maintained against the RFC destination required role SAP_BC_BMT_WFM_SERV_USER and application-specific authorization. Refer to SAP Note 1251255 on how to set up the WF-BATCH user with a specific role.

It'll be very difficult to get all application-specific roles. The workflow can fail if WF-BATCH doesn't have any specific application authorization needed to execute the functions. So it's recommended to go with the SAP_ALL kind of authorization for WF-BATCH. This is the heart of workflow execution, so you should ensure that the WF-BATCH user id is created with the SAP_ALL role before executing this step.

If you execute the **Configure RFC Destination** step, you'll get a success message, as shown in Figure 2.6.

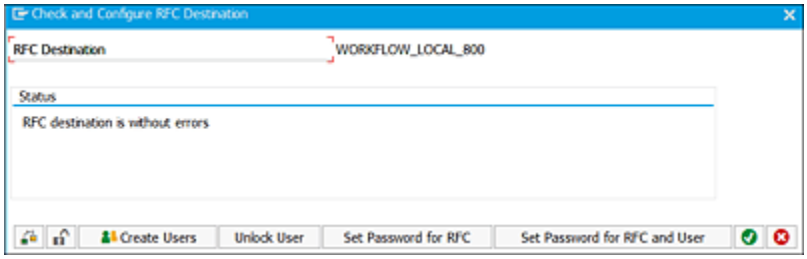


Figure 2.6 Automatic RFC Destination Configuration

Starting with SAP S/4HANA 1805, a reference RFC destination WORKFLOW_REFERENCE_XXX (where XXX = client number) is also created with a trusted connection, which avoids any issues with the synchronization of passwords

between the SAP_WFRT user and the WORKFLOW_LOCAL_XXX destination. By using the trusted RFC relationship feature, there is no need to maintain the password with the RFC destination anymore because the password is only at the user. The password of the system user SAP_WFRT can be changed at any time without disruption of the RFC execution.

Maintain Workflow System Administrator

The workflow administrator automatically gets the notification if there is any error in the workflow. The workflow administrator can be maintained at each workflow or at the system level. You can maintain the workflow administrator at the individual workflow level at the workflow header. Click on the **Basic Data** button in workflow builder or choose **Goto** • **Basic Data**, then go to the **Version-Dependent (Current Workflow Version)** tab, and finally the **Agents** subtab to set up the workflow-specific system administrator. Figure 2.7 shows where to maintain the system administrator at the workflow level.

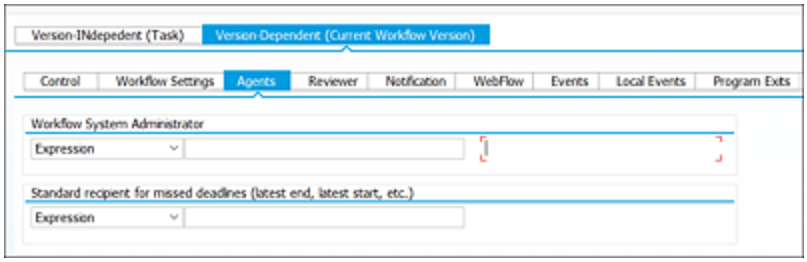


Figure 2.7 Maintain the Workflow System Administrator at the Workflow Level

The workflow administrator at the workflow level can be defined as a work center, security role, rule, user ID,

organization unit, and so on, similar to any agent determination procedure. It’s good practice to have a business domain-specific workflow administrator and set it up at the workflow level. This kind of structured approach is defined where you have a high number of workflows. If you have a very low number of workflows, then a couple of people can be responsible for managing all the workflows.

In the system level workflow system administrator maintenance, you don’t have all the flexibility to define the workflow administrator dynamically. You’ll have some fixed options such as work center, security role, user ID, and organization unit. When you execute the task automatically, the user who executes this task is assigned as the workflow administrator. You should change this after automatic generation.

It’s advisable to use any kind of HR organization unit, work center, or security role for the workflow administrator. We recommend going with the security role because it will be easy to maintain in the future. If you go with the security role, then there should be different roles for workflow developer and workflow administrator. The workflow administrator needs a few extra authorizations. So, only a few people should have this access. In a live system, you should make this configuration independent of security role changes. For example, to maintain a different derivation based on country, a dummy custom role can be created and added to the template role. Whenever a new derivation comes, it will include the custom dummy role, so the workflow administrator configuration doesn’t need to be changed.

The standard workflow system administrator must be maintained in each client. If you've already maintained a standard workflow system administrator (Transaction SWDC_RUNTIME), it's not overwritten by the automatic execution of this activity.

Maintain Active Plan Version

Only one of the plan versions created in the system can be active. This plan version (with its contents) is seen by the workflow system as the only valid plan version. All SAP workflows supplied are then in the plan version marked as **Active** automatically.

If you carry out this activity automatically, **01** is set as the active plan version. If you carry out the activity manually, you enter the plan version that you want to use as the active plan version in the **Value abbrev.** field in the group **PLOGI** with the semantic abbreviation **PLOGI**, as shown in Figure 2.8.

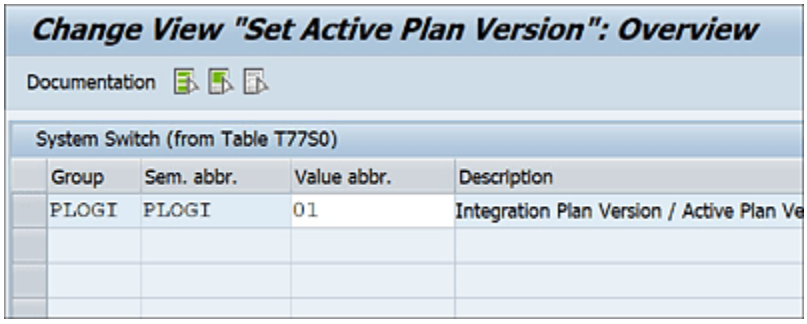


Figure 2.8 Manual Maintenance of the Active Plan Version

Classify Decision Task as General

Task TS00008267 (generic decision task) must be declared as a general task. Like all SAP tasks supplied, the generic decision task doesn't have any possible agents as standard. Follow these instructions to complete this task:

1. Carry out the activity.
2. Position the cursor on the generic decision task, and choose **Properties**.
3. Choose **General Task**.

The generic decision task classification is one of the settings made in automatic customization.

Document Generation/Form Integration

The tasks for processing documents (`TS70008298`, `TS71007944`, `TS71007945`, `TS71007945`, `TS71007946`, and `TS71007954`) are declared as general tasks. The tasks for processing forms (`TS70008112`, `TS70008113`, `TS70008114`, and `TS70008115`) are also declared as general tasks.

All users in the SAP system are allowed as possible agents of these tasks. The agent restriction must be performed by selecting responsible agents in the step definition. In addition, check the **KBA 3133006 - "Tasks (document generation, forms) = not general"** error message in Transaction SWU3.

Maintain Time Units

Here, you can set the units of measurement and time that are used by the SAP system. You (or the system) will

maintain three units of measurement: **Dimensions**, **ISO Codes**, and **Units of Measurement**, as shown in Figure 2.9. This needs to be changed only if there is a requirement to overwrite the SAP-provided default values. This can be done by executing Transaction SWU3 (Automatic Workflow Customization) and selecting **Maintain Time Units**.

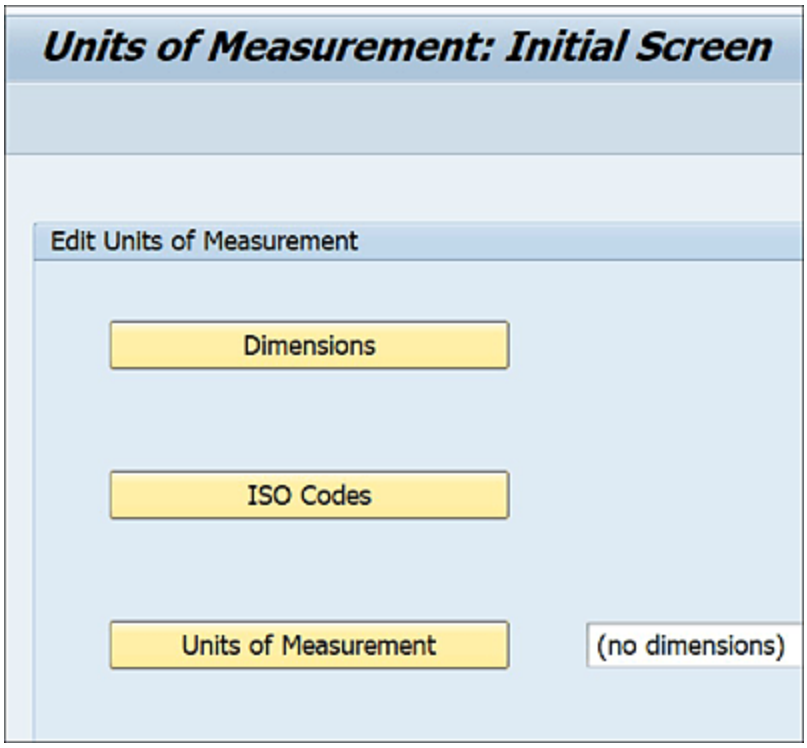


Figure 2.9 Shows the Maintenance of Units of Measurement

Schedule Background Job for Missed Deadlines

Specify a time interval at which the background job is called regularly. With each execution, the background job checks whether new deadlines have been missed since the last time it ran. Background job `SWWDHEX` is scheduled to process the missed deadline. The deadline outcome won't be triggered without this batch job. By default, it's scheduled

every three minutes. If you have too many workflows with deadlines activated, then this job may have a performance issue. So based on the number of deadlines, you may have to change the job interval.

To successfully execute this activity, the **Configure RFC Destination** activity must have been successfully executed. Report RSWWDHEX is used, and documentation for background job SWWDHEX can be found in SAP Note 1262519.

Schedule Background Job for Work Items with Errors

You use this activity to schedule monitoring and special treatment for background work items that could not be executed initially because of a temporary error in the underlying object method. The job can be scheduled from the **Schedule Background Jobs SAP Business Workflow** option under Transaction SWU3.

These background work items with temporary errors are then restarted automatically. The activity must also be scheduled if the workflow system administrator is to be notified automatically by mail in the event of application errors and system errors.

Background job `SWWERRE` is scheduled to reprocess the temporary error. By default, it will try three times in a 3-minute interval. The job won’t consider a work item for reprocessing after three times. Report RSWWERRE is used, and documentation for background job `SWWERRE` can be found in SAP Note 1676991.

Schedule Background Job for Condition Evaluation

The background job for checking the conditions for the work item start and work item end is scheduled with the standard parameters. To successfully execute this activity, the **Configure RFC Destination** activity must have been successfully executed.

Report RSWWCOND is used, and documentation for background job SWWCOND can be found in SAP Note 1677053.

Schedule Background Job for Event Queue

The background job for the event queue is scheduled with the standard parameters. If you execute the activity manually, you can schedule the background job with your own parameters. To do this, execute the **Schedule Background Jobs SAP Business Workflow** step from Transaction SWU3, and search for **Technical Job Definition** name “SAP Workflow Event”. To successfully execute this activity, the **Configure RFC destination** activity must have been successfully executed. Report RSWEQSRV is used. Figure 2.10 shows the job and its attributes for the job related to the event queue.

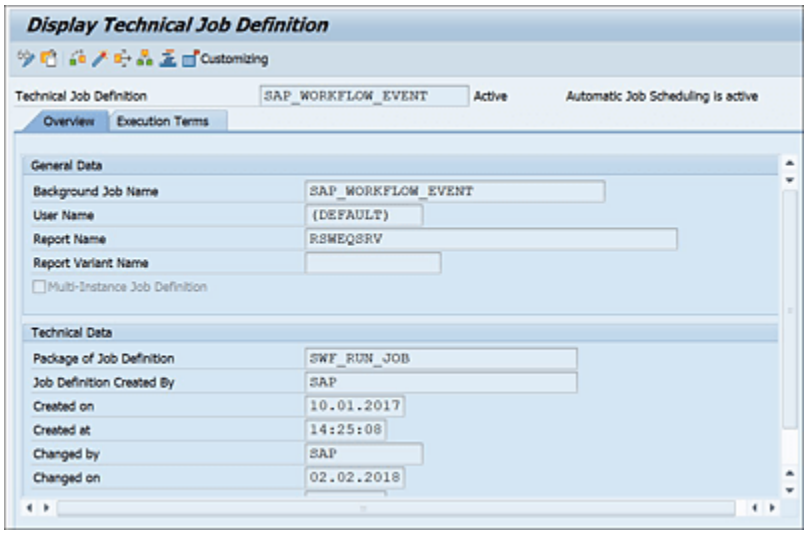


Figure 2.10 SAP Background Job Related to the Event Queue

Schedule Background Job for Clearing Report

This background job for clearing work in the workflow system deletes all job logs of the background jobs listed in Table 2.2.

Job Name	Details
SWWCOND	Work item rule monitoring
SWWDHEX	Work item deadline monitoring
SWWERRE	Work item error monitoring
SWWCLEAR	Clearing tasks in the workflow system

Table 2.2 Background Job Logs That Are Cleared When Job RSWWCLEAR Is Executed

Report RSWWCLEAR is used. Following the execution of workflow Customizing, the cleanup background job SWWCLEAR must run just once per system but not in every client. After scheduling, the background is executed once a day at 00:00

hours. To successfully execute this activity, the **Configure RFC destination** activity must have been successfully executed.

Schedule Background Job for Deleting Expired Request (CP Workflow Integration)

This report implements the central workflow system job `SAP_WORKFLOW_CPWF`, which runs in the business client if SAP Build Process Automation instances exist on the database. The report deletes finalized SAP Build Process Automation instances from table `SWF_CPWF_INST` after a retention time that is specified per entry in this table.

Tips

Starting with SAP S/4HANA 1709, the workflow system user and workflow system jobs changed. The workflow system user is called `SAP_WFRT` now instead of `WF-BATCH`. The workflow system jobs start with `SAP_WORKFLOW` now and are scheduled automatically by Transaction SJOBREPO (Technical Job Repository).

If you upgraded from an older release to SAP S/4HANA1709 or higher, refer to SAP Note 2568271.

If you have an issue where these jobs aren’t getting scheduled automatically from Transaction SJOBREPO in the SAP S/4HANA environment, see SAP Knowledge Base Article (KBA) 2770337.

2.3.2 Maintain Definition Environment

Under this section, all the activities must be executed manually as listed here:

- **Maintain Prefix Numbers**
- **Check Number Ranges**
- **Check Entries from HR Control Tables**

Maintain Prefix Numbers

This step defines how the number will be allocated when you create a workflow template, task, rule, or task group. Standard objects (workflow tasks, standard tasks, etc.) are identified with an eight-digit number. The last five digits of this number are assigned automatically by the system. The prefix number is used for the first three digits of this number.

To guarantee unique identification, define a unique prefix number for each system and client. Under this step from Transaction SWU3 in the **Maintain Prefix Numbers** section, you need to create new prefix numbers for the corresponding SAP system and client. Figure 2.11 shows how to maintain the prefix numbers for workflow and organizational management.

The prefix number applies for the following objects:

- Standard tasks
- Task groups
- Rules

The prefix should be configured only in the development system and not the quality assurance (QA) or production systems because the workflow templates/standard tasks/agent rules should be created in the development system and then imported to the follow-on systems by transport requests. Therefore, it's not necessary to change the **Edit Prefix Numbers** node in Transaction SWU3 to green in QA or production systems. See KBA 1910992 for further information on this.

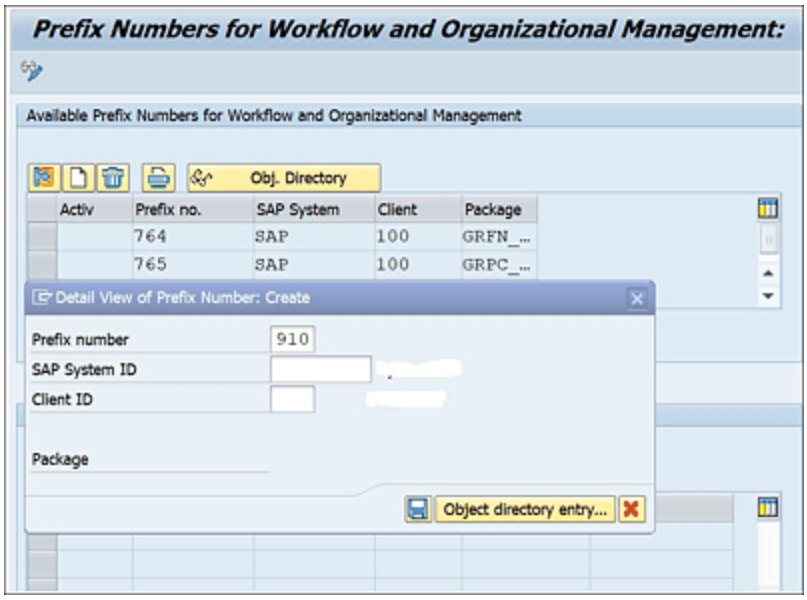


Figure 2.11 Maintain the Prefix Number for Workflow

Check Number Ranges

This routine check determines whether there is a number range for workflow tasks (**WF**) and customer tasks (**T**). Figure 2.12 shows the corresponding check under **Automatic Workflow Customizing**.

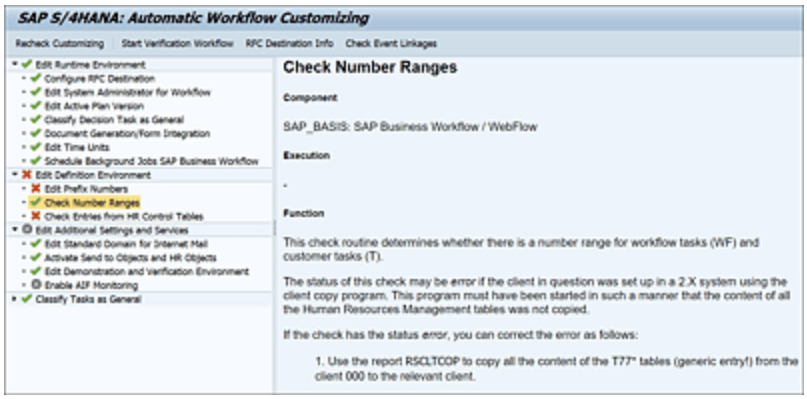


Figure 2.12 Check Number Ranges in SAP Automatic Workflow Customizing

The status of this check may be **Error** if the client in question was set up in a 2.X system using the client copy program. This program must have been started in such a manner that the content of all the human resources management tables wasn’t copied.

If the check has the status **Error**, you can correct the error as follows:

1. Use report RSCLTCOP to copy all the content of the T77* tables (generic entry) from client 000 into the relevant client.
2. Carry out the Customizing for personnel planning and workflow again.

Refer to SAP Note 31621 for more information.

Check Entries from Human Resources Control Tables

The organizational management control tables must be complete if the workflow system is to be functional. A check is made to this effect if you carry out this activity. To solve

problems of this type, use report RHTTCP77, which is shown in Figure 2.13.

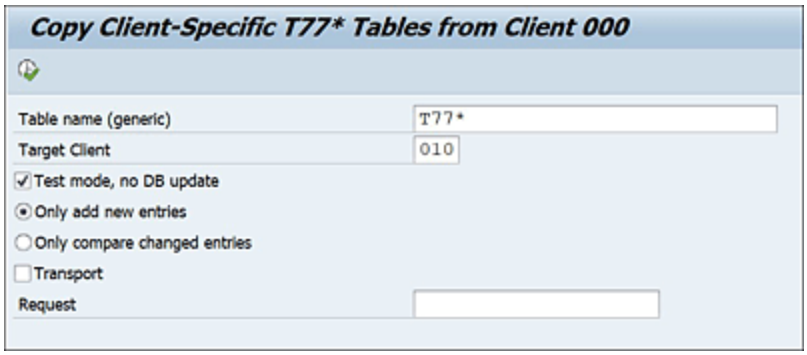


Figure 2.13 Report RHTTCP77

If there are errors regarding the transport objects `PDST` or `PDWS`, the relevant entries aren't maintained in table `SOBJ`. You can use report RHSOBJCH to rectify this later. Note that with this error, the workflow is executable in the current system, but crucial information isn't transported in the event of transports into test or production systems. In addition, refer to SAP Note 31621.

These tables are usually supplied. If entries are missing or problems can't be solved, contact SAP or try to find the relevant notes using SAP Service Marketplace.

2.3.3 Maintain Additional Settings and Services

The activities performed in this section must be executed if particular SAP Business Workflow functions are required. When you go to the **Automatic Workflow Customizing** screen, only those activities that currently have status **Error** are executed automatically.

The following activities can be performed automatically:

- Sending to objects and activating HR objects
- Maintaining demo and verification environments

Following is the full list of items that are done under this section:

- **Maintain Web Sever**
- **Maintain Standard domain for Internet Mail**
- **Activate Send to Objects and HR Objects**
- **Maintain Demo and Verification Environment**

Maintain Web Server

If you want to execute workflows that use WebFlow functions, you have to define a web server for the SAP system. For example, this is the case if the workflows to be executed contain a web activity. Automatic execution of this activity isn't possible. Follow these steps to define the web server:

1. Choose **Tools** • **Business Workflow** • **Development** • **Administration** • **Basic Settings** • **WebFlow** • **Customizing Web Server**.
2. On the **Customizing Web Server: Change** screen, enter the address of your web server and, if applicable, the port number. Optionally, you can enter another service.

Maintain Standard Domain for Internet Mail

Questions can be created with respect to work items within the workflow system. These questions can be answered using an email to the relevant work item. The answers are added to the work item’s attachments automatically. A standard domain for the current system is required for this function. Automatic customization isn’t possible for the standard domain.

For more information, refer to the SAP Library, and choose **mySAP Technology Components** • **SAP Web Application Server** • **Basis Services/Communication Interfaces (BC-SRV)** • **Communication Interfaces (BC-SRV-COM)** • **SAPconnect (BC-SRV-COM)** • **SAPconnect: Administration** • **Default Domain**.

Activate Send to Objects and HR Objects

If the workflow system is to send work items and mails to business objects and organizational objects (positions, organizational units, etc.) in Business Workplace, the relevant functions must be activated. You activate them centrally in the shared office settings of Business Workplace. For more information on the shared office settings, refer to SAP Library. Choose **mySAP Technology Components** • **SAP Web Application Server** • **Basis Services/Communication Interfaces (BC-SRV)** • **Business Workplace and Services (BC-SRV-OFC)** • **Business Workplace (BC-SRV-GBT)** • **Administration of the Business Workplace** • **Administration of the Send, Folder and Office Functions** • **Shared Office Settings**.

Maintain Demo and Verification Environment

The verification workflow and all demo workflows are declared as general, which means they can be started by all users. This is done as part of automatic customization in **Maintain Additional Settings and Services** under the **Maintain Demo and Verification Environment** section.

2.3.4 Classify Tasks as General

All the activities in this section declare SAP tasks or SAP workflows as general tasks. When you go to the **Automatic Workflow Customizing** screen, only those activities that currently have the status **Error** are executed automatically. If after performing automatic customization, the status is still `Error`, apply SAP Note 3119858 and perform the automatic customization again.

The following activities can be performed automatically:

- **Test Workflows**
- **Customizing with Workflow**
- **IDoc Interface**
- **Unit Test**
- **SAPphone**
- **Classify Scenario Tasks (Flexible Workflow) as General**
- **Processing of Replies to Appointment Request**

Figure 2.14 shows the details of this section and how to perform these steps.

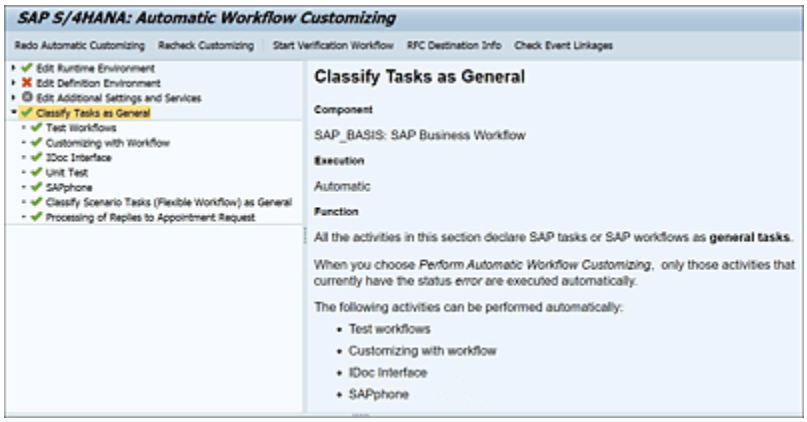


Figure 2.14 Classify Tasks as General

Test Workflows

All test workflows are declared as general, which means that they can be started by all users. The test workflows are in task group `TG70000020`. This step is performed through automatic customization.

Customizing with Workflow

The workflows are in task group `TG74500043`. This step is performed through automatic customization.

IDoc Interface

The IDoc processing SAP workflows located in task group `TG74500044` are declared as general, which means they can be started by all users. This step is performed through automatic customization.

Tips

Don’t set this to active as it will create a huge amount of unwanted work items that will eventually lead to massive performance issues! Instead, see KBA 2031151. If you’ve already set this to active by mistake and are facing performance issues when opening Transaction SBWP, see KBA 2164759 for instructions on how to remove the work items from users’ inboxes.

Unit Test

The workflows for the unit test contained in task group `TG56400001` are declared as general, as shown in Figure 2.15, which means they can be started by all users.

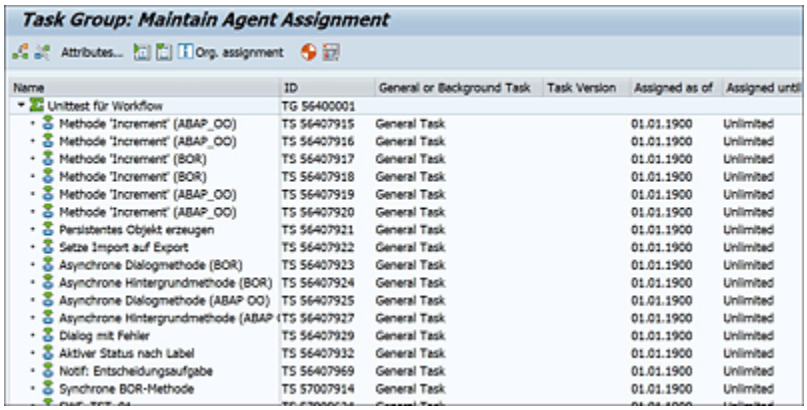


Figure 2.15 Task Group: Maintain Agent Assignment

SAPphone

The SAP tasks and SAP workflows that have SAPphone functions and are in task group `TG74500045` are classified as general. All users in the SAP System are allowed as possible agents. The agent restriction for the SAP tasks must be performed by selecting responsible agents in the step

definition. This step is performed through automatic customization.

Classify Scenario Tasks (Flexible Workflow) as General

For flexible workflows, all task used by the workflows and the workflows themselves must be declared as general tasks.

Processing of Replies to Appointment Request

The standard tasks in the workflow templates for processing replies to appointment requests are declared as general tasks. All SAP system users can be possible agents of these tasks. The tasks involved are `TS74508419`, `TS74508420`, `TS74508422`, `TS74508423`, `TS74508424`, `TS74508433`, `TS74508435`, `TS74508441`, and `TS74508469` in workflow template `WS74500804`. The tasks and the workflow template are contained in task group `TG74500047`. This step is performed through automatic customization.

2.4 Activating and Deactivating Standard Workflows

As you've seen, SAP provides a lot of standard classical workflows. The next step is activating or deactivating these workflows. In the previous section, you learned the details of activating system-related configurations for related workflows. Here, you'll see how to activate this for a specific standard workflow. This is done through Transaction SWE2 (Event Type Linkages).

The majority of standard workflows are triggered through standard events related to some standard business objects or classes. When a particular event occurs in the system (e.g., release of a purchase order) the corresponding event gets triggered, and this is the source of the event. When a particular receiver (in our case, the standard workflow) receives this event, the corresponding workflow gets triggered. This transaction basically makes a link between this event source and the event receiver. When this event is linked, then the corresponding workflow gets activated; if the link gets broken, then the workflow is deactivated.

Let's see an example of how this is done. Say you need to activate standard workflow `WS20000075` in the system. If you open the **Triggering events** tab of this standard workflow using Transaction PFTC, you'll see that this workflow gets triggered when event `RELEASESTEPCREATED` is triggered for standard business object `BUS2012`—this is the source of the event (see Figure 2.16).

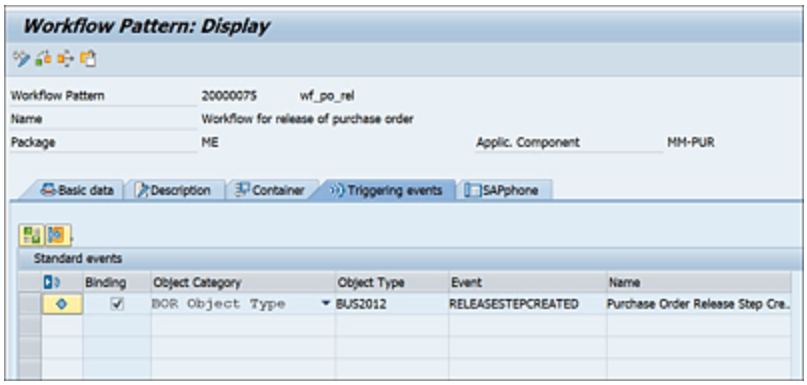


Figure 2.16 Triggering Event of Standard Purchase Release Workflow

Next, open Transaction SWE2, and browse business **Object Type BUS2012**, **Event RELEASESTEPCREATED**, and **Receiver Type WS20000075**. Select the **Linkage Activated** checkbox to complete the activation of this workflow. Figure 2.17 shows the event linkage between the workflow and the business object event.

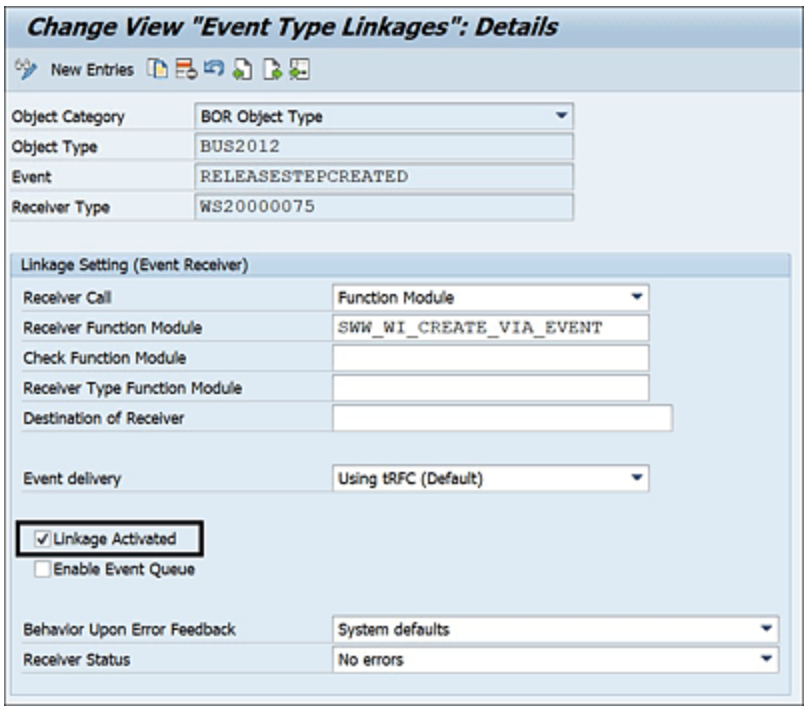


Figure 2.17 Event Linkage between Source Event RELEASESTEPCREATED (BUS2012) and Receiver Workflow Type WS20000075

Tips

If you want to add a custom check to decide whether the workflow should trigger or not, you can use the **Check Function Module** field. To determine the workflow template dynamically, you can use the **Receiver Type Function Module** field here. The details of these two function modules are shown in Figure 2.17.

2.5 Configuring Agents for Standard Workflows

In this section, you'll see how to configure agents for SAP standard classical workflows. There are different agent determination techniques for workflows. However, in the majority of standard classical workflows, the agents are determined using standard rules. But it's always necessary to check the agent determination technique first on any standard workflow. Let's see the details using the standard purchase order release workflow `WS20000075`.

Here, if you go with the standard task TS20000166, it basically sends the worktime to the corresponding agent who will release the purchase order. From the **Default rules** tab, you can see the standard agent is **20000027**, as shown in Figure 2.18.

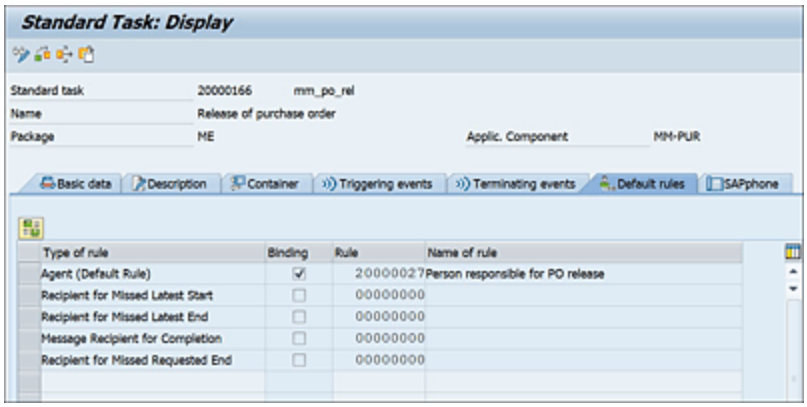


Figure 2.18 Standard Rule 20000027 Determines the Agent of Person Responsible for PO Release

For some cases, agent determination is done through functional configuration; for example, for the **Release of**

purchase order standard workflow, the agent determination is done through the purchase release group (**Grp**) and release code (**Code**) via configuration in a specific path under Transaction SPRO (see Figure 2.19).

Grp	Code	OT	Agent ID
01	Z1	US	USER1
01	Z2	US	USER2
02	Z1	US	USER3
03	Z2	US	USER4

Figure 2.19 Sample Agent Determination Based on Purchase Release Group and Release Code

However, if you want to add custom logic to determine agent using this rule, then there is a user exit (`EXIT_SAPLEBNF_005`) given inside function module `ME_REL_GET_RESPONSIBLE` (of the corresponding rule `AC20000027`), which can be leveraged.

> **Note**
>
> In this case, SAP has configured this customer exit to be triggered only when the **Role resolution for workflow** is set to **9**. This has to be maintained for the release group/release code combination in release code table `T16FC` (also maintainable through Transaction SPRO).

You'll learn the details about agent determination techniques in Chapter 6.

2.6 When to Develop a Custom Workflow

SAP provides a good number of standard workflows under different business application components. As an SAP technical consultant designing the requirements of a workflow process, the first target is to leverage SAP-provided workflows instead of developing something custom. However, not all requirements can be met using standard workflows; in some cases, you need some enhancement of the standard workflow or, at the end, may need to go for a fully custom solution.

First, we need to analyze the workflow business requirement and try to find out whether SAP has provided any standard business object (or class). If no such standard object is found, then we need to develop a custom workflow from scratch. If some standard business object is found, then we need to check whether SAP has delivered any standard workflow related to this business object. If no such workflow is found, then we need to build a custom workflow leveraging the corresponding standard business object. But if any standard workflow is found, then we first need to see whether the requirement can be met as is or with some enhancement requirement. If proper enhancement isn’t possible, then as the last option, we need to copy the standard workflow and change it accordingly based on the requirements. Figure 2.20 shows the guideline flow chart for this process.

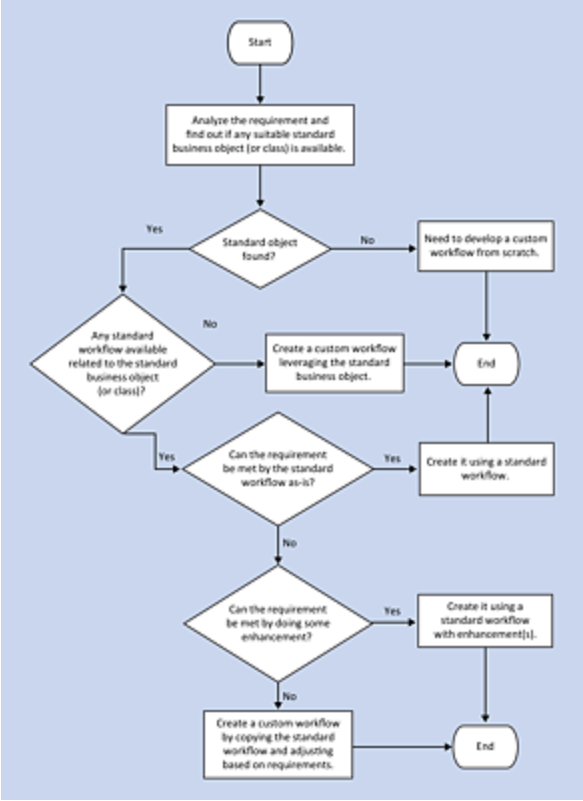


Figure 2.20 Flow Diagram Depicting When to Create a Custom Workflow

2.7 Summary

In this chapter, you've seen details about classical workflows. Though there are other workflows (e.g., flexible workflows and SAP Build Process Automation) in SAP S/4HANA, classical workflows are still significant in solution workflow requirements to a great extent. You've seen how classical workflows evolved from the very early stages of SAP R/3 to SAP S/4HANA and with multiple product evolutions and features. SAP has provided a good number of standard classical workflows, and we discussed how to search these workflows and what some of the commonly used standard classical workflows are in different ERP implementations. Then, we showed the details of system configurations from a readiness perspective, including how to activate standard workflows, how to configure agents, and so on. At the end, you saw that, in some cases, you may need to develop custom workflows. A flow diagram was included to show in what situations you should go for it.

3 Building Methods and Tasks

The building blocks of a workflow step are tasks and methods. This chapter begins with the concept of methods and tasks, exploring two approaches to building the code behind the workflow—the classic business object repository (BOR) approach and the ABAP class approach. You'll learn about building methods with both options in detail, including how to work with the parameters, handling exceptions, and performing unit testing. Finally, we'll discuss settings and options for standard tasks used in the workflow activity steps.

SAP Business Workflow was developed using an object-oriented approach in the early SAP R/3 days. During those days, object-orient programming (OOP) concepts weren't yet introduced by SAP. So, SAP developed a framework using business objects that gave the flavor of OOP to workflow, although it wasn't object oriented in a true sense. Later, as SAP introduced OOP in ABAP, SAP Business Workflow also adapted to a parallel OOP-based approach using ABAP classes and methods. Since then, SAP Business Workflow supports both the business object repository (BOR) approach as well as the ABAP class approach for building

the core business logic inside a workflow. Most existing standard workflows are built on the BOR programming approach, whereas new SAP add-on products, including flexible workflow, have moved toward the ABAP class approach.

In this chapter, we'll explore both programming techniques for workflow in detail. In both approaches, an activity step in a workflow needs a task definition, and a task definition calls a method to execute its processing. Tasks may be defined as dialog or background (see Section 3.1.4 for more details), and in turn the corresponding methods also perform actions in dialog (involving user interaction, i.e., agents) or in background (database query or update operations). Methods use parameters to read data from the task container and return data back to the task container, which in turn may be returned to the workflow container via binding (see Chapter 4, Section 4.4, for more details on containers and bindings).

Workflows make use of one or more object instances to work with the business data during its execution. However, all workflows use a primary object—often called the driving or leading business object—to carry out the various steps involved in the process flow. This leading business object usually receives the instance data from the triggering event via binding. Of course, workflows may be started even without a triggering event. In those cases, the leading business object must be instantiated by the workflow initiator (usually via ABAP code) and bound to the workflow or task container. Each BOR object or ABAP class instance uses one or more attributes to readily access data from the instance on work item texts, descriptions, and so on. These

attributes may also be read inside the task methods if required. Apart from methods and attributes, BOR type or ABAP class definitions also define events that may be used to trigger workflows or tasks, terminate work items, or create event or wait steps directly inside a workflow. In the next section, we'll study each of these components in detail from the perspective of a business object definition.

3.1 Business Object Repository Approach

A business object type definition is a framework in SAP S/4HANA that allows you to work with the data for an application. The business object layer acts like an interface above the database layer and helps in managing the data record for an application transaction; for example, the business object type for managing sales order data is `BUS2032`, and for purchase orders, it's `BUS2012`. A business object is the runtime instance of the object type definition. The definition of a business object type consists of key fields, attributes, methods, and events. Object type interfaces help group some common attributes, methods, and events, making them available in the object type definition. Figure 3.1 illustrates the basic components of an object type definition via Transaction SWO1.

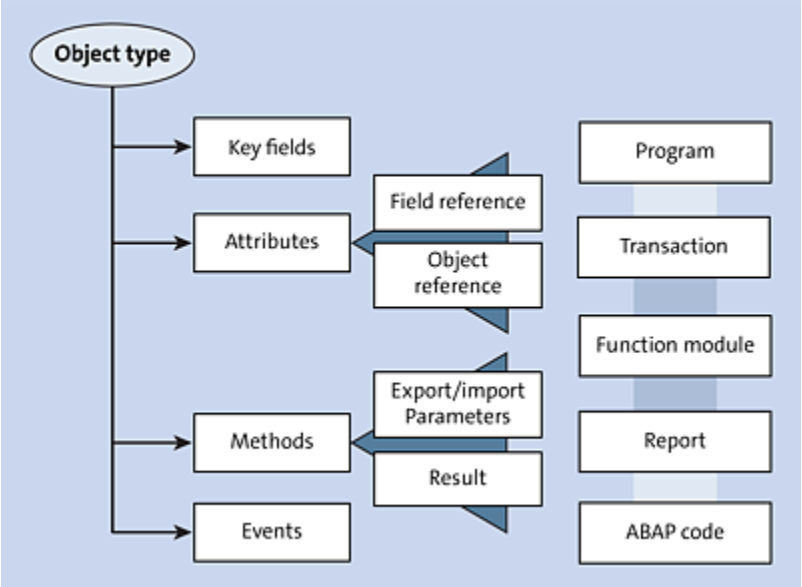


Figure 3.1 Definition of a Business Object Type with Key Fields, Attributes, Methods, and Events

In the following sections, we'll create a custom BOR object type and add some components to it. We'll go through the steps for creating each type of component in detail. First, however, we'll walk you through each element of a business object type definition.

3.1.1 Business Object Type Definition

The business object type definition integrates with the Business Application Programming Interface (BAPI), wherein specific methods of the object type are built on BAPI, calls and the object type/method names are linked to the BAPI Explorer as well. For instance, if you're working with sales orders, then you can search for all BAPIs related to sales orders in the BAPI Explorer. In Figure 3.2, and Figure 3.3, respectively, you can see that the sales order object is

linked to business object type BUS2032, and each BAPI under that object type, is nothing but a method.

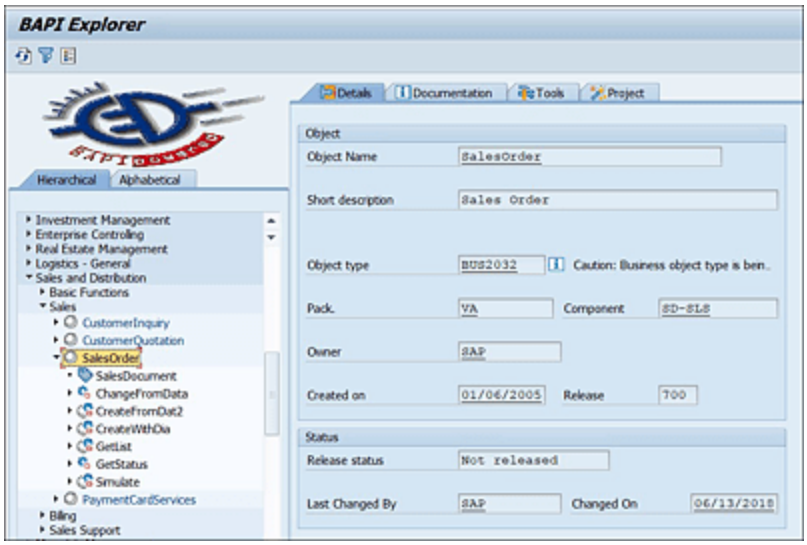


Figure 3.2 Sales Order Business Object Type from BAPI Explorer

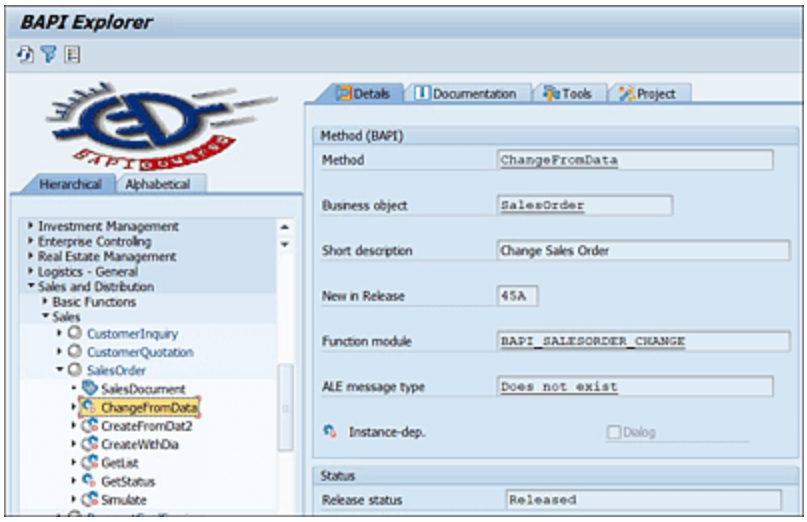


Figure 3.3 Method ChangeFromData under SalesOrder Object Type Showing the Link to BAPI

You can view the same methods from Transaction SWO1 for business object type `BUS2032`, as shown in Figure 3.4.

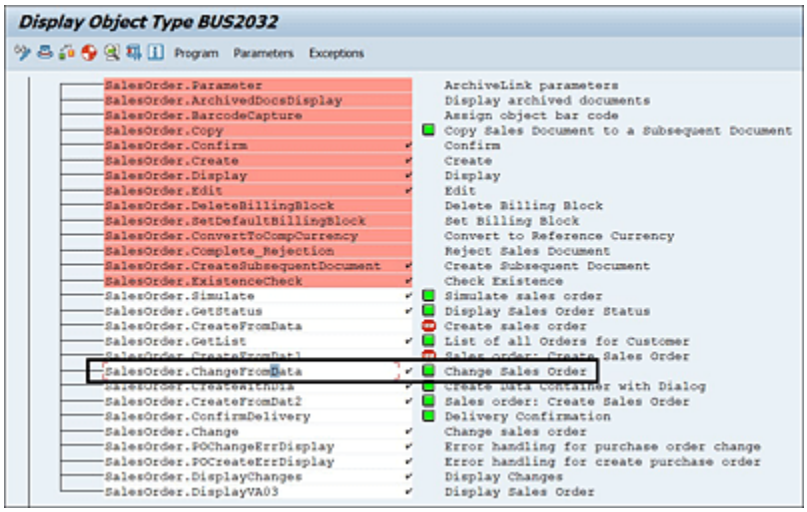


Figure 3.4 View of Methods under a Business Object Type Definition in Transaction SWO1

If you double-click on a method and navigate to the **ABAP** tab, you can see the link to the BAPI call as an application programming interface (API) function from the method, as shown in Figure 3.5.

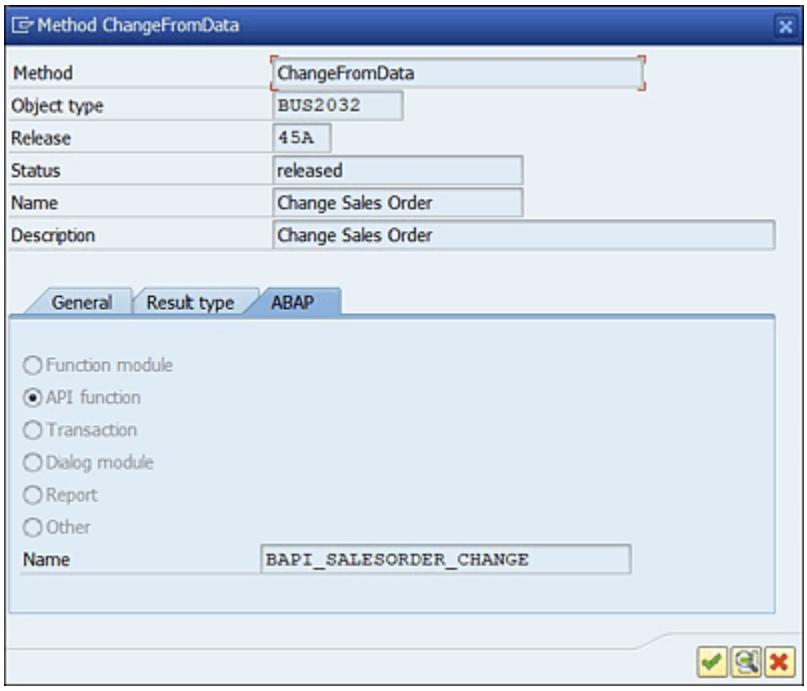


Figure 3.5 Method Definition in a Business Object Type Showing the BAPI Call as an API Function

A business object type definition (often referred to as a BOR type) in SAP consists of the following components (not all are mandatory for an object type definition):

- **Interfaces**
  Interfaces are similar to an object type definition, which are used for grouping other interfaces, attributes, methods, and events. Interfaces don't have any key fields, so they can't be instantiated. They are used for grouping and adding some common functionality to an object type definition. Interfaces can't be used on their own; instead, they must be assigned to an object type definition, and then the object type can use the components of the interface. Every business object type definition comes with some SAP standard interfaces, for instance, `IFSAP`, `IFEXIST`, and so on, as shown in Figure 3.6. The `IFSAP` Interface provides each object type with the following components:
  - Attribute `ObjectType`
  - Method `Display`
  - Method `ExistenceCheck`

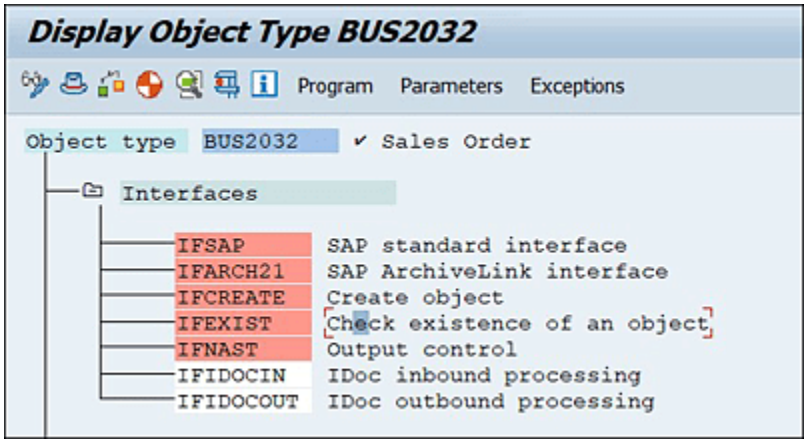


Figure 3.6 View of Interfaces Defined in a Sample Business Object Type

- **Key fields**
  The key fields of an object type are attributes used for creating an instance of the object type. The key field values must be supplied from the calling application when instantiating a business object. Instance-dependent methods can access the key fields of an object type via work area `OBJECT-KEY-<key field>`. An object type can have more than one key field in its definition, but the combined object key must not exceed 70 characters in length, as this is the maximum length of the object key in a BOR type. Key fields refer to a primary table field, based on which the object instance is created. Usually, the primary key of the table matches the object key, with few exceptions such as `BUS2001` and `BUS2054`. Figure 3.7 shows an example of a key field in a business object type definition.

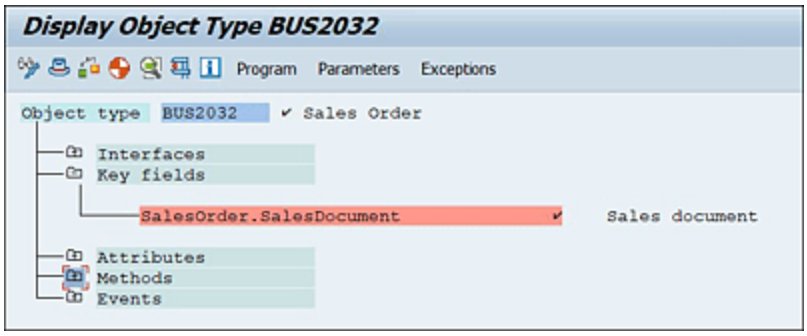


Figure 3.7 Definition of Key Field in a Business Object Type

- **Attributes**
  Attributes provide access to the object data, just like class attributes. Database attributes are non-key fields of the primary object table (defined earlier in key fields). They can be fetched directly based on the object type key field and don’t require any additional code. You just need to enter the table and field name to be selected. The table

name entered must be the primary object table or another table with the same key or partial key of the primary table. The field name entered can be any field from the selected table. Virtual attributes allow you to define your own logic to fetch additional data values. These values may be scalar or tabular. You must write your own code in the BOR object type program to populate these attributes. Figure 3.8 shows a sample list of attributes from BOR type `BUS2032`, and Figure 3.9 shows the common settings of an attribute. Here the source setting of the attribute defines the attribute as **Database field** or **Virtual**. The attribute properties checkboxes define whether an attribute is scalar or tabular (multiline), mandatory or optional, and instance independent or dependent.

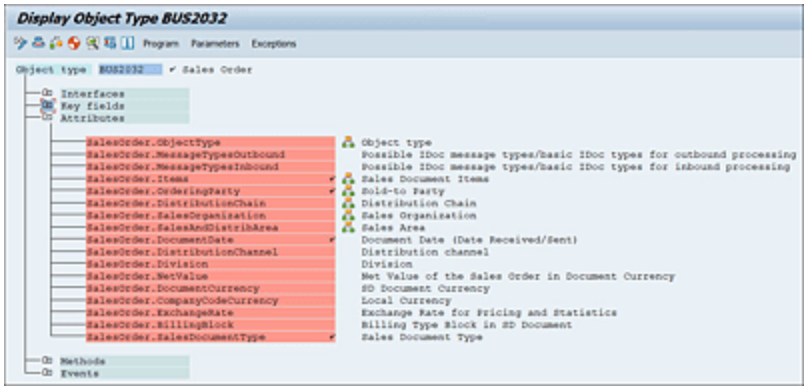


Figure 3.8 Definition of Attributes in a Business Object Type

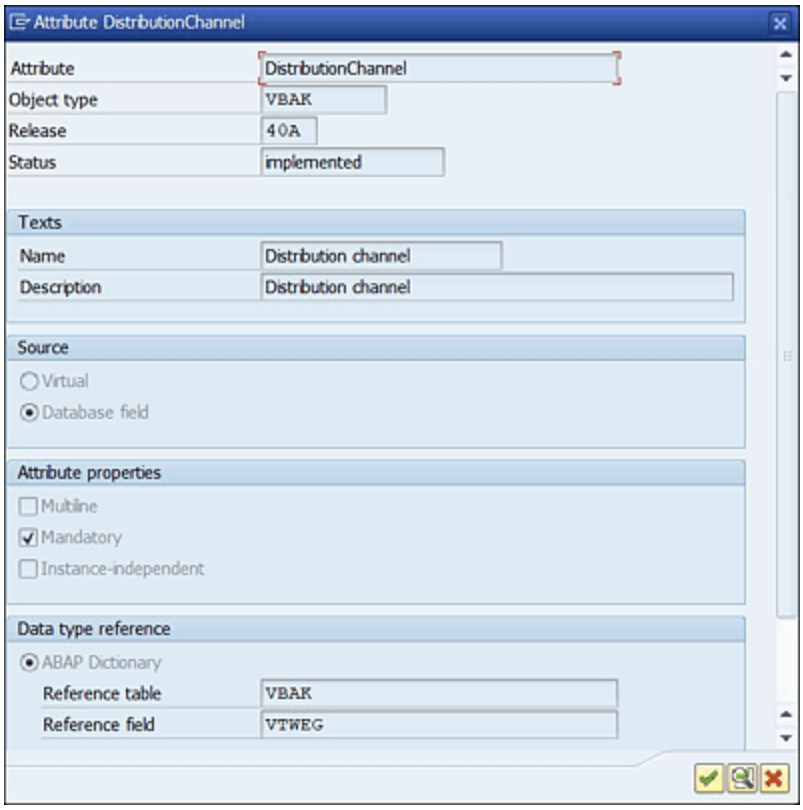


Figure 3.9 Attribute Definition Showing Database/Virtual, Multiline, Reference Table/Field Properties

Note

Another special category of attributes called **Status** is applicable to some BOR types. To make this attribute category available to your BOR type under the attribute **Source** section, you must add the interface `IFSTATUS` to your object type definition. Then, you'll be able to see this option while creating a new custom attribute. You need to enter the status number while defining the attribute, and SAP will automatically populate the attribute while creating the object instance, if the status is set for the status object number (attribute `StatusObjNumber`). Check standard BOR type `BUS2007` for an example.

- **Methods**
  Methods are procedures that can be used to fetch additional data from an object or execute some actions on a BOR object. For instance, method `GetStatus` of BOR type `BUS2032` fetches the complete status information of a sales order. Another method, `ChangeFromData`, of the same BOR type allows you to change an existing sales order.
  Methods can have import and export parameters as well as exceptions defined in them.
  Methods can be defined as *dialog* or *background*. Figure 3.10 shows the **Dialog** checkbox under the **General** tab of method attributes. If you select the checkbox then a method is defined as **Dialog** else it's background. Dialog methods are executed in foreground via a work item in a user's inbox, and they normally call some transaction to allow the user to make necessary changes. Background methods are executed by the workflow runtime system, and they are used to fetch data from database tables or execute some action in background, usually with BAPI calls. Methods with a result parameter can output a value just like a virtual attribute.
  Instance dependent or independent methods define whether the method logic depends on the object key or not. Instance independent methods (like static methods in ABAP class) don't receive the object instance at the time of execution, they only depend on import parameters to receive data.
  Methods can also be classified as *synchronous* or *asynchronous*. Figure 3.10 shows the **Synchronous** checkbox under the **General** tab of method attributes. If you select the checkbox, then a method is defined as synchronous; otherwise, it's asynchronous. Synchronous

methods finish executing before handing control back to the calling program. These types of methods can return export parameters, results, and exceptions. Asynchronous methods don’t return control back to the calling program immediately. Once executed, they depend on terminating events to communicate the results back to the calling program. Asynchronous methods may not have result parameters, export parameters, or exceptions.

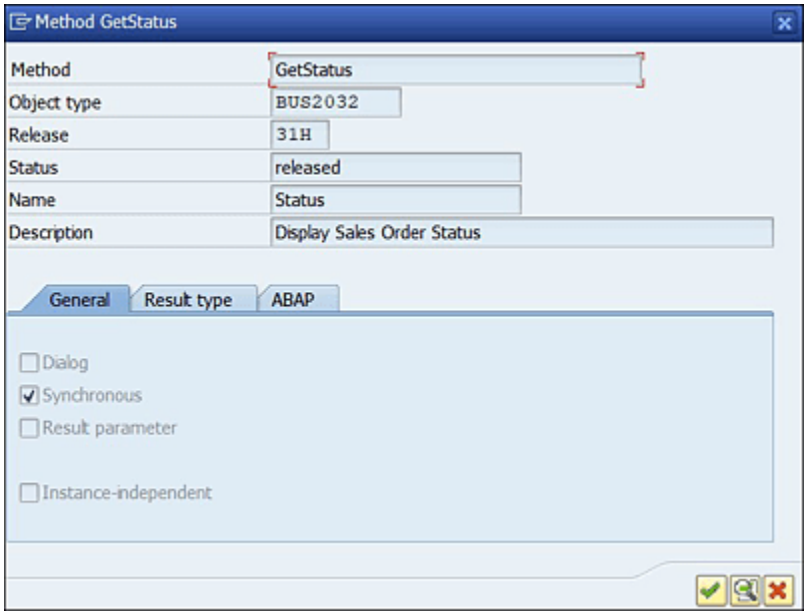


Figure 3.10 Method Definition Showing Dialog, Synchronous, Result Parameter, and Instance Independent Properties

Figure 3.10, Figure 3.11, and Figure 3.12 illustrate the common settings in a method definition. The settings on the **General** tab of the method definition define whether a method is dialog/background, is synchronous/asynchronous, has a result parameter or not, and is instance independent or dependent. The radio button options on the **ABAP** tab lets you choose between different predefined templates of generating the code for a method. You can choose between function module call,

API function call (BAPI), transaction code, dialog module, or report program submit call. If you want to write your own code in the method without using a template, then choose the **Other** option. Method parameters, as displayed in Figure 3.12, can be maintained by clicking on the **Parameters** button on the toolbar in Transaction SWO1 after clicking once on the method name.

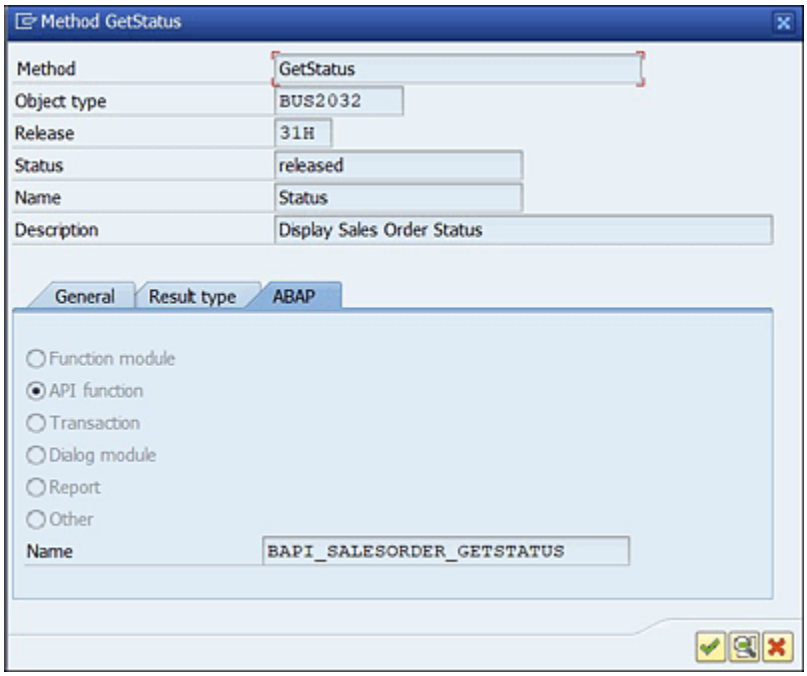


Figure 3.11 Method Definition Showing the Different ABAP Coding Options Available

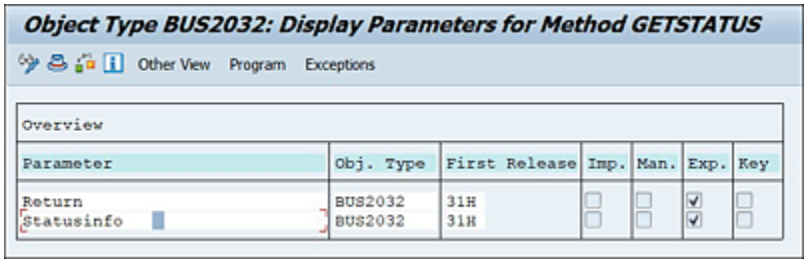


Figure 3.12 Method Parameter Definition in a BOR Type

- **Events**
  Events are components of an object type that define the

possible change in status of the object. For example, event CREATED of BOR type BUS2032 gets triggered when a sales order is created. Similarly, event CHANGED is triggered when a sales order is changed. Events can have import parameters, but no export parameters or exceptions. There is no ABAP code written behind the event definition. Event definition contains just the identifier name (and possible import parameters). Event publishing must be handled via other mechanisms, which are discussed in detail in Chapter 5. Similarly, event receiver linkage is also handled outside the scope of the BOR definition. Figure 3.13 shows some examples of event definition in BOR type BUS2032, and Figure 3.14 shows the parameter definition in an event. Event parameters may be maintained by clicking once on the event name and then clicking on the **Parameters** button in the application toolbar of Transaction SWO1.

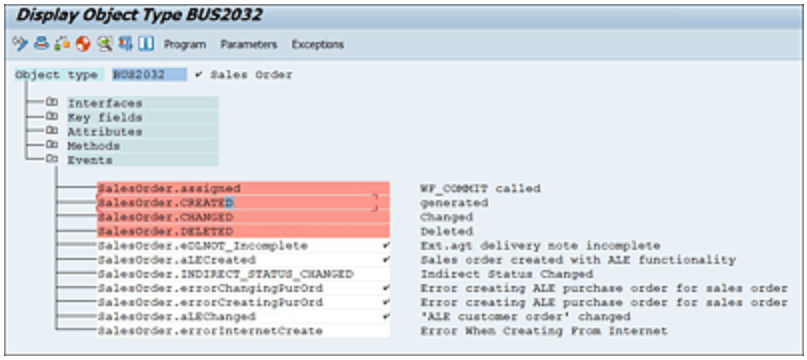


Figure 3.13 View of Event Definition within an Object Type

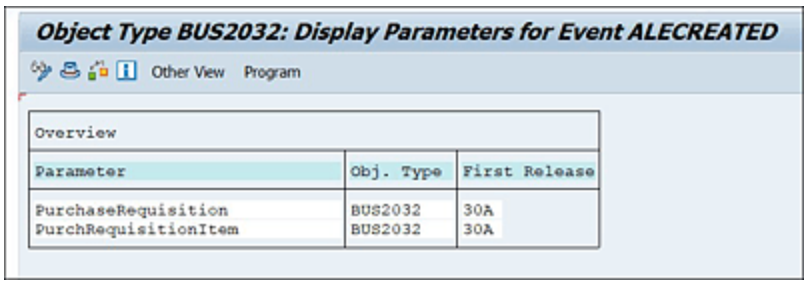


Figure 3.14 Definition of Import Parameters in an Event on a BOR Type

- **BOR program**
  Every business object type definition includes a source code program, which is maintained at the header level of the definition in the **Program** field (see Figure 3.15). This program contains all the necessary ABAP source code behind all the previously discussed component definitions, that is, key fields, attributes, and methods.

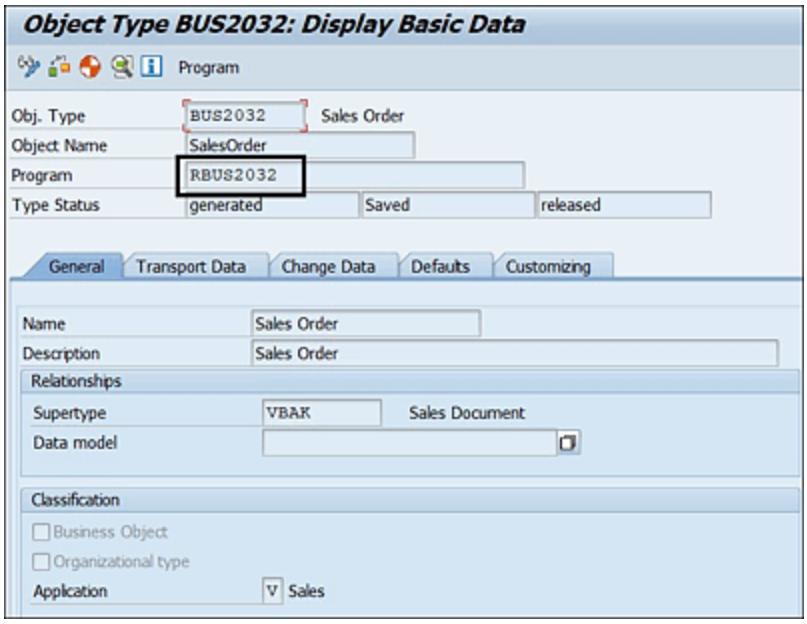


Figure 3.15 BOR Program Name Maintained at the Header Level

- **Supertype**
  Business object types may be inherited from other business object type definitions using a parent-child relationship. In this context, the parent object type is referred to as the supertype, and the child object type as the subtype. For example, standard BOR type `BUS2032` is the subtype for object type `VBAK`. Here, `VBAK` is referred to as the supertype. The supertype name is maintained at the object type definition header level, and it can't be

edited once the child object type has already been created and generated. When you enter a supertype in an object type definition, all the components from the parent (interfaces, key fields, attributes, methods, and events) are automatically inherited into the child definition. These components appear as red in the child object type and can’t be edited directly as the definition and source code of these components still point to the parent. However, you can redefine the attributes, methods, or events in the child object type. In that case, the component definition and source code are copied into the child object type BOR program and become available for changes. Figure 3.16 shows an example of a custom BOR subtype with inherited components and new custom components, which are shown on the screen in red and white, respectively.

Note

You should be very careful about redefining the components inherited from a SAP standard business object type into your own custom object type, as these may affect standard programs where the parent object is used and when delegation is maintained from the supertype to the subtype definition. In this case, we always recommend creating your own custom attribute/method/event and adding the required logic in that component instead of redefining the standard. This approach ensures that there is no impact on the standard programs due to your custom changes. Redefinition of methods may be required, however, for methods inherited from interfaces, for example, `ExistenceCheck`, which is

inherited in all object types from the standard interface `IFSAP` and needs individual BOR-specific logic to be implemented here.

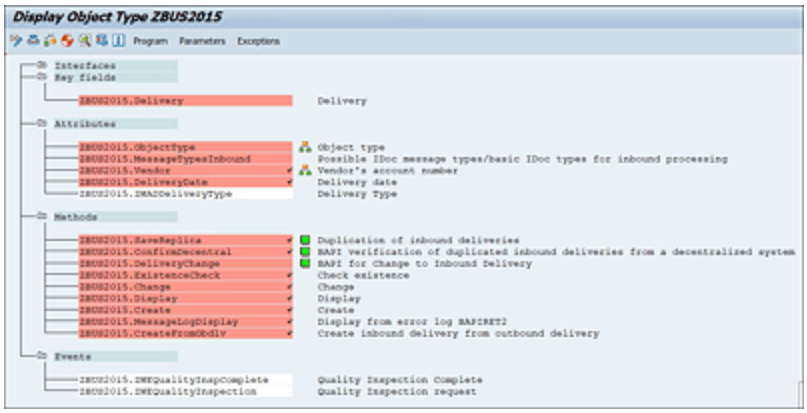


Figure 3.16 Custom BOR Subtype Definition Showing Inherited Components

Note that you can create as many subtypes (child) as you want from a supertype (parent) definition. But you can only delegate the parent to any one of those subtypes. Delegation will be covered in more detail in Section 3.1.8.

If you need to change the link to a supertype from a subtype definition, then follow menu path **Object Type** • **Inheritance** • **Change Supertype** from Transaction SWO1. This will allow you to enter a new supertype name to your current object type. However, note that this action will remove the inherited components from your previous supertype and replace them with the new one.

3.1.2 Defining a Custom Business Object Repository Object Type

In the example in this section, you'll be creating a custom BOR type for an outbound delivery in SAP. In standard SAP,

there’s already an object type available for deliveries called `LIKP`. However, if you want an object type specific to outbound delivery (`LIKP-VBTYP = J`), then you can opt for a custom BOR type definition. You can also create a subtype for outbound delivery, inheriting from the supertype `LIKP` (this approach will be discussed in Section 3.1.7).

Note

Normally, we don’t recommend creating a completely custom BOR object type definition without any link to a standard supertype in SAP. In this case, you should go for the ABAP class-based approach (explained in Section 3.2). Even though there are no technical constraints with the BOR approach, the class-based approach provides a cleaner object-oriented coding style that is compatible with the latest ABAP coding standards suitable for SAP S/4HANA. BOR programming deals heavily with macros, which are slightly difficult to follow and debug compared to OOP.

Before creating a custom BOR type in Transaction SWO1, you should always search for available standard object types that you can reuse or extend with a subtype if required. To search for standard BOR types, you can use the BOR that can be accessed via the same toolbar option from Transaction SWO1 or directly via Transaction SWO3. Figure 3.17 shows a view of the BOR in SAP with object type hierarchies grouped by application components.

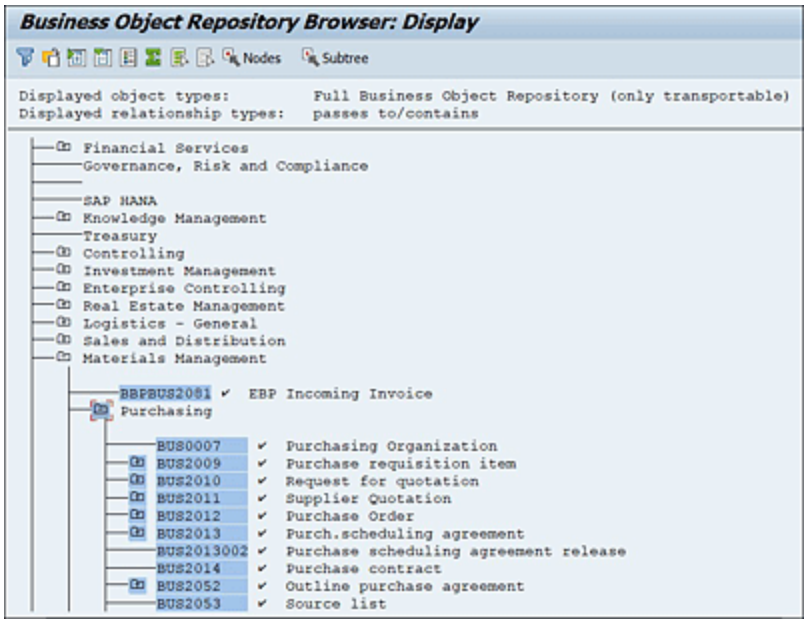


Figure 3.17 View of the BOR in SAP with Object Type Hierarchies Grouped by Application Components

To create a custom BOR type definition in SAP, go to Transaction SWO1, enter the name for your custom **Object type**, and click on the **Create** button. In the next popup screen, shown in Figure 3.18, enter the **Object name**, **Name**, **Description**, and BOR **Program** name, along with **Application**. Because you're creating a completely custom BOR type, you can leave the **Supertype** field blank.

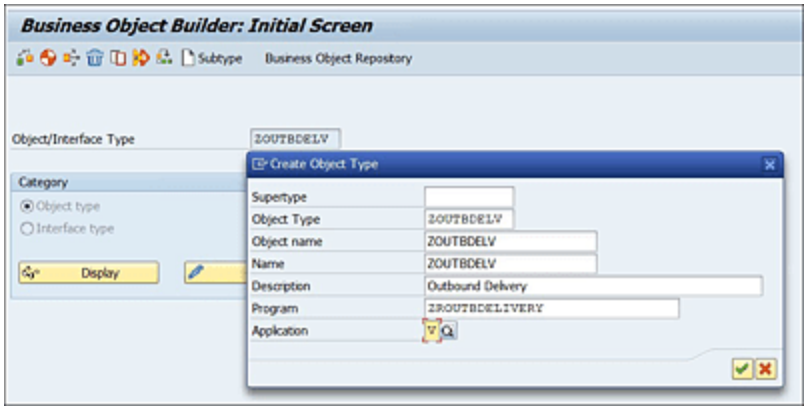


Figure 3.18 Creating a Custom BOR Type from Transaction SWO1

After entering these details, press Enter, and then enter the package and transport details when prompted. The object type definition is initially created in **Modeled** status, which is indicated by the highlighted icon beside the object type name in Figure 3.19.

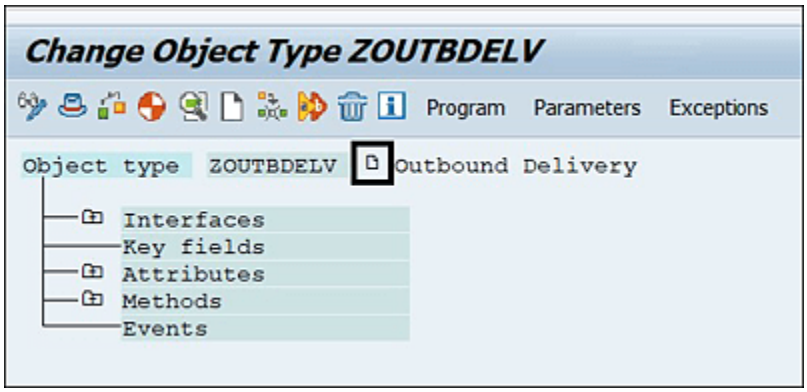


Figure 3.19 Object Type Definition Created in Modeled Status

You must now change the object type status from **Modeled** to **Implemented** by following menu path **Edit** • **Change Release Status** • **Object Type** • **To Implemented**, resulting in the screen shown in Figure 3.20.

Next, you can save and generate the object type definition. In the BOR builder, the **Generate** button acts like the **Activate** button in regular ABAP programs.

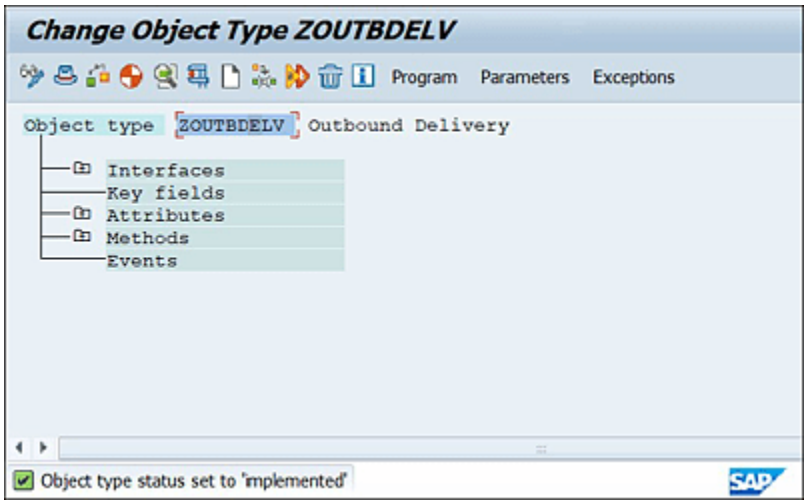


Figure 3.20 Object Type Status Changed to Implemented

3.1.3 Creating Key Fields and Attributes

The next step is to create the key field(s) for the newly defined object type `ZOUTBDELV`. You can either right-click on the **Key fields** node and choose **Create**, or you can click on the node once and click on the **Create** button on the toolbar inside Transaction SWO1. You'll receive a popup message, as shown in Figure 3.21. It doesn't matter which option you choose while creating the key field. Both have the same result in terms of object type definition. In this case, select **Yes** and proceed.

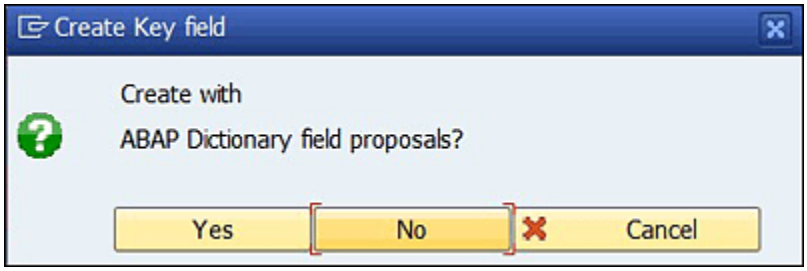


Figure 3.21 Popup to Choose between Database or Virtual Definition of the Key Field

Once you select **Yes** in the popup, another popup appears asking you to enter the primary key table and then select the key field(s) from this table. In this case, choose “LIKP” as the primary **Table** and **VBELN** (**Delivery**) as the key field (see Figure 3.22).

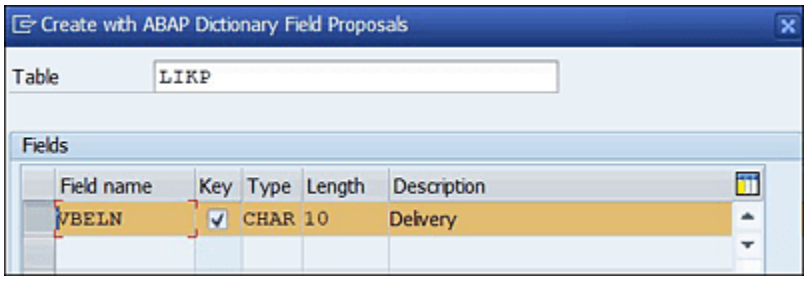


Figure 3.22 Selection of the Primary Key Table and Key Field

Press Enter after selecting the primary key, then confirm/update your key field name per your naming standards, and click on the **Create** button to proceed. Figure 3.23 illustrates this step for this example scenario, and Figure 3.24 shows the result.

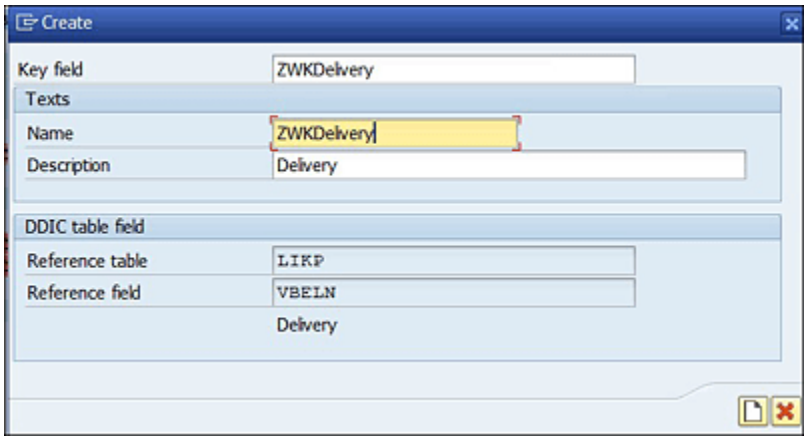


Figure 3.23 Confirming the Key Field Name

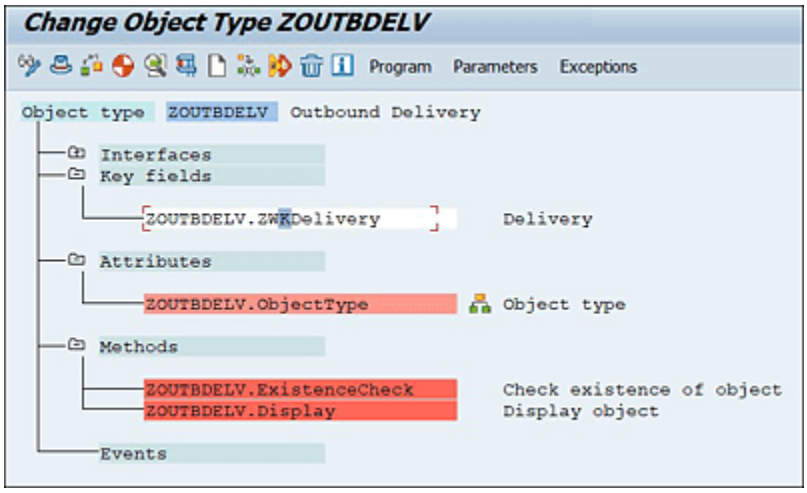


Figure 3.24 View of Object Type Definition after Creating the Key Field

Finally, you can save and generate the object type definition.

Next, you'll create two attributes in the object type—one database and one virtual. You'll create a database attribute for **Delivery Type** (`LIKP-LFART`) because it's based on table `LIKP`, and you'll create a virtual attribute for **Sales Order Type** (`VBAK-AUART`), which requires some custom ABAP coding.

Right-click on the **Attributes** node of the business object type definition, and click **Create** or choose the **Create** button from the toolbar. In the popup that appears for the **Dictionary** field proposal, choose **Yes**. Then, select the field name from the list of available fields in table `LIKP`, as shown in Figure 3.25.

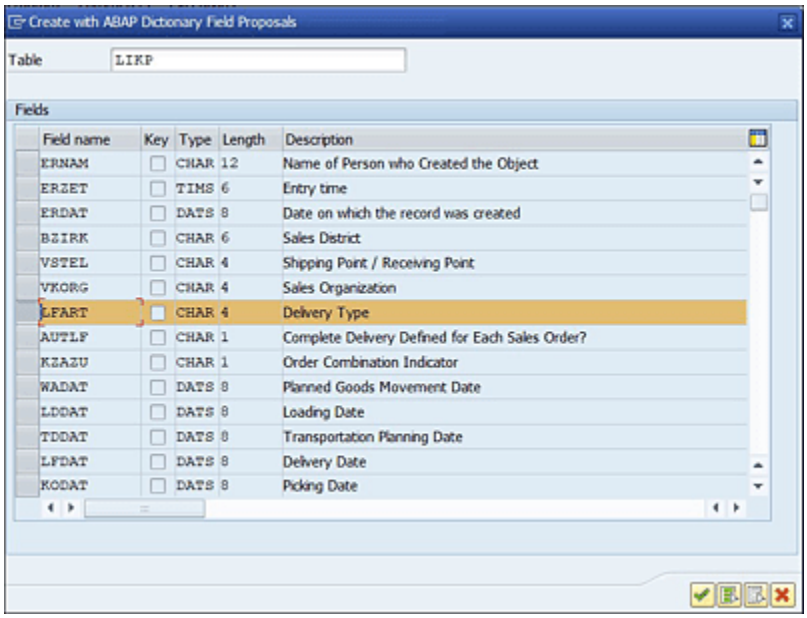


Figure 3.25 Selection of Table Fields for the Database Attribute Definition

Enter the name and description of the new database attribute to be created, as shown in Figure 3.26.

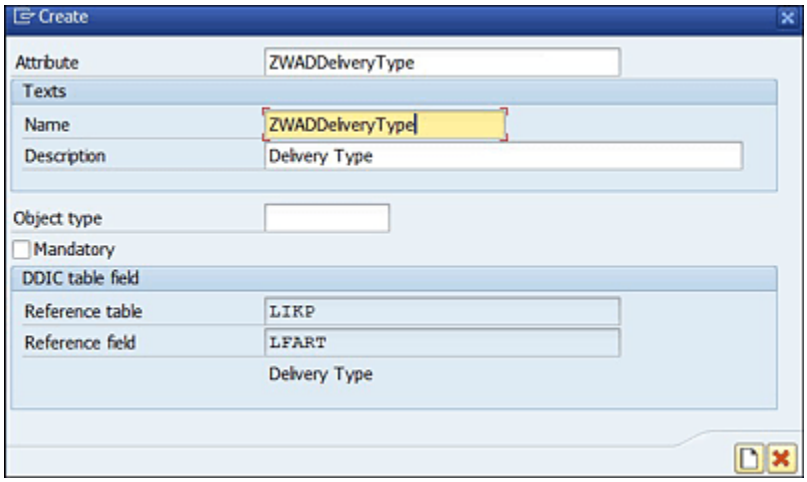


Figure 3.26 Confirm the Database Attribute Name per Naming Standards

Figure 3.27 shows the view of the object type definition after creation of the new database attribute.

Next, click once on the attribute name, and then click on the **Program** button from the toolbar. You'll be prompted to

generate a template code automatically for the new database attribute. Select **Yes**, as shown in Figure 3.28.

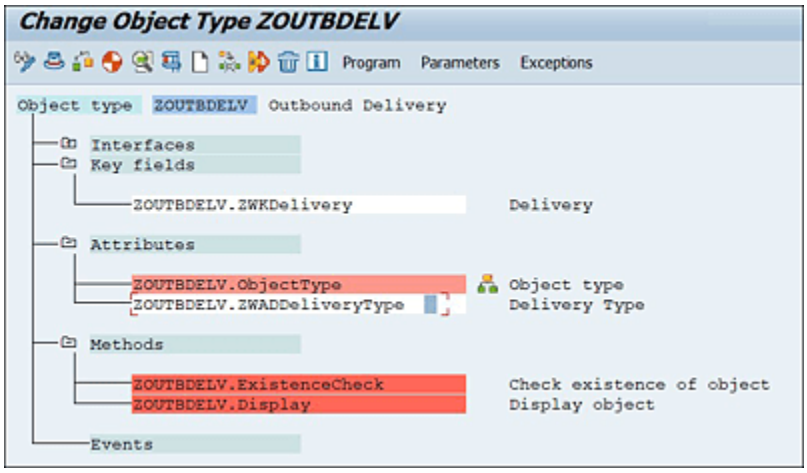


Figure 3.27 Object Type Definition after Creation of the Database Attribute

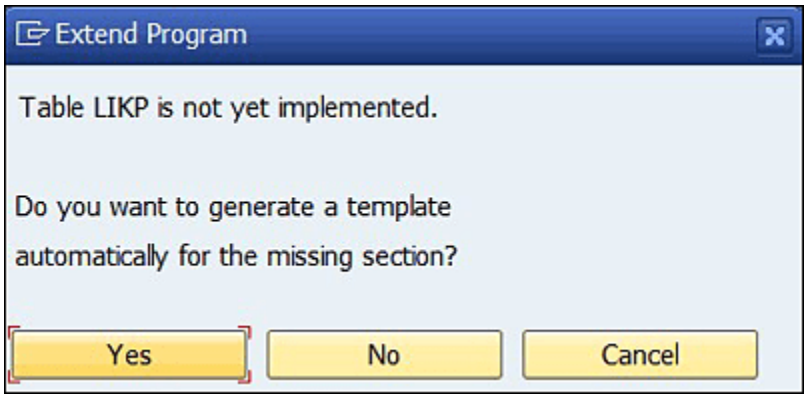


Figure 3.28 Popup to Automatically Generate Template Code for the Database Attribute

Once you click on **Yes**, the template source code to fetch data from the attribute source table is auto-generated, as shown in Figure 3.29.

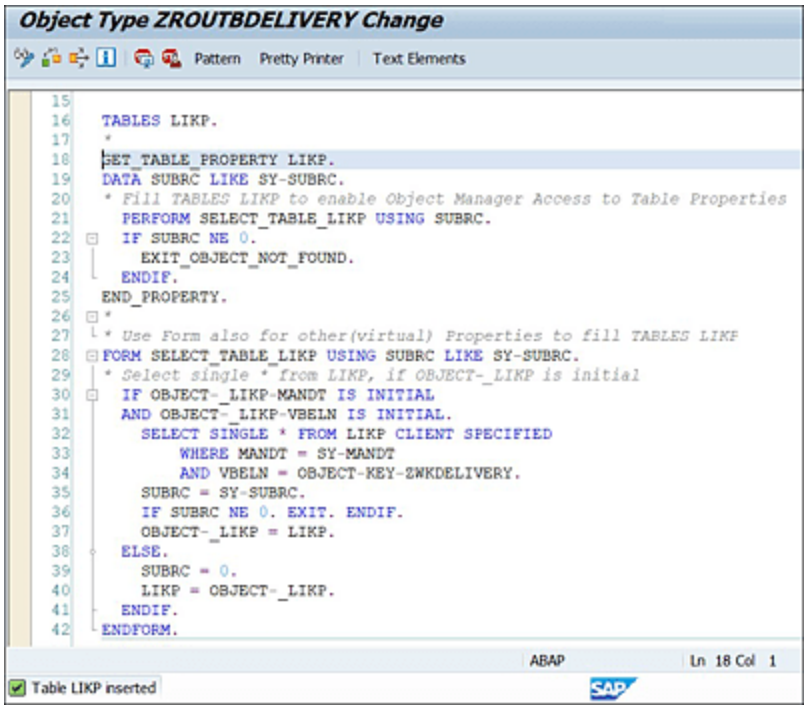


Figure 3.29 View of Automatically Generated Source Code for Database Attribute

Now go back, save, and generate the object type.

Note

The generation of template code for the database attribute is usually required only once in an object type definition. As you can see from Figure 3.29 just shown, the code is a generic `SELECT` on table `LIKP` based on the key field, which is the delivery number. If you add another database attribute from table `LIKP`, for instance, shipping point (`LIKP-VSTEL`), then the same `SELECT` query will be reused to populate this attribute.

Next, you'll create a virtual attribute for the sales order type (`VBAK-AUART`). Click on the **Attributes** node, and select **Create**. In the popup for the ABAP Data Dictionary (DDIC)

field proposal, choose **No** this time. Confirm the name of the **Attribute**, and maintain the data type properties as shown in Figure 3.30. Enter the attribute name, description, and reference table field under **Data Type Reference**.

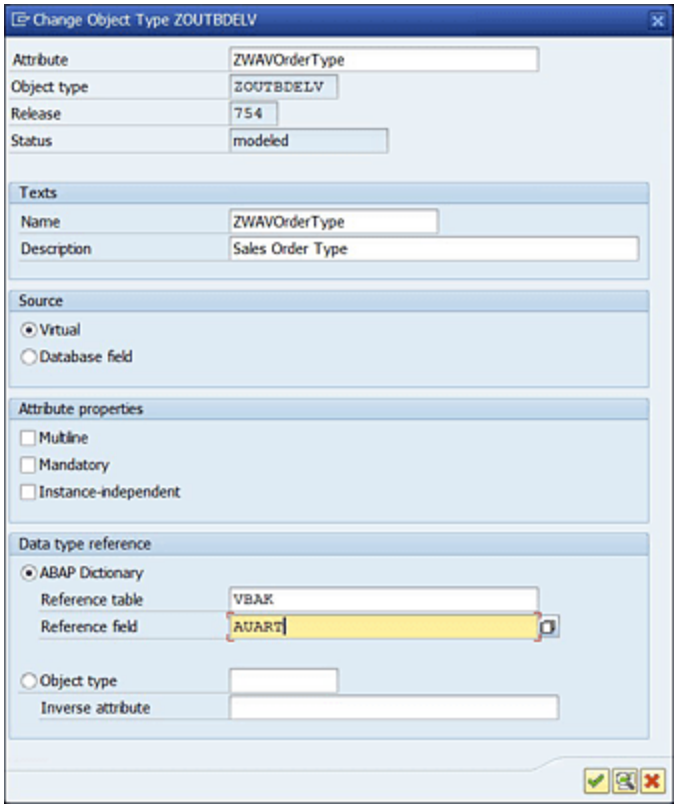


Figure 3.30 Definition of a Virtual Attribute in the Object Type

The new virtual attribute is created in **Modeled** status (page icon beside the attribute name). Figure 3.31 shows the view of the business object type after creation of the new virtual attribute.

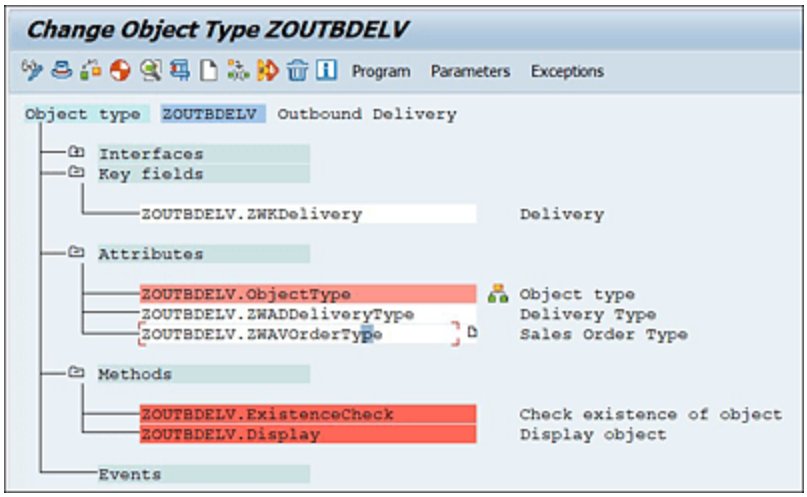


Figure 3.31 View of a Newly Created Virtual Attribute in Modeled Status

Change the status of the attribute to **Implemented** by following menu path **Edit** • **Change Release Status** • **Object Type Component** • **To Implemented**.

Now, you must maintain the source code for the virtual attribute. Before you do that, the BOR framework helps in generating an empty template code for the new attribute. Click on the attribute once, and then click the **Program** button on the toolbar. Select **Yes** in the popup prompt asking to generate a template, as shown in Figure 3.32.

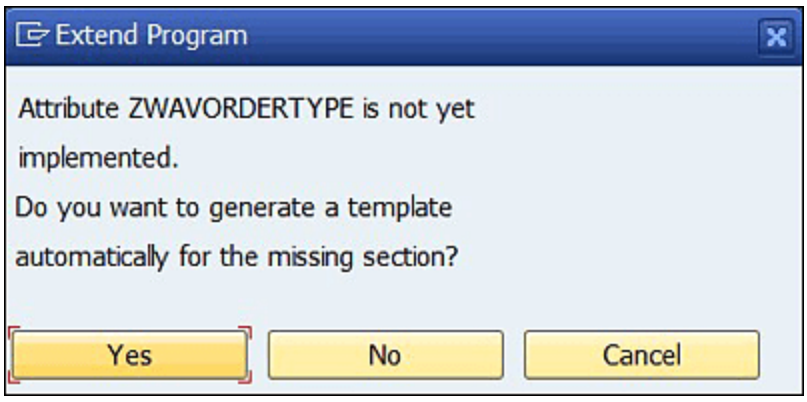


Figure 3.32 Popup Prompt to Generate an Empty Template Section for the Virtual Attribute

Figure 3.33 shows the template source code generated for the new virtual attribute. The template code contains a single line with a macro to populate attribute ZWAVOrderType between the GET_PROPERTY and END_PROPERTY section.

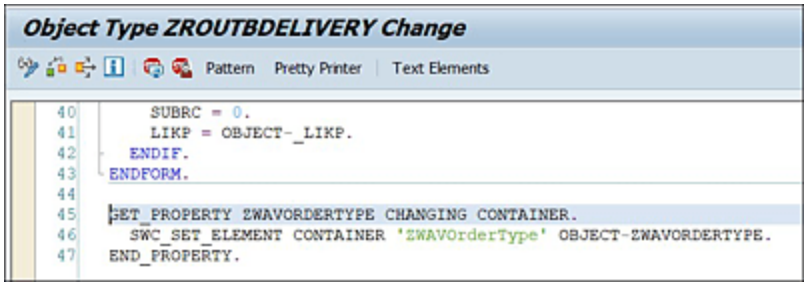


Figure 3.33 Generated Template Code Section for the Virtual Attribute in the BOR Program

Now you must write the source code with custom logic to populate the order type attribute. Listing 3.1 is a sample code for reference. You'll learn more about BOR programming techniques and about the common macros in Section 3.1.9.

```
get_property zwavordertype changing container.

***Fetch order type from preceding Sales order document
***Here we assume that each outbound delivery is created from a
***single sales order
SELECT FROM lips    " SD document: Delivery: Item data
   INNER JOIN vbak   " Sales Document: Header Data
   ON vbak~vbeln = lips~vgbel
   FIELDS vbak~auart " Sales Document Type
   WHERE lips~vbeln = @object-key-zwkdelivery
   INTO @DATA(lv_auart)
   UP TO 1 ROWS.
ENDSELECT.
IF sy-subrc = 0.
   object-zwavordertype = lv_auart.
ENDIF. " IF sy-subrc = 0
swc_set_element container 'ZWAVOrderType' object-zwavordertype.
end_property.
```

Listing 3.1 Sample ABAP Source Code for the Virtual Attribute

Go back to save and generate the object type definition.

3.1.4 Creating Methods and Defining Properties, Parameters, and Exceptions

Continuing with the development of the custom business object type, you'll now redefine inherited method `ExistenceCheck` and create a new custom background method for goods issue of an outbound delivery.

To redefine method `ExistenceCheck`, right-click on the interface method, and select **Redefine**. Then, click once on the method name, and then click on the **Program** button on the toolbar. A popup screen appears requesting confirmation to generate the template code for implementation of method **ExistenceCheck**. Click **Yes** and continue, as shown in Figure 3.34.

Figure 3.35 shows the template code being added to the BOR program for the redefined method with the `BEGIN_METHOD` and `END_METHOD` keywords. You need to add the custom logic for the method between these two lines.

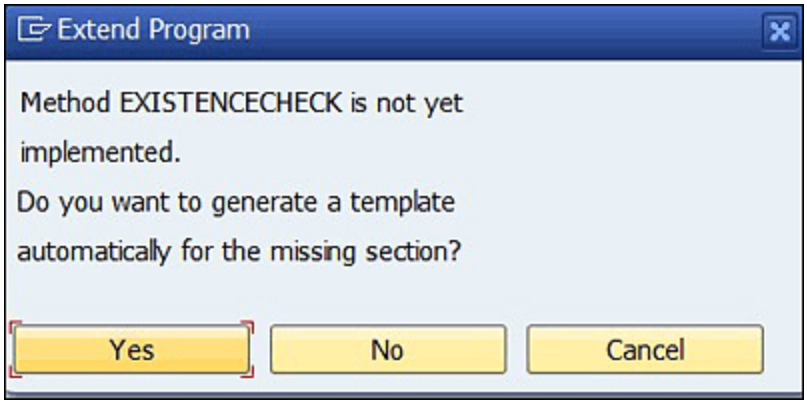


Figure 3.34 Popup Confirmation to Generate a Template Section for Method Implementation

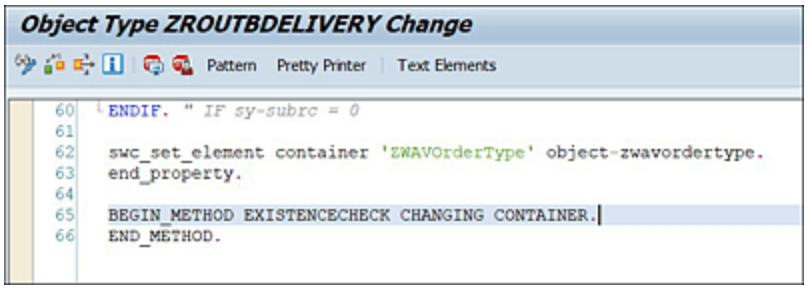


Figure 3.35 Template Section Inserted in the BOR Program with Method Implementation

Now you can insert your own code within this method implementation. The sample code in Listing 3.2 is just for reference. We'll discuss BOR programming in Section 3.1.9.

```
begin_method existencecheck changing container.
 *Check if object exists and is a valid outbound delivery
SELECT FROM likp " SD Document: Delivery Header Data
  FIELDS COUNT(*)
  WHERE vbeln = @object-key-zwkdelivery
  AND   vbtyp = 'J'.
IF sy-subrc <> 0.
  exit_return 0001 space space space space.
ENDIF. " IF sy-subrc <> 0
end_method.
```

Listing 3.2 Sample Source Code for Method ExistenceCheck of the Outbound Delivery Object Type

Go back to save and generate the object type.

Next, you'll create a new method for goods issue of the delivery. You can either right-click on the **Methods** node and select **Create**, or you can click on the **Create** button from the toolbar. You'll receive a popup message asking if you would like to create with the function module as template (see Figure 3.36).

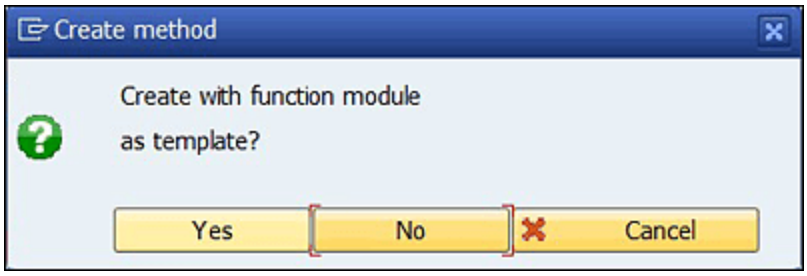


Figure 3.36 Popup to Confirm if the Method Should Be Created with a Function Module Call

This option lets you generate the method based on an API call. This approach is particularly useful when you've already modularized your method logic in a function module or when a standard BAPI or API call meets the requirement of your method. If you select **Yes**, then in the next screen, you'll be asked to select the function module name and then select the import and export parameters that should be mapped to the method interface. The generated template code will map the import parameters from the method to the function module and the export parameters from the function module back to the method export parameters.

If you select **No**, then you must write the method code yourself. Confirm the method name per your naming standards and choose the method attributes.

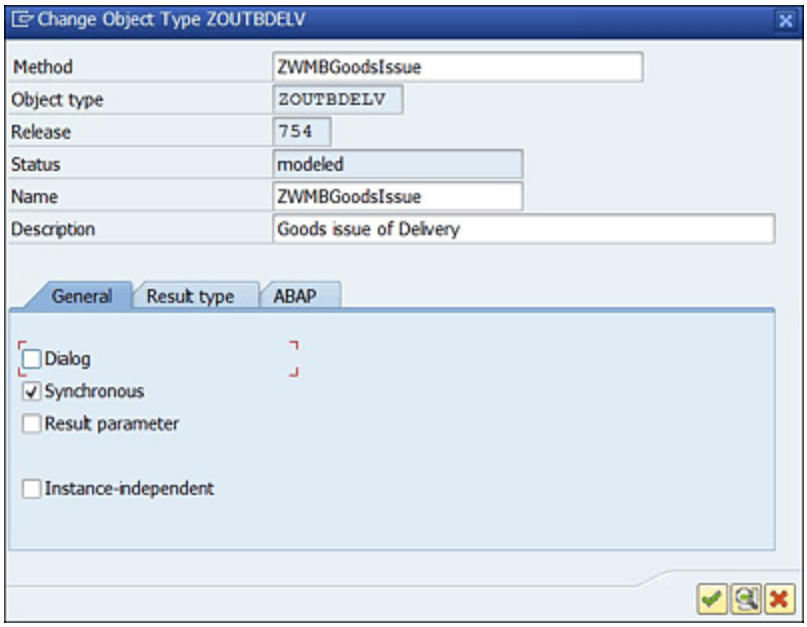


Figure 3.37 Popup Screen for Method Attributes Selection

The method attributes under the **General** tab (see Figure 3.37) have the following significance:

- **Dialog**
  If you select this checkbox, then the method will be executed in foreground mode; otherwise, the method will be executed in background. This setting is inherited from the method into the activity step task definition, which we'll explore in Section 3.3.1. Dialog methods are intended for user-based execution of a transaction. Background methods are intended for database query or update operations.
- **Synchronous**
  This setting determines whether the method execution (and the corresponding activity task execution) is set to complete when the method processing completes, or if it should wait for some external events to complete the execution status. Background methods are always

synchronous because the processing of the method and corresponding tasks completes as soon as the method logic is executed. Dialog tasks are usually asynchronous because the completion of processing depends on updating some key information in the transaction by the user, which is identified by terminating events.

- **Result parameter**
  This checkbox can be used to create a method with a result parameter. In this case, a result parameter must be maintained for the method (in the **Result type** tab of the same popup screen), which will have the output value after method execution.
- **Instance-independent**
  This checkbox marks the method as independent of the object instance; that is, it's not dependent on the key field of the object type. For instance, if you have a method to create the outbound delivery, then the delivery number (instance key) isn't relevant for the method. Another example is a method to determine the email address of the approver.

There are further settings in the **ABAP** tab of the method creation popup, which are to do with the generation of template code for method implementation. In Figure 3.38, all the highlighted options generate a template code for the method. If you prefer to write your own code, then select the **Other** radio button in this tab.

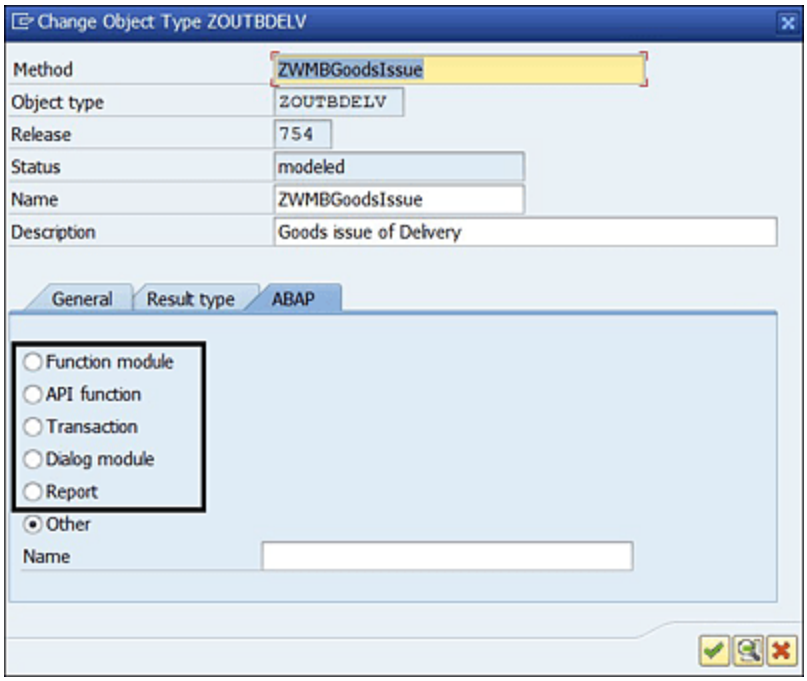


Figure 3.38 ABAP Tab Attributes for Method Creation

Once you click on execute (green check button) in the screen just shown in Figure 3.38, the method is created in **Modeled** status. You must move it to **Implemented** status via menu path **Edit** • **Change Release Status** • **Object Type Component** • **To Implemented**.

Next, you'll add the method parameters, if any. In this example, you can add the goods issue date as an import parameter and the BAPI return table as a multiline export parameter.

Click once on the method name, and then click on the **Parameters** button on the toolbar. In the next screen, click on the **Create** button. In the next popup screen, you can choose to create the parameter with an DDIC field proposal if you want to refer to the data type of the parameter from an existing SAP table/field. For this example, click on **Yes**. Enter the **Table** name as "LIKP" in the next screen and

select the field **WADAT_IST** from the list, as shown in Figure 3.39.

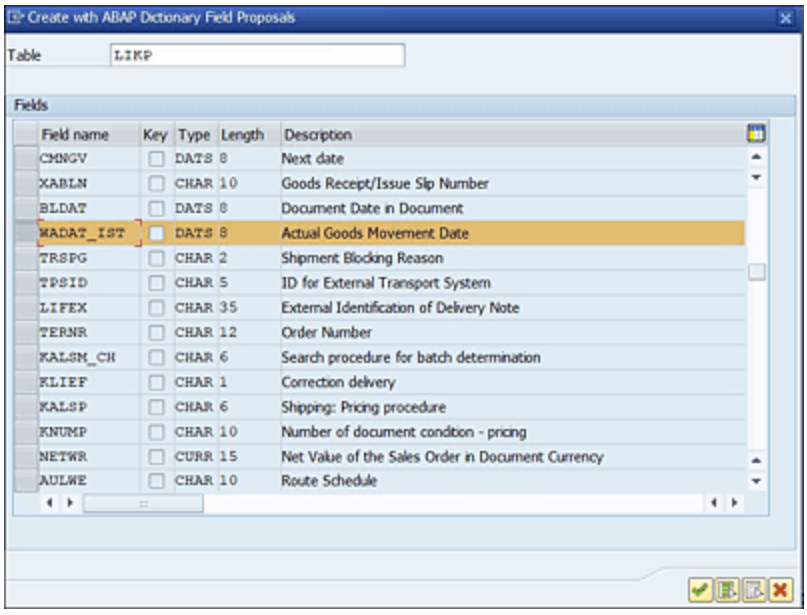


Figure 3.39 Select Table Field for Method Parameter Creation

Figure 3.40 shows a view of the method parameter creation step. In this screen, you must enter the parameter name and description, and select the attributes for the parameter type (**Import**/**Export**), scalar or tabular (**Multiline**), mandatory/optional (**Mandatory**), and whether the parameter should be auto-populated from the object type key field (**Supplied from Key**).

Once you click on **Create**, then the import parameter is created. In the same way, you can create the export parameter for the method. Figure 3.41 illustrates with an example the step to create an export parameter for the method.

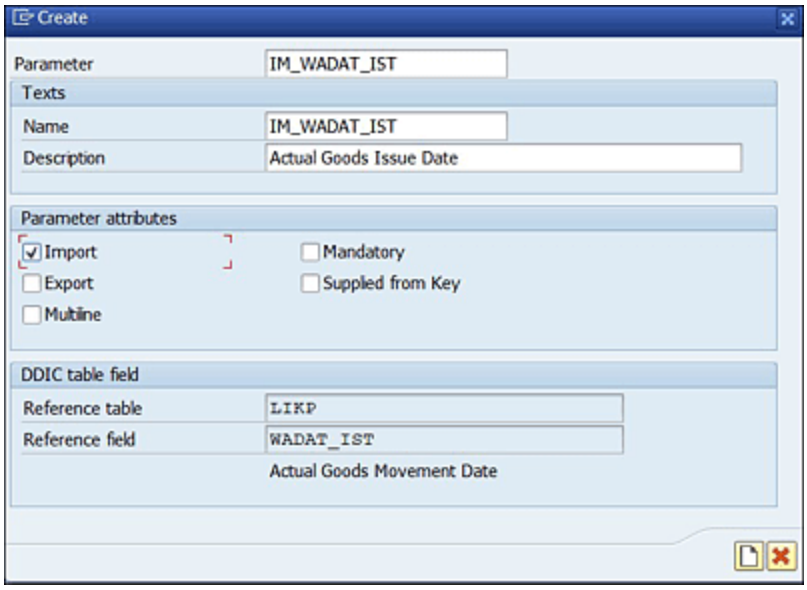


Figure 3.40 Entering the Method Parameter Name and Selecting Attributes

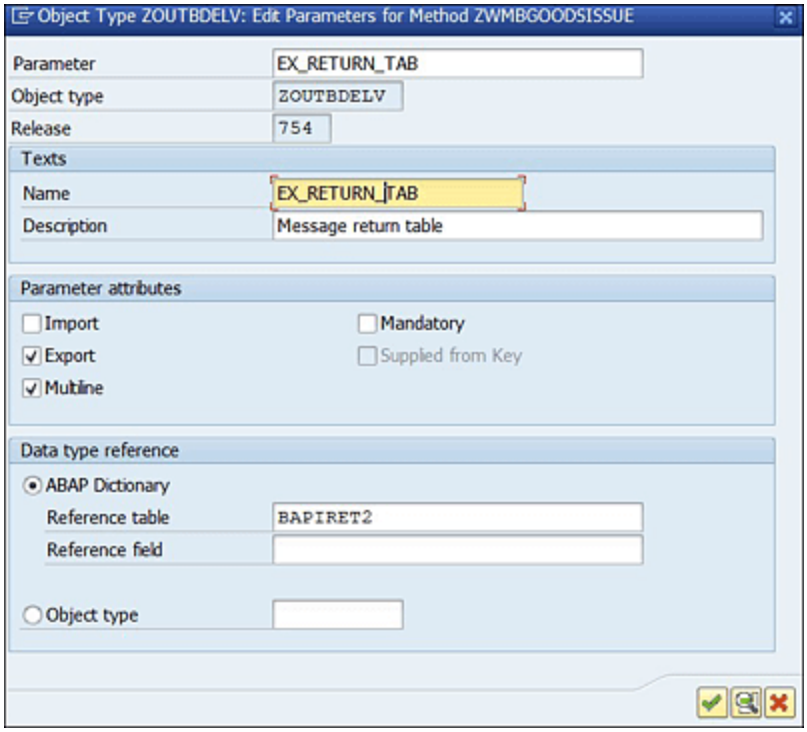


Figure 3.41 Creation of the Method Export Parameter

Figure 3.42 shows the view of method parameters after they have been created.

Similarly, you can also add *exceptions* to the method definition by clicking on the **Exception** button on the toolbar. Let's create an exception called "Goods Issue failed" for our case study. The benefit of creating an exception as opposed to a simple return message is that at runtime when the method (called from an activity task step) terminates with an exception, then the message from the exception is reflected directly on the workflow log, and the workflow is set to **Error** status. You can use subsequent **Restart after Error** functions in the workflow to reprocess the step after the issue is resolved.

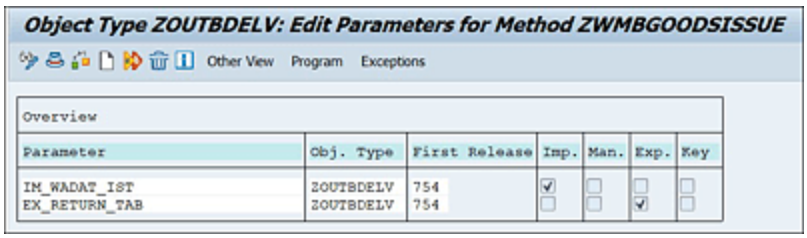


Figure 3.42 View of Method Parameters

Once you click the **Exception** button and then click on **Create**, you get the popup screen shown in Figure 3.43. Enter an **Exception** number, select the **Error type** for the exception, and then enter the **Message** class and number that will be called when the exception is triggered. Let's look at each field in detail:

- **Exception**
  For custom exceptions, you can enter any number between 9000 and 9999.
- **Error type**
  Choose from **Temporary error**, **Application error**, and **System error**. The choice of this type depends on the workflow system response in terms of handling this

exception and how the exception may be reprocessed via standard workflow jobs. Temporary exceptions are automatically reprocessed by job `SAP_WORKFLOW_SYSTEM_TEMPORARY` after this exception is raised. On the other hand, application and system errors are reprocessed by periodic job `SAP_WORKFLOW_SYSTEM`, depending on the scheduling frequency.

- **Message**
  Enter the message class and message number that will be called once the exception is raised.

Figure 3.44 shows the view of the method exception after creation.

Now go back, click once on the method name, and select the **Program** button from the toolbar. You'll be prompted to generate the method implementation in the BOR program. Select **Yes** and proceed. Figure 3.45 shows the popup message prompting you to generate the method implementation.

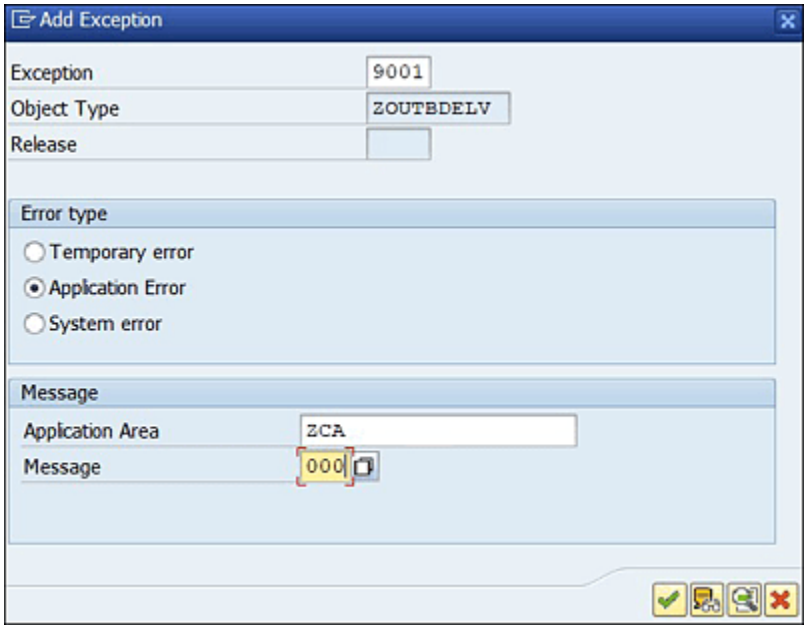


Figure 3.43 Selection of Exception Definition Attributes

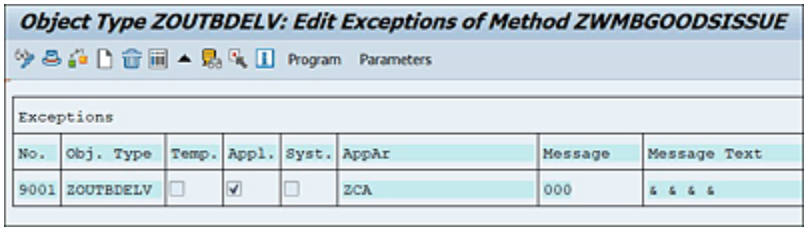


Figure 3.44 View of Method Exception Definition

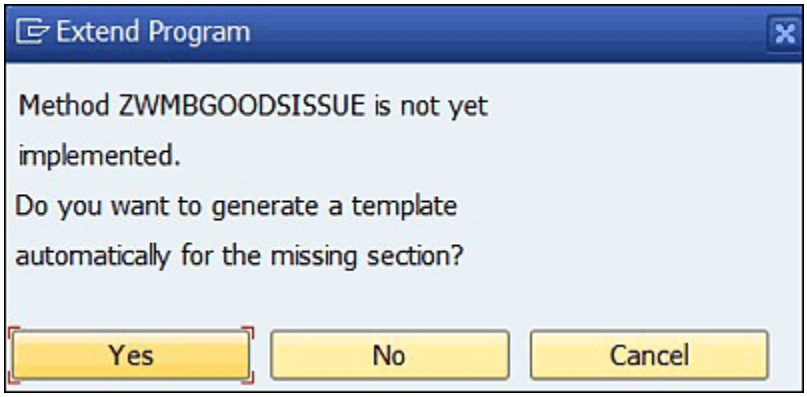


Figure 3.45 Popup Prompt to Generate Method Implementation in the BOR Program

The template code for method implementation is automatically inserted into the BOR program, as shown in Figure 3.46. You can add your own logic in the method now between the `SWC_GET_ELEMENT` macro (to read the import parameter into local variable) and the `SWC_SET_TABLE` macro (to populate the export parameter from the local internal table). Once this is completed, go back and generate the object type definition.

We're not including the source code for goods issue of delivery here as this depends on customer and system requirements. However, we'll be covering more examples for method definition under the BOR programming in Section 3.1.9.

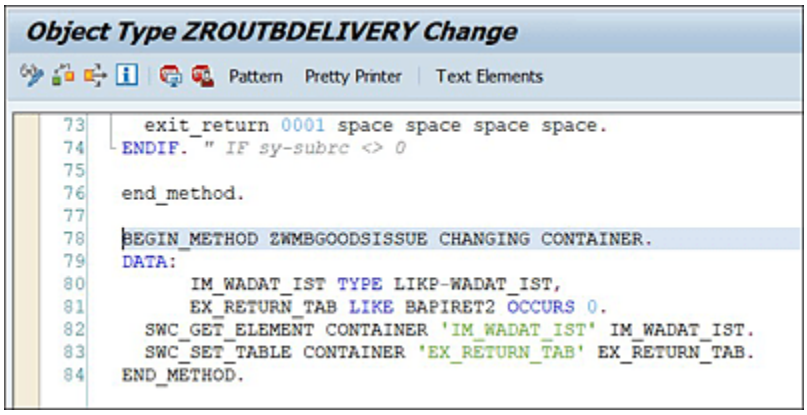


Figure 3.46 Template Code for Method Implementation Added into the BOR Program

3.1.5 Creating Business Object Repository Events

In this section, you'll add two events to the custom BOR type definition, `ZWECreated` and `ZWEGoodsIssueCompleted`. (Enter your event name per your naming standards.)

You can either right-click on the **Events** node of the object type definition inside Transaction SWO1 and select **Create**, or you click on the **Create** button from the toolbar. Figure 3.47 shows the details, such as **Name** and **Description**, that you must enter to create a new event.

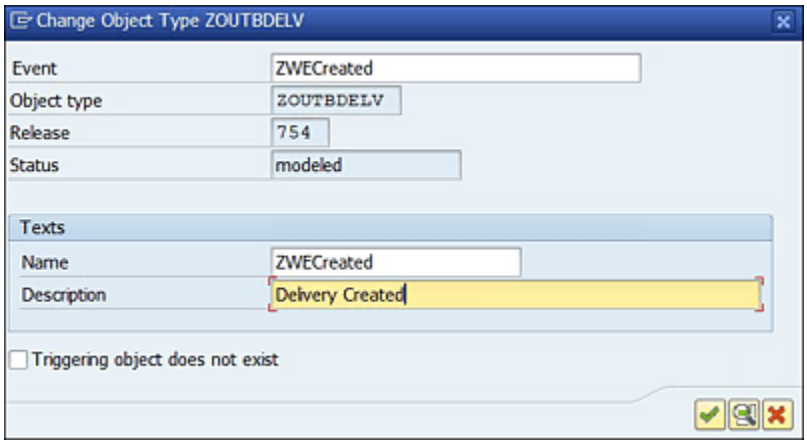


Figure 3.47 Event Definition in the Business Object Type

The checkbox for **Triggering object does not exist** makes the event instance independent, which means the event isn't dependent on the object type key field and triggering this event doesn't create an instance of the BOR object type in which the event is defined.

Once you confirm the event name in the popup screen, the event is created in **Modeled** status in the object type. You must change the status of this event to **Implemented** by following menu path **Edit** • **Change Release Status** • **Object Type Component** • **To Implemented**.

Now, follow the same steps to add another custom event, `ZWEGoodsIssueCompleted`. Once this is done, save and generate your object type.

Figure 3.48 shows the view of the business object type after creation of the events.

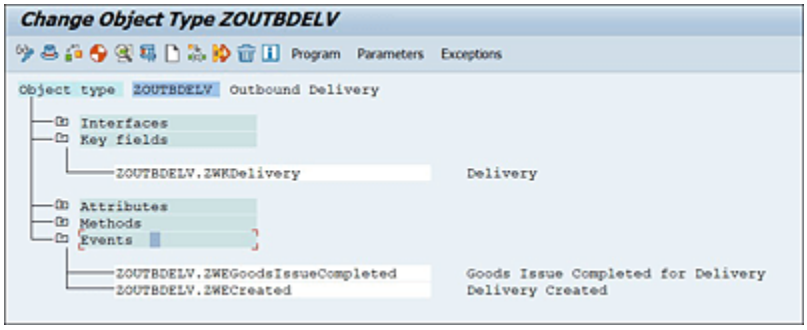


Figure 3.48 Event Definition in the BOR Object Type

You can also add event parameters (importing only) like method parameters. In this example, however, we haven't defined any event parameters.

Event definition doesn’t add any source code to the BOR program, so there is no event implementation step required like for attributes and methods. Here, we’ve only defined the event name (along with any parameters) in our custom BOR type. Event triggering and receiver linkage is handled through separate configurations discussed in detail in Chapter 5.

3.1.6 Testing a Business Object Repository Object Type

Once the definition of your custom object type is complete, you can single test the same from Transaction SWO1. Enter the object type name in Transaction SWO1, and click on the **Test** button. You can also use the **Test/Execute** F8 button inside Transaction SWO1. In this example you’ll test the custom object type `ZOUTBDELV` developed in the previous sections of this chapter. Once you click on the **Test/Execute** button, the system will prompt you to enter the key field values, in this case, the outbound delivery number (see Figure 3.49). The key field information is required to create an instance of the object type.

Enter the delivery number, and click on **Enter** to create an instance of the BOR type. All attributes will be populated per the defined mapping logic. You can also test the methods by executing them separately and entering the method import parameters (if any). Figure 3.50 shows a view of the **Test Object Type** screen after creating the object instance.

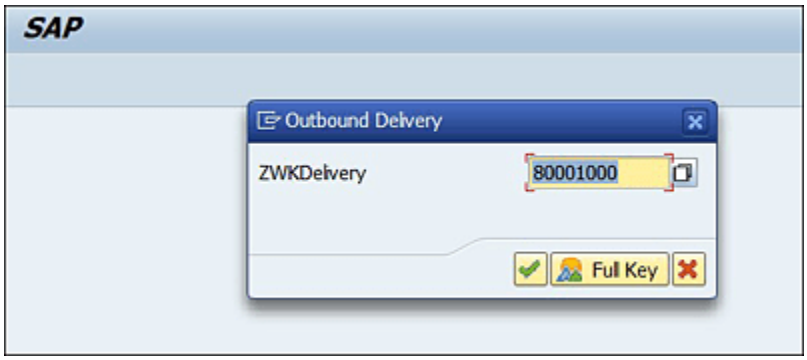


Figure 3.49 Testing a BOR Object Type in Transaction SWO1

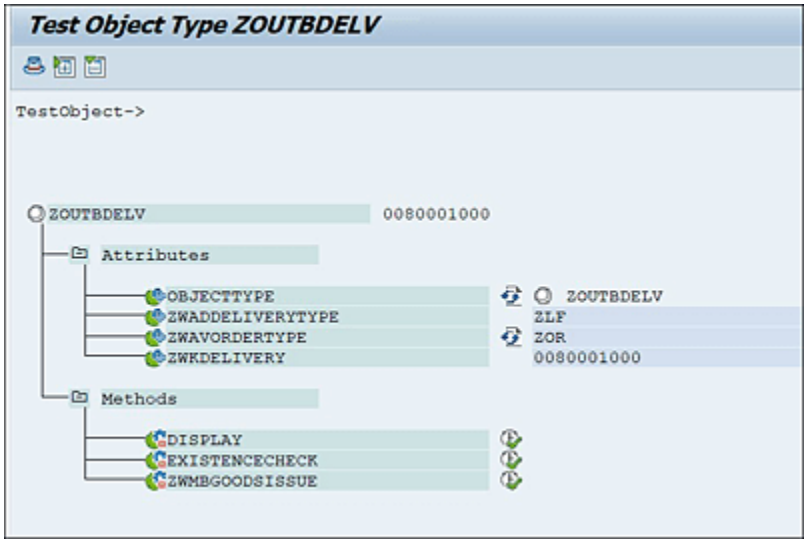


Figure 3.50 BOR Type Object Instance Displaying Attributes and Methods Available for Execution

Figure 3.51 shows an example of executing and testing a method from the BOR transaction by entering the required import parameters and then clicking on the **Execute** F8 button or the **Debugging** button (if you want to start the debugger).

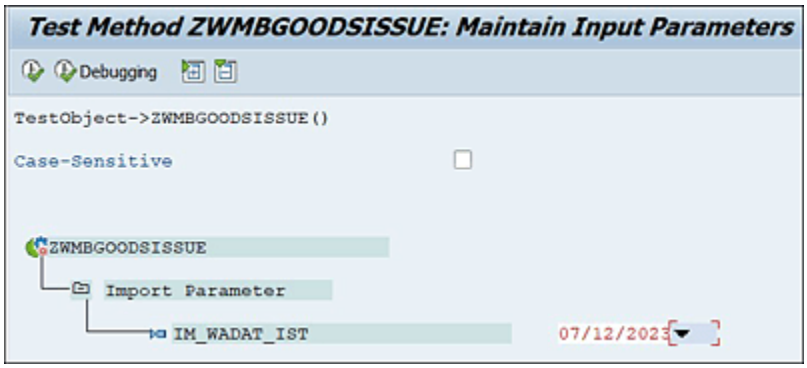


Figure 3.51 Testing a BOR Method

You can debug any method or virtual attribute code in the BOR type by setting a breakpoint on the required source code line in the BOR program.

3.1.7 Creating a Subtype of a Standard Business Object Repository Object Type

In Section 3.1.2, we looked at the steps to create a custom BOR type from Transaction SWO1. However, in most cases, you'll find that SAP has already provided a standard object type definition for your standard SAP applications, such as purchase orders, sales orders, deliveries, billing documents, and so on. In these cases, when you need to add some attribute, method, or event that isn't provided in the standard BOR type (also called a supertype), you may need to extend these object type definitions by using the subtype approach.

A subtype definition is technically a child of the supertype and is hierarchically one level below the supertype. For instance, `BUS2032` (sales order) is the subtype of `VBAK` (sales document). Semantically, `BUS2032` is the object definition of one specific category of a sales document (`VBAK-VBTYP = C`),

whereas VBAK is the BOR type for all sales documents. Now consider an example where you need to have the customer PO number (VBKD-BSTKD) as an attribute in your object type. Standard BOR type BUS2032 doesn't have this attribute, so you need to extend the standard BOR type by creating a subtype and then add your custom attribute (and methods/events if required) in the same.

After creating a subtype of a standard BOR type, you can add new attributes, methods, and events in the BOR type in the same way as described in previous sections for a custom BOR type. However, you should not edit the standard attributes, methods, or events by way of redefinition as a best practice (except for interface methods, which may be redefined in some cases). If you want to make changes to the logic of some standard attribute or method, it's better to create your custom component by copying and then making the necessary changes. Changing the key field definition of a subtype is also not allowed. Table 3.1 summarizes the editing options for each BOR component in a subtype.

BOR Component	Adding New in Subtype	Editing in Subtype
Key fields	Not allowed	Not allowed.

BOR Component	Adding New in Subtype	Editing in Subtype
Attributes	Allowed	Inherited virtual attributes may be redefined, but not recommended. Database attributes can’t be edited or redefined.
Methods	Allowed	Inherited methods may be redefined, but not recommended.
Events	Allowed	Inherited events may be redefined by adding new parameters.

Table 3.1 Summary of Component Editing Options in a Subtype Definition

To create a BOR subtype in SAP, follow these steps:

1. Enter the BOR supertype name in Transaction SWO1 in the **Object/Interface Type** field, and click on the **Create Subtype** button on the toolbar. Enter the subtype name, description, and program name details, as shown in Figure 3.52.

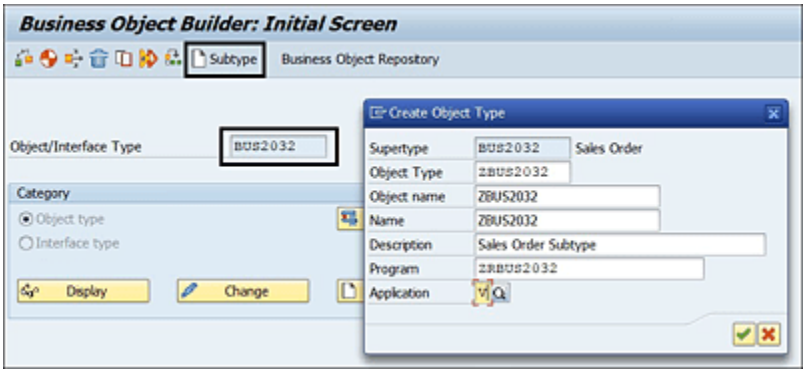


Figure 3.52 Creating a BOR Subtype Definition from Transaction SWO1

2. After creation of the subtype, the object type status is set to **Modeled**. You must change it to **Implemented** by following menu path **Edit** • **Change Release Status** • **Object Type** • **To Implemented**.
3. Figure 3.53 shows the **Change Object Type** screen where all the inherited components from the supertype appear in red in the subtype, and new components or redefined components in the subtype appear in white.

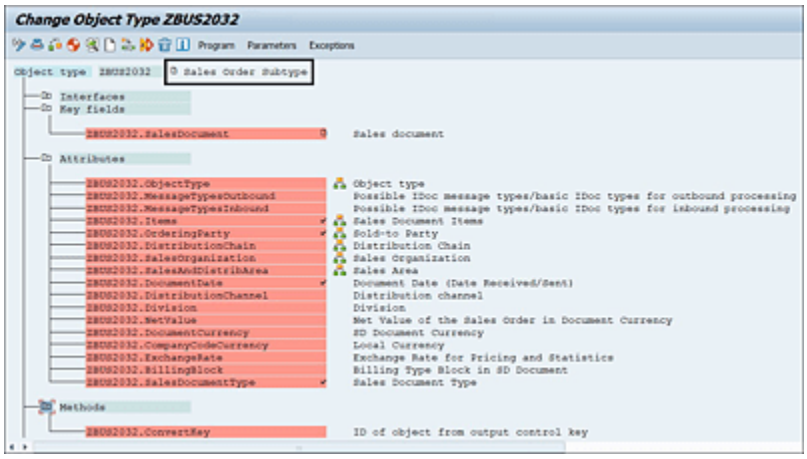


Figure 3.53 Subtype Definition Showing Inherited Components and Header Release Status

3.1.8 Delegation

Delegation is a very powerful tool in the business object type concept, which lets you add custom attributes, methods, and events to a standard BOR type via the subtype approach, but you can still access these custom components via the supertype. This allows you to go with a modification-free approach for your custom business requirements.

For example, consider that you've created a subtype of `BUS2032` called `ZBUS2032` and added new custom attributes, methods, and events in the subtype. Now, if you delegate `BUS2032` to `ZBUS2032` object type, then you can access the custom attributes, methods, and events directly via `BUS2032` in your workflow and task definition. You don't need to use `ZBUS2032` anywhere in the workflow. Standard SAP searches for the custom components in the supertype BOR program first, and if it can't find the components defined there, then it searches for them in the delegated subtype BOR program.

Note that you can create *N* number of subtypes for a BOR type definition, but you can delegate the supertype to only one subtype. This means that there can be only one delegated subtype of a business object type in SAP.

You can access the delegation setting from Transaction SWO1 via menu path **Settings** • **Delegate** or directly via Transaction SWO6. This is a cross-client configuration activity that is saved in a workbench transport. Figure 3.54 shows the delegation screen with the list of supertypes that have been delegated to a subtype.

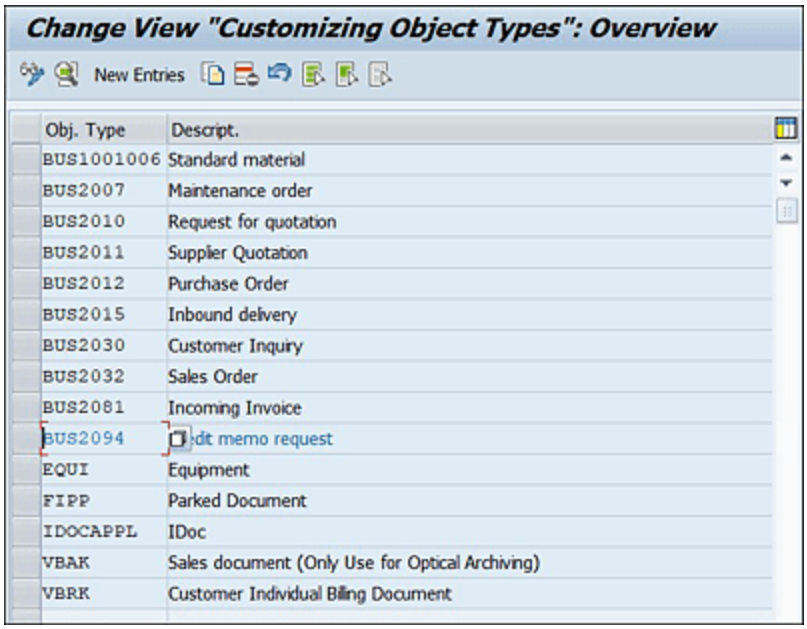


Figure 3.54 Delegation Configuration Showing the Supertypes Maintained

Now if you select any object type and click on the **Detail** button, the delegated subtype is shown. After creating your own subtype, you can maintain a new entry in this configuration.

3.1.9 Business Object Repository Programming

BOR programming is commonly used in the coding of the virtual attributes and methods of an object type definition. This programming technique makes heavy use of macros, especially for reading and updating values in the container. A container in the context of a workflow refers to a data object with a typical structure of type `SWCONT`. Containers are used in workflow definitions, task definitions, method parameters, rules, and so on. Containers may have multiple elements defined in them. These container elements are

synonymous to variables or internal tables in a program. So, if you need to read the value of a container element or update a container element with some value, these macros come in very handy. You can also use these macros in standalone report programs by including program <CNTN01>. This include contains the definition of the most-used macros in BOR programming.

Next we'll look at the various applications and use cases of macros in BOR programming. We'll start by looking at the some common macros used in defining virtual attributes and then some other macros used in defining methods.

Virtual Attribute Definitions

When you add a virtual attribute in a BOR type definition, internally it adds the attribute to a complex data structure definition named `OBJECT`. Key fields are also defined under the same object as a nested structure. Figure 3.55 shows a simple view of a business object with a virtual attribute defined.

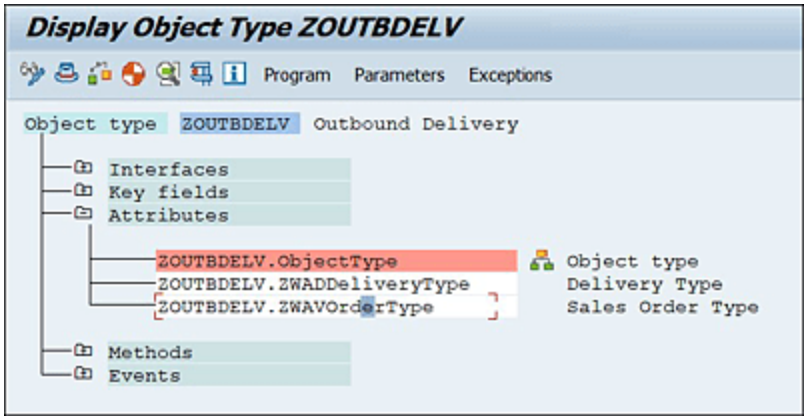


Figure 3.55 Virtual Attribute ZWAVOrderType Added in the Object Type Definition

In Listing 3.3, you can see the generated code behind the attribute definitions.

> **Note**
>
> Database attributes don’t add any code under the object data structure definition.

```
*****           Implementation of object type ZOUTBDELV            *****
INCLUDE <object>. " INCLUDE for Object Type Definition
begin_data object. " Do not change.. DATA is generated
* only private members may be inserted into structure private
DATA:
                          " begin of private,
                          "   to declare private attributes remove comments and
                          "   insert private attributes here ...
                          " end of private,
  BEGIN OF key,
    zwkdelivery LIKE likp-vbeln, " Delivery
  END OF key,
  zwavordertype TYPE vbak-auart, " Type
  _likp         LIKE likp.       " SD Document: Delivery Header Data
end_data object. " Do not change.. DATA is generated
```

Listing 3.3 Generated Source Code behind Key Field and Attribute Definition in an Object Type

The key field of this BOR type may be addressed as `OBJECT-KEY-ZWKDELIVERY`, and the virtual attribute may be addressed as `OBJECT-ZWAVORDERTYPE`, anywhere inside the BOR program. The key field(s) and attributes act as global data declarations inside the BOR program. Each object program contains the internal reference `SELF`, which refers to the object instance with which the program was called.

Commonly Used Macros in Business Object Repository Virtual Attributes

In Listing 3.4, macro SWC_GET_PROPERTY reads the attribute of an object type. When this macro is used with the addition SELF, then it means you're trying to read the attribute from the same object type. The literal Price denotes the name of the attribute you're trying to read, and the variable Price is used to return the value of this attribute in the current attribute definition logic. If the attribute to be read is a table type, then use macro SWC_GET_TABLE_PROPERTY instead.

```
GET_PROPERTY Total CHANGING CONTAINER.
 DATA : Price  TYPE  Zprice,
        Qty     TYPE Zqty.
 IF object-total is initial.
    SWC_GET_PROPERTY self 'Price' Price.
    SWC_GET_PROPERTY self 'Quantity' Qty.
    Object-Total = Price * Qty.
 ENDIF.
 SWC_SET_ELEMENT CONTAINER 'Total' object-total.
END_PROPERTY.
```

Listing 3.4 Population of a Simple Virtual Attribute Using Macros

Macro SWC_SET_ELEMENT updates element Total in the object container with the value in object-total. If the attribute is multiline (table type), then instead of using the SWC_SET_ELEMENT container, you would use the SWC_SET_TABLE container.

Refer to the example in Listing 3.5 for another piece of source code for a BOR method illustrating the usage of macros for reading/writing container elements defined as method import and export parameters. Listing 3.5 reads the import parameters of the method with macros SWC_GET_ELEMENT or SWC_GET_TABLE into local variables, performs a database query operation by calling a BAPI, and then populates the export parameters from local variables using macros SWC_SET_ELEMENT and SWC_SET_TABLE.

```
BEGIN_METHOD GETLIST CHANGING CONTAINER.
DATA:
      CUSTOMERNUMBER TYPE BAPI1007-CUSTOMER,
      SALESORGANIZATION TYPE BAPIORDERS-SALES_ORG,
      MATERIAL TYPE BAPIORDERS-MATERIAL,
      MATERIAL_LONG TYPE BAPIORDERS-MATERIAL_LONG,
      DOCUMENTDATE TYPE BAPIORDERS-DOC_DATE,
      DOCUMENTDATETO TYPE BAPIORDERS-DOC_DATE,
      PURCHASEORDER TYPE BAPIORDERS-PURCH_NO,
      PURCHASEORDERNUMBER TYPE BAPIORDERS-PURCH_NO_C,
      TRANSACTIONGROUP TYPE BAPIFLAG-BAPIFLAG,
      RETURN LIKE BAPIRETURN,
      MATERIALEVG LIKE BAPIMGVMATNR,
      SALESORDERS LIKE BAPIORDERS OCCURS 0.
  SWC_GET_ELEMENT CONTAINER 'CustomerNumber' CUSTOMERNUMBER.
  SWC_GET_ELEMENT CONTAINER 'SalesOrganization' SALESORGANIZATION.
  SWC_GET_ELEMENT CONTAINER 'Material' MATERIAL.
  SWC_GET_ELEMENT CONTAINER 'MaterialLong' MATERIAL_LONG.
  SWC_GET_ELEMENT CONTAINER 'DocumentDate' DOCUMENTDATE.
  SWC_GET_ELEMENT CONTAINER 'DocumentDateTo' DOCUMENTDATETO.
  SWC_GET_ELEMENT CONTAINER 'PurchaseOrder' PURCHASEORDER.
  SWC_GET_ELEMENT CONTAINER 'PurchaseOrderNumber' PURCHASEORDERNUMBER.
  SWC_GET_ELEMENT CONTAINER 'TransactionGroup' TRANSACTIONGROUP.
  IF SY-SUBRC <> 0.
    MOVE 0 TO TRANSACTIONGROUP.
  ENDIF.
  SWC_GET_ELEMENT CONTAINER 'materialEVG' MATERIALEVG.
  SWC_GET_TABLE CONTAINER 'SalesOrders' SALESORDERS.
  CALL FUNCTION 'BAPI_SALESORDER_GETLIST'
    EXPORTING
      CUSTOMER_NUMBER = CUSTOMERNUMBER
      SALES_ORGANIZATION = SALESORGANIZATION
      MATERIAL = MATERIAL
      MATERIAL_LONG = MATERIAL_LONG
      DOCUMENT_DATE = DOCUMENTDATE
      DOCUMENT_DATE_TO = DOCUMENTDATETO
      PURCHASE_ORDER = PURCHASEORDER
      TRANSACTION_GROUP = TRANSACTIONGROUP
      PURCHASE_ORDER_NUMBER = PURCHASEORDERNUMBER
      MATERIAL_EVG = MATERIALEVG
    IMPORTING
      RETURN = RETURN
    TABLES
      SALES_ORDERS = SALESORDERS
    EXCEPTIONS
      OTHERS = 01.
  CASE SY-SUBRC.
    WHEN 0.            " OK
    WHEN OTHERS.       " to be implemented
  ENDCASE.
  SWC_SET_ELEMENT CONTAINER 'Return' RETURN.
  SWC_SET_TABLE CONTAINER 'SalesOrders' SALESORDERS.
END_METHOD.
```

Listing 3.5 Usage of Macros for Reading and Writing Container Elements Defined as Method Parameters

Macros for Creating Object References and Containers

An object reference can be created in a method or a virtual attribute using macro `SWC_CREATE_OBJECT`. As shown in Listing 3.6, in this macro, the first parameter is the object type reference that will hold the object instance followed by the name of the object type and finally the key field value of the object type.

```
get_property salesanddistribarea changing container.
DATA : BEGIN OF salesanddistribarea,
         vkorg LIKE vbak-vkorg,
         vtweg LIKE vbak-vtweg,
         spart LIKE vbak-spart.
DATA : END OF salesanddistribarea.
DATA: salesorganization  TYPE swc_object.

swc_get_property self 'SalesOrganization' salesorganization.
swc_get_property salesorganization
                 'SalesOrganization'
                 salesanddistribarea-vkorg.
swc_get_property self 'DistributionChannel' salesanddistribarea-vtweg.
swc_get_property self 'Division' salesanddistribarea-spart.

swc_create_object object-salesanddistribarea
                  'BUS0006003'
                  salesanddistribarea.

swc_set_element container 'SalesAndDistribArea'
     object-salesanddistribarea.
end_property.
```

Listing 3.6 Example to Illustrate the Usage of Macro SWC_CREATE_OBJECT in BOR Programming

Local containers can be created and initialized in BOR programming using macros `SWC_CONTAINER` and `SWC_CONTAINER_CREATE`, as follows:

```
SWC_CONTAINER <Cont> "Declares a container in BOR program
SWC_CONTAINER_CREATE <Cont> "Initializes the container in BOR program
SWC_RELEASE_CONTAINER <Cont> "Releases the container, after which it cannot be
edited
```

Macro for a Method Call in Business Object Repository Programming

Macro `SWC_CALL_METHOD` can be used to call a method from the same object type or a different object type. The two variants of this macro are as follows:

```
SWC_CALL_METHOD <object> <method> <container>
SWC_CALL_METHOD SELF 'AllAgentsOfTaskGet' CONTAINER.
```

Macros for Method Exceptions

Exceptions in methods can be raised using macro `EXIT_RETURN <exception #> <var1> <var2> <var3> <var4>`, where `Exception #` is the exception ID defined in the method, and `<var1>`, `<var2>`, `<var3>`, and `<var4>` are the message variables for the message number tagged to the exception ID. Refer to Section 3.1.4 for details on exceptions.

Some other commonly used macros for method exceptions are as follows (these are just variants of main macro `EXIT_RETURN`):

- **`EXIT_OBJECT_NOT_FOUND`**
  Tells the workflow runtime system that the object doesn't exist.
- **`EXIT_CANCELLED`**
  Tells the workflow runtime system that an action was canceled by the user in the dialog.

- **`EXIT_NOT_IMPLEMENTED`**
  Tells the workflow runtime system that a method isn’t implemented.
- **`EXIT_PARAMETER_NOT_FOUND`**
  Tells the workflow runtime system that a mandatory parameter is missing.

3.2 ABAP Class Approach

The ABAP classes for workflow functionality was introduced in release 6.20, but there were various restrictions till release 6.40. Any normal ABAP class implementing interface `IF_WORKFLOW` can be used for workflow. Interface `IF_WORKFLOW` interface inherits two interfaces: `BI_OBJECT` and `BI_PERSISTENT`. If your workflow requirement is for a custom application or for an SAP standard application that doesn't provide a BOR type definition, then it's recommended to go with the ABAP class approach. It avoids the complexities introduced by the macros in BOR programming and provides a much cleaner and efficient coding style in concurrence with the latest ABAP coding standards and best practices.

Table 3.2 highlights some of the notable distinctions between the BOR approach and the ABAP class approach with respect to the workflow. Properties BOR ABAP Classes

Properties	BOR	ABAP Classes
Delegation	Via Customizing table	No delegation
Transaction	Transaction SWO1	Transaction SE24
Key length	Max of 70 characters	Max of 32 characters
Attribute types	Key, database, virtual, status	Key, non-key, public, private

Properties	BOR	ABAP Classes
Method types	• Dialog/background • Synchronous/asynchronous	N/A
Method parameters	Importing, exporting, result	Importing, exporting, changing, returning
Method exceptions	Temporary, application, system	All exception classes starting with `CX_BO`, for example, `CX_BO_TEMPORARY` and `CX_BO_ERROR`
Tables	Multiline parameters	Table types

Table 3.2 Primary Differences between the BOR Approach and the ABAP Class Approach

In the following sections, we'll go through a step-by-step approach to create a workflow class; add attributes, methods, and events to the class; and then test the class standalone.

3.2.1 Creating a Workflow Class

Let’s go through the various steps of creating a workflow class with the help of an example business scenario in which you’re developing a workflow for release of a billing document created in SAP. Billing documents will be automatically blocked at the time of creation based on some restrictions and then trigger an approval workflow. Once the workflow is approved, the billing document will be posted and released to accounting. Here, the key field involved is the billing document number (`VBRK-VBELN`). Now let’s look at the development of the workflow class and business logic implementation for this requirement.

Create a new ABAP class from Transaction SE24, as shown in Figure 3.56. Enter the class name and description. In **Inst.Generation**, choose **2 Public**; select the class type as **Usual ABAP Class**; and select the **Final** checkbox to mark it as final.

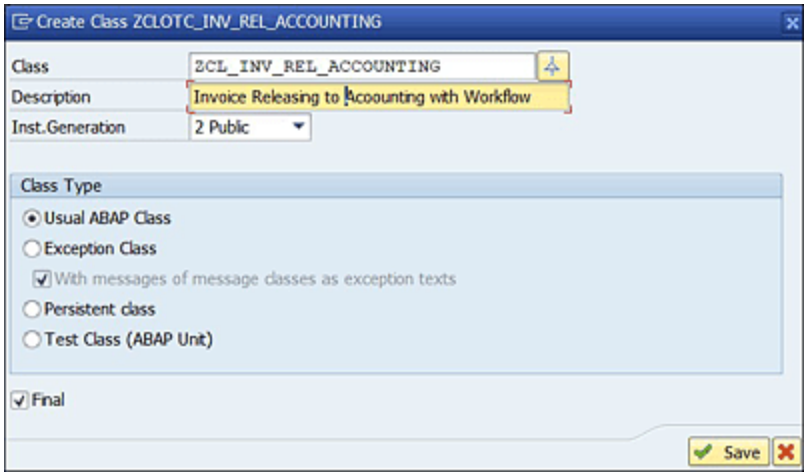


Figure 3.56 Creation of a New Normal ABAP Class from Transaction SE24

After creation of the class, go to the **Interfaces** tab, enter “IF_WORKFLOW”, and press Enter. Two more inherited interfaces, `BI_OBJECT` and `BI_PERSISTENT`, automatically get added to the class definition. The addition of interface

IF_WORKFLOW transforms your class from a regular ABAP class to a workflow business class. Figure 3.57 shows the workflow class after the addition of interface IF_WORKFLOW.

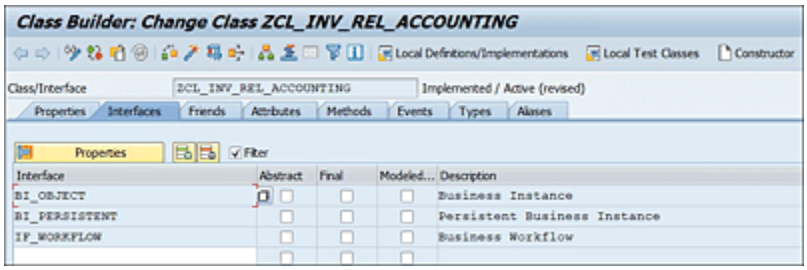


Figure 3.57 Addition of Interface IF_WORKFLOW to the Workflow Class

Now save and activate the class.

3.2.2 Defining Key Attributes and Non-Key Attributes

Attributes in an ABAP class provide access to the data from the object instance. Key attributes are specifically used to create the object instance; that is, they are the import parameters of the CONSTRUCTOR method and together form the instance ID (INSTID) in the LPOR (local persistent object reference) attribute of the ABAP class. The key attribute isn't a universal concept in regular ABAP classes. They appear in workflow business classes only through the IF_WORKFLOW interface. The LPOR attribute is a required and critical attribute in any workflow business class. This attribute is used by the workflow runtime to persist an object instance and populate the instance buffer as and when requested by the workflow runtime. All other object attributes required to be used in your workflow and task subject and descriptions should be created as Instance and

`Public`. Attributes that aren't dependent on the object instance can be created as `Static`.

Let's go ahead and create the required attributes for the workflow scenario related to billing document release in SAP S/4HANA. Because the key field for the object instance is the billing document number, you'll first create an attribute for the same.

Create attribute **MV_VBELN** in your workflow business class, mark it as **Instance Attribute** under **Level** and **Public** under **Visibility**, and select the key (checkbox in the **K** column) in the **Attributes** section, as shown in Figure 3.58.

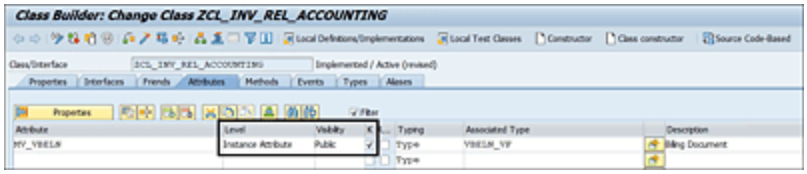


Figure 3.58 View of the Key Attribute Definition in the Workflow Business Class

Next, create the **MS_LPOR** attribute, as shown in Figure 3.59. Mark it as **Instance** and **Private** (as this attribute will only be used internally within the class). Enter "SIBFLPOR" as the **Associated Type** of this attribute.

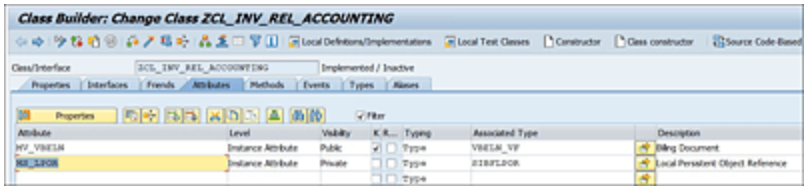


Figure 3.59 Creation of the LPOR Attribute in the Workflow Business Class

Now you need to define the instance buffer structure (under `types` declarations of the class) and then create a table type

attribute for the same. This attribute should be a static one as it will hold the collection of object instances for this workflow class at runtime in the object buffer.

> **Note**
>
> The instance buffer approach is optional. It’s a mechanism that improves the performance efficiency of a workflow business class by avoiding unnecessary instantiation of the object similar to the concept of a singleton class.

Add the code in Listing 3.7 under the local private implementation section of the class definition.

```
private section.

  types:
    BEGIN OF ty_s_inst_buffer,
      vbeln    TYPE vbeln_vf,      " Billing Document
      instance TYPE REF TO object, " class
    END OF ty_s_inst_buffer .
  types:
    ty_t_inst_buffer  TYPE STANDARD TABLE OF ty_s_inst_buffer WITH KEY vbeln .

  class-data MT_INSTANCE_BUFFER type TY_T_INST_BUFFER .
```

Listing 3.7 Local Types Declaration for the Instance Buffer in the Workflow Class

This will create the **MT_INSTANCE_BUFFER** attribute in the class, as shown in Figure 3.60.

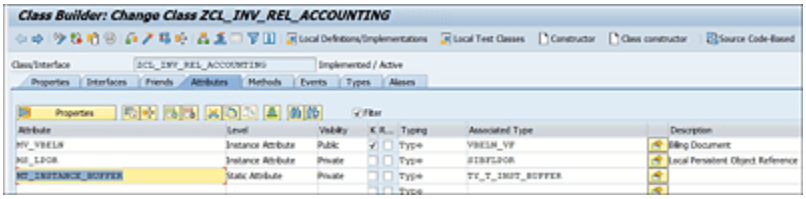


Figure 3.60 View of the Instance Buffer Attribute in the Workflow Business Class

Now, add some more attributes for the sales document type, sales organization, distribution channel, net value, and billing type in your workflow class. These optional attributes will be created as instance and public, and they are intended to be used in your workflow and task definition per your business requirement. Once the attributes are created, your workflow class will look something like Figure 3.61.

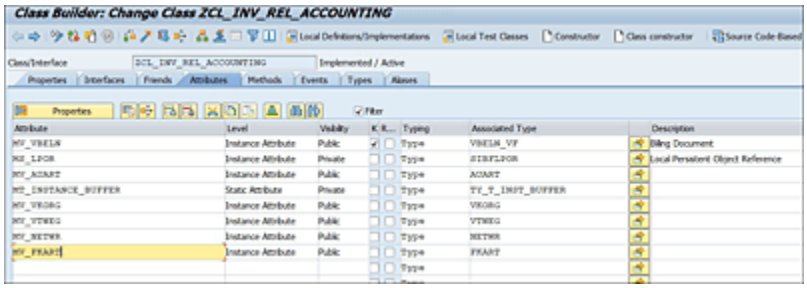


Figure 3.61 Overall View of Attributes in the Workflow Business Class

Save and activate your workflow class. In the next section, you'll be creating and implementing the methods in your workflow business class.

3.2.3 Creating Methods and Defining Attributes, Parameters, and Exceptions

We'll first start by creating the constructor method in our workflow business class. This is the most important method in your workflow class, which gets called every time a new instance of the class is requested by the workflow runtime. This class will receive the key field of the object instance (in this case, the billing document number) and populate the other instance attributes of your workflow class.

Note

If you've defined any static attributes in your workflow class, then you can use the class constructor method to populate them.

Create your `CONSTRUCTOR` method using the **Create Constructor** button in your class definition, and then add the import parameters, as shown in Figure 3.62, by entering the parameter name (**Parameter** field) and data type (**Associated Type** field).

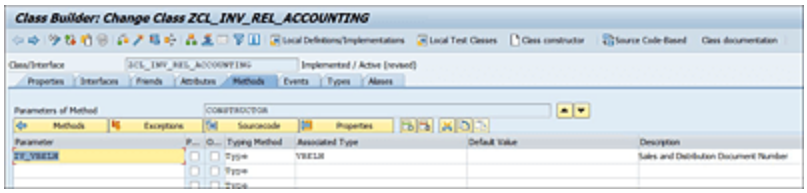


Figure 3.62 Creating the Import Parameter(s) in the Constructor Method of the Workflow Business Class

Now refer to the sample code in Listing 3.8 for the constructor method. You can add your own logic to populate the instance attributes per your requirement. Note the population of the `MS_LPOR` attribute. Here the `INSTID` is the key (concatenation in case of multiple key fields) of the object instance. This field is 32 characters long. So, if your key field(s) of a business transaction have a combined length greater than 32, consider creating an alternate key such as a GUID based on your actual key and using this alternate key for populating the `INSTID` of your workflow class.

```
CONSTANTS : lc_vbtypl TYPE vbtypl_v VALUE 'C', " Document category Sales Order
            lc_vbtyp2 TYPE vbtypl_v VALUE 'G', " Document category Contract
            lc_vbtyp3 TYPE vbtypl_v VALUE 'L', " Document category Debit memo
request
            lc_vbtypn TYPE vbtyp_n  VALUE 'M'. " Document category of Subsequent
Document

**Populate LPOR attribute
    ms_lpor-catid = 'CL'.
```

```
    ms_lpor-typeid = 'ZCLOTC_INV_RELEASE_ACCOUNTING'.
    ms_lpor-instid = mv_vbeln = iv_vbeln.

*    Populating Class attributes
    SELECT vbrk~vbeln, " Sales Document
           vbrk~fkart, " Billing Type
           vbrk~vkorg, " Sales Organization
           vbrk~vtweg, " Distribution Channel
           vbrk~netwr, " Net Value of the Sales Order in Document Currency
           vbrp~aubel, " Sales document
           vbak~auart  " Sales Document Type
           INTO @DATA(ls_invoice_details)
           FROM vbrk   " Billing Document: Header Data
           INNER JOIN vbrp
           ON vbrk~vbeln = vbrp~vbeln
           INNER JOIN vbak " Sales Document: Header Data
           ON vbak~vbeln = vbrp~aubel
           UP TO 1 ROWS
           WHERE vbrk~vbeln = @iv_vbeln
           ORDER BY vbrk~vbeln.
    ENDSELECT.
*Assigning the class attribute
    IF sy-subrc = 0.
      mv_auart = ls_invoice_details-auart. "Sales Order Type
      mv_vkorg = ls_invoice_details-vkorg. "Sales Organization
      mv_vtweg = ls_invoice_details-vtweg. "Distribution channel
      mv_netwr = ls_invoice_details-netwr. "Net value
      mv_fkart = ls_invoice_details-fkart. "Billing Type
    ENDIF. " IF sy-subrc = 0
```

Listing 3.8 Sample Source Code for the Constructor Method of the Workflow Business Class

Next, you'll implement interface method `BI_PERSISTENT~LPOR`. This method outputs the `LPOR` value to the workflow runtime, based on the `MS_LPOR` attribute that you've populated in the constructor method while creating the object instance. As mentioned before, the `LPOR` attribute is used by the workflow runtime to persist the object instance and fetch it into the object buffer as and when requested by the runtime.

Copy and paste the source code in Listing 3.9 in this method.

```
    result = MS_LPOR.
```

Listing 3.9 Source Code for Interface Method LPOR in the Workflow Business Class

Next, you'll implement interface method `BI_PERSISTENT~FIND_BY_LPOR`. This method receives the predefined import parameter `LPOR` from the workflow runtime and returns the persistent object instance based on the instance key (`INSTID`). Refer to the source code in Listing 3.10 for this method. This source code may vary between different classes based on the key field definition of the object instance. Note the use of the instance buffer concept (attribute `MT_INSTANCE_BUFFER`) to improve the efficiency.

```
    DATA : lv_vbeln    TYPE vbeln_vf,                    " Billing Document
           lo_instance TYPE REF TO zclotc_inv_release_accounting. " Class for
Invoice Releasing Acoounting

    lv_vbeln = lpor-instid(10). "Assigning billing doc no
    ASSIGN mt_instance_buffer[ vbeln = lv_vbeln ] TO
FIELD-SYMBOL(<ls_instance>).
*   If billing document object instance does not exist in buffer
    IF sy-subrc <> 0.
*     Create object instance
      TRY.
          lo_instance = NEW #( iv_vbeln = lv_vbeln ).
        CATCH cx_bo_error.
          RETURN.
      ENDTRY.
*     Fill the buffer table
      mt_instance_buffer = VALUE #( ( vbeln = lv_vbeln
                                instance = lo_instance ) ).
      result ?= lo_instance. "Filling the result object
    ELSE. " ELSE -> IF sy-subrc <> 0
*    IF billing document object instance exists in buffer
      result ?= <ls_instance>-instance. "Filling the result object
    ENDIF. " IF sy-subrc <> 0
```

Listing 3.10 Source Code for Interface Method FIND_BY_LPOR in the Workflow Business Class

The other interface methods in your workflow class are optional. A brief overview of the purpose of the other

interface methods is as follows:

- **BI_PERSISTENT~REFRESH**
  This method can be used to clear the object instance buffer and force the workflow runtime to load a new object instance on the next request. This method is equivalent to an explicit object destructor method.
- **BI_OBJECT~DEFAULT_ATTRIBUTE_VALUE**
  This method can be used to populate a default attribute such as a description of the object instance in the result parameter of the method. In this example, it could be a default description such as the customer PO number from the linked sales order. The value of this attribute is displayed beside the instance key (billing document number) when the object link is displayed in the workflow inbox.
- **BI_OBJECT~EXECUTE_DEFAULT_METHOD**
  This method can be used to call a transaction or screen when the user clicks on the object link from the inbox. In this example, you can use the method to call Transaction VF03 to display the billing document. This serves as a useful navigation link into the object for displaying further details before approving or rejecting a document via workflow.
- **BI_OBJECT~RELEASE**
  This method is used by the workflow runtime to implicitly call the garbage collector and delete the object instance from memory. It serves as an implicit destructor of the object instance. Normally, no code is required in this method.

Now that you've implemented the constructor and other interface methods of your workflow class, you can add new methods that will be called via tasks in activity steps. These methods can be created as static or instance, depending on whether they depend on the object instance (key attributes) or not, but they should always be public as they will be called from the workflow runtime. An important distinction from BOR methods is that ABAP OOP methods can't be classified as dialog or background and synchronous or asynchronous. These settings are always entered directly in the task definition, which is discussed in more detail in Section 3.3.1.

3.2.4 Adding New Methods for Dialog and Background Tasks

Now, you can create a method called `APPROVE_INV_ACCOUNTING`, which will be called to release the billing document to accounting once it's approved via workflow. This method must be declared as instance and public in its properties. We won't create any import parameters as the required data is already available in the object instance. Export parameters are also not required in this case. Error handling will be done via method exceptions. This method will be called as a background task from within the workflow. However, technically, there is no difference between a dialog and background method with respect to the ABAP workflow class. The only difference is in the purpose. Dialog methods will perform some action on a screen, transaction, or app, whereas a background method will perform a query or update operation without any user interaction. We'll now

look at the concept of exceptions in workflow classes before going through our example for creating a new method in the billing document workflow class.

3.2.5 Method Exceptions

As explained earlier in Table 3.2, there are several standard exception classes in workflow such as `CX_BO_TEMPORARY` and `CX_BO_ERROR`. You can use these standard exception classes to raise an exception that will be caught by the workflow runtime and set the workflow to **Error** status. However, these standard classes don't have any message handling, so you wouldn't be able to populate your own message texts while raising any exceptions. Therefore, it's recommended to create your own exception class by inheriting from the standard class and adding the required message interfaces. You don't need any further customization in the custom exception class. You can reuse the same exception class across multiple workflows in your system if you want to.

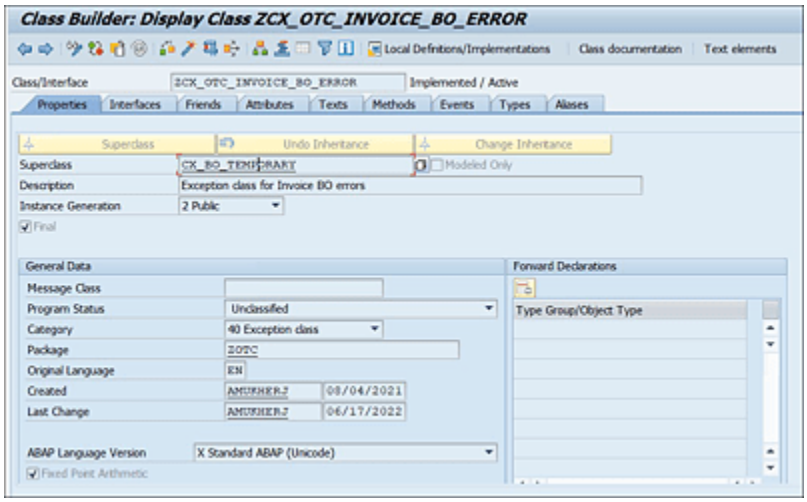


Figure 3.63 Creation of a Custom Workflow Exception Class Inheriting from a Standard Exception Class

Figure 3.63 depicts the custom exception class created for this method exception by inheriting from standard exception class `CX_BO_TEMPORARY`. Here, you enter the superclass as "CX_BO_TEMPORARY" and choose the class category as **Exception Class**. The remaining details are the same as any other ABAP class.

Exceptions for this class type when raised via workflow task method, will notify the workflow runtime to raise a temporary exception and set the workflow into **Error** status. The workflow can be automatically restarted after the error by the workflow system job `SAP_WORKFLOW_SYSTEM_TEMPORARY` in the job repository.

Add interfaces **IF_T100_MESSAGE** and **IF_T100_DYN_MSG** to this custom exception class, as shown in Figure 3.64. Enter the interface name directly in the **Interfaces** tab.

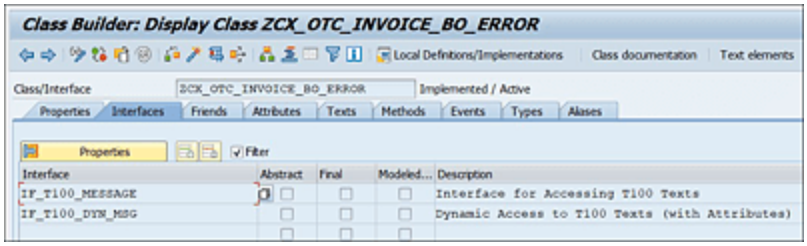


Figure 3.64 Add the Message Text Interfaces in the Custom Exception Class

Implement the constructor method of the exception class, and copy paste the source code in Listing 3.11 there.

```
CALL METHOD SUPER->CONSTRUCTOR
EXPORTING
PREVIOUS = PREVIOUS
CLASS_NAME = CLASS_NAME
INSTANCE = INSTANCE
```

```
COLLECTION = COLLECTION
LOG_NO = LOG_NO
LOG_MSG_NO = LOG_MSG_NO
FORCE_DATAFLOW = FORCE_DATAFLOW
.
clear me->textid.
if textid is initial.
  IF_T100_MESSAGE~T100KEY = IF_T100_MESSAGE=>DEFAULT_TEXTID.
else.
  IF_T100_MESSAGE~T100KEY = TEXTID.
endif.
```

Listing 3.11 Sample Source Code for an Exception Class Using Message Text Handling

Now your custom exception class is ready to be used within the workflow business class method.

Add the exception class to method definition **APPROVE_INV_ACCOUNTING** in the **Exceptions of Method** section of the screen shown in Figure 3.65.

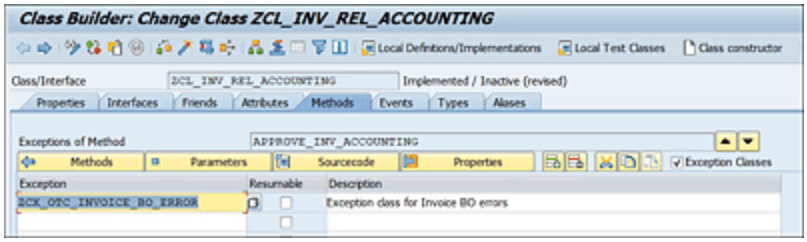


Figure 3.65 Add the Exception Class in the Exceptions of Method Tab

Now implement the method source code, as shown in Listing 3.12. Note the reference code for raising the exception using the custom exception class you just created. Listing 3.12 illustrates a typical example of a method call from a background task with some API calls for an update operation and corresponding error-handling functionality.

```
CONSTANTS : lc_posting TYPE char01 VALUE 'B',  " Posting of type CHAR01
            lc_msgty_e TYPE symsgty VALUE 'E', " Message Type
            lc_msgty_i TYPE symsgty VALUE 'I'. " Message Type
```

```
    DATA :
     lt_vbrk   TYPE STANDARD TABLE OF vbrk,    " Billing Document: Header Data
     lt_komfk  TYPE STANDARD TABLE OF komfk,   " Billing Communications Table
     lt_komv   TYPE STANDARD TABLE OF komv, " Pricing Communications-Condition
Record
     lt_head   TYPE STANDARD TABLE OF theadvb, " Reference Structure for XTHEAD
     lt_vbfs   TYPE STANDARD TABLE OF vbfs,    " Error Log for Collective Processing
     lt_vbpa   TYPE STANDARD TABLE OF vbpavb,  " Reference structure for XVBPA/YVBPA
     lt_vbrkvb TYPE STANDARD TABLE OF vbrkvb,  " Reference Structure for XVBRK/YVBRP
     lt_vbrp   TYPE STANDARD TABLE OF vbrpvb,  " Reference Structure for XVBRP/YVBRP
     lt_vbrl   TYPE STANDARD TABLE OF vbrlvb,  " Reference Structure for XVBRL/YVBRL
     lt_vbss   TYPE STANDARD TABLE OF vbss.   " Collective Processing: Sales
Documents
***Local data declarations
DATA: ls_message TYPE scx_t100key, " T100 key with mapping of parameters to
attribute names
      lv_message_text TYPE char80.      " Message_text of type CHAR80

*   filling the table with billing no
    lt_vbrk = VALUE #( ( vbeln = mv_vbeln ) ).

*    removing posting block for accounting release
    CALL FUNCTION 'SD_INVOICE_RELEASE_TO_ACCOUNT'
      EXPORTING
        with_posting = lc_posting "B
      TABLES
        it_vbrk      = lt_vbrk
        xkomfk       = lt_komfk
        xkomv        = lt_komv
        xthead       = lt_head
        xvbfs        = lt_vbfs
        xvbpa        = lt_vbpa
        xvbrk        = lt_vbrkvb
        xvbrp        = lt_vbrp
        xvbrl        = lt_vbrl
        xvbss        = lt_vbss.

*Error handling - raise exception
    IF line_exists( lt_vbfs[ msgty = lc_msgty_e ] ).
      ASSIGN lt_vbfs[ msgty = lc_msgty_e ] TO FIELD-SYMBOL(<ls_vbfs>).
      IF sy-subrc = 0.
        MESSAGE ID <ls_vbfs>-msgid TYPE lc_msgty_i NUMBER <ls_vbfs>-msgno
                WITH <ls_vbfs>-msgv1 <ls_vbfs>-msgv2 <ls_vbfs>-msgv3 <ls_vbfs>-msgv4
                INTO lv_message_text.
*Populate message key for exception
        ls_message-msgid = sy-msgid.
        ls_message-msgno = sy-msgno.
        ls_message-attr1 = sy-msgv1.
        ls_message-attr2 = sy-msgv2.
        ls_message-attr3 = sy-msgv3.
        ls_message-attr4 = sy-msgv4.
        RAISE EXCEPTION TYPE zcx_otc_invoice_bo_error EXPORTING textid = ls_message.
```

```
    ENDIF. " IF sy-subrc = 0
  ENDIF. " IF line_exists( lt_vbfs[ msgty = lc_msgty_e ] )
```

Listing 3.12 Sample Source Code for the Workflow Class Background Method

You can add more methods to your workflow class as shown here per your business requirement. Next, you'll add events to the workflow class.

3.2.6 Creating Events

You need at least one event to trigger the workflow. Although workflows can be triggered directly via an API, for custom applications, it's generally a best practice to use events for triggering workflow. This is because events provide an asynchronous delivery mechanism from the event creator to the event receiver. Event linkage can be activated or deactivated if required, and events can be added to the queue (if the event queue is active) to ensure that the workflow is triggered for all relevant cases.

You can use more than one event for a workflow. In fact, in many cases, multiple events are used for managing the workflow. Apart from the triggering event, you can use terminating events for asynchronous tasks and also use deleted or canceled events to abruptly terminate a workflow if the triggering transaction is no longer valid.

For this scenario, you'll define two events in the workflow class. Both events will be instance and public. Static events can be used if the object instance doesn't need to be created from the event. The two events are as follows:

- **`TRIGGER_INV_WORKFLOW`**
  This is the triggering event for the workflow. This event will have one parameter for `AUTO_RELEASE`. If this parameter is set by the calling application, then the approval will be bypassed, and the billing document will be released to accounting immediately.
- **`CANCELLED_INVOICE`**
  This will terminate the workflow abruptly (via a fork and wait step) if the invoice is canceled.

Figure 3.66 shows details of events added in the workflow business class. Here, we've entered the event name, selected the **Type** as **Instance Event**, set the **Visibility** as **Public**, and entered a short description for the event.

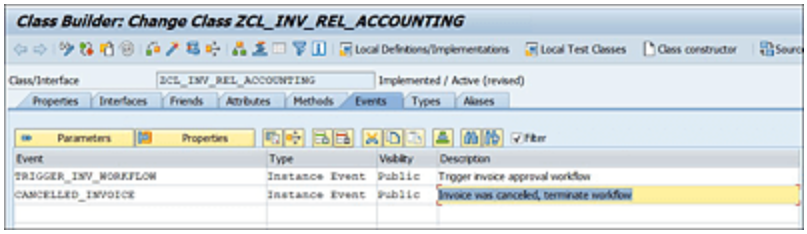


Figure 3.66 View of Events Added in the Workflow Business Class

In Figure 3.67, we've created an import parameter for the event by entering the name and data type. We've also marked the parameter as optional.

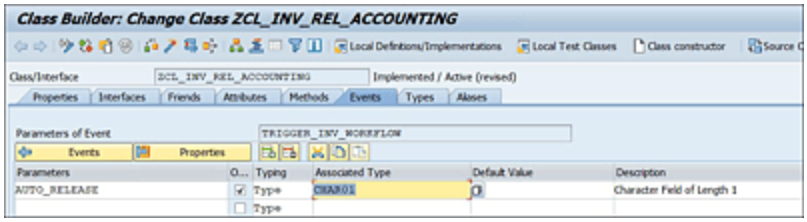


Figure 3.67 Creating an Event Parameter in the Triggering Event of the Workflow

Now the workflow business class definition is complete. You can go ahead and unit test the workflow class.

3.2.7 Testing an ABAP Workflow Class

Unit testing of an ABAP workflow class can be done via Transaction SE24, just like any other ABAP class. You just need to create an instance of the class by entering the key fields for testing or executing the instance components.

Figure 3.68 shows the unit test screen of a workflow class from Transaction SE24. The import parameters of the `CONSTRUCTOR` method are available for input here. Once you've entered the parameter values, click on the **Create Instance** button.

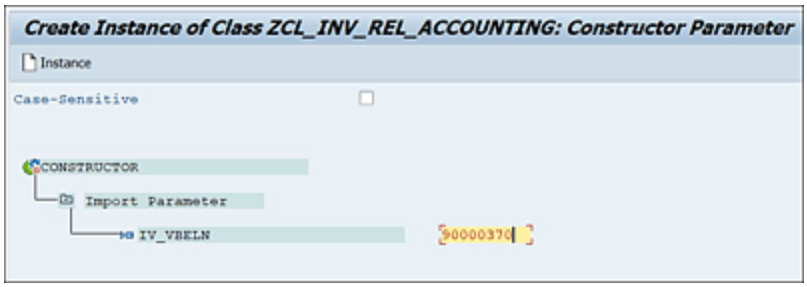


Figure 3.68 Unit Test Workflow Class from Transaction SE24

Figure 3.69 shows the result screen after the object instance creation, with the attribute values populated. Now you can execute and test any instance methods from the class.

Now you can execute your instance methods (enter parameters if any) and debug the code if required.

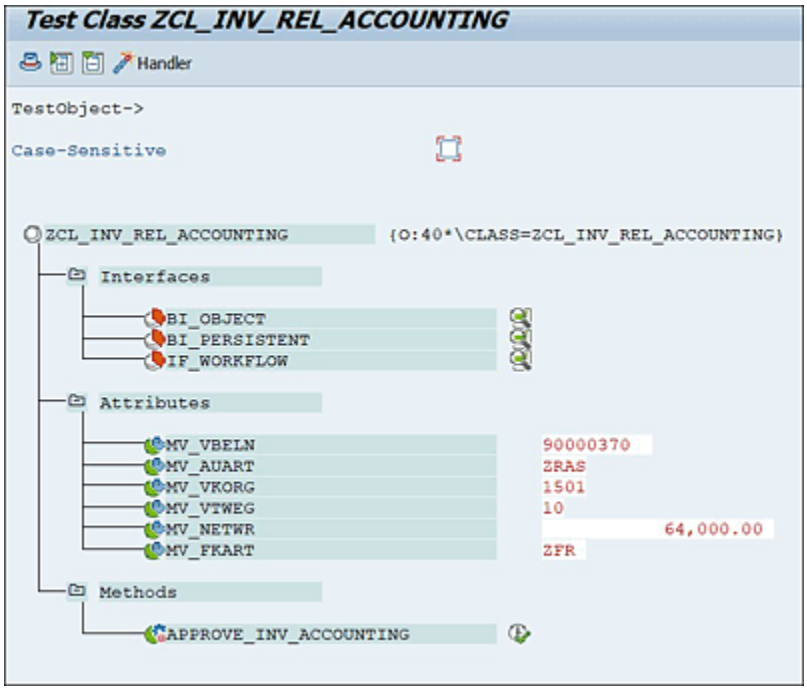


Figure 3.69 Instance Creation of the Workflow Class

3.3 Task Definition

In workflow terminology, tasks are the basic building blocks of a process flow. Tasks can be single-step or multistep, and accordingly they are more commonly classified as standard tasks or workflow templates. You can create standard tasks or workflow templates from Transaction PFTC.

In this section, we'll be discussing single-step tasks, or standard tasks, and sometimes refer to them simply as tasks. Tasks can exist standalone or can be part of a workflow definition as an activity step. These objects can execute your BOR or ABAP class method in a workflow definition. Methods with result parameters, also called functional methods, can be called via expressions in a condition step or in a container operation step of a workflow, but normal method calls are possible only via tasks in the workflow.

Apart from activity steps, tasks are also implicitly created or used in user decision steps and send mail steps in a workflow definition. These tasks refer to some standard BOR method and are used to execute some typical standard functionality in a workflow. For example, the user decision step allows the user to review a document sent for approval and then approve or reject it by clicking on the corresponding button. They can add some supporting comment to justify their action. This type of task always executes the standard BOR method `PROCESS` of BOR type `DECISION`. Similarly, the send mail steps in workflow are used to send a notification via workflow. This step always

executes the BOR method `SENDTASKDESCRIPTION2` of BOR type `SELFITEM`.

For executing your own BOR or ABAP class methods, you normally create an activity step in a workflow that executes a task, or the task can exist standalone and be called directly from a triggering event. In the next few sections, we'll specifically look at how to define a standard task. We'll also discuss the most common settings specified at task level and their implication in the workflow.

3.3.1 Defining Standard Tasks and Task Settings

Standard tasks can be created from Transaction PFTC. While creating a task, you enter the **Task type** as "TS – Standard Task" and enter a name per your naming convention, for example, "ZTSRelInvoic". Then, click on the **Create** button in the toolbar.

Under the **Basic data** tab of the task definition, enter the following details:

- **Name**
  This is a short description of the task, for example, "Release Invoice to Accounting".
- **Release Status**
  This field is purely for reference to the developer and doesn't drive any functionality of the task. You'll normally set the **Release Status** to **Implemented** for custom task definitions. For SAP standard tasks, if the **Release**

Status is **Modeled** or **Obsolete**, then it can indicate that the task isn't used anymore.

- **Work Item Text**
  This is the subject line of the work item that is created at runtime from the task. Depending on whether the task is dialog or background, the work item text is either visible in the workflow inbox as a subject or only on the workflow log. The work item text can use any variables from the task container element.
- **Object Method**
  In this section, you select the **Object Category** as **BO BOR Object Type** or **CL ABAP Class** according to the approach that selected. Then, in **Object Type**, enter either the BOR object type name or the ABAP class name, and enter the corresponding method name in the **Method** field. As soon as you press `Enter`, the other task settings are populated (if it's the BOR object type) such as **Synchronous** or **Asynchronous** and **Dialog** or **Background**. For BOR types, these settings are derived from the underlying BOR method definition and are noneditable in the task definition. For ABAP classes, the method doesn't have any setting for dialog/background and synchronous/asynchronous, so these settings are entered at the task level.
- **Confirm end of processing**
  This setting, along with the **Executable with SAPforms** setting is mostly obsolete. The **Confirm end of processing** checkbox was used for dialog tasks to add a confirmation popup to the task processor that would indicate the completion of processing for the work item,

removing it from their inbox. If the task processor doesn't confirm, then the work item remains in his inbox.

Let's create a standard task for the billing document release workflow, where you had created the custom class `ZCL_INV_REL_ACCOUNTING` in Section 3.2.1. This will be a background task that will call method `APPROVE_INV_ACCOUNTING` and release the billing document to accounting. Create the task by entering a short ID (**Abbr.**) and the other details, as shown in Figure 3.70. Note the task ID ❶ and the binding editor button ❷. Once you save the task, a number is generated based on the prefix number customizing in Transaction SWU3. The **Task** is marked as **Synchronous** ❸ and **Background** ❹ by selecting the checkboxes highlighted. In addition, the billing document number is added as a variable in the work item text by selecting from the default object type element in the task container.

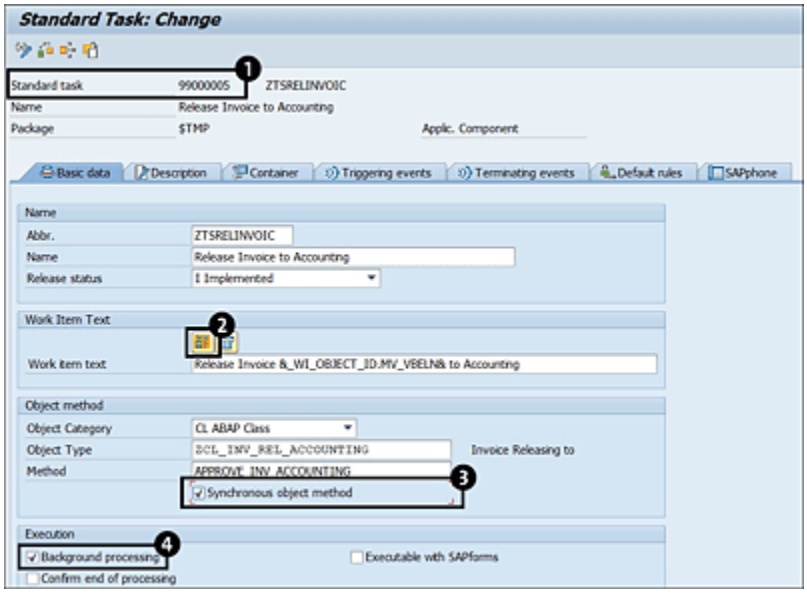


Figure 3.70 Basic Data Definition of a Standard Task for the Workflow

Let’s now look at some additional settings and tabs, as follows:

- **Binding editor**
  Binding at the task level maps the container elements from task to method and back from method to task. This means that the data flows from task container elements to the method import parameters and then the method export parameters are returned back to task container elements. The **Binding editor** button is visible in the **Object method** section under **Basic data** tab, if the underlying object method has any parameters. If the method doesn’t have parameters, then the **Binding editor** button isn’t displayed.
  In our example, method `APPROVE_INV_ACCOUNTING` of class `ZCL_INV_REL_ACCOUNTING` didn’t have any parameters, so the **Binding editor** button isn’t displayed. However, Figure 3.71 and Figure 3.72 from another task show the binding option in standard tasks.

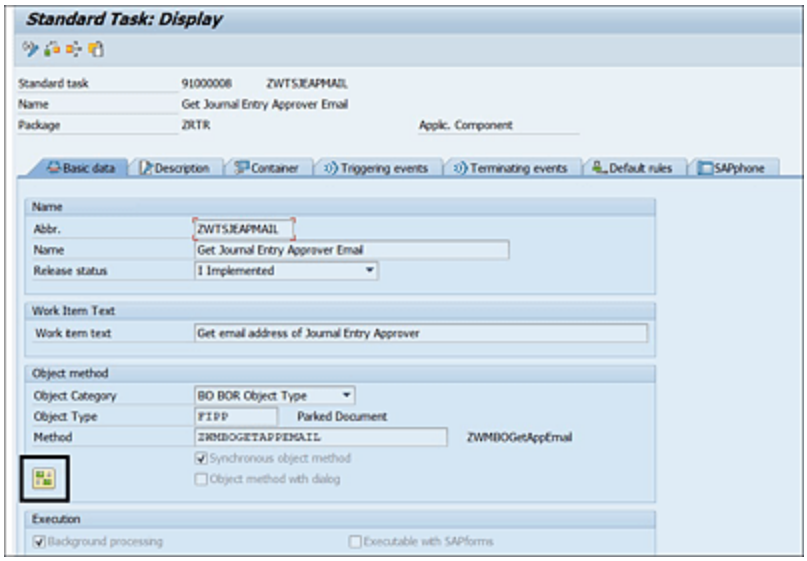


Figure 3.71 Binding Editor Button in the Standard Task Definition

In Figure 3.72, the top half of the binding editor maps the task container elements to the method import parameters and the bottom half of the binding editor maps the method export parameters to the container elements.

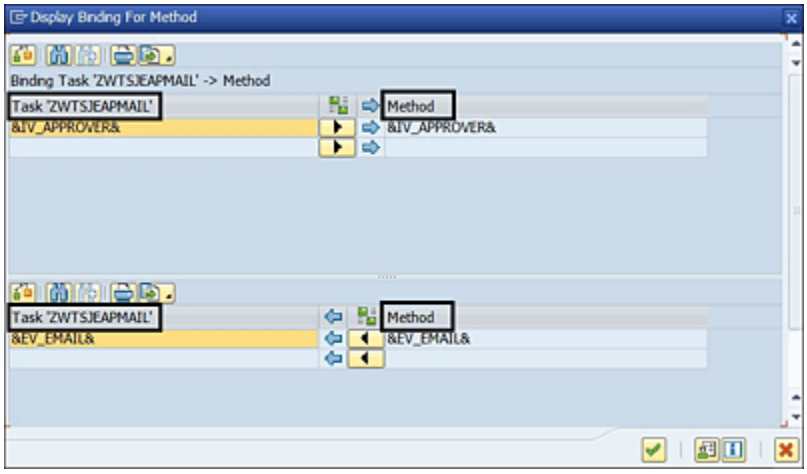


Figure 3.72 View of Binding Editor for Mapping Task and Method Container

- **Container**
  The task container lists elements used to hold the data at the task level. There are many standard container elements that come with a task definition; the most important one is the `_Wi_Object_ID` element, which is a system element for the BOR object type or the ABAP class that is entered in the **Object method** section of the task definition. This element contains the instance of the task object type. Similarly, the `_Wi_Actual_Agent` element stores the actual agent after the execution of a dialog task, and the `_Attach_Objects` element contains the instance of all attachment objects (of BOR type `SOFM`) that are linked to the task. You can also create custom container elements to receive data from the workflow container or event container. These container elements on the task can be used for many purposes:

- Explicit and implicit data mapping to the method container for method execution. The object type instance from `_Wi_Object_ID` is implicitly passed to the method, but additional method parameters must be maintained in the binding definition.
- Task container elements can be used in the work item text or the work item description. Refer to Section 3.3.2 for further details.
- Attachment objects mapped to `_Attach_Objects` element are available to the user executing a dialog task for review.
- The `_Rule_Result` container element returns the agents from the **Role** resolution based on **Default Rules** maintained in the task under the corresponding tab.

- **Triggering events**
  This tab is only relevant for single-step tasks where the task is triggered by an event. In this case, you need to maintain the triggering event from the BOR object type or ABAP class and then maintain the binding between event container and task container. The event linkage can be activated directly from this screen like a workflow definition.
- **Terminating events**
  This tab is relevant for asynchronous tasks that require a terminating event to complete the execution and return the control back to the calling step in the workflow. Here, you enter the container element that must have the type of the BOR object type or ABAP class defining the terminating event. Binding can be maintained between the event and task container to receive the object

instance along with any event parameters into the task. Refer to Chapter 5, Section 5.4, for more details on terminating events.

- **Default rules**
  If your process flow consists of a single-step dialog task only, then you'll use the default rule mechanism to perform the agent determination for the task. Here, you must enter the rule number, which has been previously created in Transaction PFAC and then maintain the binding between the task container and rule container for the import parameters of the rule. There are several other less frequently used types of rules that can also be maintained at the task level, for example, the recipients for missed deadlines. Refer to Chapter 6, Section 6.3, for more details on default rules.

3.3.2 Work Item Text and Description

Work item text is the subject line of a task, which describes in short, the primary objective of the task. For a dialog task, this normally describes the action required on a document from the selected agents of the task and is visible to the users in their workflow inbox. Some parts of the work item text can be static while other parts can be variable based on elements inserted from the task container. For a background task, this text is usually visible from the workflow log, except for send mail tasks, where the work item text is the email subject line to the recipient's email inbox.

Work item description is the long text of a task, which explains in detail the purpose of the task along with details

of the transaction, information on previous approvers (if any), comments or notes, and any other details relevant for that task. For a dialog task, these details provide the approver with all the necessary information for executing the task. For a background task, this information is usually for the administrator's reference, except in the case of a send mail task, where the task description is sent out as the body text of the email.

Along with the task description, other runtime objects that can be sent out in a dialog task include attachments and object links. Object links are BOR or ABAP class objects that are attached as a hyperlink on the task via the `_Wi_Object_ID` container element or any other object type container elements in a task. When the user clicks on the hyperlink for this object, it navigates to the transaction called by the default method (usually `DISPLAY`) on the BOR type or ABAP class. Attachments are documents such as Microsoft Office documents, text files, and PDF files that can be attached programmatically or by a user to a workflow step. These attachments provide supporting documentation for the subsequent approvers in a workflow.

Figure 3.73 and Figure 3.74 show a couple examples of work items from a user decision step and a send mail step in a workflow, respectively. Figure 3.73 shows the view of a work item text ❶ and description ❹ for a user decision step in a workflow from the My Inbox app. It also provides icons for attachments ❷ and object links ❸.

Figure 3.74 shows the view of the work item text ❶ and description ❷ for a send mail step in workflow from Outlook inbox.

The task description can be maintained like any other long text objects in SAP. Here, the variables can be inserted using the **Expression** button (see Figure 3.75), which allows you to select task container elements along with some common system elements such as date, time, client, user ID, and so on.

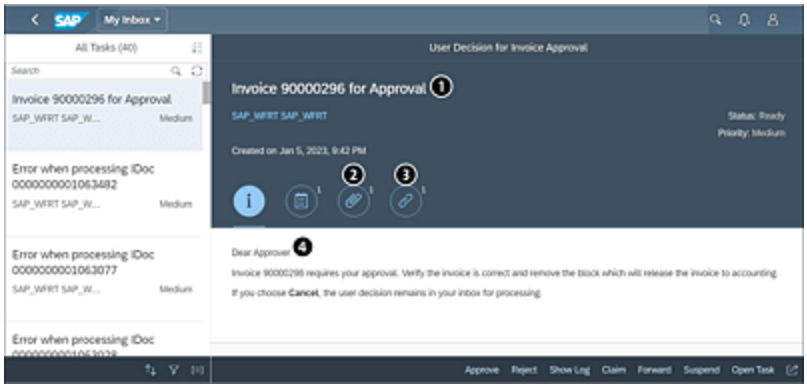


Figure 3.73 View of Work Item Text and Description for a User Decision Step in the Workflow from the My Inbox App

Figure 3.74 Work Item Text/Description for a Send Mail Step in the Workflow from the Outlook Inbox

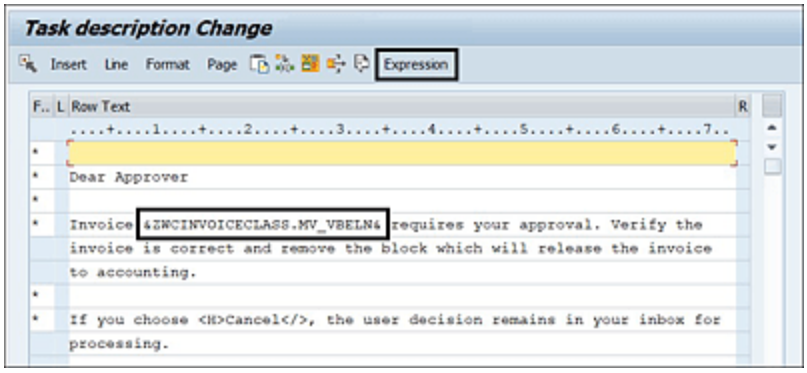


Figure 3.75 View of Task Description with Long Text Maintained Using Task Container Elements as Variables

You can also add some simple conditions in the work item description with `IF-ENDIF` statements using the command-line format **/:**, as shown in Figure 3.76. If you enter the text format as "/:", then the text is treated as a command line. In the example in Figure 3.76, the following is true:

- The command line `IF &ZWCINVOICEBO.MV_BUYER_MANAGER& NE &SPACE&.` checks whether the value of the task container element is or isn't blank.
- Then, the following line without any format, `and manager &ZWCINVOICEBO.MV_MANAGER_NAME&`, is the text that is inserted in the work item description if the preceding condition is true.
- Following the command line, `ENDIF` closes the condition started using command line `IF`.
- If the condition in the `IF` command line evaluates to false, then the following text line isn't inserted in the work item description at all. In addition, the command lines themselves don't appear in the work item description at runtime.

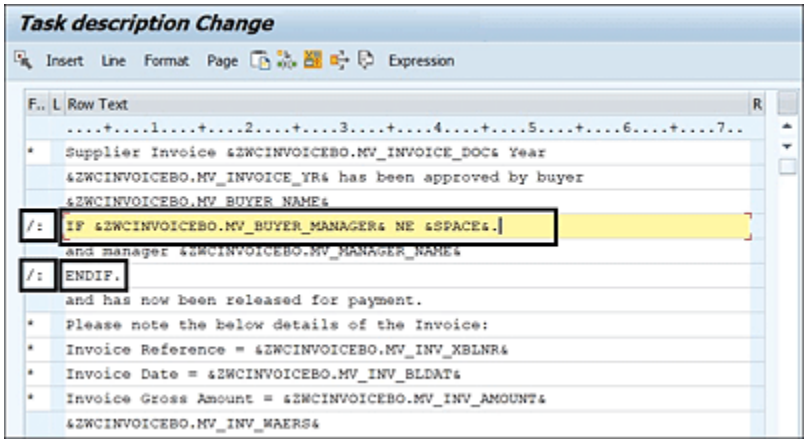


Figure 3.76 Task Description with Conditional Text Using IF-ENDIF Command Lines

3.4 Summary

In this chapter, we explored the two different approaches for building the business logic in workflow. First, we explored the BOR type approach by looking at key definitions, a step-by-step guide to building a custom BOR type with components, creating a subtype of a standard BOR type, delegation, unit testing, and BOR programming details. Next, we looked at the ABAP class approach, with a step-by-step guide to building a workflow business class, implementing standard methods, creating custom attributes and methods, and unit testing the class. Finally, we've looked at the steps to create a standard task in the workflow with all its attributes and settings, as well as how to define work item text and description in a standard task.

4 Defining Workflows and Adding Steps

This chapter teaches you how to create workflows with the Workflow Builder, an essential tool used to create, display, and process workflows. You'll become familiar with the tool and then define a workflow by adding steps, tasks, and enhancements. You'll also learn to execute and test your own workflows.

We'll start this chapter by explaining how multiple activities/steps can be stitched into an integrated process flow in the Workflow Builder. A workflow consists of a set of tasks. Each task has a purpose to complete one activity and move to the next step. The data/information from one step to the next is sent via container. This contains the workflow lifetime information and is the mechanism to flow the static and instance-specific information from one step to another throughout the lifecycle of the workflow. One of the main purposes of the workflow is to connect sequential/parallel user activities automatically, so that the person responsible for the next step doesn't need to check when he needs to perform his activities. But if a responsible person doesn't complete his tasks, then deadline monitoring can be set up to remind the responsible person to complete the activities. Deadline monitoring is also used for escalation management

and proper routing of tasks if the current owner isn’t completing his task.

We’ll discuss in this chapter the common components of workflow, how the end-to-end flow works with exchange of data, and how the workflow can be configured to continue execution if it’s stuck due to the inactivity of the current activity owner.

4.1 Workflow Builder

The Workflow Builder provides a graphical view of end-to-end activity integrated steps in one central tool. This will help to create, display, change, and test any workflow.

The Workflow Builder is accessed using Transaction SWDD. You can also use menu path **Tools** • **Business Workflow** • **Development** • **Definition Tools** • **Workflow Builder** • **Workflow Builder** to open Workflow Builder. Figure 4.1 shows the Workflow Builder and the different navigation and workflow components needed to build a workflow.

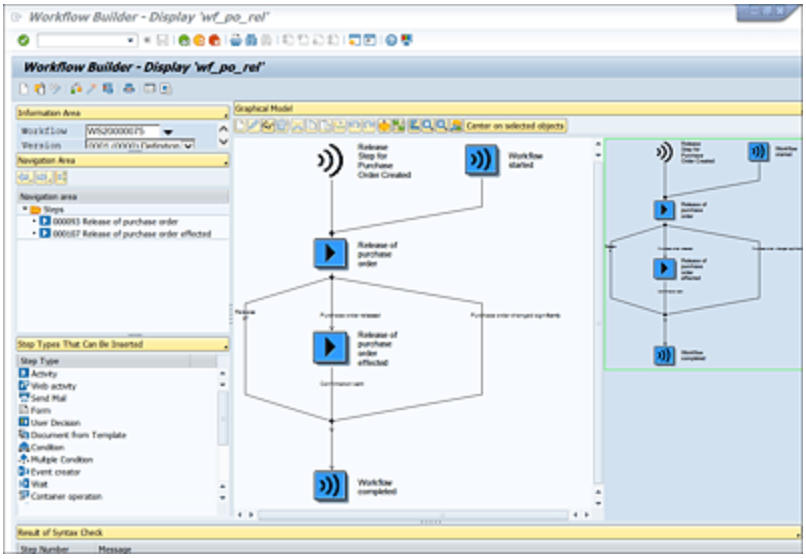


Figure 4.1 Workflow Builder Sections

The Workflow Builder has different sections that help you navigate through the flow and also create the components to build the workflow. The following describes and illustrates the different sections of Workflow Builder:

- **Information Area**
  This area contains the workflow number and version of the workflow. The current active version is loaded when Workflow Builder is open by default, but you also can select the previous version to view. The workflow follows the execution path as the active workflow when it's triggered. For example, the system triggered a workflow on November 1, 2023, and the current version of the workflow is 000. The workflow will start the activity flow per the active version (000). Let's say the developer changed the workflow by adding a few more steps and created another version called 001. So, the new active version will be 001. Any new workflow triggered will follow the 001 process flow steps. But the workflow triggered on November 1, 2023, will follow the 000 version. So, it may be needed to view the earlier version of the workflow for any old active workflow troubleshooting. Figure 4.2 shows the **Information Area**, which is normally found in the top-left corner of the Workflow Builder screen.

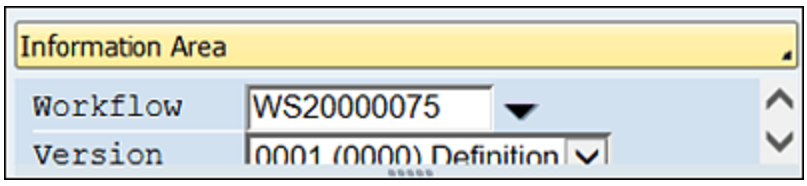


Figure 4.2 Workflow Builder: Information Area

- **Navigation Area**
  This section shows all the steps in the workflow. You can go to the details of the steps by clicking **Steps** in this section. The workflow graphical diagram can be complex, and navigation to a particular step from the workflow graphical diagram can be time-consuming. This area helps with quick navigation. It's important to provide a meaningful and easily identifiable step name so that navigation to the right step is easy. A number is associated with each step and is available in the technical log of the workflow. This will help to easily connect a workflow log step to the details view of the step in the workflow definition. Figure 4.3 shows the two steps of workflow `WS20000075` in our example.

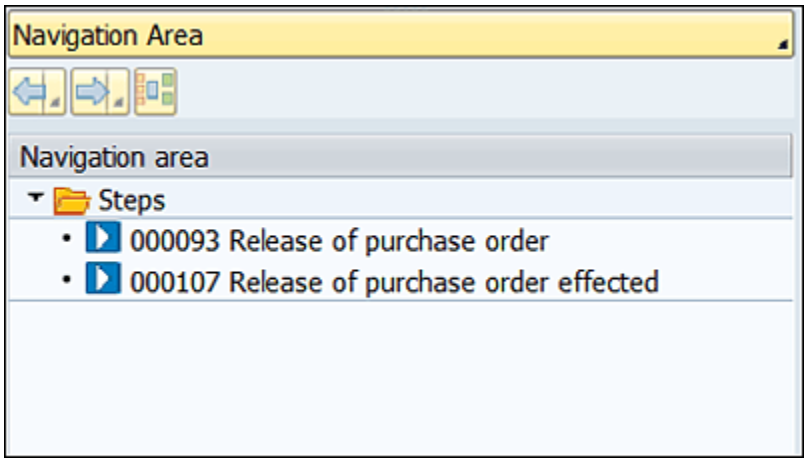


Figure 4.3 Workflow Builder: Navigation Area

- **Step Types That Can Be Inserted**
  Step Type is shown by default when you open Workflow Builder. It shows all the possible step types that can be inserted in the workflow. You can drag and drop any specific step into the location where you want to add the step in the workflow graphical area. You'll get a plus (+) sign in the workflow graphical area where the step can be

added. You can select any existing step in the workflow graphical area and double-click on the new step needed, which will create a new step after the step you selected earlier. Figure 4.4 shows the possible step types that can be added in the workflow. The details of each step will be discussed in Section 4.2.

Other information can be displayed in this section besides step type. You can click the button just above this section to get the options of more information that can be displayed in this section. You can switch into the view you want to display in this section depending on your need. Figure 4.5 shows the options that will appear when you click on the gray button just below **Navigation Area**.

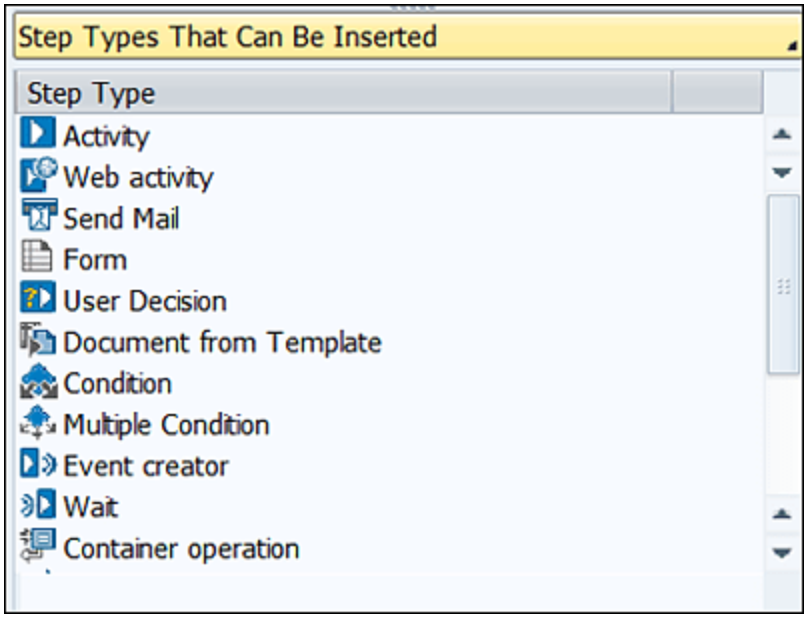


Figure 4.4 Workflow Builder: Step Types

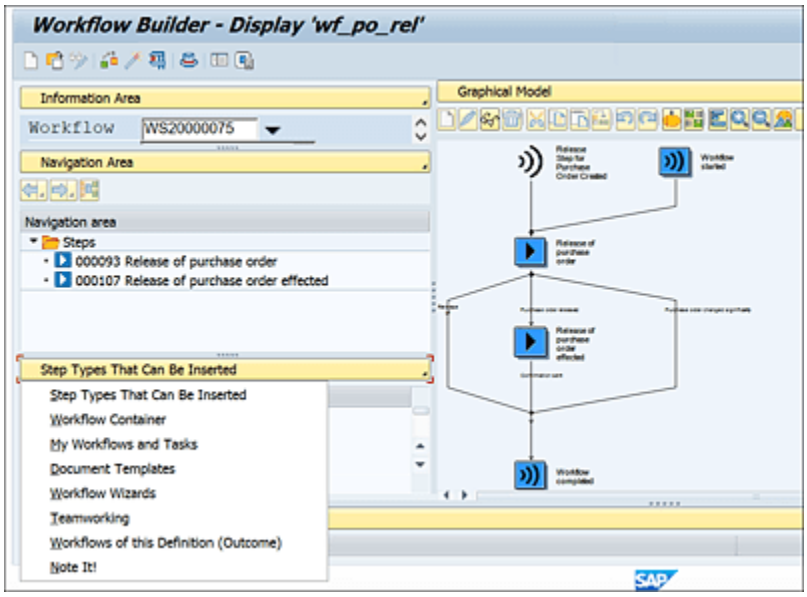


Figure 4.5 Workflow Builder: Other Components List

As Figure 4.5 shows, you can choose different components that can be shown in this area. The details of those workflow components are as follows:

- **Workflow Container**
  The workflow container stores all the data elements defined within the workflow instance. This is similar to the data variable you define for any development. There are some container elements created automatically when a workflow is created (e.g., business objects), but there are also some custom workflow containers that are also needed to pass the information from multiple steps. Workflow container elements are used to pass information from the main workflow instance to different steps. It’s also used for dynamic content of any description and texts.
- **My Workflows and Tasks**
  This list shows all the workflows you’ve created/changed.

- **Document Templates**
  There is a step type called document form templates. All the workflow document templates that can be used for the document form template step is shown here.
- **Workflow Wizard**
  If you want to create workflows using a wizard, you can try out this section. All possible standard workflow wizards are available here.
- **Teamworking**
  If multiple people are working on the same workflow, you can find out who has changed any step. You can find out those steps in the workflow with different search criteria such as last changed by, created by, and so on.
- **Workflows of this Definition (Outcome)**
  All the current running in-process workflows for this workflow definition are shown in this section.
- **Note it!**
  You can create notes and documents about this workflow in this section.

- **Message Area**
  The bottom part of the screen shows the error/warning/information messages during the activation or syntax check of the workflow.
- **Overview**
  The rightmost part of the pane shows the entire workflow in one pane. You can zoom within this section, and the zoomed portion will be displayed on the **Graphical Model** section. You'll find a green rectangular box

highlighting the zoom area. This will be very helpful for zooming into a portion of the workflow when it has lots of tasks and complex and not readable in one frame. Then you can zoom onto the required section and further navigate into details from the **Graphical Model** view.

- **Graphical Model**
  The middle part of the screen shows the graphical workflow, including a zoomed view. This section is used for further detailed navigation into tasks and other activities.

It’s recommended to design and model the workflow activity flow before you start creating the building blocks in the Workflow Builder.

Note

The Workflow Builder graphical view is used to understand the end-to-end workflow process flow. It’s also used to understand current state and historically completed steps during workflow error troubleshooting. If you want to develop a complex workflow with lots of steps, try to develop multiple subworkflows and integrate them with the main workflow to help with readability.

4.2 Common Workflow Steps

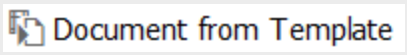
Steps are the building blocks of a workflow and are needed to model the workflow. Table 4.1 illustrates the different steps used to build the workflow.

Step Type	Step Icon	Step Functions/Purpose
Activity		Any task or subworkflow can be executed using this step. This is the main processing unit of the workflow. Runtime data is passed from the workflow container to the activity (step/subworkflow) container, and similarly data can be passed back from the task/subworkflow container to the main workflow container after the activity is complete. The activity step can be used for any background/user action processing.

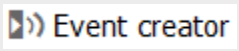
Step Type	Step Icon	Step Functions/Purpose
Web Activity		This step is used to send the data as an XML document. Any container element with ABAP Data Dictionary or XML_DOC as the data type reference is sent as an XML message. If the **Wait for Feedback** checkbox is set, the associated work item for the step is only completed once the XML response is received.
Send Mail		This step is used to send any mail. The mail can be sent to any external main address or organization object (e.g., position, job, user ID, etc.). The recipient can be dynamically passed using the workflow container.

Step Type	Step Icon	Step Functions/Purpose
Form		You can display the data in the workflow container in the form. The form can be opened in display or in edit mode. Users will have **Approve** and **Reject** buttons to process the form. Use the workflow wizard to generate the basic form, and then make changes in the form for additional requirements. The *to* and *from* data transmission can be done using the workflow container element.
User Decision		This step creates a work item that the user can take action on. The work item contains relevant text for users to make decisions. Any dynamic value can be added in the text using the workflow container element, and the user is

Step Type	Step Icon	Step Functions/Purpose
		provided with options to use in making a decision. The number of processing branches are created based on the number of options given in the user decision. Each action from the user can be processed in a separate workflow branch. User decision is the easiest way to create a step to get approval/rejection or any other action from the user. If the user doesn’t need to add/edit any information during approval, this step can be used.

Step Type	Step Icon	Step Functions/Purpose
Document from Template		This step can use the document created in the local application and passed through the workflow. The document can contain variables that are dynamically filled by the workflow container element. You have to create the template when you’re creating this step. The document is opened in the local application when the work item is executed. The document is then saved in the workflow container once it’s completed.

Step Type	Step Icon	Step Functions/Purpose
Condition		Condition is used to branch the process flow into two separate flows. Container element variables can be used to define the condition. The outcome of the condition comes in the form of **True** and **False**. The processing of the branch is carried out accordingly.
Multiple Condition		This is similar to the **Condition** step. You create multiple branches of the process flow based on a container element value. If a value isn’t in the list of predefined probable values, then a separate branch of another value is used to process.

Step Type	Step Icon	Step Functions/Purpose
Event Creator		This step is used to trigger events of the same or another workflow. The event you’re triggering should be registered against the business object/class. The business object/class information is passed using the workflow container. This should not be used for a wait event in a different branch in the same fork. It can be used to start another workflow or pass event information between other subworkflows and different branches of the workflow process.

Step Type	Step Icon	Step Functions/Purpose
Wait for Event		This step waits for a specific event to be triggered by the system. If there are dependencies with other steps not in the same fork, then use the **Wait for Event** step to take care of those dependencies on other parallel activities. The object for which the event should be triggered needs to be in the workflow container. The container element from the triggered event to the workflow container can be passed using a container binding.

Step Type	Step Icon	Step Functions/Purpose
Container Operation		This step is used for any arithmetic operation on the container element. There are two options of container operation: **Assign** and **Append**. For **Assign**, the calculated outcome is to replace the existing content of the container element. The **Append** operation can only happen for a multiline container. It will add the calculation outcome into the existing container element as a new line.
Process Control		This step is used to cancel and terminate an in-process work item or workflow. There are a few scenarios that can be handled using **Process Control**: • **Cancel work item** This will cancel another work item

Step Type	Step Icon	Step Functions/Purpose
		located in the same fork. There won't be further processing of any subsequent steps of the work item that is canceled. The step number of the workflow is maintained in the **Workflow Step** field of the **Process Control** step. • **Set work item to obsolete** This sets another work item status to **Completed** in the same workflow. You have to enter the workflow step number that should be completed. The subsequent step of the completed work item continues processing if the branch processing is obsolete. So, there should be a

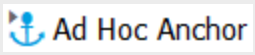
Step Type	Step Icon	Step Functions/Purpose
		processing obsolete exception handled in the work item. Normally, it’s not used to set the **Completed** status of any work item in the same fork; instead, it’s used to set the **Obsolete** status and continue processing in another part of the same workflow. • **Terminate workflow** This terminates the current workflow and sets the work item status to **Logically Deleted**. There won’t be any subsequent processing of this workflow. But if it’s a subworkflow, then the parent workflow continues the execution, and all container element binding is updated before the

Step Type	Step Icon	Step Functions/Purpose
		subworkflow is completed. • **Cancel workflow** This is similar to **Terminate workflow**. But if it’s a subworkflow, then the parent workflow branch containing this workflow won’t be processed further.

Step Type	Step Icon	Step Functions/Purpose
Process Control (Cont.)		• **Cancel workflow (including all callers)** This step cancels all work items in the current workflow and updates with a status of **Logically Deleted**. It also cancels all the work items of the parent workflow and the parent workflow as well. • **Throw exception** You can trigger local event or exceptions defined in the version-specific tab of the workflow. You have to handle the exception in a separate block.

Step Type	Step Icon	Step Functions/Purpose
Loop (Until)		This step is used to loop the sequence of activities till the loop exit condition is met. At least one execution of steps is carried out before you can exit from the loop. You have to enter the condition of exit in the condition editor. The outcome of the condition will be either **True** or **False**.

Step Type	Step Icon	Step Functions/Purpose
Fork		A fork is created when there is a need for parallel processing by multiple users. You can define how many branches are mandatory to complete the entire block of the fork. You can also enter any condition to terminate the workflow. If either number of mandatory branch or exit condition matches, then it completes the fork execution and moves to the next step in the workflow.

Step Type	Step Icon	Step Functions/Purpose
Ad Hoc Anchor		You can define the workflow that can be dynamically triggered in this step. An authorized person can replace the ad hoc anchor with one of the possible workflows maintained in this step. This new workflow doesn’t replace the existing workflow. It works similar to a subworkflow, and all the steps of this added workflow are inserted into the main workflow. The container element of the added workflow should be identical to the main workflow.

Step Type	Step Icon	Step Functions/Purpose
Block		This is used to group multiple steps together. **Block** can have a local container element that can be used by all the steps within the block, but isn’t available in the main workflow container. Any container element can be passed to and from the main workflow container using container element binding. **Block** also has a deadline monitoring capability for the entire block consisting of all steps.

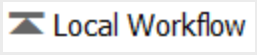
Step Type	Step Icon	Step Functions/Purpose
Local Workflow		This is triggered from local events within the main workflow. These aren't reusable for multiple other workflows. This is mainly used for exception handling. Any container element can be passed from the main workflow to the local workflow using container element binding.

Table 4.1 List of Commonly Used Workflow Steps

Let's walk through a standard workflow with some key steps and see how the entire workflow is designed. The standard SAP system has one framework workflow `WS10000051` for financial accounting document parking and release. It handles document amount release and account assignment approval through this workflow. The framework workflow uses other standard workflows depending on the level of approval needed (one level, two level, or three level approval). We won't go through the entire functionality of document parking. We'll focus on how the building blocks are used to design the workflow.

You can open the workflow from Transaction SWDD. Enter the workflow number “WS10000051” in the information area, and press the Enter key. Figure 4.6 shows the document parking workflow **WS10000051**.

The workflow has lots of steps. You have to zoom in on the overview area so that it’s readable in the **Graphical Model** frame.

The Workflow Builder can be opened from Transaction PFTC also, as well as via menu path **Tools** • **Business Workflow** • **Development** • **Definition Tools** • **Tasks/Task Groups** • **Display**. Here, enter **Task type** as “Workflow Template”, and then enter the workflow number in the **Task** field. Finally, click on **Display**. Figure 4.7 shows the Transaction PFTC screen where you have to enter the workflow details.

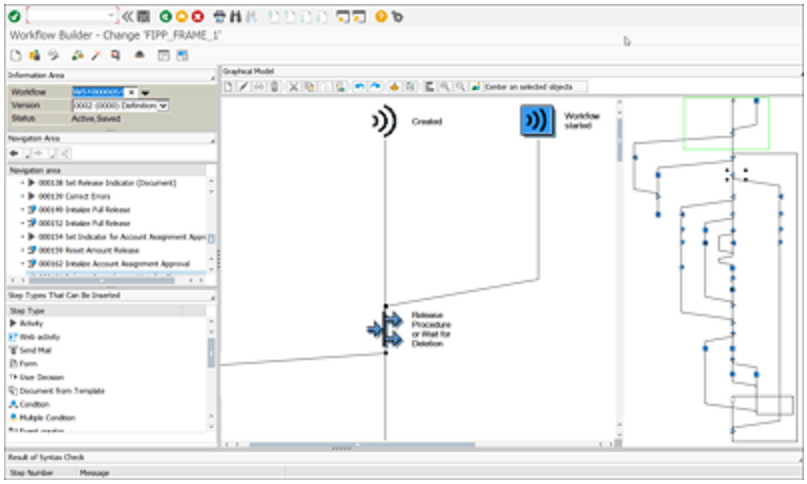


Figure 4.6 A Sample Workflow View in Workflow Builder

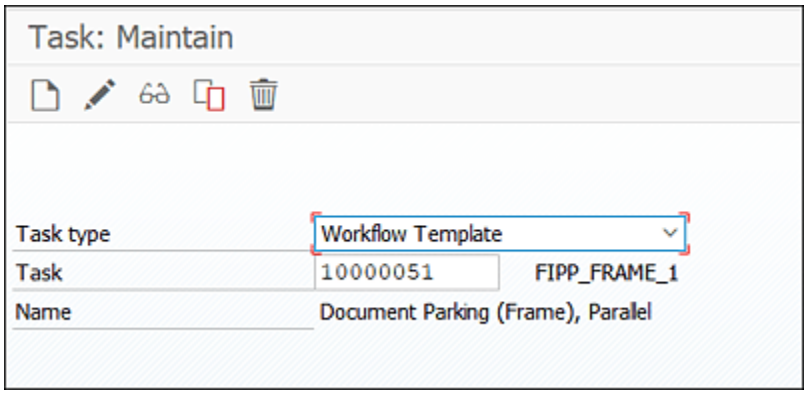


Figure 4.7 Open Workflow Builder from Transaction PFTC

This screen will display the technical details and attributes of the workflow. You can add/modify container element and triggering events in this view. You can also navigate to the workflow graphical view by clicking the **Workflow Builder** button in this screen. Figure 4.8 shows the technical details of the sample workflow `WS10000051` we're referring to here.

Click on the **Workflow Builder** button to navigate to workflow graphical screen.

You have to first think how the workflow will be triggered. If you find the Created icon in the Workflow Builder, it means this workflow can be triggered using an event. You can double-click on this icon to navigate to the event. To add a triggering event, choose **Goto** • **Basic Data** • **Version-Independent (Task)** • **Start Event**. You can define the triggering event, container element binding, and activation of triggering event here. Figure 4.9 shows the triggering event mapping for workflow `WS10000051`. You'll find the details about triggering events in Chapter 5 and container elements in Section 4.4 of this chapter.

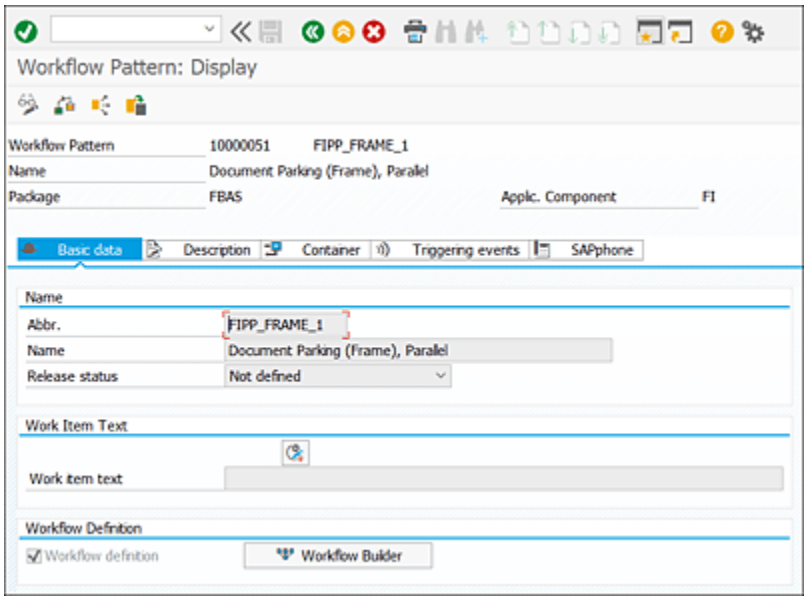


Figure 4.8 Workflow Technical Details View

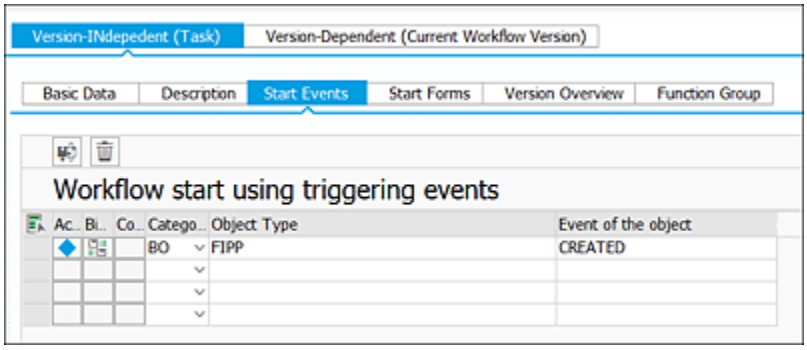


Figure 4.9 Workflow Triggering Event Mapping

Next, you have to design the flow of the process and different activities of the workflow. You can’t just add up steps without having a view of the overall design and flow of the process because you’ll end up with a complex workflow diagram, making it difficult to manage that workflow in the future. You have to think of the number of reusable steps, number of parallel branching, and exception handling before you start creating any steps. You may think of creating a subworkflow for reusable steps to make it more modular.

Determine the user activity steps and exception flow before creating other background steps.

You can start building a workflow with two branches: one for normal processing and the other to handle any deletion/withdrawal of the business transaction, if there is a scenario to delete the document. In our scenario, a **1 From 2** fork is used. One branch is waiting for any deletion event and the other branch will post the financial accounting document after successful approval. Figure 4.10 shows the workflow starting with two branches: one is waiting for deletion, and the other is the normal processing branch.

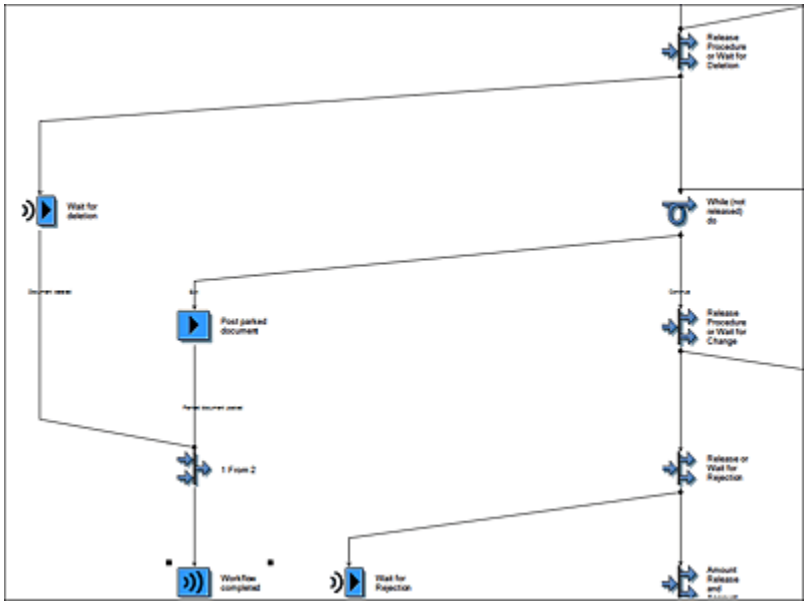


Figure 4.10 Workflow with Fork, Wait for Event, Activity, and Loop Steps

Double-click on the fork step (**1 From 2**) to see the details of the **Fork** step. Figure 4.11 show the details of the **Fork** step in our example. The number of **Parallel Branches** is **2** and **Necessary Branches** is **1**. That means the workflow will move to the next logical step if one of the two branches is complete. If **Necessary Branches** is set to **2**, then the

workflow will wait for completion of both branches before moving after the logical steps of a **Fork**.

Now we'll discuss when and how you can build a subworkflow. The example we're discussing is related to financial accounting document park and post. SAP has provided three level approval scenarios for document park and post. The main framework workflow is delivered with `WS10000051`. It's calling different subworkflows depending on the level of approval needed. There are three subworkflows created: `WS10000052` (single step approval), `WS10000053` (two step approval), and `WS10000054` (three step approval). In our scenario, main workflow `WS10000051` is calling the subworkflow dynamically.

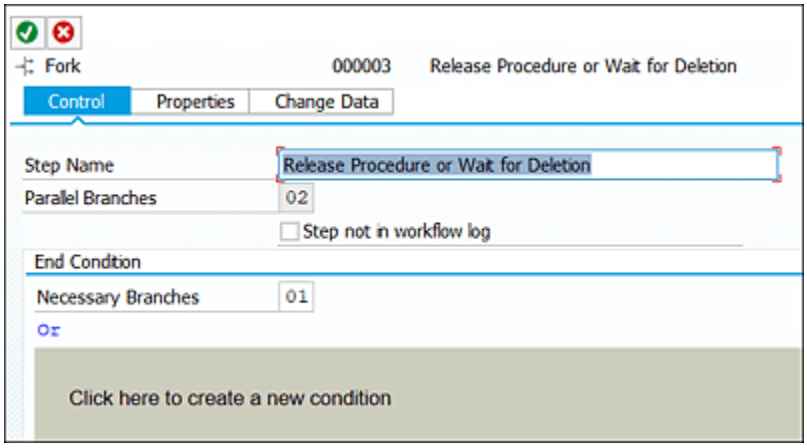


Figure 4.11 Workflow Step Fork Details

Figure 4.12 shows the background task that is determining the subworkflow to be called and subsequent call of the subworkflow. The background step, **Determine Amount Release Subworkflow**, determines the subworkflow based on the approval level needed depending on the configuration. The subsequent step, **Call Amount Release**

Subworkflow, calls the subworkflow that is determined in the earlier background step.

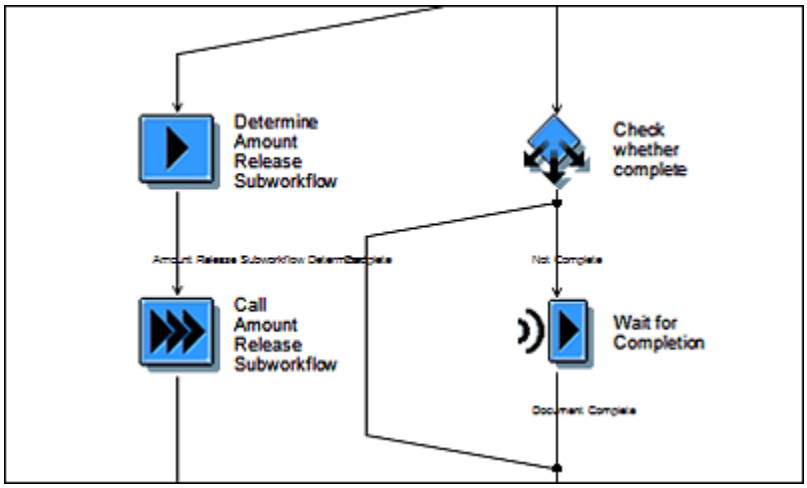


Figure 4.12 Use of Subworkflow and Background Task

Background task `TS00007913` is created to determine the subworkflow, and it passes back the subworkflow number to the main workflow. Figure 4.13 shows the background **Task TS00007913** details and container element binding. The method assigned to the background task determines which subworkflow to be called. This subworkflow number is passed through container element `&ACTIVITY_000043_TASK&`.

The determined subworkflow number is passed in the subsequent **Call Amount Release Subworkflow** whose technical details are shown in Figure 4.14. This variable containing the subworkflow number is passed into the **Expression for Task** field. This mapping will dynamically call the subworkflow. The subworkflow interface should be compatible with the workflow in the **Interfaces Compatible With** field so that proper container mapping can happen. If the **Expression for Task** field is empty, then the task entered in the **Control** tab is considered for execution.

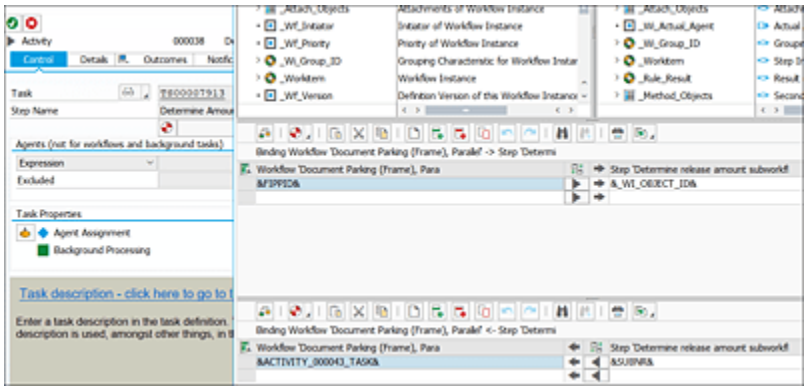


Figure 4.13 Workflow Task Control Information and Container Element Binding

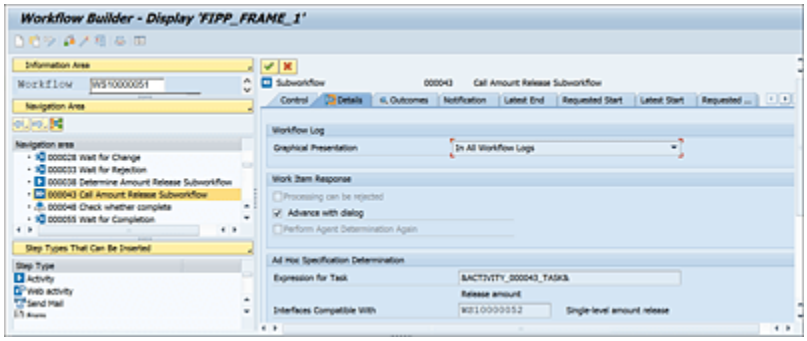


Figure 4.14 Calling the Subworkflow Dynamically

4.3 Adding Tasks

In this section, we'll discuss the purpose of the different tabs in a step and how a step can be inserted. You can drag and drop any step from the **Step Types That Can Be Inserted** list on the left side onto the outcome of the preceding step. Alternatively, you can select the outcome node and double-click on the step type that you want to add. It will open up the details screen of the step type where you'll enter all the necessary information and save it.

4.3.1 Adding an Activity Step

A step of type **Activity** is the main step for any workflow. We'll discuss the key attributes that should be maintained during an **Activity** step creation in the following lists of tabs:

- **Control**
  Figure 4.15 shows the attributes of the **Activity** step's **Control** tab. The **Control** tab contains the attributes related to execution of the task.

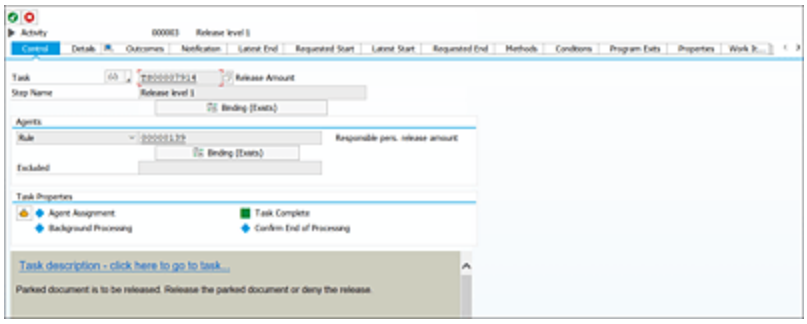


Figure 4.15 Activity Step Details: Control Tab

Here are the key attributes you have to maintain in the **Control** tab of **Activity** step:

- **Task**: Enter the workflow task or subworkflow that will be executed when this step is processed.
- **Step Name**: Enter any meaningful description of the step.
- **Binding (Exists)**: This is used to map any workflow container element to the task container element and vice versa. A task won't be able to access any workflow container variable if it isn't passed through container binding.
- **Agents**: This will determine who gets the workflow item. This is needed if the task is a foreground task. For a background task or subworkflow, this information isn't needed. There are several ways to determine agents. Rules are the most dynamic way. Figure 4.16 shows the possible type of agent determination. See Chapter 6 for more details on agent determination.
- **Task Properties**: This property comes from the **Task** property.
- **Agent Assignment**: You have to add the list of possible agents here. Click on the task button next to **Agent Assignment**. All possible agents should be updated here. Otherwise, the user won't receive the work item even if the agent determination rule successfully identifies the agent.
- **Task Description**: This comes from the task description maintained. This description appears as the work item description in the inbox.

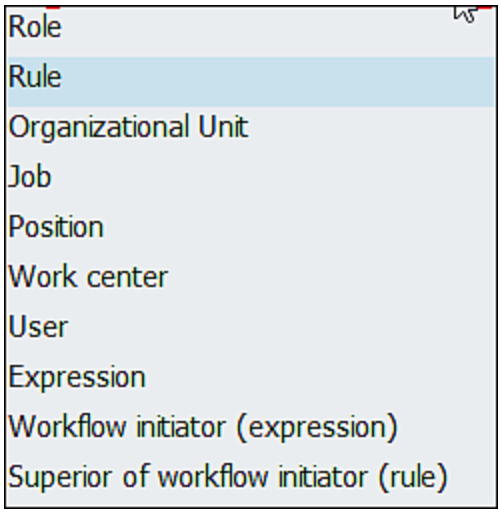


Figure 4.16 Different Agent Determination Types

- **Details**
  Figure 4.17 shows the **Details** tab attributes. The **Ad Hoc Specification Determination** section is used for dynamically calling the task. Other sections are populated by default and normally don't need any change.

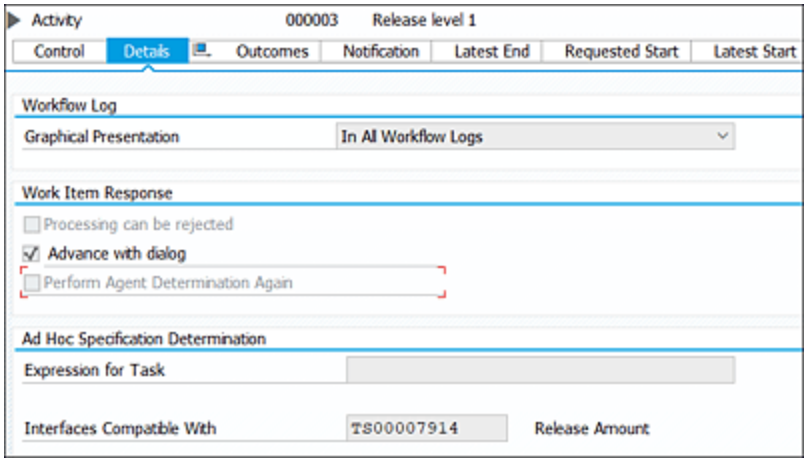


Figure 4.17 Activity Step Attributes: Details Tab

Ad hoc specification is needed when a task or subworkflow needs to be called dynamically. If this is empty, then the task/subworkflow entered in the **Control** tab is called for execution. The dynamic task/subworkflow

should be compatible with the task mentioned from the container and business object perspective.

- **Outcomes**
  This tab provides all possible outcomes of the task. The number of branches from this task is determined from the number of outcomes. If you want to use the **Process Control** step, you should use the processing obsolete outcome. Normally, the processing outcome for a task comes from being terminated or an internal event triggered from the method.
- **Notification**
  You can send any email notification when the task is completed. The recipient of the notification can be determined using any workflow agent determination procedure.
- **Requested Start**
  This tab is used to determine when a work item will be visible to the end user. If this tab isn’t maintained, then the work item is available immediately after it’s generated. For a background task, the execution of this task starts after the requested start date.
- **Latest Start**
  This tab indicates when the users should start processing their work item. Users should reserve or start processing the work item, so this is used to send warning notification if they don’t start processing, that is, opening the work item. The latest start date will be later than the requested start date but before the requested end and latest end dates.

- **Requested End**
  This is used to determine when the user is expected to complete the work item and to send a warning notification if the user doesn’t complete the work item. Requested end date should be earlier than latest end date but later than the requested start and latest start dates.
- **Latest End**
  This is used to determine when the user must complete the work item. This is normally used for the escalation process where the work item may be forwarded to the user’s manager for further processing.
  All these time-related tabs are related to deadline monitoring. Deadline monitoring can be done with warning notifications. Alternatively, it can be modeled on an outcome, which means a separate branch of processing is created for the task.
- **Conditions**
  This tab is used for exception cases. Additional conditions can be added to execute or complete a work item. If the create work item condition isn’t met, then the work item can’t be executed by the user.
- **Program Exit**
  You can execute additional methods during different phases of the work item. The class should be similar to interface `IF_SWF_IFS_WORKITEM_EXIT`. The instance of the method will contain the event name, for example, before creation, after creation, after execution, and so on. (Check the domain of the `SWW_EVTTYP` field). Custom code can be written based on the event.

4.3.2 Adding a Send Email Step

You use task type **Send Mail** to send email to users. Another option may be to create a background task and use the standard function module to send email. But the **Send Mail** task is easy to build and can be traced in the workflow log, so it's the recommended development for the email send activity.

To create this task, drag and drop the **Send Mail** activity on an outcome where you want to send email to an email address or SAP inbox. You can also select the outcome and double-click on **Send Mail** from the **Step Types That Can Be Inserted** list to open the screen shown in Figure 4.18.

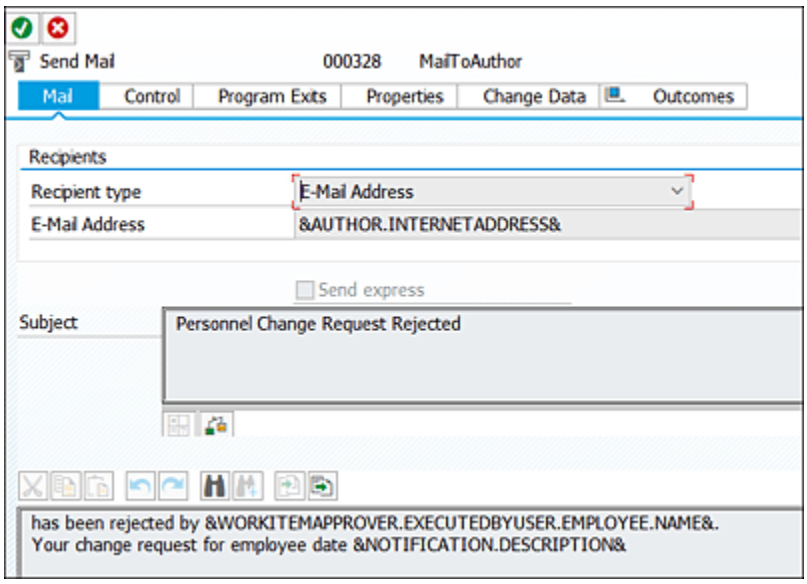


Figure 4.18 Send Email Step Type

You have to populate the following details in the **Mail** tab:

- **Recipients**
  Three types of recipients are possible: external email address, organizational object, and workflow initiator. For organization object, any job, position, organizational unit,

work center, or user can be maintained here. It's also possible to determine any kind of organizational unit dynamically in an earlier background step and pass that workflow container element in this expression. For workflow initiator, the `&_WF_INITIATOR&` workflow container will be passed as the variable. For an email address, it can be derived in an earlier background step and passed in a container element as an expression.

- **Subject**
  This will contain the subject of the email. You can add any variable from the workflow container.
- **Mail body**
  The box below **Subject** contains the mail body. You can enter any variable from the workflow container.

4.3.3 Adding a User Decision Step

We use the **User Decision** step when a work item is sent to the user to make a decision (e.g., **Approve**/**Reject**). This is a simpler way to take user action. The business object can be linked to the work item to provide the business object link so that the user can view the transaction in a separate window in display mode. If the user needs to update the business transaction, then the user decision may not be useful. A foreground activity task with transaction screen needs to be developed.

To create a user decision, drag and drop the **User Decision** task type on an outcome where you want a user to decide on the outcome to proceed to the next step in the workflow. Figure 4.19 shows a sample of a **User Decision** step.

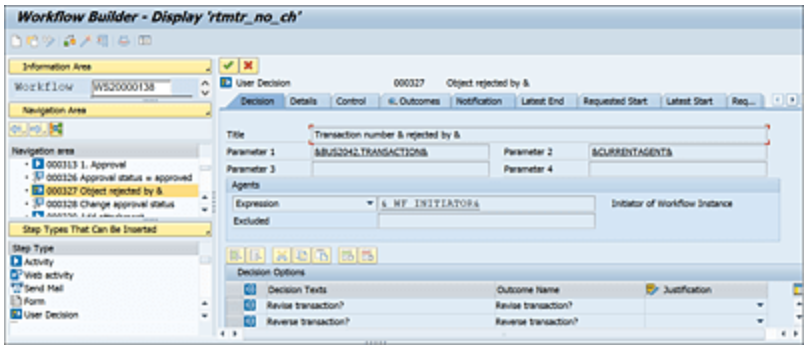


Figure 4.19 User Decision Step Type

The tabs of **User Decision** are similar to the **Activity** step with the following differences:

- **Title**
  This will appear in the work item text. You can enter four variables (**Parameter 1**, **2**, **3**, and **4** fields) from the workflow container. Consider the length of the work item text when you're adding any variable. The text may get truncated if the total text length goes beyond the allowed title length after adding variable texts. To add variable text, add "&" in the title text.
- **Agents**
  Standard workflow agent determination framework is available here. It's the same as for the **Activity** step.
- **Decision Options**
  All the decision options can be entered here. This will create the same number of outcome processing branches.
- **Control** tab
  Create a synchronous task using object type `DECISION` and method `PROCESS`. You can refer to standard task `TS00008267`. A custom task is needed to map container elements that can be used within the decision step. The task description is changed according to the business need.

4.4 Containers and Bindings

In this section, we'll discuss the data flow between different components of the workflow. Containers are used to store data in different sizes and shapes. It can store field values, multiline lists of field values, references to an object, and multiline references to an object. The data values in the container are available in the workflow log. It also helps to troubleshoot any workflow execution problems. Binding is the mechanism to flow the container values between different components of workflow (e.g., event, task, rule, etc.). Binding is always performed with reference to a container element. The binding definition is executed at runtime and is used for transferring data from one container to another container, along with assigning values to a container. We'll discuss more details on different kinds of containers and how binding occurs between them. Finally, we'll discuss custom transformations in binding.

4.4.1 Types of Containers

In general, each building block of a workflow has a container. Following are the main blocks where containers are used:

- **Workflow container for each workflow and subworkflow**
  The workflow container is used to pass the information from one building block to another of the workflow. Workflow container elements can be filled during the start

of the workflow (through the starting event of the direct workflow container binding) only through the IMPORT workflow container element. Normally, the business object instance is passed to the workflow from the triggering event. The purpose of the workflow container can be considered similar to global variables. If any field value is needed in different steps of the workflow, it's defined within the workflow container.

- **Task container**
  Task containers bind the value between the method and the workflow. IMPORT container elements of task containers are filled from the workflow container, and EXPORT container elements are transferred back.
- **Method container**
  Method containers take the field value from the task container. IMPORT container elements of the task container are filled from the task container, and EXPORT container elements are transferred back.
- **Rule container**
  The IMPORT container element is filled from the workflow container. For the default rule in the task, the IMPORT rule container is filled from the task container.
- **Event container**
  Each event (both triggering and terminating event) has a container. Events have EXPORT containers only, which means events can only pass the value to the workflow or task through this container.

Figure 4.20 shows how data flows between different container elements within the workflow.

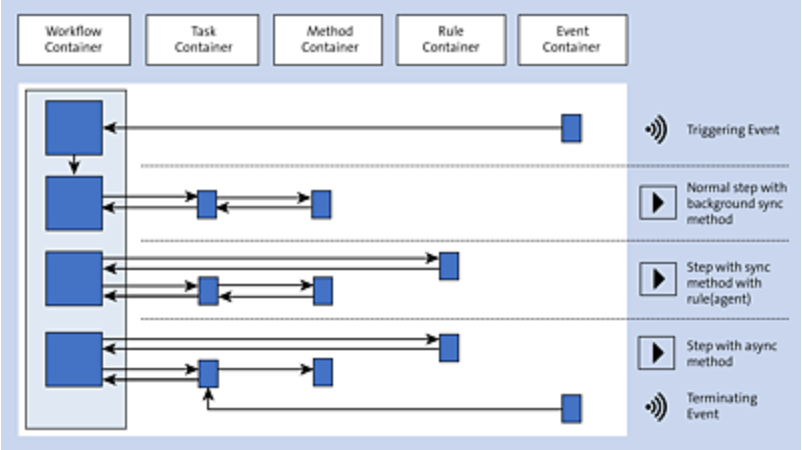


Figure 4.20 Workflow Container and Flow of Data Using Container Binding

Let’s go through this flow in the context of an invoice approval workflow. A shared service agent creates an invoice and parks it. The first step of the workflow is a background task to determine the number of approvals needed. The second step is the approval step. The agent for this step is determined using a rule, and this step is a synchronous approval step. The next step is the second level approval. This task is an asynchronous approval step. The agent is determined by a rule. Here’s the process:

1. When the shared service agent saves the document as a park document, an event is triggered from the application. In general, two parameters are passed from the event to the workflow: `&_EVT_CREATOR&` and business object `&_EVT_OBJECT&`. The `&_EVT_CREATOR&` parameter is mapped to `&_WF_INITIATOR&` and `&_EVT_OBJECT&` to a workflow container element reference to the business object. If you’re creating any custom workflow, then you have to create a container element with reference to the same business object. This element will be mapped to `&_EVT_OBJECT&`. Because this container element is created with reference to a business object, you’ll be able to

access all attributes of the business object for any application operation.

2. After the workflow is triggered and the event passes the container elements, the workflow engine will progress to the next step, which is a standard background task to calculate the number of approvers needed. You have to pass the information to the task and subsequently to the method that is needed to execute the business application logic written within the method. Rather than passing all workflow container elements, pass only the element that is needed to execute the method. When the method execution is complete, it passes back the information required for subsequent steps. For example, it will pass the number of approval steps needed for the invoice depending on the invoice amount. Normally, the task container elements are used for task description, notification variables, and other task-level actions. Method container elements are needed to execute the methods with the task.
3. The next step is to call a step with an agent in a synchronization task. The workflow engine will first call the agent determination rule. It will pass the container element information per binding definition to the rule engine. The rule determines the agents for the task and sends that information back.
4. The workflow engine then creates the task and sends it to the agent for execution. The application information is bound between the workflow container and the task container. When the user executes the task, the corresponding method gets executed, and the

application information flows to the method at runtime. If any information needs to be passed back to the workflow for subsequent activity, it’s done by the container binding between task and workflow.

5. Similarly, if the task is linked to an asynchronization method, then there is no information flow back from method container to task container. The task waits for a terminating event to proceed to the next step. The terminating event transfers the application data to the task, and then the task transfers the application data to the workflow container.

Workflow containers hold all the data used during execution. The primary purpose of workflow containers is to pass the business application data from one step to another step via the workflow container and control the workflow operation. They are also used for calculations, mathematical operations, and variable text with the workflow builder editor, in addition to the following:

- The task container is used to provide the dynamic text of the work item and pass the business application information to/from the method.
- The method container is used to execute the business application logic written in the method.
- The rule container is used to determine the agents for the work item.
- The event container passes the application information to the workflow.
- The event container is also used to troubleshoot the method executed in the task. Because the workflow is

executed in the background and there is no control to debug in production environments, some of the parameters used in the container will help in troubleshooting the issue.

Note

Try to avoid declaring too many container elements via `IMPORT` and `EXPORT`. Only export the container element that is needed in the next step and import if any container element is changed. Don’t import the container element reference to the business object. The container element reference to the business object will have the latest information corresponding to the primary key of that object.

You can create container elements for each type by going to its create/change transaction. For example, a workflow container can be created from a workflow change transaction. Go to Transaction PFTC. Enter **Task Type** as “Workflow Task”, and enter the workflow template number in the **Task** field. Then, click on the **Change** button, and go to the **Container** tab. Figure 4.21 shows the attributes of a container element.

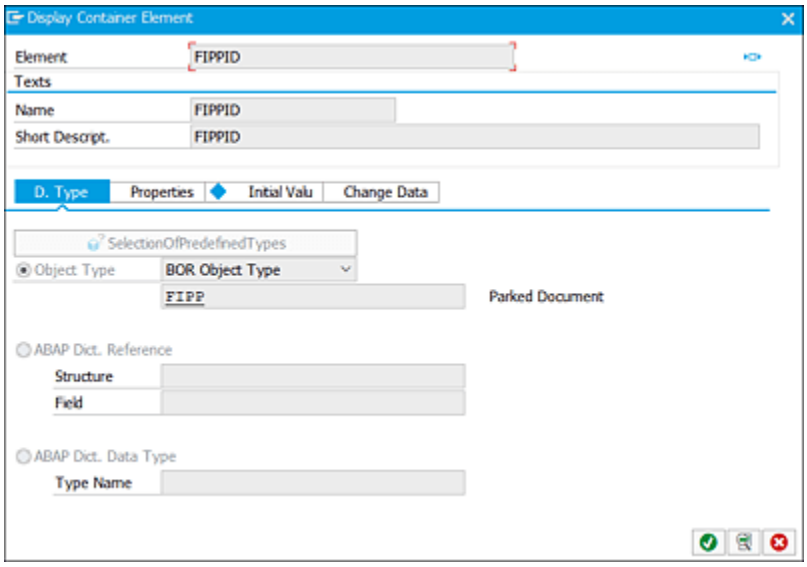


Figure 4.21 Workflow Container

As you can see, the container element can be referenced to a business object, class, or any kind of ABAP Data Dictionary objects. Similarly, you can create container elements for a task. Go to Transaction PFTC. Enter **Task Type** as “Standard Task” and **Task** as the task number. Then, click on the **Change** button, and go to the **Container** tab. You can create/change container elements for the task here. You can also define the container element for the method and event in the business object maintenance transaction. It’s defined as a parameter of that particular method or event.

To create container element for a method or event, go to Transaction SWO1, which is used to create/change/display any business object. In Transaction SWO1, enter the business object name in the **Object type** field. Click on the **Change** button to display the methods and events created for this business object. To create a container element for the method/event, select the method/event, and click on the **Parameter** button. You can create/change/display any container element for the method/event. Figure 4.22 shows

the container element of method **EDITHEADER** of the standard business object **FIPP** as reference.

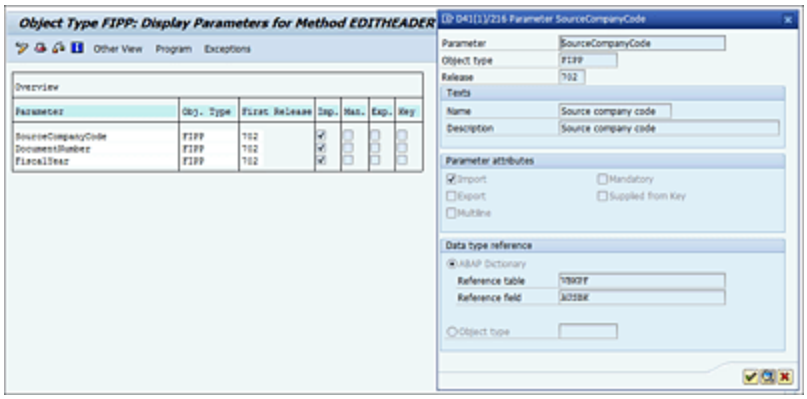


Figure 4.22 Container Element of a Method

We've created the container element for the method. But how is the container element read within a method? The container element can be accessed in the program using macros. You have to go to the method program editor and write macros to access the container element. Refer to Chapter 3 to get more details on how to write code in a method. Use macro `SWC_GET_ELEMENT` to retrieve the values of specific container elements in the rule function module or business object method. As mentioned, you have to write this macro in the method program. When methods are executed as part of workflow execution, this macro will get executed and will read the container element data content. This will be used in subsequent executions of the method code.

Here is sample code to retrieve the sales order type from a container in a rule function module:

```
swc_get_element ac_container 'ZWCAUART' lv_auart. "Sales order type
```

AC_CONTAINER is the IMPORT table of the rule function module. ZWCAUART is the name of the container element. lv_auart is the local variable where we're storing the container element value with the function module for further business application processing.

Similarly, macro SWC_SET_ELEMENT is used for passing values in the EXPORT container element.

The workflow-specific code that is executed as part of task execution can be written in a business object method or in a class method. For a class method, the container element is defined as IMPORT/EXPORT parameters of the class. You don't have to write any macros to access the container element value. It can be accessed similar to any IMPORT/EXPORT parameter you use in any classical ABAP code using a class.

4.4.2 Binding Definition and Binding Operators

Binding is the mechanism to assign the container variable value from one step to another. For example, when there is a need to pass the container variable from the workflow to a task, then the container element of the workflow container and task container should be mapped in the binding editor.

Figure 4.23 shows the container binding editor where the container element can be mapped between source and target container.

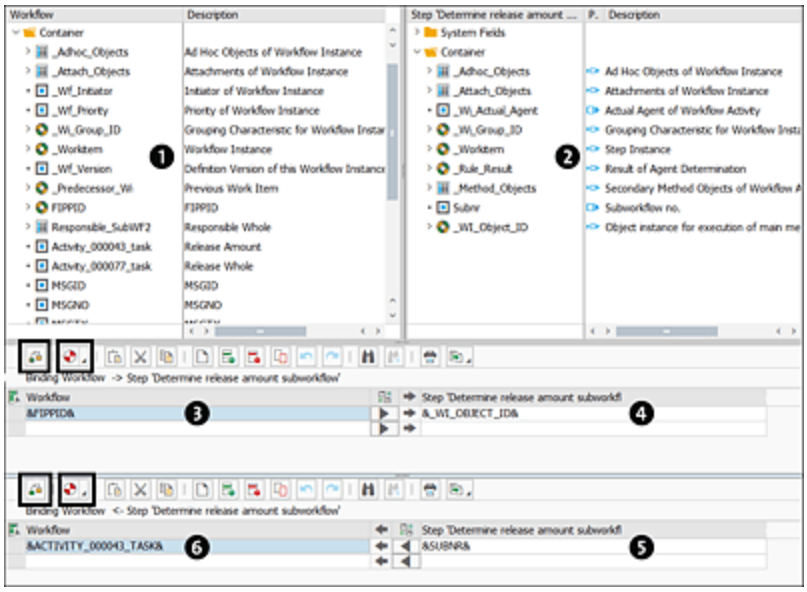


Figure 4.23 Container Binding Editor

Figure 4.23 shows the binding between the workflow container and step/task container. The binding editor between event and workflow or between task and method is similar.

The top-left side of the editor ❶ shows the source container (in this case, it's a workflow container), and the top-right side ❷ shows the target container elements. There are different icons for single value containers, table/list containers, and reference object containers. You can drag an element from the left container ❶ and drop into the target container ❷. It will automatically create a binding for the `EXPORT` side of it. In this example, source container element **&FIPPID&** is dragged into target container element **&_WI_OBJECT_ID&** from work area 1 to work area 2. It automatically creates the binding as shown in work area 3 and 4. When you drag an element from the source

container ❶ to the target container ❷, it will create the binding from the workflow to the task.

You can also drag and drop a container element from ❶ to ❸ and from ❷ to ❹. This will also create the binding for workflow to task.

Similarly, you can drag an element from the right container and drop into a left container element. This will create the `IMPORT` side binding. You can also drag and drop the container elements into the corresponding `EXPORT` and `IMPORT` side of the binding. When you drag an element from the source container ❷ to target container ❶, it will create the binding from task to workflow.

You can also drag and drop a container element from ❶ to ❻ and from ❷ to ❺. This will also create the binding for task to workflow. Areas ❸ and ❹ show the container element binding done for workflow to task, and areas ❺ and ❻ show the container element binding done for task to workflow.

You can click the  button to auto-generate the binding, but it’s not advisable to use this feature. This will try to perform binding based on the container element data type. This will unnecessarily create elements and bindings that aren’t required for your workflow development, so you should always do the container element binding manually.

You should always do a syntax check of the binding after completion. Click the syntax check button  to check if the binding is syntactically correct. This will do a compatibility mapping check of the bonded container elements.

4.4.3 Custom Transformations in Binding

The containers are bonded as a value assignment by default, but there are other complex operations that can be done during value assignment. In general, this kind of complex operation during binding isn't used. Developers prefer ABAP code to write application logic and any kind of filtration with the method.

However, these complex operations will help reduce ABAP coding. You can start with simple container value assignment and explore other options depending on the requirement. Click on  in the binding editor to get more container binding options. Container element binding has two main options:

- **Expression --> Expression**
  This is used by default to map each container element from source to target (see Figure 4.24).

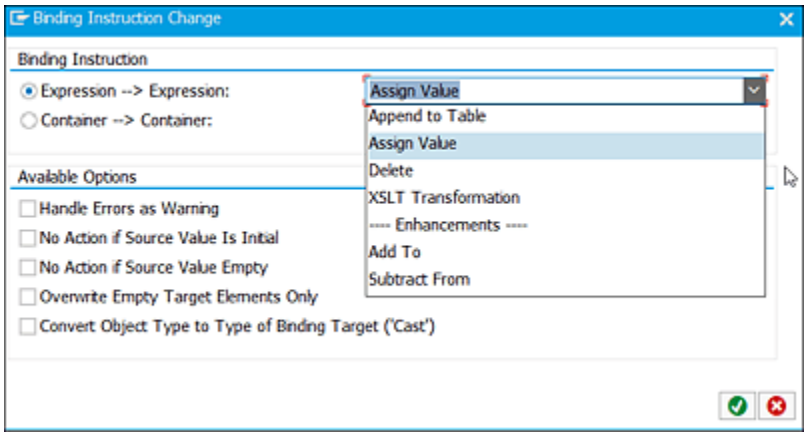


Figure 4.24 Binding Expression Operation

- **Container --> Container**
  This enables you to handle the entire container (see Figure 4.25).

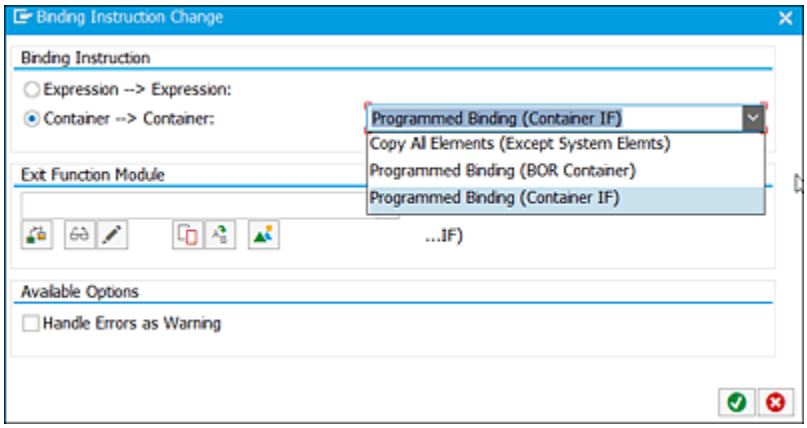


Figure 4.25 Binding Container Operation

When mapping a multiline container element, it’s possible to access a particular row or column of that multiple table container using expressions. Standard table index operations such as `&TABLE.[INDEX].ColumnName&` are supported in the binding editor.

4.5 Multiline Elements and Dynamic Parallel Processing

Some business requirements call for taking a single step in parallel. This can be a background step or a step that will go for approval. For example, a purchase order needs to be sent for approval for each line item. Each line item will have a different approver. Only once all the approvers approve is the entire purchase orders approved. Similar kind of business requirement may arise for any business objects with line items. You can use the multiline elements dynamic parallel processing technique to meet this business requirement.

In this scenario, the business object is a purchase order, and there will be only one instance of this purchase order. You have to create a multiline workflow container element that will contain purchase order line items. Figure 4.26 shows that the **Multiline** property should be checked.

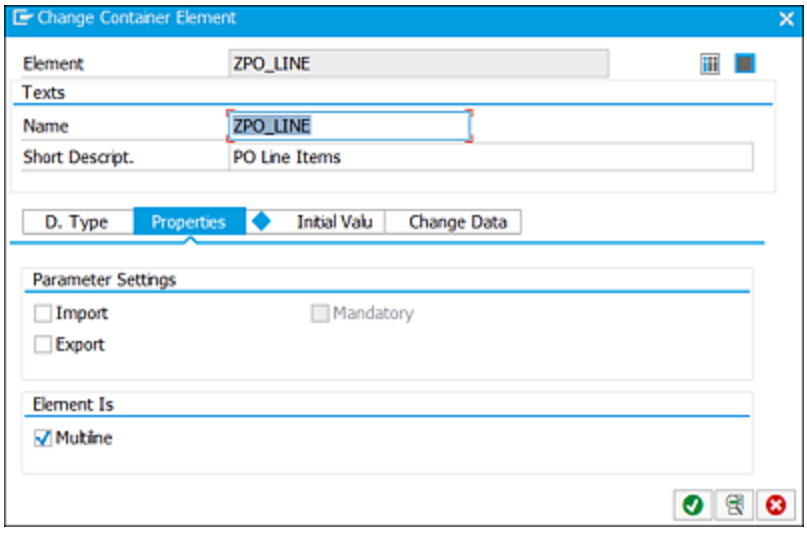


Figure 4.26 Multiline Container Element

You can create a background task or map with business object attributes to populate the line items of a business object. Once this container element is populated, you trigger a task for each entry in this container by mapping this multiline container element in the **Miscellaneous** tab of the task, as shown in Figure 4.27. Enter the multiline container element in the **Multiline Element** field.

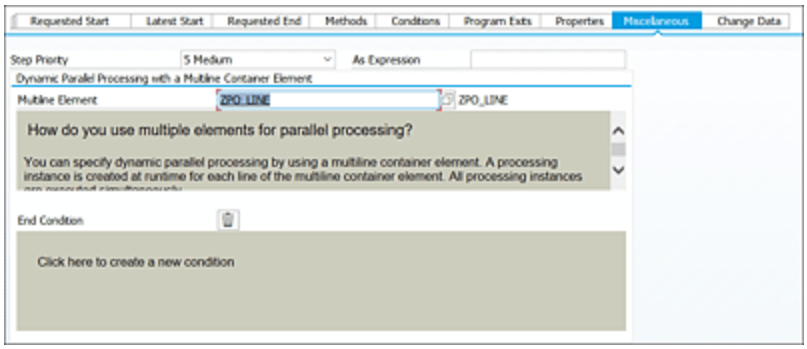


Figure 4.27 Dynamic Parallel Processing with a Multiline Container Element

With these settings, the system will generate one instance of the task for each entry in this container element. There are a few questions to keep in mind:

- **Which task/method will be executed for each instance of the generated work item?**
  The task maintained in the **Control** tab of this activity will be executed for each instance of this step. So, this method of parallel execution is applicable when multiple instances of the same business object execution is needed.
- **What container element binding system will be considered during execution?**
  The container element binding entered in the **Control** tab will be executed for each instance. That means all

instances of the step will have the same data passed to the method.

- **How can we pass the current value of the execution (purchase order line-item number, in this case) during runtime?**
  There is one container element that get created automatically—`_Wf_ParForEach_Index&`—which store the current index for dynamic parallel processing. Use this container element for further processing.

You can maintain an end condition for the dynamic parallel processing. This is checked when one of the parallel processing branches is complete.

There are different ways to design parallel processing of workflows. We've talked about the parallel processing of container elements here. This is normally done when the same business object is used for parallel processing and the total number of parallel processing items is unknown. A single task or a subworkflow can be attached here. For a simple single activity, a task is attached. If there are complex requirements of processing and user assignment, a subworkflow can be assigned.

Another option to achieve the same result is calling a subworkflow with asynchronous task with a loop. If there is a need for parallel processing with different steps for each branch, but you're aware of the number of branches needed, then the fork operation needs to be adopted.

4.6 Deadline Definition

A deadline in workflows is mainly used for escalation management. It handles the scenario of what will happen if a work item isn’t actioned within a stipulated time. In general, three subsequent actions can be taken if the work item isn’t processed in time:

- The work item can be processed automatically. If the approver only needs to approve/reject the work item, this can be an option depending on business needs. But if the approver needs to update any information in the approval form (e.g., update the general ledger account number of an invoice), then it’s not a practical option. Sometimes, businesses allow auto-approval for leave of absence and other HR-related workflows or for very-low-amount, high-volume business transactions.
- An email can be sent to the supervisor as an escalation email.
- The work item can be sent to the current approver’s manager to take action. When you’re sending the work item to the next-level manager, you can keep both the current approver and his manager as the approver for the work item. You also can only send it to the next-level manager. It’s advisable to keep the current approver as well in the escalated work item. Normally, higher-level managers will take more time to process as they don’t have all the information regarding the work item. So, keeping both as approvers will improve the processing time.

We’ll be discussing in detail the different kinds of deadline monitoring options in the workflow. There are mainly two sections of deadline information you have to populate to activate a deadline: condition to reach the deadline and action to take when it’s reached.

Figure 4.28 shows how you can set up the date/time when a deadline will be reached.

Figure 4.28 Deadline Time Calculation

In Figure 4.28, the deadline will be reached after 60 minutes of the work item creation. Table 4.2 lists the ways you can provide the reference date/time (**Refer.date/time** field).

Field	Description
Work Item Creation	Reference date is taken as the work item creation date and time. Then, additional time is added per the fixed number of minutes, days, and so on entered. This is used in the simple cases where the deadline isn’t calculated dynamically.
Workflow Creation	Reference date is taken as the workflow creation date and time. Then, additional time is added per the fixed number of minutes, days, and so on entered. This is normally not used in the business context.

Field	Description
Expression	All the fields become editable. You can use any container element that will help you dynamically set the deadline. For example, in a two-step approval process, you want to set a deadline for level 1 approval after four days and for level 2 after six days. You've used the same step within a loop. In that case, you can use a container element to set the deadline date and time and then pass that container element in the expression. Most of the complex business requirements will need to use this method to implement deadlines. It's also easy to change using some custom configuration tables. You don't have to change the workflow all the time to change the deadline.
No Deadline Monitoring Active	A deadline won't be triggered.

Table 4.2 Different Options for Calculating Deadlines from a Reference Date and Time

There are two kinds of outcomes that can be set up when a deadline is reached. You can send a notification message to the escalated agent's SAP inbox, or you can create another processing branch where you branch out a separate set of steps.

Figure 4.29 shows the settings to send a message to SAP inbox when the deadline is reached. To determine the agent who will receive the SAP inbox message, you can choose the options similar to the options available for the work item agent determination (i.e., **Rule**, **Role**, **Job**, **Expression**, etc.). You have to maintain the notification text in the **Description** tab of the standard task assigned for this step.

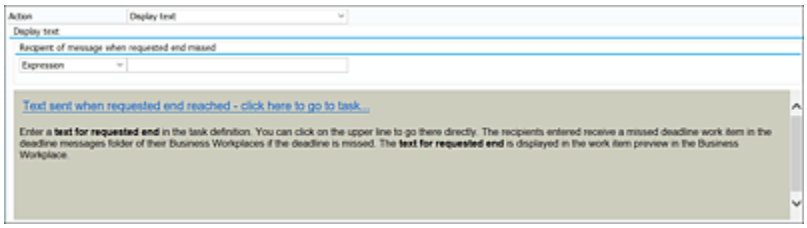


Figure 4.29 Action Notification Trigger When the Work Item Deadline Is Reached

Figure 4.30 shows the settings for a separate branch (referred to as *modeled*) when the deadline is reached. When you select the action as **Modeled**, a new branch will be created after the step. You can add more steps in that branch to execute the next steps.

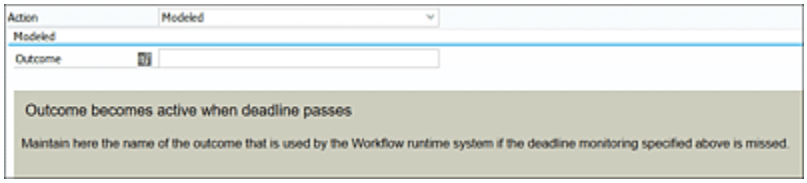


Figure 4.30 Action Event Trigger When the Work Item Deadline Is Reached

There are four kinds of deadlines that can be implemented for a work item.

- **REQUESTED START**
  If the requested start deadline is active for a dialog work item, then the work item will appear in the user inbox when the deadline is reached. For a background task, the

work item will start execution after the deadline is reached. The requested start deadline determines when the work item generation/execution begins. It doesn’t have the option to configure any text or modeled action.

- **REQUESTED END**
  This deadline checks if the work item is completed within the deadline date and time. If not, it will trigger the deadline action. You can send notification messages or modeled outcome when the `REQUESTED END` deadline is reached. This deadline is normally used to send reminder notifications to the user to complete the work item. This is normally not used for modeled outcome.
- **LATEST START**
  This deadline is triggered if the user doesn’t open the work item within this deadline date and time. This is based on the start of your work item execution and isn’t the completion of the work item. If a user has just opened the work item, but didn’t complete the task within the deadline date and time, then this deadline won’t be triggered. You can send notification messages or modeled outcome when the `LATEST START` deadline is reached. This deadline is normally used to send reminder notifications to the user to complete the work item but is normally not used for modeled outcome.
- **LATEST END**
  This deadline checks if the work item is completed within the deadline date and time. If not, you can send notifications or modeled outcome when the deadline is reached.

If any work item misses a deadline, then that work item will be highlighted in color in the user inbox. That work item will also appear in the **Overdue Entry** subfolder in the user inbox.

You can achieve similar outcomes from `REQUESTED END` and `LATEST END` deadlines, but these were given to serve different purposes. If you need to use all the `DEADLINE` options, the sequence of setting up the time will be `REQUESTED START`, `LATEST START`, `REQUESTED END`, and `LATEST END`.

`REQUESTED START` is generally used if there is some manual activity to be done before the work item creation, and there is no way to track that in the system. In that scenario, some predefined time can be added before the work item is created. However, the use case of `REQUESTED START` is very limited.

Then, the `LATEST START` deadline is planned. It's generally used to send reminder notification messages that the work item needs to be completed within the specific date and time. Because this deadline won't be triggered if the user opens the work item but doesn't complete it, in general, it's not used for modeled operations.

Next, the `REQUESTED END` deadline is planned. It's generally used to remind the user to complete the work item. You can also send the notification as an escalation to the next level along with sending the notification to the current processor. If you've planned to send the notification to both, create a container element with both IDs, and use it in the notification expression.

Finally, the LATEST END deadline is planned. It's generally used to escalate the work item to the next-level manager/processor along with the current processor. There is no change in the processing status of the work item when a deadline is triggered. The LATEST END deadline won't send the due work item to the next-level manager automatically. So, if you want to send a work item to the next level as an escalation (because the work item isn't completed within the deadline), you have to use modeled action. With modeled action, you have to cancel the current work item and regenerate a new work item. You can set new agents when the new work item is generated.

There is one more scenario to send repeated escalation notifications until the work item is completed. You have to use the modeled action in this scenario as well. One set of deadlines is triggered once for a generated work item. For example, let's say a REQUESTED END deadline is triggered, but the user still didn't complete the work item. The system won't trigger REQUESTED END again for that work item. If that work item is canceled and regenerated again, then the clock is reset and the REQUESTED END deadline can be triggered again for the newly generated work item, but it will reset the clock for the user. It will be difficult to get with standard SAP reports how long the work item is pending with the user.

It's recommended you do a modeled outcome in the LATEST END deadline and repeatedly send the notifications so that the escalation mechanism continues to work. Let's take a scenario where you want to send escalated notifications three times with one day intervals to the manager when the LATEST END deadline is reached. If the work item isn't completed in three days after the deadline, then you want

to send the work item to the current processor manager. You have to follow these steps to achieve this functionality:

1. Set up the modeled outcome in the **Latest End** tab of the step, as shown in Figure 4.31.

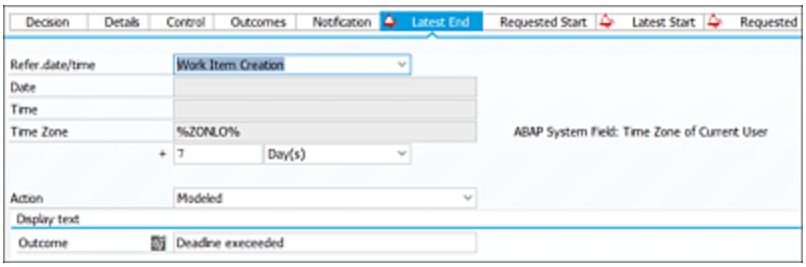


Figure 4.31 Set Up the Modeled Outcome for the LATEST END Deadline

2. Go to the **Outcomes** tab, and activate **Processing obsolete** (see Figure 4.32).

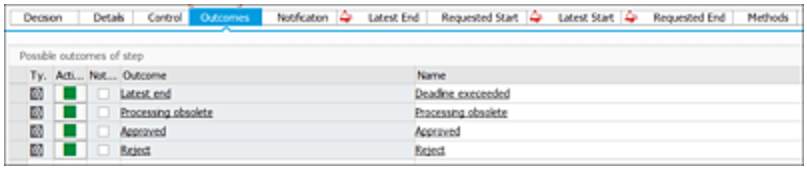


Figure 4.32 Activate the Processing Obsolete Outcome

This will create two additional branches (**Deadline exceeded** and **Processing obsolete**) of the outcome from the control step.

You have to create a `LOOP` step in the deadline exceeded and send mail using the **Send Mail** step. Create a container variable to keep the counter set at three times to send the reminder notification. Use the `REQUESTED START` deadline to send the notification the next day. This is equivalent to waiting for one day before sending the next reminder.

If the user doesn’t take any action after three reminders, you want to send the work item to the next level. To achieve

this, you have to set a PROCESS CONTROL STEP after LOOP with control function **Set Work Item to Obsolete**. This will trigger the processing obsolete outcome and will cancel the current work item. You need a subsequent process to regenerate the same earlier step with a new set of agents.

If you're planning to send the work item to a higher authority as an escalation, you have to consider a fallback option if no one acts on the work item. Sending work items to a higher authority as an escalation makes the workflow complex and maintenance a challenge. So, talk to the business to come up with an optimized way to implement deadlines.

It's also important to have correct and meaningful messages in the deadline. Sometimes, a developer activates deadlines as a general practice without a proper notification text. This confuses the end user, and a high number of production incidents get created after go live.

In real business scenarios, deadline duration isn't calculated based on a fixed seven-day week. It's based on a user business calendar that is known as the factory calendar in SAP. In a global organization, a country-specific factory calendar will exist. So, consider the user-specific factory calendar when you're calculating any deadline duration. It's recommended to have a generic task created to calculate deadline duration for any workflow work item:

1. Decide how you'll determine the factory calendar for the user. Factory calendars can be determined from different sources of configuration, such as from a plant, a user, or any custom table based on an organizational unit. This is normally done as a workflow strategy across all

workflows, not just for individual workflow developments.

2. Create a business object method to calculate the deadline. Because there are four deadlines possible, `REQUESTED START`, `LATEST START`, `REQUESTED END`, and `LATEST END`, keep the parameters to calculate all four. Then, use whatever is needed for a particular step. The business object will be generic and can be used in all workflows in the project. Create the business object with the following method parameters to calculate the deadline date and time:

   - `OFF_RS` (offset for `REQUESTED START`): `IMPORT` parameter of number type data definition
   - `OFF_LS` (offset for `LATEST START`): `IMPORT` parameter of number type data definition
   - `OFF_RE` (offset for `REQUESTED END`): `IMPORT` parameter of number type data definition
   - `OFF_LE` (offset for `LATEST END`): `IMPORT` parameter of number type data definition

   Normally the deadlines are determined based on days or hours. So, you can assume the duration unit as hours to simplify. If you see variations in the duration unit, then you can pass the duration unit as a parameter, as follows:

   - `ORG_UNIT` (organizational unit or any other parameter to determine the factory calendar): Determine this during the design phase of the project and implement across all workflows.

- DL_DT_RS (deadline date for REQUESTED START): EXPORT parameter of date type data definition.
- DL_TM_RS (deadline time for REQUESTED START): EXPORT parameter of time type data definition.
- DL_DT_LS (deadline date for LATEST START): EXPORT parameter of date type data definition.
- DL_TM_LS (deadline time for LATEST START): EXPORT parameter of time type data definition.
- DL_DT_RE (deadline date for REQUESTED END): EXPORT parameter of date type data definition.
- DL_TM_RE (deadline time for REQUESTED END): EXPORT parameter of time type data definition.
- DL_DT_LE (deadline date for LATEST END): EXPORT parameter of date type data definition.
- DL_TM_LE (deadline time for LATEST END): EXPORT parameter of time type data definition.

3. Implement the method to calculate the deadline date and time. This building blocks of the method will be as follows:
   - Find the factory calendar to be used using the organization unit or any other custom logic. The input needed to determine this will be passed as an IMPORT parameter in the workflow.
   - Call function module END_TIME_DETERMINE for each deadline type to determine the deadline date and time. These values will be exported back to the workflow.

- Bind the deadline value determined in the earlier step to the dialog step, and use it as an expression in the **Deadline** tab.
- Maintain the notification text in the **Description** tab of the task for each deadline type. You can use any container element as a variable in that text.

4.7 Summary

In this chapter, you've learned how to create a workflow with all its different steps. We've gone into detail about the different important attributes of the workflow steps, their purpose, and when to use the different steps. We discussed the container elements and how the information flows between different components of workflow. Then, we specifically covered multiple element containers and how they're used for parallel processing of work items. We focused on escalation management scenarios, which are done using deadline monitoring in workflows. We looked at both scenarios of escalation management: (1) reminders/escalated emails/notifications and (2) send the work item to the next level for processing.

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.

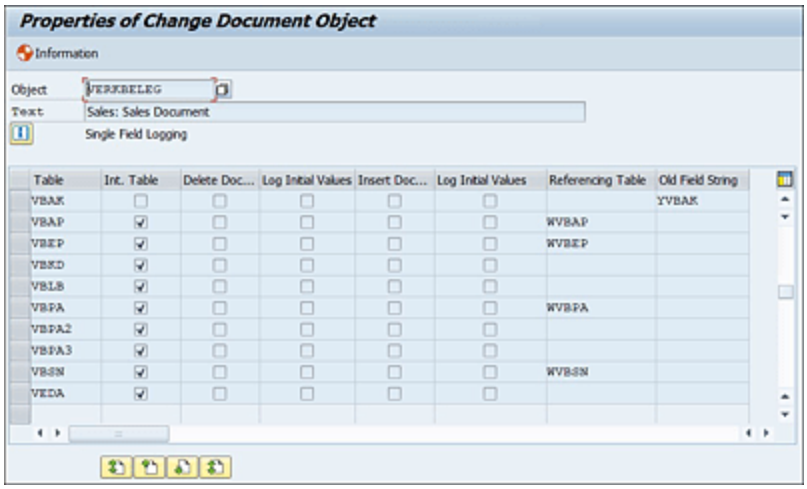


Figure 5.1 A Change Document Object in SAP

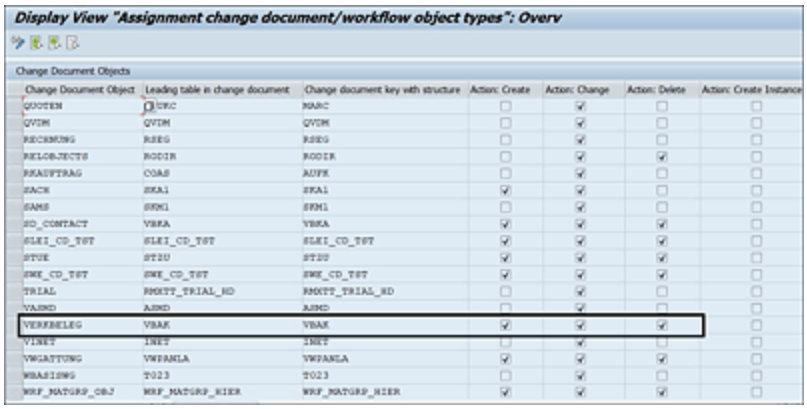


Figure 5.2 Viewing the Change Actions Maintained against a Change Document Object in Transaction SWED

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.

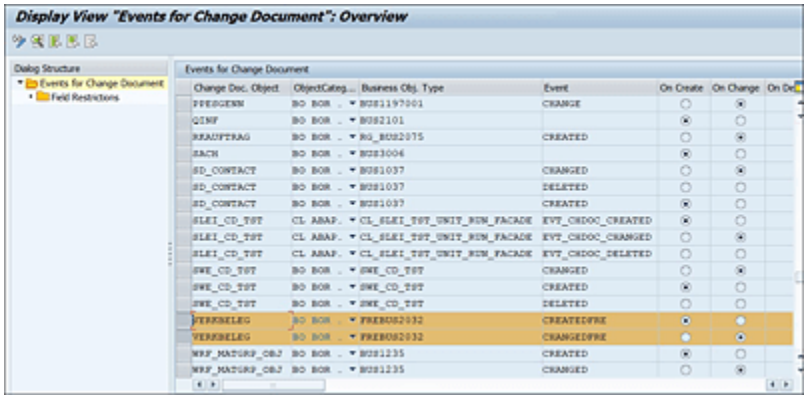


Figure 5.3 Configuring BOR/Class Events against a Change Document Object in Transaction SWEC

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.

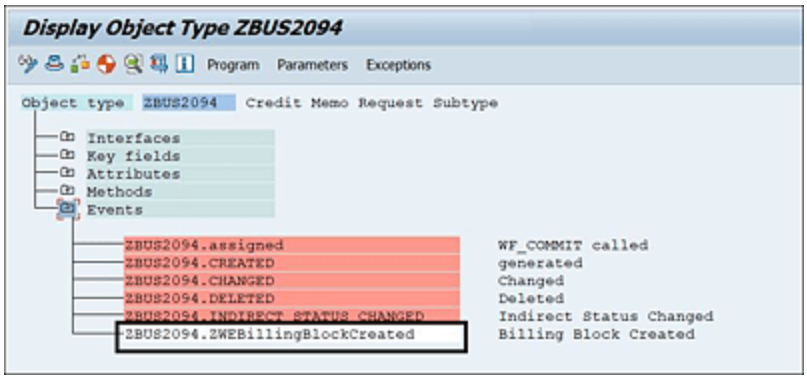


Figure 5.4 Subtype with Custom Event Added from Transaction SWO1

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 “ZWEBILLINGBLOCKCREATED”.

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

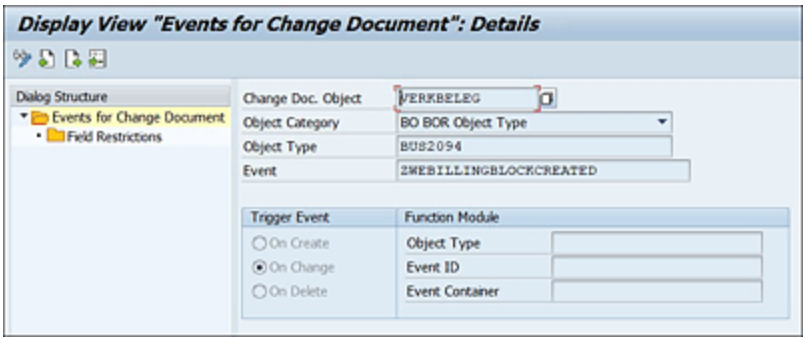


Figure 5.5 Custom Event Configured for the Change Document Object in Transaction SWEC

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.

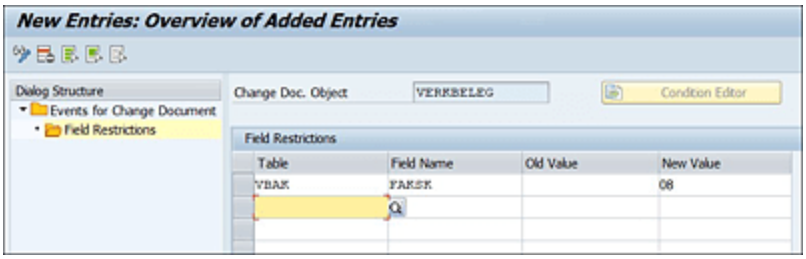


Figure 5.6 Maintaining Field Restrictions for the Event Trigger via Change Documents in Transaction SWEC

- 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.

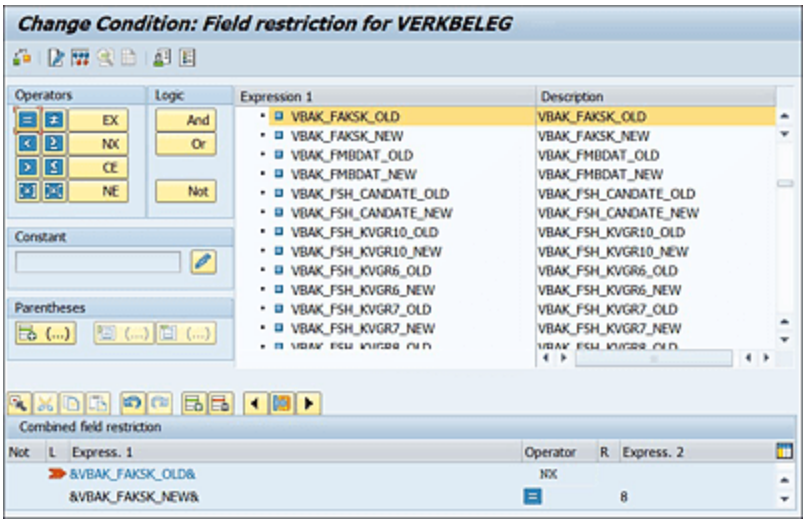


Figure 5.7 Maintaining Field Restrictions with the Condition Editor in Transaction SWEC

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.

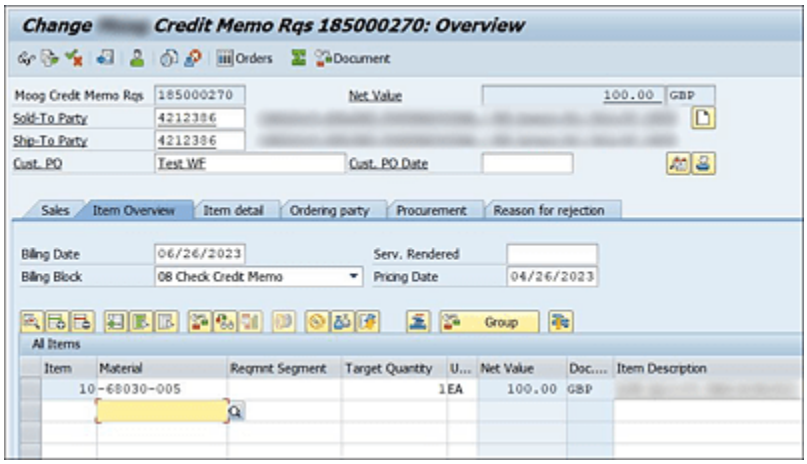


Figure 5.8 Changing a Credit Memo Request in SAP S/4HANA with the Billing Block Set

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.

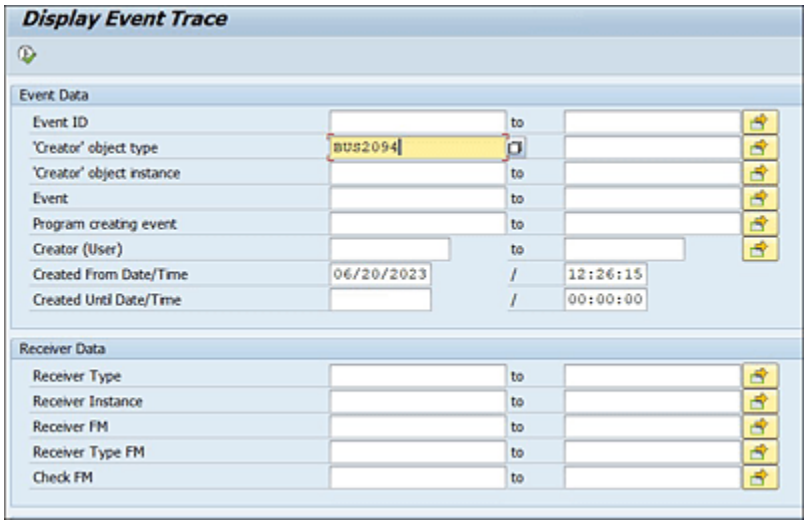


Figure 5.9 Checking the Event Trace from Transaction SWEL

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 VA01/VA02) 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.

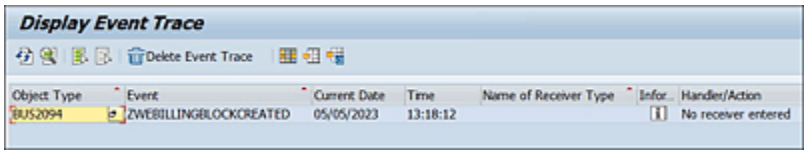


Figure 5.10 Event Trace Showing Event Has Triggered

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.

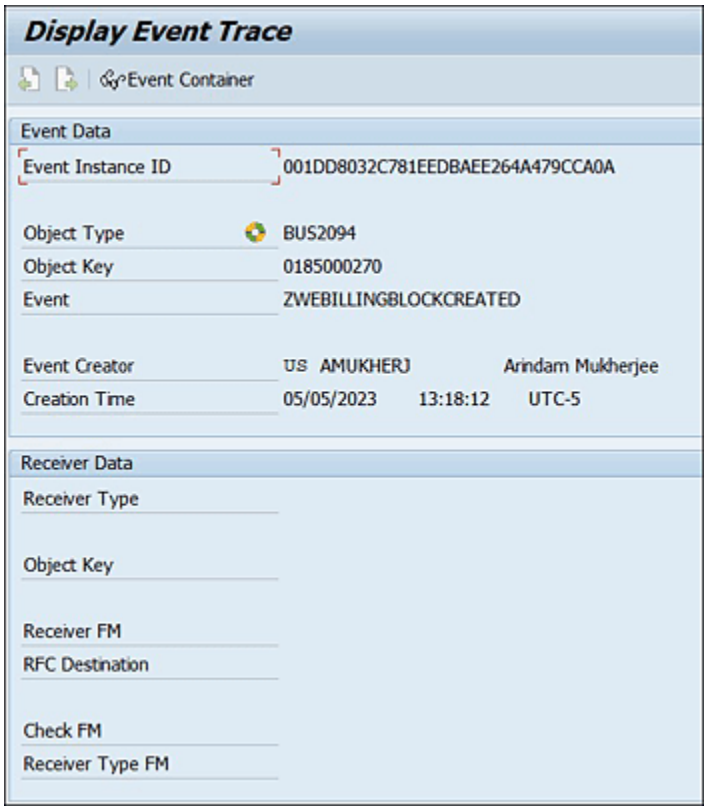


Figure 5.11 Event Trace Details Showing Event Creator Data

> **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 BS02.
- 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.

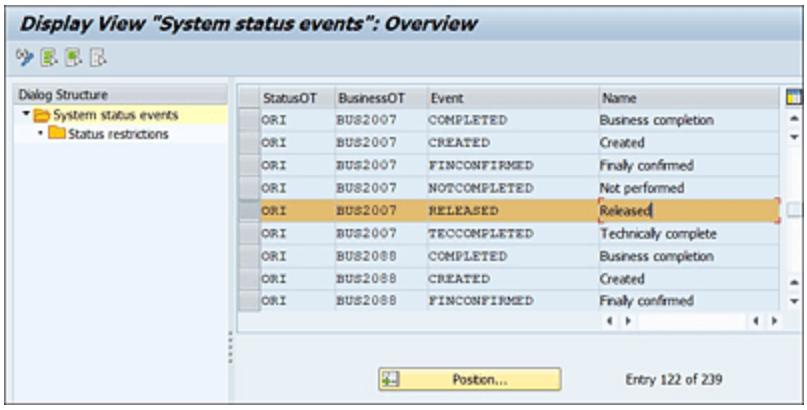


Figure 5.12 Event Configuration with System Status in Transaction BSVX

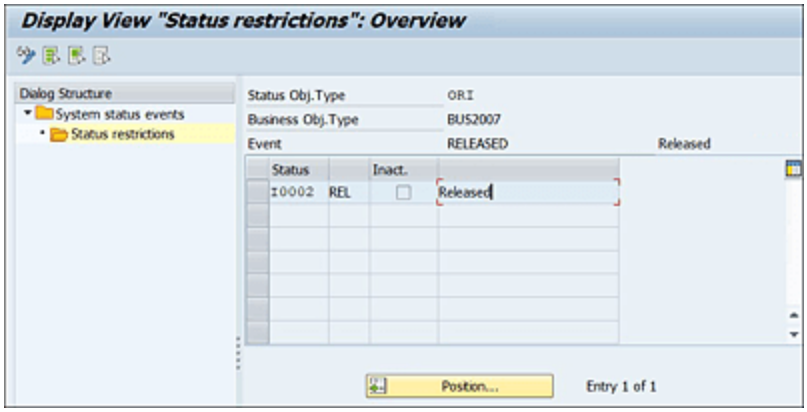


Figure 5.13 System Status Restrictions for Event Configuration

- 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).

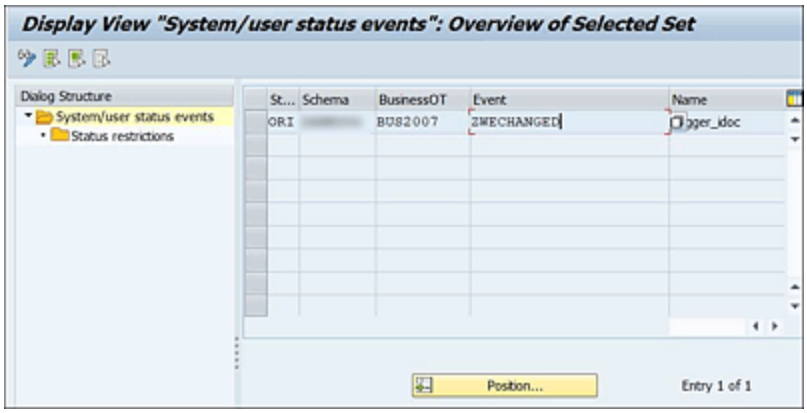


Figure 5.14 Event Configuration with User Status Profile in Transaction BSVZ

- 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.

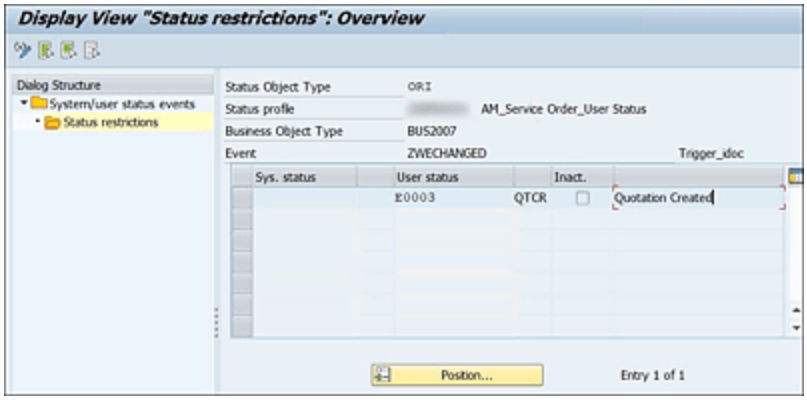


Figure 5.15 User Status Restrictions for Event Configuration

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.

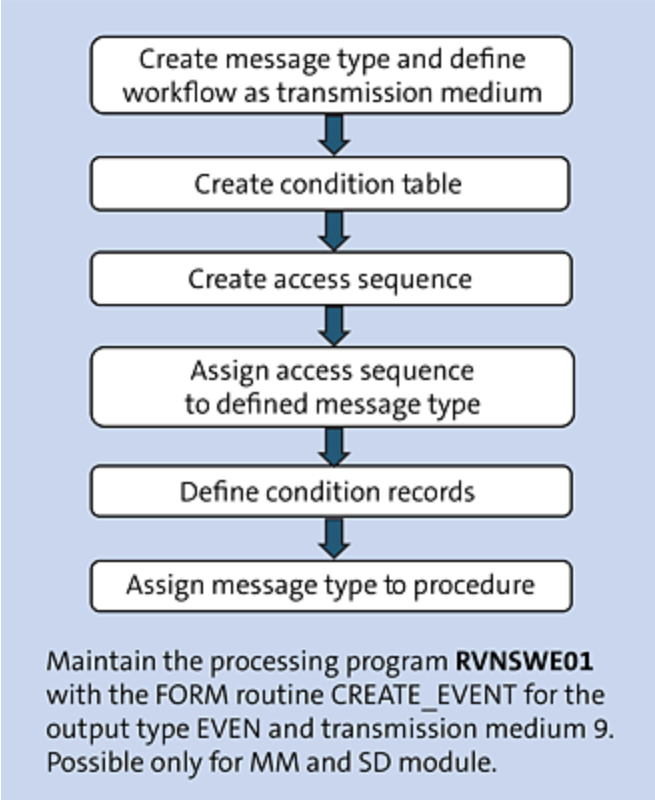


Figure 5.16 Steps for Event Configuration via Table NAST Message Control Configuration

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.

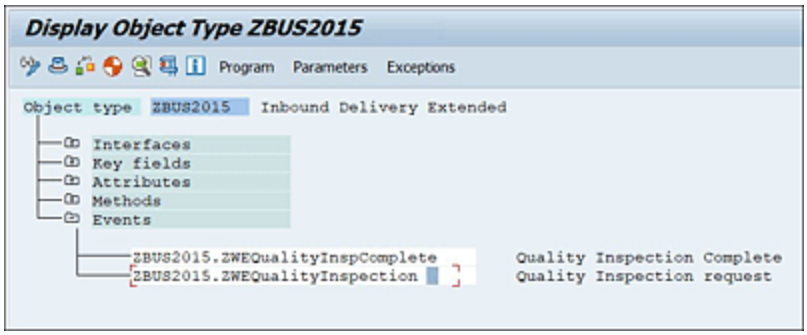


Figure 5.17 BOR Subtype Definition with Custom Events

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.

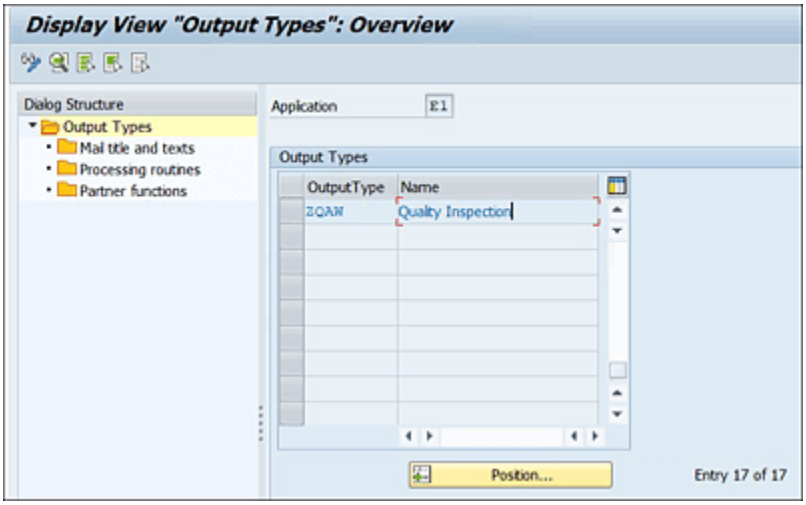


Figure 5.18 Custom Output Type Definition in Application E1 in Transaction NACE

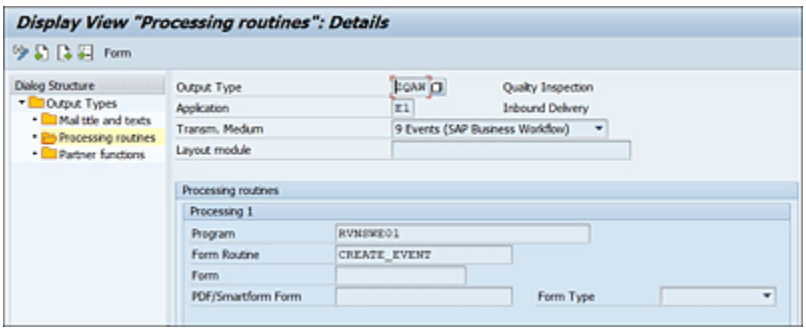


Figure 5.19 Maintenance of Processing Routine for Transmission Medium 9 in Transaction NACE

4. 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.

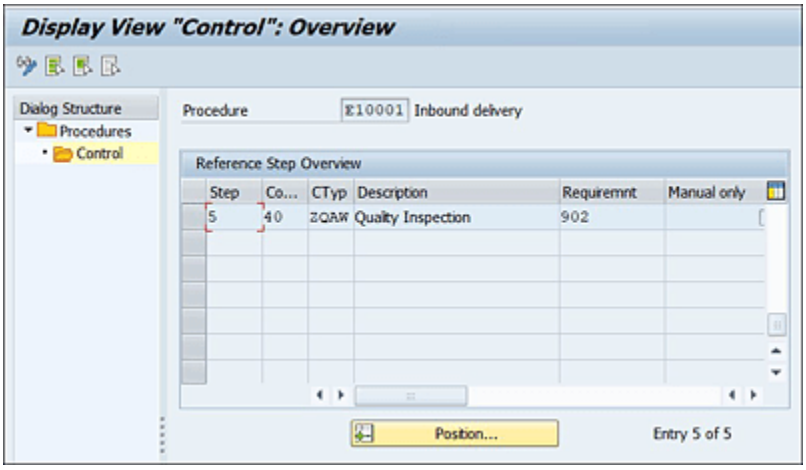


Figure 5.20 Addition of Custom Output Type to the Output Procedure along with Requirement Routine

5. 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".

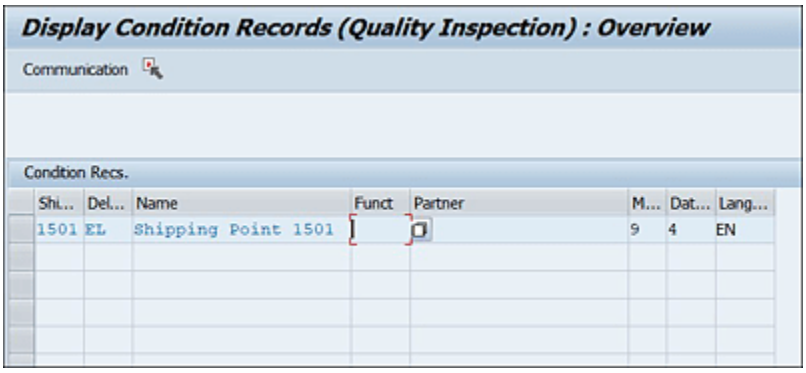


Figure 5.21 Condition Record Maintenance for Custom Output Type

6. 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.

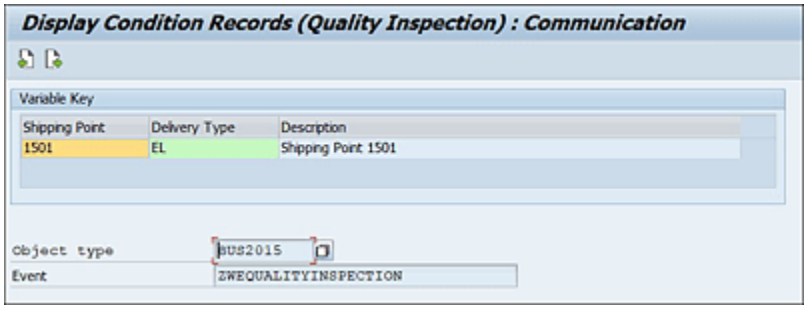


Figure 5.22 Enter the Communication Data for the Condition Record with BOR Type and Event

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.

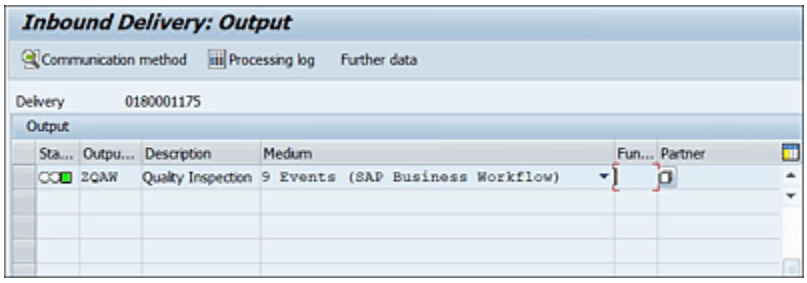


Figure 5.23 Workflow Event Output Generated after Post Goods Receipt of Inbound Delivery

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**.

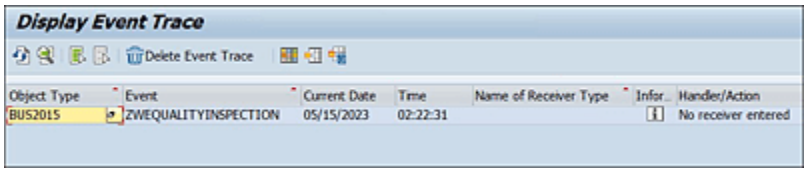


Figure 5.24 Event Trace Showing Event Triggered for the Output Type

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.

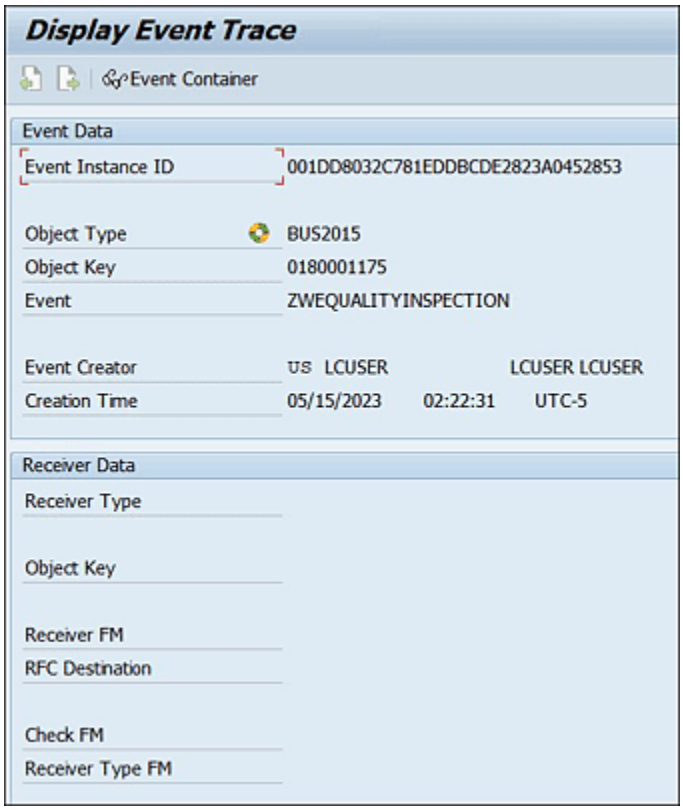


Figure 5.25 Event Trace Details Showing Event Creator Instance Information

> **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 `ZCLOTC_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 'ZCLOTC_INV_RELEASE_ACCOUNTING', " Type
            lc_event   TYPE sibfevent
   VALUE 'APPROVE_INV_WORKFLOW', " Event
            lc_catid   TYPE sibfcatid
   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 conainer
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.

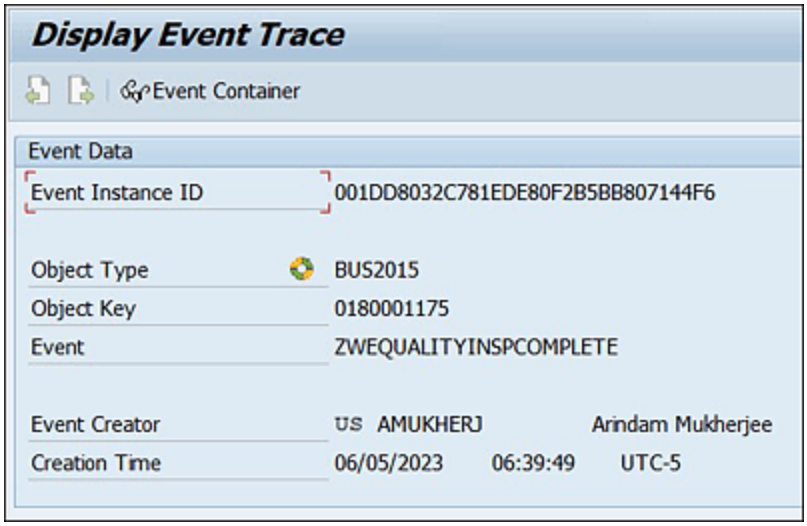


Figure 5.26 Event Trace Showing Event Creator Information

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.

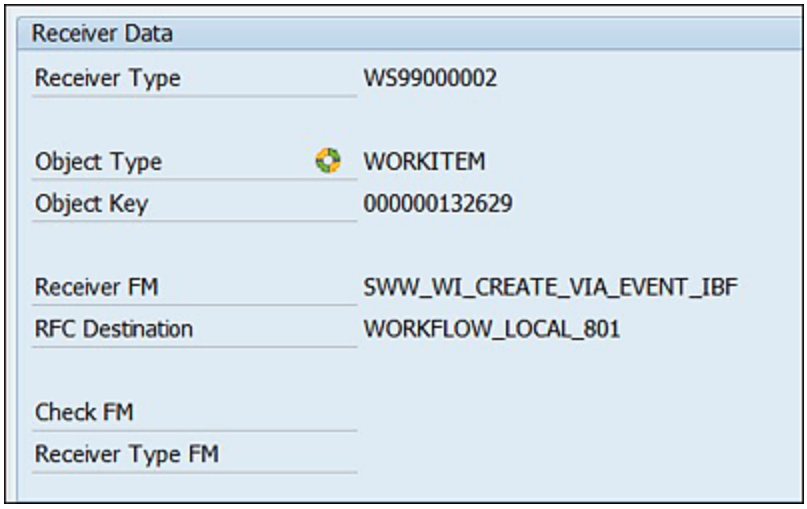


Figure 5.27 Event Trace Showing Event Receiver Information

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.

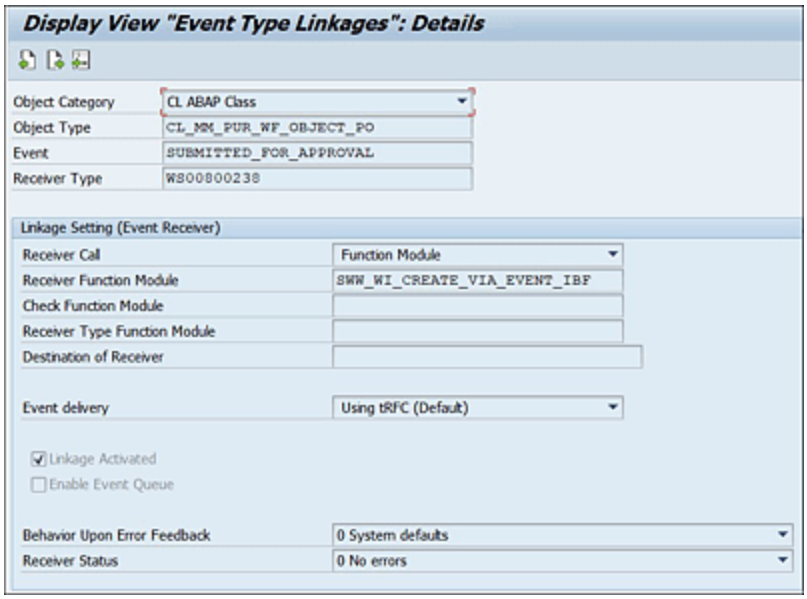


Figure 5.28 Event Linkage from Transaction SWE2 or Transaction SWETYPV

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.

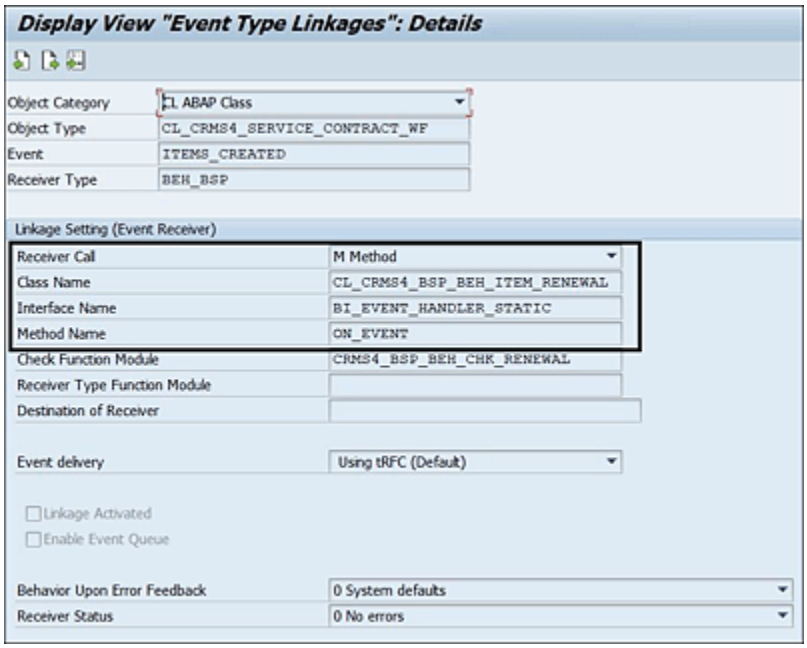


Figure 5.29 Sample Event Linkage Showing the Receiver Method Call

- **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.

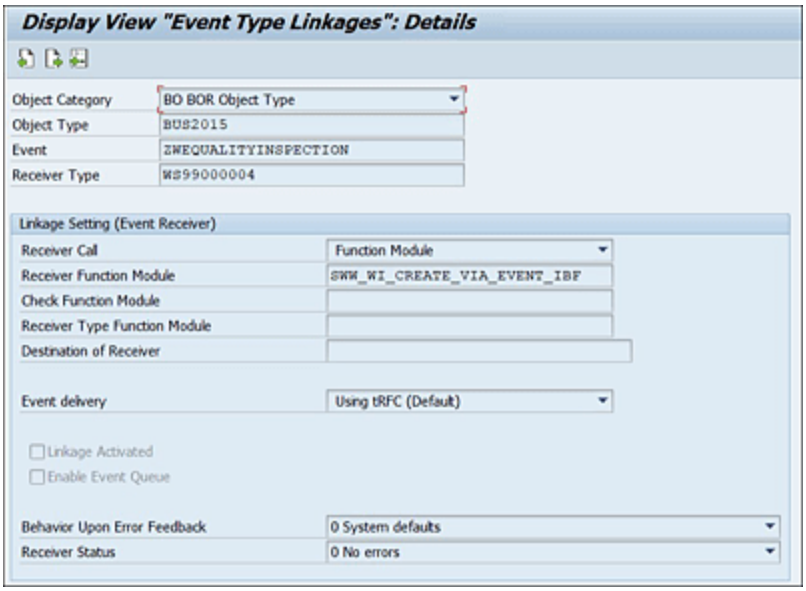


Figure 5.30 Event Linkage before Assigning a Start Condition

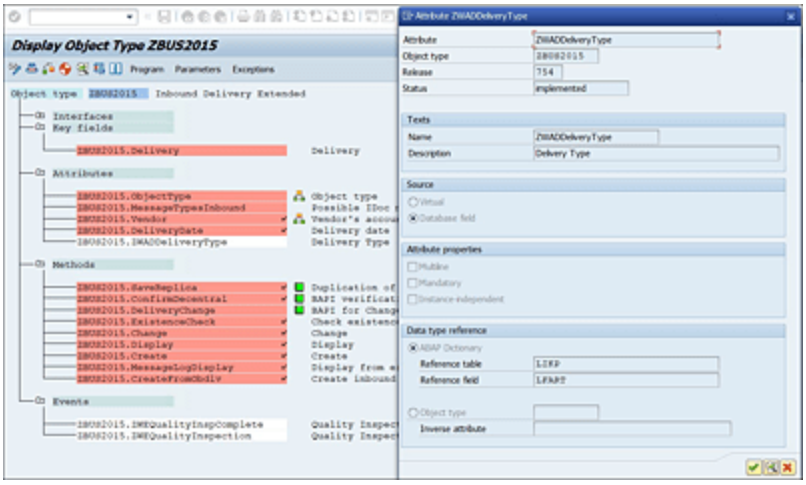


Figure 5.31 Creation of a Custom Attribute in the BOR Subtype Definition

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.)

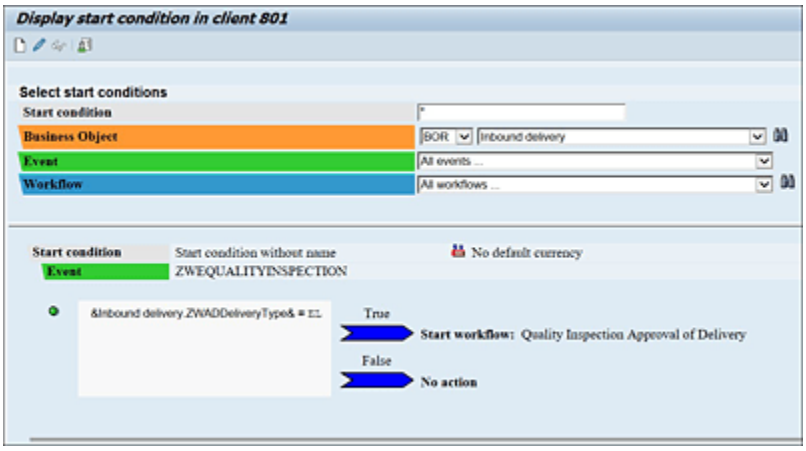


Figure 5.32 Creation of Start Condition for Event Linkage from Transaction SWB_COND

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**).

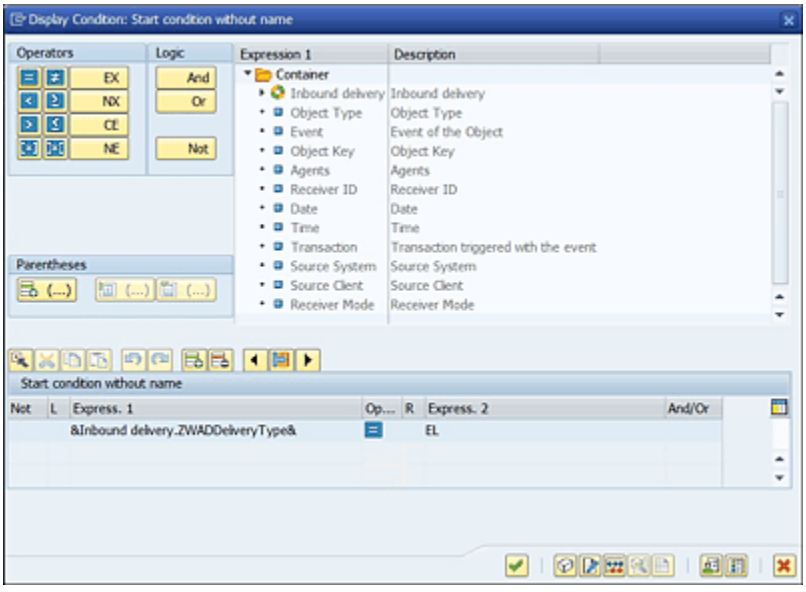


Figure 5.33 Maintaining the Start Condition Using the Condition Editor in Transaction SWB_COND

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.

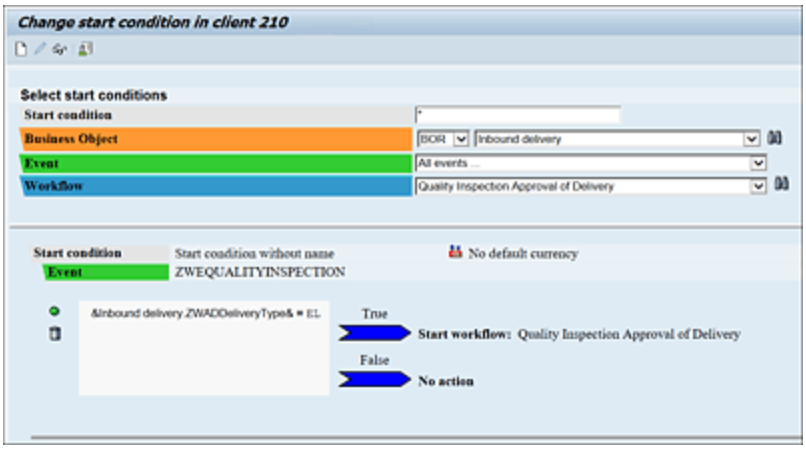


Figure 5.34 Activate the Start Condition (Indicated by the Green Icon to the Left)

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.

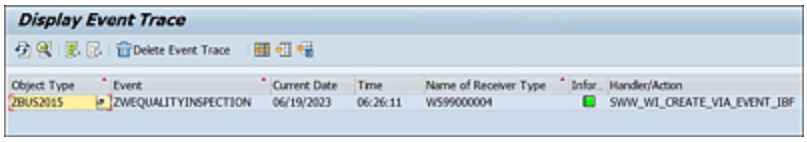


Figure 5.35 Event Trace Showing Event Triggered after Evaluation of Start Condition

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.

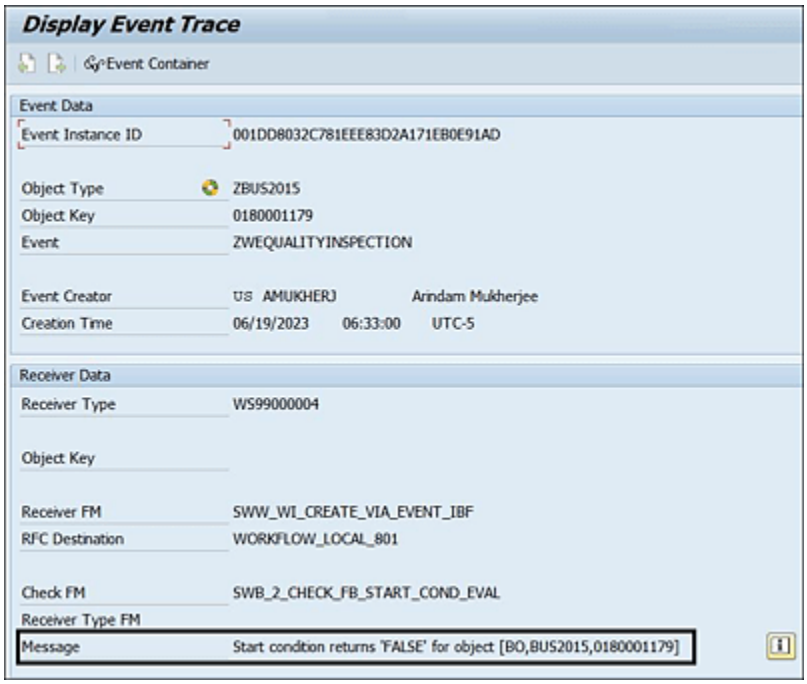


Figure 5.36 Event Trace Showing Exception Raised due to Start Condition Being Evaluated to False

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.

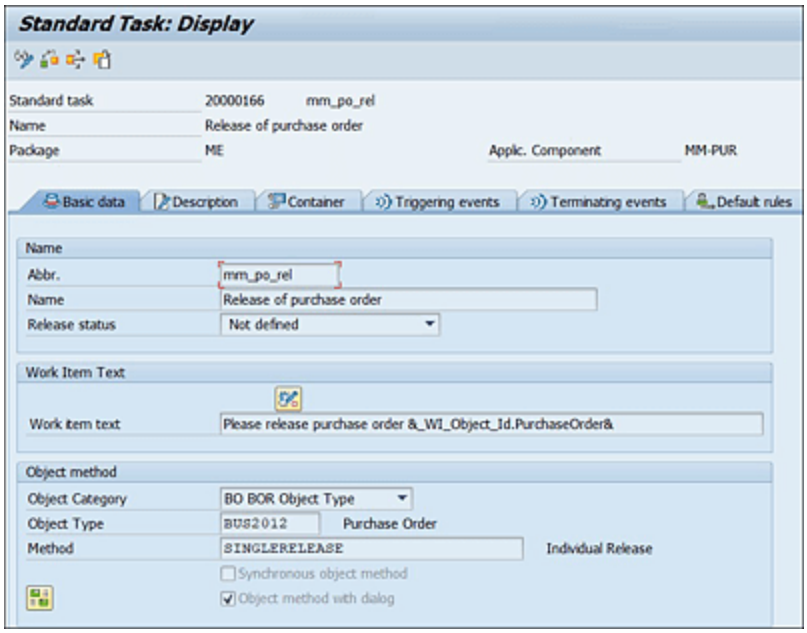


Figure 5.37 Example of an Asynchronous Standard Task

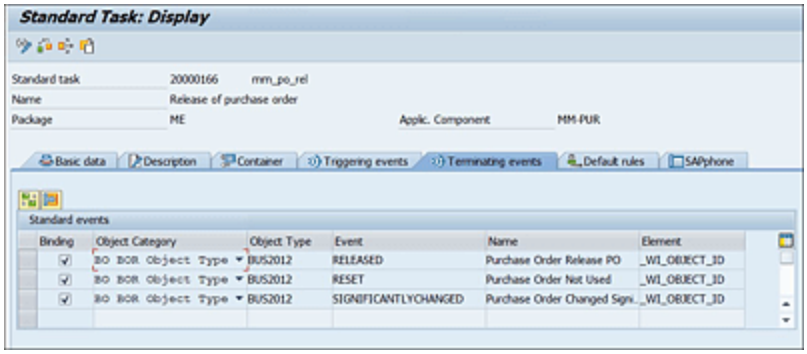


Figure 5.38 Terminating Events in an Asynchronous Dialog Task

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.

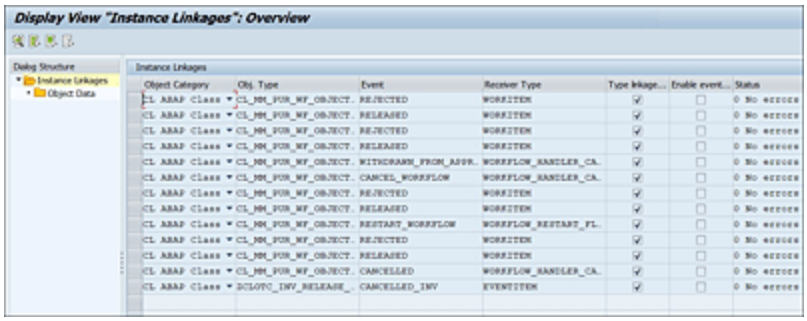


Figure 5.39 Instance Linkages from Transaction SWE3 or Transaction SWEINST

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).

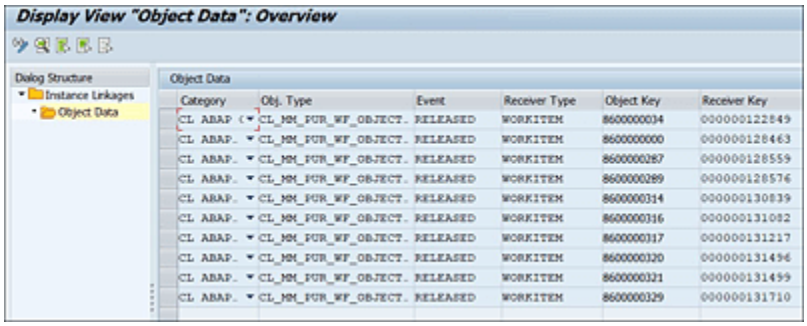


Figure 5.40 Object Instances for an Instance Linkage from Transaction SWE3 or Transaction SWEINST

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  SWETYPECOU-OBJTYPE
*"     VALUE(OBJKEY) LIKE  SWEINSTCOU-OBJKEY
*"     VALUE(EVENT) LIKE  SWETYPECOU-EVENT
*"     VALUE(RECTYPE) LIKE  SWETYPECOU-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.

6 Agent Determination

This chapter discusses different types of agents involved in workflows. We look at the definitions of each type of agent and go through some examples to illustrate the concept of each type of agent and the agent determination techniques commonly used in classical workflow development. Finally, we look at how to maintain a simple HR organizational structure for workflow and use that for agent determination.

An agent is an executor of a dialog work item in a workflow. Agents are the decision makers in the business scenario involving your workflow. Obviously, you need to take care in designing the agent determination rules for each dialog step in your workflow. The design should be such that the agent assignment process is robust, yet it remains flexible for changes, and the rules are easy to maintain from a business standpoint. This is the reason why we avoid assigning users directly to a workflow step as an agent. Instead, we make use of rules or HR organizational structure objects for agent assignment at the workflow step or task level.

Another important aspect of agent determination is the authorization or security role perspective. When designing the agent determination for a particular step, you need to make sure that the person who is supposed to execute the work item has the necessary security role to do the job. This

means that your determined agents should be a subset of a larger group of people who are authorized to do the same job.

Finally, while determining the agents for a particular step, you must keep in mind the people who should not be allowed to execute a particular step of a workflow. The people might have the relevant security roles or even be part of the same group that is determined through the agent determination rule. However, based on a specific business case, they may not be allowed to execute a particular workflow step. An example could be that of an Employee Payables group that approves employee expenses. If any member of the same group submits an employee expense of their own, then they should not be allowed to approve their own expense.

Remember that all agents involved in the workflow must have an SAP user ID, even if they are accessing a work item via an external inbox using Microsoft Outlook or some other mail service. The user ID serves as the link and the identification of the agent who executes a particular step of a workflow. This is critical in terms of audit and workflow monitoring.

Next, we'll look at the different types of agents involved in a typical workflow scenario. We'll go through the details of how each type of agent may be defined and how they are assigned to your workflow with some examples. Then, we'll discuss the different techniques used to create agent determination rules, followed by a discussion of possible agents and default roles in a workflow. Finally, we'll look at how an organizational structure is defined in SAP S/4HANA

and how it can be linked to the agents involved in your workflow.

6.1 Different Types of Agents in Workflow

Agents within a workflow can be classified into a few categories, which we'll look at in the following sections.

6.1.1 Possible Agents and Responsible Agents

The superset of all users who are allowed to execute a particular work item are called the possible agents. Possible agents are always assigned at the task level (dialog) and never at the actual step level. Because possible agents define the list of users who are actually authorized to receive the work items for a particular task definition, it's logical to tie them with security roles that authorize the user to perform the underlying function of the task. For example, if a particular dialog task is supposed to release a purchase order, then the possible agents for the tasks could be a business role tied to all the people who have the access to release the purchase order. It's also possible to define a task as a *general task* in the attributes, which means that any SAP user is authorized to receive that task in the inbox and execute it. This option should be used cautiously in a nondevelopment environment as this means that the task has no security around it. We'll discuss possible agents in more detail in Section 6.3.

Responsible agents, on the other hand, are the group of people who are selected to execute a particular work item. This group of people should be a subset of the possible agents. If not, then the person(s) won't be able to receive that work item in their inbox. Let's take the following example to better understand the difference between possible agents and responsible agents:

- **Possible agents**
  Group of purchasing managers who are authorized to approve the release of a purchase order.
- **Responsible agents**
  For a particular purchase order, based on the purchase organization, the purchasing group, and the amount on the purchase order, the specific group of purchasing managers who can approve it.

Responsible agents are usually assigned at the workflow step level via an expression or an agent determination rule. It's also possible to assign responsible agents based on organizational objects, such as an organizational unit, position, or job. We'll look at the various options available to integrate organizational structure objects into your agent assignment in Section 6.4.

You can also assign responsible agents at the task level via a default rule option. This default rule gets executed if there is no agent maintained at the workflow step level or if the agent determination fails at the workflow step level. This means that for multistep tasks, the default rule at the task level has lower priority than the responsible agents maintained at the workflow step level. However, for single-step tasks, this is the only mechanism of maintaining

responsible agents. Default rule assignment at the task level will be looked at in more detail in Section 6.3.

Figure 6.1 illustrates an example of a single-step standard task using a default rule for agent determination. Here, you can bind data from the task container to the rule container, which can be used for agent determination. This process will be covered in greater detail later in this chapter.

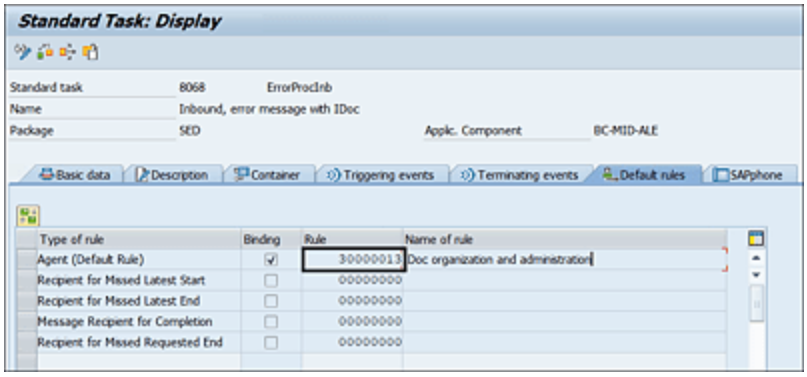


Figure 6.1 Agent Determination with a Default Rule

Figure 6.2 illustrates an example of a multistep task with an agent determination rule defined at the step level. Here, you can bind data from the workflow container to the rule container, which can be used for the agent determination logic.

Figure 6.3 illustrates an example of a multistep task with agent determination using **Expression** at the step level.

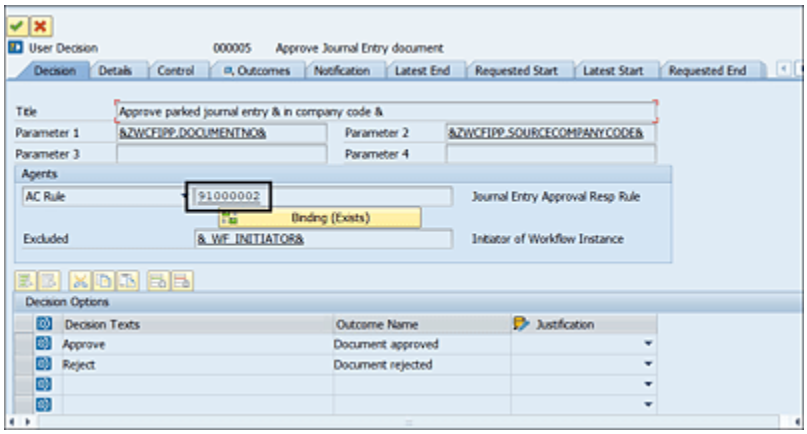


Figure 6.2 Agent Determination with a Rule at the Workflow Step Level

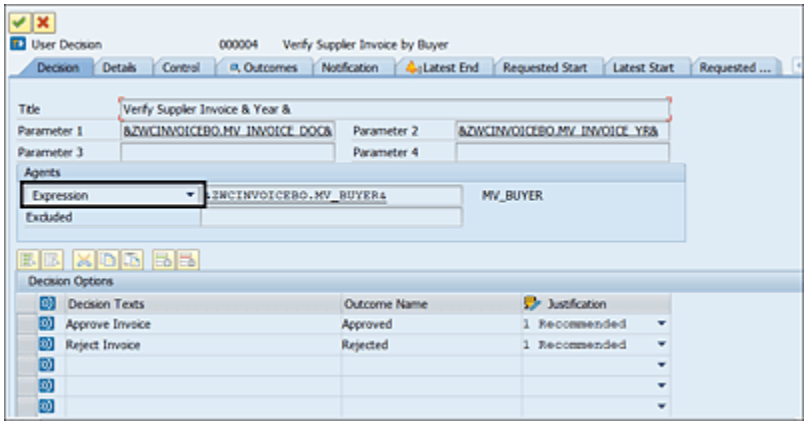


Figure 6.3 Agent Determination with an Expression at the Workflow Step Level

With the **Expression** option shown in Figure 6.3, you must select a container element from your workflow that contains the agent object (user, organizational unit, position, job, role, etc.) in the formation of structure `SWHACTOR`. Here, the first two characters (`OTYPE`) denotes the type of the agent object, for example, “US” for user ID, “O” for organizational unit, “S” for position, “C” for job, “P” for person. The next 12 characters represent the object ID associated with the type, for example, user ID, organizational unit ID, position ID, job ID or person number, and so on.

Selected agents or *recipients* are the people who receive the work item in their inbox. So, they can be considered as a runtime version of the responsible agents. Selected agents are evaluated by considering the responsible agent assigned at the workflow step level or default rule at the task level, validating them with the possible agents assigned at the task level and removing excluded agents, if any.

Figure 6.4 illustrates the concept of selected agents for different scenarios by considering possible agents, responsible agents, and excluded agents.

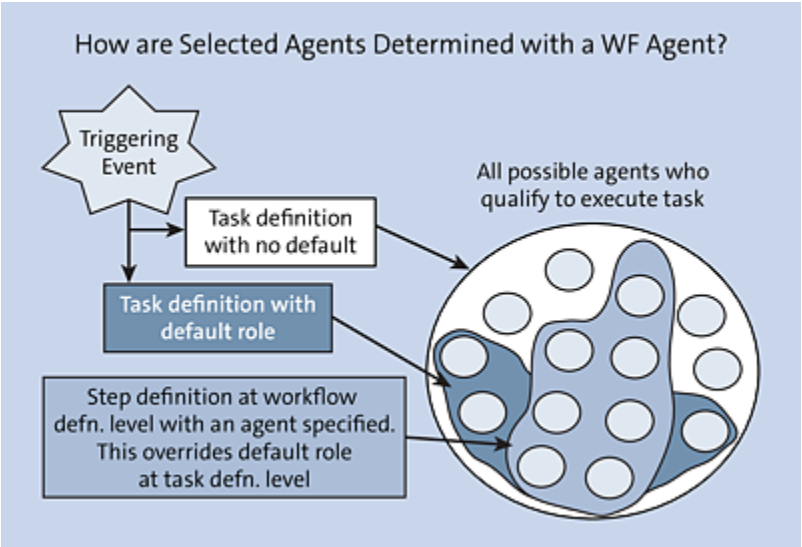


Figure 6.4 Selected Agent Determination at the Workflow Step Level

6.1.2 Excluded Agents

These are the set of people to exclude from receiving the work item for a particular workflow step. Let’s consider the example of a workflow for the release of purchase requisitions related to office supply products. The approval of the purchase requisition is done by a group of purchasing

managers based on the purchase organization and the purchasing group. Now if one of these purchasing managers requests a purchase requisition of their own, then that same person should not be able to approve his own request. This could cause an audit violation or lead to uncontrolled or even fraudulent purchases in the company. Therefore, in this case, the purchase requisition requester should be set as an excluded agent for the purchase requisition approval step.

Another example where excluded agents could be useful is where you need more than one sequential approval and don’t want the same person approving both steps (assuming that the responsible agents of the two steps have people in common). In this case, the actual agent of the first approval step should be set as the excluded agent of the next step(s).

Excluded agents are always set via expressions at the workflow step level. You can assign any workflow container element (multiline) with a base structure such as `SWHACTOR` as the excluded agent for a workflow step.

Figure 6.5 illustrates an example where the workflow initiator is assigned as an excluded agent at the workflow step level. This effectively means that after the role resolution is done for the rule at the workflow step level, the initiator of the workflow is removed from that list of agents, and the resultant list becomes the selected agent(s) of this step.

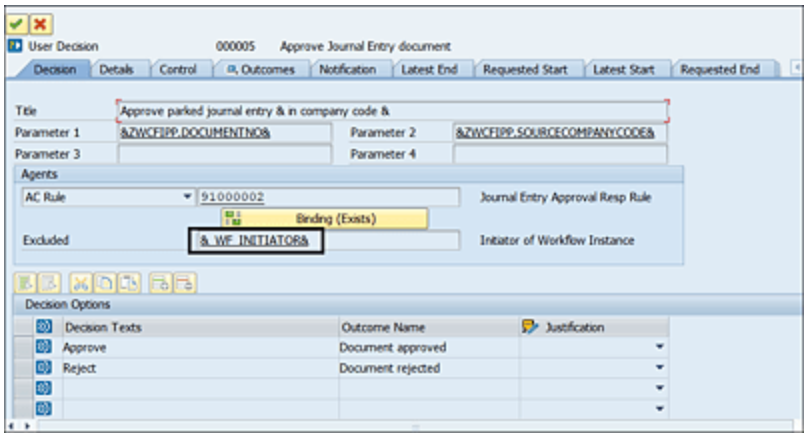


Figure 6.5 Assigning Excluded Agents at the Workflow Step Level with an Expression

6.1.3 Actual Agents

An actual agent is the person who executed a particular work item and completed the workflow step. This agent's user ID is captured by the system container element `_Wi_Actual_Agent` (`Agent`) of the dialog task once it's completed. Figure 6.6 shows a runtime view of the task container for a completed dialog work item, with the actual agent container element populated.

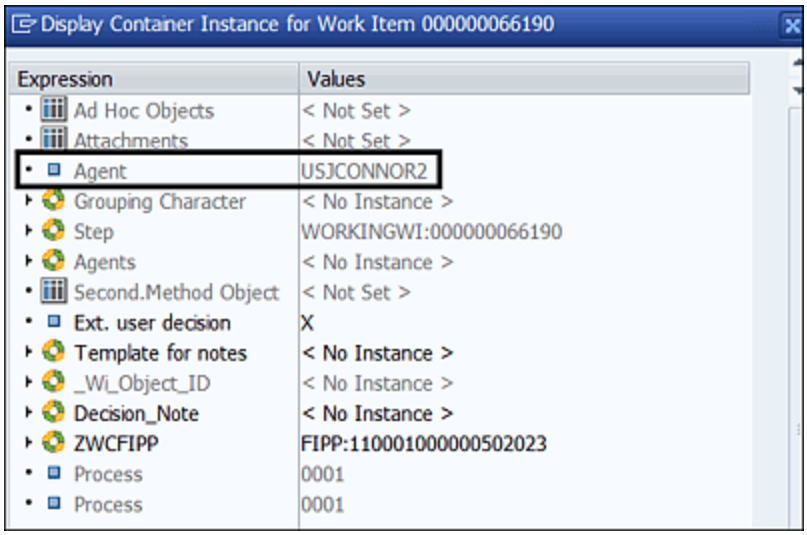


Figure 6.6 Actual Agent of a Completed Dialog Step Captured in System Element _Wi_Actual_Agent (Agent)

6.1.4 Deadline Agents

Deadline agents are entered on the relevant deadline monitoring tab (i.e., **Latest End**, **Requested Start**, **Latest Start**, **or Requested End** tabs) of a dialog workflow step. These deadline agents receive a work item in their inbox when the respective deadline is reached. Deadline agents may also be entered in the same way as responsible agents of a dialog step, that is, as HR organizational objects, security role, rule, user ID, or via an expression. Like responsible agents, the most commonly used approach is using rule or expression. Figure 6.7 illustrates an example of a deadline agent being maintained under the **Latest End** deadline tab using **Expression**.

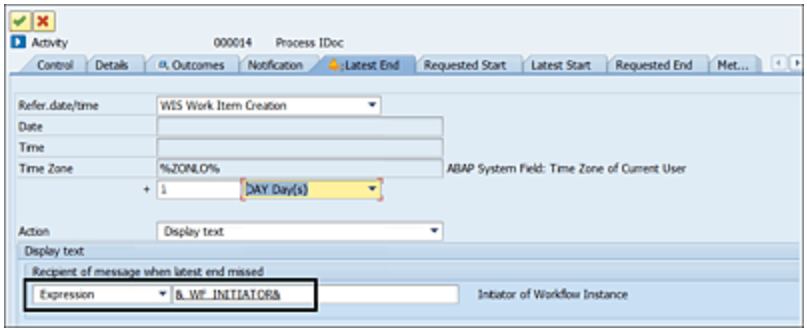


Figure 6.7 Deadline Agent Assignment at the Workflow Step Level in a Deadline Tab

It’s also possible to maintain deadline agents directly at the task definition level using the **Default Rules** tab. Here, you can assign an agent determination rule to any of the deadline recipient rules. This option is mostly useful for single-step tasks.

6.1.5 Notification Agents

Notification agents are the people who receive a notification email in their inbox on completion of a work item. These agents are entered on the **Notification** tab of the dialog step. Again, you can use HR organizational objects, security roles, rule, user ID, or an expression to assign these agents. Notification will be sent to the email address maintained in their user master record. You can customize the notification text in the task definition under the **Description** tab. You can change the text type to **Completion Text** and then maintain the corresponding text. Figure 6.8 shows the option of maintaining a notification agent for work item completion.

Figure 6.9 shows how to maintain the completion text for a notification under the task **Description** tab.

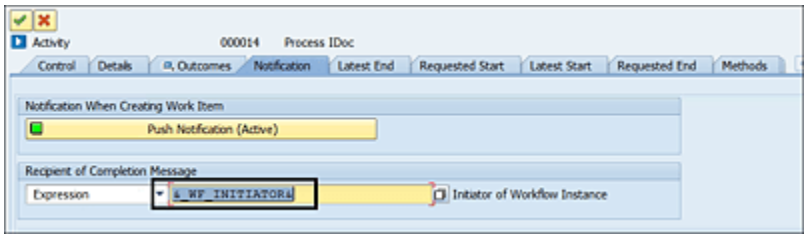


Figure 6.8 Maintain the Agent Assignment for the Work Item Completion Message at the Workflow Step Level

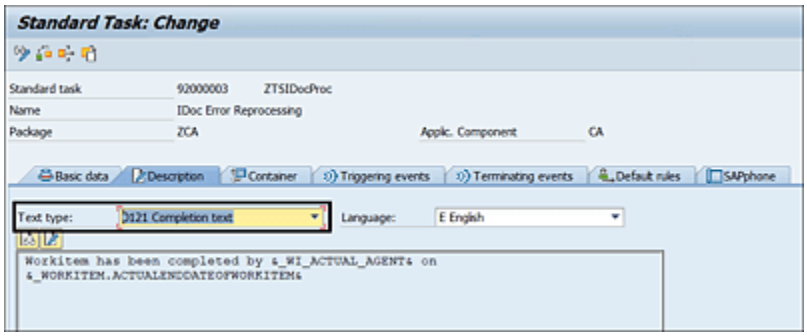


Figure 6.9 Maintain Notification of Completion Text in the Task Description Tab

It’s also possible to configure push notifications for work item creation to the selected agents based on settings made in the **Notification** tab. You can configure actions (only for the user decision step) and texts for notification. The users directly get an option to **Approve** or **Reject** the work item from the push notification message on the SAP Fiori launchpad.

6.2 Agent Determination Rules

Rules can be created via Transaction PFAC. They may be also displayed or maintained via the variant Transactions PFAC_DIS and PFAC_CHG, respectively. Each rule gets an eight-digit system-generated number (per the prefix number configuration defined in Transaction SWU3). Rules are identified by the prefix “AC” and the eight-digit number. Rules may be tested via Transaction PFAC using the **Simulate** option or by executing function module `RH_GET_ACTORS`. Rules are the most commonly used option to determine responsible agents for a workflow step (along with expressions) or as the default rule for a single-step task. Rules are also used at the task level to determine the deadline or notification agents for single-step tasks.

In the following sections, we’ll go through the details of how a rule is created, the most commonly used rule types, how to define a rule container and perform binding with the workflow step, and the steps to create each rule type in detail. These sections will help you get familiar with rule definition and the steps to develop a simple rule for your workflow.

6.2.1 Rule Definition

When you define a new rule from Transaction PFAC, you must enter a rule short ID or abbreviation and a name. Then you must choose the type of the rule under the **Category**

dropdown. The most commonly used options for rule types are the following:

- **Agent Determination: Responsibilities**
- **Agent Determination: Function to Be Executed**
- **Agent Determination: Organizational Data**

Figure 6.10 shows the detail screen for creating a new rule from Transaction PFAC once you enter the rule abbreviation name and press Enter. We'll look at each of these rule types in detail in Section 6.2.4.

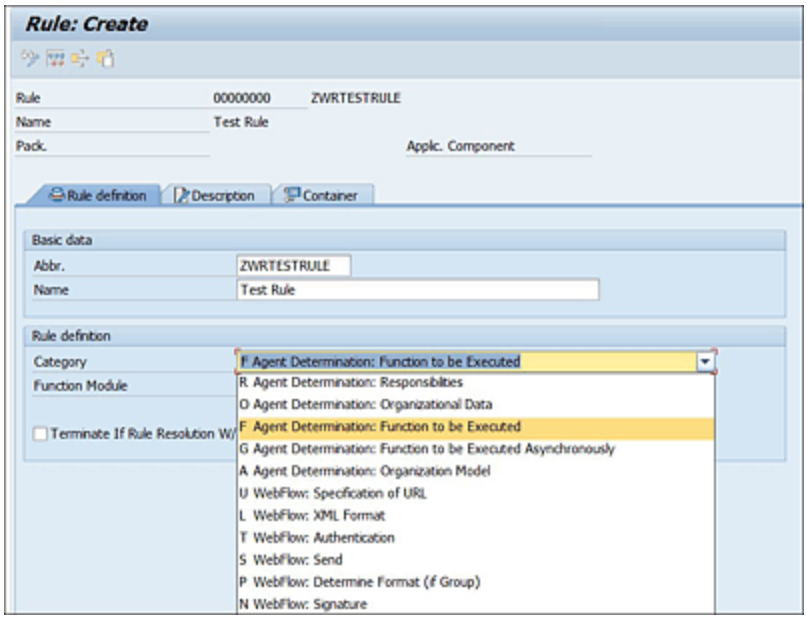


Figure 6.10 Creating a Rule from Transaction PFAC

After selecting the rule type, you can select the **Terminate if Rule Resolution W/O Result** checkbox. This checkbox if selected, would terminate the workflow and put in an error state if no agents are determined for a rule at the workflow step level. In this case, if there is a default rule maintained at the task level, then it will be ignored. The termination of the workflow occurs before even the task work item is

generated, as the rule resolution at the workflow step level happens before task work item creation. The workflow may be restarted after error, once the issue with the rule resolution has been fixed.

On the other hand, if this checkbox isn't selected, then the workflow would not go into error if rule resolution failed. In this case, the workflow would remain in progress, but the task work item would not be in anyone's inbox. These hanging work items can be analyzed and processed by the workflow administrator via Transaction SWI2_ADM1. With this transaction, the administrator can reexecute the rule if the problem has been fixed or forward the work item to another user.

Note

Both selecting and deselecting the **Terminate if Rule Resolution W/O Result** checkbox requires workflow administrator intervention in case the agent determination fails, which is feasible. However, for critical business processes, it may be better to design the agent determination in a way to always have a fallback agent. This may be done via secondary priorities or directly via ABAP code with a backup logic that gets executed when the primary logic fails.

6.2.2 Rule Container

On the **Container** tab, you define the importing container elements that will be used as criteria for executing your rule

logic. The container elements may be scalar or multiline. Create the container elements similar to workflow definitions or task definitions. Figure 6.11 shows a view of the **Container** tab inside the rule definition in Transaction PFAC, where you can define new container elements per requirement in your rule. Containers were discussed previously in Chapter 3 in additional detail.

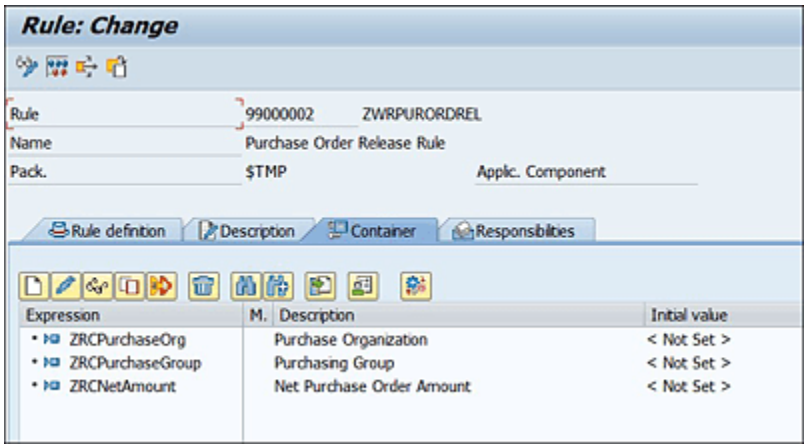


Figure 6.11 View of the Container Definition for Rule

6.2.3 Binding to the Rule Container

Once you've created your rule and you want to add it to a workflow step, you must also maintain the binding between the workflow container elements and the rule container elements. The binding editor can be opened by clicking on the **Binding** button under the rule number in a dialog workflow step from Transaction SWDD. For the task default rule, you must enter the rule number in the **Default Rules** tab of the task definition in Transaction PFTC and then click on the **Binding** button.

Figure 6.12 displays a sample binding between the workflow container and rule container. Through this binding, the data

from the workflow is mapped to the rule, which can be used in the agent determination.

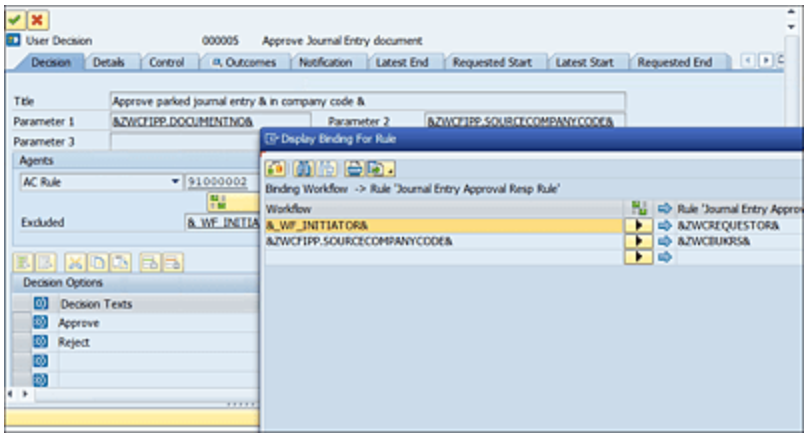


Figure 6.12 Example of Binding between the Workflow Container and Rule Container

If the rule is maintained at the task level, then the same binding must be maintained between the task container elements and the rule container elements.

The output of the rule resolution is made up of the selected agents for the workflow step or single-step task. Once the dialog step using the rule has been completely executed, you'll be able to see only the actual agent of the work item, not the selected agents determined by the rule. If you want to track the selected agents of the rule, add a binding from the system container element `_RULE_RESULT` of your rule to your workflow container element (task container element if the rule is at the task level), so that the selected agents will be saved in your workflow log.

6.2.4 Rule Types

In the following sections we'll go into detail about the three rules types we first introduced in Section 6.2.1.

Agent Determination with Responsibility

A rule definition with responsibility is used when you can segregate different groups of users based on some clearly defined criteria. Let's consider a simple example (see Table 6.1) for a rule used to determine the approvers for a purchase order release workflow.

Condition Field	Value From	Value To	Approver
Purchase Org	1,000		USER1
Purchasing Group	A01		
Net Amount (Dollar)	0	1,000	
Purchase Org	1,000		USER2
Purchasing Group	A01		
Net Amount (Dollar)	1,000	2,000	
Purchase Org	1,000		USER3
Purchasing Group	A02		
Net Amount (Dollar)	0	5,000	

Table 6.1 Business Scenario for Rule with Responsibility

In this scenario, you have three criteria based on which the agent determination for purchase order release should happen. These are purchasing organization, purchasing group, and net amount of the purchase order. These three

condition fields will be created as container elements in your rule definition.

Now you need to create a responsibility definition based on each unique combination of the three criteria field values. Some of these criteria fields may include a range, for example, the purchase order net amount. You can create a responsibility definition from Transaction OOCU_RESP.

Note

Rule definition is a cross-client workbench object. However, rule responsibilities are like master data. You need to create them in each system and client separately. Although it's possible to capture the responsibilities in a Customizing transport and move them across systems and clients, it's generally not recommended because, in most cases, the master data for the responsibility definition will vary across systems, and the users or organizational objects may be different between development, test, and production systems.

Once you enter the rule number in Transaction OOCU_RESP and click on the **Create** button inside the responsibility definition, you need to enter a responsibility short abbreviation ID (**Object abbr.**), a **Name**, and a **Start date** and **End Date**. The default start date is always today, and the default end date is **12/31/9999**. However, you can change the validity per your requirement. Figure 6.13 shows the responsibility creation screen once a rule has been entered in Transaction OOCU_RESP. You then click on the **Create responsibility** button.

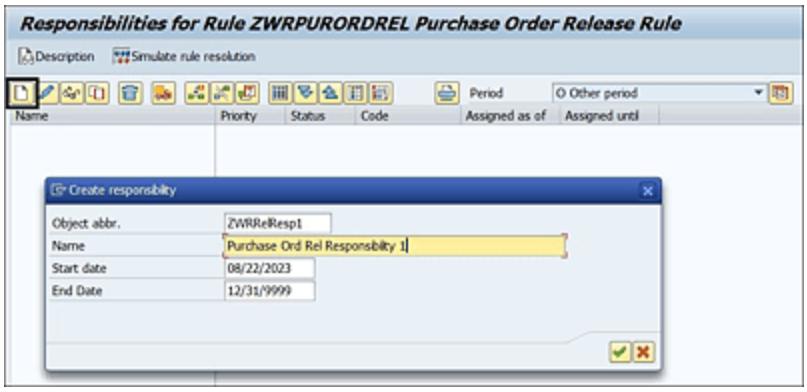


Figure 6.13 Responsibility Creation Screen in Transaction OOCU_RESP

When you press Enter, you'll be prompted to enter the combination of the rule importing data elements for your current responsibility. For example, in our example for the first responsibility definition, you'll enter the data listed in Table 6.2.

Field	Value
Purchase Org	1000
Purchasing Group	A01
Net Amount (Dollar)	0; 1000

Table 6.2 Sample Data Entered While Creating a Responsibility Definition in a Rule

Figure 6.14 shows the visual representation of the data that is entered in the **Responsibility** definition based on Table 6.2.

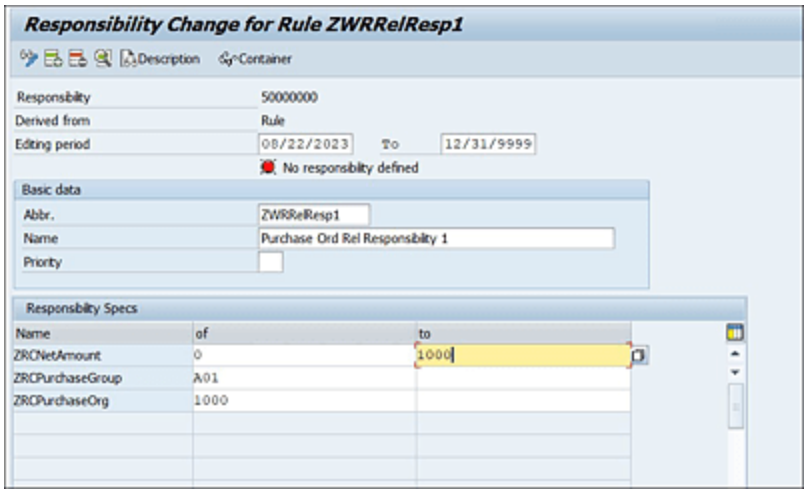


Figure 6.14 Responsibility Definition Details View inside Transaction OOCU_RESP

When you press Enter, you come back to the overview screen, where you can now select the **Insert Agent Assignment** button to maintain the agent for this responsibility definition. With this option, you are first prompted to select the agent object type, for example, **Organizational Unit**, **Position**, **Job**, **Person**, or **User**. When you select an object type, you need to enter the corresponding object ID. In our example, select the object type as **User** and enter the **Object ID** as **USER1** (see Figure 6.15). (In the actual system, **USER1** should be replaced by a valid user that you want to select for this responsibility.) Next, you select the user ID from the dropdown and the agent assignment is inserted for the selected responsibility, as shown in Figure 6.16.

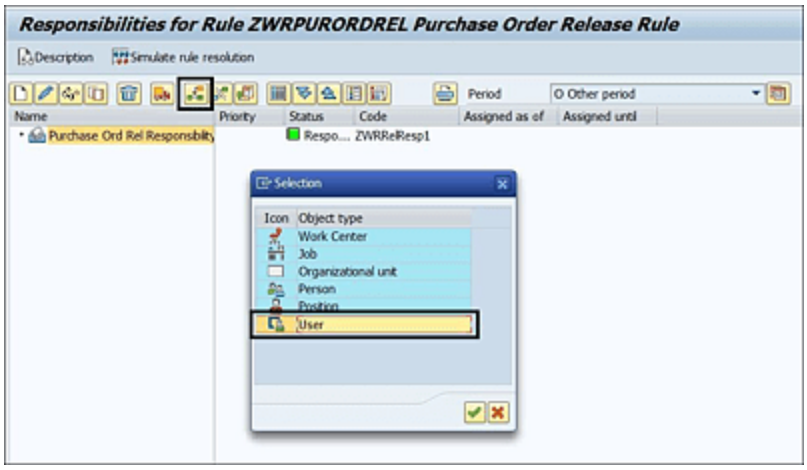


Figure 6.15 Agent Assignment for Rule Responsibility Definition: Selection of Agent Type

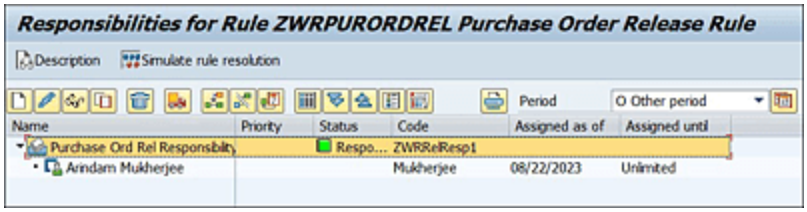


Figure 6.16 Agent Assignment Created for the Responsibility Definition

This completes the first responsibility definition for your rule. Repeat these steps for all combinations of the criteria fields per your requirement, and add as many agents to your responsibility definition as you need to.

If you need to delete an agent assignment, then select the responsibility name in the overview screen, and click on **Delete Agent Assignment** option. You can also delimit the agent assignment for a particular responsibility by choosing a validity end date.

Secondary Priority in Rule with Responsibilities

When you create or edit a responsibility definition from Transaction OOCU_RESP, you can enter a priority number.

When the rule resolution happens, the system will evaluate the responsibilities with the highest priority. For example, if you maintain your normal responsibility definitions with priority 99 and another responsibility with priority 01, the system will evaluate the 99 priority responsibilities first, and if this doesn’t match the runtime data from the workflow, then it only will proceed to evaluate the 01 priority responsibility. You can use this secondary priority approach to design a fallback agent for your agent assignment rule.

You can test your rule by using the **Simulate** button in Transaction PFAC.

Agent Determination with Function Module

This approach of rule determination gives the maximum flexibility to the developer. But on the con side, it has less visibility than the rule with the responsibility option. It’s hard for a business user to analyze failed rule resolutions with this type of rule. In concept, this approach is like the one using an expression at the workflow step level for rule determination, where you use a background task to encapsulate the logic to determine the agents for a subsequent dialog step. Generally, this type of agent determination rule is recommended for complex logic or where you already have standard or custom tables from where the agent may be determined, such as a rule to determine the cost center responsible person, a rule to determine the work breakdown structure (WBS) responsible person, or a rule to determine the approver for the purchase order release strategy.

Note

If the agent determination logic is complex with multiple levels of calculation or cross-referencing, then it may be better to add the logic in a background task, instead of a rule, which is directly assigned to a workflow step. Adding the logic in a background method gives you better control over raising proper meaningful exceptions at each stage of failure, which could be very useful during analysis of the exception via the workflow log. Rules with function modules have only one exception (mentioned in the following table), which doesn't provide any details on what failed exactly.

For creating a rule using a function module, choose the category as **Function to be executed**, as shown earlier in Figure 6.10. Enter a function module name that has the interface detailed in Table 6.3.

Parameter Type	Name	Data Type Reference	Meaning
Table	AC_CONTAINER	SWCONT	Importing rule container
Table	ACTOR_TAB	SWHACTOR	Output multiline agent list
Exception	NOBODY_FOUND		Exception for no agents found

Table 6.3 Interface Definition of a Rule Function Module

Next you need to create container elements in your rule for the importing parameter to the rule. Inside the function module logic, you'll be reading the container elements that will be used as agent determination criteria. After executing the logic, the agents must be returned in export parameter ACTOR_TAB with line structure SWHACTOR, which means that all agents must have the **Object Type** prefix (e.g., US for user, O for organizational unit, S for position, etc.), along with the corresponding **Object ID**.

Raise exception NOBODY_FOUND when no agents can be determined through the required business logic. This exception, in conjunction with the **Terminate If Rule Resolution W/o Result** setting, will enable you to terminate the workflow in error state if the agent determination fails.

Let's consider the example where we're trying to determine the approvers for a sales invoice release workflow. The approvers are maintained in a set of custom tables based on sales document type (AUART), billing type (FKART), sales organization (VKORG), and distribution channel (VTWEG). In addition, each approver has threshold amount beyond which they need to approve the invoice.

Listing 6.1 contains sample code for the agent determination rule using a function module.

```
FUNCTION z_invoice_approvers .
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  TABLES
*"      AC_CONTAINER STRUCTURE  SWCONT
*"      ACTOR_TAB STRUCTURE  SWHACTOR
*"  EXCEPTIONS
*"      APPROVER_NOT_FOUND
*"----------------------------------------------------------------------
```

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

  DATA : lt_holders TYPE STANDARD TABLE OF swhactor, " Rule Resolution Result
         ls_holders TYPE swhactor,                   " Rule Resolution Result
         lv_auart   TYPE auart,                      " Sales Document Type
         lv_vkorg   TYPE vkorg,                      " Sales Organization
         lv_vtweg   TYPE vtweg,                      " Distribution Channel
         lv_netwr   TYPE netwr,              " Net Value in Document Currency
         lv_fkart   TYPE fkart.                      " Billing Type

  CONSTANTS : lc_otype TYPE otype VALUE 'US'. "Agent type

  swc_get_element ac_container 'ZWCAUART' lv_auart. "Sales order type
  swc_get_element ac_container 'ZWCVKORG' lv_vkorg. "Sales Org
  swc_get_element ac_container 'ZWCVTWEG' lv_vtweg. "Distribution Channel
  swc_get_element ac_container 'ZWCNETWR' lv_netwr. "Net value of billing
  swc_get_element ac_container 'ZWCFKART' lv_fkart. "Billing Type

*  Reading the approver list comparing sales order type, billing type
*  sales org, distribution channel, net amount of billing

  SELECT zotc_inv_acct~auart,                " Sales Document Type
        zotc_inv_acct~fkart,                 " Billing Type
        zotc_inv_acct~vkorg,                 " Sales Organization
        zotc_inv_acct~vtweg,                 " Distribution Channel
        zotc_inv_acct~approver_group,        " Approver Group
        zotc_inv_acct~bname               " User Name in User Master Record
        INTO TABLE @DATA(lt_approverlist)
        FROM zotc_inv_acct          " Threshold amount for Inv Release
INNER JOIN zotc_inv_apprvr     "Invoice Release to Accounting Approver
         ON zotc_inv_acct~approver_group = zotc_inv_apprvr~approver_group
         WHERE auart = @lv_auart              " Sales Document Type
           AND fkart = @lv_fkart              " Billing Type
           AND vkorg = @lv_vkorg              " Sales Organization
           AND vtweg = @lv_vtweg              " Distribution Channel
           AND threshold_amount LT @lv_netwr. "Net value of Billing doc
  IF sy-subrc <> 0.
    CLEAR lt_approverlist.
  ENDIF. " IF sy-subrc <> 0

  CLEAR : lt_holders,
          actor_tab.

* Filling the agents in actor tab
  lt_holders = VALUE #( FOR <ls_approverlist> IN lt_approverlist
                        otype = lc_otype
                      ( objid = <ls_approverlist>-bname ) ).

  APPEND LINES OF lt_holders TO actor_tab.

  DATA(lv_num_lines) = lines( actor_tab ).
  IF lv_num_lines IS INITIAL.
```

```
    RAISE approver_not_found.
  ENDIF. " IF lv_num_lines IS INITIAL

ENDFUNCTION.
```

Listing 6.1 Sample Code for Agent Determination Rule Using a Function Module

Sometimes, it may be required to execute multiple rules within your rule function module, depending on different scenarios and output from the agents in your rule. For example, let's say you've developed a rule for a purchase order release, where the approvers may be the cost center responsible person or the WBS responsible person, depending on whether the purchase order item is linked to a cost center accounting object or a WBS accounting object. You already have standard rules available that determine the cost center responsible person or WBS responsible person. You can now create a function module that first checks the account assignment category (EKPO-KNTTP) of your purchase order item. If this is a cost center (K), then execute the cost center person responsible rule and get the agents; if it's a WBS element, then execute the WBS person responsible rule and get the agents.

Listing 6.2 contains sample code illustrating how to execute a rule to get the agents.

```
swc_set_element lt_container 'PONumber' lv_ebeln.

CALL FUNCTION 'RH_GET_ACTORS'
  EXPORTING
    act_object               = 'AC00800046'
  TABLES
    actor_container          = lt_container
    actor_tab                = lt_actor_tab
  EXCEPTIONS
    no_active_plvar          = 1
    no_actor_found           = 2
    exception_of_role_raised  = 3
```

```
    no_valid_agent_determined = 4
    no_container              = 5
    OTHERS                    = 6.
```

Listing 6.2 Sample Code Showing How to Execute a Rule to Determine the Agents

One final note on using evaluation paths in rules: If you have an organizational plan implemented via standard transactions that you want to reuse to determine the agents for your workflow, then you can use this option, which is a variant of the agent determination with function module approach. An evaluation path is a sequence of relationships implemented in an organizational structure in HR. Each evaluation path is identified by an ID. The input to the evaluation path is usually a user ID or a person ID, and the output is a list of organizational objects such as organizational units, positions, or user IDs.

You can use standard program RHWEGID00 to find suitable evaluation paths based on your input and desired output object types. Some commonly used evaluation paths in SAP are listed in Table 6.4.

Evaluation Path	Description
WF_ORGRUN	Find the organizational unit of a user ID or person
US_CHEF	Superior of a user
SAP_HOLD	Holder of a position
MANAGER	Manager of a user

Table 6.4 Some Common Evaluation Paths Available in Standard SAP

To create a rule with an evaluation path, you must enter the function module as “RH_GET_STRUCTURE”. When you press Enter, you’ll be prompted to enter the evaluation path in a new field. Figure 6.17 shows this example from Transaction PFTC.

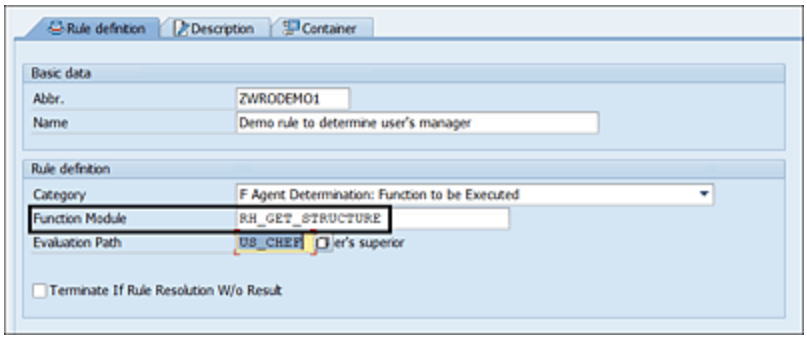


Figure 6.17 Rule with a Function Module Using an Evaluation Path

Next, you must create the rule container elements for passing the input user ID or person ID to the rule in the same screen as shown in Figure 6.11 earlier, which are detailed in Table 6.5.

Element	Data Type	Description
OTYPE	OBJEC-OTYPE	Type of the organizational management object
OBJID	OBJEC-REALO	ID of the organizational management object
ORG_AGENT	WFSYST-AGENT	Organizational management object

Table 6.5 Container Elements Required for a Rule Using an Evaluation Path

You can pass the input organizational object by filling either the `OTYPE` and `OBJID` or `ORG_AGENT` parameters.

Agent Determination with Organization Data

In this approach, you can create a rule with the category as **Agent Determination: Organizational Data**. Here you can link the code of a custom business attribute to an organizational object in the organizational structure via a business object. For example, if you have a Customizing table for a customer service center, you can link the service center code with an SAP organizational unit and determine all the users under the organizational unit as service center technicians. There are some typical configuration and development steps required for this link (see Figure 6.18):

1. Create a custom business object from Transaction SWO1 to link your custom attribute code to the organizational object. The key field of the business object should be the same attribute code. In addition, you need to classify the business object as **Organizational type** under header data on the **General data** tab.
2. Maintain the organizational object types (**Organizational unit**, **Position**, **Jobs**, **Persons**, etc.) that may be linked to your custom business object instances.
3. Maintain the assignment of organizational object IDs to your custom business object via Transaction PFOM.
4. Create your custom rule with the type as **Agent Determination: Organizational Data**, and enter your custom business object type name in **OrgObj type**.

5. To map the data of the custom attribute from your workflow to your rule, you need to create an instance of your custom business object from your workflow and bind that element to your rule container element.

The assignment of organizational object to your custom attribute is saved in HR Infotype 1208.

The output of the rule will be all the users maintained against the organizational object assigned to your custom attribute.

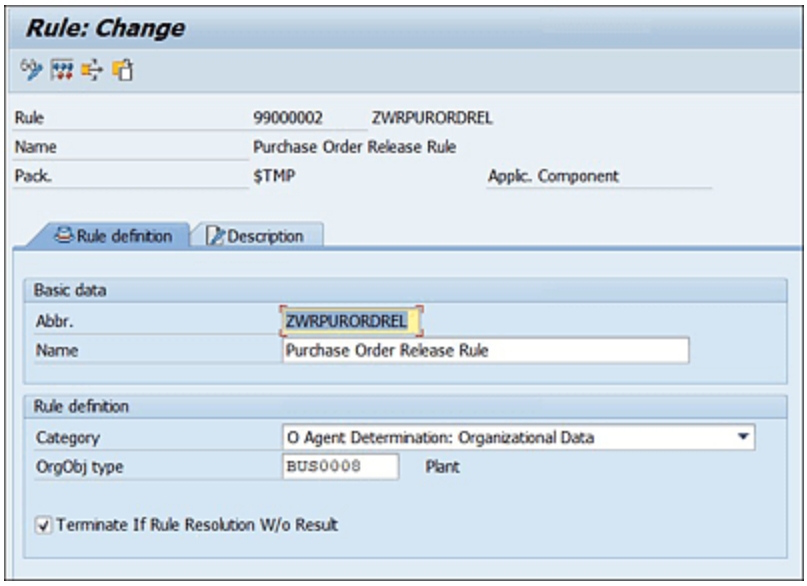


Figure 6.18 Rule Definition Using Organization Data Approach

Note

Agent determination organization data isn’t a recommended approach for creating new agent determination rules anymore. This is because it involves creating custom business objects and involves quite a bit of maintenance to keep the Customizing table entries and the organizational structure in sync. It’s preferable to use

other approaches using HR organizational structure data such as evaluation path or even a custom table with a function module–based rule in this case.

6.3 Possible Agents in Tasks and Default Rules

Possible agents may be maintained at the task definition level using menu path **Additional Data** • **Agent Assignment** • **Maintain**. The first step is to maintain the agent assignment attributes, as shown in Figure 6.19.

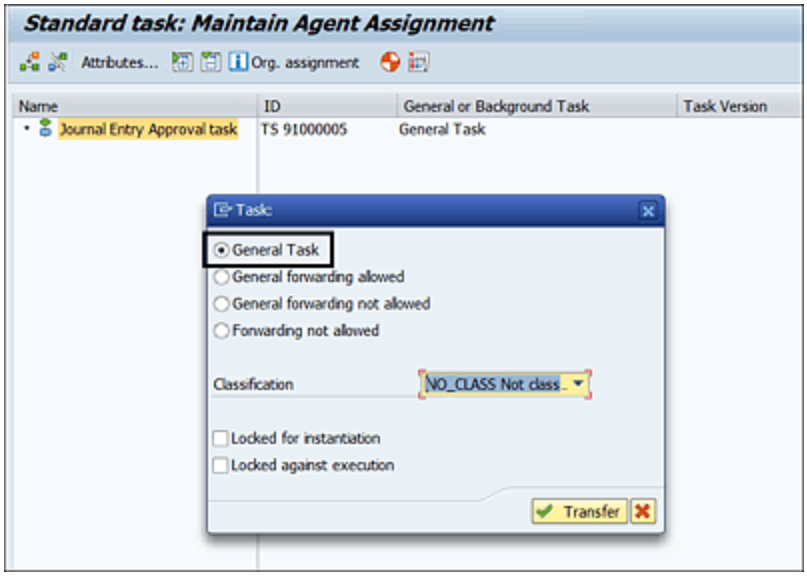


Figure 6.19 Maintain Agent Assignment Attributes at the Task Level

If you want to keep the possible agents open to all SAP users in the system, then you can choose the **General Task** radio button, as shown in Figure 6.19. In that case, you don't need to proceed to the next step. If, however, you want to allow only a specific group of users to be able to execute a particular task, then choose one of the other three radio buttons:

- **General forwarding allowed**
  This option allows you to maintain an agent assignment

for possible agents. However, any user who has received the work item for this task in their inbox will be able to forward the task to any SAP user in the system. So, with this option, the restriction with possible agents is lost at the time of forwarding.

- **General forwarding not allowed**
  This option restricts the general forwarding of the work items for this task and forces the user to respect the agent assignment done at the task level, which means that you can only forward the work item to other users who are within the possible agents group.
- **Forwarding not allowed**
  This option restricts forwarding completely. Work items for this task may only be executed by one of the users who have received it in their inbox.

Once you select the agent assignment attribute as a nongeneral task, you can create an agent assignment by clicking on the **Create Agent Assignment** button, as shown in Figure 6.20.

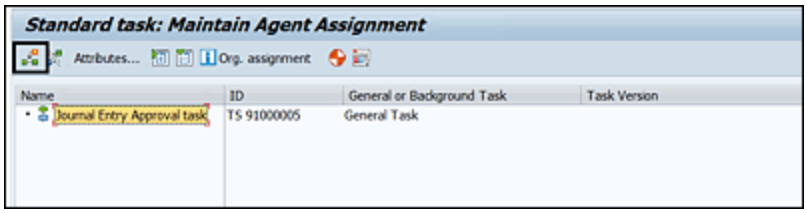


Figure 6.20 Create Agent Assignment

In the next screen in Figure 6.21, you get the option to select any one of the agent types. Here you can choose **Role**, to tie the possible agents to a security role, or you can choose **Job**, **Organizational unit**, **Work Center**, or **Position** to tie the agent assignment with the HR

organizational structure objects. The option to maintain user IDs directly isn't recommended for a productive workflow scenario as this reduces flexibility in your workflow design.

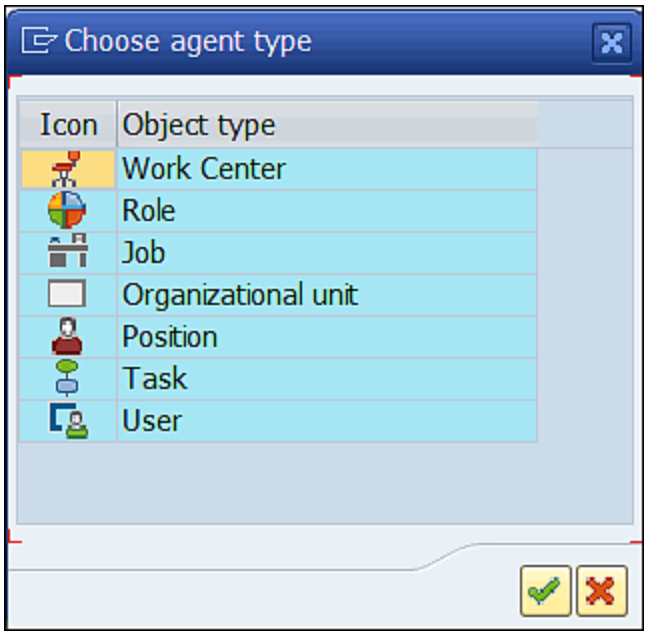


Figure 6.21 Popup to Select the Type of Agent

Once you've selected the agent type, you need to select the agent type object. For example, if you've selected **Role** as the agent type, then in the next screen in Figure 6.22, you must search and select a security role to assign as your possible agents for the task definition.

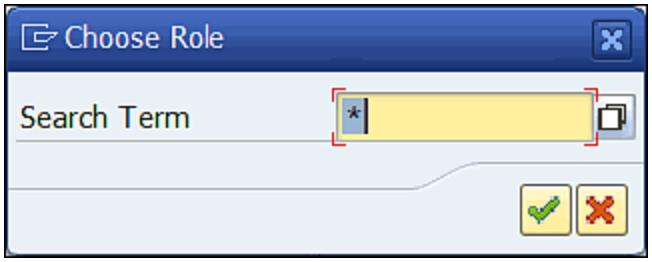


Figure 6.22 Popup to Select Role

Returning to the topic of default rules, they are always defined at the task definition level under the **Default rules**

tab in Transaction PFTC. There can be three different kinds of default rules:

- **Agent rule**
  This rule is used to determine the responsible agents for a task.
- **Deadline recipient rules**
  This type of rule may be defined to determine the agents for missed deadlines related to latest start, latest end, or request end.
- **Recipient for completion message**
  This type of rule is used to determine the email address of the user who should receive the message on completion of a work item.

Figure 6.23 shows the **Default rules** tab under the task definition in Transaction PFTC. You need to enter the **Rule** number for the appropriate default rule type. Recipients for missed latest start, missed latest end, and missed requested end are all grouped under deadline recipient rules.

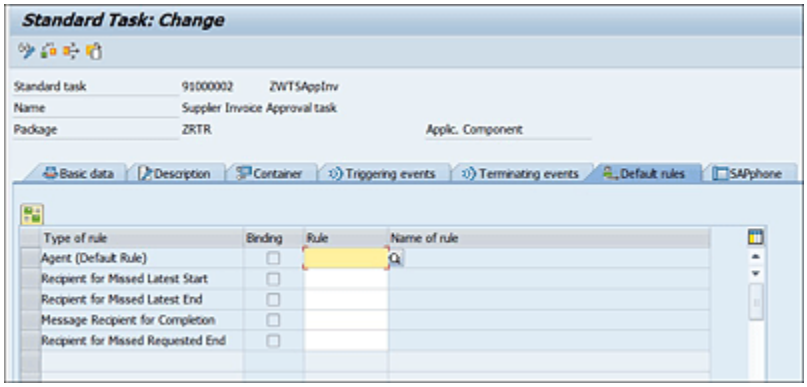


Figure 6.23 View of Default Rules under Task Definition in Transaction PFTC

All three kinds of rules have the same structure and options as that of the rule to determine responsible agents, which was detailed in Section 6.2. You only need to consider the following points while maintaining a default rule:

- Default rules have secondary priority compared to any agent determination maintained at the workflow step level for a multistep task. For single-step tasks, they are the only mechanism for maintaining responsible or deadline agents. For multistep tasks, they come into play only when the agent determination fails at the workflow step level or if there is no agent maintained at the workflow step level.
- If you've maintained a rule at the workflow step level, which has the **Terminate if rule resolution w/o Result** checkbox selected, then failure in rule resolution won't trigger the default rule.
- Binding must be maintained between the task container elements and rule container elements in this case.
- Agents determined through the default rule should also respect the possible agents assigned to the task definition. Any responsible agent determined via the default rule who isn't a possible agent won't be able to receive the work item in their inbox.

6.4 Organizational Structure Definition and Linking to Workflow Agents

Using the organizational plan to assign responsible agents to a workflow step is another common approach for agent determination. Usually, we don’t want to directly assign user IDs as agents in a workflow step because users change, move across departments, or join and leave a company. So, to keep your workflow design flexible and avoid making frequent changes to your agent determination process, it’s advisable to use the organizational structure for assigning the agents wherever possible. You can use the basic organizational plan in a system even without HR. If you do implement HR in your system, then you also have access to the extended organizational plan with some more features.

With an organizational plan, you can create or maintain organizational units, representing your business units or departments. You can link one organizational unit to another organizational unit, establishing a hierarchy and maintaining relationships. You can assign one or more positions under an organizational unit, establish relationships between them (e.g., manager and reportees), and assign holders to these positions via persons and/or users. You can also assign jobs to positions, which helps you define and group responsibilities and roles. In addition, you can delimit or delete any organizational object or relationship in the plan.

There are various transactions available to maintain the organizational structure in SAP, for example, Transaction PPOCW, Transaction PPOMW, Transaction PPOME, Transaction PPOSE, and so on. Figure 6.24 shows the view of a typical organizational structure definition in SAP with organizational units, positions, persons, jobs, and users.

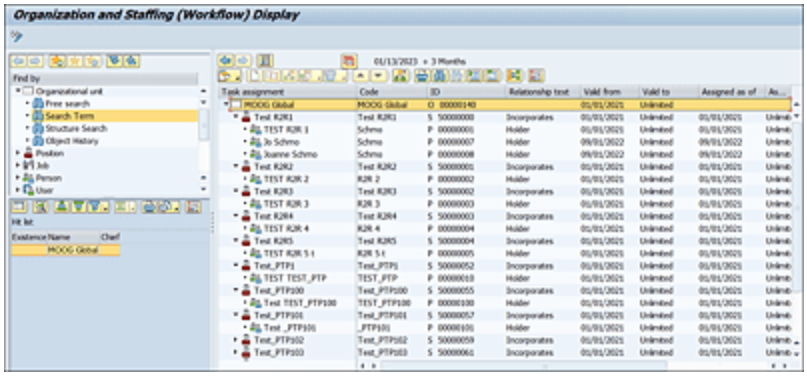


Figure 6.24 View of an Organizational Structure in SAP S/4HANA with Organizational Units, Positions, Persons, and Users

Some common recommendations and best practices in the use of organizational structure with workflow are as follows:

- Users may be linked via organizational objects to workflow agent assignment steps. For example, you can assign an organizational unit as the agent to a workflow user decision step, which would automatically pull all the users assigned to all the positions under that organizational unit as agents. Although the organizational unit or the position may be hardcoded as an agent, it’s better to use expressions or rules where the organizational object is returned as an agent dynamically based on some business attribute or relationship.
  For example, in a workflow to approve journal entry requests, you may have an organizational structure set up based on the controlling area and the profit center, and

you can read the profit center on the journal entry document and link it to an organizational unit in your plan so that all the users linked to the unit may be determined as agents.
Consider another example for approving leave requests for an employee. Here the employee submitting the leave request is linked to a position via the organizational plan, and the manager relationship of that position is determined and sent as the approver of the leave request. The holder of the manager position receives the work item in his inbox.

- Organizational objects may be assigned a validity, which is by default set from the creation date to 12/31/999. However, this validity may be updated to reflect temporary relationships, or the validity may be terminated on a certain date as part of a delimit action.
- Many standard evaluation paths are available to determine different relationships between organizational objects. These evaluation paths may be used via the rule with function module option (see Section 6.2.4) or evaluated via custom logic in a background task.
- Organization data may also be integrated into the agent assignment step via the **Rule with Organization Data** option where you can link an attribute code from a Customizing table to an organizational object from Transaction PFOM via a business object.
- Organizational objects may also be integrated into possible agent assignments at the task definition level. But it’s better to use security roles as an alternative in this case because that gives you the option to keep the

underlying transaction authorizations in sync with the job role of the people who can execute the task.

6.5 Summary

In this chapter, you've learned about the different types of agents in workflows. You now understand the difference between possible agents, responsible agents, and excluded agents, as well as the concept of selected agents and actual agents. We covered the details of agent determination rules and explored the different approaches used to build a rule in SAP Business Workflow. Then, we looked at possible agent definitions in your task definition and defining default rules in tasks in more detail. Finally, we talked about organizational structures in SAP and how to integrate organizational objects into your workflow agent determination step.

7 Email Notifications and Other Runtime Jobs

SAP Business Workflow uses different types of notification mechanisms for business users. In this chapter, you'll see how these notifications are configured and implemented in different workflow scenarios.

SAP email notifications are automated messages sent by the system to specified recipients to communicate information about various events or status updates. SAP email notifications play a vital role in optimizing business processes and improving overall productivity. Automated email notifications keep stakeholders informed in real time. There are different email notifications scenarios such as job status notifications, error notifications, workflow notifications, and so on.

However, in this chapter we'll talk about notifications related to workflows. In SAP Business Workflow, you generally use three types of notifications, as follows:

- **Regular notifications**
  These are sent to the performer of a task when the task is created. They are used to notify the receiver about the new task in their work inbox.

- **Reminders**
  These are emails sent to the performer of a task a short time before the end date or the due date of the task if the task isn’t completed.
- **Escalations**
  These are notifications sent to the performer’s supervisor or to another responsible person if the performer misses the task completion due date, as specified in their work inbox.

In this chapter, we’ll discuss in detail the prerequisites, configurations steps, and other batch jobs to achieve these workflow notifications.

7.1 Prerequisites for Setting Up Email Notifications in Workflow

There are some configurations that need to be performed as part of the email notifications setup. Before that, we’ll discuss two prerequisite activities and how to perform them: setting up email IDs for SAP users and setting up SAPconnect.

7.1.1 Setting Up Email IDs for SAP Users

Notifications are sent to external email IDs of SAP users; the email ID is retrieved from the user master record. Therefore, it’s necessary that the corresponding SAP users have the email IDs maintained in the user master record. To check the email ID of an SAP user, first open Transaction SU01 and

provide the user ID. Then display the record where you find the details of the user attributes along with the email ID, as shown in Figure 7.1.

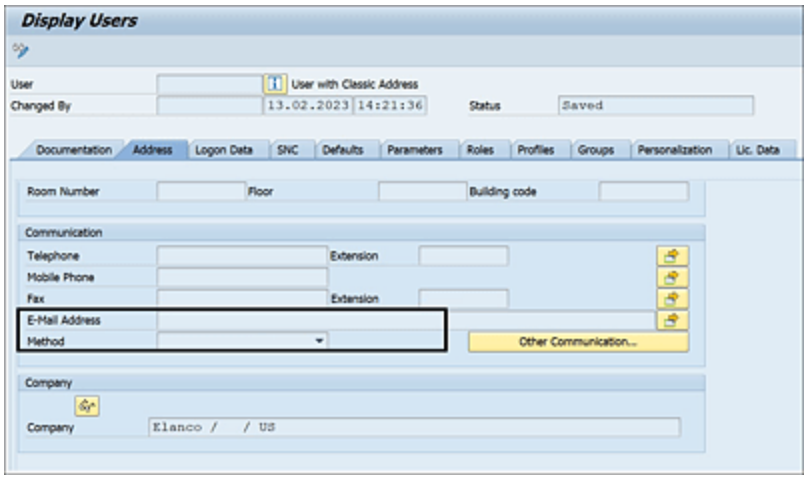


Figure 7.1 Email Address to Be Maintained in the User Master

7.1.2 Setting up SAPconnect

In Transaction SCOT, you configure the communication interface SAPconnect, which allows you to send external emails from the SAP system. You'll need to make the following settings:

1. Select **Settings** • **SMTP Connection** • **Outbound Messages** • **SMTP Nodes**.
2. Double-click on **SMTP**, as shown in Figure 7.2.
   The **SAPconnect: General Node Data of Node SMTP** screen appears, as shown in Figure 7.3.
3. Select the **Node in use** checkbox.
4. Enter the SMTP **Mail Host** name.
5. Enter the SMTP **Mail Port** number (usually, 25).

6. Click on the **Set** button next to the **Internet** option.
7. Click on the checkmark button to proceed.

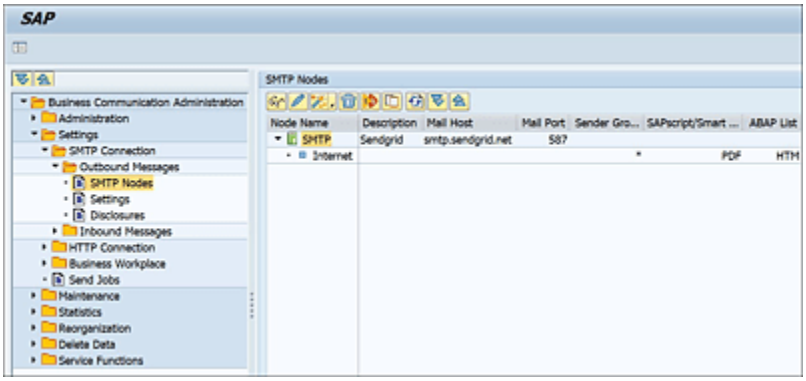


Figure 7.2 Selecting SMTP Node

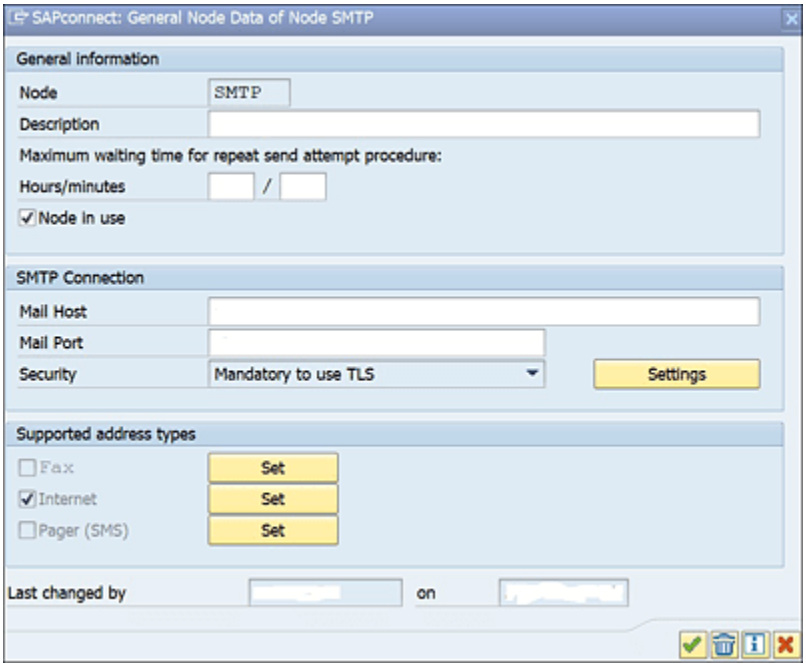


Figure 7.3 Setting Up SMTP Node

8. On the subsequent **SAPconnect : Address Type for Node** screen, enter details in the **Address areas** section (see Figure 7.4). This specifies to which address (domains) the emails can be sent. “*” means emails can

go to any email addresses, while *gmail.com means the emails can go only to users in the gmail.com domain.

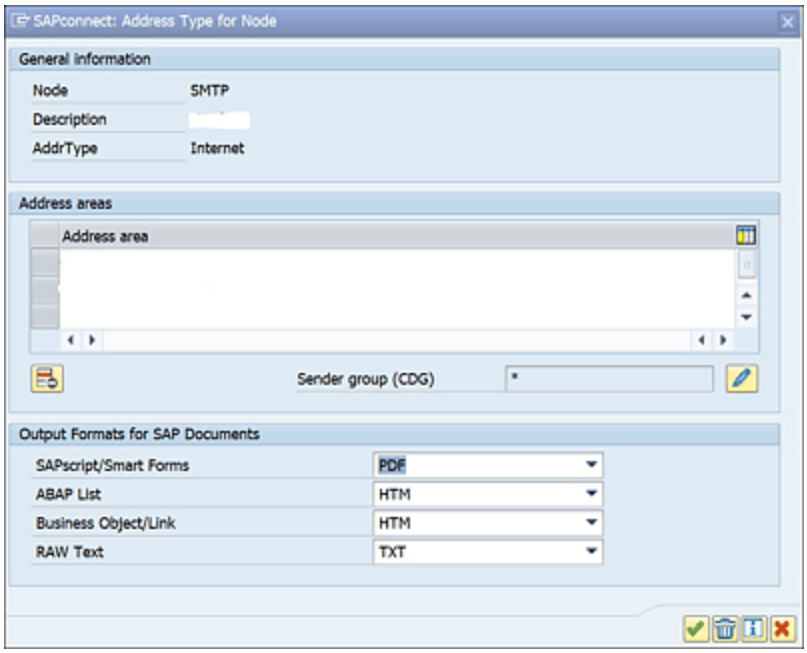


Figure 7.4 Transaction SCOT Configurations: Further Settings

Tips

It’s always a security best practice to restrict the email addresses to some domain instead of making it “*”. Generally, a company domain is used along with some specific business partners’ domains.

Next, we’ll create the send jobs and schedule the emails to be sent on a periodic basis.

7.2 Classical Work Item Email Notifications via Program RSWUWFML2

There are business requirements where you need to send notifications to external email IDs to process the work items to the corresponding users as all business users don’t use SAP inbox on a regular basis. Program RSWUWFML2 sends notifications for work items by email to SAP users who have an internet email address. You can also send SAP shortcut attachments using this report. These enable the recipient of the email to display or execute the work item directly, or to open Business Workplace in SAP GUI for Windows.

For a periodically scheduled run (not a single run), the report generates notifications for work items for which the following applies:

- The work items must have the status **Ready**.
- The work items have been created since the last run (the time and date of the last run is saved in a table).
- The work items have been changed since the last run by one of the following operations: forward, replace, end of resubmission, or requested start reached.
- The work items have a recipient with a valid internet email address.

Figure 7.5 shows the details of the selection screen of program RSWUWFML2.

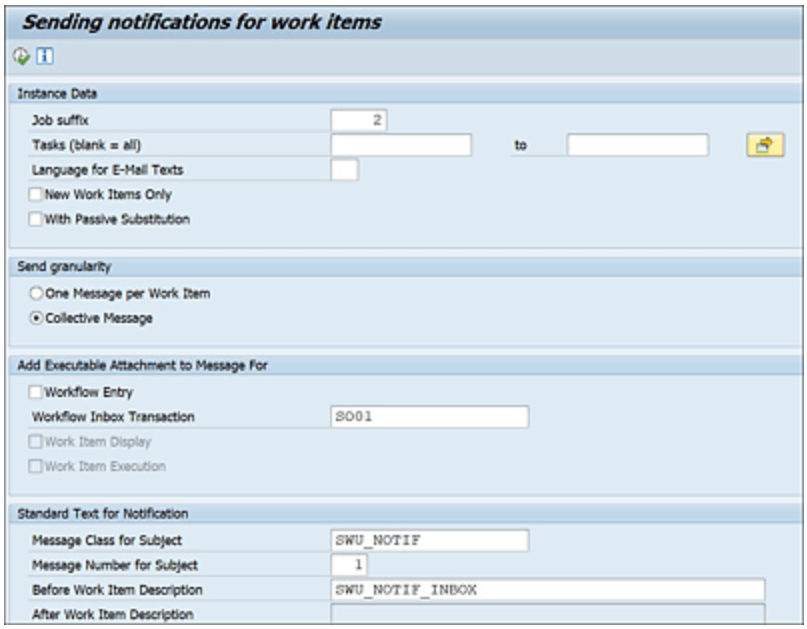


Figure 7.5 Program RSWUWFML2 to Send Notifications for Work Items

One advantage of program RSWUWFML2 is that users can be notified easily who don't normally use Business Workplace of a particular SAP system because they use another email client or another SAP system. Workflow patterns or tasks don't need to be modified. The generation of notifications doesn't need to be modeled in SAP Business Workflow.

On the other hand, there are some restrictions when it comes to program RSWUWFML2. The report excludes work items that a workflow inbox has reassigned at a later stage due to changes in the organizational structure. Position substitutions are also not included. This means that the report doesn't send any notifications for work items that appear in the workflow inbox of an agent due to a position substitution.

Normally, SAP office messages from the *Documents* folder aren't forwarded by the report. Document and object

attachments of a work item are also not sent. The messages are generated in text-only format.

There is no password caching for shortcuts. If single sign-on isn’t active, the user needs to log on to the system separately. A user can be informed about multiple work items in a collective notification. However, the collective notification doesn’t contain details about the individual work items. No container variables or container expressions can be used in the descriptive texts (before work item description, after work item description).

The language settings of the work item recipient aren’t taken into consideration when creating the message. The generated texts sent to the recipients are based on the **Language for E-Mail Texts** parameter given in the selection screen of report RSWUWFML2. Messages can only be sent to Simple Mail Transfer Protocol (SMTP) addresses (type INT) and groupware-specific forms aren’t supported.

Before we dive into using program RSWUWFML2, there are some prerequisites. The recipients of email notifications must have maintained an email address in the relevant SAP system. You can maintain a user’s email address in either the central address management (Transaction SU01) or the personal office settings in Business Workplace, under **Automatic Forwarding**. We no longer need to activate the mail group in the general office settings for program RSWUWFML2. In addition, the SAPconnect (Transaction SCOT) setup needs to be completed that we discussed in the previous section.

Let’s now discuss all the features and controls that you can use for different parameters in the selection screen of this

program.

In the **Instance Data** section (see Figure 7.6), you have the following fields:

- **Job Suffix**
  Using a job suffix enables you to have multiple instances of the report running simultaneously. Note that the times for the last run are saved for the individual instances. You can set the report parameters differently here.
- **Tasks (blanks = all)**
  You can restrict the selection of work items to work items for tasks. You can select multiple tasks here.
- **Language for E-Mail Texts**
  You can specify the language in which the work item description and the standard texts for the message are to appear in the emails. The prerequisite is that the corresponding translation is available.
  If no language is specified, the work item description is displayed in the original language of the work item and the standard texts in the language of the user where the report is scheduled.
- **New Work Items Only**
  This option enables you to deactivate the inclusion of changed work items so that the same old messages aren’t sent repetitively. This is recommended for performance reasons.
- **With Passive Substitution**
  This option enables you to define whether the report includes passive substitution. The inclusion of passive substitution will decrease performance significantly.

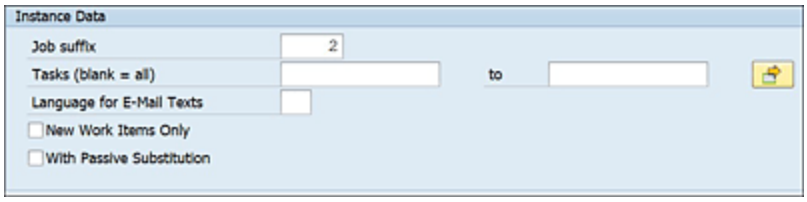


Figure 7.6 Instance Data Section of the Selection Screen

There are only two options in the **Send granularity** section (see Figure 7.7). If you change this setting, then the default settings in the lower area of the selection screen are changed dynamically:

- **One Message per Work Item**
  One message is sent for each new work item. This contains the description of the work item and accompanying texts, as specified under **Standard Text for Notification**.
- **Collective Message**
  Only one message is sent for all new work items for a particular user. The message doesn't contain any information about the individual work items.

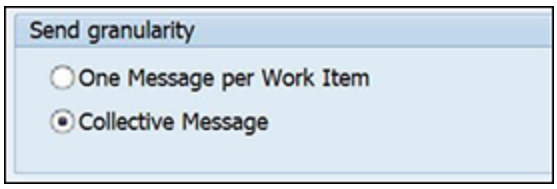


Figure 7.7 Send Granularity Section of the Selection Screen

Executable attachments are SAP shortcuts that receivers of messages can use to directly display a work item in the SAP GUI for Windows. SAP shortcuts contain the relevant logon language from the central address management (Transaction SU01) for each individual receiver. The system ID and the client contained in the shortcut correspond to the

system in which the report is running. In the **Add Executable Attachment to Message For** section (see Figure 7.8), you have the following fields:

- **Workflow Inbox Transaction**
  An SAP shortcut is added to the message so that the message recipient can start Business Workplace directly in SAP GUI for Windows.
- **Work Item Display**
  An SAP shortcut is added to the message so that the message recipient can display the work item directly in SAP GUI for Windows.
- **Work Item Execution**
  An SAP shortcut is added to the message so that the message recipient can execute the work item directly in SAP GUI for Windows. Secondary methods are also executed at the same time.

Figure 7.8 Add Executable Attachments to Message Section of the Selection Screen

The **Standard Text for Notification** section (see Figure 7.9) enables you to define the message subject and text, which is added to the work item description. The text contains general information for the user about what to do with the message. Here you have the following fields:

- **Message Class for Subject**
  Message class of a table `T100` message (Transaction SE91)

from which the subject line for the message is determined. Message class `SWU_NOTIF` contains the messages that are shipped by SAP.

- **Message Number for Subject**
  Number of a `T100` message (Transaction SE91) from which the subject line for the message is determined. The `T100` message can contain two parameters. Parameter `&1` is replaced by the system ID. Parameter `&2` is replaced by the work item text.
- **Before Work Item Description**
  ID of a SAPscript text (Transaction SE61) of type **Text in Dialog**, which is inserted before the actual work item description in the message. The text contains the form of address for the user and the reference to SAP Business Workflow. Make sure that the text matches the send granularity. The system proposes an appropriate text. Includes and symbols within a SAPscript text are expanded. Workflow-specific variables or expressions (e.g., from the container) can't be used, however.
- **After Work Item Description**
  ID of a SAPscript text (Transaction SE61) of type **Text in Dialog**, which is inserted after the actual work item description in the message. Make sure that the text matches the send granularity. The system proposes an appropriate text.

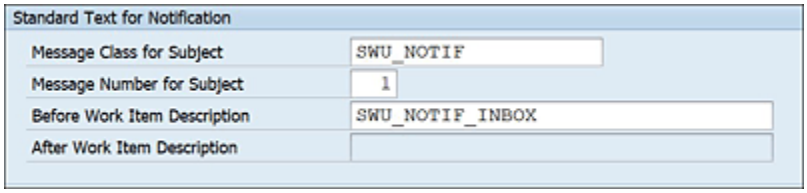


Figure 7.9 Standard Text for Notification Section of the Selection Screen

In the **SAP Shortcut Parameter** section (see Figure 7.10), you can specify the ID of an entry in the SAP Logon, which is then included in every shortcut that is sent in the `Description` parameter. This parameter determines the SAP Logon entry with which the logon to the SAP system is executed. The prerequisite for this is that you can guarantee that *saplogon.ini* contains the relevant entry on all user PCs. The parameter is only required and recommended if there are multiple *saplogon.ini* entries for the same system on the user PCs. This enables you to control a specific entry, for example, the entry that is configured for single sign-on (SSO).

Figure 7.10 Section SAP Shortcut Parameter of the Selection Screen

In the **Exits** section (see Figure 7.11), you can save function modules you've created yourself that are called at an appropriate point during the report run. The modules must have a specific signature. SAP Note 808971 describes this in more detail. You have the following two fields:

- **FM for Preparatory Phase**
  This module is called prior to the selection of the work items. You can use this module to assign values to text symbols in the SAPscript texts, for example.
- **FM for Address Determination**
  This module is called when the email address of a work item processor is to be determined. You can use this module to override the default procedure for determining

an address, for example, to determine an address from a container element of the work item.

Figure 7.11 Section Exits of the Selection Screen

Let’s now move on to the **Data for an Individual Run (Time Stamp Not Set)** section (see Figure 7.12). A single (individual) run is performed if the input fields contain a data, time, and optionally a user. A single run is recommended if work items are to be sent again from a particular time and date, for example, due to a problem when sending the emails. Contrary to a periodically scheduled run, in a single run, the time of the last run isn’t saved. Here we have the following fields:

- **From work item creation date**
  This is the first day when work items are to be included.
- **From work item creation time**
  This is the time on the first day when work items are to be included.
- **User (blank = all)**
  Notifications are only created for work items that this user has in their Business Workplace. You must not use this option to restrict the sending of notifications in the productive environment to certain users.

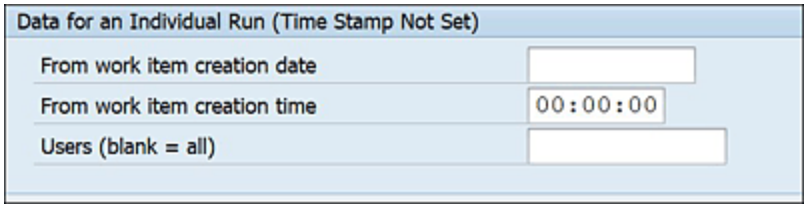


Figure 7.12 Data for an Individual Run (Time Stamp Not Set) Section in the Selection Screen

During its run, the report writes a log in a spool file and an entry in the application log. You can specify what is logged in the **Log** section (see Figure 7.13):

- **Errors Only**
  Only error messages are logged.
- **All**
  Success and error messages are logged.

You can display the application log by calling Transaction SLG1. Enter the **Object** as “WF” and the **Subobject** as “NOTIFICATIONS” as the selection values.

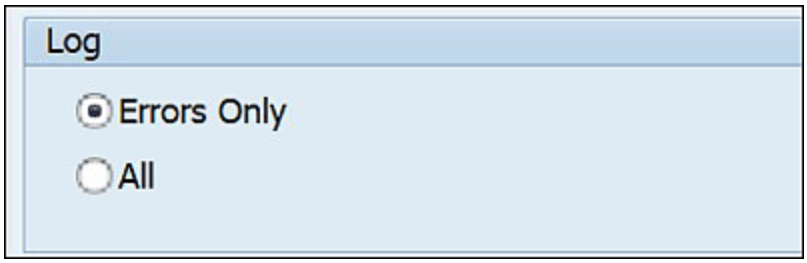


Figure 7.13 Log Section of the Selection Screen

Now let’s see some other general features of this report. If one message is sent for each work item, the message contains the task description and the standard text. If one message is sent for all work items, the message contains just the standard text. Express emails are sent for express work items.

If the messages include executable attachments, they are sent to each user individually. If executable attachments aren’t required, all messages that refer to the same work item are sent to a receiver list, which comprises the

selected agents. If you've selected a **Collective Message**, only one message is sent to each receiver per run, regardless of the number of work items for this user.

The results of database queries (e.g., email addresses) are buffered to improve system performance.

When the email messages are sent to users' inboxes, a shortcut for SAP Logon can be sent as an attachment in this notification message from report RSWUWFML2. The usual logic when using SAP shortcuts applies when logging on to execute shortcuts. If the parameter `Description` is specified in the SAP shortcut, but there is no *saplogon.ini* entry with this name, the system displays an error message when you attempt to execute the shortcut. If the parameter `Description` isn't specified, a *saplogon.ini* entry with the appropriate system is selected instead. The prerequisite for this is that the entries in SAP Logon were created from the group or server selection. Otherwise, the system ID can't be determined from the entry.

If a session with the relevant entries for system/client/user/language is already open, a new session is opened automatically without the user needing to log on. If no such session is already open, the user must log on again. If the selected *saplogon.ini* entry is configured for SSO, the system logs the user on automatically. If it's not configured, the system displays a logon window in which the user must enter their password.

Next let's see some main points related to selection of work items and recipients in this report. Report RSWUWFML2 determines the following internally:

1. The report determines the relevant work items. Dialog work items and missed deadline work items are included. Changes to the organizational structure aren’t included.
2. The recipients for each work item are also determined. Active and passive substitutions are included, but position subscriptions aren’t. Substitute profiles and task classifications are included.
3. Only receivers who have an internet email address (type `INT`) are included.
4. Create the message body (partly work item–specific).
5. Send the message to all receivers.

Standard Variants

Variants can be defined for scheduling. If you use multiple variants, you must at least use the job suffix to differentiate between them.

Executing report RSWUWFML2 generates a log file and an application log, as shown in Figure 7.14.

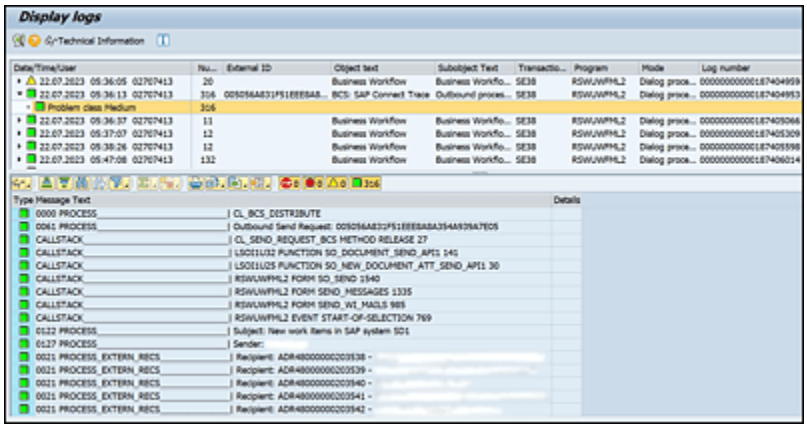


Figure 7.14 Sample Application Log Output of Program RSWUWFML2

7.3 Extended Notification Configuration with Program SWNCONFIG

There is another way of sending workflow notifications to users: extended notification. In principle, it’s the same usage as notification program RSWUWFML2, but extended notifications provide more functionality and flexibility because they are a notification framework, which is based on ABAP handler classes. These handlers can be replaced by custom code.

The processing of extended notification is divided into two parts:

- Selection of work items and creation of notifications
- Delivery of the notifications via email or SMS

Extended notifications for SAP Business Workflow are intended to inform users about work items that need to be processed. Work items to be processed are dialog work items and deadline items. The users are notified in the form of messages. A message can be an email (HTML or plain text) or SMS. Next, we’ll talk about overview of this notification process and Customizing details of this process.

7.3.1 Overview of the Notification Process

As we just mentioned, the notification process via extended notifications for SAP Business Workflow comprises two

phases:

- **Selection**
  The relevant work items are selected, and notifications are created and stored in database table `SWN_NOTIF`.
- **Delivery**
  The notifications are selected from table `SWN_NOTIF` and messages are created and sent to the users.

This initially requires Customizing via Transaction SWNCONFIG (or Transaction SWNADMIN). Afterward, program SWN_SELSEN must be scheduled periodically. Program SWN_SELSEN processes both the selection and the delivery.

Let’s get an overview of the Customizing process, which we’ll discuss in detail in Section 7.3.2:

- **Selection**
  - For the selection, a filter pair must be defined. A filter pair consists of a full filter and a delta filter. One of the filters is marked as the main filter (usually, the Full filter). The filter pair is used to identify the relevant work items, so single-step tasks can be maintained at the filter pair.
  - In addition, a category must be assigned to the filter pair. A possible category could be, for example, **Leave Request** or **Shopping Cart Approval**. The notifications, which are created during the selection phase, are assigned to this category. The category is important for the delivery later.

- Each filter of the filter pair requires a selection schedule. The selection schedule contains the information when (weekdays, time, and interval) the filter must be processed.

- **Delivery**
  - For the delivery, a subscription must be defined. The subscription defines who should be notified about which work items and how the messages will look. The subscription must be assigned to a category, as discussed earlier. By doing so, the relevant work items for the delivery are chosen. In addition, the recipient(s), the delivery type (email, SMS), and the message structure are configured.
  - The subscription requires a delivery schedule. The delivery schedule contains the information when (weekdays, time, and interval) the subscription must be processed. This means when the messages should be delivered.

Program SWN_SELSEN must be scheduled periodically. Program SWN_SELSEN is responsible for the whole notification process. At first, it processes the selections and afterward the delivery, as follows:

- **Selection**
  The program reads the available selection schedules and analyzes which of them are due. The following procedure is done for each due selection schedule:
  - The assigned filter is retrieved.
  - Based on its assigned tasks, the relevant work items are selected. This means the work items, which were

created or changed since the last processing of the filter pair are selected.

- The agents of these work items are determined. One notification per work item and agent is created and stored in database table `SWN_NOTIF`.
- In addition, obsolete notifications are marked as logically deleted in table `SWN_NOTIF`. A notification becomes obsolete when the work item isn't available anymore in the inbox of its recipient, for example, due to work item execution or forwarding to another user.

- **Delivery**
  The program reads the available delivery schedules and analyzes which of them are due. The following procedure is done for each due delivery schedule:
  - The assigned subscription is retrieved.
  - Based on its assigned category and recipient, the relevant notifications from table `SWN_NOTIF` are selected.
  - The message content is created by using a Business Server Page (BSP) application. The settings of the subscription are considered for that.
  - The messages are sent to the recipients. The message is marked a `Delivered` in table `SWN_NOTIF`.

7.3.2 Detailed Customizing

The Customizing of extended notifications can be done either via Transaction SWNCONFIG or Transaction SWNADMIN. It's based on several database tables.

Transaction SWNCONFIG is a view cluster that contains the maintenance views for most of these Customizing tables. Due to this, Transaction SWNCONFIG is rather complex but offers all possible Customizing options. Due to the complexity of the Customizing tables and of Transaction SWNCONFIG, a simplified Customizing transaction was provided. Transaction SWNADMIN was created, which is a BSP. Transaction SWNADMIN doesn’t provide all the possible Customizing options, but it hides the complexity by generating Customizing entries.

Tips

When first trying to configure extended notifications, try via Transaction SWNADMIN, and then check the corresponding Customizing in Transaction SWNCONFIG.

The following sections show the required Customizing via Transaction SWNCONFIG.

Business Scenario

After executing Transaction SWNCONFIG, select business scenario **WORKFLOW** (business workflow), as shown in Figure 7.15. The scenario **WORKFLOW** is intended for the extended notifications for business workflow, which means for notifying the users about work items via email or SMS. All the other configurations will be done under business scenario **WORKFLOW**.

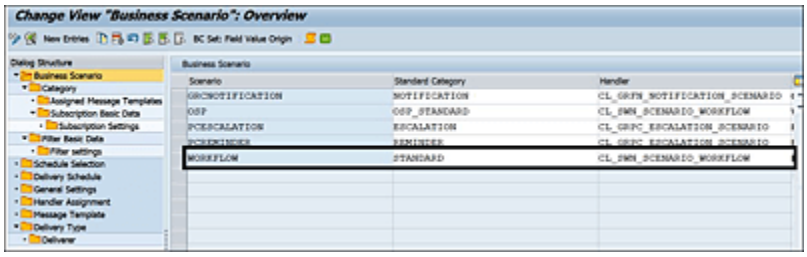


Figure 7.15 Business Scenario Workflow in Transaction SWNCONFIG

Category

From Transaction SWNCONFIG after selecting business scenario **WORKFLOW** as described in the previous step, select **Category** under **Business Scenario**. Then, create a new category, as shown in Figure 7.16.

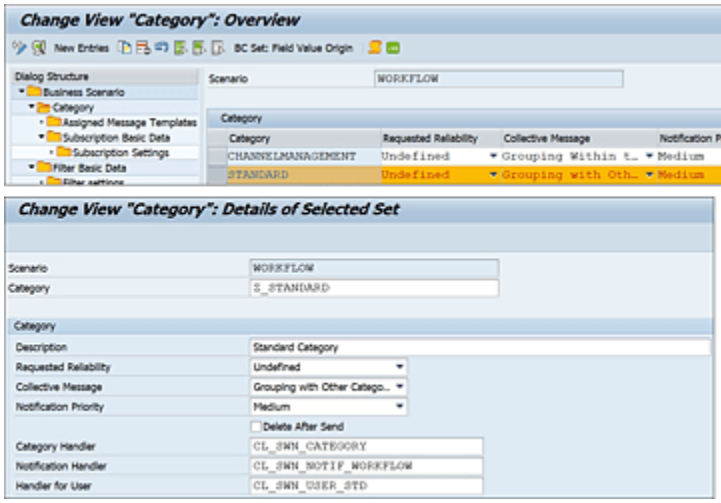


Figure 7.16 Creating a New Category by Copying the STANDARD Category

To create a new category **Z_STANDARD** under business scenario **WORKFLOW**, the easiest way is to copy category **STANDARD**, which is delivered by SAP. In this case, you can also copy the assigned message templates. Otherwise, they must be created and assigned manually. The category is used during the selection and delivery phases. During the selection phase, the work items are selected, and

notifications are created and stored in table SWN_NOTIF. The notifications are assigned to the category of the filter that is processed. The notifications are grouped via the category, and the subscription is also assigned to the category. The category tells which group of notifications should be delivered.

Next, copy the corresponding message templates under the **Assigned Message Templates** node as well, as shown in Figure 7.17.

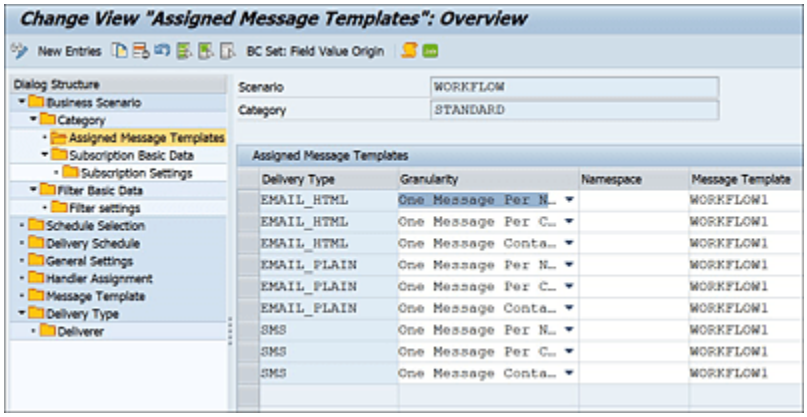


Figure 7.17 Message Templates under Category Also to Be Copied

Filter Pair

The filter pair defines which work items have to be selected. It consists of a full filter and a delta filter. When a filter is processed, the relevant work items are selected. The processing of the full filter is different from the processing of the delta filter, as follows:

- **Full filter**
  - This selects all open work items (based on filter criteria). The database selection is done from table SWWWIHEAD and is independent from any time stamp.

    - The agents of these work items, including substitutes, are determined.
    - A notification for each work item and agent is created.
    - New notifications are inserted.
    - Existing notifications that have become obsolete are logically deleted (e.g., work item isn’t open anymore, work item doesn’t exist anymore, work item was forwarded to another user, etc.).
- **Delta filter**
    - This selects all work items that were created or changed since the last run of the filter pair. The data selection is done via table `SWWWIHEAD` and table `SWWLOGHIST`. The selection includes open and completed work items.
    - The agents of the open work items, including substitutes, are determined.
    - A notification for each work item and agent is created.
    - New notifications are inserted.
    - Existing notifications that became obsolete are logically deleted.
    - Existing notifications are modified in case of special work item changes. But this case isn’t relevant for the SAP Business Workflow scenario.

To create a filter, follow these steps:

1. First, create a full filter. To create a filter, from Transaction SWNCONFIG under node **Business Scenario**, select the subnode **Filter Basic Data**. Then, click on **New Entries**. Under node **Filter Basic Data**,

select **Filter Settings**, and then add parameter **DELTA** with a blank value.

2. Under **Filter Basic Data**, the field **Category** needs to be filled up with the newly created category done in the previous step. This will link the filters with the category. This means that the notifications are assigned to this category during the selection phase. You can see the full filter in Figure 7.18.

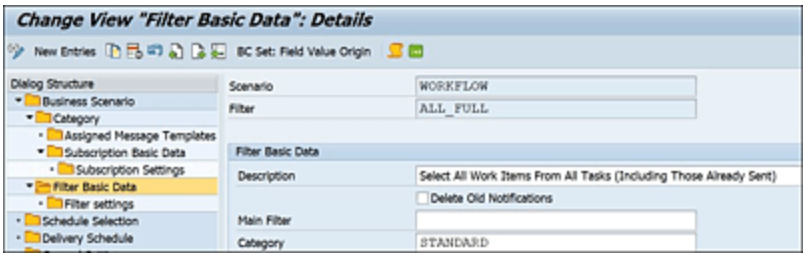


Figure 7.18 Full Filter

3. Next, create a delta filter. The initial steps are the same as for a full filter, but under node **Filter Basic Data**, when you select **Filter Settings**, add parameter **DELTA** with an “X” value, as shown in Figure 7.19.

4. For the delta filter, under the **Filter Basic Data** node, the **Main Filter** value should be filled with the **Full** filter that just got created. This will ensure that they both interact as a pair. You can see the delta filter in Figure 7.20.

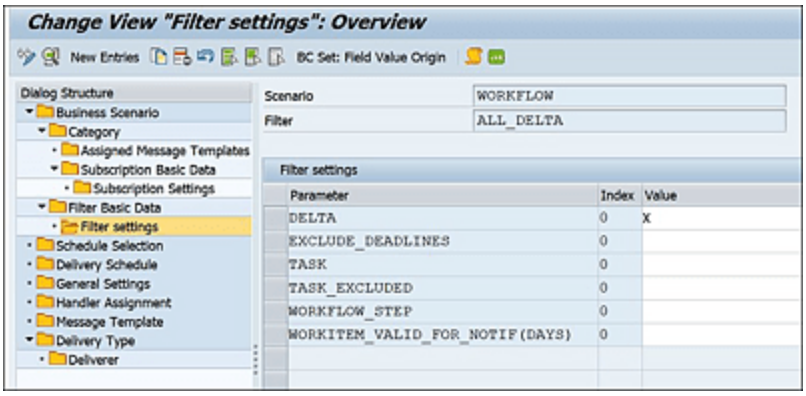


Figure 7.19 Filter Settings

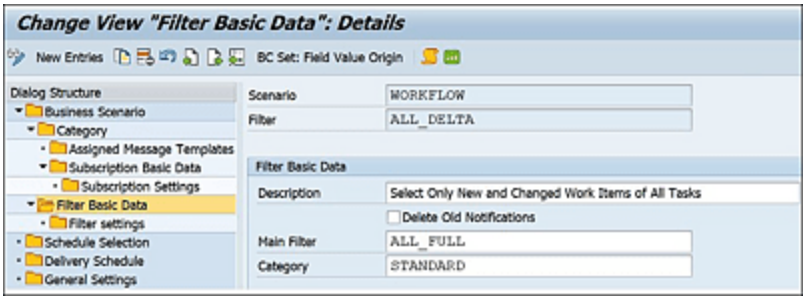


Figure 7.20 Delta Filter

Important

Delete Old Notifications in the filter should not be set in the production system. This setting is only intended to be used during the development phase if you want to start from scratch; for example, you test a configuration. When this option is set, all notifications belonging to the main filter are deleted from the database table. This means a delta handling can’t take place, and the reminder function doesn’t work. In addition, the option influences the performance in a negative way in that all notifications belonging to this filter pair are first deleted physically from the database. Afterward, the normal processing of the filter is done (work item selection, agent determination, etc.). To remove the notifications from

database table `SWN_NOTIF`, program RSWNNOTIFDEL should be scheduled periodically to keep the data amount as small as possible.

Selection Schedules

Our next step is to create a selection schedule for each filter. The selection schedule defines when (weekdays, time frame, and interval) the filter should be processed. (In addition, Transaction SWN_SELSEN must be scheduled by following the regular job selection procedure.)

To create a selection schedule for the full filter, follow these steps:

1. Create a new schedule, and choose the full filter. To do this from Transaction SWNCONFIG, select the node **Schedule Selection**, and click on **New Entries**.
2. Mark all the checkboxes next to **Weekdays** (see Figure 7.21).
3. The full filter should be processed once a day. Therefore, enter the desired time into the **Time from** field as well into the **Time to** field, and maintain the **Time Zone**. Don’t enter a value into the **Interval** field.

To create a selection schedule for the delta filter, follow these steps:

1. Create a new schedule, and choose the delta filter. To do this from Transaction SWNCONFIG, select the **Schedule Selection** node, and click on **New Entries**.

2. Mark all the checkboxes next to **Weekdays** (see Figure 7.21).
3. The delta filter should be processed more frequently. Maintain the start time (**Time From**), end time (**Time To**), and the **Time Zone**. Enter the **Interval (Minutes)** in minutes.

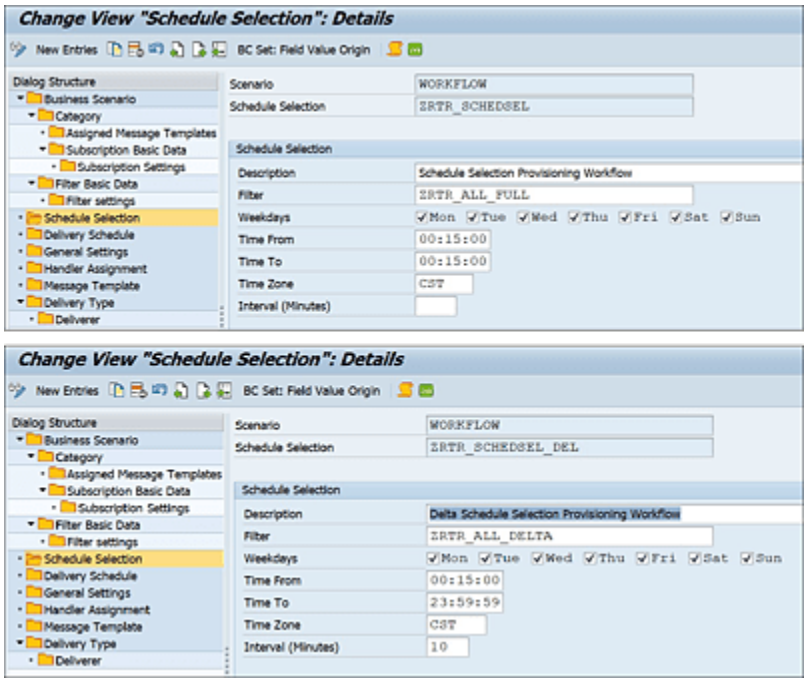


Figure 7.21 Schedule Selection for Full and Delta Filters

Delivery Schedule

Create a delivery schedule from Transaction SWNCONFIG by selecting the **Delivery Schedule** node, and click on **New Entries**, as shown in Figure 7.22. The delivery schedule tells when the emails should be sent out. The delivery schedule is later assigned to the subscription. A delivery schedule can be assigned to more than one subscription. Figure 7.22 shows the delivery schedule.

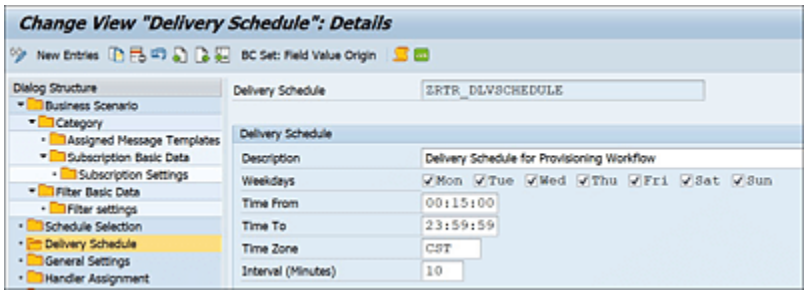


Figure 7.22 Delivery Schedule

Subscription

The subscription is required for the delivery phase. Emails are only delivered if a subscription is available. It contains the following information:

- Which notifications should be delivered (**Category**)
- When they should be delivered (**Delivery schedule**)
- What the email/SMS should look like

Following are the steps to create a subscription and define its settings:

1. From Transaction SWN_CONFIG, select the **Business Scenario** and **Category**. Under that, select subnode **Subscription Basic Data**. Choose the newly created category. (The notifications created during the selection phase were assigned to the category. By using the same category in the subscription, the notifications, which should be sent out, are chosen.)
2. From the **Subscription Basic Data** node, click on **New Entries** to create a new subscription.
3. Assign the delivery schedule created in the previous step.

4. Choose the delivery type:
   - **E_MAIL_HTML**: HTML mail
   - **E_MAIL_PLAIN**: Plain text mail
   - **SMS**
5. Choose the granularity:
   - **N**: One email contains a list of work items.
   - **1**: For each work item, one email is sent.
   - **C**: Only a general information “you have new work items” is sent out, which doesn’t contain work item–specific data.
6. Define the recipients (**Recipient Address**, **Recipient Type**, and **Handler** fields). There are several possibilities:
   - The emails should be sent to all work item agents: **Recipient Address** = * and **Recipient Type** = **RML**.
   - The emails should be sent to one particular user: **Recipient Address** = **<User ID>** and **Recipient Type** = **RML**.
   - The emails should be sent out to a list of work item agents, but *not* to all: **Recipient Address** = *, **Recipient Type** = **CUS**, and **Handler** = **ABAP class**. For further information, see SAP Note 847042 (Handler for Filtering Notifications in the Subscription).
7. Maintain the subscription settings by selecting the **Subscription Settings** node under the **Subscription Basic Data** node, as shown in Figure 7.23.

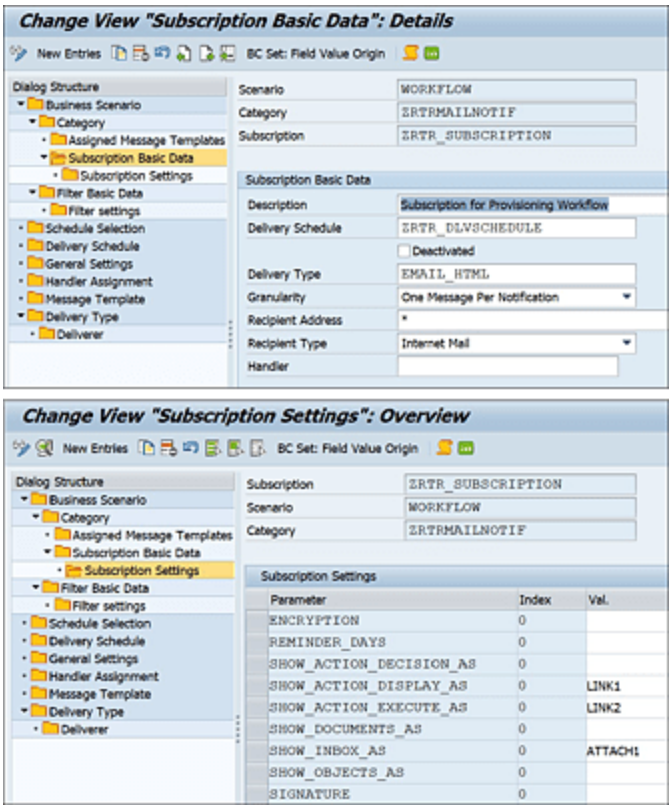


Figure 7.23 Subscription and Subscription Settings

Generally, the following parameter values are filled in this section: **REMINDER DAYS** (after x many days, a reminder is sent), **SHOW_ACTION_DISPLAY_AS** (a **Display Work Item** link is generated into the email), and **SHOW_ACTION_EXECUTE_AS** (a **Execute Work Item** link is generated into the email).

7.4 Adding Inbox and Work Item URL Links to Workflow Work Item Notifications

When the notification is sent to the user's email inbox to process the work item, sometimes it's required to provide a URL link of the corresponding work item because all business users aren't used to opening this from the SAP inbox and processing from there. Adding this work item URL is part of Transaction SWNCONFIG features and it's done through a handler class.

At the category level of Transaction SWNCONFIG, there are three handler classes: **Category Handler**, **Notification Handler**, and **Handler for User**. Out of the these, the **Notification Handler** class handles sets of different details of the notifications sent to the user. In this class, there are methods such as `ADD_LINKS_ACTION`, `ADD_LINK_ACTION_DISPLAY`, and so on, which add and display URLs in the work item. Figure 7.24 shows the handler classes and the methods for class `CL_SWN_NOTIF_WORKFLOW`.

By default, through the standard process with the help of Transaction SWNCONFIG steps that we did in previous section, the inbox and URL link to execute the work item URL link are added to the notification.

In some cases, there are requirements to customize this, for example, when executing a work item and a specific transaction or application must open. You can customize these processes by copying the standard classes into a

custom class and then enhancing these methods accordingly. Let's see a custom scenario in which when executing the work item, it needs to open an ABAP Web Dynpro application. The steps are as follows:

1. Go to Transaction SE24, and create subclass `ZCL_SWN_NOTIF_WORKFLOW`. The superclass of this subclass is `CL_SWN_NOTIF_WORKFLOW`.
2. Redefine method `ADD_LINK_ACTION_EXECUTE`. This method is used to add the hyperlink of the Web Dynpro application, and, when executed, it will open the application. In the method logic, you'll get all the values of the workflow container from the work item created by the workflow based on the work item ID with the help of FM `SWW_WI_CONTAINER_READ`. `M_WI_HEADER` is a global attribute that holds all the work item–related header data. You can see the relevant code in Listing 7.1.

```
CALL FUNCTION 'SWW_WI_CONTAINER_READ'
  EXPORTING
    wi_id                  = m_wi_header-wi_id
  TABLES
    wi_container           = i_cont
  EXCEPTIONS
    container_does_not_exist = 1
    read_failed            = 2
    OTHERS                 = 3.
```

Listing 7.1 Method Logic of SWW_WI_CONTAINER_READ

   Get the values of the element `_WI_OBJECT_ID` from the container. This holds the business object repository (BOR) and the GUID.

3. Create the ABAP Web Dynpro application URL. In this example, `lv_url` = concatenate "static part of the URL"

and “WI_OBJECT_ID”. After that, the URL will look like the following:
<schema>://<host>.<domain>.<extension>:<port>/namespace>/webdynpro/<application name>/’ObjectId’

4. Populate `C_LINK` (the changing parameter of method `ADD_LINK_ACTION_EXECUTE`) with the URL you’ve constructed and the hyperlink name that will appear in the mail, as shown in Listing 7.2.

```
wa_link-category = swn1_ref_type_tech. "('T')
wa_link-id = swn1_ref_id_execute.      "(EXECUTE_LINK_URL)
wa_link-caption = 'Execute Work Item'(002).
wa_link-url = lv_url.
APPEND wa_link TO c_links.
CLEAR wa_link.
```

Listing 7.2 Populate C_LINK

5. Open Transaction SWNCONFIG, and select the **Category** node. Replace the **Notification Handler** class as “ZCL_SWN_NOTIF_WORKFLOW”, which is created in the previous step, as shown in Figure 7.24.

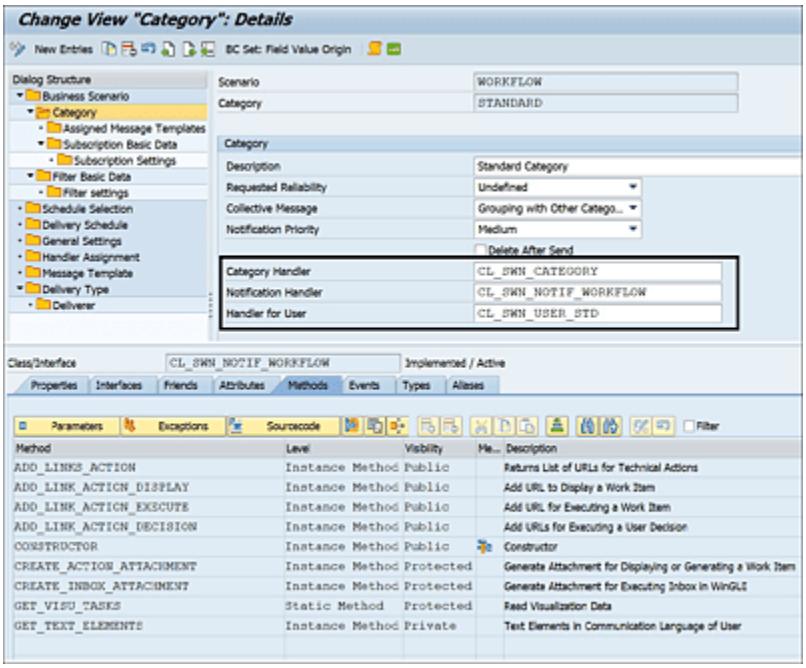


Figure 7.24 Handler Classes and the Methods of Class CL_SWN_NOTIF_WORKFLOW

7.5 Additional Workflow Runtime Jobs

There are some runtime jobs you need to schedule to complete the extended notification emails delivery and subsequent other items to complete:

- **Program SWN_SELSEN**
  Program SWN_SELSEN must be scheduled as a batch job to send out emails via the extended notifications. The frequency should be adjusted to the selection and delivery schedules created via Customizing. Usually program SWN_SELSEN is scheduled more frequently to ensure that the selection and delivery schedules can be served.
  The notification process via program SWN_SELSEN consists of two phases: selection and delivery. For each phase, schedules must be created via Customizing: selection schedule and delivery schedule. The actual processing is done only if the corresponding schedule is due.
  If the **No Time Check During Send** checkbox is marked, the program doesn’t check whether the delivery schedules are due. This means all existing delivery schedules are processed, and the emails are sent out.
  Figure 7.25 shows the selection screen details of program SWN_SELSEN.

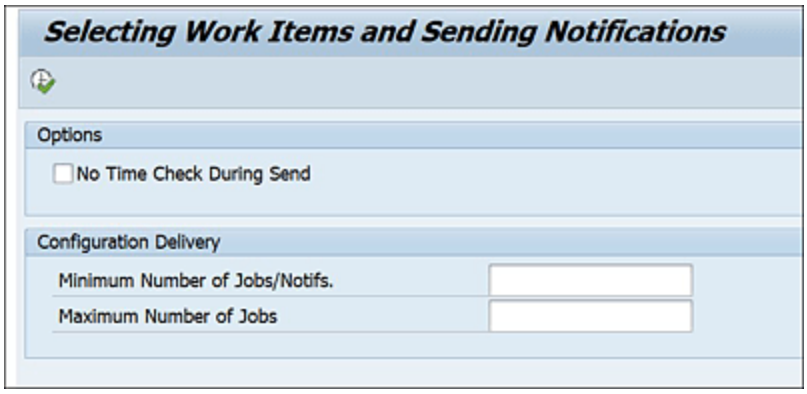


Figure 7.25 Selection Screen of Program SWN_SELSEN

- **Program RSWNNOTIFDEL**
  When a work item disappears from the user's inbox (e.g., work item is completed, canceled, forwarded to another user, etc.), program SWN_SELSEN recognizes that. The corresponding notification is set to status **Logically Deleted** (**L**) in Transaction SWN_NOTIF. The physical deletion is done by program RSWNNOTIFDEL. This program must be scheduled periodically, for example, once a day. Figure 7.26 shows the selection screen of program RSWNNOTIFDEL.

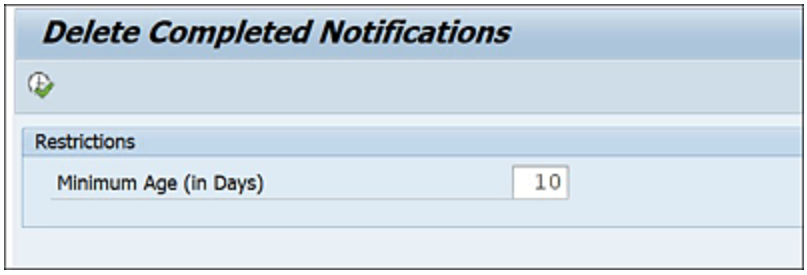


Figure 7.26 Selection Screen of Program RSWNNOTIFDEL

7.6 Summary

Email notification is one of the key features of SAP Business Workflow, and you've seen different types of notifications in workflow in this chapter. We discussed the prerequisites to make the email notification work such as setting up email addresses and setting up SAPconnect. Then, we discussed in detail about using program RSWUWFML2 to send work item email notifications. All features of this program, along with different selection screen parameters and controls, were also covered. Next, we saw a better way to handle email notifications by suing program SWNCONFIG. The Customizing details were depicted with steps. You also saw how to add work item URL links in email notifications and several other runtime jobs required to complete all the necessary items.

8 Workflow Administration, Monitoring, and Troubleshooting

This chapter introduces you to the different transactions, tools, and features available for workflow administration, monitoring, and troubleshooting. The chapter will discuss each tool and transaction, explain their features and use cases, and provide step-by-step instructions for their use. You'll learn to trace their workflows in the system, monitor logs, perform troubleshooting, and restart the workflow after errors are resolved.

Workflow isn't just about building some technical components. It's about integrating business process steps and integrating human activity. So, you can't work in a reactive mode when users will create an incident if the workflow isn't working as expected. Rather, there needs to be a constant monitoring of the processes and proactive identification of the issues to make the workflow implementation successful. The success of workflow implementations heavily depends on the setup, administration, monitoring, and troubleshooting. Workflow deals with people. As an IT support team, you won't be able to always get direct and immediate input from end users

regarding what isn’t working from a process point of view. You may see there is no incident created because the system may work technically, but the process, screen, and method of operation isn’t user friendly. By the time you get this feedback directly from end users, there will already be an impression that the workflow doesn’t work in the system. It’s very difficult and takes time to change those opinions. But if you the proper monitoring set up, you can find that something is taking longer than expected to process, and then you can proactively talk to the business, identify the improvement area, and make it better for users.

We’ll discuss these aspects of workflow development in this chapter. We’ll start with how to interpret the workflow log. It’s very important to read workflow logs because they contain all the runtime information to troubleshoot any workflow. The challenge with workflow troubleshooting is that it consists of a long process chain, so it’s sometimes difficult to simulate the entire process chain, or it takes a long time to replicate a similar issue in a lower environment. The workflow log will help you troubleshoot any issue quickly. We’ll discuss workflow administration and maintenance activities required for regular maintenance of workflows. We’ll provide a list of commonly used transactions and reports that will help you monitor the workflow system health and troubleshooting. We’ll provide examples of some common technical errors and the approach to troubleshoot those errors.

Then, we’ll discuss features and functions of the SAP user inbox. This will help you think from an end user perspective regarding what features should be enabled. We’ll discuss the scenario when someone is out of the office and how

workflow processes can be delegated. We'll end the chapter with a workflow using the ArchiveLink feature. This is a very common scenario in which the workflow process is based on incoming documents.

8.1 Workflow Log

The execution details of the workflow runtime are stored in the workflow log. Execution date/time details, user activity, container variable, and object instance information is all available in the workflow log. Workflow log is used to troubleshoot all workflow application instance-related issues. The most common way to troubleshoot is start identifying the workflow by business object. You'll get the production incident with a business object (e.g., purchase requisition), the key of that business object, and the issues. You have to check the workflow log to find the root cause of the issue.

There are five types of workflow log view available, as follows:

- User view
- Classic user view
- Technical view
- Summary (HTML)
- Classic technical view

All these views provide similar kinds of information with some small differences. Normally, user views (see Figure 8.1) don't have links to container and task ID

information immediately available. We suggest using either technical views or classic technical views when you're going through the workflow log, as we'll discuss later in this chapter. The main difference between technical and classic technical view is that you can get technical information at the bottom of the screen in the technical view (compare Figure 8.6 showing the technical view with Figure 8.8 showing the classic technical view later in this chapter). However, that will reduce the space to see all the workflow steps shown at the top part of the screen. You can decide which to use based on your own preference, but both will provide the same information, and only the navigation will be a little different. The user view provides the most significant steps and better agent information.

You can switch between user view and technical view by clicking the button or from menu path **Goto • List with Technical Details/ActiveX Version**. We recommend starting with the user view and then switching to the technical view for technical information. You can change the settings by choosing **Goto • Personal Workflow Settings** from the menu path on the workflow log screen to change the default view you'll get when you open a log. You can navigate to the workflow log from almost all workflow troubleshooting and administration transactions. Look for the workflow log icon. We'll go through the details of these three views and their navigation in this section.

Access to workflows is required to troubleshoot any issue. There are ways to go directly to an error task, but we suggest starting with the workflow log and navigating to the error steps from there.

Run Transaction SWI6 (Workflows for Object) to find the workflow instance or follow menu path **SAP Easy Access** • **Tools** • **ABAP Workbench** • **Development** • **SAP Business Workflow** • **Runtime Tools** • **SWI6 – Workflows for Object**. All SAP Business Workflow transactions are available under menu path **SAP Easy Access** • **Tools** • **ABAP Workbench** • **Development** • **SAP Business Workflow**.

Select the workflow you want to display the log, and click on the workflow log button. You'll get the key steps in the overview of the workflow log. Figure 8.1 shows the hierarchical view of the workflow steps.

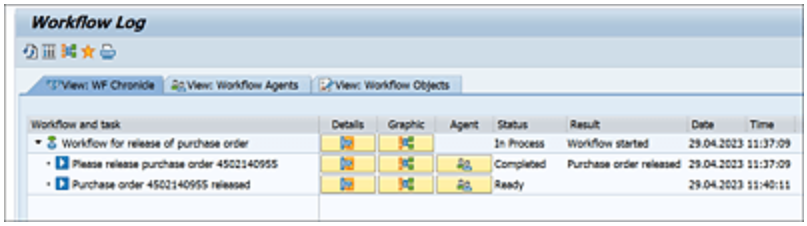


Figure 8.1 User View of a Workflow Log: WF Chronicle

As you can see, the **View: WF Chronicle** tab shows the completed key steps, their user assignment, and the pending work items and corresponding agents.

If you click on the details button, you'll find the details of the step, such as when the work item is created, who has executed this work item, whether any deadline is reached for this work item, and so on. Sometimes, users may create a ticket that they haven't received any item for approval. This log will help you get the history of the workflow and people associated with it.

This view is very useful to analyze agents. This view provides the information on selected, possible, and excluded agents. Click on the **Agent** button to see the details of the agent. Figure 8.2 shows the agent information of an unprocessed work item. You can now click on the **Agent** button to see who is the current processor of the work item.

Normally, you use this view to understand the process flow of the workflow and different agents associated with it.

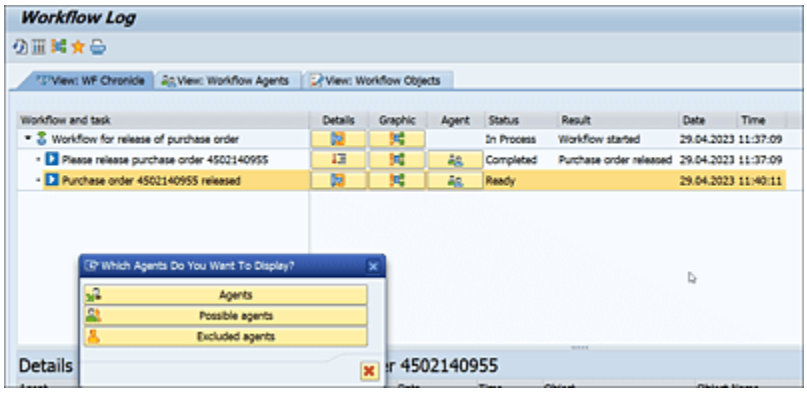


Figure 8.2 Agent View in the Workflow Log

You can click the graphical view button to see the flow of the process steps in the graphical editor. The green colored arrow will point to the position the workflow steps are completed. Figure 8.3 shows the same workflow in the graphical view.

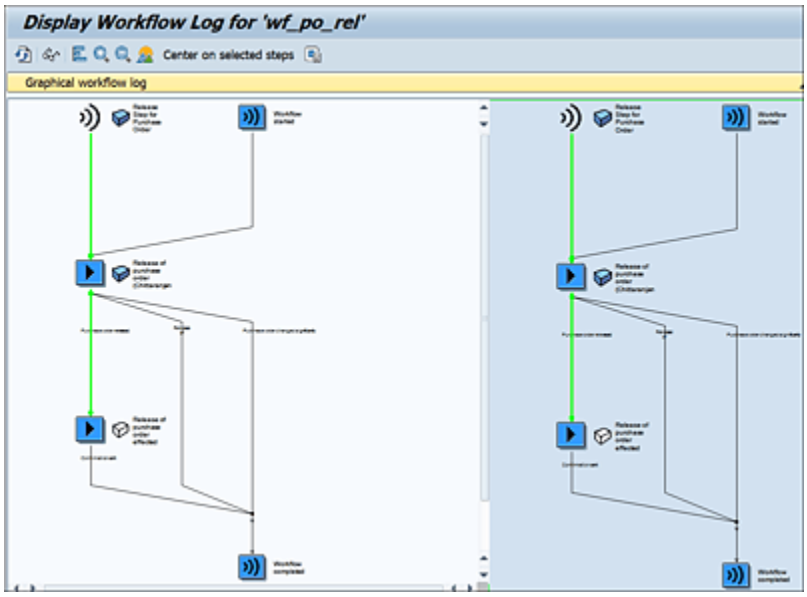


Figure 8.3 Workflow Log in Graphical View

The right pane of the graphical view shows the entire view. You can zoom in on any specific area to see it on the left pane. This may be useful for a simple workflow as shown here, but for complex workflows with loops, this view may not be that useful.

The **View: Workflow Agents** tab in the user workflow log view will show the activities of the agent involved in the workflow. It will provide a summary for each agent involved. If an agent received two work items during the lifecycle of the workflow, then this view will show both the tasks executed under one tree node, as shown in Figure 8.4.

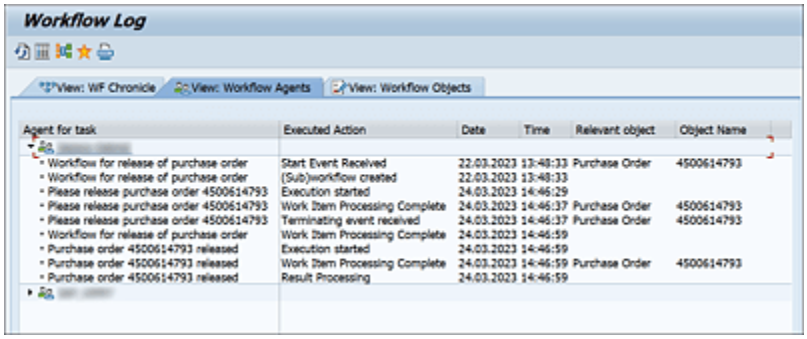


Figure 8.4 Workflow Agent View

This view mainly provides the information regarding who executed the work item and when.

The **View: Workflow Objects** tab will show the activity log related to the business object. It mainly shows the events triggered and the foreground steps executed by the end user who is linked to the workflow object. It will show you a subset of information that is available in the other two tabs. Figure 8.5 shows the workflow object view in the workflow log.

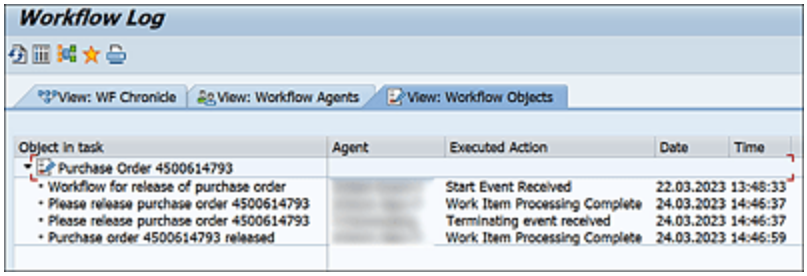


Figure 8.5 Workflow Object View in the Workflow Log

During any troubleshooting of the issue, you start from the user view **SWIG: WF Chronicle** tab. This provides the overview of the steps completed and the associated agents. The next step will be to further deep dive to get technical information about the workflow. There are two technical views available: technical view and classical technical view. Both provide similar information, so the choice comes down to user preference.

Click the **Technical Details** button to navigate to the technical view. Figure 8.6 and Figure 8.7 show the same workflow log in the technical view. You can always come back to the user view by clicking the **ActiveX version log**

button. As you can see, this view will show the execution sequence in a tree structure. This will help you understand how each step of the workflow got executed.

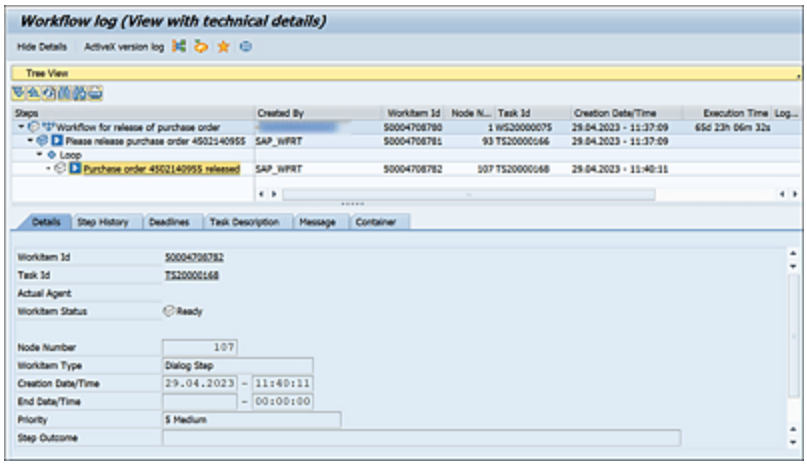


Figure 8.6 Technical Workflow Log View

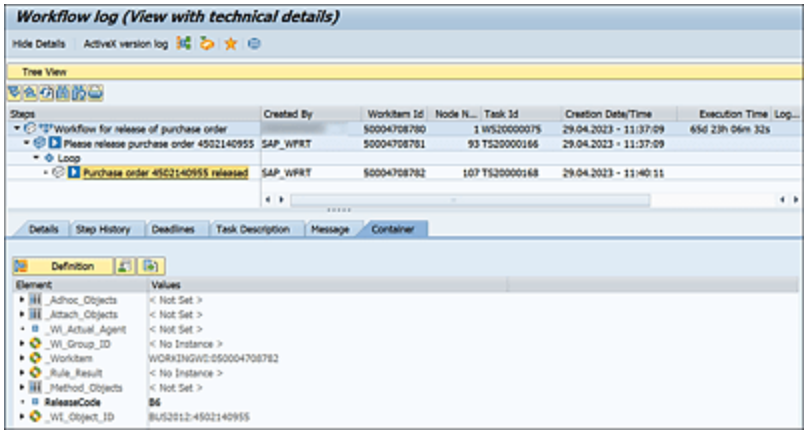


Figure 8.7 Technical Workflow Log: Container View

This view of the workflow provides all the information in a single place. The top part of the screen shows the workflow steps in a tree view, and the bottom part of the screen shows the technical details of each step. The technical details are shown in different tabs. You can click each step of the workflow to see the details of that step in the following tabs:

- **Details**
  This tab shows the technical and execution details of the workflow, such as task or workflow number, when it was created, current status, and so on.
- **Step History**
  This tab shows each activity performed under this step. If any background function module went into error during the execution, you'll find those details in this step. You can select each of the actions and then click on **Technical Details** to see the details of the error message, including the message area, message number, and so on. This tab will be very useful to troubleshoot any background errors that occurred during execution.
- **Deadlines**
  This tab shows all the deadline-related information, such as what deadlines are set up for the step and which one is missed.
- **Task Description**
  This tab provides the task description that users will see in their inbox.
- **Message**
  This tab captures the last message triggered. You can find the same information in the **Step History** tab.
- **Container**
  This tab provides the value of each container.

You won't get the agents in this view directly. To get the agents view, select the relevant task, and choose **Utilities** • **Agent**. You can see all the error messages from menu path **Utilities** • **All Error**.

You can also navigate to the administrative task for the work item from this view. Select the relevant step and choose **Goto • Workitem**, or right-click on the step, and select **Display Workitem**. It will take you to work item administration transaction (Transaction SWIA), where you can forward, execute, or replace the work item as an administrator.

You can also navigate to the technical definition of the workflow or task. Right-click on the relevant item, and click on **Display Task**. There are other options to navigate to different information. Here, we've only discussed the ones that are more useful during project work, but you should explore all the options available.

Another popular view, the classic technical view, is similar to the technical view with some navigational difference. Figure 8.8 shows the classic technical view of the same workflow.

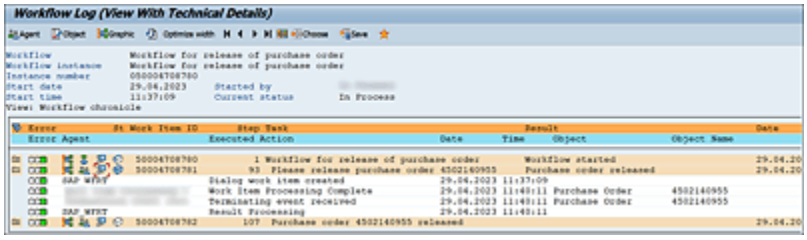


Figure 8.8 Workflow Log: Classic Technical View

The look and feel of the technical view and the classic technical view are different. The key noticeable differences from usability are listed in Table 8.1.

Element	Technical View	Classic Technical View

Element	Technical View	Classic Technical View
Agent	You have to get the agent information (agent/possible agent/excluded agent) from navigating from menu path.	You can get this view by clicking a button on the corresponding task line item.
Execution view	This view shows the workflow steps in a tree structure with a logical execution block. For example, if a step is executed with a LOOP, the step will be displayed under the LOOP block. This representation is not available in classic technical view. It's easier to compare the flow of workflow execution with the Workflow Builder.	This view shows all the steps for task execution, but they aren't like the Workflow Builder sequence of steps.
All step information in a single view	Because half of the screen will have step details in tabs, you have to scroll to navigate if you have too many steps.	You can get more steps in a single view.

Table 8.1 Key Differences in Technical View and Classic Technical View of Workflow Logs

All the features mentioned for technical views are also available in classic technical views. Which to use is up to the

user.

8.2 Workflow Administration

The workflow administrator plays a key role in establishing the success of any workflow implementation. The workflow administrator needs both the technical knowledge and the business process knowledge to support the workflow. Workflow administrators should understand the workflow basics, know the functionality of the workflow being supported, and be familiar with the error diagnosis and restarting transactions.

Because a production environment runs all day, it’s required to have more than one workflow administrator in a project to support round the clock. Normally, workflows run smoothly if they are rightly designed, so there may not be enough workflow administrator work to fully occupy a single person. In that case, other duties should be assigned to the workflow administrator. There are a few options for workflow administration structures. Because workflows may handle a few important processes such as invoice, appraisal, and so on, some errors may need to be resolved very quickly. You should consider the following when structuring a workflow administration team:

- It’s helpful to have a centralized team with a mix of technical and application owners. The technical team is also responsible for support of other systems.
- You can also consider including workflow administrators who are also workflow developers.

- The size of the team will depend on the number of workflows running in the system. It also depends on the number of workflow instances created in a day. The larger the volume, the more support is required.
- The more application areas you have using the workflow, the more business contact you need. The end user should be trained properly to minimize the confusion regarding the system functionality.
- The businessperson should be the first point of contact for any process issue. Then, the businessperson will contact the workflow administrator to investigate in more details.
- A document should be prepared with the possible errors and the solutions to the problem. Any new kind of error should be updated in this document.
- The workflow administrator should consult with the business owner and the workflow developer before restarting any error workflow.
- Make sure a workflow developer is available to help solve the problem after any new workflow is activated in production.
- The workflow administrator should keep handy all the contact numbers of business owner and workflow developer.
- Because workflows may handle a few important processes such as invoice, appraisal, and so on, some errors may need to be resolved very quickly.
- There are several cases where the workflow is working per the expected procedure. There may be questions from end users on the current status or how to provide this

kind of information. You can enable your helpdesk to provide this information. In general, workflow logs and transactions are too technical, making it difficult for end users. You can consider building custom reports to provide basic information on the workflow current status, to whom it’s assigned, and so on.

- As a workflow administrator, you have to focus on a number of areas to run the workflow engine smoothly. Let’s start by looking at workflow error handling being built into the workflow design:
  - Anticipate the type of error that may arise during execution. For example, you can set it up to handle temporary errors in the task method so they can be reprocessed. A locking error can be set up so that the system automatically reprocesses the temporary error three times (as default workflow settings). Therefore, these errors can be reprocessed automatically without workflow administrator intervention.
  - Ensure the agent determination method doesn’t cause an error. Handle the scenario with the workflow administrator as an agent where the system can’t determine the agent for a step.
  - Have an escalation matrix defined from the business side to handle the exception scenario. You can consider certain scenarios to be handled by someone from the business. For example, if an agent isn’t determined, then send the work item to some central person from the business who can forward it to the responsible person, and the workflow can continue from there.

Another important thing to focus on is workflow documentation, as follows:

- Four kinds of documentation are helpful for any workflow: the user training document, business process document, technical details document, and error handling. A detailed process flow diagram should be included in either the business process document or technical details document. User training documentation will guide the user to use the workflow properly by showing each step of the workflow with a proper screenshot. A document detailing SAP inbox set up and the SAP inbox functionality will be helpful to the end user and should include the steps to set up substitution and automatic forwarding. Other inbox functionalities such as reserving a work item, replacing a work item, and finding the work item depending on the workflow task, may be included in the document.
- The business process document will outline the detailed business process and should include a flow diagram. Any application configuration should be included. For example, vendor invoice workflows (Transaction FB50) need application configuration. The business process document should include this configuration with the volume, any authorization requirements, and any special error-handling requirements. The test conditions need to be tested, including both positive and negative conditions. The test conditions will help the developer test the object properly by getting a better view.
- The development documentation should include the workflow number, the triggering point of the workflow, and the business object used. If it’s a custom business

object, then briefly describe the functionality of the methods. You should also provide detailed logic of how work item processing agents are determined in the agent determination rule and if any responsibilities are maintained. Be sure to include any screen names, iViews, and Web Dynpro screens used.

Finally, let’s look at the workflow administration transactions and reports. There are several workflow administration transactions that are useful for day-to-day maintenance activities and troubleshooting issues. These transactions are used to determine the workflow items that need to be analyzed. Once the workflow item is identified, then navigate to the log  for further analysis. You also have to monitor the workflow engine running along with your SAP inbox. We’ll describe some key transactions that should be monitored in regular frequency. You can also think of setting up a bot to handle the allocation of these errors based on the business process area.

Before we jump into the key transactions, let’s first discuss the option to display the task and log that they each contain. The key activities that you can perform are as follows:

- **Execute the task**
  You’ll have the authorization to execute the task, but it’s not recommended for a productive environment. Only the business user responsible should execute the task. There can be audit violations if you execute any work item.
- **Execute agent rule**
  The system will re-execute the agent determination rules assigned to the task. The task will be assigned to the

newly determined agent. If the agent isn’t determined, then it will stay in the current state. You can also perform this task using Transaction SWI1_RULE. Reexecuting the agent rule is not only done for error cases. If new possible agents are added and the work item is reserved by any user, you can set the work item status back to **READY** as an administrator and re-execute the agent rule so that new possible agents also get it in their inbox. You can also re-evaluate the agent by using a special function of the workflow header event **Re-Evaluate Agents of Active Work Items**.

- **Forward work item**
  If there is no way to redetermine the agent for the rule, then you can forward the work item to the appropriate user. When you’re developing a workflow, you have to think of these scenarios, for example, if the agent can’t be determined based on rules and administrator forwards it to someone, how will the next level of approver be set? Will it be based on the person approved in the last step, or will it be based on the original line of command?

Let’s now look at the key transactions:

- **Workflow reorganization**
  You’ll find the workflow reorganization-related transactions in **Tools** • **Business Workflow** • **Development** • **Administration** • **Workflow Runtime** • **Reorganization**. You should think of the archiving workflow log. There are options for both archiving and deleting the workflow log. The workflow log can grow and take considerable disk space if it’s used heavily in the organization. You should archive it in the production

environment and consider deleting it in nonproduction environments.

- **Application log**
  This can be found via **Tools** • **Business Workflow** • **Development** • **Administration** • **Workflow Runtime-Application Log** • **SWF_APPL_DISPLAY**. This is similar to the application log display (Transaction SLG3). This filters the same transaction with object `WF`.

- **Work items without agents**
  Choose **Tools** • **Business Workflow** • **Development** • **Administration** • **Workflow Runtime** • **Work Items without Agents** (Transaction SWI2_ADM1). This report is used to determine the work items that are generated without any agent assignment. The selection screen of this report will provide you the options to select based on the time frame and tasks. The output of the report will look like Figure 8.9.

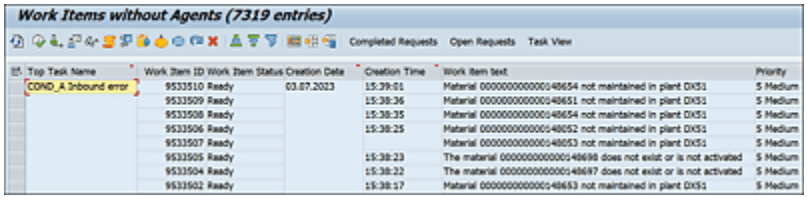


Figure 8.9 Work Items without Agents

- **Diagnosis of workflows with errors**
  Choose **Tools** • **Business Workflow** • **Development** • **Administration** • **Workflow Runtime** • **Diagnosis of Workflows with Errors** (Transaction SWI2_DIAG). This report (see Figure 8.10) will show all the workflows in error.

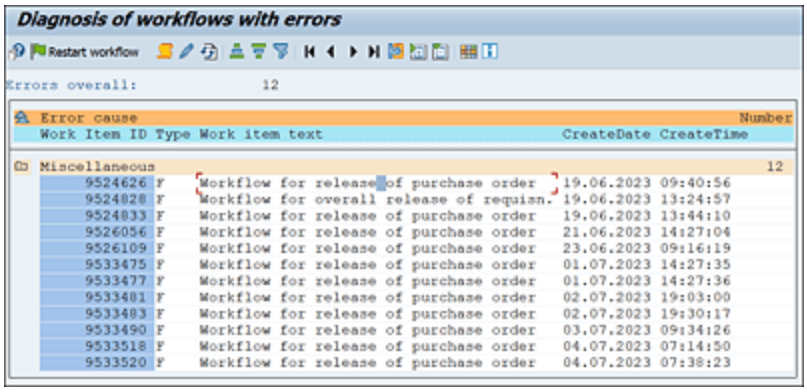


Figure 8.10 Diagnosis of Workflows with Errors

As a proactive measure, you can check these workflows regularly and take corrective action before any ticket is even logged. You can restart the workflow from the error step after taking the corrective action for the root cause. The output of this report is shown as well. A similar function can be executed from Transaction SWPR (Workflow Restart after Error).

- **Continue workflow after system crash**
  Choose **Tools** • **Business Workflow** • **Development** • **Administration** • **Workflow Runtime** • **Continue Workflow after System Crash** (Transaction SWPC). For any system crash, execute this transaction. Check if any workflow item is stuck in this queue. You can restart the workflow from here.
- **Workflow for objects**
  Choose **Tools** • **ABAP Workbench** • **Development** • **SAP Business Workflow** • **Runtime Tools** • **SWI6 – Workflows for Object** (Transaction SWI6). The preceding transactions are used as a proactive measure to check system-wide issues. But when you get productive incidents, you'll get a reference to a particular workflow object, and you have to troubleshoot for that object. Use

Transaction SWI6 to find the workflow objects that are having the issue and take corrective action (refer to Section 8.2 on how to use this transaction). Get the workflow instance using the object and key. Then go to the workflow log, get all the technical details, and analyze the issue. You can navigate to the work item display transaction or use the technical details for any of the other administrator transaction to solve the problem.

You can find more transactions in the SAP Easy Access menu. But these transactions should be enough for you to troubleshoot any issue or maintain the system.

8.3 Workflow Error Diagnosis and Resolution

Because workflows affect the business process, the business owner should be consulted about any error before making a decision. After any workflow error, the administrator should analyze the following:

- What is the business impact?
- How long will it take to fix the problem?
- Is it possible to restart the process from the error point, or does the process need to be started from the beginning?
- Who should be notified of the failure?
- Does the error have any downstream effect?
- If the failure affects the customer directly, then notify the customer.
- If there is any manual workaround before placing the actual solution, it takes time to move the code fix to the production environment in any validated system. So, the workflow administrator needs to find some quick workaround.
- The error should be documented so that it can be prevented from occurring next time.
- Don’t delete any workflows from the system. You can set the status to **Logically Deleted** for any error workflow. After analyzing the error, if the workflow can’t be restarted, then set the status to **Logically Deleted**.

Update the workflow description with the reason for the activity. This will help during the system audit. Create a document with the list of workflows logically deleted and include a detailed description of the error and the resolution for that error.

- If workflow administrator manually executes a work item, then a document should be attached to the work item with the reason for the activity. The same should be done for any administrative execution going forward.
- The workflow administrator should keep a daily report of the number of workflows started and the errors. This helps a business understand the health of the workflow system and makes them confident in the workflow.
- Handling a workflow in the productive environment and nonproductive environment differs quite a bit. In a nonproductive environment, when you get an error, you analyze the root cause, fix the error, and create a new workflow to test the fix. But in production, you create a new instance of the workflow that will be the last solution. The workflow should be built in such a way that it can be restarted from the step where it’s stuck.
- Often, incidents are created with error messages without providing the business object details. Create an instruction guide for the support center regarding the basic minimum information needed for the workflow from the end user. This helps to improve the turnaround time for production incident resolution.

We’ll now discuss the areas where you’ll mostly get errors and how to handle those errors:

- **Agent assignment error**
  This is the most common error in the system. You can use any of the workflow log analysis transactions to display the error. Refer to Section 8.1 on how to view the workflow log. You get the agent assignment–related error message, as shown in Figure 8.11.
  You'll find the details of the agent rule in the error message if the rule is assigned for the agent determination rule. If the error is due to programming, any backend configuration, or system data, then you fix that and reexecute the rule using Transaction SWI2_ADM1. Refer to Section 8.2 for more details on the administrator transactions to handle the error workflow. It's very important how you design your workflow rules. Beginners commonly make the mistake of setting the task attribute as a general task. If a task attribute is set as a general task and the agent determination rule doesn't come up with any agent or neither raised any error, then the work item goes to all users in SAP. So, you have to determine the design strategy when choosing the work item attribute. On the other hand, if you don't set it as a general task and add possible agents, then a lot of maintenance may be required for possible agents if they're not set up with some generic role or organizational units. If you decide to go with the general task attribute, you should raise an exception from the rule so that the work item doesn't go to all users.
  If the rule can't be regenerated successfully after an error due to some runtime generated data, you have to logically delete the workflow and trigger a new one with the same business key. You must document that in the

workflow and inform all the users involved in that instance of the workflow.

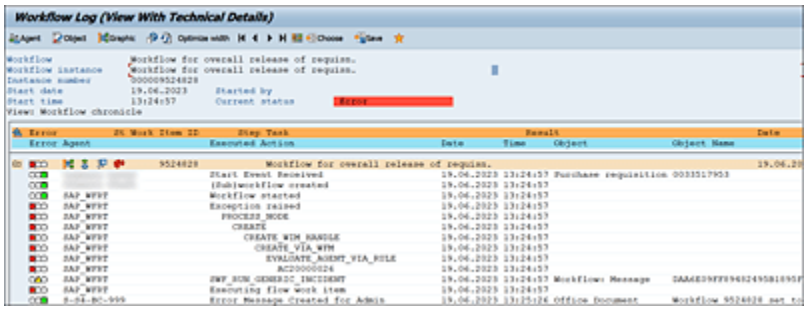


Figure 8.11 Agent Determination Error

- **Error in the business object methods**
  There can be application logic errors within the business object method ABAP code. The workflow log will show the error, and you can analyze from that step. You have to go to Transaction SWO1 to display the business object and corresponding method. You can single-test the method and debug the method code to find the error in the application logic ABAP code. If you're using ABAP classes and methods, then you have to check the ABAP code in the class and fix the application logic.
- **Container and binding error**
  Normally, a static syntax check is executed when you activate the workflow. In some cases, for dynamic and complex binding, you may get a runtime error. Another error scenario is when a mandatory parameter of a container isn't filled. You won't be able to restart these kinds of workflows unless the container element is updated or binding is changed. It's normally not recommended to change any container data manually in a productive environment. You can decide the best way to handle these errors based on your specific scenario.

- **Buffering problems**
  Sometimes you'll find workflow is picking up incorrect agents even when the workflow is set up correctly. This happens due to the buffer. You'll mostly encounter this issue in nonproductive environments where you make frequent changes. Workflow reads the majority of the organization management and agent-related information from the buffer for better performance. Because it reads from the buffer, you may find the changed organization data point isn't considered by the workflow during agent determination. You have to run Transaction SWU_OBUF to refresh the buffer. You'll find the workflow starts running perfectly after the buffer refresh.
  Transaction SWU_OBUF gets executed automatically in the system every night at midnight, so you may find something that wasn't working yesterday is running fine the next morning. This is also known as the Cinderella Effect. Normally, you won't find this issue in productive environments because organization management data doesn't change that frequently in production. Run Transaction SWU_OBUF after any workflow changes move to production. In a nonproductive environment, run it whenever you're making any workflow changes. You can also refresh the organization environment using Transaction SWUS. It's useful in the development/test environment where you can refresh the buffer for the current user ID.
- **Event linkage error**
  You may find suddenly workflows aren't starting in the system. There are two ways workflows are triggered:
  - With the triggering event

- Directly started from an application, for example, by calling function module `SAP_WAPI_START_WORKFLOW`

It’s recommended to start workflows only via the triggering event to provide better capability to troubleshoot and retrigger the workflow.
If the workflow isn’t getting triggered and is supposed to be triggered by an event, follow these steps to troubleshoot the issue:

- Check the event linkage between the event and the workflow. You can find it in the Workflow Builder (Transaction PFTC on the **Check Triggering Event** tab or Transaction SWDD via **Basic Data** • **Event**) or in the event linkage maintenance transactions (Transaction SWE2 or Transaction SWETYPV).
- Check if the event linkage is active. You can find it out in the workflow transaction (PFTC or SWDD) or in the event linkage maintenance transaction (SWE2 or SWETYPV).
- Check if event trace is on (Transaction SWEL). If the trace is on, then check the event trance log (Transaction SWELS) if the event is triggered.
- Check if there is an event check function module attached and if the check function module will allow you to trigger the event for the scenario you’re troubleshooting.
- If the event queue is activated, check the event queue regarding the status of event processing. An event can be processed using the queue or a direct transactional Remote Function Call (tRFC). It’s recommended to use

the event queue for better performance management and tracking of the event. See Section 8.7 for details on how to administrate the event queue.

- Because events are triggered using tRFC, check the tRFC log if the event is triggered (Transaction SWU2).
- Event linkage has a setting for system behavior if any event error happens. If this is set to deactivate linkage, then the event linkage will be deactivated after any event trigger error. We recommend not set tingthis attribute as deactivate linkage. Rather, send the error to the event queue for further processing.
- Check if the workflow start condition is met. You can find the start condition in Transaction SWB_COND or in the workflow header basic data on the **Start Event** tab.
- Check if there is a binding error between the event container and workflow container.

You can retrigger the event after fixing the root cause of the issue. Execute Transaction SWUE to trigger any event. Be careful about the parameters needed to trigger the event. The event initiator becomes the workflow initiator. The workflow's subsequent algorithms may be based on the workflow initiator, so you also need to take care of the workflow initiator so that it reflects the actual business scenario.
If there is an event container and workflow binding build error, you won't be able to trigger the workflow without fixing the binding. In a productive scenario, it may take time to fix this issue in development and move to production using the standard change management process. You may not have time to wait to get the fix

before starting the workflow. In that scenario, you can directly start the workflow using Transaction SWUS (Test Workflow). Be careful about populating the container element properly, including the workflow initiator.

8.4 Workflow Inbox and Features

Workflows integrate business processes and integrate people's actions. The success of a workflow depends on its usability and flexibility to deal with people's actions. When you develop a workflow, you also have to keep the design thinking aspect along with the technology. Workflows are executed by people from the workflow inbox. It's like an email inbox in that you get your actionable workflow items there. In classical workflows, the SAP GUI inbox is the primary workflow inbox where you'll get your work item to execute. There are different options for the SAP inbox. We'll highlight the most commonly used and relevant ones here:

- **SAP GUI-based workflow inbox**
  If the user is using SAP GUI for day-to-day activity, then Business Workplace is the preferable inbox option. This can be accessed from the SAP **Easy Access** menu or from Transaction SBWP. Business Workplace will keep both the actionable work item and the email notification triggered based on the work item. All of the classical work item features are available here. We'll be discussing this in details in this chapter.
- **SAP inbox**
  This is the browser-based SAP inbox. This was previously known as the universal worklist. All kinds of work items are merged into one common inbox. The details of this will be discussed in Chapter 20.
- **Additional options**
  There are other options for accessing work items such as

SAP Fiori apps for approval, and Microsoft Outlook and Lotus Notes for SAP integration. However, these are complex integrations and not that popular.

When you're working on a workflow, you have to understand how the work item will be consumed by end users in the workflow inbox. You should explore the Business Workplace features before you develop any workflow. Keep the following key points regarding usability in mind:

- What information will be visible in the workflow inbox? Work item header and item text should be captured in such a way that the key information is available immediately when the user is opening the workplace.
- A user may get different work items, so is there any way to group them? This will be very useful for user groups such as accounts payable clerks who may get hundreds of work items from different business processes. The team may be grouped by organizational unit, meaning there may be a requirement to group the workflow items rather than having all in one view. Explore the options for how the work items can be grouped for better usability.
- How will the user know the priority items?
- How will the user know when the user has to complete the work items?

We'll discuss the user aspect of Business Workplace (see Figure 8.12), which is where end users get their work items. We'll also cover the different work items that are visible in Business Workplace to help you design your work items while keeping Business Workplace in mind.

Business Workplace has email capabilities along with workflow processing, but it’s normally not used for email. It’s mostly used for workflow processing.

We’ll now discuss what information appears in which part of the inbox. Figure 8.13 divides the entire workplace into six main components.

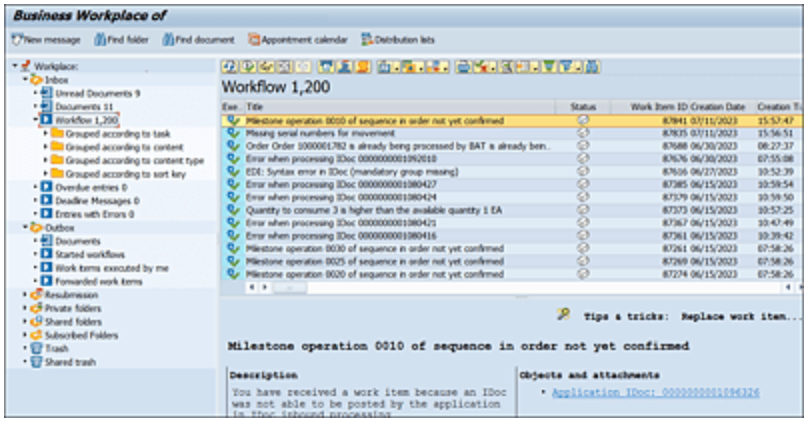


Figure 8.12 Business Workplace

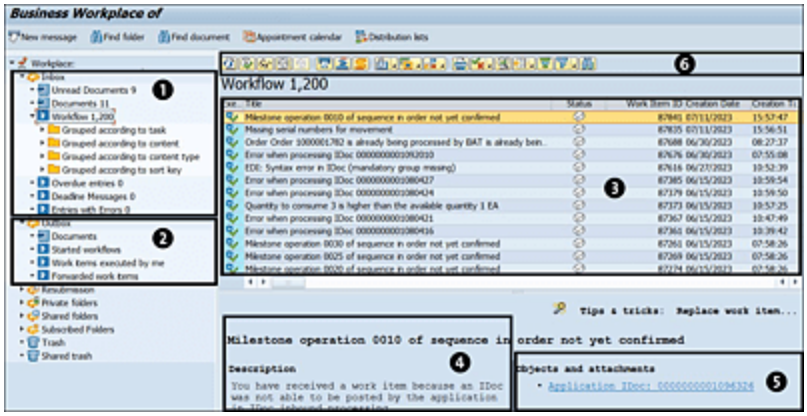


Figure 8.13 Business Workplace with Components

The main components are described in the following:

❶ **Inbox**
This section will contain both workflow work items and SAP documents (similar to email), as follows:

- The first two nodes, **Unread Documents** and **Documents**, will contain any SAP document mail sent to the user. It will also contain other SAP documents such as short dumps.
- The **Workflow** inbox will contain all the work items the user needs to execute. These work items can be grouped by task number, content, content type, and sort key, as described in Table 8.2.

Work Item Group	Work Item Group Description and Purpose
Grouped according to task	This section will group the work items according to the task number. This is very useful for business function groups of people where you'll have lots of work item to process. The task description will appear in each group, so it's important to have a meaningful task description when you're developing the workflow. You should also consider this usability criteria when reusing a task or creating a new task. For example, an accounts payable clerk needs to process financial accounting invoices and invoices that aren't from financial accounting. The work can be segregated by task.

Work Item Group	Work Item Group Description and Purpose
Grouped according to content	This will group the work items by workflow business object key. Normally, this isn’t that useful because in general one user will have one work item for a specific workflow instance.
Grouped according to content type	This will group the work items by workflow business objects. For example, the material master (`BUS1001`) workflow will be grouped into one.

Work Item Group	Work Item Group Description and Purpose
Grouped according to sort key	Earlier groupings are static groupings defined during the build of the workflow. **Grouped according to sort key** provides the options to group according to the specific content of the workflow. There are two standard work item containers: `_WI_Object_ID` and `_WI_Group_ID`. The `_WI_Object_ID` container is automatically populated with the business object instance reference. This is used for **Grouped by content** and **Grouped by content type**. `_WI_Group_ID` can be populated using container binding. This is used in **Grouped according to sort key**. For example, if you're have a global customer approval process and want to group by regional business unit (e.g., EU, NA, APAC, etc.), you can populate `_WI_Group_ID` with the region for better handling of the work items.

Table 8.2 Work Item Grouping in Business Workplace

- Normally, the grouped by feature is needed for people managing group functions where a high volume of workflow items are generated every day. Always talk to the business and understand how the work is distributed when you're developing a

workflow. This will help you understand what kind of work item grouping will be helpful.

- You can get the work items where deadlines are passed in the overdue entries bucket.

❷ **Outbox**

This section shows the workflows started, work items executed, and work items forwarded by the user.

❸ **Work items**

Work items will be shown within this window. You can change the layout by adding/removing the list of columns available. You can also dynamically define columns based on runtime data content. You can perform standard SAP List Viewer (ALV) functions such as filter, sort, and group on these columns. Work item title is one of the key pieces of information users get here, along with the kind of task to execute. When you’re developing a workflow, make sure to enter a meaningful sort text for the workflow title. It’s also recommended to enter the organizational unit (e.g., company code, sales organization, etc.) related information in the workflow title. Users sometimes use filter to select specific work items related to an organizational unit. When you’re developing a workflow, always talk to the key business users to finalize the work item text where needed.

❹ **Work item text**

When you select a work item, the work item description is shown in this area. The work item description comes from the task description. The two main objectives should be fulfilled when you create the work item text for the user. It should briefly state what action is to be taken and include key information about the business object that will help the user take the action. The work item text is also sent as an email to the user if email replication is activated. So you can also provide the link of the SAP web login portal to ease the login into the system.

Normally, the user gets familiar with the activity steps after a couple of approvals, so text related to the action to be performed can be put at the end of the work item text. The beginning of the work item text can be populated with the key business information. Work item header text is also very important from a usability perspective. People sort the work items based on who has a high number of items to be processed. So you can think of placing the organizational unit–related information first in the work item header text. Always work with the end user to come up with the work item text.

❺ **Attachment**
Any attachments and business object instances are available in this section. When you click the business object, it will execute the default method that is normally displayed and display the business object. You can also attach the attachments or view the already-attached documents. Attachments move to the next work items by default, so if the user wants to pass on any ad hoc information to the next level, then an attachment can be a good option.

❻ **Work item toolbar**
This section contains the icons representing the different actions users can take on the workflow, which are detailed in Table 8.3.

Action	Description
Refresh	This will refresh the inbox. Normally, Business Workplace automatically gets refreshed in a few seconds. But this function can be used to manually refresh it. If you've executed a work item and it has a terminating event to complete the work item, then it may take some time to get the work item out of your inbox. There may be some delay to process the terminating event.
Execute	This will execute the method associated with the task. Normally, it will open up the screen where the user can take an action.

Action	Description
Display	This will display the details of the work item. You can view the work item texts and do standard work item operation activities such as forward, resubmit, change priority, and so on.
Reserve	If a work item is sent to multiple users, then any of the users can reserve the item by using this function. Then, the work item will disappear from the other recipient’s inbox. If any user opens the work item, it’s automatically reserved by that user.
Replace	This function releases the work item to all possible recipients. With **Replace**, the work item will be visible to all selected recipients in case there are multiple recipients for the work item.

Action	Description
Forward	Current recipients can forward the work item to other users. This function will depend on the task attribute. Possible options based on task attributes are as follows: • **General Task**: This work item can be forwarded to all users. • **General Forwarding Allowed**: This work item can be forwarded to all users. • **General Forwarding Not Allowed**: This work item can only be forwarded to possible agents of the task. • **Forwarding Not Allowed**: This work item can’t be forwarded. It’s recommended not to allow forwarding to all users. A work item contains business information, so it should be limited to the users responsible.

Action	Description
Resubmit	You can resubmit a work item to a future date. Once you resubmit, the work item won’t be visible in your inbox but will appear in your **Resubmission** folder. You have to provide a date when you’re resubmitting. The work item will again appear in your inbox after that date. You won’t be able to execute the work item when it’s in the **Resubmission** folder. You have to end resubmission if you want to process it before the resubmission date. This function is similar to parking a document. If you have to park a document to get more information, then you can resubmit and keep your working inbox clean. You can document what information you’re waiting for in the attachment to maintain the history of approval.
Log	Users can check the workflow log as mentioned in Section 8.2. The workflow log is little bit complex for an end user, so normally it’s not used by the end user. A custom solution needs to be built if there is a requirement to provide visibility of the workflow progress to end users.

Action	Description
Manage Attachment	Users can view existing attached documents against the work item or create new attachments. You can upload any document or use the SAP editor to document any information. This is an effective way to pass the information to the next level or document the observation.

Action	Description
Additional Function	There are a few additional functions that can be performed here: • Set the work items as **Completed**. Some tasks need a user confirmation to complete them. In that case, a popup will appear to the user to confirm if the activity is completed. The work item will go out of the user's inbox after the confirmation only. You can set the activity to **Completed** here. • Users can reject the execution of the workflow. This option is only available if the processing rejectable property is selected for the relevant task. This isn't considered as an approval rejection, so it's not normally used in workflows. • The work item priority is set as **5** by default. You can change the priority here. The highest priority will have a different colored line, which helps users prioritize what to complete.

Table 8.3 Commonly Used Work Item Toolbar Buttons and Their Functionality

8.5 Substitution and Automatic Forwarding

Business transactions may get stuck if a user is out of the office and can’t take action on the work items. There can be four scenarios: (1) A user left the organization, and there are unprocessed work items in the user’s inbox. (2) A user leaves for a planned vacation, and someone temporarily processes the work items in the user’s inbox to keep the business running. (3) Someone fell sick, which is unplanned, and an urgent work item needs to be processed immediately. (4) For a senior manager, the secretary wants to keep a view of work items coming up in their inbox to be able to inform the manager to take required actions. If a work item isn’t actioned during an employee’s long absence, it can impact business operations such as unpaid bills resulting in penalties, missing out on discounts, employee incurring late payment fees on his credit card, and more. SAP workflow has a robust mechanism called *substitution* that will enable a work item to be processed by someone’s deputy in absence of the main user. This substitution can be activated by the user or by an administrator.

In the following sections, we’ll walk through substitutions in both SAP GUI and in SAP Fiori.

8.5.1 Substitutions in SAP GUI

Substitutions can be done in two ways, as follows:

- **Personal substitution**
  Personal substitution happens at the SAP user ID level. All work items the user has will be available for the substitute to execute. When you do personal substitution, it updates table `HRUS_D2`. Personal substitutions are normally used for temporary out of office situations when the substitute can execute the work item in the absence of the email user.
- **Position substitution**
  When a user opts for position substitution, an A210 relationship is created between the current user's position and the substitute. Position-to-position delegation is performed by the A210 relationship in the HR master. Normally, this relationship is created by the business who maintains the HR master. This isn't done by the IT support team generally. Transaction PO13 is used here. The position substitute can be created with a position, user, or personnel number. The work items assigned to the user's position will be available for the substitute. If any work item agent is determined by user ID or other position, then that work item won't be available with the substitute. Normally, this kind of substitute is used if the current user is changing his job role, and there will be a new person assigned to his old position. This substitution can be useful during the transition period.

If the user forgot to set up his substitution and went on leave, the workflow administrator can also set up the substitute. Workflow administrators can set up the substitute on behalf of another user using Transaction RMPS_SET_SUBSTITUTE.

If a user has resigned, then it’s recommended to set up the substitute before his last working day. If it’s not done and the user has work items in his inbox when he left the organization, the workflow administrator needs to reassign using the administrator report work items without agent transaction.

Let’s take the most common scenario: a user going on vacation. The user wants to substitute for his work items (those already in his inbox and future ones). The user can set up the substitution from SAP inbox. Go to SAP inbox (Transaction SBWP). Then, navigate from menu path **Settings** • **Workflow Settings** • **Maintain Substitute**. You’ll see the personal substitution screen in Figure 8.14.

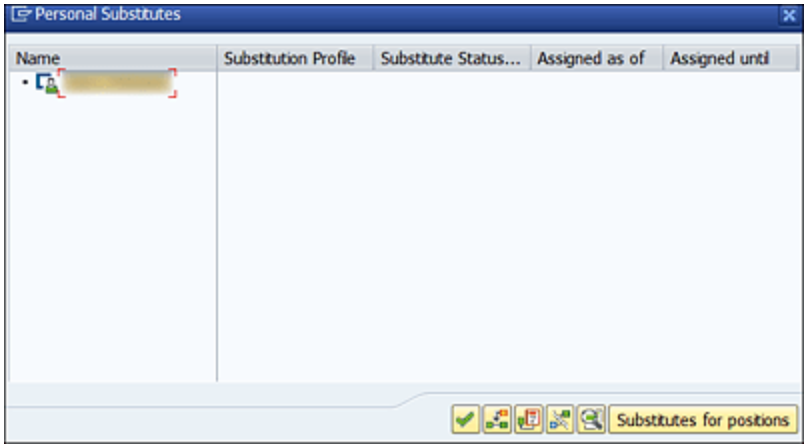


Figure 8.14 Personal Substitutes Creation Screen

Keep your cursor on the username, and click on the **Create Substitute** button . A search popup will appear where you can search for the user ID you want to substitute. Once you’ve selected the substitute, the system will provide you the option to maintain the duration of the substitute and activate the substitution. Figure 8.15 shows the details the user can maintain for the substitution.

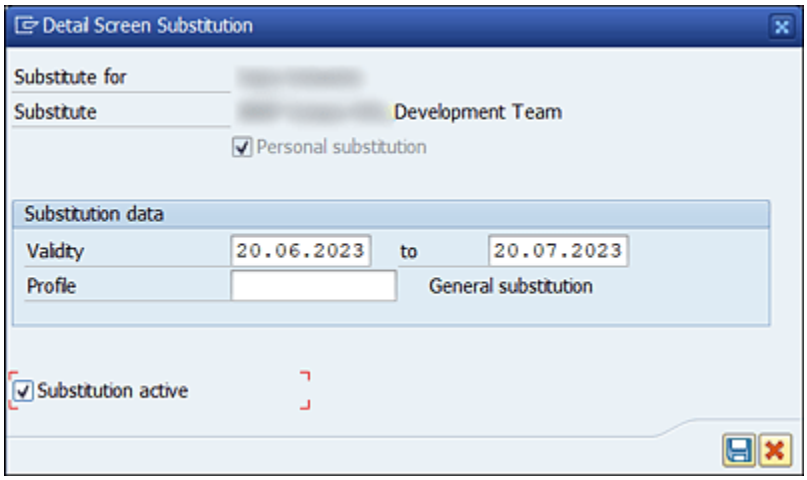


Figure 8.15 Substitute Detail Screen

The maintainable details are described here:

- **Validity**
  This will determine when the substitute will be able to access the main user work items.
- **Profile**
  This will control what kind of work items will be available to substitute. We'll discuss this in detail a little later.
- **Substitution active**
  When this is checked, the substitute will get the work items immediately in their inbox. This is known as active substitution.

For the substitute, it's important to know which work items he has received as a result of substitution and which ones the user is the original approver for. In the workflow inbox, the user needs to customize the layout and bring the **Substitute for** column with the visible layout. This column will show the name of the original owner of the work item. But once the substitute executes any work item, he becomes the owner of that work item. The **Substitute for**

field will be cleared out from that particular work item. When the substitute executes a work item (without even completing it), the work item will be reserved by the substitute. It will disappear from the main user's inbox. Let's say the main user has returned, and the substitute wants the in-process work items of the main user to go back to the main user's inbox. In that case, the substitute should replace the reserved work items.

Once the main user is back, he should end the substitution either by delimiting the substitution or deleting the substitution. You'll find the **Delimit Substitution** and **Delete Substitution** buttons in the **Maintain Substitution** screen.

Deactivation of the substation isn't enough to end the substation. If you deactivate the substitution, the substitute won't get the work items as default as he was getting for active substitution. But he can adopt the substitution and get the work items, which is known as passive substitution. For example, a senior manager wants to keep his assistant as his backup. His assistant will check his workflow inbox once a week and also process any work item on an ad hoc basis per the senior manager's instruction. This can be done using passive substitution.

Passive substitution can be activated similar to active substitutions. The only difference is that you don't have to activate the substitution (the **Active substitution** checkbox remain unchecked). Once the substitution is created, the substitute can adopt the substitution anytime during the validity of the substitution. When the substitute wants to see the main user's work item, he can follow the

SAP inbox menu path **Settings** • **Workflow Settings** • **Adopt Substitution**. Then, the original user work items will be visible in his inbox. He can switch back to his own work item view by ending the substitution view **Settings** • **Workflow Settings** • **End Substitution**. This way, the substitute can switch back and forth to see the original user's work item in his inbox.

To see the adopted work item view, a configuration is needed. Enable this function in the default workflow for personal settings. Go to Transaction SPRO configuration, and choose **SAP NetWeaver** • **Application Server** • **Business Management** • **SAP Business Workflow** • **Basic Settings (Workflow Systems)** • **Change Presettings for Personal Workflow Settings**. Select the **Also display work items of adopted substitutions** checkbox.

Substitution works at one level only. Let's say user A substitutes for User B. User B substitutes for User C. In this scenario, user C won't get the work items of user A. Substitutes need to be directed to the main user.

Another business scenario is very common where the user wants to do work item–specific substitutions. A user will have work items related to different business objects, such as HR, procurement, financial objects, or whatever. The user may want to substitute his manager for HR-related work items, someone from the procurement department for procurement-related items, and someone else for finance work items. A person may perform different roles in the organization, so he may need work item–specific multiple substitutions. This is achieved through a substitution profile.

Each task has an attribute for classification. This task classification needs to be linked to substitution profiles. A user can substitute different people for each substitution profile. The steps for this are as follows:

1. Create a classification of tasks. During workflow design, you have to come up with how all the workflows will be classified. It's recommended to classify how each workflow will be used in the system. Each task should not have a different classification. Classification can be based on business object or business process area or subarea. You can define a classification at a very high level such as process module (e.g., finance, HR, procurement) or at the business object level (e.g., purchase order, invoice, leave of absence, etc.). The number of classifications should be manageable.
   To define the classifications, go to **SPRO • SAP NetWeaver • Application Server • Business Management • SAP Business Workflow • Basic Settings • Maintain Task Classes**. Make sure the classifications have meaningful text. Figure 8.16 shows an example of task classifications (note the **INVOICE** classification has been create).

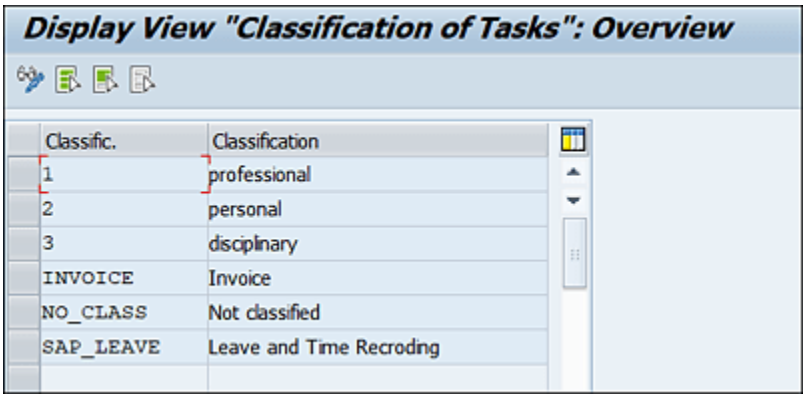


Figure 8.16 Classification of Tasks

2. Assign the task classification in the dialog task attribute. You can assign the task classification from the workflow step screen (see Figure 8.17) or from the task maintenance screen (see Figure 8.18).
3. Click the agent assignment task button  in Figure 8.17 or choose **Additional Data** • **Classification** • **Create/Change** from the menu path shown in Figure 8.18. You'll get the screen shown in Figure 8.19.

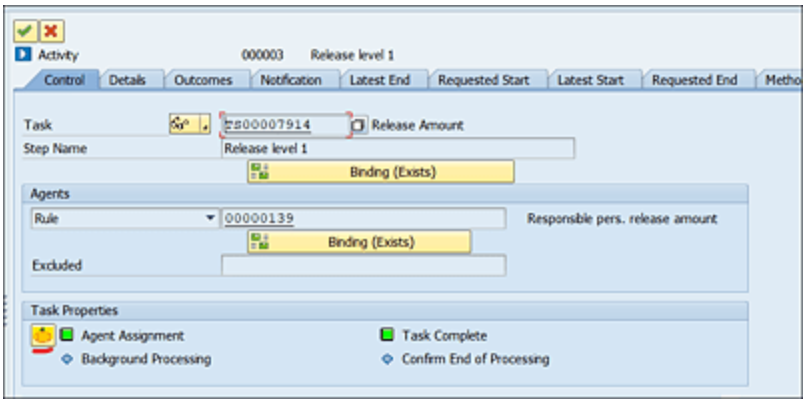


Figure 8.17 Maintain Task Classification in the Task Attribute Maintenance Screen

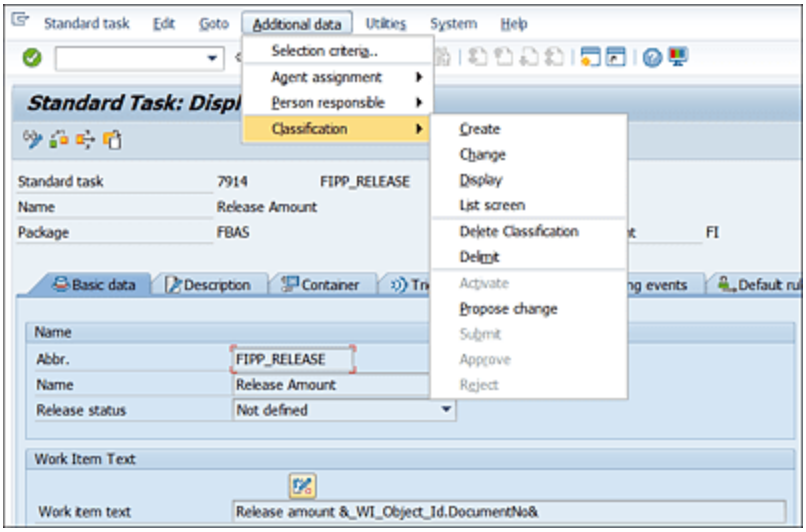


Figure 8.18 Create Classification from the Task Maintenance Screen

4. You have to select the classification value (e.g., **Invoice**) in the **Classification** field, as shown in Figure 8.19.

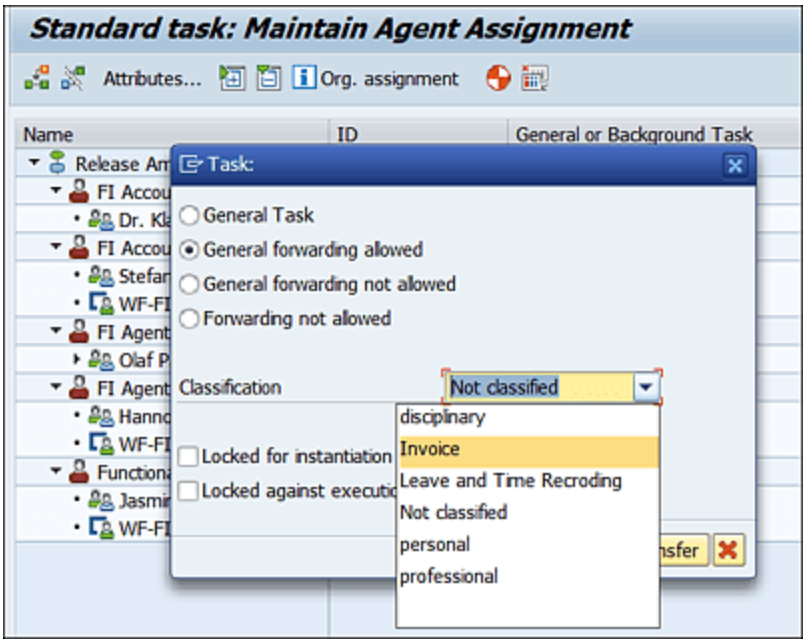


Figure 8.19 Maintenance of Task Classification

5. Create a substitution profile. You can create one substitution profile for one classification, or you can group multiple classifications into one group. To create a substitution profile, go to **SPRO** • **SAP NetWeaver** • **Application Server** • **Business Management** • **SAP Business Workflow** • **Basic Settings** • **Substitute Profile** • **Define Substitute Profile**. Create a new entry, and enter the name of the substitution profile and description of the profile. Figure 8.20 shows the substitute profiles (the **INVOICE** substitution profile is created here).

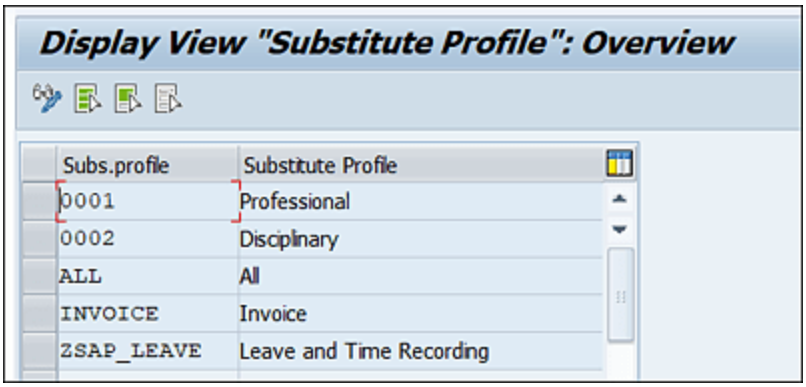


Figure 8.20 Substitute Profiles

6. Map/group the classifications against substitute profiles. Go to **SPRO • SAP NetWeaver • Application Server • Business Management • SAP Business Workflow • Basic Settings • Substitute Profile • Assign Substitute Profile**.
7. You have to create a new entry here. Enter the substitution profile and the classification name you've created in the earlier step. Figure 8.21 shows the substitution and classification profile. We've mapped the **INVOICE** substitution profile with the **INVOICE** classification in Figure 8.21.

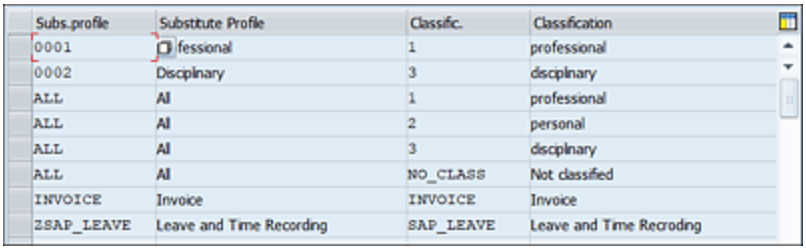


Figure 8.21 Assign the Substitute Profile to Classifications

We're now done with the configuration needed to set up substitution profiles. Substitution profiles are linked with classifications, and classifications are linked with tasks.

Therefore, there is a linkage established between the substitution profiles and work items.

Now the user can select the substitution profile when he is selecting the substitutes. Only one user can be substituted for each profile. You have to design your profile in such a way that the user should have the option for the correct selection. Figure 8.22 shows the substitute with the profile maintained.

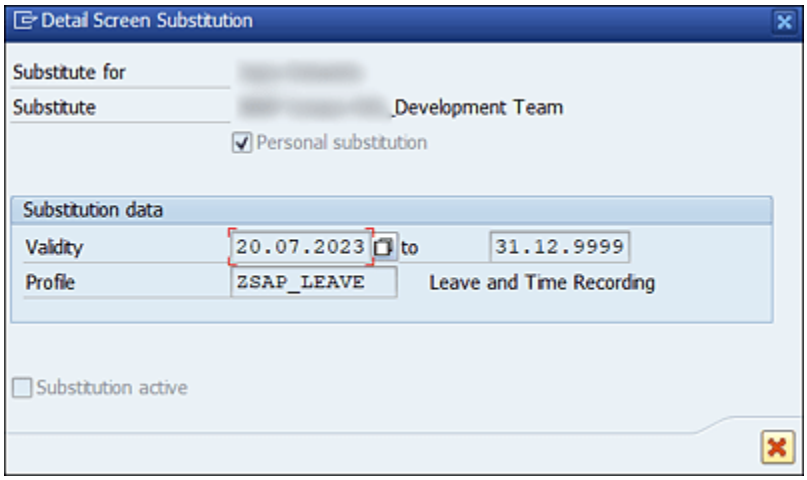


Figure 8.22 Substitute with Profile

There are some other business terms used in the context of substitution. In some places, it's mentioned as delegation or automatic forwarding of the work item as well. We've discussed manual forwarding of work items in the workflow administrator activities. For automatic forwarding of work items, you can set up a substitute. The office inbox features automatic forwarding of documents/mails. It will forward the email/task details to the external ID you've entered. You can set up the automatic forwarding of SAP office documents or tasks in email to an external mail ID by choosing **Settings** • **Office Settings** in the Business Workplace inbox, as shown in Figure 8.23. Your work item emails will be automatically

forwarded to this mail ID. You have to schedule program RSWUWFML2 in regular interval batch jobs or set up extended notification.

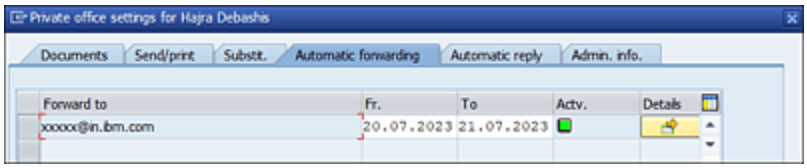


Figure 8.23 SAP Office Document Automatic Forwarding

8.5.2 Substitutions in SAP Fiori

SAP had provided substitute maintenance in the My Inbox app itself in SAP S/4HANA. You can maintain either planned or unplanned substitutes directly in SAP Fiori.

Planned Substitutes

Planned substitution is usually targeted for a scenario wherein you know the start date and the end date for your absence. Your substitute will then see your tasks directly displayed in his inbox for the period you defined.

Unplanned Substitutes

With My Inbox 2.0, you can also nominate a substitute for your *unplanned absence*. In this case, your substitute will need to accept the substitution to see your tasks in his inbox. Unplanned substitution could be looked at as a scenario where you assign permanent backup(s) for yourself. When you’re unexpectedly absent due to sickness or any other unforeseen reason, your backup

could take over your tasks and work on them until you're back.

Follow these steps to maintain substitutes:

1. Go to the My Inbox app, and click on the user profile icon on the top-right side of the SAP Fiori launchpad. Click on **Manage My Substitutes** ❶, as shown in Figure 8.24.
2. The **Manage My Substitutes** page will be opened. You can maintain planned or unplanned substitutes in different tabs on this page ❷.
3. Click on the **Add New Substitute** button to create a new substitute ❸.

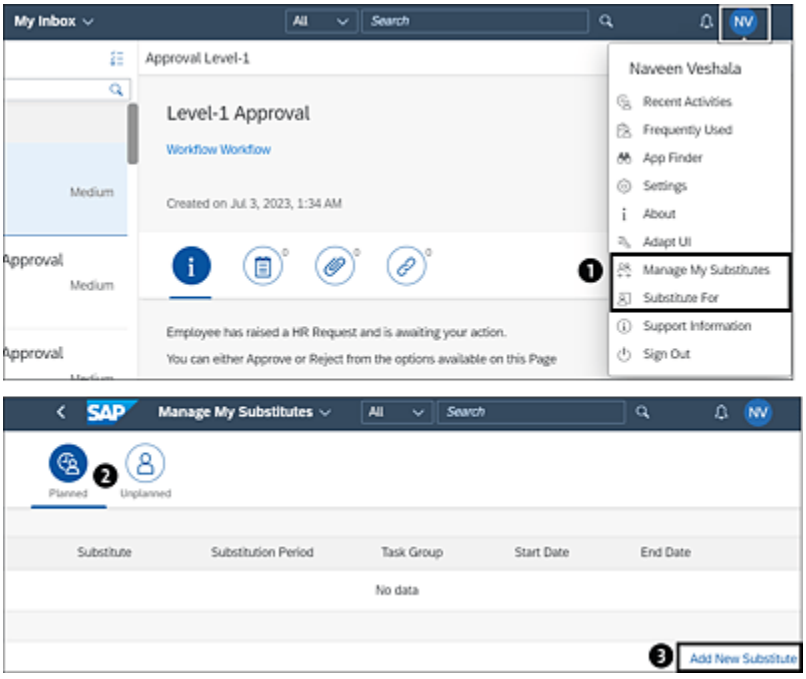


Figure 8.24 My Inbox Substitutes

4. Select the **Substitute** user ID from the popup window in Figure 8.25 ❶, and it will open another dialog to select

the substitution duration for that user ❸. You can also select the task group from the dialog ❷.

5. The selected user ID will be added as a planned substitution, which is shown in Figure 8.25 ❹.
6. Follow the same steps and add unplanned substitutions if needed. You need to select the substitution duration in case of unplanned substitution.

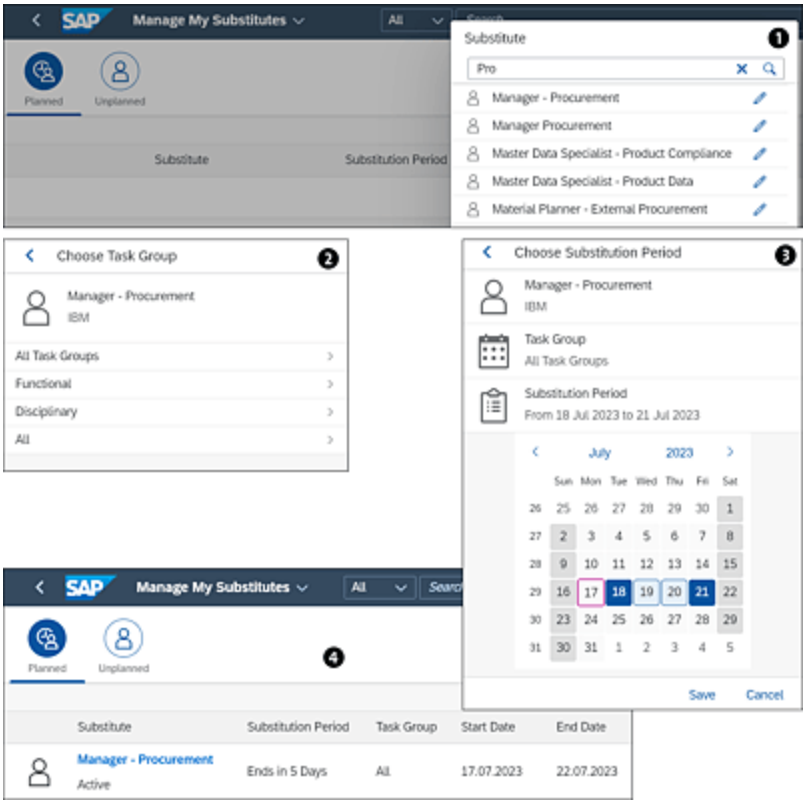


Figure 8.25 Add Planned Substitution

8.6 Display Dynamic Labels for Tasks to Display in Business Workplace

We've discussed the standard work item information available in Business Workplace. From the business data point of view, only the work item title is available in the Business Workplace work item section. There may be a need to have some other data as a column in the Business Workplace work item section. For example, *accounts payable clerk* is a group of people who process all invoices of the organization. They have an internal mechanism for allocating work items based on company codes. It would be helpful if the company code is provided as a column in the Business Workplace work item view, so then they can use the column filter of ALV to identify the work item that individuals need to process. In this case, we can add a dynamic column in the Business Workplace work item.

To add a dynamic column, you have to define the columns and map the container element value that will be shown in these columns for each task. If it's not defined for any specific task, then the field value will be blank. To define the dynamic column, go to **Tools** • **Business Workflow** • **Development** • **Definition Tools** • **Worklist Client** • **Dynamic Columns for Worklist** (Transaction SWL1).

Figure 8.26 shows an example of how this configuration can be done. You have to enter the **User Name**, **Task** number, **Header** label, and container element (**Printout** field) for

each dynamic column. You can enter a maximum of 12 dynamic columns. In the **User Name** field, if you want this dynamic column for all users, then enter “*”. Then, enter the work item task number in the **Task** field. Enter the container element of the task where the system will read the value that will be shown dynamically. Enter this container element in the **Printout** field. Enter the label for this column in the **Header** field.

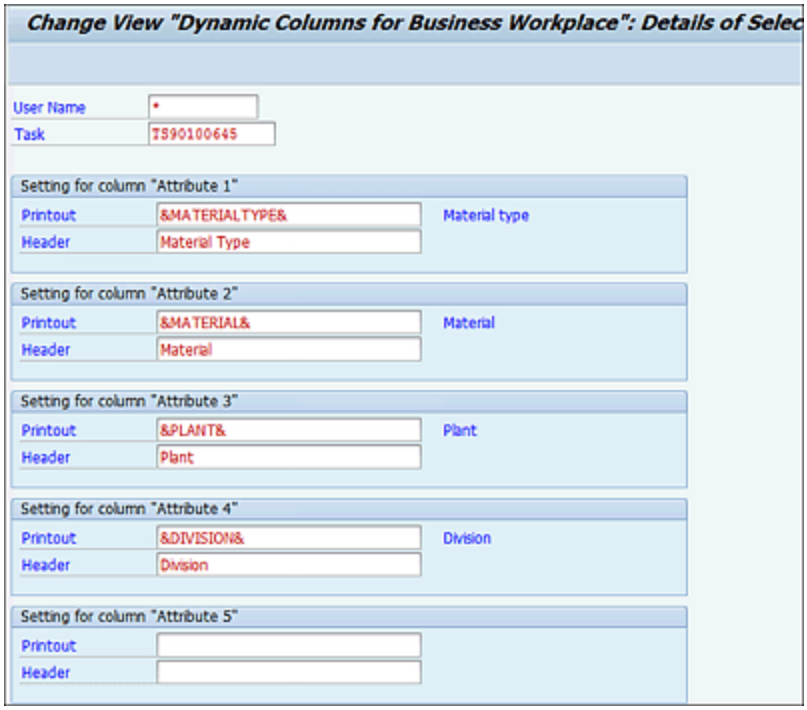


Figure 8.26 Dynamic Column Configuration for Business Workplace

You have to configure this for each task. Although this configuration can be user specific, in productive systems, it’s usually done for all users. You have to add the task number, column header, and container name that will be displayed as the data for the column. Dynamic columns are very useful for sorting, grouping, and filtering purposes.

8.7 Event Trace and Event Queue Administration

A workflow can be triggered directly from an application or via an event. The workflow administrator manages and controls the behavior of events and their receivers. There are mainly three areas to manage: event linkage configuration, event trace, and event queue.

The linkage between an event and its receiver is maintained in the event linkage transaction. Go to **Tools • Business Workflow • Development • Utilities • Events • Type Linkages** (Transaction SWETYPV) to manage the event linkage. More details about event and receiver linkage is discussed in Chapter 5.

Event trace will be helpful to troubleshoot issues if any workflow isn't triggered. You can activate the event trace and then trigger the business application to check if the event is triggered and received for that event. You can activate the event trace via **Tools • Business Workflow • Development • Administration • Event Manager • Event Trace • Switch Event Trace On/Off** (Transaction SWELS). The details of the triggered event will be found in Transaction SWEL via menu path **Tools • Business Workflow • Development • Administration • Event Manager • Event Trace • Display Event Trace**. Event trace is generally turned off in the production environment. It can be activated to troubleshoot issues when any specific workflow isn't started from an event.

The event can be managed using the event queue. It will improve the overall performance of the system and will help to manage the errors and monitoring of the events. To process an event through the event queue, you have to enable the event queue in the event linkage. Go to **Tools** • **Business Workflow** • **Development** • **Utilities** • **Events** • **Type Linkages** (Transaction SWETYPV) to enable the event queue, as shown in Figure 8.27.

If the **Enable Event Queue** checkbox is activated, then the event receiver is started via the event queue. If the indicator isn't checked or event queue isn't active, then the receiver is started immediately.

If the event queue is activated, the event triggering information is stored in temporary memory. It's then processed gradually from that queue. If there is a sudden surge of events, event queue manages that with even distribution of the workload. So, there will be a little delay between the event being triggered and the receiver starting. Events with active linkage information are only stored in the temporary memory. Events with errors are also stored in the event queue and can be reprocessed from the queue.

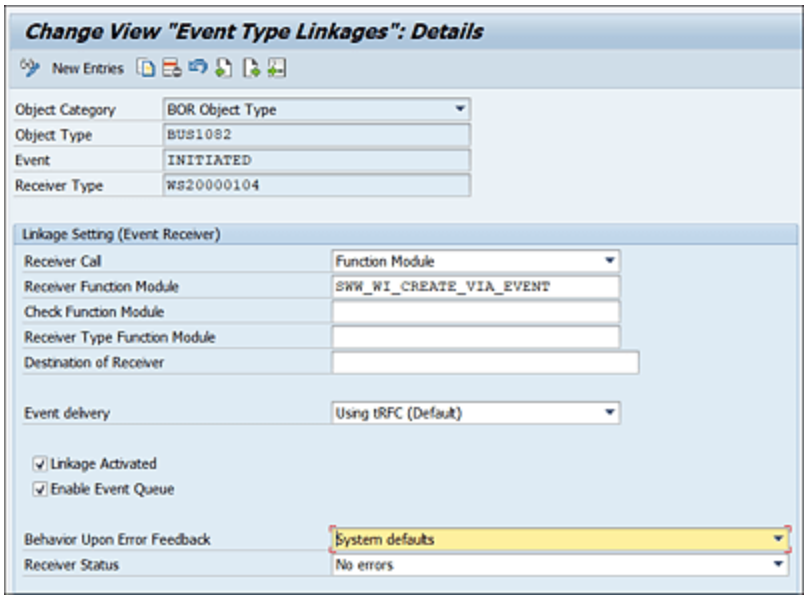


Figure 8.27 Event Linkage

Event queue is administrated from Transaction SWEQADM via menu path **Tools** • **Business Workflow** • **Development** • **Administration** • **Event Manager** • **Event Queue** (see Figure 8.28).

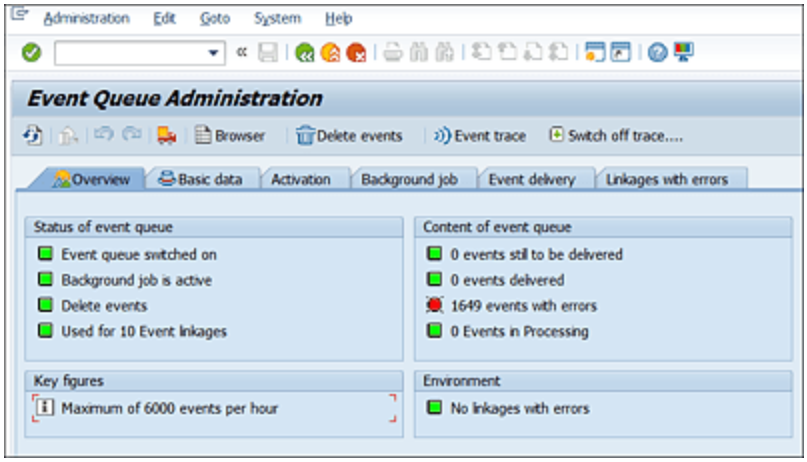


Figure 8.28 Event Queue Administration Overview

The **Overview** tab will show the summary status of the event queue, including the summary of the event queue

and all function statuses. Now let’s look at the rest of the tabs:

- **Basic data**
  The event queue administrator and the event error behavior are maintained in the **Basic data** tab (see Figure 8.29).

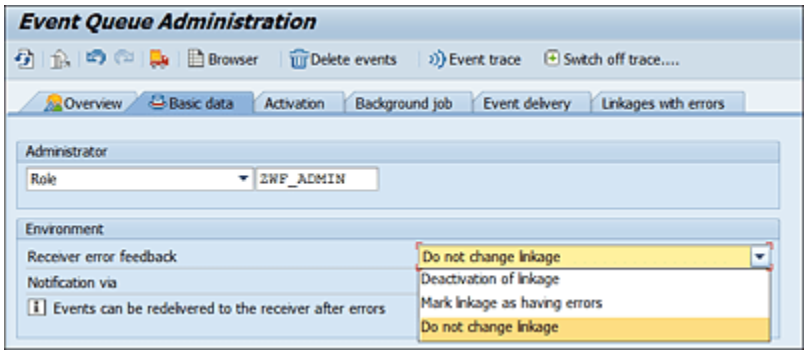


Figure 8.29 Event Queue Administration Basic Data

  You maintain the administrator here. You can maintain any specific user, role, organization unit, position, work center, or job as well. This perform will get notifications for event errors. It’s recommended not to use any user IDs here; instead, use other options relevant for your project. You don’t have to change this setting if multiple people are added as administrators or there is a change in the job role.
  Default system behavior in case of event error is defined here in the **Receiver error feedback** field. There are three options:

  - **Deactivation of linkage**
    The event and receiver linkage will be deactivated when any event triggering error happens. That will stop all subsequent receivers from starting. Because event linkage will be deactivated, event information won’t be

stored in the event queue, so retriggering of these events from the queue won’t be possible. We don’t recommend this setting because it doesn’t provide any reprocess option.

- **Mark linkage as having errors**
  In this setting, the event linkage isn’t deactivated, but the event receiver status is marked as **Error**. Subsequent event information is stored in the event queue. You can fix the root cause of the event processing error. Then, change the event receiver status as **No Error** either in the event linkage transaction or in the **Linkage with error** tab. You can then redeliver the events from the **Linkage with error** tab.
- **Do not change linkage**
  In this setting, no change happens in the event linkage. In case of an error, the event information is stored in the event queue. The system will keep on triggering the subsequent events. If there is any error in subsequent event processing, then it will be stored in the event queue. The difference between this one and the earlier one is that in the earlier one, the system won’t start processing the receiver after the first event is errored out. All subsequent event information will be stored in the queue. In this case, the system will try to process subsequent events after the first error happened. Normally, the event error is caused by parameters. Parameters are specific to any business object instance, so if an error occurs for a particular instance, it doesn’t mean that similar error will occur for all instances of

that business object. We recommend using this setting in the event queue.

- **Activation**
  You'll activate the event queue in this tab. You can switch the event queue **ON**/**OFF** here. It provides you a link to navigate to the event linkage transaction where you can activate the event queue for event receiver linkage.
  You can also set whether the event information will be deleted after the event delivery. It's recommended to delete the event information after delivery to improve event processing speed.
  The event manager tries to start the event receiver based on any triggered event. If there is any error, the information is passed back to the event manager. If the nature of the error is temporary, event manager will try to process the event again. The maximum retry count is maintained in the **Retries when temporary error** field. The error will become permanent after the system reaches the retry count.
- **Background job**
  You'll schedule the background job to process the event in this tab. The operation mode of the background job is defined here. You can set the job as **Periodic** or **Depending on Load (Dynamic)**. For periodic background jobs, the job will be executed within a fixed interval, and the number of records it will access is entered in the **Number of Events per Read Access** field. For dynamic operation mode of the background job, it will consider the time interval between two read accesses as maintained here till the queue has some events to process. If the queue doesn't have any events

to process, then it will consider the interval until event queue next checks field for the next execution. The dynamic option provides some benefits in terms of system load.
If you want to change any parameter for the periodic job, you have to unschedule the background job first. Then you can change the parameters.
You can schedule the background job for the event queue from here.

- **Event delivery**
  You'll define how the event receivers are started in this tab. Event receivers can be started synchronously (RFC) or asynchronously (tRFC). There are two options: **Sequential** or **Parallel** in the server group.
  For **Sequential**, it uses one application server and processes the event receivers one by one. It can wait till the receiver function is completed (synchronous), or it can continue with the next call without waiting to complete the receiver function (asynchronous). It's recommended to use the asynchronous mode of operation.
  You can process in parallel using a server group. Then, certain servers will be dedicated for this process. Therefore, there won't be an impact on other users if there is certain rise in the number of events. As of now, only synchronous processing is supported in parallel processing with server groups.
- **Linkage with errors**
  You'll able to handle the events with errors in this tab. This tab will show all the event events with event receiver error. You can reprocess those events from here.

As shown in Figure 8.30, you'll find all the events with errors. You can reset the receiver feedback to **No Error** by clicking the **Linkage Status** button. You can reprocess the events from here. During the reprocess, you can reprocess one event at a time or reprocess all at once. If the events can't be reprocessed, it needs to be retriggered from the business application, so you can delete those events once the event is retriggered from the business application. The **Delete Event** button will delete all the events (count is showing in the column number). You should reprocess the error and make sure you want to delete the remaining ones because they can't be reprocessed.

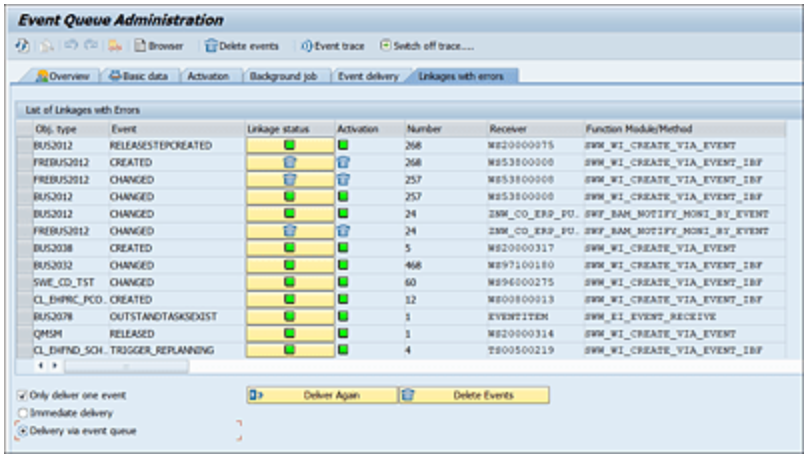


Figure 8.30 Event Queue Linkage with Errors

8.8 Process Incoming Documents with ArchiveLink

There is a common business process where the workflow needs to be started against an external document that is received. This can be for incoming vendor invoices; any customer, vendor, or material master specification; or any other business object where creation of the system document starts from a paper document. Let’s take the example of a vendor invoice. Per the business process, the vendor sends an invoice either as a paper document or electronically. High-level business requirements for this process will be as follows:

- There should be an SAP application document created for the invoice.
- The relevant data from the invoice document should be captured in the business transaction.
- The paper/electronic invoice image should be linked to the business application document for any reference.
- The invoice should be sent for approval and posted after all necessary approvals.
- This requirement is designed and put into practice with standard workflows with the ArchiveLink feature.

As mentioned earlier in this chapter, ArchiveLink is the SAP integrated service that links the archived document with the SAP businesses application document. When you upload any document in SAP or send any documents as output

attachments, you can store those within the database or in an external storage. But it doesn’t make sense to store these in a costly database. So, any incoming documents, outgoing documents, print outputs, and archived files are stored in an external content server. SAP provides standard integration to link these documents with SAP business application objects. This linking is done via ArchiveLink. Figure 8.31 shows a schematic diagram of content storage and linkage.

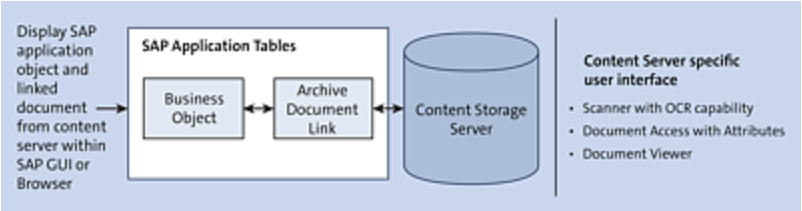


Figure 8.31 Content Server and ArchiveLink Overview

In general, the content server products provide functions to scan a physical document or retrieve electronic documents from any common inbox, store it in content server, and provide a URL that can be opened from a standard browser. It can also have an optical character recognition (OCR) system to read the content of the document and prepopulate any business objects. There are few variations of this process we’ve seen in various clients, as follows:

- **Variation 1: Storing for subsequent entry**
  - Step 1: The vendor sends an invoice either electronically to a common mailbox or in the form of physical paper.
  - Step 2: Someone in the support desk scans the physical paper. For electronic documents, the system will scan the inbox and send the attached document to the

content server. In both cases, the content server will store the document and will share a URL to the SAP application.

- Step 3: An SAP application will create a work item with the attached URL link.
- Step 4: The agent of that work item will open the task. It will show the attached document in a browser. The system will also open the business transaction (Transaction FV60, in this case) with defaulting the document type (this is just an example).
- Step 5: The user will enter the required data and save the business transaction invoice. When the invoice is created in this step, the content server document link will be linked to this invoice using ArchiveLink.
- Step 6: The saved invoice may go further in the approval process per system design.

- **Variation 2: Storing for subsequent assignment**
  - Step 1: The vendor sends an invoice either electronically to a common mailbox or in the form of physical paper.
  - Step 2: Someone in the support desk scans the physical paper. For an electronic document, the system will scan the inbox and send the attached document to the content server. In both cases, the content server will store the document and will share a URL to SAP application.
  - Step 3: An SAP application will create a work item with the attached URL link.

- Step 4: The agent of that work item will open the task. It will show the attached document in a browser. The business document is already created for this document, so the user enters the document number that should be linked to it. The system will link the content server link with the business application using ArchiveLink.
- Step 5: The saved invoice may go further in the approval process per the system design.

The difference between the two processes is the business transaction didn’t exist when the document was sent by the vendor. In the second variation, the business document already exists. There may be another supporting document the vendor has sent that needs to be linked to an existing invoice document. There can be further sophistication with OCR software where the system can read the data from the document and prepopulate the business document. In a few cases, it can create a parked document with the information scanned. In some cases, you may have to create a custom application to capture the data scanned and read by the OCR system.

All these scenarios can be achieved with a storage scenario integrated with business workflow which use Archivelink.

To use ArchiveLink, you have to do the basic configuration to connect the content repository server with SAP. The Basis team in your project will help you set this up. The configuration can be done via configuration path **SAP NetWeaver** • **Application Server** • **Basis Services** • **ArchiveLink** • **Basic Customizing**. The key configuration nodes are as follows:

- **Define Content Repositories**
  Define the logical units of the content server. You can have different logical units by system or business area.
- **Edit Document Classes**
  Define the type of document such as PDF, XML, ALE, DOC, and so on.
- **Edit Document Types**
  You have to create the document type. This document type will group similar kinds of business documents. The document type will also be linked to the storage content area. For example, you can create a document for an invoice that isn't a purchase order.
- **Edit Links**
  You'll link the ArchiveLink document type you've created in the earlier step with an SAP business object, content repository ID, and an SAP table that will connect the SAP object and ArchiveLink. Creating the document type and maintaining this link can also be done using the Document Type Customizing wizard. The business object linking with ArchiveLink should support interface `IFARCH21`. The interface has method `ArchivedDocsDisplay`, which displays the ArchiveLink objects linked to the business object. This business object will only display the ArchiveLink object. If you have to also open the transaction automatically for data entry along with displaying the ArchiveLink object, then you have to use object `IMAGE`, which also contains interface `IFARCH21`.

We've already set up ArchiveLink with the document type. The document type is also linked with an ArchiveLink

business object so that when the method gets executed, it can open the ArchiveLink document.

Now we have to link the document type with a workflow and an SAP business object where the link will be saved. SAP has provided two standard tasks for this activity. For the entry scenario (variation 1 scenario), SAP has provided task `TS30001128`, and for the assignment scenario (variation 2 scenario), it's `TS30001117`. You can use the document type Customizing wizard to link all these elements. You can find this Customizing IMG configuration via path **SAP NetWeaver • Application Server • Basis Services • ArchiveLink • Customizing Incoming Documents • Workflow Scenarios • Use Document Type Customizing Wizard**. Figure 8.32 shows all the elements of the document type and workflow linking customization.

Figure 8.32 Document Type Customizing Wizard

This ArchiveLink document needs to be linked to business object financial accounting document `BKPF`. In this context, your configuration will look like following:

- **Document Type Configuration**
  - **Document Type**: ZFUCSNONPO
  - **Document Class**: PDF_SCAN

- **Maintain Links between Document Type and Object Type**
  - **Object Type**: BKPF
  - **Document Type**: ZFUCSNONPO
  - **Relationship ID**: TOA01
- **Assign Document Type to Workflow**
  - **Document Type**: ZFUCSNONPO
  - **Object Type**: BKPF
  - **Method**: CREATE
  - **Task (Entry Scenario)**: TS30001128
  - **Task (Assignment Scenario)**: TS30001117
- **Maintain Workflow Parameters**
  - **Document Type**: ZFUCSNONPO
  - **Parameters**: TRANSACTIONCODE
  - **Contents**: FV60

Now you have to configure the responsible agent for the workflow. This is defined within the document type Customizing wizard, or you can set in the configuration node via **SAP NetWeaver** • **Application Server** • **Basis Services** • **ArchiveLink** • **Customizing Incoming Documents** • **Workflow Scenarios** • **Maintain Presettings for Storage Scenario**. You can create default settings. Under the default settings, you can set the responsible agent for each document type defined in the preceding list. You can use all elements of the workflow agent such as position, user, work center, organization unit

job, and so on. However, you can’t dynamically determine the agent in this scenario.

If the standard tasks `TS30001128` and `TS30001117` don’t meet your business requirements, then you can create your own custom workflow. Follow menu path **SAP NetWeaver** • **Application Server** • **Basis Services** • **ArchiveLink** • **Customizing Incoming Documents** • **Workflow Scenarios** where you’ll use the workflow wizard to create your own workflow. The three functions of display scanned document, enter/assign business object, and link to business object will be three separate tasks in the custom workflow. You can add your own steps in this custom workflow as well. You can also write your own agent determination rule here.

If you want to start an approval workflow of the business object after the ArchiveLink is established, you can also set it up on the back of an object link. Per the business process, once the user key is in the data, it needs to be sent for business approval based on the specific object approval matrix. When the archive link is completed, the system will trigger event `ASSIGNED`. You can set up to trigger another approval workflow using event linkage. You can use a separate standard workflow or a custom workflow that will be triggered based on the assigned `EVENT`.

The workflow for incoming documents is a very common scenario in all organizations. You can adopt a standard SAP-provided solution to easily integrate the document ArchiveLink with the business application.

8.9 Summary

You've learned in this chapter how to maintain, administrate, and troubleshoot workflows. We've discussed common workflow errors you may encounter as a workflow developer and administrator and what your approach should be to resolve them. We've explained in detail the transaction and workflow log that will be helpful to troubleshoot workflow errors. We've discussed Business Workplace where user will execute all work items. It's important to know all standard features available to help you to guide users to effectively use the workflow capability. It will also help you design the work items while keeping end user usability in mind. We have a common administrator maintenance activity on how to reassign any work item when the original responsible user ID is out of the office for a longer period of time. We discussed substitution to handle these situations.

We also discussed a very special and common workflow scenario utilizing the ArchiveLink feature. A workflow is triggered based on a scanned document stored in an archive repository. We talked about a couple of commonly used scenarios and respective solutions, including the system configuration required. You also learned how to use workflows to handle errors in IDoc processing. In the next chapter, we'll discuss the reporting capability of workflows and how these reports can help you to further improve the workflow capability in the organization.

9 Application Link Enabling and Reporting

This chapter discusses how classical workflows can be leveraged for automating Application Link Enabling (ALE) and IDoc processes, especially in the areas of inbound IDoc error handling and notifications, outbound error notifications, and active monitoring. Additionally, we'll introduce you to the common workflow data tables, application programming interfaces (APIs), and standard reports. Lastly, we'll talk about program exits and how they can be leveraged for collecting data required for your custom workflow reports.

We'll discuss in this chapter how workflows can help automate the error handling in integration and different kinds of reporting requirements in the workflow domain. When we think about workflow, the first thing that comes to mind is integrating different business process steps together. Workflow helps to minimize the lag time for handover between different step, and it provides a robust mechanism of tracking where a process is stuck. Workflow is also very helpful in automating IDoc interface error handling. There are thousands of IDocs processed in an organization every day. Some of those IDocs fail in the

application layer due to different reasons. Some organizations set up a manual process to monitor these failed IDocs and notify responsible users, but there will be a delay in any kind of manual email-centric process. The user or management won’t be able to track what has already processed and what is remaining. Normally, these manual processes are done once a day, so the system won’t have the capability to process the errors immediately.

You’ll learn in this chapter how you can set up a workflow that will be triggered immediately after an IDoc failure. It will also send a work item to the user to facilitate the reprocess from his business user inbox. We’ll also discuss how a summary notification can be sent to supervisors for any business-critical IDoc that needs to be processed immediately. In the second part of the chapter, we’ll discuss the general reporting requirements you get in the workflow area, the standard reports available, and the strategy for building the reporting capability.

For IDoc error handling, we’ll consider one business scenario where IDoc error handling is required. We’ll walk step-by-step through the configuration guide to set up workflow events and tasks to trigger a work item whenever there is an error in IDoc processing. We’ll discuss how to determine the recipients who will receive these work items. We’ll also cover the settings for both handling IDoc errors and also notifying the user of the successful IDoc posting. Then, we’ll discuss active monitoring, which is used to notify supervisors when any predefined IDoc threshold of error is reached. This will provide management visibility if any critical process will be impacted.

In the second part of the chapter, we'll cover some of the reporting requirements on the performance of the workflow, operation reporting to find where any work item is stuck, and overall tracking of workflow progress. In most of the scenarios, there will be a need to build custom reports to meet the business requirements, so we'll provide you with a list of common workflow tables and APIs that will help you build custom reports to meet business requirements.

We'll also talk about program exits in workflow. Program exits provide hooks where you can get some runtime information and store that information in custom tables. This information can be used later for analytics and other reporting requirements.

9.1 Error Handling during IDoc Processing

IDocs are widely used to exchange business application data between systems. Any IDoc failure should be investigated and reprocessed to ensure business continuity. There is always a business need to monitor IDoc failures and reprocess. This is established with manual monitoring and Excel-based tracking. That delays the reprocessing, and you'll lose track of what could be successfully processed and what needs further manual intervention. Workflows can be set up to handle IDoc failures. In this section, we'll consider one business scenario of handling sales order IDoc creation and explain how to set up and test IDoc failure handling using workflows.

We'll start by discussing a business scenario when IDoc error handling via workflow is needed. We'll cover the basic framework to set up the error handling and then elaborate on the configurations required to set up a workflow to trigger when an IDoc goes into error. Similarly, we'll go through the setup needed to notify on successful posting of IDocs. We'll also describe how to test this configuration setup in a development environment for unit testing. At the end, we'll talk about an exception scenario when IDoc processing start is controlled by the workflow. Normally, IDocs are processed immediately after they reach the target system. But in some exception cases, a user decision is needed before it can be processed, and that is controlled using a workflow.

9.1.1 Business Requirements for Inbound IDoc Error Handling

The customer order is a key element in nearly every business, so orders are important from a business point of view. Businesses should add extra reliability in its creation process and enable groups of persons to react quickly if any errors occur. This section covers how to handle errors that occur during the order creation via IDocs.

When an inbound sales order comes in via IDoc (through middleware), an error occurs occasionally that prevents the order creation in SAP S/4HANA. A customer care organization needs to be aware of these errors, so that they can take corrective action and notify the customer. If for any reason, the IDoc is in an error status such as **51**, a workflow should be triggered for members of a group, so that those

users can process the IDoc in the foreground to rectify the error. We'll explain this scenario with IDoc message type `ORDERS`.

Let's start with a simple requirement. When an IDoc with message type `ORDERS` fails, it should go to a dedicated group of people to reprocess. The basic solution is as follows:

1. An event is triggered for every IDoc failure.
2. There is a foreground task that can be triggered on the back of this event. The business object for subtype `IDOCAPPL` has standard method `INPUTFOREGROUND` to process the IDoc in the foreground.

Now, let's first configure the standard solution available. Then we'll discuss some custom variations for the requirements.

9.1.2 Handling the Inbound IDoc Error

Now we'll discuss the different configuration needed to trigger the workflow and process the work items:

1. First, we'll configure the partner profile in Transaction WE20 for the **ORDERS** message type. The partner profile setup is client dependent, so you have to set it up in each system and client. Figure 9.1 shows the partner profile configuration where the message type is linked to the process code.
2. When setting up a partner profile, you have to enter the partner number (**Partner no.**) that is the logical system name from where the IDoc is coming. Enter the message type name (in this example, "ORDERS") and **Process**

code. These are standard IDoc configurations, not specific to the workflow, so we won't discuss any further details here about the partner profile setup.

3. For postprocessing, permitted tab information is used for the IDoc error-handling scenario. The standard agent determination rule will send the error IDoc work item to the person ID as mentioned in the **Post Processing: Valid Processors** tab. You can enter a specific user ID or work center, job, organization unit, person, or position here. An example with user ID set is shown in Figure 9.2.

The function module that will process this inbound IDoc and events for any error is maintained against the process code. All these processing details are maintained in Transaction BD67. Figure 9.3 shows the process code setup details. For any standard SAP-provided process code, this configuration will already be there. If you've created a custom process code to process a custom IDoc, then you have to enter **Object Type**, **Start Event**, and **End Event** in the block IDoc in this transaction.

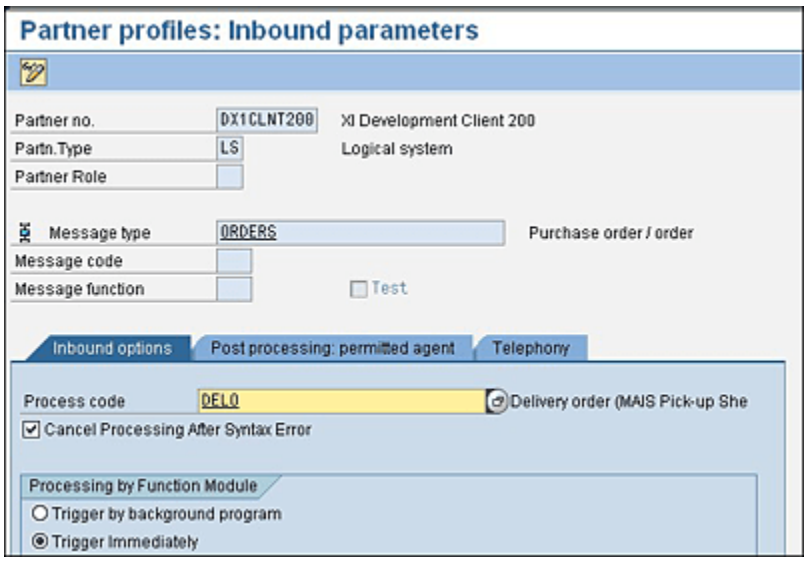


Figure 9.1 Partner Profile Configuration

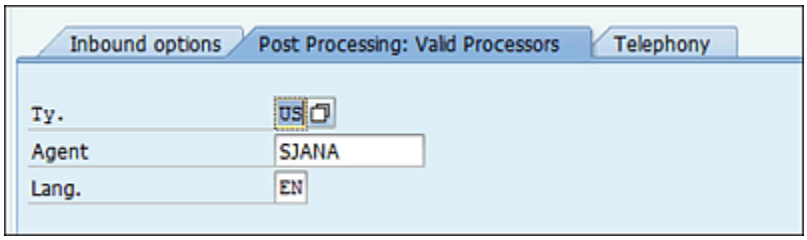


Figure 9.2 Post Processing Agent for IDoc

IDOCAPPL is the main business object for any IDoc error handling. A subtype of this business object is created for specific message types. In our scenario, SAP has created subtype IDOCDELORD of business object IDOCAPPL for delivery order IDoc error handling. Per Figure 9.3, whenever any ORDERS IDoc fails, SAP will trigger event INPUTERROROCCURRED of business object IDOCDELORD. If you're creating a custom process code, then you can create a subtype of business object IDOCAPPL.

The event details in case of IDoc failure are maintained in the **IDoc** section. Business object programming for IDoc failure handling is written in the method InputForeground and InputBackground of business object IDOCDELORD. You'll find these two methods for all IDoc error-handling business objects.

Now you know what event will be triggered when the IDoc fails. You need a workflow to be triggered based on the IDoc error event. You can find the standard task SAP has provided to process this error by going to business object builder Transaction SWO1, entering the business object name in the **Object Type** field (in this case, "IDOCDELORD"), and searching the where-used list. You'll get the standard SAP task to handle this specific IDoc error. SAP has provided TS20000117 for IDOCDELORD.

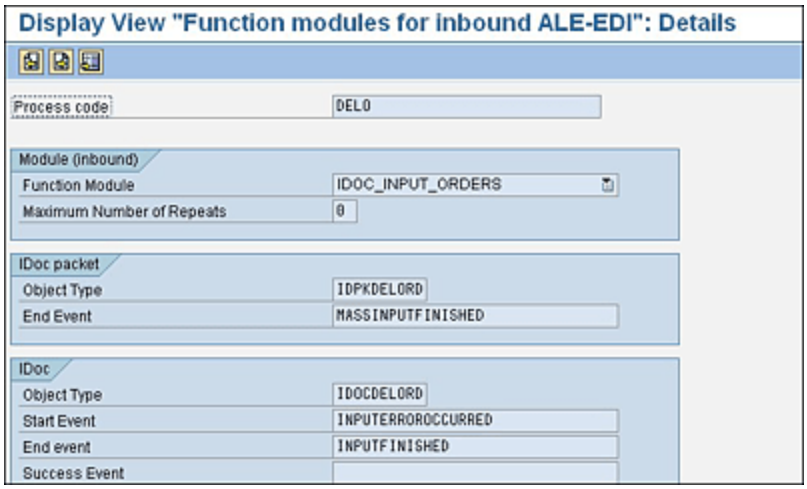


Figure 9.3 Process Code Configuration

Figure 9.4 shows the attributes of task `TS20000117` to process `DELORD` IDoc errors.

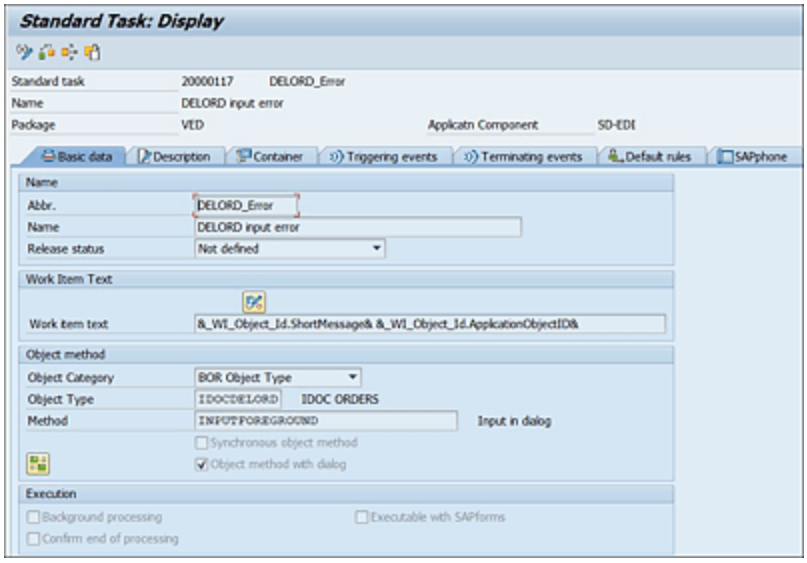


Figure 9.4 Standard Task to Process DELORD IDoc Errors

This task uses business object `IDOCDELORD` method `INPUTFOREGROUND` to process the IDoc in foreground. The business object is the subtype of `IDOCAPPL`. So, if you have to create an error-handling method for a custom IDoc, you also have to create your business object as a subtype of `IDOCAPPL`.

You'll find the triggering event to start this task and the event's details in the **Triggering Event** tab. The task will be triggered by triggering event `IDOCDELORD-INPUTERROROCCURRED`, as shown in Figure 9.5.

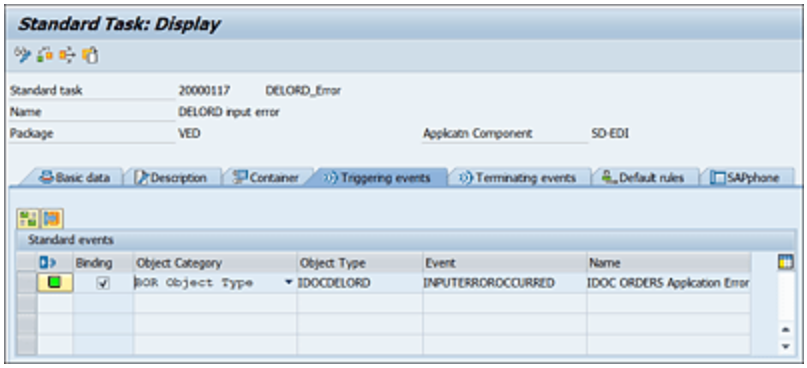


Figure 9.5 Triggering Event for IDoc Error Handling Task

The event linkage will already be maintained for standard SAP-delivered tasks, but it won't be activated by default. You can activate the event linkage by clicking the **Event Settings** button . In the popup that appears as shown in Figure 9.6, select the **Enable linkage activated** and **Enable usage of event queue** checkboxes, and save the settings. This will activate the event linkage.

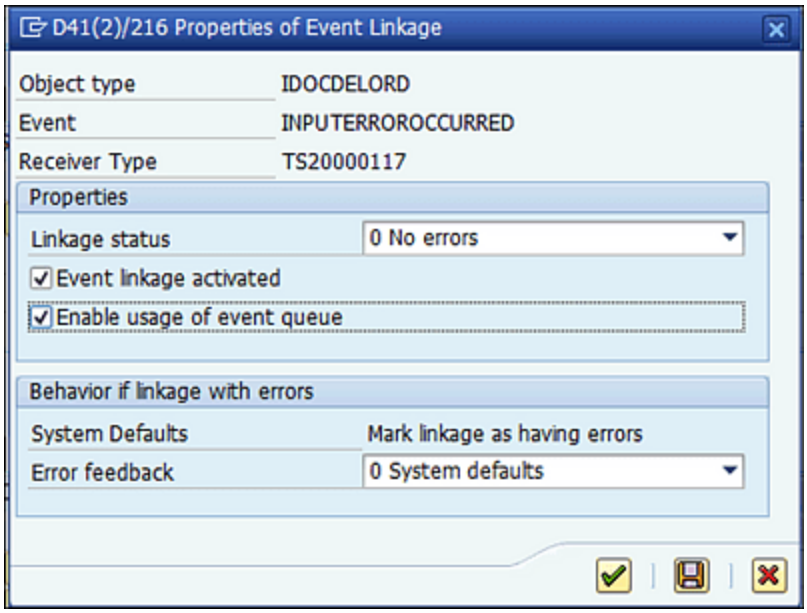


Figure 9.6 Activating Event Linkage for the IDoc Failure Processing Task

This task is an asynchronous task, so a terminating event is needed to complete the work item once the IDoc is processed, as shown in Figure 9.7. This setting will already be there in the predelivered SAP standard task, so you don't have to do anything here. This is given as a reference in case you're creating a custom task to process a custom IDoc.

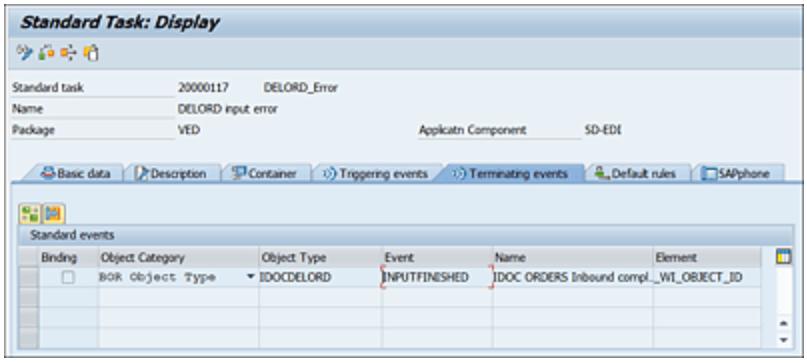


Figure 9.7 Terminating Event to Complete the Work Item

Now, who will get this work item? This is maintained in the **Default rules** tab, as shown in Figure 9.8. Standard SAP

default rule 20000046 reads the postprocessing agent from the partner profile to determine the agent. You may have different logic to determine the agents to process the workflow. In that case, you have to create a task similar to TS20000117 and a custom rule. Write your own logic to determine the agent. Assign that custom rule as a default rule for the task you've created. Refer to Chapter 6, Section 6.2, for details on how to create a custom rule.

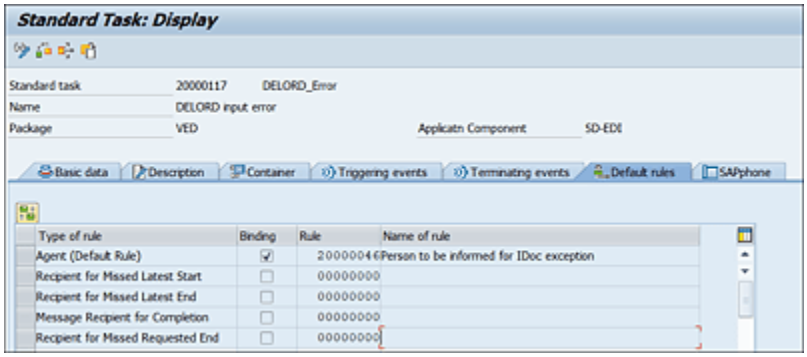


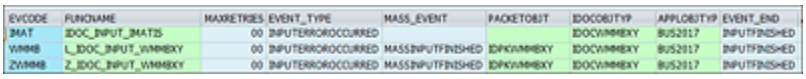
Figure 9.8 Default Rule for Work Item Agent to Process the Workflow Task

As you've seen, a triggering event is linked to the process code. It may happen that the same event is triggered for multiple message types, but you want to set up the workflow error handling only for a specific message type. You can write the start condition in the condition editor (Transaction SWB_COND) to filter further. You should check all the active IDoc error-handling events in the system; validate which ones are needed; and, if any further filtration is needed, only start the workflow for the required one.

The easiest way to find out how many process codes are linked to the same IDoc error-handling event is to check the content of table TBD52. Go to Transaction SE16, and enter table name "TBD52". Click on the table contents. In the selection screen of the table TBD52 data browser, enter

“EVENT_TYPE = INPUTERROROCCURED” and “IDOCOBJTYP = The Business Object”, and execute this selection.

If you find multiple entries in this table, that means the same error-handling event will be triggered for multiple message types. If you don’t want to activate the IDoc error-handling process for all, then you have to further filter using the event filtering start condition. Chapter 5, Section 5.3, provides details of how to create event start conditions. Figure 9.9 shows an example where the same IDoc error-handling event is used for multiple process codes.



EVCODE	FUNCNAME	MAXRETRIES	EVENT_TYPE	MASS_EVENT	PACKETOBJT	IDOCOBJTYP	APPLOBJTYP	EVENT_END
IMAT	IDOC_INPUT_IMATIS	00	INPUTERROROCCURRED			IDOCWMMBXY	BUS2017	INPUTFINISHED
WMMB	L_IDOC_INPUT_WMMBXY	00	INPUTERROROCCURRED	MASSINPUTFINISHED	IDPKWMMBXY	IDOCWMMBXY	BUS2017	INPUTFINISHED
ZWMMB	Z_IDOC_INPUT_WMMBXY	00	INPUTERROROCCURRED	MASSINPUTFINISHED	IDPKWMMBXY	IDOCWMMBXY	BUS2017	INPUTFINISHED

Figure 9.9 Same IDoc Error-Handling Event Used for Multiple Process Codes

If you don’t verify this, you may find that lots of unnecessary work items are created and sitting in someone’s inbox. Normally, these tasks are generic tasks. If the post processing agent in the partner profile isn’t maintained, then these work items will go to everyone in the organization. So, this verification is a very important part of the system setup.

There can be some custom requirements that you won’t be able to handle just with this reprocessing task. In those scenarios, you have to build a custom workflow. For example, there may be a need of escalation that requires deadline modeling. In addition, you may have to send separate emails to other user groups notifying them the IDoc error. You have to build a custom workflow in those scenarios. You can start the workflow against the same event triggering mechanism as mentioned here. Then you have to build the entire process in the workflow builder;

you'll need to create a custom business object and method for the additional logic required to process the workflow. You can use the standard business object method `INPUFOREGROUND` to process the work item.

9.1.3 Notification of Successful Posting

Similar to error processing of IDocs, you can also send notifications on successful posting of any IDoc. As there is no standard task available, you have to create a custom task to send a notification on successful posting of the IDoc. We'll take the same example of IDoc `DEBMAS`:

1. Get the process code to process the inbound IDoc as mentioned in Section 9.1.2. In our case, the process code is **DEBM**.
2. Link the successful posting event with the process code. Go to Transaction BD67 to configure the process code, as shown in Figure 9.10. Here, you need to ensure that the success event **INPUTSUCCESS** is maintained for the process code.

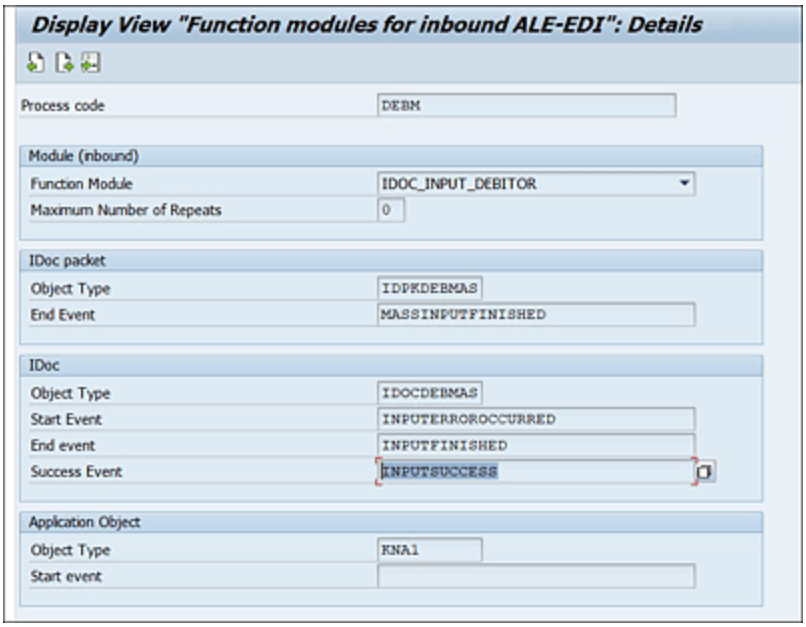


Figure 9.10 Success Event INPUTSUCCESS Configured for the Process Code

Business object `IDOCDEBMAS` is used to process this message type `DEBMAS` via process code `DEBM`. For any standard SAP process code, you'll get the business object type and event details in Transaction BD67. Business object `IDOCDEBMAS` triggers event `INPUTSUCCESS` after successful posting of the `DEBMAS` IDoc, so, this event is configured in this step. Figure 9.10 shows the mapping of the success event for the process code.

3. Create a custom workflow to send the notification. The design of the custom workflow will be per your business requirements. We won't go into the development details of the workflow here as these are covered in detail in Chapter 4. The main objective of this section is to demonstrate how to trigger a custom workflow when an IDoc is successfully in SAP. This custom workflow may be used to trigger a simple notification to a responsible user email address for the IDoc (determined based on

the responsible user in the partner profile or per your own business rules and picking up the email address from user master), or it may perform some post-processing activity on the document that is posted. In our example, we'll consider a simple workflow with a send mail step to notify the user of successful posting. For details on workflow and send mail step creation, refer to Chapter 4.

4. Once the workflow is created, open the workflow from Transaction PFTC, and navigate to the **Triggering events** tab. Here, add an entry for your IDoc business object type and success event per the details from process code settings in Transaction BD67 (refer to Figure 9.10). Figure 9.11 shows the details of the event linkage with workflow. Here, you enter the **Object Type** as "IDOCDEBMAS", enter the **Event** as "INPUTSUCCESS", and then maintain the **Binding** as prompted by the system with a suitable object type container element. Finally, activate the event linkage by clicking the button beside the event linkage entry (far left of **Standard events** table) so that it turns green. Capture the entry into a Customizing transport when prompted.

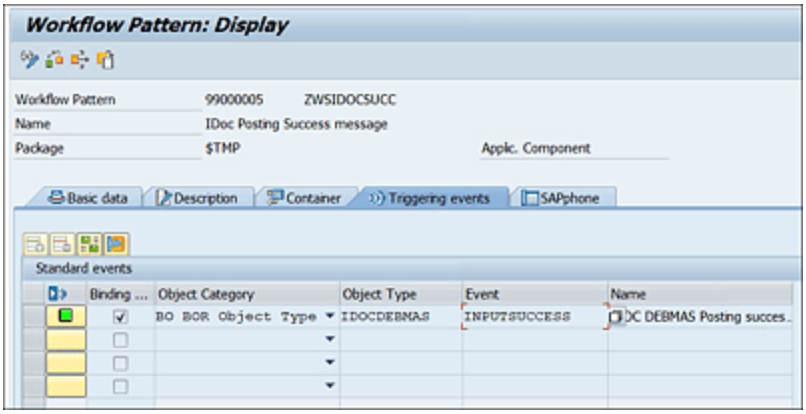


Figure 9.11 Event Linkage Entry with IDoc Object Type, Success Event, and Custom Workflow

5. The workflow definition may contain any step(s) per your requirement. In Figure 9.12, we've added a simple **Send Mail** step just to demonstrate the functionality.

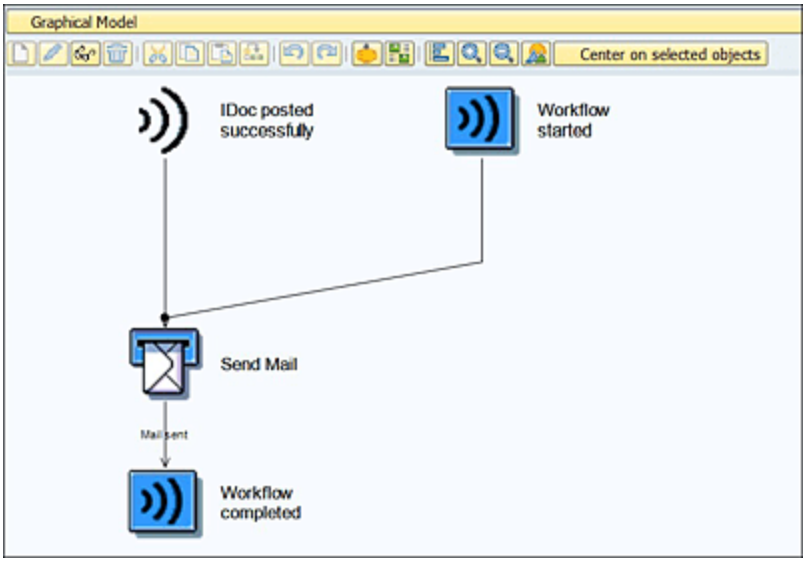


Figure 9.12 Workflow Definition for Successful Notification of IDoc Posting

The determination of the user (and their email address) who should receive the notification should be per your own business requirement.

9.1.4 Testing Procedure

You can unit test both the IDoc error-handling workflow and the successful posting workflow by creating an IDoc with IDoc test Transaction WE19. Create a test inbound IDoc with data for the scenario you're trying to test. For an error-handling case, enter some incorrect data in the IDoc; for a successful case, enter all the data correctly. The

corresponding workflow or notification will be triggered and will be available in the user's inbox

Follow these steps to test the workflow for successful posting:

1. Create a test `DEBMAS` (Customer Master) IDoc via Transaction WE19. In an actual business scenario, this IDoc will be triggered automatically from an external system or via a load program in the same system. However, for testing purposes, we'll use Transaction WE19. Here, you can create an IDoc from scratch or copy an existing IDoc that was successfully posted (status **53**). In this example, per Figure 9.13, we've copied the data from an existing `DEBMAS` IDoc in **53** status.

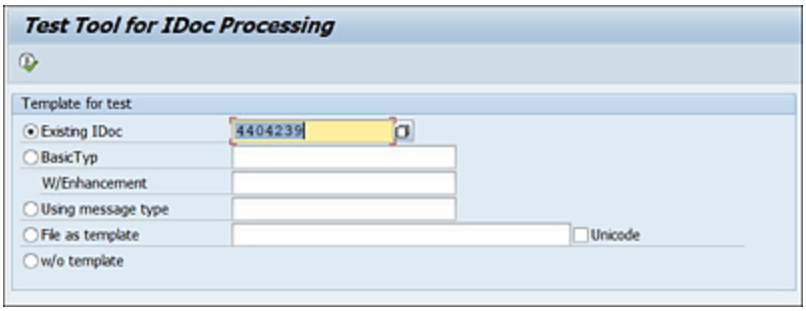


Figure 9.13 Creating an IDoc as a Copy from Transaction WE19

2. Once the IDoc data is copied, click on the **Standard Inbound** button, and then confirm the partner profile details in the subsequent popup if the status appears green. This will create a new inbound test IDoc and post it (if the partner profile is set with the **Post Immediately** setting). Figure 9.14 shows an example of IDoc posting via Transaction WE19. As soon as you click on the **Standard Inbound** button, a popup screen

comes up to confirm the partner profile based on the control data of the IDoc (see Figure 9.15). As soon as you confirm the popup, the IDoc is posted.

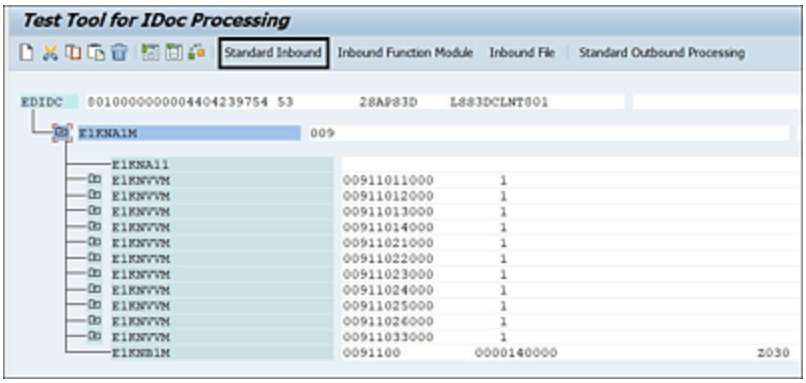


Figure 9.14 Posting an IDoc via Transaction WE19: IDoc Test Transaction

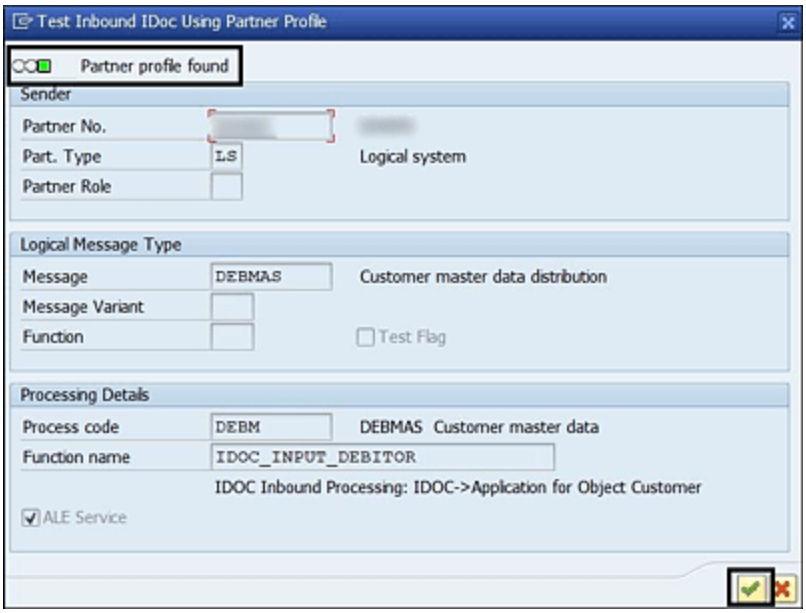


Figure 9.15 Partner Profile Confirmation Popup before IDoc Posting in Transaction WE19

3. Copy the IDoc number shown after posting the IDoc in the previous step, and enter the same in Transaction WE02 to check the status. If the IDoc status shows **53** (see Figure 9.16), then the IDoc posting was successful,

and your notification for successful posting should be created.

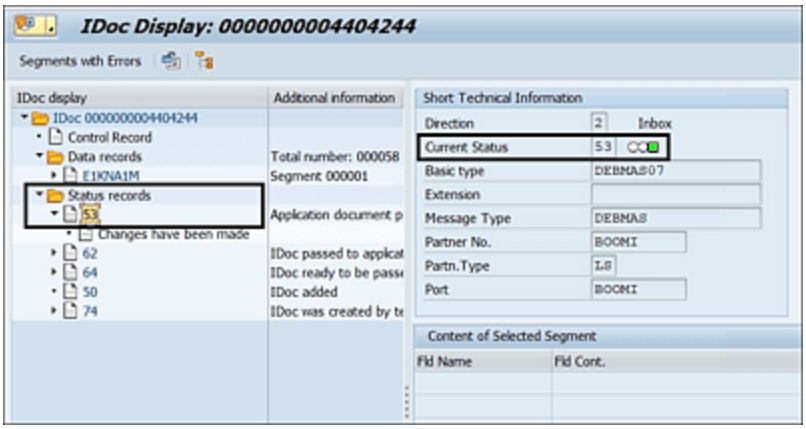


Figure 9.16 IDoc Status Check from Transaction WE02

4. Check the notification from an Outlook email inbox or Transaction SOST (if email dispatch outside SAP isn't allowed in your development system). The notification confirms the successful testing of your custom workflow. Figure 9.17 shows the details of the notification.

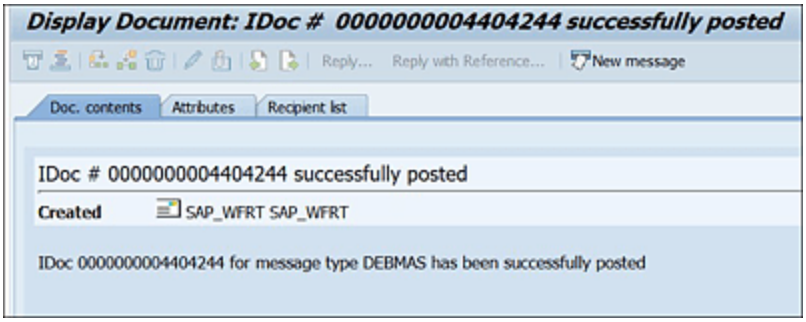


Figure 9.17 IDoc Posting Notification as Viewed from Transaction SOST

5. For IDoc error-handling workflows, the workflow is triggered when the IDoc posting fails and IDoc is set to **51** status. In this case, a work item is sent to the responsible user's SAP inbox, and the user must manually reprocess the IDoc after the root cause of the error has been fixed. Figure 9.18 shows a sample work

item for the error-handling workflow in SAP inbox (you can view the same from the My Inbox app as well).

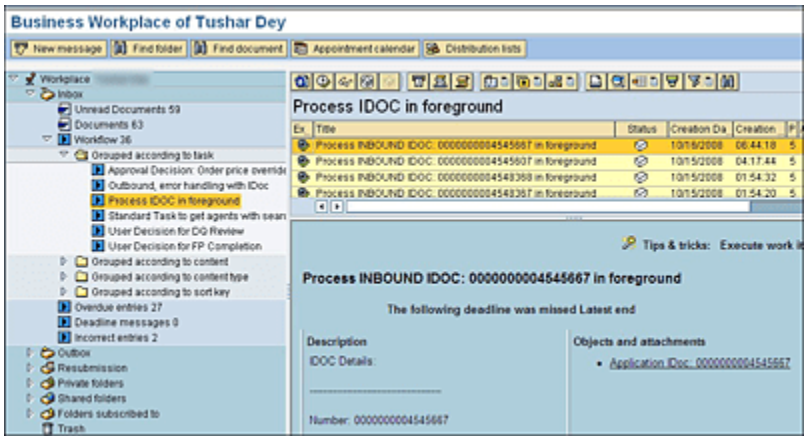


Figure 9.18 Work Item Created in the User Inbox to Process the IDoc in the Foreground

You can select the work item from the inbox and execute it by clicking the button. Once the work item is executed successfully, the work item will disappear from the user's inbox.

9.1.5 Setting Up an Inbound IDoc Process via a Workflow

Normally, IDocs are processed immediately without any manual intervention. Sometimes, however, requirements come up where the IDoc should be reviewed before it's processed. In this scenario, system setup can be done to start a workflow for an incoming IDoc. SAP doesn't provide any standard workflow for this kind of requirement. These workflows are custom-developed workflows.

The configuration steps to meet this requirement are as follows:

1. Create a workflow or standard task. This is described in detail in Chapter 4.
2. Create a new process code. Go to Transaction WE42, and click on the **New Entries** button to enter the details, as shown in Figure 9.19. Enter the **Process code**, **Description** of the process code, and **Identification** (workflow number you've created; in this case, "WS20000183"). Select the **Processing with ALE service** radio button and the **Processing by task** radio button. Now this process code will always trigger workflow `WS20000183` before processing the IDoc.
3. Assign the newly created process code in the partner profile configuration. Go to Transaction WE20. Select the inbound partner profile where you want to link the process code. Update the message type and process code with the IDoc message type and the process code you created in the earlier step.

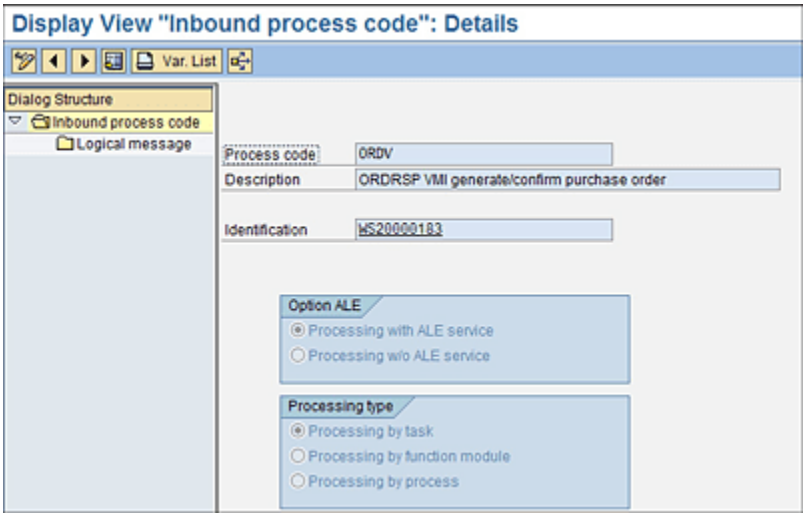


Figure 9.19 Settings to Process the IDoc with the Workflow

9.2 Active Monitoring

We've discussed in the earlier section how workflow can be set up for each IDoc failure. The responsible user gets one work item for each IDoc failure to reprocess these failed IDocs. For very critical business processes, there may be a need to send an alarm to the supervisor if there are too many IDoc failures. This can be achieved by setting up active monitoring.

What is active monitoring? Active monitoring allows you to set up a batch job that will trigger a work item when a threshold in the number of IDoc failures is reached. The threshold is the number of IDocs in a particular status within a time limit.

Let's discuss two business scenarios. You get IDocs from a third-party system in the morning that provide you the information regarding what should be ready for shipment for same-day delivery. There will be a penalty if the shipment isn't created on the same day before a stipulated time. In this scenario, you already have set up the IDoc monitoring workflow. The person responsible in the warehouse will get those IDocs for reprocessing, but there should be management visibility of the failures as this is business critical. You can set up active monitoring for this particular IDoc, which will send consolidated information via a workflow to the supervisor,

Other scenario occurs when, due to some system configuration/data issue, lots of IDocs start failing. The

individual owner of that IDoc is getting a work item to reprocess the failed IDoc. But the core issue isn't resolved in a day. Therefore, the responsible person for the business process should be notified of the high number of recurring IDoc failures. This can also be achieved by active monitoring.

Program RESIDOCA is scheduled in a batch job to set up active monitoring. You can execute the same program via Transaction WE06. Figure 9.20 shows the active monitoring program selection screen.

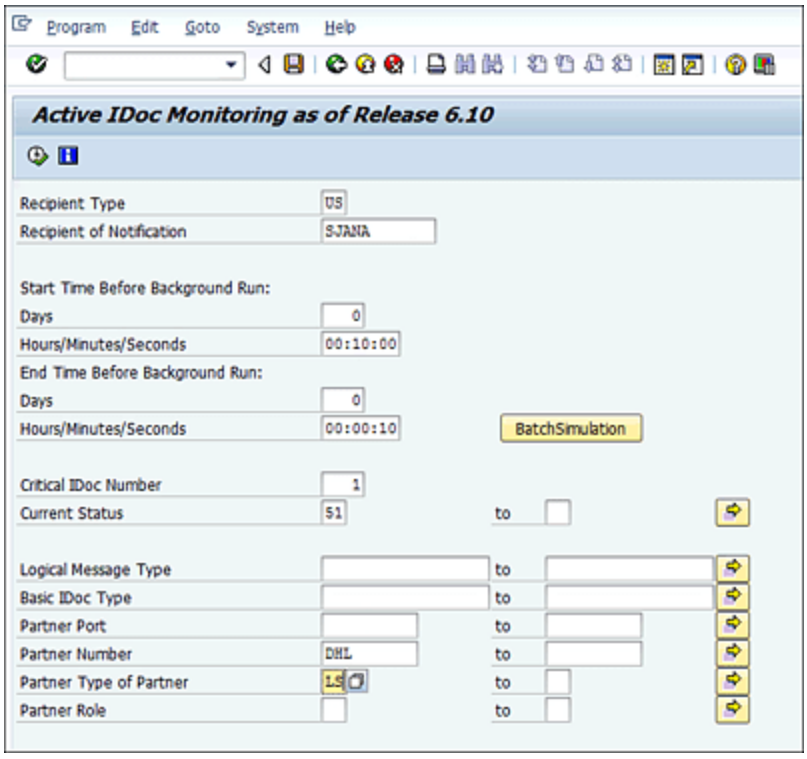


Figure 9.20 Selection Screen of the Active Monitoring Batch Job

In the recipient details, you can enter the user or any other recipient type (e.g., work center, person, position, etc.) who will receive the work item. The next block is the time duration when the program will look for erroneous IDocs. In the first business scenario mentioned, you'll know when

those IDocs come to your system. You can set up the back job and the time period for selection. In Figure 9.20, it will consider the IDoc for the last 10 hours. Other settings on this screen include the following:

- **Critical IDoc Number**
  Enter the threshold of the number of failures. In this case, the work item will be generated if the number of failures is more than one.
- **Current Status**
  Enter the IDoc error status that will be considered for selection, such as **51**, **56**, or **64**.
- **Logical Message Type**
  Enter the IDoc message type you want to set up the active monitoring.

For the other partner-related parameters that appear here, the same message type can be used for several business processes. You can enter other partner parameters for exact selection of the IDocs.

You have to schedule this batch job daily (or based on your requirement). Whenever the number of failed IDocs crosses the critical IDoc number entered in the selection screen, a work item for task `TS74508518` will be sent to the recipient.

When the user executes this work item from the inbox, it will show all the error IDoc numbers and the current status. The user can refresh the list to get the current status of the IDoc. It will help the supervisor to closely monitor the progress of error resolution for the business-critical IDocs.

9.3 Common Workflow Data Tables

Throughout different other sections, we've already seen multiple workflow-related tables. In Table 9.1, we'll highlight the different common workflow tables and their purposes.

Table Name	Details
SWWWIHEAD	Read Work Item Header: This table stores the header information of a work item, such as the work item ID, work item type, creation date, and so on.
SWW_WI2OBJ	Workflow Runtime: Relation of Work Item to Object: This table links work items to their corresponding objects, such as business objects, tasks, or processes.
SWWUSERWI	Current Work Items Assigned to a User
SWW_CONT	Workflow Runtime: Work Item Data Container
SWW_CONTOB	Workflow Runtime: Work Item Data Container (Only Objects)
SWWLOGHIST	Workflow Runtime: History of a Work Item
SWFDEVINST	Event Linkages with Instance Reference
SWFDEVTYP	Event Linkages without Instance Reference
AGR_USERS	Assignment of Roles to Users

Table Name	Details
HRS1205	Active Version of Workflows
SWF_CPWF_INST	Workflow Instance Information
SWN_NOTIF	Workflow Notifications
SWW_EVENTS	Events in Workflow

Table 9.1 Common Workflow Data Tables

Tips

The majority of workflow tables start with SWW. You can search the tables in ABAP Data Dictionary (DDIC) by using “SWW*” and then check the description. Some workflow and task definitions are stored in standard infotypes, and those tables can be searched with “HRS*”. In SAP S/4HANA, a few standard core data services (CDS) views are also given for workflows, and the corresponding DDIC SQL views can be searched using “SWW_V*”.

9.4 Common Workflow Application Programming Interfaces

SAP has provided many standard workflow APIs related to workflow triggering, reading/writing work item data, deadline processing, roles and rules handling, and more. The majority of these APIs are in the form of function modules. Table 9.2 lists the commonly used SAP workflow function modules.

Function Module Name	Details
`SAP_WAPI_START_WORKFLOW`	This function module is used to start a workflow. It requires the workflow definition ID and other necessary parameters to initiate the workflow. You can trigger the workflow using this function module without triggering any business event.
`SAP_WAPI_CREATE_EVENT`	This function module allows you to create an event in the SAP Business Workflow system. Events trigger the start or execution of workflows.

Function Module Name	Details
SAP_WAPI_READ_CONTAINER	This function module is used to read the container elements of a work item, which hold the data relevant to the workflow task.
SAP_WAPI_WRITE_CONTAINER	This function module writes the container of a work item.
SAP_WAPI_FORWARD_WORKITEM	This function module is used to forward a work item to another user or agent.
SAP_WAPI_EXECUTE_WORKITEM	This function module executes a work item in the foreground.
SAP_WAPI_CHANGE_WORKITEM_PRIO	This function module changes the priority of a work item.
SAP_WAPI_GET_DEADLINES	This function module reads the deadline information of the work item.
SAP_WAPI_WORKITEM_RECIPIENTS	This function module reads the recipient details of a work item.
SAP_WAPI_WORKITEM_DESCRIPTION	This function module reads the work item description.

Function Module Name	Details
SAP_WAPI_SET_WORKITEM_STATUS	This function module sets the work item status to another status.
SAP_WAPI_DECISION_COMPLETE	When a user decision step is encountered in the workflow, this function module is used to provide the decision or response of the user. It allows the workflow to proceed based on the user's decision.
RH_GET_ACTORS	This function module gets the agent details of a role.
RH_RESOLVE_RESPONSIBILITIES	This function module performs role resolution for standard roles with responsibilities.

Table 9.2 Common Workflow Function Modules

Tips

The majority of workflows start with SAP_WAPI, so you can search for other standard workflow function modules with "SAP_WAPI*" in Transaction SE37.

9.5 Workflow Reporting

We talked about administrative reporting in Chapter 8. These reports are useful for a workflow administrator for day-to-day support activity. However, there will be other requirements from a business perspective. To check the workflow process efficiency, key questions are asked:

- How many instances of the workflow are triggered on a monthly basis?
- What is the average time taken to complete a workflow?
- Which tasks are taking most of the time, and why?
- How many instances of the work item are escalated with deadlines triggered?
- Which work items are sitting with the approver for a long time?
- Is there a need for a centralized report combining the business data (e.g., purchase order number, vendor, amount, etc.) and approval status?

There will be lots more questions asked from the operational point of view, as well as to analyze the performance of the workflow and remove the bottlenecks. SAP provides some standard reports to analyze the data. All of these reports provide basic selection criteria and output. A workflow log is available in all the reports to find the details as required. Let’s discuss some of these useful reports:

- **Workload Analysis**
  You can run the report execution Transaction SWI5 or use

menu path **Tools** • **Business Workflow** • **Development** • **Reporting** • **Workload Analysis**. This report provides a view of work items already completed and in-progress work items. You can find how many work items are in a user's inbox and also the history of the workload. You can execute the report for the work items completed after a certain date. This can provide you an overview of workload during some peak business operation times such as year-end, holiday seasons, and so on. This report also helps workflow administrators or process owners to see which work items are pending for a long time. They can then assign those work items to someone else. Sometimes, people go on leave without a proper substitution. This report helps you identify and expedite the approval process. This report also shows the work items where there is no agent assigned. Ideally, the work item should handle this as an error if an agent isn't found. But depending on the possible agent and actual setup, there may be work items where an agent couldn't be determined. This will show you those work items as well. Figure 9.21 shows a sample output of workload analysis.

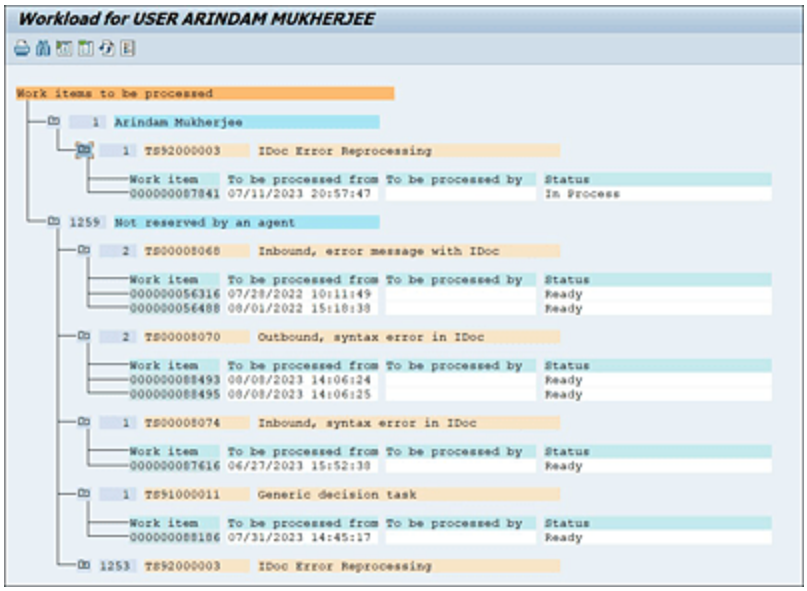


Figure 9.21 Workload Analysis (Transaction SWI5) Sample Report Output

- **Work Item with Monitored Deadline**
  You can execute Transaction SWI2_DEAD work items with monitored deadlines to analyze which work items missed the deadline. You can execute the report from menu path **Tools** • **Business Workflow** • **Development** • **Reporting** • **Workitem Analysis** • **Workitems with Monitored Deadlines**. You can group by task, task group, and so on. You can analyze if there is any trend of missed deadlines for any specific workflows. Then you can work with end users to understand the reason behind the missed deadlines. The purpose of deadlines is to force users to complete work items within a specific timeline. But if deadlines are missed, then it may take longer than the expected duration. All kinds of trend analysis can be done with data. Whether this is a user-specific issue, workflow-specific issue, or a particular time of the year, that’s the kind of viewpoint you can build based on this report output (see Figure 9.22).

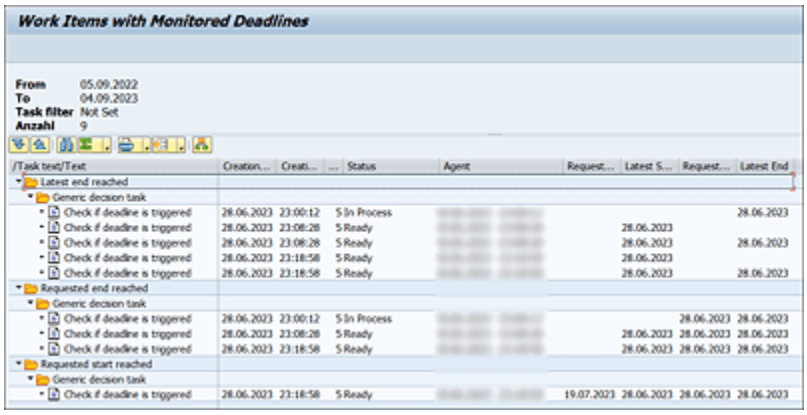
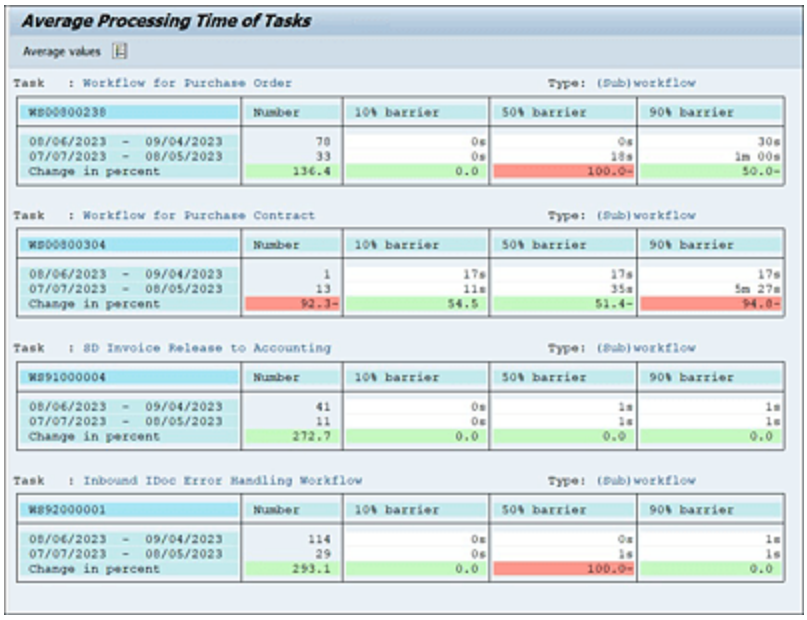


Figure 9.22 Monitored Deadline (Transaction SWI2_DEAD) Sample Report Output

- **Work Items by Processing Duration**
  You can execute Transaction SWI2_DURA to get the execution time of the work item. You can follow menu path **Tools** • **Business Workflow** • **Development** • **Reporting** • **Workitem Analysis** •**Workitems by Processing Duration** to execute this report. This report will show you the wait time and processing time. Wait time is calculated from task creation time until the execution is started. Processing time is calculated from execution started to execution completed. This data will be shown in both averages within the timeline and also details for each work item. It will also show you 10%, 50%, and 90% threshold barriers, that is, what is the maximum time taken for 10%, 50%, and 90% of the work item. In other words, with the duration reported, 10% of the work items were completed. This data point will help you analyze workflow performance. Then you can work with the end user in a design-thinking workshop to improve the performance. Figure 9.23 shows a sample output of the Work Items by Processing Duration report.



Average Processing Time of Tasks

Average values

Task : Workflow for Purchase Order — Type: (Sub)workflow

WS00800238	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	78	0s	0s	30s
07/07/2023 - 08/05/2023	33	0s	18s	1m 00s
Change in percent	136.4	0.0	100.0-	50.0-

Task : Workflow for Purchase Contract — Type: (Sub)workflow

WS00800304	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	1	17s	17s	17s
07/07/2023 - 08/05/2023	13	11s	35s	5m 27s
Change in percent	92.3-	54.5	51.4-	94.0-

Task : SD Invoice Release to Accounting — Type: (Sub)workflow

WS91000004	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	41	0s	1s	1s
07/07/2023 - 08/05/2023	11	0s	1s	1s
Change in percent	272.7	0.0	0.0	0.0

Task : Inbound IDoc Error Handling Workflow — Type: (Sub)workflow

WS92000001	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	114	0s	0s	1s
07/07/2023 - 08/05/2023	29	0s	1s	1s
Change in percent	293.1	0.0	100.0-	0.0

Figure 9.23 Work Items by Processing Duration (Transaction SWI2_DURA) Sample Report Output

There aren’t that many analytical reports available in SAP S/4HANA. The reports also don’t have enough selection fields and data output. So, you may end up writing your own custom report for workflow. There are two kinds of reports you may have to develop: operational report and analytical report. You can consider writing operational reports within SAP S/4HANA using the ABAP programming language. Consider your data warehouse (e.g., SAP Business Warehouse [SAP BW]) for writing analytical reports. There are lots of workflow-related extractors provided. Work with your SAP BW team to build the analytical report.

Normally, businesses will have requirements to have both business application data and workflow-related information in the same report, for example, develop a report to show purchase orders that are still within the approval chain. You may have to display purchase order details with the amount, vendor, and other critical information from the

business object perspective. You also have to show how long the approvals are pending, who is the current approver, and what level of approvals are already done in the same report. From the business operation perspective, high amount purchase orders need to be followed up first (this is just an example). You'll also see who should be following up in the same report. This kind of report requirement is very common. Refer to commonly used tables and APIs to build the report. Because these tables will handle high volumes of data, if you have to access workflow containers that aren't stored as simple fields of a database table, the program may face performance challenges. Always keep the performance aspect in mind when you're developing any custom reports on workflow.

9.6 Implementing Program Exits to Capture Data from Workflow Steps

Program exits in SAP workflow are exits or hooks within the various states of a work item processing where you can add your own custom logic. This custom logic may be required for capturing workflow runtime data and updating a custom table for reporting and analytics purposes. Sometimes, you may also want to capture some additional statistical data from workflow processing that may be added to the workflow log or an action log within a custom application. You can also use program exits to update some values on the workflow container or work item container before creating a work item or after execution of a work item.

Program exits are implemented via ABAP classes that implement any one of the following interfaces at the workflow header (version-dependent data) or workflow step level:

- **`IF_SWF_IFS_WORKITEM_EXIT`**
  This interface may be used at both the workflow header and work item step level. The class implementing this interface has method `IF_SWF_IFS_WORKITEM_EXIT~EVENT_RAISED`, which gets called when the corresponding event for the workflow or the task work item occurs. Check out sample class `CL_SWH_WORKITEM_EXIT` for reference.
- **`IF_SWF_IFS_WF_CONSTRUCTOR`**
  This interface may be used at the workflow header level only. Here, method `WF_CONSTRUCTOR` is executed before the

start of the workflow. Check out sample class `CL_SWF_IFS_WF_CONSTRUCTOR` for reference.

- **`IF_SWF_IFS_WF_DESTRUCTOR`**
  This interface may be used at the workflow header level only. Here, method `WF_DESTRUCTOR` is executed after the end of the workflow. Check out sample class `CL_SWF_IFS_WF_DESTRUCTOR` for reference.

Figure 9.24 and Figure 9.25 illustrate the maintenance of program exit classes at the workflow header and at the workflow step level. Once you enter a valid program exit class (an ABAP class implementing any of the three interfaces mentioned previously), then at runtime, SAP will trigger the `EVENT_RAISED`, `WF_CONSTRUCTOR`, or `WF_DESTRUCTOR` methods, depending on each change of status of the work item processing. Within these methods you'll be able to capture data from the workflow step and log into your own custom tables or an application log per your requirement.

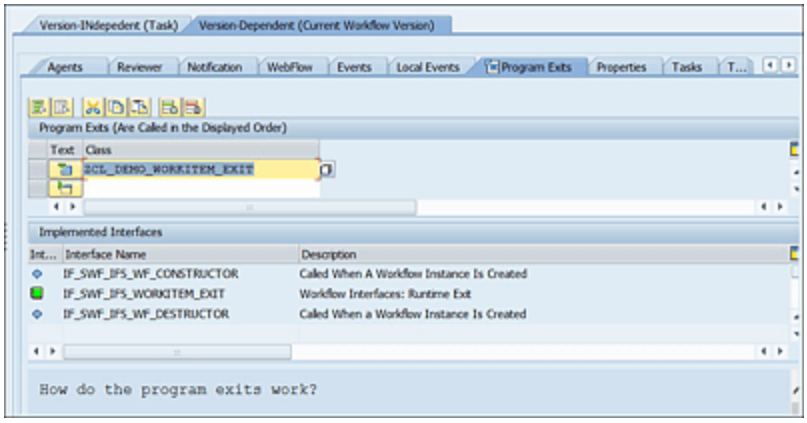


Figure 9.24 Maintenance of a Program Exit at the Workflow Header Level

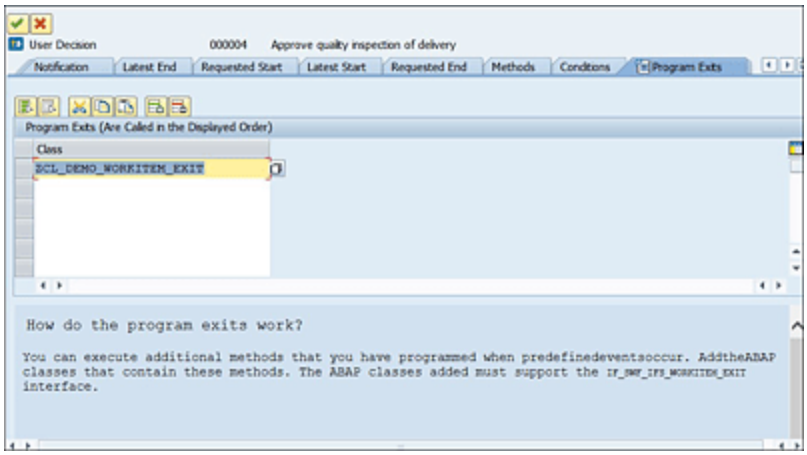


Figure 9.25 Maintenance of a Program Exit at the Workflow Step Level

Let's take a closer look at classes implementing the `IF_SWF_IFS_WORKITEM_EXIT` interface and how you can use method `EVENT_RAISED` of this interface to achieve your custom requirements. Standard class `CL_SWH_WORKITEM_EXIT` provides a sample implementation of this interface. Method `IF_SWF_IFS_WORKITEM_EXIT~EVENT_RAISED` has two import parameters:

- **`IM_EVENT_NAME`**
  This parameter supplies the event name to the program exit. The event name identifies the stage of processing of a work item at runtime. Table 9.3 provides the possible events that may be triggered during the processing of a work item. The constant values of the event names are maintained in type group `SWRCO`, which in turn points to type group `SWFCO`.

Event Name	Exit Triggering Time
`BEF_CREAT`	Before creating a work item
`CREATED`	After creating a work item

Event Name	Exit Triggering Time
BEF_EXEC	Before execution of a work item by a user
AFT_EXEC	After execution of a work item by a user
AFT_ASYINV	After asynchronous processing of a work item
BEF_REMOVE	Before removal of the work item from a user’s inbox
STATE_CHG	Any status change of a work item
AFT_REXEC	After execution of a rule for a dialog work item
BEF_DECI	Before processing the decision option for a user decision work item
BEF_ACTION	Before processing the action for a work item
AFT_ACTION	After processing the action for a work item

Table 9.3 List of Event Names along with Their Processing Time for the Program Exit Method

Checking the event name before executing your custom logic provides you with the capability to restrict the code to the relevant processing time of the work item.

- **IM_WORKITEM_CONTEXT**
  This parameter refers to an object of type IF_WAPI_WORKITEM_CONTEXT at runtime, which provides the entire context of data for the current work item as well as the main workflow. Table 9.4 lists some of the commonly used methods of this interface along with their purpose.

Method Name	Purpose
GET_HEADER	Reads the header work item data for the workflow.
GET_PROPERTY	Reads the property value based on the property name assigned on the **Properties** tab of an activity step.
GET_RULE_RESULT	Delivers the result of the current rule resolution, that is, the agents of the current dialog work item.
GET_STATE_TRANSITION	Whenever the processing status of a work item changes, this method returns the source and target status.
GET_TASK_ID	Returns the task definition ID of the current work item.
GET_WF_CONTAINER	Returns the workflow container details for the current workflow. Any element of this container may be further read using the common Business Object Repository (BOR) programming macros (refer to Chapter 3, Section 3.1.8). You can update values in the container as well.

Method Name	Purpose
GET_WI_CONTAINER	Returns the work item (task) container details for the current work item. Any element of this container may be further read using the common BOR programming macros (refer to Chapter 3, Section 3.1.8). You can update values in the container as well.
GET_TOP_CONTAINER	Returns the workflow container details for the top-level workflow. If the current task isn't called from a subworkflow, then the result of this method is the same as GET_WF_CONTAINER. You can update values in the container as well.
GET_WORKFLOW_ID	Returns the workflow work item ID of the current work item.
GET_WORKFLOW_TASK_ID	Returns the workflow definition ID of the current work item.
GET_WORKITEM_ID	Returns the current work item ID.
SET_MESSAGE_TO_LOG	Writes a message to the current workflow work item log.
GET_DEADLINES	Gets the deadline details for the current work item, including requested start, requested end, latest start, and latest end.

Method Name	Purpose
GET_DECISION_ALTS	Returns the decision alternatives for a user decision work item.
SET_DECISION_ALTS	Adds or modifies the decision alternatives for a user decision work item.

Table 9.4 List of Commonly Used Methods along with Their Purpose in Interface IF_WAPI_WORKITEM_CONTEXT

Additionally, exception CX_SWF_IFS_WORKITEM_EXIT_ERROR may be raised in case of any errors during processing, and a custom error message may be added to the exception as well.

These methods enable you to implement your own logic in the program exit class, which may be maintained at the workflow header level or at a step level. Note that you can maintain multiple program exit classes at the same level; in that case, all the classes will be executed sequentially when the predefined events are raised by the workflow runtime.

Program exits are called at various stages of processing of a work item. Some of these processing events are called in the background by the workflow batch user ID (SAP_WFRT), and others are called in dialog mode with the user ID of the agent executing the work item. Depending on your workflow design and the event, the program exits may be debugged with an external breakpoint using the workflow batch user ID or the agent's user ID.

You can use program exit classes for any custom requirement that fits the scope of the methods mentioned in Table 9.4. However, most commonly, these classes are used

to update data from the workflow and current work item into a custom table that may be readily consumed by reports used for statistical analysis of the business process implementing the workflow. Other common application scenarios for program exits include workflow and task container data manipulation and manipulation of runtime processing options (e.g., decision alternatives).

9.7 Summary

In this chapter, you've seen how classical workflows can automate ALE and IDoc processes. The chapter explains how workflow may be used in inbound processing, error notification, and outbound scenarios. You also learned about common workflow data tables and workflow APIs (primarily function modules) that can be used in different workflow-related custom applications and also how to search these table and function modules. The chapter also explains how to implement program exits in workflow steps to capture more detailed information during workflow execution, which can be useful for reporting.

10 BRFplus

This chapter teaches you about using Business Rules Framework plus (BRFplus) in a workflow scenario. You'll be introduced to general BRFplus concepts to ease your understanding, and then the integration with SAP Business Workflow will be discussed at length. Additionally, BRFplus as a guiding system can also be deployed across many processes and designs, so you can use it in many other contexts where Business Rules play a part in the overall decision management.

This chapter discusses the integration requirements of BRFplus and SAP Business Workflow to see how we can get the best of both worlds. But to better our understanding, it's best to take a closer look at the concept of business rules, then move on to understanding the management systems that are needed as a tool, and finally look at the BRFplus tool that SAP provides to build business rule applications. Let's start by discussing what business rules are.

10.1 Introduction to BRFplus

Business rules are part of every organization in almost every process execution, whether in a promotional offer from a retail store with "Buy 1, Get 1" signs or an airline offering "50% off a business class air ticket on 35,000 accrued miles." Business

rules thus speak about the operations and execution diversions in a process, as well as define the behavior of the critical systems running the enterprise.

Are these rules hard-coded in the software to bring a different contextual behavior of the system? Well, that's still the case in some organizations for a few processes, but it's certainly too costly and too slow to do so. The business might even miss a chance to apply such rules and benefit from it.

If you want to manage the deployment of the rules in a seamless manner that is both easy and fast while cutting out the entire software development lifecycle, then it's time to get introduced to *business rules management systems (BRMSs)*. In the context of SAP, we'll be looking at BRFplus. In the past, we used custom tables, constant table `TVARVC`, and custom solutions to maintain the rules, which served the purpose of maintaining the rules in some capacity. But, as the complexity of rules grow, it becomes extremely difficult to make architectural design easy to maintain and deploy. SAP came up with BRFplus as the BRMS tool of choice.

BRFplus contains functionality for rule modeling, rule storage, deployment, and monitoring. Let's first discuss BRMSs in general and then look at each of these components in turn.

10.1.1 Business Rules Management Systems

In the world where it's important to be faster in bringing the products to the market, it's equally necessary to be faster to manage any unforeseen scenarios in the business operations. Organizations today want to be ahead of the race with their competitors, thereby necessitating faster reaction time to the changing needs of the business. This, in turn, is all about

steering the mission-critical processes that make the business. The control to steer comes with the ability to manage the business rules, those governing the process, on the fly.

BRMSs help you to do so. These are the systems that are used to create, deploy, maintain, and monitor business rules. Once the rules are maintained in the order of execution, you can invoke these rules via the BRMS capability to steer the processes. The invocation of these rules is coded *only once* by the developer, and the rest of the operational behavior is controlled by changing the rules on the fly, per the business needs. Each BRMS will have environments for rule *modeling*, rule *maintenance*, and a rule *deployment* and *execution* engine.

Both the technical and nontechnical user groups are the target audiences to use the various functions of the BRMS defined in the previous statement. Business users as part of nontechnical category of the audience are expected to make changes to the rules on the fly to immediately affect the operational processes.

10.1.2 Rule Modeling

The rule modeling component is the development studio for the BRFplus developers. The BRFplus studio is also called the *BRFplus workbench*, and it's a Web Dynpro application, which can be launched using Transaction BRF+ or Transaction BRFPLUS.

Any rule modeling component needs to provide users with a set of tools to design and develop the applications. The tools should be placed to provide a better developer experience too, so the layout of the development studio is also critical. Tools

such as application building blocks, simulation engine, execution engine, version management, and lifecycle management should be integral parts of the studio.

We'll look at two very important aspects of the BRFplus rule modeling component: the layout of the workbench and the building blocks for the developers to build the whole application. The rule execution engine is separately discussed in the next section.

Layout

Figure 10.1 shows the user interface (UI) workbench for creating, updating, and deleting the business rules. This UI component can be accessed via Transaction BRF+ or Transaction BRFPLUS.

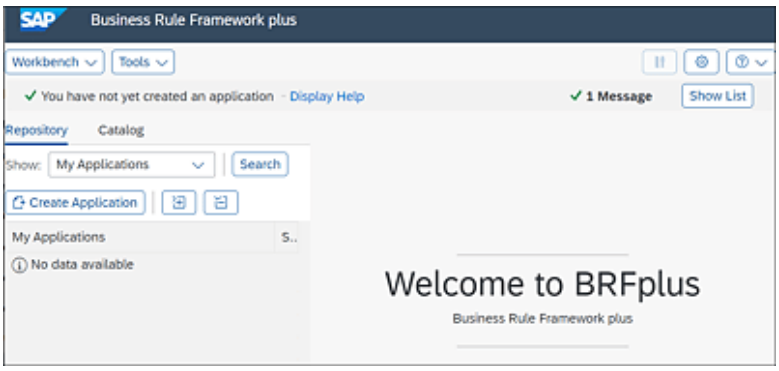


Figure 10.1 BRFplus Landing Page

Let's take a quick look at the layout of the landing page, starting at the top left:

- **Workbench**
  This button gives you the flexibility to open an application, resize the landing page to suit your needs, or add the objects as favorites.

- **Tools**
  The options under this button are primarily driven by the personalization button explained next. If the **Simple** user mode is chosen in the personalization, the user will have access to only the simulation tool. If **Expert** mode is chosen, the user will have access to the entire set of tools such as tracing, XML export and import, mass change, simulation, application administration, and so on.
- **Personalization**
  At the far-right side of the landing page, this option gives you the flexibility to define how the landing page should look, what default values should it load with, and a range of other controls for different smaller components of the overall business rules build.
- **Help**
  This button is what you use to build integration with the Help Center for giving specific documentation of your business rules application.
- **Repository**
  This view shows all the different components of the build such as data objects, expressions, rulesets, rules, functions, actions, applications, and so on.
- **Catalog**
  This view is primarily for the business users to maintain the rules in the decision tables. This is to make sure that business users can chose to concentrate more on managing the rules rather than getting exposed to the complexity of business rule development artifacts.

Building Blocks

A typical BRFplus application looks like the one in Figure 10.2.

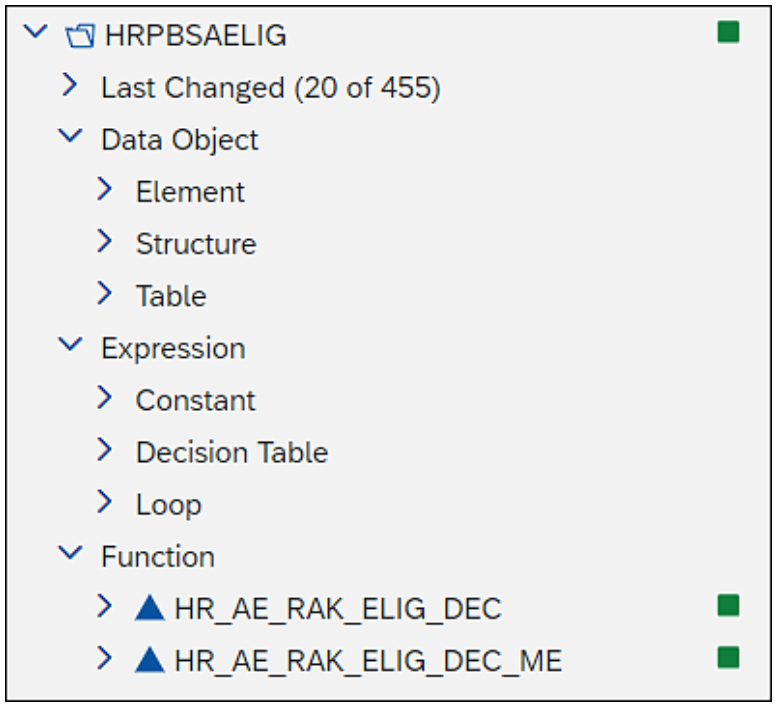


Figure 10.2 Typical BRFplus Application

Now, take a look at the different building blocks for the business rules in Table 10.1.

Components	Explanation
Application	This is the topmost component in the hierarchy (`HRPBSAELIG`, in our example) of the different smaller components and hence is the container for all the objects. It's the representation of the functionality that you want to enhance using BRFplus rules.

Components	Explanation
Data objects	This is the most granular component in the entire build of the business rules. Their definition is primarily driven by what data type they are associated with. A data object can be the following: • An element (single attribute) • A structure (a list of other data objects) • A table (a list of structures) Data objects become part of the context and result for a decision table, which is explained later in this table.
Expressions	This is the computational engine to achieve a certain business logic. There are many expression types such as case expression, Boolean, constant, decision tree, formula, function call, procedure call, and so on. The developer uses these expressions per the requirements to develop complex manipulation logic and rules. Indeed, there must be input(s) given to the expression for it to compute and produce a result per the assigned result data object.

Components	Explanation
Decision table expression	A decision table expression is where most of the BRFplus applications are designed to hold the business rules. The decision table contains a set of rows and columns, and their interaction enables each singular cell to maintain the business rule data. The initial columns form part of the *context* to the rule, whereas the right most columns are the *result* columns. Based on the requirement, there can be one or more input columns and one or more result columns. The rule engine evaluates the decision table expression from left to right and from top to bottom to determine the matching row(s). Based on the matching pattern configured, the rule engine evaluates all the rows or stops at the first match found.
Function	This is the container where all the business logic resides. This is the second layer after the application that wraps all the other objects in the business rule application development exercise. A function is the connecting link between the calling application and the rules processing framework. A function has different execution modes. The developer needs to choose correctly based on the use case to be implemented.

Table 10.1 Building Blocks of a BRFplus Application

Additional Reading

Now that you have basic information on BRFplus and its various components, we'd recommend that you take a look at *http://support.sap.com*. It's one of the best places to find information, considering the enhanced UI and search capabilities within it. You'll need an S-User or a P-User to login to the support portal. Once logged in, just search with the keyword "BRFplus" using the search bar at the middle top of the page. The search results are displayed on the **Services & Support** page where there is a wealth of knowledge articles, SAP Notes, Guided Answers, SAP Help links, and so on. The search results page looks like Figure 10.3.

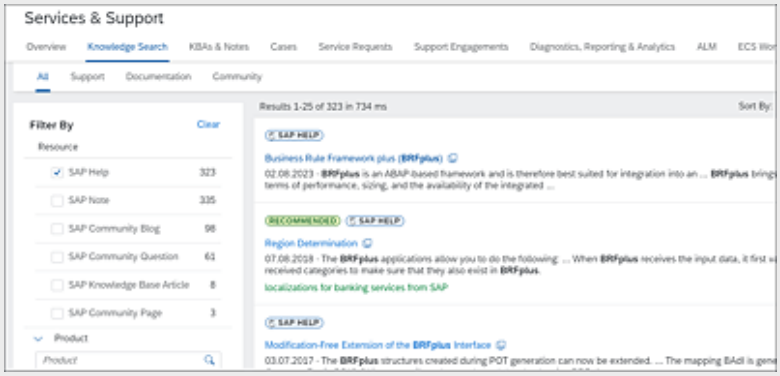


Figure 10.3 SAP Support Portal: Search Results Page

Alternatively, the you can also look for the SAP PRESS book *www.sap-press.com/brfplus-business-rule-management-for-abap-applications_2106/*, which dives deep into every aspect of the BRFplus framework.

10.1.3 Rule Execution Engine

The rule execution engine is one of the main components of any BRMS. While a developer builds a BRFplus application using the BRFplus workbench, the invocation of the rules during the business process execution are done using dedicated application programming interfaces (APIs).

The API class for on-premises BRFplus applications is `CL_FDT_FUNCTION_PROCESS`. If the business rules are built on SAP Business Technology Platform (SAP BTP) via SAP Build Process Automation, and if they are deployed to the on-premise SAP S/4HANA system, then API `CL_FDT_BRS_PROCESSOR` can be used to invoke business rules.

Class `CL_FDT_FUNCTION_PROCESS` has a method called `PROCESS` that is the actual API for invoking. Let's look at the API interface design, which is detailed in Table 10.2.

Parameter Name	Parameter Type	Description
`IV_FUNCTION_ID`	Importing	This is the GUID of the function building block. It can be found in the BRFplus workbench under the **General** tab of the artifact.
`IV_TIMESTAMP`	Importing	You specify a timestamp for the function to be processed so that the simulation of the system behavior is as it would have been at that particular point in time.

Parameter Name	Parameter Type	Description
IV_TRACE_MODE	Importing	By passing TRUE, you can record the function execution in a trace object for later reference. The trace object is referred by the EO_TRACE parameter (exporting parameter).
ITS_ID_VALUE	Importing	This is a table of ID/value pairs (ID/reference to value for each context element). From a performance perspective, this is the most efficient way of triggering BRFplus processing if you call BRFplus only a couple of times per session.
IO_CONTEXT	Importing	This is the reference to a context object. You get an object instance by calling IF_FDT_FUNCTION~GET_PROCESS_CONTEXT for a function instance. The values can be set via method IF_FDT_CONTEXT~SET_VALUE. Note that passing context values to a function with the help of this parameter leads to a decrease in performance compared to the alternatives described earlier.

Parameter Name	Parameter Type	Description
EA_RESULT	Exporting	This is a variable of a type that is compatible with the result data object of the BRFplus function. You can get an ABAP data object of this type (a reference) by calling method CL_FDT_FUNCTION_PROCESS =>GET_DATA_OBJECT_REFERENCE.
EO_TRACE	Exporting	This is the reference to the trace object used for recording trace information, depending on the setting of parameter IV_TRACE_MODE.
CT_NAME_VALUE	Changing	This is a table of name/value pairs (name/reference to value for each context element). For best performance, first transfer the values into the internal format of BRFplus. This can be accomplished by calling CL_FDT_FUNCTION_PROCESS =>GET_DATA_OBJECT_REFERENCE.

Table 10.2 Method Documentation in the Class

10.2 Integrating BRFplus Applications in SAP Business Workflow

Now that you know the ways to build BRFplus applications, let’s move on to discuss its integration with SAP Business Workflow. Here, we’ll take a simple demo scenario of a BRFplus application where the bank details of a customer are identified based on the company code and customer number as input criteria. This application will be invoked in a workflow for further processing.

10.2.1 BRFplus Application Overview

If you’re interested in using this demo scenario for hands-on practice, you need to develop the BRFplus application as depicted in Figure 10.4. The decision table will have company code and business partner as the context variables and the bank account ID as the result variable.

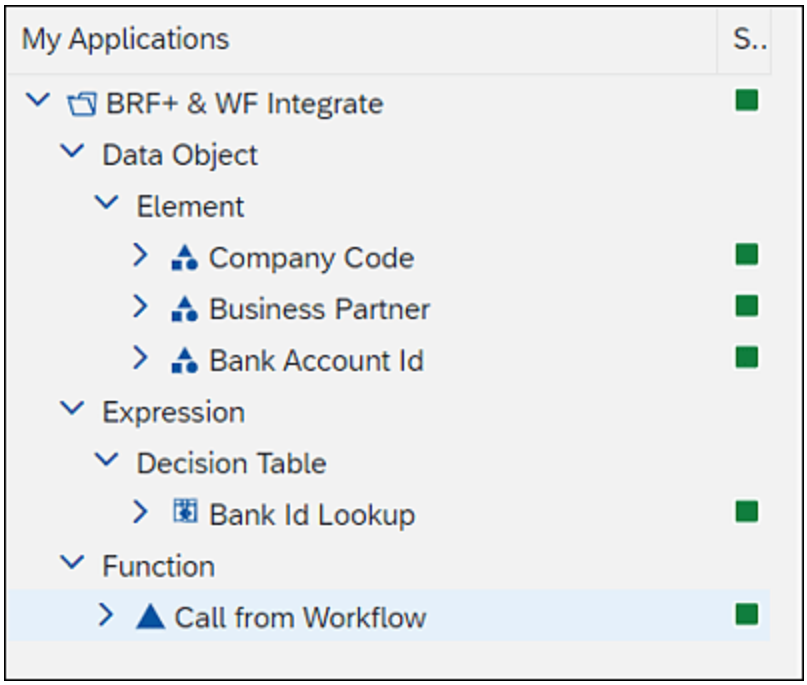


Figure 10.4 BRFplus Demo Scenario Application

10.2.2 Attach a BRFplus Function in a Business Workflow

In the business workflow, a BRFplus function is attached using the **Activate** step. Before you start to attach the BRFplus function, you'll need its ID. The ID can be found in the BRFplus workbench under the **General** section of the function artifact, as shown in Figure 10.5.

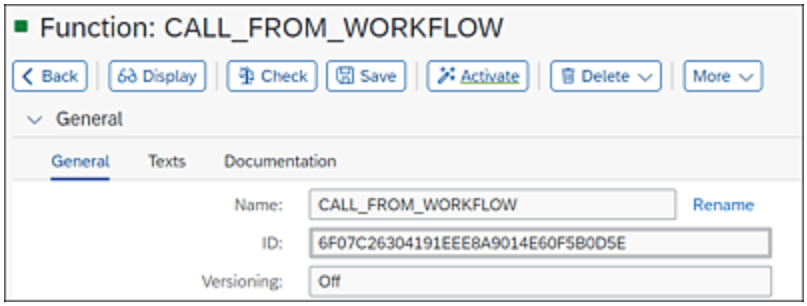


Figure 10.5 BRFplus Function ID

Once you have the function ID, you can start to create a business workflow. For that, go to Transaction SWDD. The initial screen looks like Figure 10.6.

If you've opened the transaction earlier, then the previously opened workflow loads by default. If so, then click on the **Create New Workflow** button at the top left of the screen. Alternatively, use Ctrl + Shift + F5 to create the new workflow instance.

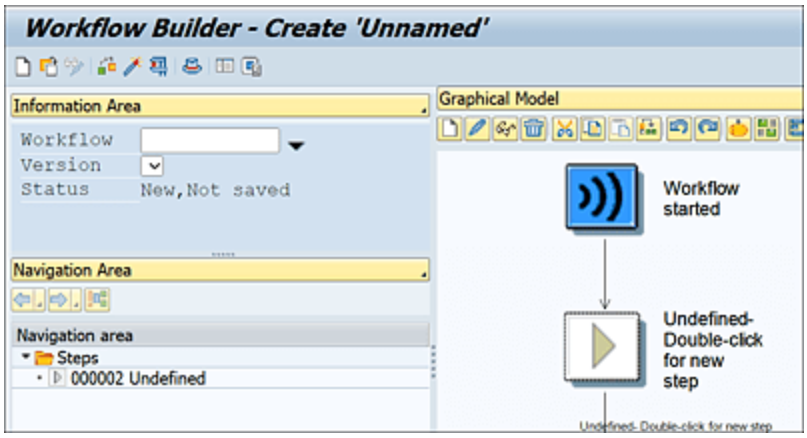


Figure 10.6 Transaction SWDD View

On the middle section of the screen, you see the workflow's initial graphical model. Double-click on the **Undefined** step, and you'll be prompted with a popup to choose the type of workflow task that needs to be created (see Figure 10.7).

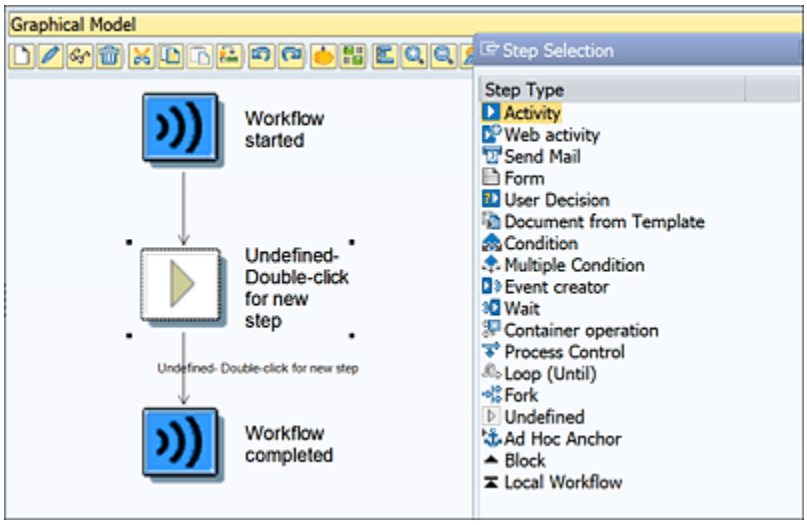


Figure 10.7 Create New Task for the Activity Type

Choose the **Activity** option, and the setup screen opens, as shown in Figure 10.8.

Click on the **Display Task** button (next to the **Task** field), and choose **Include BRFplus Function** from the dropdown list (see Figure 10.9).

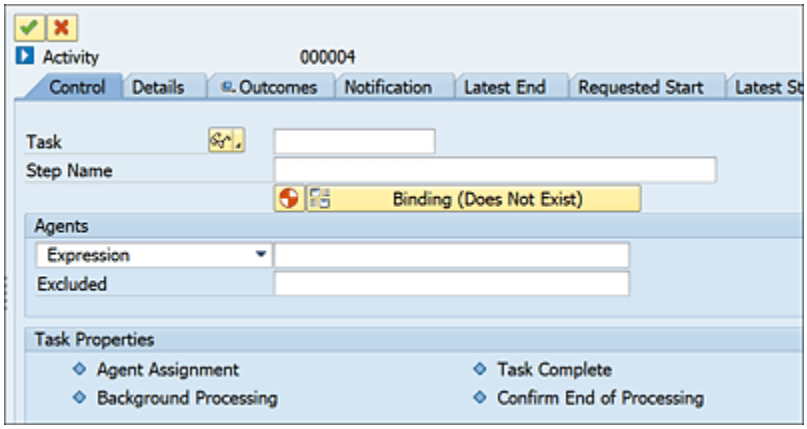


Figure 10.8 Activity Step

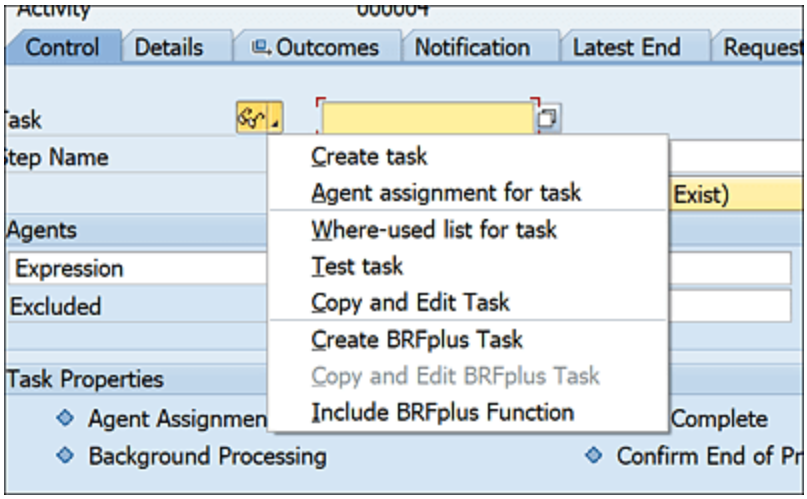


Figure 10.9 Include BRFplus Function

The popup screen shown in Figure 10.10 appears. Enter the BRFplus **Function ID** that you retrieved earlier in Figure 10.5.

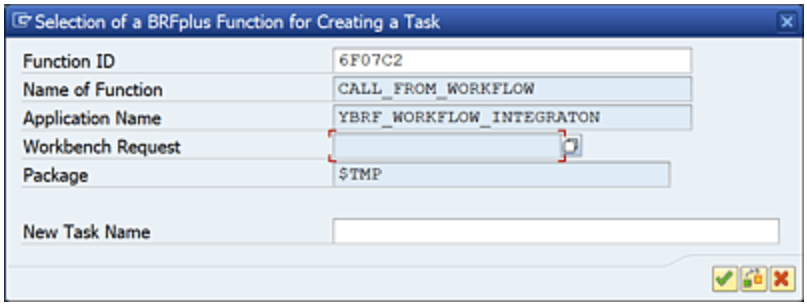


Figure 10.10 Include BRFplus Function ID

In addition, enter the **New Task Name**, and the framework automatically decides on the container elements to be created, both on the workflow and task container side. This is based on the signature of the function defined in the BRFplus workbench. Figure 10.11 depicts the popup that appears proposing the mapping between the container elements.

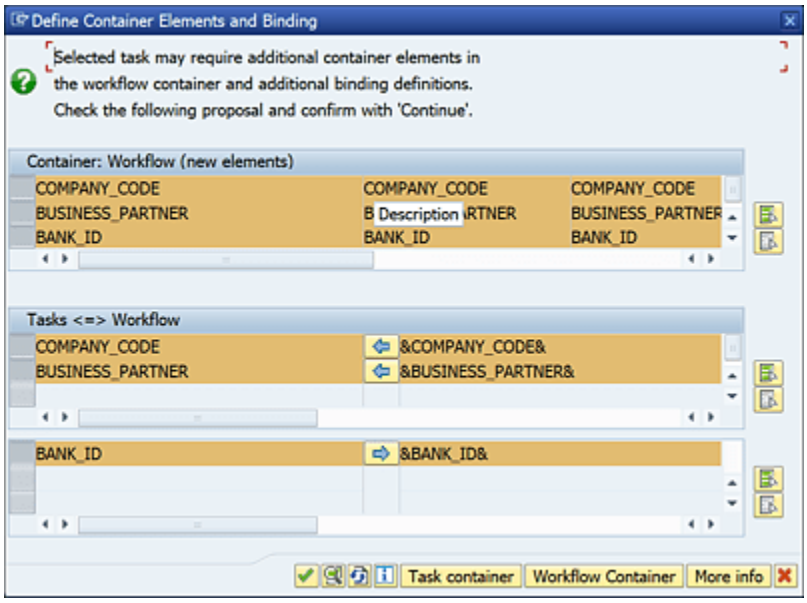


Figure 10.11 Mapping Proposal between BRFplus Function and Workflow Container Elements

When you accept the proposal of the mapping, you'll see that the task ID is created, and the description is also set as the GUID of the function. The agent assignment is also not required as the task is converted to a background task, as shown in Figure 10.12. The **BRFplus** button on the task view gives navigation to the BRFplus function in the BRFplus workbench.

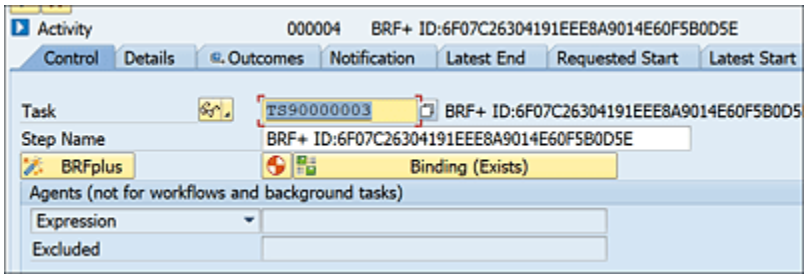


Figure 10.12 Task View after Attaching BRFplus Function

Click on the green checkmark button on the screen to take you back to the graphical model view of the workflow with the activity step added on it. Click on **Save**. You'll be asked

to provide a name to identify the workflow in a small popup, as shown in Figure 10.13.

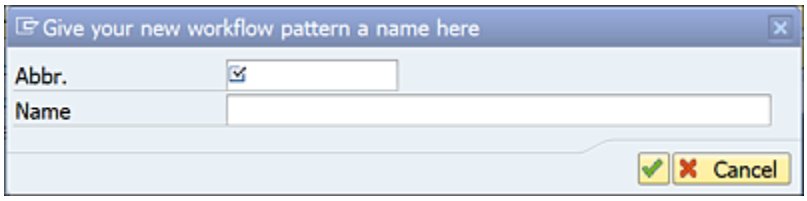


Figure 10.13 Name Your Workflow

Enter an abbreviation in **Abbr.**, enter the descriptive **Name**, and choose **OK** on the popup screen. Then, click on **Activate**. On activation, a workflow ID (WS*) will be assigned per the configuration done in the system in Transaction SWU3.

10.2.3 Executing the Workflow

To demonstrate the invocation of the BRFplus function, you'll run the workflow and provide values to the company code and business partner workflow container elements. To start with, let's look at the **Workflow Builder** view after the steps to attach a BRFplus function have been completed (see Figure 10.14).

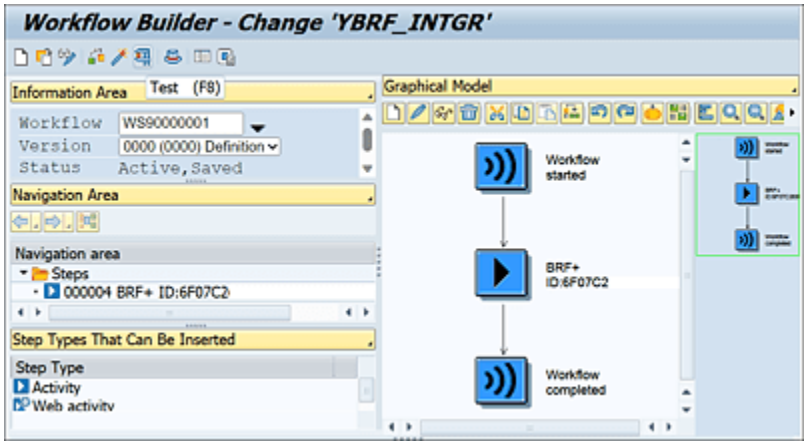


Figure 10.14 Workflow Builder View after Activation

In the left bottom section, where we see the **Step Types That Can Be Inserted** view, you need to change the view to see the workflow container by first clicking on the dropdown button and then selecting **Workflow Container**, as shown in Figure 10.15.

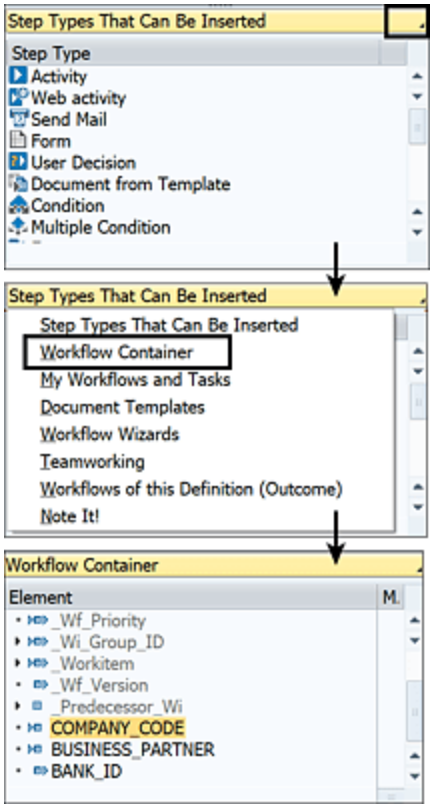


Figure 10.15 Steps to Get the Workflow Container View

You'll see the **COMPANY_CODE**, **BUSINESS_PARTNER**, and **BANK_ID** workflow container elements. Note that these container elements were auto-proposed while you were attaching the BRFplus function ID to the activity step (refer to Figure 10.11).

You now need to change the container element's properties so that `BUSINESS_PARTNER` and `COMPANY_CODE` are enabled as input

parameters and `BANK_ID` is enabled as the output parameter.

Follow this path to change attributes for each of the container elements: Double-click each of the container elements to open the popup screen, and navigate to the **Properties** tab. Select the relevant checkbox (**Import**/**Export**) in the **Parameter Settings** section for the elements (see Figure 10.16). Click **Confirm** on the popup screen or press Enter.

Because you've changed the workflow, don't forget to activate it one more time.

It's time to test the workflow. Press F8 to bring up the **Test Workflow** screen with all the relevant workflow container elements available to feed in the data. Two container elements, **COMPANY_CODE** and **BUSINESS_PARTNER**, are also available, as shown in Figure 10.17.

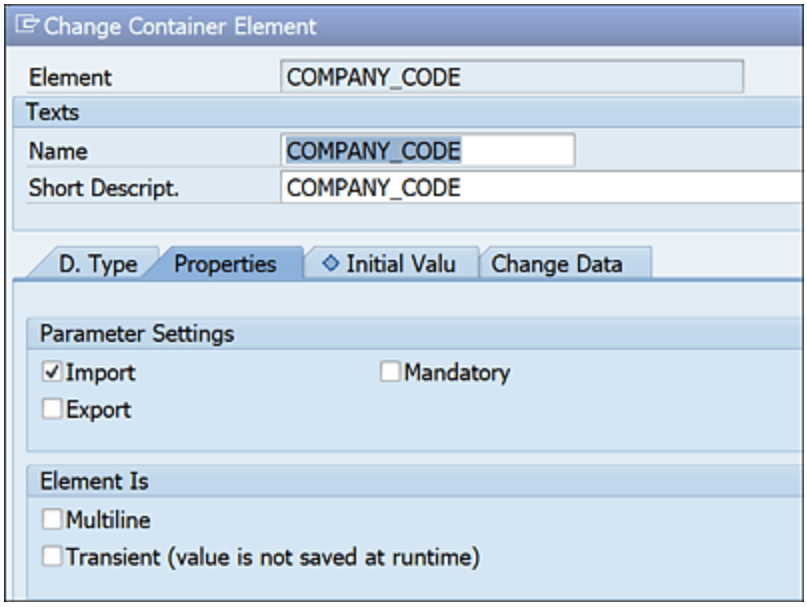


Figure 10.16 Change Container Element View

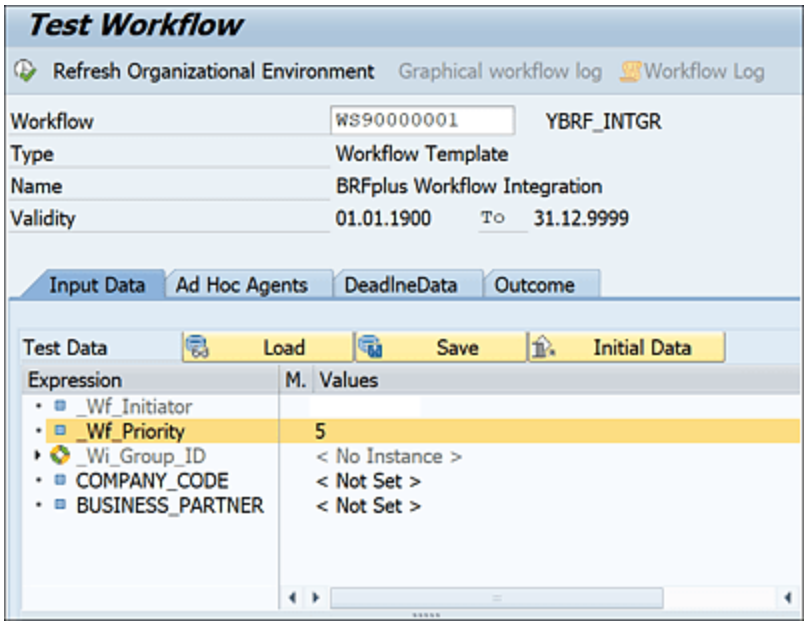


Figure 10.17 Test Workflow View

To do a positive test, put the values in the container elements that match with the decision table entries you've maintained in the BRFplus application developed earlier in Figure 10.4. For quick reference, take a look at Figure 10.18.

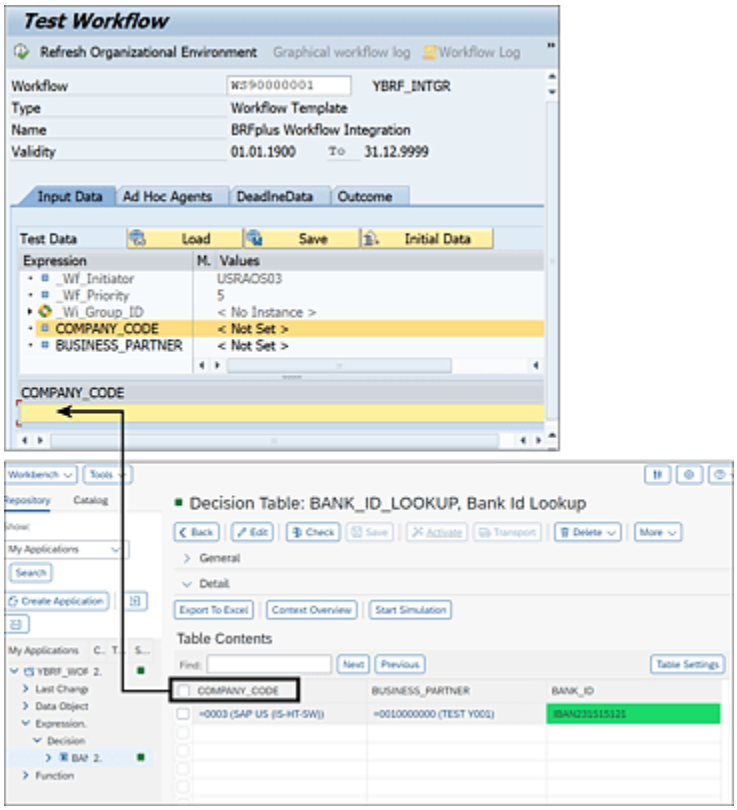


Figure 10.18 Mapping Values from the BRFplus Decision Table

After entering the matching values for the input container elements **COMPANY_CODE** and **BUSINESS_PARTNER**, press F8 to execute.

Following are the steps depicted in Figure 10.19 to test the workflow and see the execution result:

1. Press F8 on the **Test Workflow** view.
2. Once you see the Task ID generated in the message view of the screen, click on the **Workflow Log** toolbar button.

❸ You'll be taken to the **Workflow Log** view of the workflow engine. Because you need to look at the workflow container element **BANK_ID**, switch to the **Technical Details** view of the workflow by pressing Shift + F9.

❹ **BANK_ID** is shown in the container element at the task level as well as at the workflow level. The value can be verified with the value that is maintained in the BRFplus decision table.

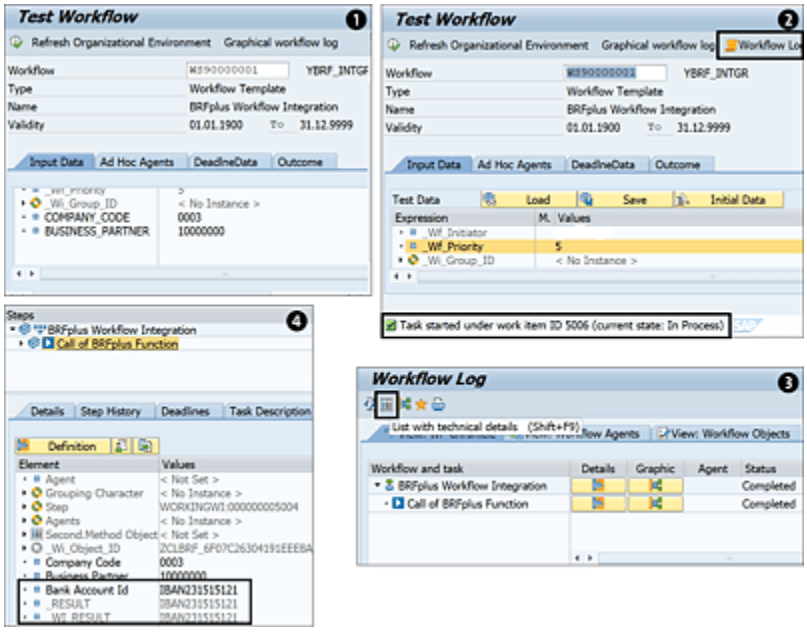


Figure 10.19 Sequence of Steps to Execute and Validate the Result

10.3 Summary

In this chapter, we discussed the need for a rule engine for enterprises and how they can be used in various business process operations. We looked at BRFplus from SAP as the tool of choice for business rule authoring, maintenance, and deployment by diving deep into understanding multiple building blocks to create a business rule application.

We used our understanding of the business rules framework to develop a sample BRFplus application and then showed how it can be integrated with a business workflow. This chapter also helps you understand how you can decouple the business rules from the main processing logic using rule engines and help bring faster deployment of the changes for ever-changing business needs.

11 Integrating Workflows with User Interface Applications and External Applications

This chapter gives you an overview of integrating workflows with user interface (UI) applications such as My Inbox. In real time, approvers access workflow work items in these UI applications and can take the appropriate actions. We'll explain how to navigate some detail pages and other apps from the My Inbox tile, as well as various My Inbox tile variants, including how to configure those tiles in detail.

We've used different types of UI worklist applications in SAP over time to check and take action on workflow work items—and these applications have improved a lot. For example, in SAP ERP systems, all classical workflows are tightly integrated with Business Workplace (Transaction SBWP). All work items are available in Transaction SBWP, and approvers open the work items and can act on them.

When SAP released SAP Enterprise Portal, work items were available in frontend applications such as the universal worklist. Agents no longer need to go to Transaction SBWP

to process work items; they can directly work on work items in the universal worklist. Now, we have the SAP Fiori app My Inbox, in addition to other options.

This chapter provides an overview of the My Inbox app, variants of the My Inbox app, and how to set up the scenario-specific My Inbox tile, which is a very common scenario in real time. We'll also explain the steps to configure navigational links to navigate to other external Web Dynpro applications and SAP Fiori apps from My Inbox.

11.1 My Inbox App Overview

SAP provides the modern SAP Fiori and SAPUI5 application called My Inbox (see Figure 11.1), that is, the workflow inbox, which is more lightweight, mobile, and compatible than earlier worklist applications in the intelligent enterprise system. By using this, you can process your standard or custom workflow tasks based on the decision options defined anytime, anywhere. You can use the My Inbox app with SAP Business Suite, SAP Business Suite powered by SAP HANA, or SAP S/4HANA 1511 onward. However, it has many additional features starting with SAP S/4HANA 1610. You can connect My Inbox with SAP ERP, older SAP Supplier Relationship Management (SAP SRM), SAP Customer Relationship Management (SAP CRM), SAP Business Process Management (SAP BPM), SAP S/4HANA, and so on, so you can access all different system-specific tasks in a single worklist.

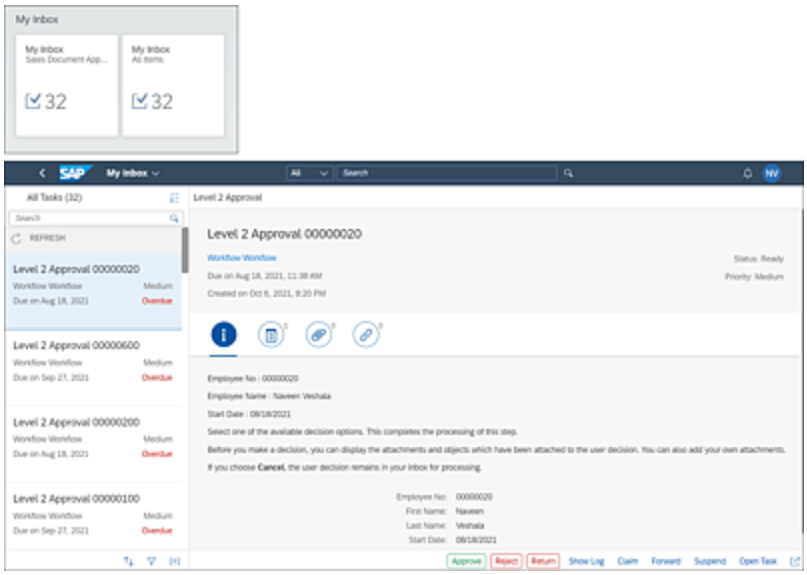


Figure 11.1 My Inbox App

My Inbox Variants

There are various **My Inbox** tile variants provided by standard SAP, and you have to enable those tiles per your business requirements:

- **All Items Inbox**
  The preconfigured **All Items** tile in the SAP Fiori launchpad enables you to easily process all your tasks. You must add the **All Items** tile from the catalog to view this tile in SAP Fiori launchpad.
- **Scenario specific inbox**
  My Inbox also offers to customize your own workflow scenarios and create scenario-specific tiles in the SAP Fiori launchpad.
- **Outbox**
  You can configure the **Outbox** tile, which will you enable to view all completed and suspended tasks.

My Inbox has many features, some of which are listed here:

- You can process tasks from SAP Business Workflow, SAP BPM, older SAP SRM servers, and third-party providers.
- You can view and add comments.
- You can view workflow and task logs.
- You can define decision options you want to display (**Approve**, **Reject**, **Return**, etc.).
- You can view, add, or delete attachments.
- You can navigate to relevant apps.
- You can create and manage substitution rules to manage the tasks in your absence.
- You can filter the tasks based on substation users.
- You can browse, sort, filter, and group tasks requiring action.
- You can add additional attributes that provide additional information about the task.

11.2 Workflow Task Integration with User Interface Applications

As explained in Section 11.1, the My Inbox – All Items app is a preconfigured app available in the system. You can get relevant roles per the SAP Fiori apps reference library assigned to your ID and use the tile in the SAP Fiori launchpad. Apart from this, you have to take certain steps to enable the scenario-specific **My Inbox** tile. Follow these prerequisites to use the **My Inbox** tiles:

- SAP Business Workflow should be active in the backend.
- SAP Fiori launchpad should be configured and available.
- All relevant My Inbox task gateway OData services and Internet Connection Framework (ICF) nodes should be active.

The following sections provide you with all the details you need to configure the scenario-specific **My Inbox** tile in the backend system and set up the SAP Fiori tile details. We'll explain launching an SAPUI5 application from a workflow task in the My Inbox app as well.

11.2.1 Set Up a Scenario-Specific My Inbox Tile

Your organization may have specific requirements that necessitate the grouping of work items related to a specific process area/business function. In this case, using the scenario-specific **My Inbox** tiles in SAP Fiori is an option to group related work items instead of using the **My Inbox – All Items** tile.

For example, let's say that the procurement department wants to separate the **My Inbox** tile, which will display only procurement-related workflow tasks. You can follow these steps to create a scenario ID and then create the tile in the SAP Fiori launchpad designer to provide to the user:

1. Create a scenario ID and consumer type via menu path **ABAP Platform** • **SAP Gateway Service Enablement** • **Content** • **Task Gateway** • **Task Gateway Service** • **Scenario Definition**.
2. As shown in Figure 11.2, click on the **New Entries** button, and create a new scenario ID using the details mentioned in Table 11.1.

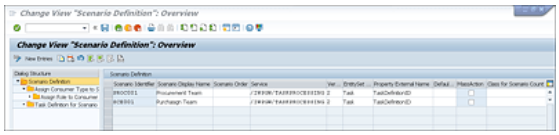


Figure 11.2 Scenario Definition

Fields	Description

Fields	Description
Scenario Identifier	Enter a new scenario ID.
Scenario Display Name	Enter a display name for the scenario.
Scenario Order	Not relevant for My Inbox
Service	Enter “/IWPGW/TASKPROCESSING”.
Version	Enter “2”.
EntitySet External Name	Enter “Task”.
Property External Name	Enter “TaskDefinitionID”.
Default Sort by Property	Enter the property you want to sort the work items in the list screen of the app. If you don’t update any value, the default value is **CreatedOn**.
MassAction	Select this checkbox if you want to, for example, approve or reject several tasks of the same type at the same time.
Class for Scenario Count	Leave this empty. Note that if the scenario isn’t part of the SAP Business Workflow engine or SAP BPM, and you want to see the number of tasks pending for this scenario, enter the name of the class that was implemented by the `/IWWRK/IF_TGW_SCENARIO` interface in the backend system.
Quick Act.	Select the **Quick Act.** checkbox to enable quick approval of workflow items by swiping on the screen. This field is only applicable when you have a touch screen device.

Table 11.1 Scenario ID Details

3. Save these details, and create a new scenario ID.
4. Next, assign the **Consumer Type** by selecting a newly created scenario ID and then clicking on **Assign Consumer Type to Scenario ID** in the **Dialog Structure** on the left, as shown in Figure 11.3 ❶.
5. You can choose consumer types of **DESKTOP**, **MOBILE**, and **TABLET** ❷ per your application and business requirements.

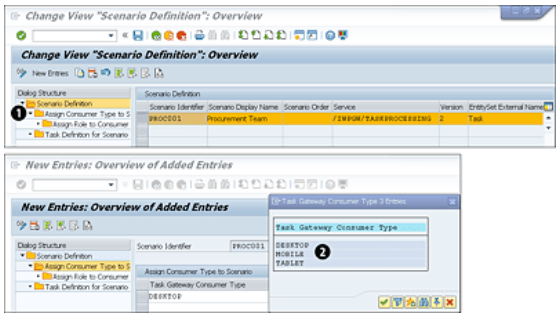


Figure 11.3 My Inbox Scenario ID Configuration

6. Optionally, you can assign consumer type and scenario to Transaction PFCG roles. That is, this scenario will be available to users who have at least one of the relevant roles assigned. You can follow these steps to assign a role:
   - Select the consumer type row, and click on **Assign Role to Consumer Type and Scenario**.
   - Choose **New Entries**.
   - In the **Role** column, assign relevant roles per your business process.
   - Save your entries.
7. Next, per your business requirement, you can assign single or multiple tasks for the same scenario. For multiple tasks, make sure you follow the same steps for each task.
8. Choose **Scenario Definition** in the **Dialog Structure**, and select the row containing your approval scenario.
9. As shown in Figure 11.4, then choose **Task Definition for Scenario** in the **Dialog Structure**.
10. Choose **New Entries**.
11. Enter the correct **SAP Alias System** for the task ID.
12. Enter the **Task Type** ID for your approval workflow.
13. Save your entries.

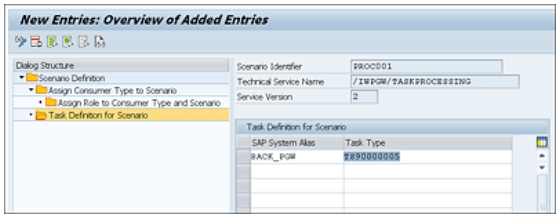


Figure 11.4 Assigning Task IDs to a Scenario ID

This is the required configuration to be done in the backend system for the scenario-based My Inbox app. Once this is done, you have to log in to the SAP Fiori designer and configure a scenario-specific dynamic tile in the relevant business catalog and assign that tile to the user.

Per your SAP Fiori launchpad configuration, you can follow either the traditional approach or implement the spaces/pages concept to create this dynamic tile. Whichever you choose, the tile creation process is more or less the same. Follow these steps to configure this tile:

1. Log in to SAP Fiori, and open the relevant catalog, which is shown in Figure 11.5.
2. Click on the **Add Tile** icon.
3. Create the tile by selecting the **App Launcher – Dynamic** option, which is shown in Figure 11.6.

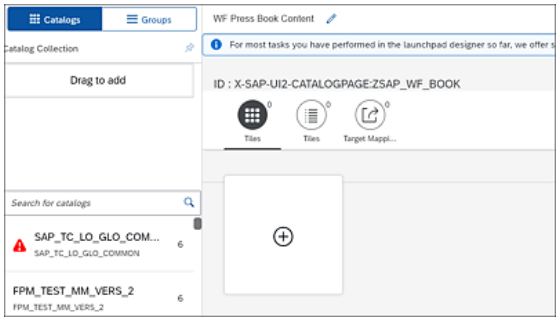


Figure 11.5 Creating the SAP Fiori Scenario Tile

Figure 11.6 Scenario Dynamic Tile

4. Enter all dynamic tile details into the screen shown in Figure 11.7, as described in Table 11.2.

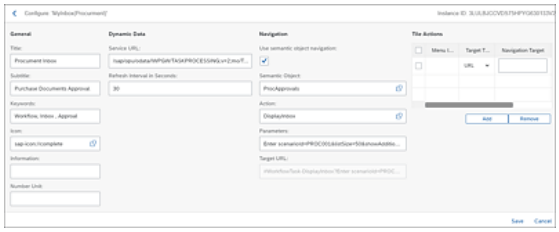


Figure 11.7 Configuring a Scenario-Specific Tile

Field	Description
Title	Enter a meaningful title for your scenario-specific app.
Subtitle	Enter a meaningful subtitle for your scenario-specific app.
Icon	Use any of the available icons.

Field	Description
Service URL	Enter “/sap/opu/odata/IWPGW/TASKPROCESSING;mo;v=2/ScenarioCollection?$filter=key eq '<your scenario identifier>'.”
Refresh Interval in Seconds	Enter the interval in seconds; this refers to the number of seconds after which data is refreshed: **0** indicates that there should be no auto-refresh.
Semantic Object	Enter the relevant name of the semantic object or workflow task.
Action	Enter “DisplayInbox”.
Parameters	Enter “scenarioId=<ID of your scenario>&listSize=<number of items you want to display on your list screen>>&showAdditionalAttributes=<true or false>.” A few notes on the details of this: • ScenarioID is mandatory. • listSize is optional, and the default is 100. • Use showAdditionalAttributes=true if you want to display custom attributes in the list view.

Table 11.2 Tile Details

5. Create a target mapping for this tile that is similar to the **My Inbox – All Items** tile shown in Figure 11.8, and then complete the tile configuration.

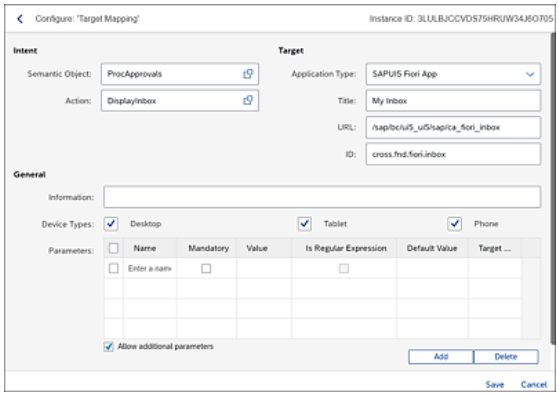


Figure 11.8 Target Mapping

Table 11.3 gives all the details about target mapping.

Field	Details
Semantic Object	Enter the semantic object that was mentioned in the tile details.
Action	Enter the action mentioned in the tile details.

Field	Details
Application Type	Choose **SAPUI5 Fiori App**.
Title	Enter a meaningful title, that is, “My Inbox”.
URL	Enter Transaction SICF node details, for example, “/sap/bc/ui5_ui5/sap/ca_fiori_inbox”.
ID	Enter the My Inbox SAPUI5 application component ID, that is, “cross.fnd.fiori.inbox”.

Table 11.3 Target Mapping Details

11.2.2 Create and Maintain User Attributes: Adding Additional Attributes for a Task

In general, the standard My Inbox app displays important details in its **Information** pane. However, SAP enables you to add even more additional details in the information pane if required, as shown in Figure 11.9. This is useful if you need to add more details for a task, as we’ll discuss in this section.

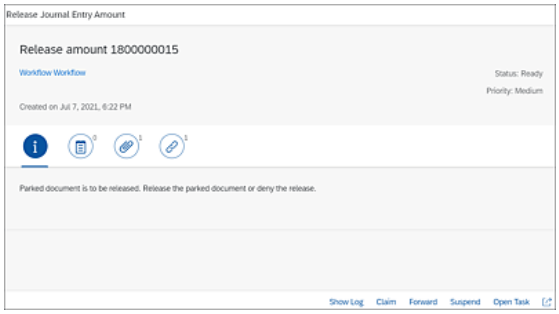


Figure 11.9 Additional Details

Perform the following steps to add additional details:

1. Go to Transaction SWF_USER_ATTR in the backend system.
2. Enter **Use Case Filter** “WF_INBOX_DC”, and select and execute the **Customizing** checkbox, as shown in Figure 11.10.

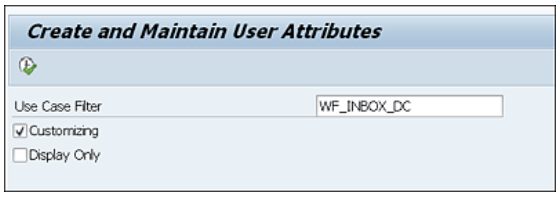


Figure 11.10 Transaction SWF_USER_ATTR

3. Click on the **Create New Entry** button and maintain the user attributes for the required task or workflow per Table 11.4. Once maintained, you can see those details in the main screen, as shown in Figure 11.11.

Field	Description
Task/Workflow	Enter the required task ID or workflow ID
Step ID	This has to be entered if you've provided a workflow ID.
Use Case	Select **WF_INBOC_DC**.
Description	Enter the label to be displayed in My Inbox.
Expression	Choose the value to display. Use the context-sensitive help to display the field. You can select any of your selected task context variables.

Table 11.4 User Attributes

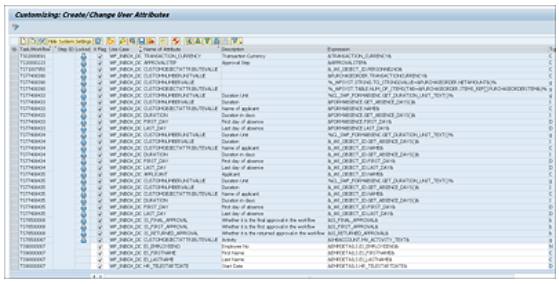


Figure 11.11 User Attributes Task Details

As shown in Figure 11.12, we've maintained the last four attributes for task TS90000007, which you can see displayed in the My Inbox app.

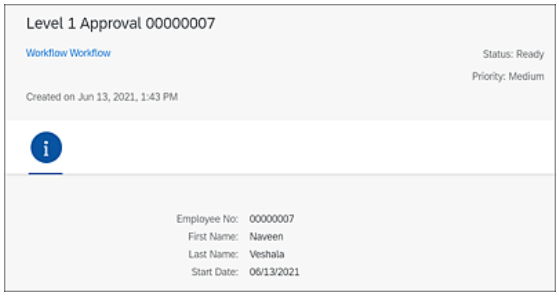


Figure 11.12 My Inbox: Additional Details

4. You can reindex the runtime data for any existing work items by clicking on the **Generate** button after maintaining these user attributes (see Figure 11.13). This is very useful for old work items that are lying in production environments.

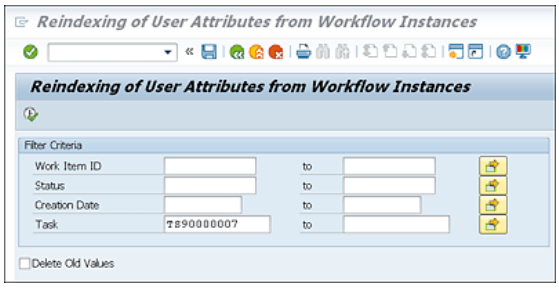


Figure 11.13 Reindexing User Attributes for Old Work Items

5. After reindexing the runtime data for this sample task, you have a newly added user attribute in the My Inbox app shown in Figure 11.14.

Figure 11.14 Updated User Attributes

11.2.3 Launching SAPUI5 Applications from Workflow Tasks

There are many scenarios in which you want to navigate to another application from My Inbox app. For example, you might be checking the summary of purchase requisition details in the My Inbox app and want to open the Purchase Requisition app to review all the details. This is a common scenario in many approval flows. SAP has provided different ways to integrate external applications with My Inbox app.

Task and object visualization parameters can be maintained in the following transactions:

- Transaction SWFVMD1 (client-independent configuration)
- Transaction SWFVISU (client-dependent configuration)

Generally, you can define two types of use cases by using these transaction codes:

- Universal worklist
- My Inbox app

Transaction SWFVISU doesn’t support both use cases together, so you need to decide which use case should be supported. If both use cases have to be supported, then Transaction SWFVMD1 must be used. The system uses the configuration of Transaction SWFVMD1 if available and uses the configuration of Transaction SWFVISU as a fallback.

The task gateway reads task visualization parameters in the following order:

1. Transaction SWFVMD1 configuration is accessed first.
2. If the Transaction SWFVMD1 configuration doesn’t exist, then Transaction SWFVISU parameters are used.
3. If both Transaction SWFVMD1 and Transaction SWFVISU configurations are missing, then standard (web GUI) visualization parameters are used.

You can follow these steps to launch SAPUI5 applications from the My Inbox app:

1. Go to Transaction SWFVISU, as shown in Figure 11.15.
2. Select **Task Visualization** in the **Dialog Structure**, and click on **New Entries** to maintain task details. You can find all task-relevant details in Table 11.5.

Task Visualization Parameter	Details
Task	Enter the required task ID.
Visualization Type	A visualization type has a list of parameters that can or must be set. There is an option to use placeholders, which allows you to provide task instance data or symbols. Based on your system/requirement, choose the correct visualization type from dropdown box.

Table 11.5 Task Visualization

3. Save the task details.
4. Choose a task ID in the **Task Visualization** table, and click on **Visualization Parameter** in the **Dialog Structure**, as shown in Figure 11.16.
5. You can see all relevant parameters per the visualization type chosen in the task details. You'll find the most common types such as **Intent-Based Navigation** and **My Inbox Generic Application**, which you can use in the My Inbox configuration.

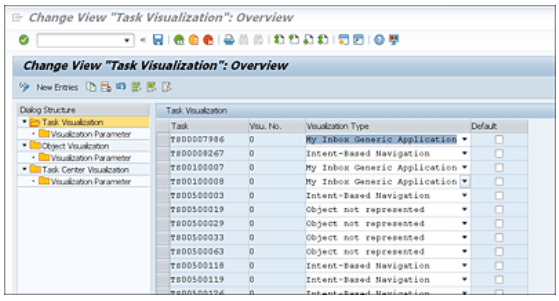


Figure 11.15 Transaction SWFVISU

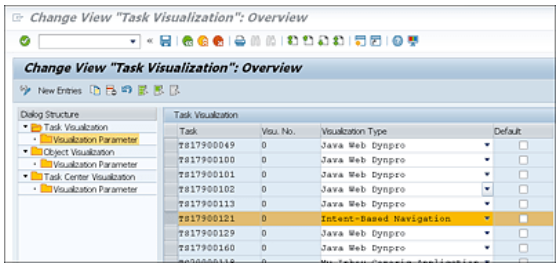


Figure 11.16 Visualization Parameter

6. As shown in Figure 11.17, you have to maintain the required values in the visualization parameters for intent-based navigation. These values are listed in Table 11.6.

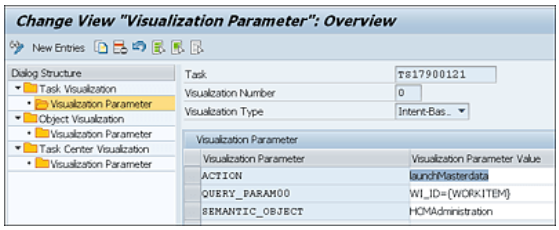


Figure 11.17 Visualization Parameters: Intent-Based Navigation

Visualization Parameter	Visualization Parameter Value
ACTION	Enter the correct action of your target application.
SEMANTIC_OBJECT	Enter the correct semantic object name of your target application. Check semantic objects details in the SAP Fiori app reference library or in the catalog.
QUERY_PARAM00	Enter the parameter value.
QUERY_PARAM01 . . . QUERY_PARAM09	Enter the parameter values.

Table 11.6 Intent-Based Navigation

7. Define visualization parameters **SEMANTIC_OBJECT** and **ACTION**. Parameters **QUERY_PARAM00 . . . QUERY_PARAM09** are used to define the application path, including the instance data and system data. The URL generation resolves the instance and system data for each parameter and concatenates the result string using the numbering of the parameters.
   Here's an example: `#<SEMANTIC_OBJECT>-<ACTION>?<QUERY_PARAM00>&<QUERY_PARAM..>`
8. Visualization type **My Inbox Generic Application** is another commonly used visualization type in My Inbox. As shown in Figure 11.18, you have to maintain all My Inbox–relevant parameter values listed in Table 11.7.

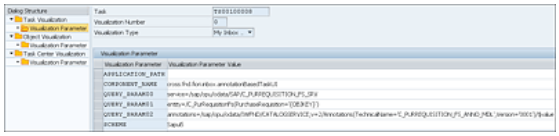


Figure 11.18 Visualization Parameters: My Inbox

Visualization Parameter	Visualization Parameter Value
APPLICATION_PATH	Enter the application path.
COMPONENT_NAME	Enter the application component name, for example: "cross.fnd.fiori.inbox.annotationBasedTaskUI".
QUERY_PARAM00	Maintain all required parameters, for example: "service=/sap/opu/odata/sap/FINCS_JRNLENTR_POST_SRV".

Visualization Parameter	Visualization Parameter Value
QUERY_PARAM01 . . . QUERY_PARAM09	Enter the parameter values.
SCHEME	Enter the scheme of the URL, for example, “http”, “https”, “sapui5”, and so on.

Table 11.7 My Inbox Generic Application

9. You can use the expressions and symbols in Table 11.8 when entering parameters. The basis of a workflow expression is always the task container. The leading object is stored in the generic container element `_WI_OBJECT_ID`, which is available in each task container.

Expression Name	Semantic
`{SYSTEM}`	System ID (`SY-SYSID`)
`{CLIENT}`	Client (`SY-MANDT`)
`{WORKITEM}`	Work item ID
`{TASK}`	Task definition ID, for example, TS78500100
`{OBJKEY}`	Business object instance ID (`{&<Workflow Expression>&}`) and result of workflow expression (e.g., `&_WI_OBJECT_ID.NAME&`)
`{SYMBOL01}` . . . `{SYMBOL05}`	Configured value in Transaction SWPA

Table 11.8 Expressions and Symbols

11.2.4 Launching Web Dynpro Applications from Workflow Tasks

Similar to SAPUI5 applications, you can configure other applications as well to launch from the My Inbox app. You need to follow the steps discussed in this section in Transaction SWFVISU to navigate to a Web Dynpro application.

As shown in Figure 11.19, follow the same steps as mentioned for SAPUI5 applications, and change parameter values as mentioned in Table 11.9.

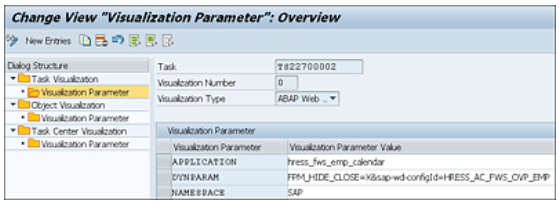


Figure 11.19 Web Dynpro Visualization Parameters

Visualization Parameter	Visualization Parameter Value
APPLICATION	Enter the Web Dynpro application name.
DYNPARM	Enter all Web Dynpro application parameters.
NAMESPACE	Enter “SAP”.

Table 11.9 Web Dynpro Visualization Parameters

After you maintain these settings in the backend, then you can navigate to the target app in the My Inbox app by clicking on the **Open Task** button, as shown in Figure 11.20.

Figure 11.20 My Inbox Open Task

11.3 Email Templates in SAP S/4HANA

Flexible workflows are the recommended approach in SAP S/4HANA. This flexible workflow framework is tightly integrated with the email templates approach. To align with SAP's "keep the core clean" approach, you always have to find the ways to minimize logins to SAP GUI and use classical ways to develop artifacts.

In SAP S/4HANA, you can create email templates using the Maintain Email Templates app. Almost all standard flexible workflows send email notifications by using these email templates out of the box. All you need to copy the standard template into the customer namespace by following prescribed naming standards, and then the standard framework will send emails to assigned recipients.

You can find complete details about these templates and the Maintain Email Templates app in Chapter 16, Section 16.2.

11.4 Integrating External Applications with SAP Business Workflow

Nearly all organizations have a hybrid environment with SAP, non-SAP, cloud, and non-cloud systems and solutions. Companies require integration between SAP and Non-SAP systems to support the new wave of automation and digitalization. There are different types of integration options (e.g., Simple Object Access Protocol [SOAP], REST, OData, etc.) available to establish communication between SAP and non-SAP systems to exchange the data through SAP Integration Suite. The preferred way of exchanging data between systems is through a whitelisted API. Whether it's SAP Build Process Automation, flexible workflow, or classical workflow, you need to connect with other systems through whitelisted APIs, which is a basic requirement to make your code cloud ready.

Another use case in today's scenarios is to integrate the workflow with Microsoft Outlook types of mail applications. End users have to check task details in their mail and take appropriate action directly from their mailbox rather than logging in to the system. SAP Business Technology Platform (SAP BTP) provides this kind of functionality seamlessly using SAP Build Process Automation techniques with any external mailing applications. This includes the no-code/low-code approach to build more robust automations. You need to explore SAP BTP if you get any such kind of requirement

in real life rather than implementing any old traditional approaches.

You can find the few latest SAP technologies in Table 11.10 for integrating external systems with SAP Business Workflow.

Technology	Details
SAP Integration Suite	SAP Integration Suite is an integration platform as a service (iPaaS) that helps you quickly integrate on-premise and cloud-based processes, services, applications, events, and data. SAP Integration Suite provides prebuilt integration flows managed and updated by SAP, as well as tools for designing, publishing, and managing APIs.
SAP Build Process Automation	SAP Build Process Automation allows you to simplify automation with visual drag-and-drop tools. It enables both citizen and professional developers to easily digitalize their workflows without writing code. There are hundreds of prebuilt content packages and connectors you can use to jump-start your automation projects and save development time. You can easily automate approval flows using SAP Build Process Automation tools and integrate seamlessly with Outlook, Excel, and so on.

Technology	Details
SAP Business Accelerator Hub	This is the first point of contact to search or find any released APIs. You can use this to discover, explore, and test the different types of content available in SAP Business Accelerator Hub to accelerate integration and extension and help you in your digital transformation journey.

Table 11.10 Integration Technologies

11.5 Summary

After reading this chapter, you should be able to set up the **My Inbox – All Items** and the scenario-specific **My Inbox** tiles, and you should be able to configure the launch of SAPUI5/Web Dynpro applications from the My Inbox app. Even if you're not a technical expert, you can configure the My Inbox app and use it for all approvals.

In the next chapter, we'll move on to discussing migrating SAP ERP workflows to SAP S/4HANA, starting with brownfield projects.

12 Migrating SAP ERP Workflows to SAP S/4HANA

This chapter provides an overview of the different migration options to SAP S/4HANA (greenfield, selective data transition, and brownfield). For brownfield conversions, you'll learn how to analyze the impact of migration on standard and custom workflows. For selective data transition and greenfield implementations, we'll discuss whether workflow migration is required and how to handle migration of open workflows, user data, and office settings from SAP ERP to SAP S/4HANA.

SAP S/4HANA is the new enterprise resource planning (ERP) product suite that runs on the SAP HANA in-memory database platform. It enables a large volume of data to process in real time, making it suitable for modern data-driven intelligent businesses. SAP S/4HANA provides a simplified data model, enhanced user experience (UX), improved real-time reporting and analytics, industry-specific solutions, new innovations, process enhancements, cloud integration, capability to integrate with modern technology, and a backbone for digital transformation. SAP has stopped all innovations in earlier SAP ERP products, and mainstream maintenance of SAP ERP will end in 2027. SAP S/4HANA is

the new future ERP product suite, so all organizations have started moving from SAP ERP to SAP S/4HANA. SAP also offers SAP Business Technology Platform (SAP BTP), which is a technical platform with collective integrated technologies and services where extended business applications and solutions can be built. The SAP landscape, product portfolio, and solution options have changed a lot with the introduction of SAP S/4HANA and SAP BTP. It's important to note that migrating from SAP ERP to SAP S/4HANA can be complex. In the workflow context, there are now multiple ways to build, and you should be able to decide the best option during SAP ERP to SAP S/4HANA migration.

In this chapter, we'll discuss different options for migrating from SAP ERP to SAP S/4HANA. We'll evaluate each option and recommend the option most suitable for a customer scenario based on requirements and current market trends. Then, we'll deep dive into each option in the context of workflow to find out how workflows can be migrated from SAP ERP to SAP S/4HANA for each option of migration. We'll talk about pre-migration activities and during migration activities, common challenges, and watchout points to help you prepare for any SAP S/4HANA migration activities. We'll only discuss workflow-related activities and not all the activities required in general for any SAP ERP to SAP S/4HANA migration.

12.1 Migration Options from SAP ERP to SAP S/4HANA

SAP S/4HANA is the successor of the SAP ERP product suite. This product is totally restructured with new technologies and more business-oriented functions and technical capabilities. The technical architecture surrounding SAP S/4HANA has changed quite a bit where business application logic and technology are residing in a different layer with string coupling. The SAP user interface (UI) has gone through drastic changes where it’s now totally based on SAP Fiori, and SAP GUI–based transactions are slowly being phased out. All new applications are built with cloud-based architecture, with web-based UIs, and on SAP BTP. In this changing business application and technology context, we have to revisit our implementation approach for workflows. This book contains all the supported technologies in the workflow area: both classical and new workflow technologies. The right technology should be selected for customer-specific scenarios.

In the SAP S/4HANA journey, there are three journey paths, as follows:

- **Brownfield project**
  This category includes organizations that want to do a technical migration to SAP S/4HANA. They want to transform their ERP system in phases slowly by building a technical backbone first without changing the process. They want to keep the change management to a minimum. Normally, the organizations have recently implemented SAP applications or are large organizations with a system supporting multiple countries globally and may prefer to have less disturbance in their stabilized system. Their business process is well aligned and stable in the system, and there is no major business driver to

change the business process or target operating model immediately. This kind of organization prefers a technical migration to SAP S/4HANA that is known as brownfield project.

- **Greenfield implementation**
  This type includes organizations that may have implemented SAP ERP a long way back. Their business process isn’t properly aligned in the system, or they may be starting a new line of business. Those organizations want to revisit the system implementation and build new, aligned with their business processes along with leveraging new technologies. They may go for a new SAP S/4HANA implementation. We call this type of SAP S/4HANA journey a greenfield implementation.
- **Selective data transition**
  The third approach is a mix of the greenfield and brownfield approaches and is known as a selective migration/smart migration/hybrid migration. Each organization has come up with a different name for this approach. Most of the existing investment is kept with some transformation capability. In this approach, a system is copied from an existing SAP ERP system. Then, the data is removed from the system, and only custom code and configuration is left in the system. The change-based configuration is aligned with the transformations needed for future business. Then, the data is moved to this system. Data migration happens in a table-to-table update mode. During the data migration, the data is changed on the fly and aligned with the new configuration. In a typical brownfield scenario, you can’t change certain types of finance configuration because

data is already there. This approach moves the data later, so it has the flexibility to overcome that limitation. This can provide a solution for data enrichment, cleanup system, business transformation, and reduced downtime. This approach is getting popular today because data is migrated directly from table to table, the SAP Partners technically can’t do this kind of data migration. There are some specific companies that have been working in this area for a long time, so they are involved in this kind of SAP S/4HANA migration approach.

Table 12.1 provides the key purpose of the different kinds of SAP S/4HANA migration approaches in the market.

Approach	Purpose
System conversion (brownfield)	Modernize the platform with minimum business process change: • In-place technical system conversion with minimum end user impact • Adoption of new innovations at the client’s own speed • Main goal of technical platform modernization

Approach	Purpose
New implementation (greenfield)	Reengineer with a new implementation or reimplementation: • Reengineering and process simplification based on latest innovations • Implementing innovative business processes with preconfigured content on a new platform • Performing initial data load • Retiring an old landscape
Selective transformation (copy, transform, deploy)	Value-driven process change in a modernized platform: • Clean up existing system from a data and customization perspective. • Clear up a lean and clean system • Reuse the existing investment and change only where business value is identified • Move the selected data needed for business operation to keep the system lean

Table 12.1 Different Options of Moving from SAP ERP to SAP S/4HANA

We’ll now discuss these three implementation approaches in the workflow context only.

12.2 Conversion Projects (Brownfield)

An SAP S/4HANA technical conversion is similar to any upgrade project, but it’s much more complex than an SAP upgrade because it brings changes and new things both in technology and business processes. We’ll first discuss how to handle the system upgrade. Then, we’ll extend the discussion to SAP S/4HANA conversion.

12.2.1 Handling System Upgrades for Standard and Custom Workflows

Like upgrade impacts any other area, there can be an impact both from the design and runtime instance perspective for workflow objects. The impact will be higher if the current system version is lower. There is a higher probability of impact if you’re upgrading your workflow from SAP version 4.7X or lower. Following are some minimum recommendations that you should be following to make your system upgrade successful and to cause minimum impact to end users:

- **Plan the upgrade activities well ahead of time.** Like any other major system change, you have to start planning your activities related to analysis and upgrade. The entire planning can be categorized under three main pillars: analysis of impact before upgrade, pre-upgrade and mid-upgrade activities, post-upgrade activities, and end user support.

- **Perform pre-upgrade analysis.**
  Understand what is changing in the upgrade and the known issues. Identify the SAP Notes that talk about the upgrade impact on the workflow. Work with your Basis team to get the list of SAP Notes that gets implemented as part of the upgrade. The Basis team can extract the SAP Notes based on the source and target SAP version. You should also ask to get the list of side effect SAP Notes for the target version you're going to. There will be a very high number of SAP Notes, and it won't be possible to check every SAP Note in that list to check workflow impact. Look for the SAP Notes with component BC-BMT-WFM. This will help you filter out the SAP Notes related to workflow. You should also look for other consulting SAP Notes that may not be part of that list. Search for "Collective SAP Note for Upgrade" to provide you some guidance on how to handle the workflow update. Table 12.2 lists some SAP Notes that will be helpful in an upgrade scenario.

SAP Notes	Description
1068627	Composite Note about Workflow Upgrade
573656	Collective Note Relating to Archiving in Workflow
910573	Structures with Incompatible Changes
3200555	Workflow Syntax Errors after Upgrade
215753	Upgrade: Old Workflows Hang

SAP Notes	Description
2665105	Workflow Start/Restart Failure after Support Package Upgrade - Binding Failure
2152421	Workflow Function Modules Aren’t Working as Expected after System/Support Package Upgrade
2760822	Workflow Customizing Is Changed after Upgrade
2075962	Binding and Container Element Errors in Workflow Runtime after Upgrading
1058159	Elements Are Missing in the Workflow Container
939489	Workflow: Container Elements Missing after an Upgrade
1228836	Compatibility of Conditions with Date/Time Constants
1732734	Binding Errors in Workflow Runtime after Updating your Basis Support Pack or Upgrading

Table 12.2 Key SAP Notes Related to the Impact of the System Upgrade on the Workflow

- **Complete the running workflow instance.**
  Most of the workflow issues occur in the existing in-process work items after an upgrade. A workflow maintains a lot of runtime information. If the workflow

container element structures are changed a lot or some existing elements aren’t compatible in the new SAP version, then there may be a chance an old workflow may give some errors. There is no definite reason the workflow will give errors after the upgrade, but we’ve seen some workflows errored out after the upgrade, so it’s recommended to complete all the existing workflows. That being said, we all know it’s not practically possible to complete all workflows in the system. You have to reasonably try to complete as many as possible. There will be some long-running workflows, so it won’t be possible to close all. Look for the workflows where the lifecycle is shorter or the technical workflows such as IDoc error handling workflows. In addition, look for workflows that are toward the end of their lifecycle. You can target those workflows as well. The chances of error will be high if the difference of source and target SAP version is high and the source SAP version is very old. Normally, within an SPS upgrade, you won’t find too many errors.

- **Identify the new workflow version in the standard workflow.**
  Check for the workflow having new version in the upgraded version of the workflow. Workflow runtime instances are version dependent. Any in-progress workflow will follow the execution path defined in the old version. All new triggered workflows will follow the new version of the workflow. So, users may report differences in behavior of the same workflow. Identify the difference between the workflow versions and document them as part of user training. When you’re analyzing new version of the workflow, you have to look for both technical

changes and end user impact changes. The end user impact is used to feed into the training content. Technical changes will help you understand if there will be any impact on existing workflow processing.

- **Recognize the impact on the modified SAP standard workflow templates.**
  It's not recommended to directly modify standard SAP-delivered workflow templates. But if you've done so, those modified workflow templates will be overwritten during the upgrade. SAP standard template modification isn't considered a repaired object. Therefore, if any standard workflow template is modified, it won't appear in the Transaction SPAU list during the upgrade. Prepare the list of such workflow objects. You have to revert back to the version or redo the changes after the upgrade.
- **Copy workflow templates from the standard SAP template.**
  You can check for the changes delivered in the standard workflow templates. If new standard SAP version of the workflow meets the business requirements, then you can replace the custom workflow with the standard one. It's always recommended to use the standard workflow templates.
- **Plan testing of the workflow.**
  It's not possible to check all the impacts on the workflow due to the upgrade, so test planning plays a bigger role in upgrades. Keep every workflow in your test plan, and be sure to keep both in-progress workflows and newly triggered workflows part of your test plan. You have to create some workflows before the upgrade happens and

keep different approval stage work items in the inbox. This will help to test all the needed scenarios. Because you won't be able to generate a workflow with a pre-upgrade version after the upgrade completes, you have to create enough work items for testing before you start your upgrade.

- **Perform a technical check in the upgraded environment.**
  Check the following system configuration in the upgraded environment. You should have a list of workflows used in the system, so only check for those specific workflows:
  - Check for event-specific configurations, as listed in Table 12.3.
  - Check syntax is free of errors for custom-developed workflow: Open all the workflows in Workflow Builder, and do a syntax check of the customer workflow. Check for the container element binding syntax error.
  - Check automatic workflow customization (Transaction SWU3). Refer to Chapter 8 for the node you should check for completeness.

  Synchronize the workflow buffer by running Transaction SWU_OBUF after all changes/activations are done.

Transaction Code	Transaction Description
Transaction SWE2 or Transaction SWETYPV	Check for event linkage activation for workflow
Transaction SWEC	Check for event activation on change document

Transaction Code	Transaction Description
Transaction BSVW	Check for workflow activation in status management
Transaction SWB_COND	Check start condition for workflow

Table 12.3 List of Event-Specific Configuration Transactions That Should Be Checked after Upgrade

- **Change in workflow container persistence data**
  The backend table to store persistence of the workflow container is changed from SAP R/3 4.7 release onward. Very few organizations are still in the previous release, so you won't find many instances of this change. Previously, it was called the container structure persistence, which is stored in table `SWW_CONTOB` and table `SWW_CONT`. Now it's used as XML persistence, which is stored in table `SWWCNTP0`. If you find any workflows still using the old persistence, switch to the new persistence for better performance. There will be an impact on the custom table if any customers are still using table `SWW_CONTOB` and table `SWW_CONT`.

12.2.2 Migrating Your SAP ERP Workflows to SAP S/4HANA

When you're migrating to SAP S/4HANA, all the points mentioned in Section 12.2.1 for system upgrades will be applicable in this scenario as well. There are more changes in the application layer and application tables in an SAP S/4HANA migration. You need to assess the impact of the

application layer logic change and ABAP Data Dictionary change impact, as follows:

- **Decide how many new features to bring into an SAP S/4HANA brownfield conversion**
  There are significant changes in how the application is used by end users and the technology used in SAP S/4HANA. The old technology and UIs are still compatible with SAP S/4HANA, so you have to decide along with the project stakeholder how much redesign and change should be brought in the workflow area in a brownfield conversion. There are several customer adoption scenarios we've seen in a brownfield SAP S/4HANA conversion. Some customers don't want to make any significant change to end user usage when going for an SAP S/4HANA conversion. They plan a subsequent project after the SAP S/4HANA technical conversion for the business transformation. Some customers bring the transformation along with the SAP S/4HANA conversion. In the workflow context, you have to check if the following scenarios are applicable in your project:
  - **Is there any change in the UI?**
    SAP Fiori is the recommended UI for SAP S/4HANA, although you can still use SAP inbox. If the users were using SAP GUI before the conversion and now SAP Fiori or WebUI will be used, then all the workflow should be tested in the new UI. With HTML for GUI, the classical workflow screens will be available to the user. Because it's a new UI, you may find out some changes or issues from an end user perspective. In addition, work with your UX team to decide what will be used for the user

inbox: SAP inbox, Businesses Workplace, or a combination of two.

- **Will any workflows be migrated to flexible workflow?**
  SAP provides flexible workflow in SAP BTP along with classical workflow. Depending on the project scope and approach, you have to decide if any of the custom or standard workflows can be migrated to flexible workflow or on SAP BTP.
- **What will your design strategy be?**
  You have to come up with a design strategy on how new workflows will be developed in SAP S/4HANA. There are three options still available: classical workflow, flexible workflow, and SAP Build Process Automation.

- **Custom code impact due to SAP S/4HANA conversion**
  There are significant changes in the application layer when you move from SAP ERP to SAP S/4HANA. The function module and ABAP Data Dictionary objects are changed. Some of these became obsolete or don’t exist in the SAP S/4HANA system. Business object or workflow class custom objects should be remediated to accommodate this. As part of SAP S/4HANA custom code remediation, you have to set up ABAP Test Cockpit with the SAP S/4HANA remediation variant. Use the SAP S/4HANA version variant that is relevant for your version. The ABAP Test Cockpit will provide you the list of custom code that needs to be remediated. It will also refer you to the SAP Note number that will tell you about the change

that happened in SAP S/4HANA. This will help you remediate your custom code.

- **Impact on container element**
  Because there are lots of simplifications in the ABAP Data Dictionary area, you have to validate each container element and binding. You can go to the Workflow Builder and container binding screen to do a syntax check of the workflow and take corrective action accordingly.
- **Impact of change in the transaction layout**
  The business application screens are changed a lot in SAP S/4HANA from SAP ERP. You'll find a totally new SAP Fiori–based transaction available in SAP S/4HANA. For the same transaction, your workflow may be using an old GUI-based transaction. If the project decides to use the new SAP Fiori-based apps, then it will be confusing for users to use old GUI-based transaction for workflow and new SAP Fiori apps for other purpose. you may have to rebuild the entire workflow in Flexible workflow. Identify all your existing workflows that call SAP ERP transactions or screens. Evaluate those workflows regarding how they should behave in SAP S/4HANA, and then remediate or rebuild the workflow.
- **Impact due to obsolete transactions used in work item processing**
  Some transactions are obsolete or don't exist in SAP S/4HANA, for example, Customer Master Create/Change/Display (Transactions XD01/02/03/06, Transactions FD01/02/03/06) or Vendor Master Create/Change/Display(Transactions XK01/02/03/06/07). Business partner is the new application object that should

be used to manage customers and vendors. If you have any workflows based on these obsolete transaction codes, then you have to redevelop your workflow in SAP S/4HANA.

- **Change in workflow system user and workflow system jobs**
  You'll find that automatic workflow Customizing is no longer green in the **Maintain Runtime Environment** node after system conversion to SAP S/4HANA. This is a change from SAP S/4HANA 1709 version onward. The SAP workflow system user `WF-BATCH` is changed to `SAP_WFRT`, so all the configuration where `WF-BATCH` was used needs to be changed to the `SAP_WFRT` user. If you have any custom code checking for the `WF-BATCH` user, those should be changed. Ideally, any custom code should not have any logic based on the user ID. You should validate the requirement and implement the solution accordingly.
  In SAP S/4HANA, the system makes sure user `SAP_WFRT` exists and has role `SAP_BC_BMT_WFM_SERV_USER_PLV01` assigned. You'll need to ensure this role is properly generated and showing the green traffic light.
  The job scheduling mechanism is also changed in SAP S/4HANA. The jobs are managed by Transaction SJOBREPO (Technical Job Repository). Workflow system jobs are scheduled automatically, and the job name starts with `SAP_WORKFLOW`. You have to deschedule all the old jobs in the system in all clients. Jobs include `SWWDHEX`, `SWWERRE`, `SWWCOND`, `SWWWIM`, `SWEQSRV`, and `SWWCLEAR`.
  In the Technical Job Repository (Transaction SJOBREPO), automatic scheduling of jobs must be switched on. System job `R_JR_BTCJOBS_GENERATOR` runs every hour by

default. This job schedules all workflow jobs starting with `SAP_WORKFLOW`. Refer to SAP Notes 2568271 and 2190119 for more details.

- **Archive workflow objects to keep the system lean**
  Archiving is always recommended to keep the system lean. Archive is more relevant for SAP S/4HANA conversion because you're moving to a platform. It's a good opportunity to do your housekeeping as well.
  There are two ways to clean up old workflow information. You can archive workflows using archive object `WORKITEM`. It considers the dependencies of work items and archives a complete logical set of workflow data. There is a standard report RSWWWIDE to delete the workflow-related content from tables. This doesn't check the dependencies of the work items, so you should be very careful when you're executing this report.
  There are two kinds of workflows in the system: *workflow* manages business transaction data that is auditable, and *technical workflow* processes errors such as IDoc error handling workflows. It's recommended to use the archiving process for auditable workflows. You can consider deleting the technical workflows that aren't required in future. It should be discussed with the audit team and project stakeholders before you finalize the design.
  Program RSWWWIDE doesn't consider the dependencies, so deleted work item–dependent data may stay in a few other tables. You should execute the RSWWWIDE_DEP and RSWW_REORG_SWWUSERWI programs to delete the redundant entries from other tables. Refer to SAP Note 1427068 for further details.

12.3 New Implementation Projects (Greenfield) and Selective Data Transition

Now let’s move on the greenfield new implementations and selective data transition. In a greenfield approach, everything is built from new. Existing workflows should be completed in the current SAP ERP system, and new workflows should be started in the new SAP S/4HANA system. You should use the appropriate workflow development option as mentioned in Chapter 1, depending on the business requirements and supported technologies. Because there is no workflow migration requirement in a greenfield implementation approach, we won’t discuss the greenfield option in this chapter.

We’ll now discuss in detail what you should consider for workflow migration when the customer has chosen the selective data migration approach to move to SAP S/4HANA. This wasn’t a very technical approach in the SAP ERP world for a new SAP installation, but we’ve seen an increase in this approach adaptation recently. We’ll start with an overview of selective data transition. You need to understand the basic technology behind this approach to help you anticipate where challenges may appear and why. This will help you make custom scenario-based decisions when needed. There are two components of migrations you should always consider: (1) developed workflow objects and configurations, and (2) runtime data in the current system. The migration strategy for developed workflow objects and

configuration should follow the approach mention in Section 12.2. Handling workflow runtime data needs special attention. We've discussed that in the subtopic in this section. Finally, we'll be talking about how to handle the user master and SAP office settings.

12.3.1 Selective Data Transition Overview

Prior to SAP S/4HANA, selective data transition was used mainly for system merges, acquisitions, divestitures, and organizational change scenarios. However, it's now becoming very popular in SAP S/4HANA migration scenarios also. SAP S/4HANA migration isn't just a technical upgrade project; it provides an opportunity to reengineer your process, cleanse and transform data, harmonize and optimize the system landscape, and merge systems to consolidate the systems. The selective data migration approach supports all these scenarios where you can keep the organization-specific past investment with the new transformation and functionalities of SAP S/4HANA and can implement it more quickly.

Let's now briefly walk through how selective data migration works, as follows:

1. Create an empty shell of the SAP ERP system. This empty shell contains only configuration and custom code from the source SAP ERP system. There are two approaches to create this empty shell: (1) You can create the empty shell from an existing SAP ERP system, which will bring the code and configuration automatically; or (2) you can do a fresh installation of an

SAP S/4HANA instance in which you have to bring the selective configuration and code using some tool or do it manually. This is known as the mix-and-match approach.

2. If the empty shell is created from the current SAP ERP system, then you do a conversion of this system to SAP S/4HANA. This will be simpler and faster because this system doesn’t contain any data.
3. You do SAP S/4HANA-specific configuration and custom code remediation in the converted SAP S/4HANA box. If it’s a fresh installation of SAP S/4HANA, you also do the new configuration and system setup here.
4. You migrate the data from the source SAP ERP system to the target SAP S/4HANA system. This data migration happens as a table-to-table migration, so you can bring historical data as well, which isn’t possible in a typical greenfield implementation where you use application programming interfaces (APIs) to move data. During this data migration, all kinds of additional data selection and transformation rules can be applied. Following are some examples of the scenarios for selective migration to SAP S/4HANA:
    - Move X years of data to the target SAP S/4HANA system.
    - Harmonize and reduce the number of charts of accounts during data migration.
    - Don’t move data related to sold companies.
    - Don’t move master data flagged for deletion and their corresponding transactional data.

- Transform the data alignment with the new organization structure, configuration, or master data.
- Merge two SAP ERP systems into one SAP S/4HANA system (system consolidation scenario).
- Carve out only specific organizational unit–related data to SAP S/4HANA (divestiture scenario).
- Use the near zero downtime approach to reduce the overall business downtime during go live.

Selective data migration to SAP S/4HANA can only be performed by some selective specialized company with their toolset. SAP established an expert group of partners in 2018 called the SAP S/4HANA Selective Data Transition Engagement. These five partners are SNP, SAP, CBS, Datavard (now part of SNP), and Natuvion. If you're considering the selective migration approach for SAP S/4HANA migration, then you have to work with one of these partners. You can't build your own tool for selective data migration.

12.3.2 Managing the Technical Migration of In-Process Workflows

Now we'll discuss what should be considered when we're thinking of data migration of workflows and selective data migration. This is one of the items that comes up in all project discussions: whether workflow-related data can be migrated to SAP S/4HANA and what are the limitations. As a workflow expert, you'll be involved in those discussions.

Before you decide if the workflow data should be migrated to SAP S/4HANA systems, you need to know when it can or can’t be migrated to SAP S/4HANA. You’ll see three scenarios of selective data migration:

- **Full data migration to SAP S/4HANA system without any change or data filter**
  You can consider moving the workflow-related data into your SAP S/4HANA system. The data will be migrated as is from SAP ERP to SAP S/4HANA. Follow the recommendation provided for the brownfield conversion in this case as well. Though it’s possible to move all workflow data as is, it’s recommended to complete workflows in the source system as much as possible.
- **Selective data migration where selection is based on number of years/date and time**
  This selection rule is normally applicable for business application objects. How this selection logic will be applicable for workflow objects should be verified. If the work item creation date is with the date selection, that is still doable. But it can create problems with dependencies. For example, per the project requirement, you have to move the data created/changed in the past three years. A sales order approval was started a little bit earlier than three years, but after the approval, the sales order is eventually created within the “past three years” window. If you now go by the selection rule, that is, data created within the past three years, you’ll move the sales order, but you won’t move the workflow because it was started earlier than three years. So, you’ll have a sales order in the system, but you won’t find the approval history of that sales order. You should consider all these kinds of

dependencies and set up business expectations accordingly. Some of the dependencies can be handled technically. But it will be too complex and may not be worth it to build such complex logic during data migration. You have to balance between what is technically possible and what is realistic in the context of the project.
You should also consider the workflow-dependent technical data. For example, if a work item meets the selection rule, then you have to bring the entire workflow and all other work item and log information of that particular workflow. You should consider these exceptions and business user impact if you still want to bring workflow data. These kinds of dependencies make the extraction rule complex and performance intensive, increasing downtime. It's technically possible to bring this workflow data, but it will have other impacts and complexities. You have to consider this and define your data selection approach accordingly.

- **Selective data migration in which selection is based on an organizational unit (e.g., company code)**
  In this scenario, you want to bring only certain organizational unit– or master data–specific workflows. The workflow container contains the business data in an XML persistence layer. You have to read individual business object instance or workflow container instance data, extract that information, and then decide if this specific workflow should be migrated or not. Though it sounds technically possible, it has a huge performance impact during selection, so it's recommended not to bring this kind of workflow, if possible. It's better to complete

the workflow in the source system and retrigger it again in the target SAP S/4HANA system.

- **Other selective data migration where data is transformed during migration**
  If the data is transformed during the import part of the data migration, it's not possible to move workflow data into the target system. Normally, the transformation rule is applied at the domain level. It's not possible to apply the same rule for the workflow, so workflow data usually isn't moved to the SAP S/4HANA system if data is transformed during migration.

You should think of how to close the existing workflow and how to restart partially completed workflows in the new systems if workflows aren't migrated to a new system. Change management activity will occur, which should be planned early.

You also need to think of attached documents for workflow migration. The documents attached in a workflow are usually stored in a content management repository system. SAP tables only contain the link to the documents in the content repository. There won't be any impact if there is no change in the content management repository system. But SAP S/4HANA projects may also involve infrastructure migration. If your content repository is also migrating, check that there is no change in the object link.

12.3.3 Migrating User Data and SAP Office Settings from SAP ERP to SAP S/4HANA

Usually, user master data isn’t migrated directly from the SAP ERP production system to the SAP S/4HANA production system in a selective migration scenario. There are other standard options to migrate user master data from existing SAP ERP systems to SAP S/4HANA systems. Therefore, the base principle of table-to-table migration for selective migration isn’t generally used here. Keep in mind that SAP ERP security roles will go with significant remediation and change when you’re migrating from SAP ERP to SAP S/4HANA. There will be new roles introduced for SAP Fiori apps and any new business process if you’re activating. There will be adjustments to existing roles to make them compatible with new roles and replaced transactions in SAP S/4HANA. These role changes will move in a transport, so you have to keep in mind the sequence of role change transport and user master migration. Role change transport should be moved after the user master migration. Transport movement happens during uptime as part of the system build. This activity is completed a couple of weeks before the downtime. That means you have to move the user master data a couple of weeks before, so there will always be a mismatch of delta that is updated during these days. It’s advisable to have a change freeze during this time. Here are some of the common approaches normally taken to move user master data:

- **User master provisioning tool**
  Lots of organizations use a user master provisioning tool to create and update user master records, such as SAP Identity Management. Those applications have the capability to re-push the current user master records. You can consider these tools to recreate the user master in

the productive system. You have to check if there is any exception such as system users that aren’t managed using the user provisioning tool. Those need to be handled manually.

- **User master export/import**
  You can export the user master from SAP ERP and import it into the SAP S/4HANA system. If the name of the roles aren’t changed in the target SAP S/4HANA system, then it should get imported without an error. There are some concerns that come up because you’re importing user master in a different version of the product, so it’s not a preferred option in some projects. However, we’ve seen it work. It should be tested in a nonproductive environment first before doing it in production.
- **Custom program**
  You can write a custom program to read the user master data and create it using a business application programming interface (BAPI). It works mostly, but you won’t be able to bring the new password.
- **Manual creation**
  If the number of users is too low, you may think of recreating it manually, but this isn’t a preferred option. There can be manual input errors during creation.

SAP office settings represent another area where you have to clearly discuss with your Selective Data Migration partner regarding the strategy of migration. In general, if the workflow is moved from the source SAP ERP system to the target SAP S/4HANA system, then the office settings also get migrated along with the same data objects. All objects related to office settings, for example, workflows, SAP office

folder, distribution lists, and so on, are interdependent. It’s recommended to move everything in office rather than have further selection for tables. That will reduce the data migration complexity and create a better UX after the SAP S/4HANA migration.

12.4 Summary

We’ve discussed three implementation approaches for SAP S/4HANA: brownfield conversion, greenfield implementation, and selective data migration. We’ve talked about different scenarios and market trends of implementation scenarios for each implementation approach. We’ve discussed in detail what should be considered and validated in a system upgrade and an SAP S/4HANA conversion project. Then, we explained the different workflow build approaches available in SAP S/4HANA and when to adopt these. We’ve discussed the selective migration approach with different use cases and explained the additional complexities the selective migration approach brings in the context of workflow migration.

13 Workflow in SAP Master Data Governance

SAP Master Data Governance (SAP MDG) uses SAP Business Workflow to define and execute validation and approval workflows for master data change requests. These workflows route change requests created by requestors to data stewards and other stakeholders, allowing them an opportunity to review and approve any changes before these changes become effective.

Master data records such as customers, suppliers, materials, and general ledger accounts are used in many purchasing, sales, and accounting business processes. Having accurate master data records is a critical requirement for running these business processes efficiently. For example, having an accurate delivery address for customers ensures that deliveries are made on time, and better material master data improves inventory management and ordering processes. Many organizations have legal and regulatory requirements for data privacy, retention, and destruction of sensitive data, as well as to maintain the audit trail of changes to critical data records. Master data records are also used in reporting, and analytics and having accurate master data records is a prerequisite for achieving timely

and accurate reports that are essential for effective decision-making.

Organizations implement SAP Master Data Governance (SAP MDG) to improve quality and trust in their master data by having a centrally governed single source of truth for master data records. When various business units of an organization share master data, each master data record has multiple stakeholders from one or more business units. Any changes to a shared master data record can impact business processes in multiple business units. Organizations need to ensure that changes to master data records are reviewed and approved by all stakeholders before these changes become effective. Approval processes depend on the type of master data that is being changed and the type of changes that are being requested. SAP MDG provides preconfigured workflow templates that implement commonly used approval processes for customer, supplier, material, and financial master data across various industries.

In this chapter, you'll learn about various workflow templates available in SAP MDG and how these templates can be used, customized, and enhanced to meet specific requirements. We'll explore workflow templates used in validation and approval of business partner, supplier, customer, material, and financial master data. We'll also cover a rule-based workflow template that can be configured to model simple as well highly complex approval workflows.

13.1 Introduction to SAP Master Data Governance

SAP MDG is a master data management solution providing preconfigured data models, workflow templates, and user interface (UI) application to centrally create, change, and distribute master data records to SAP and non-SAP systems across the entire system landscape.

A typical process in SAP MDG starts when a user creates a change request using a UI application. Change requests can also be created using a system interface when changes are initiated by another system and sent to an SAP MDG system using a system-to-system interface. Either way, a change request is created, and a workflow process is initiated. A data specialist reviews the changes requested and ensures that all required fields are populated and that the requested changes meet the data validation rules defined for the type of master data record that is begin changed. Many of these validation rules are executed by the system in the background, and the results of the validation are available to the data specialist.

If changes are incorrect, the data specialist can reject the change request and send it back to the requestor with a note explaining why the change request is being rejected. The requestor can make additional changes and resubmit the request or withdraw it. Once the data specialist approves the request, the request is typically forwarded to the data steward for final approval. At this step, the final approver can approve or reject the request. When the request is finally approved, changes are activated and are

visible using the standard data maintenance applications and transactions. Changes are also often distributed to one or more systems so that all these systems receive the updated and approved master data records. Figure 13.1 shows typical steps in a change request process approval workflow.

At any point in time, all stakeholders, including requestors, data specialists, and data stewards, can view the status of the change request, a list of actions taken so far, and the next steps in the workflow. This audit trail is available even when a change request is finally approved or rejected and is used to analyze the change request process and prepare reports on master data changes.

Master data undergoing changes is copied to a staging area, where changes are made to the data and validations are executed. This ensures that changes being requested and reviewed aren’t active and don’t affect ongoing business processes. Only after changes are approved are they activated and copied from the staging area to the active area. Only the activated changes are then distributed to other systems in the landscape. While a master data record is undergoing changes, it’s typically locked for further editing to ensure that multiple changes don’t overlap. However, parallel change requests are also possible, which allows multiple changes to be executed.

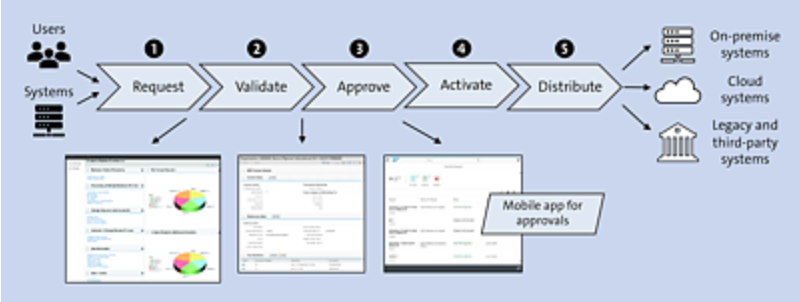


Figure 13.1 Typical Steps in a Change Request Approval Workflow Process in SAP MDG

13.1.1 Data Models in Central Governance

A data model in SAP MDG defines the structure of the data that is being governed. A data model consists of entities, attributes, and relationships. SAP delivers data models for business partner, material, and financial master data records. An entity in a data model is a set of attributes used to store different types of data in the data model. An entity in SAP MDG is like a table in SAP Data Dictionary; in fact, a table is generated for each entity in the data model. An entity contains several attributes, and each attribute defines one field of a master data record. Entities are linked with each other using relationships. The relationship type determines whether one entity type is at a higher level than another entity type or whether it's to be copied as an attribute of the other entity type. Figure 13.2 shows standard data models delivered by SAP. Data models can be displayed using Transaction MDGIMG via menu path **Master Data Governance, Central Governance • General Settings • Data Modeling • Edit Data Model**.

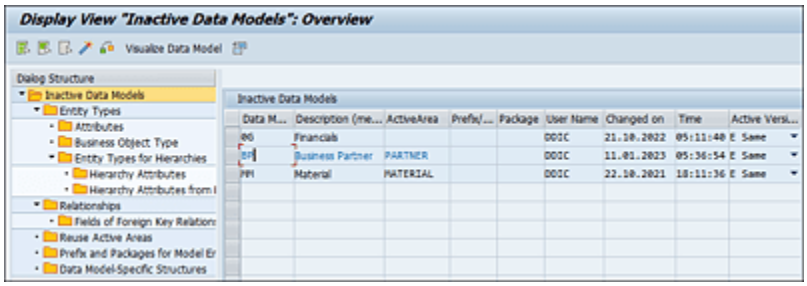


Figure 13.2 Standard Data Models Delivered by SAP

A key concept in an SAP MDG data model is storage area. There are two storage areas in SAP MDG, as follows:

- **Staging area**
  The staging area contains data that is currently being changed though a change request. The system uses the data model to generate tables for storing the data that is undergoing change. When a change request is created, data is copied from the active area into the staging area, and all changes are applied to the records in the staging area tables. Validations and derivations are executed on the data held in the staging area. If a change request is activated, data is copied from the staging area back to the active area and removed from the staging area. If a change request is rejected or withdrawn, data in the active area remains unchanged and is removed from the staging area.
- **Active area**
  The active area contains data that is ready to be distributed to other systems or ready to be used by other applications and systems. Data can be loaded in the active area using standard data migration tools. Data in the active area can be used to create a change request. When a change request is created, data is copied from the active area into the staging area. The system copies data

from the staging area into the active area once a change request is activated.

13.1.2 Change Request in Central Governance

A change request in SAP MDG is a container that holds changes to the master data that is being changed. A change request is created for create, read, update, and delete (CRUD) operations and is configured to support the end-to-end process of requesting master data changes, deriving attributes, validating changes, approving or rejecting the changes, and finally activating or withdrawing changes. In the following, we'll discuss some of the important properties of change requests that are relevant to workflows:

- **Change request type**
  A change request is always associated with a change request type. A change request type determines the type of data that can be changed and type of change. The standard system provides several preconfigured change request types, for example, change request type `SUPPL1P1` is used to create a supplier, and change request type `MAT02` is used to change material master data. Figure 13.3 shows some of the standard change request types delivered by SAP. Change requests can be edited with Transaction MDGIMG using menu path **Master Data Governance, Central Governance** • **Process Modeling** • **Change Requests** • **Create Change Request Type**.
- **Change request step**
  Every change request type has one or more change

request steps associated with it. Examples of change request steps are submission, validation, approval, revision, activation, and so on. A change request can go through one or more of these steps depending on the type of data that is undergoing change, type of changes that are being made, and the decisions made by user agents during the processing of the change request. Depending on the type of workflow template used for the change request, change request steps are defined differently. For rule-based workflow, change request steps are defined using Transaction MDG following menu path **Master Data Governance, Central Governance • General Settings • Process Modeling • Workflow • Rule-Based Workflow • Define Change Request Steps for Rule-Based Workflow**. You'll learn about rule-based workflow in Section 13.1.10. Figure 13.4 shows change request steps defined for change request types that use rule-based workflow.

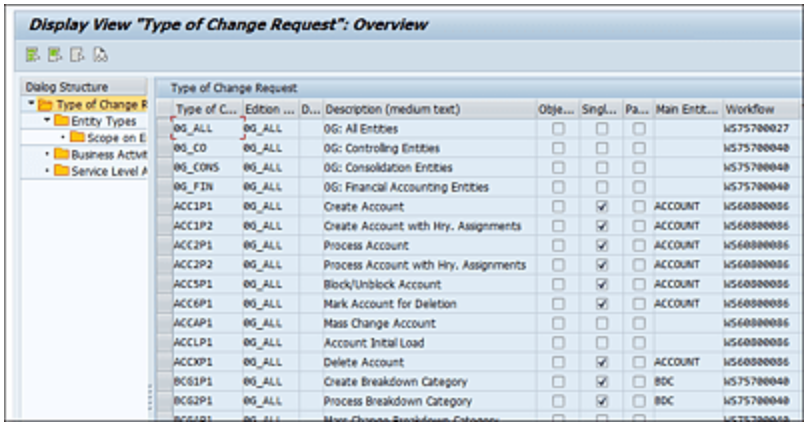


Figure 13.3 Standard Change Request Types Delivered by SAP

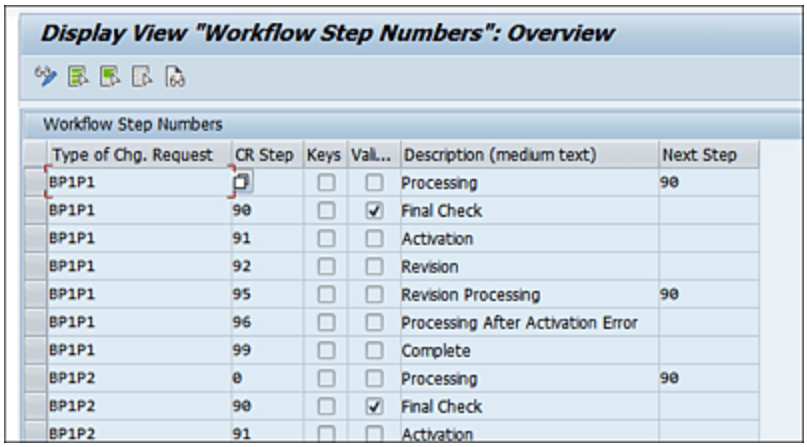


Figure 13.4 Change Request Steps for Change Request Types That Use Rule-Based Workflows

For change request types that use preconfigured workflow templates, change request steps can be defined using Transaction MDGIMG and following menu path **Master Data Governance, Central Governance • General Settings • Process Modeling • Workflow • Other MDG Workflow • Define Change Request Step Numbers**. Figure 13.5 shows change request steps defined for preconfigured workflow templates.

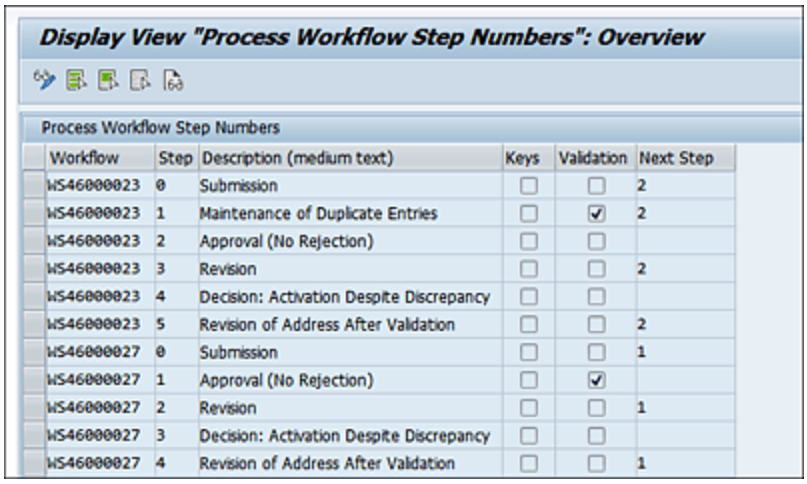
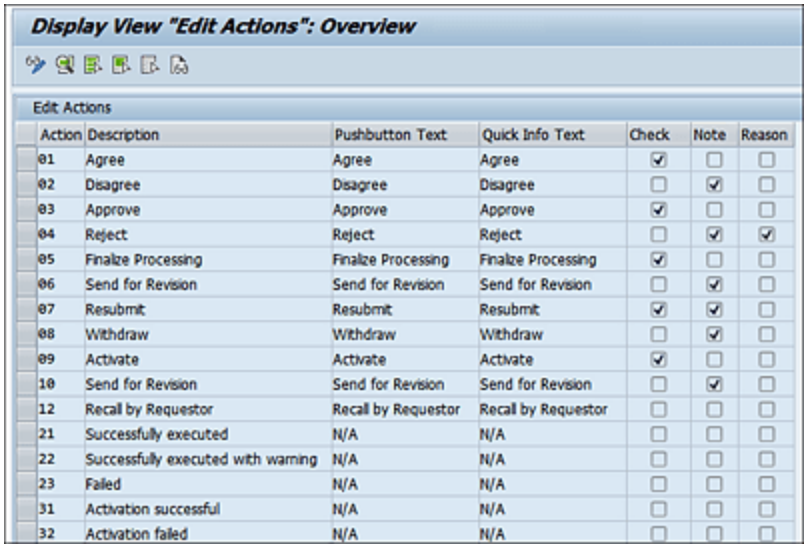


Figure 13.5 Change Request Steps Defined for Preconfigured Workflow Templates

- **Change request actions**
  A user that is processing a change request can perform one or more actions that correspond to the buttons on the UI application. If a change request task is a background task, the system executes the task, and the result of the execution is determined as one of the possible actions for the step. Figure 13.6 shows some of the standard change request actions delivered in the standard system.



Display View "Edit Actions": Overview

Edit Actions

Action	Description	Pushbutton Text	Quick Info Text	Check	Note	Reason
01	Agree	Agree	Agree	☑	☐	☐
02	Disagree	Disagree	Disagree	☐	☑	☐
03	Approve	Approve	Approve	☑	☐	☐
04	Reject	Reject	Reject	☐	☑	☑
05	Finalize Processing	Finalize Processing	Finalize Processing	☑	☐	☐
06	Send for Revision	Send for Revision	Send for Revision	☐	☑	☐
07	Resubmit	Resubmit	Resubmit	☑	☑	☐
08	Withdraw	Withdraw	Withdraw	☐	☑	☐
09	Activate	Activate	Activate	☑	☐	☐
10	Send for Revision	Send for Revision	Send for Revision	☐	☑	☐
12	Recall by Requestor	Recall by Requestor	Recall by Requestor	☐	☐	☐
21	Successfully executed	N/A	N/A	☐	☐	☐
22	Successfully executed with warning	N/A	N/A	☐	☐	☐
23	Failed	N/A	N/A	☐	☐	☐
31	Activation successful	N/A	N/A	☐	☐	☐
32	Activation failed	N/A	N/A	☐	☐	☐

Figure 13.6 Examples of Change Request Actions Delivered with the Standard System

- **Change request step type**
  A change request step is associated with a step type. A step type determines what actions are available for a change request step. Figure 13.7 and Figure 13.8 show an example of change request step types and available actions for a change request step type, respectively.

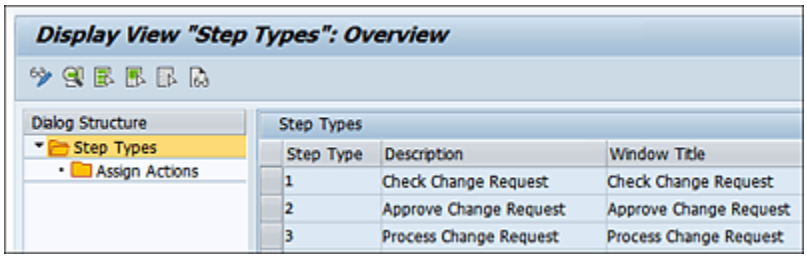


Figure 13.7 Change Request Step Types

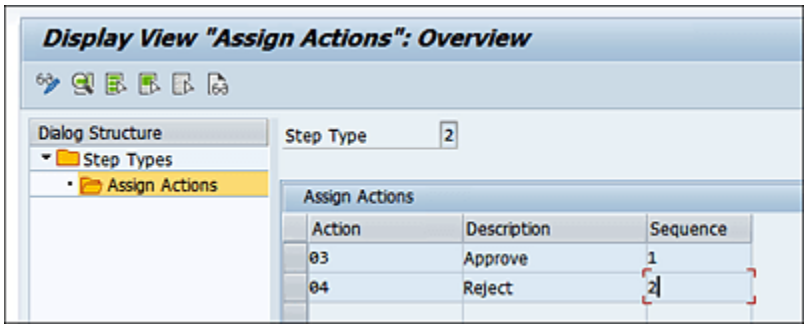


Figure 13.8 Actions Assigned to Change Request Step Type Approve Change Request

- **Workflow template**
  An important property of a change request type is the workflow template that is used to control the change request process. A workflow template determines the steps that are executed, the order in which these steps are executed, the agents assigned to execute these steps, and the status of the change request after each step. Several preconfigured workflow templates are delivered by SAP. We'll discuss these workflow templates in detail in Section 13.2. Additionally, a rule-based workflow template is also delivered in the standard system. This workflow template is configured using a set of decision tables that are then used by the template to control the workflow. We'll discuss rule-based workflow in Section 13.1.10.

13.1.3 Business Object BUS2250

Every change request in SAP MDG uses a workflow definition. Object type BUS2250 (SAP MDG change request) is delivered in the standard system for workflows associated with SAP MDG change requests. Figure 13.9 shows events associated with object BUS2250.

Event type linkage for event CREATED of object BUS2250 must be activated to enable triggering a workflow when a change request is created. The workflow system will use SAP MDG Customizing to determine which workflow must be triggered. Event linkage can be activated using Transaction MDGIMG by following menu path **Master Data Governance, Central Governance • General Settings • Process Modeling • Workflow • Activate Event Type Linkage**. Figure 13.11 shows all the event linkages for object BUS2250.

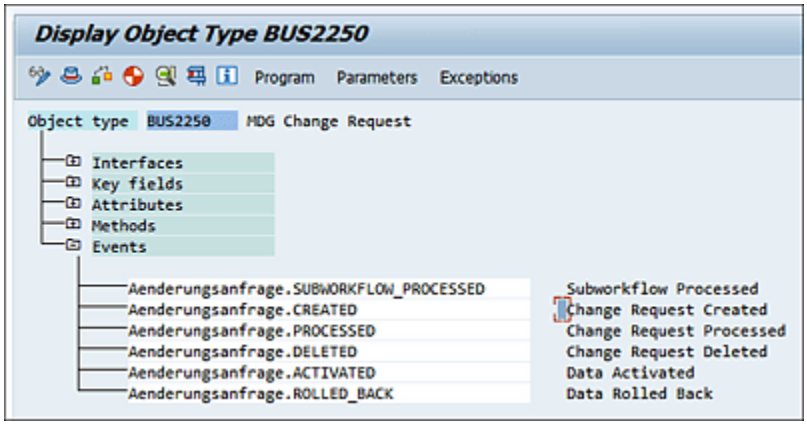


Figure 13.9 Business Object BUS2250 (SAP MDG Change Request)

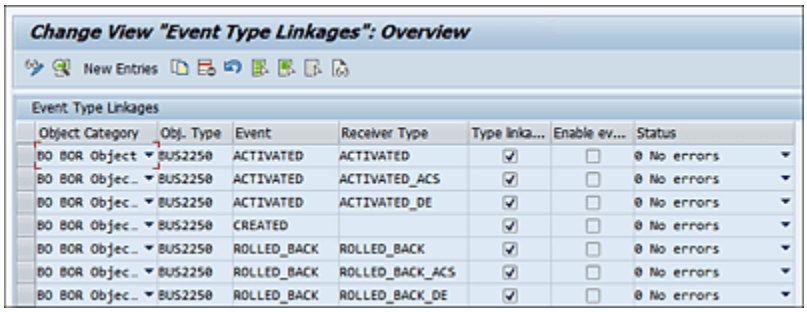


Figure 13.10 Event Type Linkages for Object BUS2250 (SAP MDG Change Request) Part 1

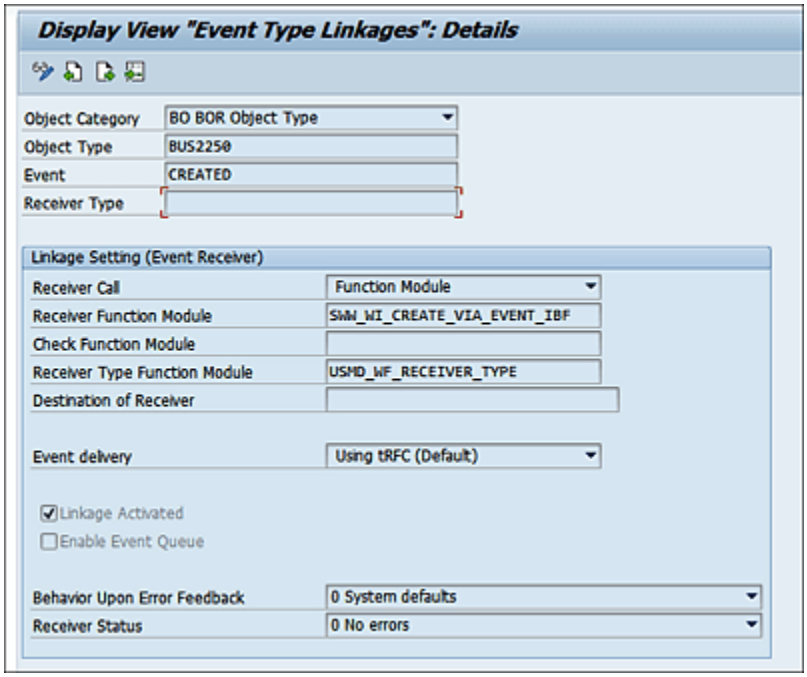


Figure 13.11 Event Type Linkages for Object BUS2250 (SAP MDG Change Request) Part 2

Note

In event type linkage Customizing, don’t assign a receiver type. Receiver type is determined based on the change request type used to create the change request. If a receiver type is specified, the system will always use this workflow template for every change request type.

Similarly, don’t activate event linkage in the header of the workflow template. If a workflow template is started in response to the `CREATED` event of the `BUS2250` object, the system will always start this workflow template for all change request types.

The `CREATED` event has the following parameters:

- **`CR_CREATOR`**
  User ID of the requestor who submitted the change request.
- **`CR_CREATOR_NAME`**
  Name of the requestor who submitted the change request.
- **`CR_WORKFLOW`**
  Workflow template associated with the change request type.
- **`CR_TYPE`**
  Change request type used to create the change request.
- **`CR_EDITION`**
  Edition used to create the change request.
- **`CR_CONTEXT_PARAMETER`**
  A list of context parameters that are available throughout the process.

These parameters are used in the workflow templates for deciding the flow and agents for certain work items, sending notifications of work item creation, and determining work item texts.

13.1.4 Standard Dialog Tasks Used in Workflow Templates

Once a change request is created using a UI application or using an application programming interface (API) call, the system will trigger the workflow associated with the change request type that was used to create the change request. One or more work items will be created when the workflow starts. If the work item created is a dialog work item, it must be assigned to dialog users so that the users will be able to execute the work item. To enable the assignment of users as agents to work items, a dialog task needs to be categorized as a general task. If the task isn't categorized as a general task, and if a processor is assigned to the change request step that isn't assigned as a possible agent for the workflow task, no user will be able to execute the task, and the workflow will end in error. To categorize the workflow tasks as general tasks, call Transaction MDGIMG and follow menu path **Master Data Governance, Central Governance** • **General Settings** • **Process Modeling** • **Workflow** • **Configure Workflow Tasks**. The application component for the SAP MDG application framework is CA-MDG-AF. Click on the **Assign Agent** link, and classify any task that isn't a background task as a **General Task**, as shown in Figure 13.12.

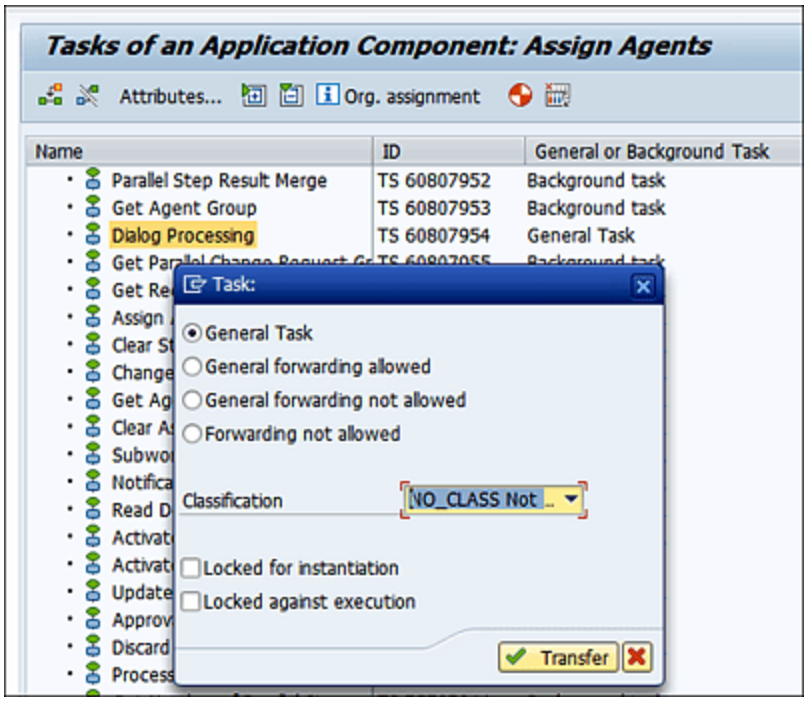


Figure 13.12 Categorizing Workflow Tasks as General Tasks

In addition to categorizing tasks as general tasks in the application framework component (CA-MDG-AF) you must categorize the tasks in the SAP MDG application components for the master data domains that you'll be using. Table 13.1 shows application components for each of the standard data models provided by SAP.

Application Component	Application Component Description
CA-MDG-APP-CA	SAP MDG for Contract Accounting
CA-MDG-APP-FIN	SAP MDG for Financials
CA-MDG-APP-CUS	SAP MDG Customer Central Parts
CA-MDG-APP-MM	SAP MDG Material
CA-MDG-APP-SUP	SAP MDG Supplier (Central Parts)

Application Component	Application Component Description
CA-MDG-APP-BP	SAP MDG Business Partner (Central Parts)

Table 13.1 Application Components Used by SAP MDG

13.1.5 Standard Dialog Tasks

Preconfigured workflow templates delivered by SAP use many dialog tasks for work items that must be executed by users. Examples of such work items are necessary actions on change requests such as enriching change request data provided by the requestor, evaluating changes, reviewing validation errors and warnings, and activating change requests. Dialog tasks are used to launch the change request UI application associated with the change request type and change request step. The system will evaluate the SAP MDG Customizing to determine the Web Dynpro application and application configuration and then launch the application. This will allow the user to process the change request. The following dialog tasks are delivered as standard:

- **Evaluate change request (`TS75707979`)**
  This task is used when the change request is created with only basic attributes such as description and notes, and data specialists haven’t yet added any master data records to the request.
- **Process change request (`TS75707943`)**
  This task is used to process the change request after the

requestor or data specialist has added details of the changes.

- **Approve change request (`TS75707980`)**
  This task is used to approve (or reject) the change request after changes are evaluated and all validation errors are resolved by data specialists.
- **Revise change request (`TS75707981`)**
  This task is used by the requestor to revise the change request after it has been rejected by an approver.

All these standard dialog tasks call method `PROCESS` of business object `BUS2250`. All task containers share the following import parameters:

- **Workflow step number (`APPSTEP`)**
  This parameter holds the value of the change request step (and not the workflow step, as the name would indicate). The system determines several Customizing attributes defined at the change request step level, including UI application and configuration, validations to executed, enrichment spots to be processed, and so on using this parameter.
- **Workflow step type (`STEP_TYPE`)**
  This parameter is used to distinguish between different dialog tasks and is used to determine the work item description.
- **SAP MDG change request (`_Wi_Object_ID`)**
  This parameter holds the value of the change request number and is populated in the task container from the value provided by the `CREATED` event of business object

BUS2250. This parameter indicates which change request should be processed by the task.

13.1.6 Standard Background Tasks

In a typical workflow template for SAP MDG, dialog tasks are used for taking input from users while background tasks are used to process the input received from the user. Background tasks are used to set the status of the change request, perform validation of the changes, update the validation log, activate the change request, write the activation log, run derivation rules and call third-party services to enrich change request data, send custom notifications, determine agents for the next step of the change request, evaluate results of the parallel step, and so on. These background tasks use the parameters received from the workflow container to perform the task and then update the workflow container with the results. The specific import and export parameters used vary depending on the task. Following are some of the most used background tasks in different workflow templates:

- **Set status of change request (TS75707951)**
  This task sets the status of the change request to reflect the result of the previous task. The workflow template will use the change request status to determine the next step and agents for the next step, configure UI properties, and so on.
- **Check change request (TS75707952)**
  This task calls the VALIDATE method of the BUS2250 (SAP MDG change request) business object to run validation

rules designed for the change request type. Results of the validation process are written in the validation log and can be accessed using Transaction SLG1. UI applications used in SAP MDG also allow users to display validation logs. Change request steps can be configured to convert validation errors into warnings to allow users to submit change requests with partial or incorrect data, which can then be enriched by experts. Similarly, change request steps can be configured to ensure that all validation errors are resolved before further processing of the change request is allowed.

- **Path and process finder (`TS60807945`)**
  This task is used by the rule-based workflow template to find the next step in the workflow process and number of parallel paths for the next step. Rule-based workflow template uses a set of decision tables, and this task determines the next steps by reading the decision table records. Section 13.1.10 provides an overview of the rule-based workflow template.
- **Activate change request (`TS60808005`, `TS60808002`)**
  Once a change request is successfully validated and approved, it's activated. During activation of the change request, master data records in the change request are copied from the staging area to the active area and distributed. At the time of creating the change request, the system copies the records from the active area and staging area, and changes are applied to this copy. Task `TS60808005` is designed to perform this activity. At the time of activation of the records, the system compares the snapshot of the record that was copied to the staging area with the current snapshot and, if there is a difference,

raises an error. This will avoid overwriting any changes to the active area records that may have happened outside of the governance process. If changes should be overwritten, task `TS60808002` can be called by the workflow, which will bypass the snapshot and activate the change request.

- **Discard change request (`TS75707936`)**
  Standard workflow templates allow requestors to withdraw the request if the changes aren't necessary or can't be approved. When the requestor decides to withdraw the change request, task `TS75707936` is called to roll back the change request by removing the records in the change process from the staging area without copying them back to the active area. This action also unlocks the records so that these records can be included in the change requests in the future.

13.1.7 Agent Determination

Agent determination or assignment of agents to tasks is an important task in workflows used in SAP MDG. Organizations that implement SAP MDG often have elaborate agent determination rules to ensure that all the required stakeholders are involved in the approval process without slowing down the approval process. Often, these are conflicting objectives because involving more stakeholders will increase the number of approval steps, which will slow down the process. Having an optimum agent determination strategy will ensure that all stakeholders that must be involved in the approval process are part of the process

without causing a significant delay in how long the approval process runs.

Agents are often determined based on the type of data that is undergoing change and the type of change that is being carried out. For example, the approval process for creating new spare parts is often different from the approval process for creating new hazardous chemical materials. Similarly, changing the bank data of a supplier involves more steps than updating the address of a customer. Organizations have data experts that specialize in specific types of business processes and master data used in those processes. Data experts who can provide and validate financial data of customers and suppliers are often different from those who can provide and validate sales data for customers, and yet another set of specialists exists who can provide purchasing data for suppliers. For material master data, different data experts are involved in the approval process depending on the material type or material group. Approvers for finished goods are often different from approvers for service materials. For financial data, there can be different stakeholders for financial accounting data such as general ledger accounts and financial reporting structures that are different from stakeholders for management accounting data such as profit centers and cost centers. In some organizations, a committee often exists that creates and validates changes to all types of financial master data at the group level to ensure uniform financial master data across the organization.

Agent determination rule `AC75700139` (workflow processor of change request) is used to determine the agents. This rule uses parameter `AGENT_FILTER` to select a business add-in

(BAdI) implementation for the agent determination. The workflow templates provide the value of the agent filter attribute. Several standard BAdI implementations are delivered by SAP for different filter values. Filter value `STANDARD` is used to select agents by reading the SAP MDG Customizing tables. Filter values `ERP_MDGS` and `ERP_MDGC` are used to select BAdI implementations that use a Business Rules Framework plus (BRFplus) application to determine the agent. The following sections describe the agent determination used by standard and rule-based workflow templates.

Agent Determination for Standard Workflow Templates

SAP has delivered several workflow templates for different change request types used in SAP MDG. Section 13.2, Section 13.3, and Section 13.4 provide a detailed description of these workflow templates. These workflow templates use a set of rules that determine agents for various work items that are created during the execution of the workflow instance. SAP MDG uses the HR organization structure to determine agents for various work items. Objects in the HR organization structure such as organizational unit, job, or position are used to determine the agents. Users are assigned directly or indirectly to these objects using Transaction PPOCW (Organization and Staffing [Workflow] Create) or Transaction PPOME (Organization and Staffing Change). Staffing assignments for these objects are then used to determine the users that will receive a work item. There are two ways to configure agents for standard workflow templates:

- **Agent determination using SAP MDG Customizing tables**
  SAP MDG provides an easy way to assign agents to workflow steps using Customizing tables that can be accessed via Transaction MDGIMG and menu path **Master Data Governance, Central Governance** • **General Setting** • **Process Modeling** • **Workflow** • **Other MDG Workflows** • **Assign Processor to Change Request Step Numbers (Simple Workflow)**. Figure 13.20 later in the chapter shows an example of agent assignment using SAP MGD Customizing tables.
- **Agent determination using the BRFplus application**
  For implementing advanced agent determination rules, BRFplus applications are used. Agent determination for workflow templates used for governance of customers and suppliers use BRFplus applications. SAP has delivered BRFplus application `MDG_BS_ECC_SUPPLIER_WORKFLOW` for supplier master data and `MDG_BS_ECC_CUSTOMER_WORKFLOW` for customer master data. Both these BRFplus applications have a function named `AGENT_DETEMINATION`, which is called by the agent determination task of the workflow template. This function uses decision table `GET_AGENT` in the application to determine the agents. A decision table allows multiple columns as search columns and two columns as output columns (for object type and object ID). The search columns are used to search for a record that closely matches the parameters from the workflow instance, and the output columns provide the objects (from the organization structure) that are then used to resolve the valid agents for the work item. Because decision tables provide pattern matching and other

features, a complex agent determination strategy can be built using these applications. Refer to Section 13.2.3 for an example of how the BRFplus application is used to assign agents to a workflow template.

Agent Determination for Rule-Based Workflow Templates

In addition to standard workflow templates, SAP has also delivered a rule-based workflow template (`WS60800086`). Section 13.1.10 provides a detailed description of the rule-based workflow template. This workflow template uses a set of decision tables that determine the number of steps and agents that are assigned to the work items created for these workflow steps. Agents can be assigned to the workflow step using the HR organization structure objects such as organizational unit, job, or position. Additionally, security roles can be used to determine agents for specific work items. Users that are assigned to these organizational objects or security roles are then determined as agents. A special type is also supported by the rule-based workflow template, which allows either the workflow initiator or agent that executed the previous step to be determined as an agent for the current step. This is often helpful when a change request workflow must be routed back to the previous user or all the way back to the requestor due to rejections.

13.1.8 Workflow Container Used by Workflow Templates

All the workflow templates in SAP MDG use a set of attributes as part of the workflow container. These attributes are initially populated by the `CREATED` event of the `BUS2250` object when the workflow is triggered and later by the execution of the workflow tasks. These attributes allow the correct routing of various steps based on the result of the previous step, determine the agents for various steps, and set correct descriptions of various work items. Figure 13.13 shows typical attributes of a workflow container used by SAP MDG workflow templates.

The following are some of the important attributes of the workflow container.

- **`CHANGE_REQUEST`**
  This is the `BUS2250` object that triggered the workflow. This object stores all the information related to the change request such as change request type, ID and name of the creator, change request number, and any context parameters.
- **`BRANCHES`**
  This multiline attribute holds all the individual `BUS2250` objects created for each branch of a parallel workflow.
- **`PROCESSOR`**
  This is the current processor of the work item.
- **`ACTION_RESULT`**
  This is the result of the previous step. Workflow templates often use the results of the previous step to determine the next step as well as the agents for the next step.
- **`AGENT_FILTER`**
  This filter value is used by the agent determination rule

AC75700139.

- **ACTIVATION_ERROR**
  This flag indicates if an error occurred during the change request action.

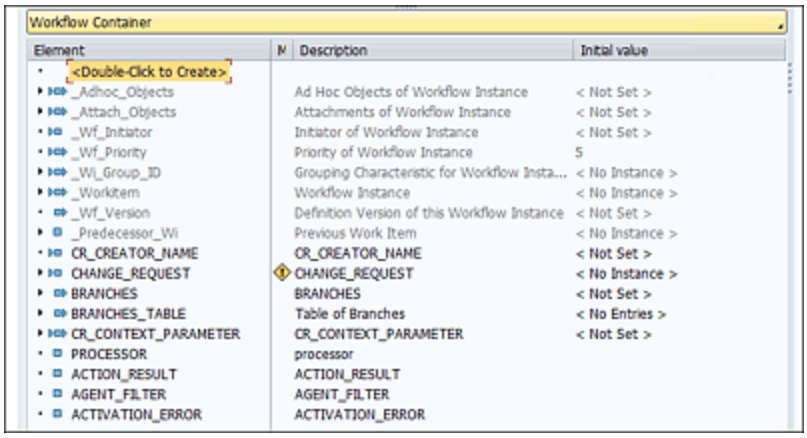


Figure 13.13 Typical Workflow Container Used by Standard Workflow Templates in SAP MDG

13.1.9 Workflow Log for Change Requests

During the execution of the change request workflow, several dialog and background tasks are executed. For dialog tasks, a user must take an action for the workflow process to continue to the next step. Many organizations require that an audit trail or log of these actions must be maintained so that it can be reviewed later to determine the users who are involved in the change request process, the actions they took, and the date and time the action was taken. For certain industries, there are legal and regulatory requirements that such an audit trial be maintained for critical master data change requests. In addition to legal or regulatory requirements, many organizations also have certain service-level agreements for some type of master

data management tasks, and they need to review the workflow log to ensure that change requests and steps within the change request process are executed within the given time frame. For workflows that aren’t yet complete, administrators need to know the status of the workflow and who are the next people that need to act for the process to move forward. The workflow system can also send automatic reminders when certain work items aren’t executed within a given time frame.

Figure 13.14 shows the workflow log for a change request in SAP MDG. The workflow log contains the work item ID, work item description, processor who processed the work item (for completed work items) or list of processors (who are assigned as the agent) for work items that are ready to be processed, work item status, date and time the work item was created and processed, and action taken for completed work items. Background tasks are also included in the work item log.

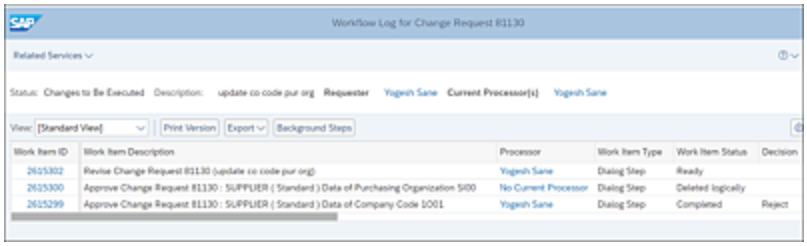


Figure 13.14 Workflow Log for an SAP MDG Change Request

13.1.10 Rule-Based Workflow Template

SAP delivers several workflow templates (discussed in detail in Section 13.2, Section 13.3, and Section 13.4) for the most-used governance process patterns in different

industries for different types of master data objects. If an organization's requirement for the governance process is only marginally different from the process pattern in the workflow templates delivered by SAP, it's often easier to make a copy of the standard template and adjust. However, if the process requirements differ by a large degree, it may not be possible to adjust the standard templates. For such cases, SAP has delivered a rule-based workflow template (`WS60800086`) that uses a set of decision tables to decide the number of steps, agents for these steps, and control of the flow. By configuring these decision tables, it's possible to model everything from a simple two-step sequential approval process to a multistep process with parallel steps, exceptional approvals, and dynamic branches. The flexibility offered by the rule-based workflow template makes it suitable in cases where requirements are often simple initially but evolve into more complex requirements over time. Figure 13.15 shows examples of process patterns that can be built using the rule-based workflow.

For each change request type that uses the rule-based workflow, the system generates a BRFplus application with an application that contains the change request type. The generated application contains all the required functions, rulesets, data elements, and structures required for the application. Workflow template `WS60800086` isn't designed specifically for a data model or a change request. It contains generic tasks that read the information stored in the generated application to decide the steps, agents, and control of the process flow. The system-generated BRFplus application can be accessed by calling Transaction MDGIMG and menu path **Master Data Governance, Central**

Governance • **General Settings** • **Process Modelling** • **Workflow** • **Configure Rule Based Workflow**. BRFplus applications can also be accessed using Transaction USMD_SSW_RULE.

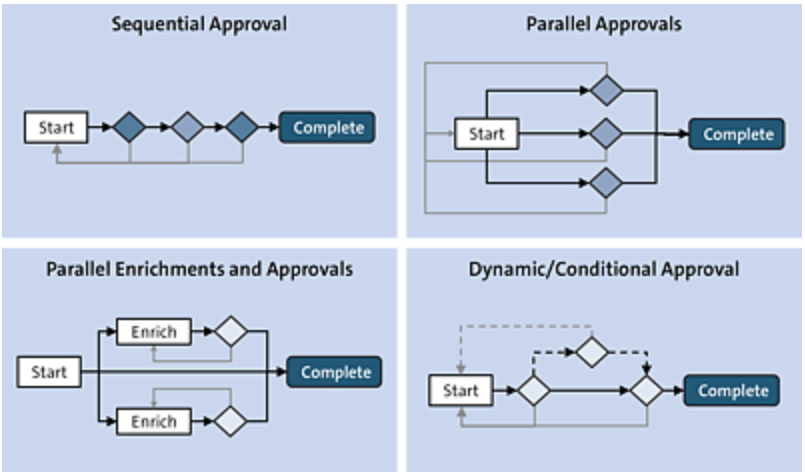


Figure 13.15 Examples of Process Patterns Built Using the Rule-Based Workflow Template

Workflow Template WS60800086

Workflow template `WS60800086` contains tasks that use the system-generated BRFplus application and the decision tables configured in the application to determine the next step and agents for the next step primarily using the previous step, result of the previous step, and other parameters. The workflow template isn't designed for any specific data model, but the data model–specific data is read from the generated BRFplus application. Figure 13.16 shows the process diagram for the rule-based workflow template.

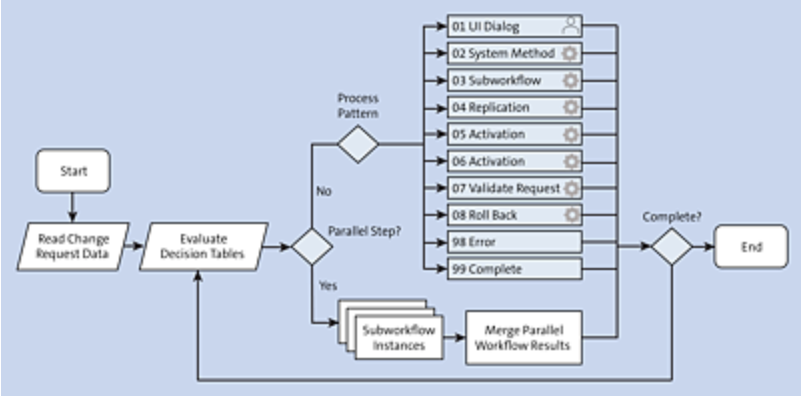


Figure 13.16 Process Diagram for Rule-Based Workflow Template WS60800086

The workflow is started when a change request is created. The workflow template doesn’t have any event linkage, but the workflow is started when event `CREATED` of business object `BUS225` is triggered. The system reads the change request type and related attributes from the event object and SAP MDG Customizing tables and then transfers the data to the workflow container. The system evaluates the single-value decision table based on the previous step and previous action along with other parameters to determine the condition alias for the next step. The user agent and nonuser agent tables are then searched for the condition alias to determine if the next step is a dialog step or a background step. The system also evaluates if parallel processing is enabled for the next step.

Parallel processing for a step can be enabled by configuring multiple agent groups for the condition alias for the step. In this case, one subworkflow is created for each group. Each of these subworkflows is an instance of the rule-based workflow template using the same generic template, that is, the same set of decision tables, and other data is used for the subworkflows and the parent workflow. Once all

subworkflows are completed, the system calls BAdI `USMD_SSW_PARA_RESULT_HANDLER` to merge the results of the individual results into a single result, which will then be evaluated to decide the next step.

If parallel processing isn’t enabled for the next step, the system determines the process pattern to be used and calls the associated task. The service name configured in the nonuser decision table is used to select a specific task or populate attributes of the task container. If the process pattern isn’t 99 (which is used to exit the loop), the system loops back to the evaluate decision table to determine the next step in the process. After execution of the process pattern, the system updates the change request status, date and time of the execution, processor, and result of the task into the workflow container. These attributes combined with the decision table entries are used to evaluate the next step in the workflow.

Because the workflow template itself doesn’t have any steps specific to a data model or type of change, and the steps are determined by the decision table configuration, this template allows a highly flexible way to design governance processes. Agent determination supported by the rule-based workflow template includes security roles and special users (change request initiator and last processor), which aren’t supported by the preconfigured workflow templates. The rule-based workflow template can also be extended by a set of BAdIs to add functionalities that can’t be implemented by configuring the decision tables. The upcoming section “Extending the Rule-Based Workflow Template with Business Add-Ins” provides details of the various BAdIs available.

Process Patterns Used in the Rule-Based Workflow Template

The loop used in workflow template `WS60800086` executes one process pattern in each iteration. Selection of the process pattern depends on the condition alias (described later in this section) determined for the current step. Depending on the process pattern, the system will call the corresponding task in the workflow template. Following are the standard process patterns and their functions:

- **01 UI Dialog**
  This process pattern is used to start dialog processing task `TS60807954`. This process pattern can only be selected by the system (and configured manually in the nonuser decision table) when a record matching the condition alias is found in the user agent decision table.
- **02 Call Synchronous Method**
  This process pattern is used to call a BAdI implementation to implement a custom processing logic when none of the standard process patterns provide the required functionality. Task `TS60807949` (system method) is called to trigger the BAdI implementation. The filter value to search for the BAdI implementation is read from column `SERVICE_NAME` of the nonuser decision table. Refer to the upcoming section “Extending the Rule-Based Workflow Template with Business Add-Ins” for details.
- **03 Call Subworkflow**
  This process pattern is used to trigger a subworkflow using the standard task `TS60807944` (subworkflow for single step workflow) to include processing steps defined in an existing workflow as a part of the current workflow. The

subworkflow ID is read from column SERVICE_NAME in the nonuser decision table.

- **04 Call Data Replication**
  This process pattern is used to initiate replication of master data in the change request process after the change request is activated by calling standard task TS60807976 (change request replication). The system uses the data replication framework to distribute the change request.
- **05 Activation (Don’t Bypass Snapshot)**
  This process pattern is used to activate the change request after final approval by calling standard task TS60808002 (activate change request) with an empty value for parameter IGNORE_SNAPSHOT_DIFF. If a parallel change was made to the data record in the active area, the system will detect the difference in the snapshot taken at the time of creation of the change request and at activation, and activation will fail to prevent overwriting the changes in the active area that aren’t available in the staging area.
- **06 Activation (Bypass Snapshot)**
  This process pattern is used to activate change request after final approval by calling standard task TS6080802 (activate change request) with parameter IGNORE_SNAPSHOT_DIFF set to X. If there are parallel changes made to the same master data record in the active area, these will be ignored and may be overwritten.
- **07 Validate Change Request**
  This process pattern is used to trigger validation of the change request in the background and write the

validation log to the application log. Logs are accessible from the UI application. Standard background task `TS75707952` (check change request) is used with this process pattern.

- **08 Roll Back Change Request**
  If a change request is rejected and needs to be withdrawn, this process pattern is called to remove the master data records in the change request from the staging area without copying them to the active area. The status of the change request will be set to **06** (final check reject rejected) to indicate the end of processing of the change request. This process pattern uses standard background task `TS75707936` (discard change request).
- **98 Error**
  This process pattern is used to trigger error handling by calling standard background task `TS60807951` (default handler).
- **99 Complete (Sub)Workflow**
  This process pattern is used to exit the loop processing in the rule-based workflow template for both the main workflow and the subworkflow. For the subworkflow, the control will pass to the main workflow. Flag `Workflow Instance Complete` in the workflow container will be set by this task.

Decision Tables Used in BRFplus Applications for the Rule-Based Workflow

In addition to the system-generated functions and data structures that reflect the data model used by the change

request type, a set of decision tables are generated, and these tables are configured to model the required process pattern. Each decision table has a set of input columns (also known as condition columns) and output columns (also known as result columns). Input columns are used to search for a record based on the data attributes read from the context of the change request and workflow, and the output columns determine the result of the process. The three decision tables generated for each BRFplus application are used together to determine the output, which is then used by the workflow template to determine the next task, agents, and container values. Each generated BRFplus application contains the following three decision tables:

- **Single-value decision table**
  The single-value decision table generated by the system has the name `DT_SINGLE_VAL_<Change Reuqest Type>` and determines the flow between the change request steps, including the parallel steps. This table reflects the process flow as it determines the next step in the workflow process based on the previous step, previous action, and other parameters. The most important condition columns used in this table are previous step and previous action. Based on the values for these two columns and other parameters, this table determines the next step and other parameters, especially the condition alias that is used as a condition column to search the other two tables. Table 13.2 provides details of the single-value decision table columns.

Column	Type	Details

Column	Type	Details
PREVIOUS_STEP	Condition column	Previous change request step.
PREVIOUS_ACTION	Condition column	Result of the previous step (both dialog and background step).
CR_PRIORITY	Condition column	Current priority of the change request.
CR_REASON	Condition column	ID of the change request reason.
CR_REASON_REJ	Condition column	ID the change request rejection reason.
PARENT_STEP	Condition column	Previous change request step from which a parallel workflow is initiated; used in combination with PAR_AGT_GRP_NUM.
PAR_AGT_GRP_NUM	Condition column	Agent group number of subworkflow generated from the parent workflow. Each subworkflow is another instance of the rule-based workflow using the same template and BRFplus application.

Column	Type	Details
COND_ALIAS	Result column	A key used to search the condition columns of the other two decision tables. This allows linking the two tables with this table.
NEW_STEP	Result column	Next step in the workflow.
NEW CR_STATUS	Result column	New change request status.
EXP_COMPLETE-_HOURS	Result column	Expected hours in which the next step should be completed. If not, a system-generated email notification is sent.
MERGE_TYPE	Result column	Determines how the results of the parallel steps should be combined to determine the single result value. Only value B is supported, which calls a BAdI implementation using the filter value stored in column MERGE_PARAM.
MERGE_PARAM	Result column	Filter value to select a BAdI implementation to merge results of parallel workflow steps.

Column	Type	Details
DYNAMIC_AGT_SEL-_SERVICE	Result column	Used as a filter value to determine agents for the next step using a BAdI implementation instead of using the user agent decision table.

Table 13.2 Columns in the Single-Value Decision Table

- **User agent decision table**
  The user agent decision table generated by the system has the name DT_USER_AGT_GRP_<Change Reuqest Type> and determines the user agents for a condition alias and which actions are available on the UI for the next step in the process. This table has only one condition column (COND_ALIAS) and it's used to link the single-value decision table with this table. Table 13.3 describes the columns of the user agent decision table.

Column	Type	Details
COND_ALIAS	Condition column	Links the row in this table with the single-value decision table having the same value. This column is a result in column in single-value decision table.
STEP_TYPE	Result column	Step type for the next step. Step type determines which actions/buttons are available on the UI.

Column	Type	Details
USER_TYPE	Result column	Type of the user agent specified in column USER_VALUE. Possible values are **User**, **Organizational Unit**, **Job**, **Position**, **Role**, or **Special User**.
USER_VALUE	Result column	Values of type identified in USER_TYPE. This can be a single value or a list. Special user values such as INIT (initiator of the change request) or LAST (processor of the last step) can be maintained.

Table 13.3 Columns in the User Agent Decision Table

- **Nonuser agent decision table**
  The nonuser agent decision table generated by the system has the name DT_USER_AGT_GRP_<Change Reuqest Type> and determines background steps associated with a condition alias. This table has only one condition column (COND_ALIAS), and it's used to link the single-value decision table with this table. Table 13.4 describes the columns of the user agent decision table.

Column	Type	Details

Column	Type	Details
COND_ALIAS	Condition column	Links the row in this table with the single-value decision table having the same value. This column is a result column in the single-value decision table.
AGENT_GROUP	Result column	Used to identify a branch in parallel workflow processing. One subworkflow is created for each agent group.
PROCESS_PATTERN	Result column	Process pattern listed in the “Process Patterns Used in the Rule-Based Workflow” section.
SERVICE_NAME	Result column	Value depends on the process pattern. For example, for the call system method process pattern, this value is used as a filter to search for BAdI implementations.

Table 13.4 Columns in the Nonuser Agent Decision Table

Designing a Rule-Based Workflow

A rule-based workflow template makes extensive use of the decision tables to control the flow of the process by determining the next step based on the previous step and result of the previous step. Designing a governance process

largely involves populating the decision tables with the correct records so that the resulting process accurately reflects the desired pattern. Decision tables can be prepared using a spreadsheet and uploaded, but populating the decision tables with correct data, especially correctly linking the three decision tables with each other, requires careful planning. The process typically starts with drawing the outline of the process and then populating the outline with the attributes, such as step number, action result, change request status, condition alias, process pattern, and so on, so that these values can then be easily translated into a set of records. Figure 13.17 shows the outline of a process implemented using the rule-based workflow.

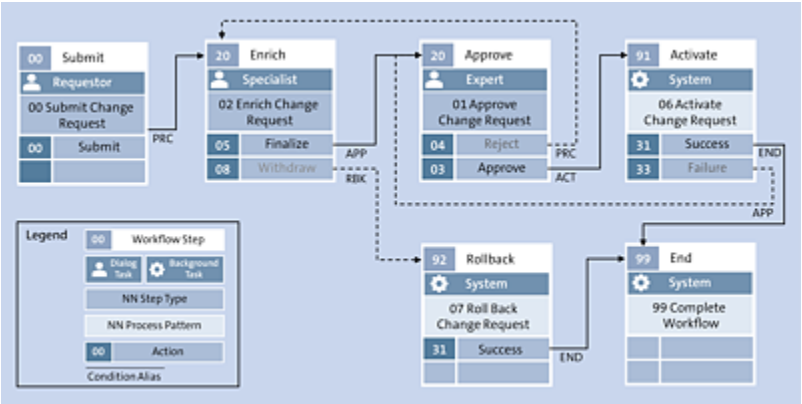


Figure 13.17 Outline of a Process Implemented Using the Rule-Based Workflow

Figure 13.18 shows the decision tables populated using the data gathered from this outline diagram.

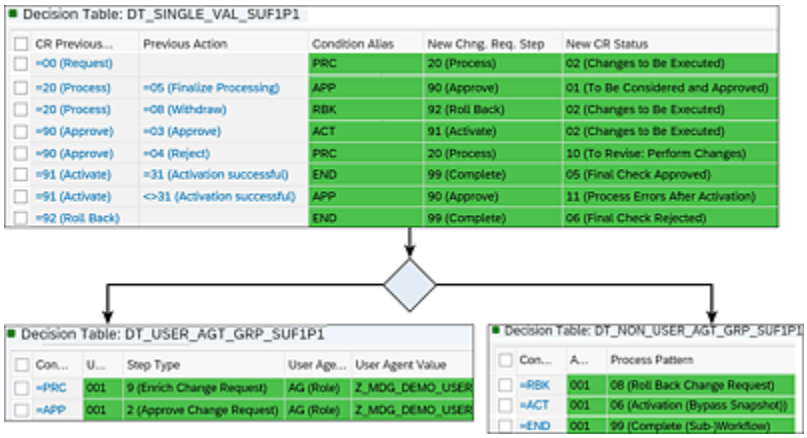


Figure 13.18 Decision Table Entries for the Preceding Process

The change request type used in this process is `SUF1P1`. Figure 13.18 shows the single-value decision table (`DT_SINGLE_VAL_SUF1P1`), user agent decision table (`DT_USER_AGT_GRP_SUF1P1`), and nonuser decision table (`DT_USER_AGT_GRP_SUF1P1`) generated by the system for this change request type. The condition columns are shown with a gray background, and the output columns are shown with a green background. You'll observe that the **Condition Alias** column is a result column in the single-value decision table, but it's the only condition column in the other two tables. An important observation is that the values populated in the condition alias column in the single-value decision table are provisioned in either the user agent decision table or the nonuser agent decision table. Similarly, all possible results of all the steps are provisioned in the single-value decision table. This is necessary to ensure that the rule-based workflow template will always be able to find the next step based on the result of the previous step.

The process outlined in Figure 13.17 is a simple three-step approval workflow process that starts when a requestor submits a change request. Submission of a change request

is represented as a dialog task, and the step number for this step is always defaulted to 00. The role assigned to this task is requestor, and there is only one action possible—to submit the change request. The requestor can also save the request in draft status and submit later, but the workflow doesn’t start until the change request is submitted.

The next step in the flow is to process or enrich the change request by a data specialist. The step number for this step is **20**, and the name of the step is **Enrich**. The output column condition alias is set to **PRC**, and the next step is **20**. Because this is a dialog step, this condition alias is provisioned in the user agent decision table. Because this isn’t a parallel step, only one record with condition alias equal to **PRC** is necessary, and user agent group is set to **001**. The step type for this step is **9** because this step type allows the specialist to either finalize the processing or withdraw the change request. The agent type for this step is security role, and the security role used is `Z_MDG_DEMO_USER`. At runtime, users assigned to this role will receive a work item to enrich or process the change request.

Once the processing is finalized by the specialist, the change request is sent to an approver. The step for this task is **90**, the step name is **Approve**, and the condition alias is **APP**. Because this step is triggered only when the result of the previous step **20** is **05** (Finalize Processing), a row is inserted into the single-value decision table with condition column **CR Previous Step** populated as **00**, and condition column **Previous Action** populated as **05**. The output column condition alias is set to **PRC**, and the change request status is set to **02**. This status doesn’t allow changes to the change request data records because an

approver can only approve or reject the changes requested but not make further changes. This ensures that all changes are always reviewed by at least two agents. Because this is a dialog step, a record with condition alias **PRC** is configured in the user agent decision table and the step type is set to **02** (**Enrich Change Request**). This step type allows the approver two actions: **03** (**Approve**) and **04** (**Reject**). The agent assignment for this step is like the **Enrich** step.

After final approval of the change request by the approver, the change request should be activated. This is configured with one row in the single-user decision table with the previous step as **90** (**Approve**) and previous action as **03** (**Approve**). The condition alias is **ACT**, and the next step number is **91** (**Activate**). Because this is a background step, a record is configured in the nonuser agent decision table with condition alias **ACT**. The process pattern for activation is **06** (Activation [Bypass Snapshot]).

Upon successful activation of the change request, the system will set the result of the activation step as **31**. An entry is configured in the single-value decision table with the previous step as **91** (**Activate**) and the previous action as **31** (**Success**) with condition alias **END**. Because this step is a background step, an entry is configured in the nonuser decision table with condition alias **END** and process pattern **99** (**Complete Workflow**). This process pattern will end the rule-based workflow process.

At the processing step, the processor has the option to withdraw the changes if the requested changes should not be carried out. If the processor chooses to withdraw the

change request (action = **08**), changes should be rolled back. This a background step, and the process pattern is 08 (Roll Back Change Request). Similarly, at the approval step, the approver has the option to reject the change request, and it will be routed back to the specialist. An entry in single-value decision table with the previous step as **90** (**Approve**) and the previous action **04** (**Reject**) is configured for this routing. The condition alias for this step is **PRC**, which is already configured in the user agent decision table.

Extending the Rule-Based Workflow Template with Business Add-Ins

Several BAdIs are available to extend the functionality of the rule-based workflow template. Following is a list of available BAdIs:

- **Rule Context Preparation for Rule-Based Workflow (`USMD_SSW_RULE_CONTEXT_ PREPARE`)**
  This BAdI is used to add additional columns to the decision tables in the system-generated BRFplus application for the change request type and to populate these columns at runtime using the attributes of the change request and workflow context. Additional columns, especially when used as condition columns allow the designer to build a process flow that depends on the values of these columns at runtime. For example, account group can be added as a condition column in the single-value decision table for customer and supplier master data so that routing of the workflow can be based on the account group of the customer or supplier.

- **Calling of System Method for Rule-Based Workflow (`USMD_SSW_SYSTEM_METHOD_CALLER`)**
  This BAdI is used to call code to process a background task. The system provides several process patterns to execute common tasks, but if these tasks aren’t sufficient to meet the requirements, this BAdI allows calling a method of a class that implements the BAdI and passes the change request data. The custom code can then use the change request attributes and workflow process attributes to execute the required action and pass back the result of the step. This result will then be used to determine the next step in the workflow process.
- **Dynamic Selection of Agent in the Rule-Based Workflow (`USMD_SSW_DYNAMIC_AGENT_SELECT`)**
  This BAdI allows agent determination using custom code instead of using the user or nonuser decision table. It can also be used to enable parallel processing for a step at runtime based on the attributes of the change request. This BAdI is used if the agent determination logic is implemented outside of the decision tables or the decision table functionality isn’t sufficient to meet all the requirements.
- **Handling of Parallel Results in the Rule-Based Workflow (`USMD_SSW_PARA_RESULT_HANDLER`)**
  This BAdI is used to merge results of all subworkflows that are started for a parallel step into a single result. The system calls this BAdI implementation when all the subworkflows are completed with a list of results of all individual subworkflows. A typical implementation of this BAdI is used to merge results of parallel approval steps such that if all approvers have approved the previous

step, the result is set as **Approved**; otherwise, the result of the step will be set as **Rejected**.

- **Check User Agent/Nonuser Agent Table for the Rule-Based Workflow (`USMD_SSW_CHECK_AGENT_TABLE`)**
  This BAdI is used to implement dynamic checks in the user agent or nonuser agent decision tables.

13.1.11 SAP Master Data Governance, Consolidation and Mass Processing

SAP MDG supports consolidation of data from multiple sources into a single source of data by combining and comparing data from various sources into a single best record. SAP MDG, consolidation and mass processing, also supports updating several master data records together in a single process. Both consolidation and mass processing use a process template that allows an administrator to configure various steps in the process and configure options for each of the process steps. A typical SAP MDG, consolidation and mass processing, process includes data loading, editing, validating, and activating steps. A workflow template is used to determine the user agents for these steps. Using the workflow template allows the administrator to enforce the four-eye principle for a process template. This ensures that at least two different people are involved in consolidation and mass processing of master data records using the consolidation and mass processing module of SAP MDG. SAP delivers workflow template `WS54500001` for SAP MDG, consolidation and mass processing. Figure 13.19 shows process templates delivered in the standard system

and assignment of the workflow template to the process template.

Figure 13.19 Process Templates Used in SAP MDG, Consolidation and Mass Processing

SAP MDG, consolidation and mass processing, uses business object `BUS2240`. Use Transaction SWE2 to activate event linkage for the `STARTED` event of this object. This will allow the system to start the workflow template assigned to the consolidation process template when a consolidation process is created. Figure 13.20 shows the event linkage for event **STARTED** of business object **BUS2240**.

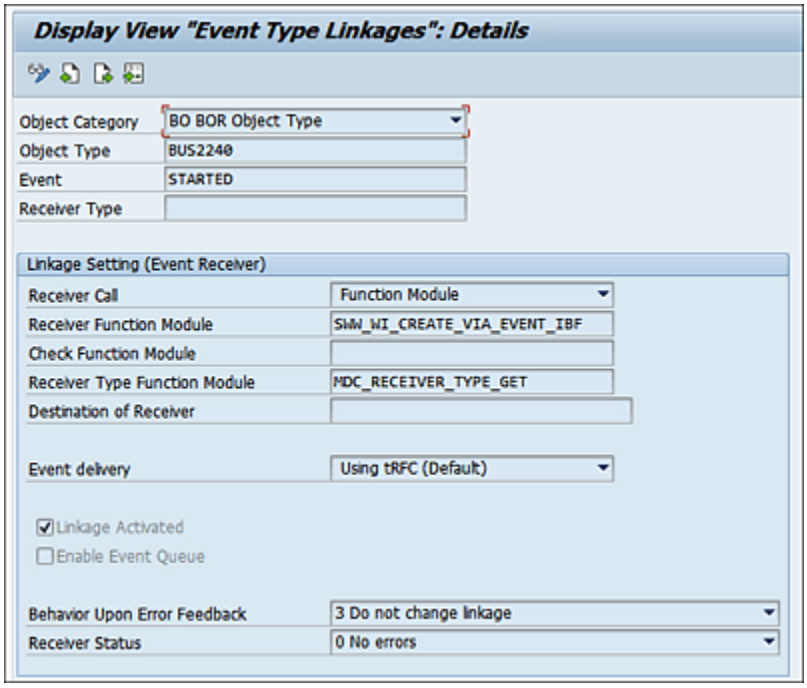


Figure 13.20 Event Type Linkage for the STARTED Event of Business Object BUS2240

13.2 Business Partner Workflows

In SAP, a business partner is an organization, person, or group of people or organizations in which your company has a business interest. A business partner can be a customer, supplier, bank, government entity, and so on. A business partner can take multiple roles as well. It's common to have a business partner as both a customer and a supplier. Business partner master data is used in sales for processing sales orders, in purchasing for processing purchasing orders with suppliers, in accounting for managing account receivables and payables, and in human resources for modeling employees as suppliers. Business partner data is also used in credit analysis, background checks, and risk and compliance management. These are just a few examples of how a business partner is used in SAP. Having a single repository of business partners is essential to ensure that all of these processes use the same set of attributes and that errors arising out of mismatches in data records are minimized.

13.2.1 Business Partner Data Model and Approach

A business partner data model consists of general data, customer data, and supplier data. The general data of a business partner consists of name and address data, tax numbers, ID numbers, roles, payment cards, credit profile and credit segment data, and bank accounts. Customer data of a business partner consists of customer general data, sales data, and company code data. Supplier data of a business partner consists of supplier general data, purchasing organization data, and company code data. Depending on the roles assigned to a business partner and business processes implemented in SAP, various attributes of the business partner master data are populated. For example, if a business partner is a customer and sales and distribution business processes are implemented in SAP, customer general data and sales data will be populated along with business partner general data. If accounts receivable processes are implemented in SAP, company code data for the customer role will also be populated. Figure 13.21 provides an overview of business partner data model in SAP MDG.

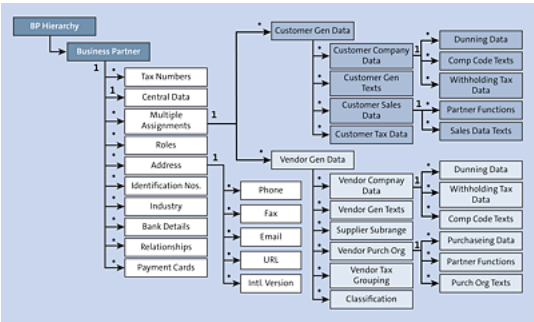


Figure 13.21 Business Partner Data Model Used in SAP MDG

In SAP S/4HANA, a business partner approach is mandatory, that is, all customers and suppliers must be created as business partners, and changes to the customer and supplier master data can only be done by updating the business partner master data and synchronizing the business partner master data with customer and supplier master data

using the customer/vendor integration (CVI) functionality. In SAP ERP 6.0 and earlier versions, the business partner approach is available, but it's optional. Customer and supplier data can be created and managed without first creating a business partner. SAP MDG uses business partners as the main object for updating business partners as well as customer and supplier master data. SAP MDG always uses the business partner approach for governance of business partner master data.

13.2.2 Change Requests for Business Partner Master Data

SAP MDG provides three sets of change requests for governance of business partner master data: change requests for business partner data, change requests for customer data and change requests for supplier data. Depending on how the business partner is used in the organization, one or more of these change requests can be used to manage business partner master data. Table 13.5 provides a list of change request types and linked workflow templates for the business partner data model.

Type of Change Request	Description	Workflow
BP1P1	Create Business Partner	WS60800086
BP1P2	Create Bus. Partner w. Hry. Assignment	WS60800086
BP2P1	Process Business Partner	WS60800086
BP2P2	Process Bus. Partner w. Hry. Assignment	WS60800086
BP5P1	Block/Unblock Business Partner	WS60800086
BP6P1	Mark Business Partner for Deletion	WS60800086
BPCC1	Process Business Partner Cleansing Case	WS60800086
BPHP1	Process Business Partner Hierarchies	WS60800095
BPLP1	Business Partner Initial Load	WS72100006
BPMP1	Business Partner Mass Maintenance	WS60800095
CUST1P2	Create Customer (Parallel WF)	WS54400001
CUST1P3	Create Customer w. Hierarchy Assignment	WS54300003
CUST2P1	Process Customer	WS54300003
CUST2P2	Process Customer w. Hierarchy Assignment	WS54300003
CUST5P1	Block/Unblock Customer	WS54300003
CUST6P1	Mark Customer for Deletion	WS54300003
CUSTMRP1	Multi-Processing for Customer Financials	WS54300003
CUSTMRP2	Multi-Processing for Customer Sales	WS54300003
SUPPL1P1	Create Supplier	WS54300005
SUPPL1P2	Create Supplier w. Hierarchy Assignment	WS54300003

Type of Change Request	Description	Workflow
SUPPL2P1	Process Supplier	WS54300007
SUPPL2P2	Process Supplier w. Hierarchy Assignment	WS54300003
SUPPL5P1	Block/Unblock Supplier	WS60800059
SUPPL6P1	Mark Supplier for Deletion	WS60800068
SUPPMRP1	Multi-Processing for Supplier Financials	WS54300003
SUPPMRP2	Multi-Processing for Supplier Purchasing	WS54300003

Table 13.5 List of Change Request Types Used in Business Partners and the Associated Workflow Template

Note

Change request types for business partners can also be used for governing changes to customers and suppliers.

13.2.3 Workflow Templates Used in Business Partner Change Requests

SAP has provided some preconfigured workflow templates based on the industry best practices for governance of business partner master data records. For example, change request type CUST2P1 (process customer) implements typical governance processes used in management of customer master data. These workflow templates have a preconfigured set of tasks that are executed in a specific order depending on the type of data that is undergoing changes and action taken by users who are processing the change requests. After each task is executed, the template will determine the next step to be executed and the status of the change request based on the action taken in the previous step. A key aspect of these templates is that they allow users to be assigned using Customizing tasks in Transaction MDGIMG. In some cases, a separate Web Dynpro application is provided to configure a complex agent assignment scenario.

In addition to preconfigured workflow templates, a rule-based workflow template is used for some change request types. This is a generic workflow template that uses a set of decision tables to control the creation of work items and users that are assigned to execute these work items. In the following sections, we'll discuss each of these templates in detail.

Note

Multiple workflow templates may have similar steps and process flows. This is because workflow templates are added when business functions for different SAP MDG domains are activated and related Business Configuration Sets (BC Sets) are imported. If BC Sets for business partners, customers, and suppliers are imported, multiple workflow templates may be present in the system with similar configurations.

Workflow Template WS60800086

Workflow template WS60800086 is the rule-based workflow template used by change request types BP1P1, BP1P2, BP2P1, BP2P2, BP5P1, BP6P1, BPCC1, and BPCC2. Refer to Section 13.1.10 for a detailed description of the rule-based workflow template.

Workflow Templates WS60800095 and WS72100006

Workflow template WS60800095 is used by change request types BPHP1, BPHP2, BPMP1, and BPMP2. These change request types are used to maintain business partner hierarchies and mass maintenance of business partners.

This is a simple three-step sequential approval process workflow. The workflow starts when a requestor submits a change request. The system performs validation of data in the background, and the results are written to a validation log. The change request is then routed to a data specialist who performs manual validations. If any further changes are required, the request is sent back to the requestor. If all changes are correct, the request is approved and sent to an approver. The approver then performs additional checks and can either approve or reject the request. If the request is rejected at any point, it's sent back to the requestor who can then make further changes and resubmit the request or withdraw the request if the changes can't be approved.

Once the changes are finally approved, the system will activate the changes. If there are any errors during the activation, the request is routed to a data steward who can decide if the change request should be activated by bypassing the snapshot or should be sent for revision. If the change request could be activated by bypassing snapshot, the system will write change documents to record the changes. If the change request could not be activated even after bypassing the snapshot, the system will route the request to the data steward who can then send it back to the requestor with a note explaining what additional changes should be made before the change request can be activated. If the change request is withdrawn by the requestor, changes are rolled back, and no change documents are written. Through the process, the system will maintain an audit trail of workflow process steps. Figure 13.22 shows the process diagram for workflow template WS60800095.

Agent determination for workflow templates WS60800095 and WS72100006 is determined using a BRFplus table. Agents are determined based on a combination of change request type, change request step, and a flag indicating if the central data of the business partner was changed. SAP has delivered BRFplus application MDG_BS_ECC_SUPPLIER_WORKFLOW, which has function AGENT_DETEMINATION that is called by the agent determination task of the workflow template. This function uses the GET_AGENT decision table in the application to determine the agents.

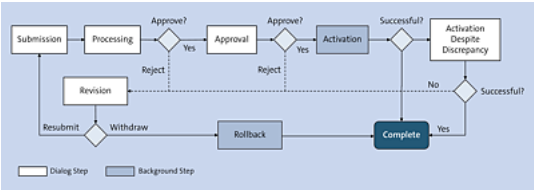


Figure 13.22 Workflow Process Diagram for WS60800095 and WS72100006

To assign agents, follow menu path **MDGIMG • Master Data Governance, Central Governance • Central Governance for Supplier • Workflow • Assign Processor to**

Change Request Step Number in BRFplus for Supplier. Figure 13.23 shows an example of an agent determination using a BRFplus decision table. This example shows how agents are determined for change request type BPHP1. Each step in the change request, the system will determine the object type and object ID using the decision table. Object types can be **Organizational Unit**, **Position**, **Job**, or a **User** ID. All users assigned to the specified position, job, or organizational unit will be determined as agents for the workflow step.

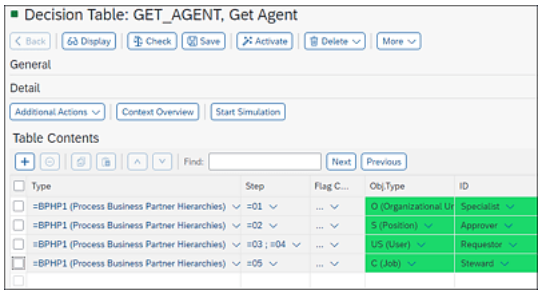


Figure 13.23 Example of Agent Determination Using a BRFPlus Decision Table

Workflow Templates WS54400001 and WS54300005

Workflow template WS54400001 is used by change request type CUST1P2, and workflow template WS54300005 is used by change request type SUPPL1P1. Change request type CUST1P2 is used to create a customer master record, and change request type SUPPL1P1 is used to create a supplier master record. Both of these workflow templates use parallel workflows to trigger multiple data enrichment and approval steps to speed up the master data creation process. Process flow and agent determination are similar for both of these templates.

The process starts when a requestor submits a request to create a customer master data or vendor master data. A master data approver then receives a work item to approve the central data such as names, addresses, bank details, tax numbers, and so on. If any changes are required, the request can be sent back to the requestor who can either update the change request or withdraw the change request. If no further updates are required, the approver can approve the request.

After this step, data specialists for financial, sales, and purchasing data receive work items. Financial specialists receive work items for both customer and supplier master data. Sales specialists receive work items for customer master data, and purchasing specialists receive work items for supplier master data. The number of work items created is based on the number of company codes, sales areas, and purchasing organizations configured for the customer or supplier.

One work item is created for each company code, sales area, and purchasing organization each. All of these work items are created in parallel so that all data specialists can enrich the change request at the same time. This ensures a speedy processing of master data creation. Each data specialist task is followed by an approval task where a data approver can approve or reject the data provided by the data specialist. If the changes are rejected, the data specialist gets a work item to rework the data entered and the same can be resubmitted to the approver. Because each data specialist is working on only one piece of

company code, sales area, or purchasing organization data, there is no option to withdraw the request at this step.

Once all approvers approve the changes, the change request will be activated. If there is no error in the activation, the process is complete. If there is an error in the activation of the change request, a data steward gets a work item and can decide to retry the activation process or send the request back to the requestor who can either make further changes or withdraw the request.

A subworkflow (`WS54400002` for customer master data and `WS54300006` for supplier master data) is used for the data maintenance and approval steps. This allows the main workflow to create multiple parallel instances for the subworkflow for each of the company codes, sales areas, and purchasing organizations. While the subworkflows are active, the main workflow will wait for all the subworkflows to be completed. When all subworkflows are completed, the main workflow will resume.

Figure 13.24 shows the process diagram for workflow templates `WS54400001` and `WS54300005`.

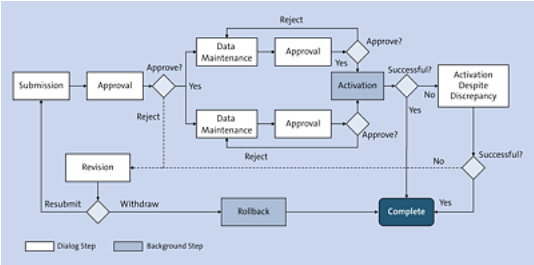


Figure 13.24 Process Diagram for Workflow Templates WS54400001 and WS54300005

Agent determination for workflow templates `WS54400001` and `WS54300005` is determined using a BRFplus table. For customer master data, agents are determined based on a combination of change request type, change request step, sales area, company code, and central master data change flag. For supplier master data, agents are determined based on a change request type, change request step, company code, purchasing organization, and central master data change flag.

SAP has delivered BRFplus application `MDG_BS_ECC_SUPPLIER_WORKFLOW` for the supplier master data workflow template. This BRFplus application has function `AGENT_DETEMINATION`, which is called by the agent determination task of the workflow template. This function uses decision table `GET_AGENT` in the application to determine the agents. To assign agents, follow menu path **MDGIMG • Master Data Governance, Central Governance • Central Governance for Supplier • Workflow • Assign Processor to Change Request Step Number in BRFplus for Supplier**.

SAP has delivered BRFplus application `MDG_BS_ECC_CUSTOMER_WORKFLOW` for the customer master data workflow template. This BRFplus application has function `AGENT_DETEMINATION`, which is called by the agent determination task of the workflow template. This function uses decision table `GET_AGENT` in the application to determine the agents. To assign agents, follow menu path **MDGIMG • Master Data Governance, Central Governance • Central Governance for Customer • Workflow • Assign Processor to Change Request Step Number in BRFplus for Customer**.

Table 13.6 shows an example of an agent assignment using the BRFplus decision table. In this decision table, agents are determined based on a change request type, change request step, company code, sales organization, distribution channel, division, and central data changed flag. When a change request is submitted, the requestor will specify the sales areas (a combination of sales organization, distribution channel, and division) and company codes relevant for the customer master data. The system will create one work item for each sales area and company code. Agents are determined for each work item separately using the BRFplus decision table. In the following example, the system will determine USER1 as the agent for company code C001 and USER2 as the agent for company code C003. Note that the BRFplus decision table supports multivalue attributes and patterns. For example, USER5 is determined as the user for both company codes C001 and C003 for step 6. By carefully configuring the decision table, a complex agent determination logic can be implemented.

Change Request Type	Change Request Step	Company Code	Sales Organiza-tion	Distribu-tion Channel	Division	Central Data Change	Object Type	Obje ID
CUST1P2	1							
CUST1P2	4	C001					US	USER
CUST1P2	4	C003					US	USER
CUST1P2	5		S001	01	01		US	USER
CUST1P2	5		S002	01	01		US	USER
CUST1P2	6	C001; C003					US	USER
CUST1P2	6		S001; S002				US	USER
CUST1P2	7						US	USER

Table 13.6 Agent Assignment for Workflow Template WS54400001 Using the BRFplus Decision Table

The agent determination table can be prepared using Excel and uploaded into the system. Existing agent determination table entries can be downloaded into Excel for further editing.

Workflow Template WS54300003

Workflow template `WS54300003` is a basic two-step approval workflow template used in scenarios where the four-eye principle of governance needs to be enforced without a complex process. This workflow template is used by change request types `CUST1P3`, `CUST2P1`, `CUST2P2`, `CUST5P1`, `CUST6P1`, `CUSTMRP1`, `CUSTMRP2`, `SUPPL1P2`, `SUPPL2P2`, `SUPPMRP1`, and `SUPPMRP2`. These change request types are used to update existing customer/supplier master data, block or delete customer/supplier master data, and process multiple customer/supplier master data records in a single change request.

The process starts when a requestor submits the change request. The change request is routed to an approver who can either approve or reject the change request. If the change

request is rejected, it's routed back to the requestor who can make further changes and resubmit the request or withdraw the request. If the request is resubmitted, it's routed to an approver for approval. If the request is approved, the system will try to activate the changes. If the changes are activated successfully, the process ends. If there is any error during the activation, the system will write to the activate log and route the request to a data steward. The data steward can review the log, and if the underlying error is already resolved, can retry activation. If the error can't be resolved, and further changes are required, the data steward can route the request to the requestor. The process will resume as before. Figure 13.25 shows the process diagram for this workflow template.

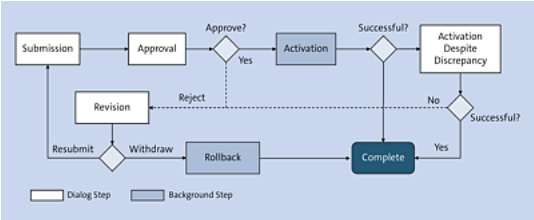


Figure 13.25 Workflow Process Diagram for Template WS54300003

Workflow template `WS54300003` uses agent determination rule `75700139` to determine agents for work items created for all the dialog tasks used in this template. This rule uses SAP MDG Customizing to determine agents for each step of the workflow. To set up agents for various steps in the workflow, call Transaction MDGIMG and follow menu path **Master Data Governance, Central Governance • General Settings • Process Modeling • Workflow • Other MDG Workflows • Assign Processor to Change Request Step Numbers (Simple Workflow)**. Figure 13.26 shows an example of agent assignment for change request type `CUST2P1`, which uses this workflow template.

Agents are assigned based on the change request type and change request step. Agents can be assigned as users, organization units, jobs, or positions. An organization structure must be created using Transaction PPOMW, and organization units, jobs, or positions created in this structure can be used as agents in this Customizing activity. If multiple users are assigned to an organization unit, job, or position, all users receive a work item, but only one of the users must act. Once the work item is processed by one of the users, other agents can't process the same work item. User IDs can be assigned as agents, as shown in Figure 13.26. Typically, multiple users are assigned as processors of work items to ensure speedy processing of the work items. To assign multiple user IDs as agents, create one row for each user in this table, and the system will assign the work item to all the users so any one of them can process the work item.

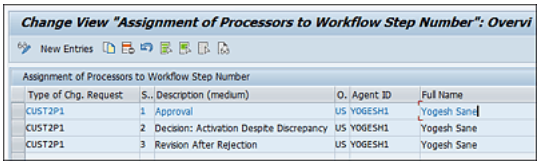


Figure 13.26 Agent Assignment for Workflow Template WS54300003

Workflow Templates WS54300007, WS60800059, and WS60800068

Workflow templates WS54300007, WS60800059, and WS60800068 are used by change request type SUPPL2P1, SUPPL5P1, and SUPPL6P1, respectively, to approve changes to supplier master data, including blocking supplier data and marking supplier data for deletion. This workflow template provides implementation of the four-eye principle and parallel work items for central, financial, and purchasing organization data changes.

The process starts when a requestor submits the change request to update the supplier master data. Depending on the type of data that is changed, the system creates one or more parallel work items. The system creates one work item for each company code and each purchasing organization data item that is changed, blocked, or marked for deletion, and one work item for central data if the central data is also changed or blocked in the change request. These work items are assigned to the respective agents, and all work items can be processed in any order. Each of these work items can be assigned to multiple agents, and only one agent must act for each work item. If any of the work items are rejected, all other approval work items are logically deleted (i.e., work items are marked for deletion, and no processing is possible, but the work item itself exists in the system), and the request is sent back to the requestor.

The requestor has an option to resubmit the request or withdraw the request. If the requestor resubmits the request, the system will create another set of work items and assign them to the respective agents. Depending on the data that is changed in the revised request, one or more approvers will receive a work item to approve, even if they have approved the same request before and there is no change to the data that they have approved. Once all approvers approve the change request, the system will activate the change request. If the change request can be activated successfully, the process will end. If there is any error during the activation step, a work item will be created and assigned to a data steward who can then try activation or send the request back to the requestor. Figure 13.27 shows the workflow process diagram for workflow template WS54300007.

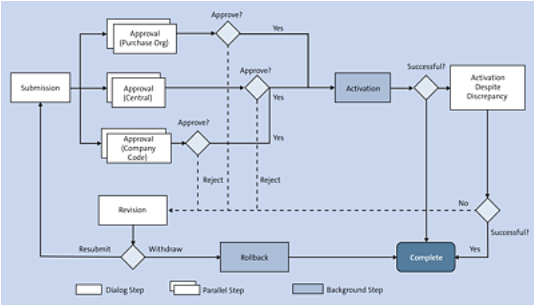


Figure 13.27 Workflow Process Diagram Template for WS54300007

Agent determination for workflow template WS54300007, WS60800059, and WS60800068 is determined using a BRFplus table. Agents are determined based on a change request type, change request step, company code, purchasing organization, and central master data change flag.

SAP has delivered BRFplus application MDG_BS_ECC_SUPPLIER_WORKFLOW for the supplier master data workflow template. This BRFplus application has function AGENT_DETEMINATION, which is called by the agent determination task of the workflow template. This function uses decision table GET_AGENT in the application to determine the agents. To assign agents, follow menu path **MDGIMG • Master Data Governance, Central Governance • Central**

Governance for Supplier • Workflow • Assign Processor to Change Request Step Number in BRFPlus for Supplier.

Table 13.7 shows an example of an agent assignment using a BRFplus decision table. In this decision table, agents are determined based on a change request type, change request step, company code, purchasing organization, and central data changed flag. When a change request is submitted, the system will analyze the type of data that is changed and create one work item for each company code and purchasing organization level data item that is changed. Additionally, if the central data is changed, one work item for central data will be created.

In the example in Table 13.7, the system will determine USER1 as the agent for the approval work item for central data. For company code 0001, agent FINUSER1 will be selected and FINUSER2 will be selected for all other company codes. If purchasing organization level data was changed, the system will determine user PURUSER1 as the agent for purchasing organization A001, and for all purchasing organizations with IDs starting with “A”, the system will determine user PURUSER2 as the agent. By carefully configuring the decision table, a complex agent determination logic can be implemented. Table 13.7 shows how the agent determination criteria can be populated in an Excel spreadsheet and uploaded to the BRFPlus application.

Type	Step	Company Code	Purchasing Organization	Central Data Changed Flag	Object Type	ID
=SUPPL2P1	=01			=X	US	USER1
=SUPPL2P1	=01	=0001			US	FINUSER1
=SUPPL2P1	=01	<>0001			US	FINUSER2
=SUPPL2P1	=01		=A001		US	PURUSER1
=SUPPL2P1	=01		Starts with “A”; exclude =A001		US	PURUSER2

Table 13.7 Agent Assignment for Template WS54300007, WS60800059, and WS60800068.

13.3 Finance Workflows

SAP MDG, Financials allows you to govern creation and changes to the financial master data. Financial master data that is typically under governance includes general ledger accounts, cost centers, and profit centers. In addition to these commonly governed master data objects, other master data objects that can be governed in SAP MDG include cost element, financial reporting structure, company, and consolidation entities. The following provides a brief description of some of the most governed master data objects in financials:

- **General ledger accounts**
  The general ledger is the most fundamental structure for recording financial information about a business. A general ledger account is an item within the general ledger. General ledger accounts are used to record different types of financial transactions and their values. Most of the data that is reported in a financial report is derived from general ledger accounts.
- **Cost center**
  During the normal course of business, every organization incurs costs. The cost center is a master data object in controlling that represents a delimited location where costs occur. These costs can include payroll costs, rent and utility costs, or any other costs relevant to a cost center. The posting and assignment of costs to cost centers enables managerial accounting and controlling of costs.

- **Profit center**
  A profit center is a management-oriented organizational unit used for internal control purposes. Structuring your organization as hierarchies of profit centers enables you to assign responsibility related to revenues and costs. Profit centers can be viewed as companies within a company.

13.3.1 Finance Data Model

SAP MDG provides a standard data model called 0G to govern financial master data entities. This data model includes entities such as chart of accounts, general ledger account, company code, cost center, cost center groups and cost center hierarchies, profit center groups and hierarchies, and so on. Many of these entities are related to each other. For example, a cost center is assigned to a profit center so there is a relationship between cost center and profit center. Relationships are used to link these entities together. Figure 13.28 shows an overview of all the entities in the finance data model for which a change request can be created.

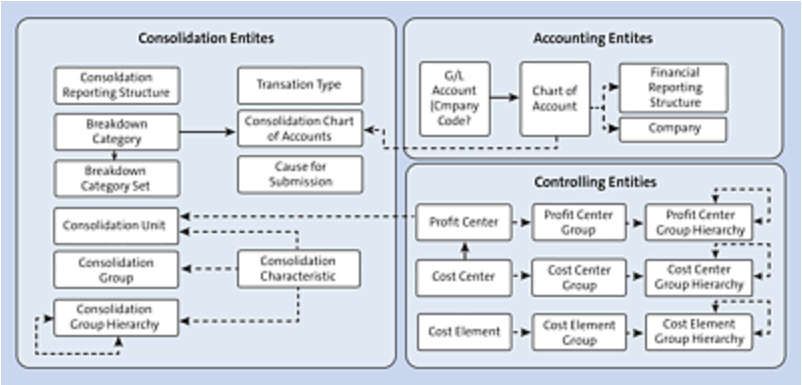


Figure 13.28 Entities and Relationships in the Finance Data Model in SAP MDG

Entities in the finance data model can be placed into three broad groups: accounting entities, controlling entities, and consolidation entities. Accounting entities include general ledger account, chart of account, financial reporting structure, and company. Entities in controlling include profit center, cost center, and their groups and hierarchies. Unlike the business partner data model, the finance data model has many entities for which a change request can be created. Finance data models also use many hierarchies, and a separate entity and change request types are used for management of hierarchies. For example, cost centers and profit centers are organized as hierarchies, and each cost center or profit center can be assigned to a hierarchy.

13.3.2 Change Request Types in the Finance Data Model

SAP delivers change request types for each of the create/change/delete operations for all the entities. Additionally, change request types are available for processing hierarchies of cost and profit centers, financial reporting structure, and consolidation group hierarchy. Each change request type is associated with a workflow template that determines the number of steps in the process, the agents that are assigned to execute these steps, and the order in which these steps are executed.

A key aspect of financial data is that multiple types of financial master data records are often created or updated

together, and these changes are approved as a single unit and distributed as a single unit. Either all changes are approved or all are rejected to ensure that various master data records are consistent with each other.

To facilitate maintenance of multiple types of records together, SAP has delivered four types of change requests, one change request each for accounting entities, controlling entities, and consolidation entities, and then one change request that allows changes to all financial master data records. Depending on how financial master data changes are governed, change requests for individual entities, change requests for multiple entities, or a combination of these two can be used.

13.3.3 Workflow Templates Used in Finance Change Requests

There are two types of workflow templates used in change requests in the finance data model: simple workflow and advanced workflow. An extended workflow template is also delivered by SAP. However, this template isn't configured as the default template for any of the change request types. In the following sections, we'll look at the two workflow templates for the simple and advanced workflows.

Workflow Template WS75700040

Workflow template `WS75700040` implements the simple workflow process pattern consisting of a submission step followed by two approval steps. This template is used by all change request types except for change request type `0G_ALL`,

which uses the extended workflow template. Validation and approval are done at the change request level, and all records are validated and approved as a single unit. A single work item is created for each step of the workflow. This template is used by change request types to create or update general ledger accounts, cost centers, profit centers, financial reporting structures, and so on.

The process starts when a requestor submits a request. The request is routed to a processor who validates all the data entered in the request and ensures that requested master data records can be created and distributed to all systems. The processor has the option to reject the change request and send it back to the requestor if any changes are required. If the processor finalizes change request processing, the request is routed to an approver who provides the final approval. Once the change request is finally approved, changes are activated and distributed. The process ends once changes are approved and distributed. At any point, the request can be rejected and sent back to the requestor with a note explaining why the change request is being rejected and changes are required so that request can be approved. The requestor has the option to make the required changes and resubmit the request or withdraw the request if the required changes can’t be made. If the request is withdrawn, all changes are rolled back. Figure 13.29 shows the steps in the workflow process for template `WS75700040`.

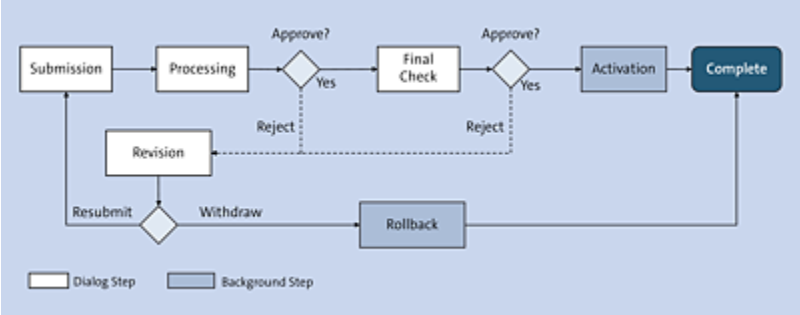


Figure 13.29 Workflow Process Diagram for Template WS75700040

A key observation for this workflow template is that all changes are approved or rejected as a single unit by a single user agent at each step. This workflow template is used for creation or updating of master data records of a single type, that is, creation or mass change of general ledger accounts or cost centers. This workflow template is also used by change request types that allow master data entities from the same group; for example, change request type `0F_FIN` allows changes to all financial accounting entities such as general ledger accounts, financial reporting structures, and companies using a single change request.

Agents for workflow template `WS75700040` are determined using an SAP MDG Customizing table. To set up agent determination for a simple workflow template, call Transaction MDGIMG, and follow menu path **Master Data Governance, Central Governance • General Settings • Process Modeling • Workflow • Other MDG Workflows • Assign Processor to Change Request Step Number (Simple Workflow)**. Figure 13.30 shows an example of agent assignment for change request type `CCT2P1`.

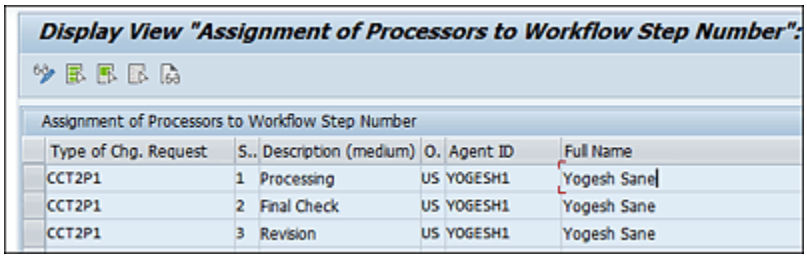


Figure 13.30 Agent Assignment for Workflow Template WS75700040

Agents are assigned based on a change request type and change request step number. Agents can be assigned as users, organization units, jobs, or positions. An organization structure must be created using Transaction PPOMW, and organization units, jobs, or positions created in this structure can be used as agents in this Customizing activity. If multiple users are assigned to an organization unit, job, or position, all users receive a work item, but only one of the users must act.

Workflow Template WS75700027

Workflow template `WS75700027` implements the advanced workflow process pattern consisting of a submission step followed by four approval steps. It supports parallel processing at each of the approval steps by creating multiple work items and assigning these work items to one or more agents. All work items created for a step can be processed in any order. Change request type `0G_ALL`, which allows changes (including creation) to all types of financial master data entities, uses the advanced workflow template.

The process starts when a requestor submits a request. This workflow template is used by the `0G_ALL` change request type that supports changes to all types of financial master data

objects. Typically, this change request is created as a mass change request where multiple objects that are linked to each other are created or updated together. Once the change request is submitted, the system determines one or more agents for evaluation depending on how many agents are configured for this step. If there are multiple agents configured for this step, the system creates one work item for each agent (or agent group), and all work items can be executed in parallel in any order. If any of the evaluators rejects the request, all other work items that aren't yet processed are logically deleted, and the request is sent back to the requestor for further changes.

When all evaluators approve, the request is sent for approval. After approval, the request is sent to a processor who will then make sure that all requested changes can be executed and makes additional changes to the records, if required. Once the request is successfully processed, it's sent for the final check. Once the final check is approved, the request is activated and distributed. The process ends after successful activation of the changes. At any point, the request can be rejected and sent back to the requestor. The requestor can make further changes as suggested by the evaluators or processors and resubmit the request or withdraw it if the suggested changes can't be made. If the request is withdrawn, changes are rolled back, and the process ends. Figure 13.31 shows the process diagram for the advanced workflow template.

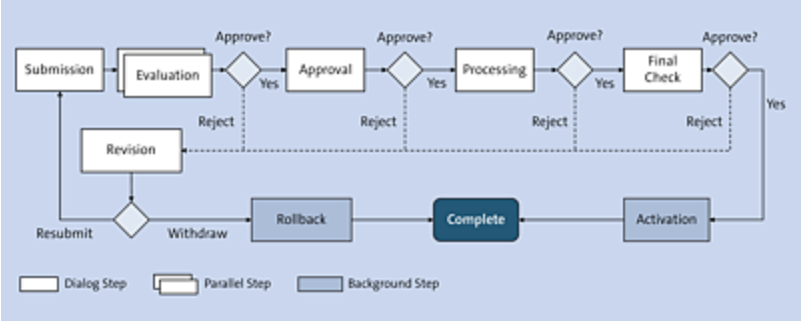


Figure 13.31 Process Diagram for Advanced Workflow Template WS75700027

Agents for workflow template `WS75700040` are determined using an SAP MDG Customizing table. To set up agent determination for a simple workflow template, call Transaction MDGIMG and follow menu path **Master Data Governance, Central Governance • General Settings • Process Modeling • Workflow • Other MDG Workflows • Assign Processor to Change Request Step Number (Simple Workflow)**. Agents are assigned based on a combination of change request type and step number. For the evaluation step, multiple rows can be created, and the system will create one parallel work item for each row for this step. For other steps, only one work item is created. If multiple agent assignment entries are created in this Customizing table, the system will assign all the agents to the same work item, and any one agent can process the work item.

13.4 Material Workflows

Master data governance for materials allows you to govern creation of new materials and changes to existing material master data. There several different types of materials, and different industries use different types of materials. Governance processes also vary widely across industries, and there is no standard process that is followed across various industries. There are no preconfigured workflow templates available in SAP MDG for material master data. Instead, organizations use the rule-based workflow template (`WS60800086`) to build governance process patterns. Section 13.1.10 provides details on the rule-based workflow template.

13.5 Summary

This chapter provides a detailed description of how SAP Business Workflow is used to implement governance processes using SAP MDG. It covers the different types of master data records that can be governed using SAP MDG and key SAP MDG concepts such as change requests and change request steps. The chapter covers the preconfigured workflow templates provided by SAP for creating and changing business partners, customers, suppliers, and financial master data records. For each workflow template, a process diagram and agent determination strategy are provided. This chapter also provides a detailed description of the rule-based workflow template and the use of the BRFplus application, including decision tables used by the rule-based workflow template.

With the insights gained in this chapter, you'll be able to appreciate how SAP Business Workflow is used in real-world applications by various organizations across different industries.

Part II

Flexible Workflow in SAP S/4HANA

In this part, you will learn to use flexible workflow in SAP S/4HANA. This part will compare flexible workflows to classic workflows, describe standard and custom development scenarios, and teach you to set up standard workflows and develop custom workflows with step-by-step instructions.

14 Introduction to Flexible Workflow

SAP has introduced flexible workflows as an innovative and efficient way of designing the flow of sequential actions in SAP S/4HANA to meet industry standard digitalization requirements rapidly. Flexible workflows simplify the approval process with no code/low code efforts. It provides flexibility to business users to configure workflow templates based on predefined standard scenarios without the need of a developer. Flexible workflow is the default workflow engine in SAP S/4HANA.

Flexible workflow is a framework introduced in SAP S/4HANA as part of the SAP S/4HANA workflow engine. It allows developers to create simple workflow scenarios and allows functional/business process specialists to model the process flow very easily by using SAP Fiori apps.

Flexible workflows support both dialog approval and background execution of sequential business processes. The flexible workflow is based on a set of predefined implementation objects, such as workflow scenarios, activities, start and step conditions, and agent determination rules. It integrates the functions of email notification, deadline overdue monitoring, and exception

handling during approval. The flexible workflow framework is tightly coupled with the new SAP Fiori user experience (UX) framework. Consultants and business users should understand how to work with SAP Fiori apps to use the benefits of flexible workflows.

SAP has clearly segregated scenario development and scenario modeling. Scenario development will be done by developers in the backend system, and the same scenario is modeled per business flow by using the SAP Fiori app called Manage Workflows. By using this app, functional consultants and business process specialists can model single-step and multistep approval processes very easily. They can even configure start conditions to start the flow, assign agents who act on the process, define exception handling, send email notifications if a work item is overdue, and so on. The Manage Workflows app provides lot of flexibility to business users to simplify/model their process flow, which were configured at the code level by developers in classical workflows.

SAP tightly integrated the approval process with the My Inbox app. All flexible workflow work items will be routed and available in the My Inbox app, so approvers can check their work items in their My Inbox app and take appropriate action based on the workflow setup.

The purpose of this chapter is to teach you how to use flexible workflows in day-to-day approval scenarios. To accomplish this, we'll explain the basic configuration, required authorizations, and required SAP Fiori apps to develop and model flexible workflows. In addition, we'll compare its features with classical workflows and walk

through one custom flexible workflow scenario. We'll also explain simple troubleshooting techniques to check in case any issues arise. We'll cover the end-to-end approver scenario by integrating with My Inbox and also touch on the transport process to move scenarios across the system landscape.

14.1 Authorizations and SAP Fiori Applications Required for Development

Different users need different accesses and SAP Fiori roles to work on flexible workflows. You can segregate authorizations based on user roles and details as mentioned in this section. Table 14.1 and Table 14.2 provide details on important SAP Fiori apps and their relevant technical/business catalogs required for different personas. You can submit a request to your security team for access to these apps based on your role in the project:

- **Developers**
  Developers require all workflow-relevant authorizations. In addition to workflow access, they need access to new Transaction SWDD_SCENARIO (Scenario Editor) to work on flexible workflow scenarios, and they need SAP Fiori app access (see Table 14.1) as well.
  Developers should also have authorization to access OData service `SWF_FLEX_DEF_SRV`. SAP released SAP Note 3100365, which has all the required role details.

SAP Fiori App	Technical/Business Catalog
Manage Workflow Scenarios	SAP_BASIS_TCR_T
Manage Workflows	SAP_BASIS_TCR_T
Configure Software Packages	SAP_BASIS_BC_EXT
Register Extensions for Transport	SAP_BASIS_BC_EXT
Maintain Email Templates	SAP_BASIS_TCR_T
Manage Teams and Responsibility	SAP_CA_BC_RSM

Table 14.1 Developers: Apps and Catalogs

- **Business process specialists and functional consultants**
  These people mainly configure workflow templates by using the Manage Workflows app with predefined scenarios. They need all the apps and required roles listed in Table 14.2.

SAP Fiori App	Technical/Business Catalog
Manage Workflows	SAP_BASIS_TCR_T
My Inbox	SAP_BR_MANAGER
Manage Teams and Responsibility	SAP_CA_BC_RSM

Table 14.2 Business Process Specialists: Apps and Catalogs

- **Line approvers**
  These are actual business users who check work items in the My Inbox app and take appropriate action. They need the My Inbox app access in addition to their business roles (see Table 14.3).

SAP Fiori App	Technical/Business Catalog
My Inbox	SAP_BR_MANAGER

Table 14.3 Business Users: Apps and Catalogs

> **Note**
>
> This section discussed minimum access/SAP Fiori apps for different personas, and you'll have to work with your security team to get any other roles/access required during the implementation phase.

14.2 Flexible Workflow Scenarios

You'll come across different approval scenarios in your day-to-day business activities. SAP had provided many scenarios to support these approval processes such as purchase order approvals, journal entry approvals, invoice approvals, service sheet entries, sales order flows, and so on. This list goes on and on, and there are many out-of-the-box scenarios available in the system that the business process specialist can check and make use of. However, every organization has some unique requirements that can't be met by using standard scenarios. In these cases, they can plan for custom scenarios.

Flexible workflows fall into two broad categories: standard flexible scenarios and custom flexible scenarios. We'll explain these two categories in the following sections at length, so you'll be comfortable working in both standard and nonstandard scenarios. We'll also spend time comparing the two scenarios to help you know how to choose between them.

14.2.1 Standard Flexible Scenarios

Many classical workflow templates are available in the system for processes such as invoice approval, purchase requisitions approval, and so on to cater to many such approval flows over the years. SAP provides similar flexible workflow templates for many of those standard approval flows in SAP S/4HANA, and this list is growing day by day.

SAP has provided exhaustive help documentation as well to make use of these templates.

When business process specialists/functional consultants get any requirements for approval flow in standard transactions, they can quickly configure and enable these appropriate standard scenarios. This is an absolutely no-code/low-code option in the SAP S/4HANA system. We'll explain where to find these standard scenarios and the required configuration to enable them in Transaction SPRO in Section 14.4.

14.2.2 Custom Flexible Scenarios

As mentioned, there are certain scenarios you can't meet by using standard templates. SAP has provided flexibility to create custom scenario in such cases, such as a custom HR form filled in by the requester and sent to the approver. The approver will check the data and approves it if everything is fine. Upon approval, data is posted back into SAP. Because it's purely a custom requirement, there is no standard template available in the system that applies.

We have to create a custom scenario and workflow to cater to this type of custom requirement. The developer has to model the requirement and create the custom scenario in the backend system. The business process analyst/functional consultant then creates a workflow template in SAP Fiori by using the Manage Workflows app.

Hopefully you understand now when to go for standard and when you need to implement custom flexible scenarios.

We'll compare both options in detail in the following sections.

14.2.3 Comparing Flexible and Classical Workflows

Let's compare the main differences between classical and flexible workflows to give you more insights into flexible workflows. As listed in Table 14.4, you can see the key differences between classical and flexible workflows, which will make it easier to develop these functionalities.

Feature	Classical Workflow	Flexible Workflow
Availability	SAP ERP, SAP S/4HANA	SAP S/4HANA
Configured by Manage Workflows app	No	Yes
Work items	Yes	Yes
Integration with My Inbox	Yes	Yes
Type of process supported	Sequential/parallel	Sequential

Feature	Classical Workflow	Flexible Workflow
Ad hoc workflows	No	Yes, in the product lifecycle management (PLM) module flexible workflows
Events	Yes	Yes
Manage Teams and Responsibilities app integration for agent determination	No	Yes, but not available for custom workflows
Substitution available (active/passive)	Yes	Yes
Workflow configured by	Developers (IT team)	Business process specialist
Exception handling	Handled with flow modeling	Exception handling in the Manage Workflows app for negative actions

Feature	Classical Workflow	Flexible Workflow
Deadline monitoring	Yes	Yes (from SAP S/4HANA 1909 version onward)
Maintain Email Template app integration	No	Yes (from SAP S/4HANA 1909 version onward)
Workflow Builder	Transaction SWDD	Transaction SWDD_SCENARIO
Workflow logs	Yes	Yes
Complexity of workflows	Low to complex can be modeled	Typically, complex flows split into smaller condition sequential flows in Manage Workflows app

Table 14.4 Comparison between Classical and Flexible Workflows

14.2.4 Choosing Between Classical and Flexible Workflows

You'll come across many scenarios in real-world business processes in which it's very difficult to choose which between a classical or flexible workflow to fulfill business requirements. Figure 14.1 helps you make a decision and choose the right framework to deliver on the requirements.

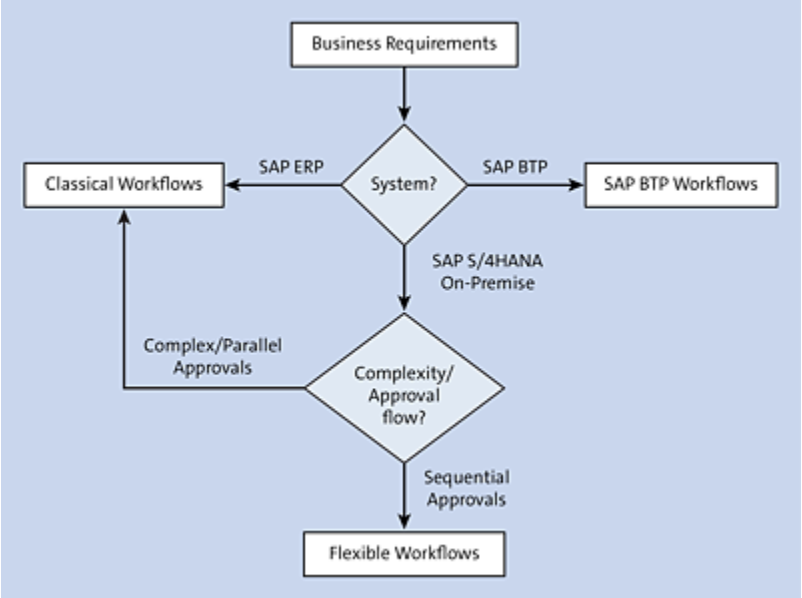


Figure 14.1 Choosing between Classical and Flexible Workflows

14.3 Migrating to Flexible Workflows

All new key words are used when migrating to flexible workflows from classical workflows. Table 14.5 lists the new keywords.

Classical Workflows	Flexible Workflows
Workflow template	Workflow scenario
Agent responsibility rules	Teams and responsibilities
Web Dynpro/module pool	SAPUI5/SAP Fiori
Universal worklist/Business Workplace	My Inbox
Transaction SWDD	Transaction SWDD_SCENARIO

Table 14.5 New Key Words in Flexible Workflows

Let’s also look at our migration path, which is useful when migrating to flexible workflows. Brownfield implementations are more common these days, so the decision is whether to develop a new flexible workflow or not during migration. Figure 14.2 will be helpful to make the right decision when migrating from an SAP ERP system to SAP S/4HANA (both on premise and in the cloud).

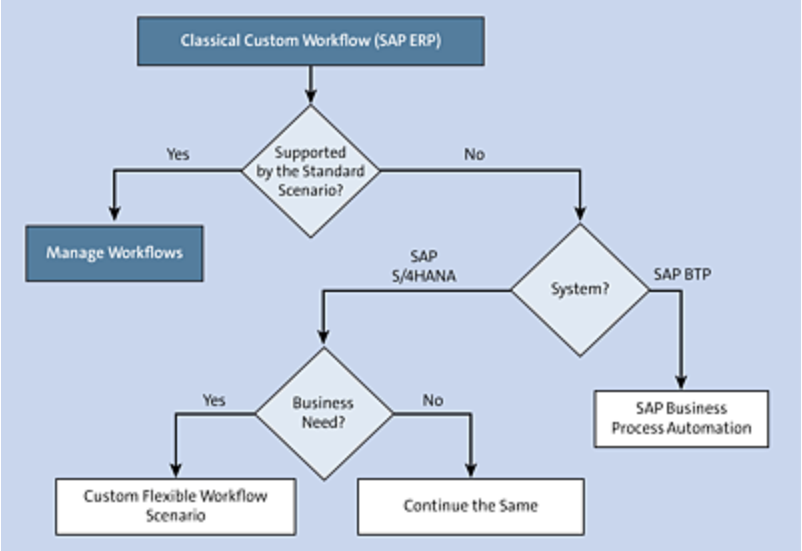


Figure 14.2 Migration Path

14.4 Setting Up a Standard Flexible Workflow Scenario

We'll explain how to find standard flexible scenarios using different methods and how to configure and activate one standard template with step-by-step instructions. Let's take a purchase requisition flexible scenario as an example and see how to find this scenario in the SAP Help site, how to locate it in the scenario editor, how to activate the scenario, and how to configure end to end by using the Manage Workflows app.

14.4.1 Finding Standard Workflows on SAP Help

SAP Help has provided detailed step-by-step instructions to activate many standard scenarios. You'll find the appropriate scenarios under the relevant functional module help links. For example, we can walk you through the purchase requisition flexible scenario in SAP Help (the link for the same is *http://s-prs.co/v569701*).

For approval of purchase requisitions, you have two options:

- **Overall release**
  The entire purchase requisition is approved. This type of approval is also known as header level approval.
- **Release of purchase requisition items**
  The items of the purchase requisition are approved individually. This type of approval is also known as item-level approval.

14.4.2 Finding Workflows in Scenario Editor and the Manage Workflows App

Developers can find the flexible workflow scenarios (by clicking on the **Scenario** field value help in the left-side navigation panel) in the scenario editor using Transaction SWDD_SCENARIO (see Figure 14.3 to see what this looks like in an on-premise SAP S/4HANA system). You can check all artifacts in the scenario editor and runtime class details. We'll explain more about the scenario editor in Chapter 15.

You can click on the **Scenario** field value help in the left-side navigation panel in Transaction SWDD_SCENARI to see all the scenarios available in the system in the value help popup window. Choose the appropriate scenario, and explore all details. For example, we've selected **Approve Purchase Requisition Item Level Scenario**. Generally, functional consultants and business process specialists aren't interested in the technical details of the scenario; they just want to see the business version so they can tailor it per business requirements.

To tailor it, you'll use the Manage Workflows app. Scenario should be activated in the backend to be available in the Manage Workflows app. We'll explain those steps in Section 14.4.4.

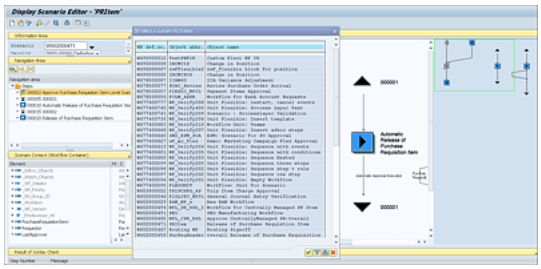


Figure 14.3 Transaction SWDD_SCENARIO

For now, let's just discuss how to find a workflow in the Manage Workflows app. Log in the SAP Fiori launchpad and access the Manage Workflows app (see Figure 14.4), provided you've got relevant roles assigned to your user ID. You can see all the activated scenarios in the **Workflows** dropdown.

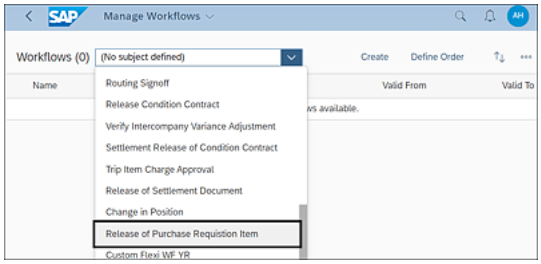


Figure 14.4 Manage Workflows App: Workflows List

14.4.3 Activating the Scenario

Figure 14.5 explains the sequential flow of activities to be performed to activate any flexible workflow scenario in the system. A few are mandatory and a few are optional. Let's take the purchase requisition scenario as an example and go through all these steps one by one.

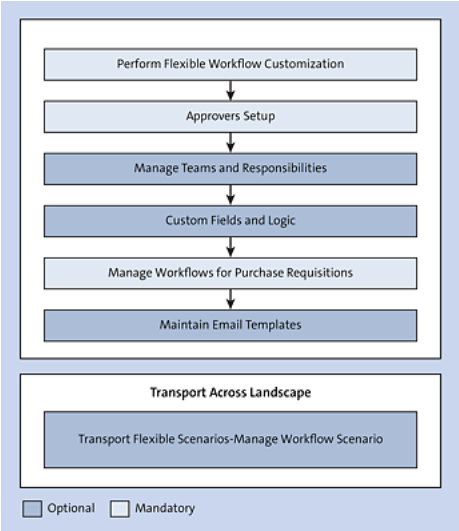


Figure 14.5 Scenario Activation Steps

The scenario activation steps are described in the following:

1. **Perform flexible workflow customization.**
   As a prerequisite, you must complete all regular workflow configuration steps in the backend system that are mentioned in Chapter 2, Section 2.3. Ensure that the automatic Customizing is active for the workflow functionality. Check whether all Customizing steps in Transaction SWU3 (see Figure 14.6) listed here have a green checkmark:

   - **Configure RFC Destination**
   - **Edit System Administrator for Workflow**
   - **Edit Active Plan Version**
   - **Classify Decision Task as General**
   - **Document Generation/Form Integration**
   - **Edit Time Units**
   - **Schedule Background Jobs SAP Business Workflow**

   If not, click **Redo Automatic Customizing**, or select one of the activities, and click **Execute Activity** to make the required settings.

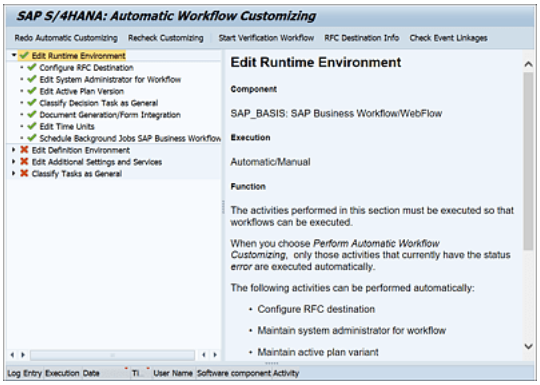


Figure 14.6 Transaction SWU3

2. **Activate flexible workflow scenarios.**
   Activate the relevant type of business document flexible workflow scenario. In our example, it's purchase requisition. In Transaction SPRO (Customizing) under **Application Server** • **Business Management** • **SAP Business Workflow** • **Flexible Workflow** • **Scenario Activation**, you'll add the scenario ID you want to use and activate it. This is one of the prerequisites to get scenarios in the Manage Workflows app.
   As shown in Figure 14.7, click on the **New Entries** button, enter the required scenario, and check the **Active** checkbox in the **Activating a scenario** table. The screen shows the ID added and activated for the purchase requisition line-item approval scenario. Similarly, you need to maintain the scenario ID for the required workflow.

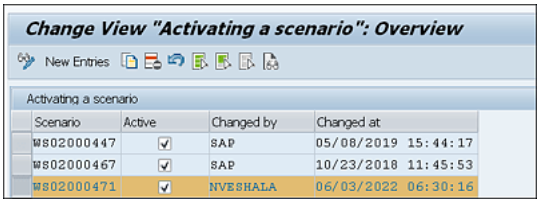


Figure 14.7 Activate the Flexible Scenario

3. **Define step names and decision options in My Inbox.**
   Generally, all tasks will appear in the approver's My Inbox app whether they are relevant or not. This becomes a tedious job to filter required tasks by approvers, and they might ask to send only relevant work items to their inbox. You'll maintain these settings in Transaction SPRO path **SAP NetWeaver • Application Server • SAP Gateway Service Enablement • Content • Workflow Settings • Enable Task Filter**.
   To display work items in the My Inbox app, you must maintain the required relevant workflow decision task ID details via Transaction SPRO menu path **SAP NetWeaver • SAP Gateway Service Enablement • Content • Workflow Settings • Maintain Task Names and Decision Options**, which is shown in Figure 14.8.
   You'll maintain all step details as mentioned in Table 14.6 by clicking on the **New Entries** button shown in Figure 14.8.

Step	Details
Workflow ID	Maintain the required workflow ID.
Step ID	Maintain the required dialog step ID in the workflow.
Icon MIME Repository Path	Enter the icon MIME repository.
Step Description	Enter the step description.

Table 14.6 My Inbox Step Details

You can see how it has been maintained in the purchase requisition scenario example in Figure 14.8. You have to maintain in a similar way for other scenarios as well.

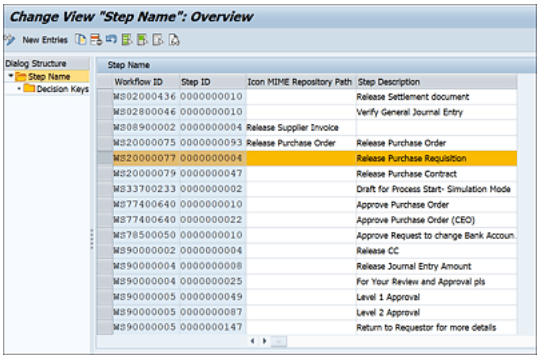


Figure 14.8 My Inbox Settings

You need to select **Step ID** and maintain decision keys for that step ID. As shown in Figure 14.9, maintain the following values in the **Decision Keys** maintenance view:

- **Decision Text**: This value is the dialog step outcome in the workflow task. This will be displayed as button text in the My Inbox app.
- **Nature**: This value is the workflow dialog task step action nature. Based on this value, button colors will be changed in the My Inbox app.

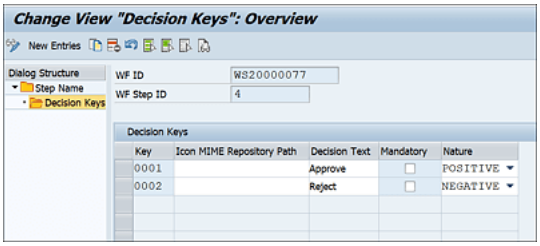


Figure 14.9 My Inbox: Task Decision Keys

4. **Define visualization metadata for My Inbox.**
These are other important settings that you need to perform to integrate work items properly in the My Inbox app. This is required to navigate to another page from the My Inbox app per business requirements. Go to Transaction SWFVISU, and maintain these settings for the required workflow tasks.
You can see how these details are maintained in the example scenario in Figure 14.10. You've already gone through these details in Chapter 11, Section 11.2.3. That is why we've outlined only our sample scenario purchasing requisition details in Figure 14.10. You can refer to the same chapter if any further details required.

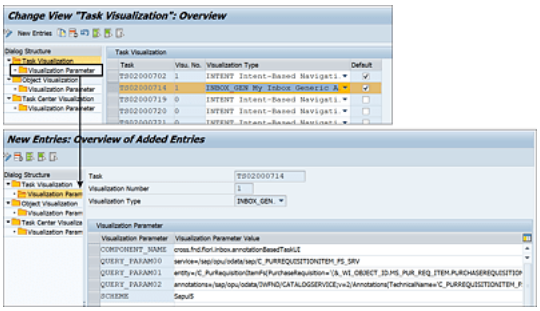


Figure 14.10 Transaction SWFVISU

Select a task, click on the **Visualization Parameter** node, and then maintain all the parameters shown in Table 14.7.

Parameter Name	Visualization Parameter Value
COMPONENT_NAME	cross.fnd.fiori.inbox.annotationBasedTaskUI
QUERY_PARAM00	service=/sap/opu/odata/sap/C_PURREQUISITION-ITEM_FS_SRV
QUERY_PARAM01	entity=/C_PurRequisitionItemFs(PurchaseRequisition='{&_WI_OBJECT_ID.MS_PUR_REQ_ITEM-.PURCHASEREQUISITION&}',PurchaseRequisitionItem='{&_WI_OBJECT_ID.MS_PUR_REQ_ITEM.PURCHASEREQUISITIONITEM&}')

Parameter Name	Visualization Parameter Value
QUERY_PARAM02	annotations=/sap/opu/odata/IWFND/CATALOG-SERVICE;v=2/Annotations(TechnicalName='C_PURREQUISI-TIONITEM_FS_ANNO_MDL',Version='0001')/$value
SCHEME	Sapui5

Table 14.7 My Inbox Visualization Parameters

5. **Deactivate event type linkages for old workflows.**
   As a rule, activating flexible workflow scenarios is sufficient to override old workflow scenarios. To be on the safe side, you can additionally deactivate any old workflow scenarios you may have used previously. You can activate all relevant OData services in Transaction /IWFND/MAINT_SERVICE for values helps, and so on per the SAP Fiori app reference library configuration details.

These are all the minimum activities that must be completed in the backend SAP S/4HANA system. Once these are completed, then you have to log in to SAP Fiori launchpad and configure the workflow template by using the Manage Workflows app. We'll explain those details in the following section.

14.4.4 Setting Up a Standard Scenario Using the Manage Workflows App

Business process specialists and functional consultants will configure the workflow template by using the Manage Workflows app, which is available by logging in to the SAP Fiori launchpad. We'll explain this setup by using our example scenario: **Release of Purchase Requisition Item**. Select this scenario in the Manage Workflows app (see Figure 14.11).

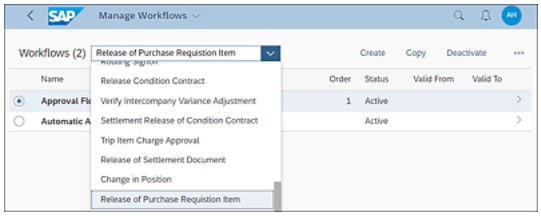


Figure 14.11 Manage Workflows Homepage View

By default, you can locate one active scenario in the list. We've outlined the sample business requirements in the following; you can configure the template quickly in the Manage Workflows app for these business requirements:

- The workflow should be triggered when the net amount is greater than 1,000 USD.
- The work item should be routed to the initiator's manager.
- The purchase requisition item should belong to purchase group Z02.
- The overdue item should be displayed in the approver's inbox if he won't approve after 24 hours.
- The workflow should be canceled when the approver rejects the line item.

The first step in the workflow template configuration is to click on the **Add** button in the Manage Workflows app homepage view to create the new template (see Figure 14.12).

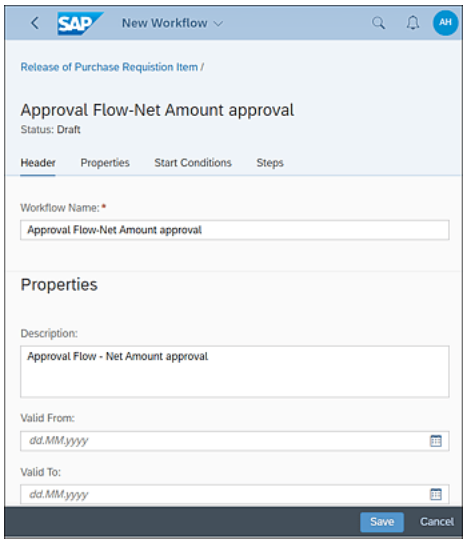


Figure 14.12 Manage Workflows: New Template

Here you'll fill out the following information:

- **Workflow Name**
  You must enter a meaningful name for your template.
- **Description**
  Enter a description of the workflow.
- **Valid From**
  If you're activating the workflow approval flow for a certain period, then enter a **Valid From** date.
- **Valid To**
  If you're activating the workflow for a certain period such as a campaign, then enter a **Valid To** end date. The workflow won't be triggered after the end date.

Per our business requirement, the workflow should be triggered when the net amount of a purchase request is greater than 1,000 USD. It's called a condition-based workflow triggering, and you have the same option available in flexible workflows.

We've added this condition as shown in Figure 14.13. You can add more than one start condition by using the **Create Alternative Preconditions** button as well, if required.

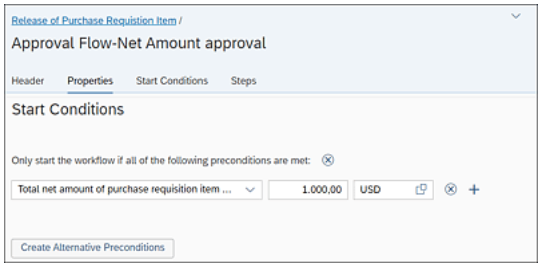


Figure 14.13 Manage Workflows: Start Conditions

Once the condition setup is completed, you have to create the required steps in the workflow. As shown in Figure 14.14, click on the **Create** button to add a new step. You can add either dialog activity or background activity here.

You can add dialog activity if any human intervention is needed (i.e., manager or any other persona per requirement will get the work item per the dialog step agent determination and will check the work item and take appropriate action). For background activity, the step will execute predefined business logic and complete the process.

Per our sample business requirement, you need to send the work item to the approver's inbox when the amount is greater than 1,000 USD. To achieve this requirement, you'll add one purchase requisition release step that will be routed to the initiator's manager. To do so, click on the **Create** button in the **Workflow Steps** section on the **Start Conditions** tab, which is shown in Figure 14.14. This will open the next screen where you'll provide all the step-related details and save the configuration.

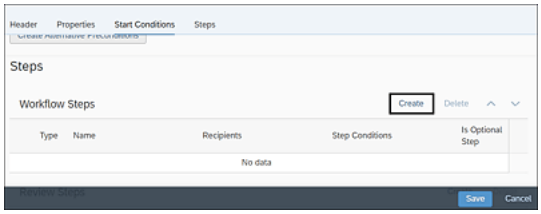


Figure 14.14 Manage Workflows: Add Steps

Specifically, as shown in Figure 14.15, maintain all step details such as agents, step conditions, deadline, exception details, and so on.

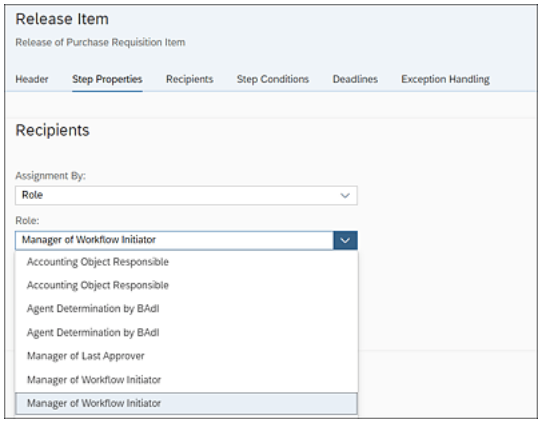


Figure 14.15 Manage Workflows: Step Details

On these tabs, you'll maintain the following:

- **Step Name**
  This is a meaningful step name per the business requirement.
- **Step Type**
  You can choose a relevant step type per the business requirement and scenario definition.
- **Assignment By**
  You'll configure how agents will be determined by using this dropdown value. In general,

you can choose either **Role** or **User**. If **Role**, then you can find agents based on different options, If **User**, then a user ID has to entered.

- **Role**
  You'll get agent determination options based on the scenario definition in backend. You'll choose the appropriate option to select the agent for your work item. In our purchase request approval example, we've selected **Manager of Workflow Initiator**.

The next tab you'll configure is **Step Conditions**. You'll configure conditions for steps as well, which means whenever a step condition occurs, then only that step work item will be triggered. Based on your business requirements, you must configure appropriate step conditions.

Per our business requirement, the work item will be triggered only when the purchasing group of the purchase requisition is **Z02**, as configured in Figure 14.16.

You can add more than one condition by using the **Add Alternative Preconditions** button on the **Step Conditions** tab.

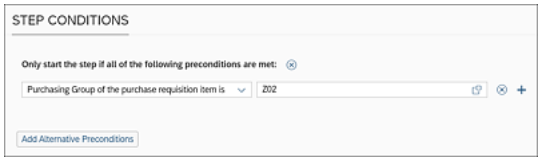


Figure 14.16 Manage Workflows: Step Conditions

The next tab is **Deadlines** in which you'll configure overdue work items and deadline email notifications. You can mark any work item overdue based on your business requirement.

Per our purchase requisition example business requirement, you must mark a work item as overdue if the approver won't take any action after 24 hours. To do this, select **Deadline calculation start with creation of workflow Instance**, and then select **1 Day(s)**, as shown in Figure 14.17.

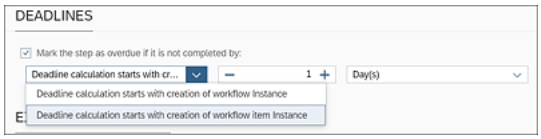


Figure 14.17 Manage Workflows: Deadlines

The next tab is **Exception Handling** (see Figure 14.18) where you'll maintain all workflow routing exception cases. All of these actions are based on **1 Day(s)** backend scenario design. All negative outcomes appear in this section, and we'll configure those outcomes.

Per our purchase requisition example requirement, the workflow should be canceled when the approver rejects the line item. You have to configure the **Exception Handling** section to meet this requirement. In the **Required Action** dropdown, all negative outcomes in the step definition in the backend will be available in this dropdown. Based on the requirement, choose the required action. The **Action Result** can be as follows:

- **Cancel Workflow**
  This will cancel the workflow.

- **Continue**
  The workflow will be continued further.
- **Repeat Step**
  The same step will be repeated again.
- **Restart Workflow**
  The workflow will be restarted from the beginning.

This configuration was done for our example scenario to cancel the workflow. Finally, click on the **Apply** button on the footer toolbar and create the step. Click on the **Save** button.

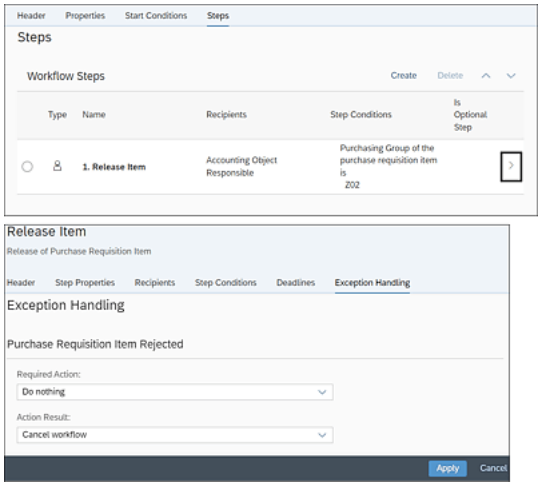


Figure 14.18 Manage Workflows: Exception Handling

Once the workflow is saved, you'll see the **Activate** button on the top toolbar (see Figure 14.19), which will activate the workflow.

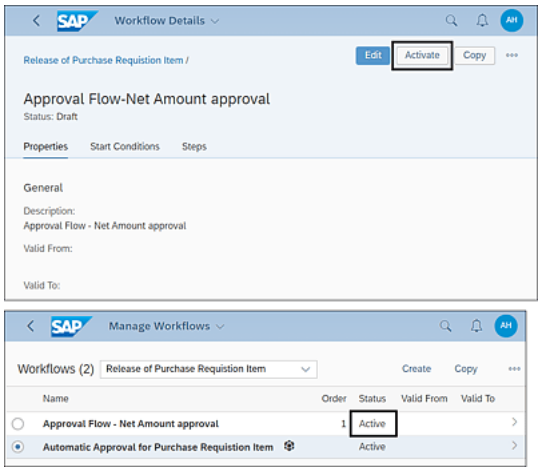


Figure 14.19 Manage Workflows: Activate the Workflow Template

We've explained all the steps to configure and activate the flexible workflow template by using the Manage Workflows app.

Apart from these options, you can configure deadline email notification and final approval or rejection email notifications as well in the Manage Workflows app. However, you must

create these templates during the scenario definition in the backend system, and then you'll only get those features in the Manage Workflows app. We'll explain those details in the following section.

14.5 Extending the Standard Flexible Scenario

SAP has provided multiple ways to enhance standard flexible scenarios per business requirements, and a few of those techniques are discussed in this section. SAP provided various business add-ins (BAdIs) to enhance agent determination rules, business context properties, and so on. You have to find relevant BAdIs in the BAdI repository and go through the BAdI documentation to implement the same.

Let’s look at one such BAdI to enhance preconditions in flexible workflows. SAP provides a set of preconditions and step conditions that we can use in the Manage Workflows app when creating the template. None of the existing conditions are meeting the business requirement, so SAP provided the way to create our own custom conditions by using BAdIs.

SAP provides two BAdIs to define the custom condition and evaluate the new custom condition and details mentioned:

- **`SWF_WORKFLOW_CONDITION_DEF`**
  This BAdI is used to define the new custom preconditions. This is a filter BAdI, so you have to provide the scenario ID when implementing this BAdI.
- **`SWF_WORKFLOW_CONDITION_EVAL`**
  This BAdI is used to evaluate the new precondition entered by the user. This is also a filter BAdI so you provide the scenario ID when implementing this BAdI.

Note

You can check the BAdI documentation to implement these BAdIs and also check sample class `CL_SWF_FLEX_IFS_BADI_COND_SAMP` to understand the coding pattern for custom condition definition and evaluation.

14.6 Summary

This chapter explained the basic details about flexible workflows. You saw the differences between classical and flexible workflows and how to find and activate standard scenarios using the Manage Workflows app. We also covered the minimum configuration required in the backend SAP S/4HANA system to activate any scenario and how to perform My Inbox integration. We’ll provide more details about custom scenario development by exploring some examples in the next chapter.

15 Custom Scenario Development

This chapter explains how to develop a custom flexible workflow. You'll learn about different areas of scenario development, such as flexible blocks, tasks, conditions, agent determinations, email templates, and so on. You'll also understand how to configure the business objects for flexible workflow. We'll walk through an end-to-end sample use case that will include configuring a custom scenario inside the Manage Workflows app as well.

As you recall from Chapter 14, flexible workflows are an innovative and new concept in SAP S/4HANA as a follow-up to SAP Business Workflow. They give more control and flexibility to business process experts to create workflows from predelivered content. You've seen the key differences between classical workflows and flexible workflows. You also learned how to set up out-of-the-box workflow scenarios, activate the predelivered flexible workflows, and extend the standard flexible workflows.

In this chapter, you'll learn more about creating custom workflow scenarios and flexible workflows on top of that. Beyond the set of workflow scenarios delivered by standard SAP and options to extend those standard scenarios, SAP

enables you to create custom flexible workflow scenarios for highly customized requirements in your business processes. You'll see different ABAP objects and steps needed to build an entirely custom flexible workflow to meet such individual customer needs. We'll dive deeper into the elements of flexible blocks, which are the basic building blocks of standard as well as custom workflow scenarios.

In addition, we'll take a sample use case to walk through the different objects while we go through the custom scenario development. Let's assume a very generic process in the hire-to-retire area: an HR application where an employee raises an HR request that subsequently goes for two levels of approval before updating into the actual database.

15.1 Workflow Class Development

As you've read about the ABAP class-based approach of building the core programming logic behind the classical workflow in Chapter 3, a similar approach is carried into the flexible workflows in SAP S/4HANA as well. ABAP classes form the basic building blocks that represent the business entity and hold the core technical and business functionalities of the workflow. In this section, you'll see the significance of classes with respect to flexible workflow scenarios, attributes, methods, and standard callback classes for flexible workflows.

15.1.1 Use Case for Walkthrough of Custom Scenario

We’ll be setting up the flexible workflow for the sample use case of the HR request we introduced at the beginning of this chapter. Let’s assume a very basic use case having two levels of approval. The breakdown of the process flow into the subsequent steps is illustrated in Figure 15.1 and listed here:

1. Employees submit an HR request, and the flexible workflow is triggered.
2. The workflow will be acted upon (approved/rejected) by the immediate line manager.
3. Once the line manger approves, the workflow will pass to the next approver, for example, the HR administrator.
4. After each step, relevant email notifications should be sent to approvers for pending action and to the employee regarding final completion of the workflow request.

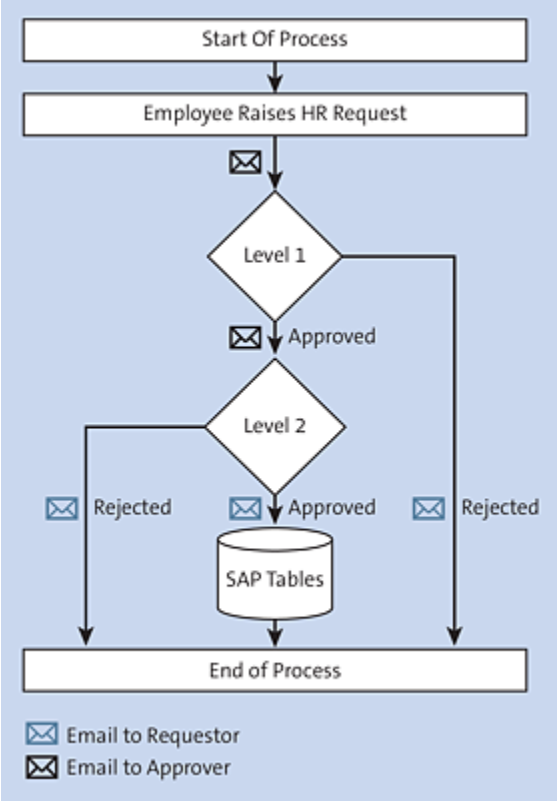


Figure 15.1 Flow Diagram for the Sample Use Case

In each relevant section of this unit, we'll take this sample use case and show the relevant artifacts being created.

15.1.2 Classes

Each workflow scenario has a leading object associated with it. A leading object represents the business entity in the SAP system and encapsulates the attributes, that is, the business data and its corresponding functions and the methods in an object-oriented fashion. Examples include an employee travel request, a purchase order, sales order, and so on. The leading object of the workflow scenario can be a business repository object or an ABAP class implementing interface `IF_WORKFLOW`. We prefer the more recent and

efficient approach—the class-based one for custom development. This class will provide the methods for individual tasks, start of event, and other events of the flexible workflow scenario.

In addition, callback classes `CL_SWF_FLEX_IFS_DEF_APPL_BASE` and `CL_SWF_FLEX_IFS_RUN_APPL_BASE` are responsible for handling the framework-level events in flexible workflow. You'll learn more about the significance of these classes in subsequent sections.

For our use case, we'll be creating custom class `ZCL_EMPLOYEE_REQUEST_WF`. Open Transaction SE24 (Class Builder), and click **Create** to create a new class. Provide the details as shown in Figure 15.2. Enter a meaningful description in the **Description** field, select **2 Public** for the **Inst.Generation** (instance generation) dropdown, and keep the **Class Type** set as **Usual ABAP Class**. Click **Save**.

In addition, create custom callback class for runtime data: `ZCL_EMP_FLEX_RUN_APPL_BASE` in a similar fashion. You'll add interfaces to these classes in the next section.

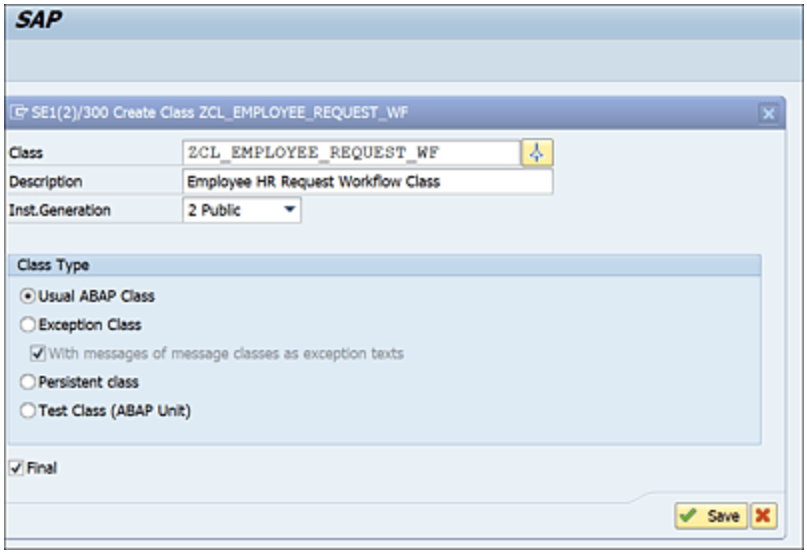


Figure 15.2 Custom Class for the Sample Use Case

15.1.3 Interfaces

Similar to classical workflows, every ABAP class being attached to a workflow scenario as a leading object should implement interface `IF_WORKFLOW`. This interface is the basic building block of ABAP class-based workflow implementation. Once you use this interface in the **Interface** tab in ABAP Class Builder (Transaction SE24) for the custom class, it adds `BI_OBJECT` and `BI_PERSISTENT` as well. Open the custom class `ZCL_EMPLOYEE_REQUEST_WF` via Class Builder using Transaction SE24 in change mode, navigate to the **Interface** tab, and enter "IF_WORKFLOW" as the **Interface**. Save and activate the class. Figure 15.3 shows the interfaces implemented in our use case's custom class

Similarly, the runtime and definition callback classes of the flexible block in a workflow scenario should implement interfaces `IF_SWF_FLEX_IFS_RUN_APPL` and `IF_SWF_FLEX_IFS_DEF_APPL`, respectively.

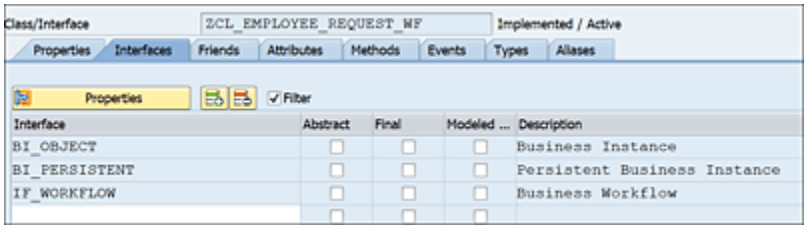


Figure 15.3 Interface for the Sample Use Case

15.1.4 Attributes

Attributes define the properties of the leading object, that is, the business entity being processed. For example, an

employee object may have attributes such as employee number, employee name, department code, and so on. Like classical workflows, there is a local persistence object reference. For this, you'll need an attribute of type `SIBFLPOR`. This local persistence object reference will be used to convert the generic class reference created by the workflow framework to a specific instance of the ABAP leading object class.

In our use case, `mv_pernr` will be the key attribute of the leading object class, which will uniquely identify the business object, that is, the employee in the organization, by using his personnel number. Add a new line in the attribute section of **Class ZCL_EMPLOYEE_REQUEST_WF**, enter the **Attribute** name as "mv_pernr", select **Level** as **Instance Attribute**, set **Visibility** as **Private**, select the **Key** checkbox, enter the **Associated Type** as "PERNR_D", and add a meaningful **Description**. Similarly, create another instance attribute called "Mv_por" of **Associated Type** "SIBFLPOR" with **Private** selected under **Visibility** in the **Attribute** section of custom class **ZCL_EMPLOYEE_REQUEST_WF**. This will be the persistent object reference as shown in Figure 15.4.

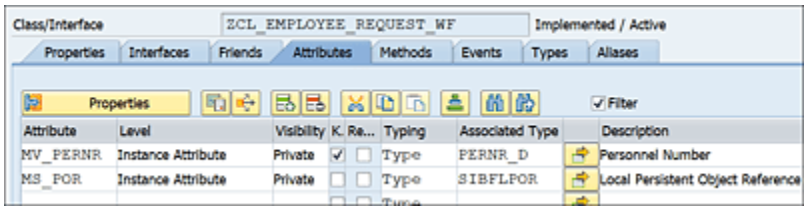


Figure 15.4 Attributes of the Workflow Class for the Sample Use Case

15.1.5 Events

Events are a more flexible and effective way to communicate with workflows. Events are messages that can be raised by a calling application program and published throughout the system. All the receiving workflows linked to that event react according to the type of event. The event concept of classical workflows is carried as is into flexible workflows. These events have parameters that are used to communicate data between application programs and workflow instances.

The following types of events are available:

- Triggering event of a task or a flexible workflow
- Terminating event of a task
- Cancellation event of a flexible workflow
- Event to restart the workflow

These events are also used as a part of exception handling; that is, you can define how the system should behave when any dialog work item is rejected by the approver. For our custom use case, create an event START in the custom class by following these steps (see Figure 15.5):

1. Open custom class **ZCL_EMPLOYEE_REQUEST_WF** in change mode via Transaction SE24.

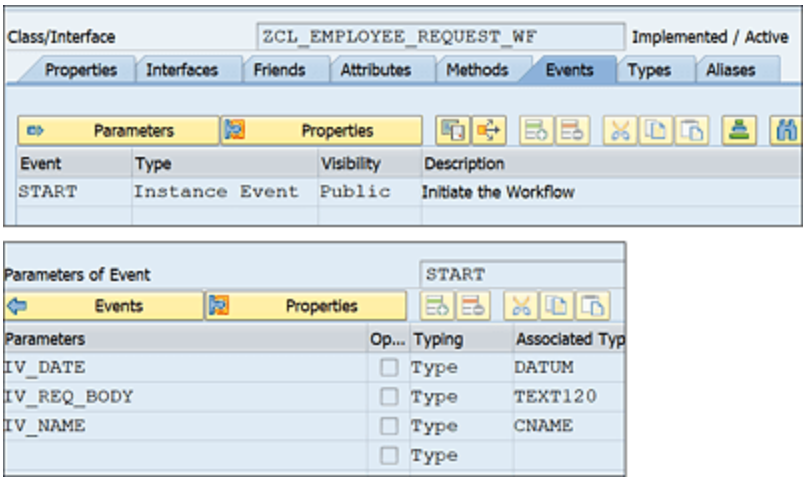


Figure 15.5 Event in the Custom Class for the Sample Use Case

2. Move to the **Events** tab, and add an **Instance Event** named **START** with **Public** visibility and a meaningful **Description**.
3. Add the event parameters by clicking the **Parameters** button. These parameters will be used to pass the required input from the calling application into the workflow.

15.1.6 Standard Methods

As with classical workflows, while creating the class for the leading object of a workflow scenario, each of the methods inherited from interface `IF_WORKFLOW` should be implemented or activated with empty source code at least.

The following methods are inherited from interface `IF_WORKFLOW` (i.e., appears in the **Methods** tab once the interface is added) of which `FIND_BY_LPOR` and `LPOR` are used to convert the workflow generic reference to the specific

instance of our ABAP class and vice versa. The purpose of all these methods is the same as in the classical workflows:

- `BI_PERSISTENT~FIND_BY_LPOR`
- `BI_PERSISTENT~LPOR`
- `BI_PERSISTENT~REFRESH`
- `BI_OBJECT~DEFAULT_ATTRIBUTE_VALUE`
- `BI_OBJECT~DEFAULT_ATTRIBUTE_VALUE`
- `BI_OBJECT~RELEASE`

Figure 15.6 shows the standard methods inherited from the interface that are to be implemented in our custom class of the HR request use case. In this use case, the employee number will be passed to the instance ID by a call to the `FIND_BY_LPOR` method by the workflow event handler. This employee number will be stored in the attribute of classes `mv_pernr` and `ms_por` via the constructor method during instantiation of the workflow object and will be used throughout the workflow processes.

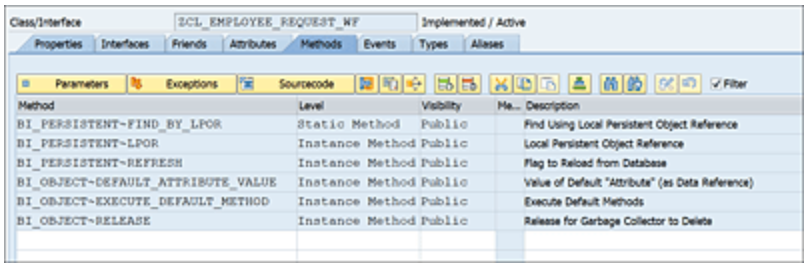


Figure 15.6 Methods of the Workflow Class of the Use Case

When it comes to callback classes, SAP has provided different methods, as listed in Table 15.1, which are plugged in at different framework level executions, such as before creation of an activity, after creation of an activity, and so

on. For our use case, we have the callback class for runtime data: `ZCL_EMP_FLEX_RUN_APPL_BASE`.

Methods in Runtime Callback Class	Description
`BEFORE_CREATION_CALLBACK`	Called before the next activity (work item) is created
`ON_CREATION_CALLBACK`	Called on creation of activity
`AFTER_CREATION_CALLBACK`	Called after the current activity is created
`BEFORE_COMPLETION_CALLBACK`	Called before activity completion
`AFTER_COMPLETION_CALLBACK`	Called immediately after activity completion
`MITIGATE_AGENT_RULE_EVALUATION`	Called when agent determination fails
`MITIGATE_START_COND_EVALUATION`	Called when the start condition evaluation fails
`RESULT_CALLBACK`	Called when the entire workflow is completed
`ON_CANCELLATION_CALLBACK`	Called when the entire workflow is canceled
`INIT`	Called on initialization of the object

Table 15.1 Methods in the Standard Callback Class

Each of these methods has different importing and exporting parameters with which you can read the context elements (workflow containers and task containers), outcomes of user decisions, and other workflow technical details at runtime. You can use these methods for performing tasks such as updating custom tables of your application, updating staging or intermediate tables for statuses, calling business application programming interfaces (BAPIs) and function modules (especially on completion of the workflow) to update specific objects, sending highly customized emails with complex business logic, and so on. You can implement the logic to cater to technical aspects such as creating application logs, creating change documents, and so on.

Let’s implement the `RESULT_CALLBACK` method of `ZCL_EMP_FLEX_RUN_APPL_BASE` for demonstration. In this method, the importing parameter `IO_CONTEXT` contains the reference to the scenario context. The container elements can be read using methods `get_workflow_container()` and `get_task_container()` for the workflow and task containers, respectively.

Using method `get_result()` of the importing object reference `IO_RESULT`, you can fetch the result of the current task of the workflow instance, such as approved or rejected based on the nature of the work item execution (positive/negative), the agent/actor of the user decision task, any notes added during approval, and so on.

Using these values at runtime, you can implement the required business logic. The sample implementation is shown in Listing 15.1.

```
  METHOD if_swf_flex_ifs_run_appl~result_callback.
*/ Get Result
    io_result->get_result(
      RECEIVING
        ry_result = DATA(ls_result) ).

*/ Get Container - The Values of Application Object at runtime
    io_context->get_workflow_container(
      RECEIVING
        ro_container = DATA(lt_wf_container) ).


    io_context->get_task_container(
      RECEIVING
        ro_container = DATA(lt_task_container) ).

*/ Update Custom Tables if any
*   ...

*/ Call BAPI or other logic
*   ...

*/ Application Logs if needed
*   ...

*/ Set Workflow Outcome - will be used by other processing like Email Templates
    ev_outcome = COND #( WHEN ls_result-nature = 'POSITIVE' THEN 'APPR' ELSE 'REJ'
).

  ENDMETHOD.
```

Listing 15.1 Sample Implementation of the RESULT_CALLBACK Method of the Callback Class

15.2 Business Objects

In today's realm of object-oriented programming, classes and objects play a major role in tying together related entities, properties, and functions. Business objects represent a real-time business entity like a company, material, purchase order and so on. These entities are modeled using ABAP core data services (CDS) views. These CDS views are tied to the flexible workflow scenario and its associated classes using a set of configurations. For example, in the workflow scenario for the overall release of the purchase requisition, leading object `PURCHASEREQUISITION` is assigned to class `CL_MM_PUR_WF_OBJECT_PR` and corresponding CDS view `I_PURCHASEREQUISITIONAPI01` (see Figure 15.7). In the following sections, you'll learn how to maintain business object types and object node types, as well as how to link them with CDS views that will be consumed by flexible workflows.

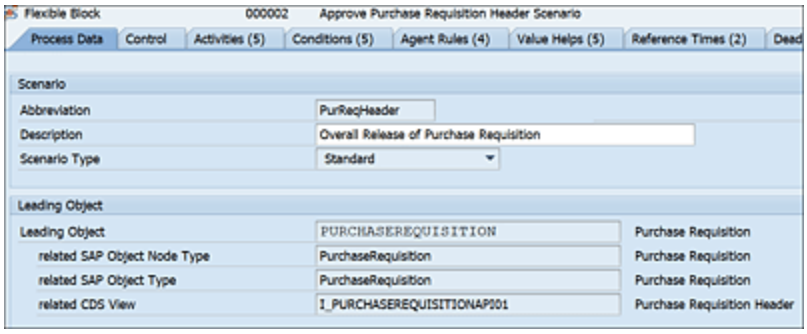


Figure 15.7 Leading Object and Related CDS View for Purchase Requisition

In our use case, we have a sample custom CDS view, `ZI_EMPLOYEE_REQ`, on top of underlying tables, for example, a custom table that represents the entity model for the

application object and is associated to the leading object employee. This CDS view contains fields such as **Personnel Number**, **Employee Name**, **Request Number**, **Created On**, and so on.

15.2.1 Maintain Business Object Type (V_BO_TYPE)

The SAP object type is a generalization of business object types, which has been proposed as the central entity in the SAP S/4HANA meta model. This content will be available within the SAP S/4HANA runtimes for consumption by various frameworks such as business event handling, extensibility, SAP Fiori reuse services, and so on. The SAP object type is a unique representation of the business entity.

Go to view maintenance using Transaction SM30, enter the **Table/View** name as “V_BO_TYPE”, and click the **Edit** button to open this view in change mode. Click the **New Entries** button to create a new entry for our use case, as shown in Figure 15.8. **SAP Object Type** is a character string without spaces. **Object Type Name** can be any meaningful string.

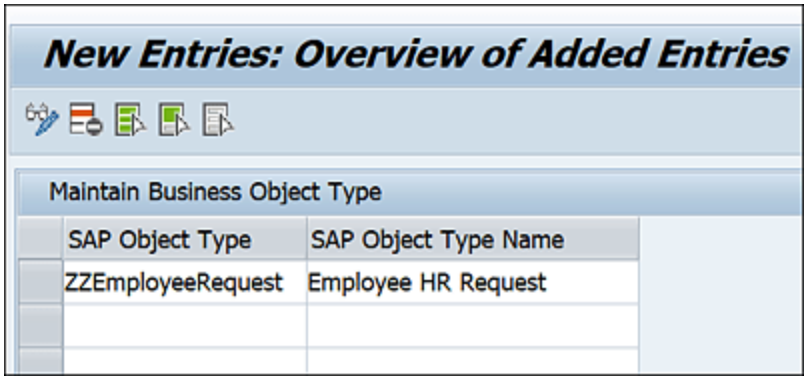


Figure 15.8 Maintain the Business Object Type

15.2.2 Maintain Object Node Type (SBO_V_NODETYPE)

In this Customizing activity, you maintain the object node type for the object type. Each object type is associated with a set of object node types. The object type groups various nodes. There is a dedicated root node for each object type.

Go to Transaction SM30, and maintain entries in view `SBO_V_NODETYPE`, as shown in Figure 15.9. Open this view in **Edit** mode, and click the **New Entries** button. **SAP Object Node Type** is a character string without spaces, and **SAP Object Node Type Description** is a meaningful description for it. Set the **Root Indi** field, which indicates the current object is the root entity. Provide the **SAP Object Type** as "ZZEmployeeRequest" created in the previous step. In addition, maintain the workflow class representation (or business object repository [BOR] representation as needed). In our case, map custom class `ZCL_EMPLOYEE_REQUEST_WF`.

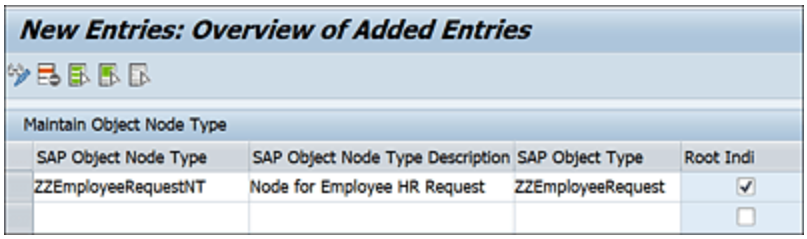


Figure 15.9 Maintain Business Object Node Type.

15.2.3 Maintain Core Data Services View (V_SBO_NODE_CDS)

The leading business object, which represents a business entity in real life, has an associated CDS root entity. This

forms the base for the entire entity model, which relates to different entities.

In this Customizing activity, you'll maintain such CDS views for the corresponding SAP object node types. Open the view maintenance for `V_SBO_NODE_CDS` via Transaction SM30, and add the new entries as shown in Figure 15.10 for our use case. Here, **SAP Object Node Type** is "ZZEmployeeRequestNT" (same as the one created in Section 15.2.2), **CDS Type** is **Representative View**, and **CDS View Name** is "ZI_EMPLOYEE_REQ".

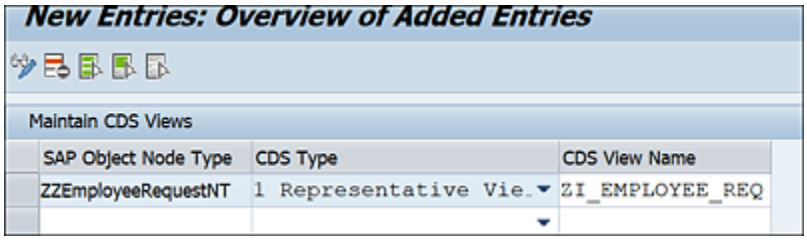


Figure 15.10 Maintain CDS Views

15.2.4 Maintain Object Representation

In this Customizing activity, you maintain the different object representation for the object types. The object representation can be a BOR object ID, class, behavior definition, SOA number, or key fields of an object. Maintain the object representation as custom workflow class `ZCL_EMPLOYEE_REQ_WF`, as shown in Figure 15.11 by the view maintenance via Transaction SM30 for view `V_BODEF`. For this, create a new entry where the **SAP Object Type** is "ZZEmployeeRequest" (same as the one created in Section 15.2.1), **Rep Type** (representation type) is **Class**, and **SAP Object Representation** is "ZCL_EMPLOYEE_REQUEST_WF".

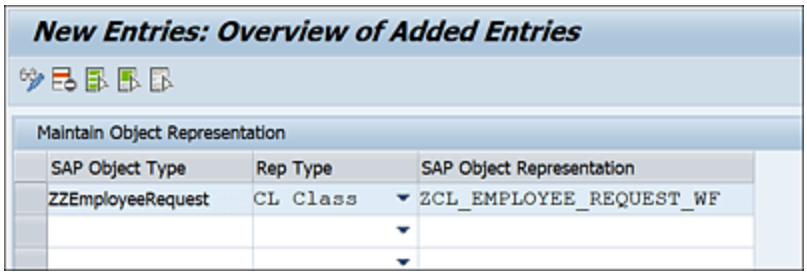


Figure 15.11 Maintain Object Representation

Note

All of these configurations are available in the SAP Customizing Implementation Guide (Transaction SPRO) at menu path **ABAP Platform** • **Application Server** • **Business Management** • **SAP Object Type Repository**.

Once all of these configurations are completed, the class (or business object) for the leading object, the object representation, and the corresponding CDS views are linked to each other and will be available for further consumption.

15.3 Scenario Development

The workflow scenario is the soul of flexible workflows. They provide the skeleton for a flexible workflow. In other words, flexible workflow uses the building blocks provided by its corresponding workflow scenario. Each workflow scenario consists of a *flexible block* that contains necessary components required by flexible workflow: context element, process data, control data, agent rules, activities, conditions, and so on.

Context elements are a kind of variable used by flexible workflow. *Process data* and *control data* hold the different classes that form the foundation of flexible workflow execution. *Activities* contribute different tasks (background and dialog) to be executed, and *agent rules* provide different methods of agent determination available for that flexible workflow for the actionable dialog tasks. As the name suggests, *conditions* provide the preconditions that can be used by flexible workflow for validation before executing any specific task or the workflow itself.

We'll go through each of these components and other related artifacts in detail in this section.

15.3.1 Context Element

Context element or scenario context is the terminology for workflow container in flexible workflows. In simple terms, these are placeholders or containers for a set of variables and structures that can hold the business data (values) at

runtime, that is, during workflow execution. The elements of a scenario context are described by the ID, data type reference, and other properties (e.g., import, export, and mandatory). The scope of these scenario context elements is for the entire life span of that workflow instance. Every scenario context has some default system elements and control parameters, as well as the elements that store the actual business data. Figure 15.12 shows the **Scenario Context** area in the left side pane of the workflow scenario builder from Transaction SWDD_SCENARIO.

The importing context elements hold the value that is passed when a workflow instance is started. They must either be defaulted with the initial value or passed from the calling application through the triggering event. The exporting elements are used by calling workflows for nested workflow calls. These features remain the same as the workflow container elements in classical workflows.

In our use case for demonstration, we've created context element `Employee`, which is an object of custom class `ZCL_EMPLOYEE_REQUEST_WF`. Open Transaction SWDD_SCENARIO. Double-click on the text **<Double-Click to Create>** under **Scenario Context**, and create context elements with **Element** name "Employee", enter a **Name** for it, and provide a meaningful description in the **Short Descript.** field, as shown in Figure 15.12. The **Object Type** of this element will be **CL ABAP Class**, and **ZCL_EMPLOYEE_REQUEST_WF** will be the corresponding class assigned to it. As you can see, these steps are similar to the workflow containers in classical workflows.

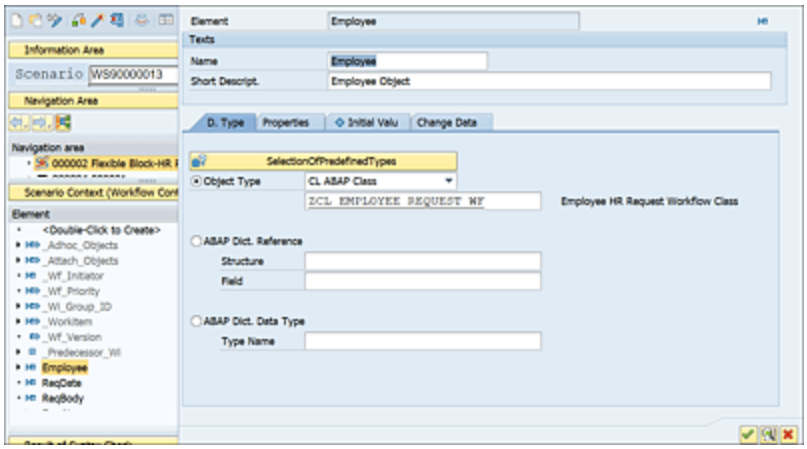


Figure 15.12 Scenario Context in the Workflow Builder

15.3.2 Process Data

The **Process Data** tab of the flexible block is the main configuration area for a workflow scenario. Figure 15.13 shows a sample view of process data for a new workflow scenario creation in Transaction SWDD_SCENARIO. The important fields here are as follows:

- **Abbreviation**
  The 12-character abbreviation helps users to quickly recognize the workflow scenario.
- **Description**
  This holds a meaningful description that best describes the purpose of the workflow scenario.
- **Leading Object**
  Leading object represents a business object/entity in the SAP system. It's a mandatory parameter of the entire process, created inside the scenario context, that is, the workflow container. The leading object of the workflow scenario can be a business repository object (Transaction SWO1) or an ABAP class (Transaction SE24). The object

instance of this leading object essentially contains the actual business data of the application (e.g., employee or sale order) at runtime. It contains all the events and methods to be used by other workflow elements as well in each of the steps.

- **Workflow Start Events**
  This section contains the details of the events that will trigger the creation of the flexible workflow instance. Here, you provide the ABAP class/BOR details and its corresponding event. Like classical workflows, the event linkage will be activated from this section, and the event container needs to be bound with the workflow container; the external program will use this container to pass the values into the workflow instance while starting the workflow via the `Start` event.
- **Workflow Runtime Events**
  This section is used to configure the other events of the workflow: cancel workflow and restart workflow. The class event associated will be triggered accordingly.

Let’s create a new workflow scenario. Open Transaction SWDD_SCENARIO, and create a flexible workflow scenario. The screen shown in Figure 15.13 opens for further steps to create the custom workflow scenario.

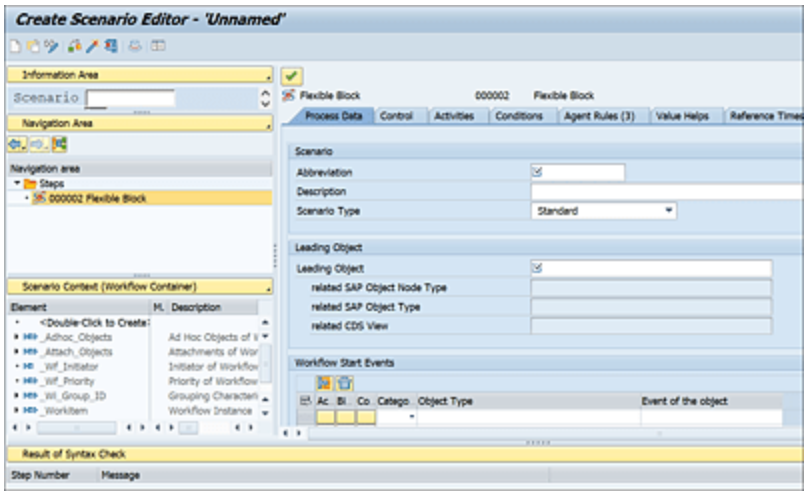


Figure 15.13 Creation of a New Workflow Scenario for the Sample Use Case

In the next screen, you'll enter the basic scenario details such as "Employee Req" for **Abbreviation**, a meaningful **Description** of the flexible workflow, and **Leading Object** details inside the **Process Data** tab of the flexible block, as shown in Figure 15.14. Use the context element `Employee` as we had created earlier of type `ZCL_EMPLOYEE_REQ_WF`, in this **Leading Object** field of the **Process Data** tab.

Now assign the workflow start events and bind the event parameters by clicking on the **Bind** button in the second column of the **Workflow Start Events** table, similar to the event bindings of classical workflows (see Figure 15.15). Therefore, whenever the `START` event of class `ZCL_EMPLOYEE_REQ_WF` is raised, this custom flexible workflow will be triggered. This event can be raised from any user exits, business add-in (BAdI), enhancement points, or any other custom program/class methods per the business requirements.

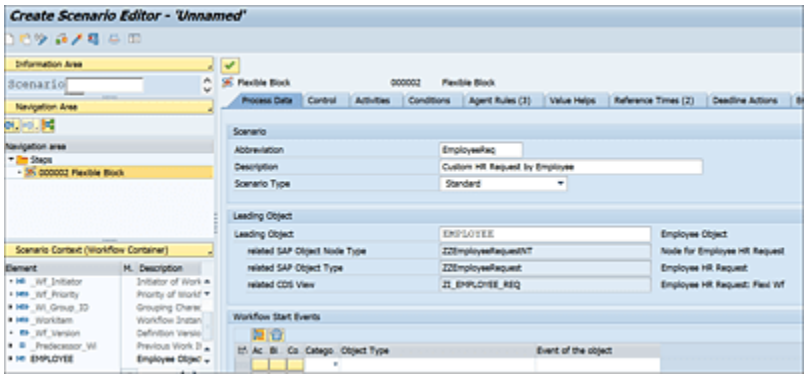


Figure 15.14 Process Data in a Flexible Block of the Sample Workflow Scenario

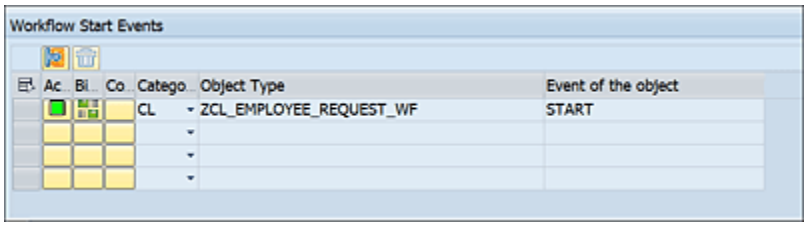


Figure 15.15 Start Event for the Custom Use Case

Additional Information

In some cases, while extending the standard processes, you may need to replace standard workflow with a customized one. Deactivate the standard event linkage and then activate the one with the custom workflow in such cases. Transaction SWETYPV is used to activate/deactivate event linkages.

15.3.3 Control

The **Control** tab (see Figure 15.16) of the flexible block contains the details of the callback class. `CL_SWF_FLEX_IFS_DEF_APPL_BASE` and `CL_SWF_FLEX_IFS_RUN_APPL_BASE` are default standard classes provided by SAP for definition

and runtime data, respectively. Once you create the custom workflow scenario, SAP defaults on the **Control** tab with these standard classes. You can create a custom callback class inheriting `CL_SWF_FLEX_IFS_RUN_APPL_BASE` per customer need, and the developer must handle the runtime behavior accordingly. The custom callback class for runtime data should implement interfaces `IF_SWF_FLEX_IFS_RUN_APPL` and `IF_SWF_FLEX_IFS_RUN_APPL_STEP`. Any custom class for **Definition Data** should implement interface `IF_SWF_FLEX_IFS_DEF_APPL`. More commonly, runtime data callback classes are extended as needed, and the standard settings are kept intact for definition data.

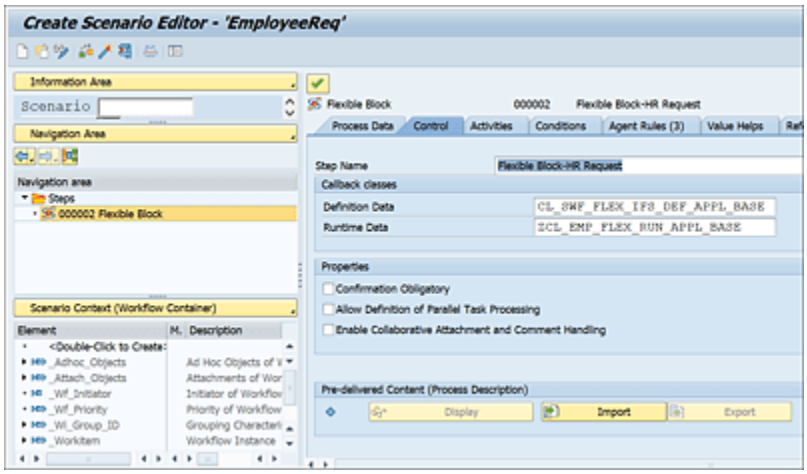


Figure 15.16 Control Tab of the Flexible Block

Now proceed to the control data of the workflow scenario created in the previous section. Assign the callback class for **Runtime Data**. You'll be using the standard class for definition data in our sample use case. Assign custom runtime callback class `ZCL_EMP_FLEX_RUN_APPL_BASE`.

Warning

To prevent inconsistencies in existing instances, avoid changing the callback class name after the activation of the workflow scenario.

15.3.4 Activities

Activities are the collection of predefined steps consisting of user decisions and workflow tasks. These are defined in the flexible block of the scenario definition and are available to model the step sequence of a flexible workflow using the Manage Workflows app.

As shown in Figure 15.17, each activity should have a **Unique Name**, meaningful **Description**, **Activity Type** (user decision, task-based activity, etc.), system-generated **Activity** number, and corresponding **Task** ID. Table 15.2 lists various kinds of activities available in the flexible workflow scenario.

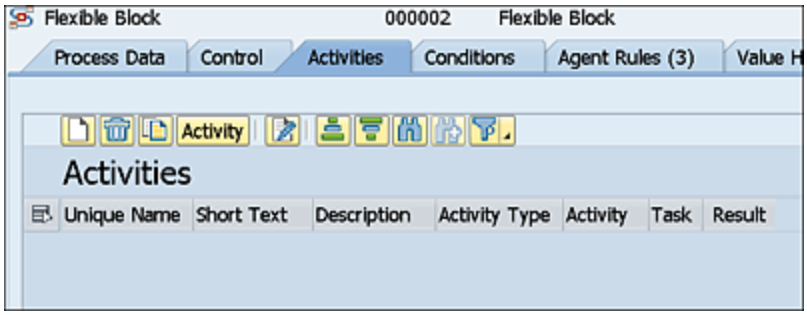


Figure 15.17 Activities in the Flexible Block of the Workflow Scenario

Types of Supported Activities	Description
Activity	Dialog or background task

Types of Supported Activities	Description
User decision	Dialog step with decision options
Review	Special dialog task for review workflows
Extension	Starts a workflow service instance

Table 15.2 Types of Activities Available in a Workflow Scenario

Whenever there is a need for user action, for instance, leave approval, you'll use user decisions/activities with a dialog task. When there is no user intervention in the step execution, use a background activity, which executes the method attached to the task in background mode.

Let's configure the **Activities** tab for a custom use case. Click the **Create** button (refer to Figure 15.17), and in the popup screen that appears, provide a **Unique name** of the activity, enter a **Description**, and choose the type of activity under the **What do you want to create?** section. In our use case, we're creating an activity for a user decision, which corresponds to the approval steps. Figure 15.18 shows the activity creation for our use case.

Click **OK** (the checkmark button) to proceed to configure further inputs of the user decision. As shown in Figure 15.19, provide the **Title** (this will be the default work item text), and then enter the parameters, if any, and the possible decisions for the user decision work item: **Approve** and **Reject**. Tag the decisions with the relevant texts and a

unique **Outcome ID** (“APPR”, “REJ”). Set the approval decision as **POSITIVE** and the rejection as **NEGATIVE** in the **Nature** column.

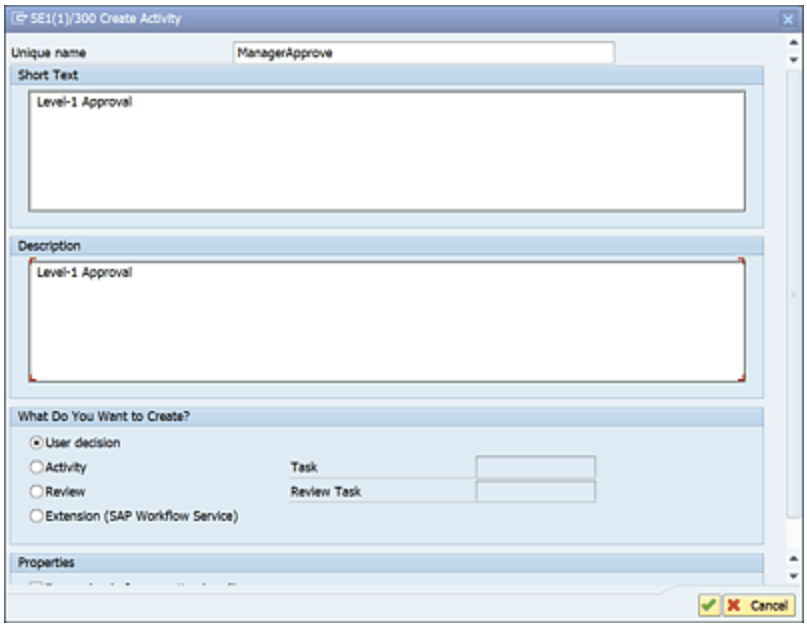


Figure 15.18 Create Activity Screen

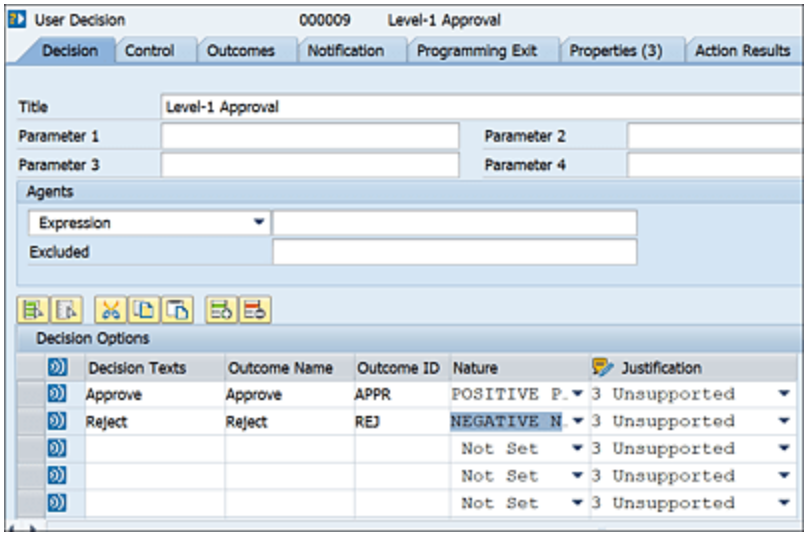


Figure 15.19 User Decision Activity for Line Manager (Level-1)

Similarly create another activity “HRApprove” of the **User decision** activity type, for the second level of approval, as

illustrated in Figure 15.20. The next approver will be an HR administrator who is determined based on the SAP role.

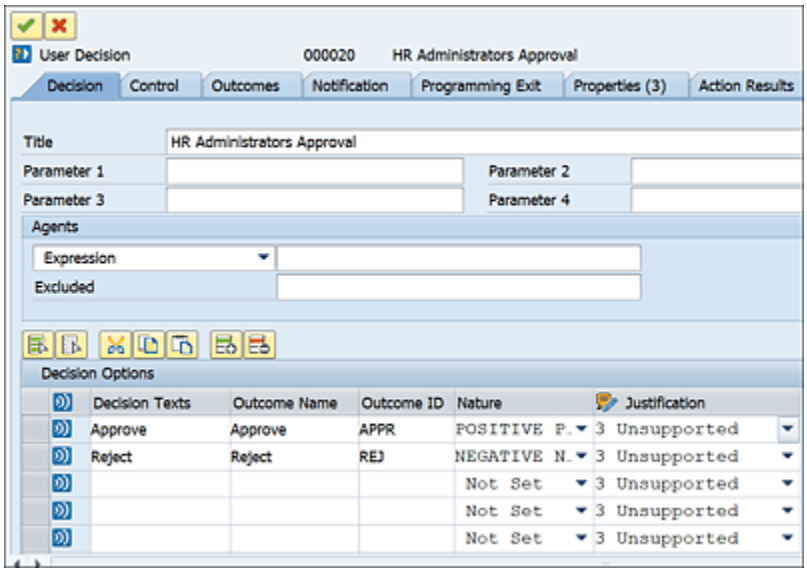


Figure 15.20 User Decision Activity for HR Administrator

Now you have two activities that can be used in the flexible workflow, as shown in Figure 15.21.

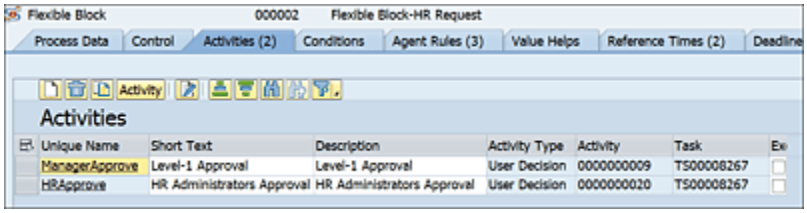


Figure 15.21 Activities Overview Screen in the Flexible Block

15.3.5 Conditions

Conditions define the criteria for execution of the workflow steps. They are checked before execution of a workflow step or entire workflow to validate certain business conditions, and the steps are executed or skipped according to the current data being processed. For example, you can have different workflows configured based on the document type,

company code, net amount and so on for a purchase requisition approval flexible workflow.

Conditions can have from zero to a maximum of two parameters. In the Manage Workflows app, conditions are available from the set of conditions defined in the workflow scenario. The parameter will be passed based on the selected condition. Figure 15.22 gives a quick view of conditions available in the standard approve purchase requisition item workflow scenario as an example.

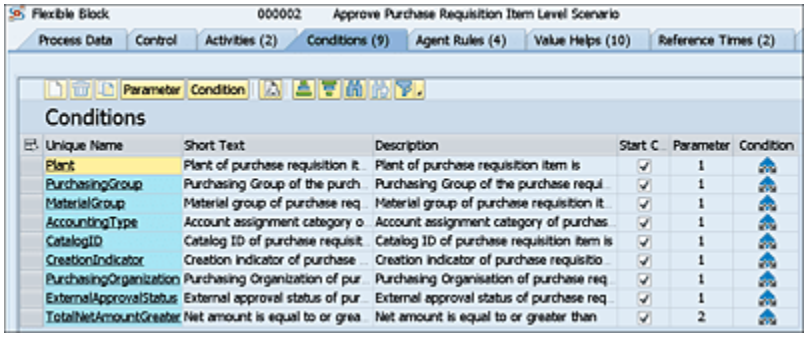


Figure 15.22 Conditions in the Approve Purchase Requisition Item Level Standard Workflow Scenario

SAP provides a set of predefined conditions for most of the standard flexible workflow scenarios. They can be used directly while creating a flexible workflow in the Manage Workflows app.

For implementing custom conditions for individual business requirements, SAP has provided enhancement spot `SWF_PROCESS_WORKFLOW_CONDITION`. This spot contains two BAdI definitions, as follows:

- `SWF_WORKFLOW_CONDITION_DEF` to add custom conditions
- `SWF_WORKFLOW_CONDITION_EVAL` to add evaluation logic for custom conditions

You can implement these BAdIs to extend the standard workflow scenarios as needed.

In the case of entirely custom solutions, new conditions will be added into the custom workflow scenario. To create a condition that will decide the very start of the workflow itself, select the **Start Condition** checkbox. You can add up to two parameters and make them mandatory if needed.

The following types of parameters are supported in workflow scenario:

- Integer
- Decimal
- Long
- String
- Date
- Time
- Boolean

Each parameter will be mapped to a scenario context, and you can define the condition from the parameters in the condition editor using various Boolean operators. Remember that the start condition can use only importing or changing parameters of the scenario context.

15.3.6 Agent Rules

Agent rules are responsible for the agent determination of flexible workflows. These agent rules are used in the Manage Workflows app to assign a recipient to an actionable

step or a notification. The basic concept of agent determination remains the same as for classical workflows. The key difference here is that the recipient of a work item can be configured from the set of agent rules available in the workflow scenario while configuring flexible workflows from the Manage Workflows app, thus enabling more on-the-fly control to key users and business process experts over choosing the approvers.

Every flexible workflow scenario comes with a specific set of standard agent rules, as shown in Figure 15.23. Even when you create a custom workflow scenario, these standard agent rules are automatically added to it. You can keep or remove them as needed.

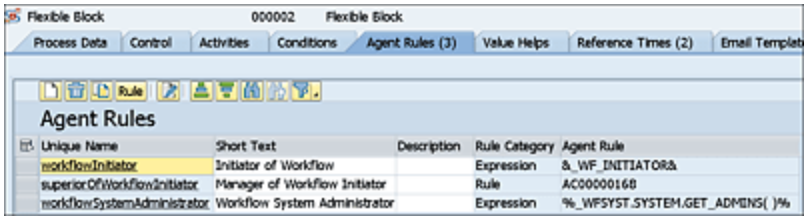


Figure 15.23 Standard Agent Rules in the Flexible Block Defaulted while Creating a Custom Scenario

The standard agent rules are as follows:

- **workflowInitiator: Initiator of Workflow**
  This is an expression with system variable `&_WF_INITIATOR&`.
- **superiorOfWorkflowInitiator: Manager of Workflow Initiator**
  This is basically an agent rule configured in Transaction PFAC. This standard rule will return the reporting manager of the workflow initiator from his HR organizational management structure.

- **workflowSystemAdministrator: Workflow System Administrator**
  This contains the workflow administrator details configured in automatic workflow Customizing (Transaction SWU3) or maintained in administration data (Transaction SWDC_RUNTIME).

Additionally, standard workflows have an agent rule for BAdI-based agent determination. This type of agent rule can be used to enhance/extend the agent determination in a standard flexible workflow with custom approval logic. Each standard workflow scenario can also have other standard agent rules specific to the business process they belong to, for example, accounting object responsible, manager of workflow initiator's manager, and so on.

Apart from these predelivered standard agent rules, you can create your own custom agent rules tailored to specific business needs. The **Rule Category** can be any of the standard agent assignment methods: roles, rules, simple expressions, any SAP user ID, or HR organization parameters similar to the classical workflows.

Note

Whenever the agent rule maintained in flexible workflow fails to return any valid agents while resolving the agent determinations during workflow execution, the workflow instance goes into error. Such erroneous workflows need to be monitored by the administrator.

The flexible workflow frameworks provide an option to handle such agent-resolution errors programmatically by

using `MITIGATE_AGENT_RULE_EVALUATION` of the runtime callback class of the workflow scenario.

In classical workflows, whenever any work item fails due to agent determination, we relied on the notifications sent by the workflow framework to the administrator in the SAP inbox.

In flexible workflows, we're given a provision to programmatically handle such scenarios. Whenever the agent rule is unable to return an agent for any agent resolution of the task at runtime and the custom mitigation programmed in `MITIGATE_AGENT_RULE_EVALUATION` is also failing to return a valid agent, callback method `ON_NO_AGENT_FOUND` is called. The application can be programmed by the customer to handle such situations per need, such as creating application logs, sending notifications to the administrator, and so on.

With respect to our use case, we'll use the standard `superiorOfWorkflowInitiator` agent rule for the first level of approval. For the second level, create a new agent rule by clicking on the **Create** icon. Enter a name and description for the agent in the **Unique name** and **Description** fields in the popup, as shown in Figure 15.24. In the next screen, select the agent type as **AG Role** when prompted, and provide the custom role name as shown in Figure 15.25.

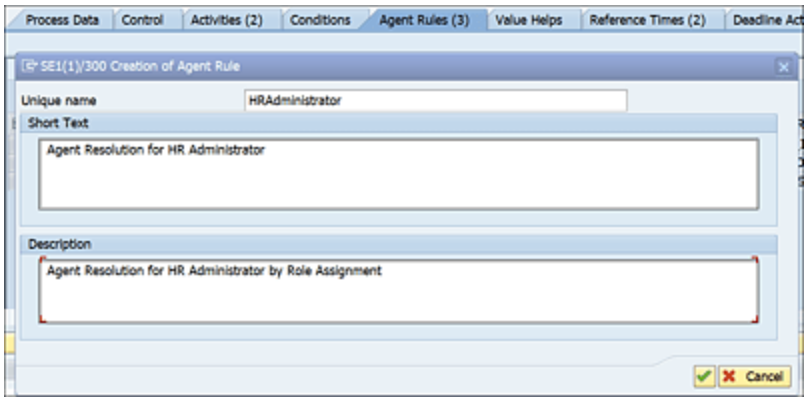


Figure 15.24 Create New Agent Screen

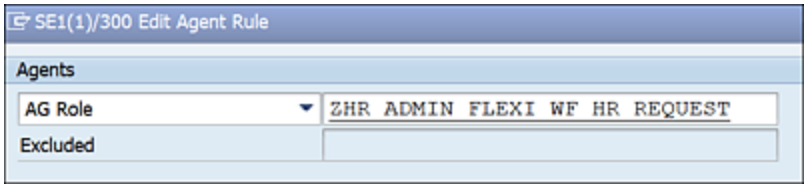


Figure 15.25 Role-Based Agent Rule Configuration

Note

A test role, for example, `ZHR_ADMIN_FLEXI_WF_HR_REQUEST`, needs to be created from Transaction PFCG and assigned to any test user.

Now with all the agents set up, the **Agents** tab looks something like that shown in Figure 15.26.

Figure 15.26 Agent Rules

Note

In our use case, we're using two different activities for user decision. But you can create a single user decision activity that can be reused as well by assigning different agents for two different steps while configuring the flexible workflow in the Manage Workflows app.

15.3.7 Value Helps

SAP has provided standard value helps for parameters in standard flexible workflows, which are used to give suggested values to the business process expert while configuring the parameters for any given condition. These value helps are mapped to an OData service in the backend and fetches data through an `EntitySet` of that OData service. For example, the value helps shown in Figure 15.27 correspond to the conditions that were shown earlier in Figure 15.22.

For custom workflow scenarios, where you have a custom condition to be evaluated, you can reuse any existing standard OData entity inside the custom scenario, for example, a custom condition where you require a list of company codes in the value help. Otherwise, you can go for your own implementation of the OData service as well. The OData service need not be necessarily based on a CDS view as its data source; it can be an ABAP code–based implementation as well.

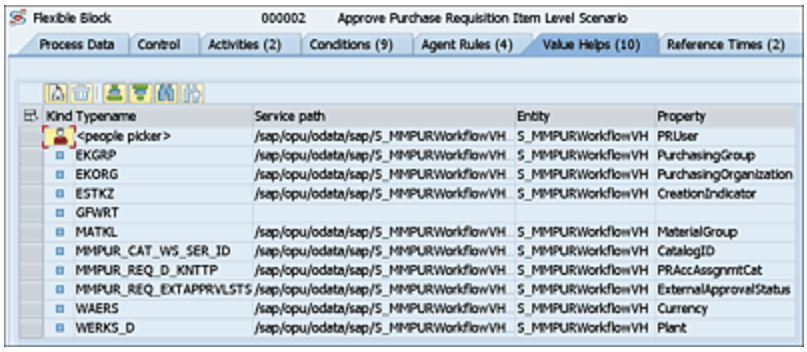


Figure 15.27 Value Helps: Approve Purchase Requisition Item Level Scenario

The mapping of the OData service and corresponding `EntitySet` to the parameter is carried out in the **Parameter** section of the **Conditions** tab itself and then displayed in the **Value Help** tab of the flexible block. Click on the **Help** button on the far right of the parameter, which will open the window to maintain the OData details: **Service Path**, **Entity** set name, and **Property** name in the corresponding entity type, as shown in Figure 15.28.

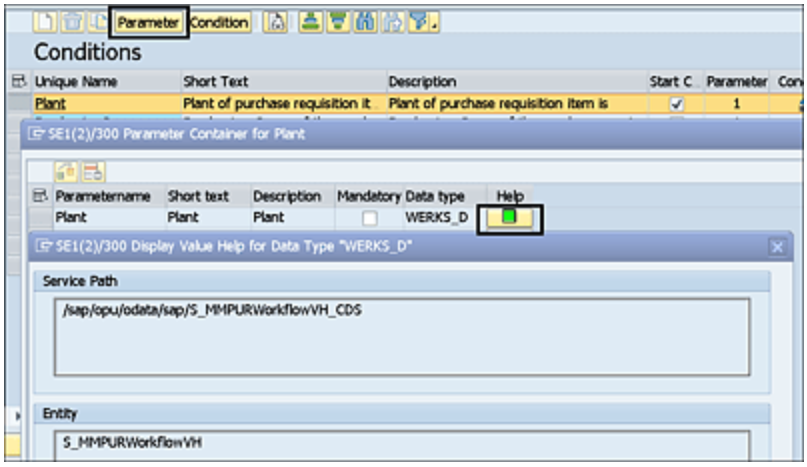


Figure 15.28 Value Help via Conditions

Note

Remember, the **Entity** field should contain the entity set name, not the entity type name itself.

15.3.8 Email Templates

As you saw in previous chapters, in the flexible workflow scenario, SAP has come up with a very flexible and feature-rich functionality for notifying users over email—email templates. These notifications can be sent when there are any new work items pending their actions in the SAP inbox (the My Inbox app) or when a request is approved/rejected. Moreover, maintaining the email templates can be performed by business process experts either with plain text or HTML formatting via the Maintain Email Templates app.

In the Maintain Email Templates app, you use a set of predelivered email templates that will be triggered at predefined actions. Whenever you need to extend the email templates or create a new one for a custom flexible workflow, you'll copy he predelivered template `SWF_CRT_NOTIFY_RECIPIENTS`.

Apart from this, you can also create custom email templates which can be used for other events that aren't part of the out-of-the-box solution. Such functionality is configured in the **Email Template** tab of the flexible block. We can configure the email functionality for the following use cases:

- **Workflow End**
- **Workflow Cancellation**
- **Workflow Restart**
- **Step Deadline**
- **Exception Step Deadline**

These custom email templates fetch the data for dynamic values in the email body from custom CDS views. The developer needs to create a CDS view in Eclipse ABAP Development Tools (ADT) that provides leading object data depending on the work item ID (`WorkflowTaskInternalID`). This custom CDS should join standard CDS view `I_WorkflowTask` with `I_WorkflowTaskApplObject` and again with the CDS view for your application (generally, a custom CDS view for the application object). The join with application object CDS will happen based on `SAPBusinessObjectNodeKey` values.

Following are the prerequisites before creating the email templates for the flexible block:

1. The custom CDS joining the CDS views of work items and application object as described earlier should be ready. This CDS must have `WorkflowTaskInternalID` as the key field, for example, `ZI_WF_EMP_REQ` in our use case. Refer to Listing 15.2 for a sample code snippet of ABAP CDS view `ZI_WF_EMP_REQ`.
2. Redefine method `RESULT_CALLBACK` of the callback class, for example, `ZCL_EMP_FLEX_RUN_APPL_BASE` in our use case. Set export parameter `EV_OUTCOME` of method `RESULT_CALLBACK` depending on the process result or status of your leading business object.
3. The email subject and body should be created in SAP email templates. This template should use CDS view `ZI_WF_EMP_REQ`. Now the work item variables and the application object's values at runtime can be populated into the email body using the CDS view, as shown in Figure 15.29.

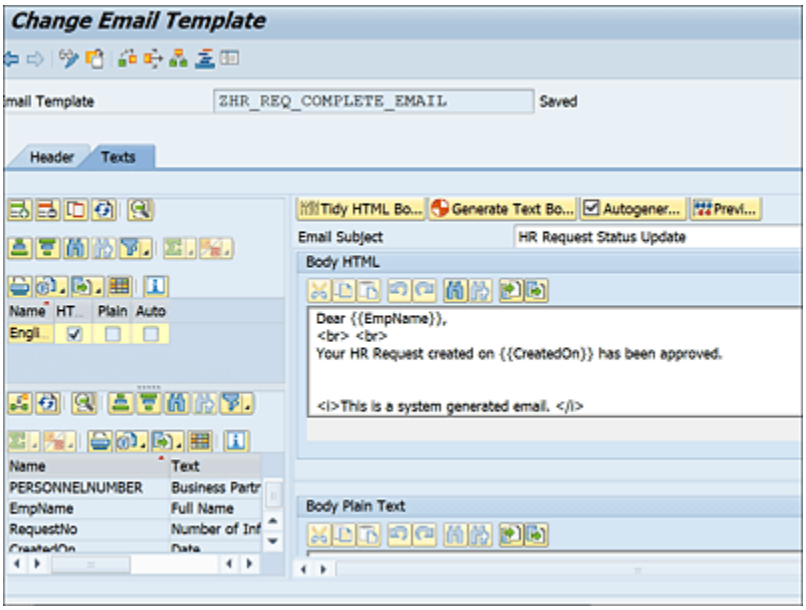


Figure 15.29 Email Templates Maintenance

Note

To maintain email templates, go to the ABAP Workbench (Transaction SE80), right-click on the relevant package, and select **Create** • **Others** • **Email Template**.

```
define view ZI_WF_EMP_REQ
  as select from  I_WorkflowTask as WfTask
  left outer join I_WorkflowTaskApplObject as WfTaskAppObj
  on  WfTaskAppObj.WorkflowTaskInternalID = WfTask.WorkflowTaskInternalID
  and WfTaskAppObj.WorkflowObjectRole     = '01'
  left outer join ZI_EMPLOYEE_REQ as EmpReq on EmpReq.PERSONNELNUMBER =
WfTaskAppObj.SAPBusinessObjectNodeKey1
   {
   key WfTask.WorkflowTaskInternalID,
   key WfTaskAppObj.WorkflowTaskObject,
    ….

    /* Applciation Object */
   EmpReq.PERSONNELNUMBER,
   EmpReq.EmpName,
   EmpReq.RequestNo,
   EmpReq.CreatedOn,
   /* Associations */
   WfTask._TaskApplicationLeadingObject,
   WfTask._TaskApplicationObject,
```

```
  ….. .
}
```

Listing 15.2 Custom CDS for Workflow Email Template

Now using the **Wizard** option, you can set up the email templates for the flexible workflow scenario. In this wizard, choose the event/use case for which email triggering needs to be enabled. Define the outcomes, such as approve, reject, and so on, as needed. Provide runtime class name and email template names when prompted. Once the wizard completes, all the relevant artifacts are linked to the workflow scenario, as shown in Figure 15.30. Now this template can be configured from the Maintain Email Templates app per the requirements.

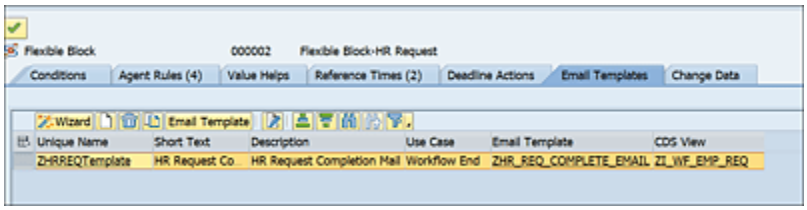


Figure 15.30 Email Templates in the Workflow Scenario

Note that, if you go to the Maintain Email Templates app, this email template *ZHR_REQ_COMPLETE_EMAIL* will be available in the predelivered templates. It needs to be copied and customized as needed. For example, search for custom email template "ZHR_REQ_COMPLETE_EMAIL" in the Maintain Email Templates app, and click on the **Copy** button, as shown in Figure 15.31.

Once copied, you'll see the screen shown in Figure 15.32, wherein you can define the email subject and email content (**Email Subject** and **Body HTML** fields, respectively). In addition, you can use the runtime values of the application

object from the **Show Data Fields** button, which brings the variables from the CDS views you created. The variables (CDS fields) can be identified with double braces, for example, {{EmpName}}.

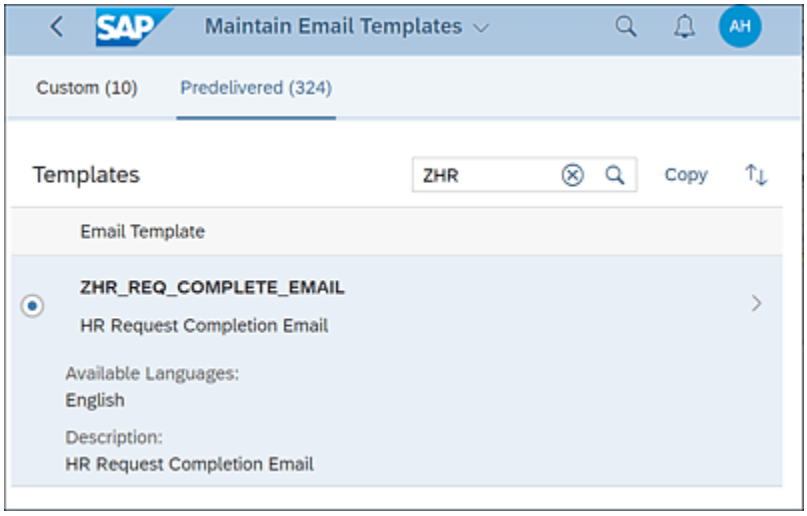


Figure 15.31 Email Template from Custom Workflow Scenario Definition

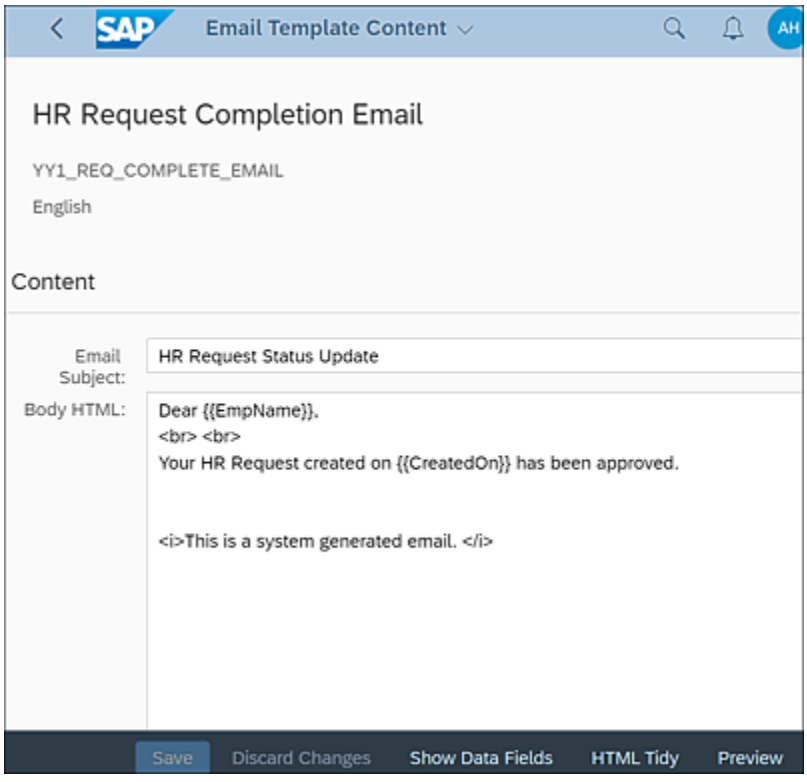


Figure 15.32 Copying Email Content in the Maintain Email Templates App

15.4 Create a Workflow Template Using the Manage Workflows App

Now that you've seen the scenario editor, the elements of the flexible block, and various objects used by the workflow scenario, let's look into creating the flexible workflow based on this workflow scenario. In this section, we'll talk about the Manage Workflows app with reference to the custom scenario and take our use case as a practical example.

The steps to configure the flexible workflow remain the same, irrespective of whether the workflow is the standard SAP-delivered scenario or the custom scenario that you created. As you saw in Chapter 14, Manage Workflows is the key SAP Fiori app behind setting up any flexible workflow. This app provides an easy way to model, set up, and change the flexible workflows as and when needed by the key users, which shifts the duties away from the IT department. This SAP Fiori app allows partners as well as the business process expert to create a specific flexible workflow based on the workflow scenario template per business requirements. Figure 15.33 illustrates a sample view of the Manage Workflows app opened from the SAP Fiori launchpad.

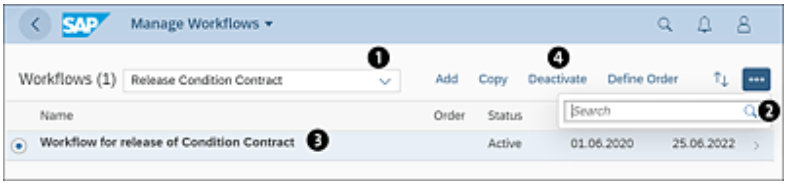


Figure 15.33 Manage Workflows App

As quick recap, let's take a look at the functionalities available on this screen:

❶ From the **Workflows** dropdown, you can see and select the workflow scenarios for which the flexible workflow is to be configured. The dropdown contains an exhaustive list of all scenarios available in the system.

❷ You can search for a specific workflow using the search bar.

❸ Once you select the workflow scenario from the dropdown, you’ll see all the flexible workflows already configured, if any, in the search results table.

❹ Once you select any flexible workflow using the radio button in the search results, different actions become available for that workflow:

- **Add**: Business process owner can add a new flexible workflow for the given scenario using **Add** button. This can happen when we're setting up the workflows for first time, or create an additional workflow with different precondition, or even deactivate old one and create new one with changed business requirements.
- **Copy**: We can copy and existing one into new flexible workflow and do the changes per need using **Copy** button.
- **Deactivate**: When due to some business requirement changes at any point of time, any old workflows are no longer needed, we can **Deactivate** them.
- **Define Order**: We can define the priority order in which the workflows are checked for execution depending on the matching start condition.

You can see the functionalities of the sections within flexible workflows and how they are dependent on the elements of the workflow scenarios in Table 15.3 and Table 15.4.

Section	Field	Purpose	Workflow Scenario Element
Header	**Workflow Name**	Unique name of the flexible workflow.	

Section	Field	Purpose	Workflow Scenario Element
Properties	**Description**	Describe the business purpose/functionality of the workflow.	
Properties	**Valid From**	Start date.	
Properties	**Valid To**	End date.	
Start Conditions	**Start Conditions** (dynamic dropdown)	Select from available conditions. This tells the flexible workflow which business entity needs to be evaluated during runtime.	Conditions
Start Conditions	**Start Conditions** (text box appears on screen dynamically)	Provide the value to be compared. The flexible workflow is executed only if this value tallies with the runtime value for the same business entity (e.g., company code).	Value helps

Section	Field	Purpose	Workflow Scenario Element
Step Sequence	**Steps**	It can be any of the activities to be executed by the workflow—a background task or user decision.	Activity
Step Sequence	**Recipients**	The agents who will be processing the work item of the given activity in the **Steps** column.	Agent rule
Step Sequence	**Step Conditions**	Condition to be checked before executing that particular step (activity).	Conditions

Table 15.3 Add New Workflow

Section	Field	Purpose	Workflow Scenario Element
Header	**Step Name**	An optional step name.	

Section	Field	Purpose	Workflow Scenario Element
Header	**Step Type**	This dropdown will display all available activities from the workflow scenario.	Activity
Recipients	**Role**	This dropdown contains all available agent rules that can be recipients of that work item and need those resolved agents' actions for their completion.	Agent rules
Recipients	**Step to be completed by**	You can set whether all the agents resolved by the agent rule or any one of them is enough to complete the work item.	
Step Conditions	**Step Conditions** (dynamic dropdown)	Select from the available conditions.	Conditions

Section	Field	Purpose	Workflow Scenario Element
Step Conditions	**Step Conditions** (a text box appears on screen dynamically)	Provide the value to be compared. The given step/activity is executed only if this precondition is met.	Value helps
Deadlines		This is used for deadline monitoring and to send reminders of pending actions.	Reference times

Table 15.4 Add New Step inside the Step Sequence

Now, choose the custom workflow scenario **Custom HR Request by Employee** from the dropdown menu (see Figure 15.34), and click **Create**.

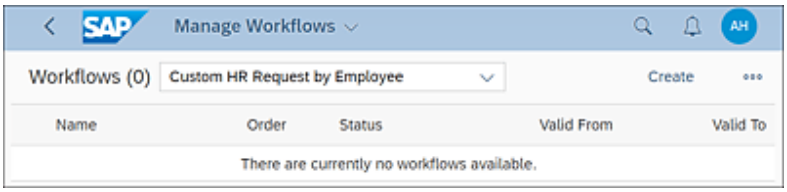


Figure 15.34 Create New Workflow

Enter the **Workflow Name** and a meaningful **Description** of the flexible workflow in the subsequent screen, as shown in Figure 15.35.

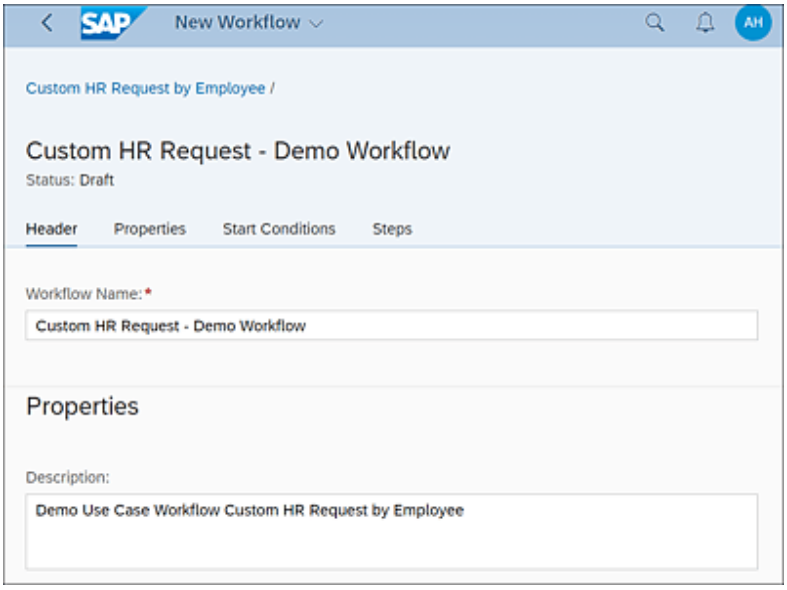


Figure 15.35 Properties of the Custom Flexible Workflow

Navigate to the **Steps** tab. Click on the **Create** button in this section to add a new step from the custom workflow scenario, as shown in Figure 15.36.

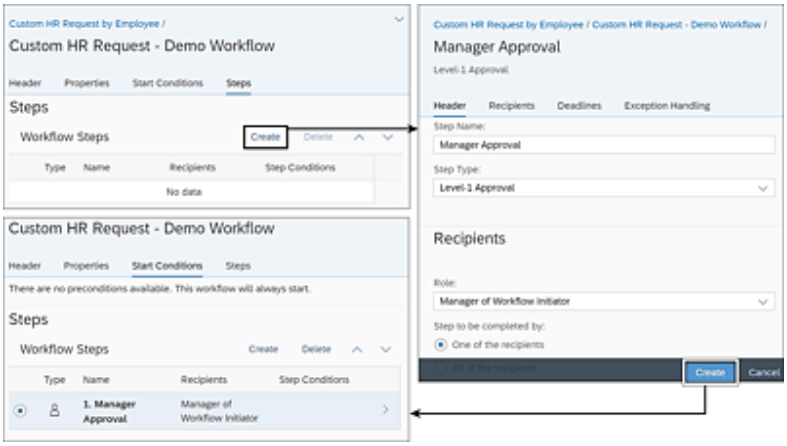


Figure 15.36 Manage Workflow: Create Steps for the Custom Use Case

In the next screen, select the **Step Type** (from the dropdown of different activities in the workflow scenario) and the recipient's role of the work item into the **Role** dropdown under the **Recipients** heading (which contains the agent

rules defined in the workflow scenario). Click the **Create** button at the bottom of the screen.

For the first level of approval, choose the recipient role as **Manager of Workflow Initiator**. Similarly, create another step for the second-level approval, and assign the **HR Administrator** as the recipient role (see Figure 15.37). As you can see, this time, the custom agent rule—**Agent Resolution for HR Administrator**—is chosen in the **Role** dropdown.

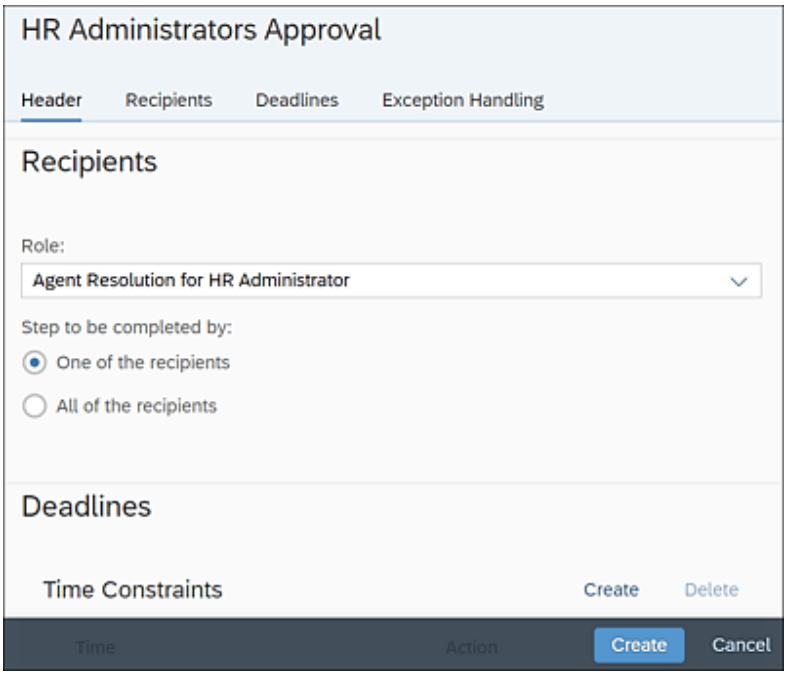


Figure 15.37 Agent Assignment in Flexible Workflow

Note

The manager of workflow initiator is determined from the HR organizational reporting structure, so the corresponding manager's details need to be maintained in the organizational management infotype of the employee accordingly.

Save the flexible workflow and navigate back. Now the workflow will be in **Draft** status. Select the workflow, and click **Activate** (see Figure 15.38).

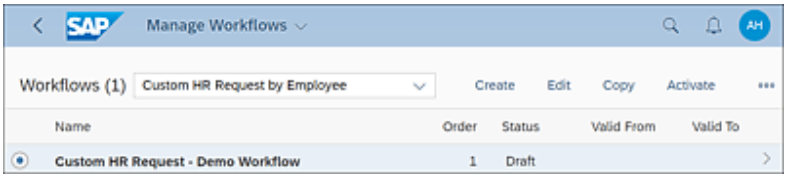


Figure 15.38 Activate the Draft Workflow of the Sample Use Case

Now let's set up the email notifications. Open the Maintain Email Templates app. Navigate to the **Predelivered** tab. Search for standard template "SWF_CRT_NOTIFIY_RECIPIENTS", select the template, and click **Copy** (see Figure 15.39).

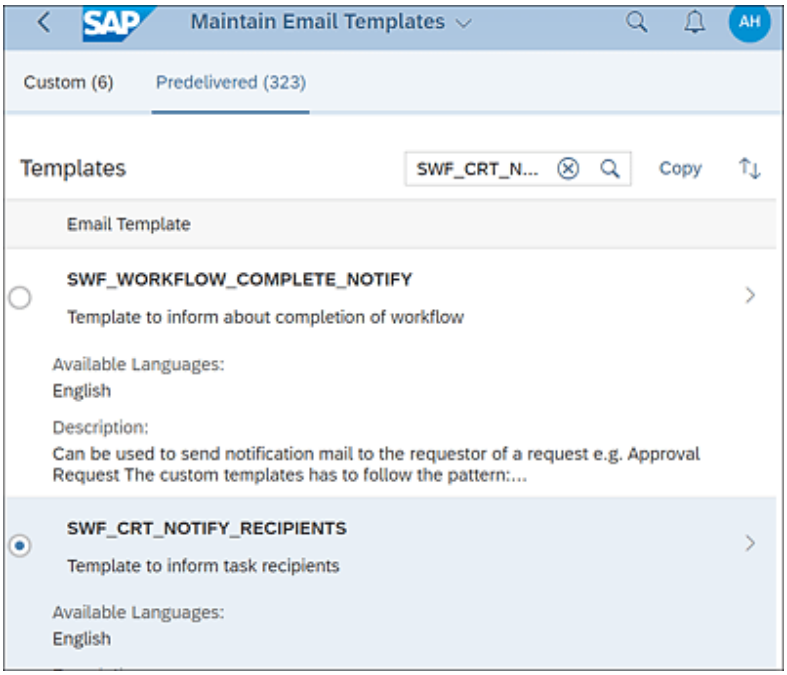


Figure 15.39 Copying the Standard Email Template

Change the name of the template in the **Email Template** field in accordance with the naming conventions to be used, as mentioned in previous sections. In our custom use case,

90000013 is the scenario ID, 9 is the step number for level-1, and 20 is the step number for level-2 of the approval hierarchy. So, the email template name looks like that shown in Figure 15.40. For the **Name** field, you can enter any meaningful description.

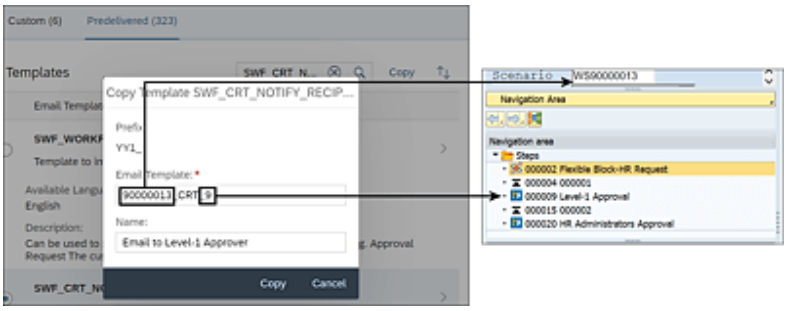


Figure 15.40 Naming the Custom Email Template

Figure 15.41 shows the actual email body while navigating inside the email template. Now the flexible workflow is completely set up and is ready to be used for approval processes.

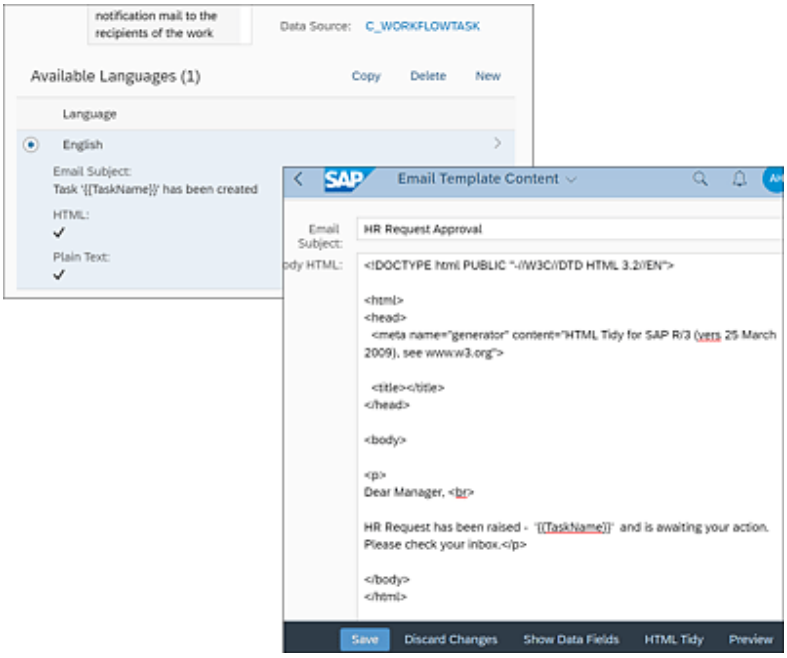


Figure 15.41 Email Templates Subject and Body Contents

15.5 Initiating the Custom Flexible Workflow

In real-time scenarios, the custom flexible workflow will be triggered by invoking the corresponding workflow event from any user exits, BAdI implementations, and so on within any transaction, application, or custom program of any business process per your business needs. Here, you'll create a dummy custom program to raise the sample use case workflow event. Inside the custom program, you can use the standard classes and methods available to invoke the corresponding workflow events, which in turn triggers the flexible workflow.

The high-level overview of the approach to call the workflow event for the custom HR request use case is as follows:

1. Create an instance of the custom event container using static method `get_event_container` of class `cl_swf_evt_event`.
2. Set the event containers, for example, date `IV_DATE`, using the `SET` method of the preceding instance. Set all relevant event parameters as needed.
3. Call method `raise` of class `cl_swf_evt_event` and also pass the object key. For example, Listing 15.3 shows a sample code snippet in ABAP language that calls the `raise` method of class `cl_swf_evt_event` to trigger custom event `START` of custom class `ZCL_EMPLOYEE_REQUEST_WF` to pass the event parameters via object instance `lo_event_parameters`.

```
CALL METHOD cl_swf_evt_event=>raise
          EXPORTING
            im_objcateg        = cl_swf_evt_event=>mc_objcateg_cl
            im_objtype         = 'ZCL_EMPLOYEE_REQUEST_WF'
            im_event           = 'START'
            im_objkey          = lv_objkey
            im_event_container = lo_event_parameters.
```

Listing 15.3 Sample ABAP Code for Raising a Custom Workflow Event

Now the entire flexible workflow and its triggering mechanism are available for consumption. Once you trigger the event for the workflow, for example, using the custom program, the flexible workflow is initiated, and you can see the work item awaiting user action in the recipient inbox.

However, even though the workflow will get triggered, you'll need to set up the My Inbox app configurations to be able to view the work item of custom flexible workflows in the app. In the next section, we'll go through the steps for My Inbox integrations.

15.6 My Inbox Integration

Flexible workflows use the same My Inbox app and underlying configurations as classical workflows such as the system alias, task gateway configurations, and so on, for work item approvals. So, you need to just maintain the workflow task and decision steps to bring the work item into My Inbox. Similarly, maintain the visualization metadata for passing the workflow runtime parameters to the target application that will open the work item. In this section, we'll discuss maintaining My Inbox–related configurations for a custom flexible workflow. In addition, we'll take a quick look at how the work item looks in My Inbox for our custom HR request use case.

15.6.1 Define Step Names and Decision Options

Go to Transaction SPRO (SAP Customizing IMG), navigate to **ABAP Platform** • **SAP Gateway Service Enablement** • **Content** • **Workflow Settings**, and execute **Maintain Task Names and Decision Options**. As shown in Figure 15.42, maintain the **Workflow ID** and **Step ID** ❶ for the approval, that is, user decision activity. Navigate to **Decision Keys** in the **Dialog Structure** ❷ for each of the steps, and maintain the possible decisions—**Approve** (positive outcome) and **Reject** (negative outcome). You can configure the approval/rejection notes to be **Mandatory** as well. Our custom scenario ID is **WS90000013**, and the step

numbers for the first and second level of approvals are **0000000009** and **0000000020**, respectively ❸.

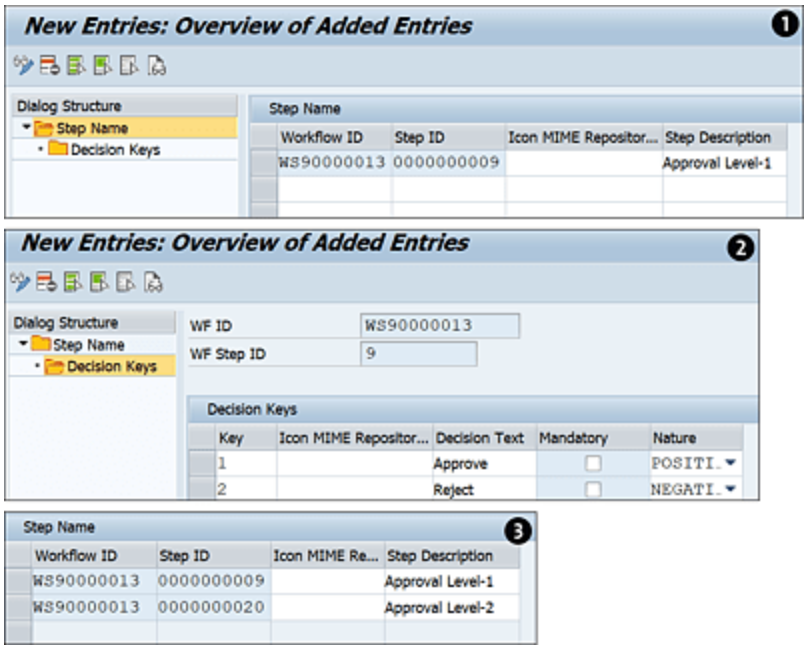


Figure 15.42 Define Steps for My Inbox Approval

15.6.2 Define Visualization Metadata for My Inbox

This customization steps remain the same as those for any classical workflow or standard flexible workflow. As a quick recap, use Transaction SWFVISU or Transaction SWFVMD1 to maintain the visualization parameters. If you're going to use any custom Web Dynpro or SAP Fiori apps to display the work item details, rather than the standard work item viewer, then these settings are needed to pass the required parameters from the work item in the inbox to the target application. The target application can read these parameters, read the workflow containers, and display the content accordingly. Further details on this process were

already covered in Chapter 11, Section 11.2.3, and a brief overview for flexible workflows can be found in Chapter 14, Section 14.4.3, as well.

15.6.3 My Inbox for Custom Scenario Walkthrough

Now the entire flexible workflow, its triggering mechanism, and My Inbox settings are available for consumption. Once you trigger the event for the workflow, for example, using the custom program as described in Section 15.5, the flexible workflow is initiated, and you can see the work item awaiting user action in the recipient inbox.

Figure 15.43 shows that the work item is seen in the first approver's, that is, line manager's My Inbox app along with the corresponding heading and work item texts. The work item texts can be customized with finer details per your business needs.

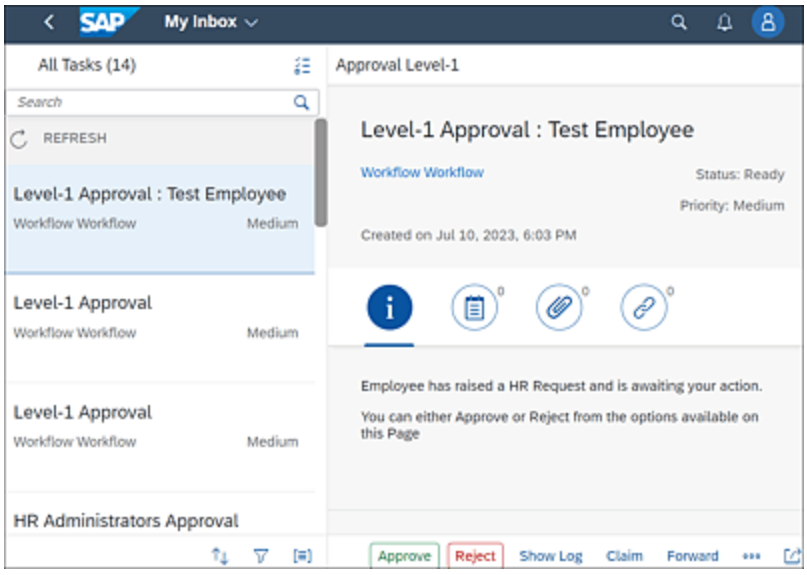


Figure 15.43 Work Item in the First Recipient's Inbox

Similarly, once the work item is executed positively (approved), the work item is received by the second approver (i.e., HR administrator’s My Inbox app), as shown in Figure 15.44.

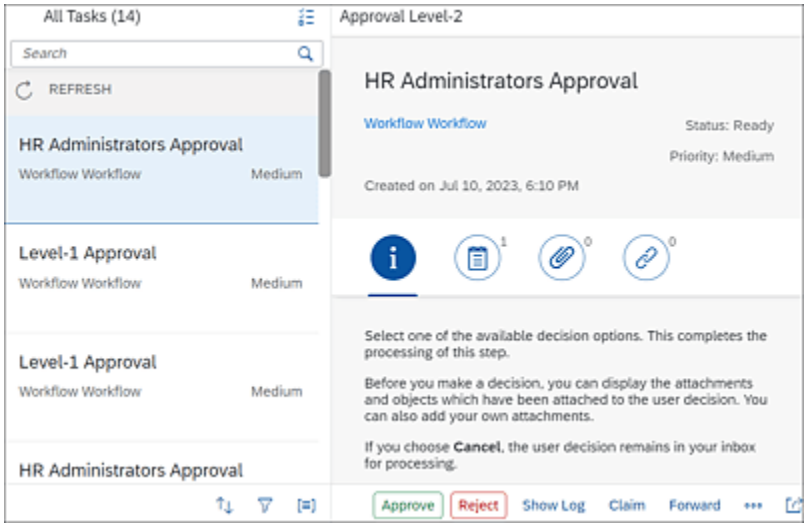


Figure 15.44 Work Item in Second Recipient’s Inbox

15.7 Troubleshooting

In this section, we'll discuss the key points to ponder while troubleshooting any flexible workflow execution. Similar to classical workflows, while executing flexible workflows, you may encounter different kinds of errors—incorrect master data, missing configurations, other technical errors, and so on.

Incorrect master data is one of the biggest contributors to issues reported for flexible workflows in productive systems, be it standard or custom flexible workflows. All the workflow tasks are dependent on the underlying master data of the leading business object for which the workflow is being executed.

Agent determination failure is a common error due to master data inconsistency. We observe errors such as **Agent determination for step failed** or **Error in resolution of rule "ACXXXXXXXX" for step** while analyzing the workflow from the technical logs using Transactions SWIA, SWI1, or SWI2_DIAG. For example, Figure 15.45 shows the technical logs of a work item for the overall release of the purchase requisition flexible workflow in Transaction SWI2_DIAG.

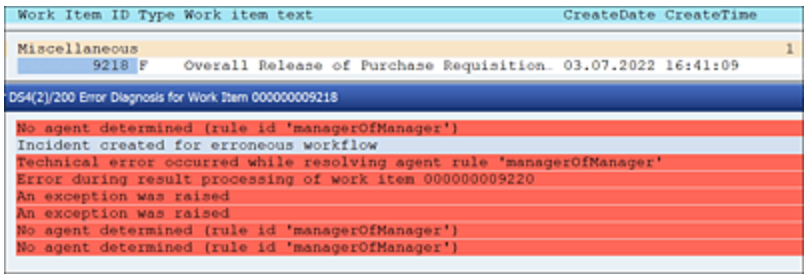


Figure 15.45 Agent Determination Error from Transaction SWI2_DIAG (Diagnosis of Workflows with Errors)

As you can see, the agent with rule `managerOfManager` could not be determined for this work item, so the work item has raised an exception. The following shows the details of errors logged, like which exception has been raised, on which of the work items the exception occurred, etc.:

- Any user activity step requiring a business user's action is highly dependent on the master data, which determines the workflow recipient. If you're using the manager of workflow initiator agent rule or any other HR using the reporting hierarchy, it must be maintained in the HR organizational management–relevant infotypes.
- For role-based agent rules, at least one SAP user must be assigned with the business role.
- For rule-based agent rules, the exceptions in the function module must be handled correctly, and it should always return a valid agent. In nonproductive systems, the replicated use case can be troubleshooted by simulation using Transaction PFAC.
- Similarly, if emails aren't received in the recipient's inbox, the email address of the agent recipient should be maintained. The technical logs can display errors in which the step email couldn't be found.
- If the workflow is successfully triggered and work item can be seen assigned to the intended agent/recipient in the workflow logs (Transaction SWIA/SW1), but the work item is not seen in My Inbox, then you should take the following actions:

- Make sure the user activity tasks are general tasks.
- Check whether the workflow scenario ID and step number are configured in the configuration **Maintain Task Names and Decision Options** (refer to Section 15.6).
- Check Transaction SU53 for any authorization errors, and assign the authorization roles to the user accordingly.
- Check whether no substitutions are maintained in table `HRUS_D2`.
- Use Transaction SWI5 (Workload Analysis) to verify whether the work items are assigned to the user by entering the **Type** as “US” and selecting the **User ID**.
- If any error is observed in the technical logs as **You’re not one of the possible agents of task XXX**, make the task a general one for user activity (dialog work item).

Note

Do keep an eye on authorization checks for workflow batch users as well. In addition, remember that `WF-BATCH` user has been replaced with `SAP_WFRT` in SAP S/4HANA systems.

- If the emails aren’t being sent after a work item is created or approved, the workflow logs need to be analyzed for any errors logged in the **Send Email** step. In the technical details of the step history in Transaction SWI1, the dialog work item log should contain this information.

Some of the common reasons include the email ID of the recipient isn’t available, email template for the flexible workflow scenario isn’t configured, the naming convention of the template isn’t followed, SMTP node configurations in Transaction SCOT aren’t available, and so on.

- A common issue observed with email notifications is the unavailability of email templates in productive systems. Remember to add the email templates into a transport request manually using the Register Extensions for Transport app.
- Sometimes, email notifications may not be sent if the job containing program RSWF_OUTPUT_MANAGEMENT has been suspended.
- Similarly, make sure that the dependent Transaction SPRO configurations for activating the flexible workflow are transported into the target systems before importing the flexible workflow from the Manage Workflow Scenarios app. Otherwise, it may happen that the flexible workflow is imported into the target system, but the corresponding workflow scenario isn’t activated. In such cases, the workflow will fail to be triggered.

Figure 15.46 shows a quick view of the technical logs for the workflows initiated for our sample use case walkthrough. The left side of the figure ❶ shows the **Workitem Id**, execution status (in the **Step History** section), **Task Id** of the work item being executed, **Level-1 Approval**, **Creation Date/Time**, and so on. The right side ❷ shows the log entries for the second dialog work item—**HR Administrators Approval**.

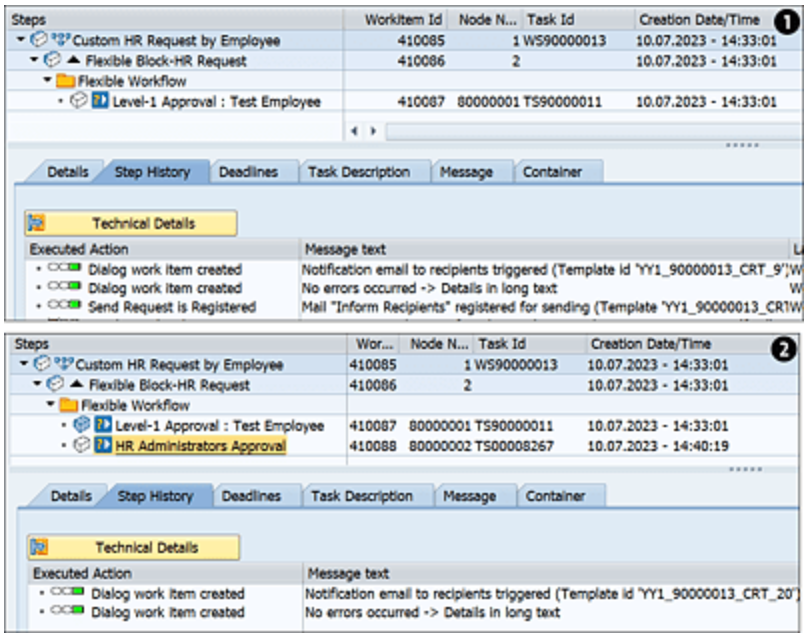


Figure 15.46 Workflow Technical Log for HR Request Use Case

Note

Email templates are a part of the *key user extensibility* concept in SAP S/4HANA. For adding email templates to a transport request, the relevant package must be registered using the Configure Software Packages app, a transport request must be assigned to it, and finally the email template will be added to the package contained in the transport via the Register Extensions for Transport app.

Tip

Here are a few tips to consider:

- As an alternative approach, if the emails aren’t triggered or the agent couldn’t be determined, you can keep a break point in the corresponding hook method of

the runtime class and debug it to find the exact root cause, for example, `GET_APPL_AGENT_RULE_EXECUTION`.

- Remember to check the general ABAP transactions—Transaction SLG1 (Applications Logs) and Transaction ST22 (ABAP Dumps)—as well as workflow-specific tools while troubleshooting.
- If the flexible workflow isn’t getting triggered after performing the intended action, use Transaction SWUS (Workflow Test Tool) to trigger the workflow manually. It will display error details about missing configurations or other errors that stop the workflow being executed successfully.

15.8 Summary

In this chapter, you've seen various technical artifacts required to be built for setting up a custom workflow scenario. We started with the underlying ABAP class developments and business object configurations. Then, we studied the actual workflow scenario development: context elements, process data, different classes used in the workflow scenario, defining various kinds of activities, different approaches for agent determinations, value help, email templates, and so on.

You also saw how business process experts or key users can use this workflow scenario in the Manage Workflows app to create flexible workflows per their business needs. We also set up the My Inbox integration—the last mile of configuration for the entire development.

We walked through a sample use case of a generic HR request process along with each section of this chapter by showcasing how different artifacts are created and how the final output (i.e., the work item) is seen in the My Inbox app.

Even though a predelivered set of flexible workflow scenarios is available, customers often have some highly specific requirements that can't be met by standard flexible workflows. In such cases, developers can choose this path of custom scenario development to fulfill specific business needs.

16 SAP Fiori Applications for Flexible Workflow

A few important SAP Fiori apps can be used in flexible workflows in general, and it's very important to understand those SAP Fiori apps in order to work on flexible workflows effectively. This chapter explains all those important apps and how you can use them in real-time use cases.

In Chapter 14 and Chapter 15, we explained how to activate standard flexible workflows and build custom scenarios, respectively. We've used email templates, agent determinations, and different kinds of notification features in flexible workflows. In this chapter, we'll discuss the prerequisite setups to enable in-app extensions to create email templates, transport these email templates to other systems, and maintain responsibility rules to manage agent determinations using SAP Fiori apps. We'll start by setting up the adaptation transport organizer (ATO).

16.1 Adaptation Transport Organizer Setup

Setting up the adaptation transport organizer (ATO) is one of the prerequisite steps to be carried out in the SAP S/4HANA system before implementing any in-app extensions by using SAP Fiori apps. You must follow these steps to set up ATO (note that this setup is required only in the development system):

1. Open Transaction S_ATO_SETUP (ATO Setup), as shown in Figure 16.1, in your SAP S/4HANA development system.
2. Check if ATO is configured or not. If it's not configured, proceed with setup.

You can find all the setup details in Table 16.1. This one-time setup needs to be carried out in the development system. Otherwise, you'll get an error stating that it needs to be set up.

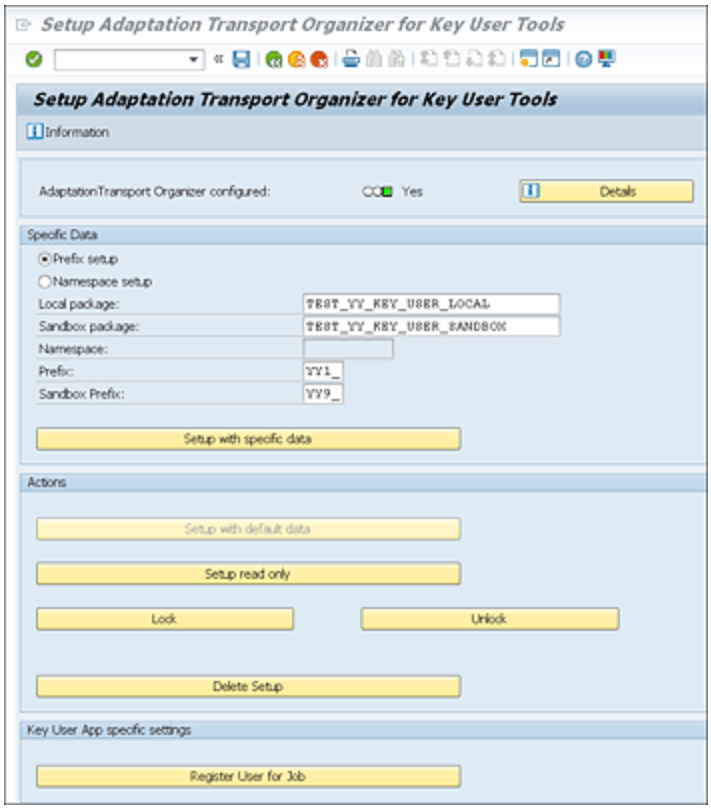


Figure 16.1 Transaction S_ATO_SETUP

Field	Description
Specific Data	
Prefix setup	Prefix for the objects created in the backend via the key user tools, which are connected to the ATO. The suggested prefix is *YY1_*. SAP recommends that enhanced email templates start with *YY1_*. So, if you enter "YY1_" as the **Prefix**, then you can name those email template extensions correctly. We'll provide more details on email templates in the following sections. This prefix is used to create objects that can be transported to follow-on systems in the landscape. For example, as maintained in Figure 16.1, **Prefix YY1_** means that the ABAP object name has the following format: *YY1_<object name>*.
Namespace setup	This is the namespace for all the objects created by using key user extension apps, for example, *<Namespace><Generated object name>*.

Field	Description
Local package	Enter the local package name, for example, “TEST_YY_KEY_USER_LOCAL”. ABAP objects created in the local package will be transported to the target system after assigning them to the transportable package. Don’t start the package name with the “$” symbol. Packages starting with “$”will be deleted during upgrade, as mentioned in SAP Note 2478895.
Sandbox package	Enter the sandbox package name, such as “TEST_YY_KEY_USER_SANDBOX”. ABAP objects located in **Sandbox package** are created locally and will never be transported.
Namespace	Enter the namespace if the **Namespace setup** radio button is selected.
Prefix	Enter a prefix such as “YY1_” if the **Prefix setup** radio button is selected. These objects will be transported to target systems.
Sandbox Prefix	Enter a prefix such as “YY9_” to distinguish them as ABAP objects that much be created locally and not transported to target systems.

Field	Description
Set up with specific data	After entering all the preceding specific details per your project requirements, click on this button to complete the ATO setup.
Actions	
Set up with default data	Click on this button and complete the setup by using SAP suggested data. This option is only available when you do the setup for the first time.
Setup read only, **Lock**, **Unlock**	Use these actions to set up ATO in read-only mode. This is mostly useful in nondevelopment environments to display the key user development items in the related SAP Fiori app.
Delete Setup	This action will delete the existing ATO setup, so use this action with caution.

Table 16.1 ATO Setup Options

16.2 Maintain Email Templates App

Now that you've set up ATO, it can be used to configure and customize all email templates in general. These templates are tightly integrated in a flexible workflow notifications framework.

Open the Maintain Email Templates app, as shown in Figure 16.2. It will display all custom and predelivered templates on the main page. Per the requirement for this use case, we must copy any predelivered template and customize it with our own details.

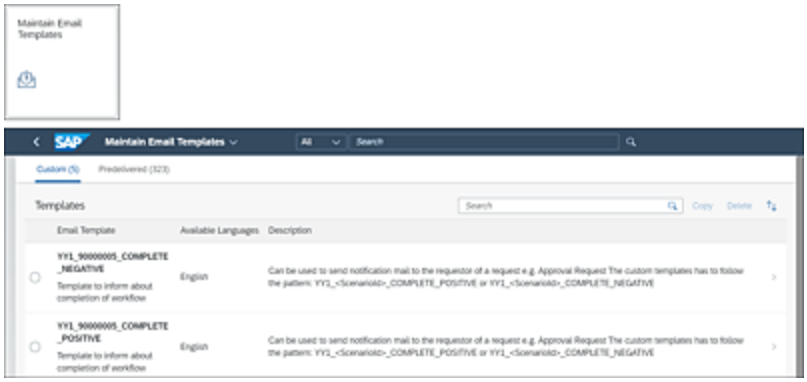


Figure 16.2 Maintain Email Templates App

Let's copy one task notification template for our standard purchase requisition scenario and customize it with our custom details. Copy the predelivered task notification email template `SWF_CRT_NOTIFY_RECIPIENTS` to the customer namespace by following the naming standards discussed later in Table 16.2.

Note

You can get the relevant scenario ID by opening the Manage Workflow Scenarios app, as shown in Figure 16.3. The relevant scenario ID is shown on the top-left side.

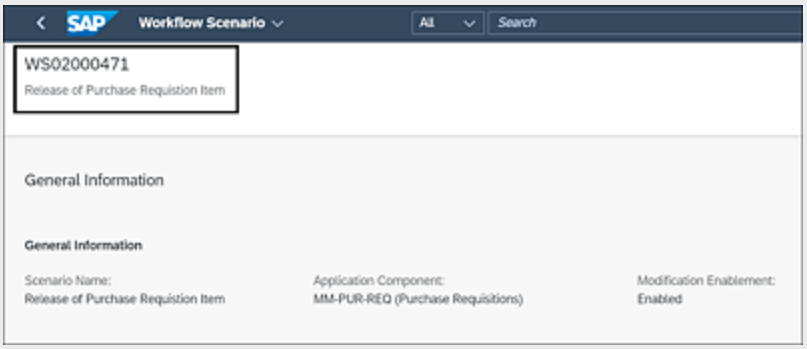


Figure 16.3 Scenario ID

Follow these steps:

1. Go to the **Predelivered** (templates) tab in the Maintain Email Templates app, and search for the standard template. As shown in Figure 16.4, choose the standard template, click on the **Copy** button, and provide the template name per the naming standards provided in Section 16.3.

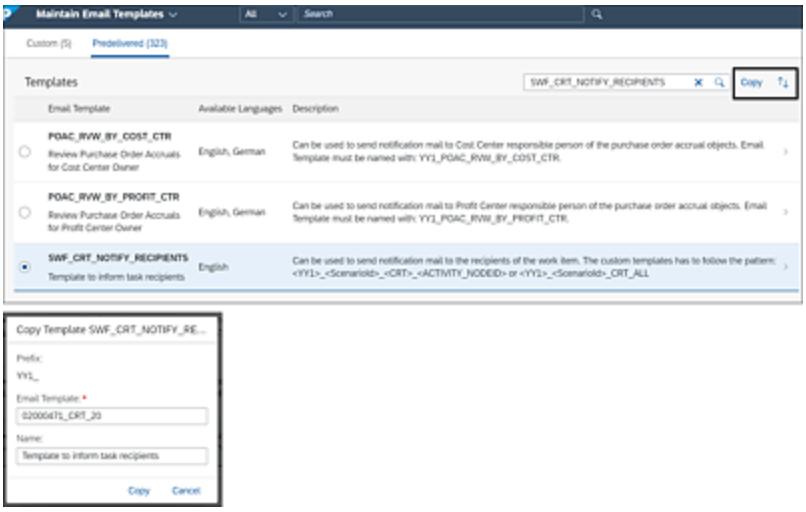


Figure 16.4 Copy Standard Email Template

2. This copied template will now be available in the **Custom** (templates) tab. You can double-click on the copied template to navigate to the email details page. You can change the name and description in this page. Click on the **Change Details** button and a popup will appear where you can make those changes to the **Name** and **Description** fields (see Figure 16.5).
3. Click on the > symbol marked in Figure 16.5 to go to the email **Content** page shown in Figure 16.6. Customize it per requirements, and then click on the **Change Details** button.

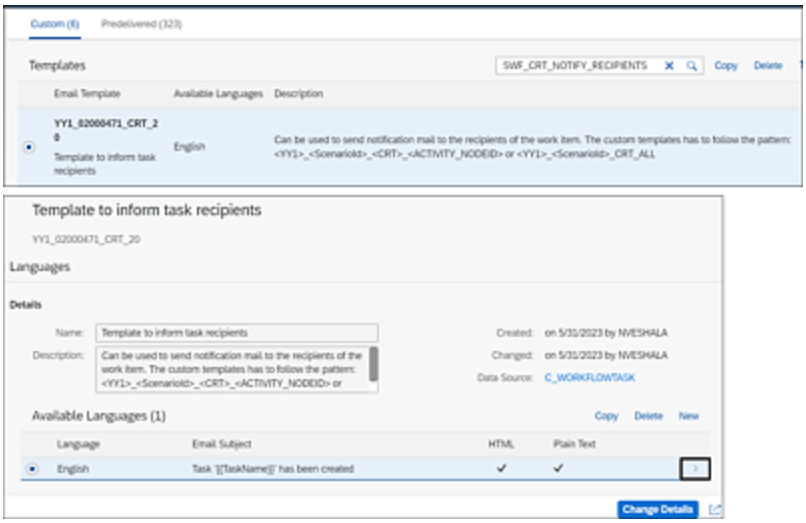


Figure 16.5 Email Template Details

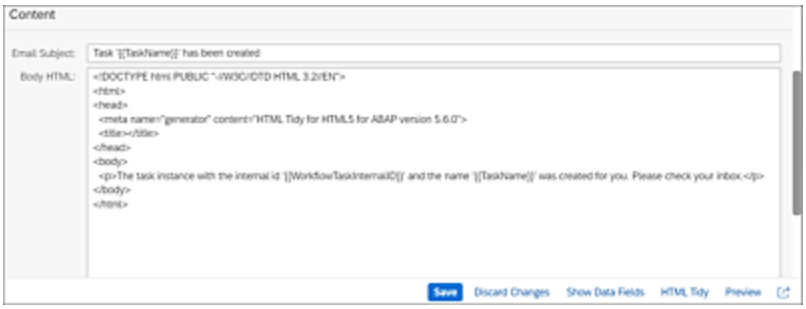


Figure 16.6 Email Content

16.3 Notification Features

Similar to classical workflows, SAP has provided an easy way to send mail notifications during the flexible workflow process. You can send email notifications when a work item gets created to notify agents and request approval/rejection of work items by creating relevant custom email templates, as explained in Section 16.2. Apart from these, notifications can also be sent for final completion/rejection of workflows, deadline monitoring, and so on by using the scenario definition standard framework in Transaction SWDD_SCENARIO.

Open the Maintain Email Templates app, copy the predelivered standard email templates *SWF_CRT_NOTIFY_RECIPIENTS* and *SWF_WORKFLOW_COMPLETE_NOTIF* to the customer namespace by following the template naming standards explained in Table 16.2, and create the templates for the required flexible workflow scenarios.

Template Naming Standard	Example	Details
YY1_<Scenario_ID without WS>_CRT_<StepID>	*YY1_00800173_CRT_32*	Triggers email to recipient immedia after cre the work of the st maintain the temp name (h for step
YY1_<Scenario_ID without WS>_CRT_ALL	*YY1_00800173_CRT_ALL*	Triggers email for dialog st the scen as soon work iter created
YY1_<Scenario_ID>_COMPLETE_POSITIVE	*YY1_00800173_COMPLETE_POSITIVE*	Triggers email to requesto when a w item the created been app

Template Naming Standard	Example	Details
YY1_<Scenario_ID>_COMPLETE_NEGATIVE	*YY1_00800173_COMPLETE_NEGATIVE*	Triggers email to requesto when a w item they created b been reje

Table 16.2 Email Template Naming Conventions

In addition to these standard template customizations, there are a few other email notifications use cases you can create in Transaction SWDD_SCENARIO while building a scenario which are explained in Table 16.3.

As shown in Figure 16.7, you need to go to the **Email templates** tab in the custom scenario in Transaction SWDD_SCENARIO and create an email template for the required use case. So, it's necessary to define the **Use Case** of an email template to limit the selection on the frontend side in the Manage Workflows app to the emails that make sense (e.g., while defining a deadline).

Use Case	Details
Workflow End	At the end of the workflow, the initiator will be informed about the overall result of the approval.
Workflow Cancellation	A notification will be sent if a workflow was canceled.
Workflow Restart	A notification will be sent while restarting a workflow.
Step Deadline	Before an activity is overdue, the possible agents will be notified so they get a chance to process the activity in time.
Exception Step Deadline	This will be a reminder email to notify agents to complete exception handling actions, as per workflow design.

Table 16.3 Email Notification Use Case

As shown in Figure 16.7, different use cases are available in Transaction SWDD_SCENARIO.

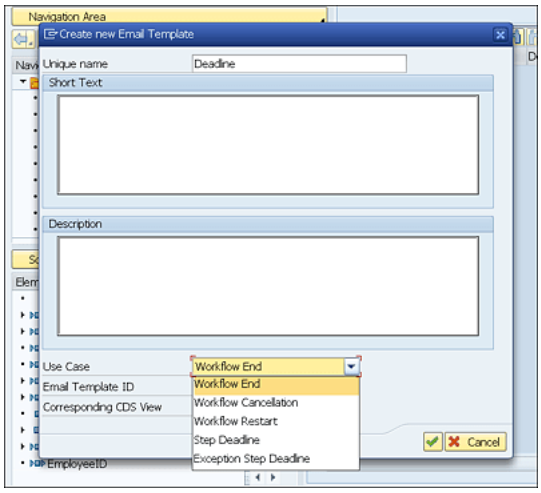


Figure 16.7 Email Notification Use Cases

Let’s create one use case for deadline monitoring, and the same steps will be applicable for all other use cases as well:

1. As shown in Figure 16.8, go to the **Email Templates** tab, and click on the **Create** button, which is marked.

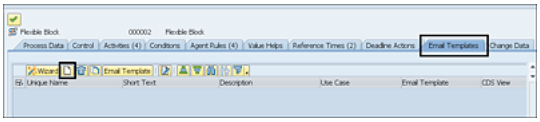


Figure 16.8 Deadline: Email Template

2. The **Create new Email Template** popup will be displayed as shown in Figure 16.9. Enter a meaningful name in the **Unique name** area, select **Step Deadline** in the **Use Case** dropdown, and choose the email template that was created in the backend for deadline monitoring.

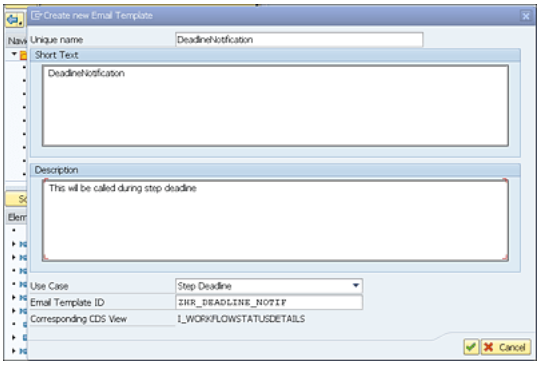


Figure 16.9 Deadline: Create New Email Template

3. Save and activate the scenario in Transaction SWDD_SCENARIO. You can see the newly created template in the **Email Templates** tab in this scenario, as shown in Figure 16.10.

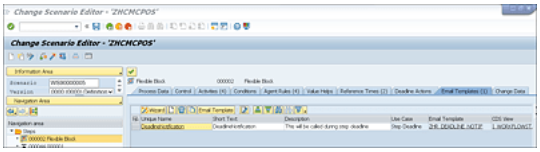


Figure 16.10 Scenario: Email Template

4. Go to the Manage Workflows app, and choose this email template to send deadline notifications for step deadlines.

16.4 Manage Teams and Responsibilities App

As we're moving toward an intelligent enterprise, the enterprise should have capabilities to determine responsible agents to work on tasks. It's essential to define and manage these responsibilities, including authorizations for various contexts, and retrieve responsible agents that can react to tasks and activities. SAP provided a new functionality in SAP S/4HANA to determine agents by using teams and responsibilities.

You can search for the Manage Teams and Responsibilities app in the search bar of the SAP Fiori launchpad and open the app, which is shown in Figure 16.11.

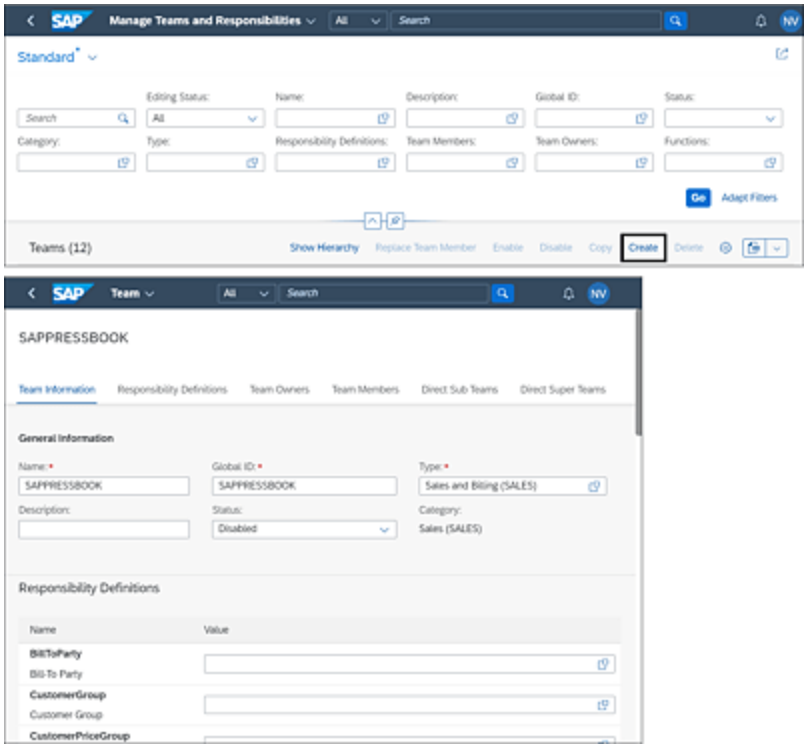


Figure 16.11 Manage Teams and Responsibilities

You can use this app to perform the following tasks:

- Create, edit, and copy teams.
- Assign team members (business partner associated with SAP user) to teams.
- Create a global ID to reference teams in other systems.
- Change the team's status via **Enable**/**Disable**. This helps you indicate whether a team can be used for agent determination in flexible workflows.

After you click on **Create** in the Manage Teams and Responsibilities app, you need to fill out all the relevant details as listed in Table 16.4.

Artifacts	Details
Name	Enter a meaningful name for the team.
Global ID	A team global ID uniquely identifies a team across multiple systems. For example, when you need to have the same team in production and quality systems, or any system or service that can store a team in responsibility management, you need a team global ID that is the same in all systems. A team internal ID is generated automatically but is unique to the system in which it's created, so you need a global ID to reference the team in other systems.

Artifacts	Details
Type	Type is linked with category, and, internally, it generally refers to business processes such as sales, procurement, distribution, journals, and so on. SAP provided a few standard categories for which you can create teams by using this app. Note: SAP didn’t provide any customization to add custom categories.
Status	**Enabled**: Team can be used in agent determination in workflows. **Disabled**: Team can’t be used.
Responsibility Definition	Responsibility definitions are nothing but team attributes. It’s a name-value pair that can be used to query teams, for example, plant = 1010 in the procurement category.

Table 16.4 Teams Artifacts

In general, the business process owner creates multiple teams for different scenarios by having different responsibility definitions, and then they can use those team when setting up workflow templates in the Manage Workflows app. As of now, this functionality is available for a few standard workflow scenarios, and you can’t use the same for your own custom scenarios.

16.5 Transporting Extensions

You must follow a two-step approach to transport extension items (e.g., extensions from the Custom Fields and Logic app, email templates, form templates, etc.) by using SAP Fiori apps in the SAP S/4HANA system. You have two SAP Fiori apps, Configure Software Packages and Register Extensions for Transport, that are used to transport these extensions. We'll explain how to transport scenarios to other systems using the standard view and the Manage Workflow Scenarios app.

You have to work with the security team and get relevant roles (standard template role: `SAP_NW_APS_EXT_ATO_PK_CFG_APP`) to access these SAP Fiori tiles.

16.5.1 Configure Software Packages

The Configure Software Packages app is used to configure software packages for transporting extension items. If you remember, all extension items are saved in the local package you've configured in the ATO setup (refer to Figure 16.1). Once development is over, you must transport those extension items across the landscape by saving them into a normal package. So, with this app, the delivery lead/project manager can configure a normal package for transporting these extension items.

As shown in Figure 16.12, you have to search for and select the Configure Software Packages app in the SAP Fiori launchpad.

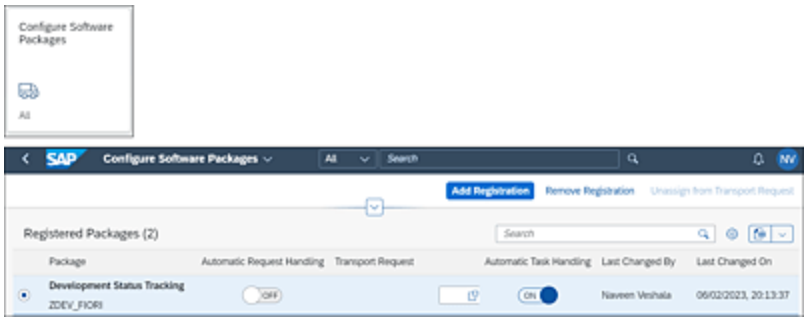


Figure 16.12 Configure Software Packages

You can find app details in Table 16.5.

Button	Description
Add Registration	This is used to register the existing package for transport. You must choose the required package in the popup and register that for transport.
Remove Registration	This is enabled when you choose any registered package in the list, and it’s used to unregister the package.
Unassign from Transport Request	This is used to unassign the package from the transport request.
Automatic Request Handling	This activates the automatic creation of transport requests. This will enable the **Activate Task Handling** option as well.
Transport Request	This assigns the package to an existing transport request, and you can see the assigned transport request in this table column.

Table 16.5 Configure Software Packages

You can follow these steps to register packages and assign transport requests to the registered package:

1. Click on the **Add Registration** button, and choose the required package from the popup window, as shown in Figure 16.13. The package will be displayed in the **Register Package** list.

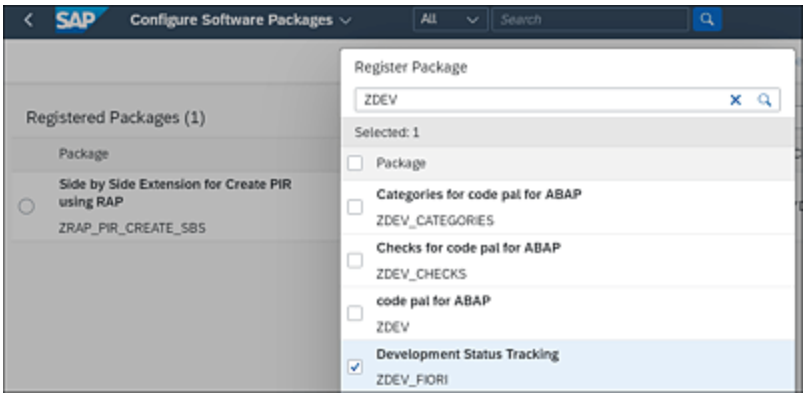


Figure 16.13 Register Package

2. Once the package is registered, the next step is to assign the transport request. Choose the package from the **Package** column, and then click on the value help in the **Transport Request** column to see all the transport requests in the popup. Choose the correct transport, and it will be shown in the **Transport Request** column in the main table (see Figure 16.14).

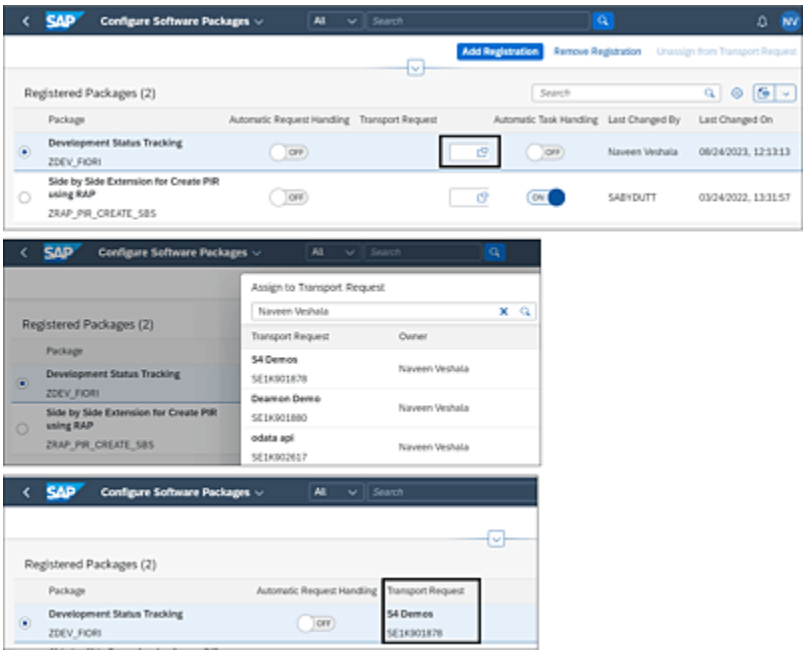


Figure 16.14 Assign the Transport to the Registered Package

16.5.2 Register Extensions for Transport

The Register Extensions for Transport app is used to assign extension items to software packages and transport requests. The email template we've created in earlier sections is saved in local project `TEST_YY_KEY_USER_LOCAL`, which we set up in ATO. You have to move this email template extension item from this local package to the transportable package by following these steps (see Figure 16.15):

1. Open the Register Extensions for Transport app via its tile, as shown in Figure 16.15.
2. Select the extension item that needs to be transported, and then click on the **Reassign to Package** button. The extension item will be moved from the local package to the actual transportable package.

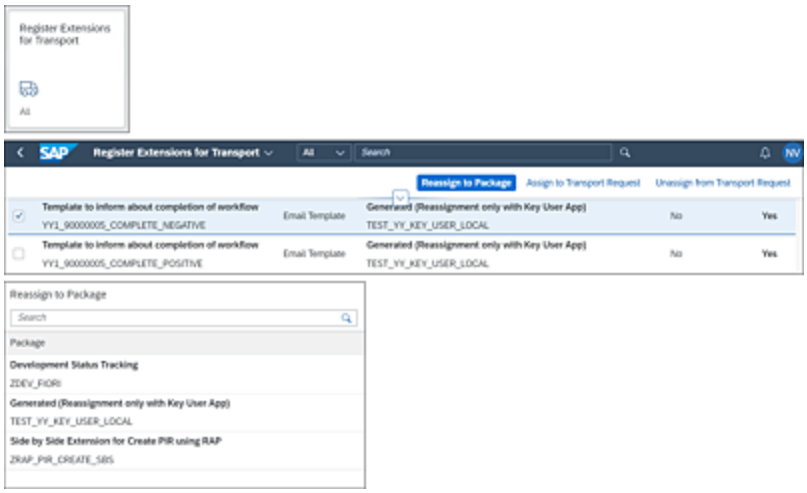


Figure 16.15 Register Extensions for Transports

16.5.3 Transporting Workflow Scenario Content

Next, you want to transport workflow definitions that you've created in the development system to the production environment. The first transport is always a full transport, and subsequent transports are only delta transport requests. The transport can contain the following content:

- Workflow definitions, that is, workflows modeled with the Manage Workflows app
- Order of workflow definitions
- Workflow templates

To transport scenario content within an on-premise environment, you must add the transportable objects to a Customizing transport and then follow these steps:

1. Go to Transaction SM31 in the development system.

2. Enter “V_SWF_FLEX_SCACT” in the **Table/View** field, and click on the **Edit** button.
3. Select the scenario you want to transport, and click on the **Transport of scenario content** button, as shown in Figure 16.16. Alternatively, you can select the scenario and press Ctrl + F7.

Another option to transport the scenario content that was created in the Manage Workflows app is directly available in the Manage Workflow Scenarios app. There is an **Export/Import** option used to export or import workflow templates into the local drive. This is also useful to create a backup. Generally, after completion of development, you can export the scenario in the development system and then import the same into the quality system.

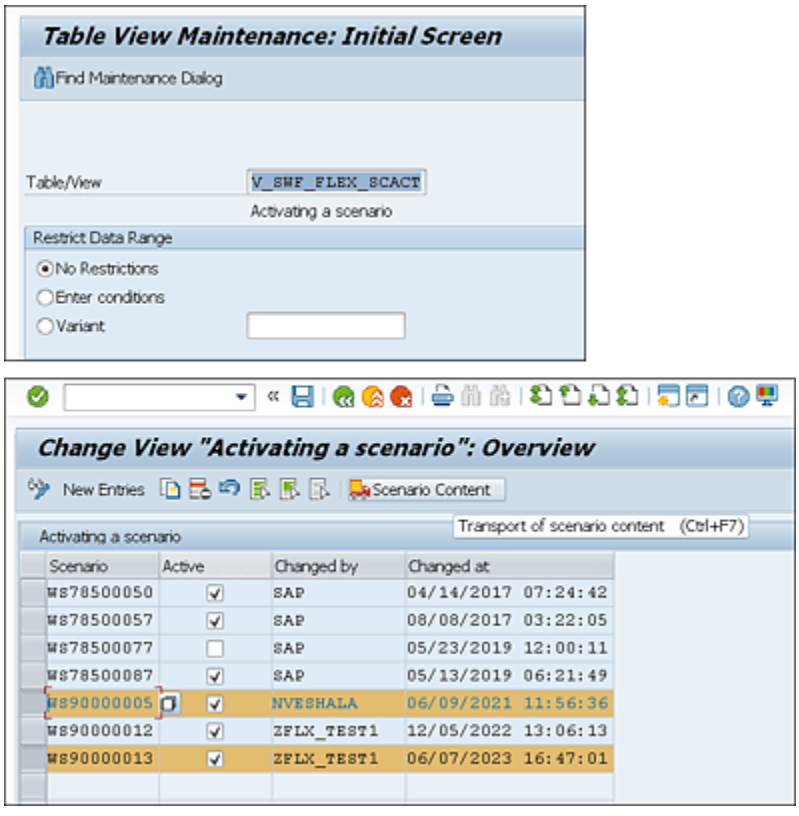


Figure 16.16 Transport Scenario

Note

Importing replaces any artifacts, for example, workflow templates, that already exist in the target system. These artifacts are replaced by the newly imported one.

The export is also useful when you initially modify the scenario in the nondevelopment system and later want to switch to maintaining the scenario in the development system. In this case, you first export the scenario in the nondevelopment system and later import the file into the development system.

16.6 Summary

In this chapter, you've learned about setting up ATO, different types of notification features, the Manage Teams and Responsibilities app, the Maintain Email Templates app, and transporting flexible workflow scenarios across the landscape.

In the next chapter, we'll move on to discuss workflows with SAP Business Technology Platform, specifically focusing on SAP Build Process Automation.

Part III

Workflows with SAP Business Technology Platform

This part will teach you about workflow services for SAP S/4HANA available via SAP Business Technology Platform. You will learn about SAP Build Process Automation and build and design workflows with SAP BTP tools and services.

17 Introduction to SAP Build Process Automation

So far, you've learned how to develop workflows inside SAP S/4HANA. It's equally important to know how to develop workflows in SAP Business Technology Platform (SAP BTP). Going forward, based on business cases, many developments of workflows will be done in SAP BTP as a side-by-side or cloud native solution. You'll learn in this chapter how to set up and develop workflows in SAP BTP.

This chapter introduces you to the SAP Build Process Automation service and describes how it can be used with SAP S/4HANA for workflows. You'll set up the service, learn to navigate the SAP BTP cockpit, and configure basic security. You'll work with this service in more depth in the remaining chapters in this part.

17.1 Overview

SAP Build Process Automation is a service within SAP BTP This service is the replacement for the SAP Workflow Management service. The main reason behind retiring SAP Workflow Management and introducing the SAP Build Process Automation service is to include all automation

features inside one service. SAP Workflow Management was mainly used to build workflows only, whereas the SAP Build Process Automation service includes business process automation using workflows and task automation using robotic process automation (RPA). This help organizations bring process automation easily and rapidly to their business.

Note

The SAP Workflow Management service is still available with customers who are already subscribed to this service and will continue to be available. However, any new SAP BTP subscription won’t get the SAP Workflow Management service.

The key features of SAP Build Process Automation are as follows:

- Intuitive workflow management and RPA functionality
- Low-code/no-code tools to simplify workflow creation, decision management, and process visibility
- Automation of routine tasks using software bots
- Intelligent document processing using artificial intelligence (AI) and machine learning
- Native process integration functionalities
- Prebuilt functionality, templates, and predeveloped industry-specific content
- Trusted enterprise-grade platform

Key benefits of using SAP Build Process Automation are as follows:

- Build processes and workflows using graphical interfaces in an intuitive way
- Develop form-based workflows using the drag-and-drop feature, which helps with rapid development, and enable citizen developers to build the business solution with less pro-code
- Easy maintenance of workflow decision logics
- Develop solution-combining process and task automation rapidly
- Prebuilt contents of SAP business processes to reuse
- Predefined software development kits (SDKs) are given to be used (e.g., Google Vision AI SDK to read documents)

17.2 Typical Use Cases for SAP S/4HANA

These days, we see a trend toward organizations sticking more closely to standard SAP functionality in order to, in SAP terminology, *keep the core clean* ("core" here refers to SAP S/4HANA) so that upgrades and feature packs can be applied easily to introduce new functionalities that accompany new versions of SAP S/4HANA. This approach helps organizations bring innovation rapidly into their business. However, keeping standard means not adopting organization-specific customizations. So, to adopt customization inside SAP S/4HANA, SAP provided features such as in-app and developer extensibility, which help to keep the standard functionality while allowing organizations to add their business-specific functionalities. However, there is another concept of extensibility called side-by-side extensibility that takes place in SAP BTP. This type of extensibility is usually adopted in loosely coupled custom solutions to perform extensions outside SAP S/4HANA itself.

For workflows, SAP provides flexible workflows with the option of customization using in-app and developer extensibility. However, there are scenarios where you need to build complete custom workflows or extend workflows in such a way that can't be done using in-app and developer extensibility completely. Those are the business situations where you'll build workflow solutions using SAP Build Process Automation, which runs on SAP BTP.

Use cases of workflows where you can use SAP Build Process Automation on top of SAP S/4HANA are as follows:

- The standard workflow provides only partial coverage of the use case needed by the business.
- The standard workflow doesn’t exist at all in SAP S/4HANA.
- The workflow will get data from SAP and/or non-SAP systems.
- The workflow will push approval/rejection data to SAP and/or non-SAP systems.

Let’s look at one example of a workflow use case where SAP Build Process Automation can be used. Suppose a business needs approval workflow for dispute management in the financial supply chain management (FSCM) area of SAP S/4HANA. The requirements for the workflow are as follows:

- The dispute administrator will create a dispute case in SAP S/4HANA.
- Creation of the dispute case will trigger a workflow.
- Based on the country and company code, a business rule will decide how many levels of approvals are needed and who will receive the workflow task.

Here, creation of the dispute case is a standard feature of the FSCM area in SAP S/4HANA. Triggering a workflow with multilevel approvals isn’t a standard feature of the FSCM area in SAP S/4HANA.

The standard workflow for FSCM dispute management only sends email notifications. So, in this case, the standard

workflow won’t fulfill the business needs completely.

This is a typical example of when you can use the SAP Build Process Automation service in SAP BTP to create your workflow for this business case. How to build the solution technically is discussed in detail in Chapter 18.

17.3 System and Service Requirements

SAP Build Process Automation runs on SAP BTP as a service. To use this service, you don’t need any server to be available on-premise. However, you can choose a hyperscaler (Microsoft Azure, Amazon Web Services [AWS], or Google Cloud) on which SAP BTP services will run. All the services of SAP BTP aren’t always available by default to all hyperscalers or to all data centers. So, before onboarding the service, you need to check the SAP Build Process Automation documentation or the SAP BTP cockpit to determine in which hyperscaler and in which locations this service is available.

17.4 Setting Up the Required Services

Setting up the SAP BTP environment properly is the most important part to start workflow development work using the SAP Build Process Automation service. In this section, you'll learn how to do this, but before that, you need to either have access to your organization's SAP BTP subaccount or have already activated an SAP BTP trial account. Usually, the SAP BTP administrator of your organization will provide you access to the SAP BTP subaccount.

To log in to the SAP BTP cockpit, paste the following link to Chrome or any supported browser, and click on **Sign in** (see Figure 17.1): *https://account.hana.ondemand.com/#/home/welcome*.

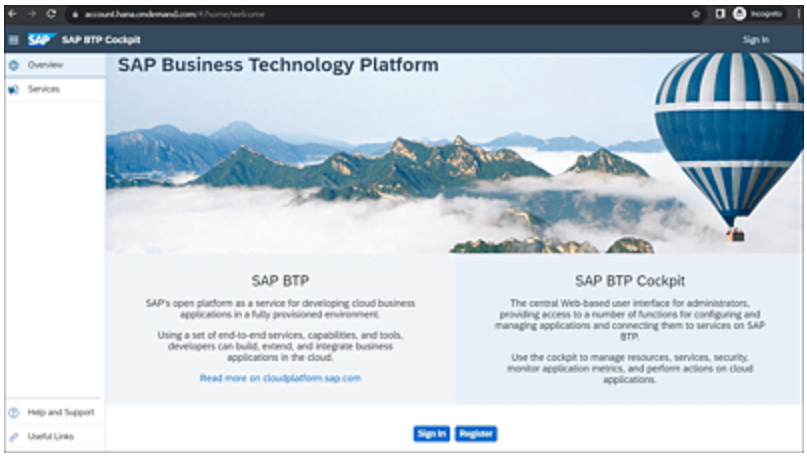


Figure 17.1 SAP BTP Cockpit Sign-In Page

All services aren't added by default to all subaccounts in SAP BTP, as you may have different subaccounts for

different purposes. So, you now need to add the service to the subaccount where you want to run workflow using SAP Build Process Automation.

After login, you'll see the SAP BTP cockpit, as shown in Figure 17.2.

Now, you need to check the entitlements to see if the required service is missing, so you can add it to the subaccount if necessary. For trial accounts, by default, all required services, including SAP Build Process Automation, are added. However, for your organization's subaccount, you need to select **Entitlements • Entity Assignments** on the left side of the screen, click the **Configure Entitlements** button on the right side, and click **Add Service Plans**, as shown in Figure 17.3.

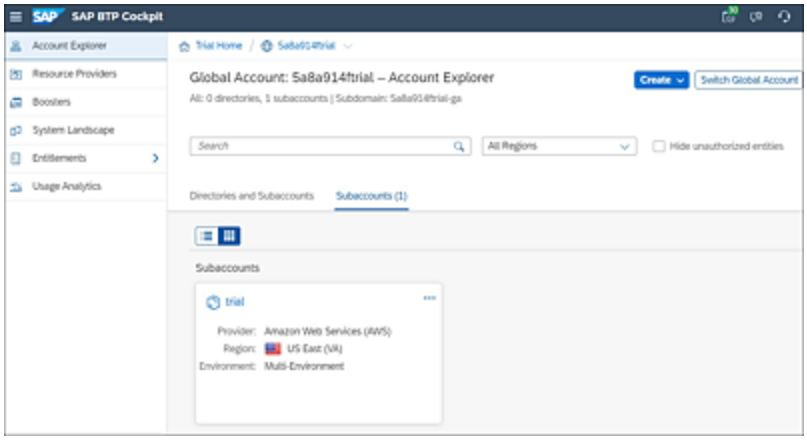


Figure 17.2 SAP BTP Cockpit after Login

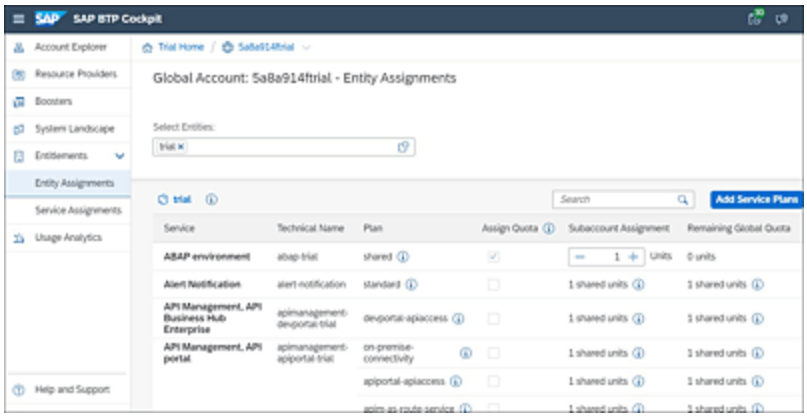


Figure 17.3 Add Service Plans

A popup will appear with the list of services that are entitled for you. Enter search term “build”, and you’ll see the **SAP Build Process Automation** option, as shown in Figure 17.4. For a trial account, all services are preselected, but for your organization’s SAP BTP account, you need to select the service plan from the right side. Select the **Standard** plan, and click on **Add 1 Service Plan** to add it to your subaccount. Now you’re ready to start the service setup in your SAP BTP account.

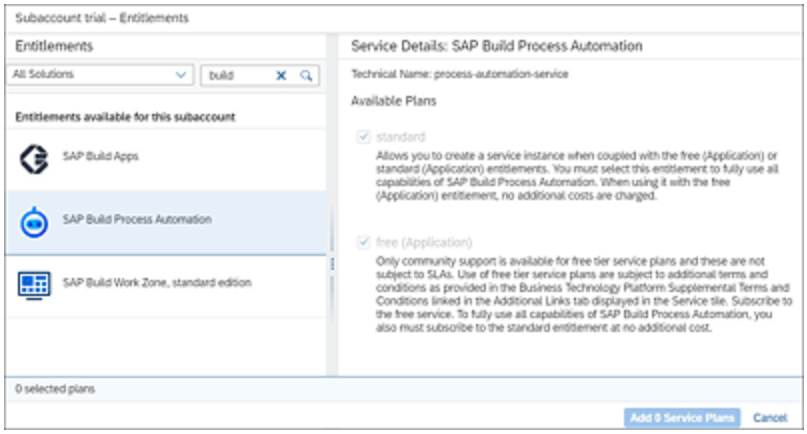


Figure 17.4 SAP Build Process Automation: Available Service Plans

It’s always recommended to use boosters to set up the SAP Build Process Automation service to create and configure

service automatically. Don’t try to do the setup manually as you’ll likely end up with multiple errors. Let’s start the SAP Build Process Automation setup by clicking on the **Booster** option (see Figure 17.5). Note that you should have administrator access in the SAP BTP account to run boosters. You’ll be taken to the following screen in the right panel of the SAP BTP cockpit. Select **Extension Suite - Digital Process Automation** from the dropdown.

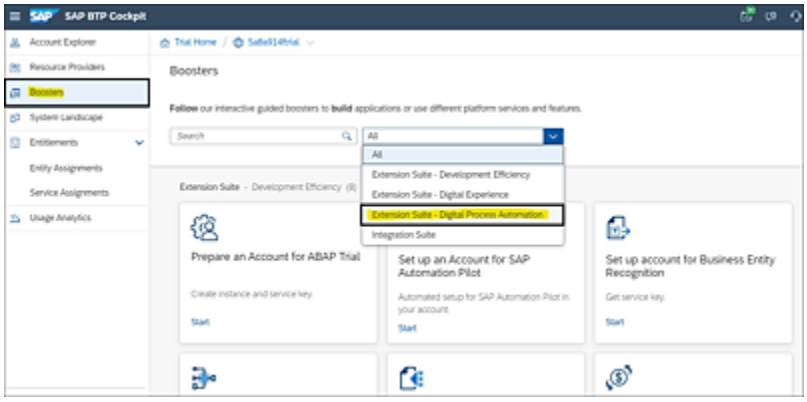


Figure 17.5 Start Service Configuration with the Booster Option

You’ll get the screen shown in Figure 17.6. Start the setup by clicking the **Start** button from the **Set up account for SAP Build Process Automation** tile.

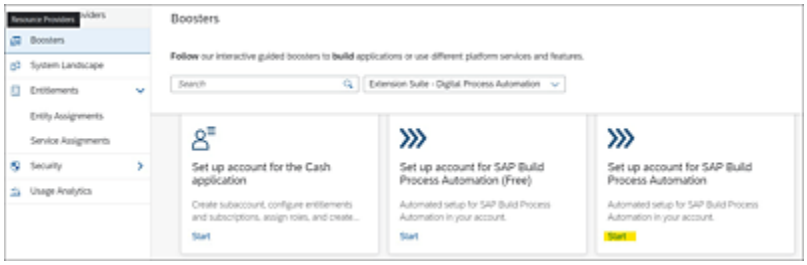


Figure 17.6 Start Booster Configuration of the Service

After clicking the **Start** button, the setup screen will appear. The first screen will check for your authorization and

entitlement of the service for the SAP BTP account that you selected. Once that’s all done, move to the next screen.

The next screen, as shown in Figure 17.7, will ask you to select the subaccount under which you want to create the workflow service and the space that you created earlier.

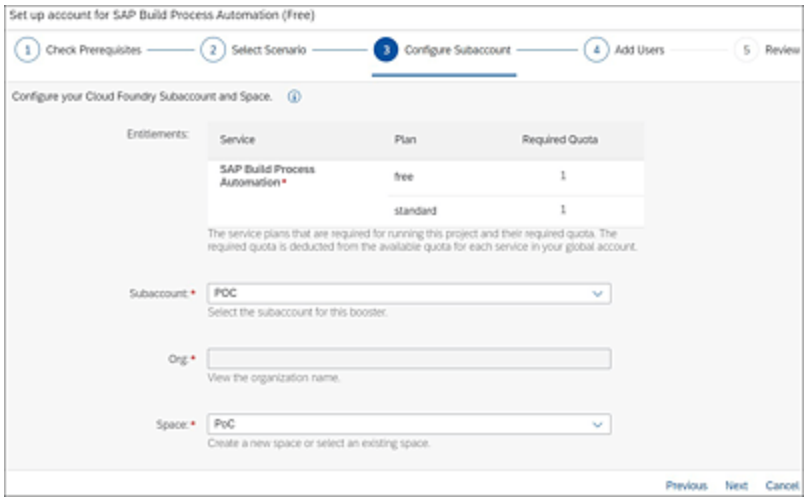


Figure 17.7 Configuration of the SAP Build Process Automation Service to a Selective Subaccount and Space

If no space was created earlier, you’ll be given the option to create a space. We didn’t create any spaces in the previous section, so you’ll now create a new space named “PoC”. Once you’ve entered the name “PoC” in the **Space** field, the booster will create the space for you.

Now, move to the next screen where you’ll select the default identity management for user authentication (see Figure 17.8). Click on **Next** and then on the **Finish** button. The process will take some time to finish. Note that if your organization has a custom Identity Authentication service, that also will be picked up in the **Custom Identity Providers** field. You also can add the administrator via email ID (who are part of the default identity service) in this

screen. Don’t add developers now because developers will be added when the security team creates organization-specific custom roles after service deployment when the standard roles are available.

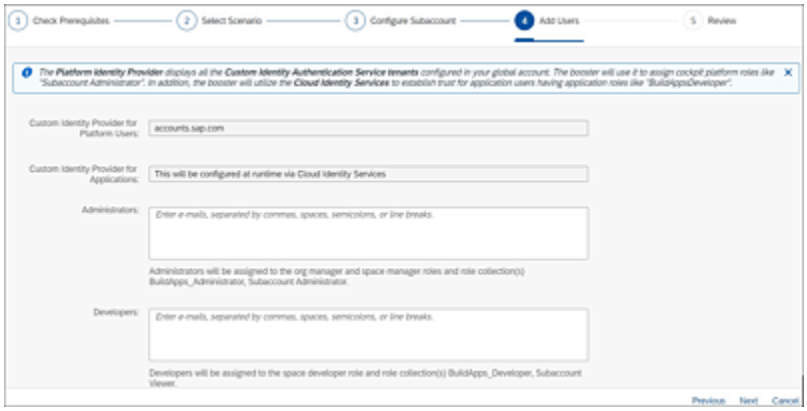


Figure 17.8 Identity Authentication Service Selection Screen

Once the process is completed, you’ll see the popup screen shown in Figure 17.9. Navigate to the subaccount where you created the SAP Build Process Automation service.

Now go to **Instances and Subscriptions** from the left menu, and you’ll see the new subscription of **SAP Build Process Automation**, as shown in Figure 17.10.

Now, click on the **SAP Build Process Automation** service link to start your first workflow development. You’ll see the screen shown in Figure 17.11 to start working with SAP Build.

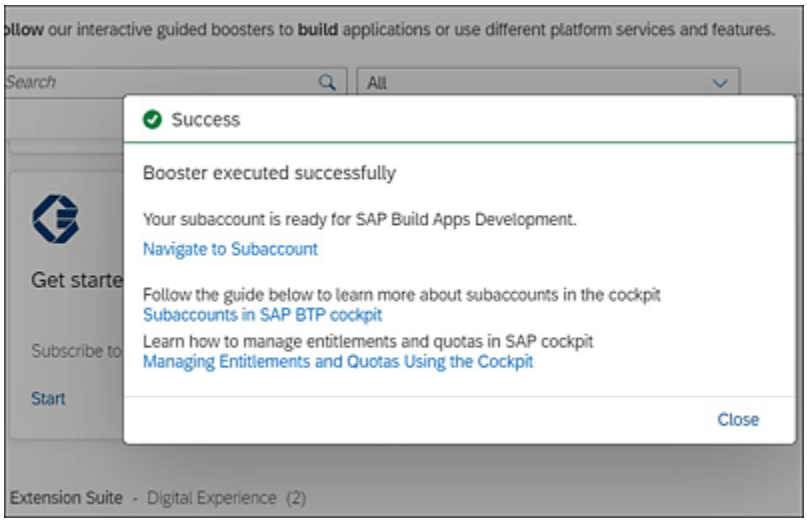


Figure 17.9 Booster Created the Service Successfully

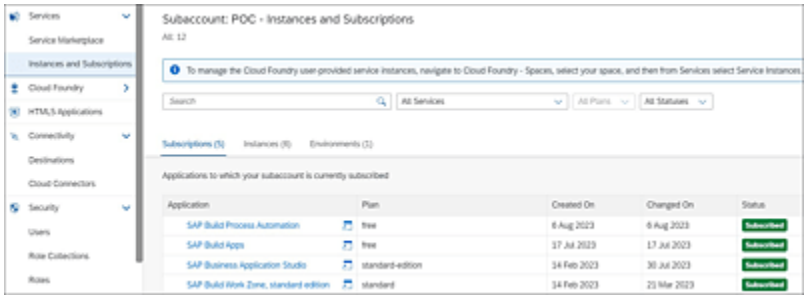


Figure 17.10 SAP Build Process Automation Service Created and Ready to Use

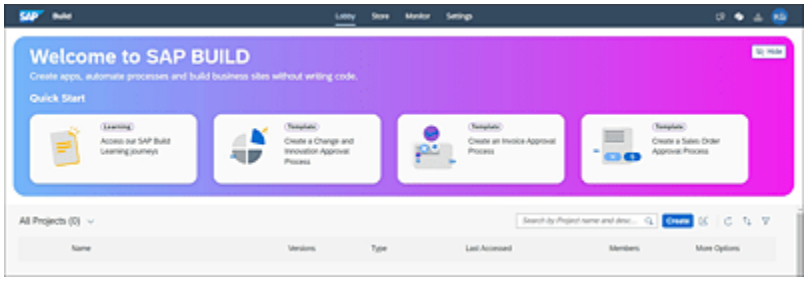


Figure 17.11 SAP Build Process Automation: Tool Landing Page after Login to Service

17.5 Working with the Workflow Cockpit

Once you're inside SAP Build Process Automation, the first tab is **Lobby** where you'll see a list of projects and also a list of prebuilt templates that you can use for rapid development.

The next tab is **Store** (see Figure 17.12), which has a list of templates, SDKs, and third-party integrations. You can add any template, SDK, or third-party service that you want to use in your workflow.

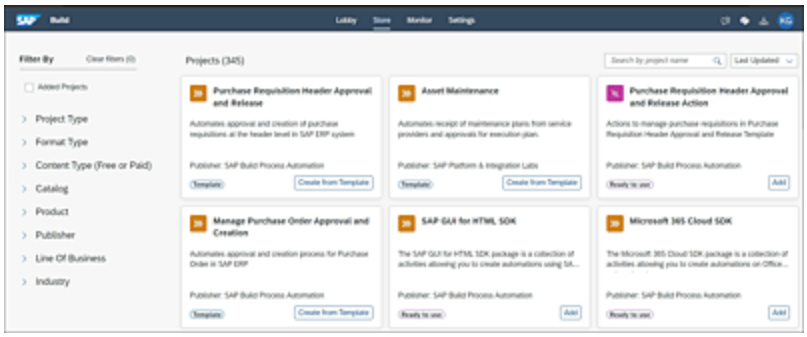


Figure 17.12 SAP Build Process Automation: Store

Next, the **Monitor** tab (see Figure 17.13) is where you'll get services such as **Business Rules** setup, process visibility setup (**Visibility Scenarios**), and so on.

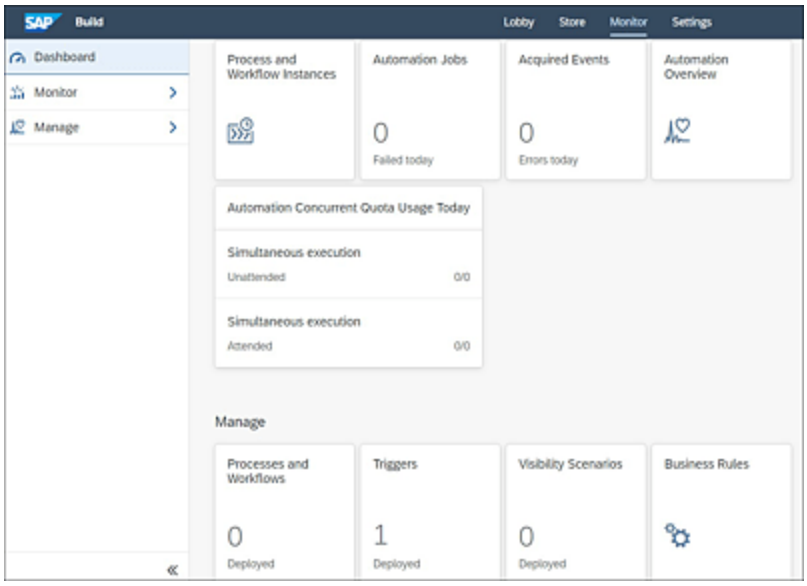


Figure 17.13 Monitor Workflow: Manage Business Rules and Process Visibility

Let’s now start with our first workflow project. To do so, go to the **Lobby** tab, and click on the **Create** button, which will open a popup with the three options shown in Figure 17.14. Select **Build an Automated Process.**

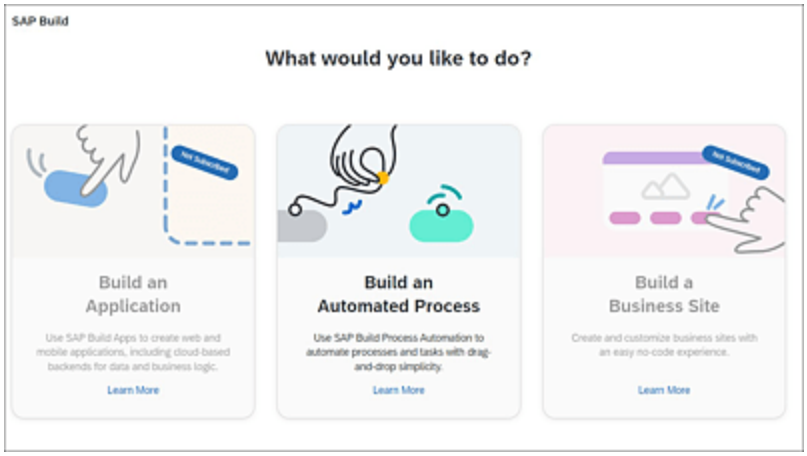


Figure 17.14 Select Build an Automated Process

The screen shown in Figure 17.15 appears. Because you’re creating a workflow project, select **Business Process** to start a workflow project.

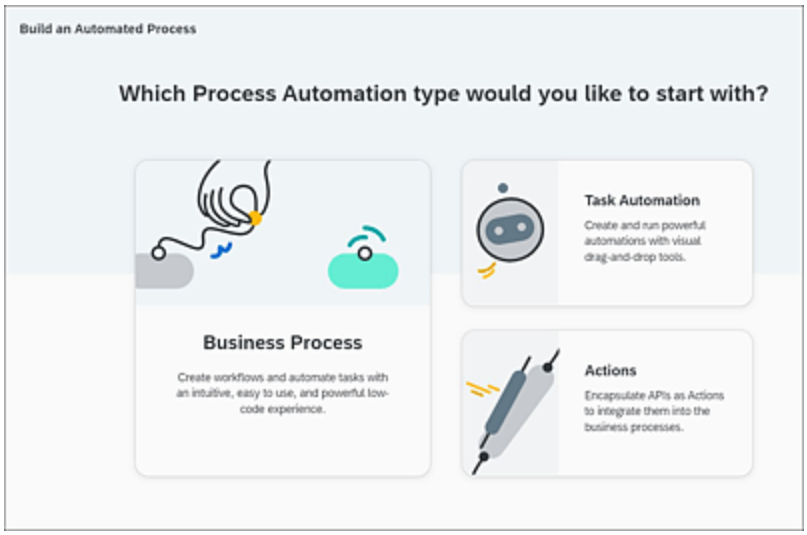


Figure 17.15 Select Business Process to Start a Workflow Project

Now enter a name in **Project Name** (and, optionally, a short description) for the project on the next screen to start your first workflow build, as shown in Figure 17.16.

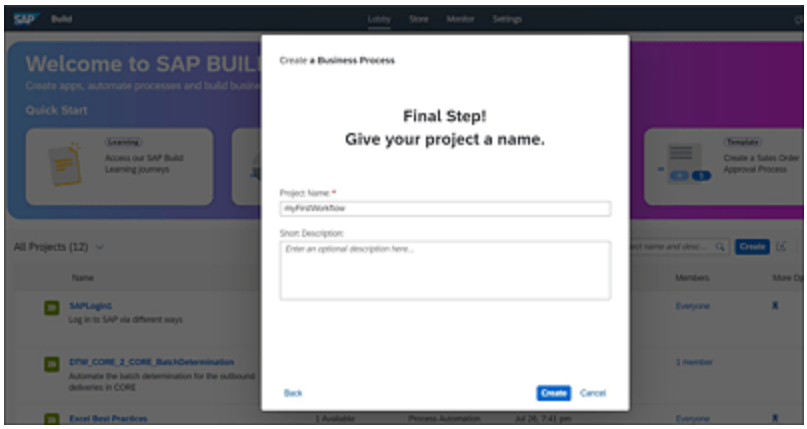


Figure 17.16 Name Your Project

Figure 17.17 shows the workflow editor. This is the design-time place where you'll design your workflows, tasks, and subflows. You'll also be able to link main flows to subflows, call APIs for external data, send mail, push workflows for approval to inboxes, and so on.

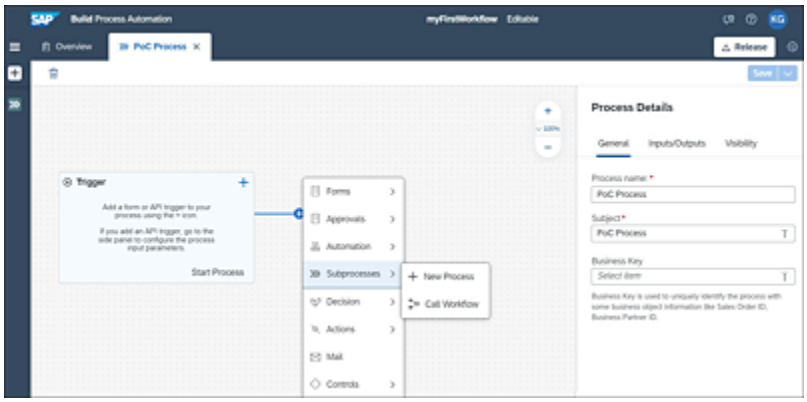


Figure 17.17 Workflow Editor to Start Creating the Workflow

17.6 Security

SAP Build Process Automation security comes with three role collections, as follows:

- Process Automation Admin
- Process Automation Developer
- Process Automation Participants

All three role collections will be available in the **Role Collection** section of SAP BTP when the SAP Build Process Automation service setup is completed. An organization's security team should assign these roles based on tasks to various types of users to maintain proper authorization based on business uses.

17.6.1 Process Automation Admin

This role collection is for the administrator of SAP Build Process Automation. Based on organizational needs, you can create a custom role collection with the required roles from the following list for the administrator:

- `AdvancedUser`
- `Document_Information_Extraction_UI_Templates_Admin`
- `EventsAdmin`
- `FormsAdmin`
- `IRPAOfficer`
- `IRPAPersonalDataAccess`

- PVAdmin
- PVEventSender
- PVTenantOperator
- RegistryAdmin
- RegistryDeveloper
- RuleExecutor
- RuleRepositorySuperUser
- SchedulerAdmin
- WorkflowAdmin
- WorkflowContextAdmin
- WorkflowTenantOperator

17.6.2 Process Automation Developer

This role collection is for developers. As mentioned earlier, you can always create your custom role collection and add only the required roles from the following list. As an example, if you don’t need to use RPA during development, but only workflow tasks, then you don’t need to assign RPA roles to the developer inside your custom role collection. It’s suggested to not change anything in the following standard role collection:

- AdvancedUser
- Document_Information_Extraction_UI_Templates_Admin
- EventsDeveloper
- FormsDeveloper

- IRPAProjectMember
- PVDeveloper
- PVOperator
- RegistryDeveloper
- RuleDeveloper
- RuleExecutor
- SchedulerDeveloper
- WMDeveloper
- WorkflowBusinessExpert
- WorkflowDeveloper

17.6.3 Process Automation Participants

This role collection is for the end user who will use process automation for approvals and accessing forms to submit. As mentioned earlier, for example, if this same user won't access RPA and workflows, then you can divide this role collection into two parts and assign roles accordingly. The standard roles for participants are as follows:

- FormsParticipant
- IRPAAgentUser
- IRPAParticipant
- RegistryDeveloper
- WorkflowInitiator
- WorkflowParticipant

17.7 Troubleshooting

During the SAP Build Process Automation setup, especially when performing the setup in your SAP BTP account, you may get issued a quota warning during the booster run. You'll the warning shown in Figure 17.18, which states that there are missing entitlements with a required quota of 1.

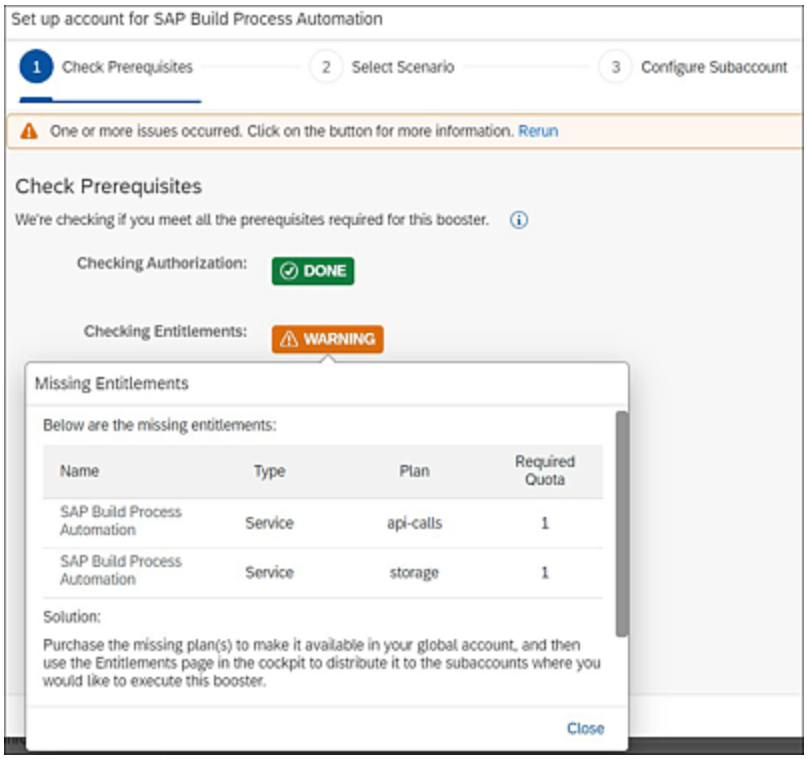


Figure 17.18 Issue during Booster Setup

If such an issue appears, as mentioned in the **Solution** section of Figure 17.18, investigate your entitlements from the **Entitlements** page in the SAP BTP cockpit (choose **Entitlements** • **Service Assignments**). If you have entitlements, then add them; if not, connect with SAP to provide you a larger quota.

17.8 Summary

In this chapter, you learned about the SAP Build Process Automation functionalities and how to use this service. We also covered in detail how to set up this service in SAP BTP and how to set up roles for authorizations. Finally, we also touched on troubleshooting of the service during setup.

18 Process Development

In the previous chapters, you saw how SAP Business Workflows and flexible workflows are designed and developed in the SAP S/4HANA system. With the advent of clean core solutions, companies are moving more toward SAP Business Technology Platform (SAP BTP) as a base for the customization and extension requirements, especially with workflows. In this chapter, we'll introduce you to the world of SAP BTP-based workflows, design techniques, and options, along with various design concepts. Finally, we'll implement a typical use case with the available flavors of workflows.

In the previous chapter, we discussed what SAP Build Process Automation is and where it can be used, depending on your use case requirements in SAP S/4HANA. We also saw how to set up the various services for SAP Build Process Automation, which got us ready to kick start the design process. In this chapter, we will first discuss some techniques to design a workflow and then we will walk through the two options for creating workflows on SAP BTP: SAP Workflow Management using SAP Business Application Studio and SAP Build Automation Process using the process builder tool. The former is a *procode* tool, which an SAP BTP developer might be more interested in, and the latter is best

suited to a *citizen developer* who wants a platform that allows for the rapid design and development of processes with low-code/no-code tools.

In this chapter, we'll first look at the design techniques for a typical workflow and then cover the various design elements (controls) we can use for both SAP Business Application Studio-based design as well as for process builder-based design. Then we will go step-by-step through building a typical workflow, using dispute process management as our example, using the process builder and then deploy that workflow. We will also discuss the same use case built on SAP Business Application Studio.

We will close the chapter with coverage of how a few of the key dependencies (like destinations) are set up, how to use SAP Cloud Transport Management to transport our workflow artifacts, and how authentication is set up. We will also provide an overview of the key APIs SAP offers that we can use for workflows.

18.1 Workflow Design Techniques

SAP Build Process Automation is a very new service (as of September 2023). Many features that were available in SAP Workflow Management aren't included in this service yet. You can create simple or medium complex workflows using SAP Build Process Automation. Some examples of this are as follows:

- Workflows with multiple level approvals

- Workflows with subprocesses or calling a separate workflow from a current workflow
- Workflows with branches based on decision and/or condition
- Workflows with wait time
- Triggering workflows via an event or application programming interface (API)
- Triggering workflows from a form or SAPUI5 applications
- Adding a robotic process automation (RPA) process inside the workflow

So, if you have business requirements that include any of these criteria, you'll be able to adopt such use cases within SAP Build Process Automation workflows. A majority of these cases will satisfy your business needs, so you can use the SAP Build Process Automation service. However, let's look at some complex business cases that have the following requirements:

- Adding complex business logic within the workflow with scripts between passing data from one approver to another
- Complex user interface (UI) for approvers with data addition and validation
- Wait time for approvers, such as if a workflow isn't approved after 48 hours, the workflow will move to another person

In such cases, we won't be able to meet business needs using Workflow Builder editor, which is provided for low-code/no-code developers. In such situations, we need to

move to a pro-code solution. Pro-code solution refers to using the SAP Business Application Studio workflow plug-in. Yes, we still can develop complex workflows using SAP Business Application Studio and deploy to the SAP Build Process Automation service runtime. These workflows also will work perfectly. More details are available in Section 18.2.

Now, let's talk about the design techniques for creating an effective workflow. There are three key elements involved to design a fool-proof workflow:

- **Input**
  What are the input parameters needed to start a workflow and to complete each task? This involves providing the right business input.
- **Processes**
  What business logic has to be processed to get the required output?
- **Output**
  This is the process outcome that will be given to the next task or to finish the workflow.

Let's discuss in detail how you can achieve the design of an effective workflow:

1. Understand the business flow: every workflow is built to cater to a business process flow, so it's important to understand the business logic that has to be implemented for the workflow.
2. Create a flowchart of the process steps or tasks and document each task with a clear start- and end-of-flow indicator. Once done, validate with your business

analyst whether the process flow you've drawn is reflecting the use case build requirements.

3. Once validated, start mapping your input and output of workflow tasks. This is the time when you'll be planning the input parameters and output parameters of each task, that is, what input parameters are needed to start the flow, what parameters will be passed by `task1` to `task2`, and so on.
4. Identify the steps where you need to add waiting, where you need to call external application program interfaces (APIs), and where you need to add business logic.
5. Identify process steps that can be reused for and from other workflows, and put them in subflows. If your workflow is very complex in nature, it's always better to spilt them into multiple subflows.
6. Finally, most important is identifying the tasks where you can introduce automation. In fact, this is the main reason for introducing SAP Build Process Automation and retiring SAP Workflow Management. With SAP Build Process Automation, you can add multiple automation features to your workflows.

Additionally, it's important to keep in mind that workflows aren't just used for approvals; you can use workflows for business process automation as well.

18.2 Creating a Workflow Using SAP Business Application Studio

Let’s dive deep to see in detail how to use the various design elements and controls to build a meaningful workflow using SAP Business Application Studio for the SAP Workflow Management service. You’ll see how to use SAP Business Application Studio to create a workflow module and details about the different controls available for creating a workflow in the following section. Later, you’ll see how to use these controls to create an end-to-end workflow for a specific use case, deploy it, and run it.

18.2.1 Create a Workflow Module

Let’s start by creating a workflow module in SAP Business Application Studio, as follows:

1. Create a new project in SAP Business Application Studio by choosing **File** • **New Project from Template**, as shown in Figure 18.1.
2. Choose **Basic Multi Target Application**, and click the **Start** button.
3. Provide a project name in the **Enter a project name** field (here, “LeadToCashDemoProject”), and click **Finish**, as shown in Figure 18.2.
4. Right-click on the *mta.yaml* file created in the MTA project, and select **Create MTA Module from Template** (see Figure 18.3).

5. Choose **Workflow Module**, and click **Start**, as shown in Figure 18.4.

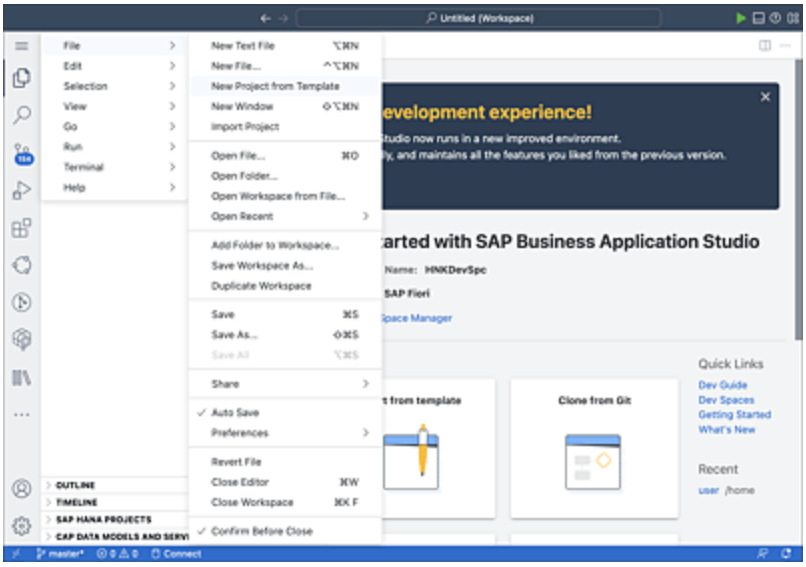


Figure 18.1 Create a Multi-Target Application Project in SAP Business Application Studio

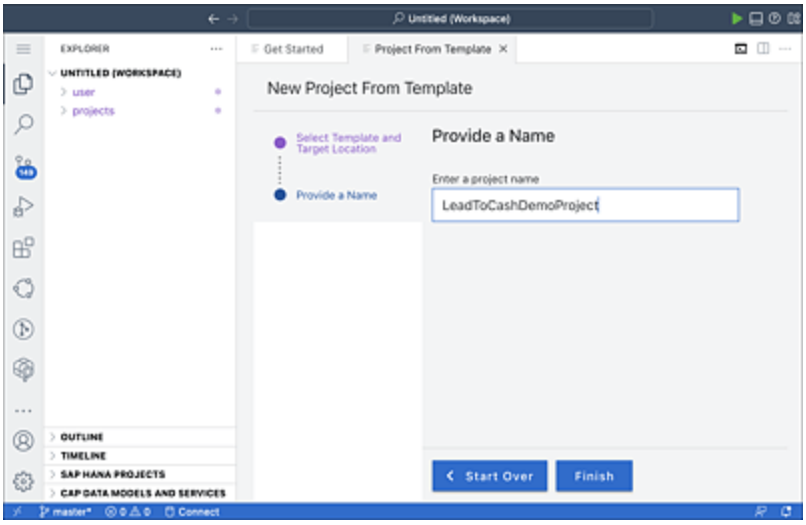


Figure 18.2 Create an MTA Project Folder

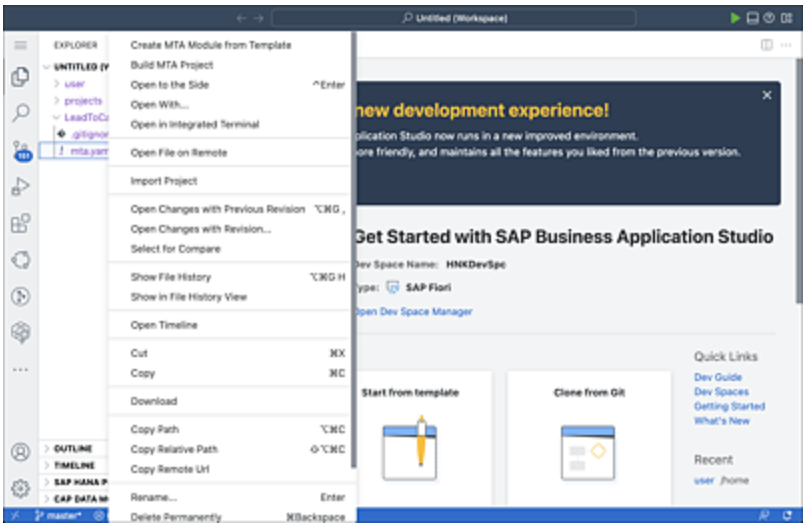


Figure 18.3 Create a Workflow Module (Part 1)

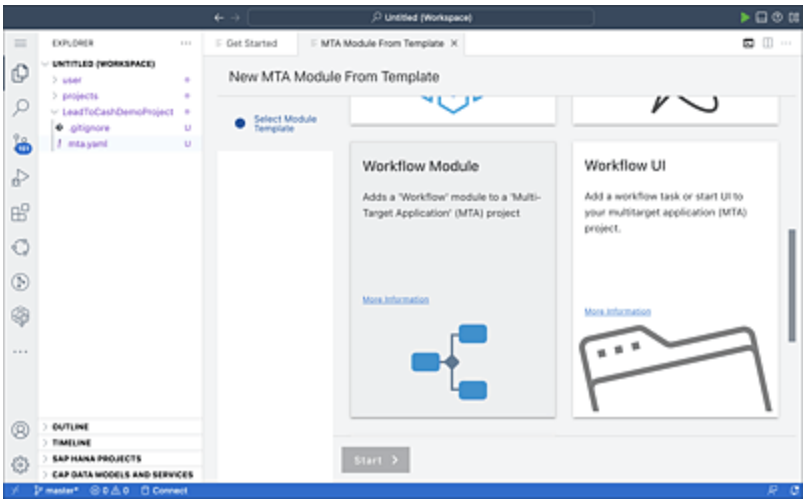


Figure 18.4 Create a Workflow Module (Part 2)

6. On the screen shown in Figure 18.5, provide the path in the **Enter the path to the folder where the module should be generated** field and the workflow module name in the **Enter the module name** field. Click **Next** to continue to the next screen.

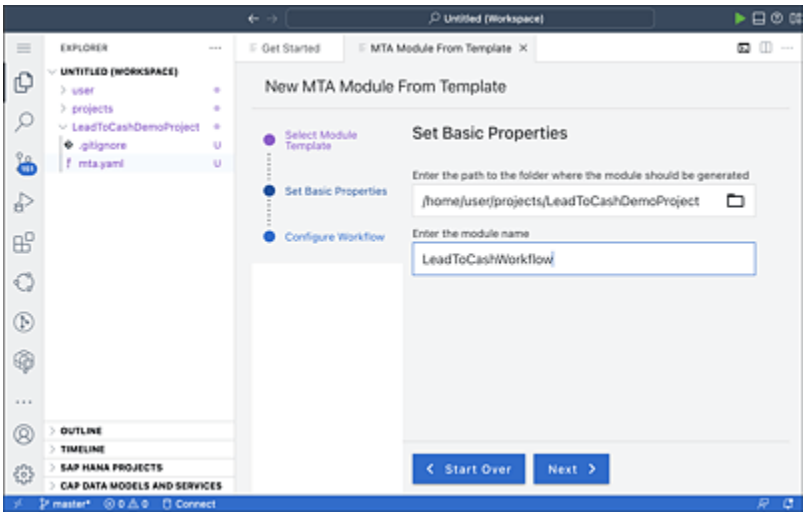


Figure 18.5 Name Your Workflow Module

7. On the screen shown in Figure 18.6, provide a namespace in the **Enter a namespace** field, a workflow name in the **Enter a workflow name** field, and a description in the **Enter a workflow description** field. Click **Finish**.

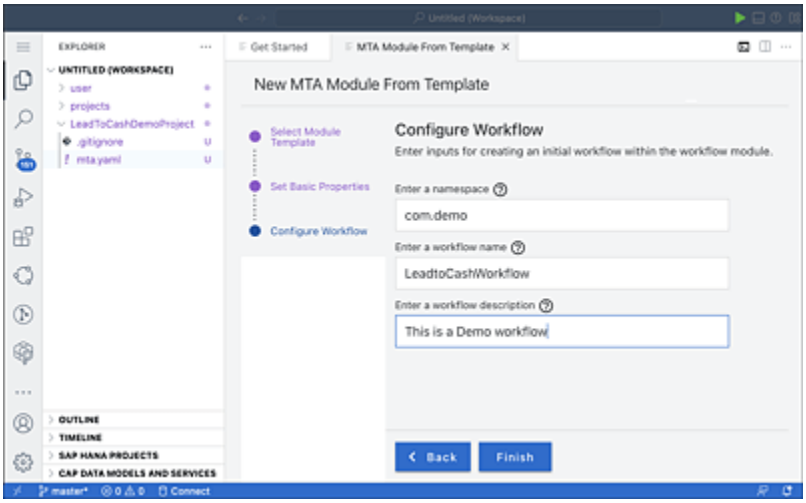


Figure 18.6 Provide a Namespace, Workflow Name, and Description

8. Open the newly created workflow in the Workflow Editor, as shown in Figure 18.7.

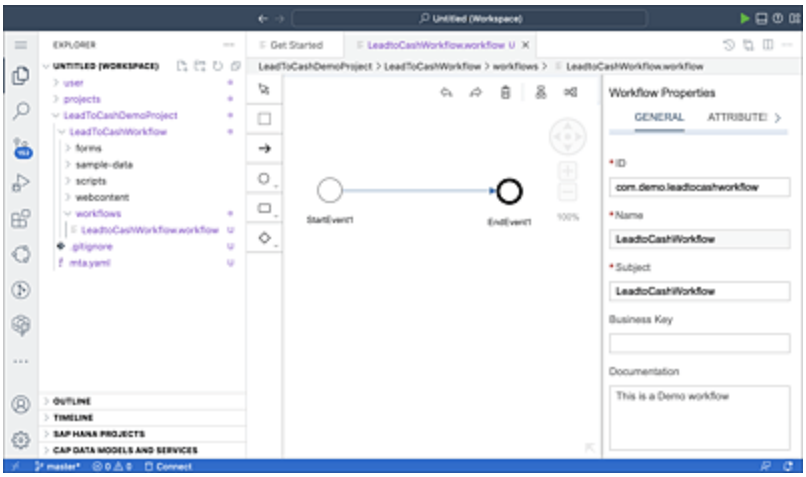


Figure 18.7 Workflow Editor

When you add the workflow module, it should open the primary design canvas automatically. If not, navigate to the folder named **workflows**, and double-click the workflow file that was created. Note that there are two controls automatically added in the designer: **StartEvent1** and **EndEvent1**. This represents the workflow start event and end event. You'll add more controls later between these two events.

Now you're all set to start designing your workflow.

18.2.2 Tasks in Workflows

The Workflow Designer provides the ability to create the following tasks:

- Script tasks
- Service tasks
- User tasks
- Mail tasks

You'll see each of these tasks in detail in the following sections.

Script Task for Data Manipulation

One of the important activities when you design a workflow is having ways to manipulate data. This is done using script tasks. You define script tasks whenever you need to have some data prepared for the next task, constructing a data payload for a service task, and so on. A script task is associated with a JavaScript-based program and gets executed automatically when the workflow reaches this step in the flow. Let's see how to create a script task, as follows:

1. In your Workflow Designer that you created in the previous step, click the **Tasks** icon in the side menu, and select **Script Task,** as shown in Figure 18.8.

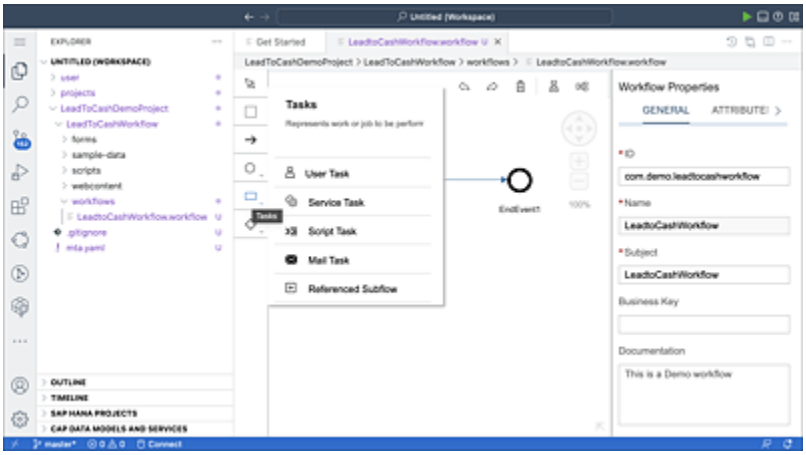


Figure 18.8 Create a Script Task

   A new script task is added in the designer as shown in Figure 18.9.

2. Now you need to create a JavaScript file that will get executed when the workflow reaches this step. On the

right-side menu, you'll see a link and a button under the **Script File** header. If you have a JavaScript file already created, you can add that using the **Select** button. As we don't have any yet in this use case, you need to create a new JavaScript file. Click on the **Create File** link (refer to Figure 18.9), and provide the name of the JavaScript file as "InitialTask". You can see that a *.js* file got created under the scripts folder in your project, as shown in Figure 18.10.

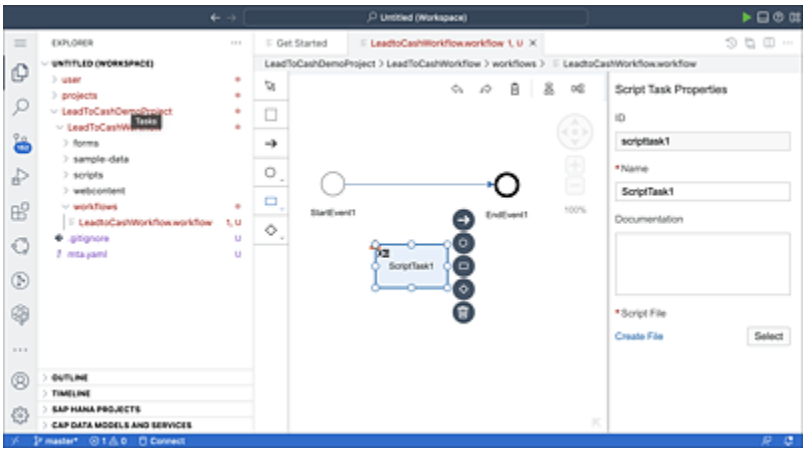


Figure 18.9 Script Task Created

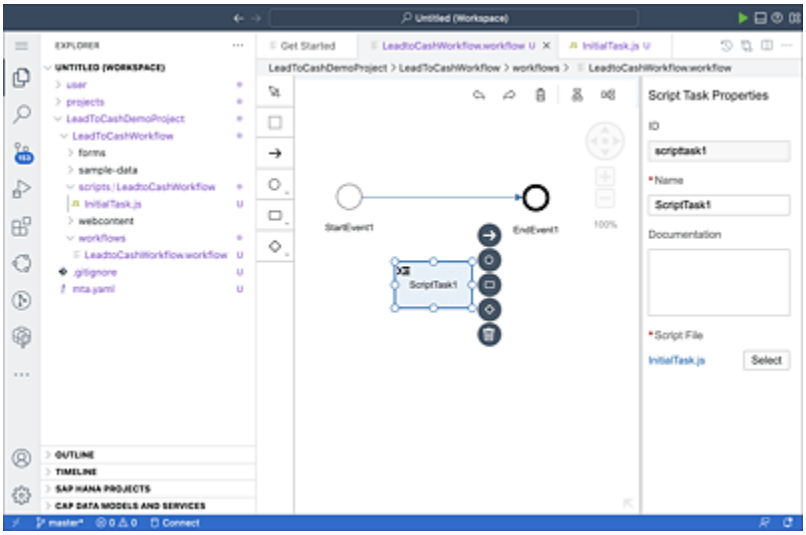


Figure 18.10 Creating a .js File for the Script Task

3. You can now write your logic in this JavaScript file, as shown in Figure 18.11.

Figure 18.11 Write the Data Handling Logic in the .js File

4. You can see that there are two events automatically created in the workflow: **StartEvent1** and **EndEvent1**. We'll talk about these events in detail in Section 18.2.4. For now, let's connect the start event to our newly created script task. For this, select the arrow between the start and end events and delete it. Then, click on **StartEvent1**. From the resulting context menu, select the arrow icon on the top, which represents the sequence flow, and drop it on the script task. Do the same from the script task, and drop it on the end task. The resulting screen will look like Figure 18.12.

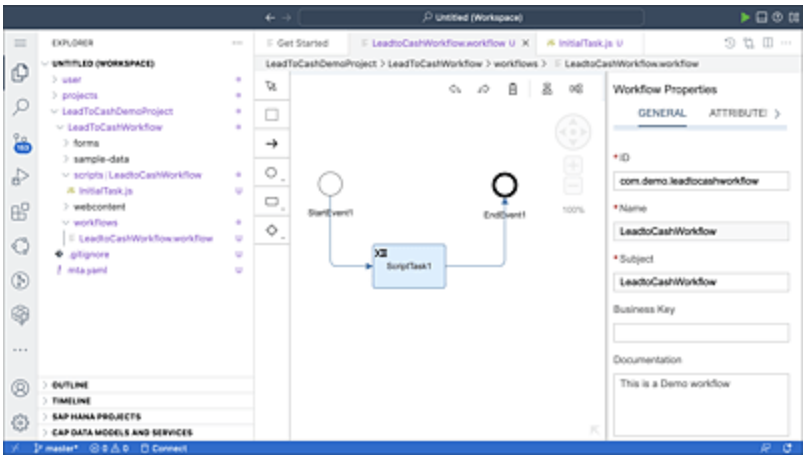


Figure 18.12 Workflow with a Script Task

Service Task for Create, Read, Update, Delete Operations

Now that you know what a script task is, let’s look at the service task. A workflow should be able to read, write, and delete data objects of any given system of records. Whether an SAP system or non-SAP system, a workflow does this using API calls. In an SAP system, this will be done using an OData call; for non-SAP systems, it will be a REST API call. For this, you have the service task. Let’s create a service task in our flow by clicking on the **Tasks** icon and selecting **Service Task**, as shown in Figure 18.13.

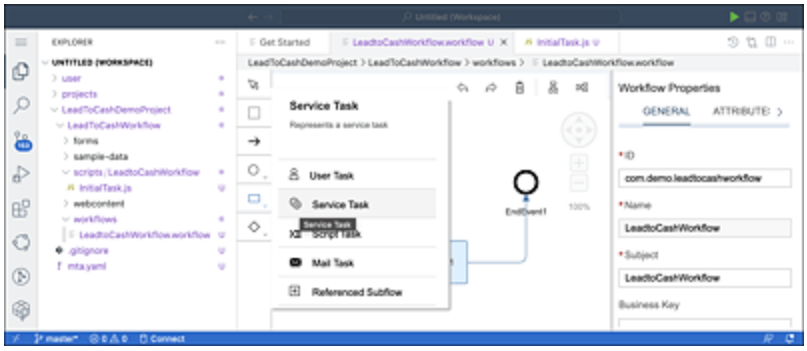


Figure 18.13 Create a Service Task

Once the task is added to your Workflow Designer, select the task. You can see the **Service Task Properties** in the right-hand pane. The **GENERAL** tab lets you provide the general properties of the task such as **Name** and **Description**. Name the task “Call S/4HANA OData”.

Now, click the **DETAILS** tab, which is the most important tab where you need to configure the OData/REST API details that this task will execute (see Figure 18.14).

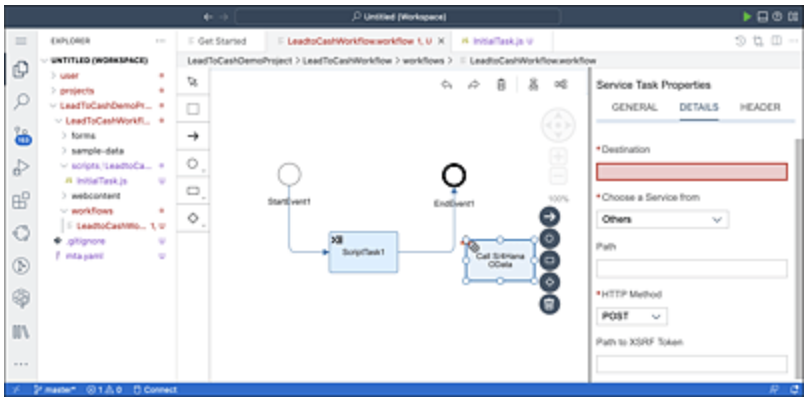


Figure 18.14 Configure the Service Task

Configure the service task as shown in Figure 18.15 by following these instructions:

- **Destination**
  This is the property where you need to set the destination name of your target system where the API needs to be called. You define destinations in the **Destination** section of the SAP BTP cockpit, which is described later in this section. For example, we have an on-premise SAP S/4HANA system where we need to execute OData through this task. We define the destination in the SAP BTP cockpit, that connects to SAP S/4HANA using the cloud connector.

- **Choose a Service from**
  This property lets you select APIs either from SAP API Business Hub or by defining on your own when you select **Others**.
- **Path**
  This is the most important part where you define the path to the API endpoint, which is the path to the entity set that we're trying to call in OData.
- **Path to XSRF token**
  Here, you define the path where the workflow module can retrieve a Cross-Site Request Forgery (XSRF) token. Generally, for OData, you enter the path to the OData name removing the entity set.
- **Request Variable**/**Response Variable**
  These are custom context variables you define that store the data payload to be sent to OData and the response you get after execution, respectively.
- **Principal Propagation**
  Choose this option if you need to pass on the authentication credentials per the logged-in user executing this task.

The third tab in the properties section is for **HEADER** information. In this section, you can add required HTTP headers if needed, especially for REST API calls, such as the content type, API key, and so on.

Similar to how you connected the start event to the script task, let's connect the script task to the service task to make a complete flow, as shown in Figure 18.15.

Now you need to configure a destination for the service task. Let’s understand why you need a destination and how to configure it. A destination is a proxy object that provides connectivity to a remote system of record, which enables outbound communication from workflows when you make a call through an exposed API. Whether it’s SAP S/4HANA or a non-SAP system, you need to define a destination. Apart from providing a means of communication, it also helps in taking care of the authentication and other properties centrally. You define and configure destinations in the SAP BTP cockpit.

Let’s see how it’s done. Log in to your SAP BTP cockpit, and navigate to your subaccount. Under the **Connectivity** section on the left side menu, choose **Destinations**.

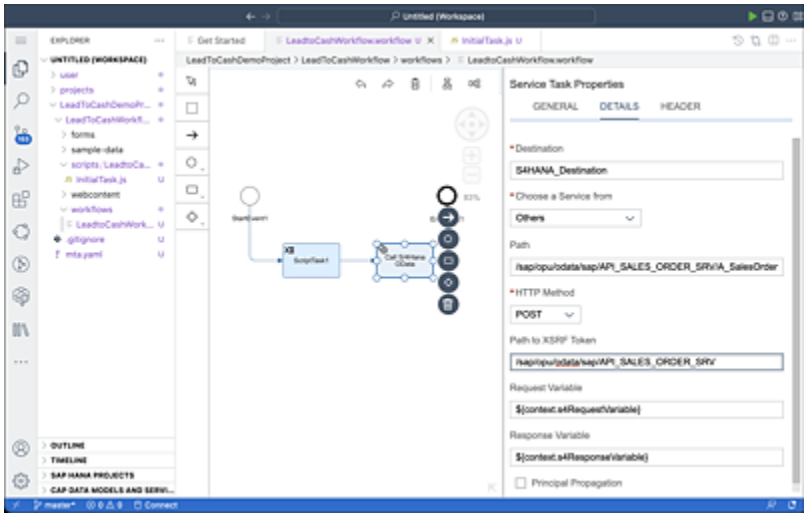


Figure 18.15 Configured Service Task

Click **New Destination**, and configure the destination. Provide the details of the on-premise system. A sample screen is shown in Figure 18.16. The key steps in this task are as follows.

1. Provide a meaningful **Name** for the destination.

2. Select the destination **Type**, such as **HTTP**, **LDAP**, **EMAIL**, or **RFC**, according to the requirement.
3. The next key element is the destination host **URL**. This should represent the host name set in the cloud connector (either original or virtual host).
4. The **Proxy Type** needs to be set to **OnPremise** for on-premise systems and to **Internet** for other cloud systems.
5. Next, select the **Authentication Type**. This should be based on the kind of authentication set for the selected destination, such as Basic, OAuth 2.0, and so on. Mostly for on-premise destinations, you'll select **BasicAuthentication**.
6. The **Location ID** represents the parameter set in the cloud connector. Leave this blank if this isn't set in the cloud connector.
7. Finally, provide the **User** ID and **Password** for basic authentication, or, for OAuth 2.0, provide the client ID and client secret. It would require multiple parameters for other types of authentications.
8. Under the **Additional Properties** section, you can add multiple parameters that may be used by the consuming services. For SAP Business Application Studio, for example, a few additional parameters are required, such as `WEBIDE Usage = true`. You can also define the SAP S/4HANA client ID as `sap-client`. This will be the destination name that you'll be using in the workflow service task whenever you need to make an OData call to this system.

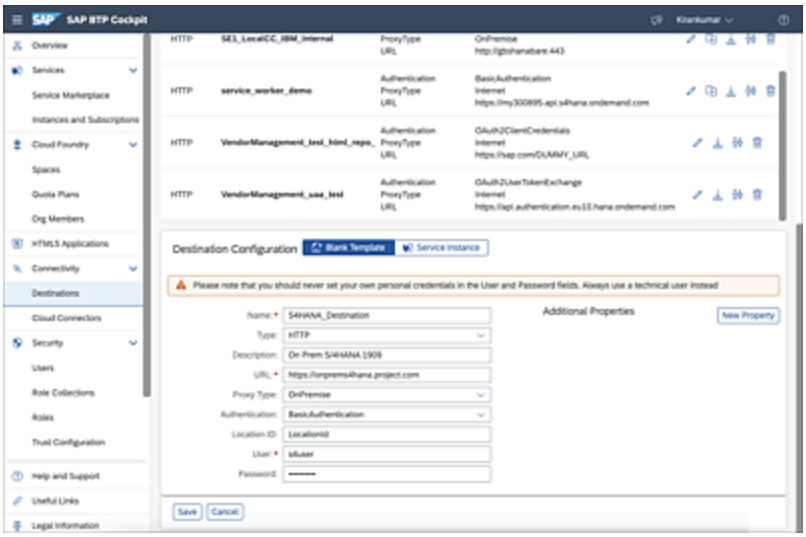


Figure 18.16 Creating a Destination

User Task

One of the most important task controls in a workflow is the user task. Using a user task, a manual input can be introduced in the workflow, especially when the flow needs a user input such as an approval for the workflow to move forward.

Let’s see how a user task can be configured in the workflow. Like you created the service task, select **User Task** from the side menu bar under the **Tasks** menu. (Note that you can create any controls from the context menu for any of the controls already added in the designer.) Click on the design screen and you can see a **User Task** control is placed as shown in Figure 18.17.

A user task has quite a few properties that need to be set. Let’s see them in detail. The **Properties** section on the right has the **DETAILS**, **USER INTERFACE**, and **ATTRIBUTES** tabs, along with the **GENERAL** tab.

Figure 18.17 Create a User Task

In the **DETAILS** section, you set the properties that should appear for the user task in My Inbox, as shown in Figure 18.18:

- **Priority**
  This can be set **to Low**, **Medium**, **High**, and **Very High**.
- **Subject**
  This is the text shown as the heading in the My Inbox task.
- **Description**
  This is the text description of the task.
- **Recipients**
  This is where you need to set the user to which the task needs to be assigned. This can be set as a static email ID/user ID, but usually you want this to be resolved at runtime. This is achieved by setting a Java Unified Expression Language (JUEL) expression to assign the `userid` from a context variable. You can set **Groups** email ID as well.
- **Due Date**
  You can define a due date for the user task by setting this checkbox. It can be set based on either a duration or a

time stamp. If its duration, you'll get an option to select a static value or a dynamic value that is resolved using an expression. The duration could be a numeric value that would represent minutes, hours, days, months, or years.

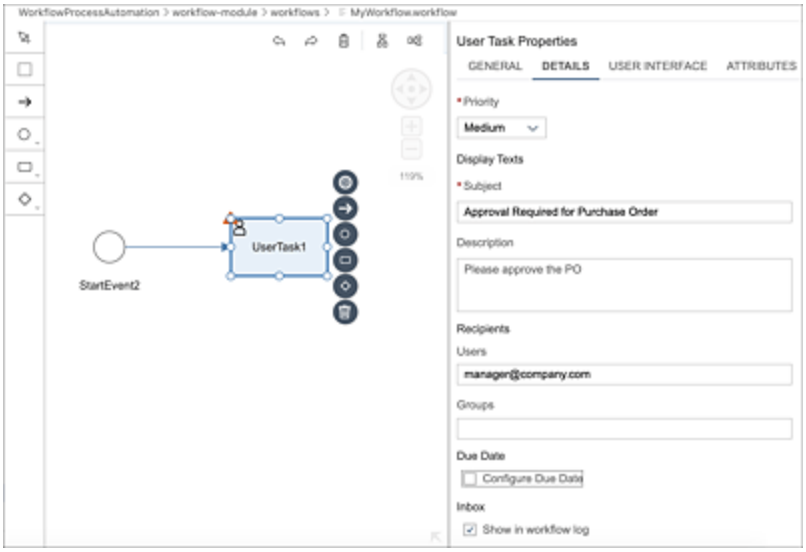


Figure 18.18 Configure Details Section of the User Task

Figure 18.19 shows how to set the **Due Date** of the user task.

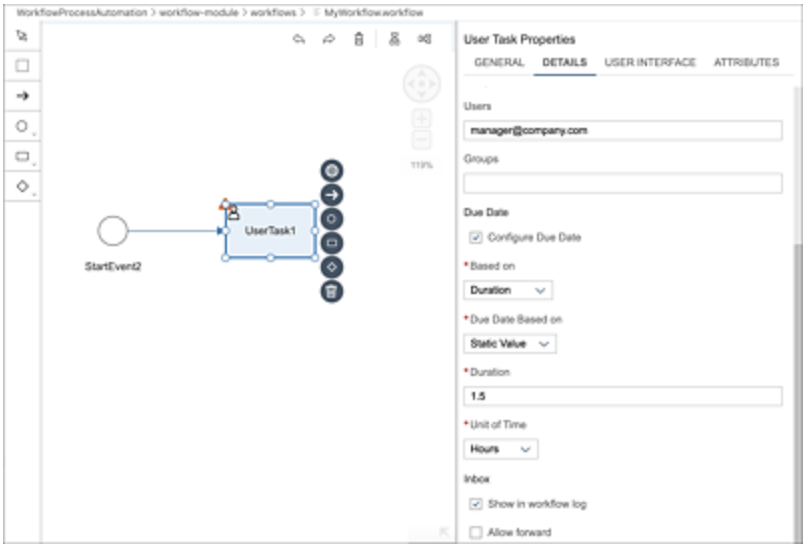


Figure 18.19 Set a Due Date for the User Task

Now let’s see the most important properties section of the user task, which is the **USER INTERFACE**. A user task always requires a UI screen either as a static form or a dynamic UI, which can enable the user to add/edit data. This can be achieved in two ways: using a form or using an SAPUI5 component, as shown in Figure 18.20.

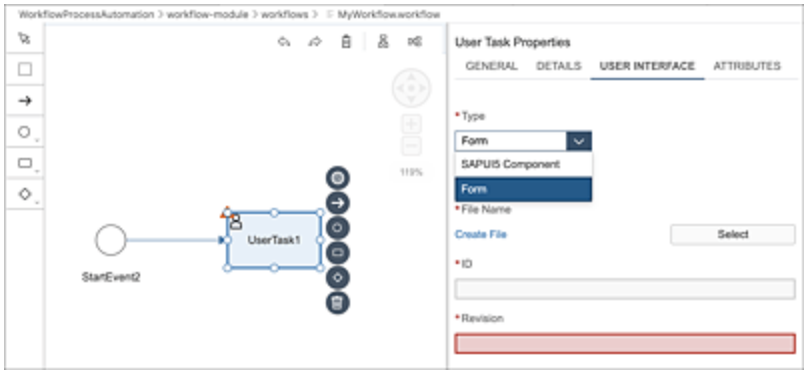


Figure 18.20 Types of Forms to Create for a User Interface

First, let’s discuss using a form. You can create a simple form, add form controls using drag and drop, and configure it. This option is a no-code option to create a rudimentary form that can be used as a UI.

Select **Form** from the dropdown under the **Type** property. If you already have a form created before, you can select and add that using the **Select** button. If not, click the **Create File** link. This option will let you create a new form. Add a **Name**, **ID**, and **Revision** for the form, and then click **Create**, as shown in Figure 18.21.

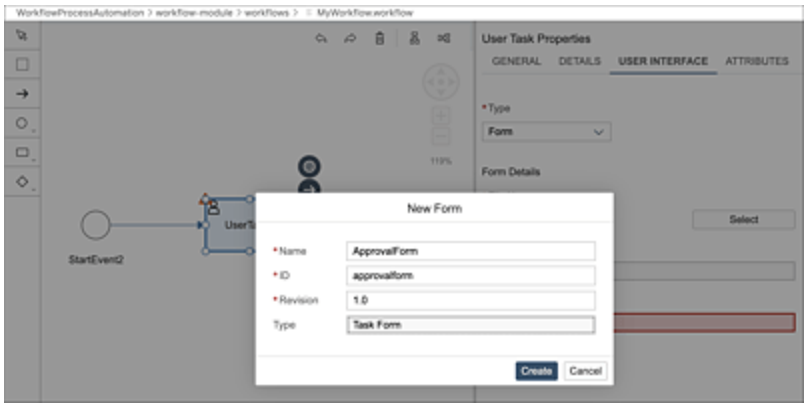


Figure 18.21 Create a New Form

You can see now a new form is created under the *forms* folder of your project and is assigned to your user task automatically. The screen editor will open automatically as well, where you can add the UI controls, as shown in Figure 18.22.

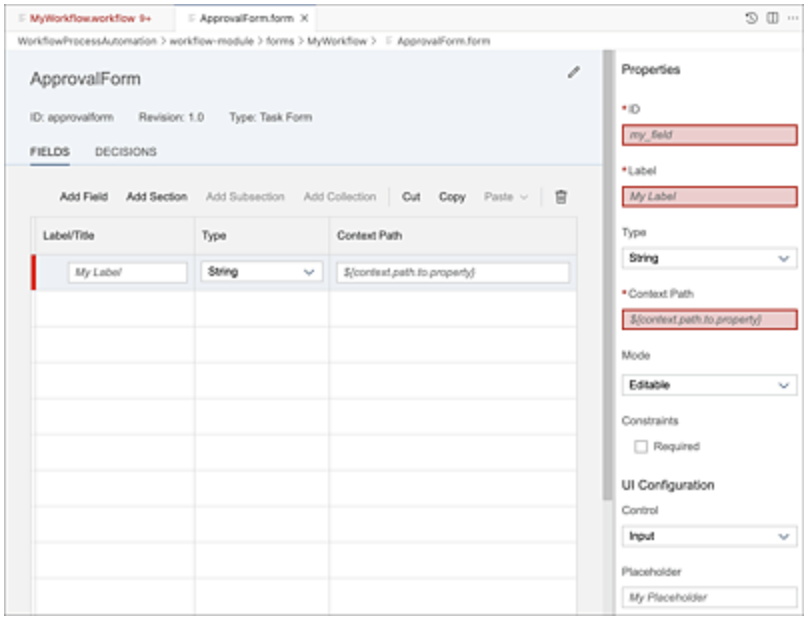


Figure 18.22 Editor Screen of the New Form

Here, you can add labels, fields, sections, subsections, and collections, as shown in Figure 18.23.

You can now add a decision to the form. Remember, this will be a user task where the user has to approve or reject the task. Go to the **Decisions** tab of the form. Here, you can add a decision, decision text, ID, and semantic color.

At runtime, the form automatically stores the user's chosen decision value in the context. To process the workflow further, you need the decision value the user has chosen, which can be done in a script task by accessing the decision value by a small code snippet:

```
${usertasks.usertask1.last.decision}
```

Here, `usertask1` represents the task ID of the user task we've configured.

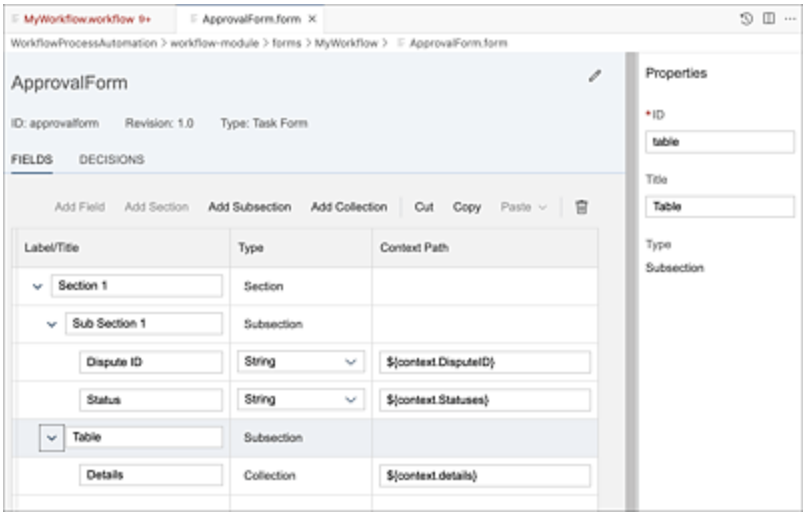


Figure 18.23 Adding Controls and Configuring the Form

Finally, you have the **ATTRIBUTES** tab, where you can add custom task attributes. This data can be read at runtime using the workflow management API and will come in handy when you need to query a specific instance of the user task and manipulate outside the workflow, such as when using an API trigger.

Now let’s move on to discussing creating a UI using an SAPUI5 component. You just saw how you can create a simple form with low or no coding required. You might need a more complex UI with more complex controls where data editing with multiple backend calls may be required. In such cases, you go for a custom UI developed using SAPUI5.

You can create a separate SAPUI5 project in SAP Business Application Studio or a UI module within your workflow project. Either way, once the SAPUI5 module is created and built in the same workspace as your workflow module, you can add this UI module in your user task by selecting the HTML5 app.

Once configured as shown in Figure 18.24, the selected SAPUI5 app will be used as the UI for the configured User Task. Here the user decisions need to be stored in the context manually and later used in the workflow just as we previously described.

The development of the SAPUI5 applications isn’t in the scope of this book, so we leave it to you to explore more on SAPUI5 development using SAP Business Application Studio.

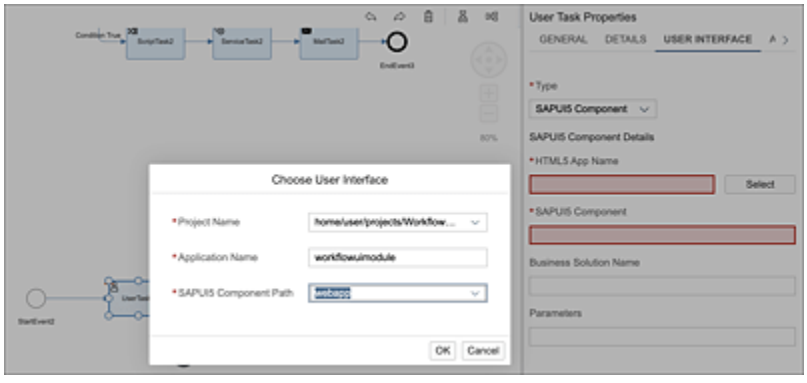


Figure 18.24 Adding an SAPUI5 Project as the UI for a User Task

Email Task

One of the most required features in a workflow is the ability to send emails. Workflows invariably have user tasks that require email triggers, but with SAP Workflow Management, the email functionality is more than just an add-on to the user task. There are possibilities of emails to be triggered for process visibility actionable insights, which you'll see in the process visibility section.

For now, let's see how an email task can be configured in our workflow. Add an email task from the workflow by clicking the **Tasks** icon in the side menu and selecting **Mail Task**. Name it "Send Email", as shown in Figure 18.25.

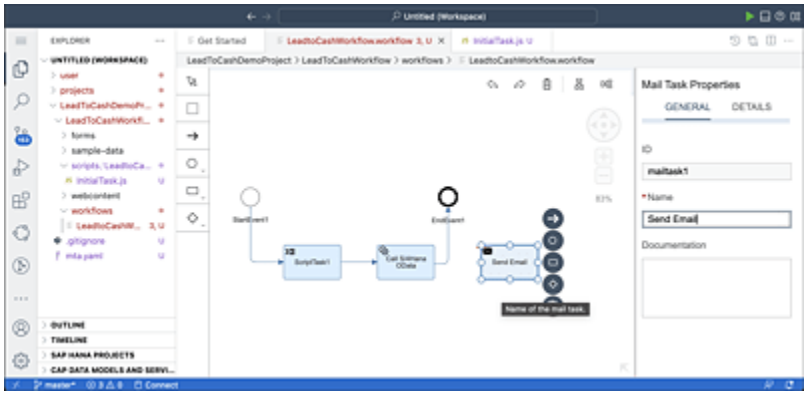


Figure 18.25 Create an Email Task

Go to the **DETAILS** tab (see Figure 18.26) in the **Properties** section. Here, you set the recipient email ID, subject, and body of the email. You can choose between having a plain text or HTML file to be included as the body of email.

Again let's connect the tasks and complete the flow, as shown in Figure 18.26.

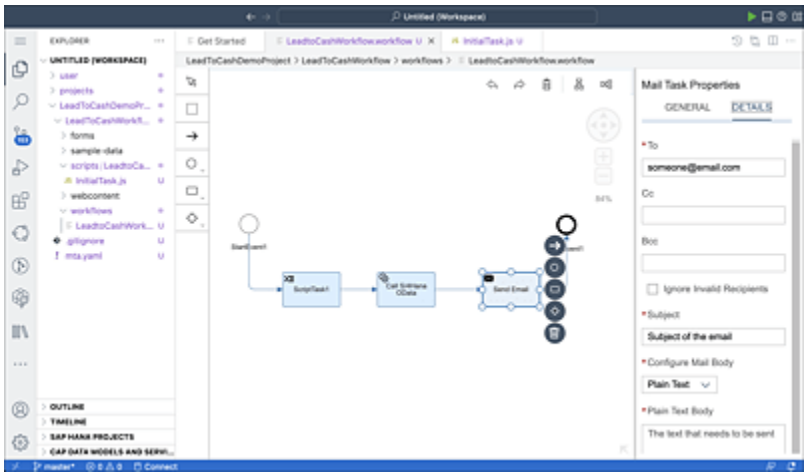


Figure 18.26 Configure the Email Task

Note

One of the important things to note for email tasks is the Simple Mail Transfer Protocol (SMTP) destination configuration. For the emails to be sent from workflows, a destination with the name `bpmworkflowruntime_mail` needs to be configured.

In addition, workflow management only supports SMTP servers that are available on the internet. It doesn’t support SMTP servers configured on premise that are available only over Cloud Connector.

18.2.3 Using Gateways

Another major feature of a workflow is the ability to control the flow by making decisions based on conditions and also to do parallel processing via gateways. There are two types of gateways, as follows:

- **Parallel gateway**
  As the name suggests, a parallel gateway gives you the ability to design a parallel flow.

  ParallelGateway1

  This gateway allows the flow to be split into two or more different flows that will be executed in parallel without checking a condition. In addition, this gateway allows all the split flows to converge back into the main flow. The key point to note here is that when you use parallel gateways for a join, all split flows should complete and reach this gateway to proceed further.

- **Exclusive gateway**
  An exclusive gateway lets the flow proceed to the next step based on a condition.

  ExclusiveGatewa
  y1

  You can make use of this gateway when the process flow needs to be directed to a path based on satisfying a condition. A typical example is a user task where a decision could be either approve or reject. Based on the action, the flow needs to be redirected.

Let’s see how to configure these two gateways. Click on the **Gateways** control on the side menu bar, select **Parallel Gateway**, and click on the designer screen. A parallel

gateway control is now placed. Give it a name in the **Properties** section.

You can now use the context menu to add different controls from this control. Say you need to branch the flow into two different sets of tasks that can be run in parallel. You create a Script Task 1, a Service Task 1, and Email Task 1 in the one branch, and you create a Script Task 2, Service Task 2, and Email Task 2 in another branch. When this is executed, the control will flow to both branches simultaneously, and all the tasks will get executed in parallel. Make sure to join the parallel flows back again with the same parallel gateway control to ensure the rest of the flow proceeds only when all the branches of the parallel flow are executed. Without this, the workflow can potentially end up in a hanging state. A parallel gateway is configured as shown in Figure 18.27.

Now let's see how an exclusive gate is configured. It's very similar to a parallel gateway except you need to define a condition to choose a branch of execution. From the menu bar, select **Exclusive Gateway** from the **Gateways** menu, and click on the designer screen. An exclusive gateway control is placed. Under the **Properties** section, provide a **Name** for the control. Now create the same set of tasks as you did for the parallel flow.

You'll see a difference here, where the sequence flow arrows show a warning symbol. This is because for an **Exclusive Gateway** to work, you need to define a condition for each of the **Sequence Flows**.

Click one of the **Sequence Flow** arrows. Under the **Properties** section, give a meaningful name for that branch of the sequence, such as "Condition True". Likewise, name

the other branch sequence as “Condition False”. You can set one sequence as default via the **Default** checkbox, which means if the condition is either false or the condition evaluates to error, there is always a way forward and the workflow isn’t stopped.

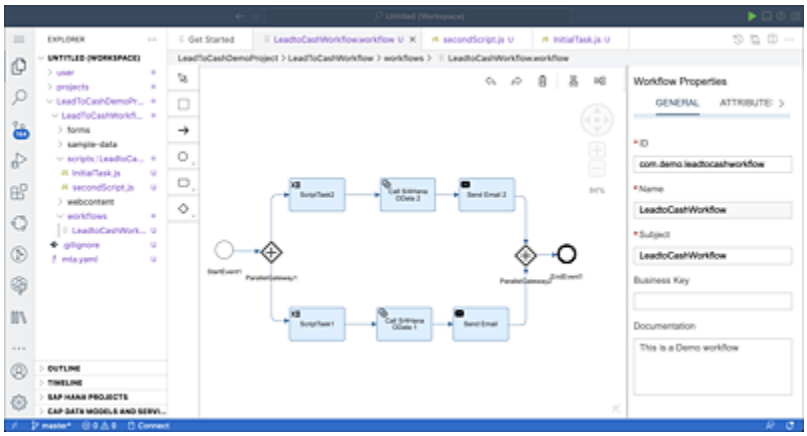


Figure 18.27 Configuring a Parallel Gateway

Figure 18.28 shows how the sequence’s properties are set. A condition can be set as a JUEL expression using a context variable as shown in the figure. If you set the **Default** checkbox, you don’t need to provide a condition.

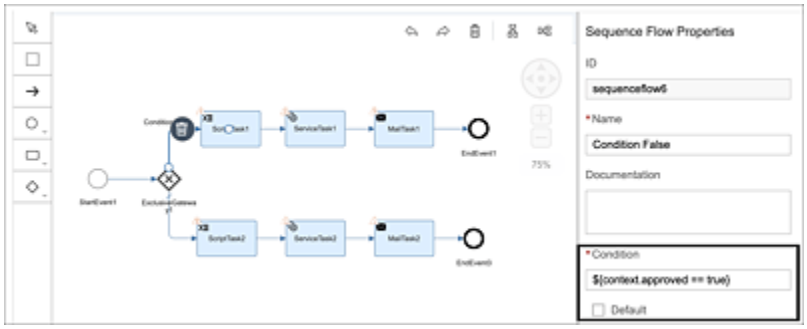


Figure 18.28 Configure the Sequence Condition for an Exclusive Gateway

An exclusive gateway is configured as shown in Figure 18.29. As we described earlier, the exclusive gateway

has a condition defined based on which the flow is chosen for execution at runtime.

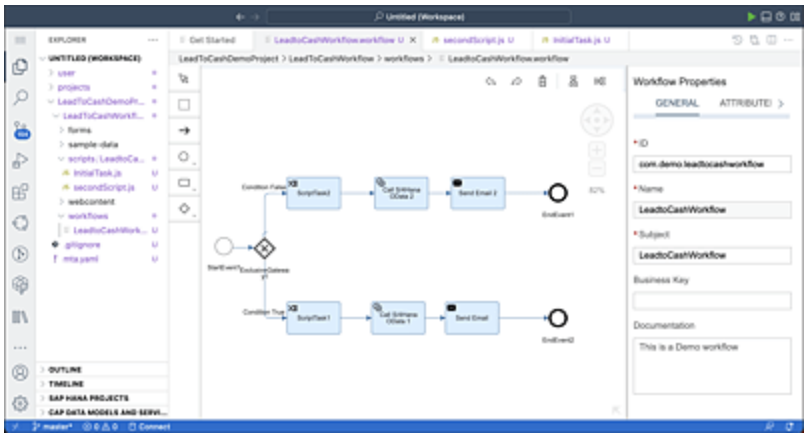


Figure 18.29 Configuring an Exclusive Gateway

> **Note**
>
> In an exclusive gateway, you need to define the condition mandatorily at least for one of the flows. In Figure 18.28, you can see a condition defined in the properties in JUEL, which translates to a Boolean. One of the paths could also bear a condition or be set as default where a condition isn't needed.

18.2.4 Events

Modern architecture principles tend to favor decoupled systems where events play a major role in connecting the processes. Every system generates events for major process steps, and it's the responsibility of the target system to react to these events, thereby achieving a decoupled architecture.

SAP Workflow Management provides capability to design event-driven workflows. This means the workflow can subscribe to specific events in SAP S/4HANA and proceed with the next steps in the process flow or make decisions. Workflows can receive events in two ways:

- Directly using SAP Workflow Management APIs
- Using an event broker such as SAP Event Mesh

When we talk about events in SAP Workflow Management workflows, we need to differentiate the concept explicitly. Events are not only the business incidents of a process but also technical design components of a workflow. Before we talk about the “actual” events, let’s get familiar with the workflow events, as follows:

- **Start event**
  Every workflow instance starts with a start event, and each instance should have one start event. When you create a new workflow module, a **Start** and **End** event is automatically created for you (refer to Figure 18.7). You can also add it manually from the side menu bar under the **Events** control.

Start Event

  You can configure a sample context as a JavaScript Object Notation (JSON) file, as shown in Figure 18.30, which will set the initial context attributes during this event execution. This could be sent as a JSON payload from a

form or an API call depending on how you want to trigger the workflow.

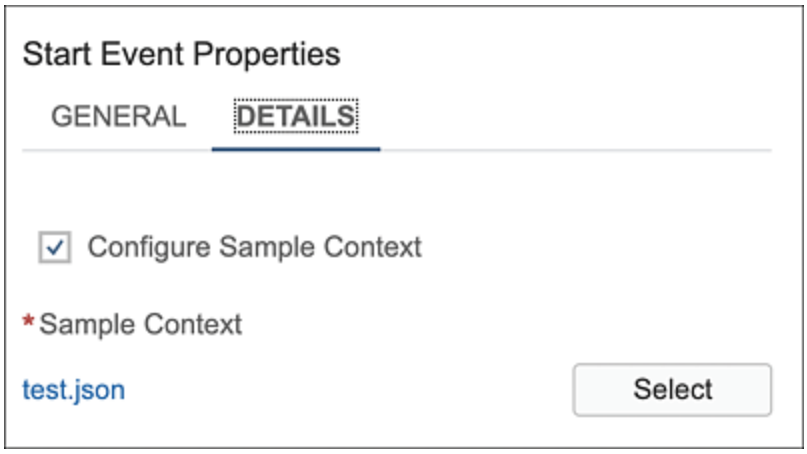


Figure 18.30 Setting a Sample Context for Start Event

- **Intermediate message events**
  This is arguably the most important of the event components of a workflow. An intermediate message event can represent an actual process event that occurs in a source system. You define an intermediate message event between the process steps. When the workflow execution reaches this step, it waits for a message, and, until then, the instance waits. The message can be delivered to this event using a workflow management REST API endpoint.

In the **DETAILS** section of the intermediate message event properties, you can set the name of the message that event expects and also capture the event’s data

payload in a context variable using a JUEL expression, as shown in Figure 18.31.

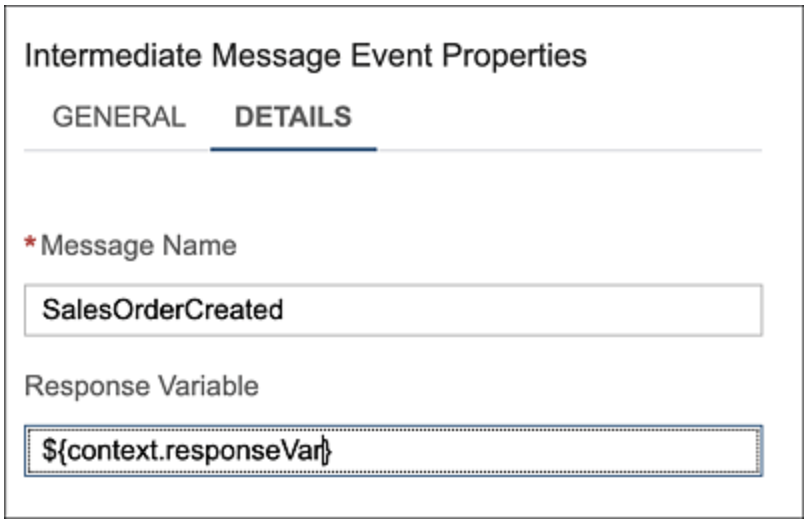


Figure 18.31 Configuring the Intermediate Event Message Properties

More details on the API and how it can be used are given in Section 18.7.

- **Intermediate timer events**
  Another key feature you need in a workflow is the ability to pause or put a wait in the process. For example, you have a set of two parallel workflow tasks, one of which is dependent on the other task completion that takes an unusually long time to complete. You can add an intermediate timer event and define a specific interval until which the workflow instance pauses from executing the next steps, as shown in Figure 18.32.

The timer duration could be configured as a static duration or an absolute time stamp. A duration will wait for the set amount of time (minutes, hours, days, months, and years), and a time stamp will translate to an actual time when the event will be triggered.

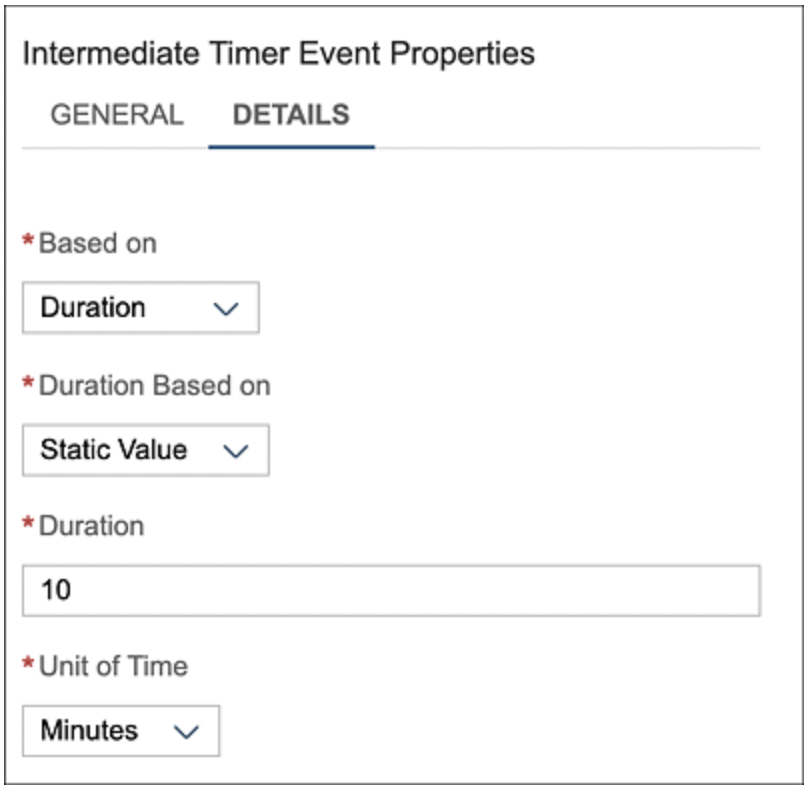


Figure 18.32 Configure the Intermediate Timer Event

- **Intermediate escalation events**
  An intermediate escalation event is used to raise an event from a referenced subflow to its parent workflow instance that started the subflow. This is especially handy when the referenced subflow needs to communicate a particular state of the instance to the parent instance with which the parent instance can take appropriate action. For example, if the referenced subflow execution has an unexpected error situation or delay in a particular step, it can send an

event to the parent instance. Based on the message, the parent instance can trigger a user task along with an email task to an operational user to take action on the issue and resolve it.

An escalation event takes only an **Escalation Code** as its configuration parameter, which is passed to the parent instance when this event is executed (see Figure 18.33). The parent instance should have a matching boundary escalation event with the same escalation code.

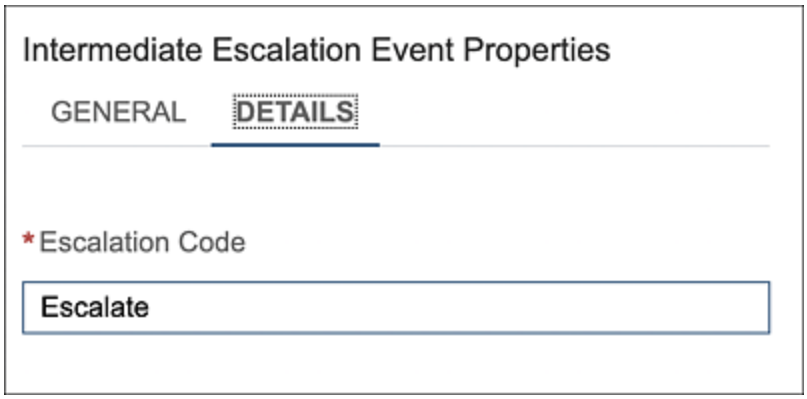


Figure 18.33 Configuration of an Intermediate Escalation Event

- **End event and terminate end events**
  Finally, for each instance start, there should be an end. An end event and terminate end event is used to achieve this. When an end event is reached, the execution stops for that branch of the flow, but if there are more branches, the execution continues until all branches reach an end event. When a terminate end event is reached, the entire execution of that workflow instance is terminated, even

though other branches haven’t finished. You can toggle an end event to make it a terminating end event and vice versa in the **Properties** section.

There are two additional events bound to the user task and referenced subflows, as follows:

- **Boundary timer event**
  A boundary timer event is used in user tasks or referenced subflows if the flow needs to be diverted to an alternate flow based on a timer. On the **User Task** context menu or **Referenced Subflow** context menu, you’ll see the option to add a **Boundary Timer Event**. You can configure the timer based on either **Duration** or **Timestamp**. The **Duration** is configurable as a **Static Value**, **Expression**, or **Task Due Date** that can be provided in the user task context, as shown in Figure 18.34.

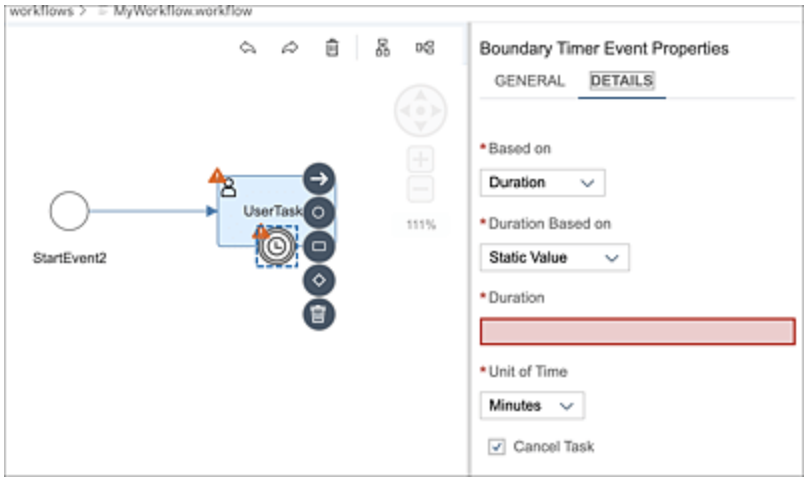


Figure 18.34 Add a Boundary Timer Event

- **Boundary escalation event**
  Similarly, a boundary escalation event can be triggered from referenced subflows to divert the process to an alternate flow based on an escalation event code. To set a boundary escalation event, click the **Referenced Subflow** control, and select the Boundary Escalation Event from the context menu, as shown in Figure 18.35.

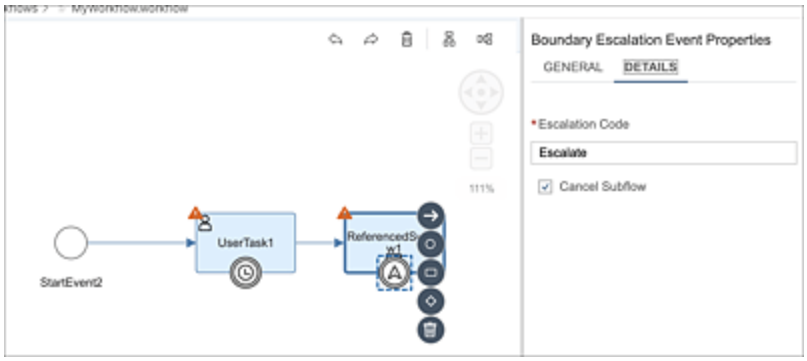


Figure 18.35 Configure a Boundary Escalation Event for a Referenced Subflow

Note that the **Escalation Code** set in this event properties needs to be sent from the referenced workflow, which then gets caught in the parent workflow instance where the escalating subflow instance is referenced from. A common property for both the boundary timer event and escalation event is the **Type**, which can be either canceling or noncanceling depending on whether the **Cancel Task** checkbox or **Cancel Subflow** checkbox is selected. This parameter, if selected, cancels the task or subflow, respectively.

18.3 Creating a Workflow Using Process Builder

Let’s now move on to see how you can use SAP Build Process Automation to create, deploy, and run a business process in SAP BTP. Once you’ve set up SAP Build Process Automation, you should see a link in the subaccount under the **Instances and Subscriptions** section of the SAP BTP cockpit. Clicking on this link will open the lobby of SAP Build (see Figure 18.36).

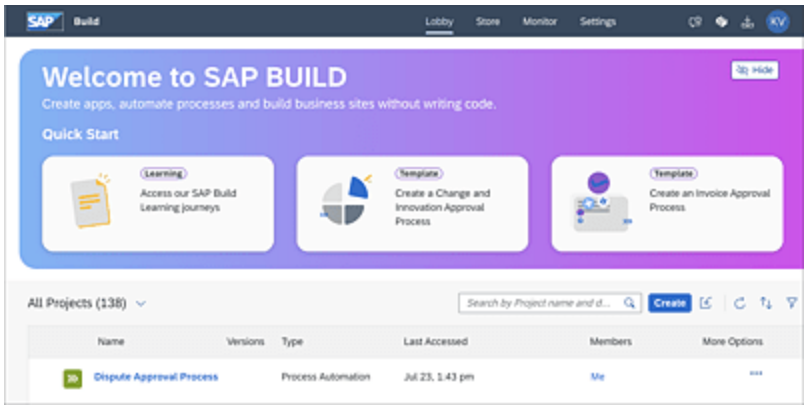


Figure 18.36 Lobby of the SAP Build Process Automation Cockpit

This is where you start your build journeys for creating apps and automation scenarios. All of your created projects get listed here from which you select the individual project to open it in the Build Editor (see Figure 18.37).

This is the place you create you process artifacts that allow you to create, build, deploy, and monitor multistage workflow processes. You could also create automation tasks from here, which could be standalone or part of a business process. (We won’t be discussing automation in detail in this

book.) Following are the key artifacts that you can create for a process:

- Processes
- Forms
- Approvals
- Visibility scenarios
- Automations
- Decisions
- Custom data types

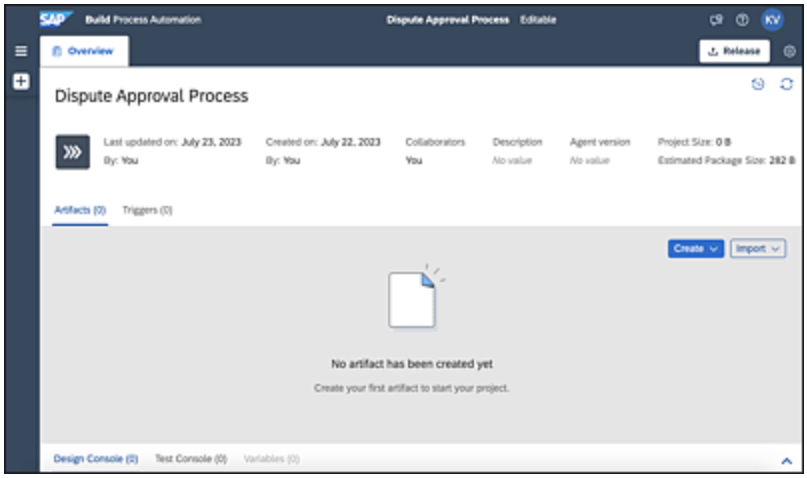


Figure 18.37 Landing Page of the Build Editor

An end-to-end process is created in process builder using a combination of the preceding artifacts along with various controls available for a workflow process. Once you create a process with all its artifacts, it’s then released and deployed in sequence from this cockpit. Let’s look at each of them in some detail.

18.3.1 Using Process Builder

From the lobby of SAP Build, you click on the **Create** button. This gives you the **Build an Application** option using **Build an Application**, **Build an Automated Process**, or **Build a Business Site**, as shown in Figure 18.38. Click on the **Build an Automated Process** option.

Next, in Figure 18.39, you see options to create **Business Process**, **Task Automation**, and **Actions**. Click on **Business Process**.

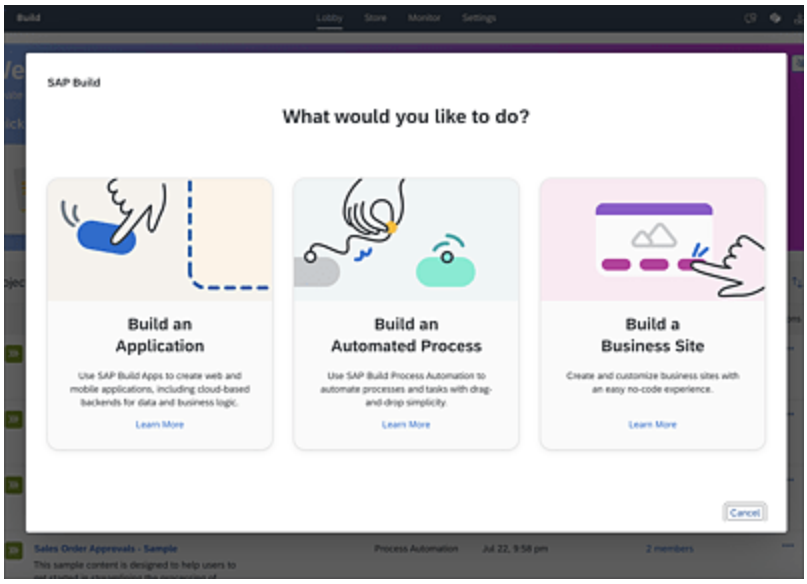


Figure 18.38 Selecting Build an Automation Process

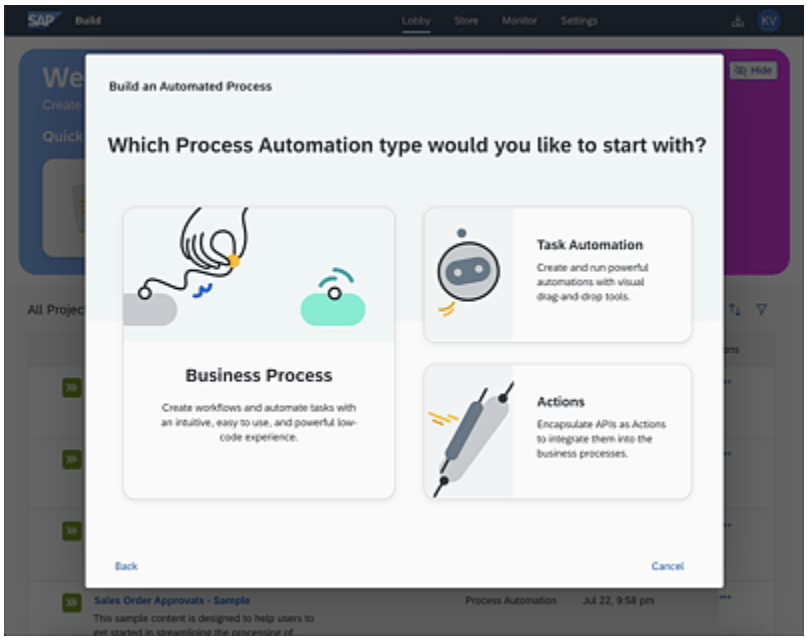


Figure 18.39 Selecting the Process Automation Type

As shown in Figure 18.40, you're prompted to name your project. Provide a meaningful name for your project (and optionally a description), and click on **Create**.

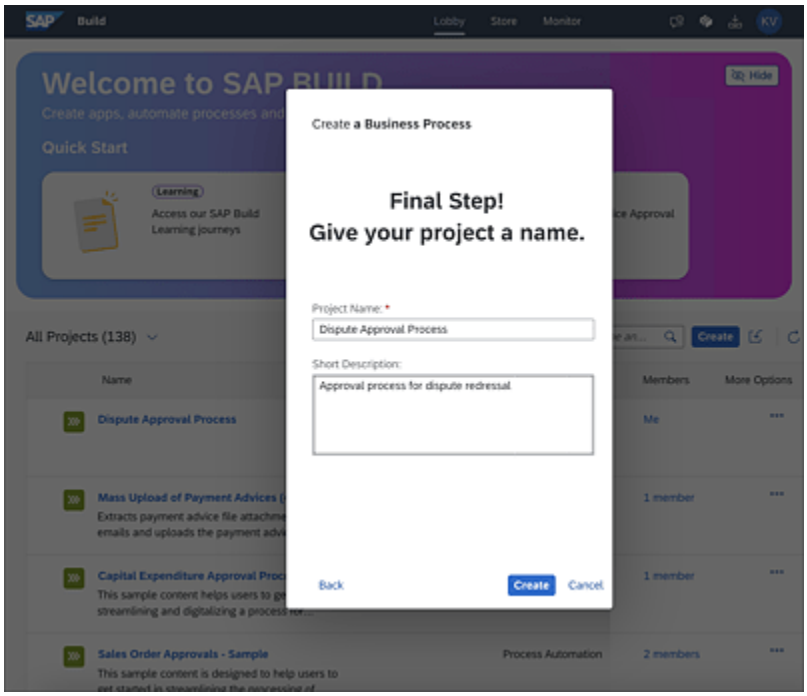


Figure 18.40 Naming the Project

This will lead to the project artifact repository, which you saw in Figure 18.37. On this screen, you create and maintain all your project artifacts, including both process artifacts and automation artifacts, as well as the process visibility artifacts for your process.

Clicking on the **Create** button here gives you the options to create various artifacts for your process. As shown in Figure 18.41, here you see options to create the following:

- **Automation**
  You can create artifacts here that can be used to define a step in your process.
- **Process**
  This takes you to the process builder that helps you design a process.
- **Form**
  This is used as a trigger to your process or as a user interaction later in the process.
- **Approval**
  This is used in an approval step from My Inbox where user interaction is required.
- **Visibility Scenario**
  This is used to provide an end-to-end view of your process with customized key process indicators and alerts.
- **Decision**
  This is used to define a business rule for your project.

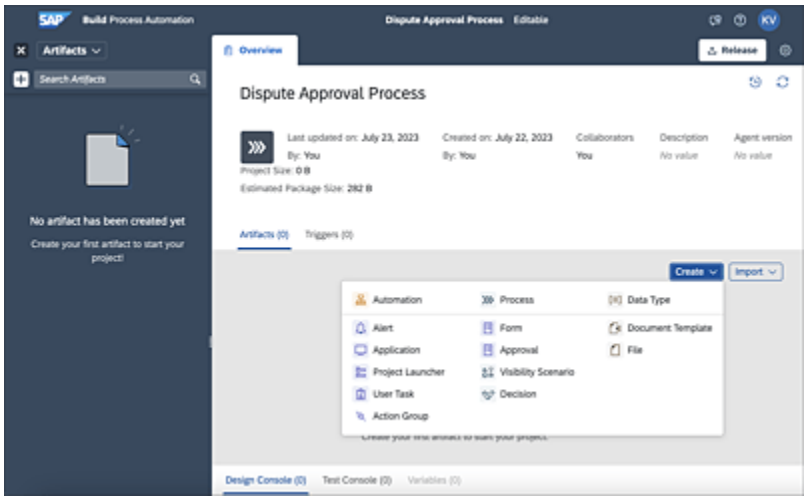


Figure 18.41 Create Your Process Artifacts

Select to create a **Process** first, provide a meaningful name to your process artifact, and click **Create**, as shown in Figure 18.42. The **Identifier** field is auto-populated, and you can either keep it as is or modify accordingly. In addition, add a meaningful description.

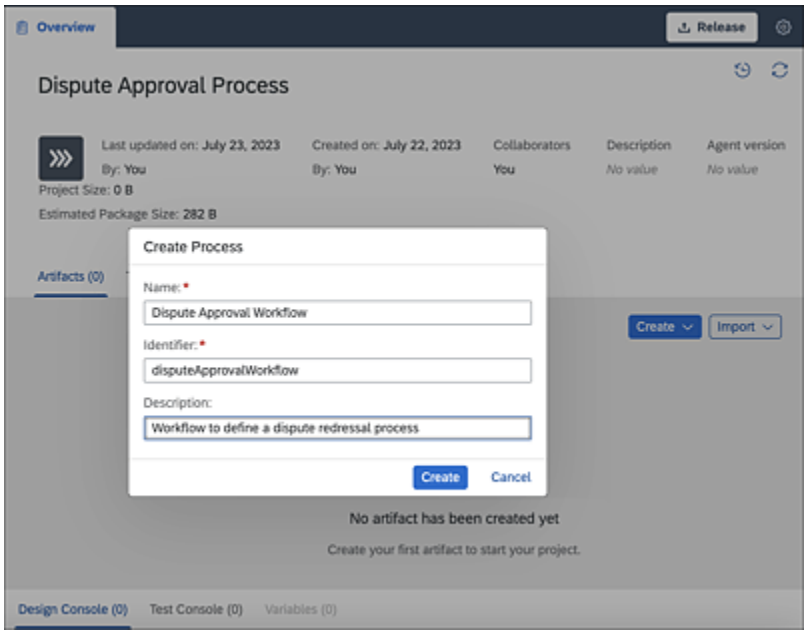


Figure 18.42 Creating a Process with a Meaningful Name

The editor now opens the process builder screen where you start designing the process. You can see there is a start task and an end task automatically added. The start task is nothing but a trigger to the process. This trigger can be defined from a built-in form or using an API call from an external service or portal such as SAP Build Work Zone.

Under the **General** section on the right side (see Figure 18.43), you can see the name of the project you had given along with a third field called **Business Key**. This is a key parameter that helps uniquely identify an instance of your process.

You can also see a couple more tabs under **Process Details**: **Inputs/Outputs** and **Visibility**. Under the **Inputs/Outputs** tab, you can define data types that need to be utilized in each step under **Input parameters**. These data types could be coming from actions or subprocesses and will be available to all steps in your process. Likewise, the outputs will be the data types that can be made available to another subprocess or workflow. Under the **Visibility** tab, you can define those data types available in your process that need to be utilized for a **Process Visibility** dashboard.

Figure 18.43 Process Builder with a Predefined Design Template

Now your preliminary template to design your workflow is ready. Let’s see the different artifacts and controls you can add to your workflow to define a logical flow. Click on the **+** symbol between the **Trigger** and **End** steps. You’ll see the various options you can select from as follows (see Figure 18.44):

- **Forms**
  Create a simple form for user input or to trigger a workflow.
- **Approvals**
  Create a simple form for making the approval/reject scenario visible in the My Inbox app.
- **Automations**
  Create bot-based automations to do mundane and repetitive tasks.
- **Subprocess**
  Modularize your complex flow into multiple subflows.
- **Decisions**
  Control your process using simple-to-build business rules.
- **Actions**
  Create API endpoints that can be consumed by the steps in your process.
- **Mail**
  Create email notification tasks in your process.
- **Controls**
  Add conditions, create multiple branches, and add a timer to your process flow.

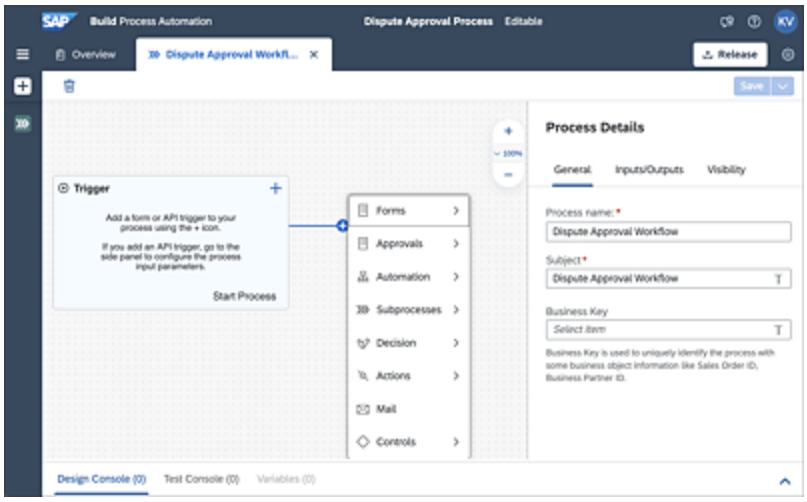


Figure 18.44 Artifacts Available in the Process Builder

Creating Artifacts

You can create these artifacts either from the project **Overview** page as shown in Figure 18.41 and use it in your process, or you can create the artifacts on the fly from the process builder itself. We've explained the creation of artifacts using the latter option.

Now that we've seen all the artifacts you can create for a process flow, let's discuss how to use these artifacts when you build your process.

18.3.2 Using Triggers

You saw in the previous section that when you create a process in process builder, a trigger task is created automatically. This is the starting point of the process. A trigger can be done using a form or from an external API call.

Click on the + button on the trigger task, and you'll see the options to assign a trigger using the following (see Figure 18.45):

- **Forms**
  Selecting **Forms** will let you create a form and assign it automatically to the task. It also gives you the options to navigate to a form builder tool where you can design the form.
- **API**
  Selecting **API** will let you configure the trigger as an API call. You can define the input data you require during the start of the process under the process **Input/Output** section as described in the previous section.

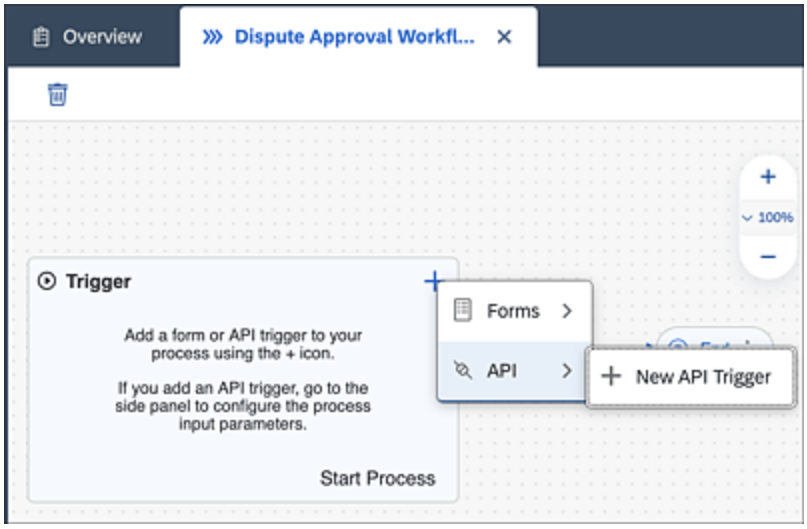


Figure 18.45 How to Use a Trigger to Start the Process

Note

An API trigger is done using standard process automation APIs from your application or service. It can be tested from any REST client. While doing so, you need to use the

definition ID of the process that will be available after you release and deploy the process.

18.3.3 Using Forms

Form, as it literally means, is a simple way of creating a user input application. This can be either a trigger to the process or a task to receive input from a process user or to define an approval scenario where a user can either approve or reject the task, by which you can define the future steps of the process.

Clicking on **Forms** let you create a new form and adds a task in the process flow (see Figure 18.46). Here you can create a simple form using drag-and-drop controls.

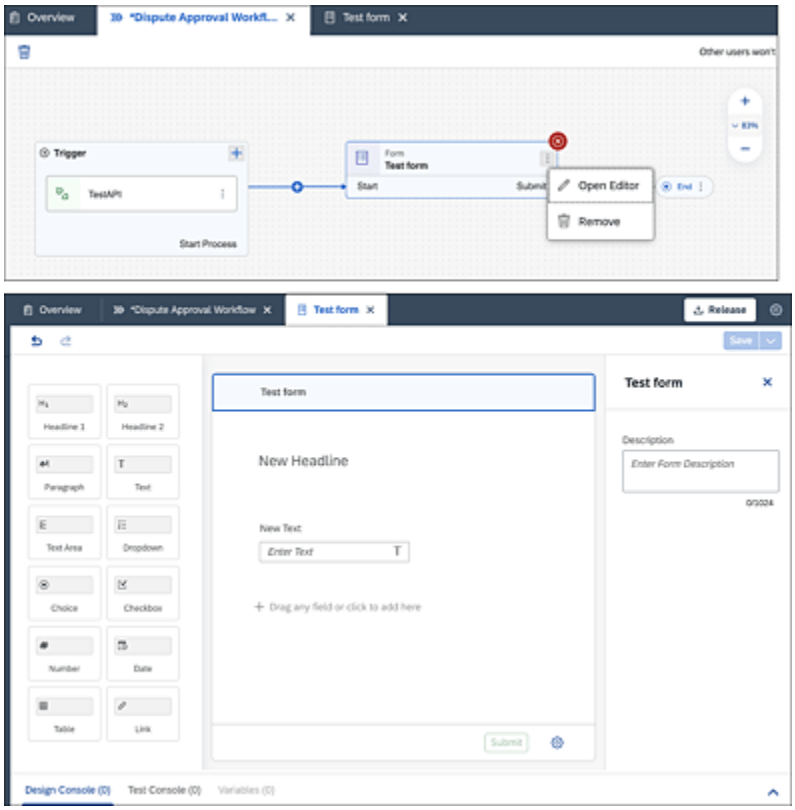


Figure 18.46 Creating a Simple Form

The form is built by dragging and dropping UI controls from the palette from the left. The palette has rudimentary UI controls such as **Text**, **Heading**, **Checkbox**, and so on, plus a few complex controls such as **Dropdown** and **Table**. Once you add all the controls you need and save it, each of these UI controls needs to be bound to a context attribute. This can be done from the process builder screen by selecting the **Form Task** and editing the **Input** and **Output** tabs of the **Properties** section.

A form has only one action, which is **Submit**. The click of this action button submits the data to the context and continues to the next step in the process.

18.3.4 Using Approval Forms

Approval forms are another type where the user gets to choose a decision either to approve or reject. The action of the user decides the next step in the process. The form is built in a similar way as a normal form, as explained in the previous section.

In the outlined portion in Figure 18.47, you can see two actions: **Approve** or **Reject**. The **+** button emanating from both of these can be used to proceed the flow based on the selected decision by the user. For example, if **Approve**, the process moves to the next step; if **Reject**, the process can be diverted to a previous step.

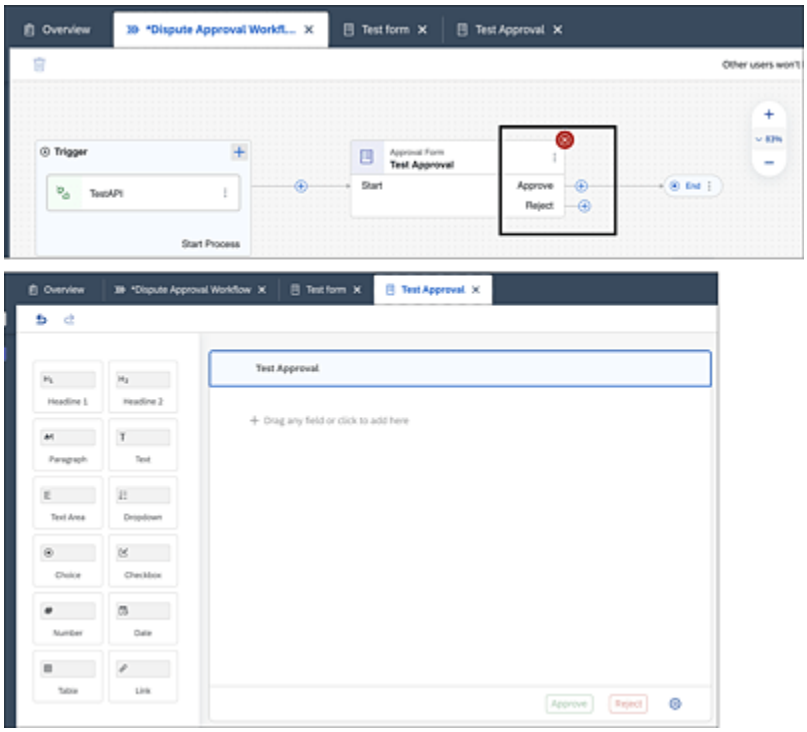


Figure 18.47 Create an Approval Form

Otherwise, creation of the approval form is the same as creating a simple form.

18.3.5 Using Automation Tasks

You can create automation projects either from the lobby or from the process **Overview** of the selected project using the menu shown earlier in Figure 18.41. The creation of an automation itself isn't explained here as this is out of the scope of this book. Instead, we'll discuss how to use a previously created automation as a task in your process. There are two ways to add an automation task:

- From the + button menu, chose **Automation**, as shown in Figure 18.48. The process builder will list the existing automation tasks within your project that you may have

created earlier and let you select. In Figure 18.48, **Trigger Dispute Creation** is an already created automation that is available for you to select.

- You can also choose to create a **New Automation** from the menu. If you select this option, process builder will automatically try to detect the agent version installed in your machine before proceeding to the next step. Otherwise, you can skip this and select the version of the agent installed manually. In the next step, enter the name of the task, and click **Create**.

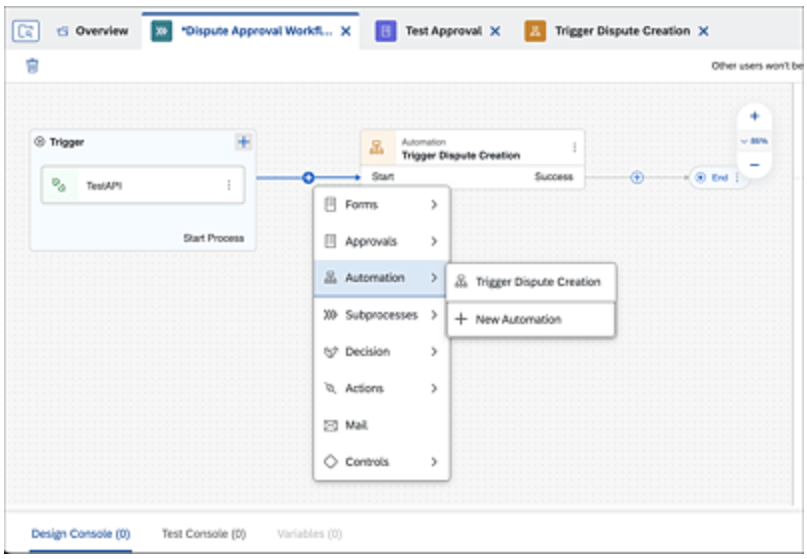


Figure 18.48 Adding an Automation Task in Your Process

Agent Availability

At the time of writing, the agent is available only for a Windows-based desktop; there is no agent for the Mac OS. If you need to run your bot on Mac, you need to make use of a virtual desktop infrastructure (VDI).

18.3.6 Using Decisions

The majority of the business process depends on some sort of rules to proceed or determine the next course of action, such as choosing an approver based on the sales order amount, choosing between manual and automatic approval based on the purchase order quantity and amount, and so on to name a few simple ones. This can be achieved easily using the **Decisions** option that lets you create a table- or text-based decision matrix that can be consumed in your process as a decision task (see Figure 18.49).

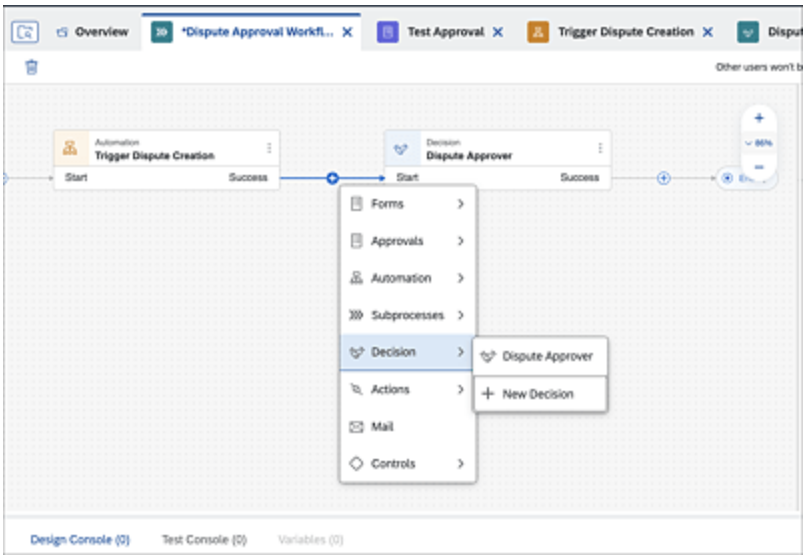


Figure 18.49 Creating a Decision Task

A decision task is made up of a set of decision rules that you need set up for this task to execute. But before you create a rule, you need to define the input and output parameters for this task that you'll use to hold the output result from a given set of data inputs, as shown in Figure 18.50.

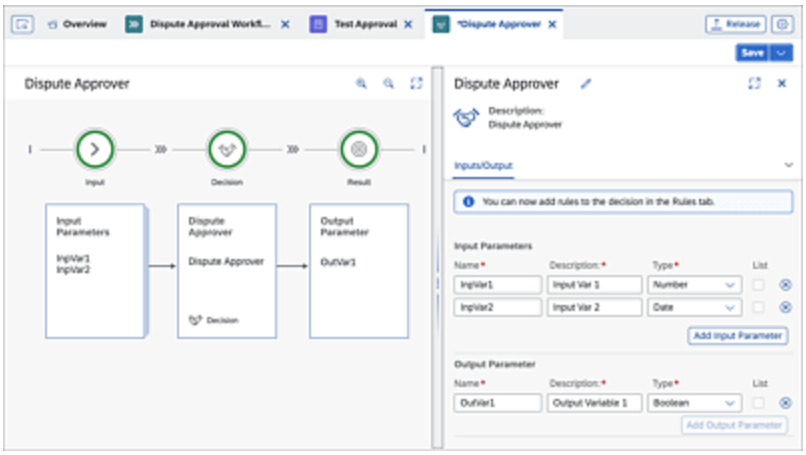


Figure 18.50 Creating and Editing the Decision Rules

To configure the rules, you need to edit the decision task. For this, either open the **Decision** task from the **Overview** page, or use the context menu in the task itself in the process (three dots) to open the editor.

As you saw earlier, you have two options when creating a business rule:

- **Decision Table**
  Multiple rule expression can be combined to produce a specific output in a tabular fashion.
- **Text Rule**
  A simple if-then condition check is used to output a decision as a specific decision.

Let’s see how both look in action. Navigate to the **Rules** tab in the decision editor (refer to Figure 18.51), and click **Add Rule**. Sometimes the tab may be hidden as a dropdown based on the screen width. If you don’t see the tab name in the screen, use the dropdown option to select the tab.

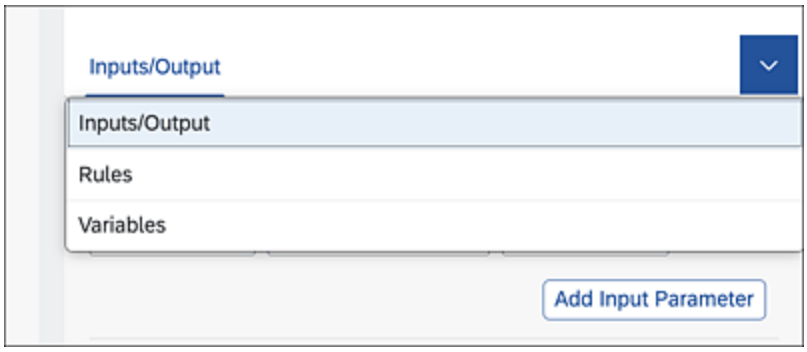


Figure 18.51 Navigating to the Rules Tab

In Figure 18.52, you can choose between **Decision Table** and **Text Rule**. Select **Text Rule**, give a meaningful name to your rule, and click **Next**.

Under the **Configure Results** step, select the **Result Vocabulary** field where the result needs to be published. Proceed to the next step, review, and create the rule.

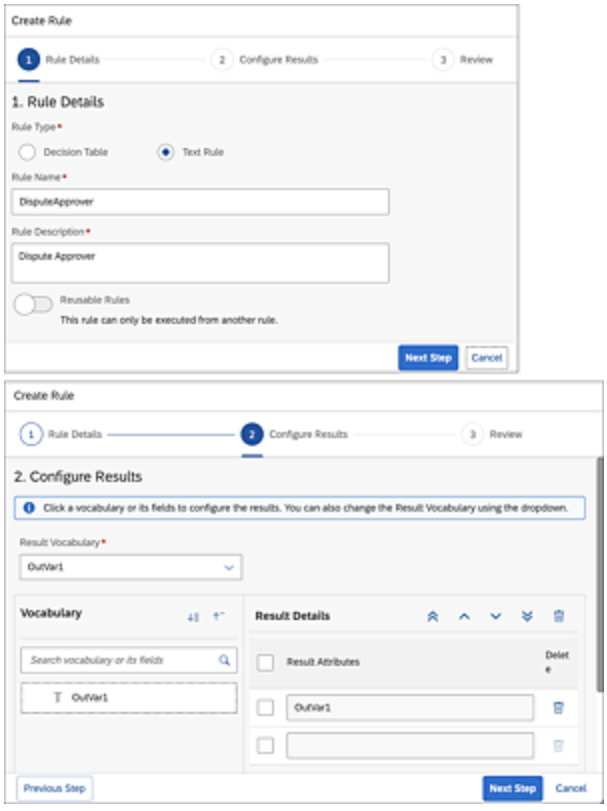


Figure 18.52 Creating a Text Rule

Once you create the rule, you can configure the if-then condition. Using operators available to select from the context menu and context help, you can set a simple rule. For example, you have the **InpVar1** which is an input variable available in the context. You can check if it's greater than 10,000 using the greater than symbol, which the system will prompt you to choose when you're editing this field, as shown in Figure 18.53.

Save the rule, and now you're ready to use the business rule in your process.

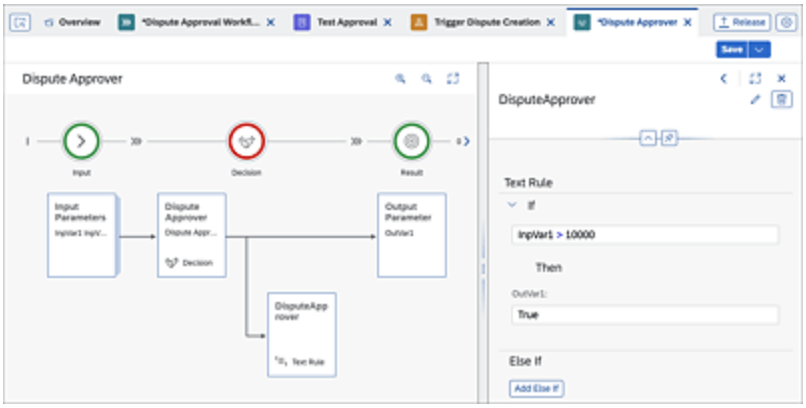


Figure 18.53 Creating a Rule Condition and Configuring the Output

Errors

While saving your rule project, if there are any errors, they will get listed at the bottom of the process builder window under **Design Console**.

Now let's see what a decision table looks like. This time, after clicking on **Add Rule**, select **Decision Table**, and provide a meaningful name, as shown in Figure 18.54.

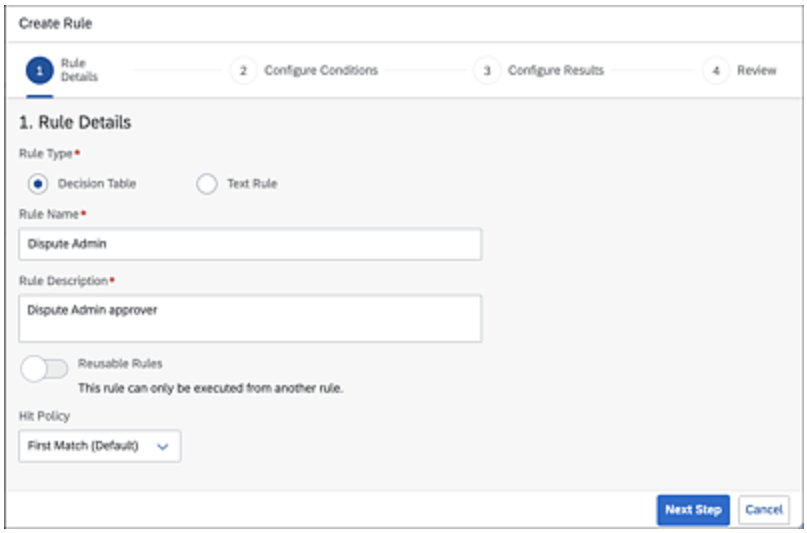


Figure 18.54 Creating a Name for the Decision Table Rule

In the second step, configure the input variables and the conditions, as shown in Figure 18.55.

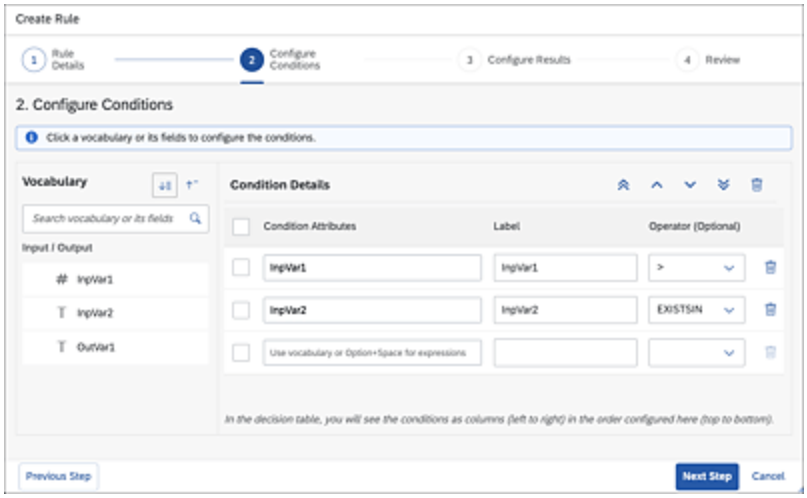


Figure 18.55 Configuring the Input Condition Attributes

In the third step, configure the output variables, as shown in Figure 18.56.

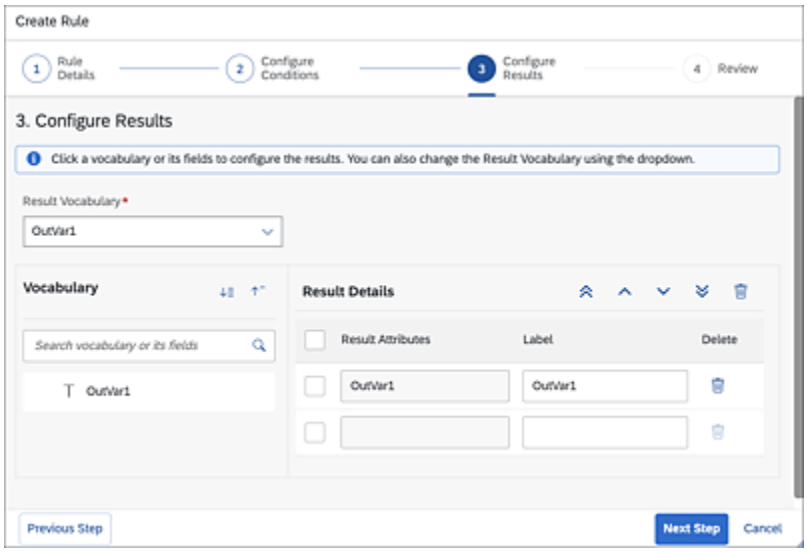


Figure 18.56 Configuring the Output Condition Attributes

Finally, review and create the rule, as shown in Figure 18.57.

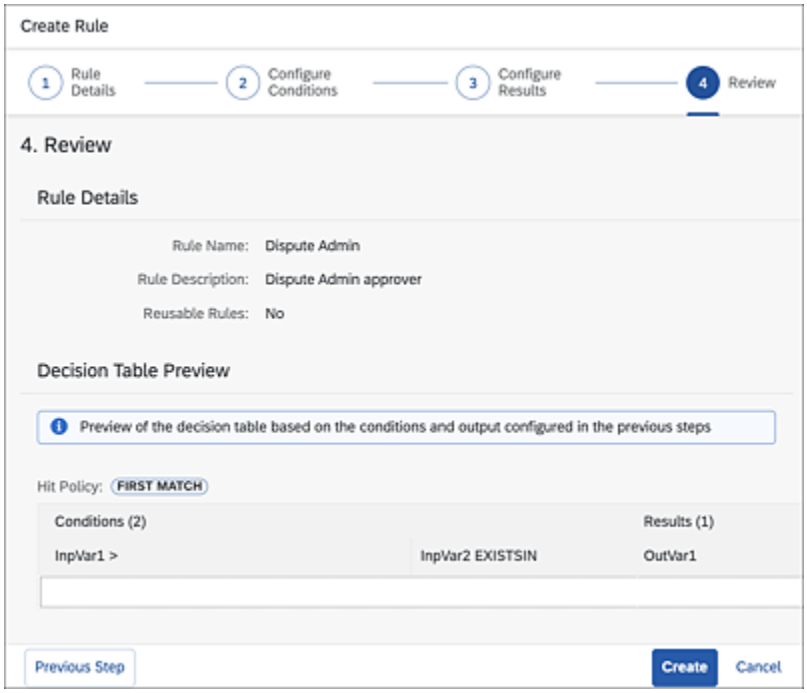


Figure 18.57 Reviewing and Creating the Rule

After creating the decision table, now let’s configure the conditions as you did before for text rules. You can add multiple rows of values that represent multiple rule

conditions and their corresponding results in a tabular fashion. Compared to a text rule that evaluates a single if-then condition, a decision table can evaluate multiple conditions based on the input values. Each of the evaluating conditions and results is represented by a row in the decision table, as shown in Figure 18.58.

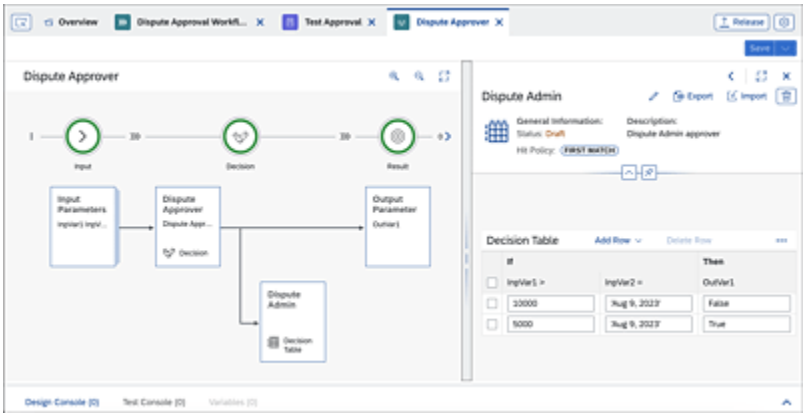


Figure 18.58 Configuring the Decision Table

Now that we've created the business rule either as a text rule or a decision table, you can use it in the process to make a runtime decision.

18.3.7 Using Subprocesses

The ability to modularize a process flow is one of the most attractive design features one could hope for when defining a complex process. This is exactly what subprocess tasks bring into the design of processes. You can divide a complex flow into a main process of multiple subprocess flows and call them from each other based on the requirement there by also reusing the subprocesses wherever needed. You can pass data from the calling flow to the called flow and back too. Let's see how this is done.

You can either create a subprocess from the lobby the same way as you create a process or from your process flow itself using the + symbol where you want to insert/call the subprocess (see Figure 18.59).

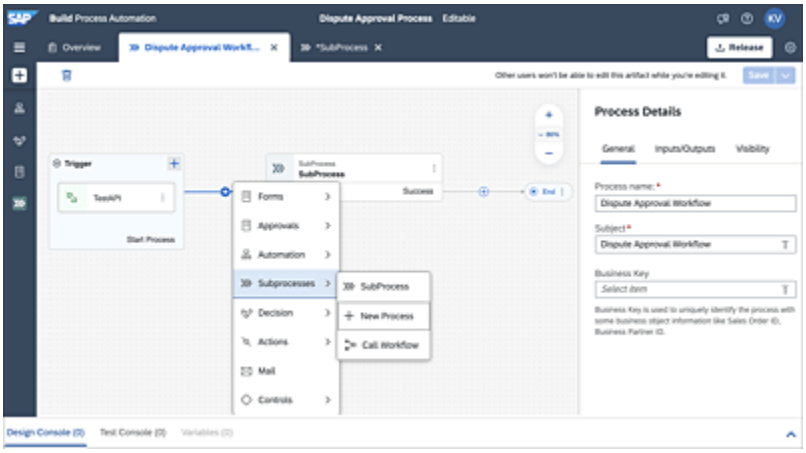


Figure 18.59 Adding a Subprocess

Note

The subprocesses that you want to use in your main process flow should be created under the same project.

In the menu, you can see all the other processes you've created listed so they can be added as your subprocess (**SubProcess**). You can create a + **New Process** using the option and start designing it.

The most interesting option is **Call Workflow**. You saw how you can create workflows using SAP Business Application Studio in Section 18.2. SAP Build Process Automation gives the option to call these workflows as subprocesses from your process. This is a very convenient option that provides great flexibility to develop the workflow using either of the options and call them with ease.

If you select **Call Workflow**, you're presented with a workflow menu from which you can click on **Add** to add the required SAP Business Application Studio–based workflow (see Figure 18.60).

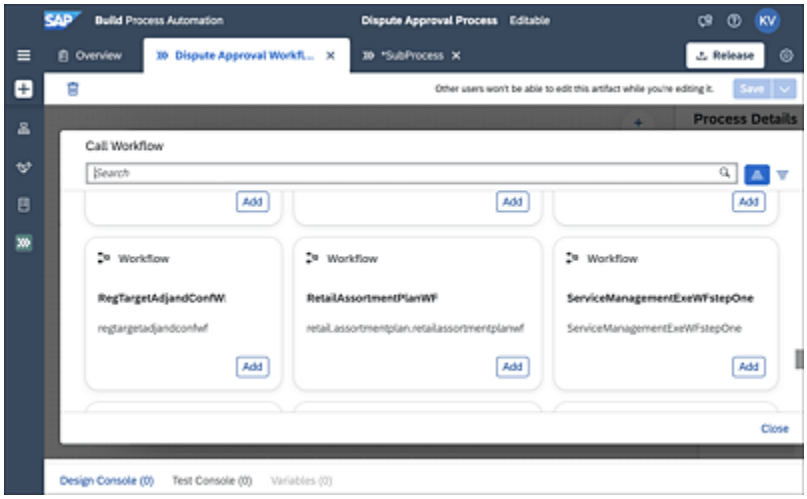


Figure 18.60 Adding an SAP Business Application Studio–Based Workflow as a Subprocess

> **Note**
>
> When you add a subprocess between two steps—say step A and step B of the main process—once step A triggers the subprocess, it should run end to end without errors to come back to the main flow and start step B.

18.3.8 Using Actions

Two of the main requirements from a workflow are reading data from a given data source and writing data. The data source could be anything, such as an SAP system, database, or any legacy or cloud-based system of records. Workflows designed in SAP Build Process Automation provide the

ability to make such create, read, update, delete, query (CRUDQ) operations by using an API call either as an OData or a REST call to the data source. Actions provides you this ability to add such API endpoints and use them in your process flow.

From the **Lobby**, click on **Create** to see the different **Process Automation types** you can create, as shown earlier in Figure 18.39. Here, you see the **Actions** option, which is where you start adding API references that can be used in your processes.

Once you click the **Actions** tile, you'll be presented with three options:

- **Business Accelerator Hub**
  Use this option to choose an API from the SAP Business Accelerator Hub (*https://api.sap.com*) to choose a specification of the listed SAP standard APIs.
- **Upload API Specification**
  Use this option to add a custom API specification, such as Swagger or an SAP OData metadata file, which is of the format XML, EDMX, or JSON by either dragging and dropping or browsing and adding. This is particularly useful when calling custom OData or REST endpoints exposed by third-party source systems.
- **Graph**
  Use this option to directly add API specifications from a connected SAP Graph instance. SAP Graph is the API management capability withing SAP Integration Suite.

Let's see how you can add and use an OData API in the process flow, as follows:

1. In the **Action** menu, choose **Upload API Specification**. Here you can either drag and drop your specification file or browse the location and add it (see Figure 18.61).

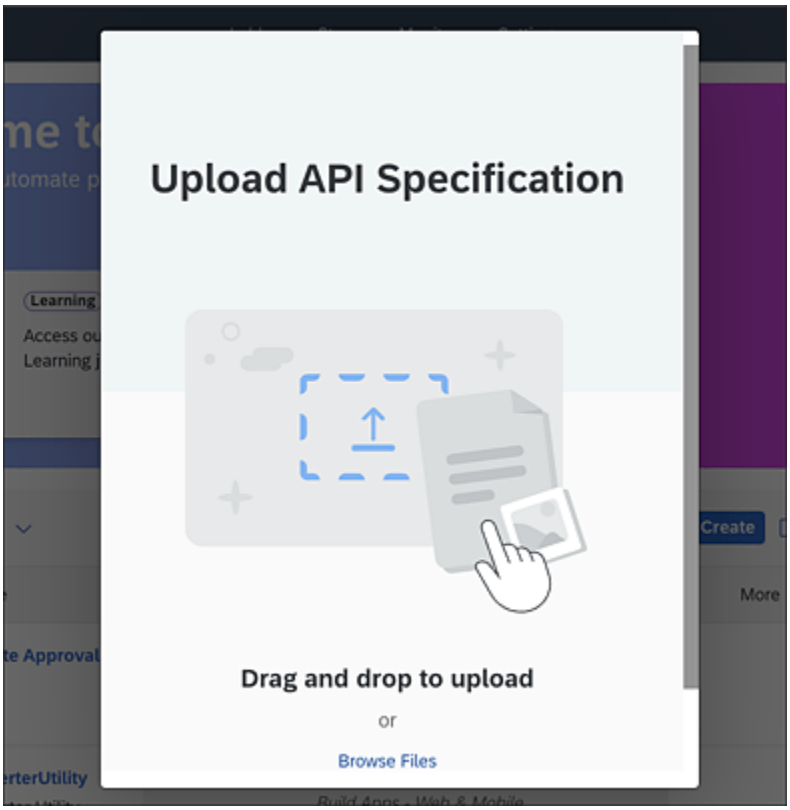


Figure 18.61 Uploading an API Specification

2. Let’s add an OData specification in EDMX format. Take an example of `API_SALES_ORDER_SRV`, which is a standard OData specification available in SAP S/4HANA. Save the metadata file to your local machine as an XML file.
3. Click **Browse Files**, and add this specification file. Click on **Next** (see Figure 18.62).

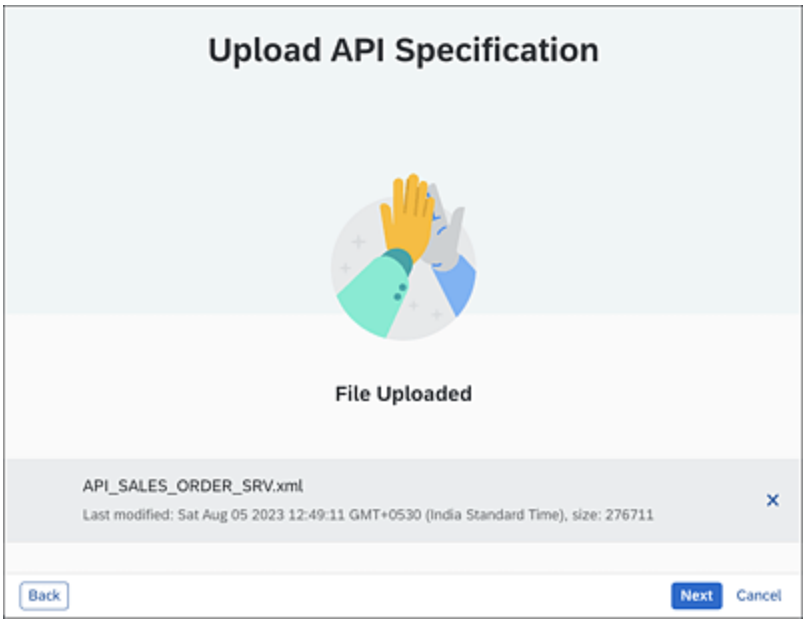


Figure 18.62 Uploading the API Specification

4. Give the action a **Project Name**, and click **Create** (see Figure 18.63).

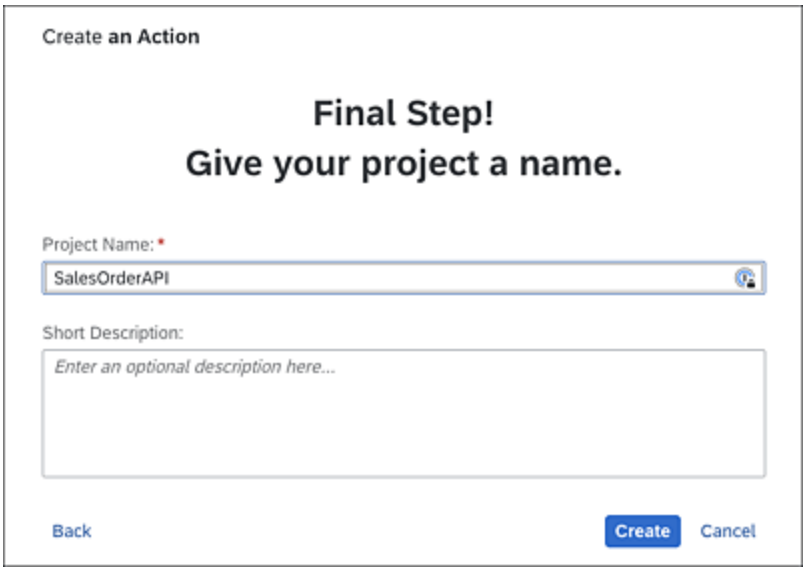


Figure 18.63 Naming the Action Project

5. The action project is now created, and you can select, add, and configure your required actions based on the

entity sets available in your API. Let’s select the **GET** action for **/A_SalesOrder** entity (see Figure 18.64).

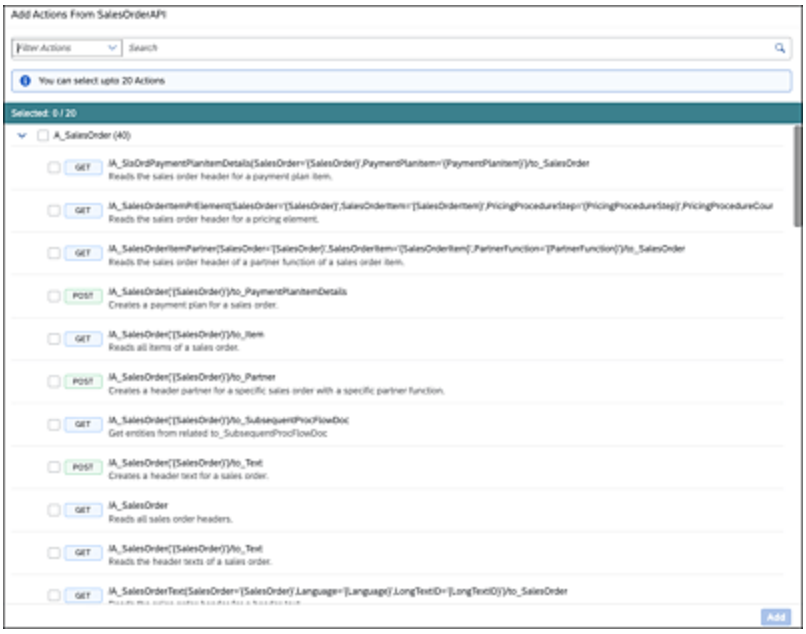


Figure 18.64 Selecting the Actions

6. In the next step, you can select any **Input** and **Output** parameters, such as standard OData parameters (**$skip**, **$top**, etc.) or the OData entity set parameters, as shown in Figure 18.65.

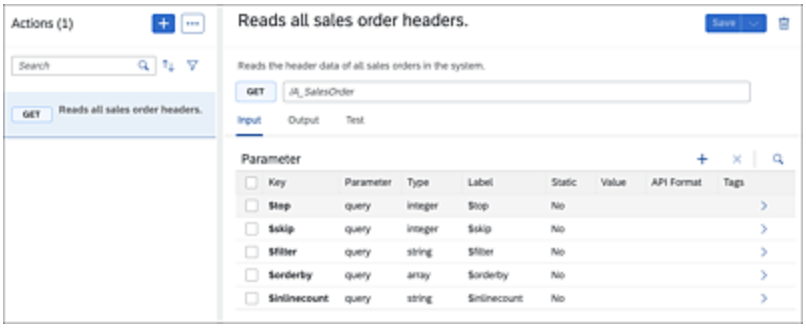


Figure 18.65 Configuring the Selected Actions

7. You’re now ready to release and publish (see Figure 18.66) the action.

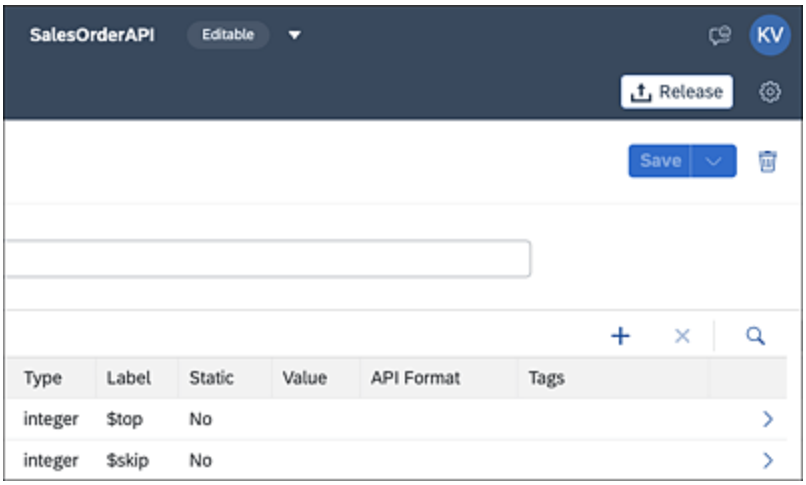


Figure 18.66 Releasing the Project

These two actions do the following:

- **Release** deploys your action to the SAP BTP runtime.
- **Publish** (see Figure 18.67) makes your action available to be added in your process using the **Browse** action.

Figure 18.67 Publishing Your Action

8. Now go to your process, and click on the + icon where you want to add this action. From the menu, select **Actions**, and click **Browse**. Search for your action project, and you can see that the action is now listed. Click **Add**, as shown in Figure 18.68.

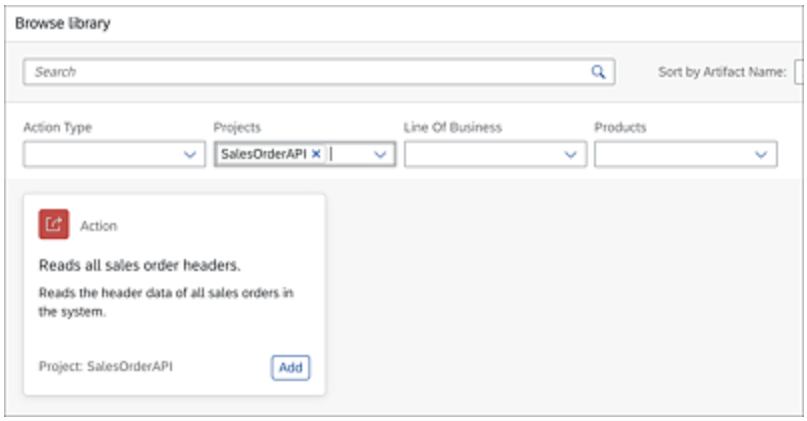


Figure 18.68 Selecting and Adding the API

You've now added an API action successfully to your project.

Note

To make your action call, it's mandatory to configure the **Destination variable**. This is the destination configuration to the SAP S/4HANA system in this case. In general, this will be the destination configuration to the source system from where this API needs to fetch data.

Now, add this action as a task in your process per Figure 18.69, and define the **Destination variable** for this action.

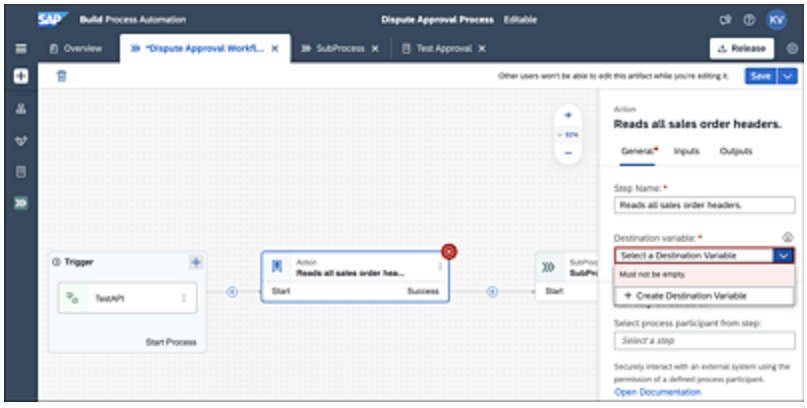


Figure 18.69 Adding the Action to Your Process

Under the **Inputs** tab, you can configure the request parameters that you configured in the previous steps. Now click the **Outputs** tab under the action configuration. Here, you can see the entire data structure the API returns after a successful call to the data source. This data can be easily consumed in a future step, say to be mapped in a form or passed to another subprocess for further processing.

You've successfully configured an API action and consumed it in the process.

18.3.9 Using Mail

Another task you can create in your process is a mail task. As the name suggests, this will trigger an email to the assigned user. From the **+** icon, select a **Mail** task.

On the right panel, select an email ID from the process context for the **To** field. You can add **CC** and **BCC** as additional options. Click on **Open Mail Body Editor** (see Figure 18.70) to add the body of the email, and add additional variables from the process context.

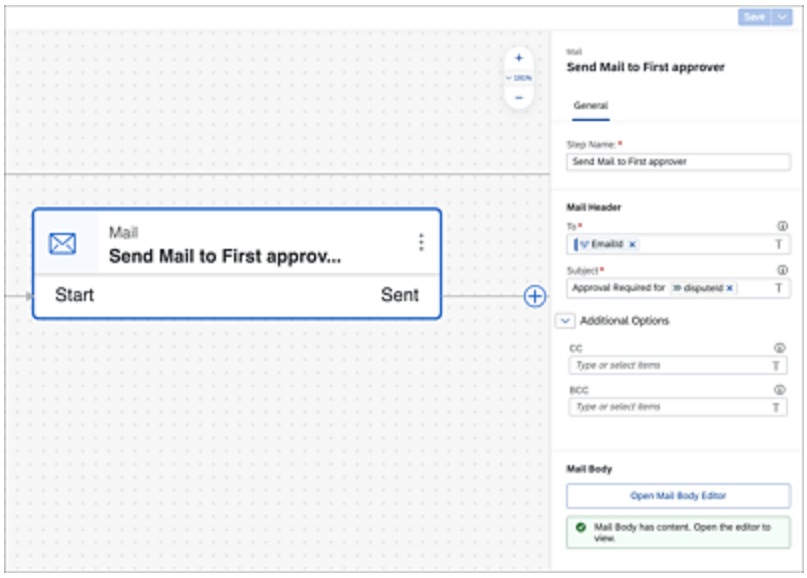


Figure 18.70 Creating a Mail Task

In the email body editor, you can add the email content text that needs to be sent to the user. You can directly reference variables from the context wherever you need to add dynamic variables, as shown in Figure 18.71.

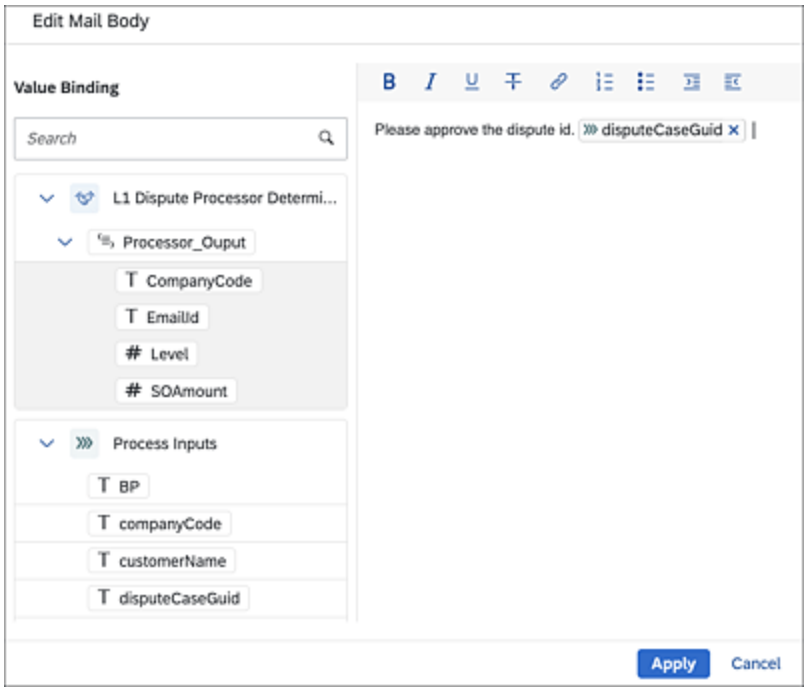


Figure 18.71 Editing the Mail Body

18.3.10 Using Controls

Now for a workflow process, it’s required that the main flow can branch into multiple flows, with or without a condition. You need such controls to channel the flow. For this, SAP Build Process Automation provides three different controls (see Figure 18.72).

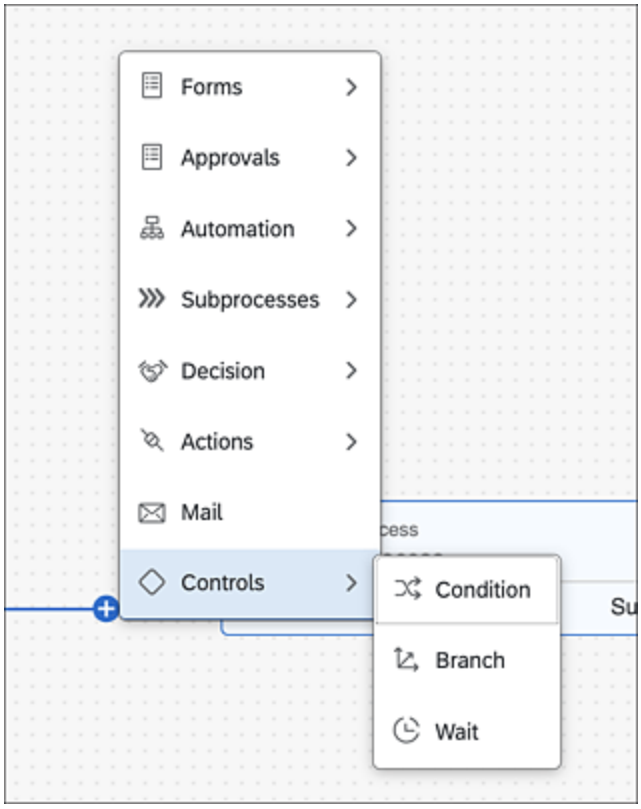


Figure 18.72 Controls in a Process

The three controls are described as follows:

- **Condition**
  A condition control is used to split a flow into two branches based on the condition evaluating to true of false. Once you add a condition to your flow, you can do the following.
  - Rename the **Step Name**.

- Rename the “If” **Branch Name**.
- Add a condition to the “If” branch using the **Open Condition Editor** button, as shown in Figure 18.73. This condition should be based on the value of a process input’s parameters you configured during the creation of the process. You can use comparison operators in this value to be evaluated to true or false.

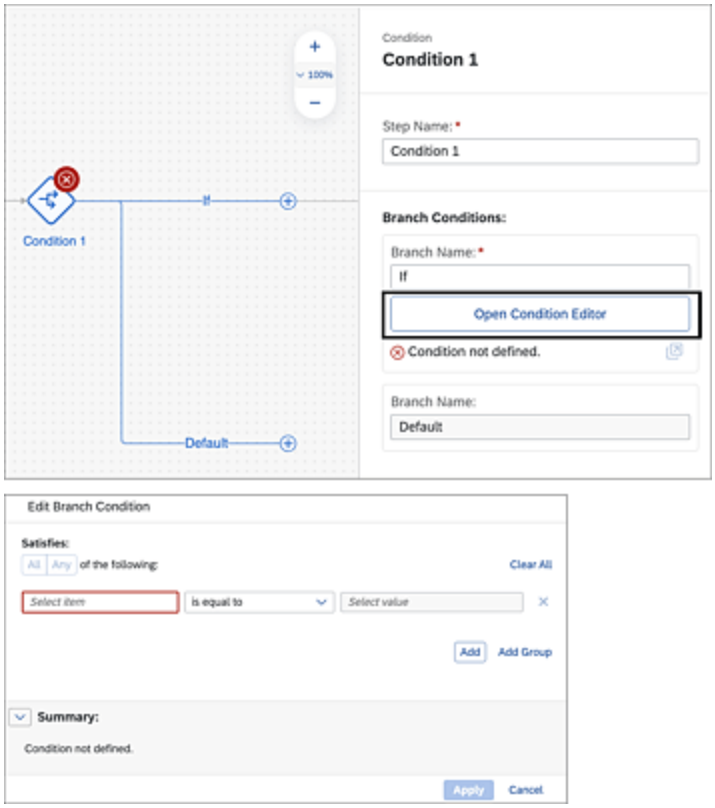


Figure 18.73 Condition Control

- **Branch**
  A branch is a control that can be used to split a flow into multiple flows. All these flows run in parallel and converge at the next process step. You can add up to 10 branches per control. Referring to Figure 18.74, you can see the branch control can create split flows that perform some

tasks in parallel and converge back at the subprocess task after executing all the branched tasks.

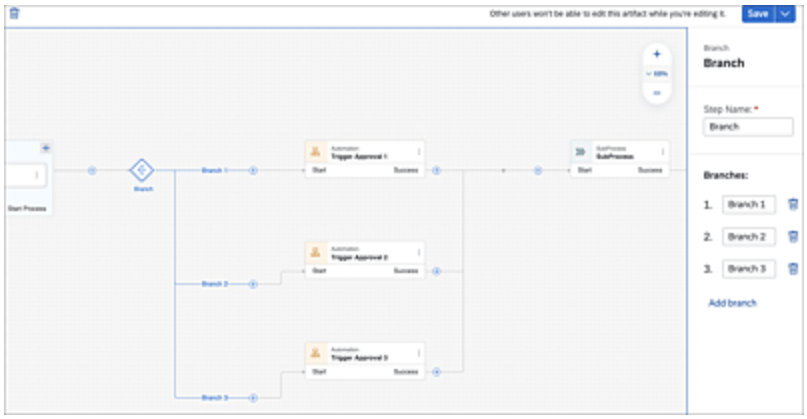


Figure 18.74 Branch Control

> **Note**
>
> When multiple branches converge to a task, the convergent task is executed only after all the branches are executed successfully.

- **Wait**
  A wait control is a timer. You can introduce a timed delay before the next step of the process is executed. The **Duration** could be set to any integer from 1 to 999, and it can be set in the following increments (see Figure 18.75):
  - **Minutes**
  - **Hours**
  - **Days**
  - **Months**

Figure 18.75 Wait Control

We’ve gone through the design artifacts and controls that you have in SAP Build Process Automation. With this, we can move on to how to build and deploy such a project.

18.4 Building and Deploying the Project

Now that you've gone through the various artifacts and got a glimpse of how you can design a process, let's see how you can deploy an SAP Build Process Automation project in SAP BTP and run it.

Users from a developer background will be familiar with the term *build*. This is a technical term used for the process of assembling and packaging your project artifacts and creating a deployment descriptor, a configuration file, and so on, and then making them ready to be deployed to the runtime.

In SAP Build Process Automation, because it's a low-code/no-code product, you need not worry about these lengthy and cumbersome technical challenges. It's quite simple. Only two clicks of a button are needed: first click on the **Release** button and second click on the **Deploy** button. In the following sections, after we discuss those two actions, we'll cover how to run the project.

18.4.1 Release

Let's see what the **Release** button does. Once you've created your artifacts for the project, saved it, and made sure there are no errors under the **Design Console** tab, you click the **Release** button on the top-right corner of the

process builder, as shown in Figure 18.76. You can also click **Release** from the lobby.

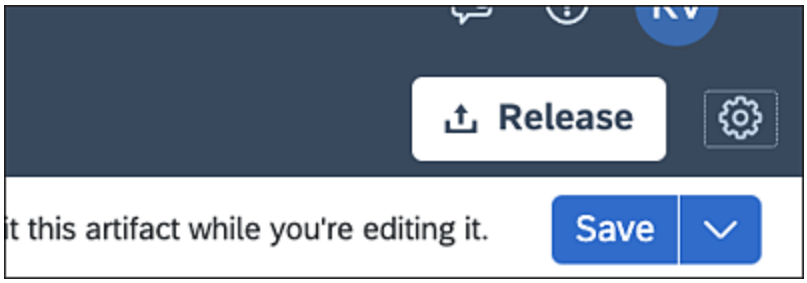


Figure 18.76 Releasing a Project

This action creates a version of your project artifact and makes it ready for deployment. It changes the state of the current editable project to **Released** status. Once you click, you can assign a version number based on the kind of change you're making.

The first time you release a project, the version is defaulted to 1.0.0 that you can't change. The next time and onward, you can choose if it's a patch, minor change, or major change based on which the version numbers will be automatically incremented (see Figure 18.77). This is how the lifecycle of your build process is maintained.

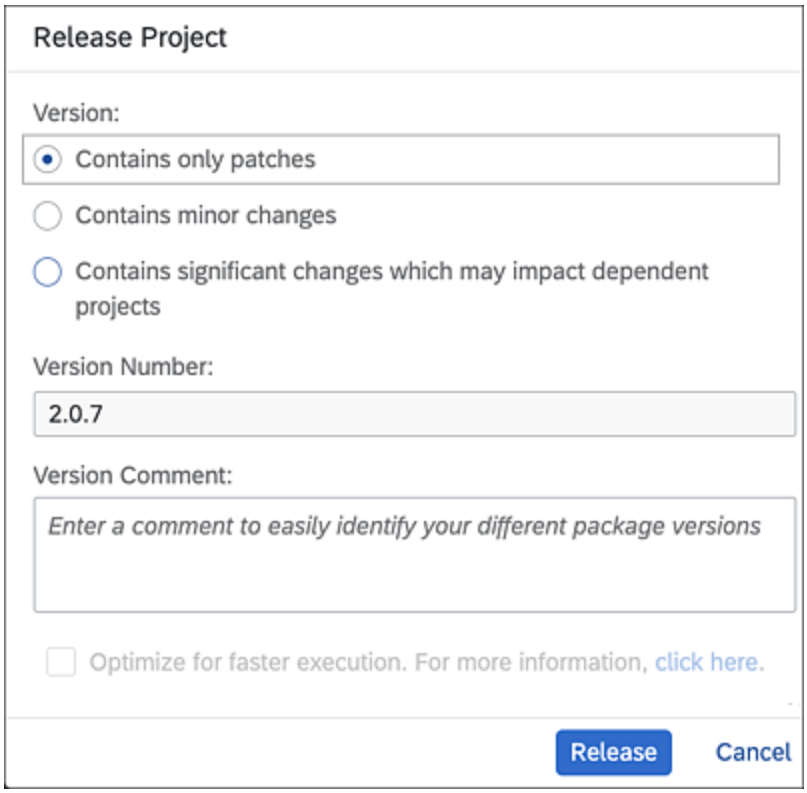


Figure 18.77 Versions When Releasing a Project

18.4.2 Deploy

Once you release the project, you can see now the same button reads as **Deploy** (see Figure 18.78). Clicking on this will deploy the entirety of the project and make it ready for testing and later for productive usage.

Figure 18.78 Deploy Button

On the top center of the process builder window, you'll see the status of the project, and you can toggle between the

versions. One version will always be **Editable**, and you can use the dropdown to select any **Deployed** or **Released** version (see Figure 18.79).

You can choose to **Undeploy** (the **Deploy** button will become **UnDeploy** when a deployed version is selected) a deployed version where it will make the status of the project **Released** from which you can deploy it later. Note that you can delete a released version, but you can't delete a deployed version.

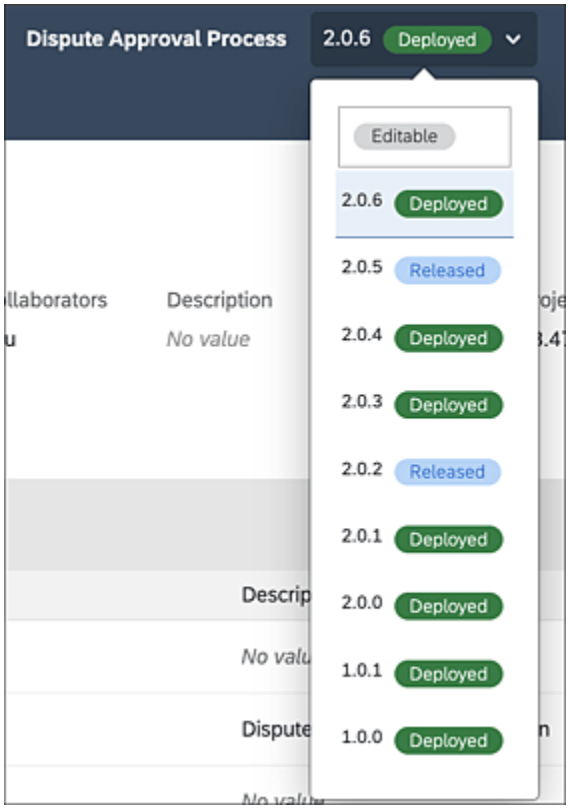


Figure 18.79 Versions of the Project

18.4.3 Run

Finally, once you release and deploy the project, you can test the process using a Postman-like REST client. This is

applicable if the process trigger is an API trigger.

You can also test the process from the Process and Workflow Definition app under the **Monitor** section. Here, you need to supply the context data payload in JSON format. You can see the required context data structure under the **Triggers** section and choose the **View** option of the deployed trigger.

Clicking on **Start New Instance** (see Figure 18.80) lets you trigger a new instance of the selected process. This is something you can use for testing your workflow.

To start the process, you need to pass a set of input data that you defined while designing this process as a JSON payload. You'll be prompted to enter this JSON data when you click the **Start New Instance** button. Once entered in Figure 18.81, click **Start New Instance** and then **Close** to trigger the workflow instance. You can see the running instances from the **Show Instances** button.

Form Trigger

A form-based trigger can be tested using the **Form Link** generated under the trigger properties of the deployed project.

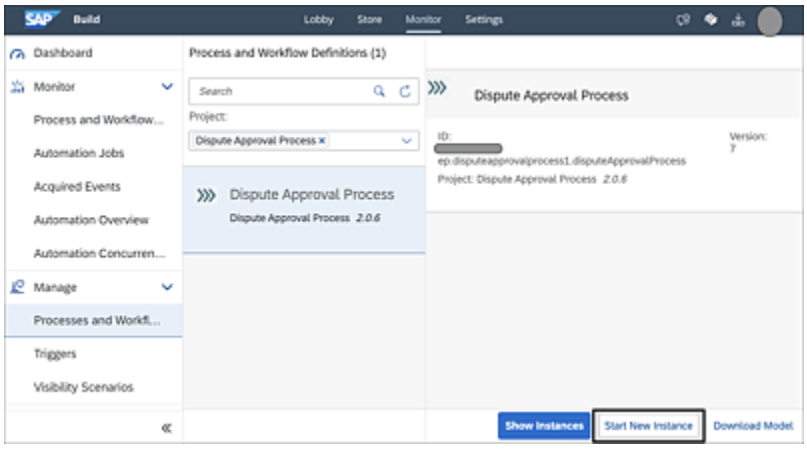


Figure 18.80 Testing or Starting a New Instance of the Deployed Process

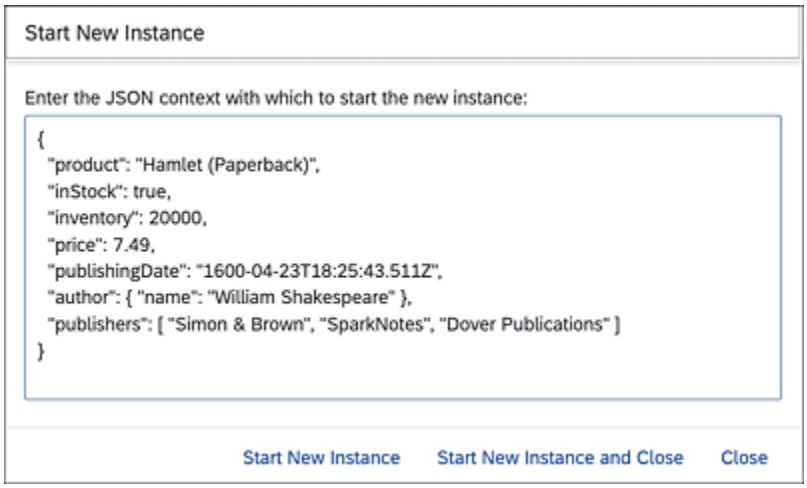


Figure 18.81 Start a New Instance after Entering a JSON Payload

18.5 Destination Configuration with Authentication

For making an API or OData call, you need to have an authenticated and authorized endpoint of the host system where this API/OData service is available. For configuring such endpoints, you use destinations in SAP BTP. This same destination concept is used in SAP Business Process Automation as well for making action calls to different API endpoints. You configure a destination for each system and define the authentication and authorization mechanism for the same that can easily be consumed by the action task in the process. Let’s see how a destination is configured and how the authentication mechanism setup.

18.5.1 Destination Setup

You need to configure destinations for your action projects to communicate with the respective backend data sources. Whether it’s an SAP S/4HANA system or any SAP or non-SAP system, the destination is mandatory for the API call.

The process of creating a destination is the same as you do for the other SAP BTP services. This is done for each subaccount in your SAP BTP cockpit. But to use this destination in your SAP Build Process Automation service, you need to add a couple of additional properties, as follows:

```
sap.applicationdevelopment.actions.enabled = true
sap.processautomation.enabled = true
```

For creating a destination, you can either navigate to the SAP BTP cockpit and choose **Subaccount** • **Destinations** or from the SAP Build Process Automation choose **Lobby** • **Settings** • **Destinations**. Once here, click on **Open** in the SAP BTP cockpit.

You saw how to configure a destination earlier in Section 18.2.2. As a sample, we've created an on-premise SAP S/4HANA destination with basic authentication of the required parameters for usage in SAP Build Process Automation, as shown in Figure 18.82.

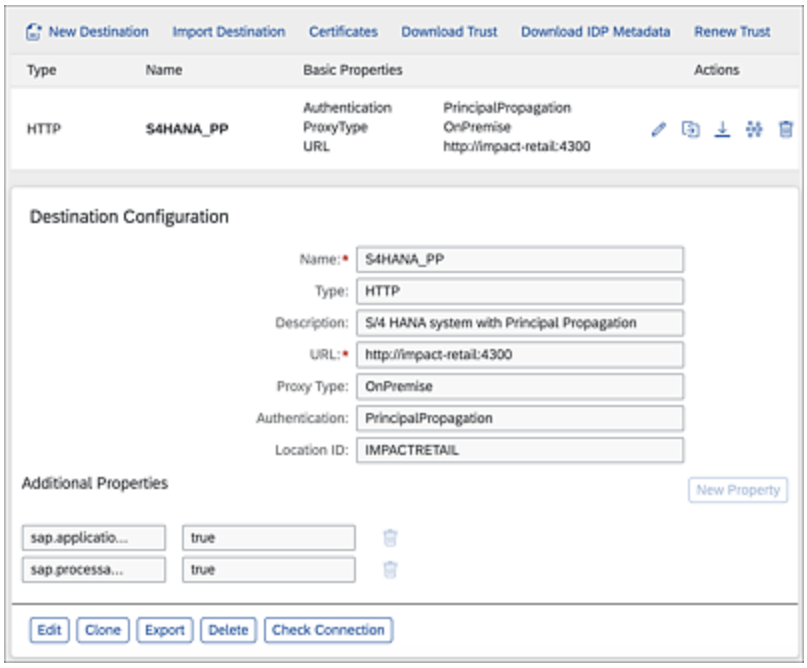


Figure 18.82 Additional Parameters for the Destination

Once you create the destination, you can check the connection with a ping, which should give a **Connection Successful** message. Your destination is now ready to use.

Once this is configured, navigate to the SAP Build Process Automation lobby, and click **Settings**. Navigate to **Destinations**, and click on the **New Destination** button.

You can see the newly created destination for **S4HANA_PP** is available to be added. Select the **S4HANA_PP** destination, and click on **Add**, as shown in Figure 18.83.

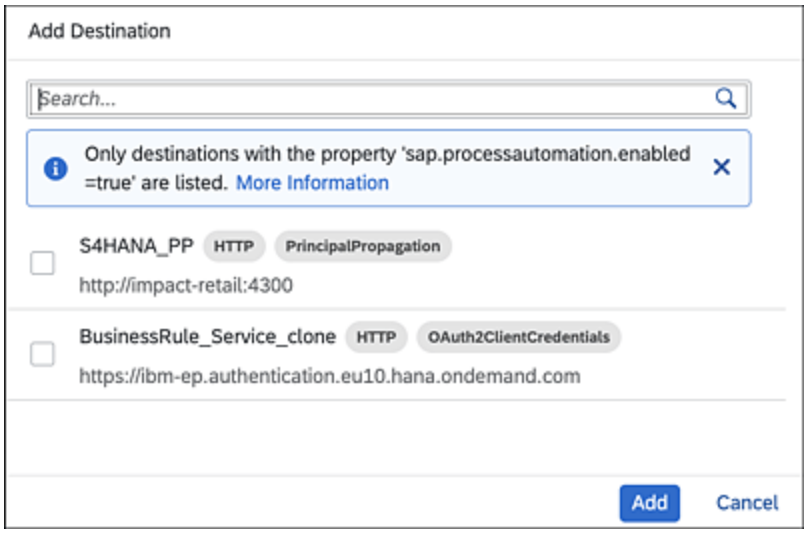


Figure 18.83 Adding the New Destination

Now the destination is available for use in your action project. Once you design a process that has an actions project, while deploying, you need to map this destination to an environment variable that you created for the action task.

When you add an action task in your process, you'll now need to define a **Destination variable** mandatorily under the **General Properties** of **Action**. Figure 18.84 shows the **+ Create Destination Variable** option in the dropdown, which you should click on if you haven't defined a destination variable already. This will be an environment variable that gets created for your project (see Figure 18.85). You need to map this environment variable to the destination you just created in the next step.

This destination can be directly created by clicking the + **Create Destination Variable** option in Figure 18.84 or from clicking the **Settings** icon in the process builder.

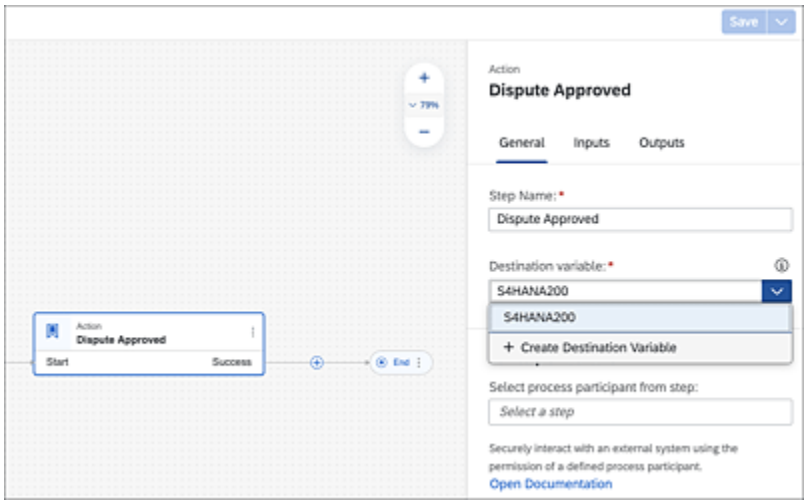


Figure 18.84 Adding a Destination Variable for Your Action Project

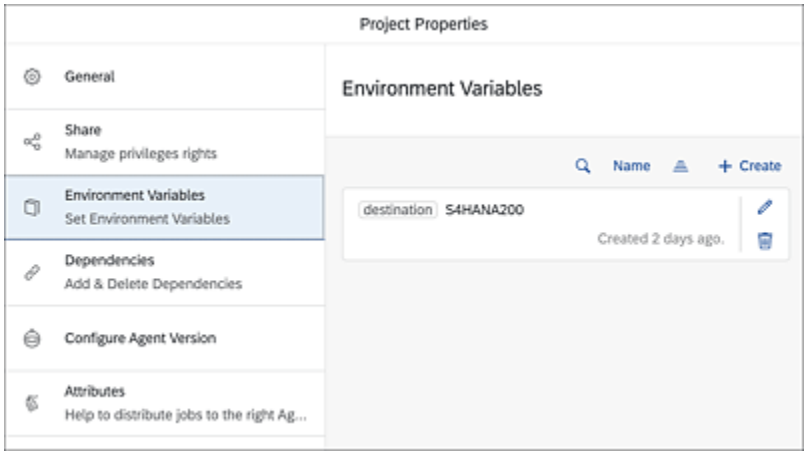


Figure 18.85 Adding Destination Environment Variable under Project Properties

Once the destination variable is configured, this variable needs to be mapped to the actual destination you created in the previous step while deploying the project. When you click the **Deploy** button after releasing your build, you'll be presented with the screen shown in Figure 18.86. The first step will summarize the artifacts that you're deploying, and,

in the second step, you'll need to map the environment variables you defined to the actual destination you created.

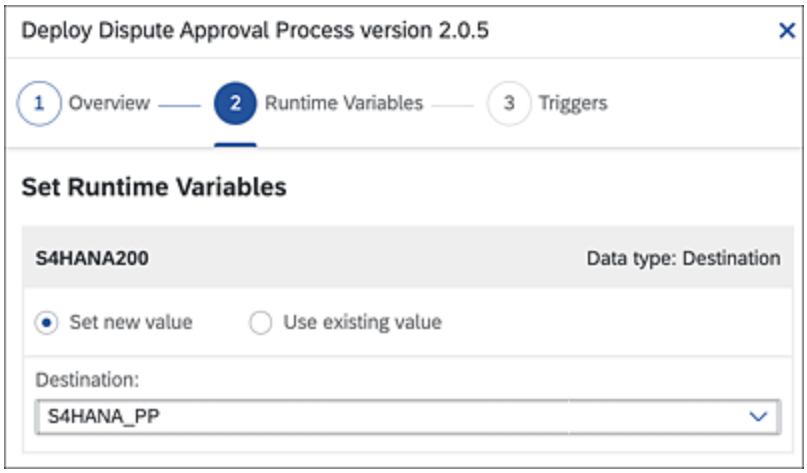


Figure 18.86 Mapping the Environment Variable to the Destination while Deploying

18.5.2 Authentication

An API destination needs to have authentication, and SAP BTP supports most of the standard authentication mechanisms, such as basic authentication, OAuth 2.0, Security Assertion Markup Language (SAML), and so on.

SAP Build Process Automation currently supports only the following types of authentication for a destination that can be used in an action task:

- Basic authentication using the user ID and password
- OAuth2ClientCredentials using the client ID and client secret
- No authentication

18.6 Transport Management

Once the design, development, and testing are completed, now comes the task of moving the artifacts across the landscape. Most companies have a typical three-system or even a four-system landscape consisting of a development, quality, and production system, sometimes with an acceptance system. Most often, the SAP BTP subaccount setup will be matched to the SAP S/4HANA landscape. How do you move your built and tested project artifacts across these subaccounts and finally to production?

This is where the SAP Cloud Transport Management service comes into the picture. SAP Cloud Transport Management is used to transport SAP BTP–based development, mostly any artifact that can be packaged as an *mtar* file. Along with SAP Cloud Transport Management, it's mandatory to have the SAP Content Agent service also subscribed and configured to transport SAP Build Process Automation artifacts. This isn't required for the SAP Workflow Management service artifacts. The rest of the process remains the same for both.

Configuration of the SAP Cloud Transport Management nodes isn't explained here because it's assumed that the SAP Cloud Transport Management service is configured for your landscape along with the SAP Content Agent service.

Once you release a project, go to the lobby, and under the **Versions** dropdown (see Figure 18.87), click the three dots against a deployed or released project version that you

want to transport. Here, you'll see an option to **Promote** the project. The **Promote** feature allows the movement of artifacts as transport to other environments through the SAP Cloud Transport Management and the SAP Content Agent services.

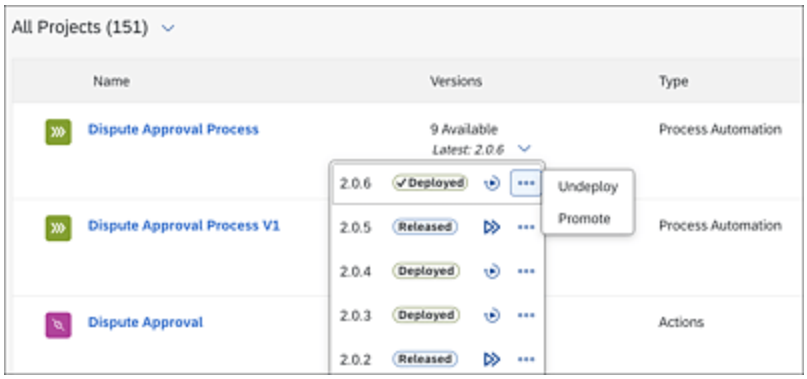


Figure 18.87 Transport Using the Promote Feature

If SAP Cloud Transport Management and the SAP Content Agent service are configured properly, a transport request will be created, and the *mtar* file containing all the artifacts in the project will be added to the transport request and will be available for import in the starting node of SAP Cloud Transport Management. From here, the transport request can be imported to the next system and subsequently to production.

Figure 18.88 shows a starting node in the landscape where a transport request containing an *mtar* archive is created and has been imported. You can see the status of the transport request as **Succeeded**.

Once the **Dev** node import is successfully completed, the transport request will show in the next node in the landscape as **Initial** and waiting to be imported. Figure 18.89 shows that the same transport request from

the development node is now waiting in the production node to be imported. You can select the transport request and click the **Import All** button to import the transport request to the production subaccount. This import can also be controlled by SAP Solution Manager using a Change Request Management process.

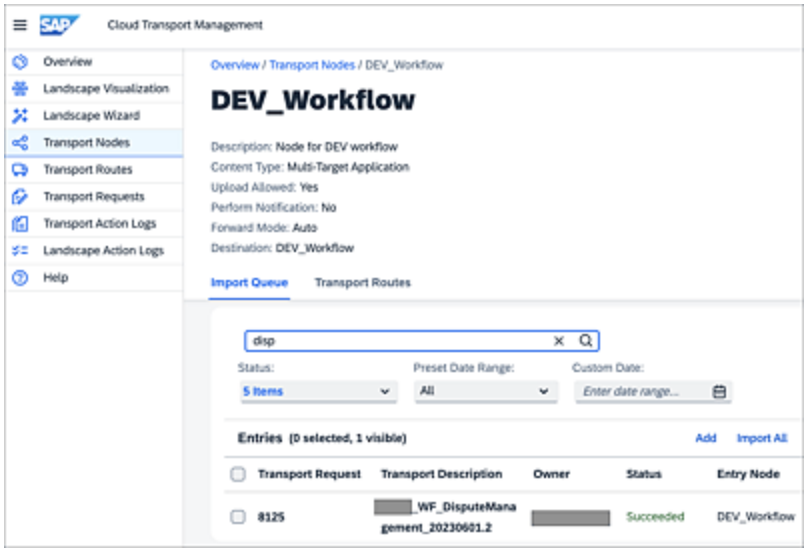


Figure 18.88 Transport Created in the Dev Node of SAP Cloud Transport Management System

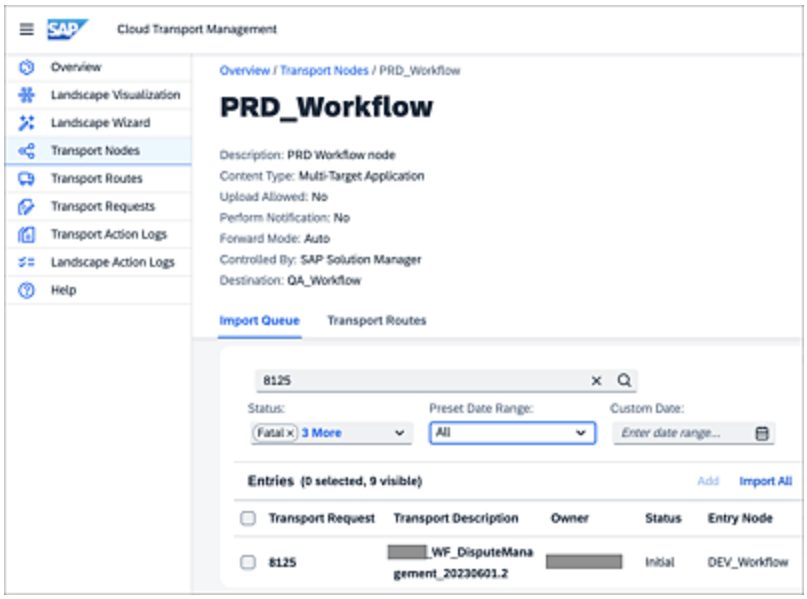


Figure 18.89 Transport Waiting in the Production Node to Be Imported

18.7 Using APIs for SAP Build Process Automation

SAP has provided both REST-based and OData-based APIs that can be readily used to achieve various tasks within SAP Build Process Automation. We'll be looking at mainly the APIs available for the process-related topics that we discussed in previous sections, as follows:

- **Workflow**
  These sets of REST APIs can be used to manage the process flows and related tasks.
- **Decision**
  These REST APIs allow you to write and retrieve business rules decision-related data.
- **Inbox**
  These OData APIs let you retrieve My Inbox items as well as perform operations and substitutions.

> **Note**
>
> More information on the available APIs and references for SAP Build Process Automation are available at *https://api.sap.com/package/SAPProcessAutomation/all*.

We won't be discussing all available APIs here. We'll highlight the key APIs that will come in handy when designing a workflow process using SAP Build Process Automation.

18.7.1 Application Programming Interfaces for Workflow

Table 18.1 lists some of the key APIs that can be used to post and read workflow instance–related information (e.g., user task information and workflow instance details) and to trigger intermediate message events.

API	Description
User Task	
GET /v1/task-instances	Can retrieve all the user task instances from all the workflow instances unless the status is specified. Various parameters can be added to this filter based on workflow definition, workflow instance, and so on.
PATCH /v1/task-instances/{taskInstanceId}	Can be used to update the properties, context, and status of a user task. This is extremely useful if you want to implement some sort of mass approval scenarios that My Inbox doesn't support.
Workflow Instances	

API	Description
POST `/v1/workflow-instances`	Arguably the most important API, which can be used to start a new workflow instance. This is very useful when triggering a workflow from an external UI form or service using an API trigger.
PATCH `/v1/workflow-instances/{workflowInstanceId}`	Using this API, you can modify the status of a running instance.
PUT `/v1/workflow-instances/{workflowInstanceId}/context`	Using this API, you can overwrite the entirety of a workflow instance’s context data.
PATCH `/v1/workflow-instances/{workflowInstanceId}/context`	Using this API, you can overwrite any part of a workflow instance’s context data.
Messages	

API	Description
POST `/v1/messages`	This API is used to trigger an intermediate message in SAP Workflow Management service-based workflows designed in SAP Business Application Studio. As of now, SAP Build Process Automation workflows don’t support intermediate messages or events, so this is particularly applicable only for SAP Workflow Management workflows.

Table 18.1 Key APIs Available for SAP Build Process Automation Processes and Workflows

Warning

Take extreme caution while using the context update APIs. Make sure the workflow isn’t being executed while updating the context as this can cause unpredictable outcomes or even corrupt the workflow. Another option is to first suspend the instance and then update the context. You can resume it after.

18.7.2 Application Programming Interfaces for Decisions

Table 18.2 describes the key API for decision tasks where you need to execute and receive the decision rule data.

API	Description
POST `/v2/rule-services`	This API is used to trigger a decision rule service with the necessary vocabulary input. Response will provide the decision output data according to the rule.

Table 18.2 API for Decision Tasks

18.7.3 Application Programming Interfaces for My Inbox

Table 18.3 lists the key APIs used for My Inbox–related manipulation.

API	Description
Operations	
POST `/Forward?SAP__Origin='{SAP__Origin}'&InstanceID='{InstanceID}'&ForwardTo='{ForwardTo}'`	This API can be used to forward an inbox task to another user from a remote trigger.
Substitutions	

API	Description
POST `/SubstitutionRuleCollection`	This API can be used to create a substitution rule for a given user. This is useful when administrators need to remotely reassign critical tasks on the absence of an explicit substitution set by a user.

Table 18.3 Key APIs for My Inbox

18.8 Design a Process/Workflow for the Use Case

Now that we've gone through details of how a workflow can be designed using SAP Business Application Studio and process builder, deployed it to SAP BTP, and run it, let's walk through a typical use case to see how a workflow can be designed using SAP Business Application Studio and process builder.

18.8.1 Use Case and Solution

Most manufacturing companies have a market-to-cash process. A few also have an additional process to manage disputes from their customers arising out of sales completed. SAP S/4HANA doesn't have an out-of-the-box workflow that matches the requirements if the company wants to implement an approval process at multiple levels.

Here's the scenario: Within the organization, a dispute administrator is the one who will be responsible for managing and solving the disputes and, if required, involving other teams (billing, logistics, finance, commerce, etc.) based on the nature of the dispute. Within this context, SAP Business Workflow is used to support the routing of disputes to the different teams involved in the resolution, record the approval, and generate commercial credit notes.

As this would be a custom workflow, the question is whether you're going clean core or not. If the answer is yes, then the

obvious choice of designing the workflow will be in SAP BTP (either SAP Build Process Automation or SAP Business Application Studio).

Coming back to the process itself, in the SAP S/4HANA system, when disputes are created and moved to **In Progress**, the dispute administrator can route the cases to other teams for resolution or approval depending on the amount of the sales order. When the analysis has been completed, and all details have been populated in the backend system, the dispute administrator can save the dispute, leaving the processor empty and letting the system automatically determine the processor, along with the number of levels of approvals needed and the teams it needs to go to. Once all the approvals are in place, the dispute gets updated, and commercial credit notes are automatically generated. This takes away the responsibility from the dispute administrator to route to the right team, and the process can be automated with a preset business rule.

For this use case, we've decided that the standard workflow won't be suitable and need something custom. The choice is to customize the standard workflow, but we're going clean core. One of the key principles of the clean core strategy is to keep customization and extension of standard processes away from the core SAP S/4HANA system as much as possible.

So, the obvious option to go the SAP BTP way. Design the custom workflow using the SAP Build Process Automation service and trigger it from SAP S/4HANA. Once the required

steps are complete, update the dispute and release the credit note.

Now that we know the intended solution, let’s see how this can be implemented. We’ve decided that to design and run the workflow in SAP BTP, but how does the trigger happen? Another key strategy used in modern solutions is event-driven architecture. SAP S/4HANA systems support events for all business processes. All the important steps in a business process can generate events, and so does our dispute creation step. When the dispute administrator creates and saves a dispute, an event gets generated. This event can be used to trigger our workflow in SAP BTP using standard APIs. The dispute details are sent across as the data payload within the event.

Once the workflow gets triggered, it’s up to the workflow to decide the level of approvals required and which user or group it should go to. This part is achieved using business rules, or decisions, as it’s called in SAP Build Process Automation. We’ll see how simple it is to achieve this.

You saw in the previous section criteria based on which you choose to use SAP Build Process Automation process builder against SAP Business Application Studio to design the workflow. Considering both as a possibility, let’s dive in and see what it looks like and how easy or complex it is to build the workflow using both these options.

18.8.2 Design Using Process Builder

We’ve seen the various artifacts that can be used to build a process in process builder in previous sections. We’ll put a

few of them in use to build this process flow. Start with creating a new SAP Build Automation Business project as described previously in Section 18.3.1. Name it “Dispute Approval Process”, and click **Create**. Open the newly created process in the process builder.

Add the set of context variables that you’ll be using in the flow going forward. For this, click the **Inputs/Outputs** tab, and then click on the **Configure** button for process inputs. Add the necessary data fields, as shown in Figure 18.90 (provided in Table 18.4), and click **Apply**.

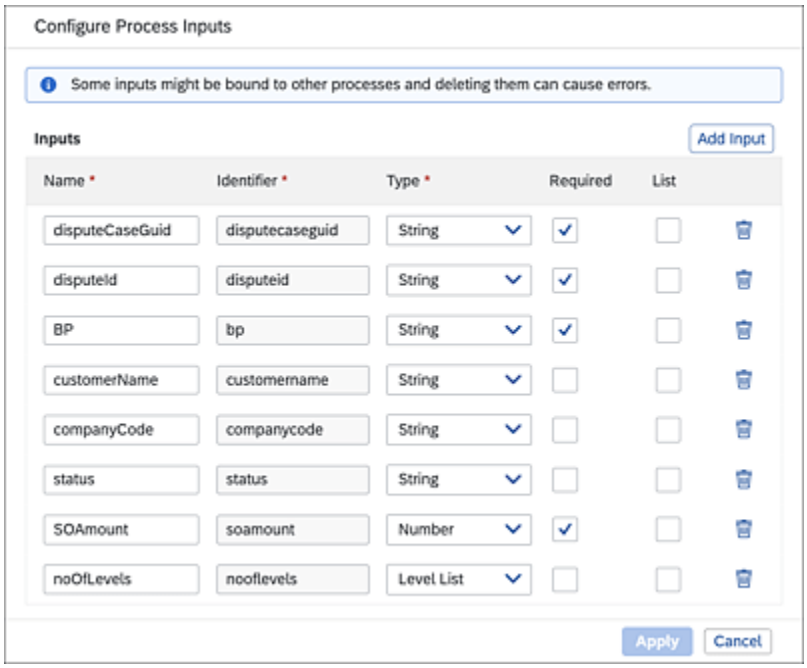


Figure 18.90 Adding Context Fields

Name	Identifier	Type
disputeCaseGuid	disputecaseguid	String
disputeId	disputeid	String
BP	bp	String

Name	Identifier	Type
customerName	customername	String
companyCode	companycode	String
status	status	String
SOAmount	soamount	Number
noOfLevels	nooflevels	Level List

Table 18.4 Process Input Parameters

Note that **Level List** is a custom data type that you can create in SAP Build Process Automation as an artifact. It's nothing but a simple structure. You can create such custom data types to use in the automation process. Figure 18.91 shows the structure of the custom data type.

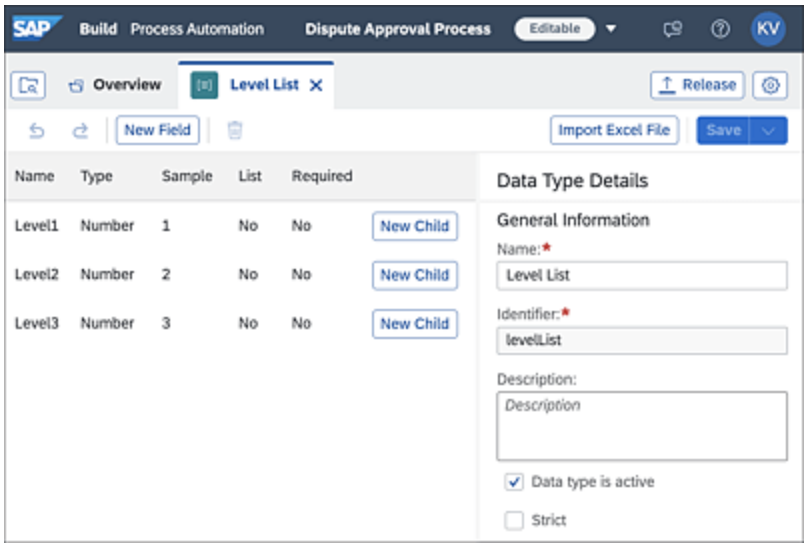


Figure 18.91 Creating a Custom Data Type

Additional Custom Data Fields

If you need to add a custom data type such as a list, you can use **Create Data Type** from the project **Overview**. This is especially useful to create custom arrays that can be used in the workflow. Here we're creating a simple list called *Level List* that will hold the approval level indicator that needs to be fed to the business rules.

You start the process with a trigger, and as this is an event-based trigger, you need an API trigger. An API trigger is added to the **Start Process** step. Click the **+** icon on the **Trigger** step, and add a trigger named "TriggerDisputeApprovalProcess", as shown in Figure 18.92.

You know it's going to be an event-based trigger, so whenever the dispute administrator creates a dispute in the SAP S/4HANA system, an event gets generated. This event can be published and subscribed to through an event broker such as SAP Event Mesh. You can subscribe to this event mesh using a webhook, which can be the standard API used to trigger our process.

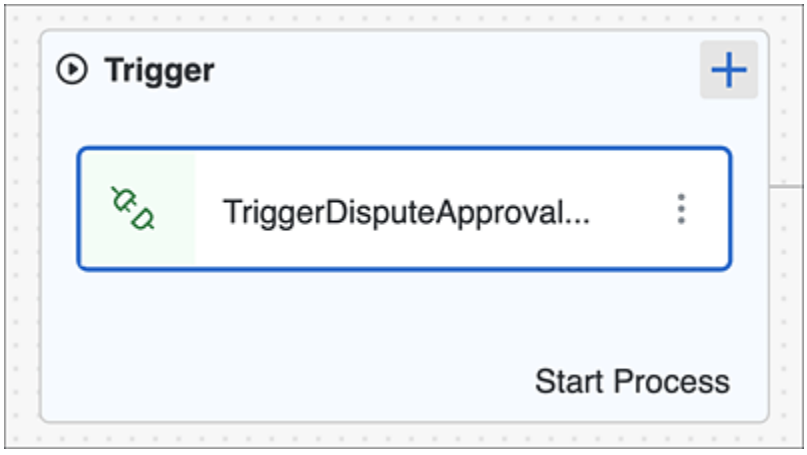


Figure 18.92 Configuring the Trigger

The standard API looks something like this:

https://<spa-service-id>.cfapps.<region>.hana.ondemand.com/workflow/rest/v1/workflow-instances

This can be obtained once you release and deploy your workflow. Navigate from the **Lobby** to **Monitor** • **Manage** • **Triggers**, and under the **Actions** menu of your deployed trigger, click **View**. In Figure 18.93, you'll see the Trigger's URL endpoint. The payload/input is nothing but the context data that you'll receive from your event payload through the API.

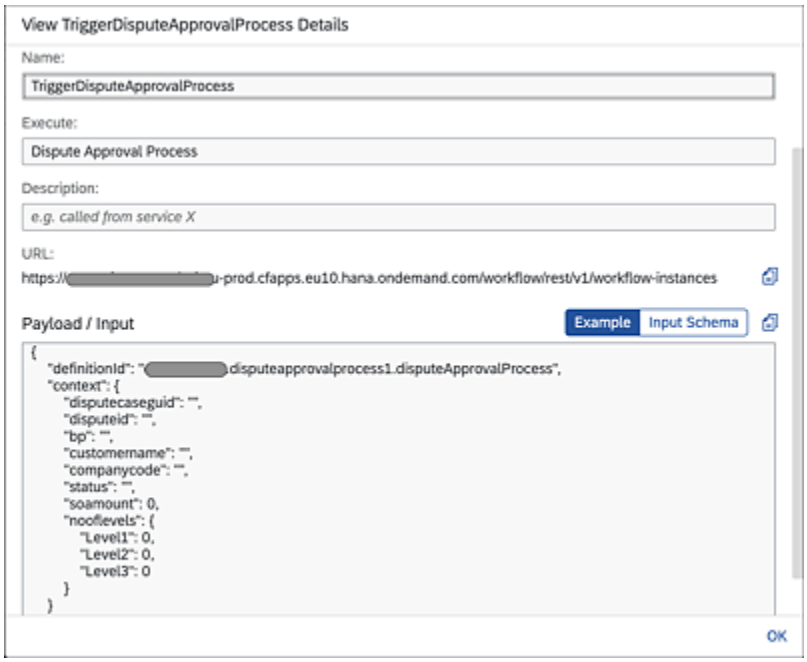


Figure 18.93 View Trigger Details

Tip

Note the `definitionId` in the trigger details. This needs to be used to trigger your process from the API. To get this information, from the **Lobby**, navigate to **Monitor** •

Manage • **Process and Workflow Definitions**, and then select your deployed process.

Figure 18.94 shows a preview of the initial process steps that you need to design in the process builder screen.

Figure 18.94 Initial Steps of the Process

Once the process is triggered, the first step in the process is the processor determination. Per the requirement, it's the responsibility of the process to determine the processor and level of approvals required for the given dispute data. This is achieved using a decision artifact, as follows:

1. You can create a new decision artifact from either the **Overview** page or from the process itself. Click the **+** icon on the line between the trigger step and the end step, and add a new decision.
2. Name the decision "Dispute Processor Determination".
3. Click on the *three dots* icon on the step, and click **Open Editor.**
4. Here you need to configure the **Inputs/Outputs** parameters that will be used while executing this step.
5. Before you add the parameters, for convenience, create a custom **Data Type** called **DisputeProcessor**, which will be a simple list (see Figure 18.95).

6. Now, it lets you continue editing the **Decision Task**.
7. Click in **Add Input Parameters**, and enter the **Name** as “Processor Input”. Add the **Description**, and select the type as **DisputeProcessor**.
8. In the same way, set the output parameters. Click in **Add Output Parameters**, and enter the **Name** as “Processor Output”. Add the **Description**, and select the type as **DisputeProcessor** (see Figure 18.96).

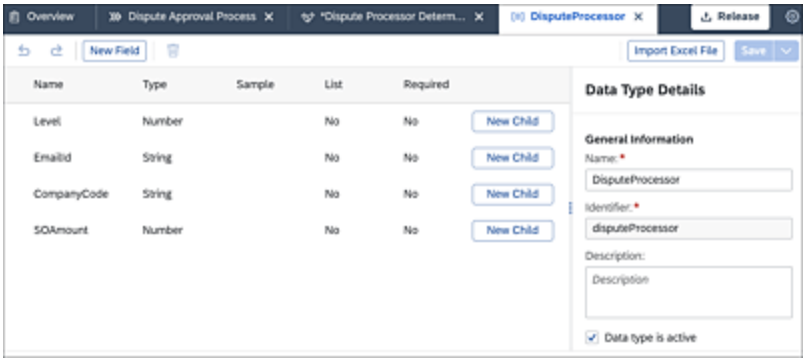


Figure 18.95 Creating a Custom Data Type

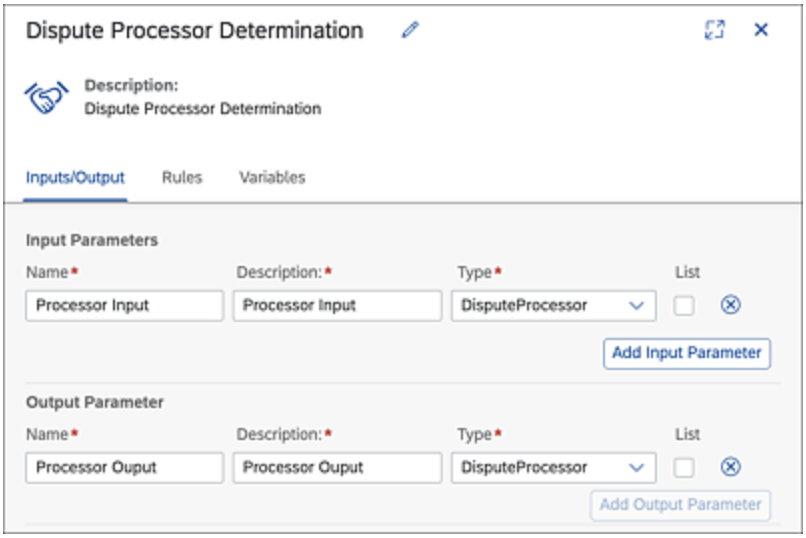


Figure 18.96 Setting the Decision Parameters

9. Click on the **Rules** tab, and click **Add Rule**.

10. On the **Rule Details** step, enter the **Rule Name** and **Rule Description**. Select **Hit Policy** as **First Match (Default)**, as shown in Figure 18.97.

11. Click **Next Step**, and add the rule condition parameters. From the **Vocabulary** section, click **CompanyCode**, **SO Amount**, and **Level**. These fields get added to the **Condition Attributes**. You can either add the **Operators** here or when you add these fields to the decision table. Click on **Next Step**, as shown in Figure 18.98.

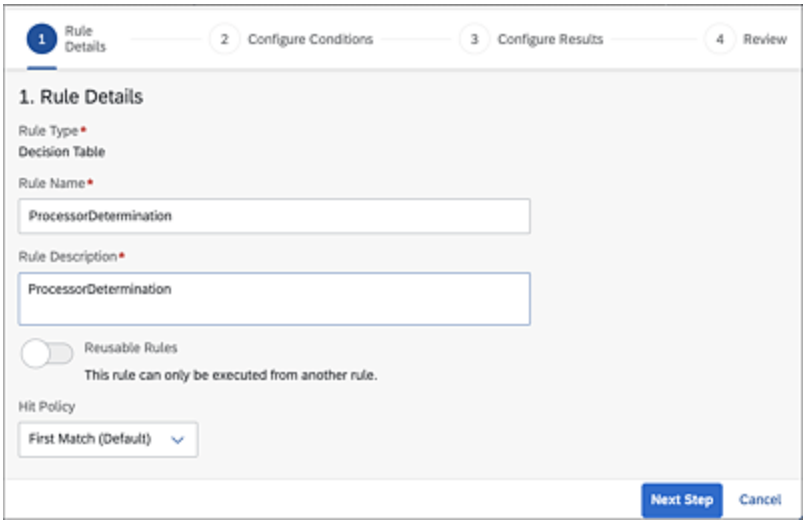


Figure 18.97 Rule Details

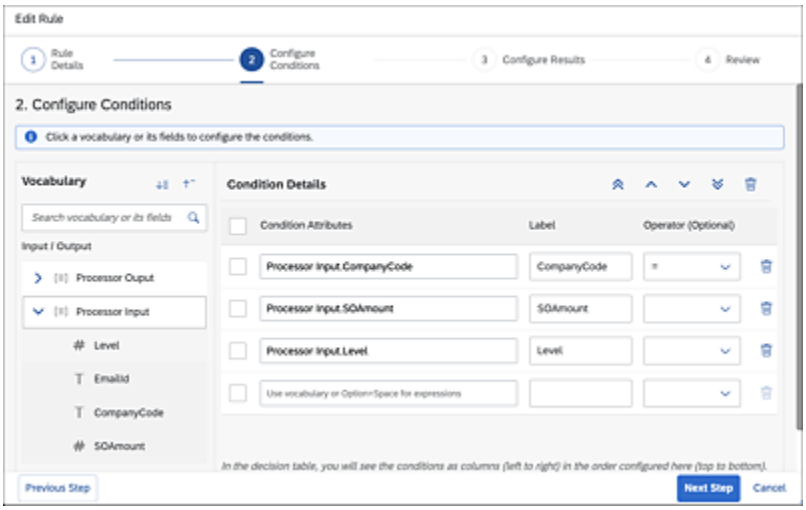


Figure 18.98 Configuring the Conditions

12. Set up the results by selecting the **EmailId** field from **Processor Output Vocabulary**. Click **Next Step**, as shown in Figure 18.99.

13. In the final step, you can review the configuration you've done. Click **Finish**, as shown in Figure 18.100.

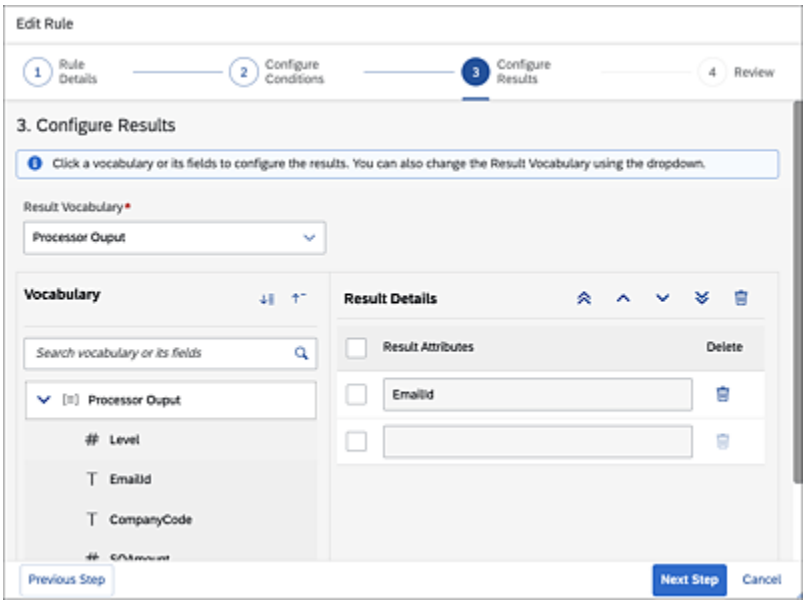


Figure 18.99 Configuring the Output Results

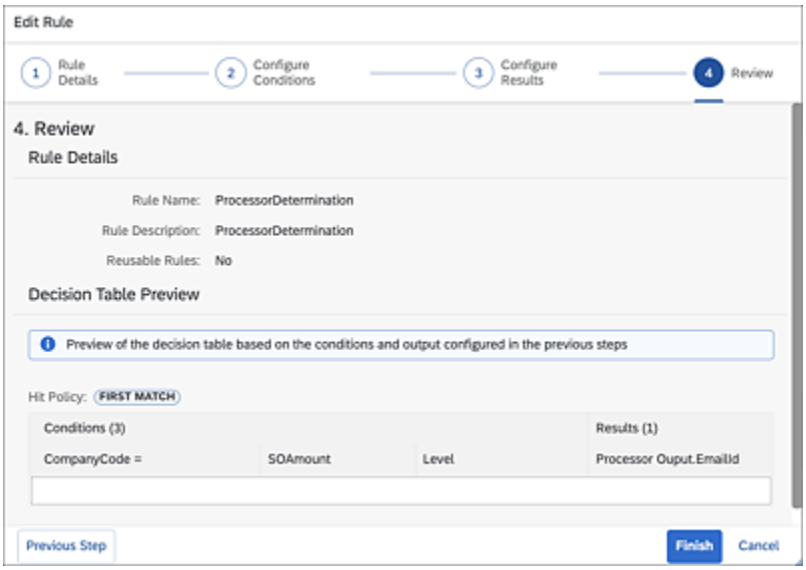


Figure 18.100 Reviewing the Rule

14. Now, in the newly created rule (**Decision Table**), add the values as shown in Figure 18.101. Make sure you change the **EmailID** field, which should be a valid user in the SAP BTP subaccount in which you're running this process. The values to add are listed in Table 18.5.

'XNL1'	<= 1000	=1	L1Approver@com.com
'XNL1'	IN (1000 .. 10000]	=1	L1Approver@com.com
'XNL1'	IN (1000 .. 10000]	=2	L2Approver@com.com
'XNL1'	> 10000	=1	L1Approver@com.com
'XNL1'	> 10000	=2	L2Approver@com.com
'XNL1'	> 10000	=3	L3Approver@com.com

Table 18.5 Data to Be Maintained in the Decision Table

15. Make sure to maintain a valid email ID for the **EmailID** field.

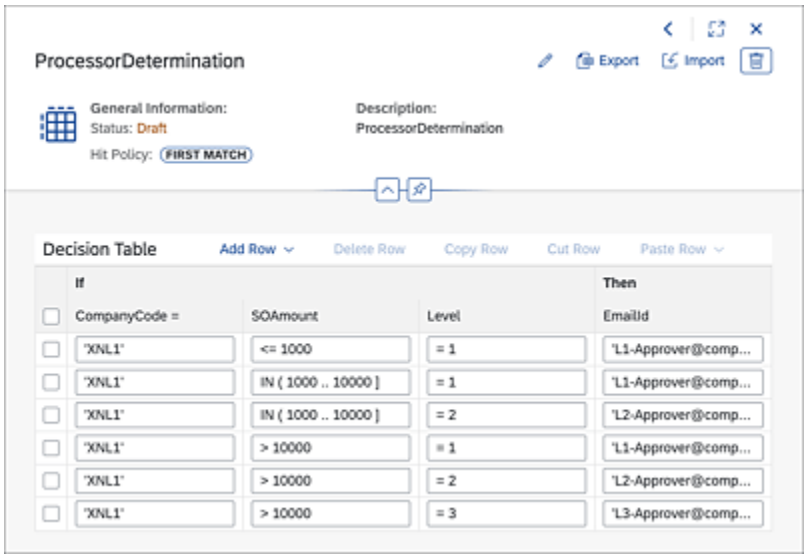


Figure 18.101 Decision Table

Let’s keep the rule simple for the sake of this exercise. Based on the company code, the sales order amount, and the number of levels, the approver is determined, as follows:

1. For a given company code, if the sales order amount is less than 1,000, it should have one level of approval.
2. For a given company code, if the sales order amount is between 1,000 and 10,000, it should have two levels of approval.
3. For a given company code, if the sales order amount is greater than 10,000, it should have three levels of approval.

These inputs are mapped from the event data payload when the dispute is created.

You've created the rule and decision step in the process flow. Now you need to map the data elements. Click on the **Decision** step on the process flow, and click **Inputs** on the right-hand side panel. Set the parameters for the fields as shown in Figure 18.102 from the **Process Inputs** structure.

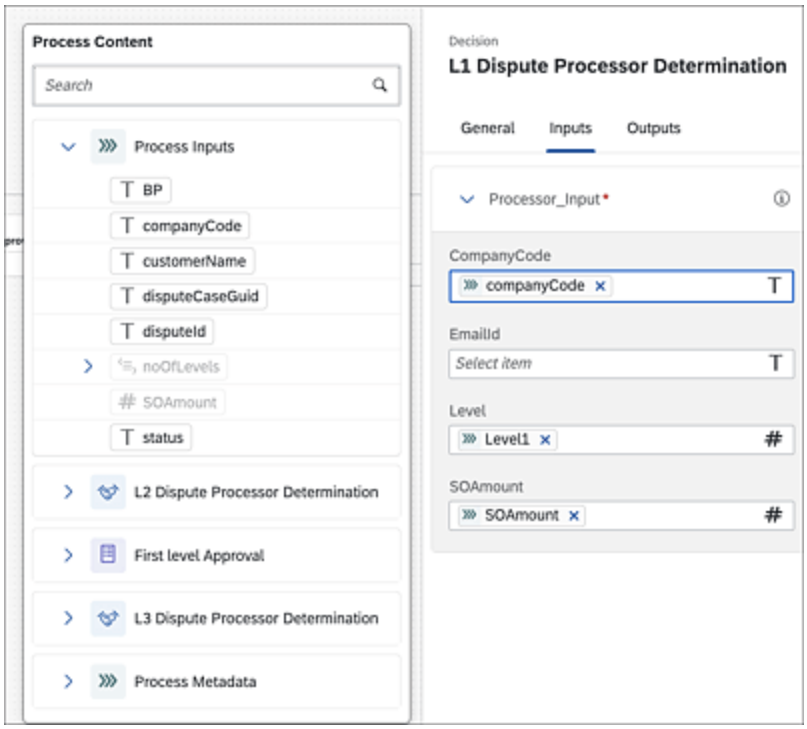


Figure 18.102 Decision Field Mapping

Now, when the decision table is executed, the first-level approver is determined. A My Inbox task is created in the next step using an approval form.

Let's see how an approval form is created. As you saw in Section 18.3.4, an approval form is a simple form with two actions: **Approve** and **Reject**. An approval takes the process to the next step, and a rejection ends the process. Follow these steps:

1. From the **Overview** tab, create a new approval form, and name it "First Level Approval".

2. Create two more approval forms, and name them “Second Level Approval” and “Third Level Approval”.
3. Add the screen elements shown in Figure 18.103 for all three forms. Save the form.

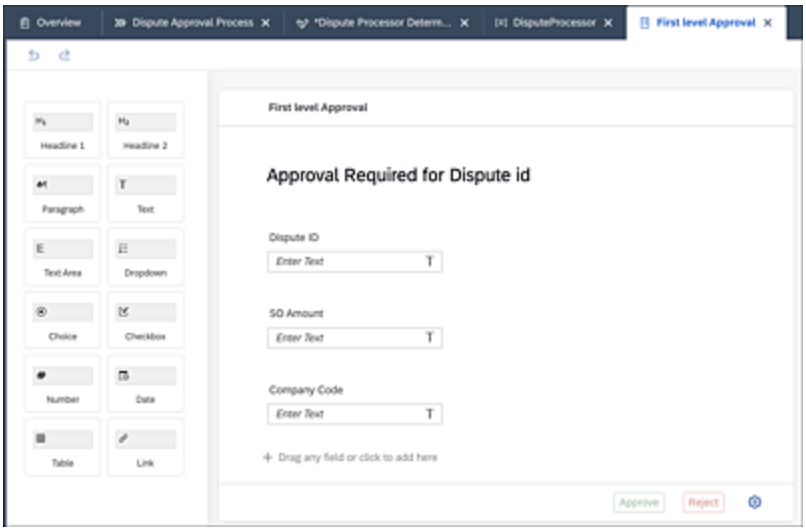


Figure 18.103 Create Approval Forms

The screen should have three text controls labeled **Dispute ID**, **SO Amount**, and **Company Code**.

4. Add the **First Level Approval** form after the decision step using the + icon. Map the data fields as shown in Figure 18.104. The mapping part is simple, just click on each field. The context help will automatically open with the **Process Content** screen where you can just click the relevant field for each of the **Input** fields to map them.

Per our process, once the first-level approver approves it, the process moves to the next step. Here a condition needs to be checked per the business requirement. Based on the sales order amount, the condition is checked if it needs further approvals. If not, the process is ended.

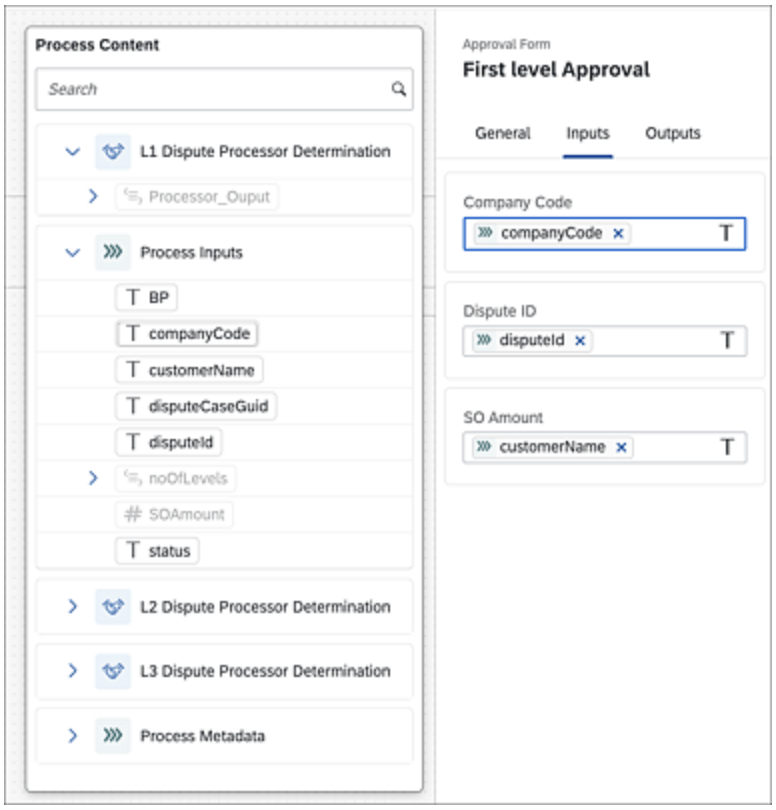


Figure 18.104 Form Field Data Mapping from the Context

Let’s add a condition, as follows:

1. Click the + icon on the line after the **First Level Approval** step’s **Approve** endpoint, and select **Controls** and then **Condition**.
2. Name the condition, and click **Open Condition Editor** on the right panel, as shown in Figure 18.105.
3. Add the conditions as shown in Figure 18.106, and click **Apply**.
4. The values should look like the following:
   - **SOAmount is greater than 1,000**
   - **SOAmout is less than or equal to 10,000**

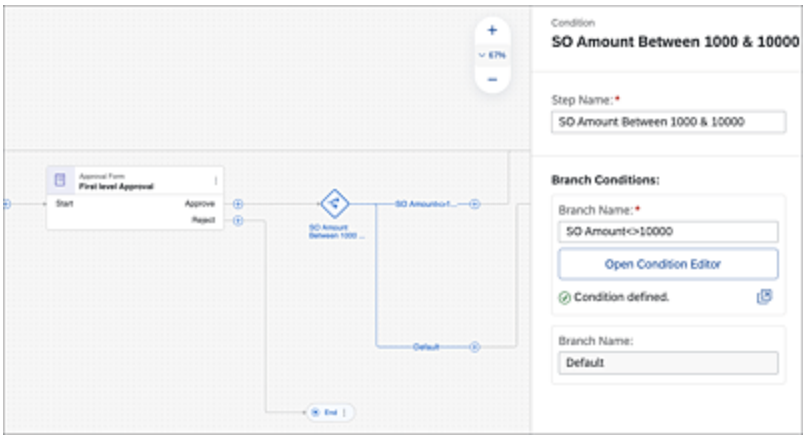


Figure 18.105 Creating a Condition

Figure 18.106 Adding Condition Criteria

5. You'll now see two branches created from the condition step. One is **Default**, and the other is **SO Amount <>10000.** If the first-level approver rejects the case, the dispute is closed. So, let's end this branch of the flow by adding an **End** event.
6. From the **SO Amount <>10000** branch, add the existing **Decision** you created in the previous step but set the parameter for the **Level** as **Level2** from the **Process_Inputs** structure (see Figure 18.107). This step decides the second-level approver.

7. Next, add another **Approval Form** for the second-level approval. Remember, you've already created the **Second Level Approval** form in the previous step. You just need to add that form and map the parameters from the context to the fields defined, as shown in Figure 18.108.

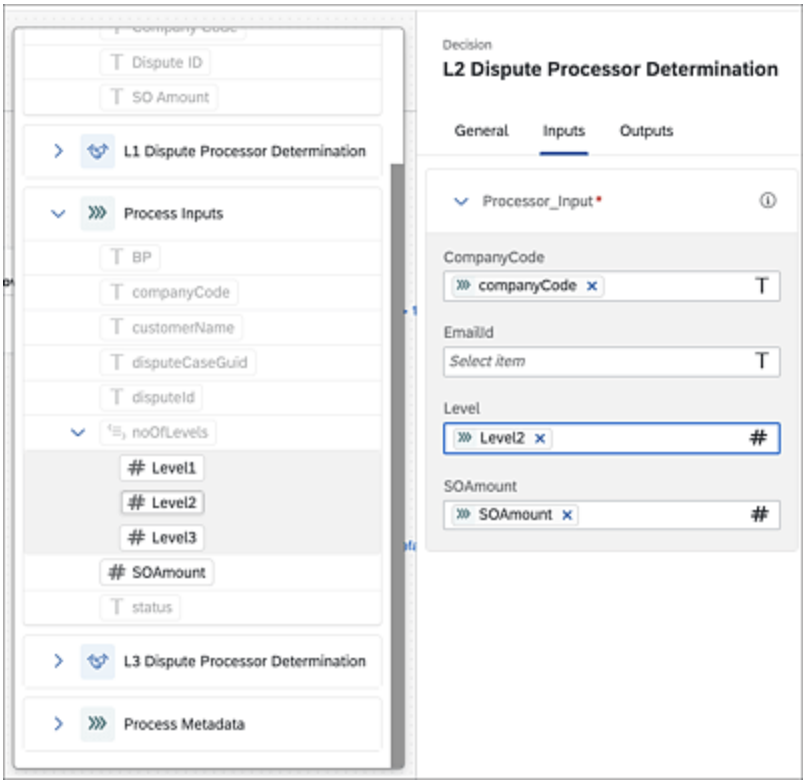


Figure 18.107 Second Decision Step

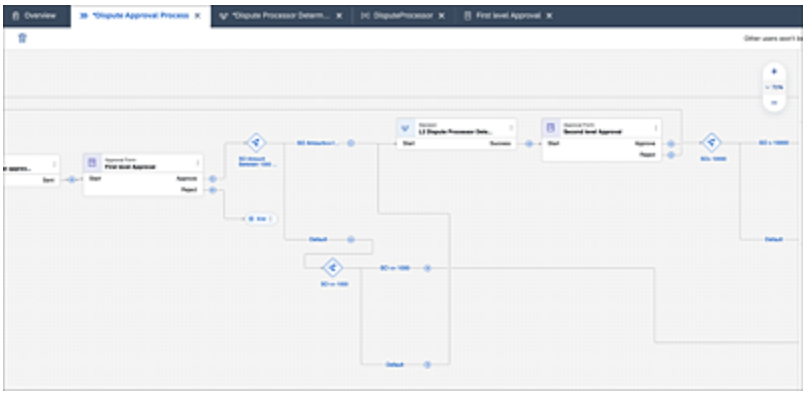


Figure 18.108 First Condition Check and Second-Level Approval Steps

You've now created the sequence flow after the **Condition SO Amount Between 1000 & 10000** that checks if the sales order amount value is between $1,000 and $10,000, which resolves to `true`. You need to configure the default flow now, which resolves to `false`.

Add another **Condition** into the **Default** flow of the first **Condition** and name it "SO<=1000". Here, you're going to check if the sales order amount is less than or equal to $1,000, as shown in Figure 18.109. If it's less, considering the approval has already taken place, end the flow after connecting it to the action task, which you'll configure later in the exercise.

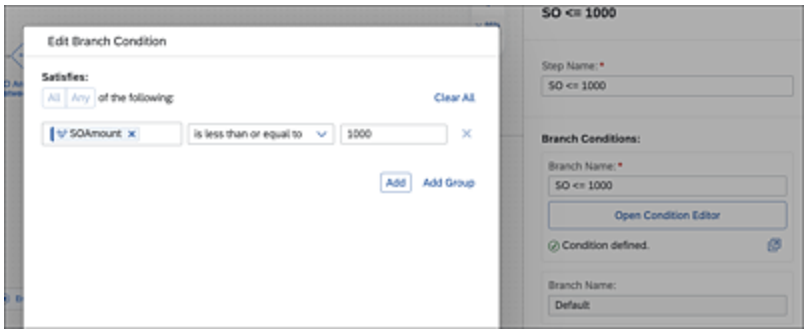


Figure 18.109 Second Condition Configuration

If the sales order amount is greater than $10,000, it should go to the **Second Level Approval** and then to the **Third Level Approval**. So, you connect the **Default** flow to the **L2 Dispute Processor Determination** step you created before. Remember, the second-level approval can be triggered from two conditions resolving to `true`: sales order amount between $1,000 and $10,000, and sales order amount greater than $10,000. After the second-level approver approves, you need to check if it indeed needs to

go to the third level or not. This is what you'll do next with another **Condition**.

Add another **Condition** from the menu via the **+** icon emanating from the **Approve** flow in the **Second Level Approval Form**. Add the condition as shown in Figure 18.110: **SOAmount is greater than 10,000**.

Figure 18.111 shows how the process design looks with the third-level approval and the action task added to align with the end event.

You've seen that if the first-level approver rejects it, the process ends, but if the second- and third-level approver rejects it, the control comes back to the first-level approver, and the process repeats. To achieve this, drag and connect the ends emanating from the **Reject** flow of the **Second Level Approval** and **Third Level Approval** to the **L1 Dispute Processor Determination** task.

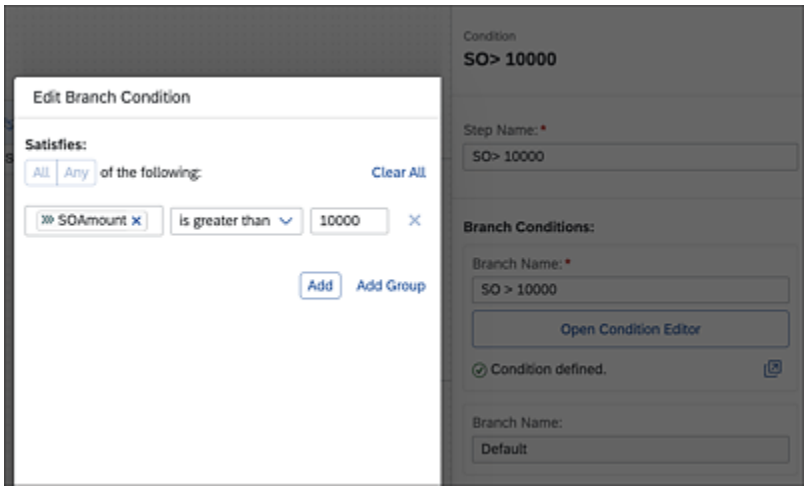


Figure 18.110 Third Condition Check

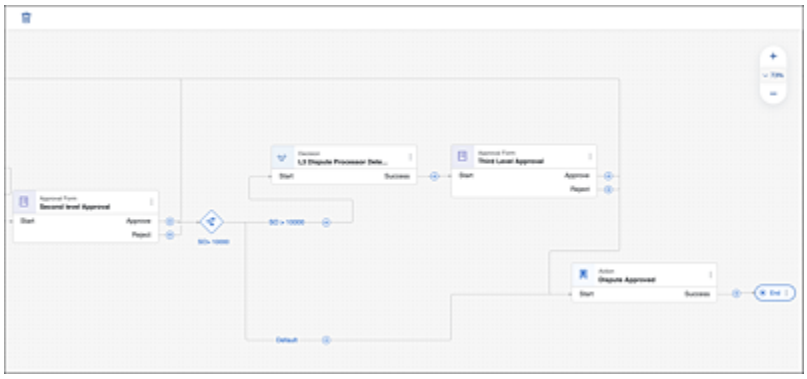


Figure 18.111 Third Condition Check and Action to Update the Dispute

Finally, if all the approvers complete the approval, an **Action** call is made to update the dispute status in the SAP S/4HANA backend and to generate a credit note, and the process ends. This is achieved by creating an action project with the OData details as described in Section 18.3.8. Once this is successfully done, you can add an **Action** task in your flow. Here we're using custom OData.

From the **Approve** endpoint of the **Third Level Approval** task, click the + icon to select an **Action**, and go to **Browse** the action library. Search for the **Action** project you created for this OData call, and click **Add** as shown in Figure 18.112.

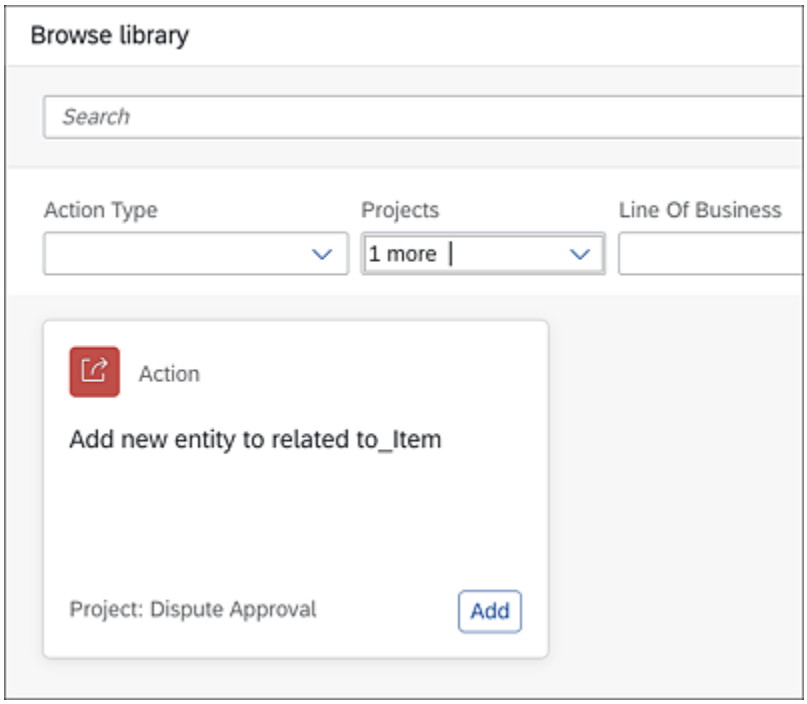


Figure 18.112 Add Dispute Update Action to the Process

Now you have an **Action** call to the dispute update OData. You now just need to drag and drop to map the relevant context data to the OData **Inputs** parameters, as shown in Figure 18.113.

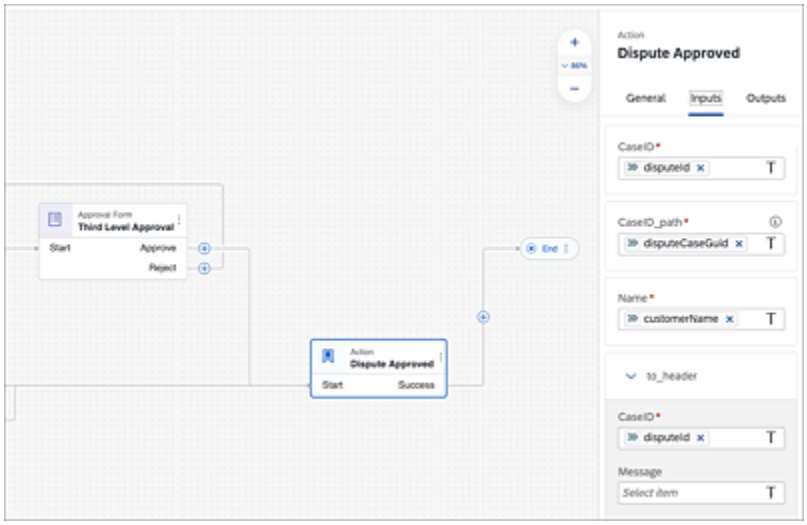


Figure 18.113 Setting the OData Parameters to Update the Dispute Case

Well, that’s it—you’ve designed the workflow process. Now release the project and deploy it after fixing any errors shown in the **Design Console** at the bottom of the editor.

To test this process, you can trigger this API from a REST client such as Postman with the same endpoint described previously, (refer to Figure 18.93) using the service’s OAuth 2.0 credentials. Once it’s triggered, you can check the status of the process under **Monitor** • **Manage** • **Process and Workflow Definitions** shown in Figure 18.114

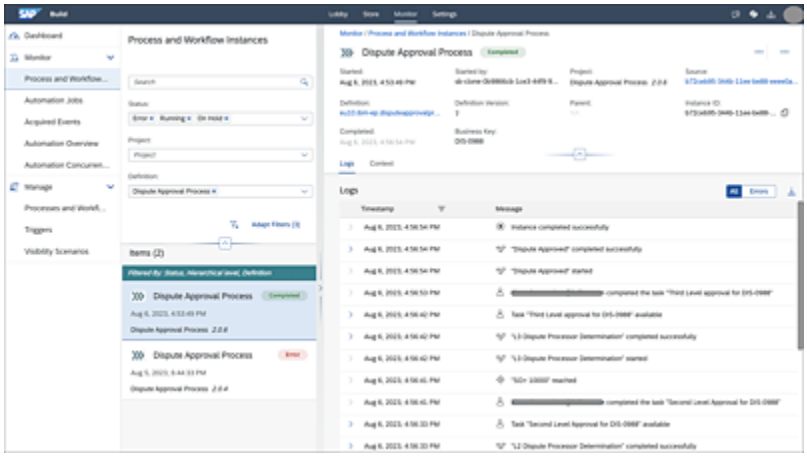


Figure 18.114 Process Log Showing Successful Execution of the Process

A mail task is also configured for first-level approver to receive an email with the notification to process the dispute case. You can add the mail task before the second- and third-level approver task as well.

Note

For email tasks to work, you need to configure an SMTP destination in SAP BTP. Refer to the SAP configuration guidelines to do this at *http://s-prs.co/v569703*.

Approvers can log in to My Inbox where they see the task as shown in Figure 18.115.

That completes the build and test of the use case. You can see how simple and uncomplicated SAP Build Process Automation can be for use case scenarios like this.

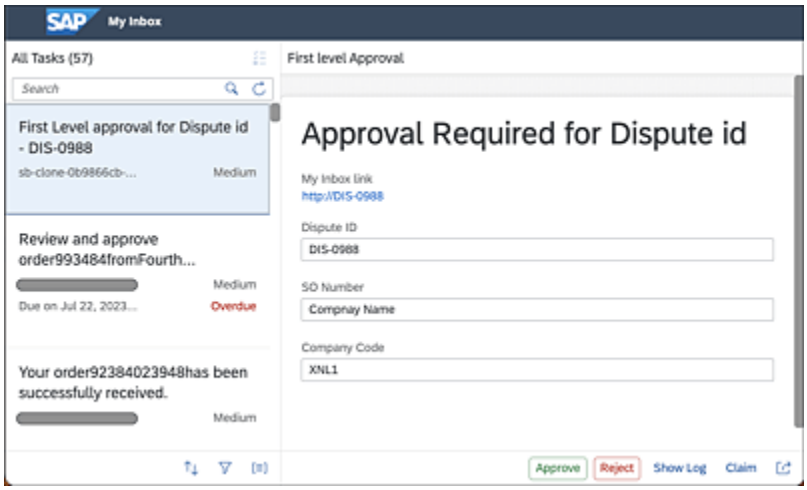


Figure 18.115 My Inbox Task

18.8.3 Design Using SAP Business Application Studio

The same process can be designed using SAP Business Application Studio as you do with the SAP Workflow Management service. Remember that the criteria based on which you choose SAP Business Application Studio and process builder for a workflow is discussed in the previous section. Based on such complexity, you can use SAP Business Application Studio to create more complex scenarios and more controls, along with the option to massage the data using custom scripts. On the contrary, SAP Business Application Studio–based design is more pro developer and goes against the citizen developer flavor as it

involves more coding to achieve the same set of tasks. Let’s see how that can be done.

> **Note**
>
> SAP Build Process Automation is still evolving as of writing this book. Many features available in SAP Business Application Studio–based workflow design aren’t available in process builder.

In SAP Business Application Studio, you create a project with a workflow module using the available templates and start designing the same flow you created in process builder, but let’s add some complication to it:

- You need to do some data manipulation before passing the data from first-level approver to the second and third level.
- You need to have a complex UI along with data addition and validation before its approved at the first-level approver.
- You need to add a timer for user approval to make sure the process isn’t stuck for a long time in the user approval task.
- You need an event-based trigger to end the workflow after the dispute is approved and the update is confirmed by the SAP S/4HANA system. SAP S/4HANA will generate an event once the dispute case is updated.

You start off with creating a workflow-based project as described in Section 18.2.1. We named our project “DisputeApprovalProcess” and the workflow

“DisputeApprovalWorkflow”. If you created the project from the template correctly, your project structure will look like Figure 18.116.

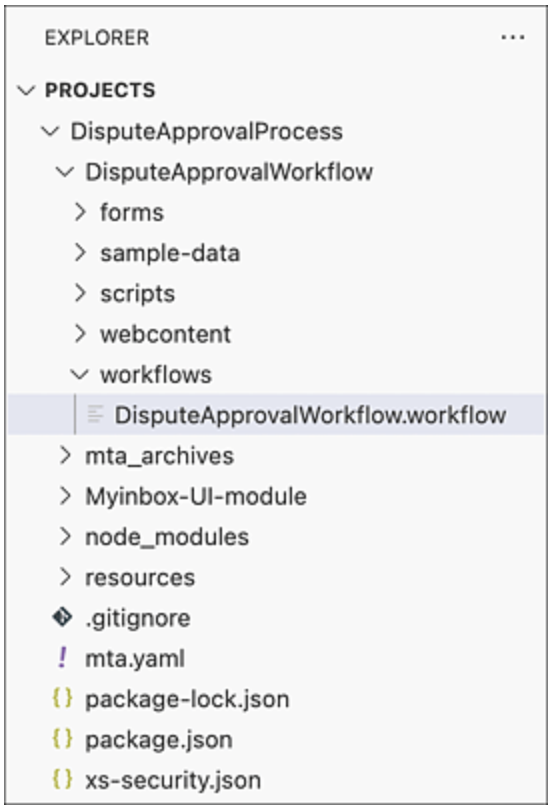


Figure 18.116 Creating a Workflow Project

In the *DisputeApprovalWorkflow.workflow* file, start designing the workflow. The naming of the individual tasks and other script files are up to you.

Create the **Start event**, and connect it to a **Script Task**. The script task is to prepare the request payload for the business rule service call in the next step. The script task may not only be a data preparation task, but also can be used to implement, say, validation of the data, to convert from one data type to another (e.g., string to integer), or to write a bit more complex calculations.

Create a script file that contains the code in Listing 18.1. This will be the payload for the business rules service task. We're setting this structure to the context variable that will be used in the next step. Make sure to change the `RuleServiceId` in the following code to the ID of your deployed business rule service.

```
var ruleRequest={
    "RuleServiceId": " 2da7c433c1024c09a7db210da1c35ffc ",
    "Vocabulary": [
      {
        "DisputeObject": {
          "ApprovalLevel": 1,
      "CompanyCode":$.context.companycode,
          "SOAmount": $.context.soamount
      }
      }
    ]

  }

$.context.FirstLevelBRReqPayload=ruleRequest; //set to context variable
```

Listing 18.1 Simple JavaScript Code for the Script Task to Fetch the Business Rules Decision

The script task will be the next step to start event in the workflow that you're designing and should look like the one in Figure 18.117.

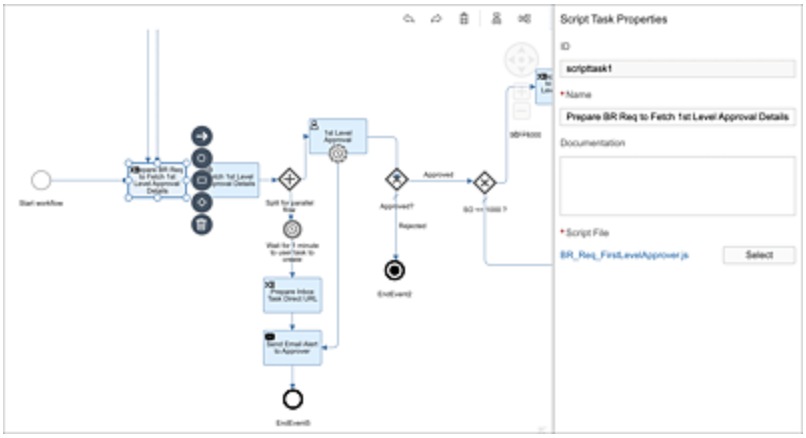


Figure 18.117 Creating a Script Task

Next create a **Service Task** to call the business rule to fetch the first-level approver. Here, set the request and response parameters as shown in Figure 18.118. Remember, you created the response variable in our previous **Script Task**, you need to use the same.

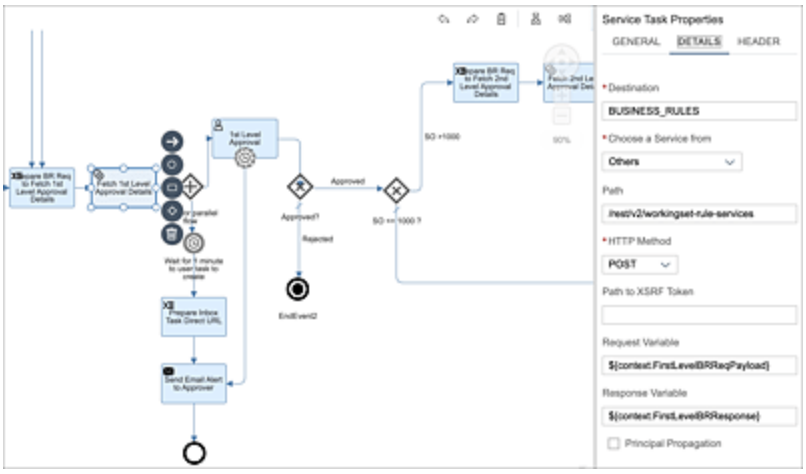


Figure 18.118 Creating a Service Task to Fetch the Business Rule

The next step is to create the user task for the first-level approval. In the flow shown in the previous figure, we're using a **Parallel Gateway** to split the flow into two. One flow goes directly to the **User Task**. The other goes to an **Email** task. Here, you're using two controls. One is an **Intermediate Timer Event** for 1 minute. This is to allow sufficient time for the **User Task** to be created by the workflow runtime before you trigger the **Email** task because we're going to fetch the **Task ID** of this user task instance to be added to the email task as the **My Inbox** link on the email body. This will be easy for the user to click and open the task in My Inbox. If this delay isn't set, there is a chance that the email task gets triggered before the **User Task** is created, and email won't have the **Task Instance** reference. In addition, we're using a **Script Task** here to

build the My Inbox URL with the **Task Instance ID** to be used in the email task.

> **Note**
>
> If you need to add the **My Inbox** URL in the email body in the process builder design, it needs to be sent to the process as an initial payload field of the trigger API or in an action call response that can be mapped as URL in the email body.

The script used here may look something like Listing 18.2.

```
var
wfManagementCockpit="https://mycomp.cockpit.workflowmanagement.cfapps.eu10.hana.ondemand.com/"
var instanceId=$.usertasks.usertask1.last.id;

var link=wfManagementCockpit+"/cp.portal/site#WorkflowTask-DisplayMyInbox?sap-ui-app-id-hint=cross.fnd.fiori.inbox&/detail/NA/";
"/comsapspaprocessautomation.comsapspainbox/inbox.html#/detail/NA/"+instanceId+"/TaskCollection(SAP__Origin='NA',InstanceID='"+instanceId+"')"

$.context.MyinboxLink=link;
```

Listing 18.2 Script to Add a My Inbox Task URL to the Email Task

The `link` variable will be sent in the email body. Please remember to change the `wfManagementCockpit` URL to your SAP BTP workflow management URL in the preceding code.

Next, note that we've added a **Boundary Timer** to the **First-Level Approval** user task. This is to make sure if the user task isn't processed by a set time, a reminder email will be triggered that is the same as the email task you configured in the previous step. Here, we're reusing a task.

Another key point here to note is the UI for the **User Task**. Requirement says you need a complex UI where there is a requirement to validate and edit data before approval. So, we're adding a custom UI module in the *mta* project. Once you add the UI module, your project structure should look something like Figure 18.119.

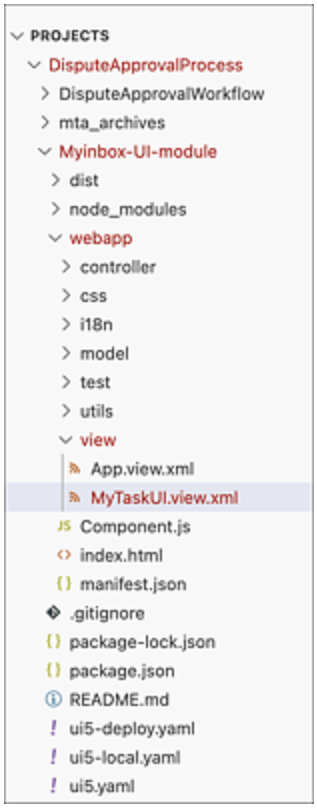


Figure 18.119 UI Module for the User Task

This newly created UI module is used in the **User Task, First Level Approval** shown in Figure 18.120.

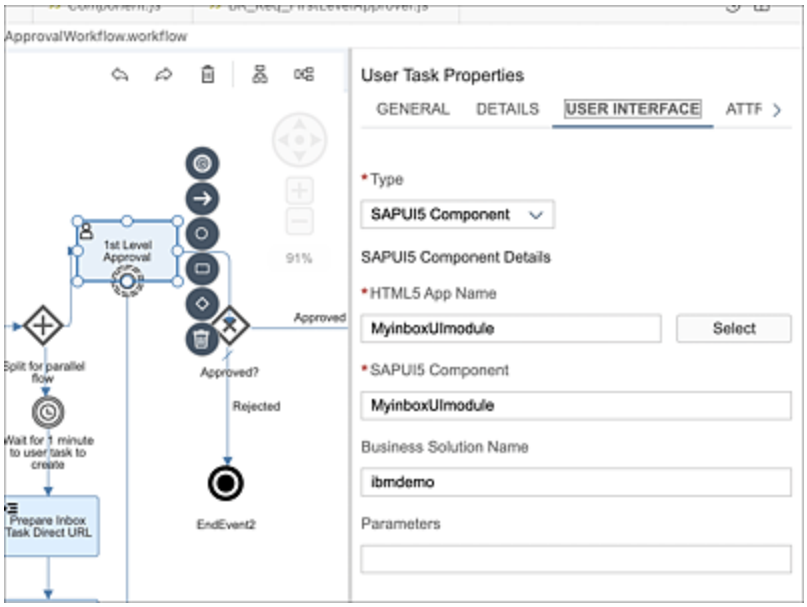


Figure 18.120 Adding the UI Module to the User Task

The next step is the **Condition** check to see if the dispute case is approved or rejected. This remains the same as you did in the process builder–based flow. The remaining flow looks something like Figure 18.121. The set of step repeats for the second-level and third-level approval.

Figure 18.121 and Figure 18.122 show how the flow should look when you finish the design.

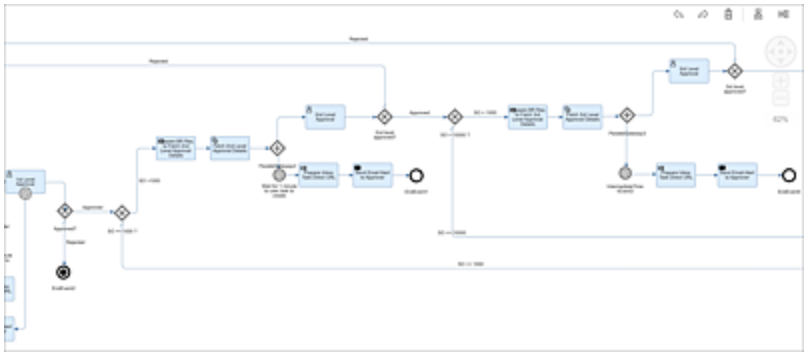


Figure 18.121 Remaining Flow (Part 1)

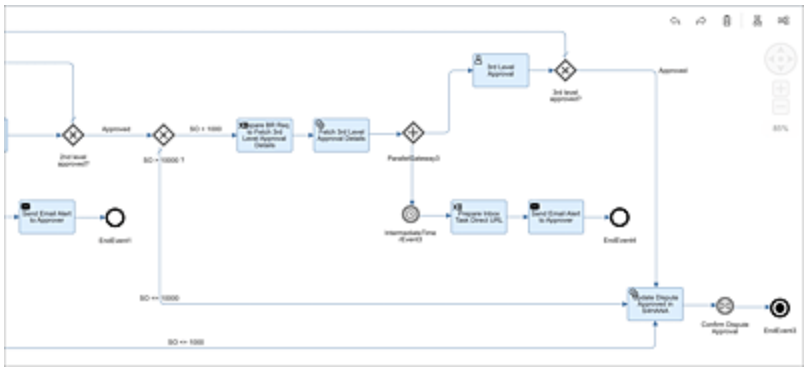


Figure 18.122 Remaining Flow (Part 2)

Finally, after the dispute is approved, the case is updated. But you need a confirmation from the backend before you complete the workflow instance. For this, you use an **Intermediate Message Event**. What this control does is put the workflow in wait mode until it receives the event trigger that this case has been updated. The event is triggered from the SAP S/4HANA system and is published through an event broker such as SAP Event Mesh. You use the workflow API for messages (refer to Section 18.7.1) to subscribe to this event topic in SAP Event Mesh and trigger this **Intermediate Message Event**.

See Figure 18.123 to configure the **Intermediate Message Event**. Here, the message name is **ConfirmDisputeApproval.** This is the name that the API will be using to trigger this intermediate message event for a given workflow instance ID.

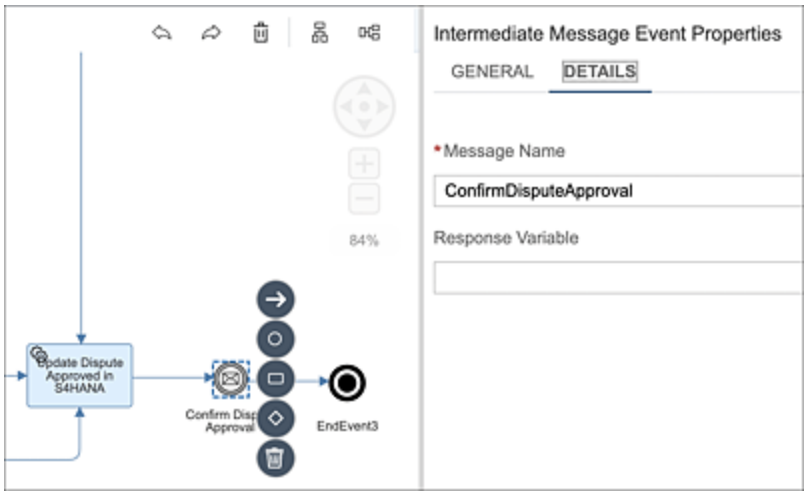


Figure 18.123 Intermediate Message Event That Waits for a Business Event

Now that we've designed the workflow, let's build it and deploy. Once deployed, you can trigger this workflow using Postman or directly from the Manage Process and Workflows app, same way as you did with the process builder design in the previous section. The payload also remains the same.

> **Note**
>
> All your workflow definitions and instances can be centrally monitored and triggerred from the SAP Build **Monitor** tab from the **Lobby**. Workflows created using SAP Business Application Studio and process builder will appear here and can be monitored.

You can see entire execution logs of our process in the Process and Workflow Instances app. You can see the workflows completion steps and approvals. Note the final highlighted steps in the logs (see Figure 18.124). This is the event (intermediate message event in our workflow) that was generated from the SAP S/4HANA system when the dispute case was updated.

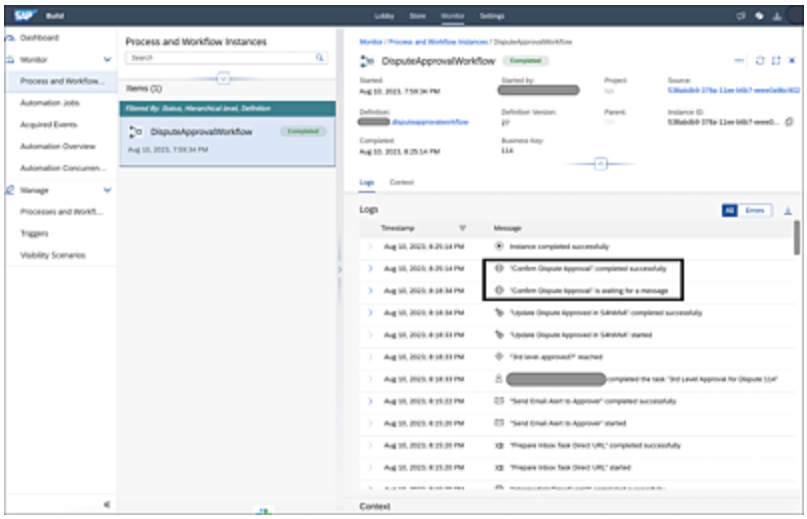


Figure 18.124 Execution Logs of the Workflow

We talked about the custom UI for the inbox and the additional complexity you added. The requirement was to have a button in the screen that would make an OData call to the backend to validate the data on the screen before the approver completed the task. In addition, if the approver wants to navigate to full details of the task in the SAP S/4HANA system, a button needs to be added in the footer. You were able to achieve this using the custom UI, which isn't yet possible within the process builder design.

Figure 18.125 show how the My Inbox screen would like with the custom SAPUI5 module. You can see the **Validate** button that validates the data before submission using an OData call, and another button on the footer called **Additional Information** that lets the user navigate to a specific screen in the SAP S/4HANA system.

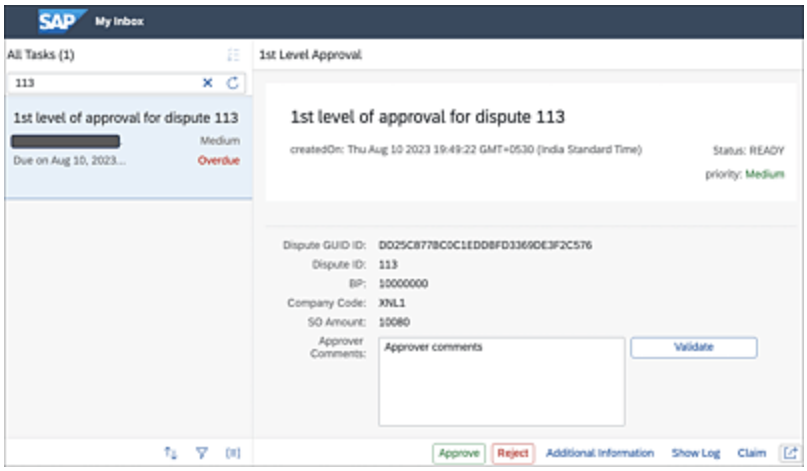


Figure 18.125 Custom My Inbox Screen

18.9 Summary

In this chapter, you learned the designing technique for workflows in SAP BTP. You saw that SAP BTP offers two flavors of workflows. The more mature SAP Business Application Studio–based workflow design targeted more toward developer-centric designing of complex workflows, and the process builder–based workflow design that is part of the SAP Build Process Automation and SAP's low-code/no-code platform to build rapid workflows meant for citizen developers. You also saw a typical use case and how it can be designed using both the flavor and criteria to decide what design tool to use.

SAP Build Process Automation is a relatively new product from SAP. It's in the process of getting more features and is expected to mature in due course.

19 Process Visibility

Any discussion on the workflow management capability in SAP Business Technology Platform (SAP BTP) can't be complete without talking about the process visibility services. This is the one of the most interesting and valuable features in SAP Build Process Automation.

Process visibility refers to a specific offering within the SAP BTP portfolio that provides advanced process monitoring and analytics capabilities. It used to be part of SAP Workflow Management, and while the underlying architecture for SAP Workflow Management components changed, the process visibility component remained the same and it was included in SAP Build Process Automation. Process visibility enables organizations to gain real-time insights into their business processes across SAP and non-SAP systems, driving operational excellence and informed decision-making. It combines various technologies and services to deliver comprehensive process monitoring and analytics capabilities. Some key features and components of process visibility include the following:

- **Process monitoring**
  Organizations can monitor end-to-end business processes, capturing process events, activities, and data from various systems and applications. It provides real-time

visibility into process execution, allowing stakeholders to track the progress and performance of critical processes.

- **Process analytics**
  Process visibility leverages analytics capabilities to transform process data into meaningful insights. It offers advanced analytics tools and prebuilt dashboards for analyzing process performance, identifying bottlenecks, and uncovering optimization opportunities.
- **Event-driven architecture**
  The solution follows an event-driven architecture, capturing and processing events from different systems, applications, and sensors. It enables organizations to respond in real time to events and triggers within their business processes, facilitating proactive decision-making and action-taking.
- **Intelligent business operations**
  Process visibility incorporates intelligent technologies such as artificial intelligence (AI) and machine learning to enhance process monitoring and decision-making. It enables organizations to apply predictive and prescriptive analytics, identify patterns and anomalies, and automate decision-making processes.
- **Integration capabilities**
  Process visibility integrates with various SAP and non-SAP systems, allowing organizations to monitor and analyze end-to-end processes across heterogeneous landscapes. It supports data integration, connectivity, and interoperability, ensuring seamless visibility across systems, applications, and processes.

- **Collaboration and alerts**
  The solution provides collaboration features to facilitate communication and collaboration among stakeholders involved in the process. It allows for sharing information, exchanging messages, and initiating actions based on process events. Alerts and notifications can be set up to notify stakeholders about critical process events or deviations.

Process visibility empowers organizations to drive process optimization, improve operational efficiency, and enhance customer satisfaction. By gaining real-time insights into their business processes, organizations can identify bottlenecks, streamline operations, and make data-driven decisions to stay competitive in today's digital landscape.

It's important to note that SAP BTP is a comprehensive platform offering with various services and components, and process visibility is one of the capabilities provided within this platform. Organizations can leverage other services within SAP BTP, such as integration services, application development tools, and data management services, to enhance their process visibility capabilities further.

In this chapter, we'll discuss how to configure process visibility, explore the testing process visibility scenario, walk you through process monitoring and real-time insights, show you how to add workflow actions to a dashboard, and discuss using application programming interfaces (APIs).

19.1 Configuring Process Visibility

Process visibility comes with SAP Build Process Automation. In the following sections, you'll see how to set up process visibility scenarios in SAP Build Process Automation. There are four broad steps:

- **Configuring the visibility scenario with one or more processes**
  Adding one or more processes is the first step for configuring a visibility scenario.
- **Defining the correlation between processes**
  This is only applicable if more than one process is included.
- **Configuring phases, states, statuses, sub-statuses, attributes, and actions**
  Phases refers to a sequence of process steps, states are various possible states of the instance of a scenario, statuses and sub-statuses refer to the status of a scenario instance, attributes are meaningful information about a scenario instance, and actions refer to how to act on certain situation in a process instance.
  Configuring the performance indicators that show the aggregated information of a measure, grouped by dimension, with the applied filters.

These steps are described in detail in the following sections.

19.1.1 Roles

Access to different features in process visibility can be controlled through the roles listed in Table 19.1. Before you begin configuring process visibility, make sure that your user ID has the `PVAdmin` and the `PVDeveloper` roles assigned to

it. We'll use the same dispute management process defined and used in Chapter 18 to build a process visibility scenario.

Role	Role Description
`PVAdmin`	This role provides permission to do the following: • Access the Event Acquisition and Monitor Visibility Scenarios tiles • Push events • View all the deployed scenario definitions • View the acquired events • Trigger processing of data • View errors related to data acquisition • View information and errors related to the processing of data
`PVDeveloper`	This role provides permission to do the following: • Access the Configure Visibility Scenarios tile • View all the scenarios • Create, edit, save, and activate scenarios • Export the scenarios • Import the scenarios

Role	Role Description
`PVOperator`	This role provides permission to do the following: • Access the Process Workspace app • View metadata of the available scenarios • Fetch all the scenario instances • Perform aggregation, filtering, and search on the scenario instances
`PVEventSender`	This role provides permission to push events to process visibility.
`PVTenantOperator`	This role provides permission to do the following: • Delete events • Delete scenarios definitions

Table 19.1 Key Roles to Work with Process Visibility

Note

Custom roles can also be created out of these standard roles to apply further control on access to users. There are two types of data level authorization possible in process visibility:

- **Scenario level**
  Granular access can be given to a scenario level, for example, the lead-to-order scenario.

- **Scenario attribute level**
  This enables further granular control at an attribute level of a process visibility scenario. For example, the user can access the lead-to-order scenario data where the attribute `salesOrganization` equals `SO1`.

19.1.2 Process Preparation

Before creating a visibility scenario, you need to prepare the process by adding the attributes that you want to include in the scenario:

1. In the process builder, open the process, as shown in Figure 19.1.

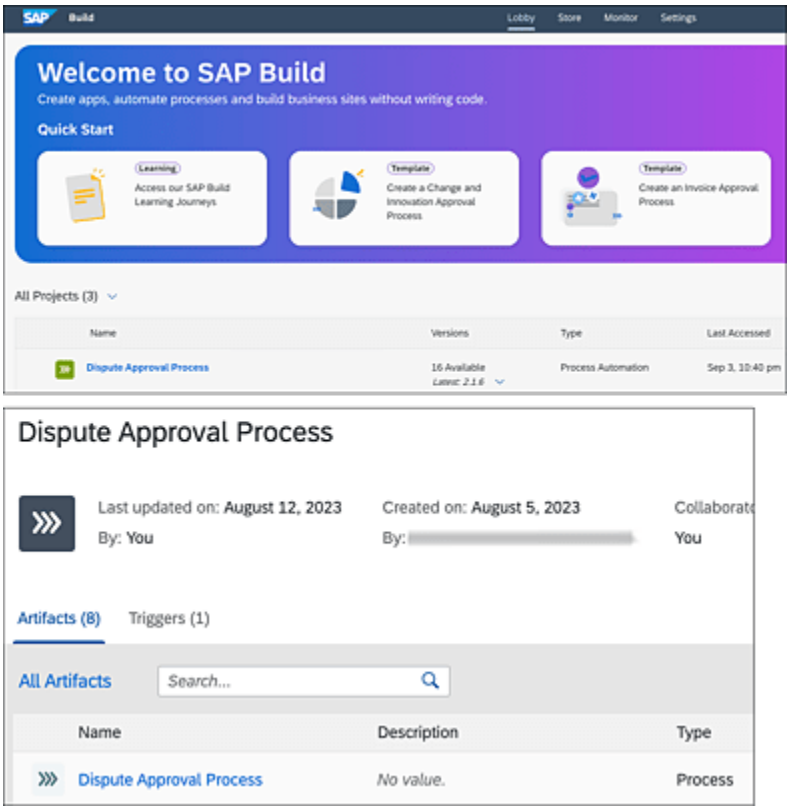


Figure 19.1 Opening the Process in Process Builder

2. On the right-hand side of the screen shown in Figure 19.2, select the **Visibility** tab ❶.
3. Click on the + sign on the right panel ❷ to add the relevant attributes from the process.
4. Add the required attributes as shown in Figure 19.2 ❸.

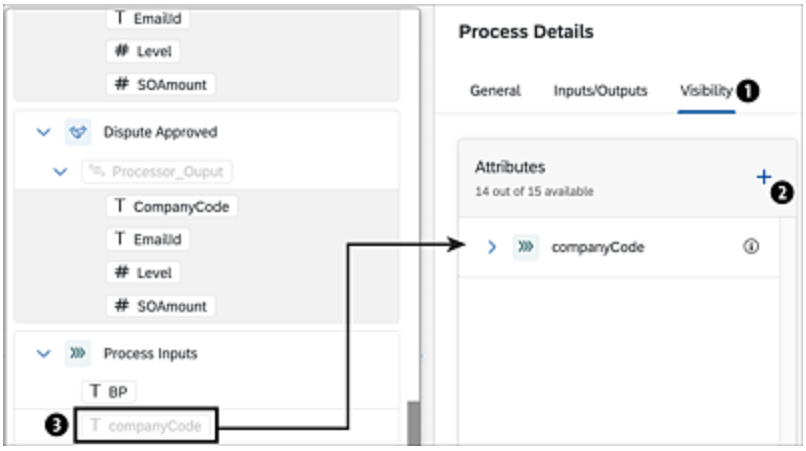


Figure 19.2 Adding Attributes to the Visibility Scenario

5. Follow the same approach to add other attributes as well (see Figure 19.3).

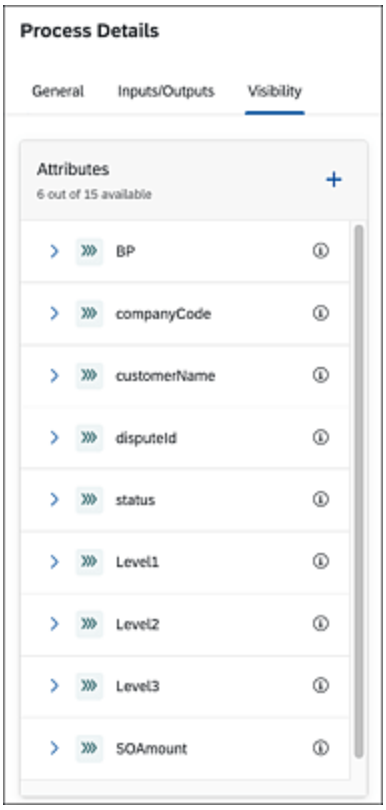


Figure 19.3 Required Attributes Added

6. After all the required attributes are added, click on the **Save** button at the top-right corner of the screen to save the work.

19.1.3 Create Visibility Scenario

There are different ways to create a visibility scenario: from the left navigation pane, or from the process **Overview** screen as described in this section.

From the left navigation pane, click on the + button. Then click on **Create** • **Visibility Scenario**, as shown in Figure 19.4.

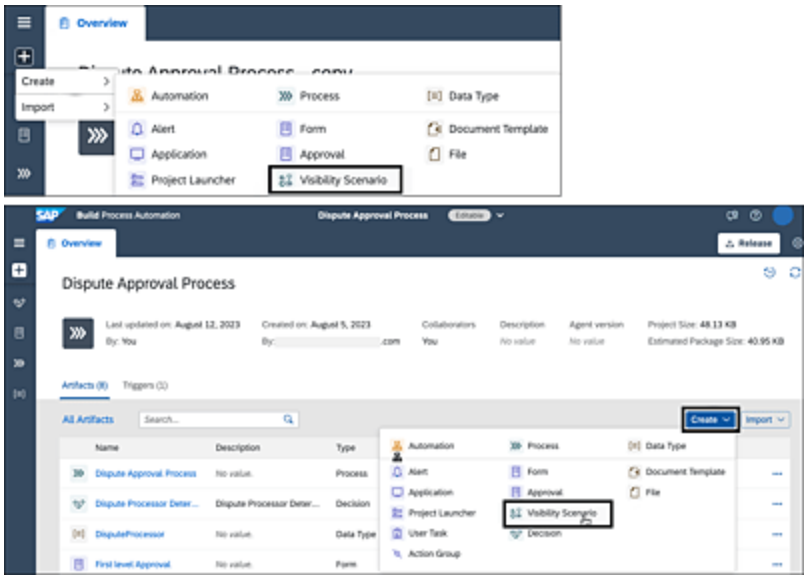


Figure 19.4 Creating the Visibility Scenario

Set **Name**, **Identifier**, and **Description** to something meaningful, and then click on **Create**, as shown in Figure 19.5. The **Identifier** field gets populated automatically as you fill in the **Name** field. This is the unique identifier of the visibility scenario. There can't be another scenario with the same identifier.

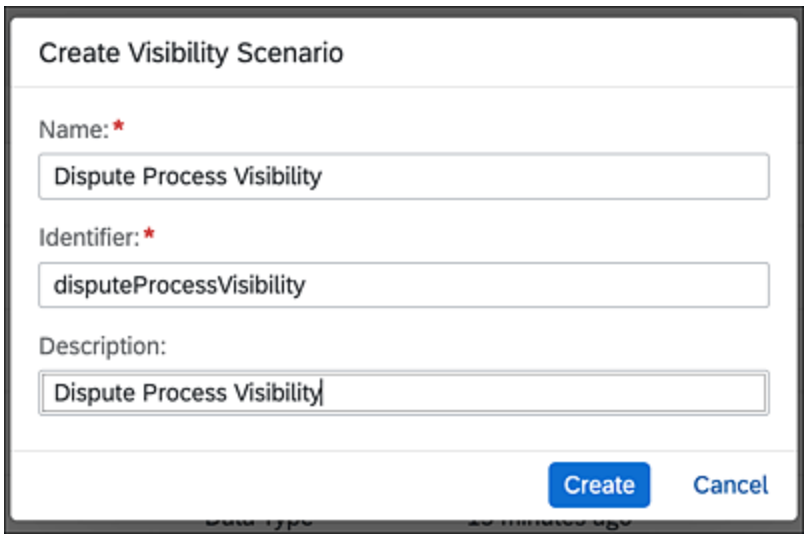


Figure 19.5 Creating the Visibility Scenario

Now that the visibility scenario is created, you need to configure it.

19.1.4 Configure Visibility Scenario

When you open a visibility scenario, by default, the **Processes** tab gets opened. Before you add the required process (or processes), you need to do some minor changes in the **General** tab. This isn't critical as far as the desired functionality is concerned, but it's a good practice to follow. To make it easier to see the instances that are being processed, go to the **General** tab ❶, and enter the value of the **Instances Label** field to "Disputes" and the **Instance Label** field to "Disputes" ❷, as shown in Figure 19.6.

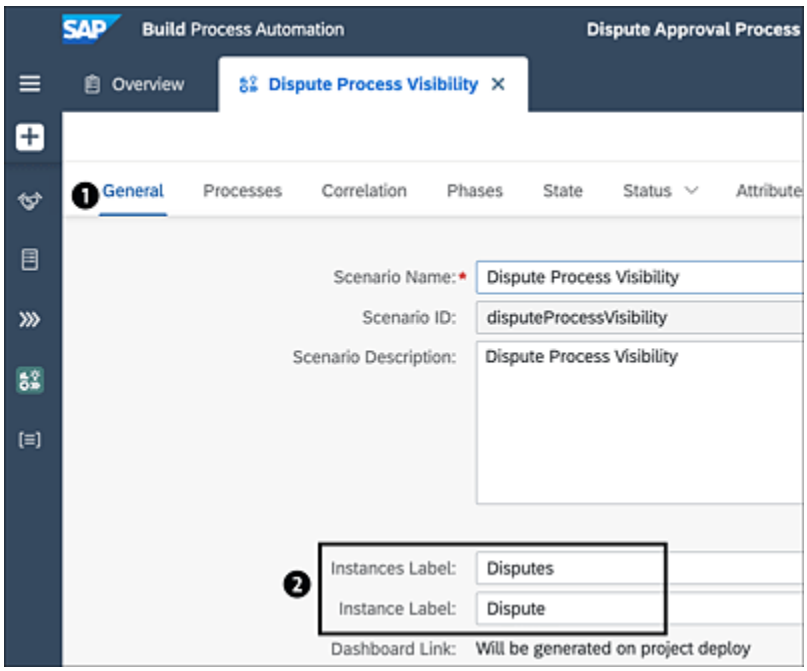


Figure 19.6 Configuring the Visibility Scenario: General Tab

19.1.5 Add Process to the Visibility Scenario

Next, click on the **Processes** tab ❶, click on the + sign ❷, and then click on **Add Process** ❸, as shown in Figure 19.7. In the popup that appears ❹, select the **Dispute Approval Process**, the creation of which is discussed in the previous chapter of this book.

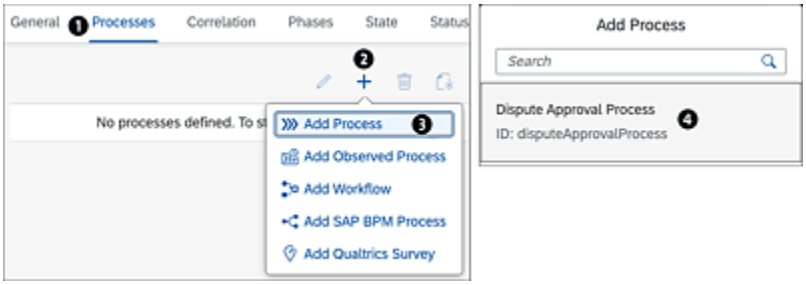


Figure 19.7 Adding the Process to the Visibility Scenario

Note

Here are a couple notes on this step:

- More than one process can be added into a process visibility scenario.
- If any changes are made in the process used in the visibility scenario, it needs to be reimported. This will ensure that the changes made in the process are also available to the visibility scenario.

Once the process gets added to the visibility scenario, you'll be able to see the events that may occur during the process and the context attributes that are available within the process, as shown in Figure 19.8.

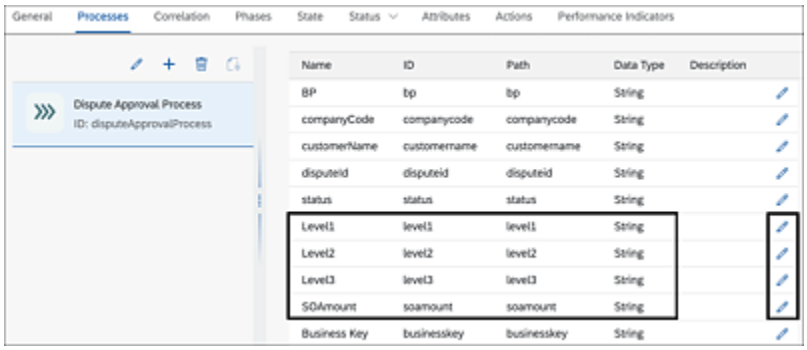


Figure 19.8 Events and Context Attributes Available in the Visibility Scenario

Click the pencil icon next to a context attribute to check the data type and (as necessary) change it to match the correct data type that you designed for the attributes.

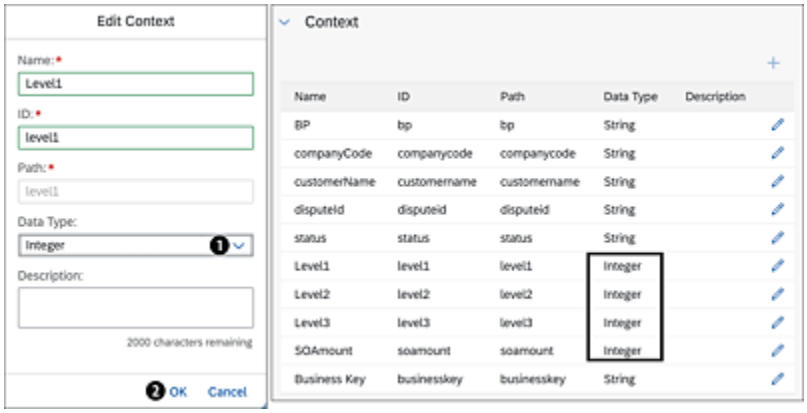


Figure 19.9 Correcting the Data Types of Context Attributes

As shown in Figure 19.9, in this case, the data type of the **Level1**, **Level2**, and **Level3** attributes needs to be changed to **Integer** ❶, after which you should click **OK** ❷. **SOAmount** also needs its data type to be changed to **Double**.

19.1.6 Configure Phases

A process consists of various parts that can be identified through the concept of *phases*. Phases help identify a

sequence of process steps that are part of the visibility scenario. They become handy when the business user needs to track and analyze specific parts of the visibility scenario. Steps to configure a phase are as follows:

1. Go to the **Phases** tab, and then click on **+** to add a phase, as shown in Figure 19.10 ❶.
2. In the **Add Phase** dialog box, provide a **Name** and **ID** for the phase ❷, and then click on **OK** ❸.

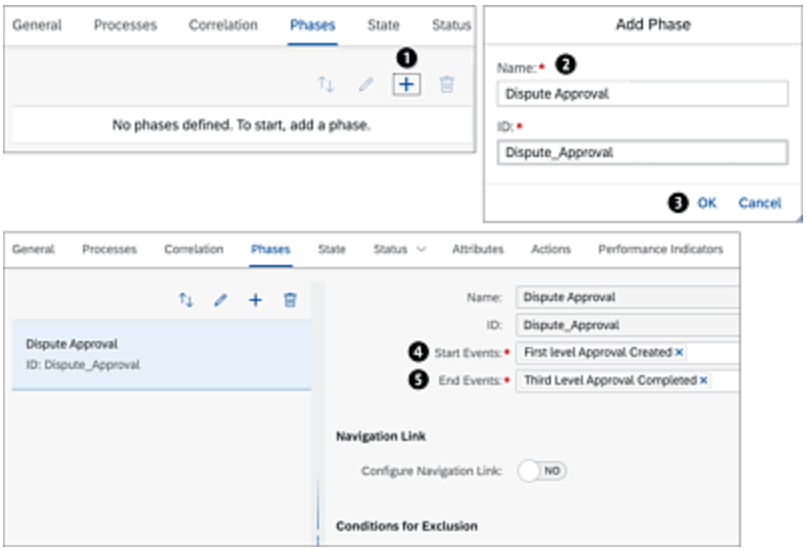


Figure 19.10 Configuring Process Phases

3. Define the start of this phase by choosing the required start event from the **Start Events** dropdown list ❹. Here, you choose **First level Approval Created** (see Figure 19.10).
4. Define the end of this phase by choosing the required end events from the **End Events** dropdown list. Here, you choose **Third Level Approval Completed** ❺.
5. Save the process by clicking on the **Save** button on the top-right corner.

Note

Some optional configurations are available here, as described in the following:

- **Navigation Link**
  If you want the user to refer to some additional information available externally to the running instance of the process, it can be enabled through this navigation link by section. Note that an environment variable of type **Destination** needs to be configured for this purpose. It can be either a complete application URL or can have the host to connect to. The path attribute here needs to be filled with the relative URL.
- **Conditions for Exclusion**
  A phase of the process can be hidden by defining this attribute. Phases of the process will be displayed in the instance details view based on the evaluation of the phase conditions.

19.1.7 Configure Status

Status shows the instance status based on how the instance is progressing. The progress of an instance is determined by targets, phases, and custom events. The process visibility scenario provides some standard statuses (see Table 19.2) and sub-statuses (see Table 19.3).

Status	Description

Status	Description
Critical	An open instance of the process under consideration is in **Critical** status when the elapsed time is more than the target cycle time, or a certain condition is met for the instance, for example, one of the sub-statuses defined under it becomes true.
At Risk	An open instance of the process under consideration is in **At Risk** status when the target cycle time is higher than the threshold target cycle time, or a certain condition is met for the instance, for example, one of the sub-statuses defined under it becomes true.
On Track	An open instance of the process under consideration is in **On Track** status when it’s neither critical, nor at risk.
Completed without Violation	An instance of the process under consideration is in **Completed without Violation** status when it’s completed within the target cycle time without any deviation.
Completed with Violation	An instance of the process under consideration is in **Completed with Violation** status when the process is complete, but the elapsed time is higher than the target cycle time.

Table 19.2 Statuses Available Out-of-the-Box in Visibility Scenarios

There is a default sub-status for each of the statuses as mentioned in Table 19.3. These are used for more fine-grained performance indicators, which is one of the key features of process visibility.

Status	Sub-Status	Description
Critical	**Overdue**	This sub-status appears only if the target is defined.
At Risk	**Threshold Violation**	This sub-status appears only if the target is defined.
On Track	**On Time**	This sub-status denotes that the process instance is completed before the target cycle time decided for the process.
Completed without Violation	**Completed without Violation**	This status appears when a process instance is completed on time without any deviation regarding the allocated time.
Completed with Violation	**Completed with Violation**	This status appears only if the target is defined.

Table 19.3 Sub-Statuses Available Out-of-the-Box in Process Visibility

In our dispute management process, there are three levels of approvals needed based on the sales order amount (SOAmount). You want to measure the process through performance indicators by these approval levels. For

example, you might want to see the number of instances of the dispute management process where the first-level approval is in place but the final, that is, the third-level approval, is still missing. For that purpose, you need to define the following sub-statuses:

- **Critical**
  You want to flag the process instances in **Critical** status when second-level approval is complete, but the third-level approval is still pending beyond a certain target cycle time. In the interest of time, you make this threshold time duration as 2 minutes, although making it a day or two may be more practical in reality. Steps for creating the custom sub-statuses are as follows:
  - Click on the **Status** tab, which is shown in Figure 19.11.
  - Under the **Sub-Stats** section, select the **Critical** sub-status.
  - Click on the **+** button.
  - Add a unique name in the **Name** field. The sub-status **ID** will automatically be populated.
  - Click on the **OK** button
  - Choose **Event A occurred and not Event B in a Timeframe** from the **Expression Type** dropdown. Set the **Duration** to **2 Min** for easy testing of the scenario.

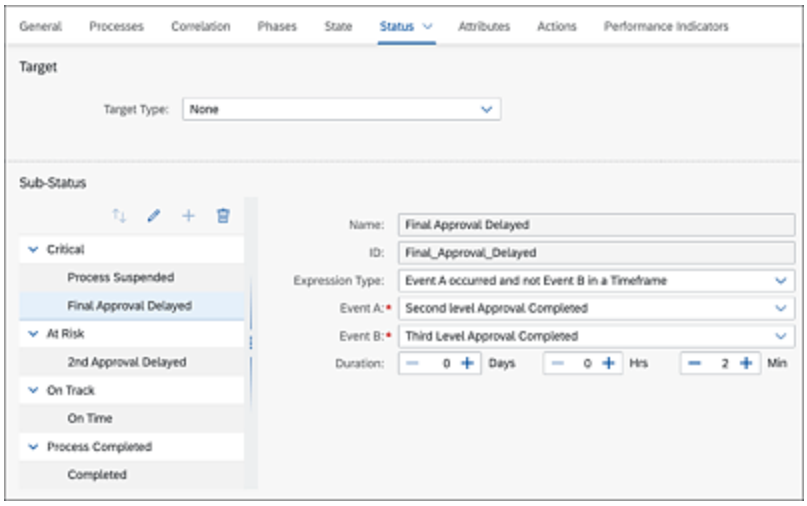


Figure 19.11 Configuring Process Phases

- **At Risk**
  Similar to the previous status, you want to flag a process instance at risk when the first-level approval is complete, but the second-level approval is pending. Follow the same steps as earlier to create this custom sub-status under **At Risk**, which will look like Figure 19.12.

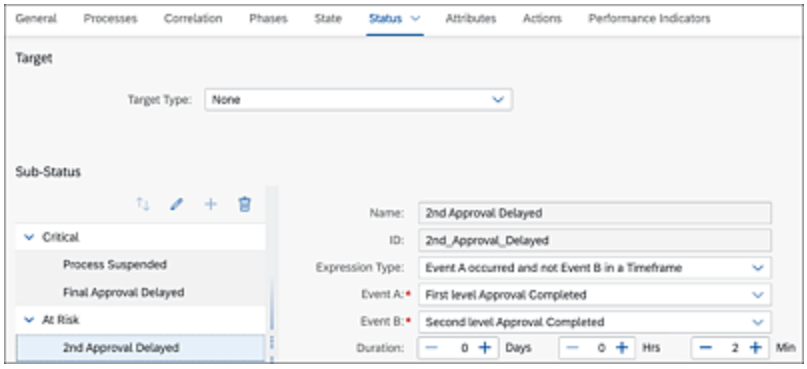


Figure 19.12 Defining More Sub-Statuses

19.1.8 Configure Performance Indicators

The performance indicators give the business users a holistic view of the process and enables them to understand

process performance at a glance. It shows the health of the process grouped by these performance indicators. Each of the indicators are represented by a card in the visibility dashboard.

There are a few performance indicators that come out of the box with process visibility. Additionally, you can configure more per requirement. Here’s an example to show how such custom performance indicators can be created:

1. Choose the **Performance Indicators** tab, as shown in Figure 19.13.

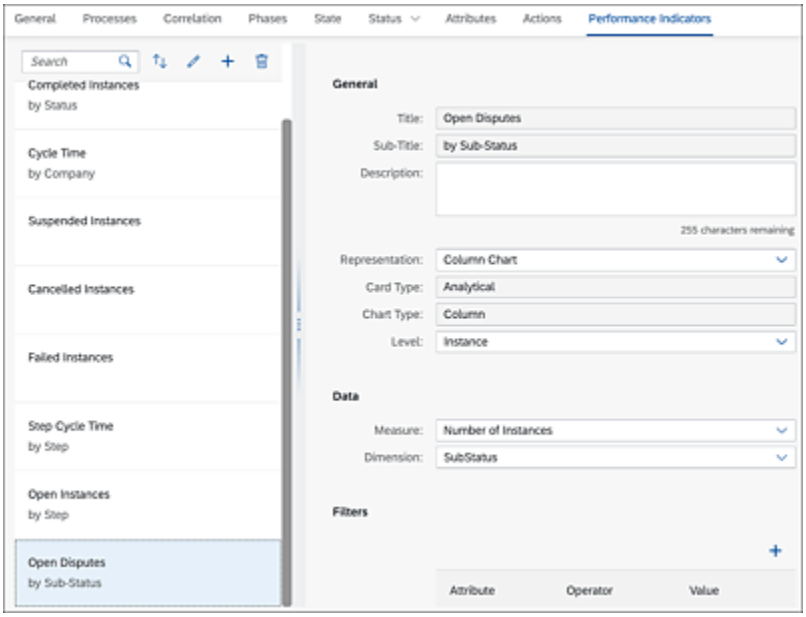


Figure 19.13 Configure Custom Performance Indicators

2. Choose + to add a performance indicator.
3. In the **Add Performance Indicators** dialog box, provide a **Title**, **Sub-Title**, and **ID** for the performance indicator.
4. In the **General** section, select one of the options from Table 19.4 in the **Representation** dropdown list to

decide how to display the performance indicators in the Process Workspace app. While the table shows the various options available for representation, you have options to measure the selected indicator by attributes available at the instance, phase, step, or entity level. These options are available in the **Level** dropdown.

Representation	Description
Header	Shows the title, subtitle, aggregated value of the selected indicator.
List	Shows the title, subtitle, and aggregated value of the selected indicator in the header area. It also shows the indicator grouped by the selected dimension in the chart area.
Bar Chart	Besides showing the title, subtitle, and aggregated value of the selected indicator in the header area, it shows the indicator grouped by the selected dimension in the chart area.
Donut Chart	Information is represented in the same way as the bar chart with the only difference being the type.

Representation	Description
Column Chart	Information is represented in the same way as the bar and donut chart, but the representation is through column chart.
Line Chart	Information is represented in the same way as bar and donut chart, but the representation is through a line chart.
Tables	Title and subtitle are shown in the header area with the list of records showing the selected attributes.

Table 19.4 List of Representation Options

5. Under the **Data** section, because you want to measure the number of disputes, select **Number of Instances** as **Measure**, and select **SubStatus** for **Dimension**.

19.1.9 Release the Project

Now that you're ready with the process visibility scenario, follow these steps to release it:

1. Click the **Release** button at the top-right corner of the screen, as shown in Figure 19.14 ❶.
2. When the **Release Project** popup appears, select the appropriate **Version** radio button and the relevant version-specific comments.

3. Click on the **Release** button at the bottom of the popup screen ❷. This will release the project along with the visibility scenario and make it ready for deployment ❸.

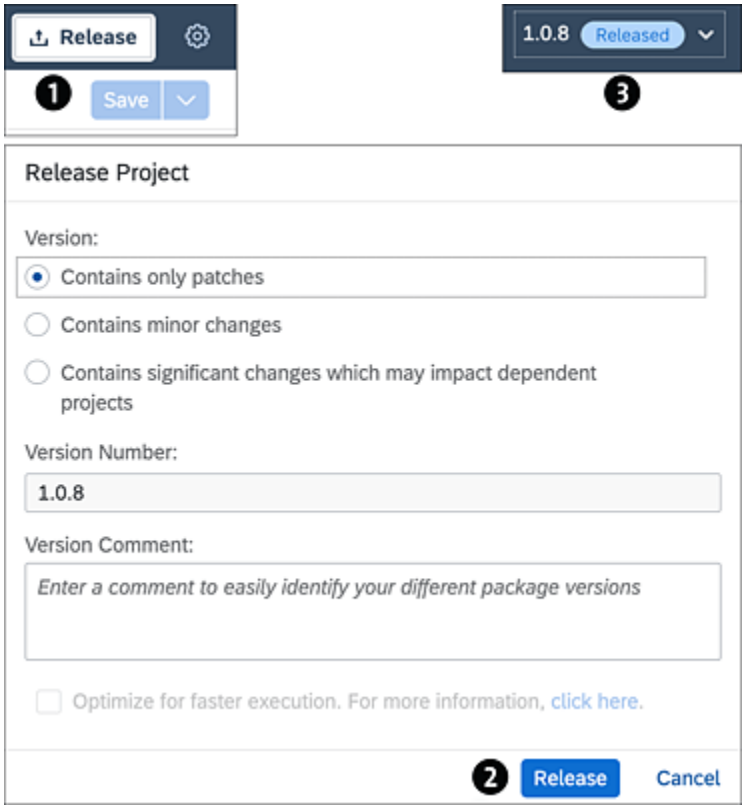


Figure 19.14 Releasing the SAP Build Process Automation Project

Note

Each version of a project is independent and has its own lifecycle status as follows:

- **Editable**
  This is the default status when a project is created. An editable project is considered a draft and doesn't appear in the list of projects in the lobby.
- **Released**
  Denotes that the project is ready to deploy. For already-

released projects, it needs to be released again to create a new project version. You can delete a released project.

- **Deployed**
  A project can only be run when it’s deployed.

The lifecycle status of a project can be reset to its previous status. To delete a project, it must be in **Released** status. You can’t delete a **Deployed** project.

Version Control

Following are a few notes about the **Version Number** field:

- **Contains only patches**
  Used mainly for bug fixes denoted by the number after the second decimal, for example, 1.0.1.
- **Contains minor changes**
  Used for minor changes and bug fixes denoted by the number after the first decimal, for example, 1.1.0.
- **Contains significant changes that may impact dependent projects**
  Used when there are major changes that may lead to incompatibility between versions and impact other projects. This is denoted by the first number, for example, 2.0.0.

19.1.10 Deploy the Project

Now that the project is in **Released** status, the next step is to deploy the project as shown in Figure 19.15:

1. Click on the **Deploy** button at the top-right corner of the screen. In the popup window that appears, click on the **Next** button at the bottom-right corner.

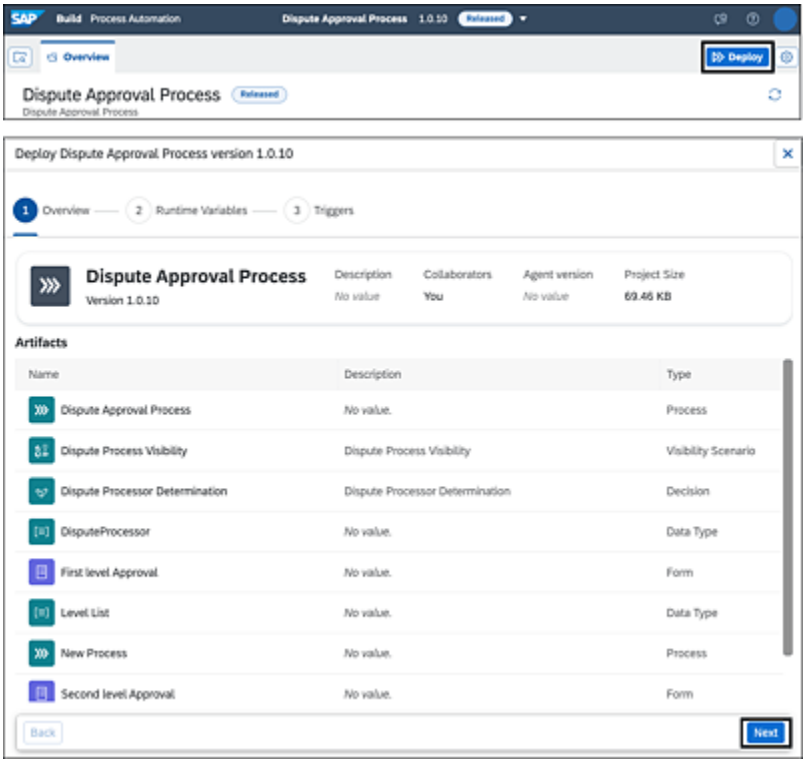


Figure 19.15 Deploy Project: Step 1 and Step 2

2. Under **Runtime Variables**, you can choose the **Set new value** option or **Use existing value** option. This attribute decides the SAP BTP destination for the SAP Build Process Automation project.
3. If you choose the **Use existing value** option, as shown in Figure 19.16, you can choose your target SAP BTP destination from the destination dropdown list.

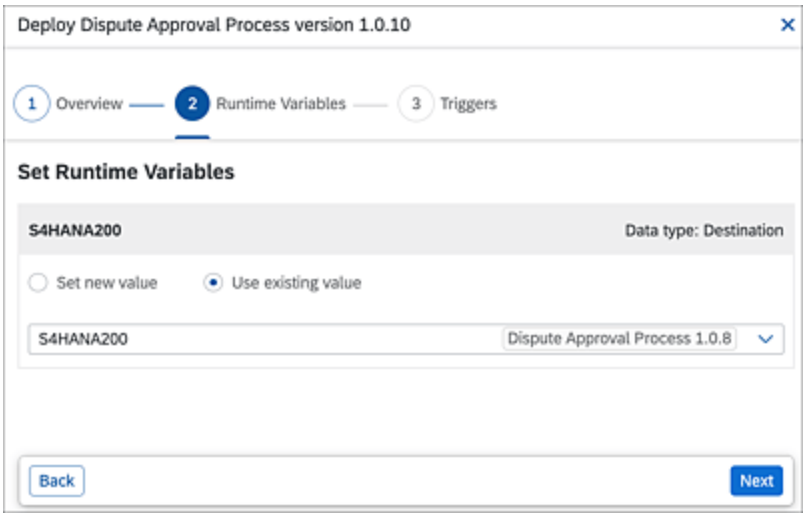


Figure 19.16 Deploy Project: Step 3

4. The final step is to select the trigger, and click on the **Deploy** button, as shown in Figure 19.17.

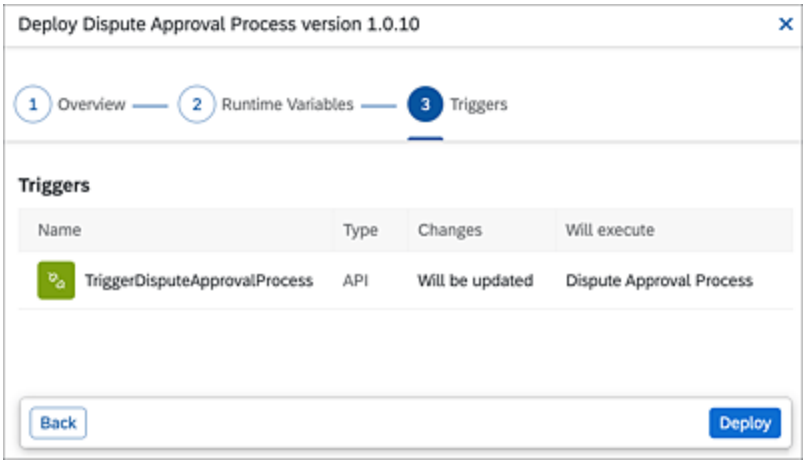


Figure 19.17 Deploy Project: The Final Step

Once the project is successfully deployed, the status at the top will be changed to **Deployed**, as shown in Figure 19.18.

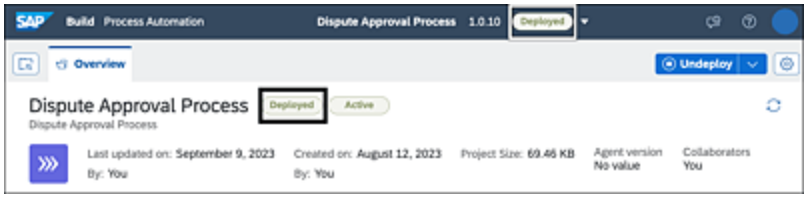


Figure 19.18 Project Status Post: Successful Deployment

19.2 Testing the Process Visibility Scenario

Now that the project is successfully deployed, let’s test the visibility scenario. There are different ways to launch a process visibility scenario. For this purpose, you need to follow these steps:

1. Open the SAP Build lobby, and click on the **Monitor** option at the top, as shown in Figure 19.19 ❶.
2. Click on the **Visibility Scenarios** tile or in the left navigation pane.
3. In the **Visibility Scenario** screen, identify the desired visibility scenario from the list. You can use the search option to quickly find the scenario that you want to open, as shown in Figure 19.19 ❷.
4. You may need to click on it to see the details such as activation date, version, and so on to identify the correct one.
5. Click on the **Navigate to Dashboard** button ❸ at the top right corner.

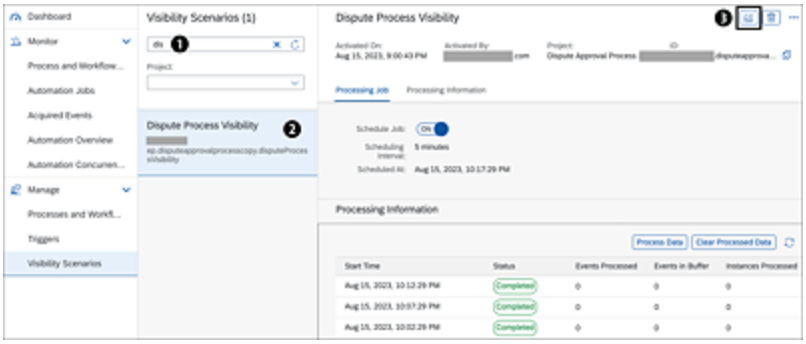


Figure 19.19 Launching the Visibility Dashboard: Steps 1 and 2

Finally, the process visibility dashboard is launched, as shown in Figure 19.20. You can see the sub-status that you defined earlier to measure disputes by sub-status.

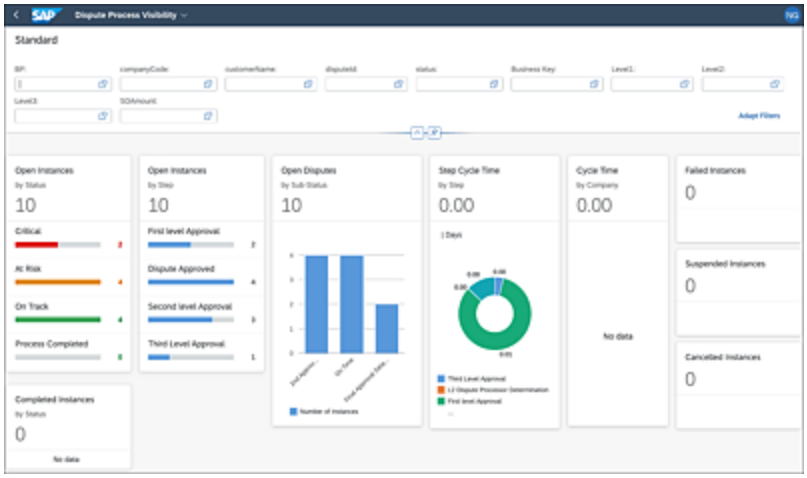


Figure 19.20 Process Visibility Scenario: Launched

Note

Each tile in this dashboard is interactive. Clicking on them shows the details about the objects under consideration. The **Standard** tab at the top of the screen is for fine-grained search.

19.3 Process Monitoring and Real-Time Insights

Now that you have some idea about the interactive feature, let’s explore it further. If you want to see the details of the disputes that are stuck at the second-level approval, you need to click on the **Open Instances** tile. This will take you to the detail view of those disputes, as shown in Figure 19.21.

You can drill down further to see when the second-level approval was given and how much time it took to get it, as shown in Figure 19.22. All this information can play an important role in improving the business process being measured through the process visibility scenario.

From the example used here, it may look like process visibility can only cover SAP systems. However, process visibility can be very effective in gaining end-to-end process visibility across heterogeneous landscapes.

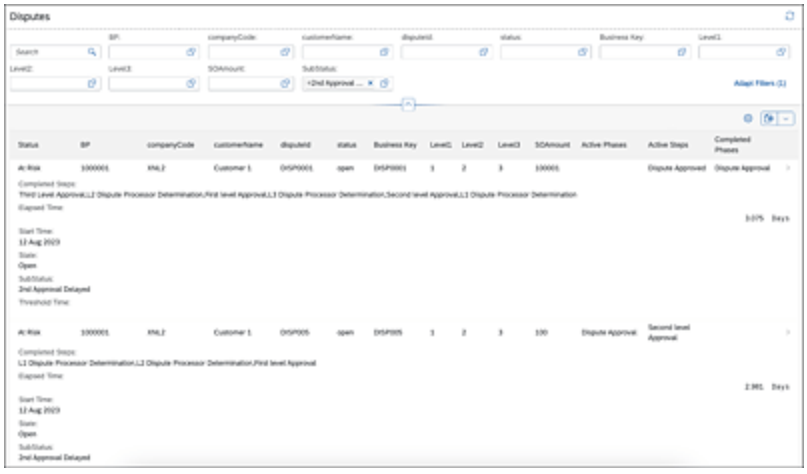


Figure 19.21 First-Level Drilldown Feature in the Process Visibility Dashboard

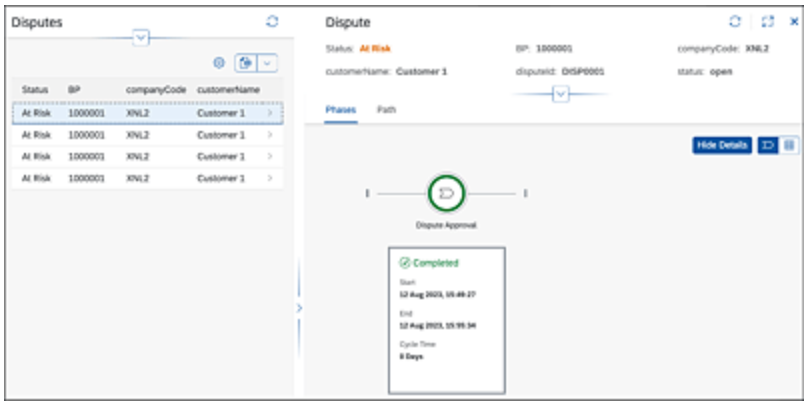


Figure 19.22 Second-level Drilldown Feature in the Process Visibility Dashboard

19.4 Add Workflow Actions to the Dashboard

It’s possible to trigger certain actions based on the situations in the processes for which a visibility scenario is created. Trigger workflow is the feature through which this can be achieved by following these steps (see Figure 19.23):

1. Open the visibility scenario, and go to the **Actions** tab.
2. Click on the + sign ❶.
3. Once the popup window appears, enter a name in the **Name** field ❷. The **ID** field automatically populates ❸. There are two types of action possible as available in the **Type** dropdown ❹, as shown in Table 19.5.

Action Type	Description
Navigational	If you want a web application to be opened for further reference/action to respond to a specific situation in the process, this option can be used to give the user the option to open it from the visibility dashboard.

Action Type	Description
Trigger Workflow	If you want a separate workflow to be triggered by the user or system, this option needs to be used. Note that as of now, this only supports workflows created in SAP Business Application Studio and deployed in the now deprecated SAP Workflow Management service in SAP BTP.

Table 19.5 Action Types Available

4. Select one of the following options from the **Sentiment** field ❺, which determines the appearance of the action in the visibility dashboard:
   - **Neutral**: Provides a neutral appearance while providing the navigation link/workflow trigger button.
   - **Positive**: Provides a positive appearance while providing the navigation link/workflow trigger button.
   - **Negative**: Provides a negative appearance while providing the navigation link/workflow trigger button.
5. Click **OK** ❻.

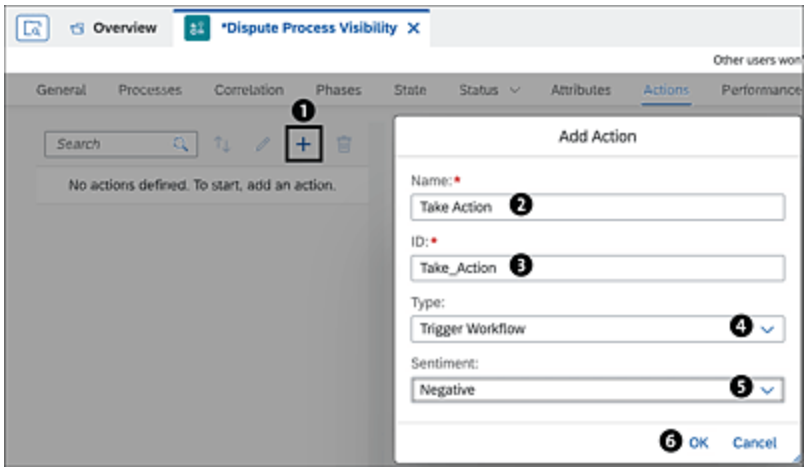


Figure 19.23 Adding Action into the Visibility Scenario

Now that the action is added, let's define the condition based on which this action will be triggered (see Figure 19.24):

1. Click on the + sign under **Condition**.
2. Select the **Attribute** from the list of all attributes available for the visibility scenario that you defined earlier. Here you want the action to be triggered when the process instances are suspended for some reason, so, you choose **Status** in the **Attribute** field.
3. Select the **Operator**. The only two options are **equal to** and **not equal to**, which are self-explanatory. Choose **equal to**.
4. Select **Value** of the sub-status that you want the action to be triggered. Among all the sub-statuses available to this visibility scenario, choose **Process Suspended** for our objective.
5. Click on the **OK** button to complete defining the condition.

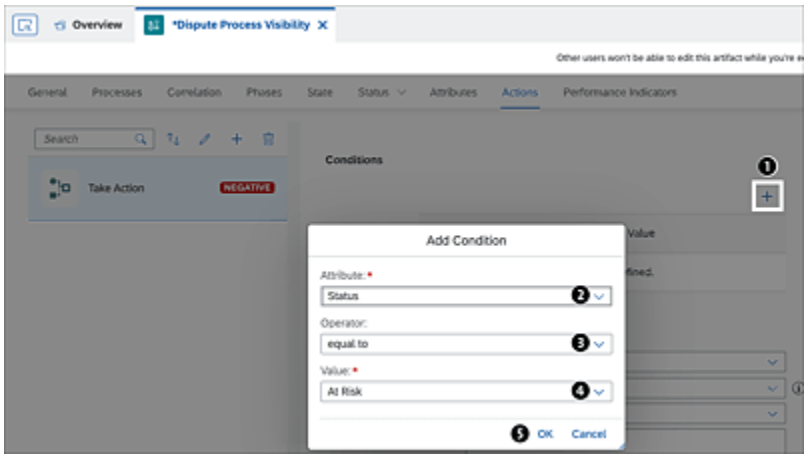


Figure 19.24 Adding a Condition for the Action

Note

Had you chosen the **Navigational** type of action, you would have to fill in the following two fields:

- **Environment Variable**
  This points to a relative path to another application.
- **Path**
  If you have a fixed URL that needs to be opened as corrective action, you don’t need any value in the **Environment Variable** field.

Here, you’re using the other type of action, **Trigger Workflow**, by following these steps and shown in Figure 19.25:

❶ **Tigger Type**
There are two options available in this dropdown field: **User** and **System**. You choose **User** because you want to give the user an option to trigger this action from the visibility screen. If you want an automatic trigger, you have to choose the **System** option.

❷ **Environment Variable**
The environment variable can be of the type **Destination** which needs to be created in the business process project. This is a very important prerequisite for the **Trigger Workflow** type of action. Certain additional parameters need to be added while configuring this destination. The same is explained nicely in the SAP Help document (see *http://s-prs.co/v569702*). This same destination needs to be chosen during deployment of the project, as shown in step 2 (**Runtime Variables**) of Figure 19.26.

❸ **Workflow Definition**

All the workflows developed in SAP Business Application Studio and deployed in the SAP BTP subaccount are visible in this dropdown field. For this purpose, use the **DisputeApprovalWorkflow** that was already deployed. This workflow triggers an email based on values passed through the context.

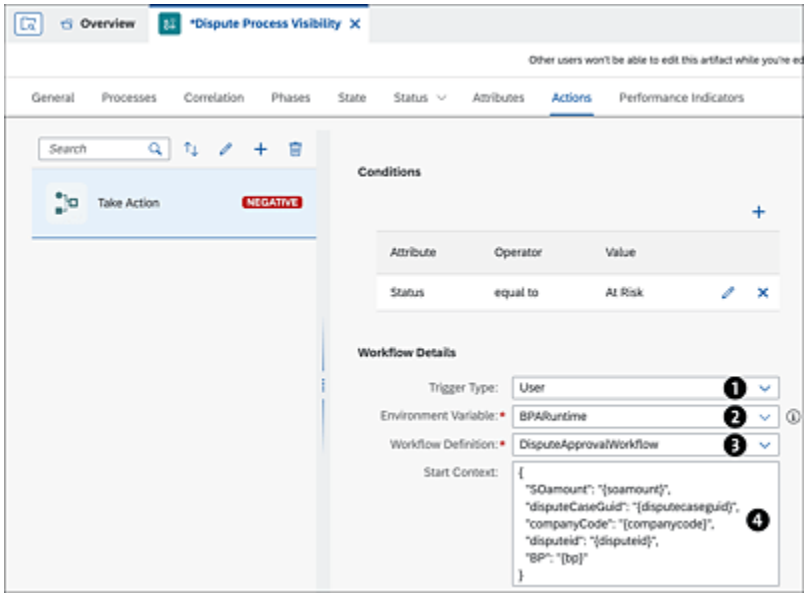


Figure 19.25 Add Action Workflow to Visibility

❹ **Start Context**

You need pass on the value through this field to our target **DisputeApprovalWorkflow** that will be triggered by the user per the condition you set earlier (refer to Figure 19.24).

Finally, you save these configurations by clicking on the **Save** button at the top-right corner of the screen. Because you made all these changes, you must release and deploy the business process project again following the steps

shown in Figure 19.14. While there is no change in the release process in general, deployment of the project needs the additional **BPARuntime** destination, as shown in Figure 19.26.

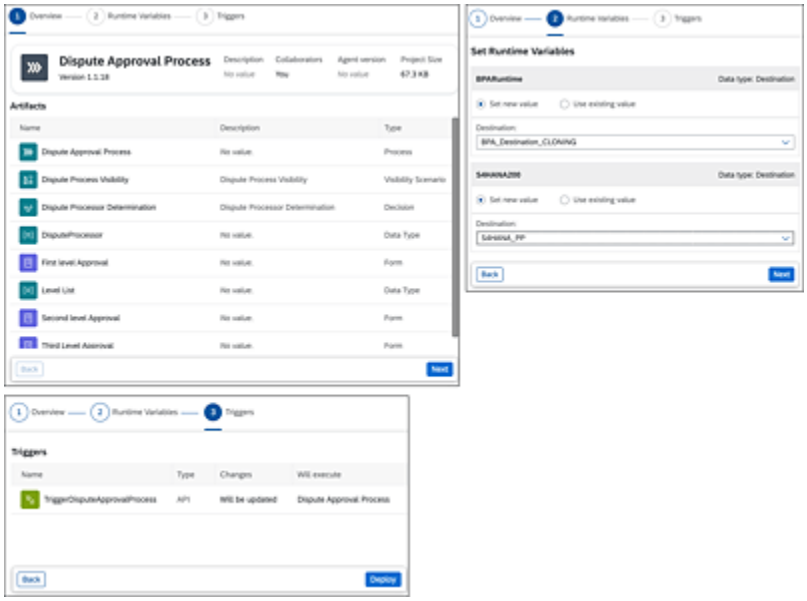


Figure 19.26 Deployment of the Project after Configuring the Action to Trigger Workflow

Now, let's see the effect of this **Trigger Workflow** action. For this, launch the visibility scenario for the process, as shown in Figure 19.27.

Because you defined the condition to trigger the workflow for the process instances where the status is **At Risk**, click on the **At Risk** section of the **Open Instances** tile in the visibility dashboard, as shown in Figure 19.27 ❶. It will navigate to the detail screen.

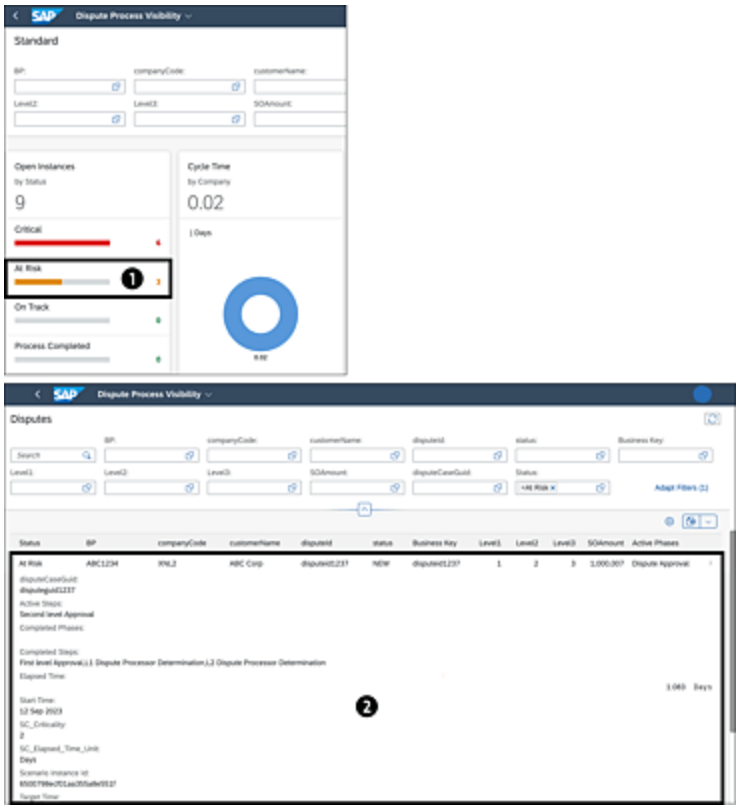


Figure 19.27 Testing Trigger Workflow Action: Steps 1 and 2

In the detail screen, click on the record ❷. This will navigate to the next level of detail, as shown in Figure 19.28 and Figure 19.29. On clicking any of the records in the left-hand pane ❸, the right-hand side of the screen will show further details. On the top-right part of the screen, the **Take Action** button appears, showing the action workflow that you configured previously. Before you click on this button to trigger the workflow action, let's go to the **Action Logs** tab in Figure 19.28 ❹.

If you click the **Take Action** button ❺ in Figure 19.29, the `DisputeApprovalWorkflow` gets triggered, and a record will appear in the action logs list ❻.

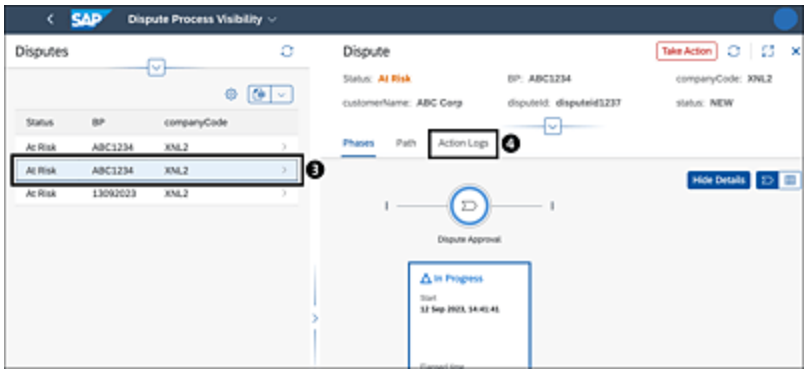


Figure 19.28 Testing Trigger Workflow Action: Steps 3 and 4

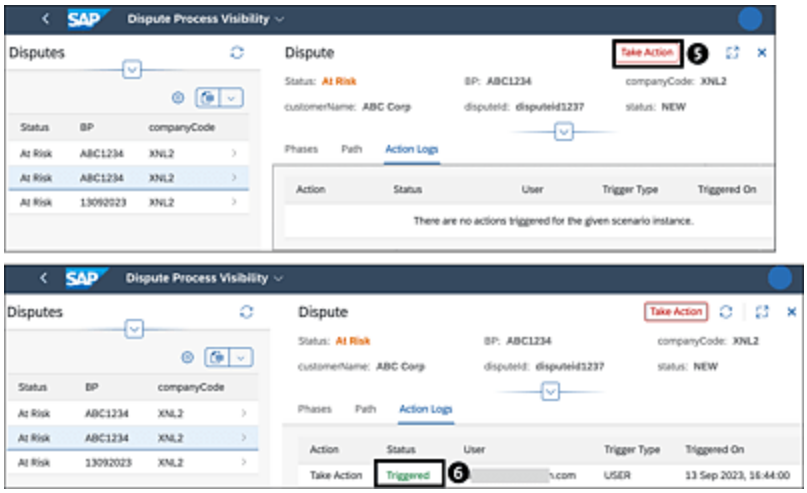


Figure 19.29 Testing Trigger Workflow Action: Steps 5 and 6

19.5 Using Application Programming Interfaces for Process Visibility

The Process Monitoring dashboard is a key component of the process visibility service. What this dashboard offers is configurable to a large extent, it's limited by the look and feel, that is, the UI aspects. To address this, SAP has provided a whole bunch of APIs with which you can deploy a new visibility scenario, push events to the scenario, and get details of the process scenario instances. A complete custom dashboard can also be developed using these APIs.

There are two types of APIs available for process visibility: OData-based APIs and REST-based APIs. As is common for any API-based communication, authentication is required to use these APIs. Currently, OAuth2 type of authentication is supported.

SAP Business Accelerator Hub provides the complete list of these APIs, but here are some of the most useful ones:

- **OData-based APIs**
  The OData-based API exposes scenario instance information to gain visibility on end-to-end processes, as shown in Table 19.6.

API	Description	Roles Require
`/odata/v1/metadata/Scenarios`	Retrieves metadata of scenarios that can be accessed by the user	`PVOperator`, `PVRestrictedOper`
`/odata/v1/{scenarioId}/Instances('{scenarioInstanceId}')`	Retrieves scenario instance details for the scenario instance ID	`PVOperator`, `PVRestrictedOper`
`/odata/v1/{scenarioId}/Instances('{scenarioInstanceId}')/Phases`	Retrieves phase details of a scenario	`PVRestrictedOper`

Table 19.6 Useful OData APIs for Process Visibility

- **REST-based APIs**
  The REST-based APIs help list and manage scenario definitions, as shown in Table 19.7.

API	Description	Roles Required

API	Description	Roles Required
`/rest/v1/scenario-definitions/{scenarioId}/data-executions`	Triggers process execution for the scenario ID for which input is given here	`PVAdmin`
`/rest/v1/scenario-definitions/{scenarioId}/data-executions`	Fetches 20 of the most recent process data execution logs for the given scenario ID	`PVAdmin`
`/rest/v1/data-acquisition/error-messages`	Fetches the error messages that occurred during data acquisition	`PVRestrictedOperator`

Table 19.7 Useful REST APIs for Process Visibility

19.6 Summary

The process visibility component of SAP BTP offers a real-time end-to-end visibility of a running process irrespective of whether they are running in and/or out of SAP backend systems. This is an important component in the former SAP Workflow Management service and now the new SAP Build Process Automation. More than one process can be monitored through this service. This is a no-code tool through which certain statuses and sub-statuses of a process instance can be configured. The process monitoring dashboard is the most interesting component of the process visibility service. It provides an out-of-the-box dashboard with meaningful graphs and charts to show the health of a business process. Besides the out-of-the-box performance indicators, several custom performance indicators can be configured. It also provides many APIs, using which all the information that is available to the out-of-the-box monitoring dashboard can be made available to the external world. Using these APIs, you can develop a completely custom monitoring dashboard per requirement. In summary, we can say that the process visibility service is one of the most interesting and useful components of SAP BTP.

20 Task Processing with My Inbox

Now that you've seen how to design and deploy a workflow, we need to look at how to get the user task assigned to a given user and how the user can access it. You already learned that the user task gets assigned to the My Inbox app. In this chapter, we'll see how to process the task using My Inbox and the various options available in on-premise and SAP Business Technology Platform (SAP BTP).

You've learned about the various flavors of workflows for SAP S/4HANA and about the options available in SAP BTP. You also saw that each workflow option has a critical task called the user task. As you know, one of the major tasks in a workflow is the approval task. In most cases, a workflow will have one or more user tasks where one user (or a set of users) needs to take an action to approve or reject a particular set of data before it gets processed further, such as purchase requisition and purchase order approval in the source-to-pay process, leave approval in the HR process, and so on. All these user tasks need to present the set of data in a screen and give the option for the user to approve or reject, and the data needs to be processed according to

the decision. My Inbox is the app that lets you do this activity in a workflow.

As you know, workflows can be designed either in SAP S/4HANA or SAP BTP. My Inbox also comes with respective options for SAP S/4HANA and for SAP BTP. We'll look at the app for SAP BTP in the following sections and discuss the SAP Task Center, which is the latest offering from SAP to combine multiple inboxes into a universal inbox.

Note

My Inbox has been discussed at length in earlier chapters of the book. To get an overview of the on-premise version of My Inbox, check out Chapter 11, Section 11.1. To see how to set up scenarios and implement visualizations, see Chapter 11, Section 11.2. To learn more about substitutions in My Inbox, see Chapter 8, Section 8.5.

20.1 My Inbox for SAP Business Technology Platform

We talked about SAP Build Process Automation and SAP Workflow Management in previous chapters. There we came across a user task similar to on-premise workflows in which the user needs to take decisive action on a workflow instance to approve or reject a request or take another action depending on the requirement. This is done using an SAP Fiori app called My Inbox, but SAP BTP uses a different app than the on-premise one. All of our on-premise workflows use the on-premise version of My Inbox, but all

the SAP BTP–based workflows, whether SAP Build Process Automation or SAP Workflow Management, use My Inbox on SAP BTP. Let’s delve into the configuration and set up of the SAP BTP version of My Inbox.

20.1.1 Standard App in SAP Build Process Automation

In our SAP Build Process Automation example build, we created three levels of approvals. All of these approval steps are either an approval form in SAP Build Process Automation or a user task in a workflow based on SAP Workflow Management.

The My Inbox app is available from the **Lobby** screen of the SAP Build Process Automation service, as shown in Figure 20.1. This can be used for testing purposes.

Figure 20.1 My Inbox Link in the Lobby of SAP Build Process Automation

The My Inbox in SAP BTP offers all the features explained previously in Chapter 11, Section 11.1 and Section 11.2, and in Chapter 8, Section 8.5, including substitutions. You can view/hide the workflow logs, claim/release the work item, and so on.

In Figure 20.2, you can see what the My Inbox looks like. You can see the list of tasks in the left-hand side, which gives the key details of the task. Clicking on the task provides a

detailed view with various actions you can take on this task, such as **Approve** or **Reject**.

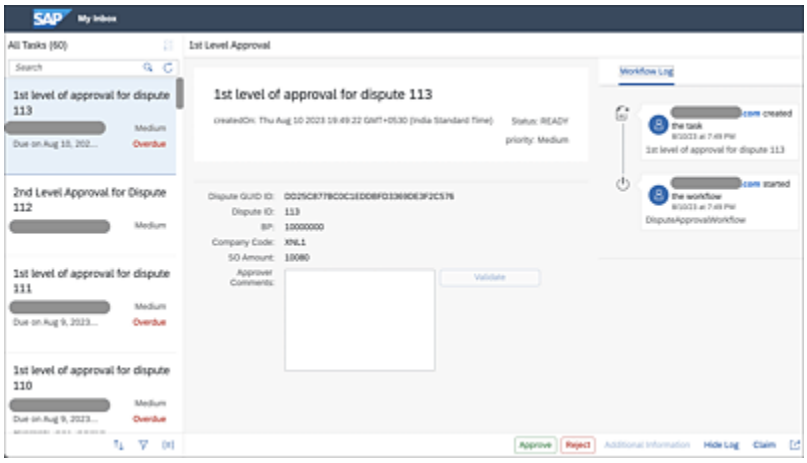


Figure 20.2 My Inbox in SAP BTP

20.1.2 Configure My Inbox in SAP Build Work Zone

Although the My Inbox application is accessible from the **Lobby** screen of SAP Build Process Automation, this shouldn’t be where an end user accesses My Inbox. The user should use the SAP Build Work Zone launchpad where a **My Inbox** tile appears just like for the on-premise version. Let’s see how this is achieved.

Make sure you have proper authorization as an SAP Build Work Zone launchpad content manager. The My Inbox application is available as part of the HTML5 application repository, and this can be used to configure your content in an SAP Build Work Zone launchpad site that you’ve created. Similar to creating any new tile in your site, you can create a new tile using the My Inbox application.

From the **Content Manager** • **Content Explorer** section of your SAP Build Work Zone site, select **My Inbox**, and then click **Add** to add it your contents, as shown in Figure 20.3.

Now the My Inbox app is available in your **Content Manager** screen, as shown in Figure 20.4.

From here onward, follow the standard steps to create a catalog group, and assign this app to the catalog.

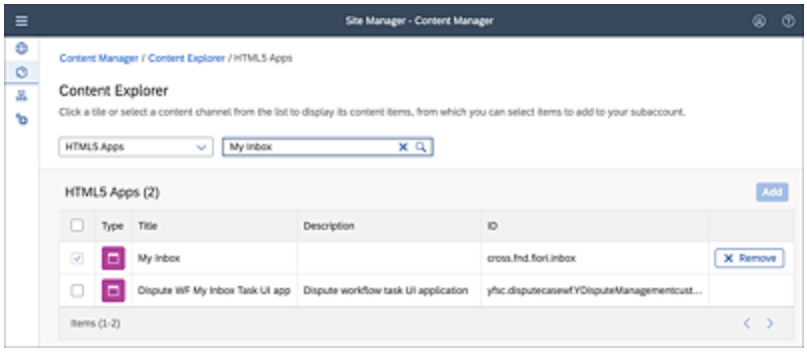


Figure 20.3 Adding the My Inbox App to the Contents of Your Site

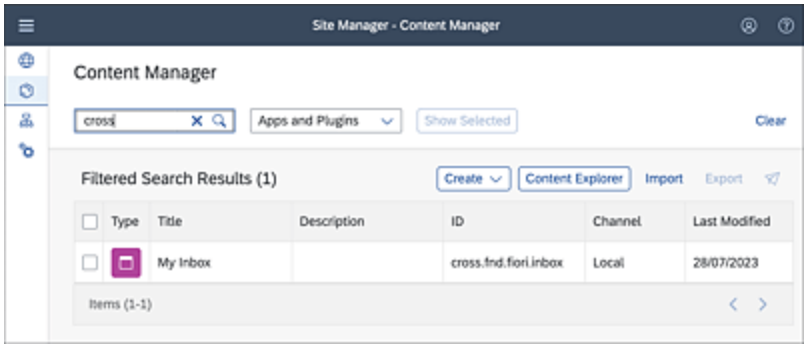


Figure 20.4 My Inbox App Now Available in Content Manager

Note

To create the catalogs, groups, spaces, and pages, refer to the standard documentation for SAP Build Work Zone launchpad configuration.

Later to a role and once this role is assigned to the end user, My Inbox will be available for the user in the launchpad. Here, the user can use substitutions from the **User Action** menu of the launchpad just as with the on-premise launchpad.

Figure 20.5 shows the My Inbox tile as it appears for the user in the SAP Build Work Zone launchpad with the count of task items assigned to the user.

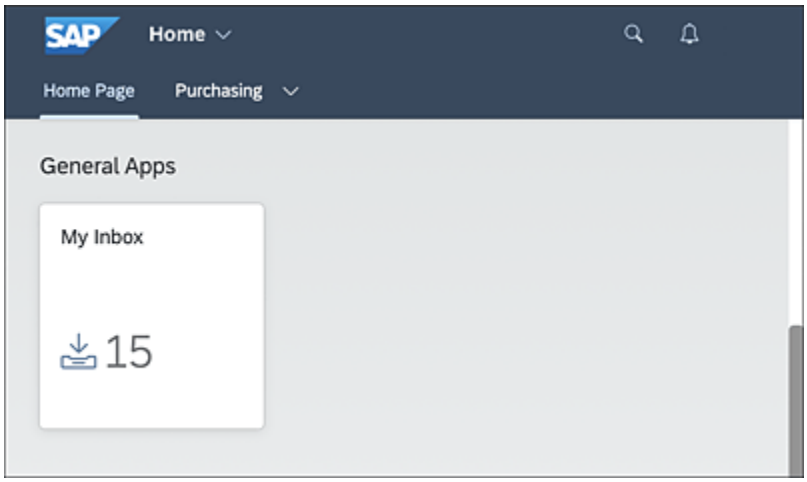


Figure 20.5 My Inbox Made Available in the Launchpad

To set substitutions, navigate to the **User** • **Settings** • **Manage My Substitutes**, as shown in Figure 20.6.

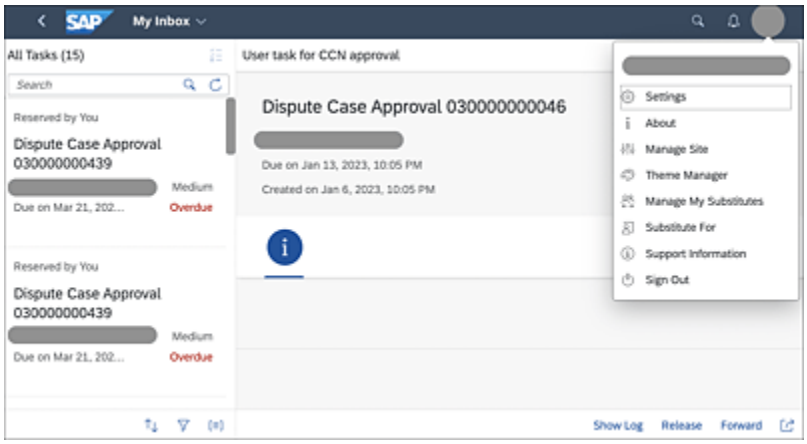


Figure 20.6 My Inbox User Action Menu Option: Manage My Substitutes

A substitute can be set in the Manage My Substitutes app by entering the SAP BTP user ID for a particular user and setting a time frame for this substitution to be in effect, as shown in Figure 20.7.

> **Note**
>
> The Manage My Substitutes app doesn’t provide the option to search and assign the user as compared to on-premise My Inbox. You need to either enter it manually or copy and paste the SAP BTP user ID of the user or from the user profile details.

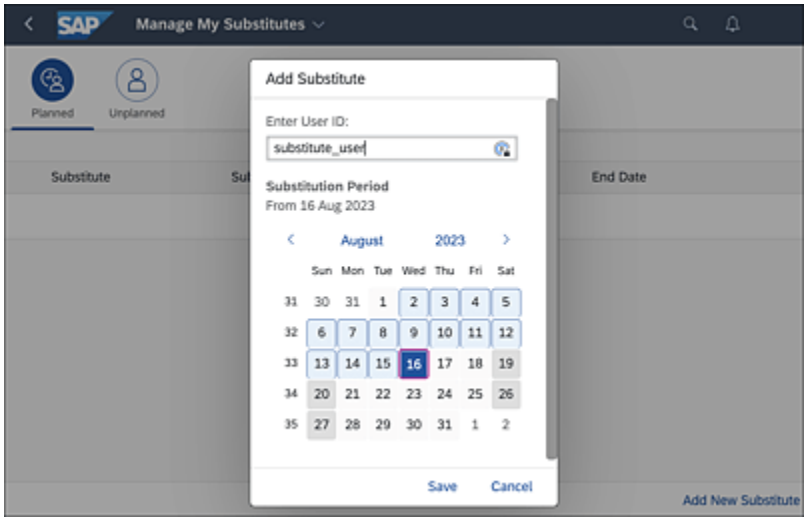


Figure 20.7 Adding a Substitute

You can also view whom you’re substituting for from the Substitute For app in the **User Action** menu.

In the My Inbox app, there are some important configuration parameters to be aware of. Most importantly, if you need to enable substitutions, you have to set the parameter `substitution` to `true`. Various parameters that you can set are shown in Figure 20.8.

> **Note**
>
> Unlike the on-premise version of My Inbox, there is no such option to set various visualizations for the My Inbox details page. You can use the approval form control in SAP Build Process Automation and either as a simple form or a custom SAPUI5 application in a workflow based on SAP Business Application Studio. One thing to ensure is that the UI modules you develop in SAP Business Application Studio are deployed with a managed AppRouter, instead of standalone AppRouter, for it to be used in the My Inbox app in the SAP Build Work Zone launchpad.

Under the **Navigation** tab, of the My Inbox app in the **Content Manager** screen, the following parameters can be set that affect the behavior of the app at runtime:

- **`listSize`**
  This takes a numeric value that represents the number of tasks to be listed in the My Inbox app.
- **`expertMode`**
  This parameter enables the expert view in My Inbox (`true`/`false`).
- **`userSearch`**
  This parameter enables user search for both substitutions and forward functionalities (`true`/`false`).
- **`Substitution`**
  This parameter is used to make the Manage My Substitutes app available for users (`true`/`false`).
- **`showAdditionalAttributes`**
  This parameter lets you use and display customer-specific

attributes related to business content (`true`/`false`).

- **`showLog`**
  This parameter enables the **Show/Hide Log** button (`true`/`false`).

- **`enablePaging`/`pageSize`**
  The `enablePaging` parameter enables pagination in the **Master** view (`true`/`false`), and `pageSize` sets the limit.

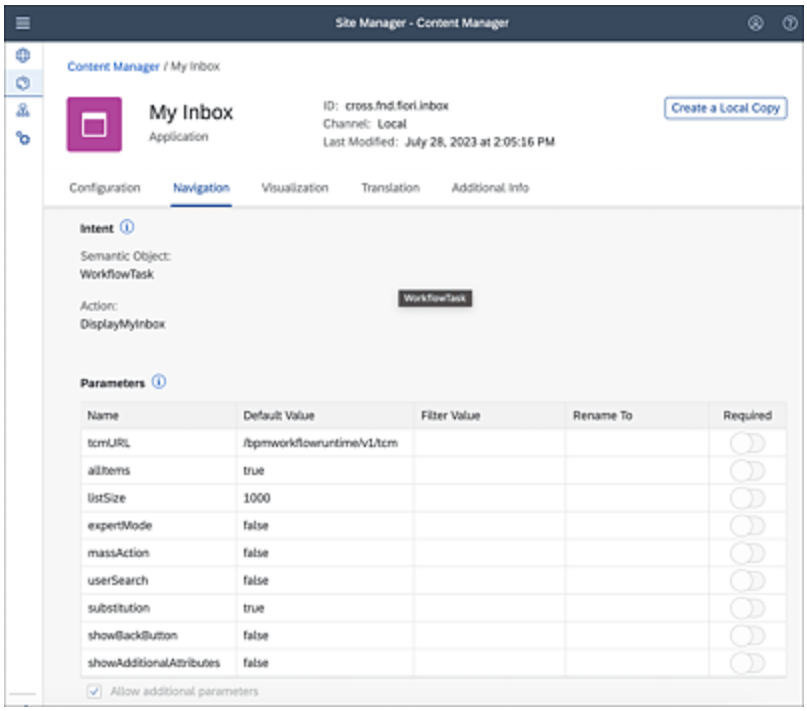


Figure 20.8 My Inbox Parameters

20.2 SAP Task Center

To get approval items into an inbox, SAP has two inbox applications:

- **My Inbox app in SAP S/4HANA**
  This app is the single source of all approval items generated from SAP S/4HANA standard or custom workflows.
- **My Inbox on SAP BTP**
  This out-of-the-box application comes with the SAP Build Process Automation service. All the workflow approvals generated from this service are shown to this inbox application.

As you can see, to approve, a user needs to open two inbox apps, which isn't a very friendly solution from a usability perspective. SAP solved this problem with the introduction of a new service in SAP BTP called SAP Task Center. The SAP Task Center service is a single point of entry for all kinds of approvals from SAP products.

As of now, the SAP Task Center service fully supports on-premise SAP S/4HANA; SAP S/4HANA Cloud, public edition; SAP S/4HANA Cloud, private edition; and SAP Build Process Automation. All approvals from SAP S/4HANA and SAP Build Process Automation will be pushed to one single place for approval, and approvers will see all approvals in one single place—SAP Task Center. This service also supports SAP products such as SAP Ariba, SAP Fieldglass, and so on. Table 20.1 shows the list of SAP products that support approval workflow integration with SAP Task Center. As of now, there is no way to integrate third-party product tasks into SAP Task Center. However, from Q4 2023, SAP has plans to expose the API to integrate third-party product tasks as well per the SAP road map of SAP Task Center.

Products	Support Type
SAP Ariba	No types of approvals
SAP Cloud for Customer	No types of approvals
SAP Concur	No types of approvals
SAP Fieldglass	No types of approvals
SAP Marketing Cloud	No types of approvals
SAP Build Process Automation	All types of approvals
SAP S/4HANA	All types of approvals
SAP S/4HANA Cloud, public edition	All types of approvals
SAP S/4HANA Cloud, private edition	All types of approvals
SAP SuccessFactors	No types of approvals

Table 20.1 Product Support for the Approval Workflow in SAP Task Center

SAP Task Center by default isn't available as a service in SAP BTP subaccounts. You need to add this service to your subaccount and then configure the service to make it work. This

service runs from both web and mobile. It's recommended to configure the service via a booster. If you use a booster, during the creation of the service, all the destinations (e.g., in SAP Build Process Automation; SAP S/4HANA; SAP Build Work Zone, standard edition; etc.) will be created in your subaccount. You can add this service to your SAP Build Work Zone, standard edition launchpad to make it available to users based on their authorization. The SAP Mobile Start app also supports SAP Task Center.

When you run the booster for SAP Task Center to setup, you'll have three options:

- Set up an account for SAP Task Center with SAP Build Work Zone, standard edition.
- Set up an account for SAP Task Center with SAP Build Work Zone, advanced edition.
- Set up an account for SAP Task Center with SAP SuccessFactors Work Zone.

We're setting up here with SAP Build Work Zone, standard edition.

Now, let's discuss how you can integrate SAP Task Center with SAP Build Process Automation. Because we choose to use SAP Build Work Zone, standard edition, SAP Task Center will be available in our launchpad.

As we followed the booster approach to create SAP Task Center, a destination called "SAPBuildPA" is created. Now we need to copy that destination and create another destination that will push workflow approval tasks to SAP Task Center. It's assumed that you've already configured Identity Authentication via **Security** • **Trust Configuration** in SAP BTP.

Let's look at the steps to create a destination for SAP Build Process Automation to establish a connection with SAP Task Center when SAP Task Center and SAP Build Process Automation services are running from the same subaccount.

For creating a destination, you can either navigate to the SAP BTP cockpit and choose **Subaccount** • **Destinations**, or from the SAP Build Process automation **Lobby**, choose **Settings** • **Destinations.** Once here, click on **Open** in the SAP BTP cockpit and then add the information shown in Table 20.2.

Property	Instruction
Name	Name of the destination per the customer's naming convention
Type	**HTTP**
Description	(Optional)
URL	Endpoint URL of key created from taking the instance of SAP Buil Automation and then appending "/internal/workflow/rest/v1" to th
Proxy Type	**Internet**
Authentication	**OAuth2SAMLBearerAssertion**
Audience	"<https://<subaccount>.authentication.eu<server_number>.har
AuthnContextClassRef	"urn:oasis:names:tc:SAML:2.0:ac:classes:PreviousSession"
Client Key	The `clientid` from SAP Build Process Automation key.

Property	Instruction
Token Service URL Type	**Dedicated**
Token Service URL	URL from the SAP Build Process Automation key with appended “
Token Service User	`clientid` from the SAP Build Process Automation key
Token Service Password	`clientSecret` from the SAP Build Process Automation key

Table 20.2 Destination Properties to Be Configured For SAP Task Center

Now add additional properties with the details listed in Table 20.3.

Property	Instruction
nameIdFormat	Security Assertion Markup Language (SAML) name identifier formats such as user ID, email address, and so on. Value will be like the following: “urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress”
tc.enabled	**True** (this property will enable the SAP Build Process Automation destination to be used in SAP Task Center)
tc.provider_type	**SPA** (to enable the **Filter** tab in SAP Task Center for SAP Build Process Automation approval tasks)
tc.ui.group and **tc.ui.group.[language_code]**	Optional property for language groups
tc.ui.label and **tc.ui.label.[language_code]**	Optional property used for adding language-specific labels

Table 20.3 Additional Properties to Be Configured for SAP Task Center Destination

Once the destination is created, it will look like Figure 20.9.

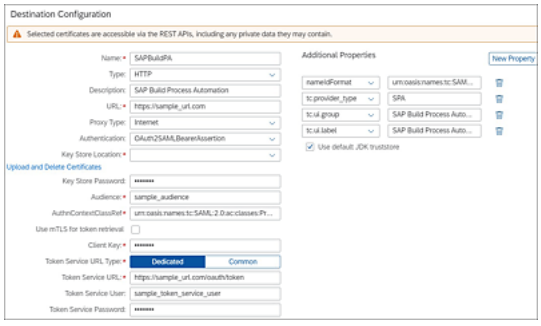


Figure 20.9 Destination Setup for SAP Build Process Automation with SAP Task Center

You can also set up the destination with a remote subaccount (i.e., if your SAP Task Center is running from a separate subaccount than SAP Business Process Automation). In that case, you need to clone destination “SAPBuildPA_rem”, set up principal propagation between two subaccounts, and set up user propagation between Cloud Foundry applications.

Some restrictions while using SAP Task Center for SAP Build Process Automation approval tasks are as follows:

- Don’t create two destinations of SAP Build Process Automation for one SAP Task Center because this can cause duplication of tasks.
- The default Identity Authentication service, which is provided by SAP, won’t work, so a customer-specific identity provider must be set up in the subaccount.
- SAP Task Center will show the task, but task processing will take place inside the SAP Build Process Automation UI inbox, which means the SAP Build Process Automation service inbox app is also needed.

20.3 Summary

Where there is a workflow, no matter whether it’s on-premise or in the cloud, you can anticipate an intervention task. A visualization must be made available for the user with a proper and intuitive UI. This is achieved using the My Inbox app, which is a kind of universal app for approval, as well as other user-intervention scenarios. In this chapter, you saw the options for on-premise workflows and SAP BTP–based workflows, including how to configure them and use them to their full potential in both flavors. In addition, you learned about SAP Task Center, which is SAP’s universal inbox connecting not just an SAP S/4HANA system but other SAP on-premises and cloud products.

A The Authors

Sabyasachi Dutta is a senior SAP application architect and consultant with more than 17 years of experience in the SAP technical domain. At IBM, he is a product lead for ABAP. He served in various technical roles in end-to-end SAP implementation projects for global clients from a number of different industries. Sabyasachi has played a key role in building the SAP HANA technical competency at CIC India. He is also highly involved in developing different technical assets, providing training, and establishing process, methods, and tools for SAP technical delivery. To do so, he uses his knowledge of the code review process, ABAP, SAP BTP, the ABAP programming model for SAP Fiori, the ABAP RESTful application programming model, in-app extensions, workflows, DevOps, and generative AI. Sabyasachi can be reached at *www.linkedin.com/in/sabyasachi-dutta-s4arch*.

Nilay Ghosh is a globally acknowledged technical leader for the SAP practice at IBM Consulting. He has more than 20 years of experience working with SAP's technology stack, including SAP ERP, SAP S/4HANA, SAP BTP, and other on-premise and cloud products. He has had several technical leadership roles at IBM for both global client projects and internal strategic programs. He also set up the SAP eSOA, SAP UX/mobile, and SAP BTP practices at IBM's largest global delivery center in India. In addition to being a lead architect for the global SAP S/4HANA transformation program, Nilay is currently the chief architect for all the intelligent workflows developed under the strategic Evolution Program between SAP and IBM. A core member of IBM's SAP Global CTO Network, and a frequent speaker with a few patents to his name, he can be reached at *www.linkedin.com/in/nilay-ghosh/*.

Kousik Goon is a globally acknowledged technical leader in the SAP practice at IBM Consulting. He has more than 18 years of experience working in SAP's technology stack, including SAP ERP, SAP S/4HANA, SAP BTP, and other on-premise and cloud products. He has held several technical leadership roles in IBM—both for global client projects and internal strategic programs. He was instrumental in setting up the SAP UX/mobile and SAP BTP practices at IBM's largest global delivery center in India. In addition to his role as an integration/extension architect for a global SAP S/4HANA transformation program, Kousik is the key SME at IBM Consulting for SAP S/4HANA extensions and integration. A core member of IBM's SAP Global CTO Network, and a frequent speaker, he can be reached at *www.linkedin.com/in/kousikgoon*.

Sandip Jana is a global technical leader for the SAP practice area at IBM Consulting. He has more than 22 years of experience with enterprise architecture, system landscape design, and simplifications for different SAP technology platforms, both on-premise and in the cloud. He has led different IBM strategic offerings helping customers to embark on their SAP S/4HANA journey. He has been the lead architect for global SAP transformation projects. Sandip is currently leading the SAP technical practice at the IBM global delivery center in India and is responsible for building capabilities in new technical areas. He leads IBM's SAP HANA impact assessment that guides customers on their SAP S/4HANA transformation journey. He also leads IBM's global SAP S/4HANA transformation offering: IBM Rapid Move. A core member of IBM's SAP Global CTO Network, and a frequent speaker, he can be reached at *www.linkedin.com/in/sandip-jana*.

Arindam Mukherjee is a senior technical architect and consultant who has worked for IBM for more than 17 years, specializing in SAP Business Workflow, ABAP, ABAP on SAP HANA, OData, and SAP S/4HANA embedded analytics. Arindam is an established subject matter expert in the SAP Business Workflow space, implementing workflow solutions

for several global SAP customers as well as driving the growth of workflow competency and the knowledge base content for classical workflows at IBM. Arindam has also expanded his skills to include flexible workflows with SAP S/4HANA, SAP Fiori, and OData integration and has led multiple SAP S/4HANA implementations by engaging in end-to-end solution design, development, and support.

Srinivas Rao is an application architect for the SAP practice at IBM with more than 16 years of consulting experience. He is passionate about clean core solutions and has extensive expertise with them. He has the SAP certifications for SAP Cloud Platform extensions and SAP Cloud Platform integration. He has extensive experience in delivering complex SAP S/4HANA implementations projects with varied experience in building and delivering solutions. He is also passionate about automation and was the winner of one of the SAP Workflow Management Hack2Build events. He can be reached at *www.linkedin.com/in/srinivas-rao-58275311/*.

Yogeendar Rao is a senior SAP S/4HANA technical consultant working for IBM India. He has more than 10 years of global SAP consulting experience, with expertise in ABAP on SAP HANA, SAP Business Workflow, flexible workflows, OData, and SAP Conversational AI. He has worked on enterprise digital transformations and delivered technical use cases for global SAP customers using his proficiency in ABAP on SAP HANA and SAP Business Workflow. He has mastered flexible workflows and their integration with SAP Fiori apps, doing work on solutioning, designing, and extending the flexible workflows in many SAP S/4HANA implementations. Yogeendar can be reached at *www.linkedin.com/in/yogeendar-rao/*.

Yogesh Sane is a subject matter expert and a senior consultant at IBM with more than 20 years of experience in

implementing SAP solutions, including SAP R/3, SAP ERP, SAP S/4HANA, and SAP BTP for clients in multiple regions and industries. Yogesh leads the master data governance delivery center at IBM India and has implemented SAP Master Data Governance for a number of clients. He set up an internal training program on SAP Master Data Governance and has trained hundreds of SAP consultants. He has also set up a center of excellence to perform design reviews and troubleshoot implementation issues for SAP MDG implementation projects delivered by IBM. Yogesh has created several assets to benefit clients by harnessing the experience and knowledge of the entire SAP MDG practice.

Naveen Veshala is a senior SAP application architect and consultant with more than 20 years of experience in the SAP technical domain. He was worked for IBM for close to 18 years, specializing in SAP S/4HANA, SAP Gateway, SAP Fiori, flexible workflows, ABAP, and SAP ERP. Naveen is an established subject matter expert within IBM’s SAP practice where his main focus is SAP S/4HANA, flexible workflows, and OData. He also contributed to SAP Community and published blogs on flexible workflows. He has led multiple SAP S/4HANA implementations by engaging in end-to-end

solution design, development, and support. Naveen can be reached at *www.linkedin.com/in/naveenveshala*.

Kiran Viswanathan is a technical architect and senior consultant with more than 21 years of extensive experience in SAP user experience, mobility, SAP BTP, orchestration, and integration. He has experience as a user experience and extensibility solution consultant and as a technical delivery manager for multiple clients across numerous industries for their SAP S/4HANA implementations. He also leads and manages the SAP UX, mobility, and SAP BTP practice area for the IBM India global delivery center. Additionally, he is an architect and realization lead in IBM's strategic Evolution Program for developing intelligent workflows for several industry use cases. He can be reached at *www.linkedin.com/in/kiran-viswanathan/*.

Index

A ⇑

B ⇑

C ⇑

D ⇑

E ⇑

F ⇑

G ⇑

I ⇑

J ⇑

K ⇑

L ⇑

M ⇑

N ⇑

O ⇑

P ⇑

Q ⇑

R ⇑

S ⇑

T ⇑

U ⇑

V ⇑

W ⇑

WS54300007, WS60800059, and WS60800068 [→ Section 13.2]

WS54400001 and WS54300005 [→ Section 13.2]

WS60800086 [→ Section 13.1] [→ Section 13.2]

WS60800095 and WS72100006 [→ Section 13.2]

WS75700027 [→ Section 13.3]

WS75700040 [→ Section 13.3]

Workflow wizard [→ Section 4.1]

Service Pages

The following sections contain notes on how you can contact us. In addition, you are provided with further recommendations on the customization of the screen layout for your e-book.

Praise and Criticism

We hope that you enjoyed reading this book. If it met your expectations, please do recommend it. If you think there is room for improvement, please get in touch with the editor of the book: *Meagan White*. We welcome every suggestion for improvement but, of course, also any praise! You can also share your reading experience via Twitter, Facebook, or email.

Technical Issues

If you experience technical issues with your e-book or e-book account at SAP PRESS, please feel free to contact our reader service: *support@rheinwerk-publishing.com*.

Please note, however, that issues regarding the screen presentation of the book content are usually not caused by errors in the e-book document. Because nearly every

reading device (computer, tablet, smartphone, e-book reader) interprets the EPUB or Mobi file format differently, it is unfortunately impossible to set up the e-book document in such a way that meets the requirements of all use cases.

In addition, not all reading devices provide the same text presentation functions and not all functions work properly. Finally, you as the user also define with your settings how the book content is displayed on the screen.

The EPUB format, as currently provided and handled by the device manufacturers, is actually primarily suitable for the display of mere text documents, such as novels. Difficulties arise as soon as technical text contains figures, tables, footnotes, marginal notes, or programming code. For more information, please refer to the section Notes on the Screen Presentation and the following section.

Should none of the recommended settings satisfy your layout requirements, we recommend that you use the PDF version of the book, which is available for download in your online library.

Recommendations for Screen Presentation and Navigation

We recommend using a sans-serif **font**, such as Arial or Seravek, and a low font size of approx. 30–40% in portrait format and 20–30% in landscape format. The background shouldn’t be too bright.

Make use of the **hyphenation** option. If it doesn't work properly, align the text to the left margin. Otherwise, justify the text.

To perform **searches** in the e-book, the index of the book will reliably guide you to the really relevant pages of the book. If the index doesn't help, you can use the search function of your reading device.

Since it is available as a double-page spread in landscape format, the **table of contents** we've included probably gives a better overview of the content and the structure of the book than the corresponding function of your reading device. To enable you to easily open the table of contents anytime, it has been included as a separate entry in the device-generated table of contents.

If you want to **zoom in on a figure**, tap the respective figure **once**. By tapping once again, you return to the previous screen. If you tap twice (on the iPad), the figure is displayed in the original size and then has to be zoomed in to the desired size. If you tap once, the figure is directly zoomed in and displayed with a higher resolution.

For books that contain **programming code**, please note that the code lines may be wrapped incorrectly or displayed incompletely as of a certain font size. In case of doubt, please reduce the font size.

About Us and Our Program

The website *https://www.sap-press.com* provides detailed and first-hand information on our current publishing

program. Here, you can also easily order all of our books and e-books. Information on Rheinwerk Publishing Inc. and additional contact options can also be found at *https://www.sap-press.com*.

Legal Notes

This section contains the detailed and legally binding usage conditions for this e-book.

Copyright Note

Your Rights as a User

You are entitled to use this e-book for personal purposes only. In particular, you may print the e-book for personal use or copy it as long as you store this copy on a device that is solely and personally used by yourself. You are not entitled to any other usage or exploitation.

In particular, it is not permitted to forward electronic or printed copies to third parties. Furthermore, it is not permitted to distribute the e-book on the internet, in intranets, or in any other way or make it available to third

parties. Any public exhibition, other publication, or any reproduction of the e-book beyond personal use are expressly prohibited. The aforementioned does not only apply to the e-book in its entirety but also to parts thereof (e.g., charts, pictures, tables, sections of text).

Copyright notes, brands, and other legal reservations as well as the digital watermark may not be removed from the e-book.

Digital Watermark

This e-book copy contains a **digital watermark**, a signature that indicates which person may use this copy.

If you, dear reader, are not this person, you are violating the copyright. So please refrain from using this e-book and inform us about this violation. A brief email to *info@rheinwerk-publishing.com* is sufficient. Thank you!

Trademarks

The common names, trade names, descriptions of goods, and so on used in this publication may be trademarks without special identification and subject to legal regulations as such.

All of the screenshots and graphics reproduced in this book are subject to copyright © SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany. SAP, ABAP, ASAP, Concur Hipmunk, Duet, Duet Enterprise, ExpenseIt, SAP ActiveAttention, SAP Adaptive Server Enterprise, SAP

Limitation of Liability

The Document Archive

The Document Archive contains all figures, tables, and footnotes, if any, for your convenience.

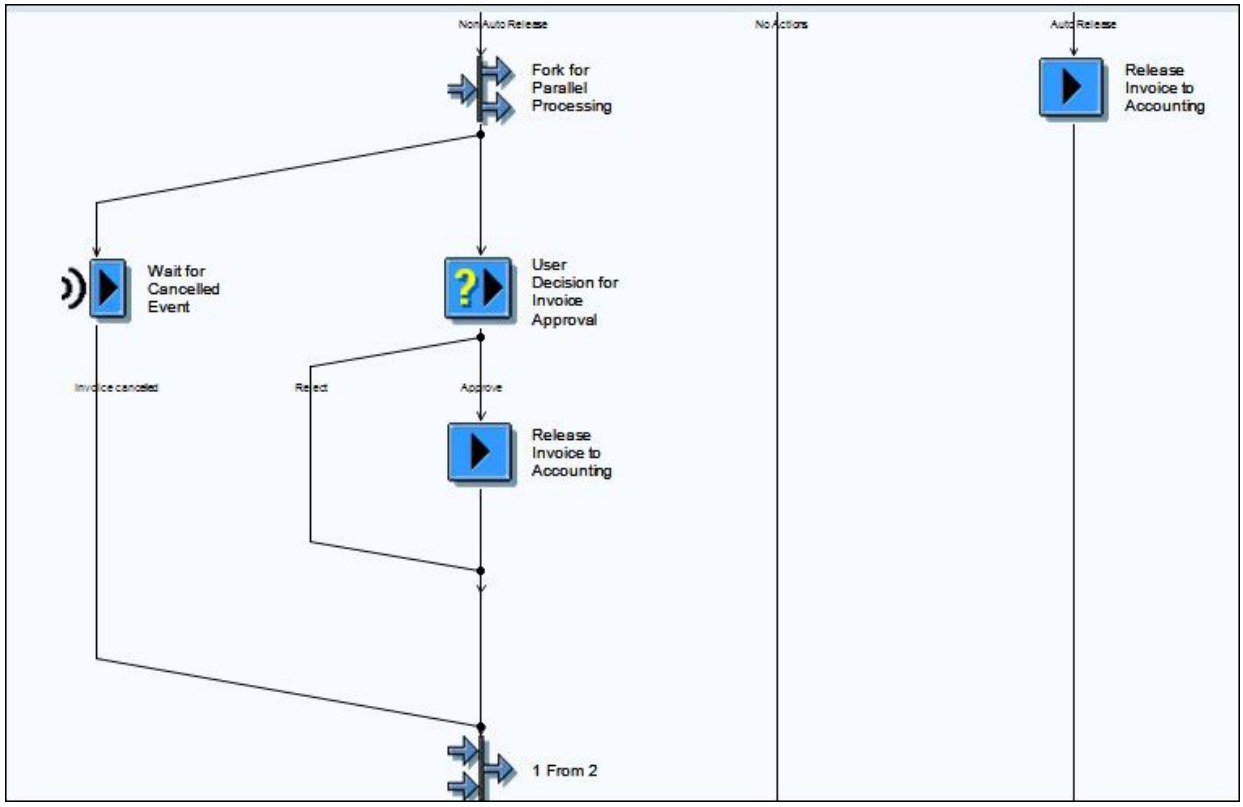


Figure 1.1 Sample Workflow Definition Using the Classical Workflow Approach

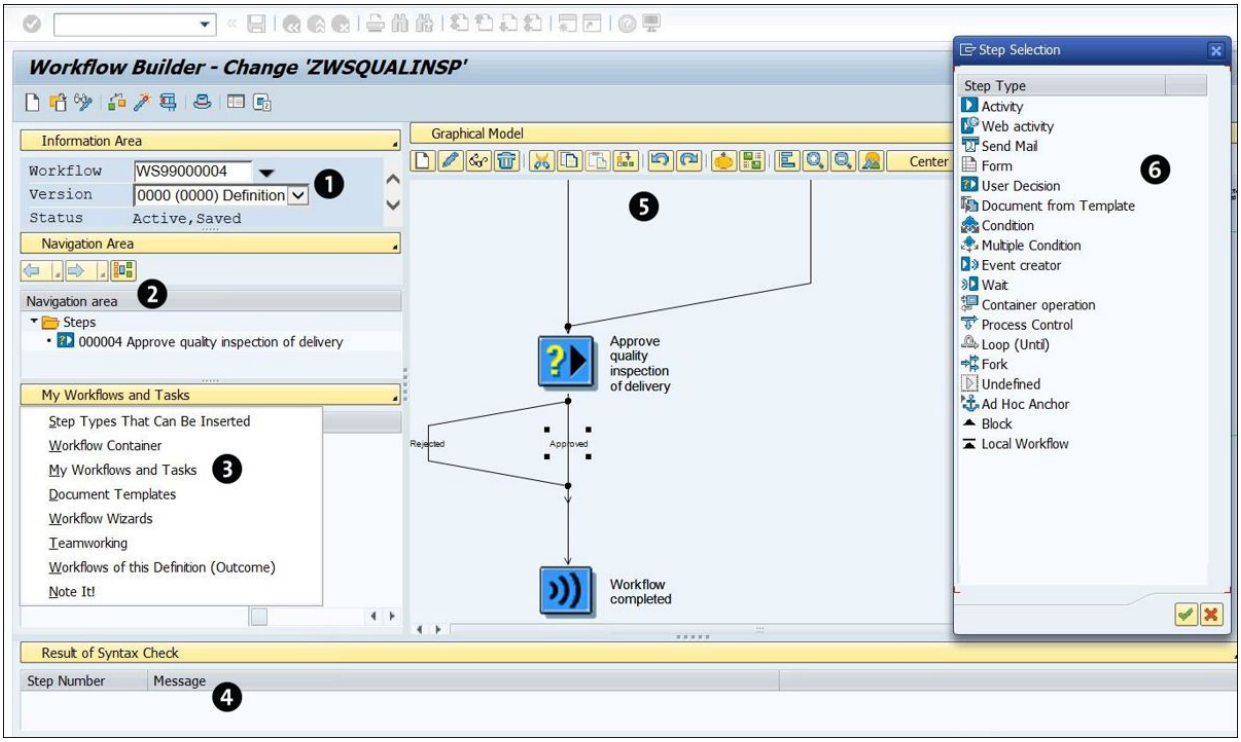


Figure 1.2 View of Workflow Builder with Different Development and Navigation Options

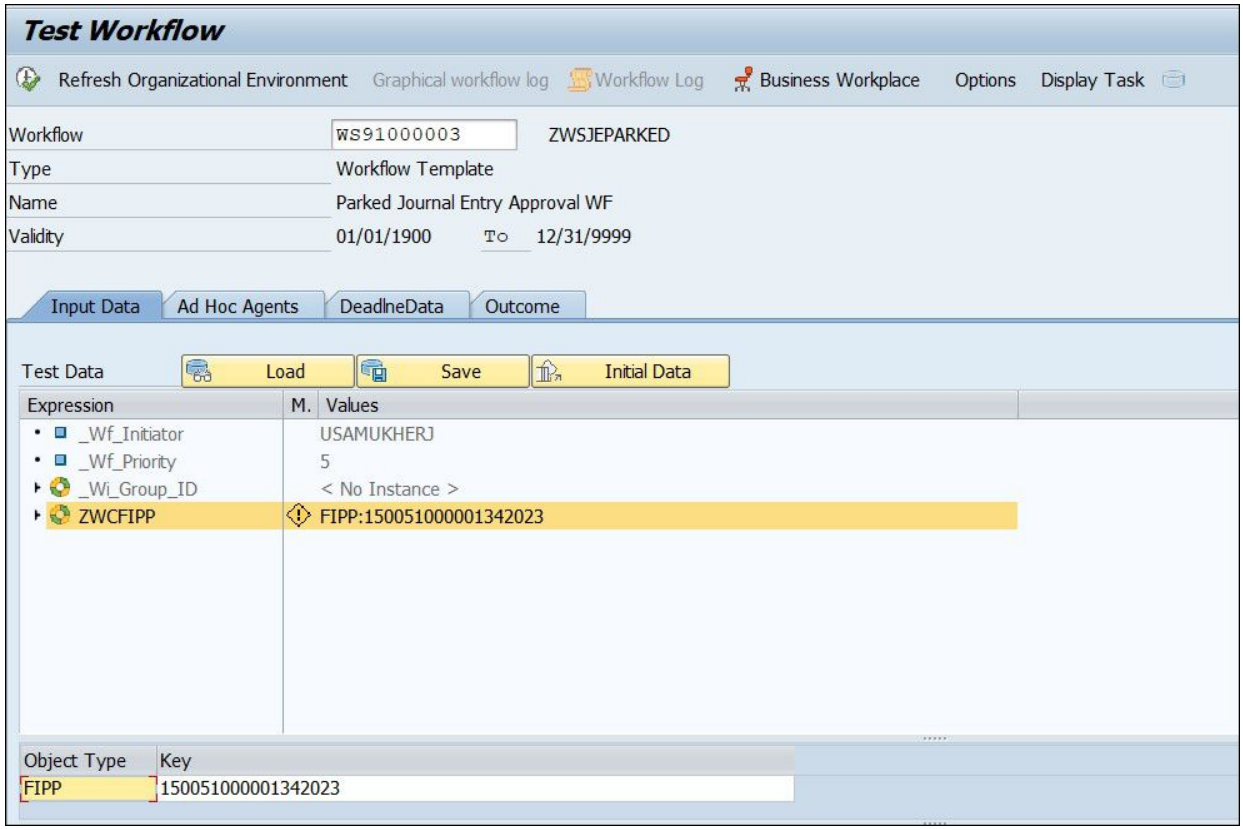


Figure 1.3 Example of Workflow Single Testing via Transaction SWUS

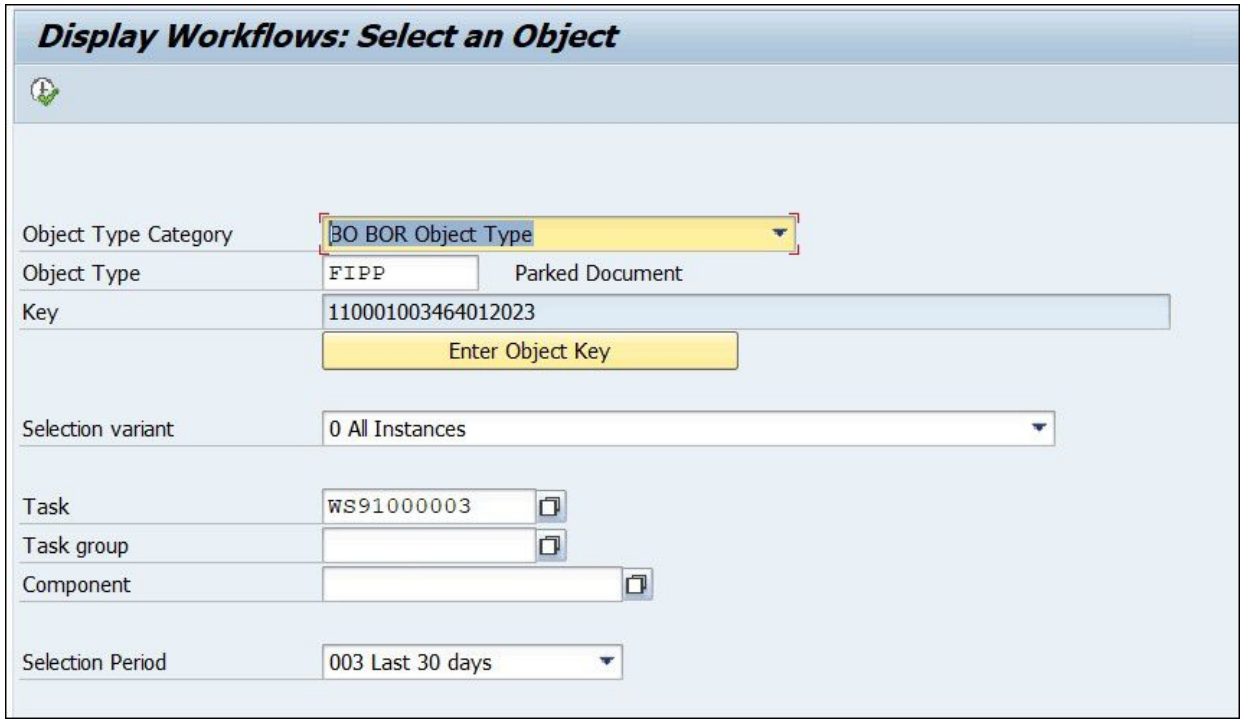


Figure 1.4 Sample Selection Criteria for Parked Journal Entry Approval Workflow Search Based on Document Key

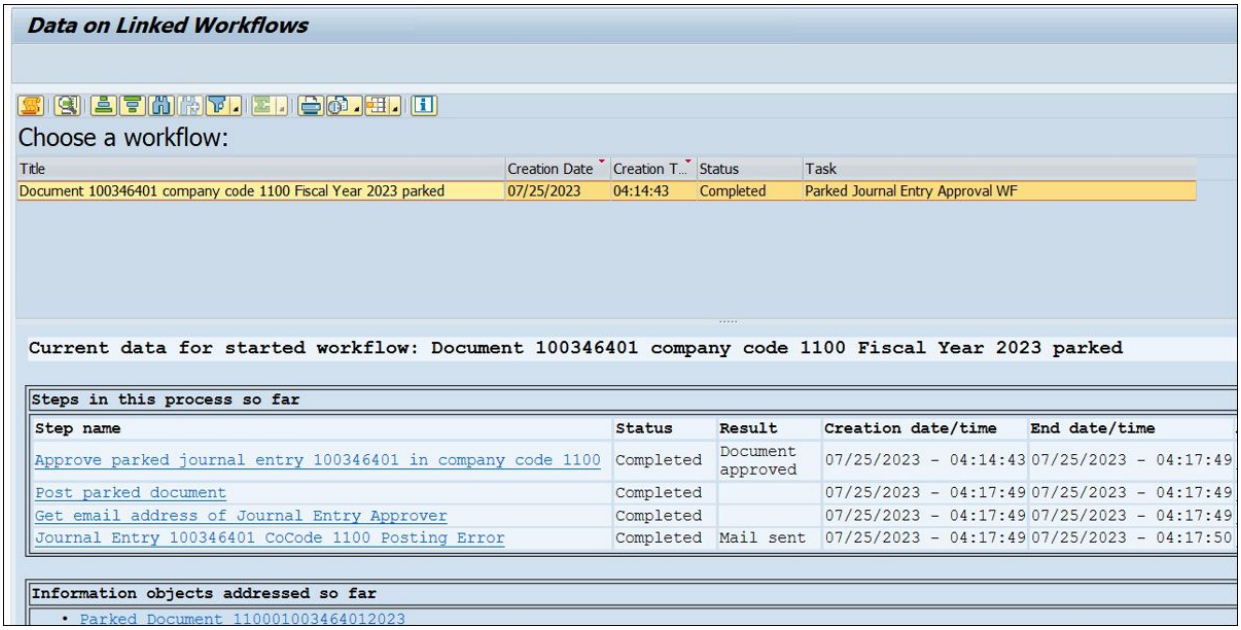


Figure 1.5 Sample Output of Transaction SWI6 with Details of Workflow Triggered for a Parked Journal Entry Document

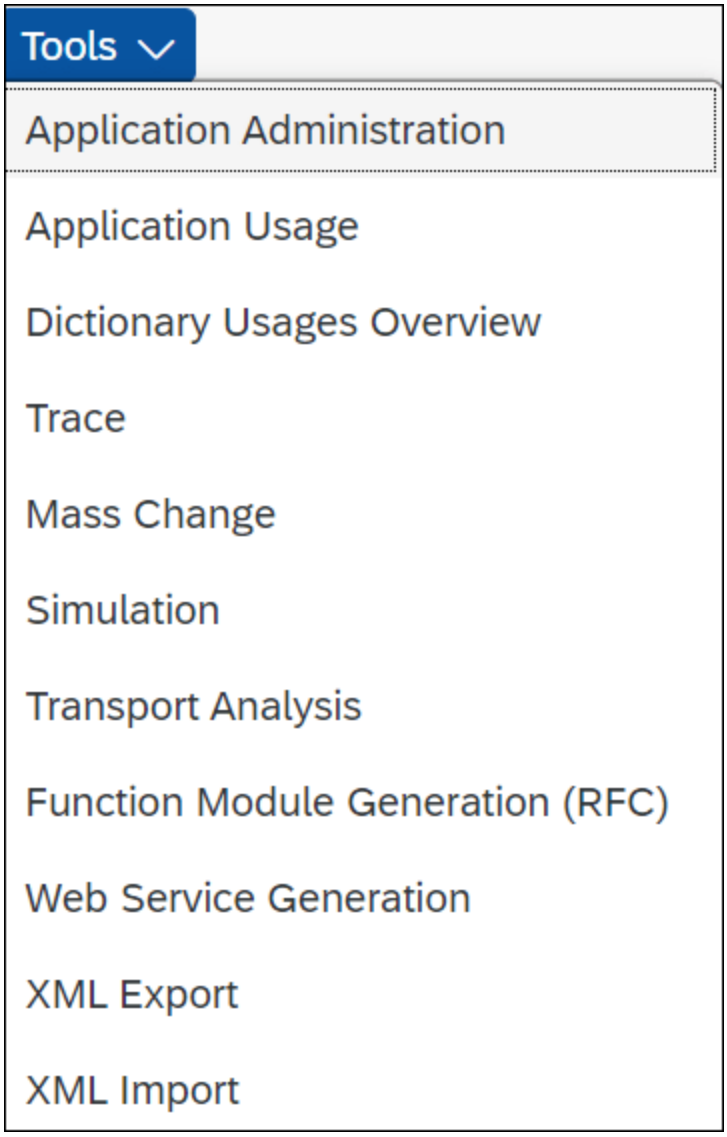


Figure 1.6 Development Tools

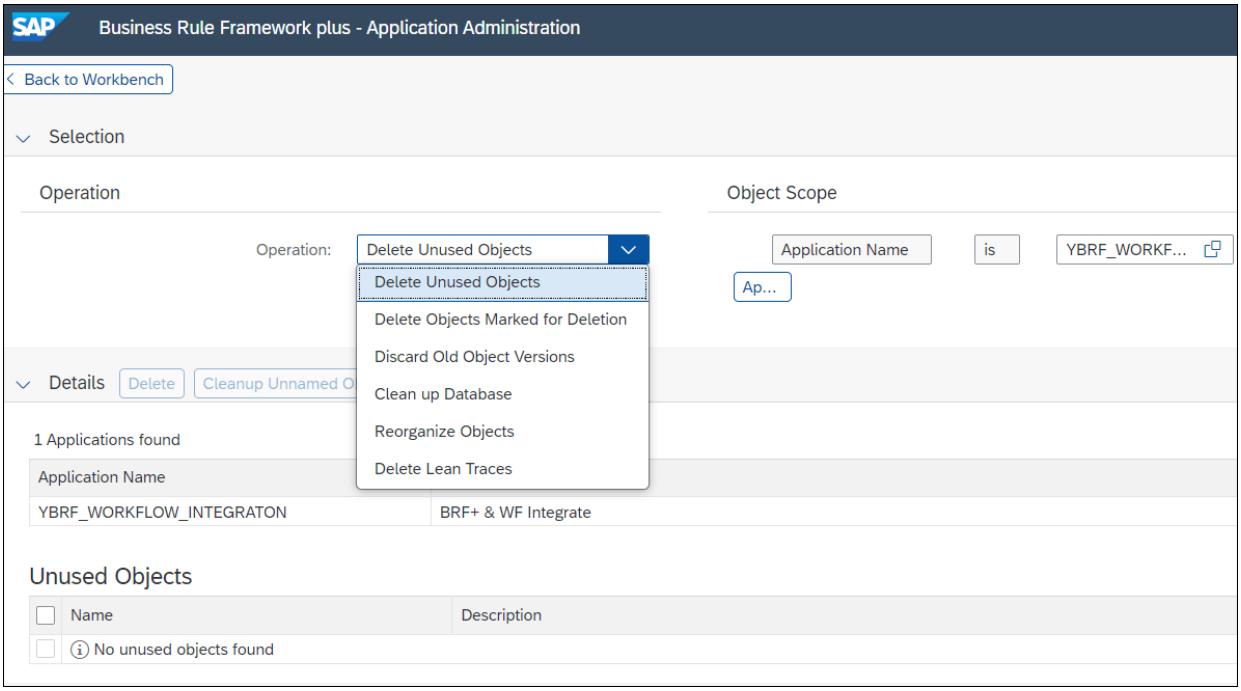


Figure 1.7 Operations Possible via the Application Administration Tool

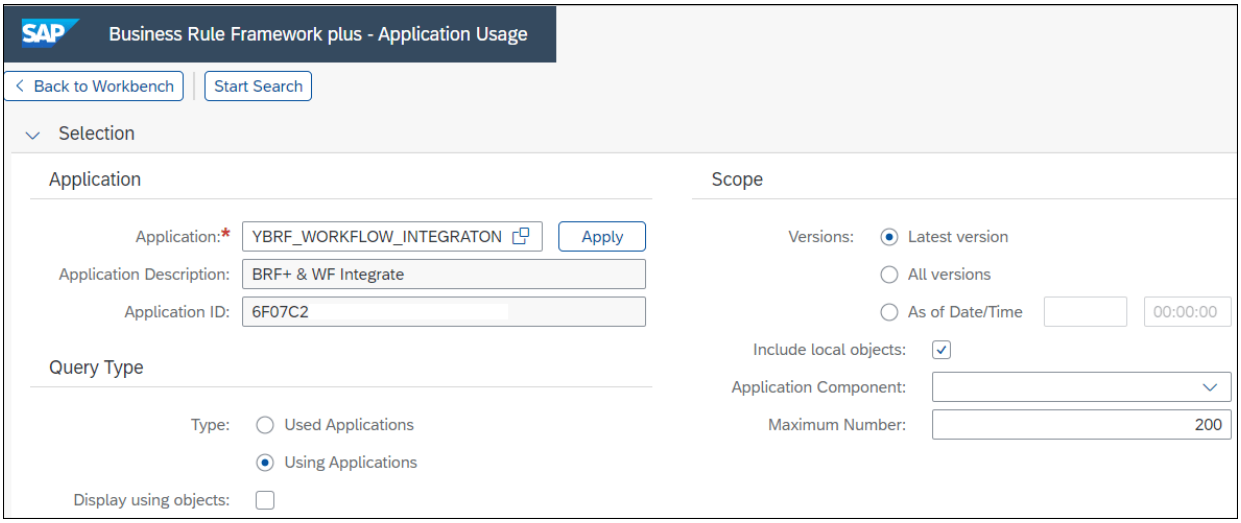


Figure 1.8 Search Screen for the Application Usage Tool

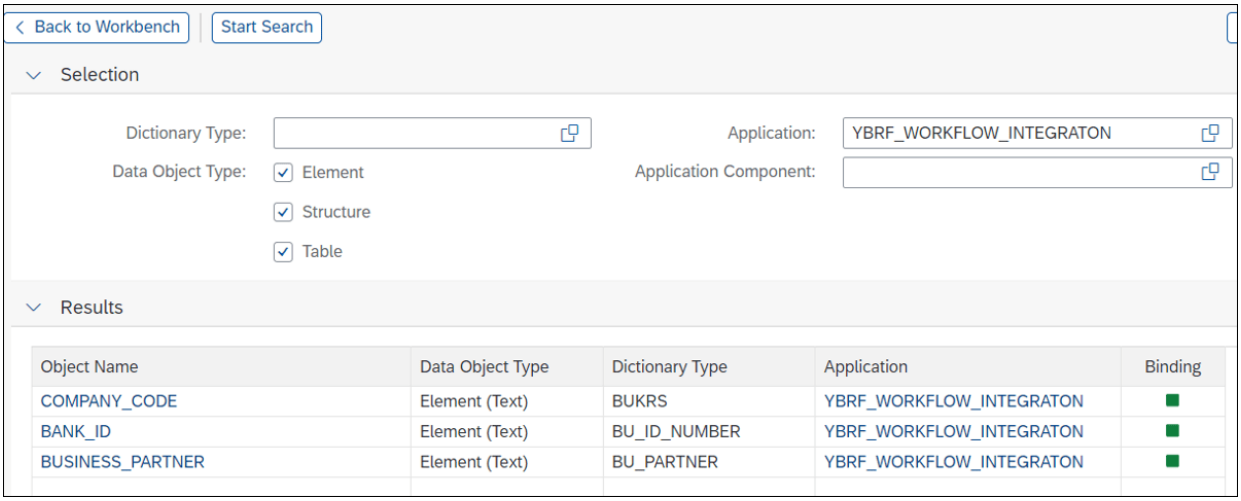


Figure 1.9 Dictionary Usages Tool

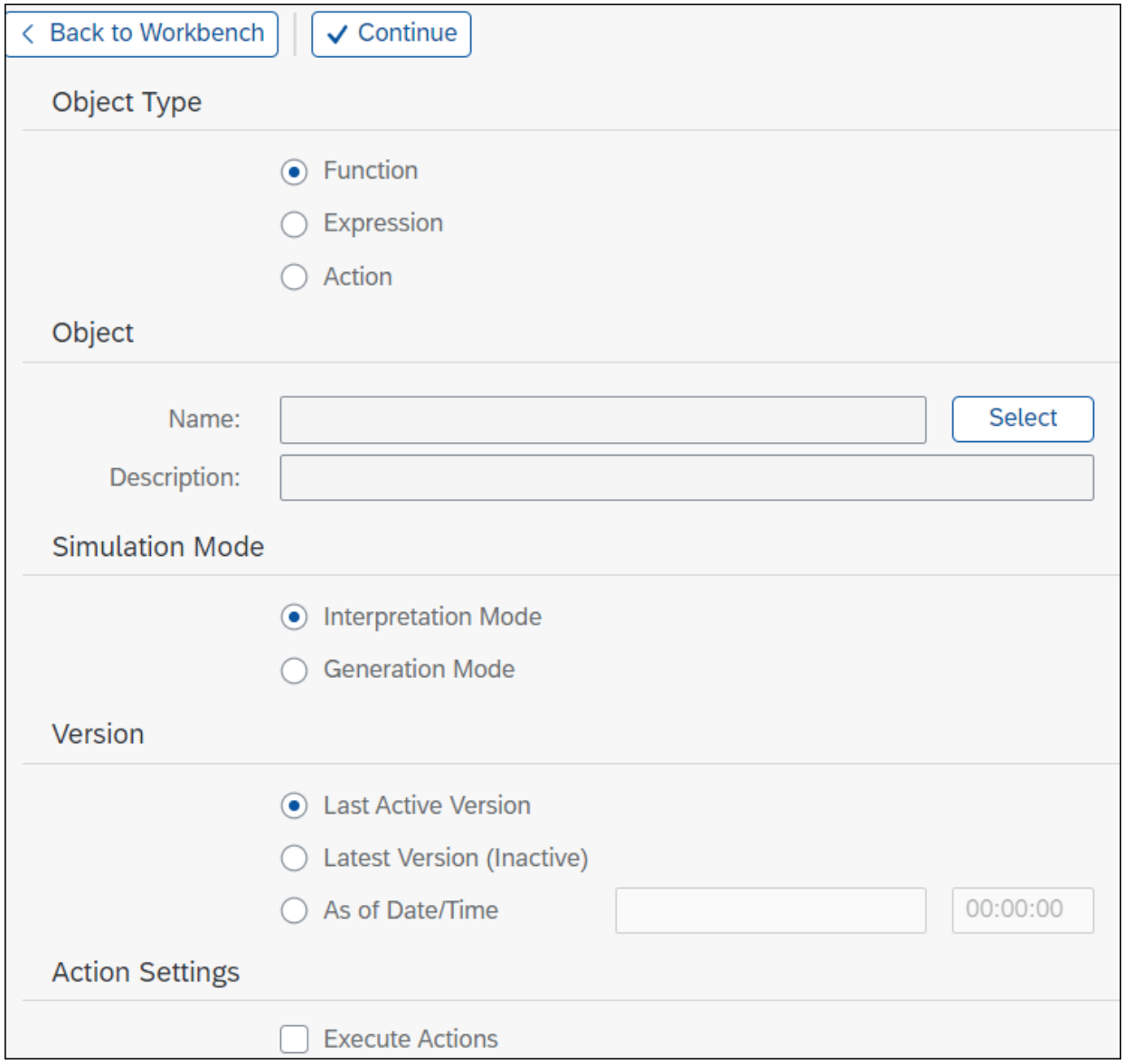


Figure 1.10 Selection Screen of Simulation Tool

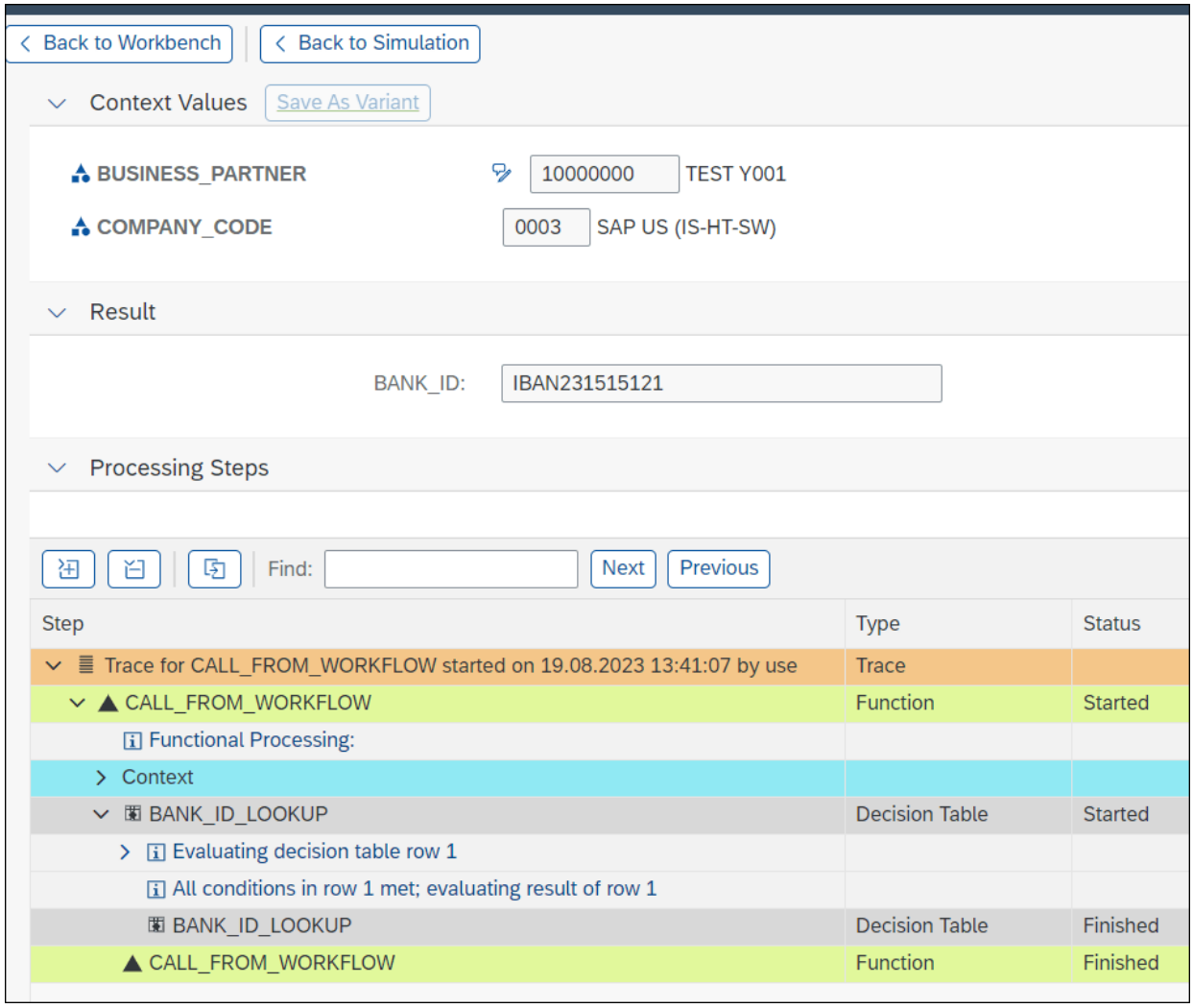


Figure 1.11 Simulation Outcome

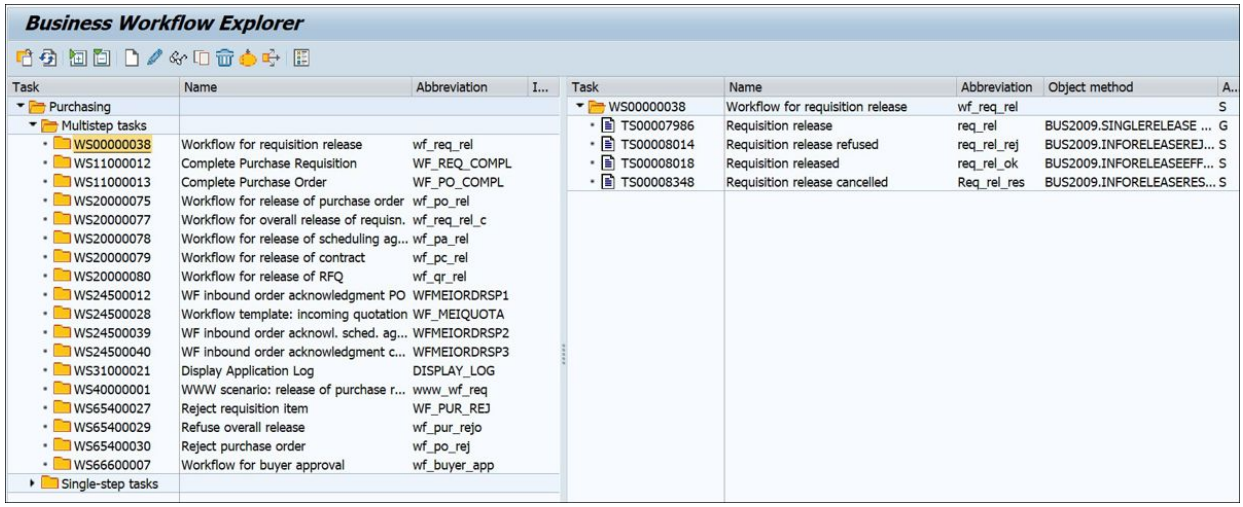


Figure 2.1 List of Workflow Templates in Business Workflow Explorer

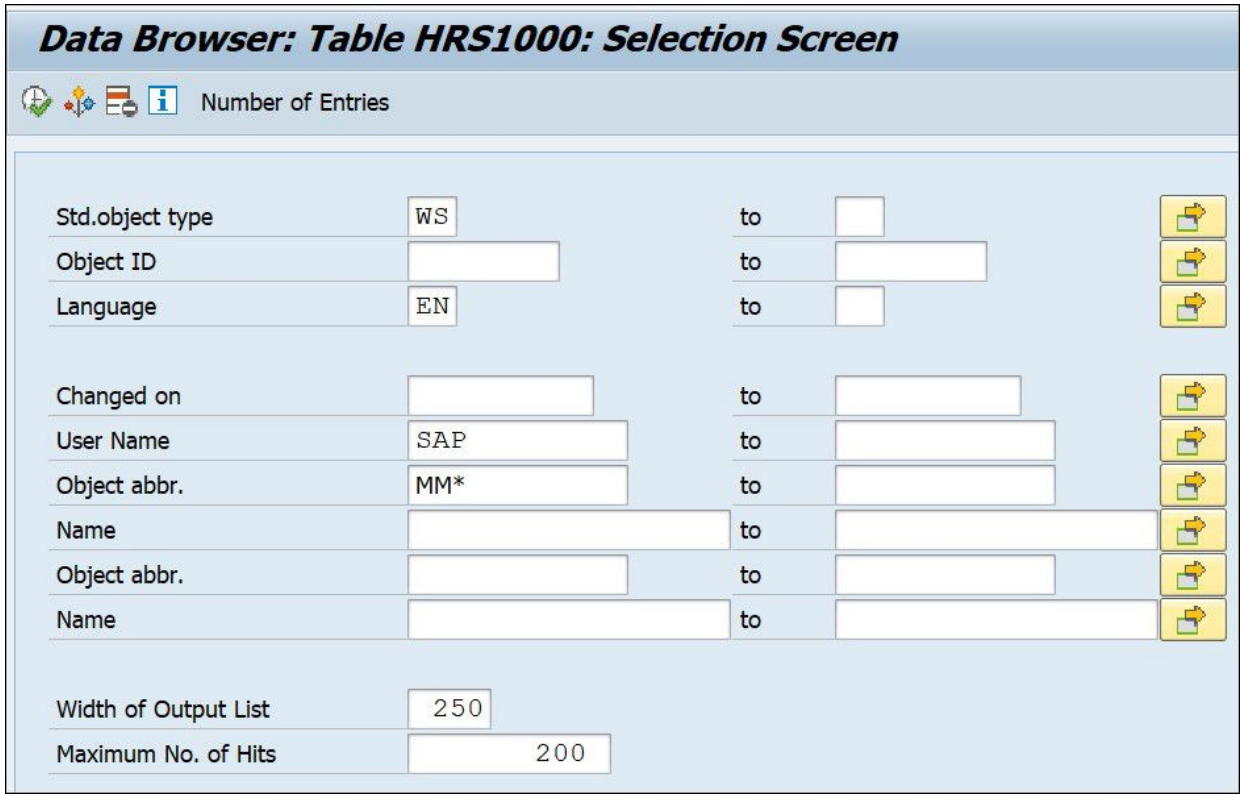


Figure 2.2 Selection Criteria to Fetch All the SAP Standard Workflow Templates in the Material Management Area

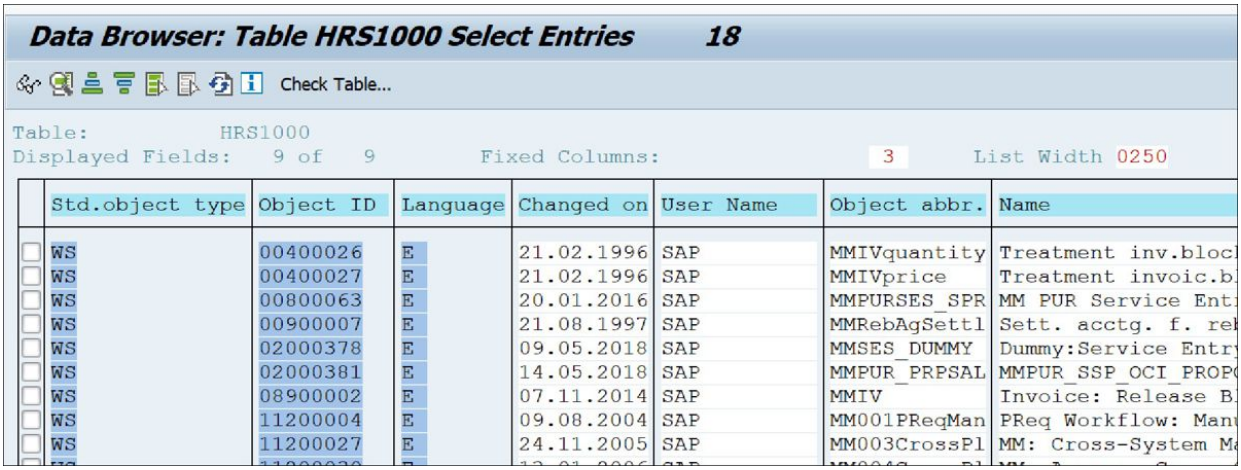


Figure 2.3 Results of the Preceding Selection Showing the Standard Workflow Template in Materials Management

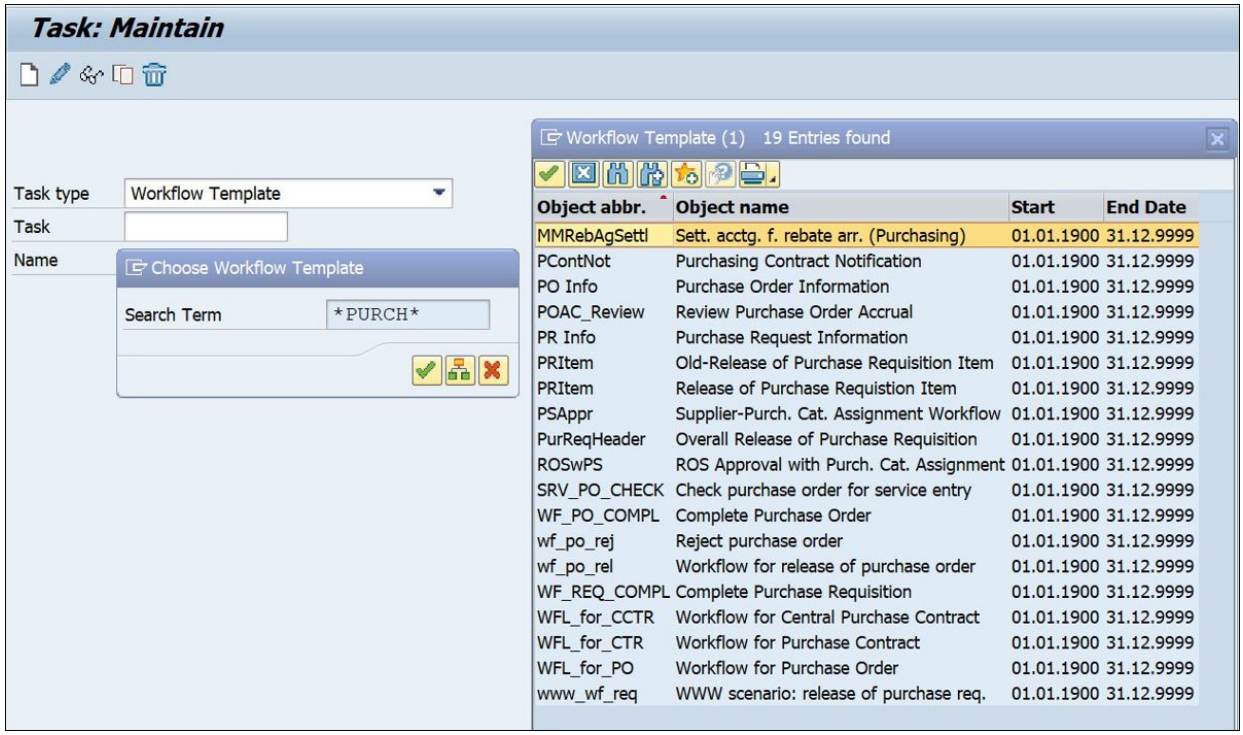


Figure 2.4 List of Purchasing Workflows from Transaction PFTC

Figure 2.5 Automatic Workflow Customizing

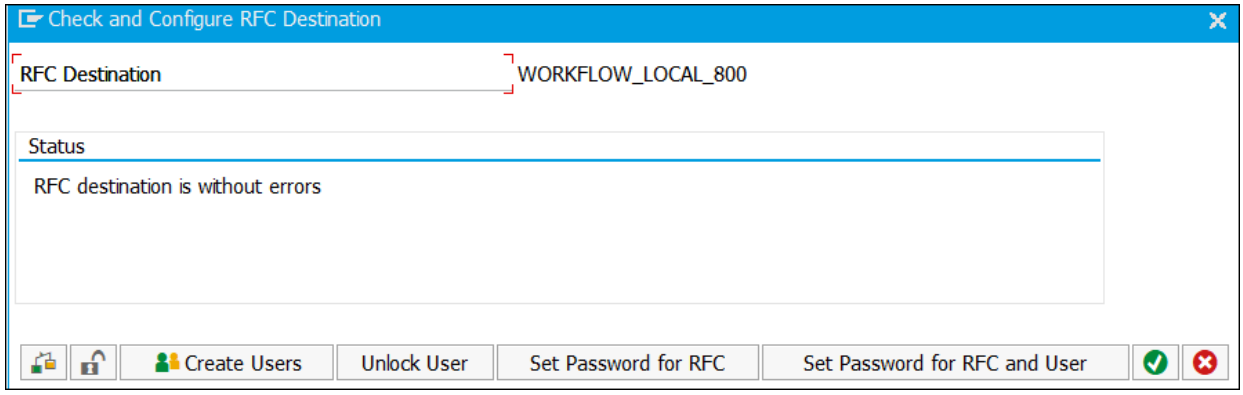


Figure 2.6 Automatic RFC Destination Configuration

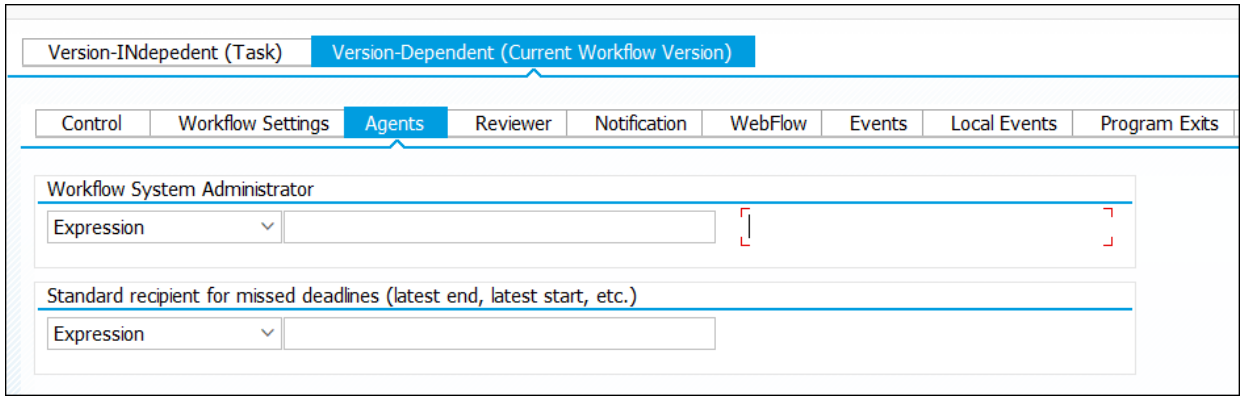


Figure 2.7 Maintain the Workflow System Administrator at the Workflow Level

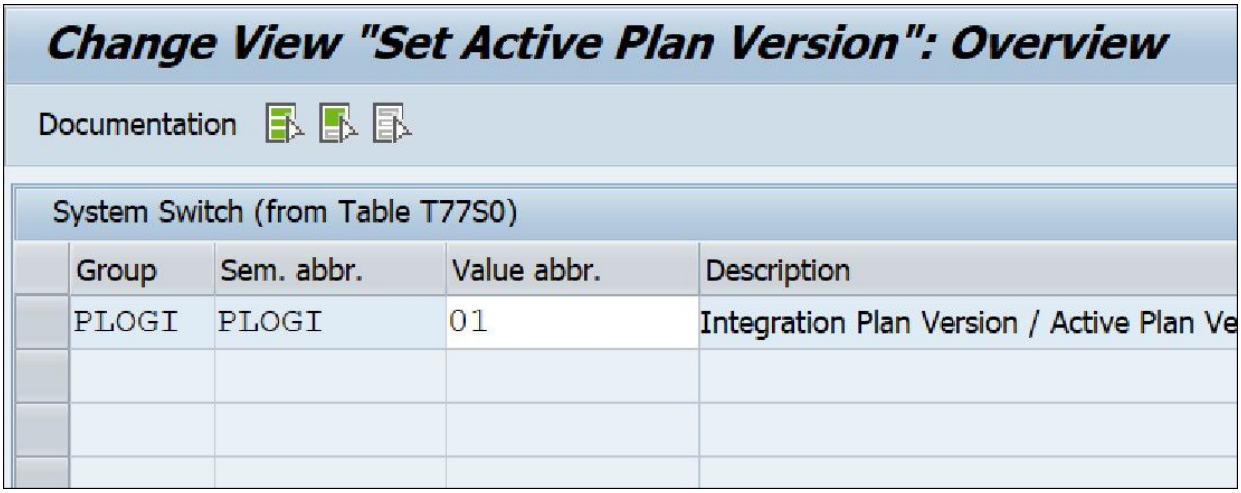


Figure 2.8 Manual Maintenance of the Active Plan Version

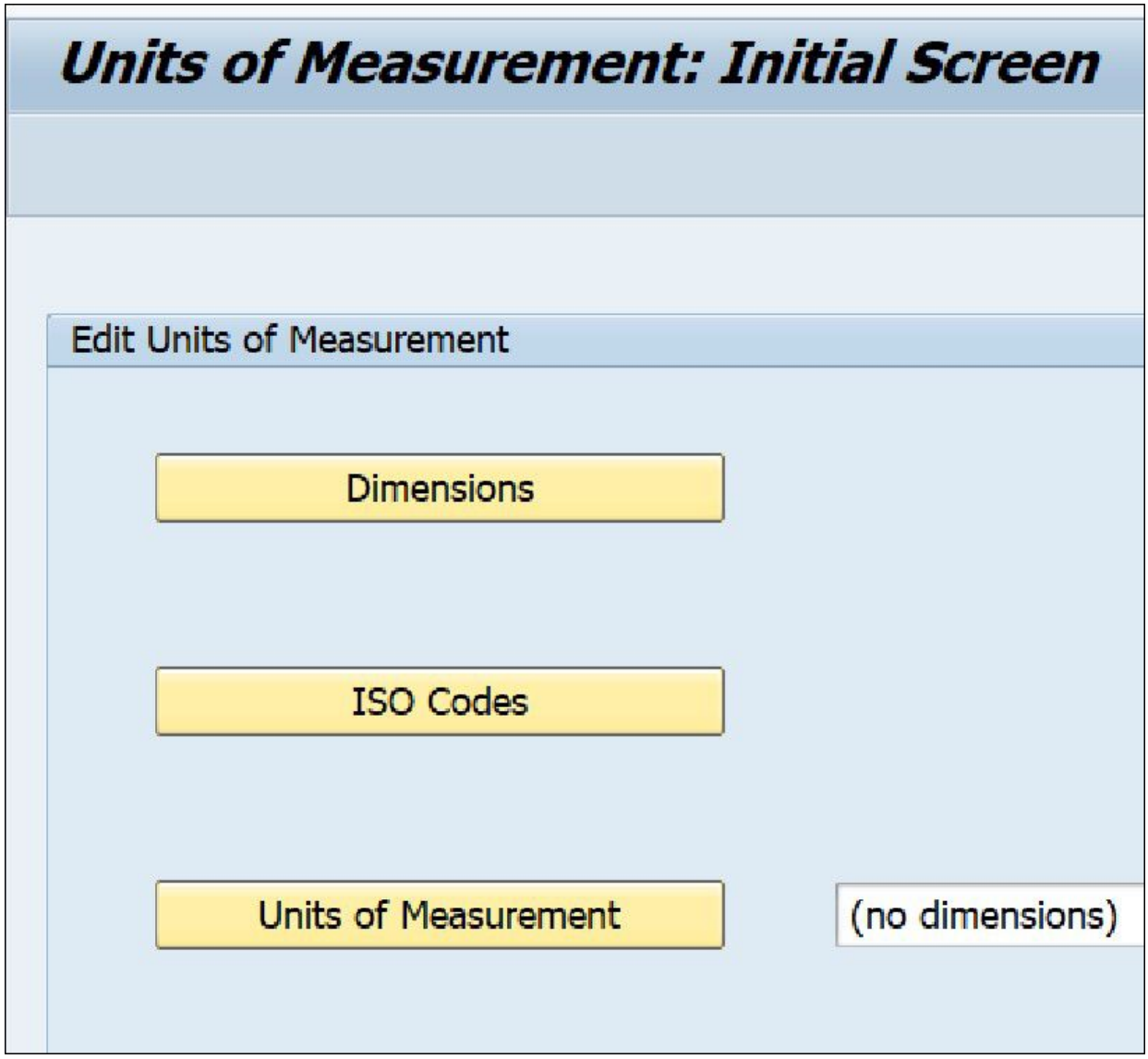


Figure 2.9 Shows the Maintenance of Units of Measurement

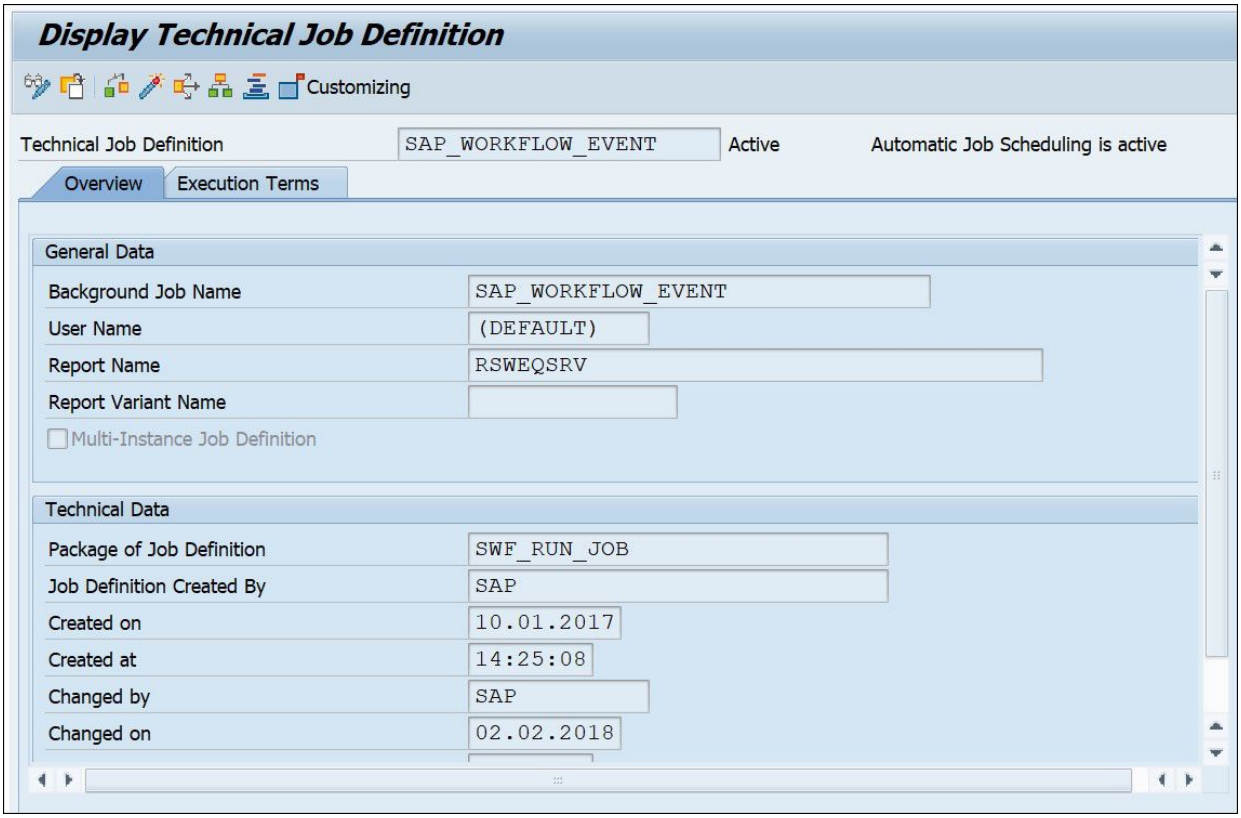


Figure 2.10 SAP Background Job Related to the Event Queue

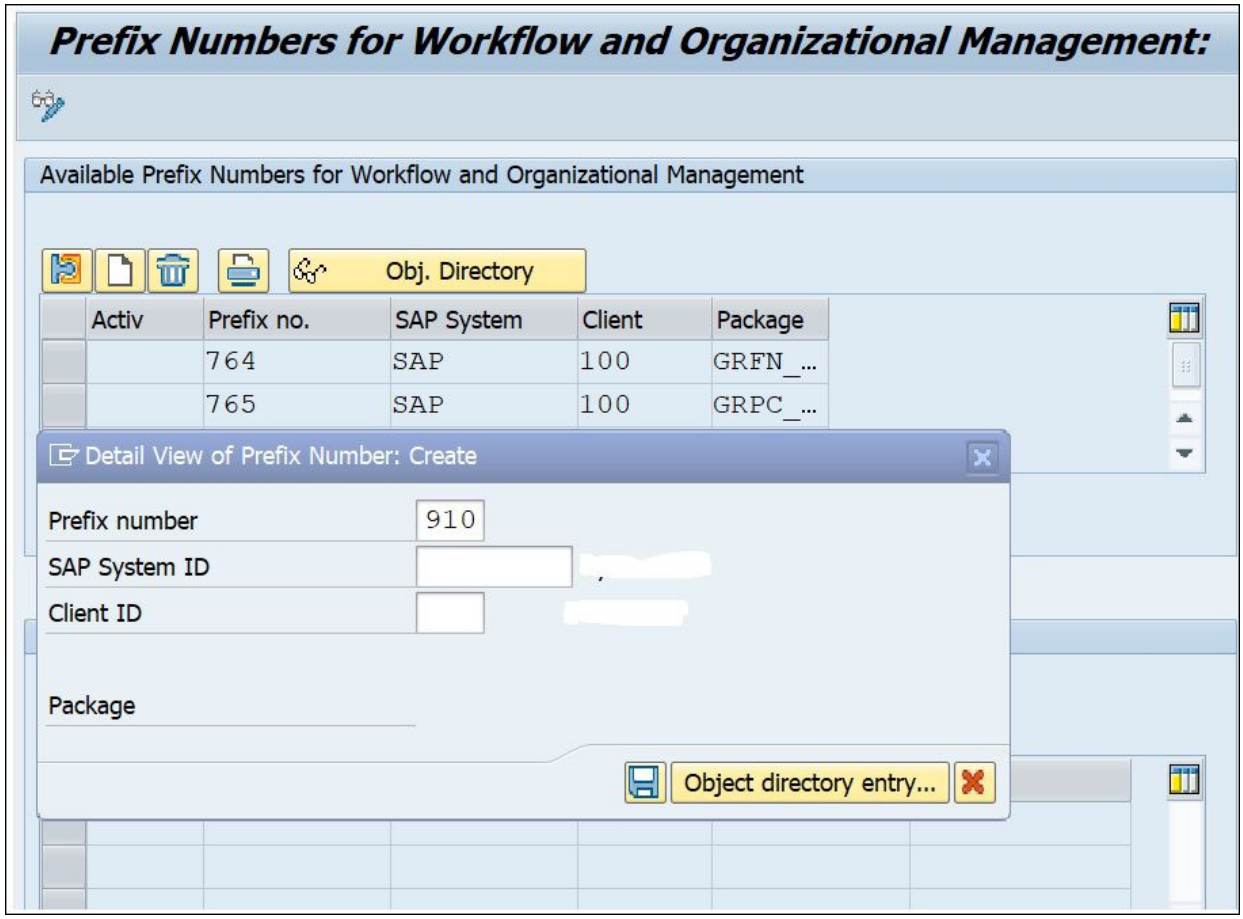


Figure 2.11 Maintain the Prefix Number for Workflow

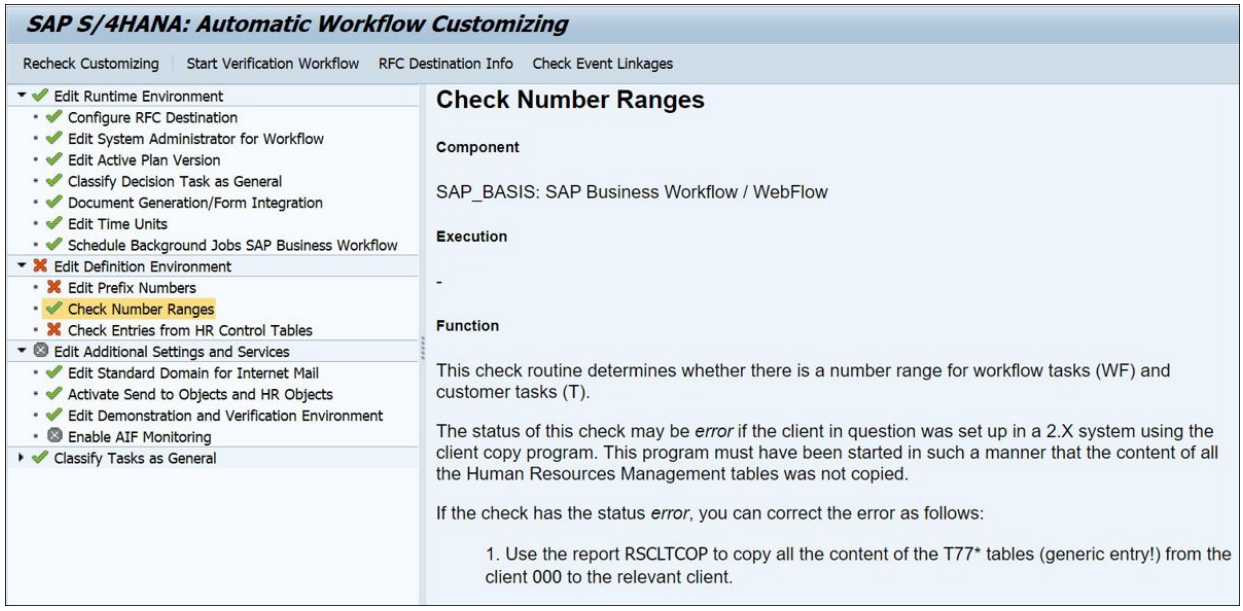


Figure 2.12 Check Number Ranges in SAP Automatic Workflow Customizing

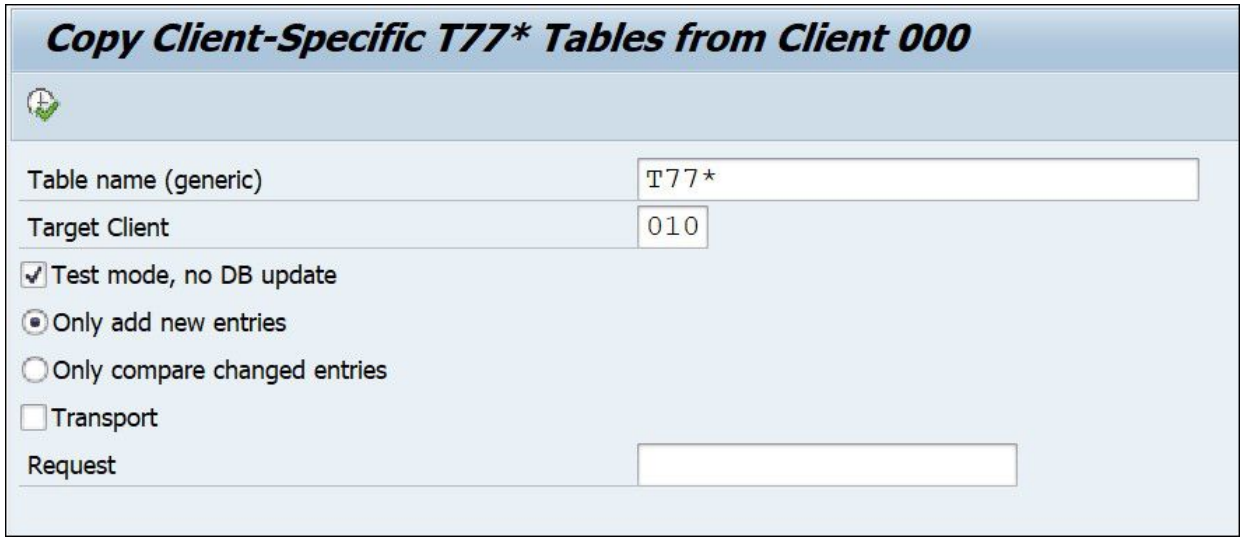


Figure 2.13 Report RHTTCP77

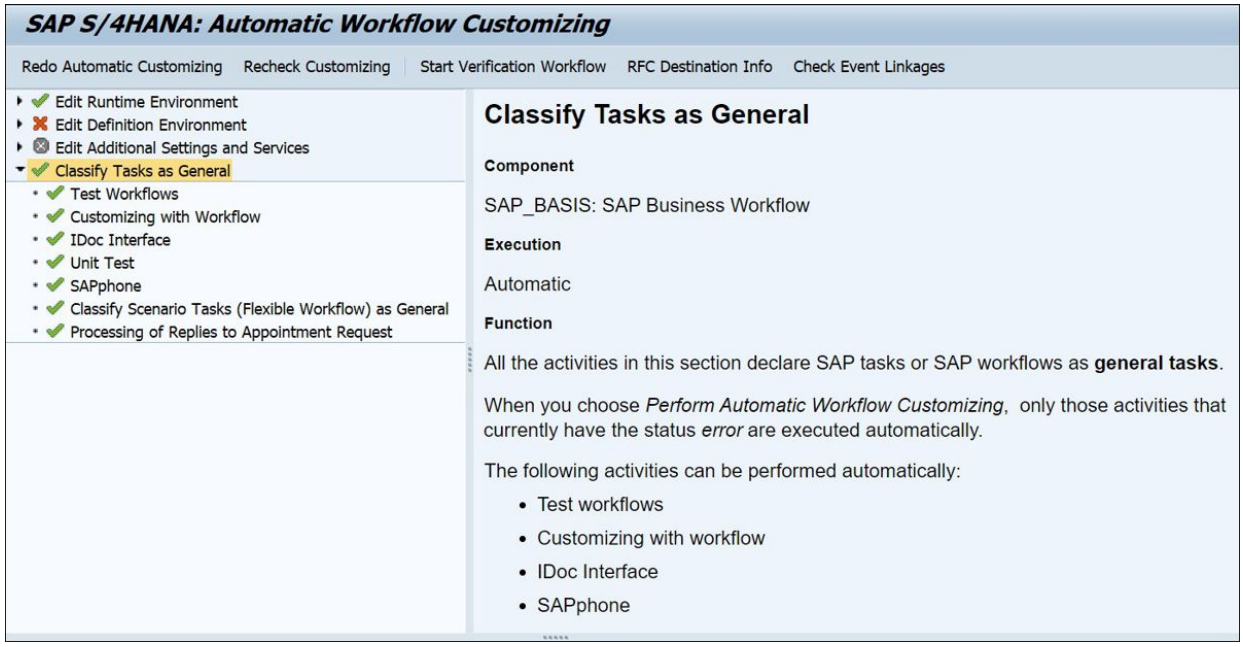


Figure 2.14 Classify Tasks as General

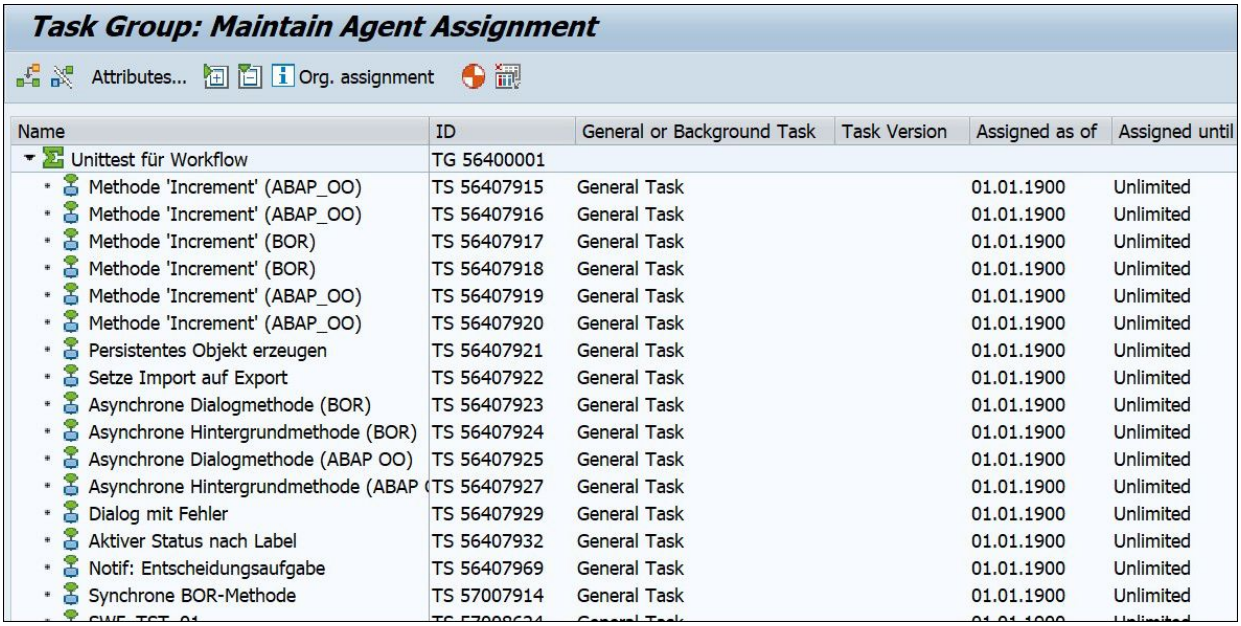


Figure 2.15 Task Group: Maintain Agent Assignment

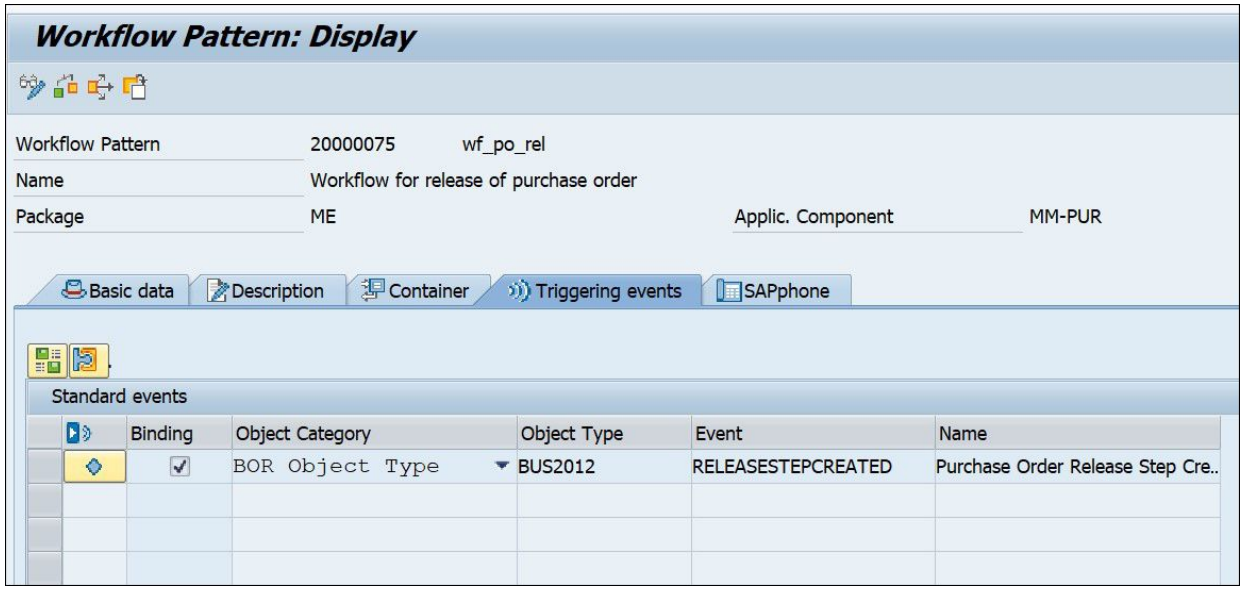


Figure 2.16 Triggering Event of Standard Purchase Release Workflow

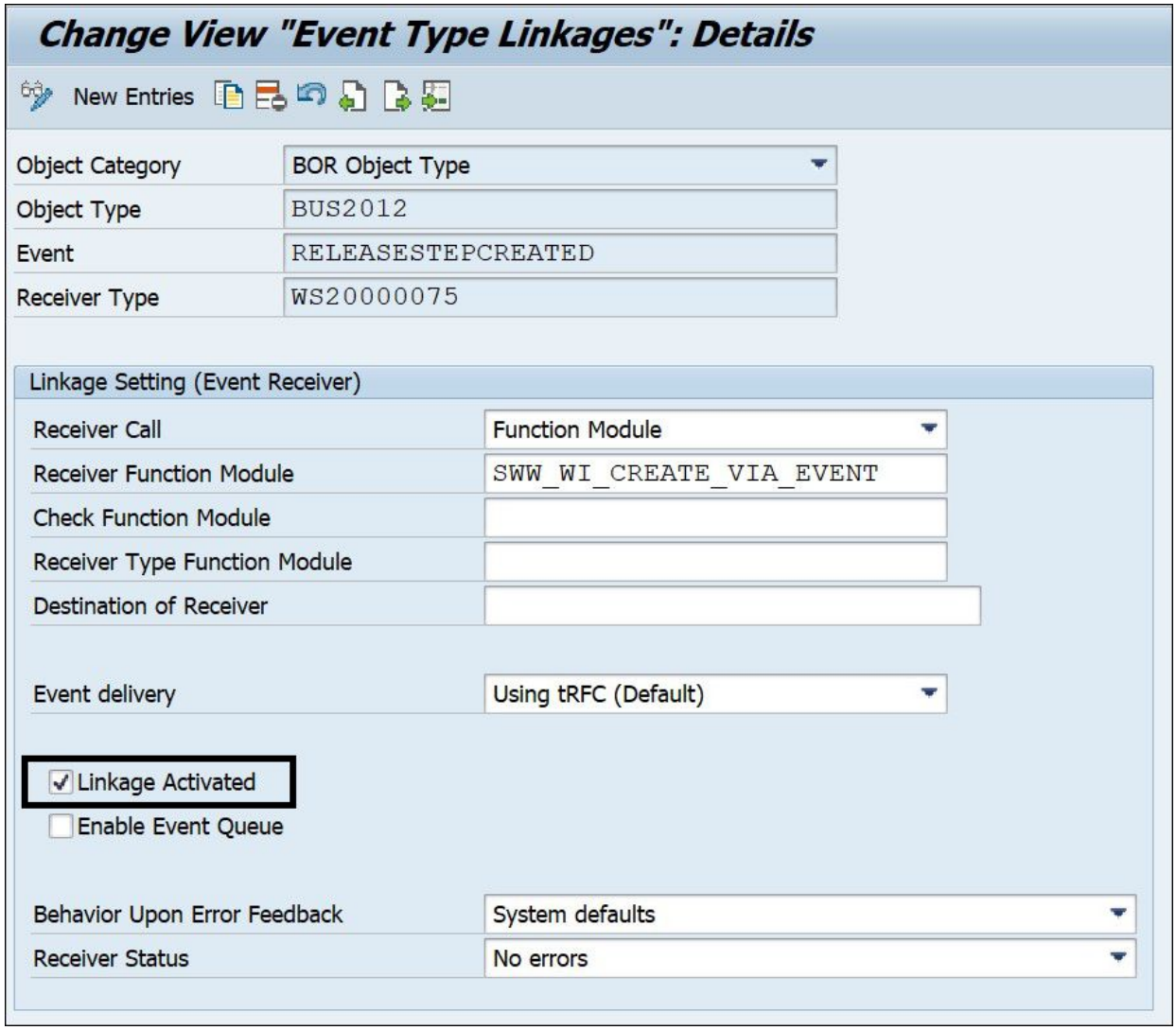


Figure 2.17 Event Linkage between Source Event RELEASESTEPCREATED (BUS2012) and Receiver Workflow Type WS20000075

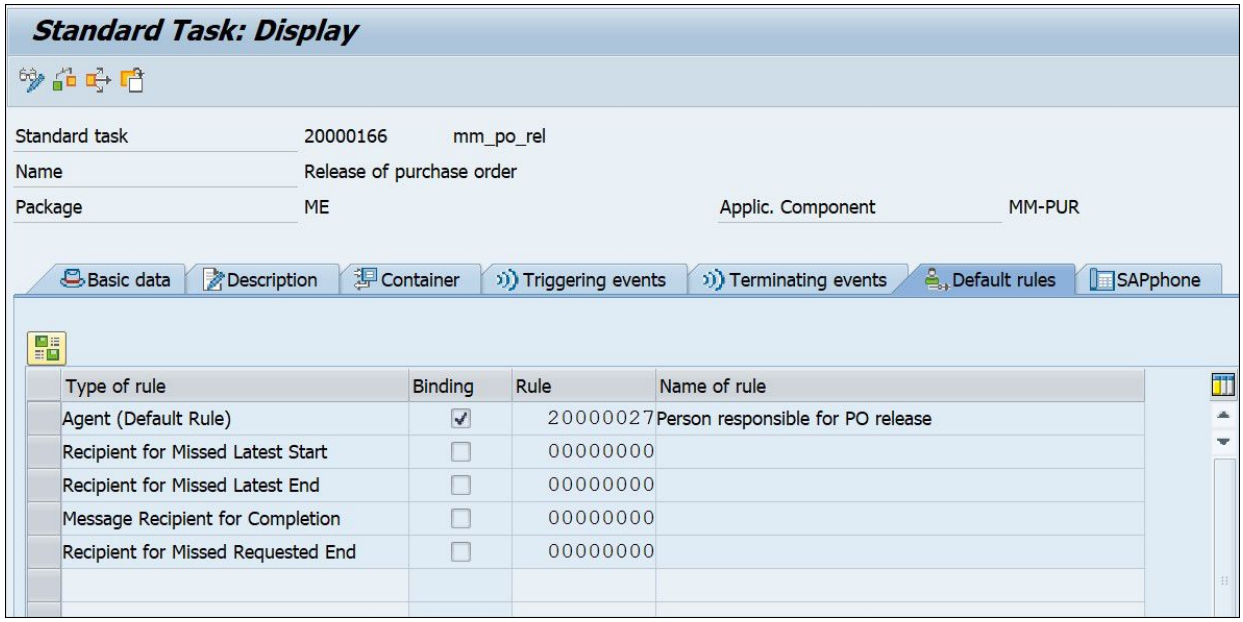


Figure 2.18 Standard Rule 20000027 Determines the Agent of Person Responsible for PO Release

Grp	Code	OT	Agent ID
01	Z1	US	USER1
01	Z2	US	USER2
02	Z1	US	USER3
03	Z2	US	USER4

Figure 2.19 Sample Agent Determination Based on Purchase Release Group and Release Code

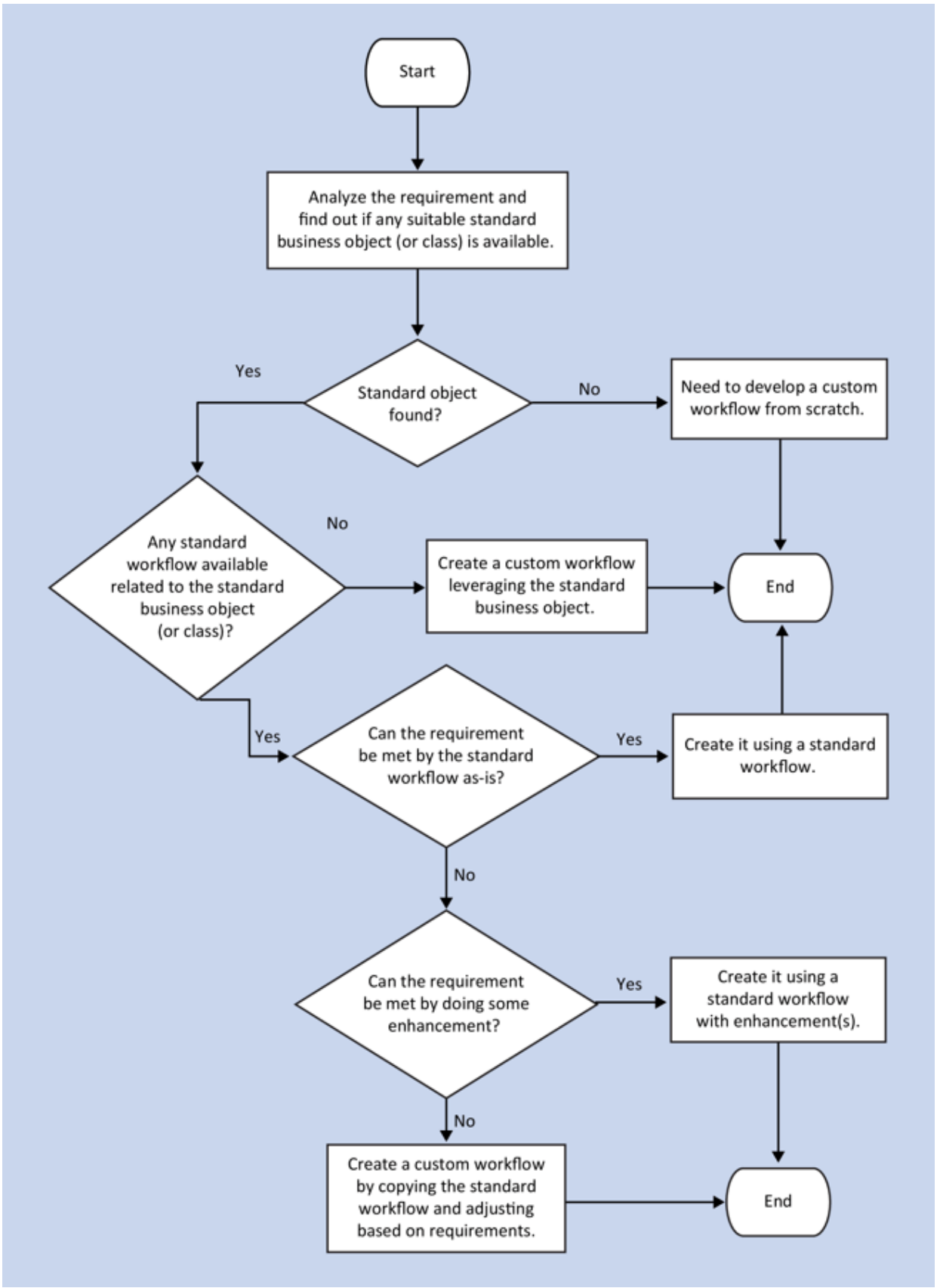

Start
Analyze the requirement and find out if any suitable standard business object (or class) is available.
Standard object found?
Yes
No
Need to develop a custom workflow from scratch.
Any standard workflow available related to the standard business object (or class)?
No
Create a custom workflow leveraging the standard business object.
End
Yes
Can the requirement be met by the standard workflow as-is?
Yes
Create it using a standard workflow.
No
Can the requirement be met by doing some enhancement?
Yes
Create it using a standard workflow with enhancement(s).
No
Create a custom workflow by copying the standard workflow and adjusting based on requirements.
End

Figure 2.20 Flow Diagram Depicting When to Create a Custom Workflow

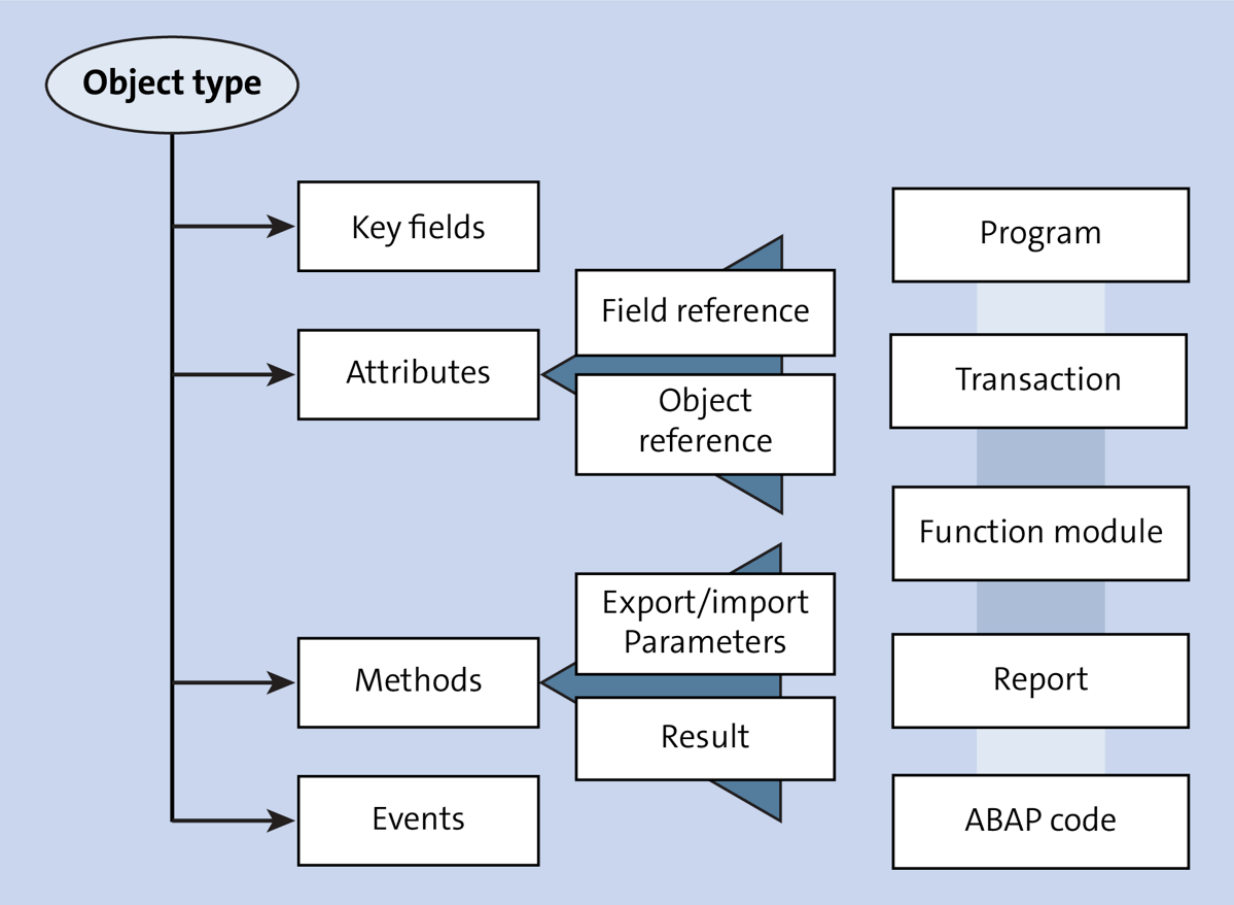


Figure 3.1 Definition of a Business Object Type with Key Fields, Attributes, Methods, and Events

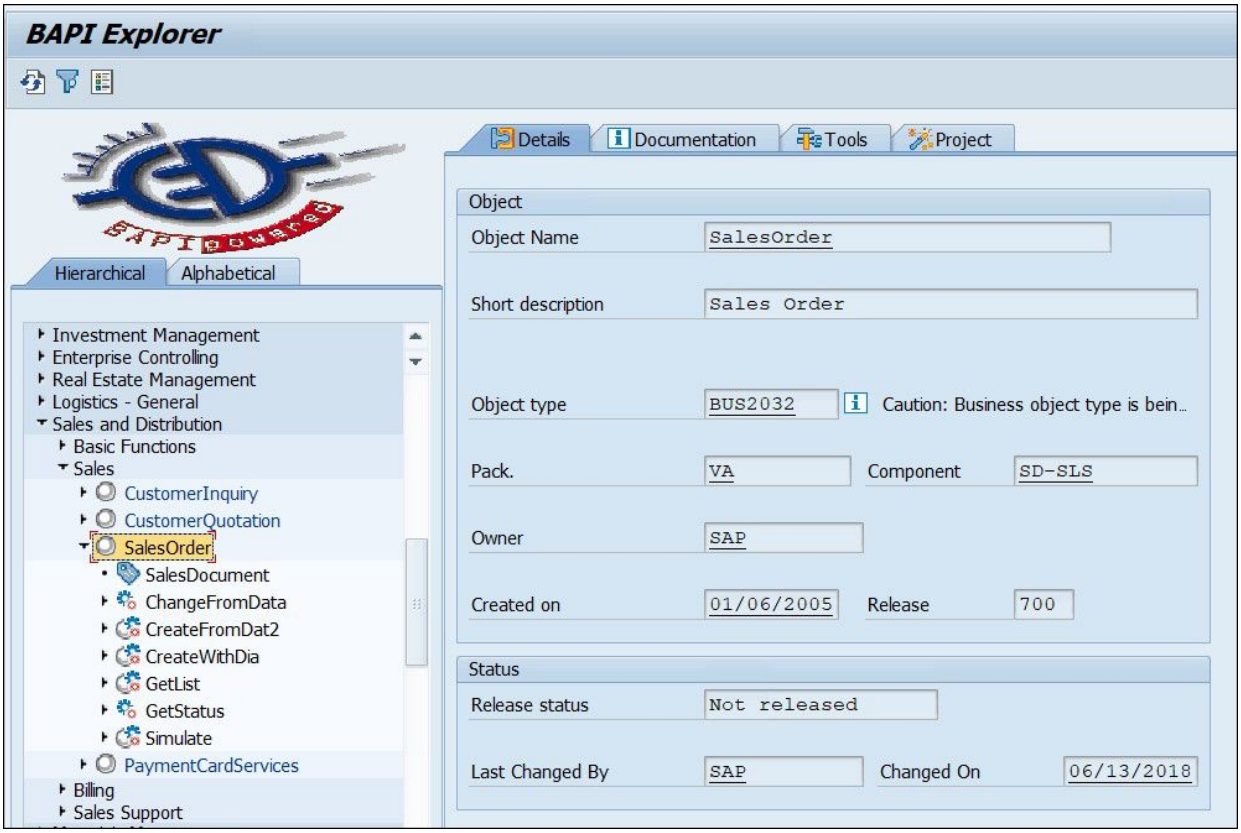


Figure 3.2 Sales Order Business Object Type from BAPI Explorer

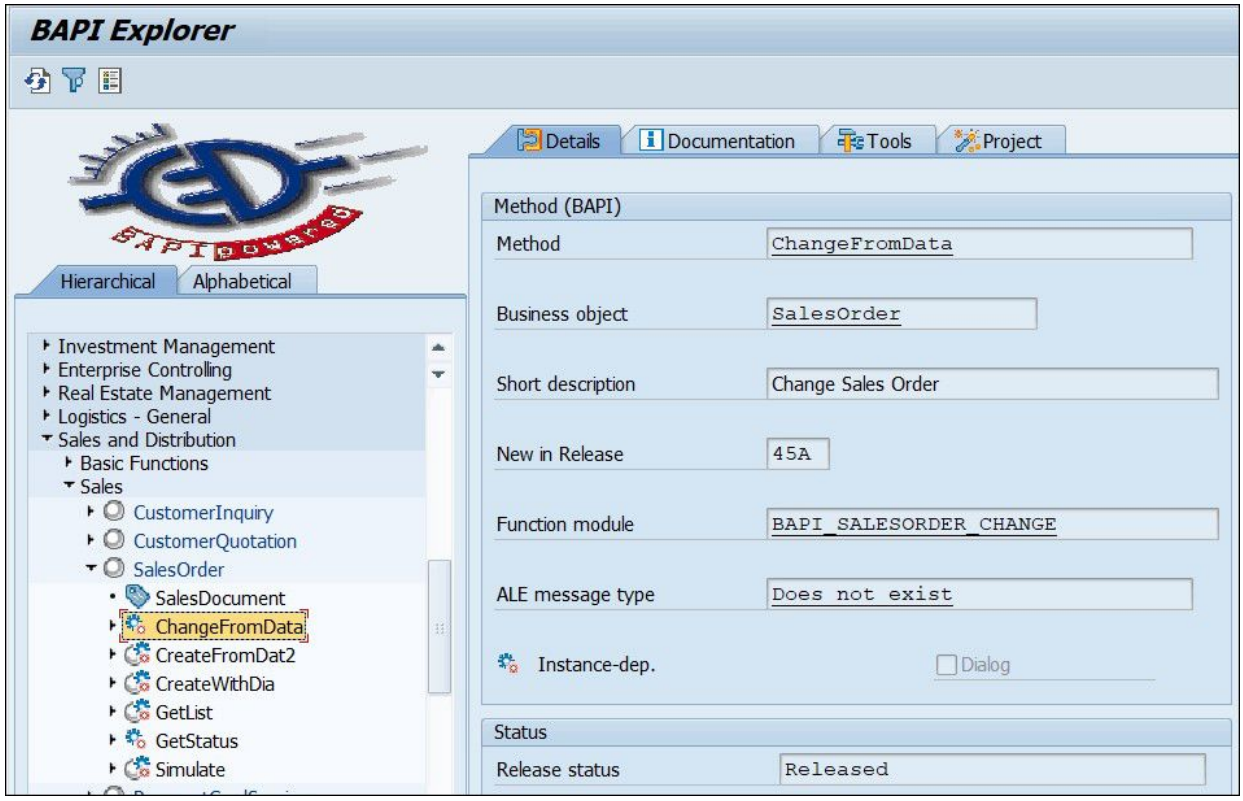


Figure 3.3 Method ChangeFromData under SalesOrder Object Type Showing the Link to BAPI

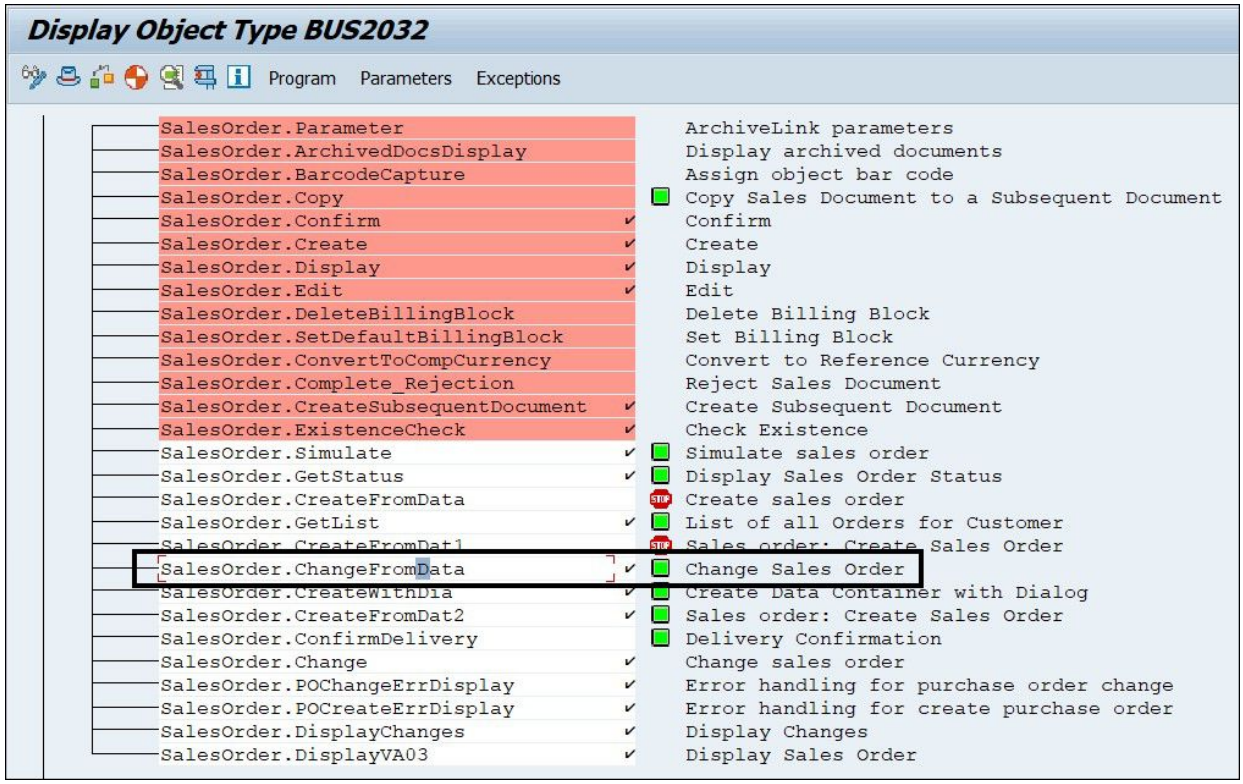


Figure 3.4 View of Methods under a Business Object Type Definition in Transaction SWO1

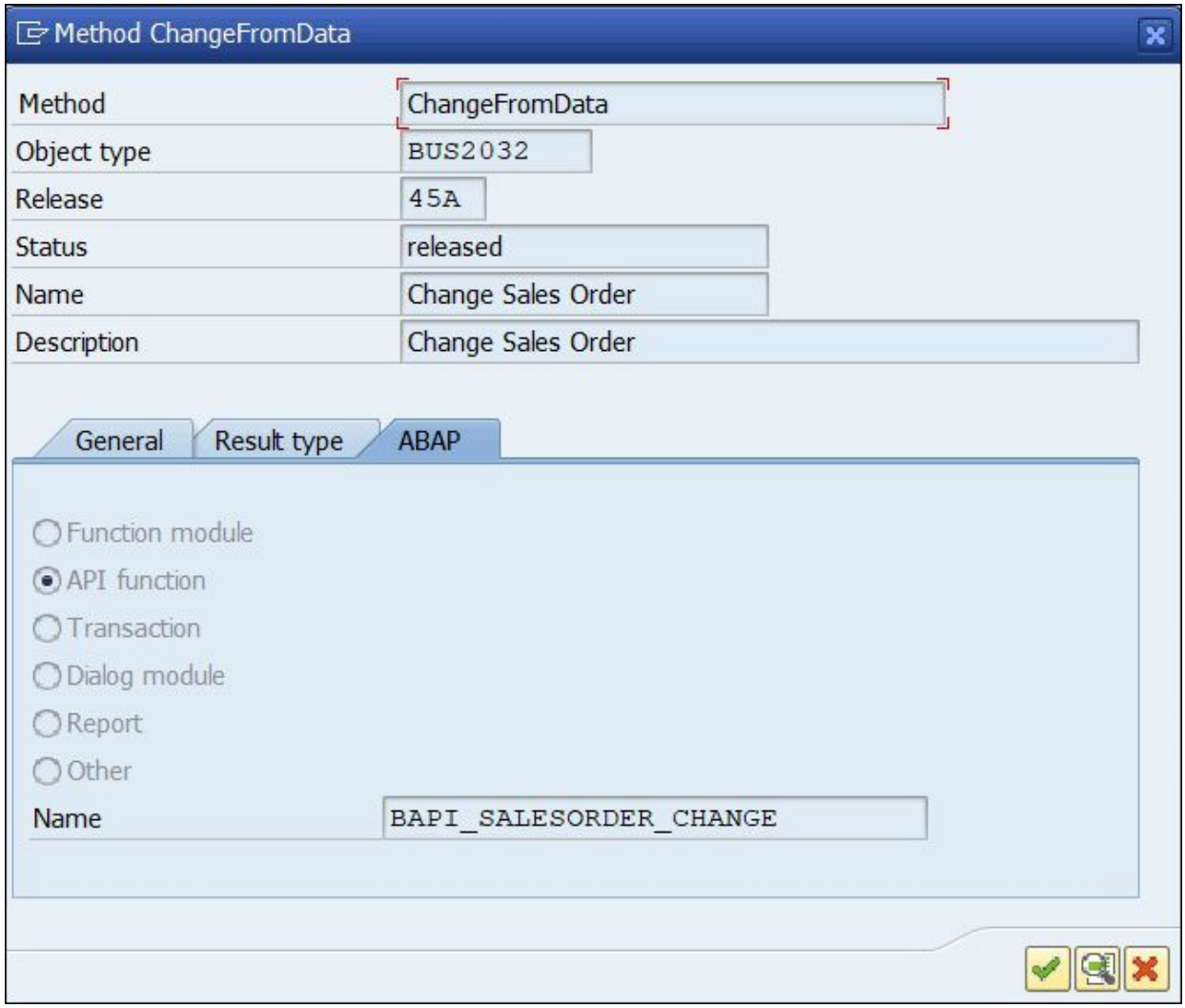


Figure 3.5 Method Definition in a Business Object Type Showing the BAPI Call as an API Function

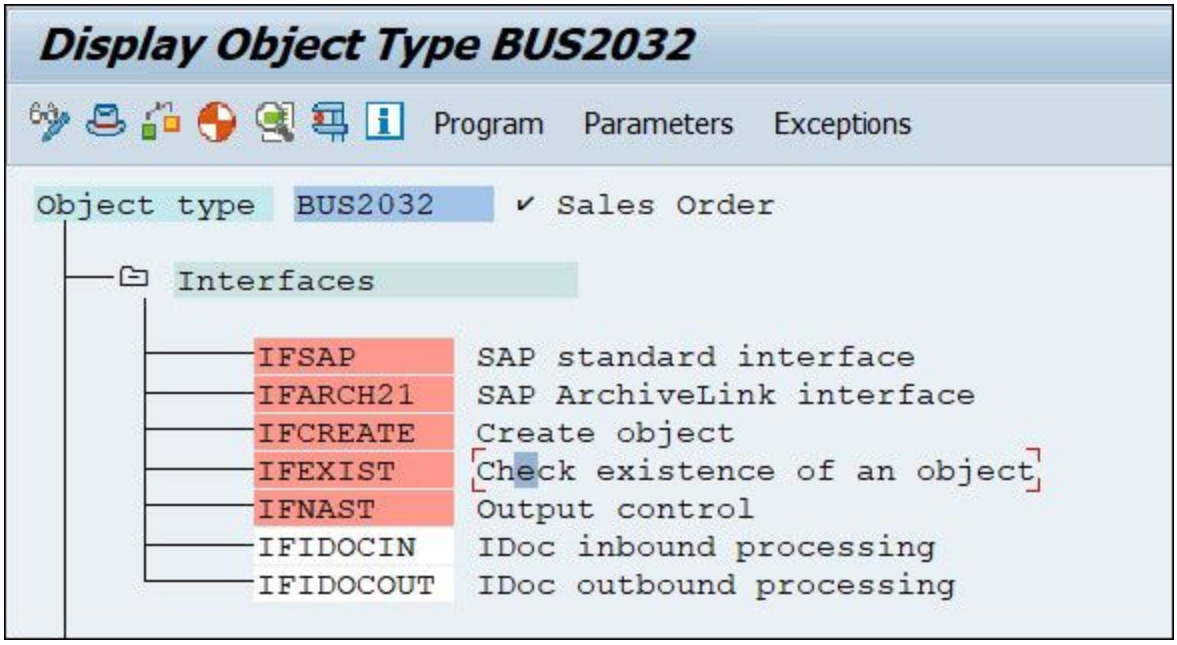


Figure 3.6 View of Interfaces Defined in a Sample Business Object Type

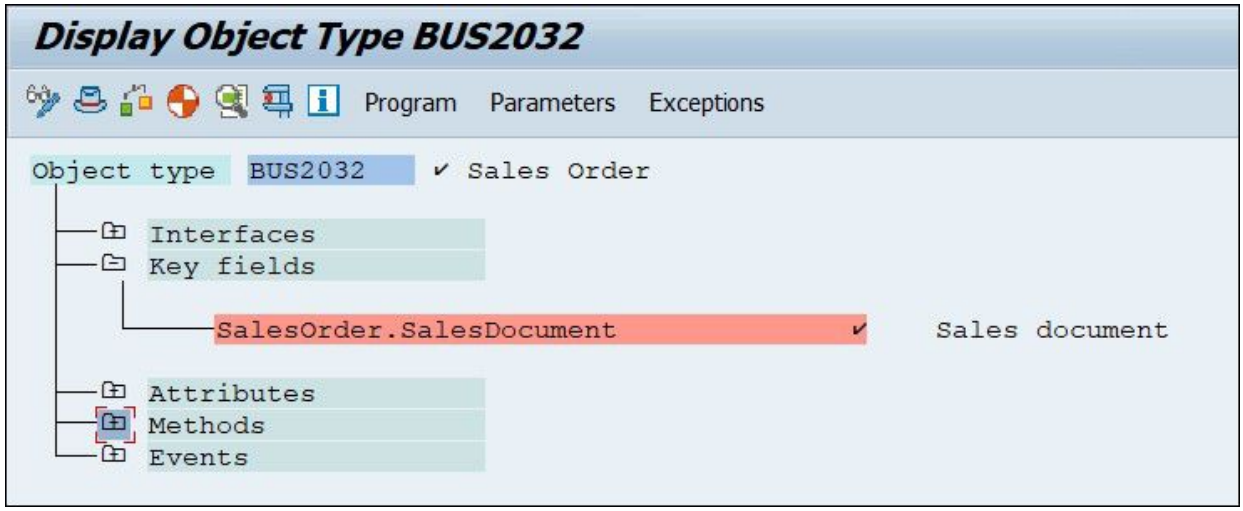


Figure 3.7 Definition of Key Field in a Business Object Type

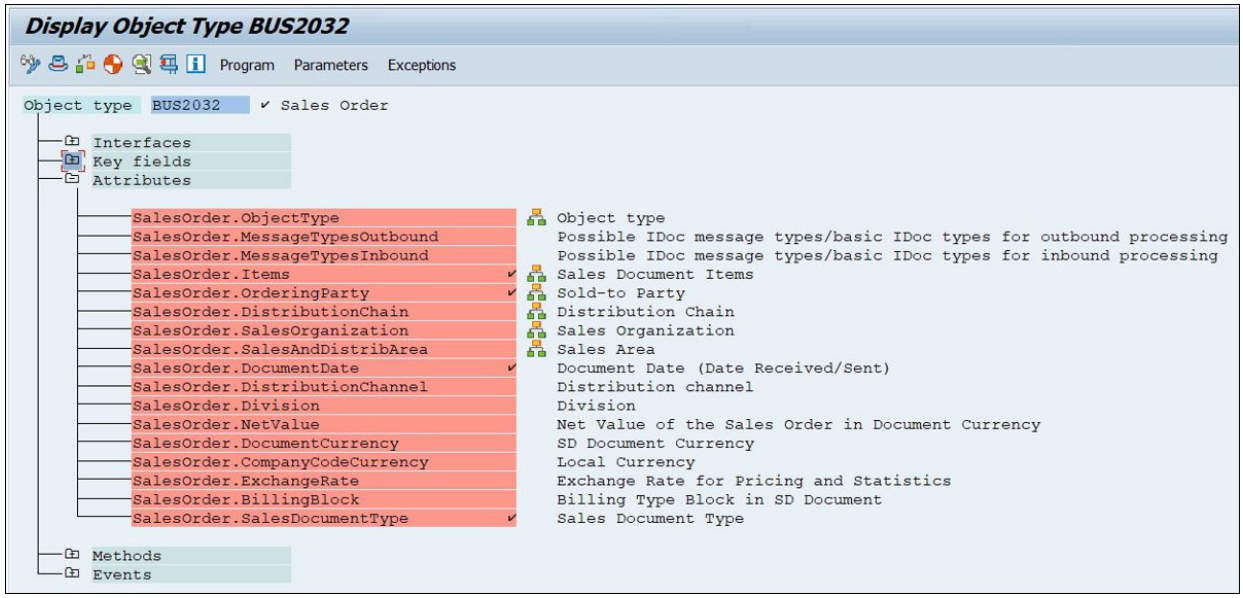


Figure 3.8 Definition of Attributes in a Business Object Type

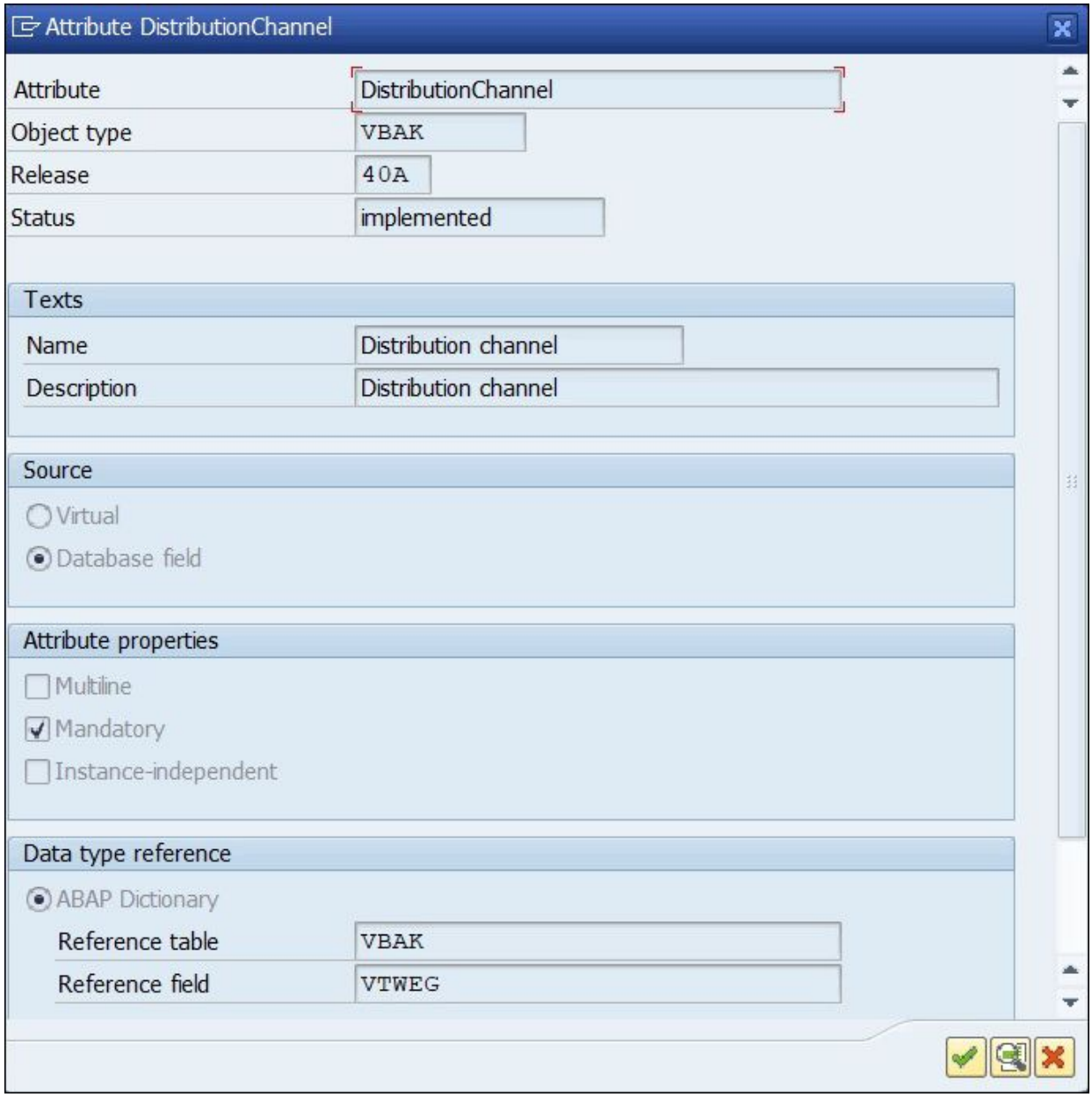


Figure 3.9 Attribute Definition Showing Database/Virtual, Multiline, Reference Table/Field Properties

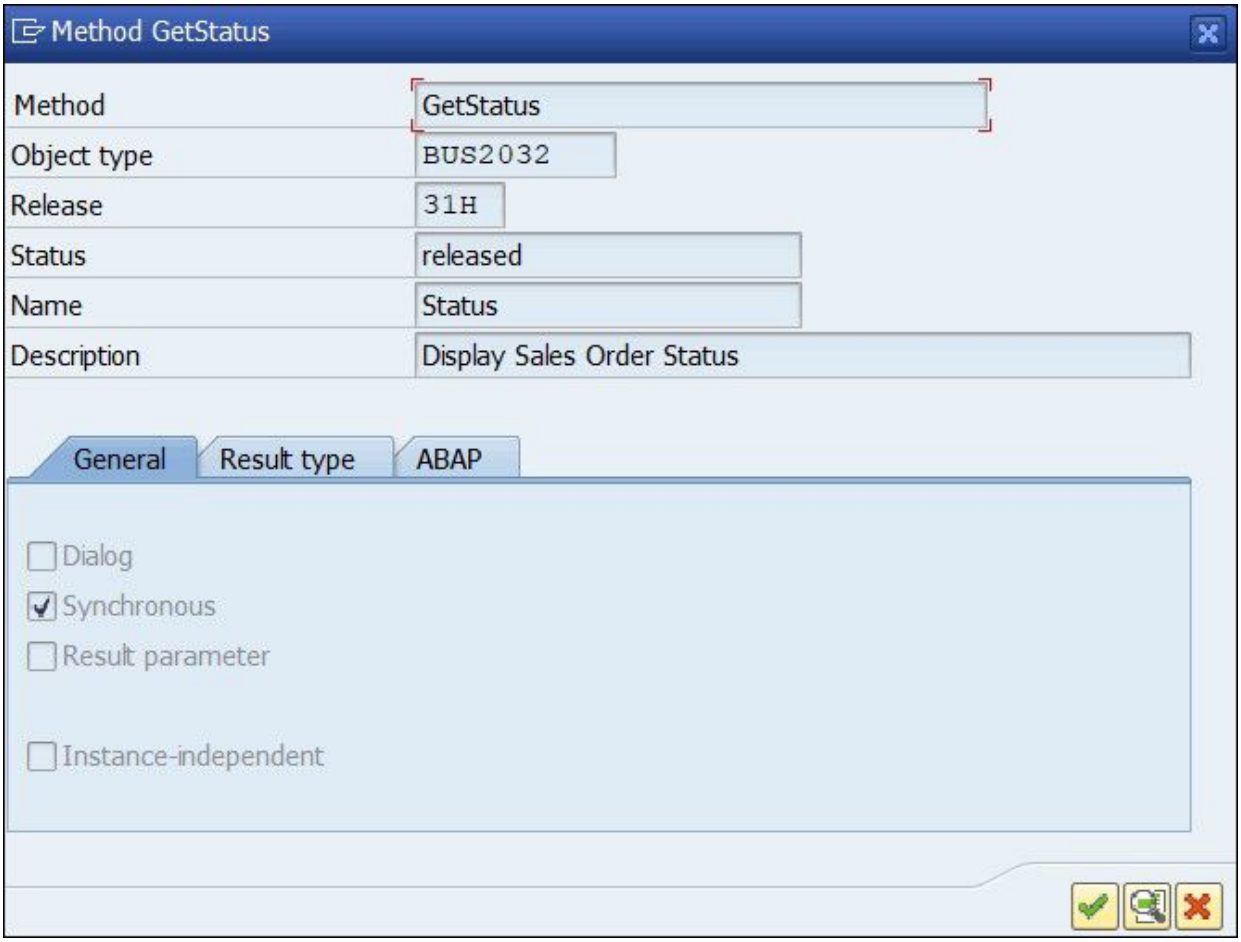


Figure 3.10 Method Definition Showing Dialog, Synchronous, Result Parameter, and Instance Independent Properties

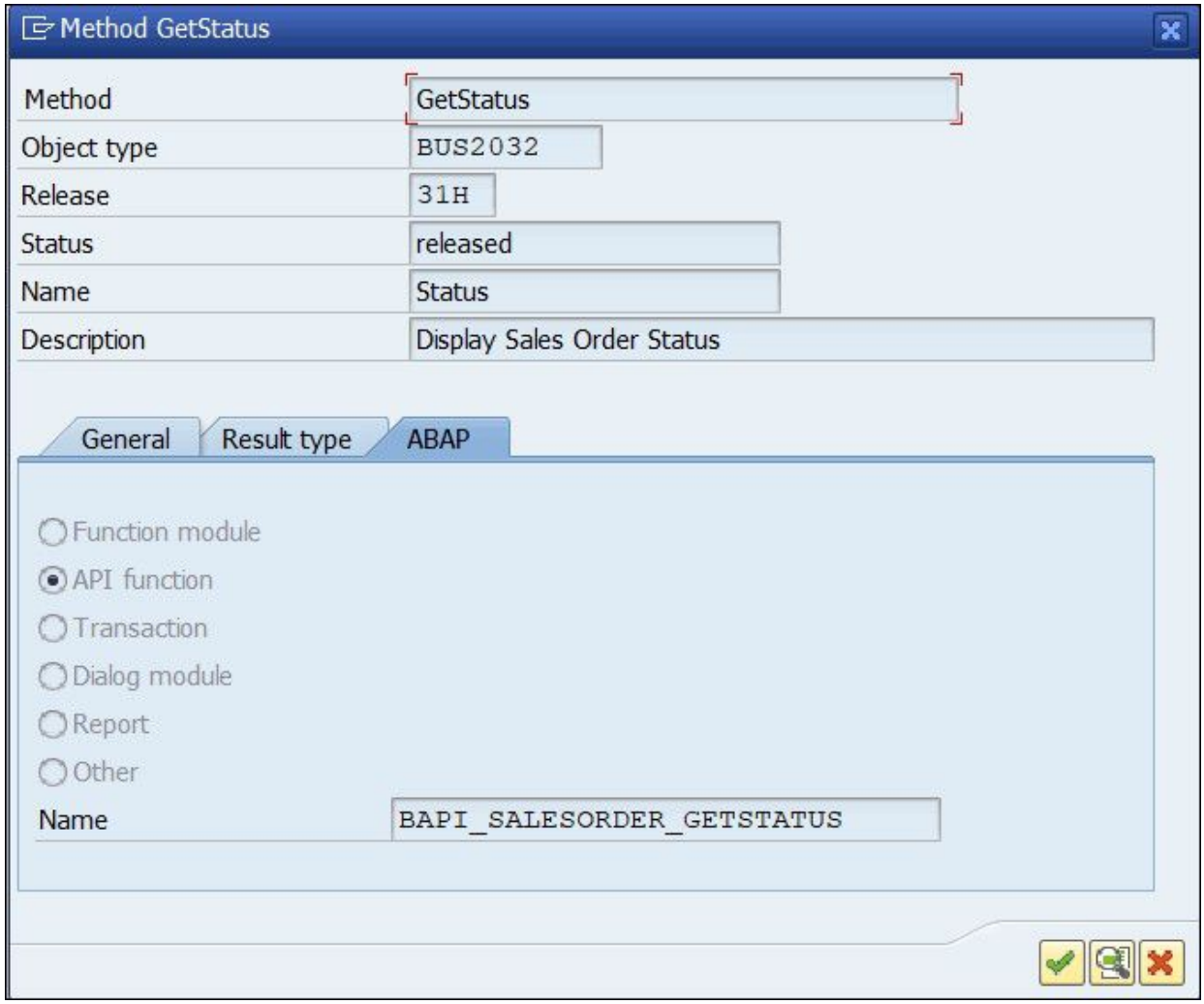


Figure 3.11 Method Definition Showing the Different ABAP Coding Options Available

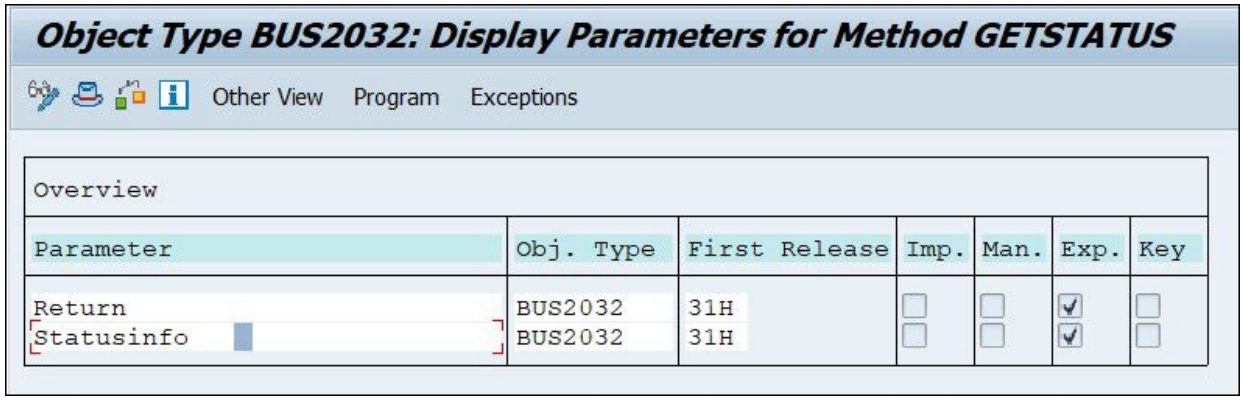


Figure 3.12 Method Parameter Definition in a BOR Type

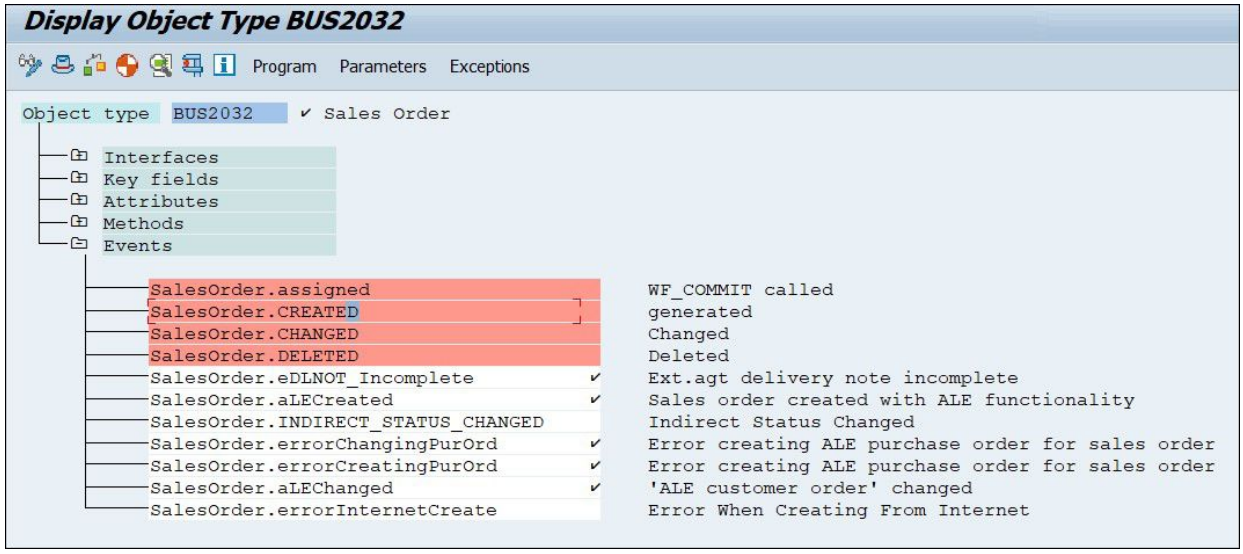


Figure 3.13 View of Event Definition within an Object Type

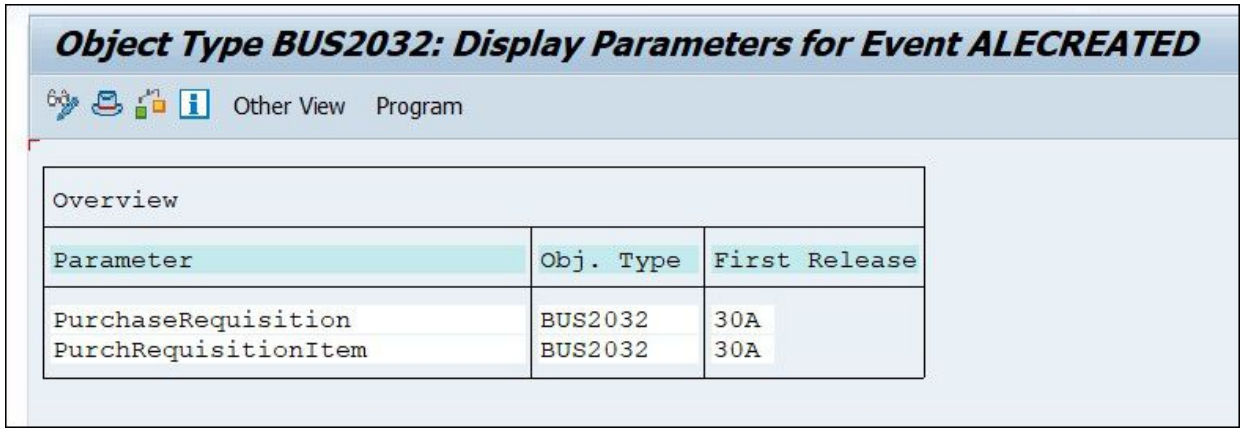


Figure 3.14 Definition of Import Parameters in an Event on a BOR Type

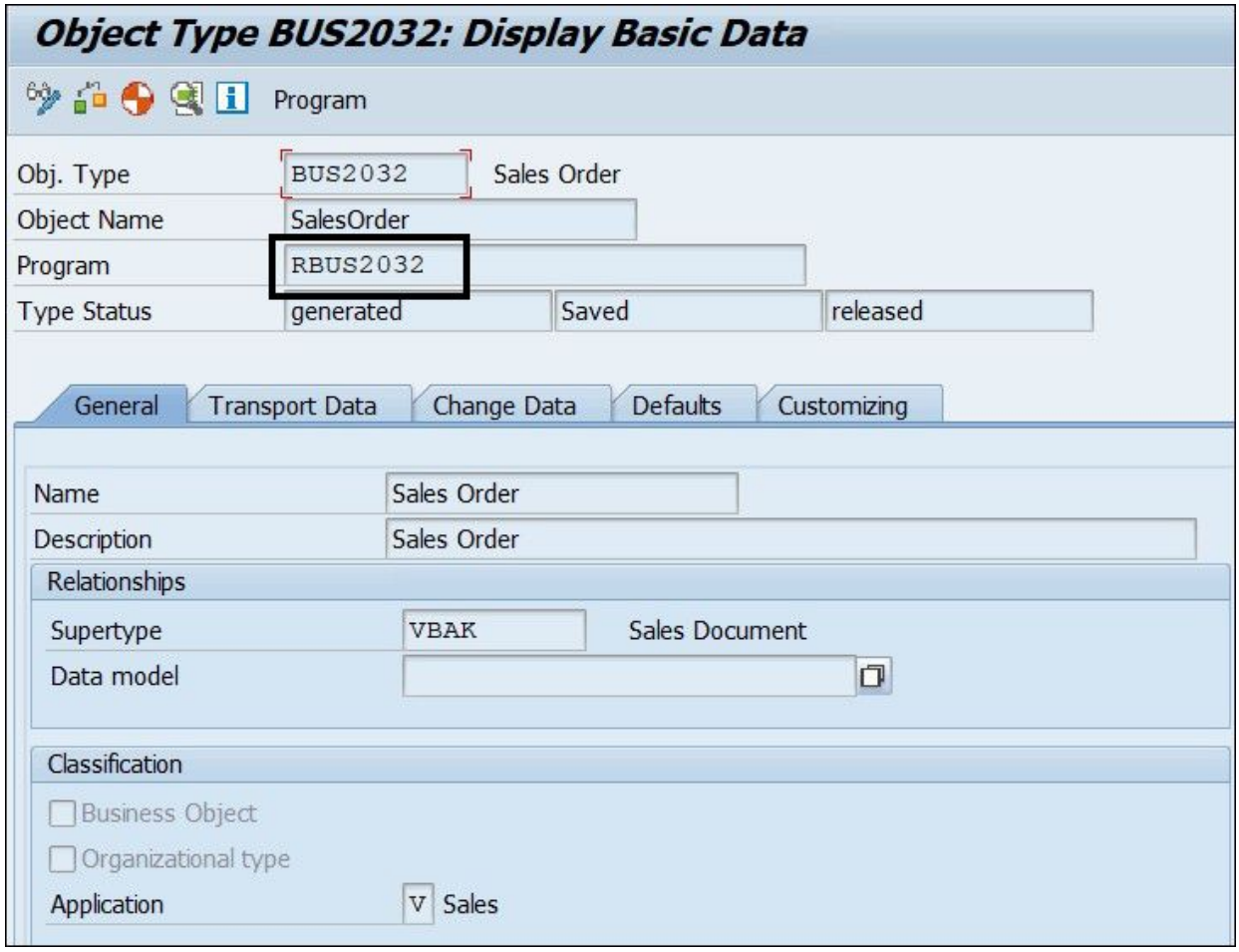


Figure 3.15 BOR Program Name Maintained at the Header Level

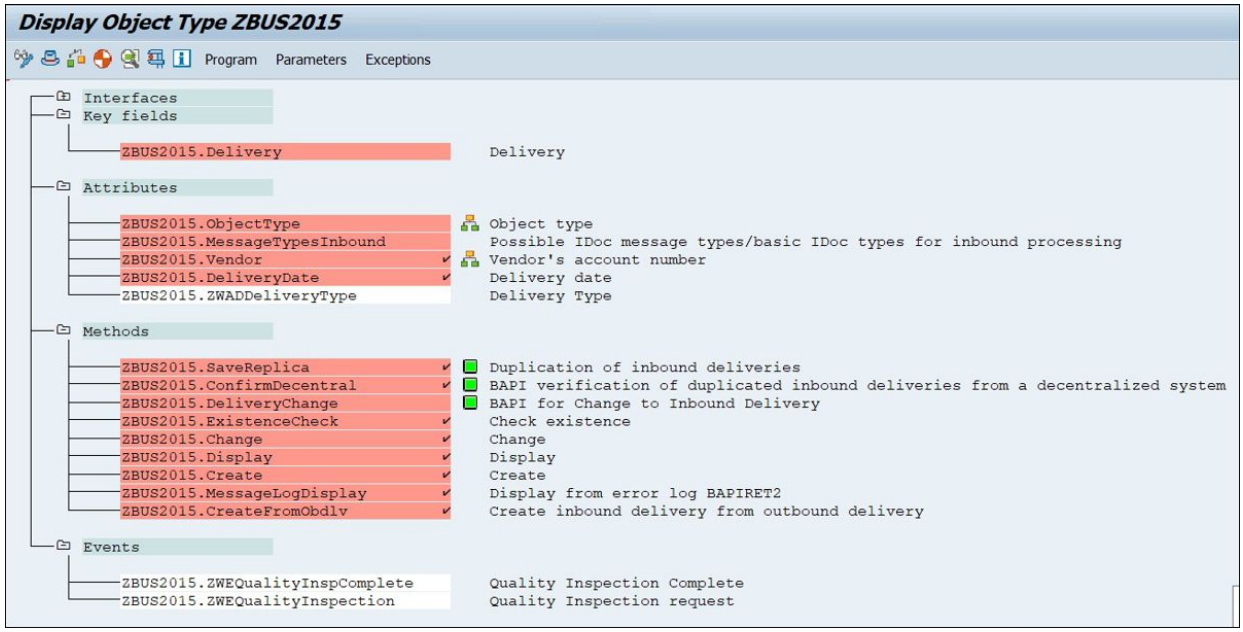


Figure 3.16 Custom BOR Subtype Definition Showing Inherited Components

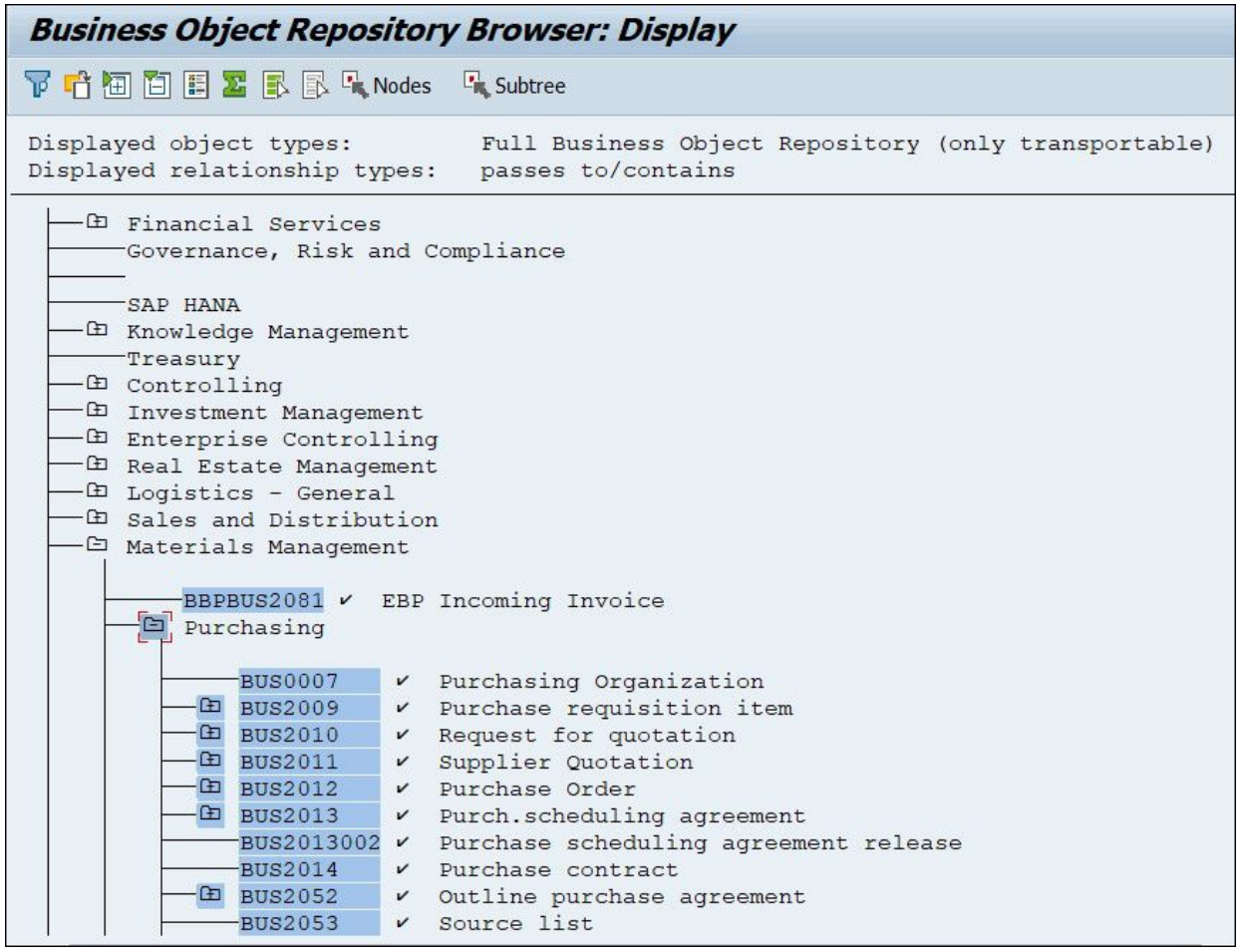


Figure 3.17 View of the BOR in SAP with Object Type Hierarchies Grouped by Application Components

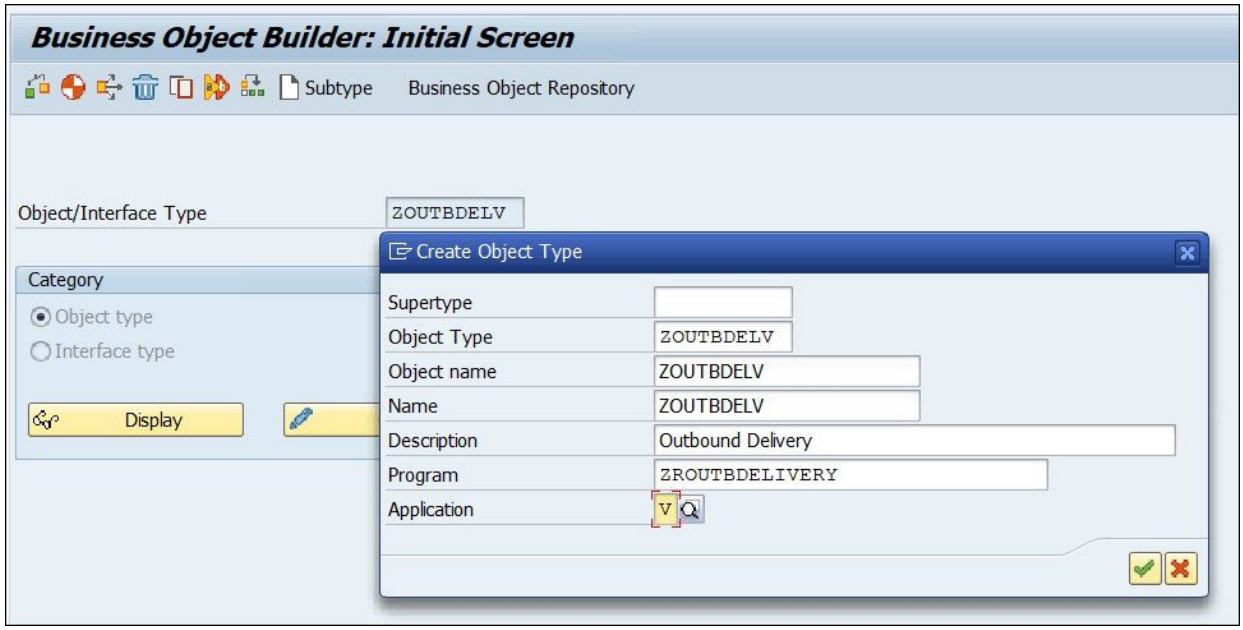


Figure 3.18 Creating a Custom BOR Type from Transaction SWO1

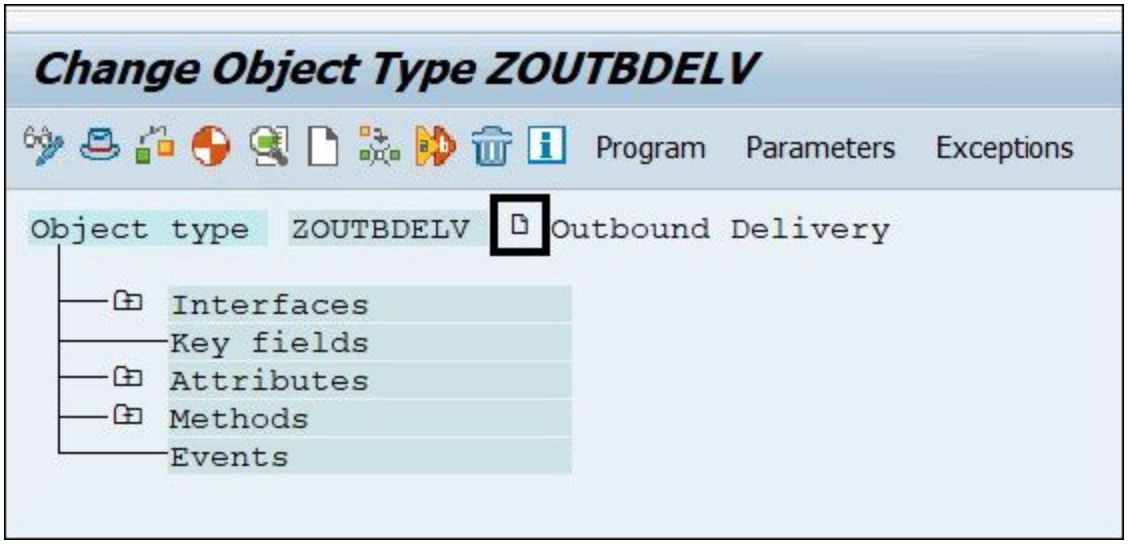


Figure 3.19 Object Type Definition Created in Modeled Status

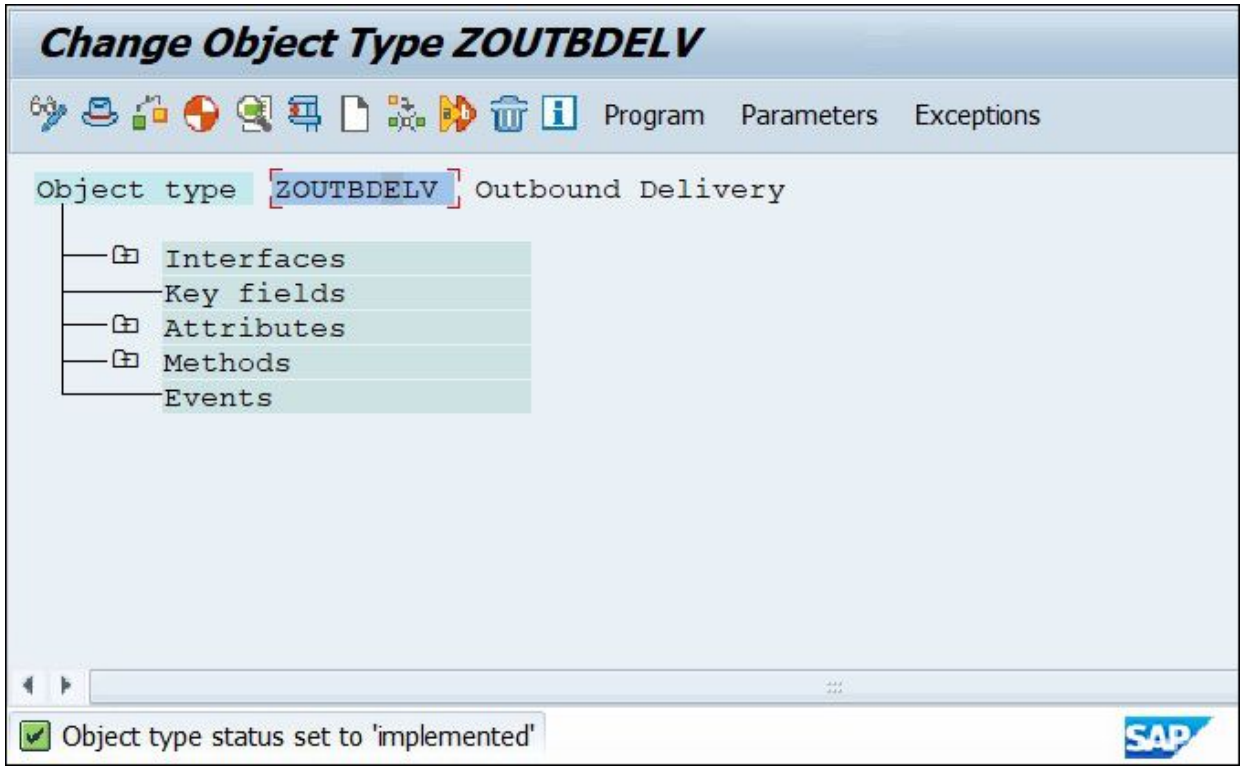


Figure 3.20 Object Type Status Changed to Implemented

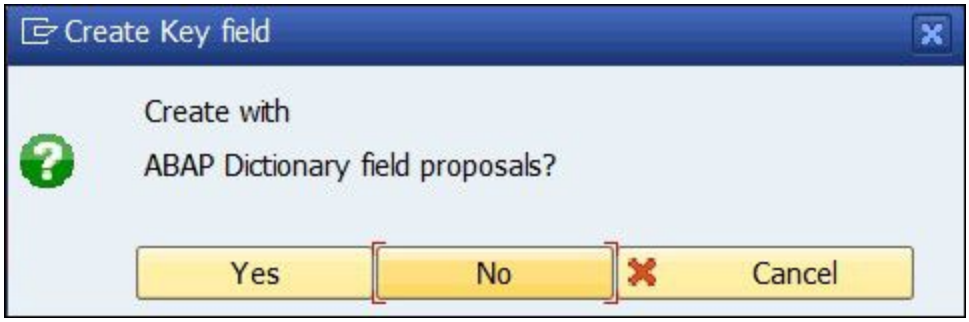


Figure 3.21 Popup to Choose between Database or Virtual Definition of the Key Field

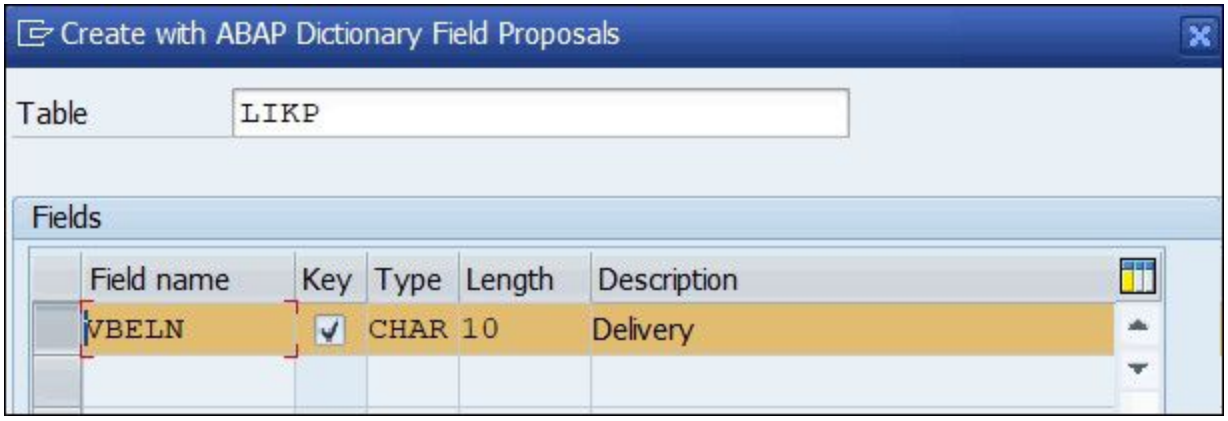


Figure 3.22 Selection of the Primary Key Table and Key Field

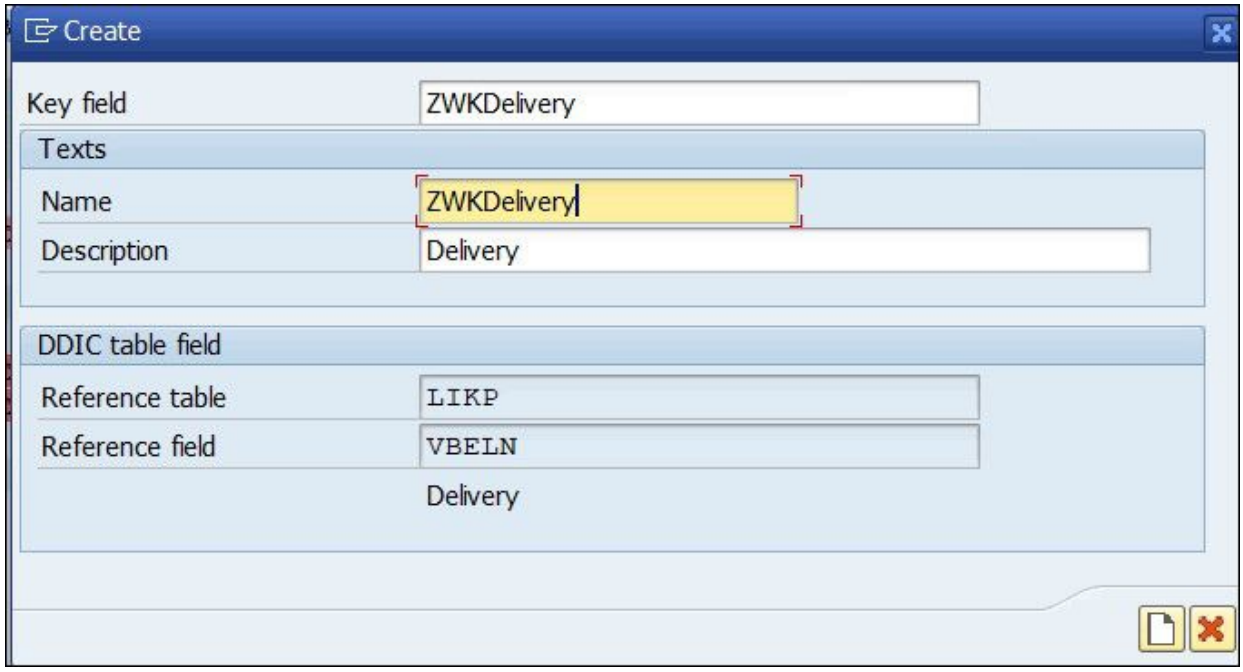


Figure 3.23 Confirming the Key Field Name

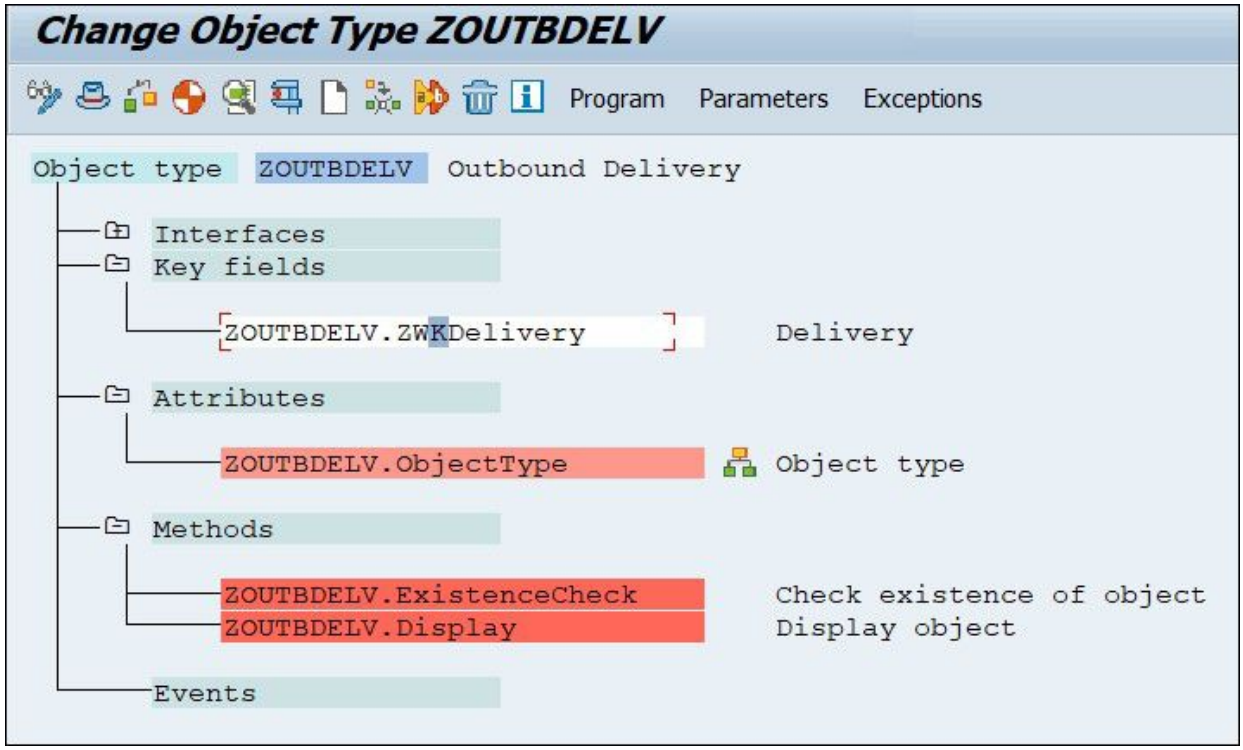


Figure 3.24 View of Object Type Definition after Creating the Key Field

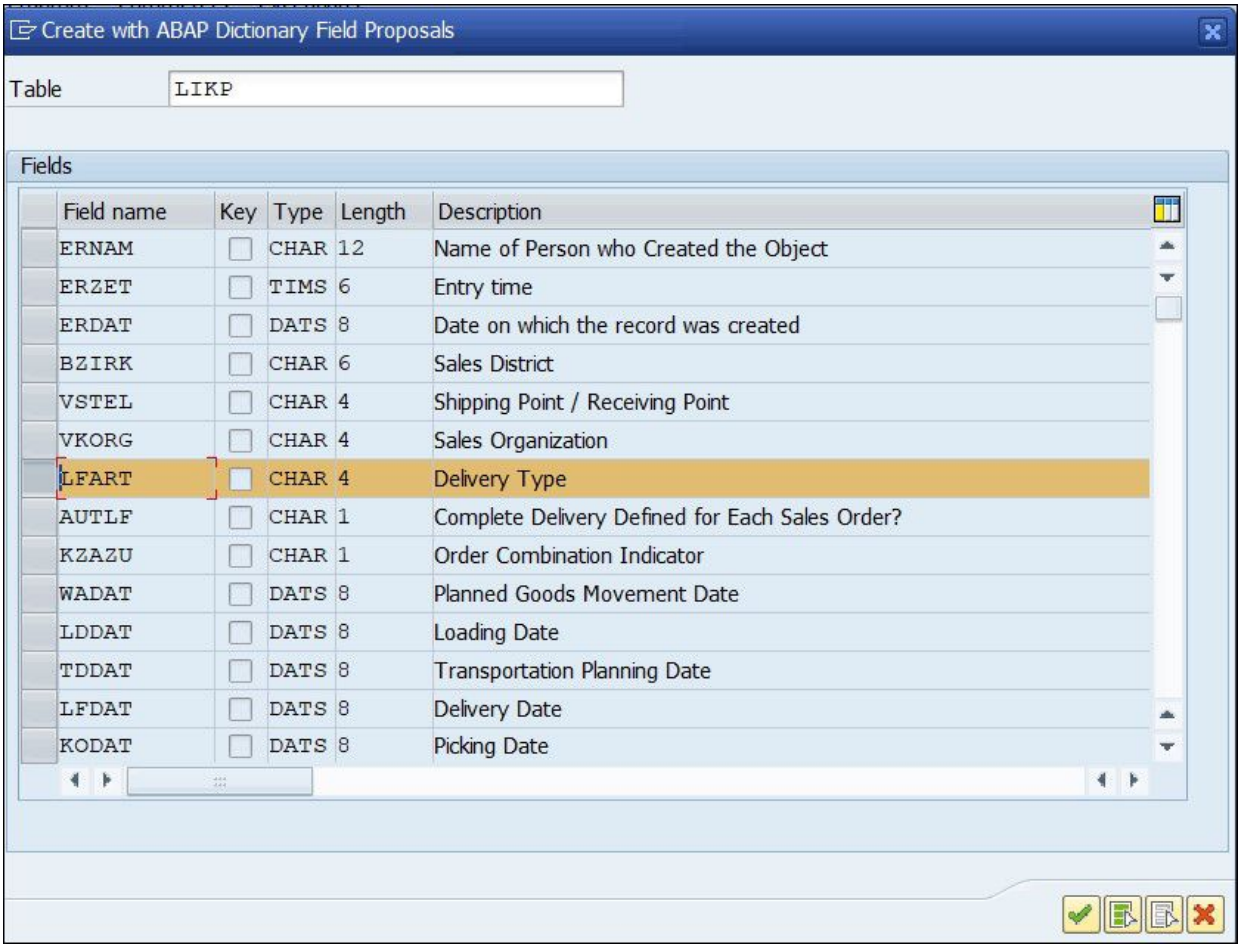


Figure 3.25 Selection of Table Fields for the Database Attribute Definition

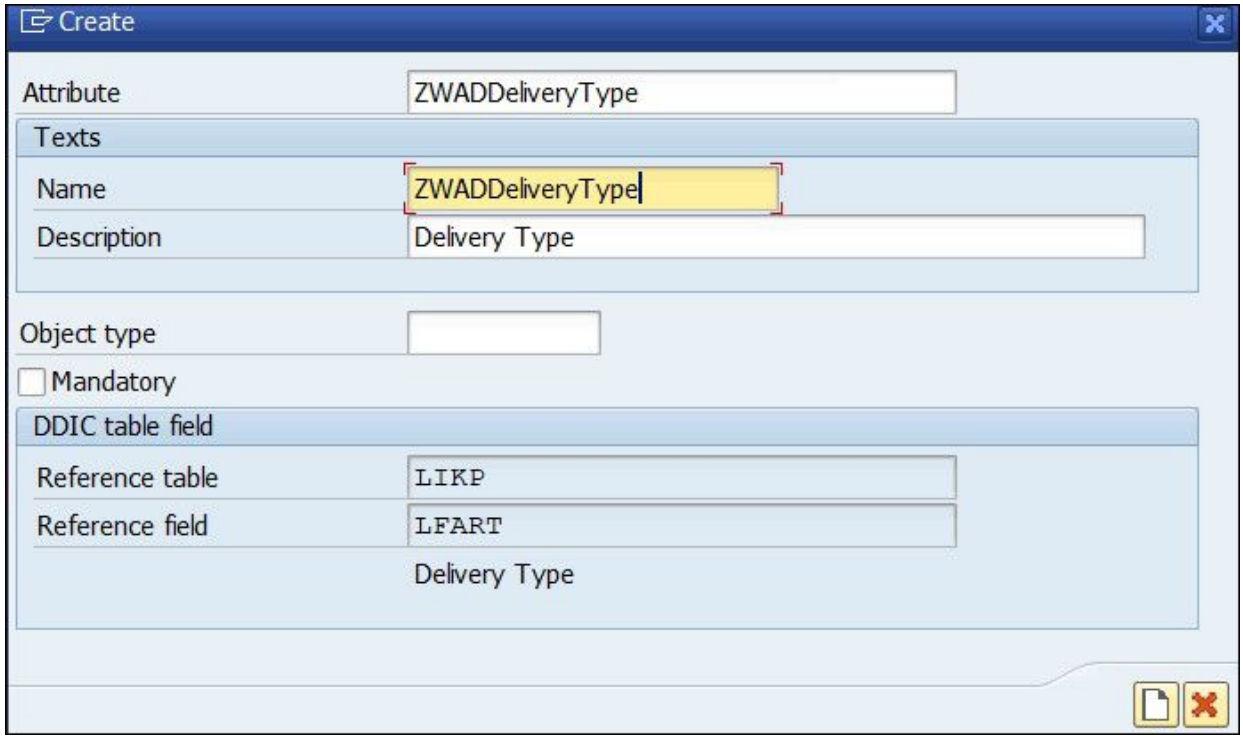


Figure 3.26 Confirm the Database Attribute Name per Naming Standards

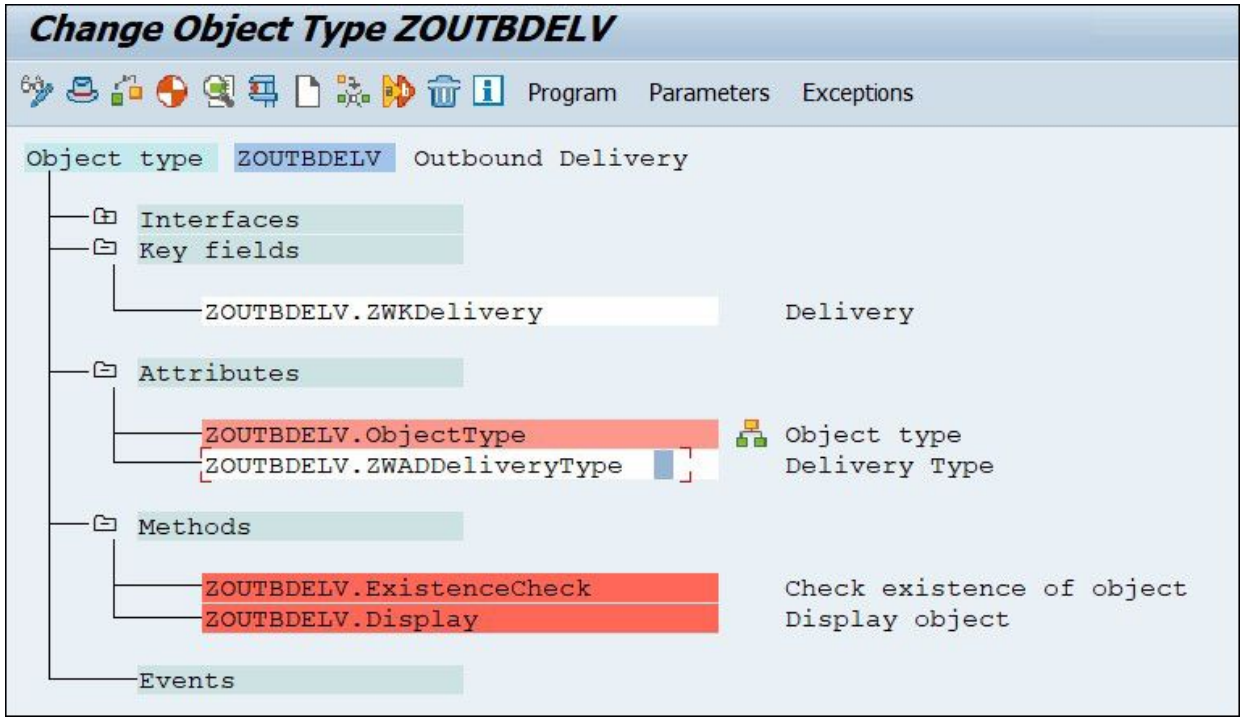


Figure 3.27 Object Type Definition after Creation of the Database Attribute

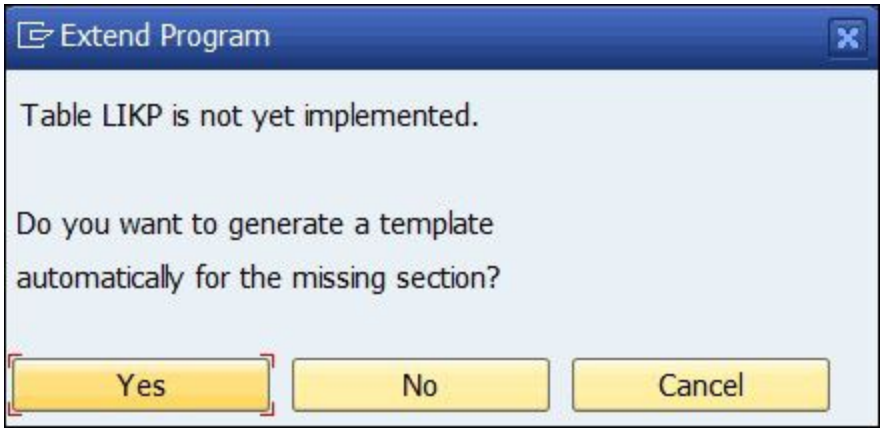


Figure 3.28 Popup to Automatically Generate Template Code for the Database Attribute

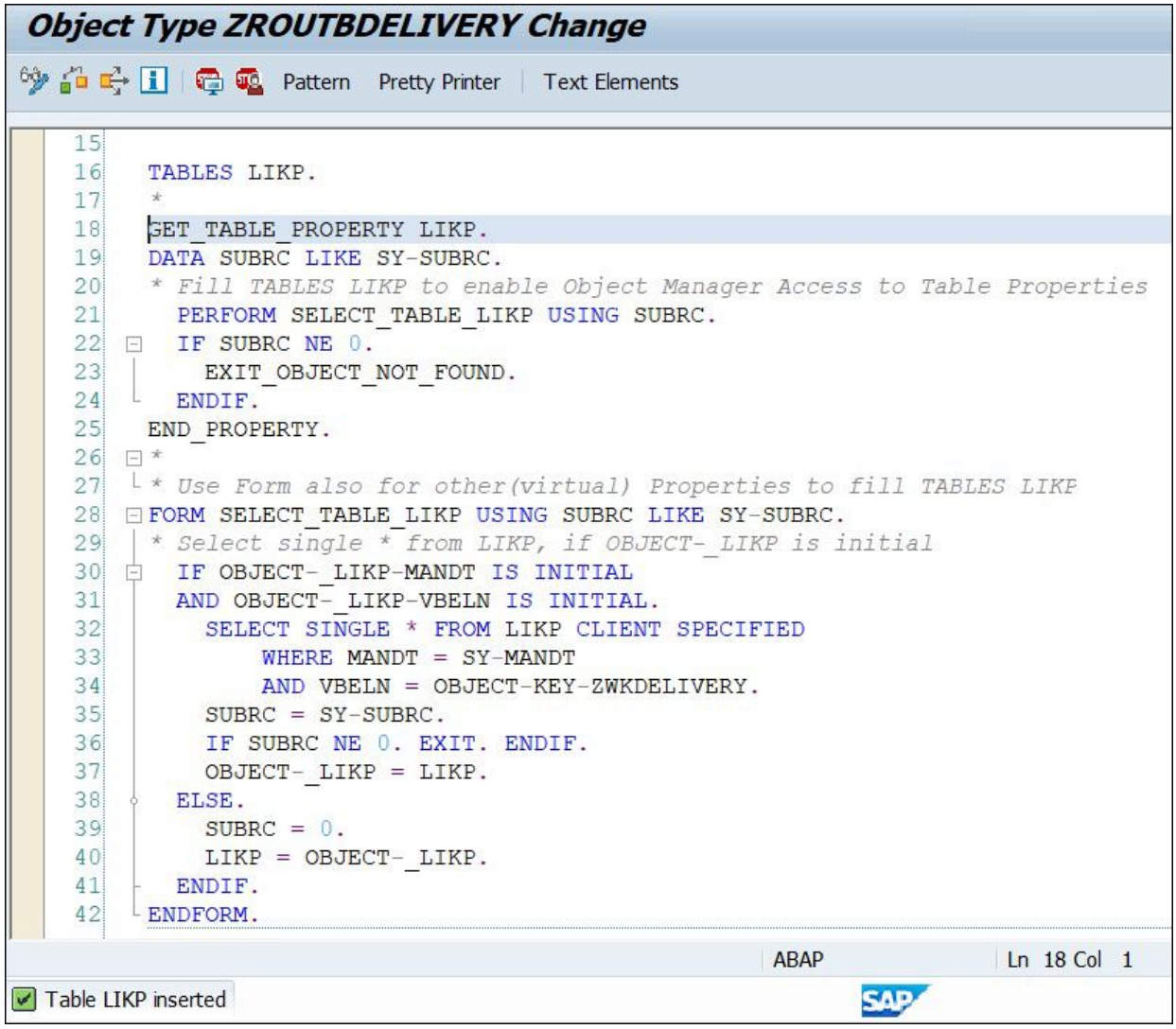


Figure 3.29 View of Automatically Generated Source Code for Database Attribute

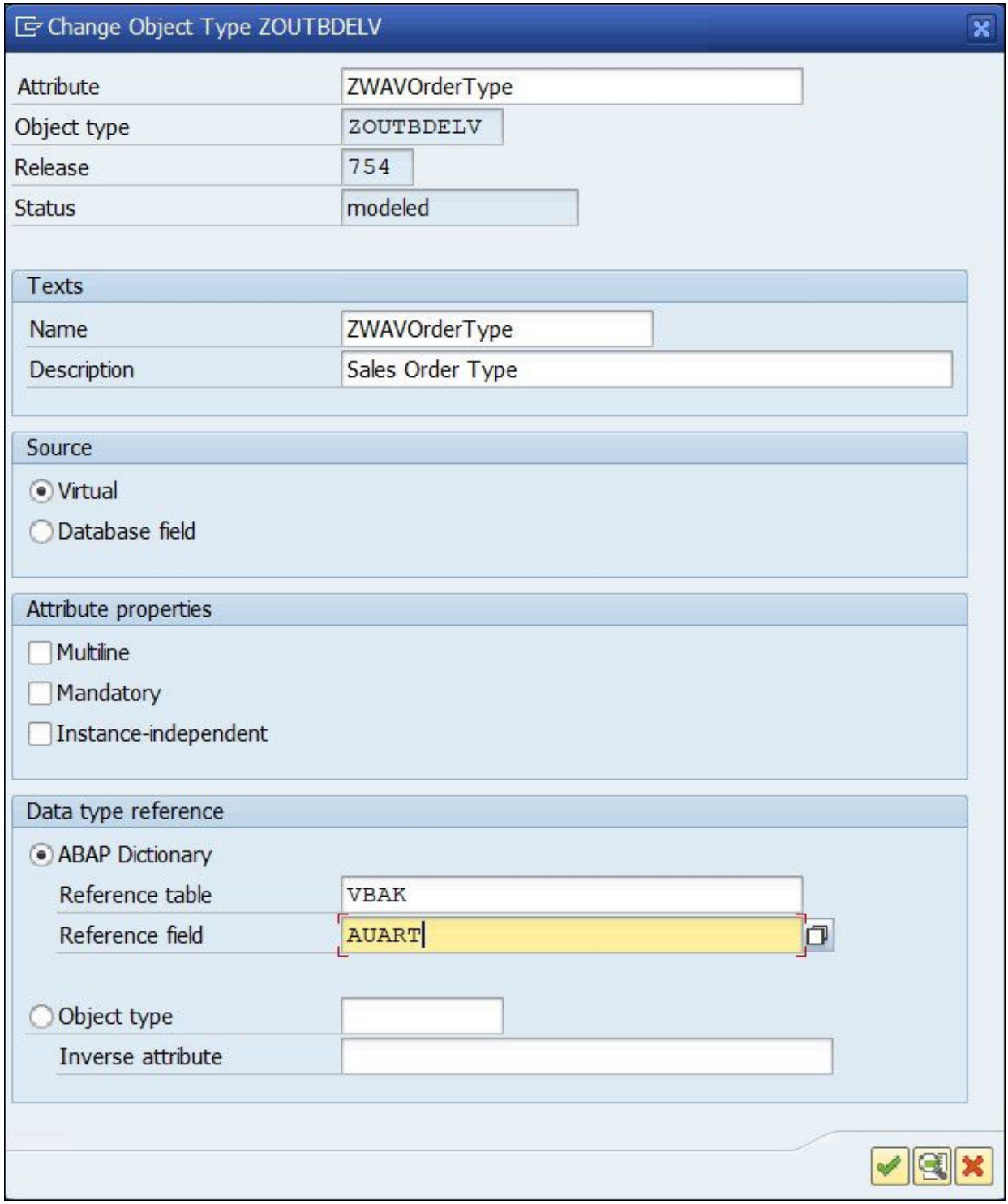

Change Object Type ZOUTBDELV
Attribute
ZWAVOrderType
Object type
ZOUTBDELV
Release
754
Status
modeled
Texts
Name
ZWAVOrderType
Description
Sales Order Type
Source
Virtual
Database field
Attribute properties
Multiline
Mandatory
Instance-independent
Data type reference
ABAP Dictionary
Reference table
VBAK
Reference field
AUART
Object type
Inverse attribute

Figure 3.30 Definition of a Virtual Attribute in the Object Type

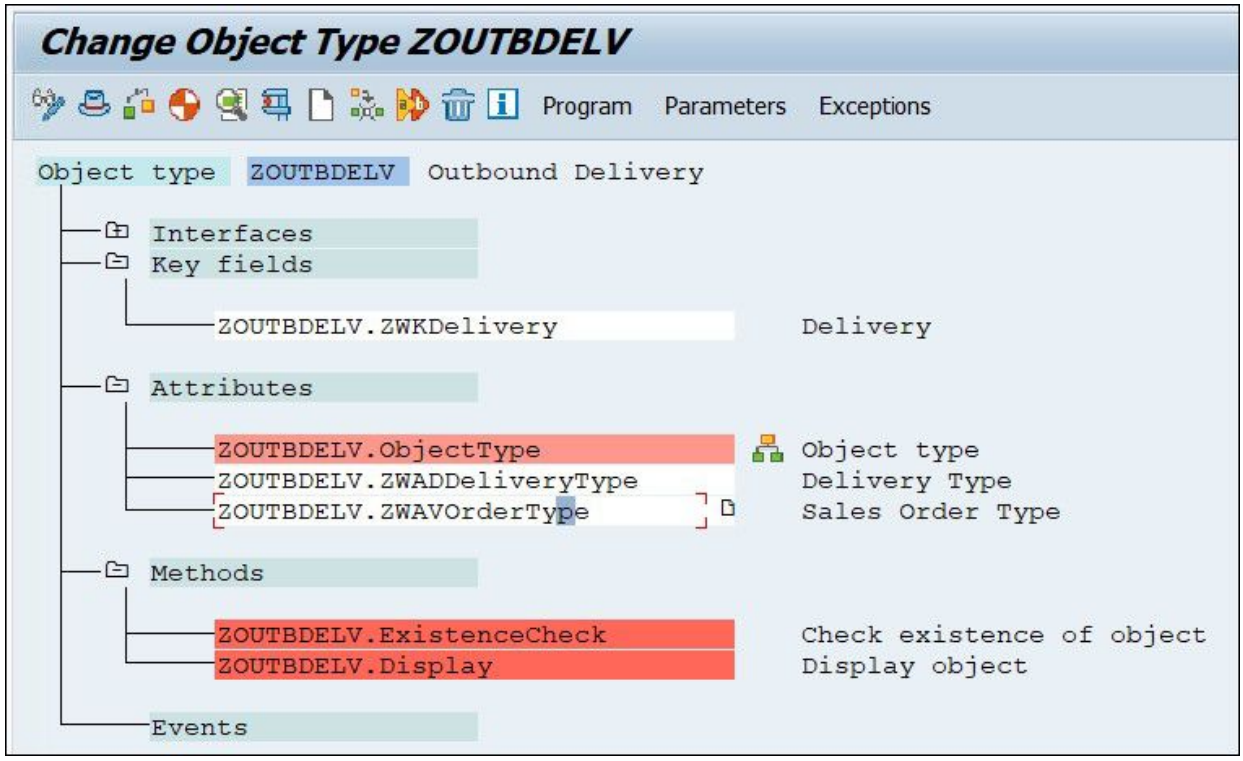


Figure 3.31 View of a Newly Created Virtual Attribute in Modeled Status

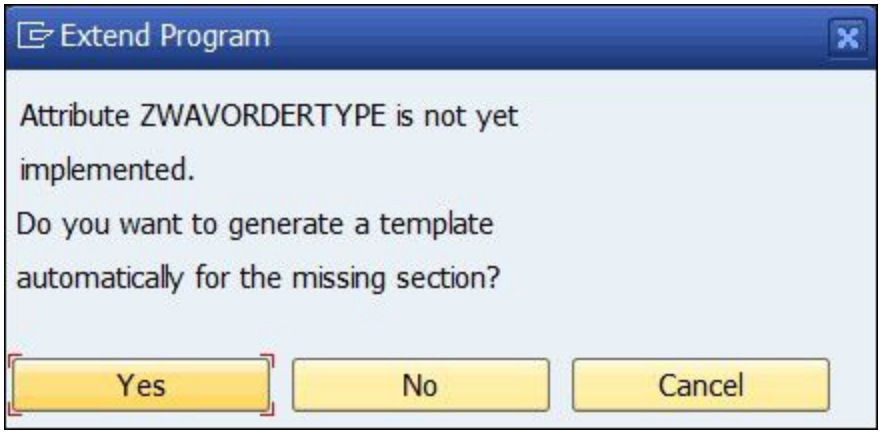


Figure 3.32 Popup Prompt to Generate an Empty Template Section for the Virtual Attribute

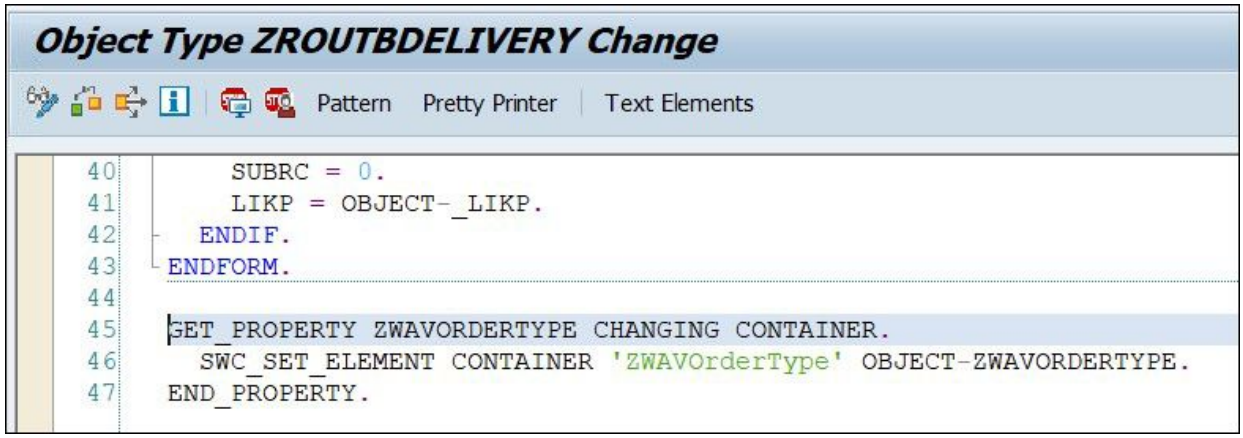


Figure 3.33 Generated Template Code Section for the Virtual Attribute in the BOR Program

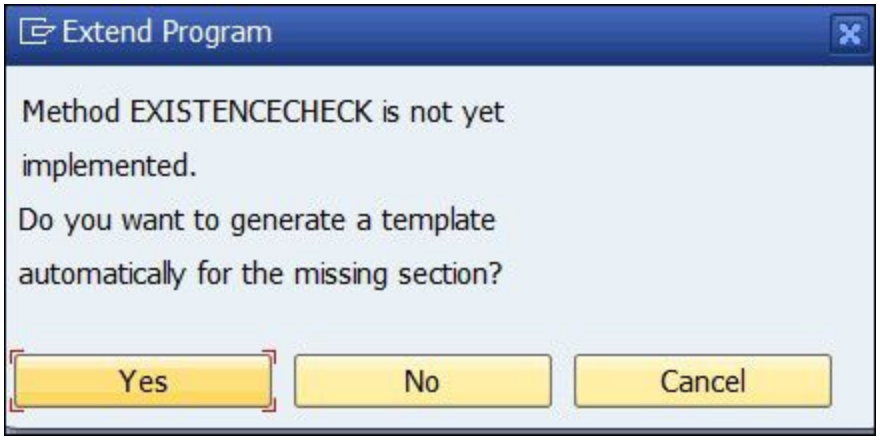


Figure 3.34 Popup Confirmation to Generate a Template Section for Method Implementation

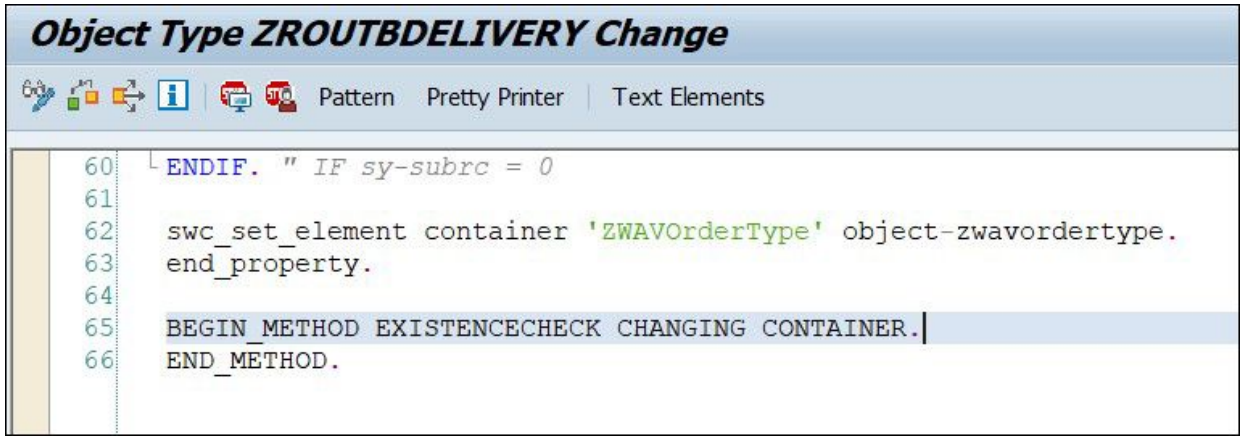


Figure 3.35 Template Section Inserted in the BOR Program with Method Implementation

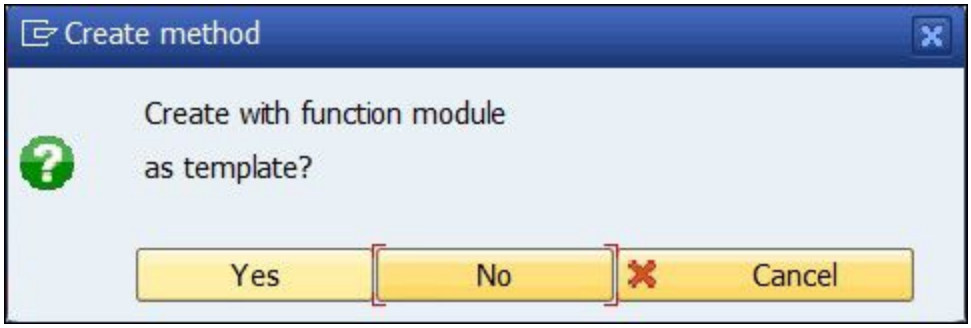


Figure 3.36 Popup to Confirm if the Method Should Be Created with a Function Module Call

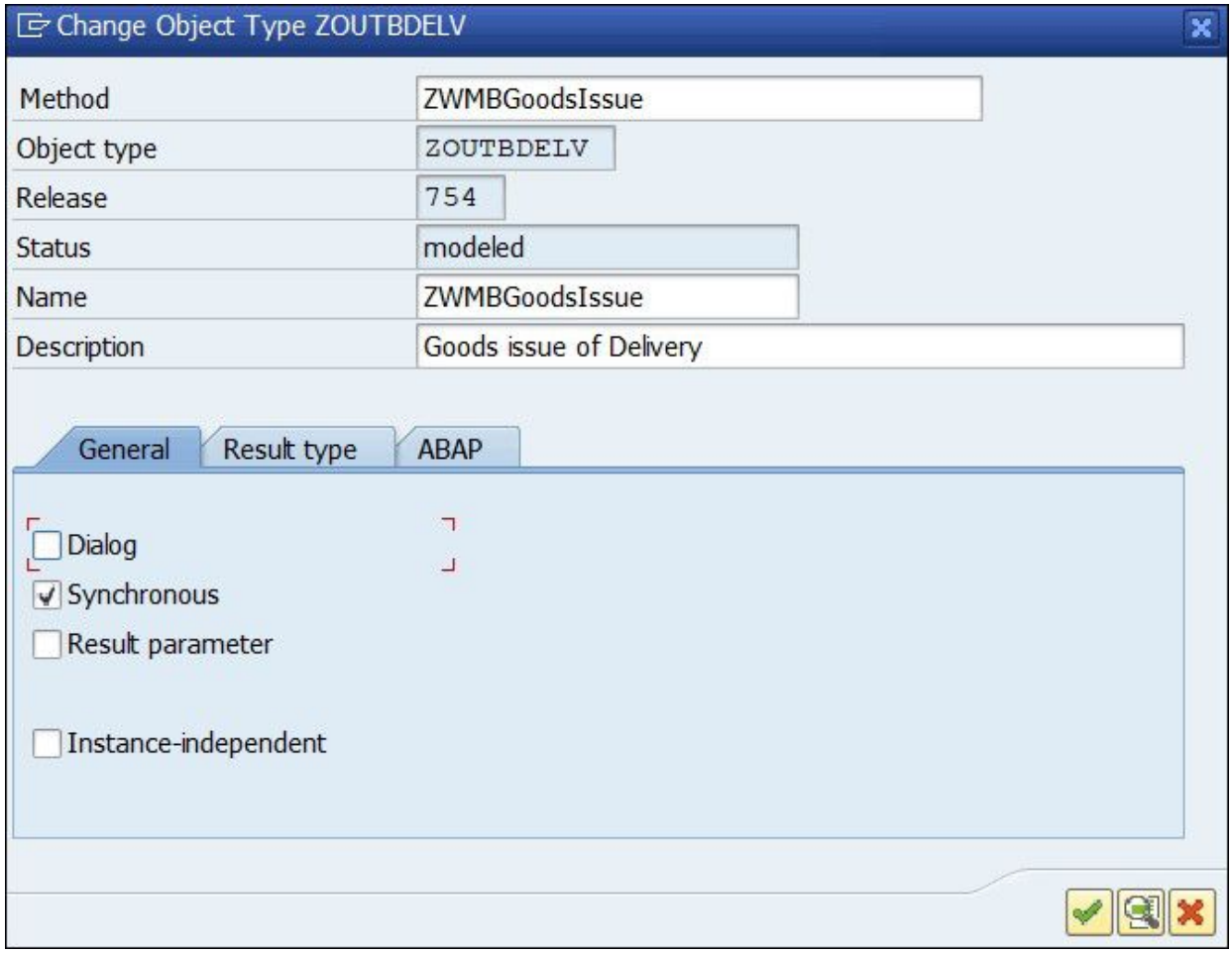


Figure 3.37 Popup Screen for Method Attributes Selection

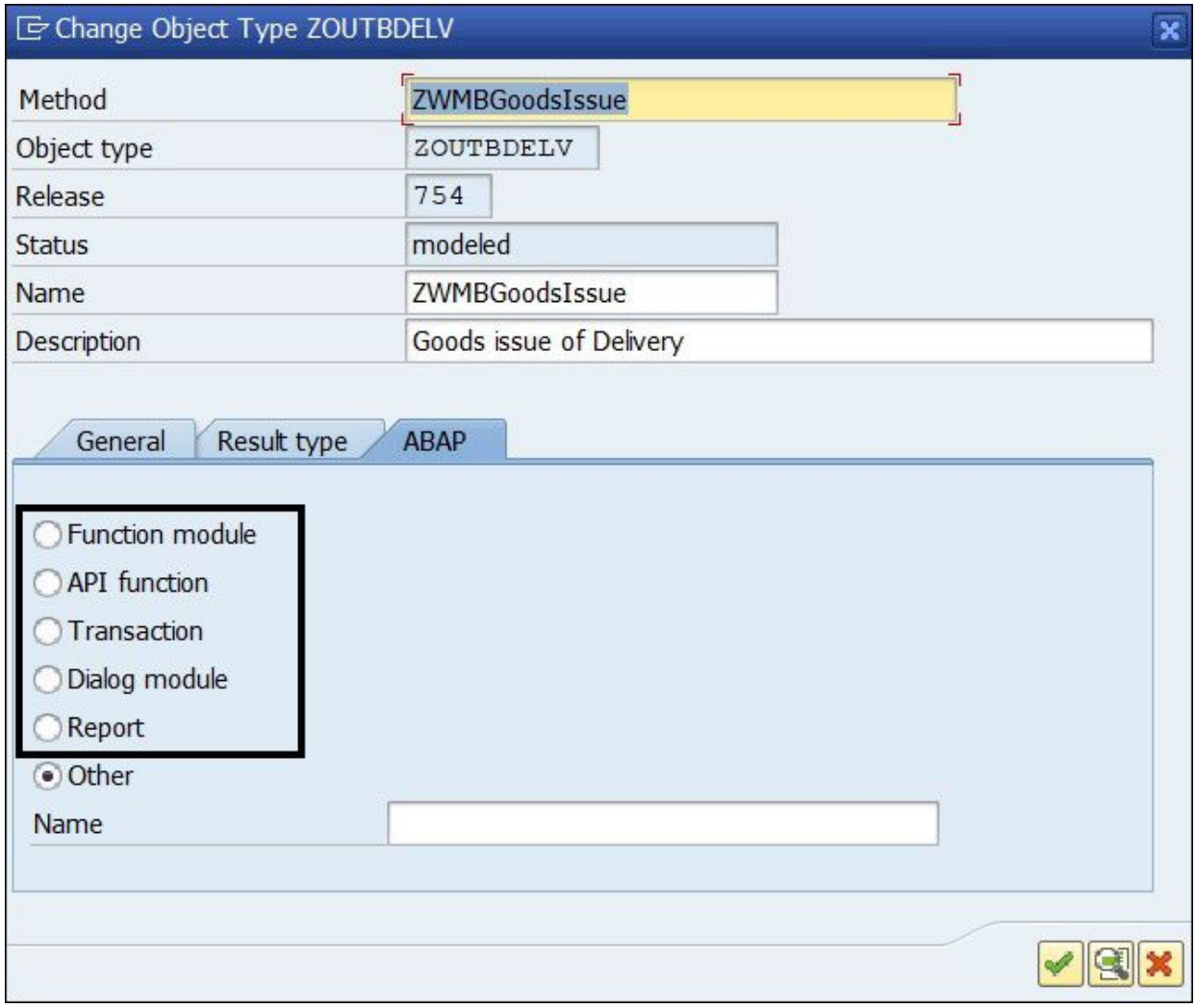


Figure 3.38 ABAP Tab Attributes for Method Creation

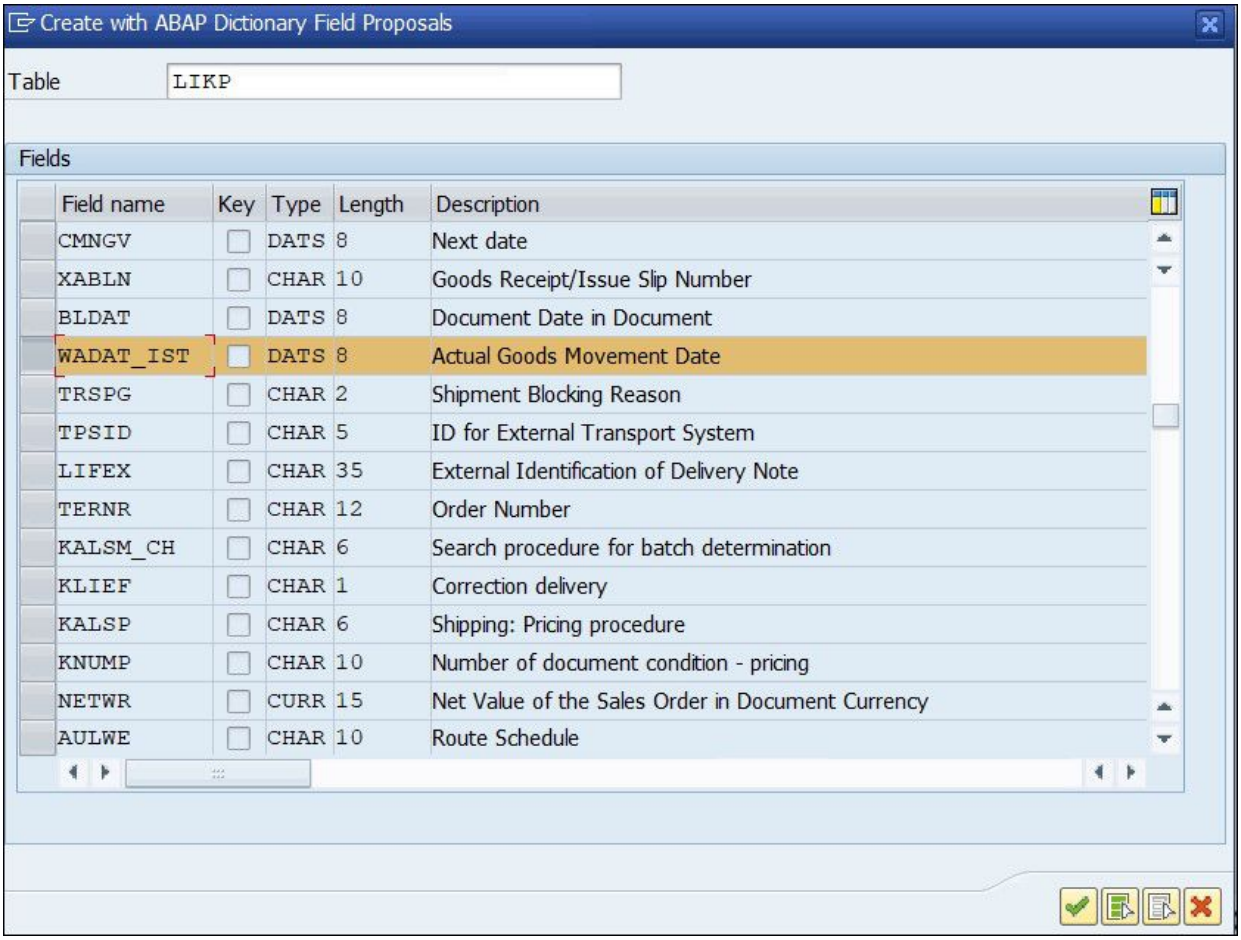


Figure 3.39 Select Table Field for Method Parameter Creation

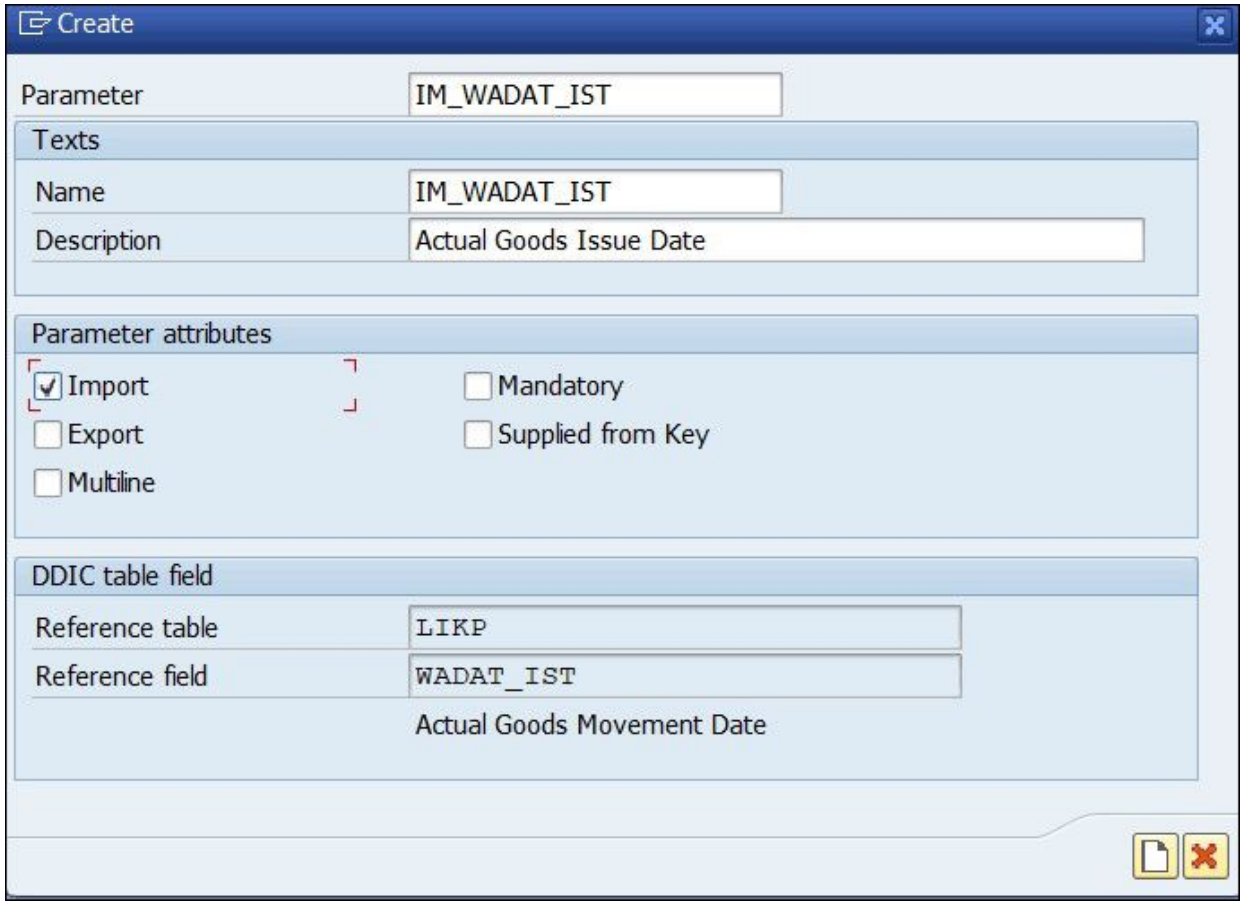


Figure 3.40 Entering the Method Parameter Name and Selecting Attributes

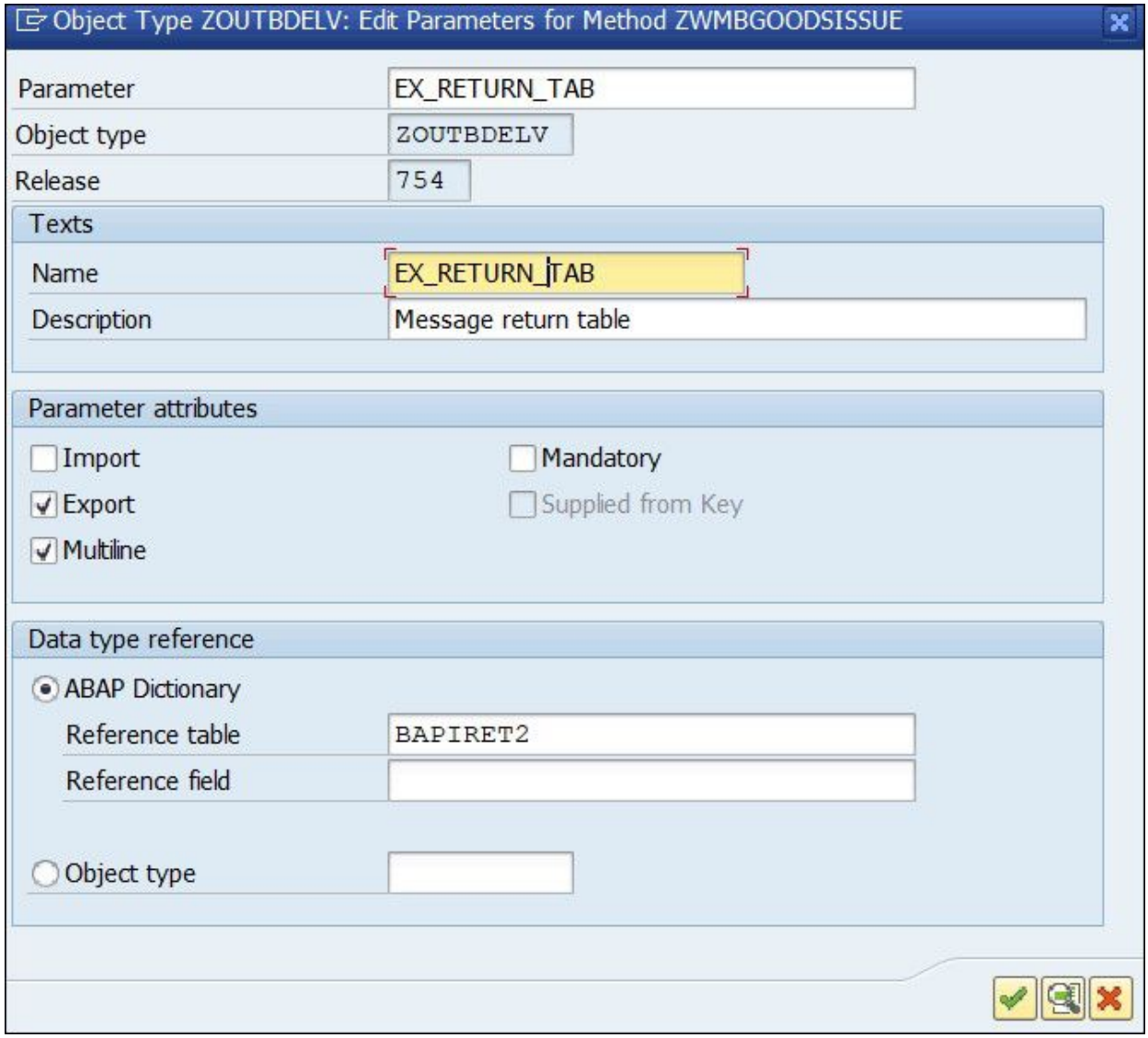


Figure 3.41 Creation of the Method Export Parameter

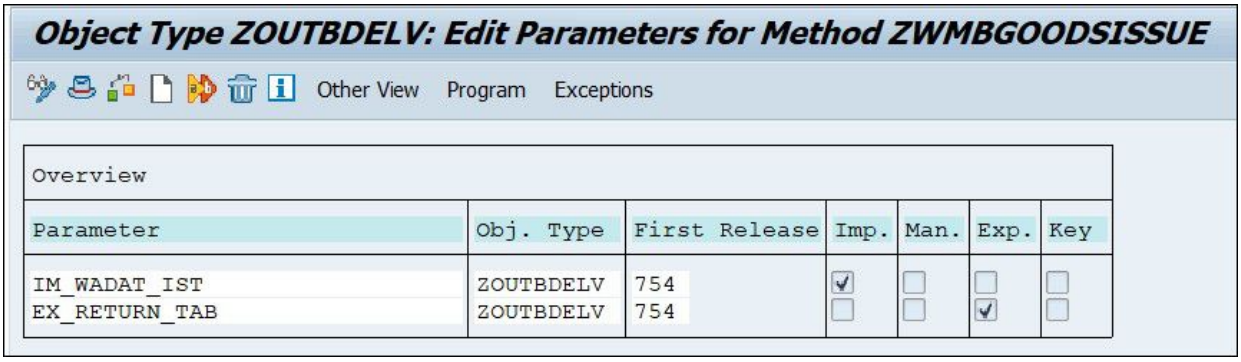


Figure 3.42 View of Method Parameters

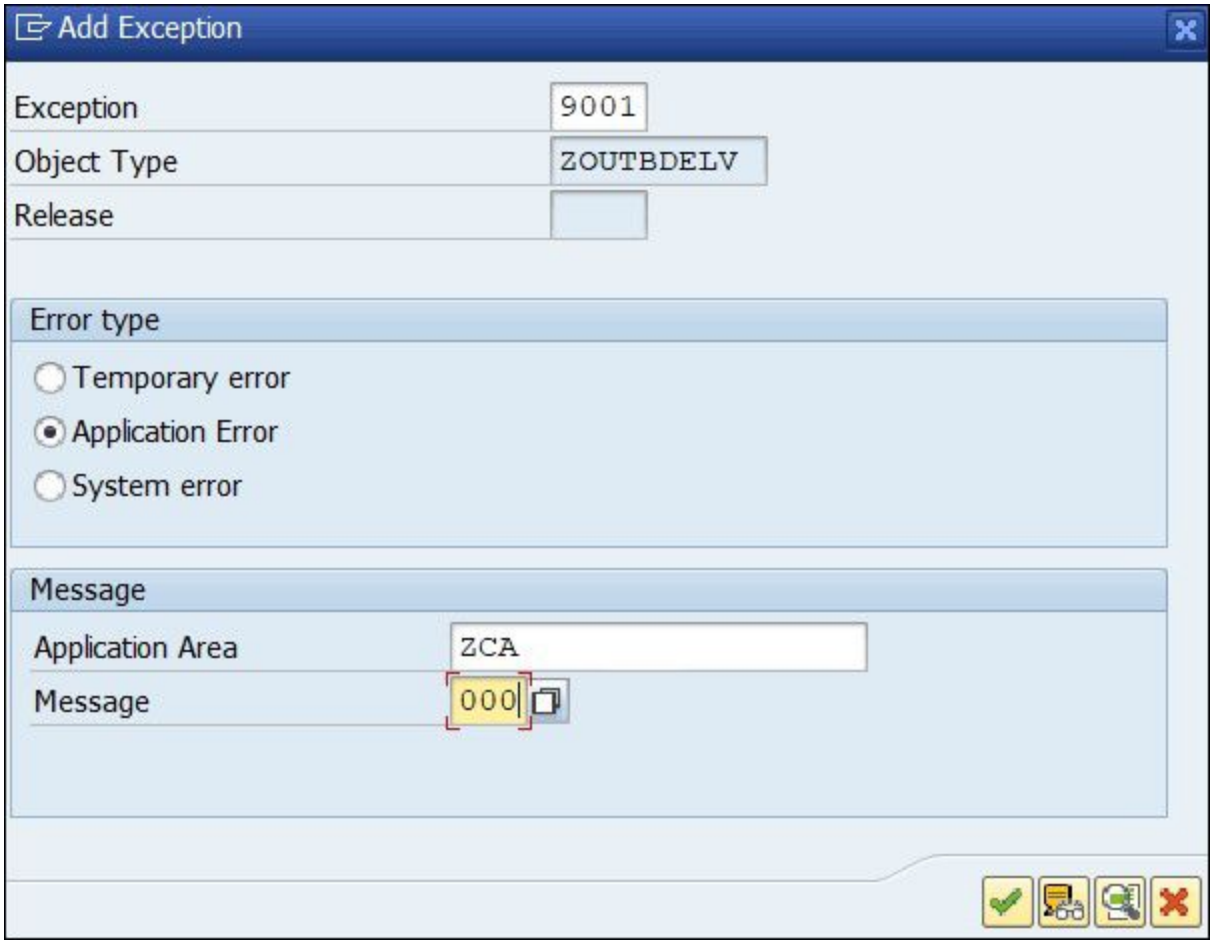


Figure 3.43 Selection of Exception Definition Attributes

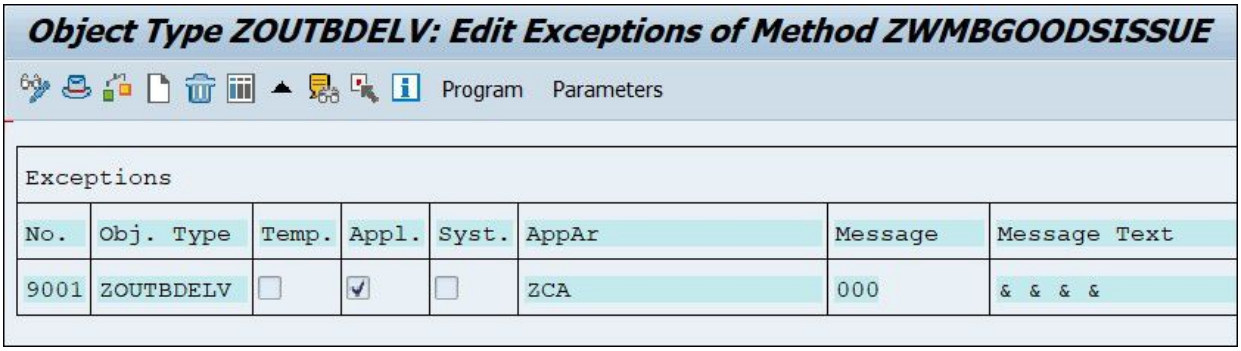


Figure 3.44 View of Method Exception Definition

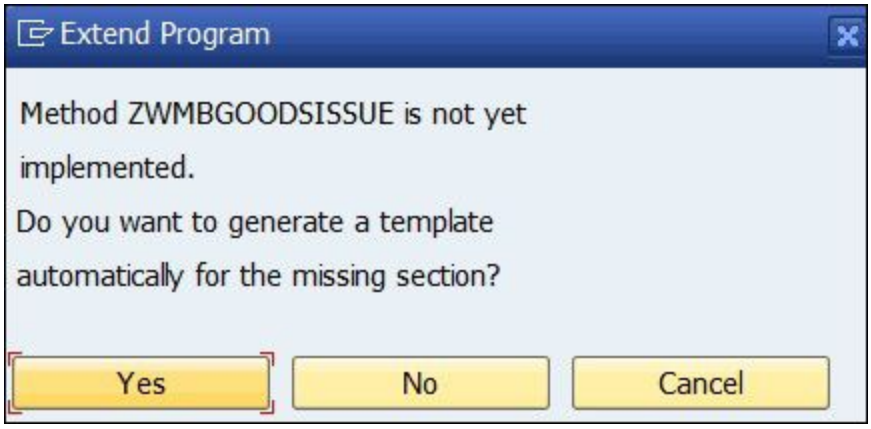


Figure 3.45 Popup Prompt to Generate Method Implementation in the BOR Program

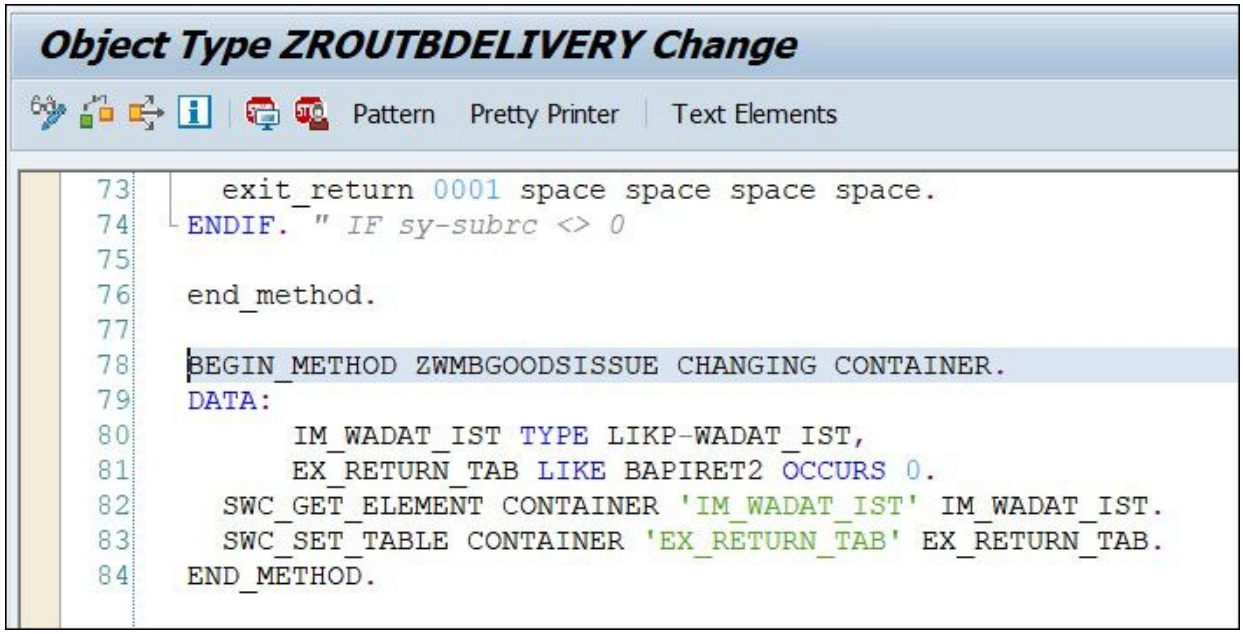


Figure 3.46 Template Code for Method Implementation Added into the BOR Program

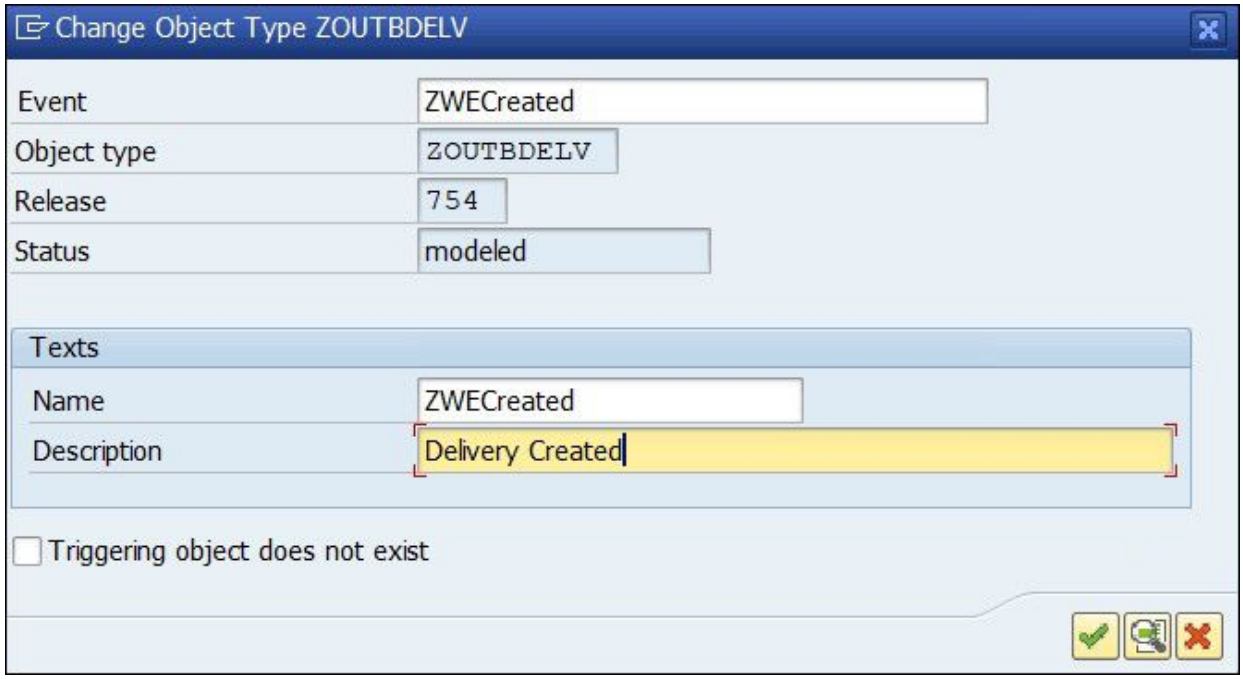


Figure 3.47 Event Definition in the Business Object Type

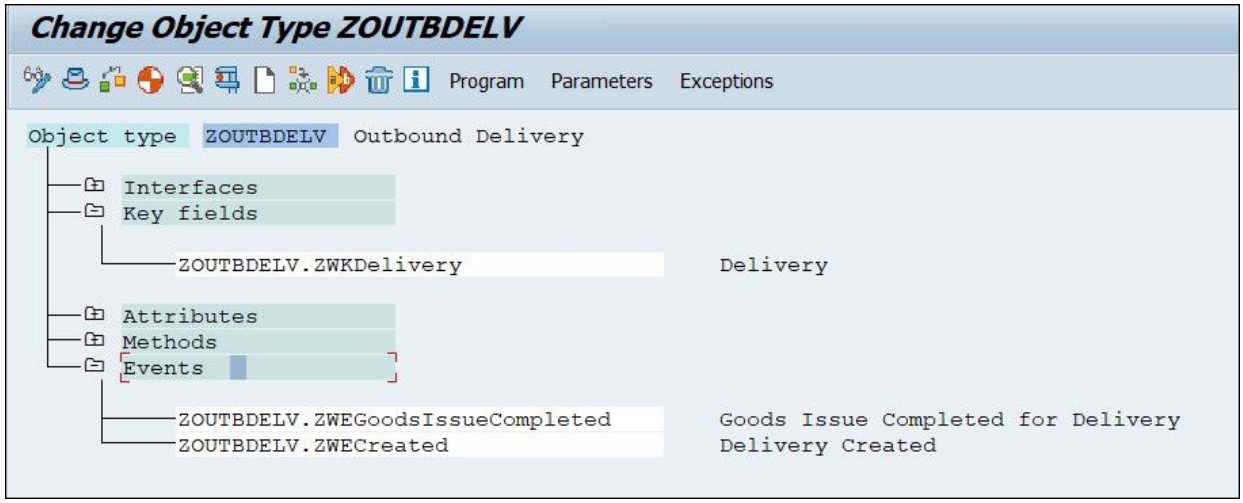


Figure 3.48 Event Definition in the BOR Object Type

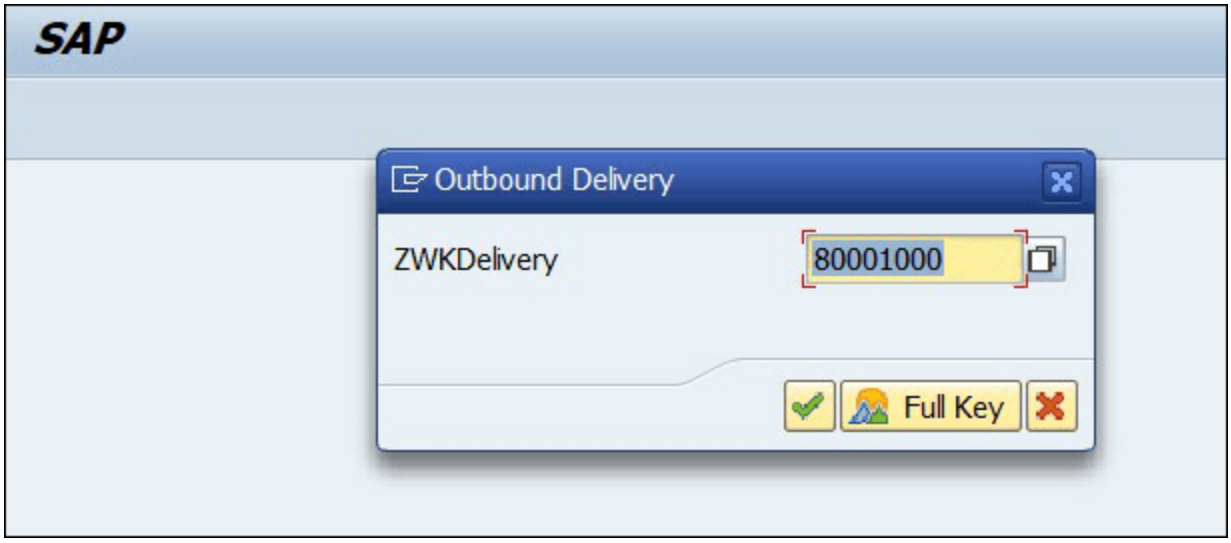


Figure 3.49 Testing a BOR Object Type in Transaction SWO1

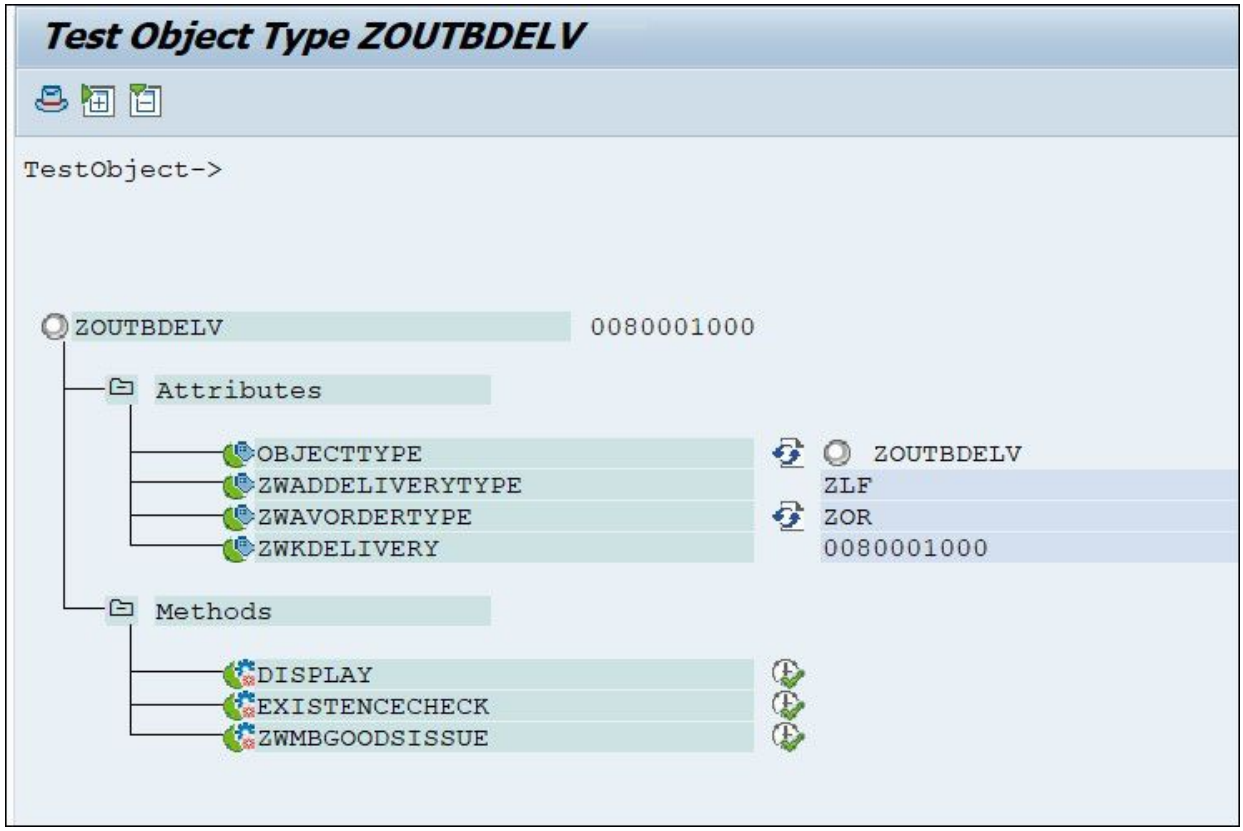


Figure 3.50 BOR Type Object Instance Displaying Attributes and Methods Available for Execution

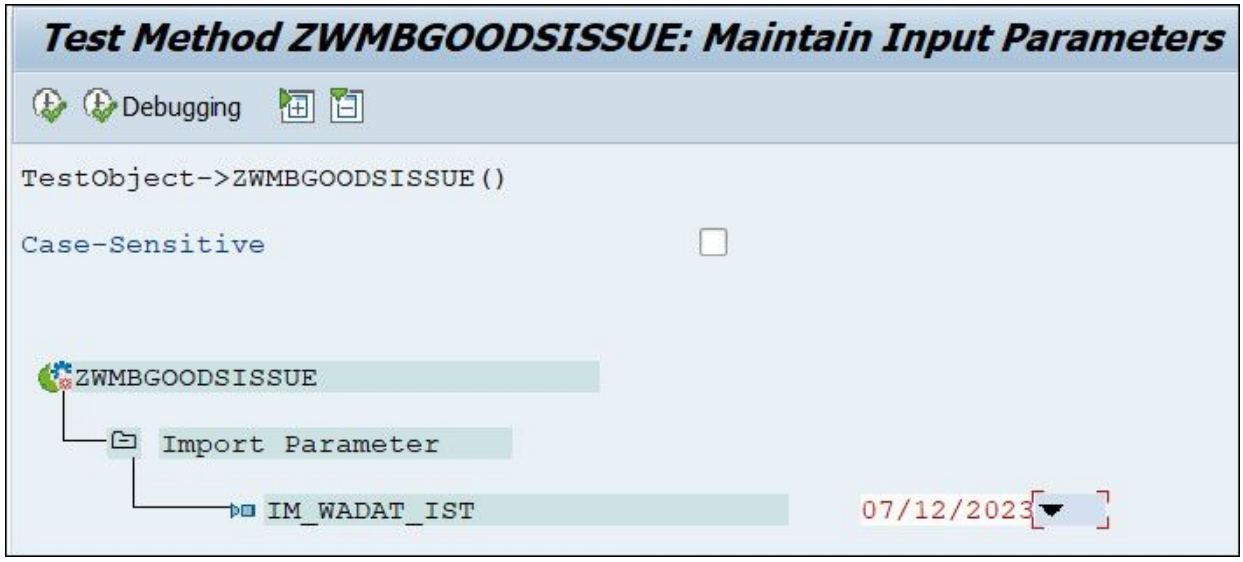


Figure 3.51 Testing a BOR Method

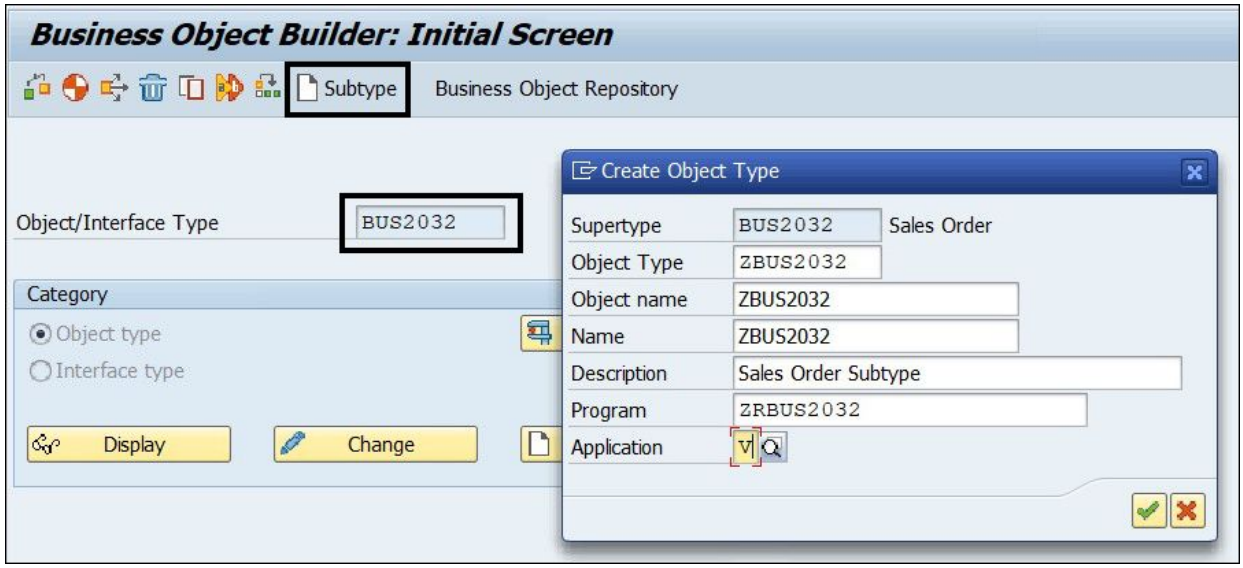


Figure 3.52 Creating a BOR Subtype Definition from Transaction SWO1

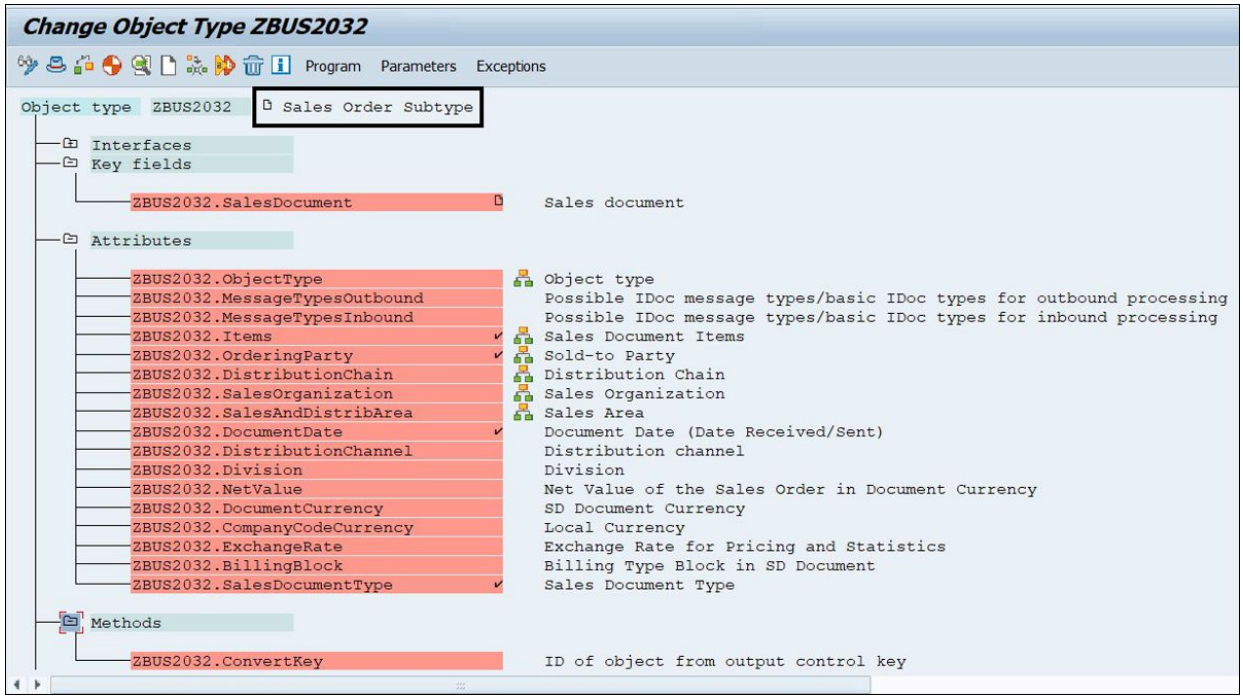


Figure 3.53 Subtype Definition Showing Inherited Components and Header Release Status

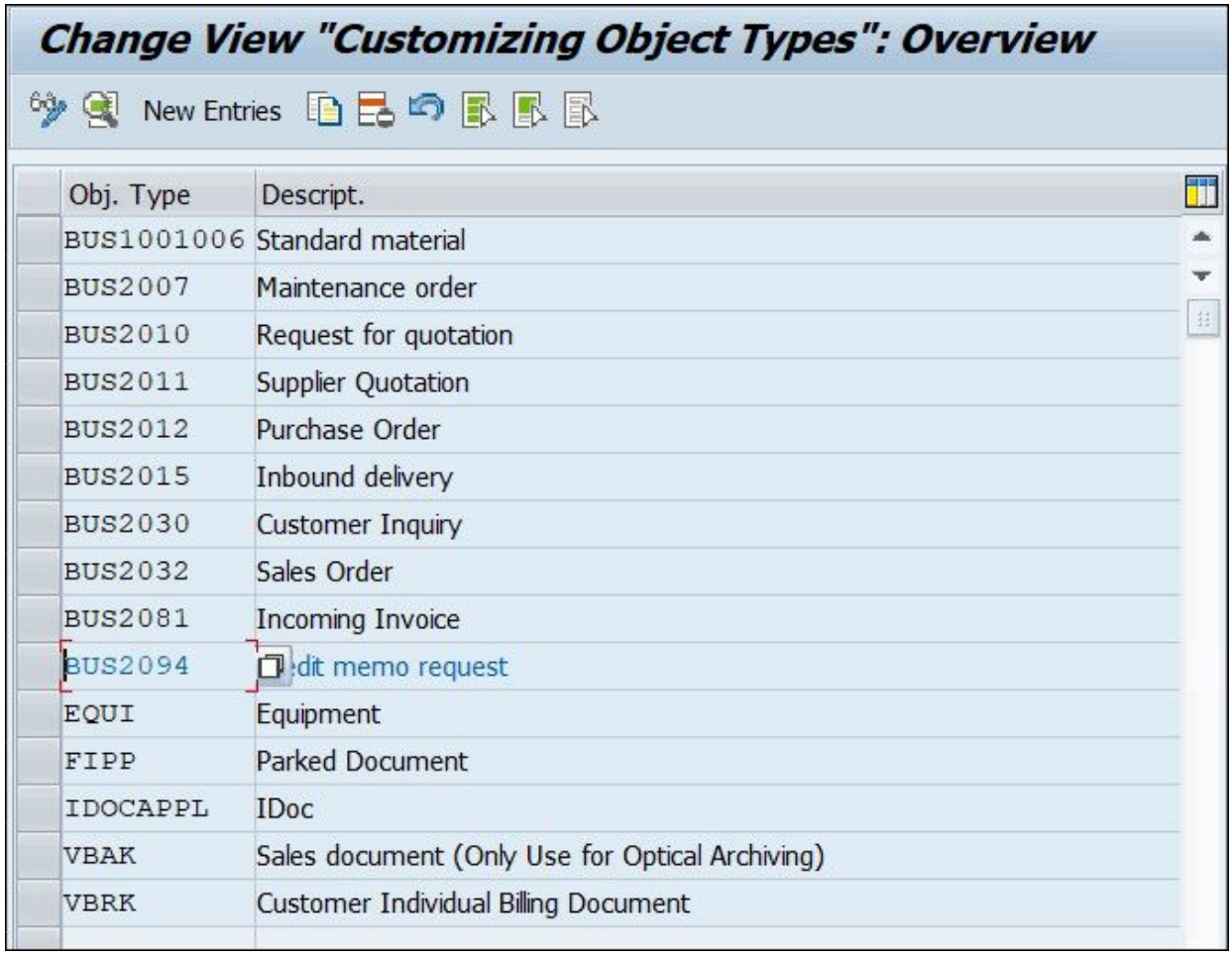


Figure 3.54 Delegation Configuration Showing the Supertypes Maintained

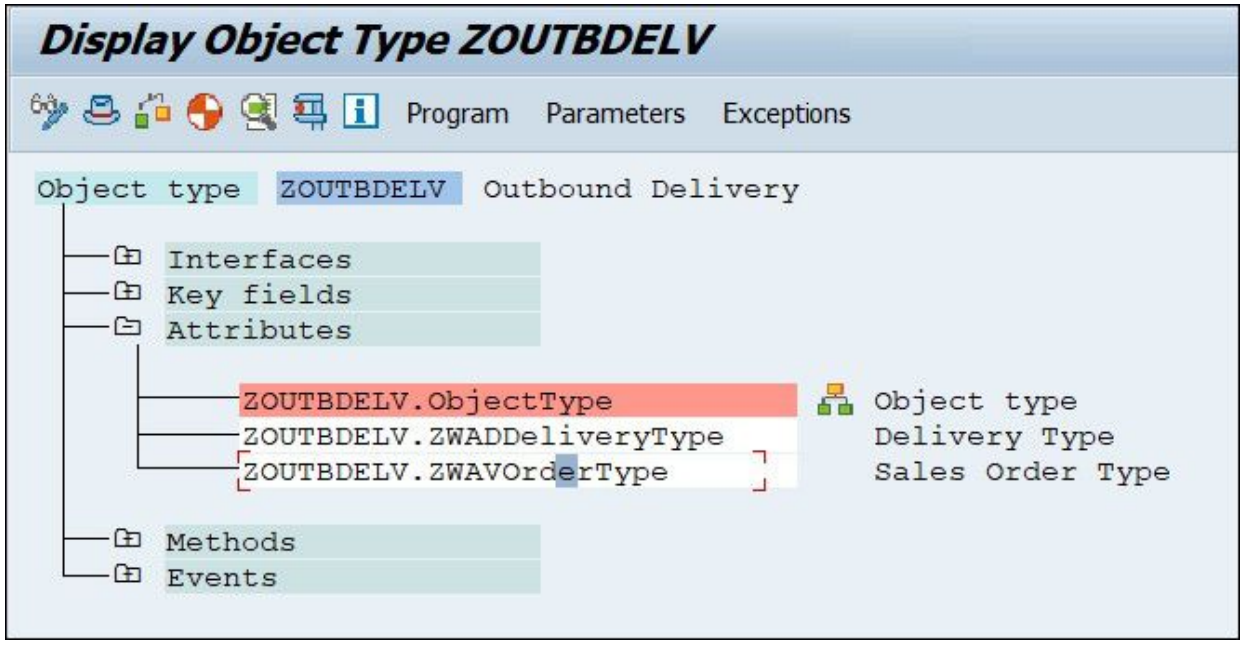


Figure 3.55 Virtual Attribute ZWAVOrderType Added in the Object Type Definition

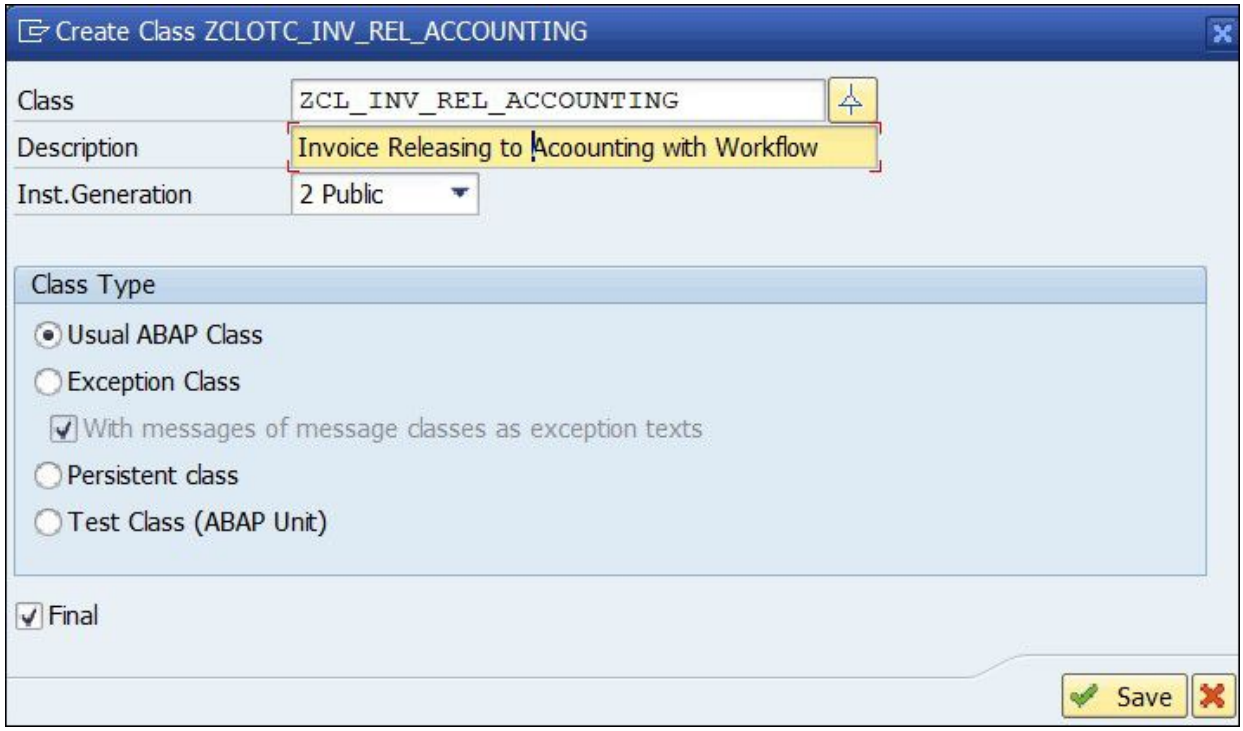


Figure 3.56 Creation of a New Normal ABAP Class from Transaction SE24

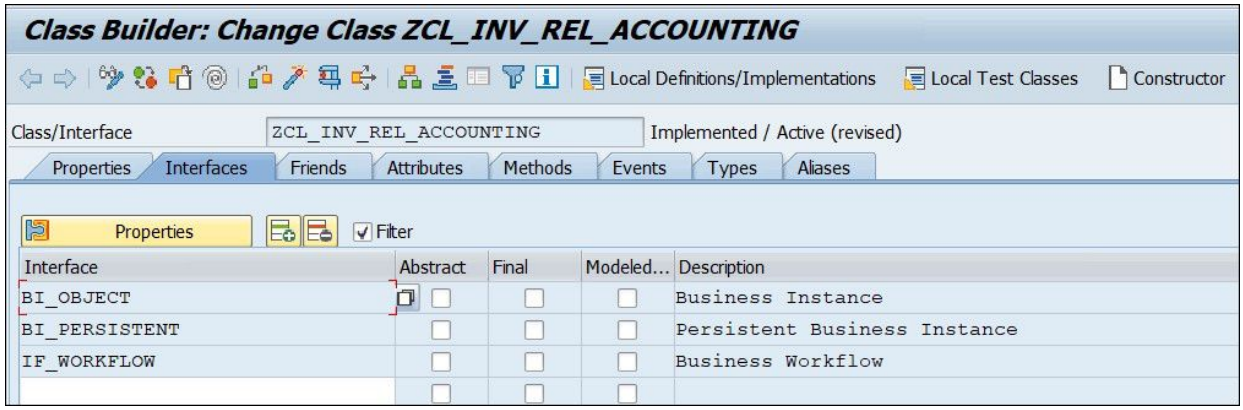


Figure 3.57 Addition of Interface IF_WORKFLOW to the Workflow Class

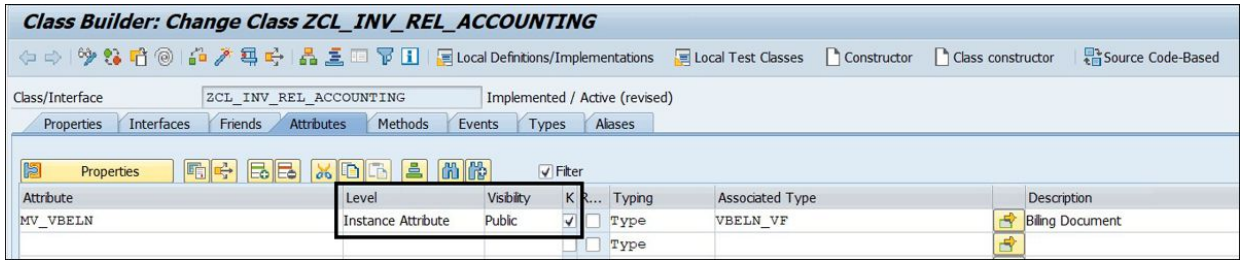


Figure 3.58 View of the Key Attribute Definition in the Workflow Business Class

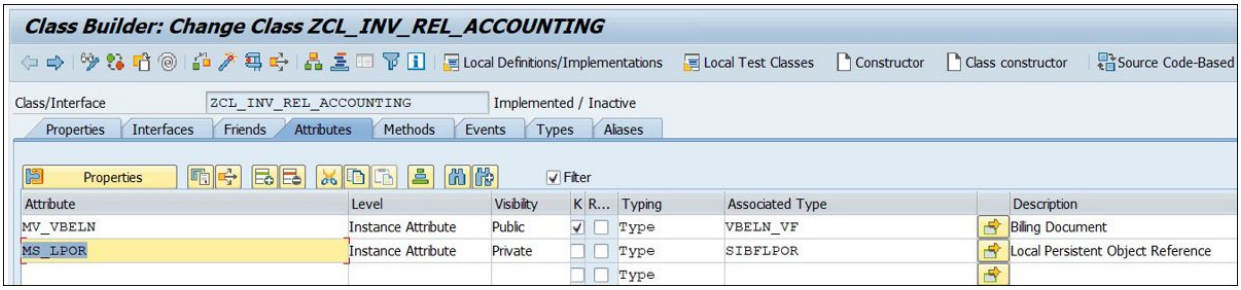


Figure 3.59 Creation of the LPOR Attribute in the Workflow Business Class

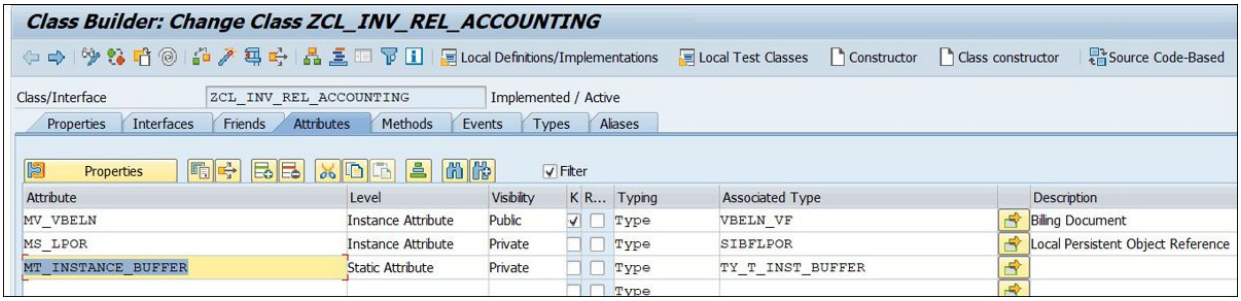


Figure 3.60 View of the Instance Buffer Attribute in the Workflow Business Class

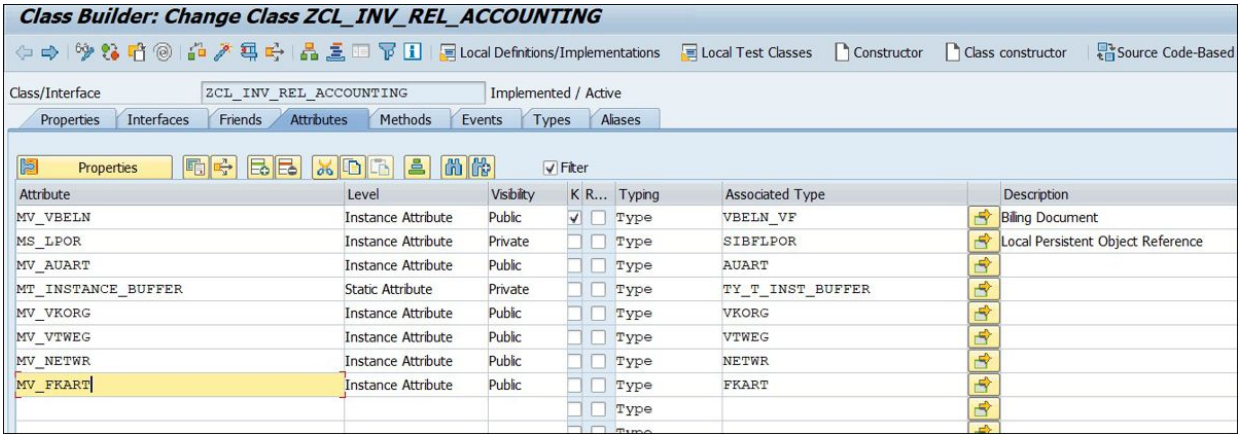


Figure 3.61 Overall View of Attributes in the Workflow Business Class

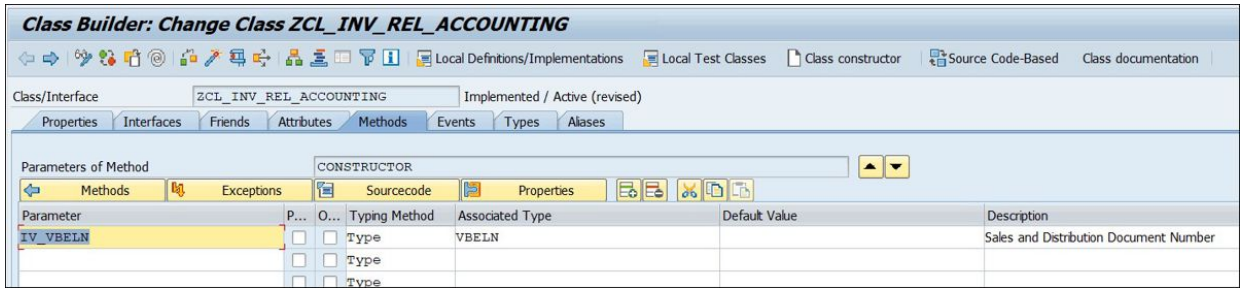


Figure 3.62 Creating the Import Parameter(s) in the Constructor Method of the Workflow Business Class

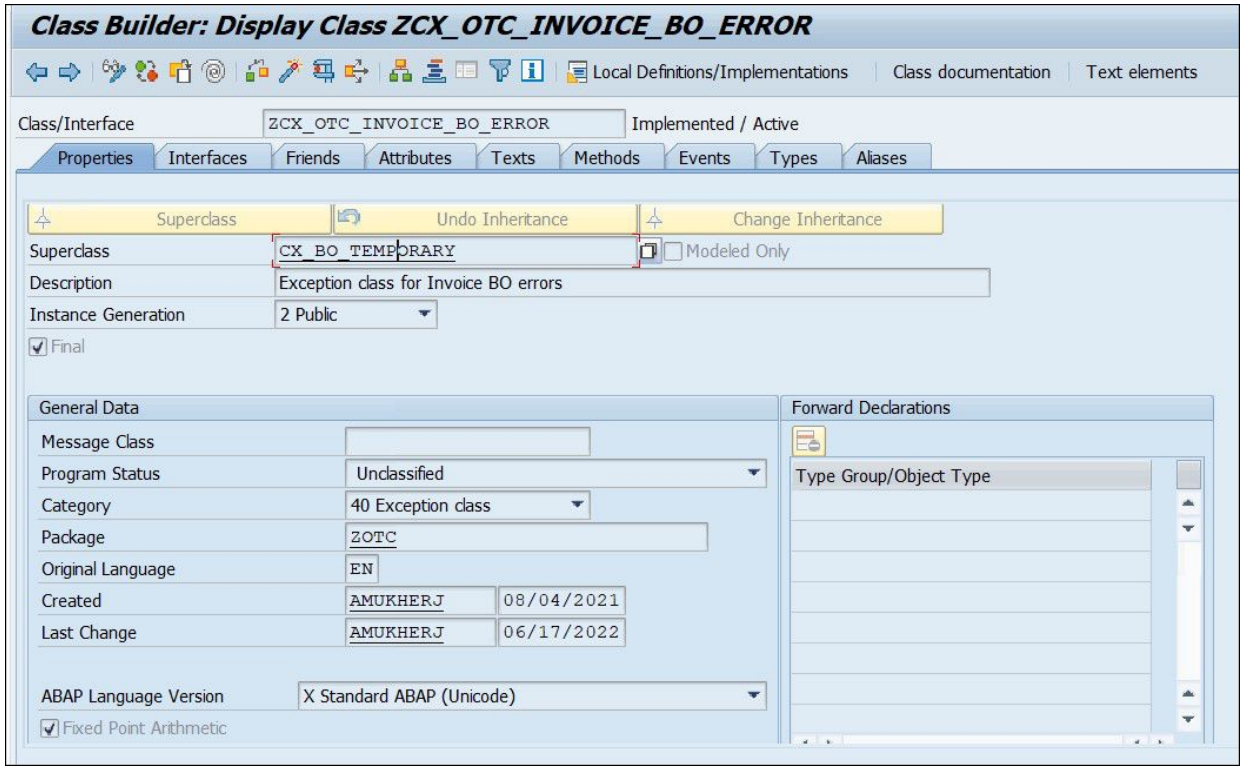


Figure 3.63 Creation of a Custom Workflow Exception Class Inheriting from a Standard Exception Class

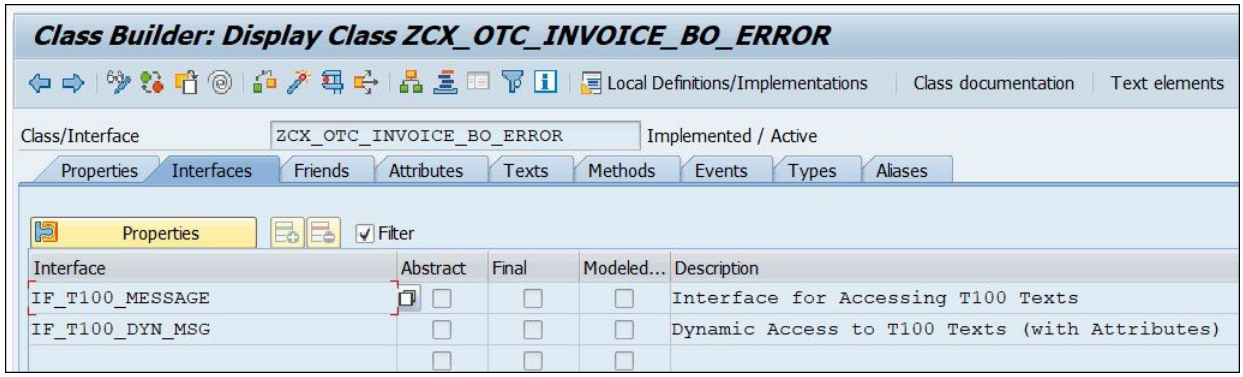


Figure 3.64 Add the Message Text Interfaces in the Custom Exception Class

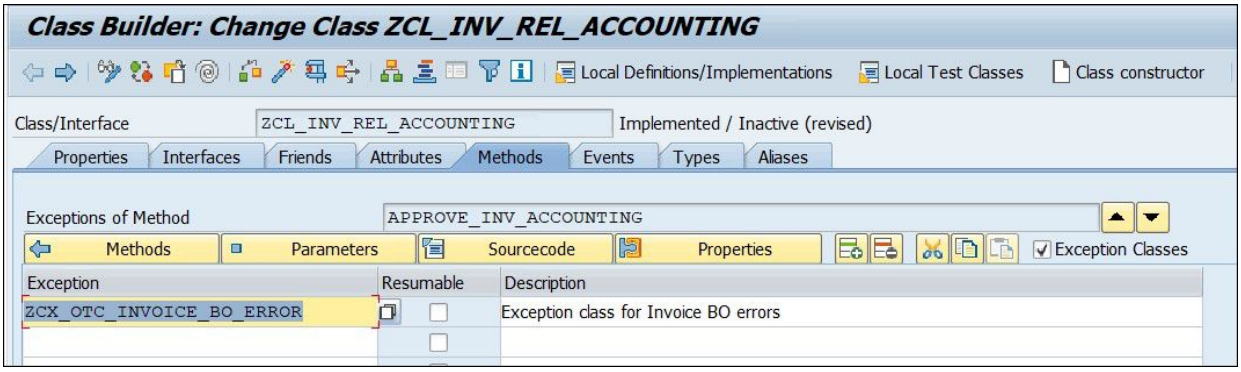


Figure 3.65 Add the Exception Class in the Exceptions of Method Tab

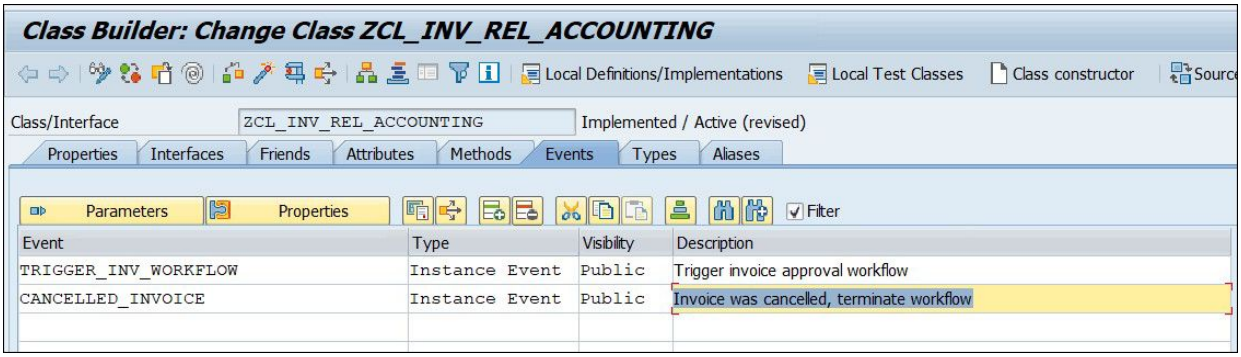


Figure 3.66 View of Events Added in the Workflow Business Class

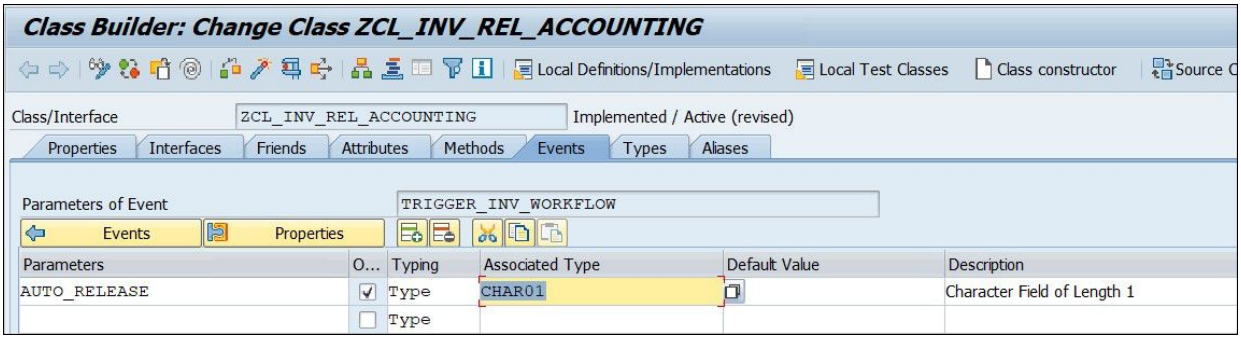


Figure 3.67 Creating an Event Parameter in the Triggering Event of the Workflow

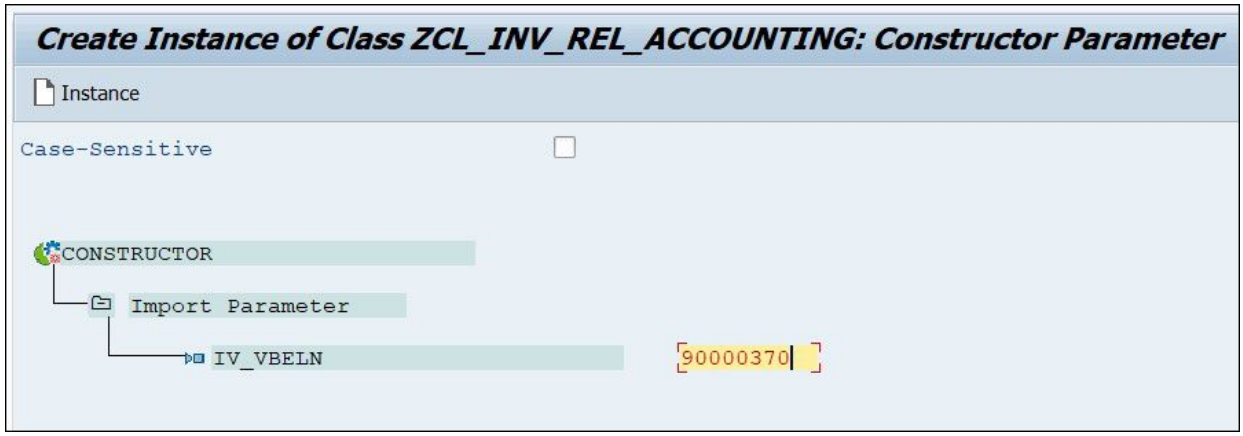


Figure 3.68 Unit Test Workflow Class from Transaction SE24

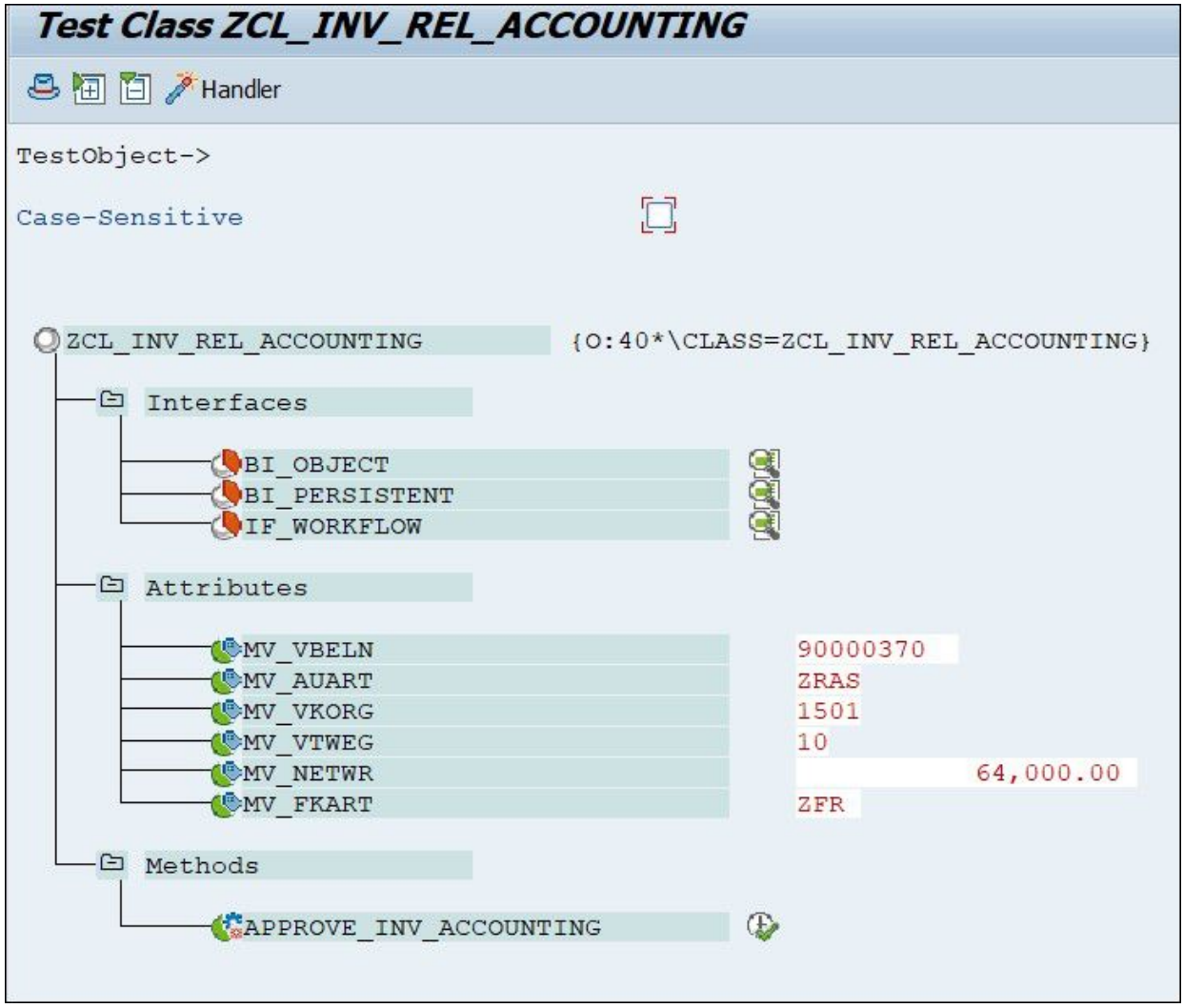


Figure 3.69 Instance Creation of the Workflow Class

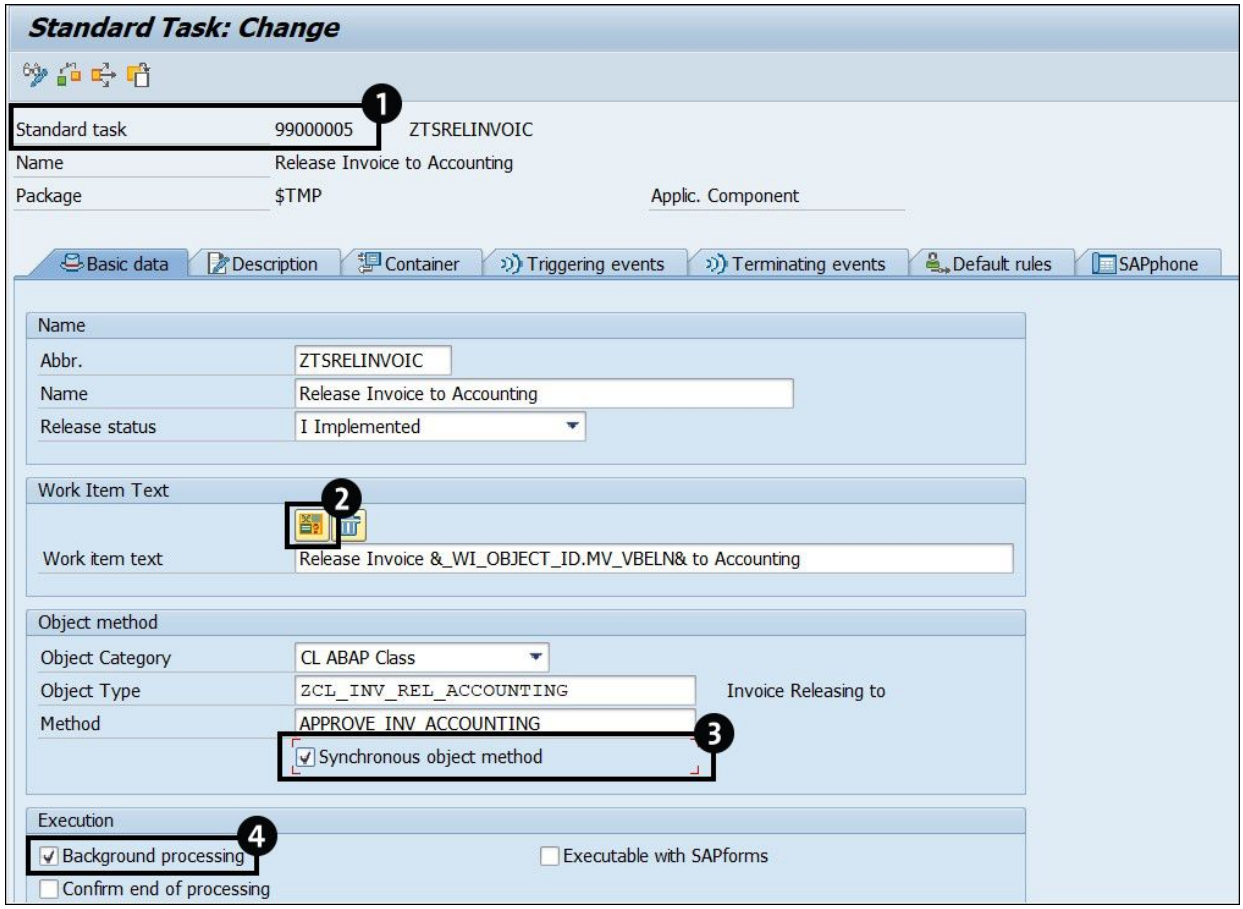


Figure 3.70 Basic Data Definition of a Standard Task for the Workflow

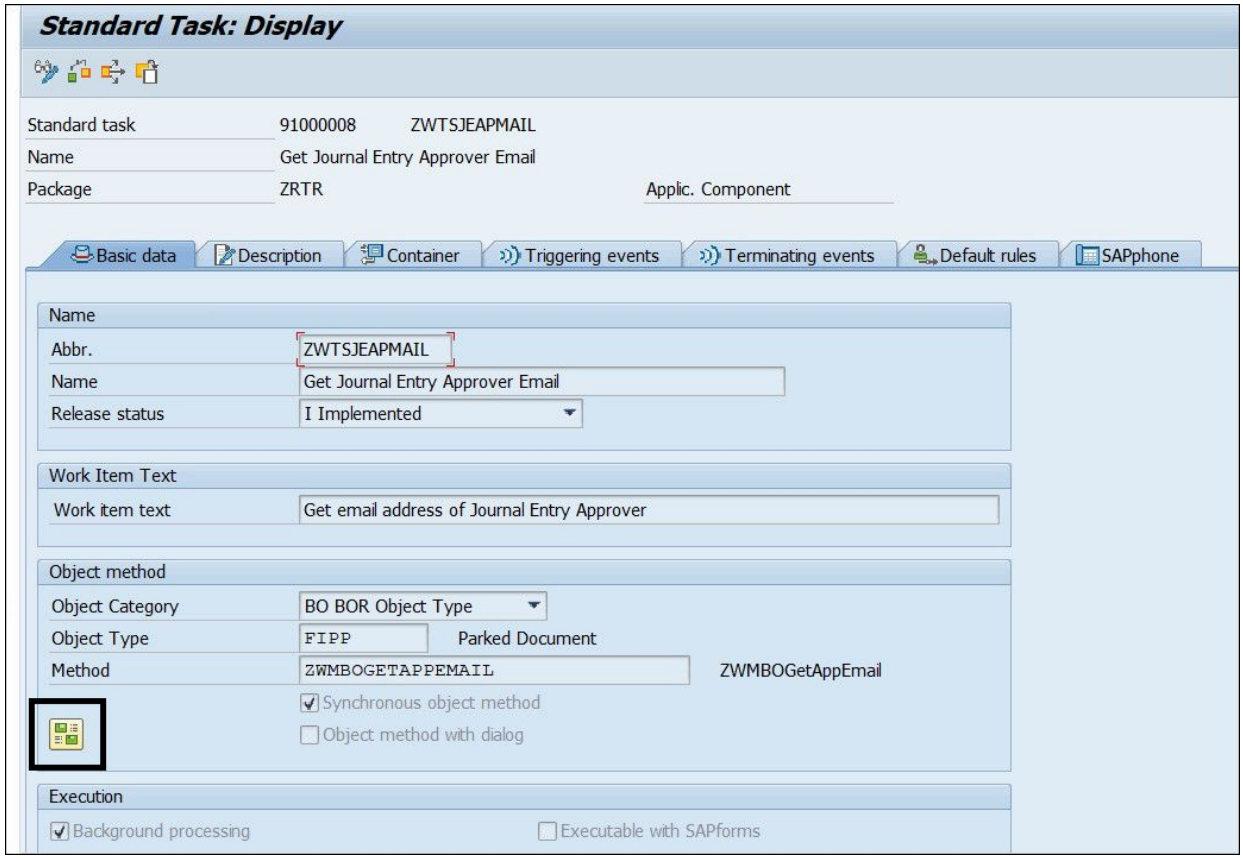


Figure 3.71 Binding Editor Button in the Standard Task Definition

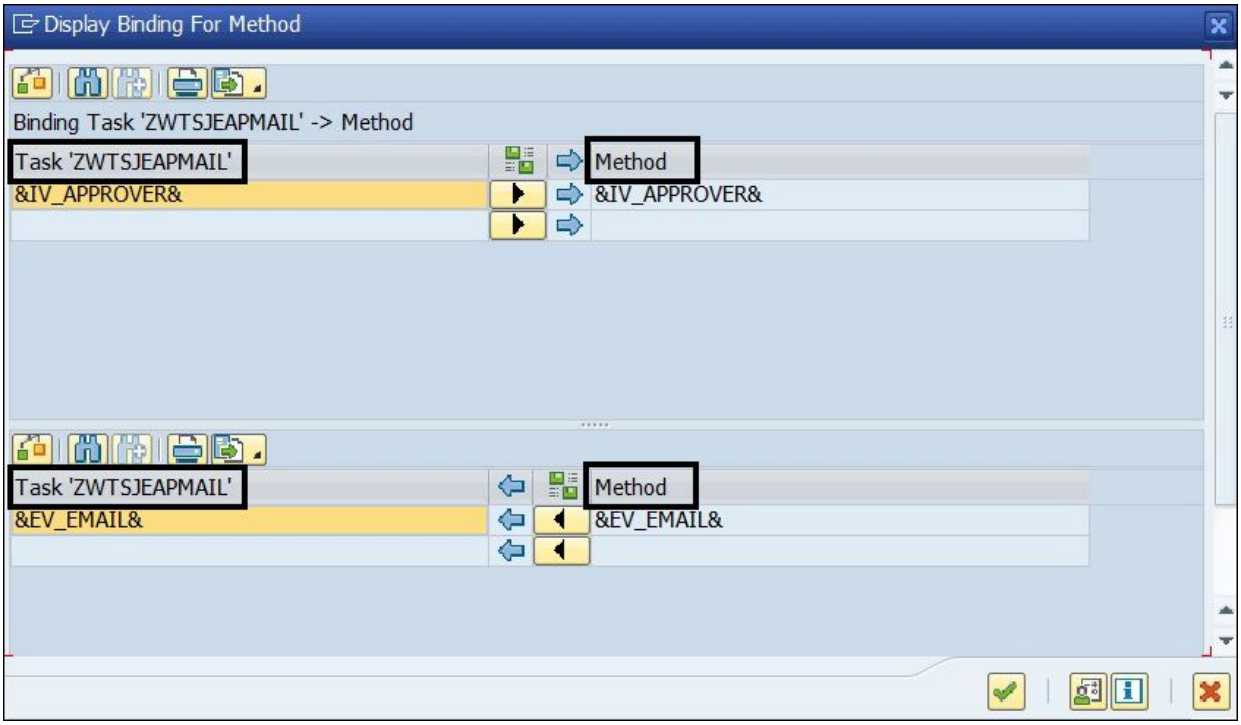


Figure 3.72 View of Binding Editor for Mapping Task and Method Container

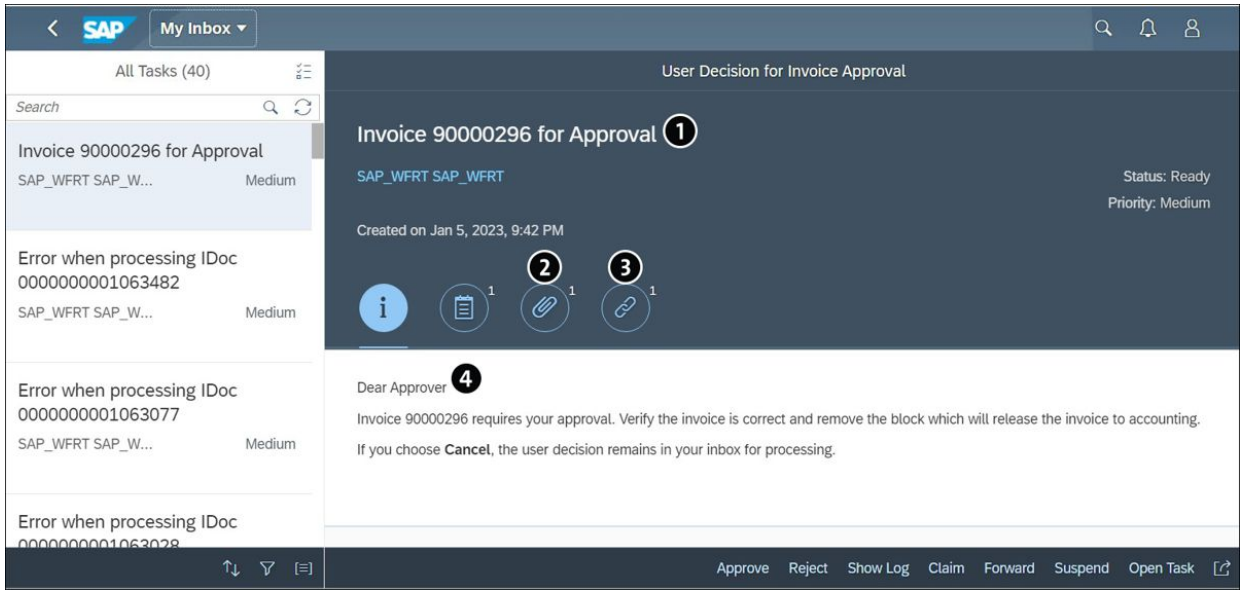


Figure 3.73 View of Work Item Text and Description for a User Decision Step in the Workflow from the My Inbox App

Figure 3.74 Work Item Text/Description for a Send Mail Step in the Workflow from the Outlook Inbox

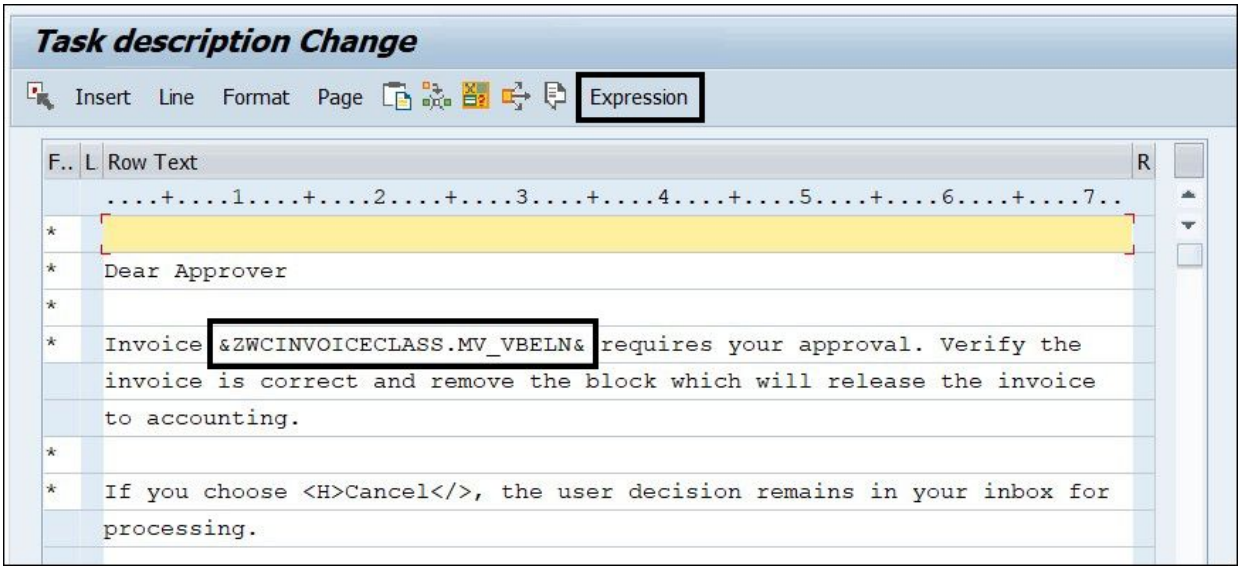


Figure 3.75 View of Task Description with Long Text Maintained Using Task Container Elements as Variables

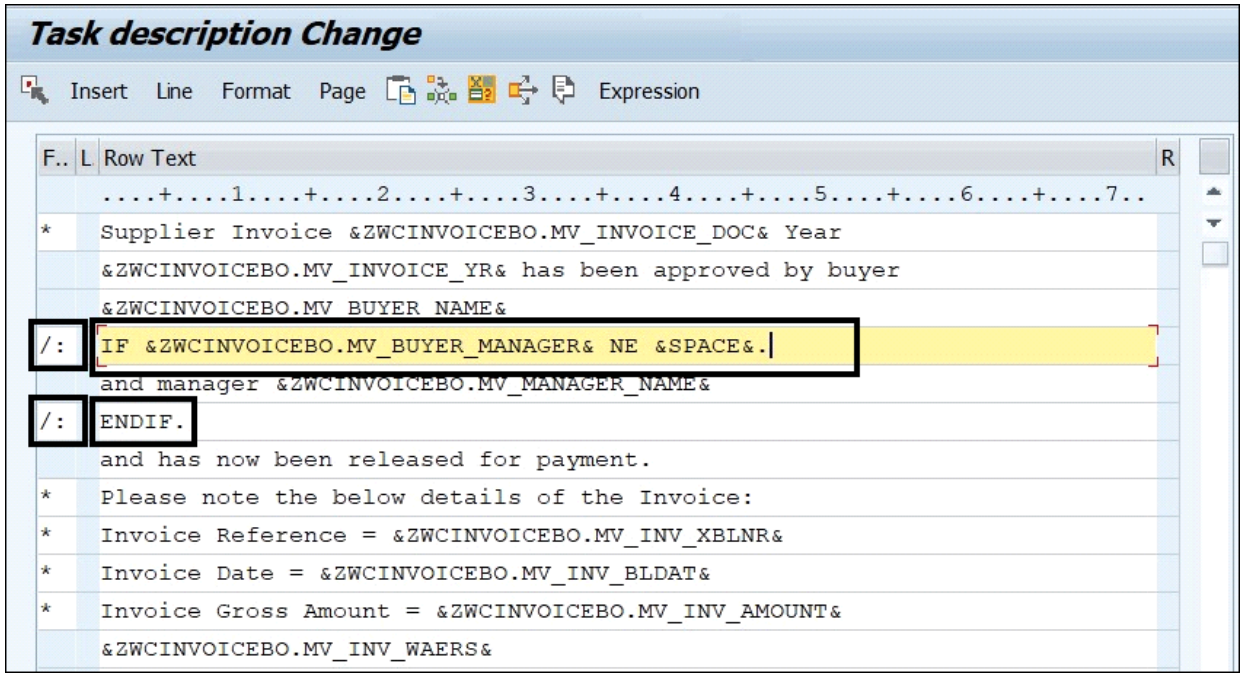


Figure 3.76 Task Description with Conditional Text Using IF-ENDIF Command Lines

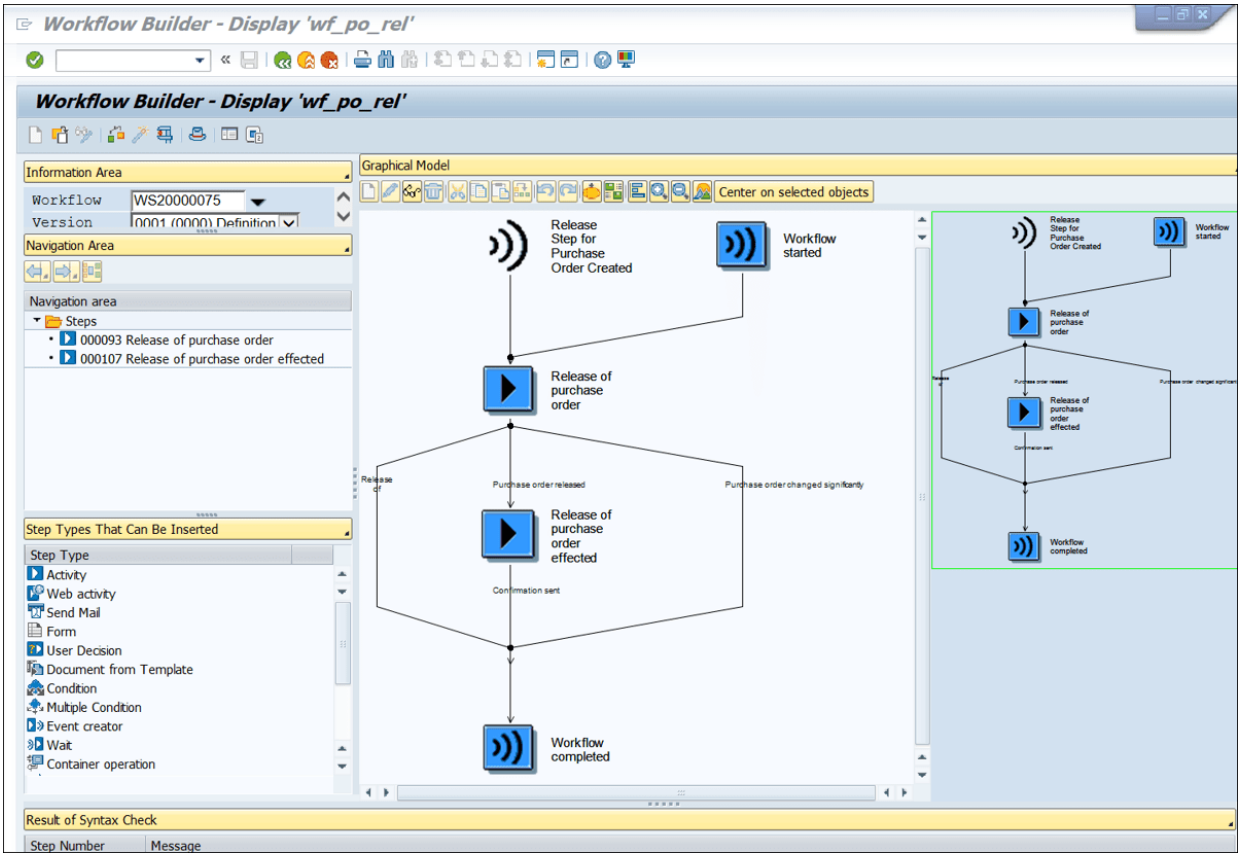


Figure 4.1 Workflow Builder Sections

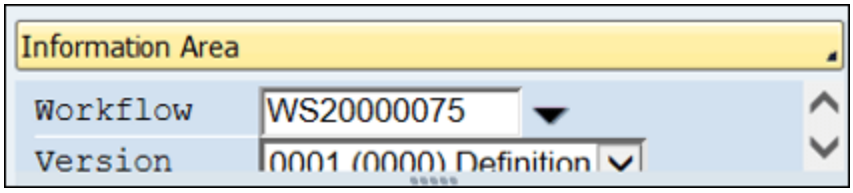


Figure 4.2 Workflow Builder: Information Area

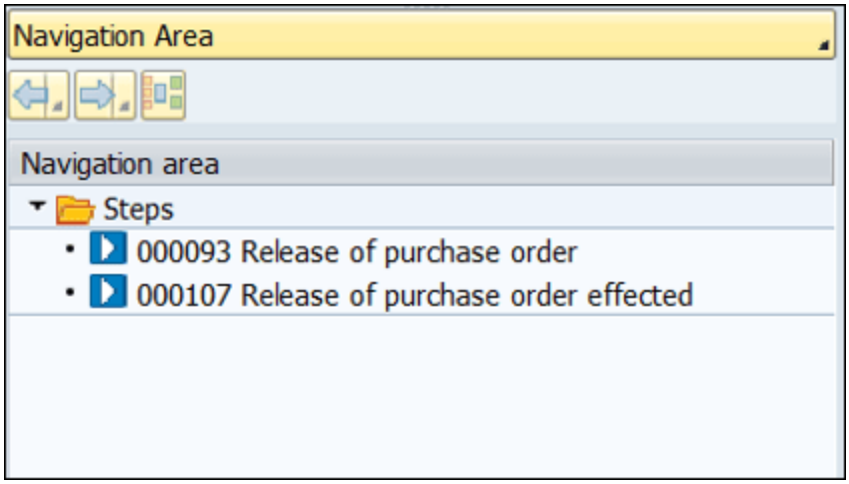


Figure 4.3 Workflow Builder: Navigation Area

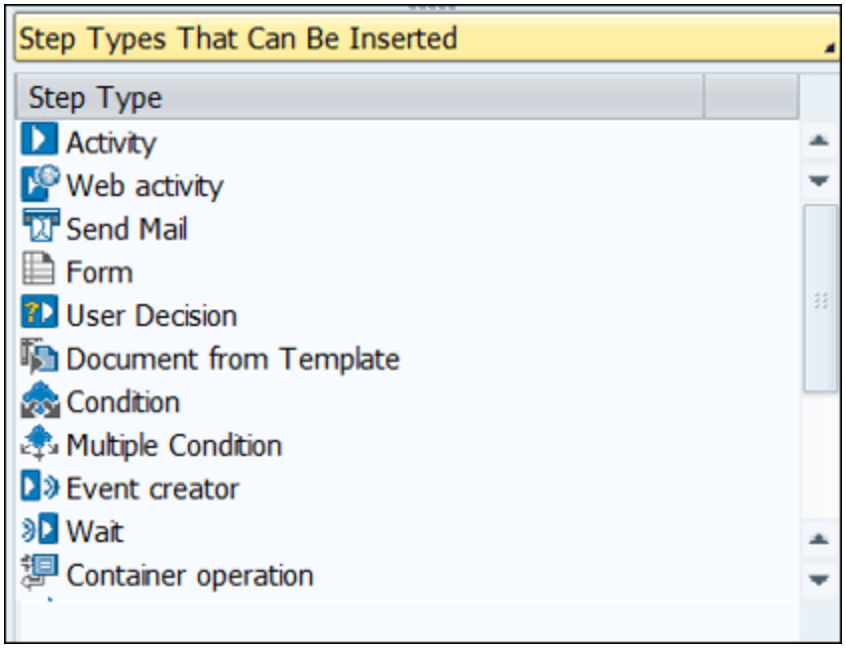


Figure 4.4 Workflow Builder: Step Types

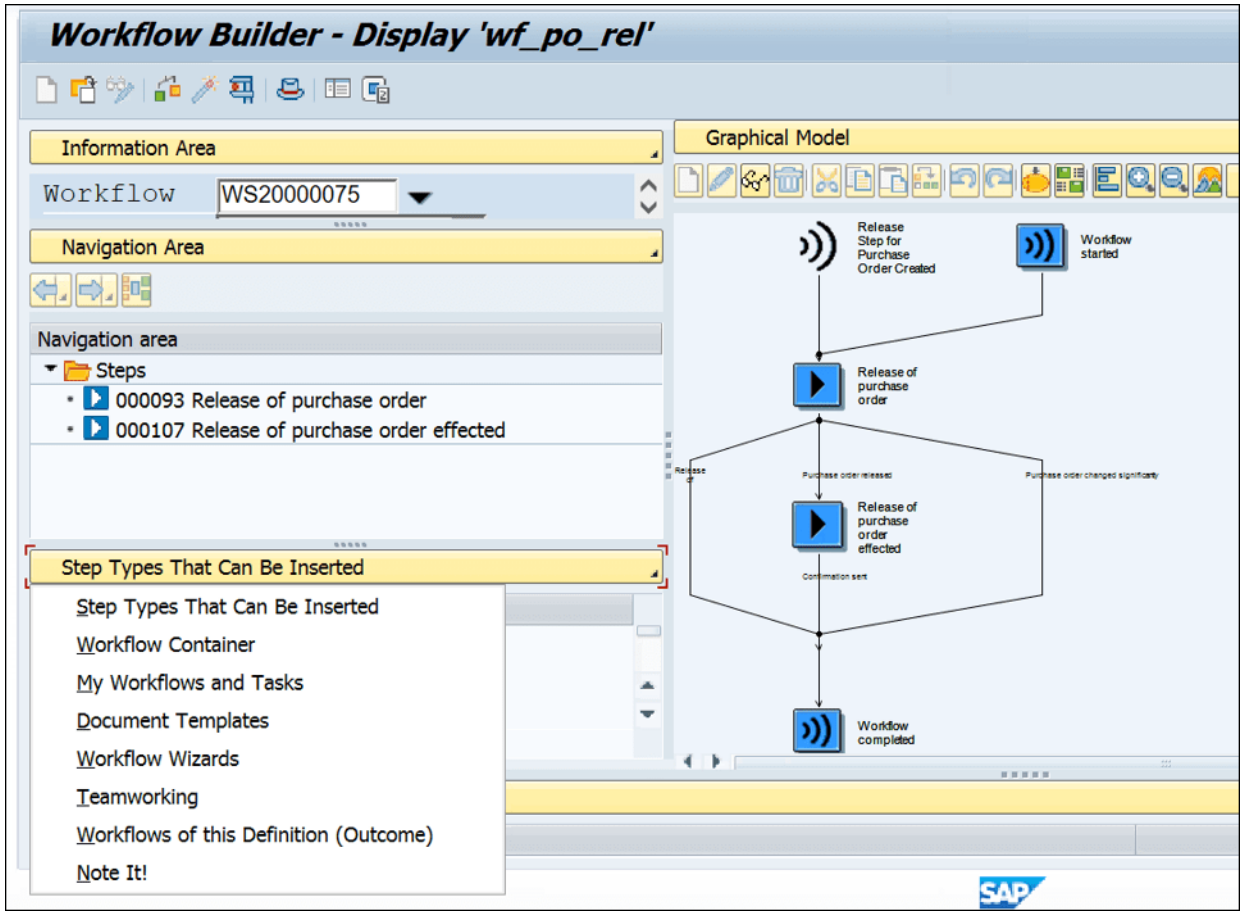


Figure 4.5 Workflow Builder: Other Components List

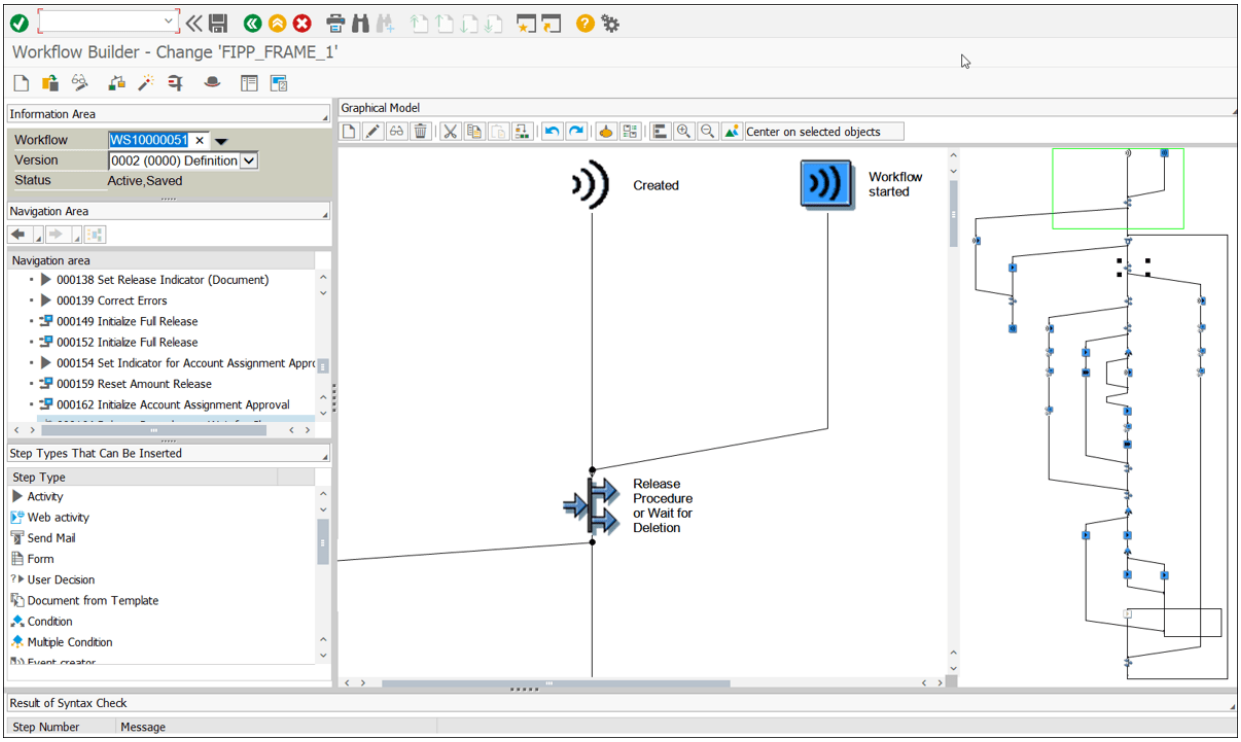


Figure 4.6 A Sample Workflow View in Workflow Builder

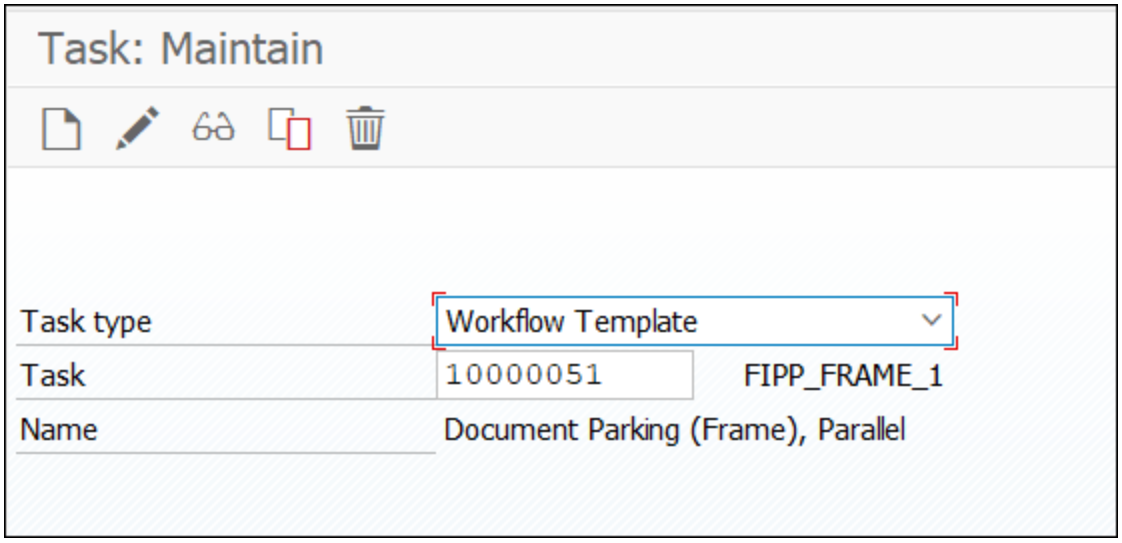


Figure 4.7 Open Workflow Builder from Transaction PFTC

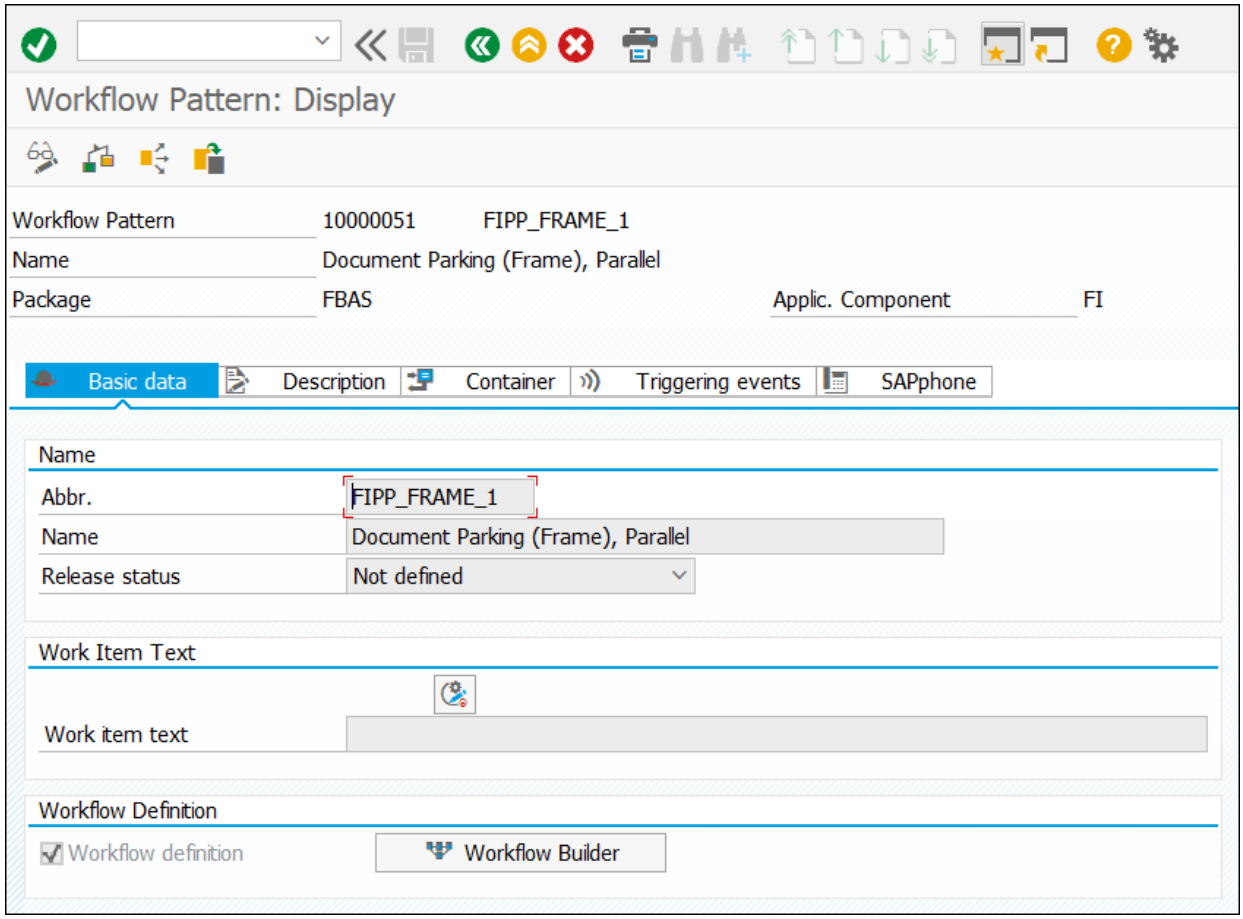


Figure 4.8 Workflow Technical Details View

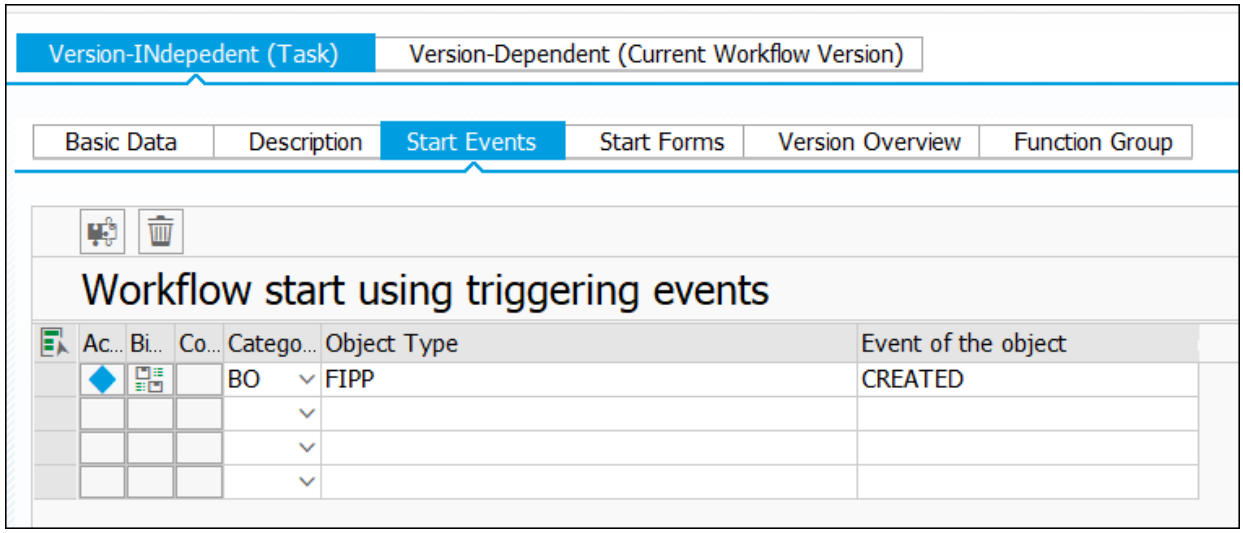


Figure 4.9 Workflow Triggering Event Mapping

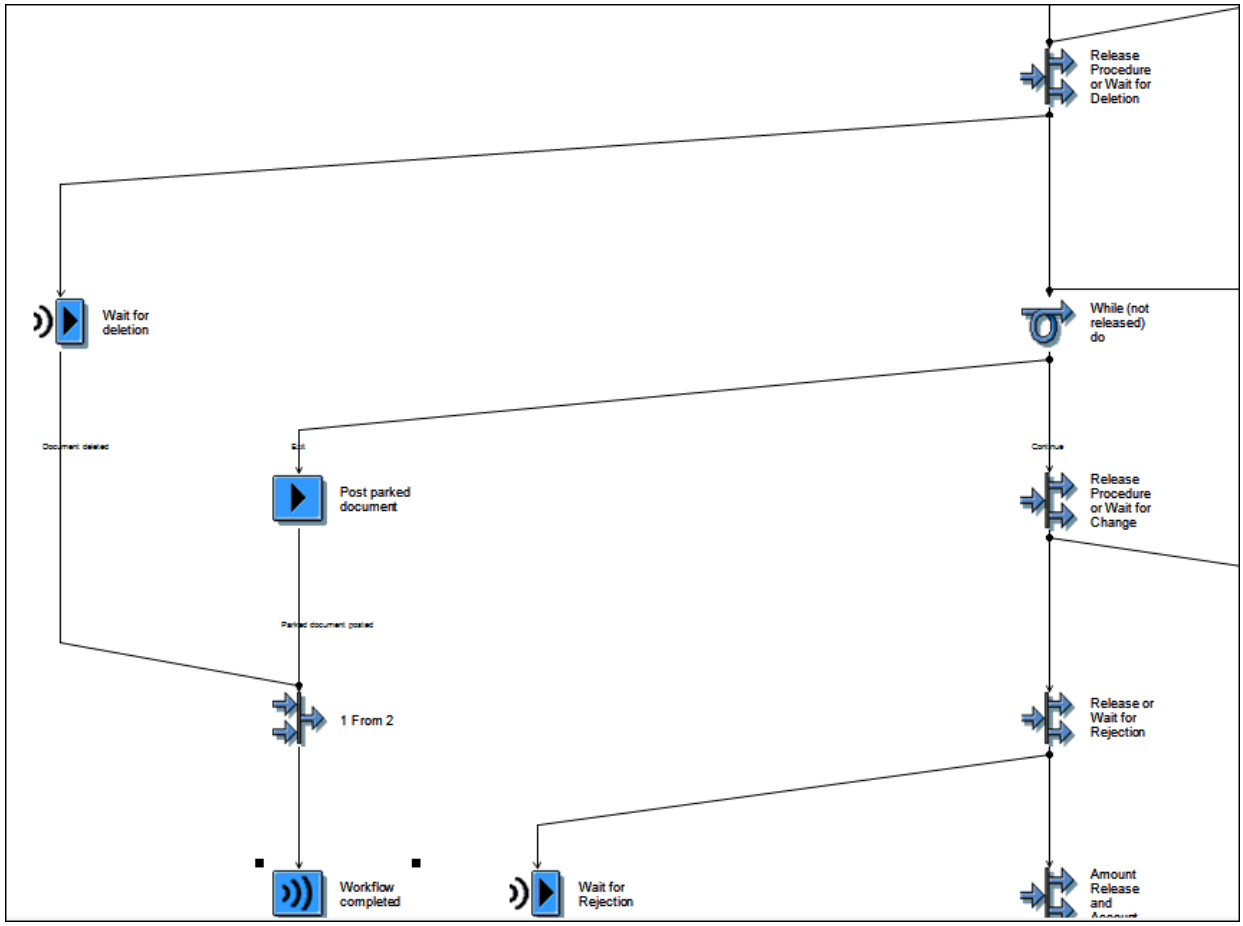


Figure 4.10 Workflow with Fork, Wait for Event, Activity, and Loop Steps

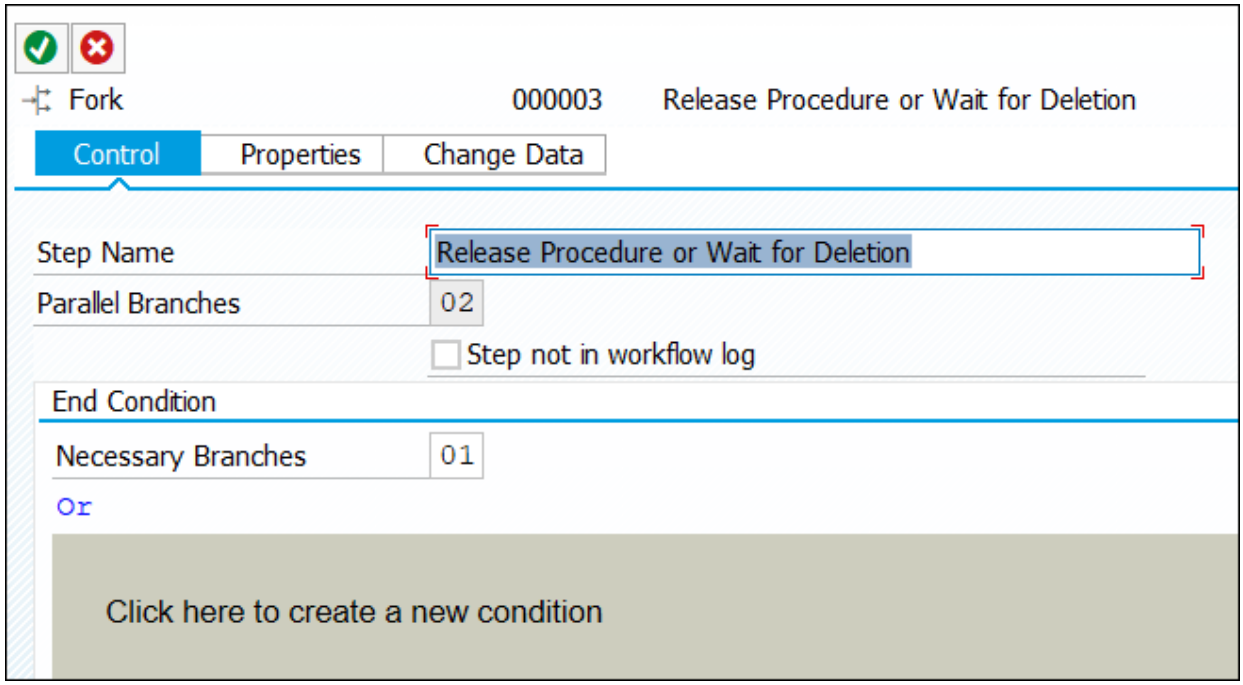


Figure 4.11 Workflow Step Fork Details

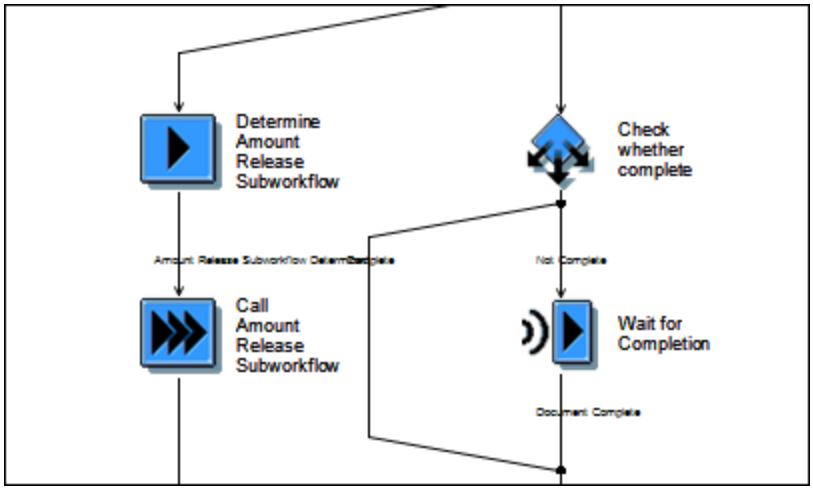


Figure 4.12 Use of Subworkflow and Background Task

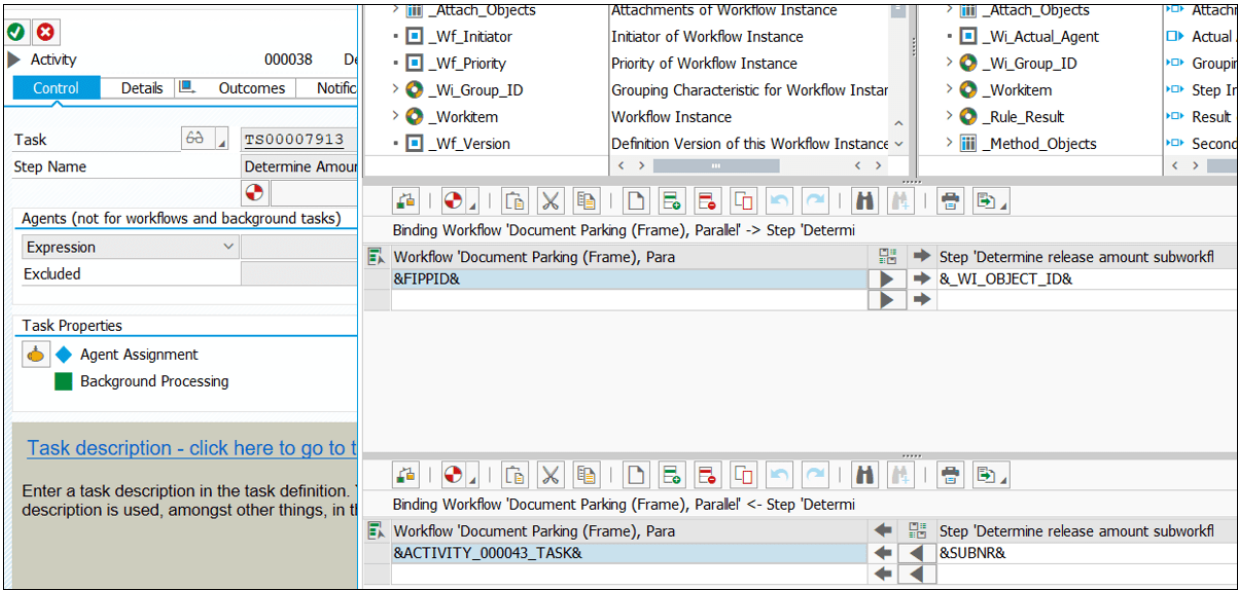


Figure 4.13 Workflow Task Control Information and Container Element Binding

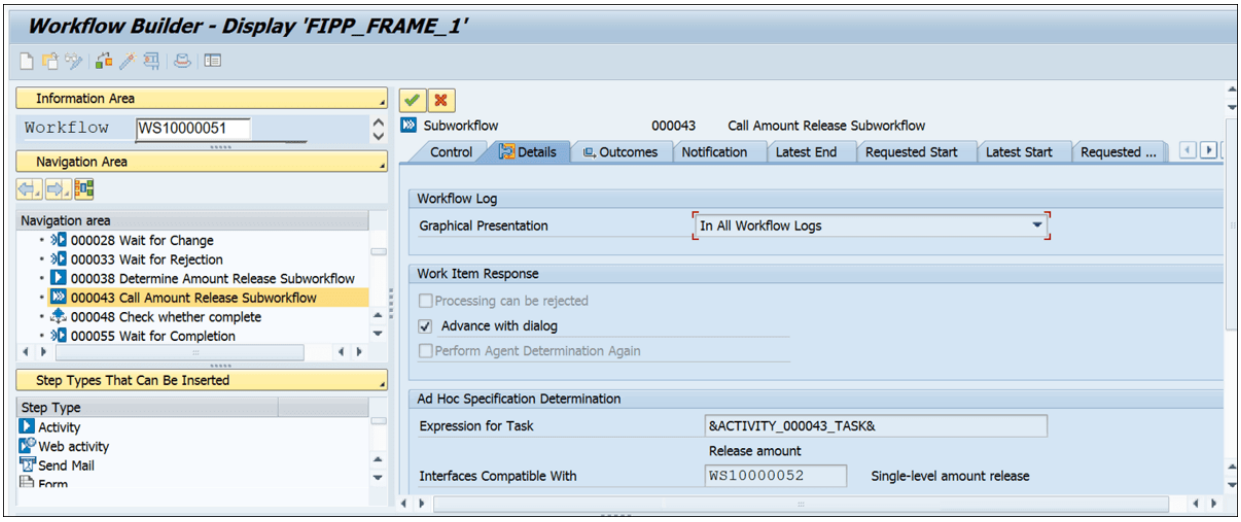


Figure 4.14 Calling the Subworkflow Dynamically

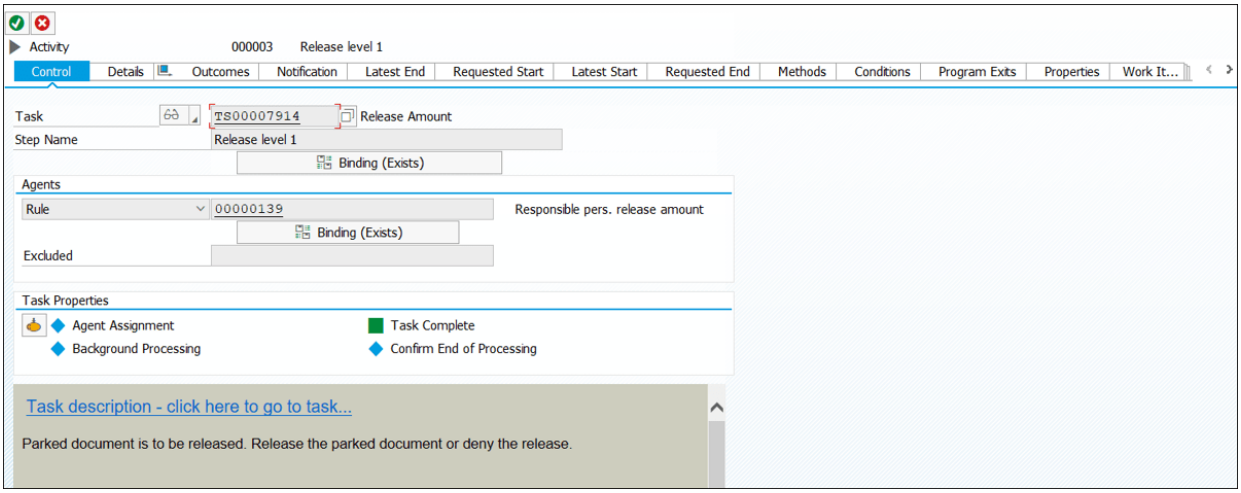


Figure 4.15 Activity Step Details: Control Tab

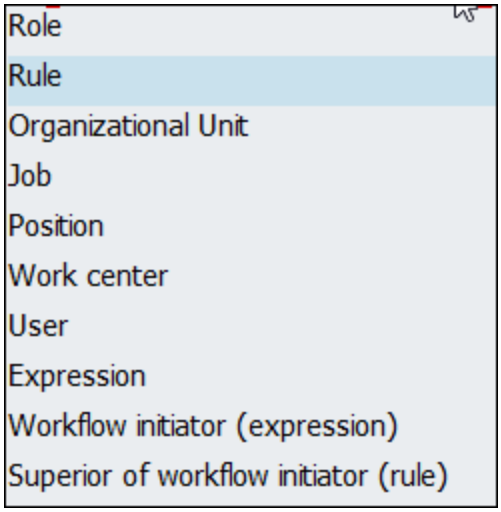


Figure 4.16 Different Agent Determination Types

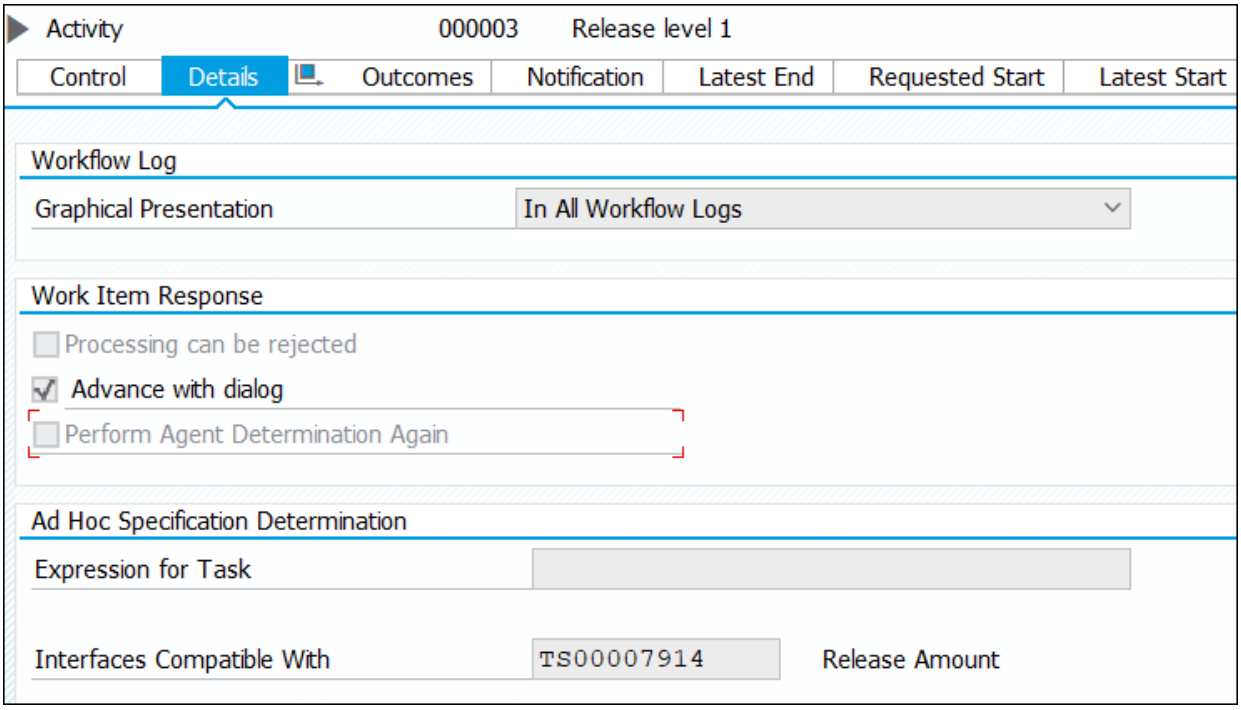


Figure 4.17 Activity Step Attributes: Details Tab

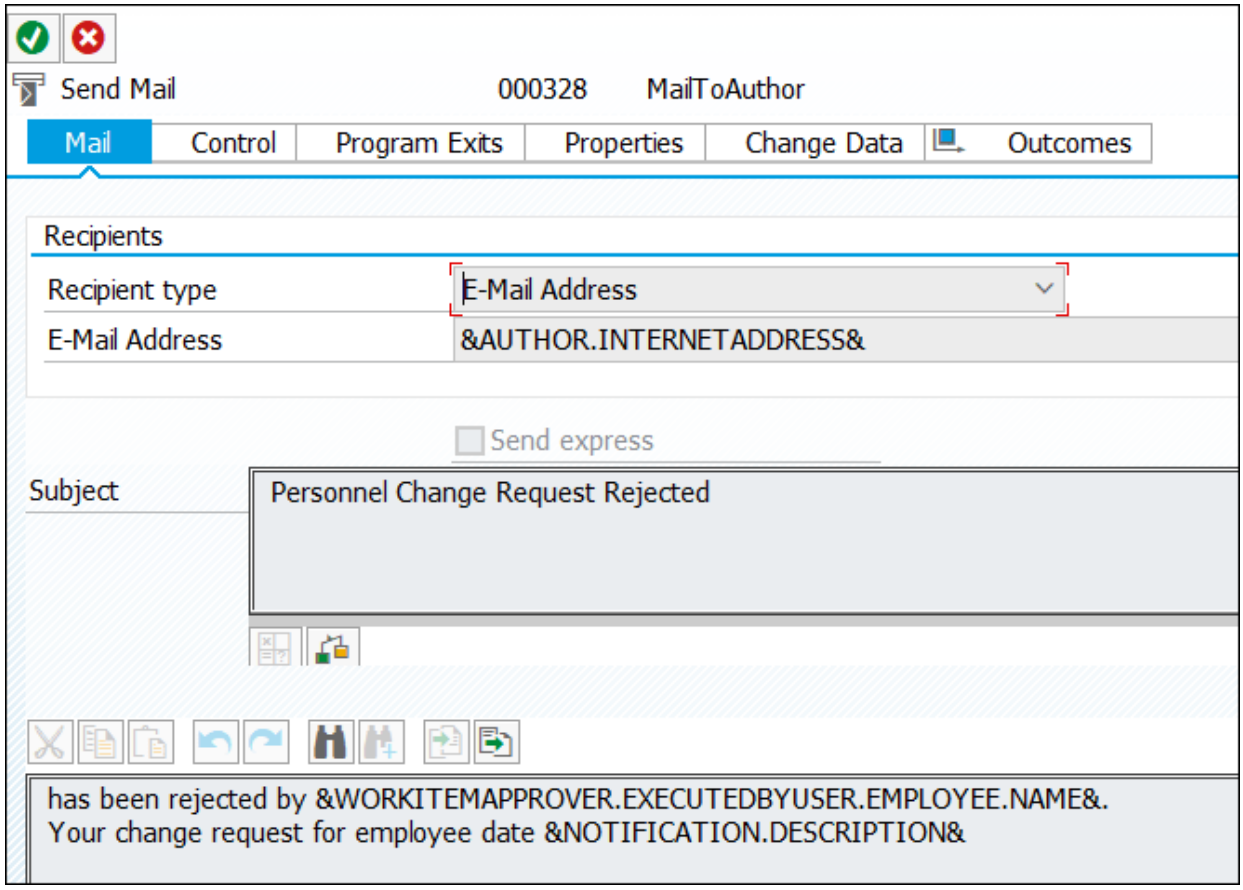


Figure 4.18 Send Email Step Type

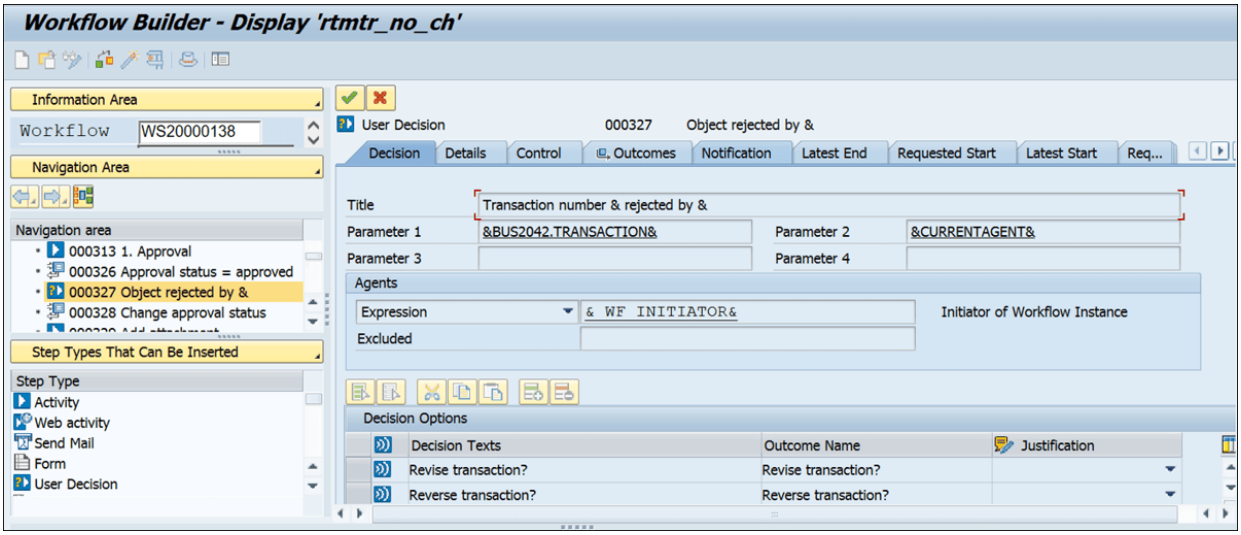


Figure 4.19 User Decision Step Type

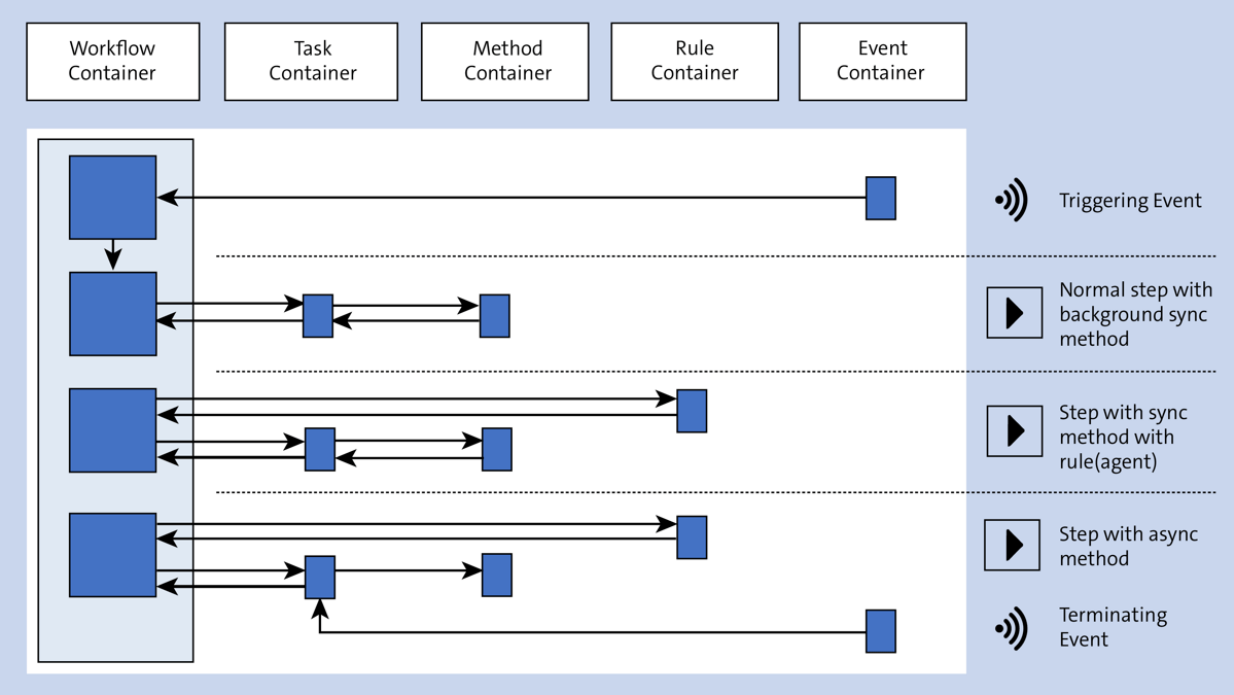


Figure 4.20 Workflow Container and Flow of Data Using Container Binding

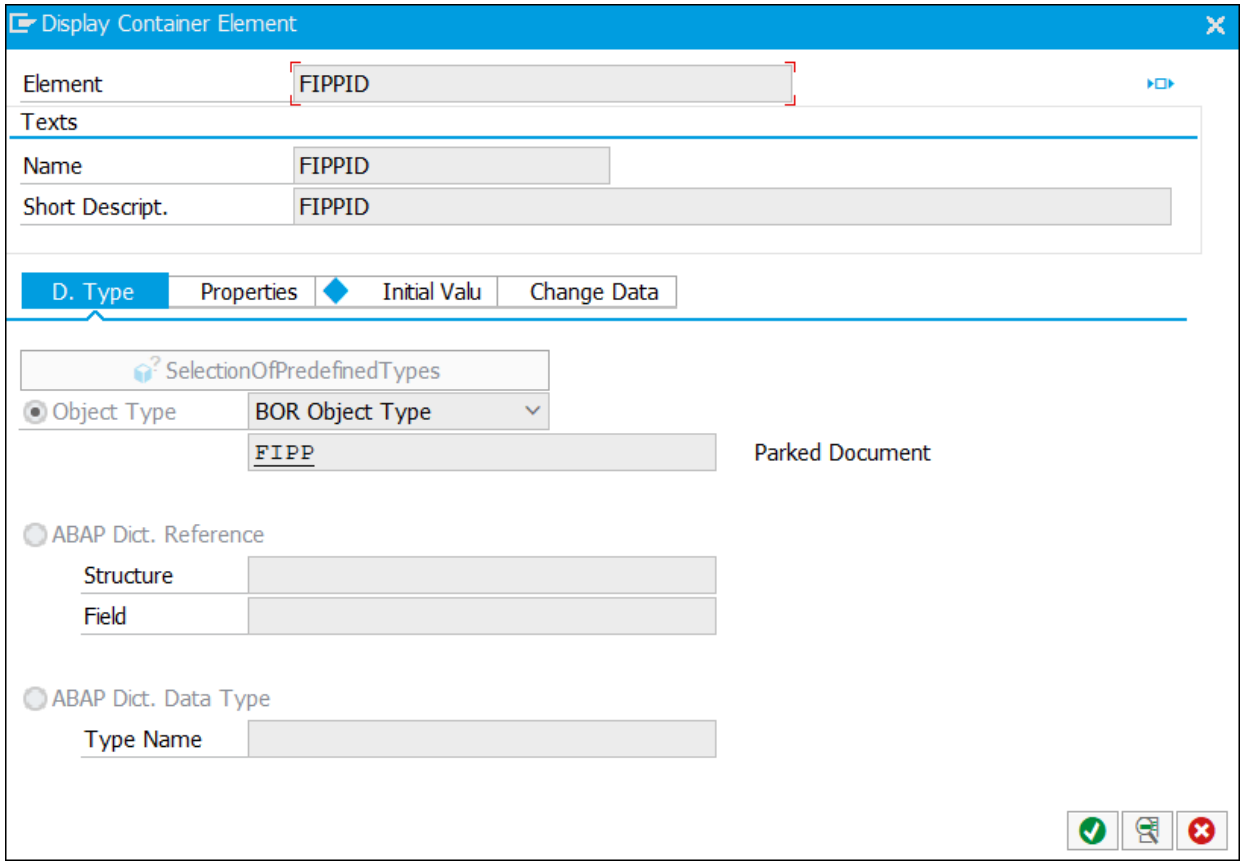


Figure 4.21 Workflow Container

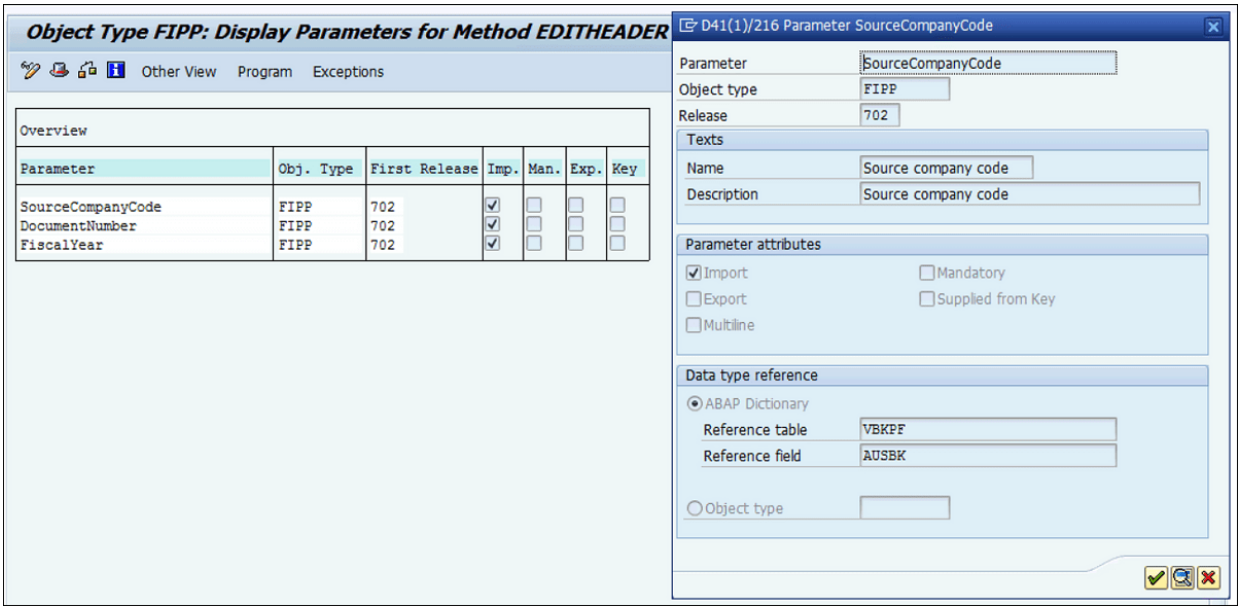


Figure 4.22 Container Element of a Method

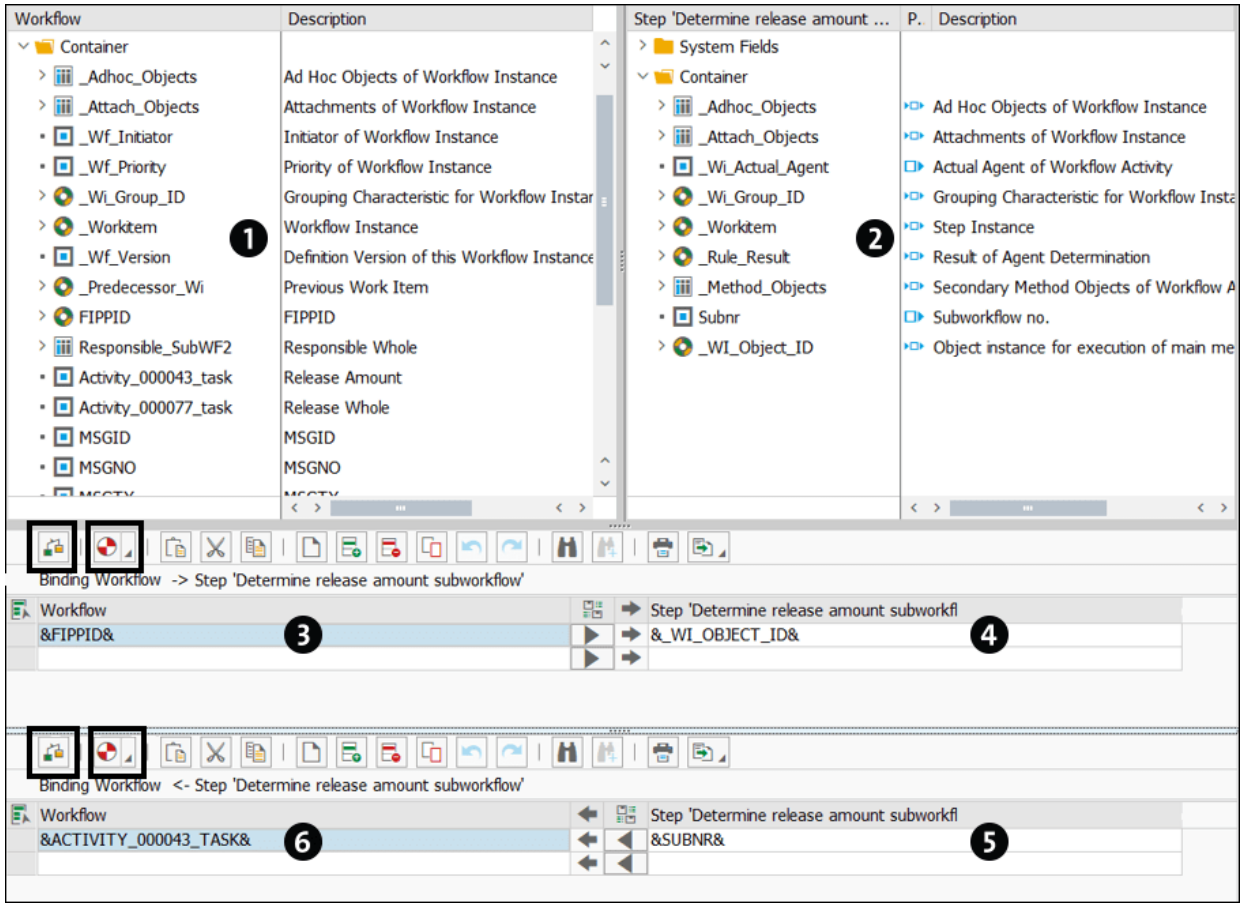


Figure 4.23 Container Binding Editor

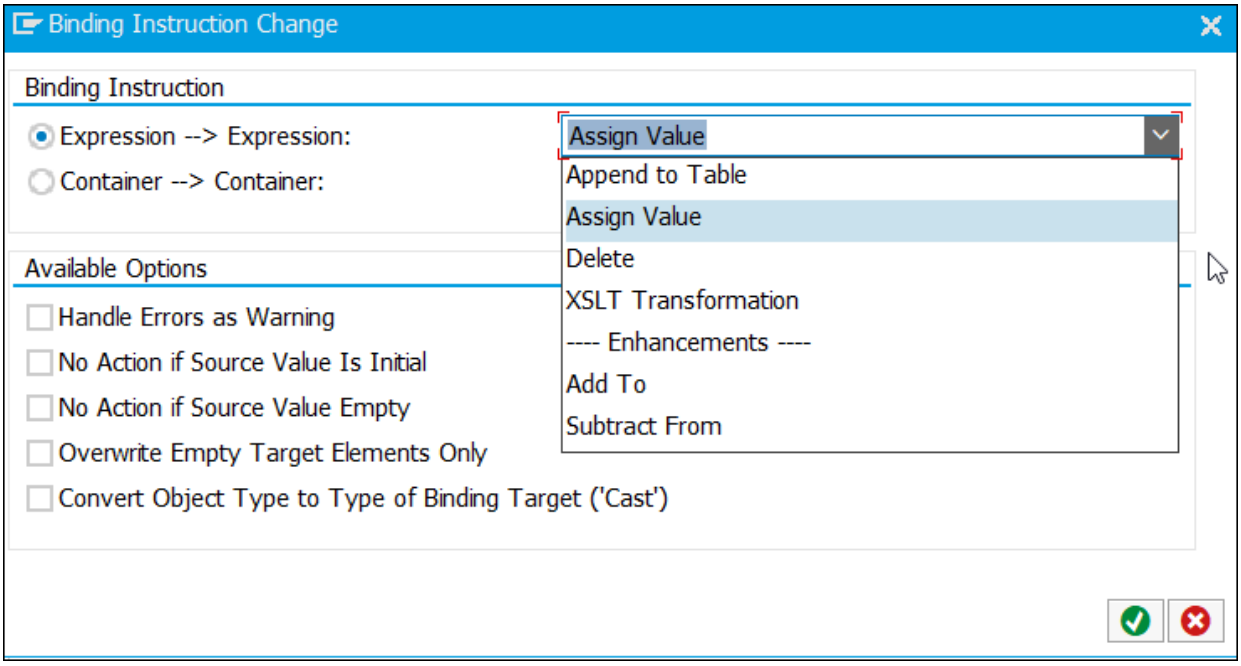


Figure 4.24 Binding Expression Operation

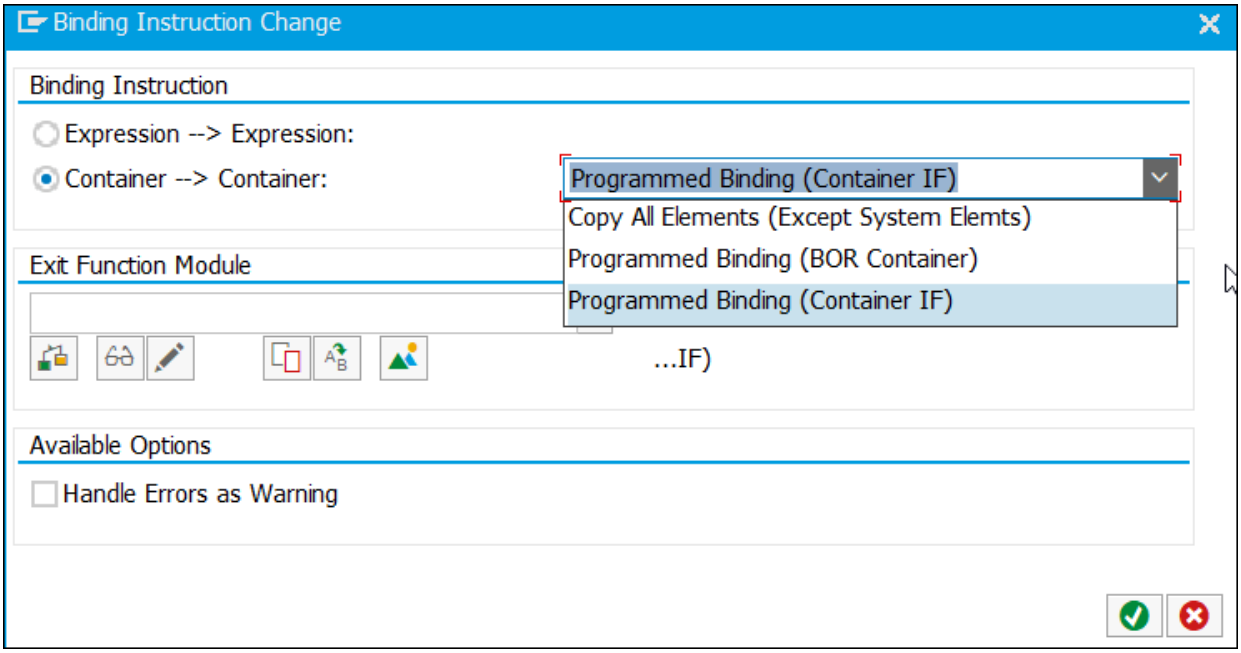


Figure 4.25 Binding Container Operation

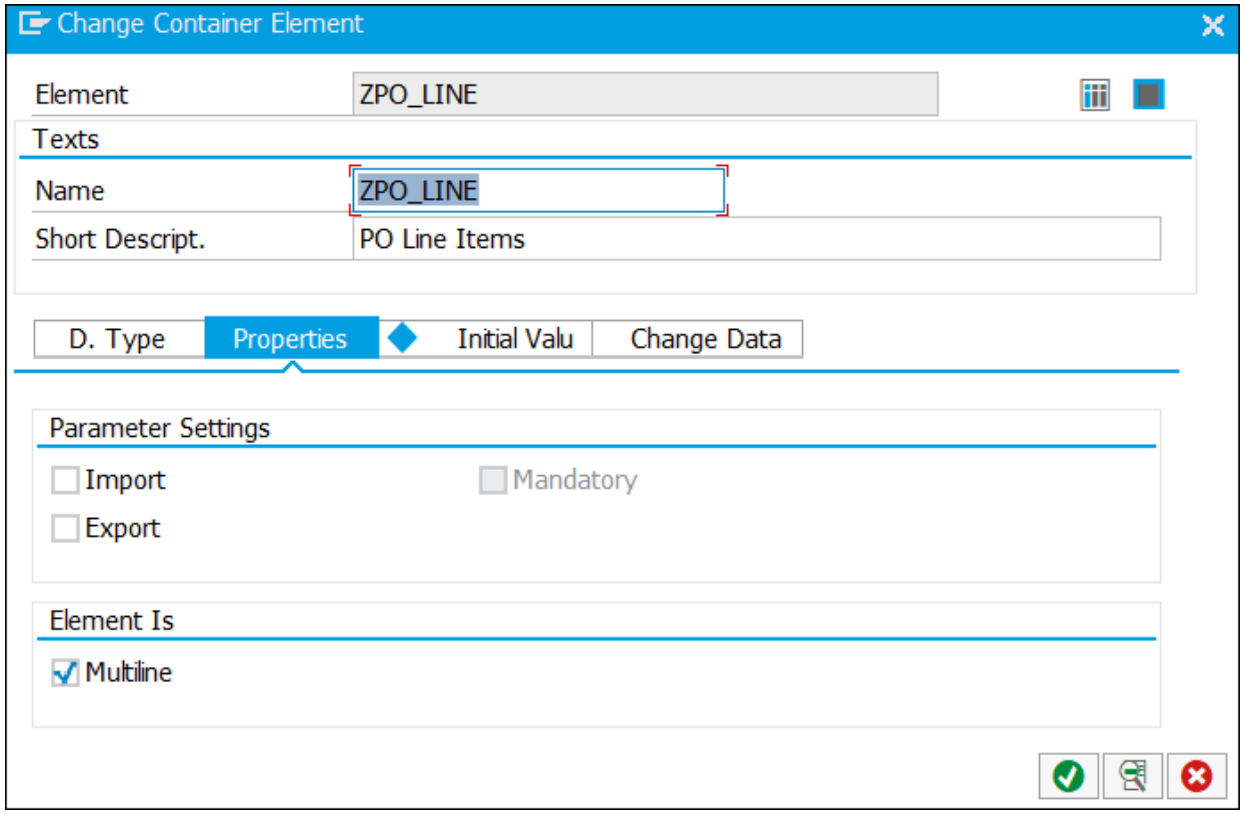


Figure 4.26 Multiline Container Element

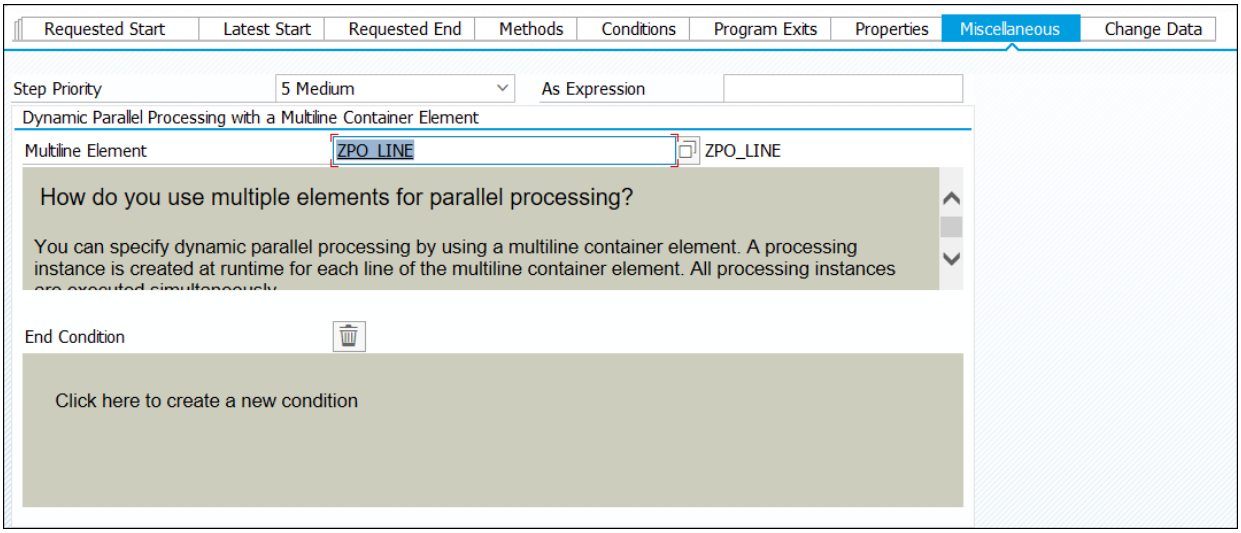


Figure 4.27 Dynamic Parallel Processing with a Multiline Container Element

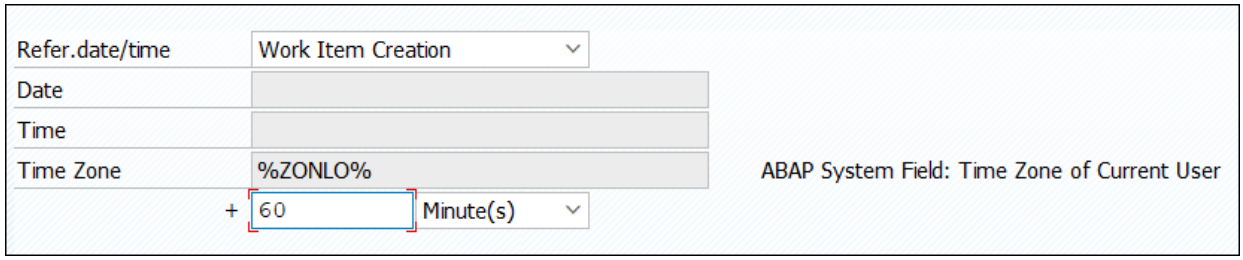


Figure 4.28 Deadline Time Calculation

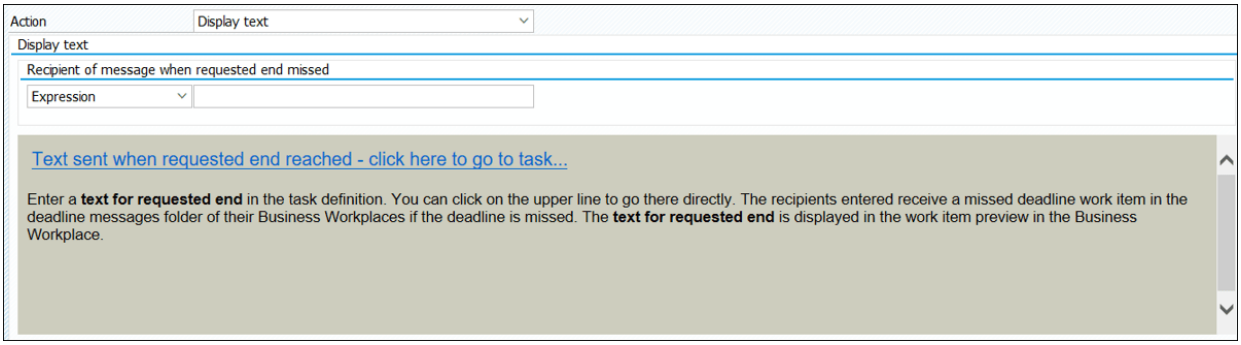


Figure 4.29 Action Notification Trigger When the Work Item Deadline Is Reached

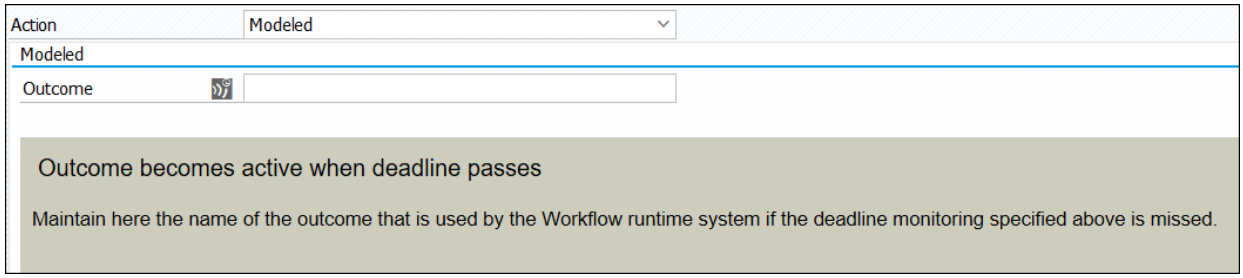


Figure 4.30 Action Event Trigger When the Work Item Deadline Is Reached

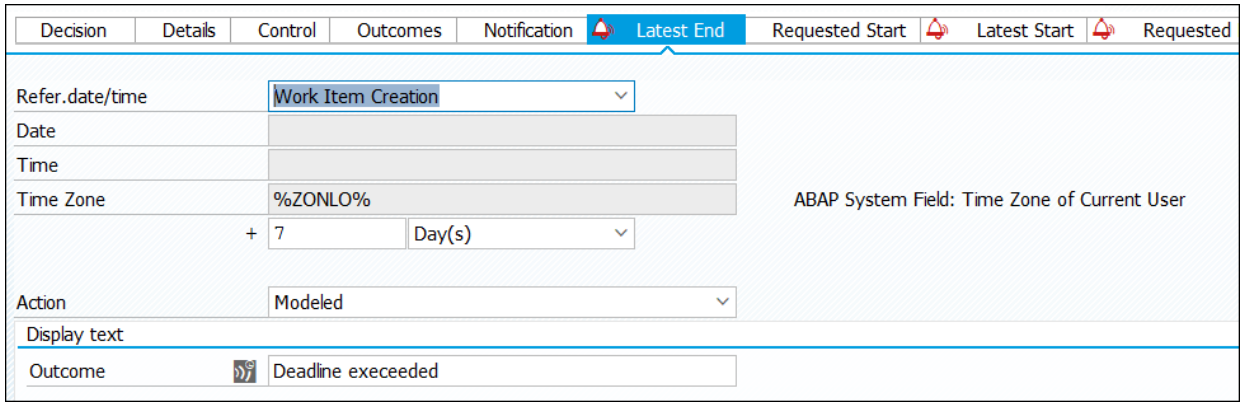


Figure 4.31 Set Up the Modeled Outcome for the LATEST END Deadline

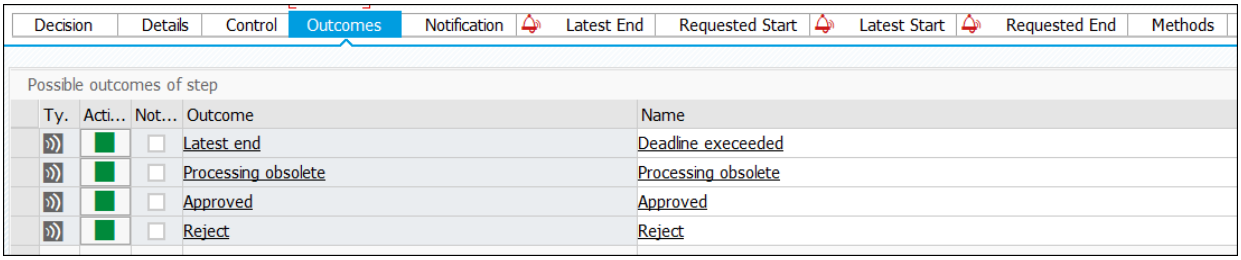


Figure 4.32 Activate the Processing Obsolete Outcome

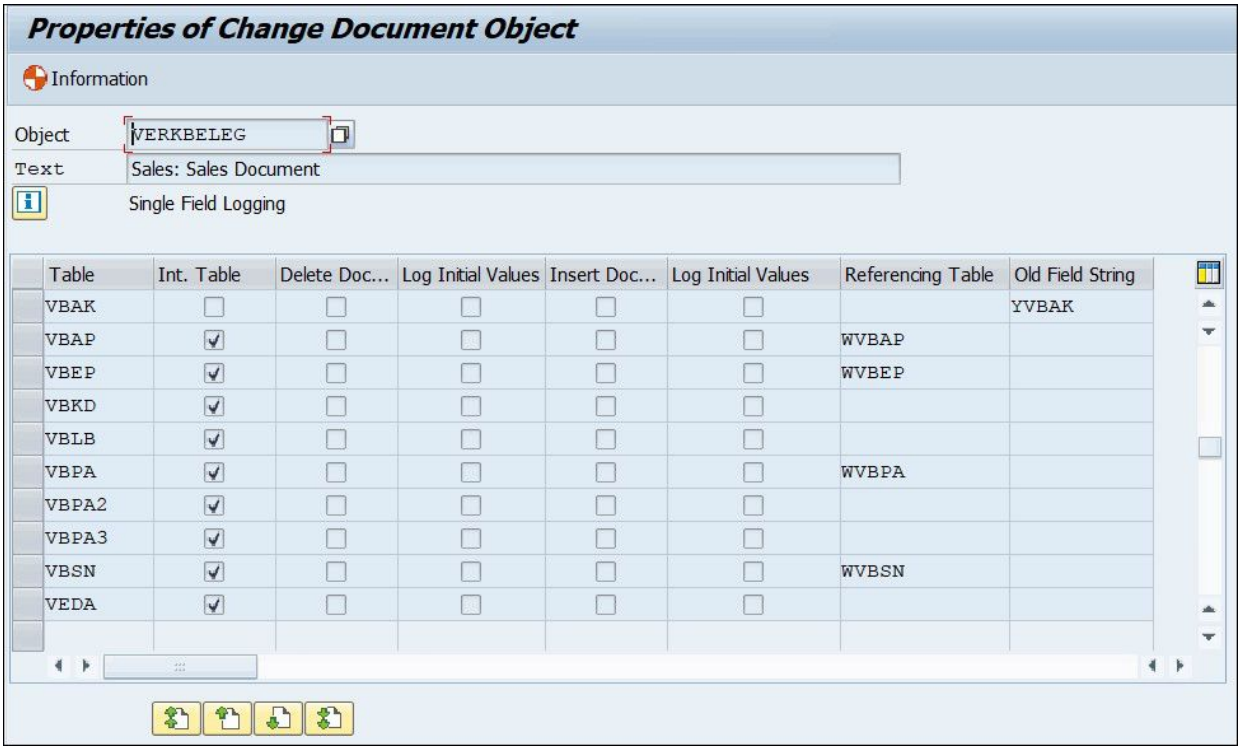
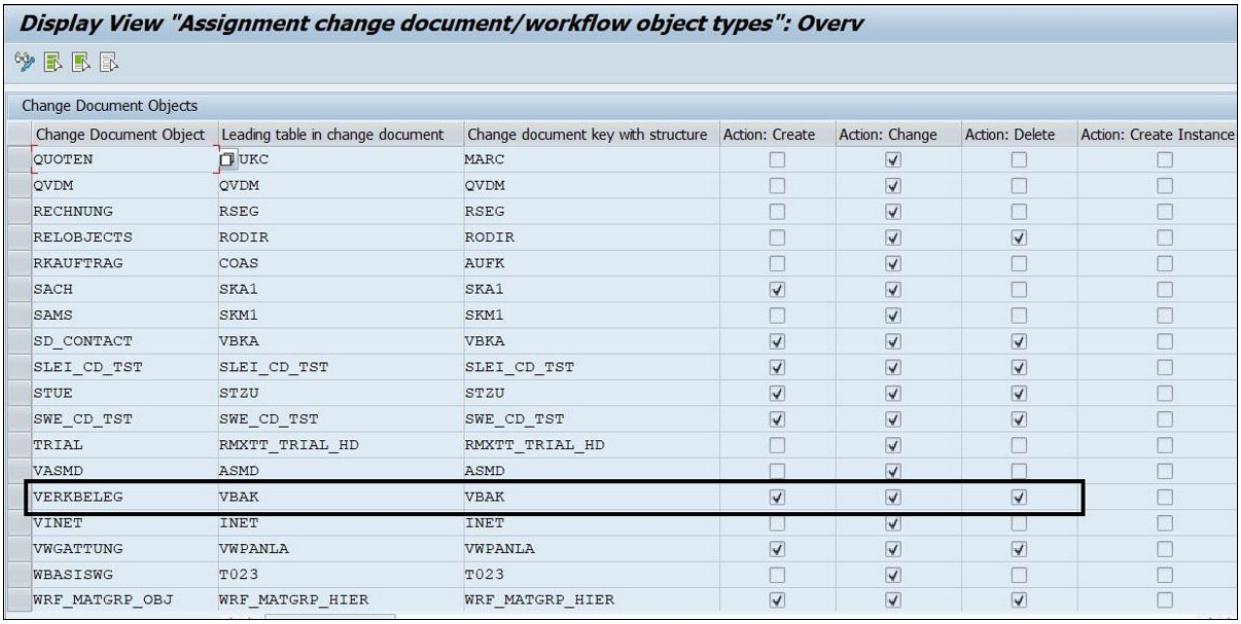


Figure 5.1 A Change Document Object in SAP



Change Document Object	Leading table in change document	Change document key with structure	Action: Create	Action: Change	Action: Delete	Action: Create Instance
QUOTEN	UKC	MARC	☐	☑	☐	☐
QVDM	QVDM	QVDM	☐	☑	☐	☐
RECHNUNG	RSEG	RSEG	☐	☑	☐	☐
RELOBJECTS	RODIR	RODIR	☐	☑	☑	☐
RKAUFTRAG	COAS	AUFK	☐	☑	☐	☐
SACH	SKA1	SKA1	☑	☑	☐	☐
SAMS	SKM1	SKM1	☐	☑	☐	☐
SD_CONTACT	VBKA	VBKA	☑	☑	☑	☐
SLEI_CD_TST	SLEI_CD_TST	SLEI_CD_TST	☑	☑	☑	☐
STUE	STZU	STZU	☑	☑	☑	☐
SWE_CD_TST	SWE_CD_TST	SWE_CD_TST	☑	☑	☑	☐
TRIAL	RMXTT_TRIAL_HD	RMXTT_TRIAL_HD	☐	☑	☐	☐
VASMD	ASMD	ASMD	☐	☑	☐	☐
VERKBELEG	VBAK	VBAK	☑	☑	☑	☐
VINET	INET	INET	☐	☑	☐	☐
VWGATTUNG	VWPANLA	VWPANLA	☑	☑	☑	☐
WBASISWG	T023	T023	☐	☑	☐	☐
WRF_MATGRP_OBJ	WRF_MATGRP_HIER	WRF_MATGRP_HIER	☑	☑	☑	☐

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

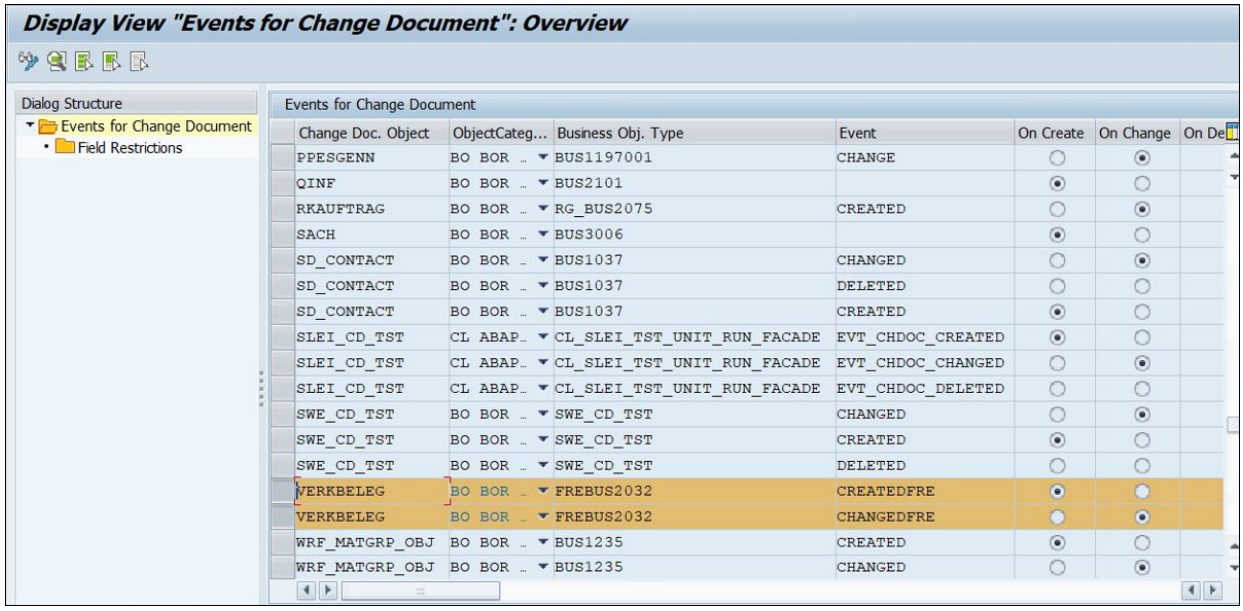


Figure 5.3 Configuring BOR/Class Events against a Change Document Object in Transaction SWEC

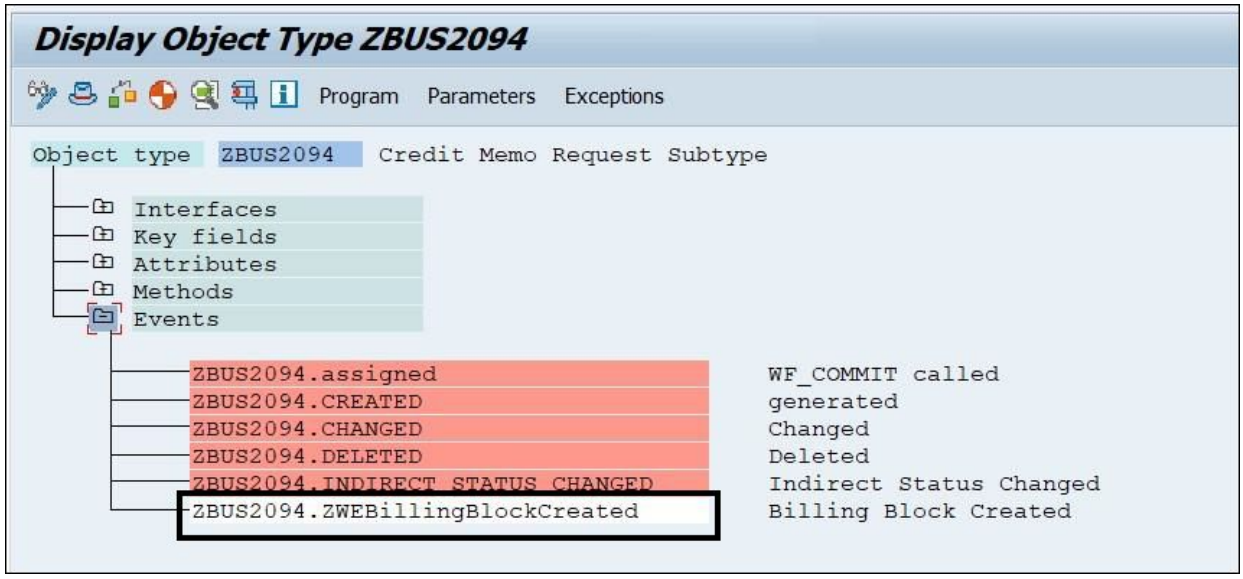


Figure 5.4 Subtype with Custom Event Added from Transaction SWO1

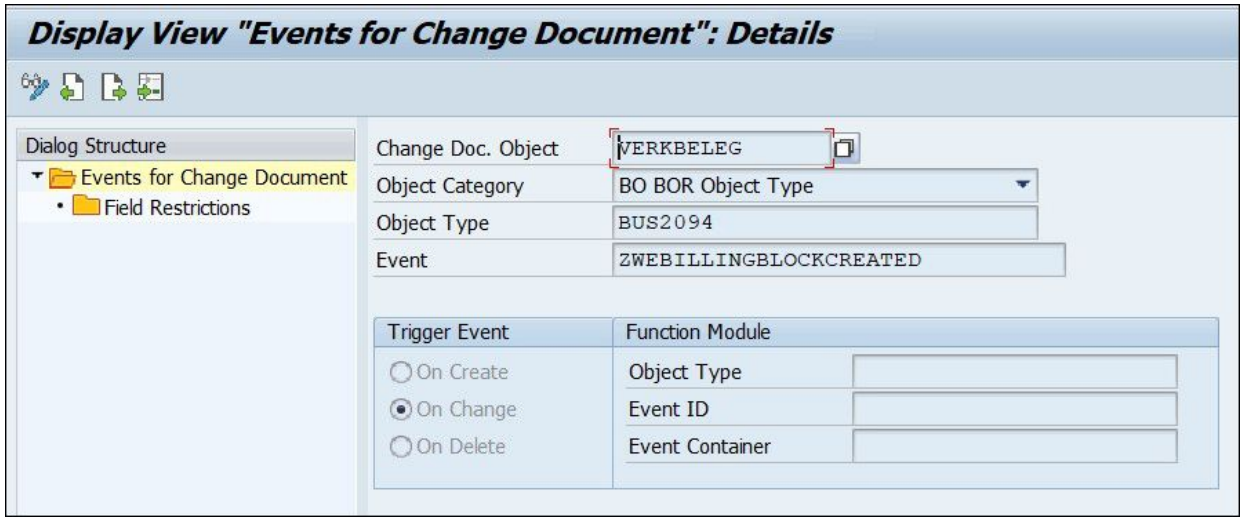


Figure 5.5 Custom Event Configured for the Change Document Object in Transaction SWEC

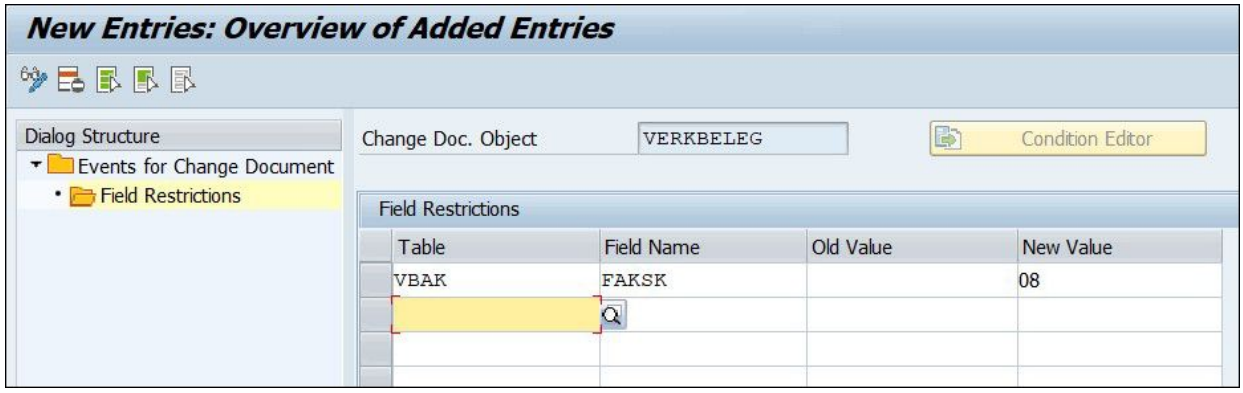


Figure 5.6 Maintaining Field Restrictions for the Event Trigger via Change Documents in Transaction SWEC

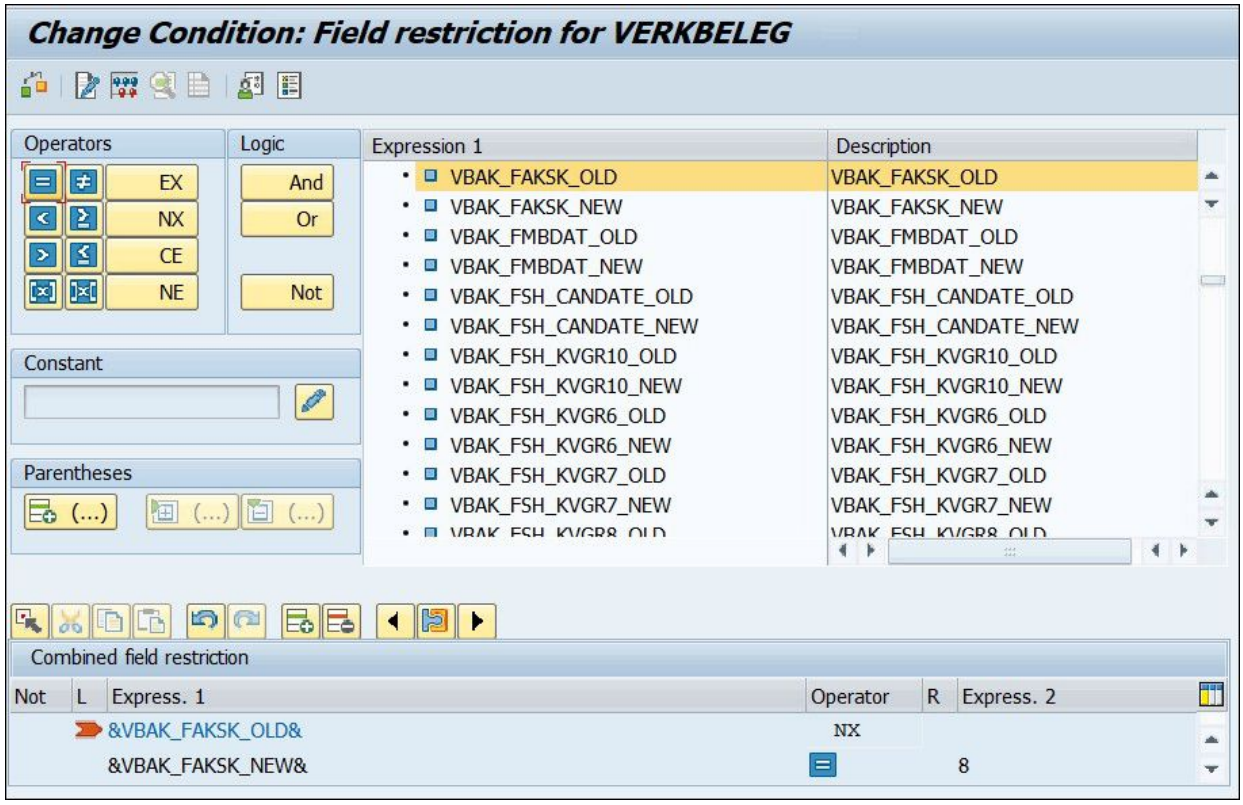


Figure 5.7 Maintaining Field Restrictions with the Condition Editor in Transaction SWEC

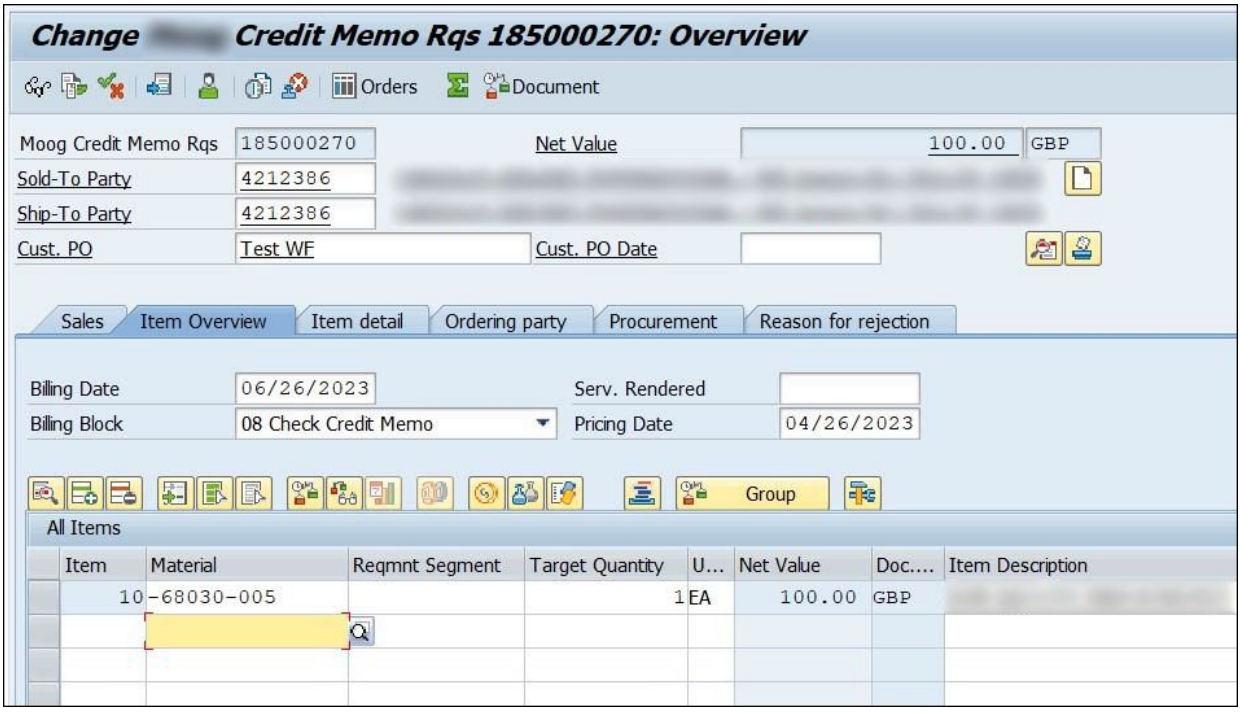


Figure 5.8 Changing a Credit Memo Request in SAP S/4HANA with the Billing Block Set

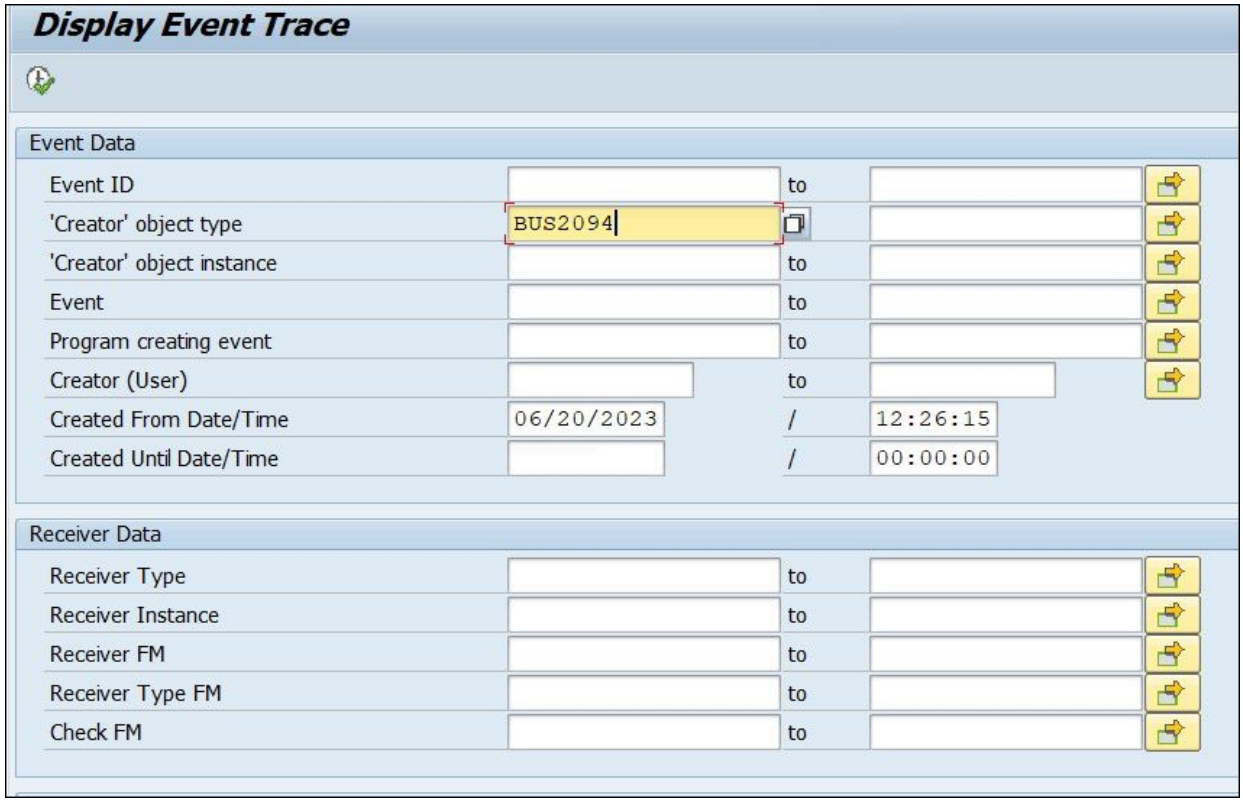


Figure 5.9 Checking the Event Trace from Transaction SWEL

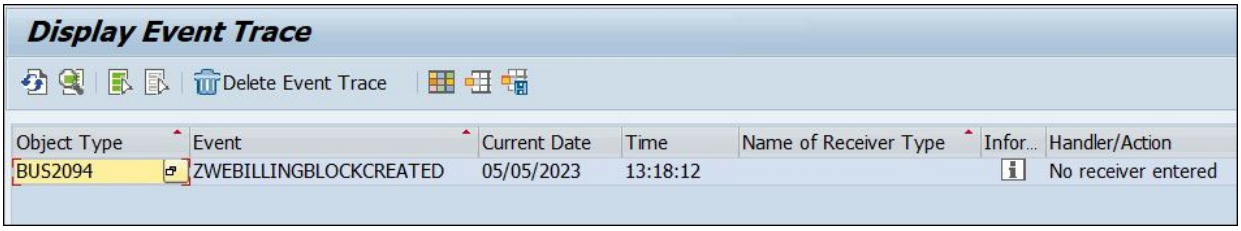


Figure 5.10 Event Trace Showing Event Has Triggered

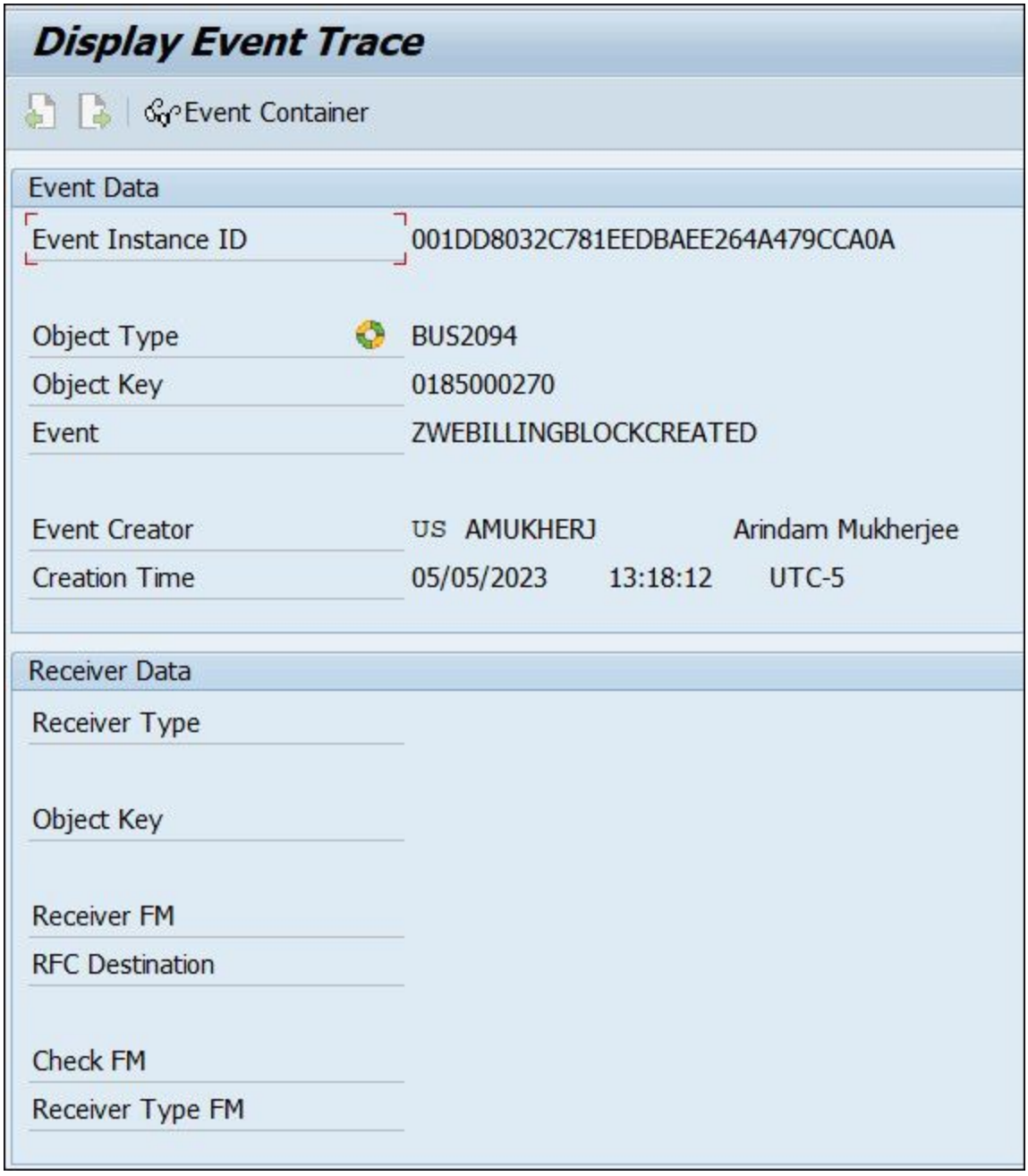


Figure 5.11 Event Trace Details Showing Event Creator Data

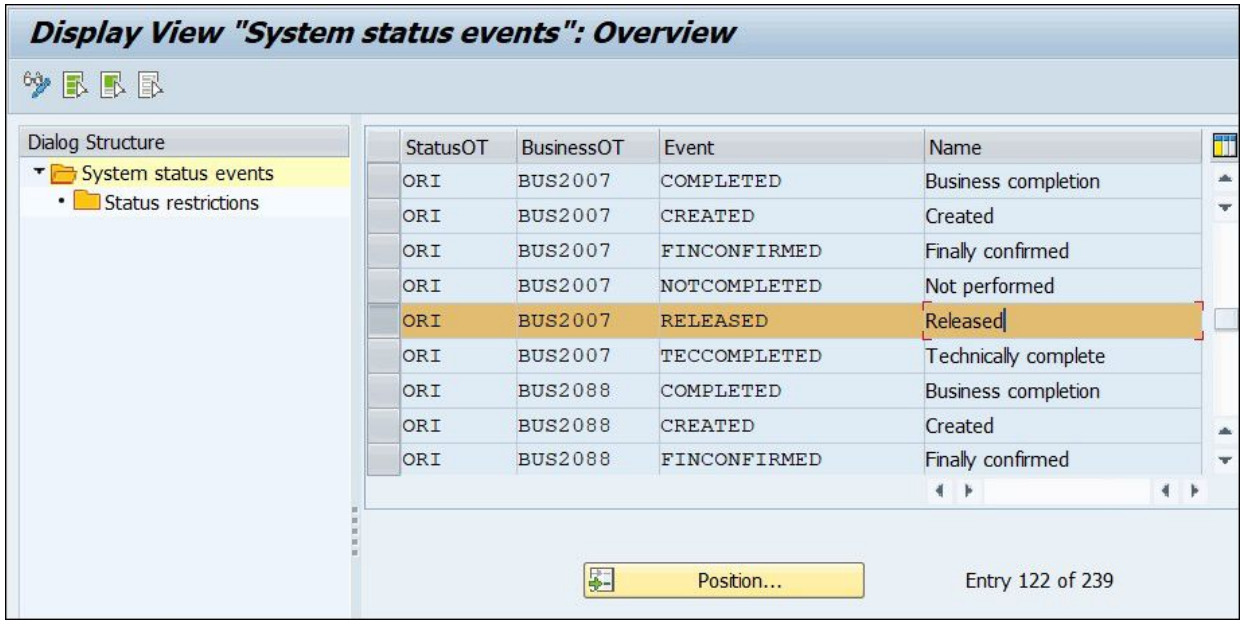


Figure 5.12 Event Configuration with System Status in Transaction BSVX

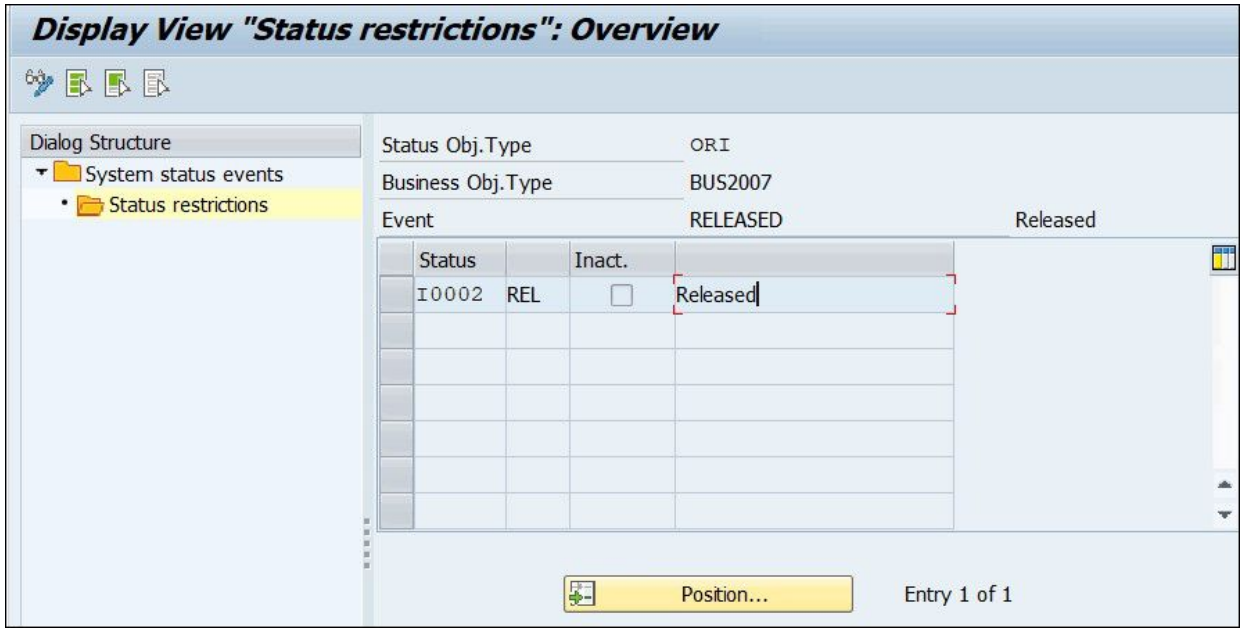


Figure 5.13 System Status Restrictions for Event Configuration

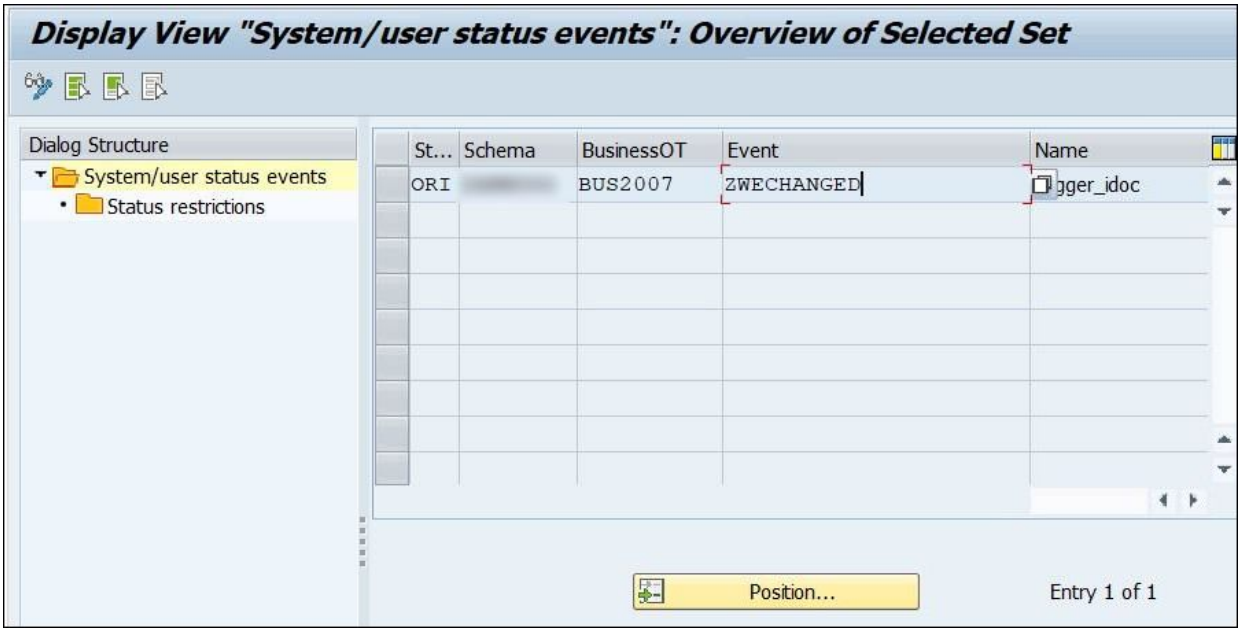


Figure 5.14 Event Configuration with User Status Profile in Transaction BSVZ

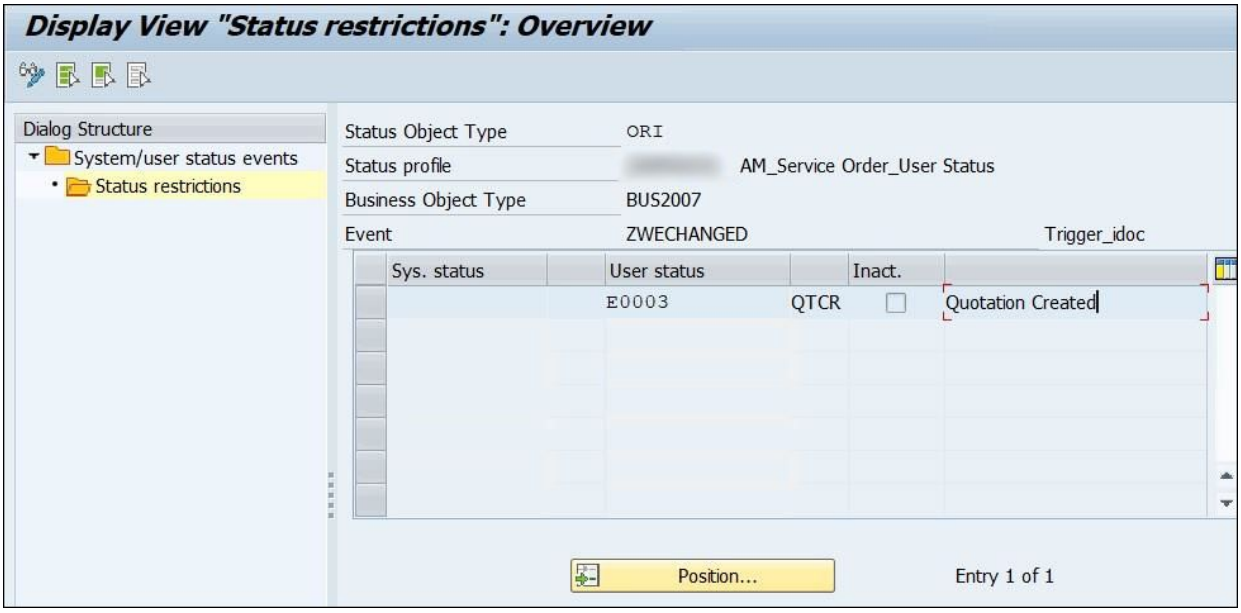
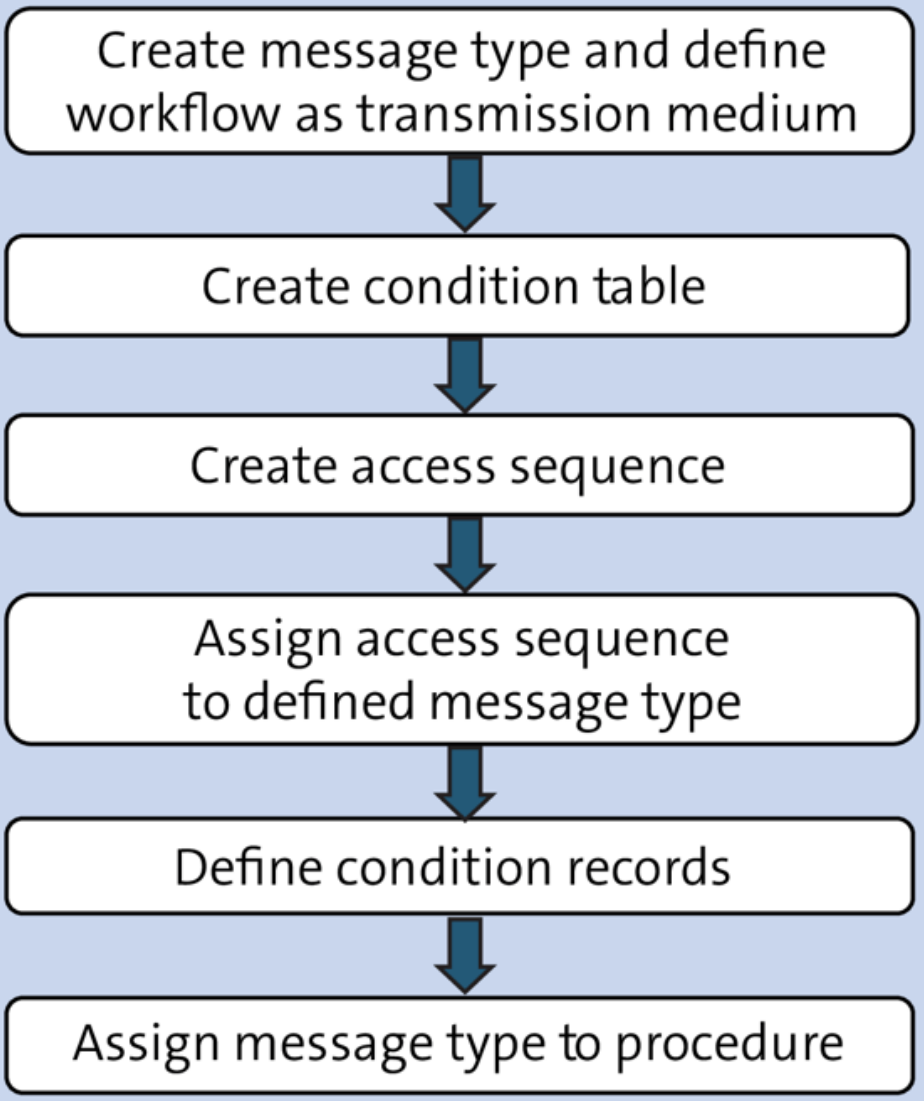


Figure 5.15 User Status Restrictions for Event Configuration



Maintain the processing program **RVNSWE01** with the FORM routine CREATE_EVENT for the output type EVEN and transmission medium 9. Possible only for MM and SD module.

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

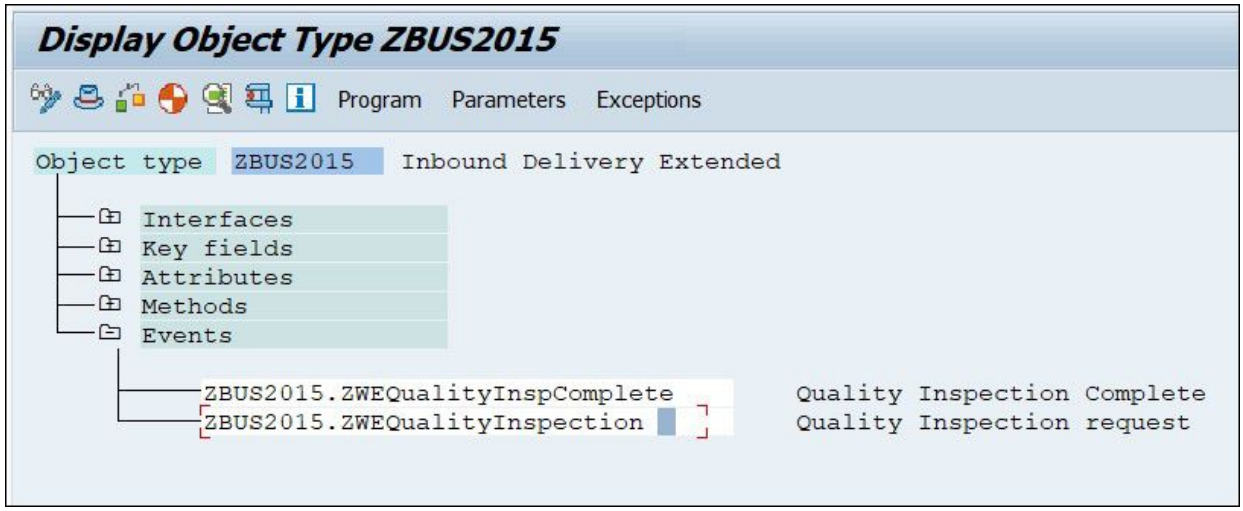


Figure 5.17 BOR Subtype Definition with Custom Events

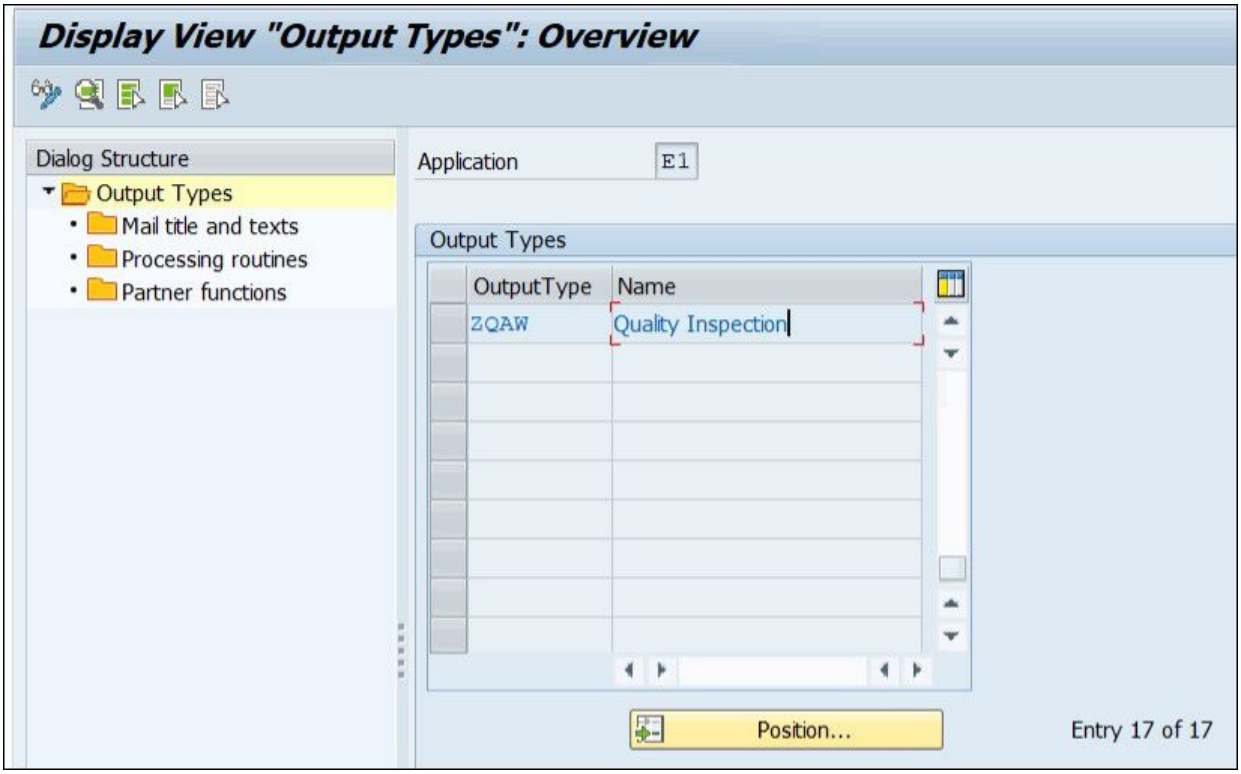


Figure 5.18 Custom Output Type Definition in Application E1 in Transaction NACE

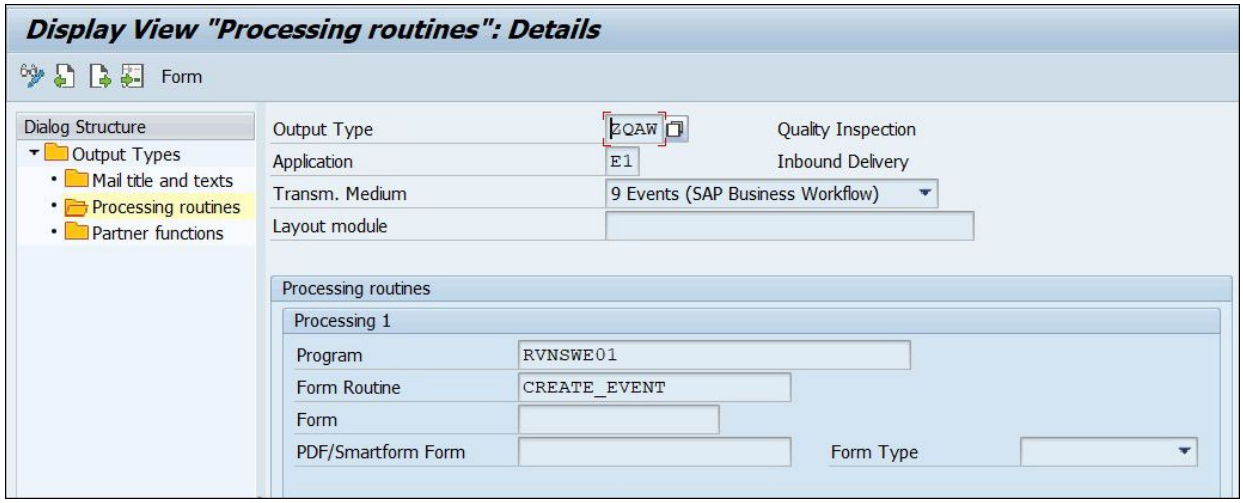


Figure 5.19 Maintenance of Processing Routine for Transmission Medium 9 in Transaction NACE

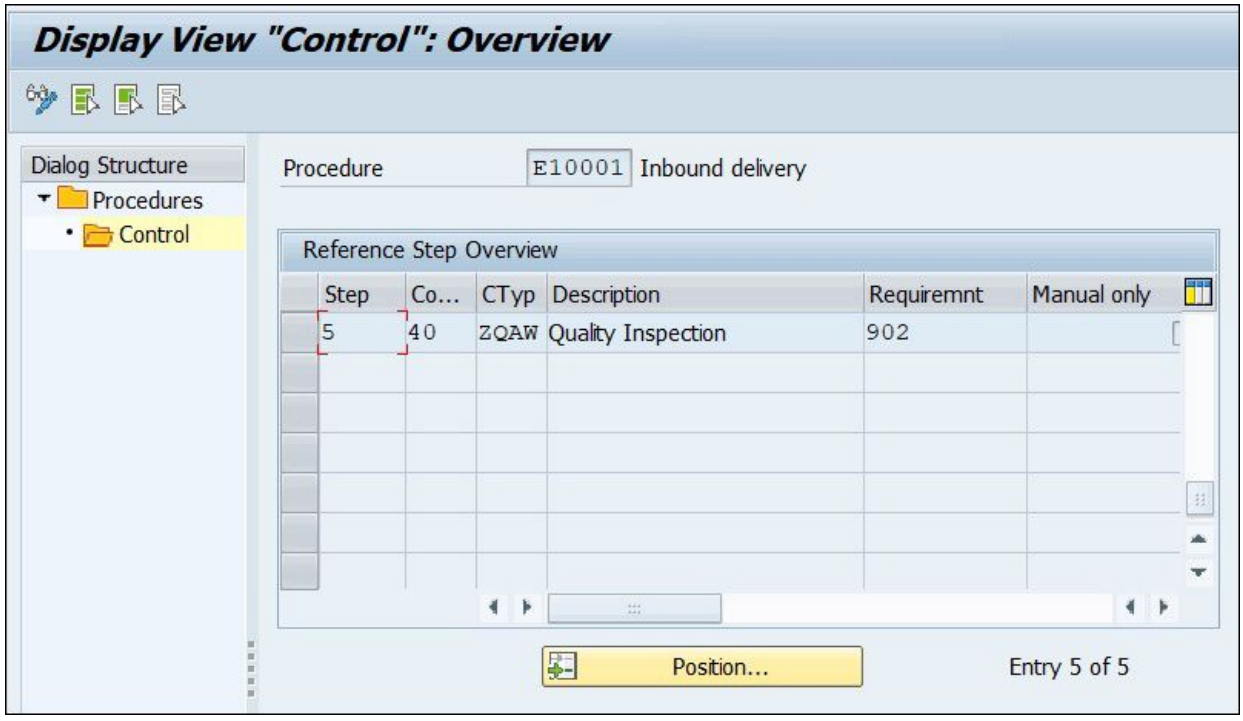


Figure 5.20 Addition of Custom Output Type to the Output Procedure along with Requirement Routine

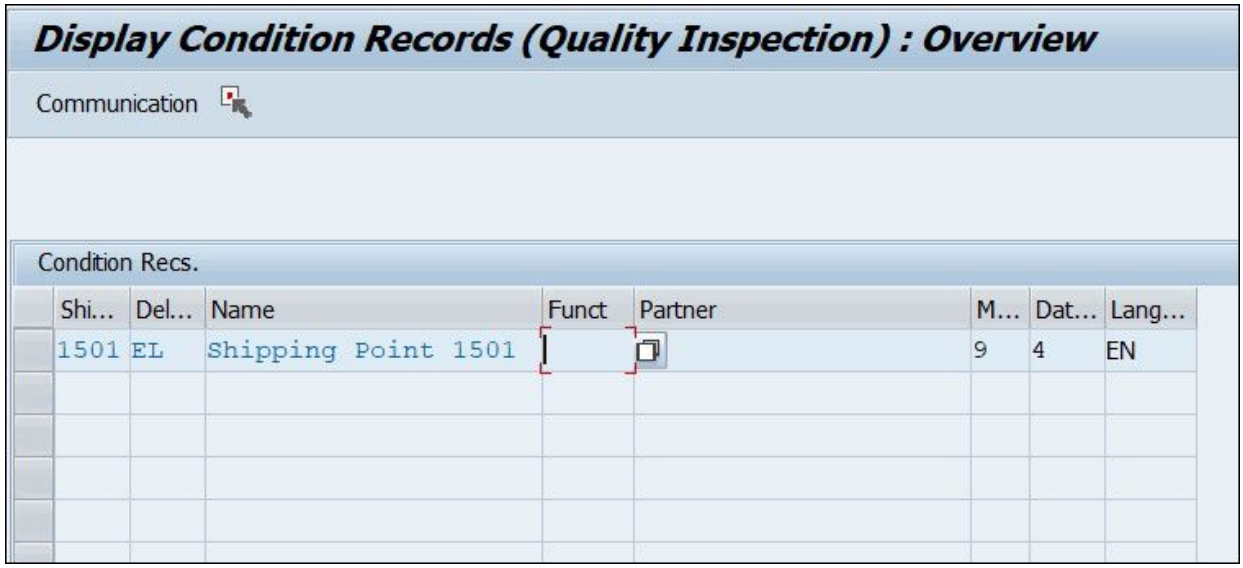


Figure 5.21 Condition Record Maintenance for Custom Output Type

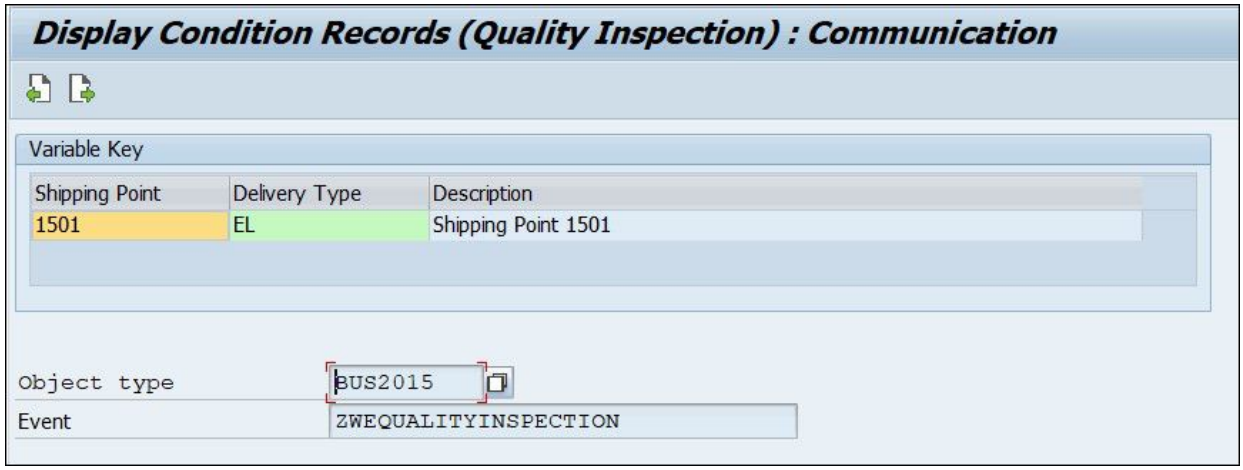


Figure 5.22 Enter the Communication Data for the Condition Record with BOR Type and Event

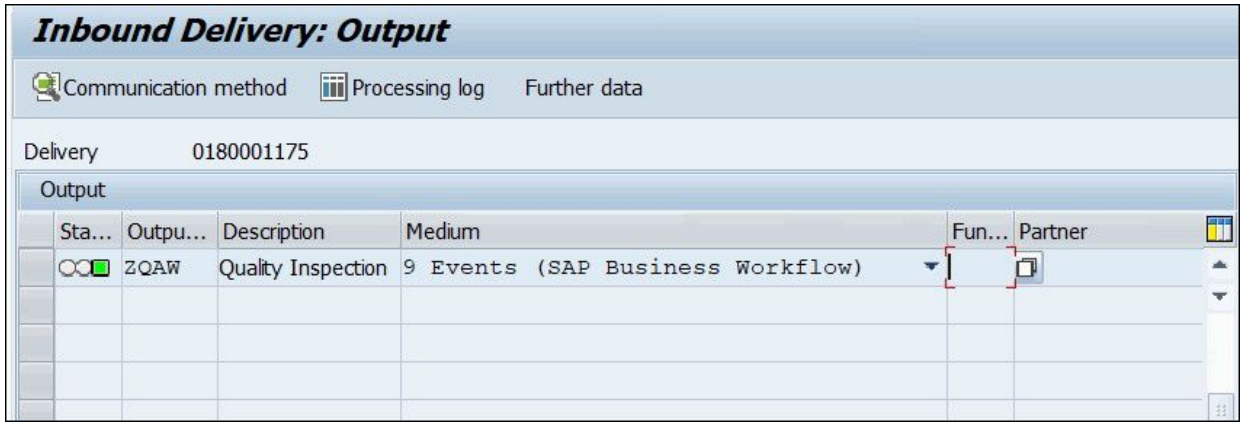


Figure 5.23 Workflow Event Output Generated after Post Goods Receipt of Inbound Delivery

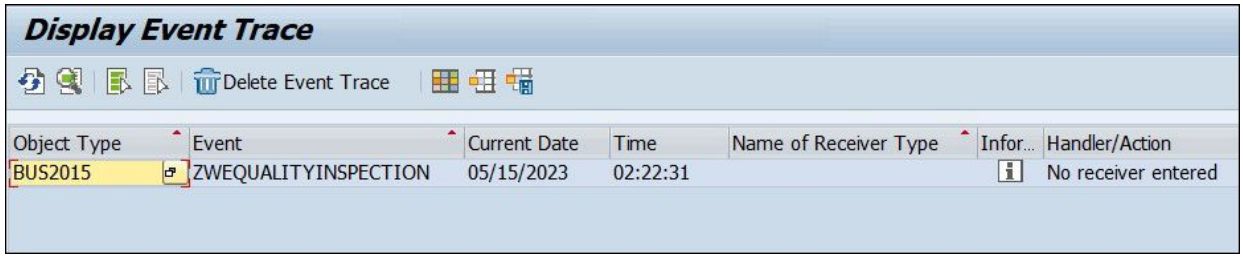


Figure 5.24 Event Trace Showing Event Triggered for the Output Type

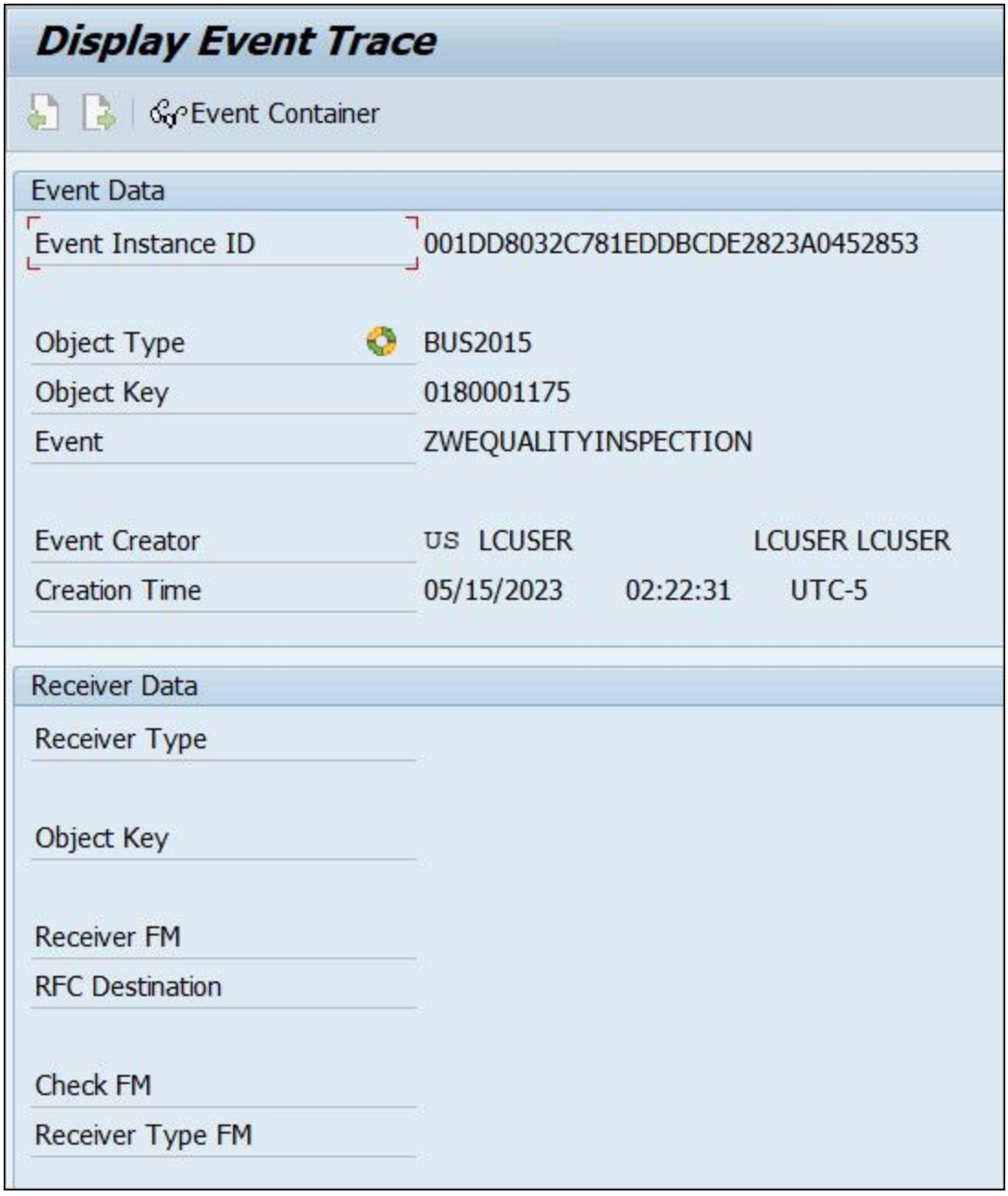


Figure 5.25 Event Trace Details Showing Event Creator Instance Information

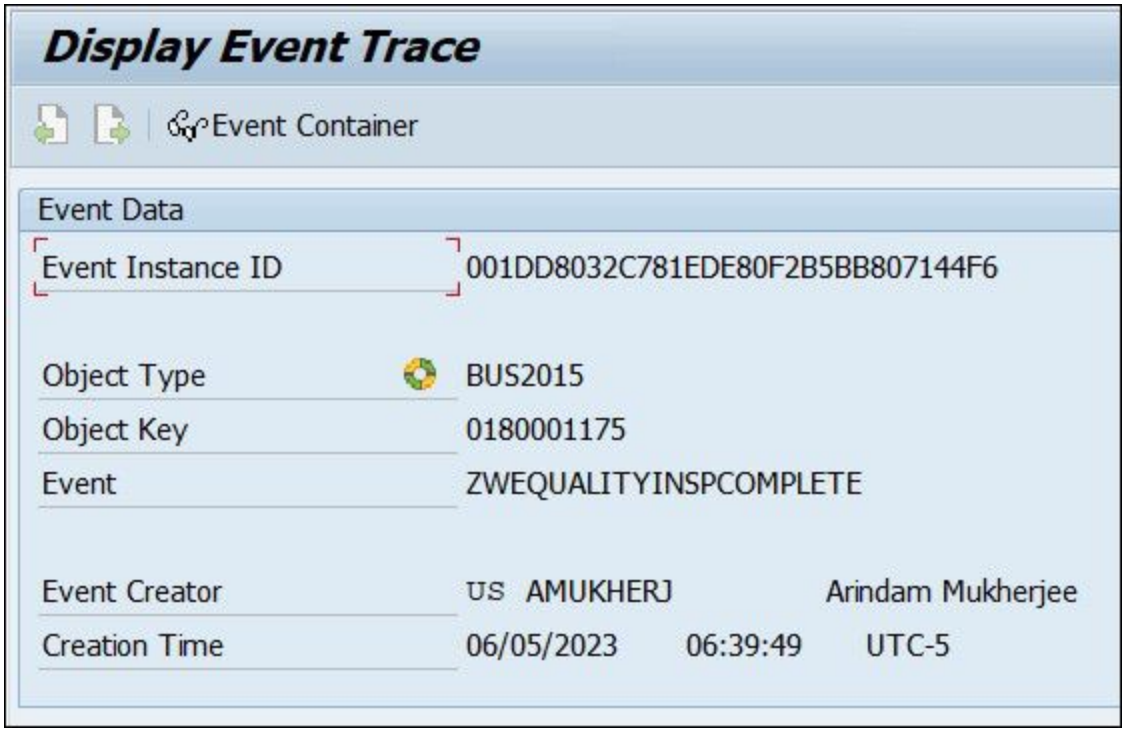


Figure 5.26 Event Trace Showing Event Creator Information

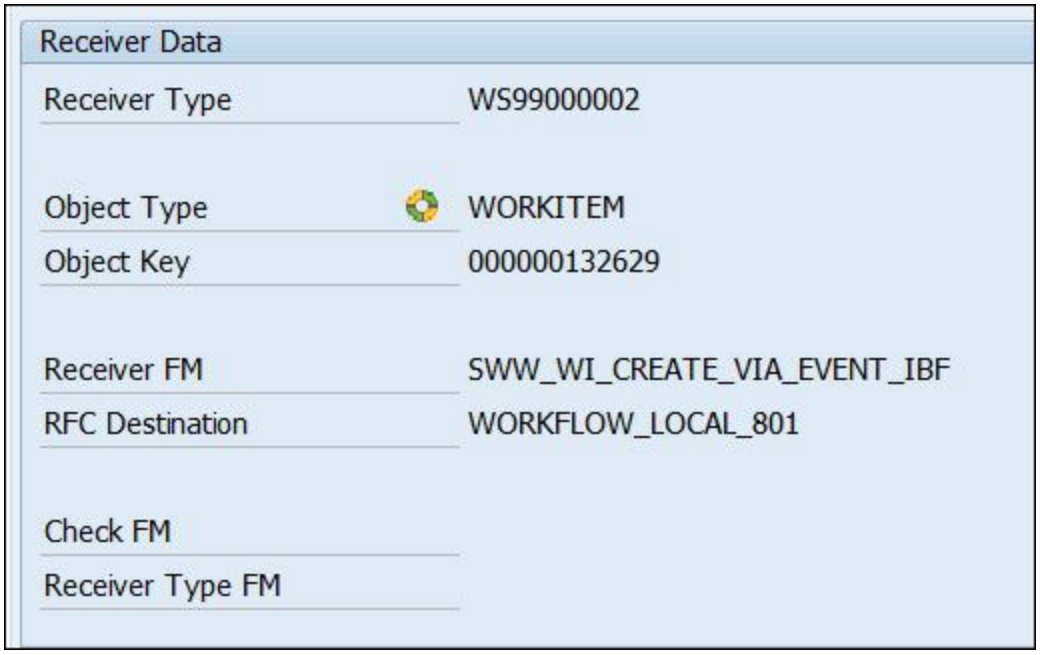


Figure 5.27 Event Trace Showing Event Receiver Information

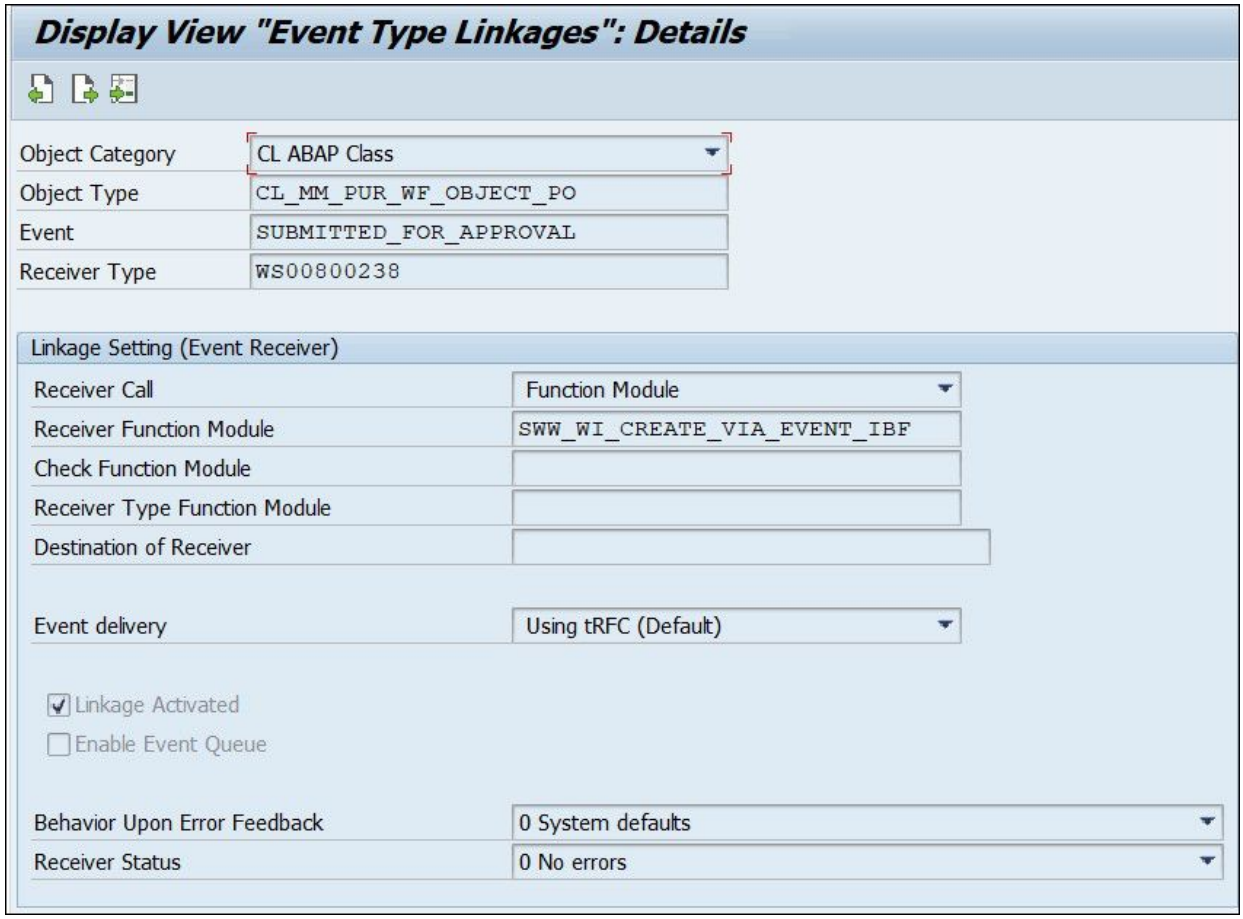


Figure 5.28 Event Linkage from Transaction SWE2 or Transaction SWETYPV

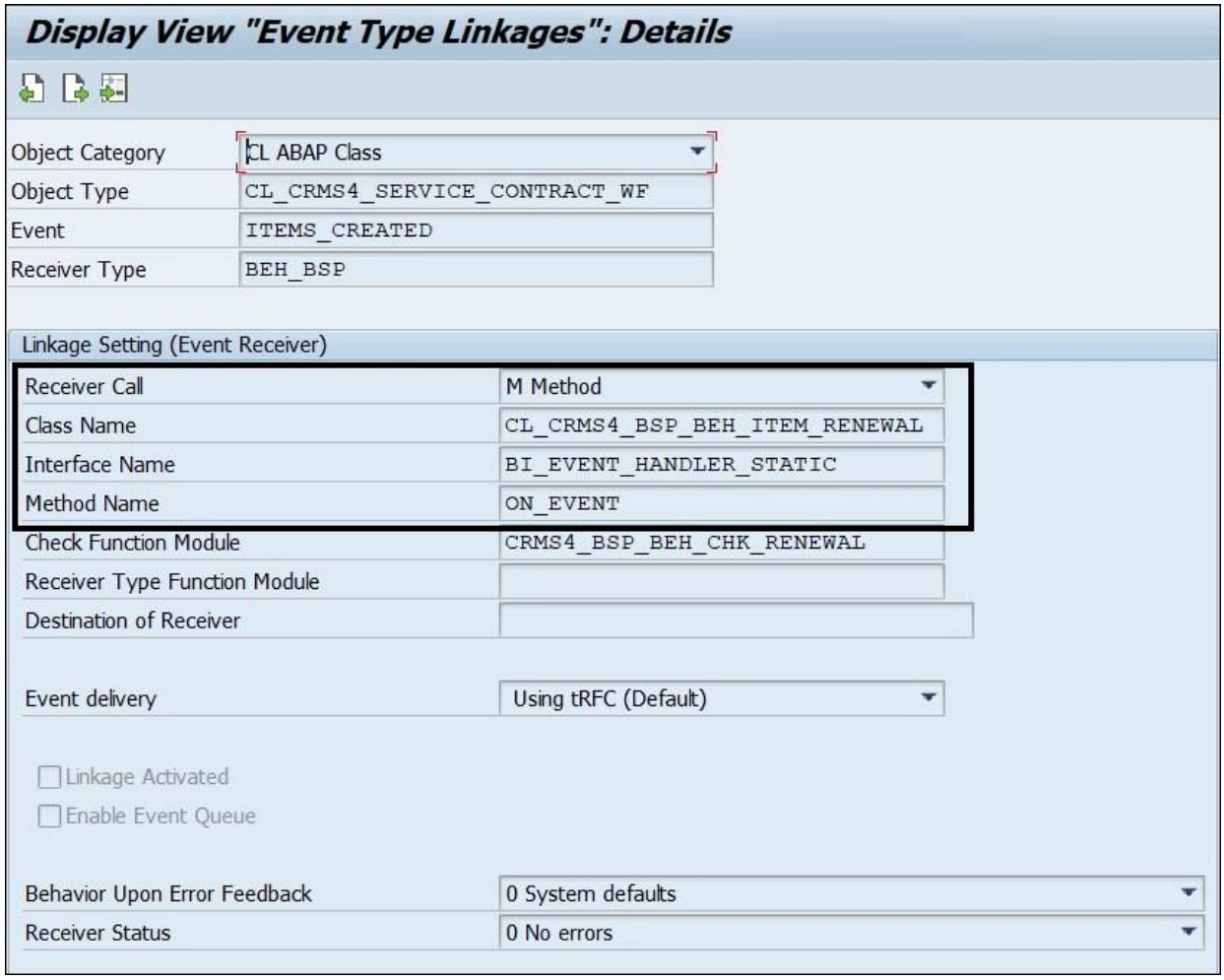


Figure 5.29 Sample Event Linkage Showing the Receiver Method Call

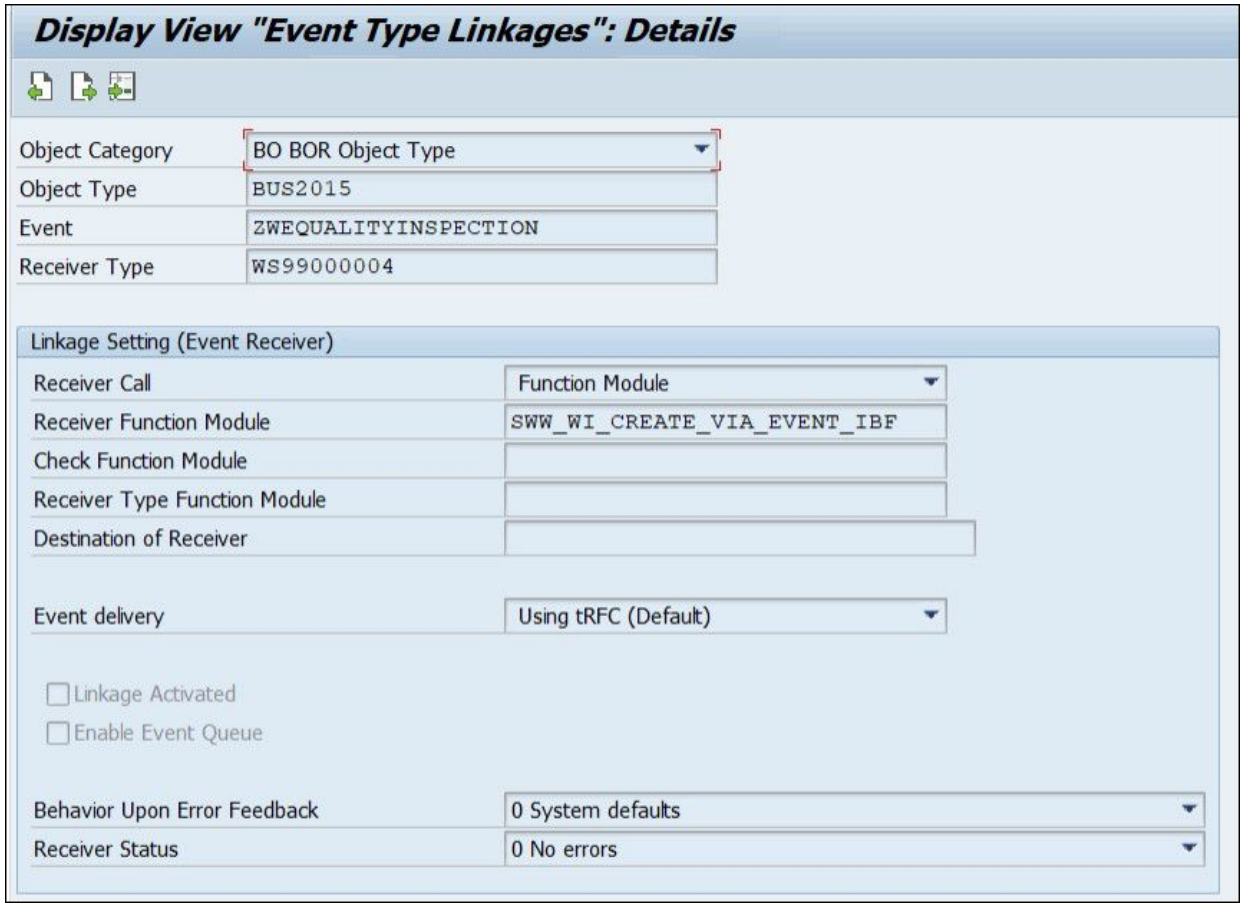


Figure 5.30 Event Linkage before Assigning a Start Condition

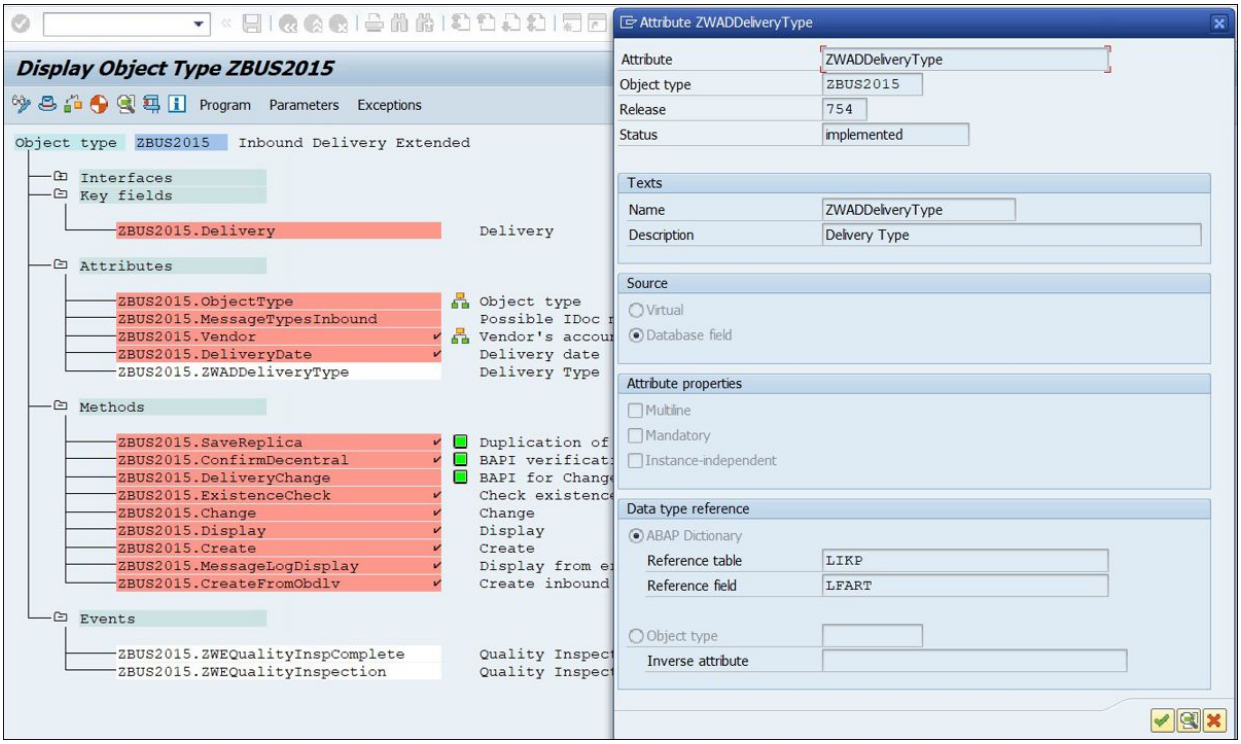


Figure 5.31 Creation of a Custom Attribute in the BOR Subtype Definition

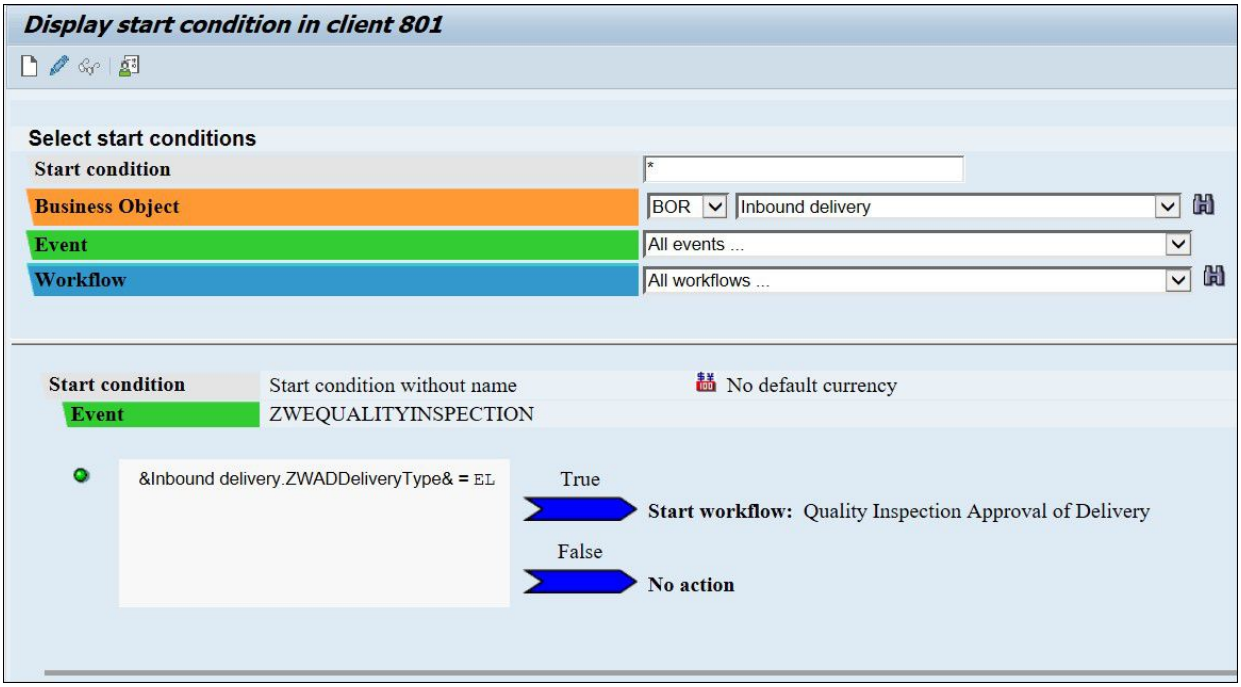


Figure 5.32 Creation of Start Condition for Event Linkage from Transaction SWB_COND

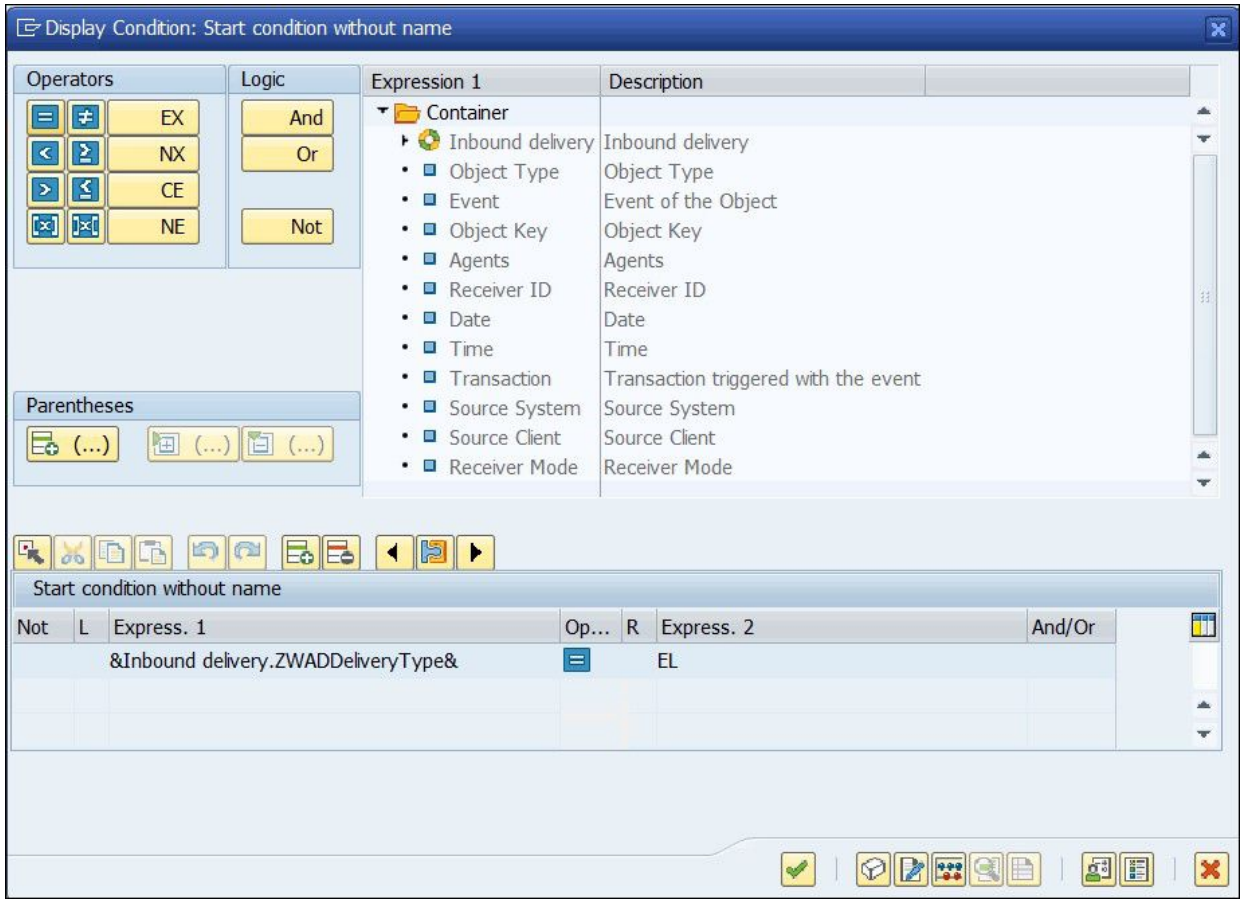


Figure 5.33 Maintaining the Start Condition Using the Condition Editor in Transaction SWB_COND

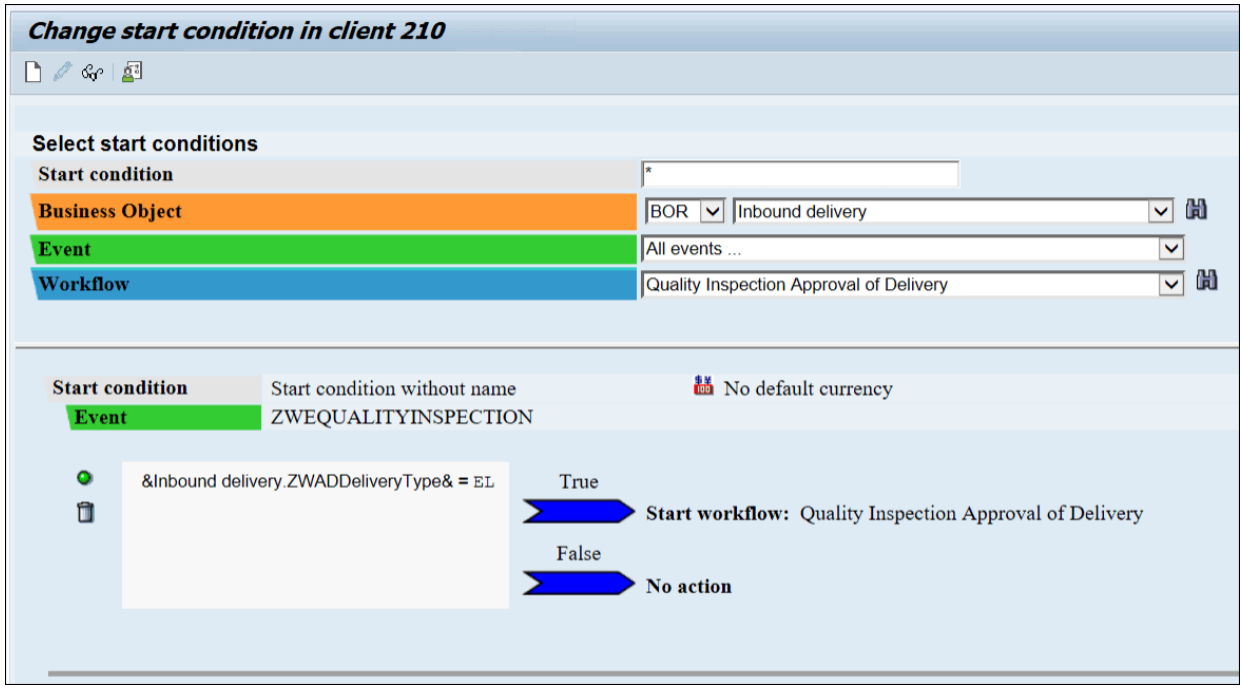


Figure 5.34 Activate the Start Condition (Indicated by the Green Icon to the Left)

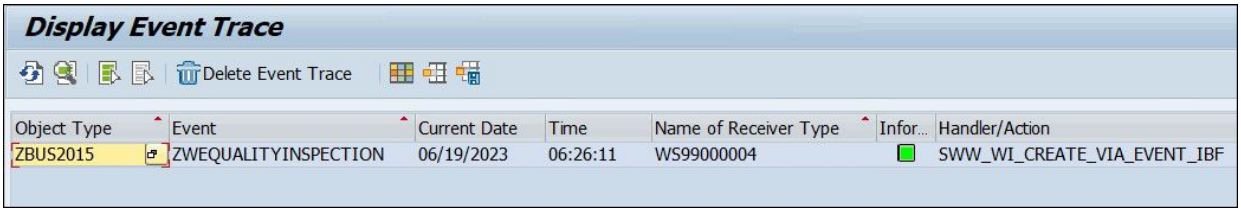


Figure 5.35 Event Trace Showing Event Triggered after Evaluation of Start Condition

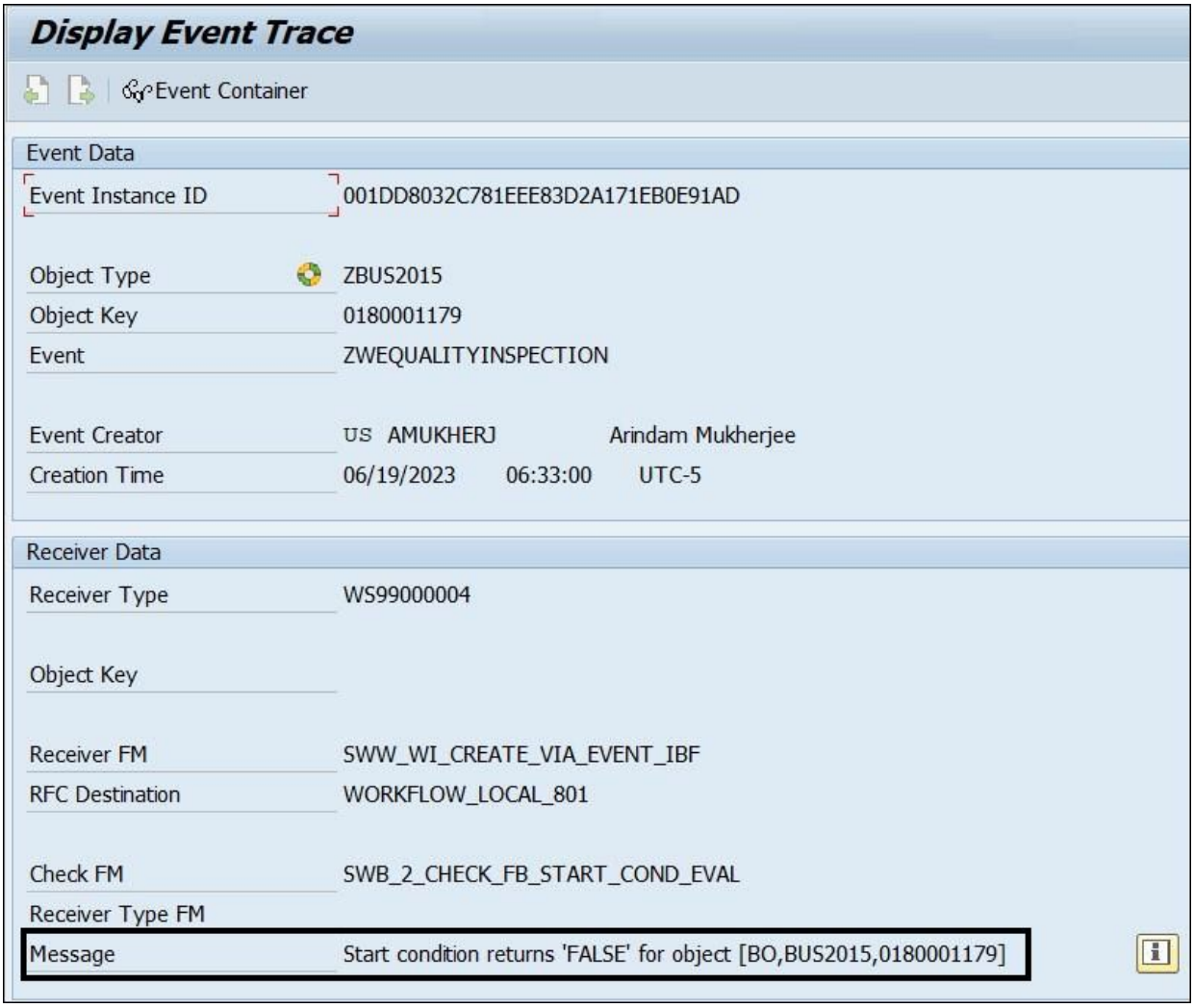


Figure 5.36 Event Trace Showing Exception Raised due to Start Condition Being Evaluated to False

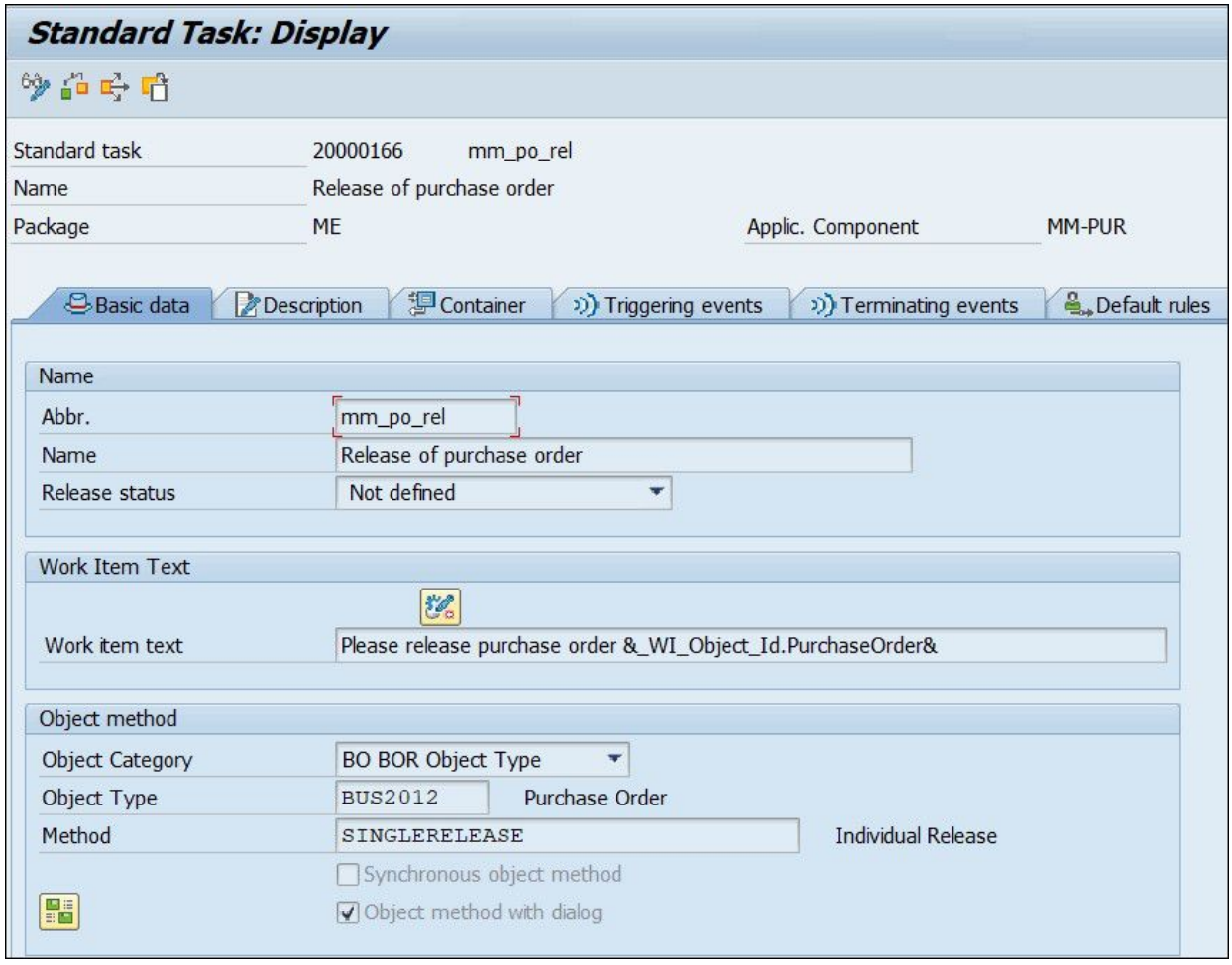


Figure 5.37 Example of an Asynchronous Standard Task

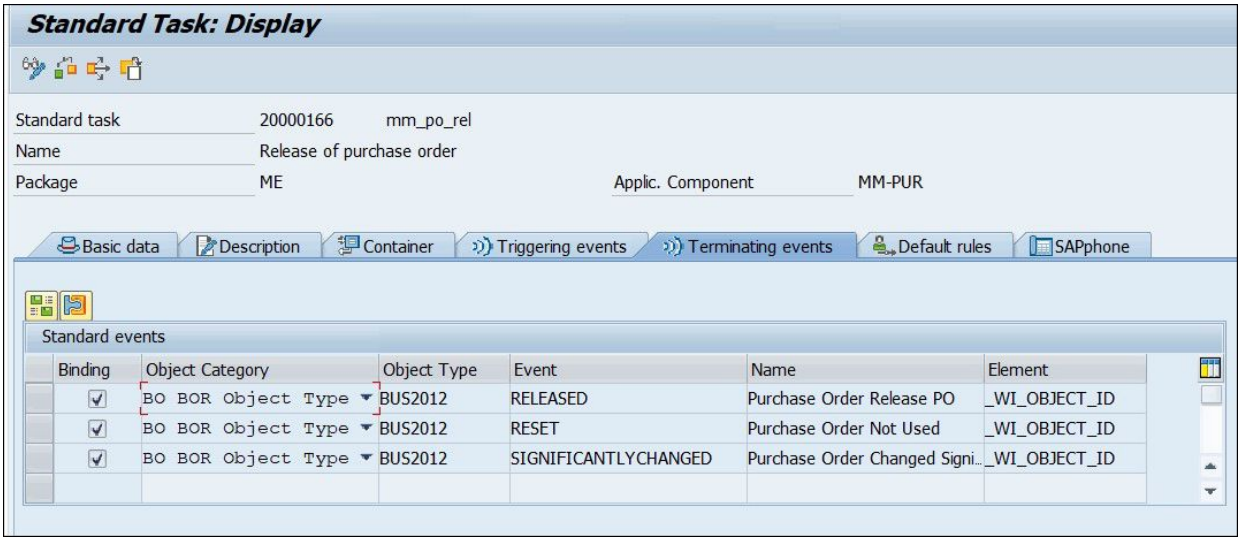


Figure 5.38 Terminating Events in an Asynchronous Dialog Task

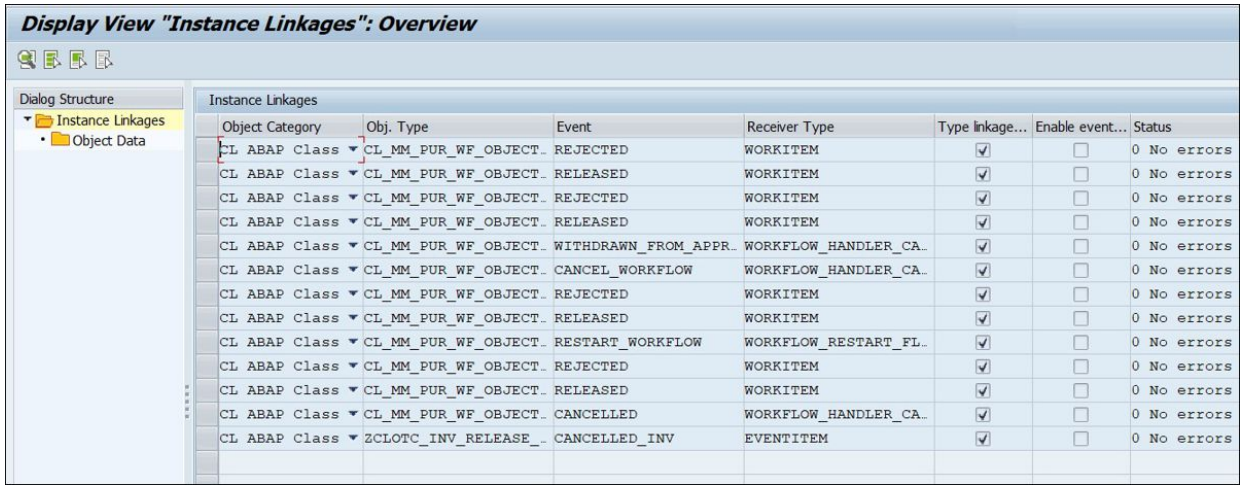


Figure 5.39 Instance Linkages from Transaction SWE3 or Transaction SWEINST

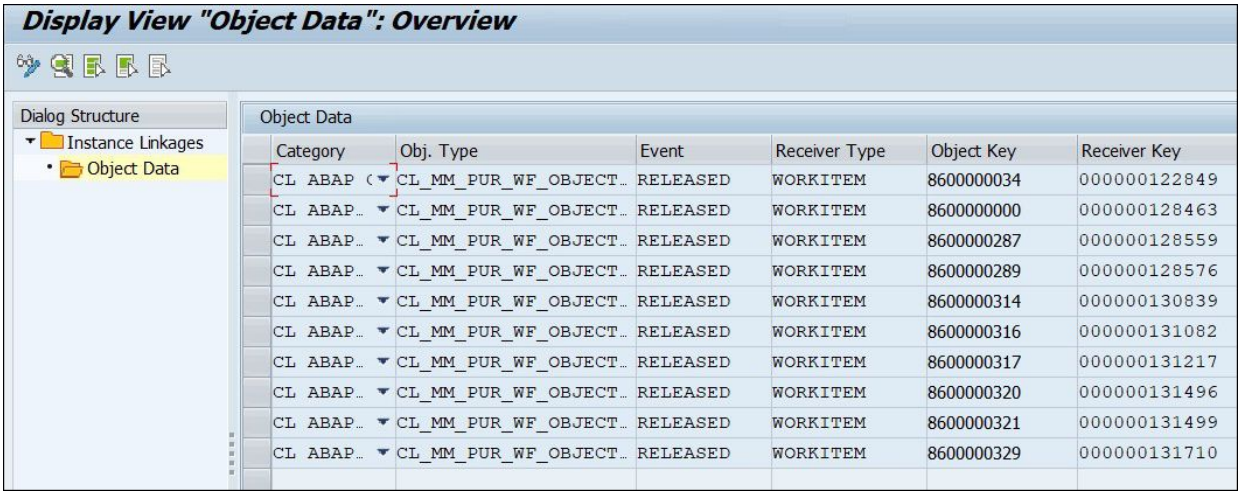


Figure 5.40 Object Instances for an Instance Linkage from Transaction SWE3 or Transaction SWEINST

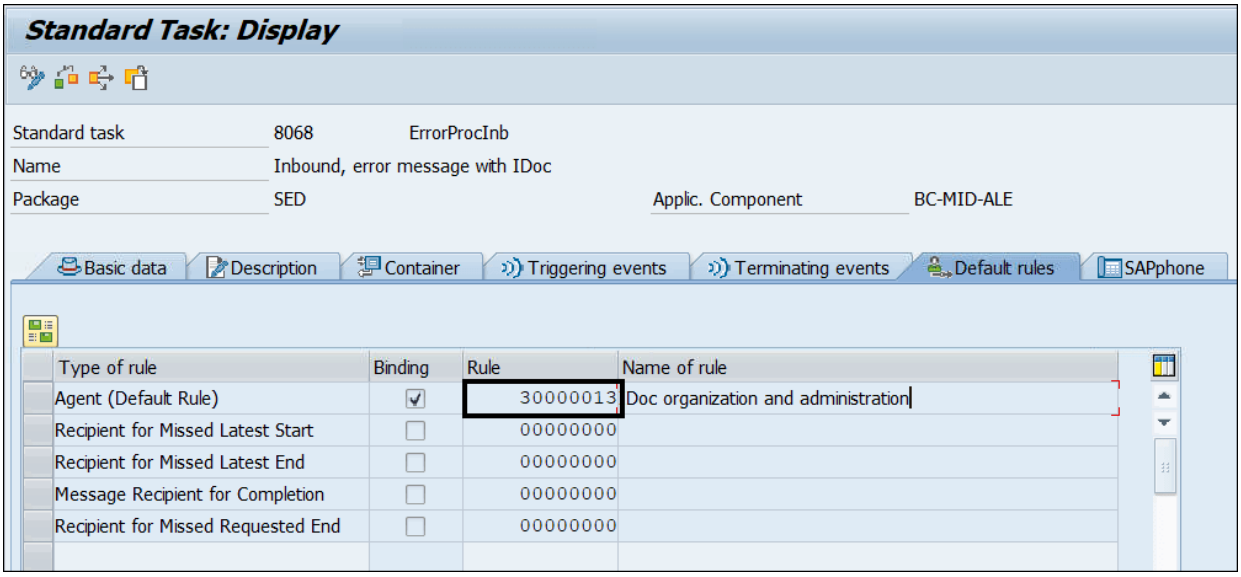


Figure 6.1 Agent Determination with a Default Rule

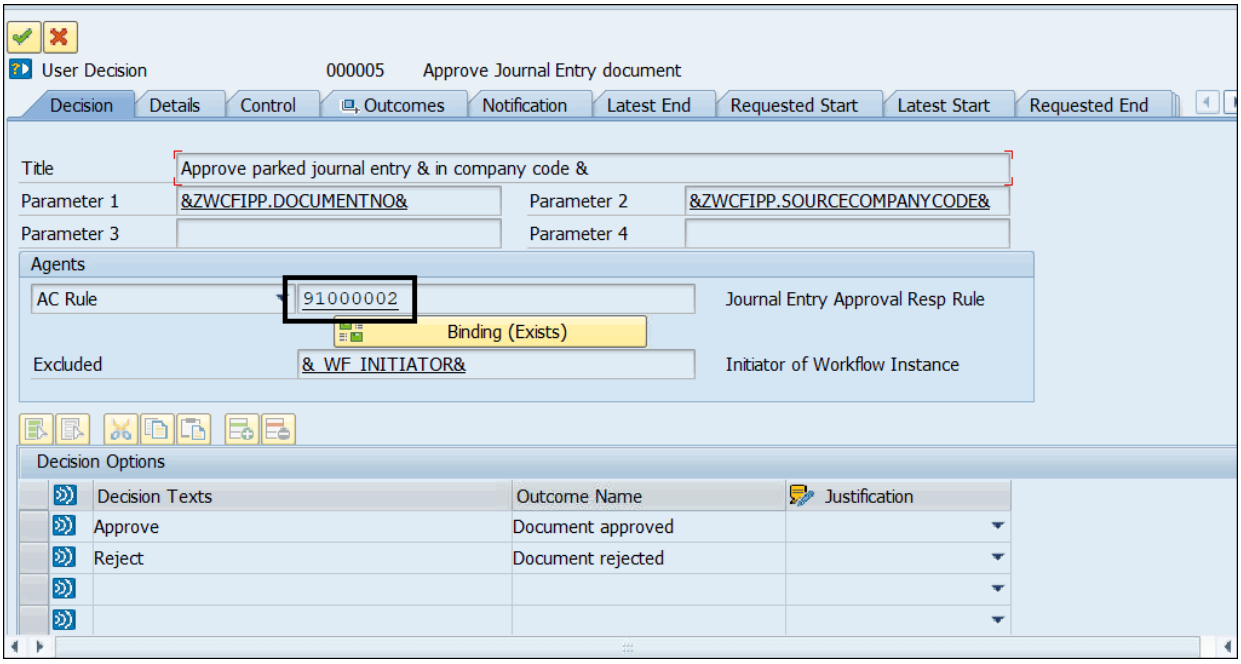


Figure 6.2 Agent Determination with a Rule at the Workflow Step Level

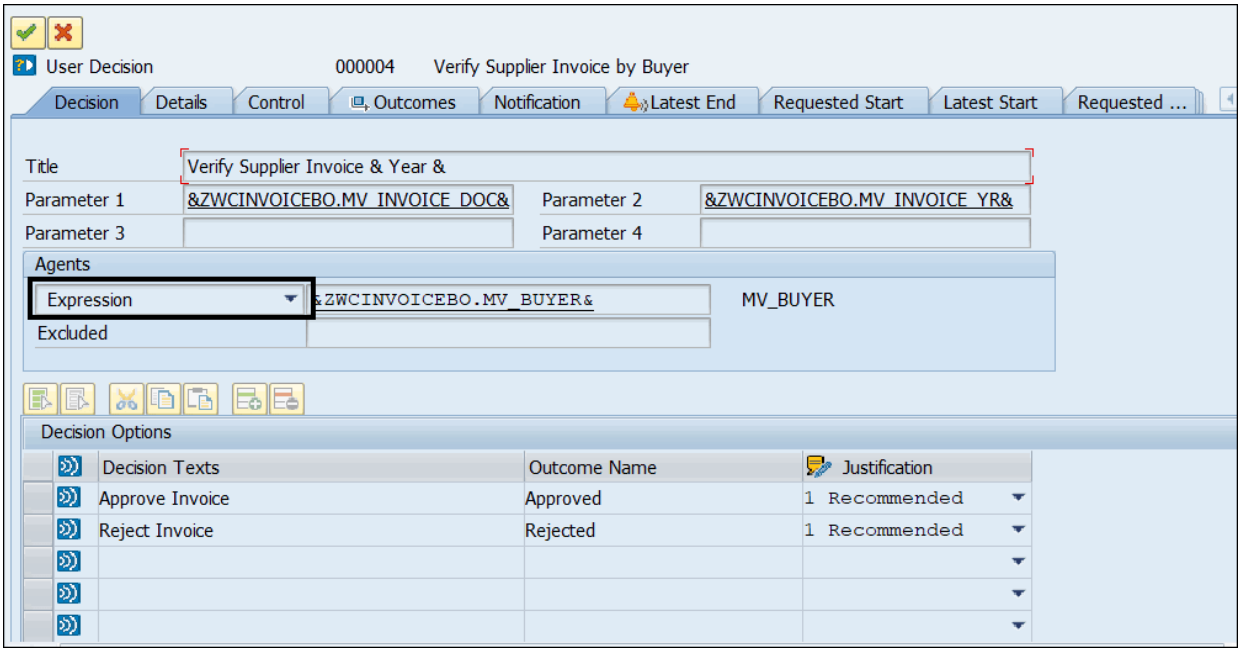


Figure 6.3 Agent Determination with an Expression at the Workflow Step Level

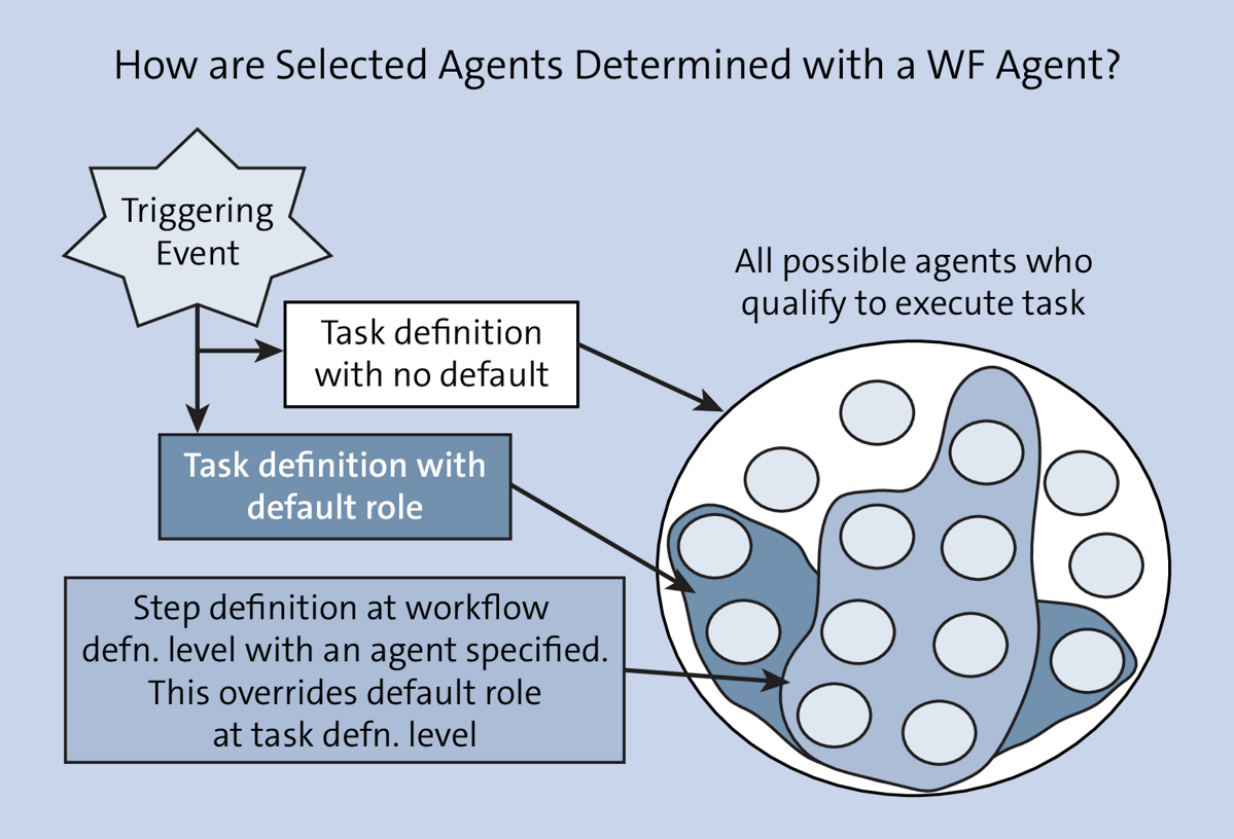


Figure 6.4 Selected Agent Determination at the Workflow Step Level

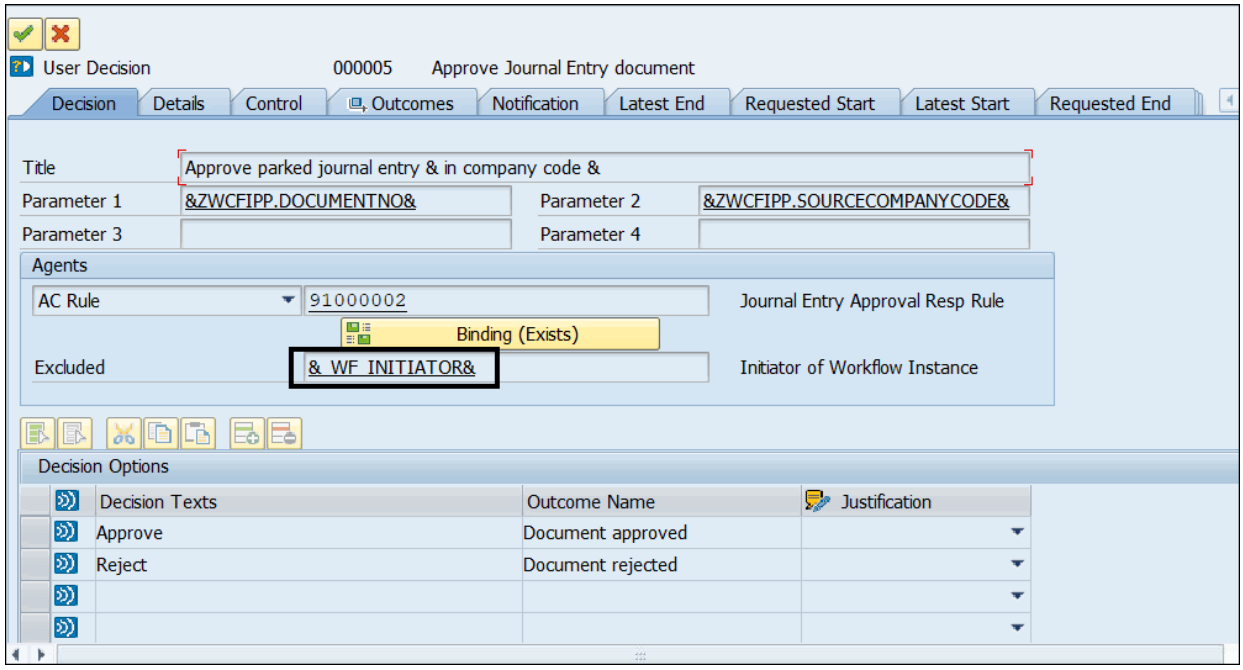


Figure 6.5 Assigning Excluded Agents at the Workflow Step Level with an Expression

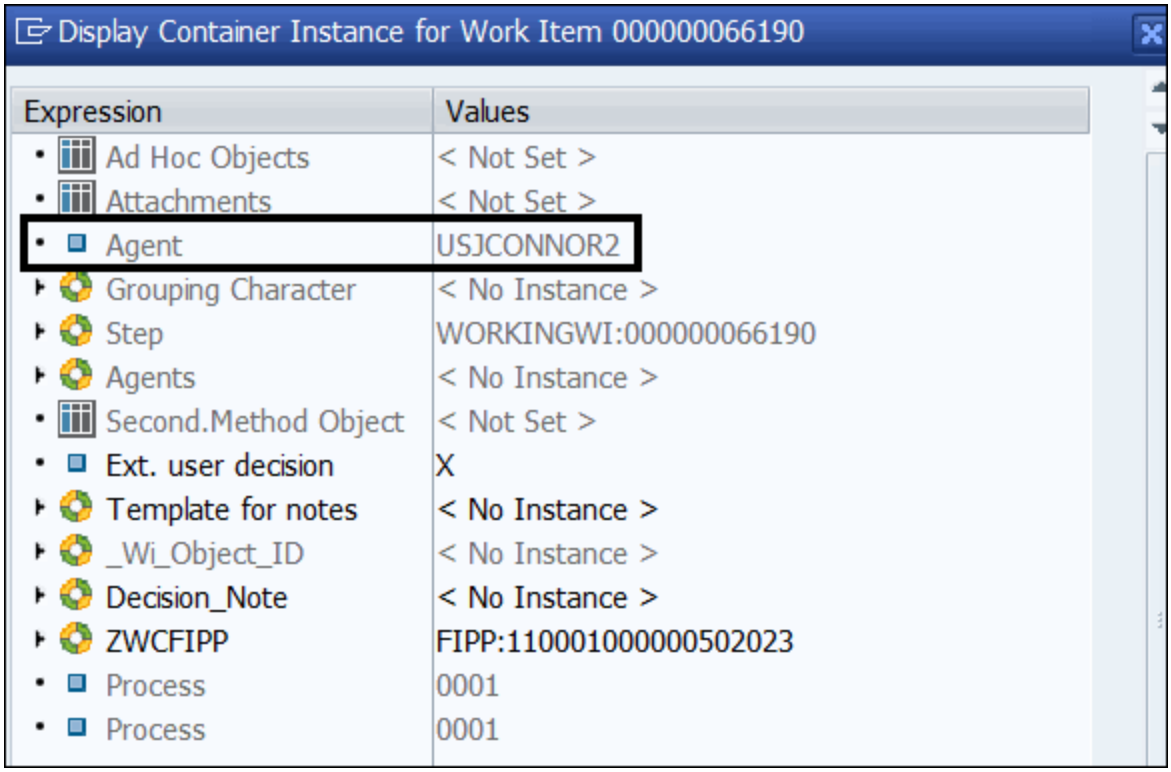


Figure 6.6 Actual Agent of a Completed Dialog Step Captured in System Element _Wi_Actual_Agent (Agent)

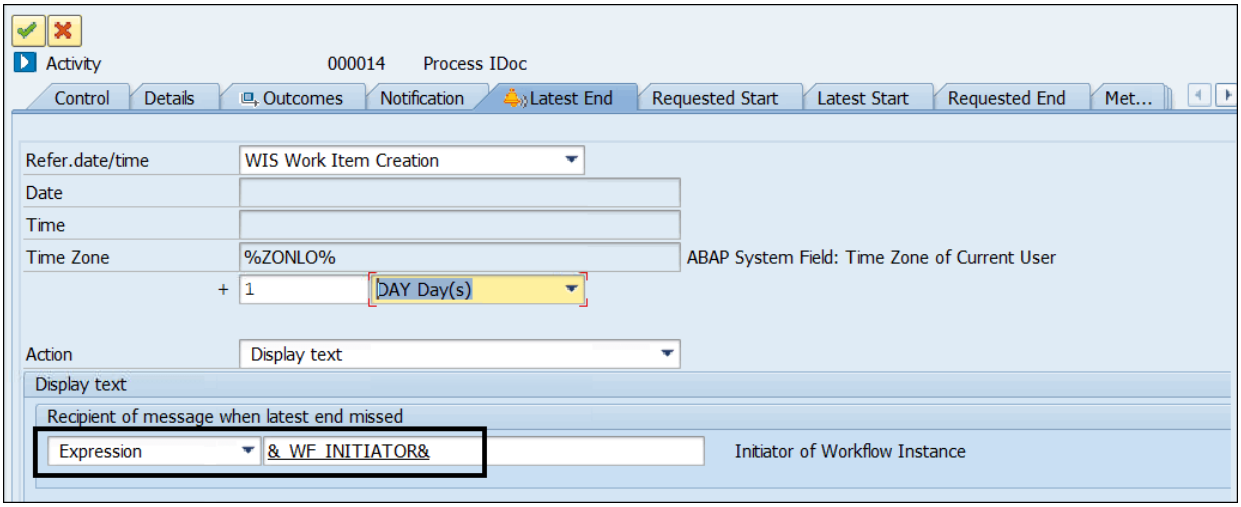


Figure 6.7 Deadline Agent Assignment at the Workflow Step Level in a Deadline Tab

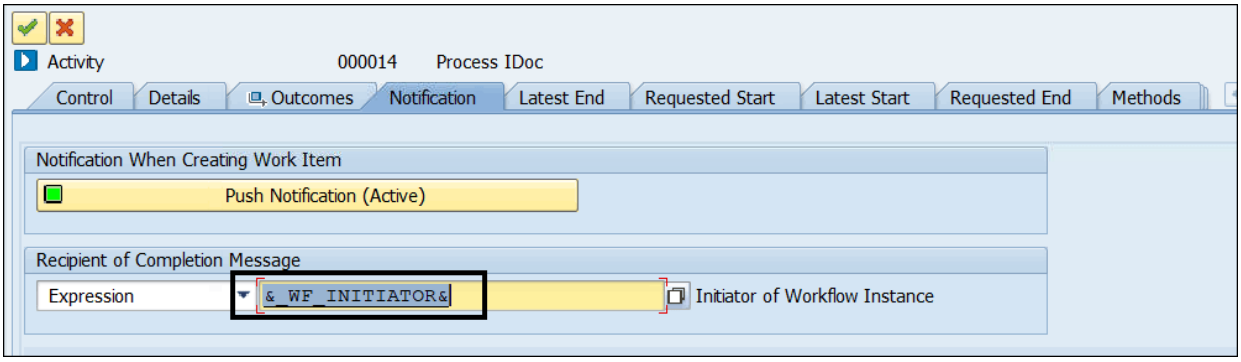


Figure 6.8 Maintain the Agent Assignment for the Work Item Completion Message at the Workflow Step Level

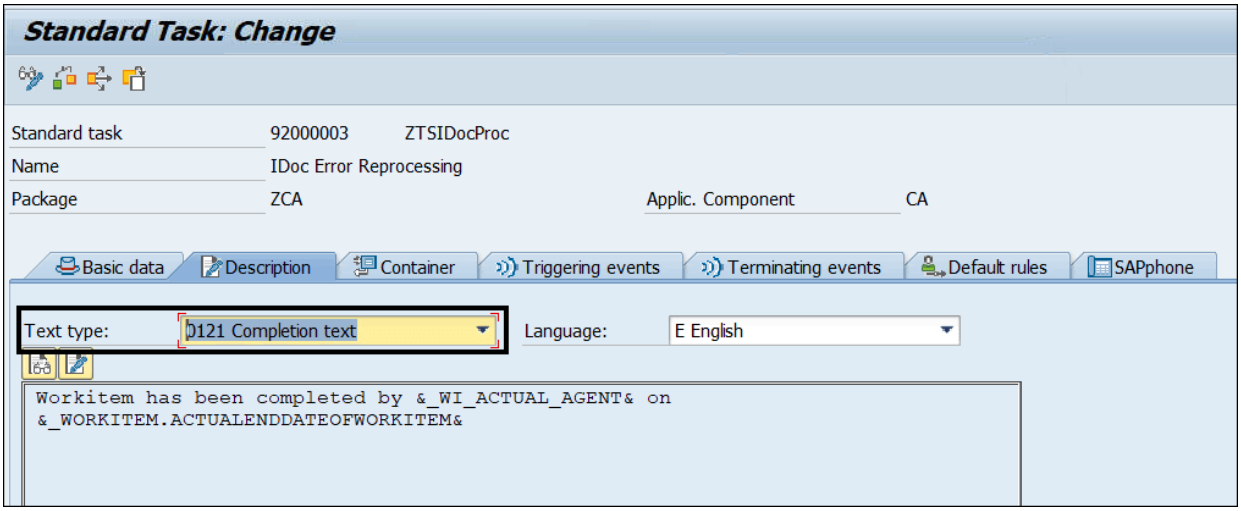


Figure 6.9 Maintain Notification of Completion Text in the Task Description Tab

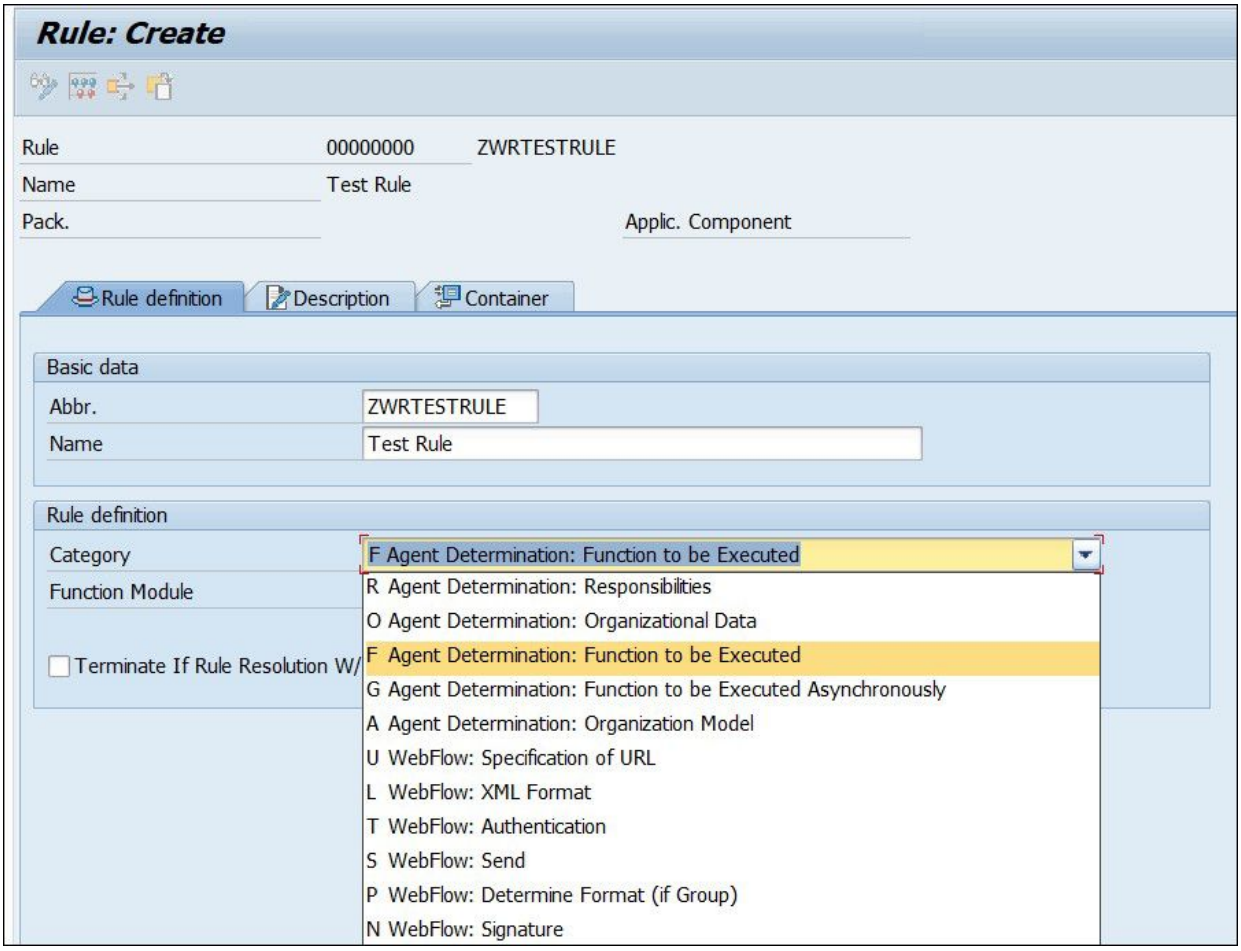


Figure 6.10 Creating a Rule from Transaction PFAC

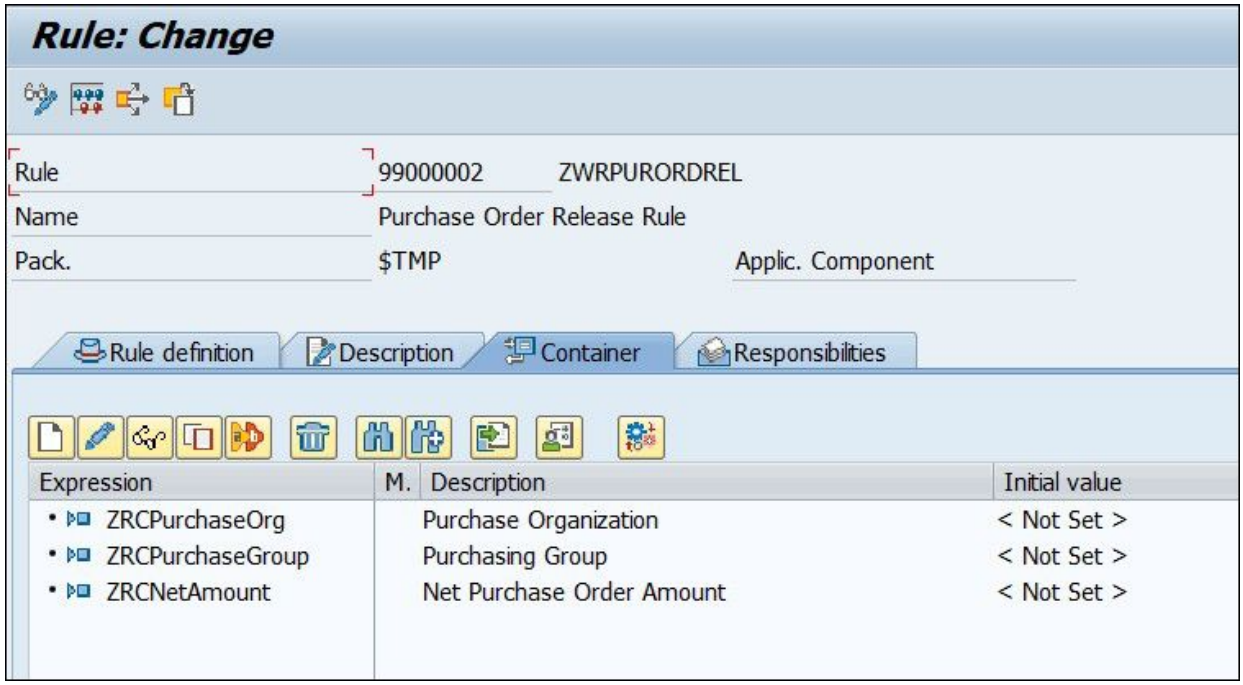


Figure 6.11 View of the Container Definition for Rule

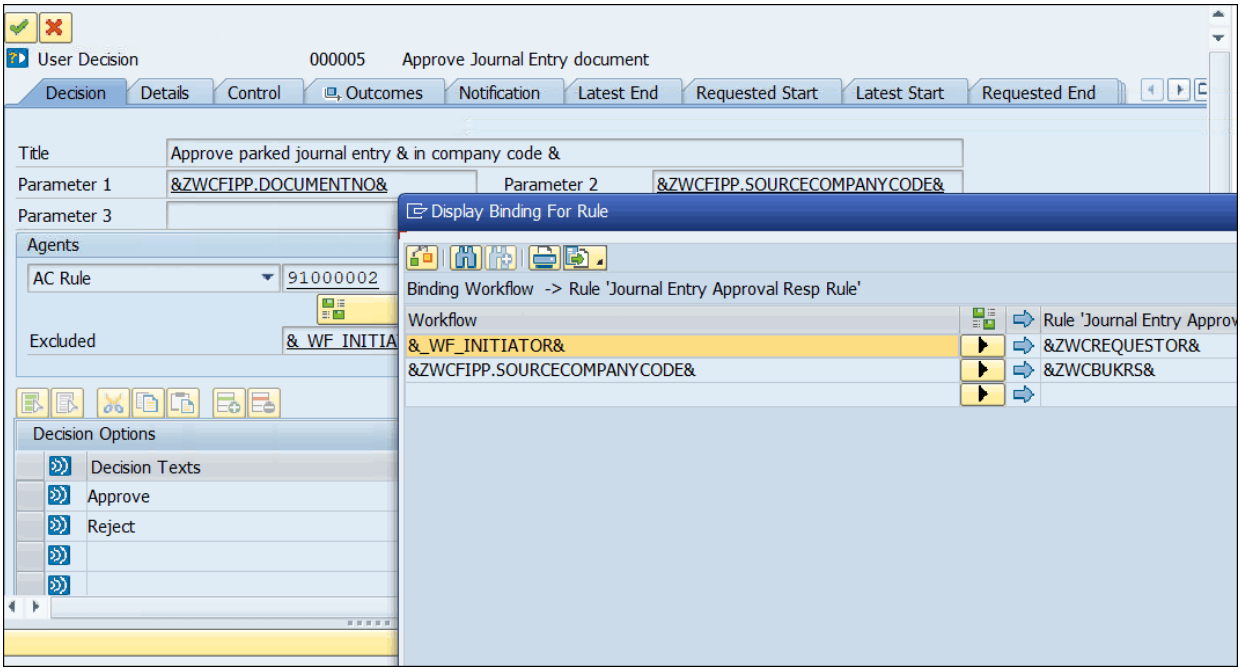


Figure 6.12 Example of Binding between the Workflow Container and Rule Container

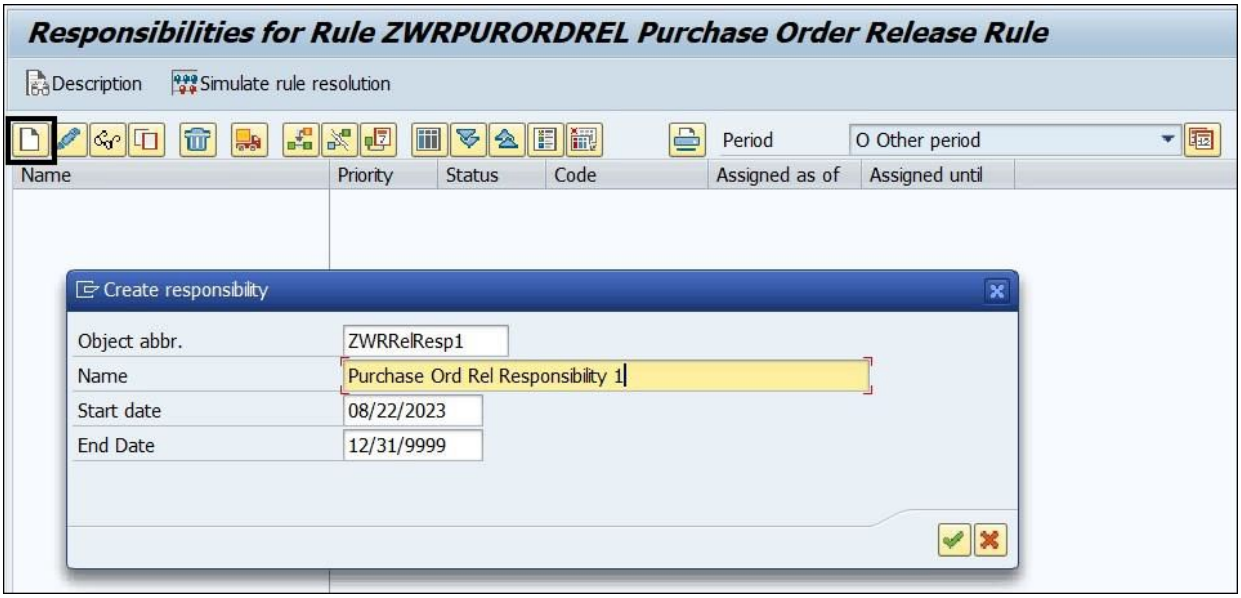


Figure 6.13 Responsibility Creation Screen in Transaction OOCU_RESP

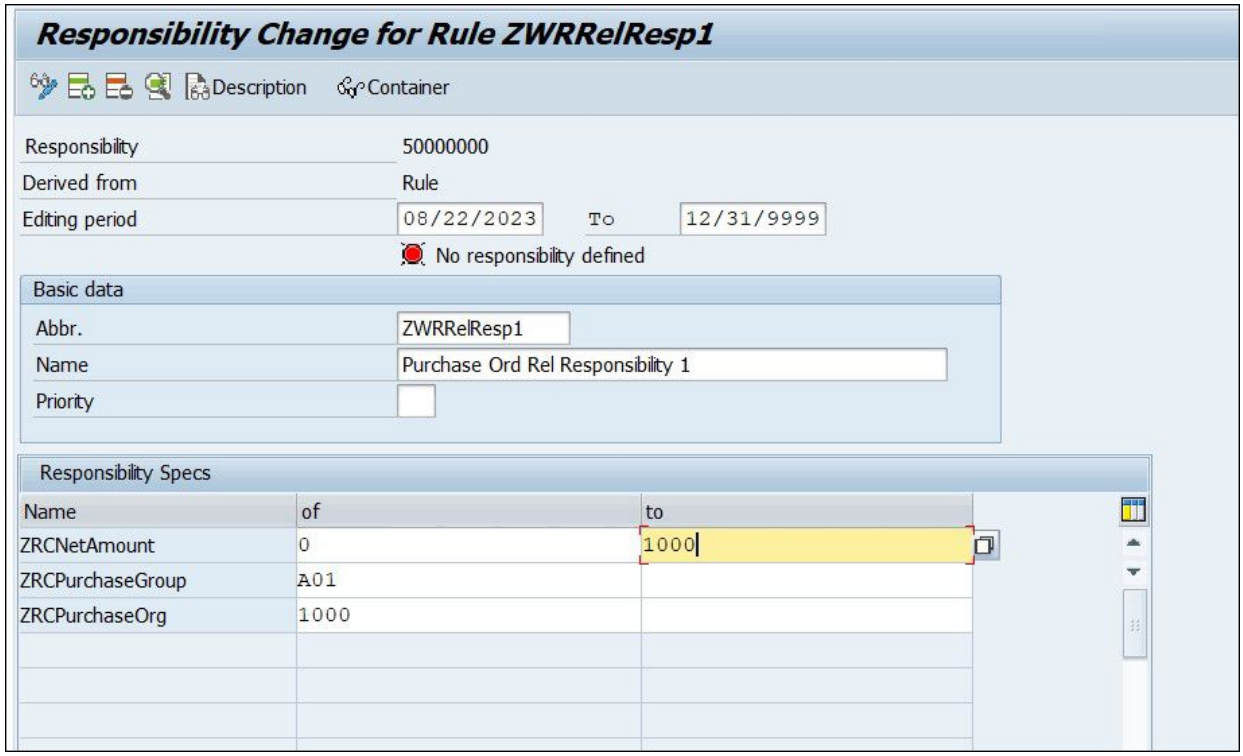


Figure 6.14 Responsibility Definition Details View inside Transaction OOCU_RESP

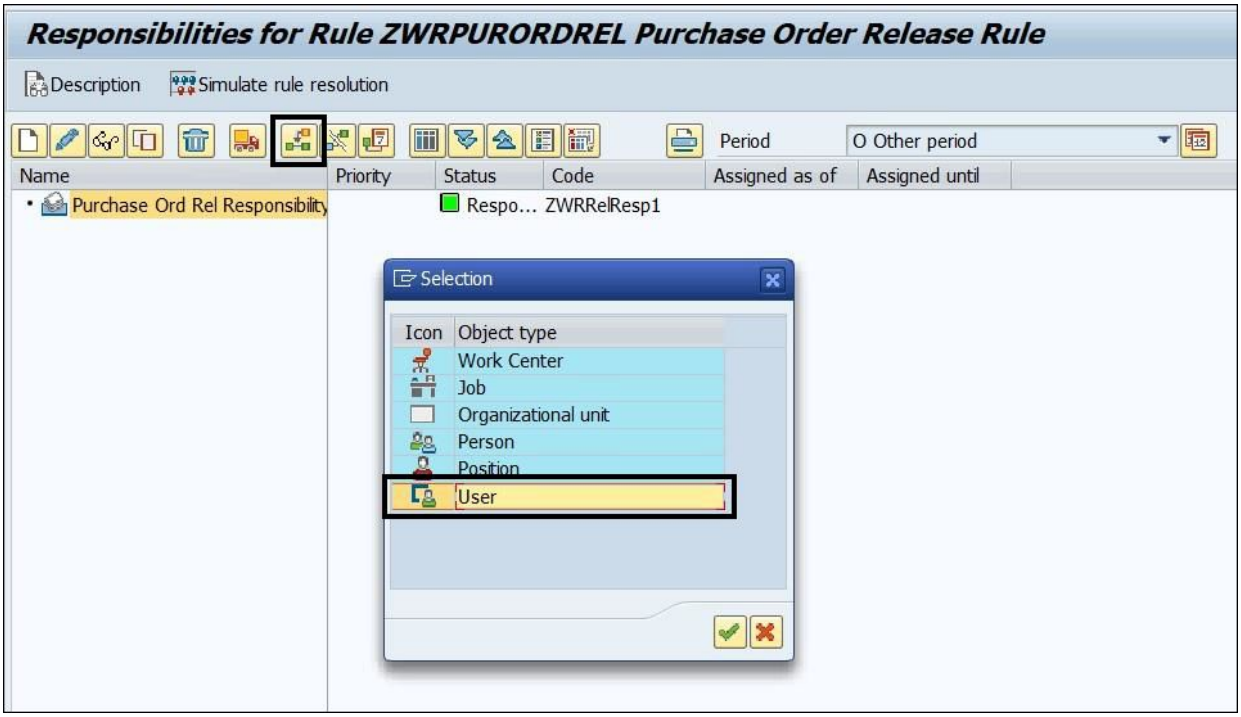


Figure 6.15 Agent Assignment for Rule Responsibility Definition: Selection of Agent Type

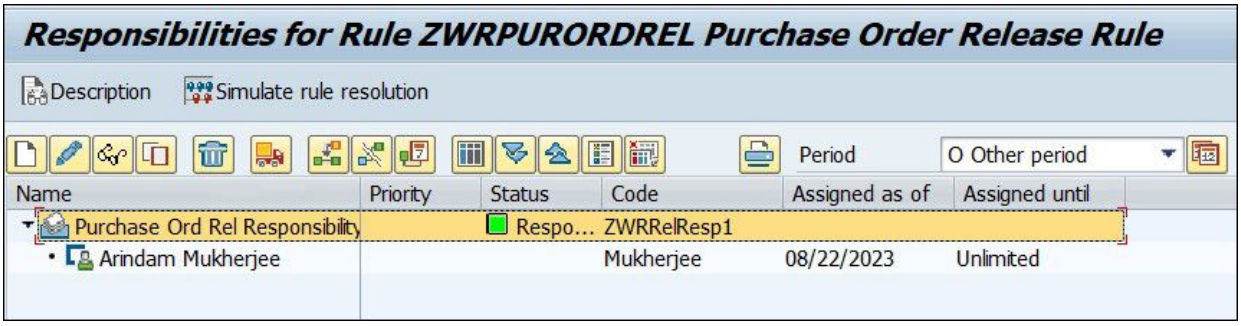


Figure 6.16 Agent Assignment Created for the Responsibility Definition

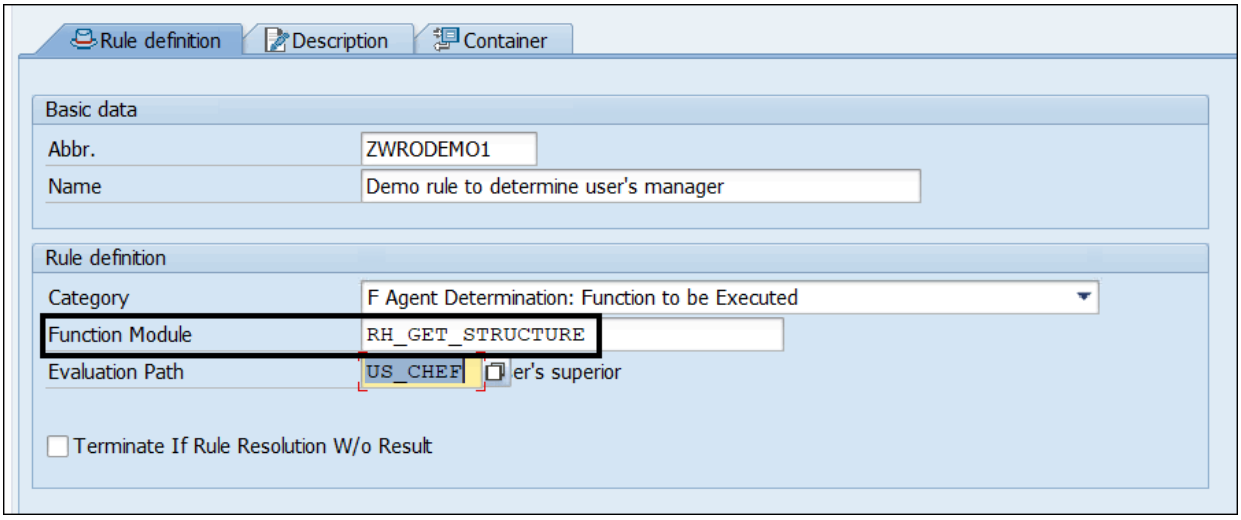


Figure 6.17 Rule with a Function Module Using an Evaluation Path

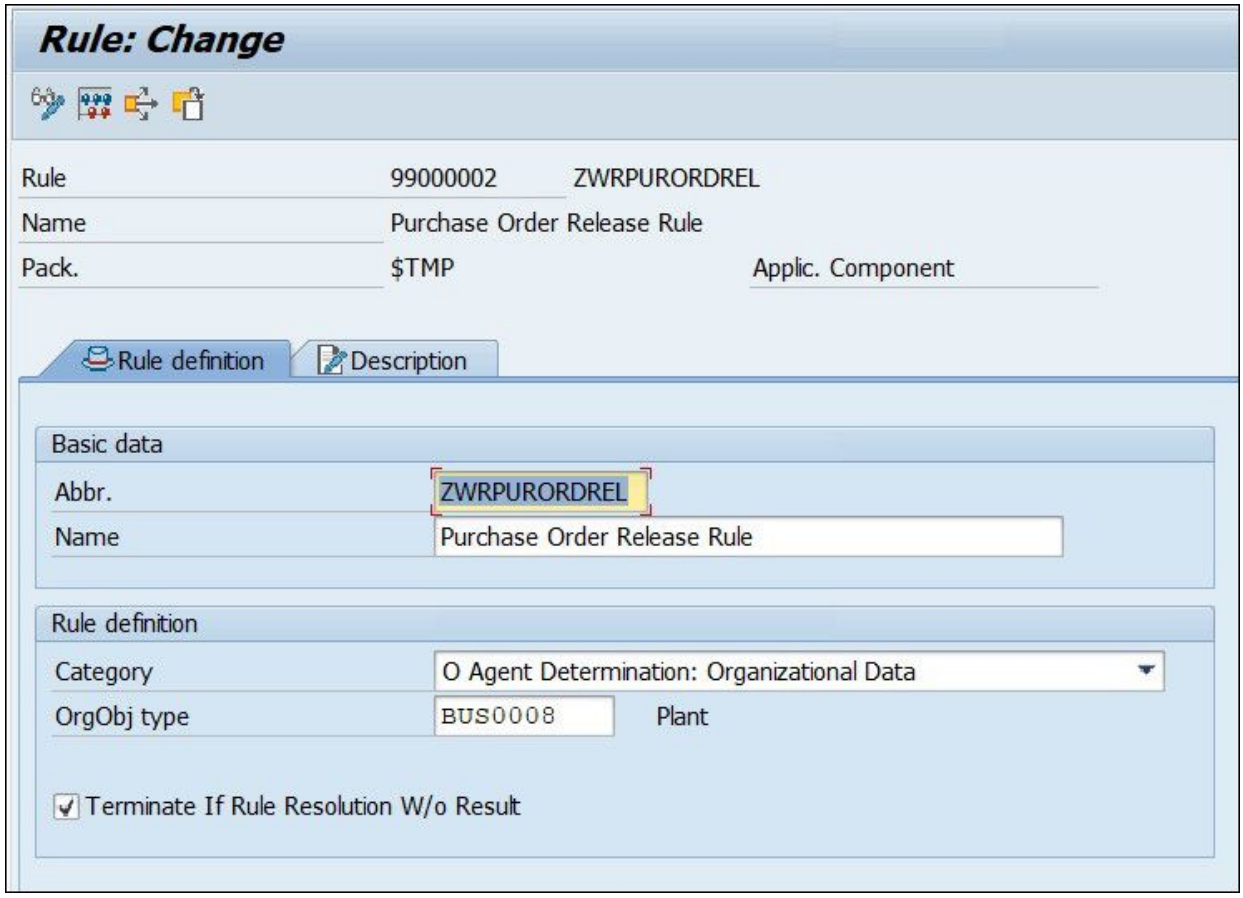


Figure 6.18 Rule Definition Using Organization Data Approach

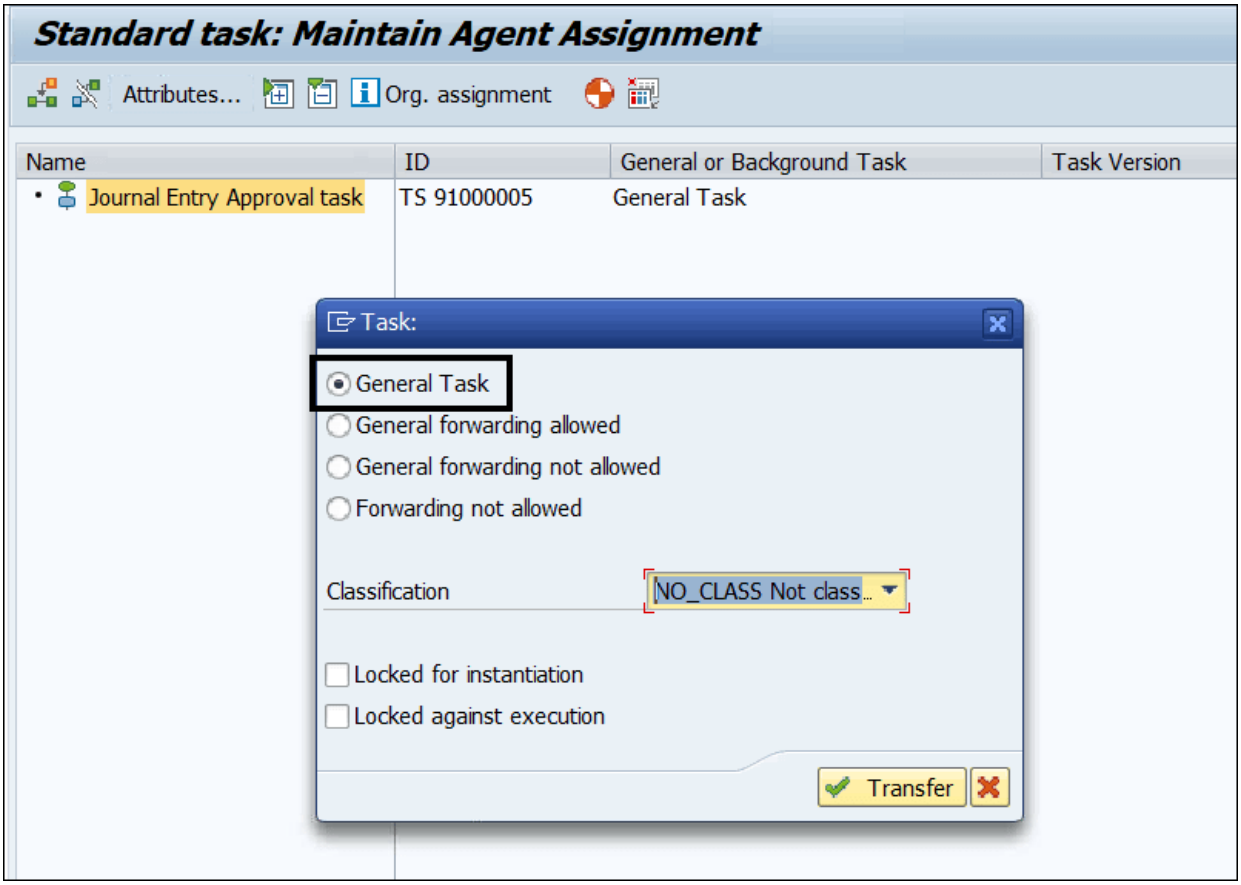


Figure 6.19 Maintain Agent Assignment Attributes at the Task Level

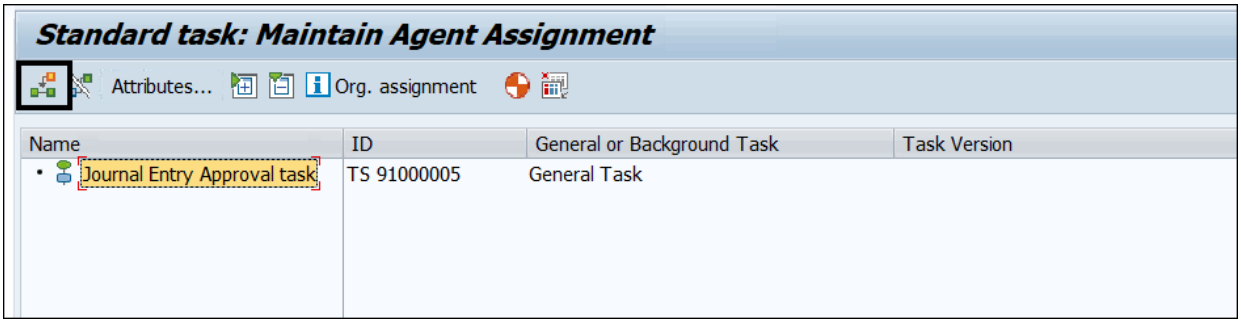


Figure 6.20 Create Agent Assignment

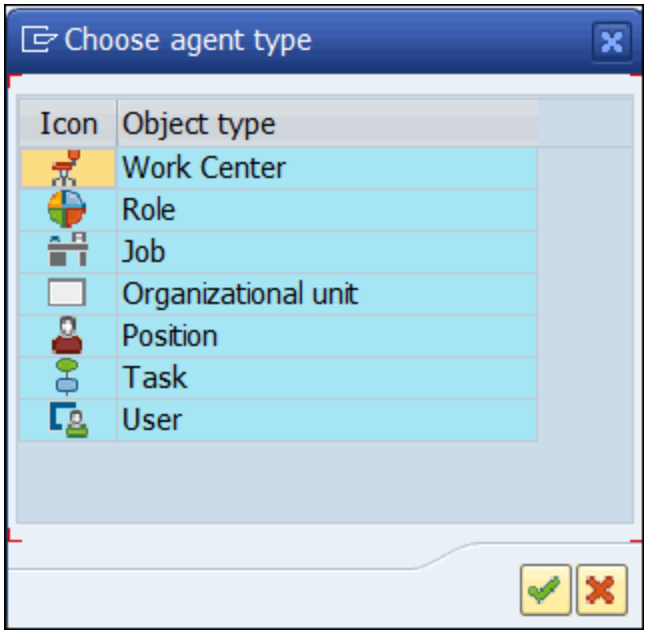


Figure 6.21 Popup to Select the Type of Agent

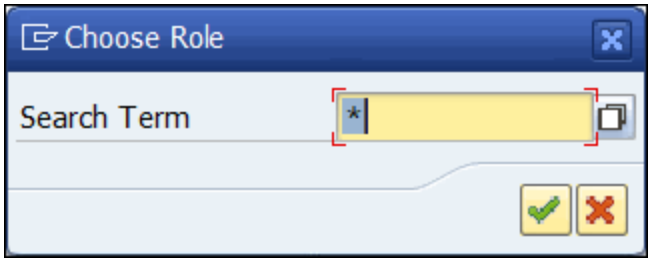


Figure 6.22 Popup to Select Role

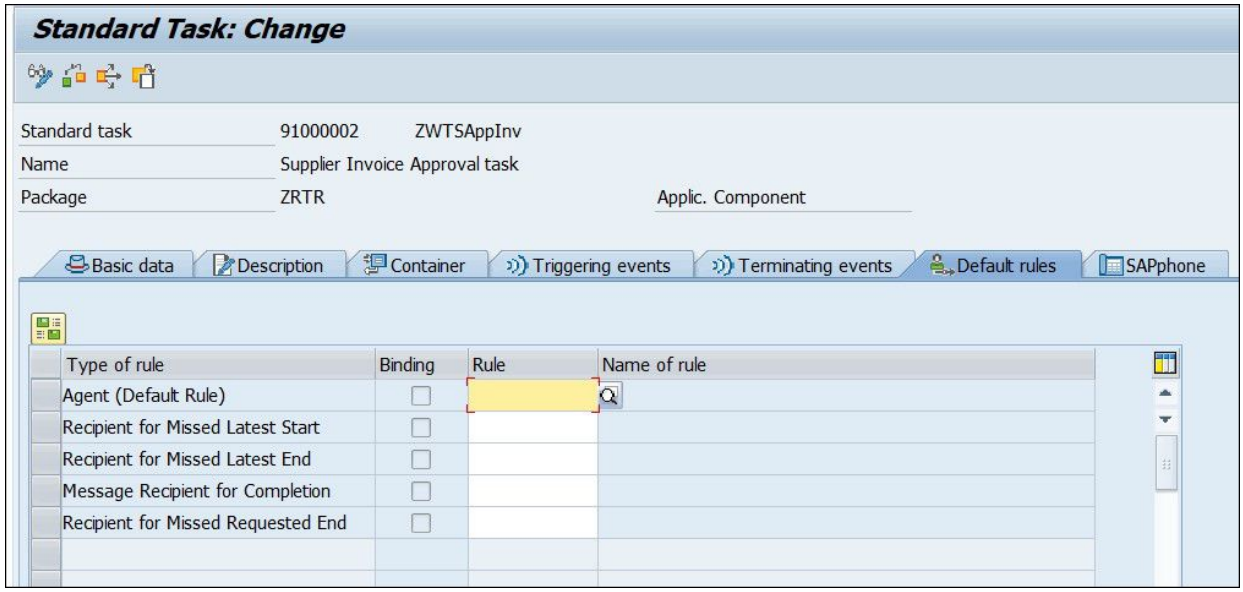


Figure 6.23 View of Default Rules under Task Definition in Transaction PFTC

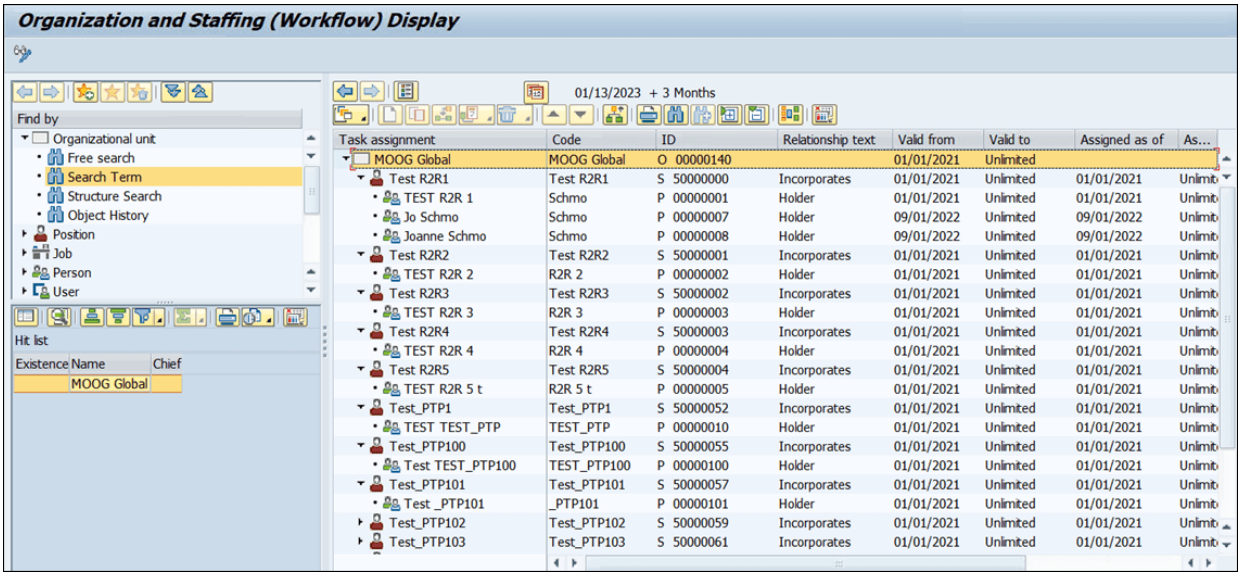


Figure 6.24 View of an Organizational Structure in SAP S/4HANA with Organizational Units, Positions, Persons, and Users

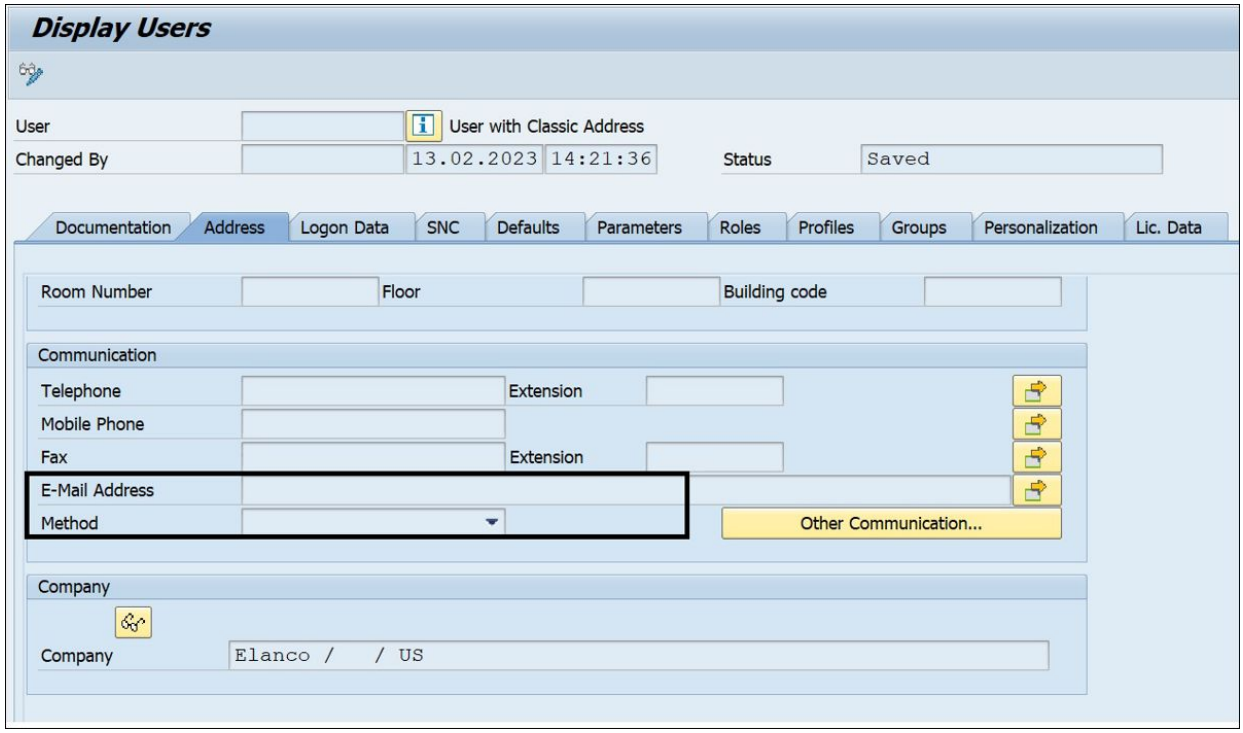


Figure 7.1 Email Address to Be Maintained in the User Master

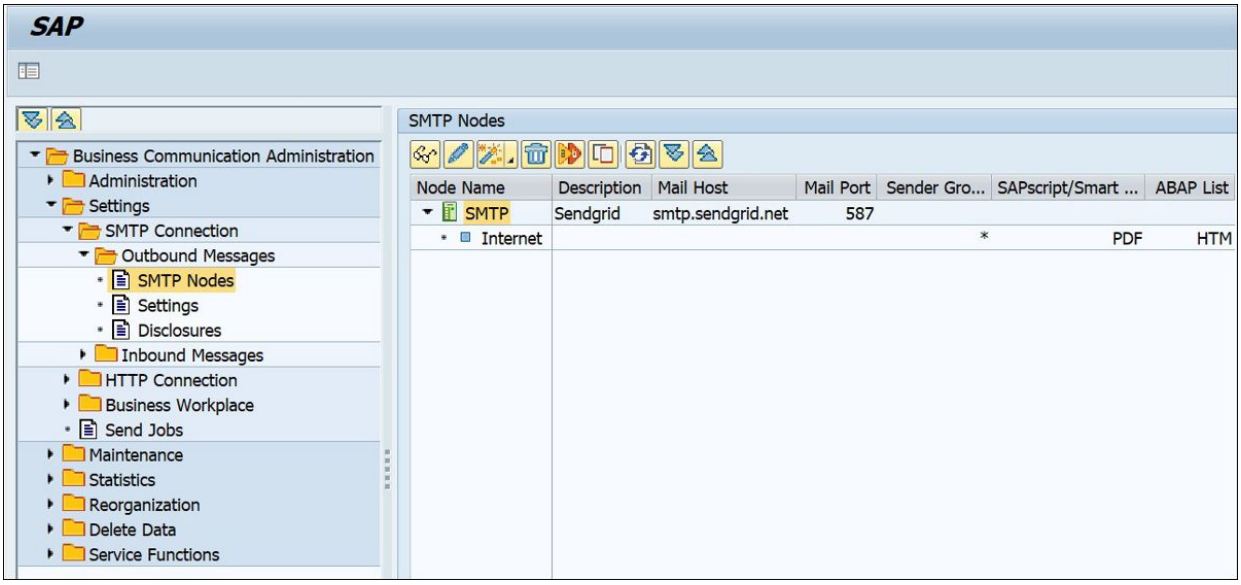


Figure 7.2 Selecting SMTP Node

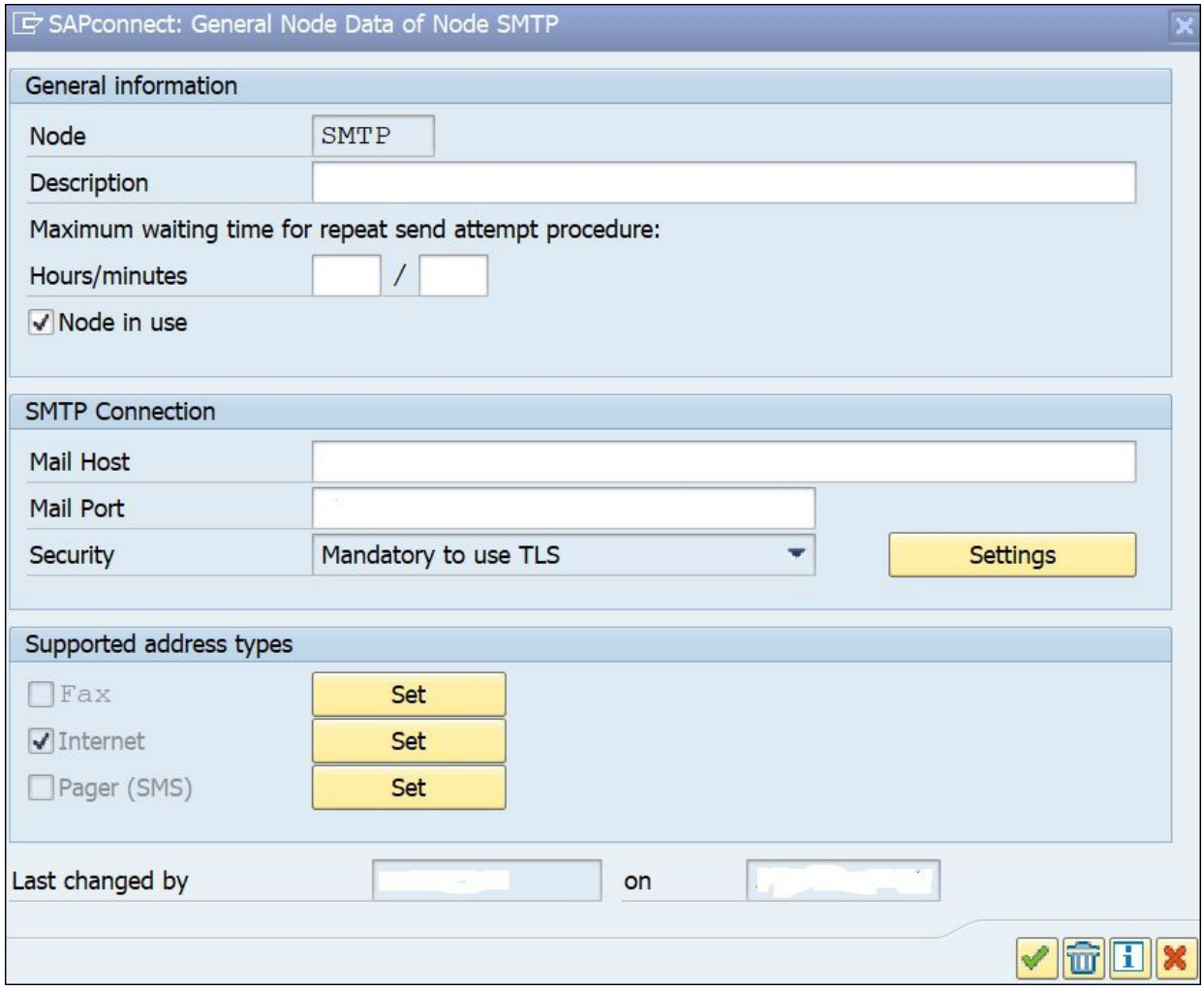


Figure 7.3 Setting Up SMTP Node

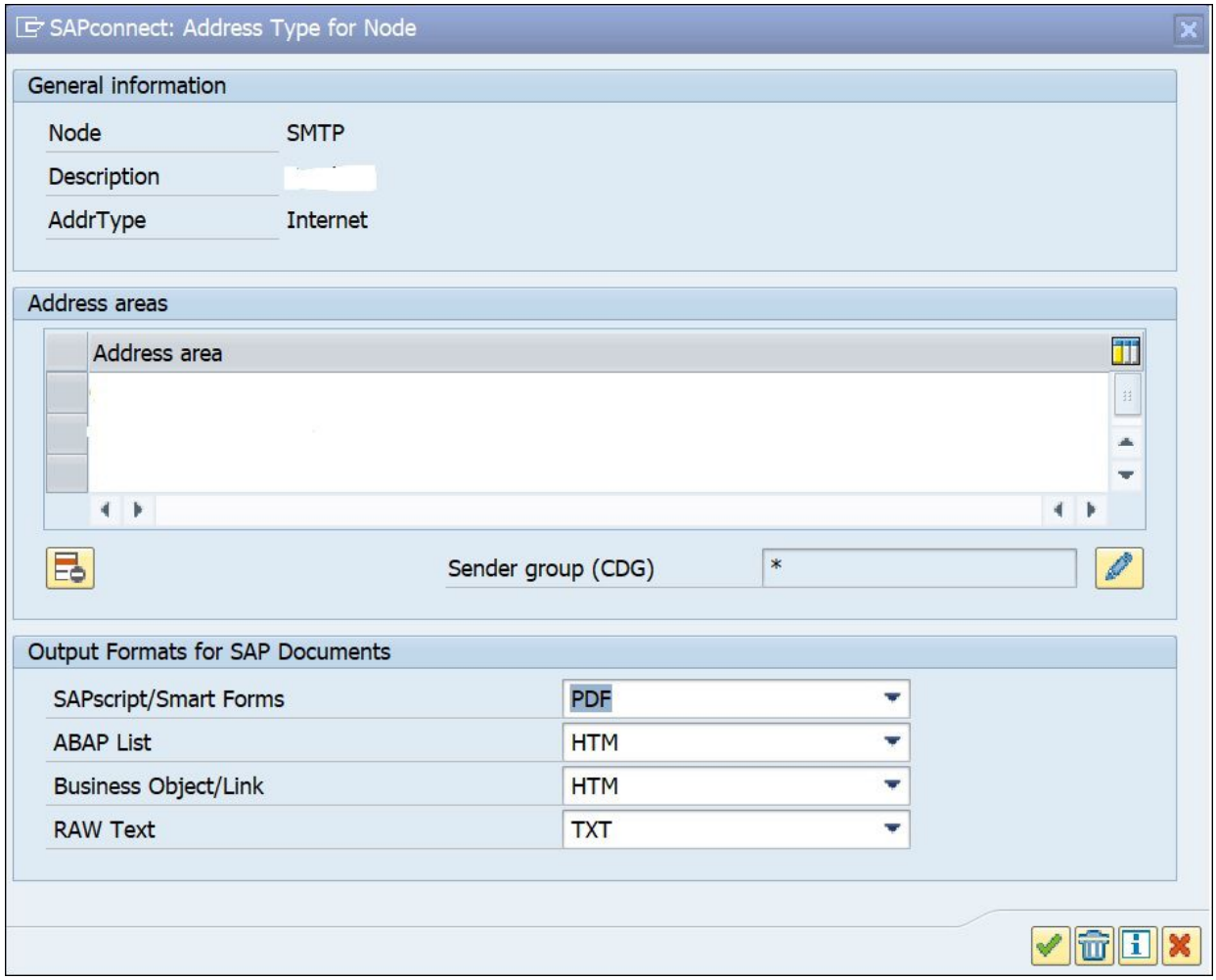


Figure 7.4 Transaction SCOT Configurations: Further Settings

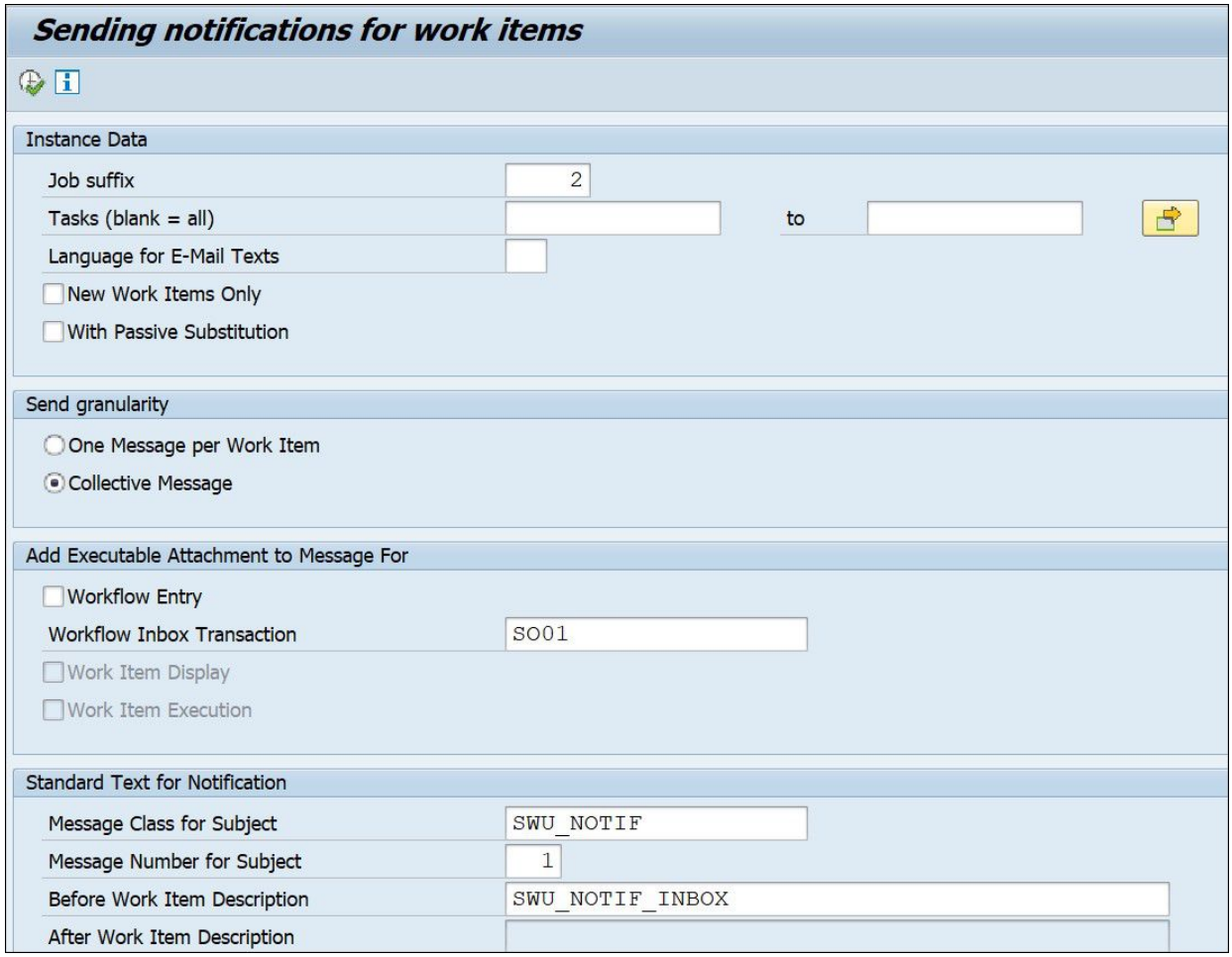


Figure 7.5 Program RSWUWFML2 to Send Notifications for Work Items

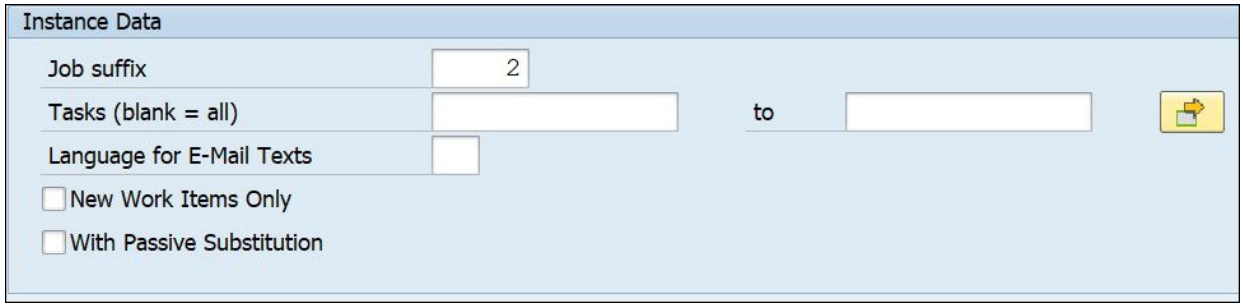


Figure 7.6 Instance Data Section of the Selection Screen

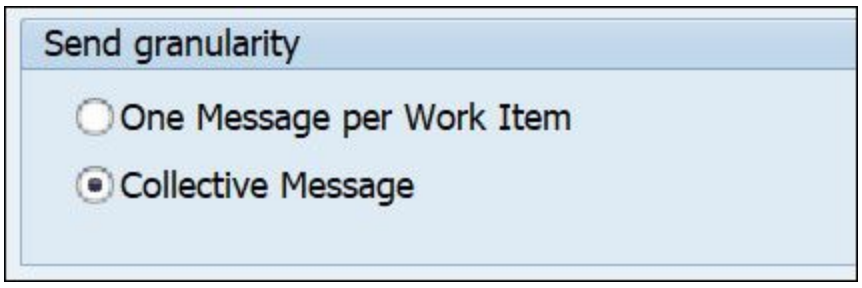


Figure 7.7 Send Granularity Section of the Selection Screen

Figure 7.8 Add Executable Attachments to Message Section of the Selection Screen

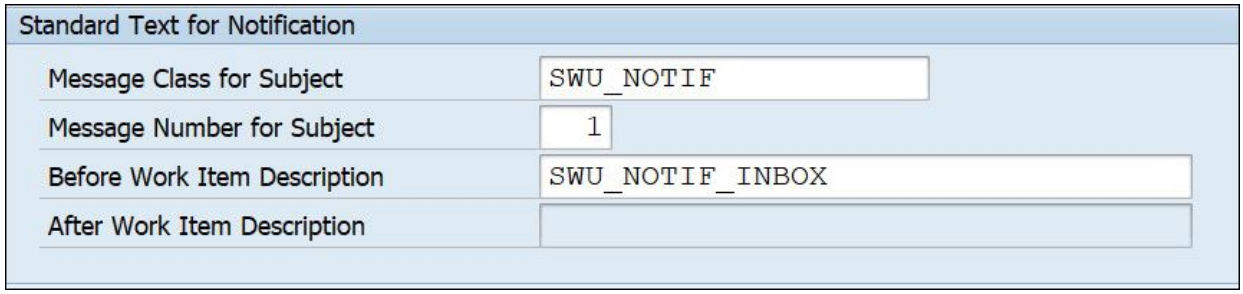


Figure 7.9 Standard Text for Notification Section of the Selection Screen

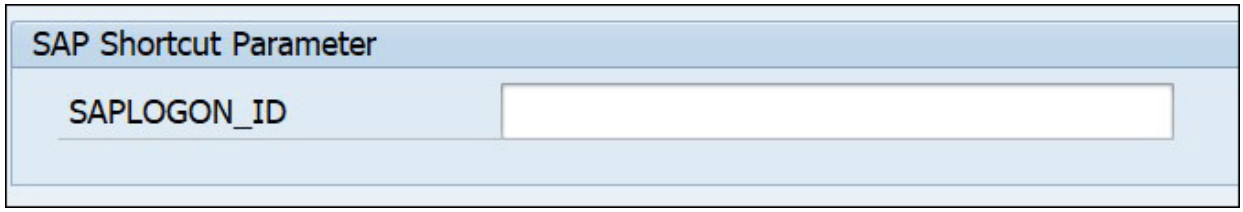


Figure 7.10 Section SAP Shortcut Parameter of the Selection Screen

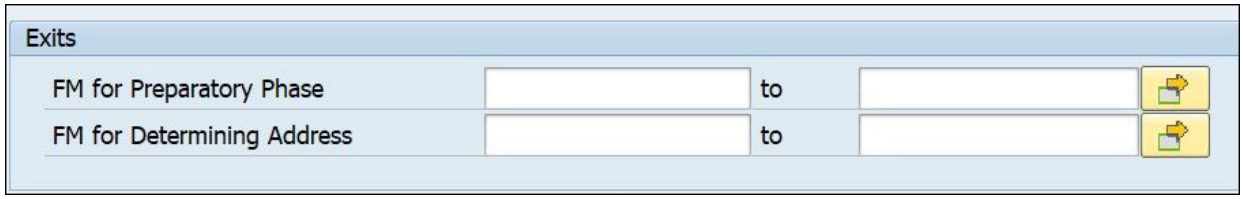


Figure 7.11 Section Exits of the Selection Screen

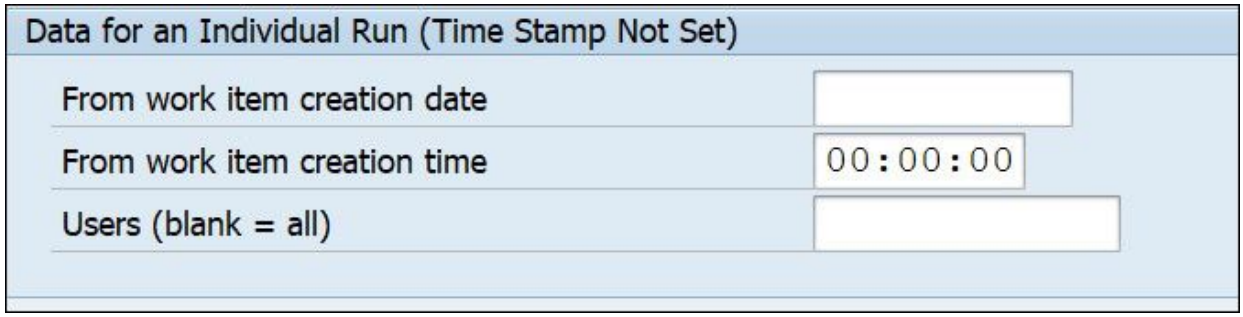


Figure 7.12 Data for an Individual Run (Time Stamp Not Set) Section in the Selection Screen

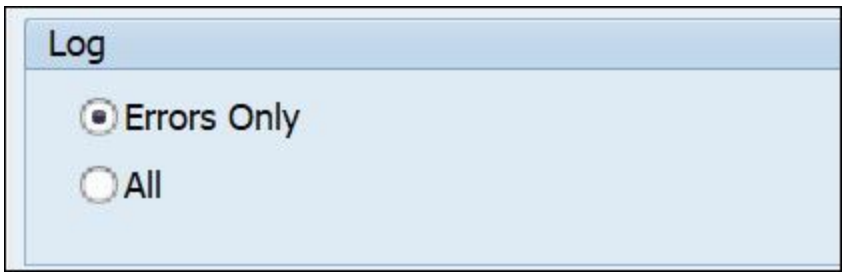


Figure 7.13 Log Section of the Selection Screen

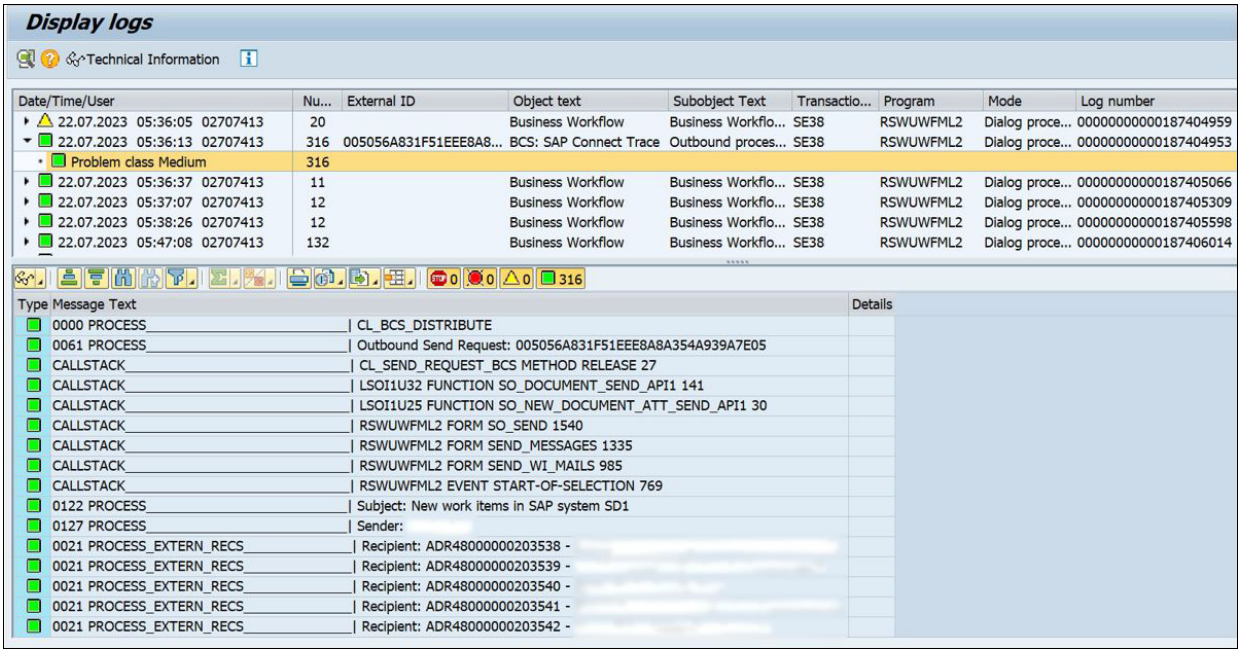


Figure 7.14 Sample Application Log Output of Program RSWUWFML2

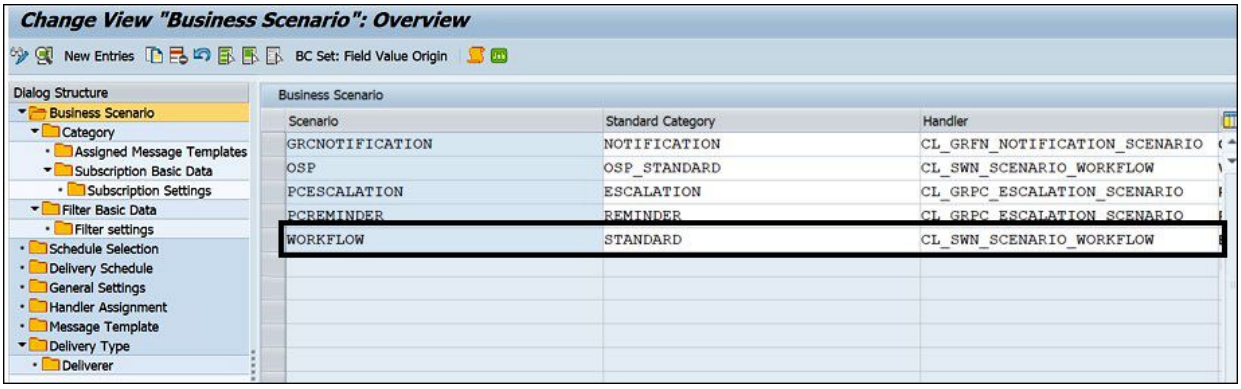


Figure 7.15 Business Scenario Workflow in Transaction SWNCONFIG

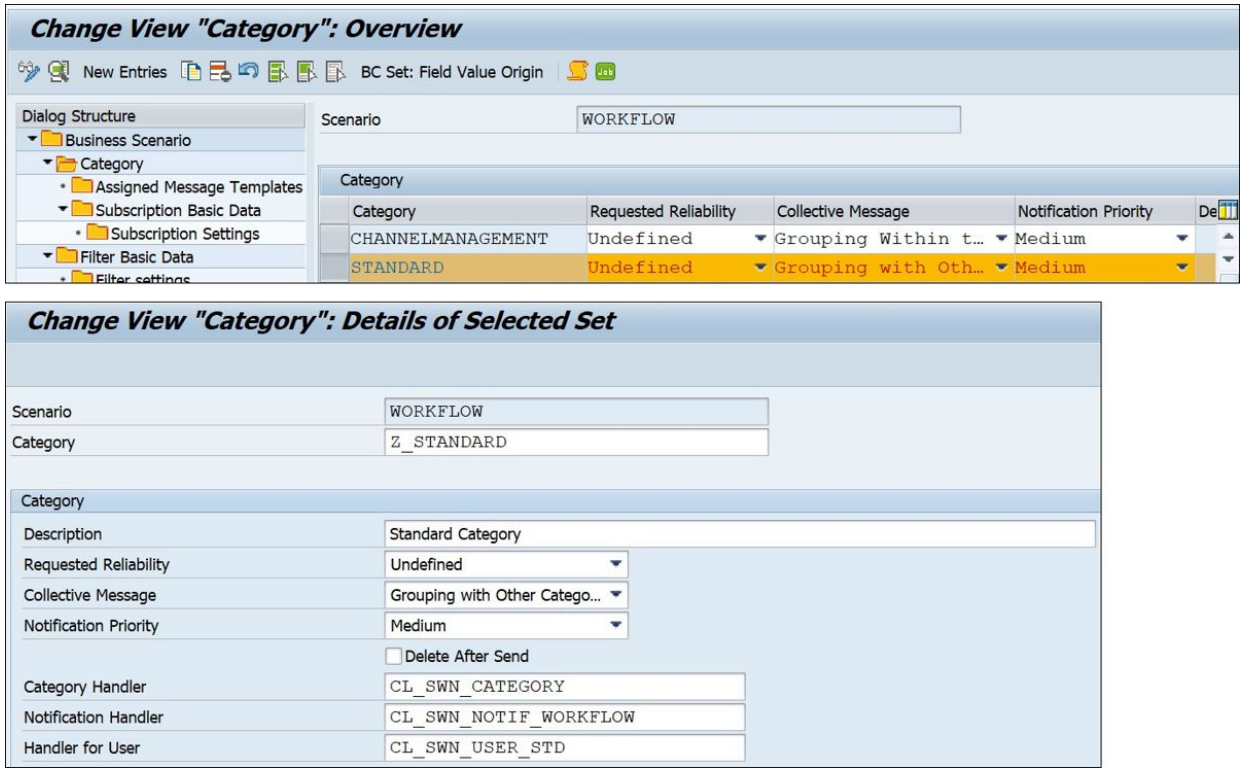


Figure 7.16 Creating a New Category by Copying the STANDARD Category

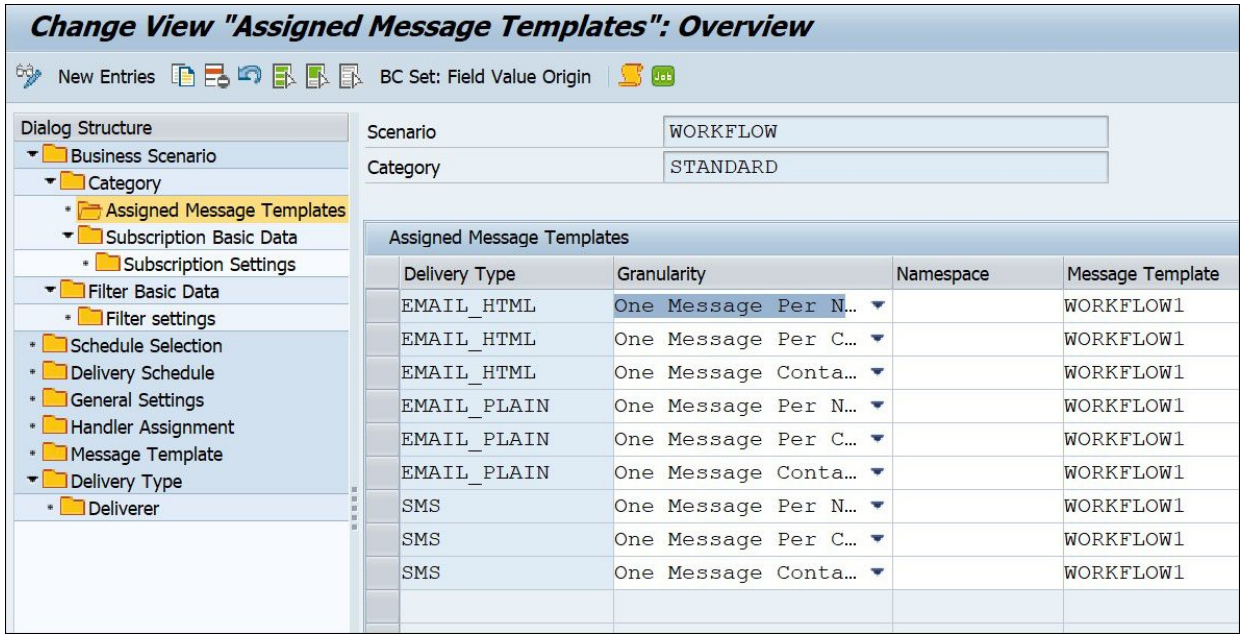


Figure 7.17 Message Templates under Category Also to Be Copied

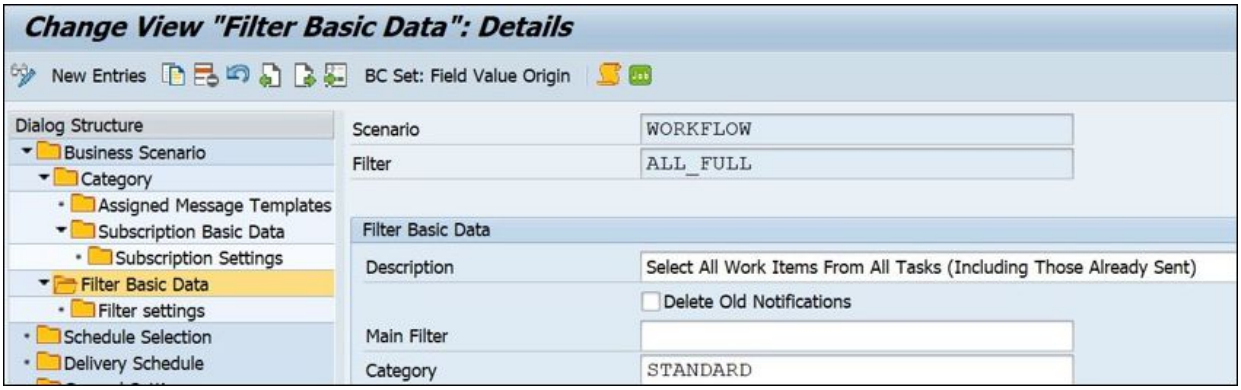


Figure 7.18 Full Filter

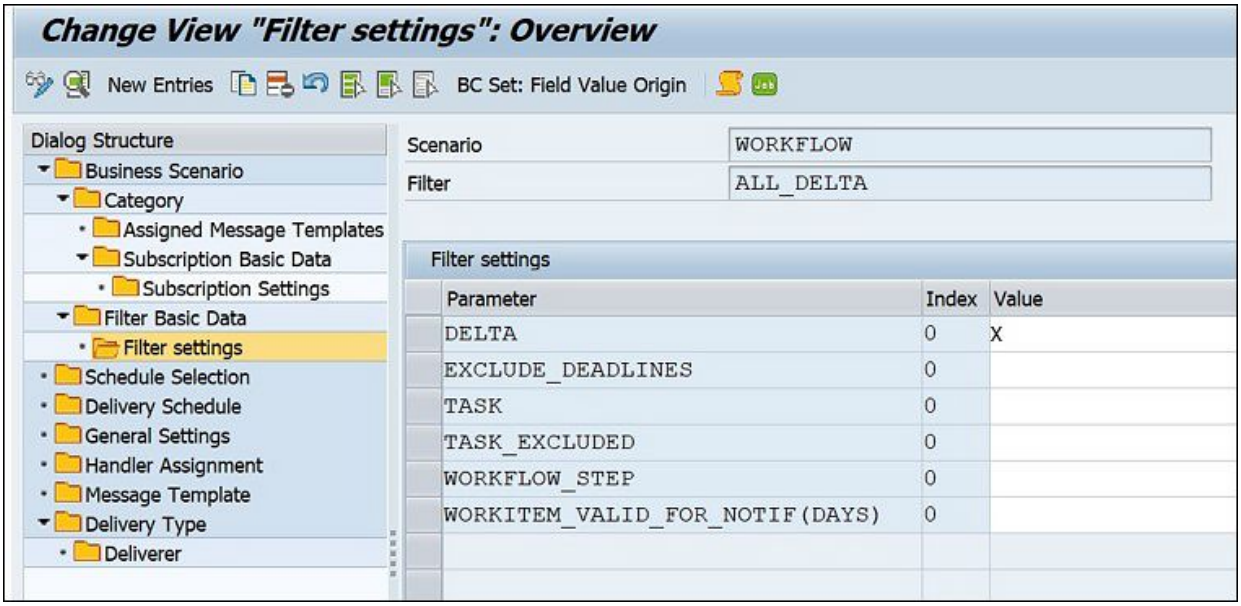


Figure 7.19 Filter Settings

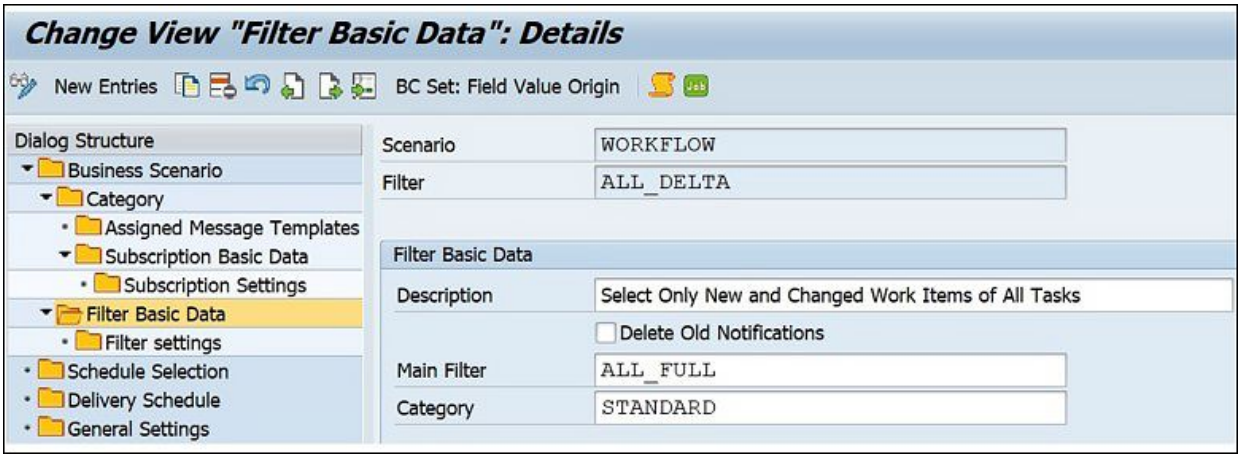


Figure 7.20 Delta Filter

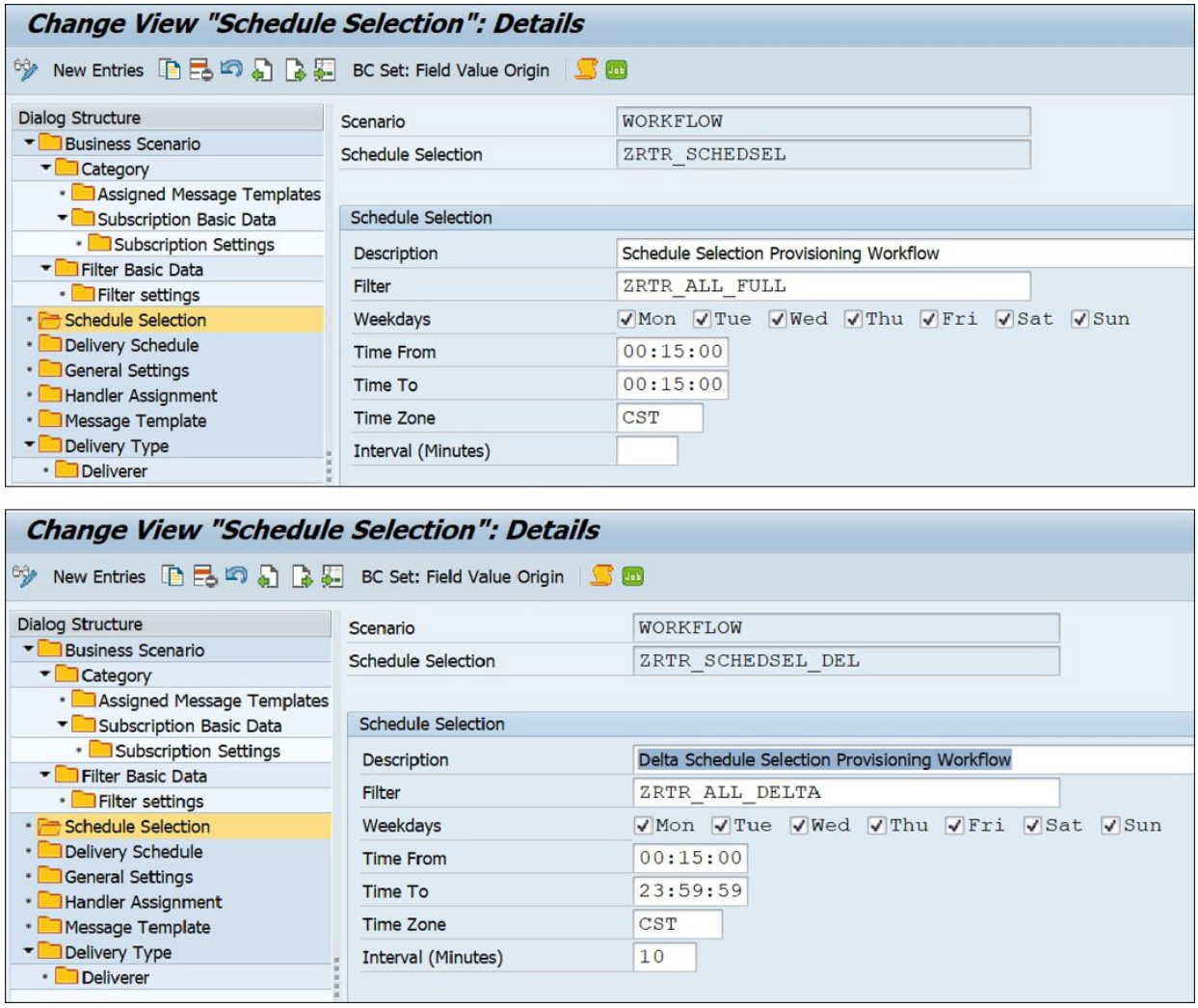


Figure 7.21 Schedule Selection for Full and Delta Filters

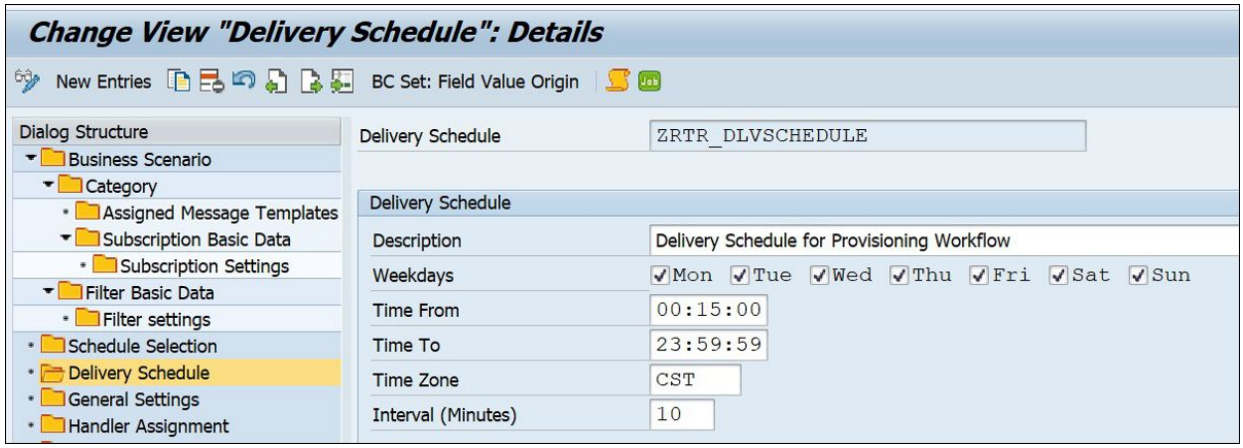
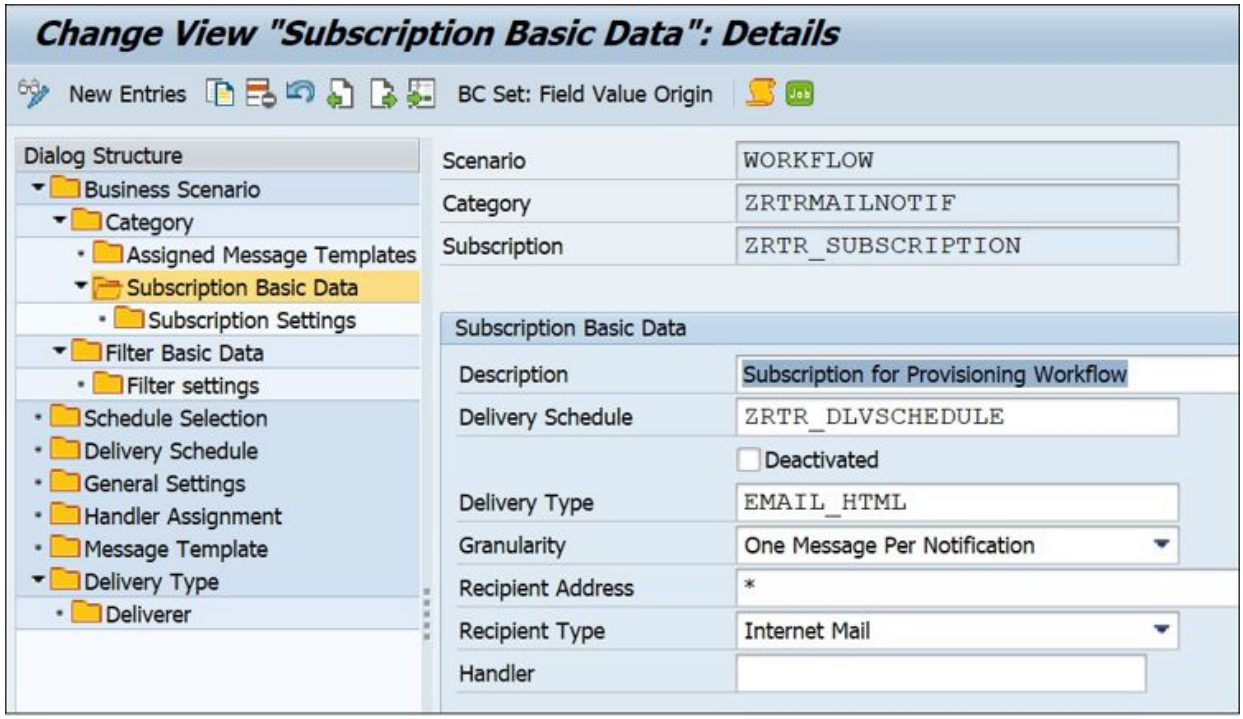


Figure 7.22 Delivery Schedule


Change View "Subscription Basic Data": Details
New Entries
BC Set: Field Value Origin
Dialog Structure
Business Scenario
Category
Assigned Message Templates
Subscription Basic Data
Subscription Settings
Filter Basic Data
Filter settings
Schedule Selection
Delivery Schedule
General Settings
Handler Assignment
Message Template
Delivery Type
Deliverer
Scenario
WORKFLOW
Category
ZRTRMAILNOTIF
Subscription
ZRTR_SUBSCRIPTION
Subscription Basic Data
Description
Subscription for Provisioning Workflow
Delivery Schedule
ZRTR_DLVSCHEDULE
Deactivated
Delivery Type
EMAIL_HTML
Granularity
One Message Per Notification
Recipient Address
*
Recipient Type
Internet Mail
Handler

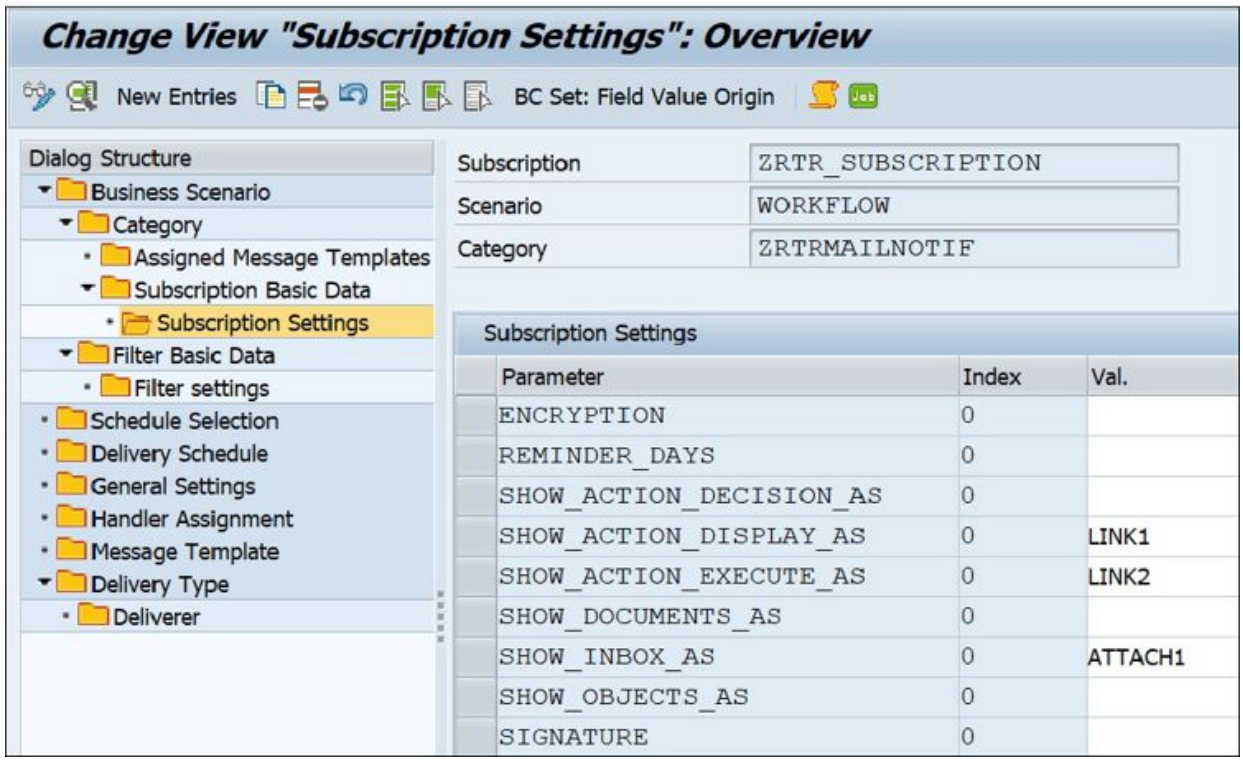

Change View "Subscription Settings": Overview
New Entries
BC Set: Field Value Origin
Dialog Structure
Business Scenario
Category
Assigned Message Templates
Subscription Basic Data
Subscription Settings
Filter Basic Data
Filter settings
Schedule Selection
Delivery Schedule
General Settings
Handler Assignment
Message Template
Delivery Type
Deliverer
Subscription
ZRTR_SUBSCRIPTION
Scenario
WORKFLOW
Category
ZRTRMAILNOTIF
Subscription Settings
Parameter
Index
Val.
ENCRYPTION
0
REMINDER_DAYS
0
SHOW_ACTION_DECISION_AS
0
SHOW_ACTION_DISPLAY_AS
0
LINK1
SHOW_ACTION_EXECUTE_AS
0
LINK2
SHOW_DOCUMENTS_AS
0
SHOW_INBOX_AS
0
ATTACH1
SHOW_OBJECTS_AS
0
SIGNATURE
0

Figure 7.23 Subscription and Subscription Settings

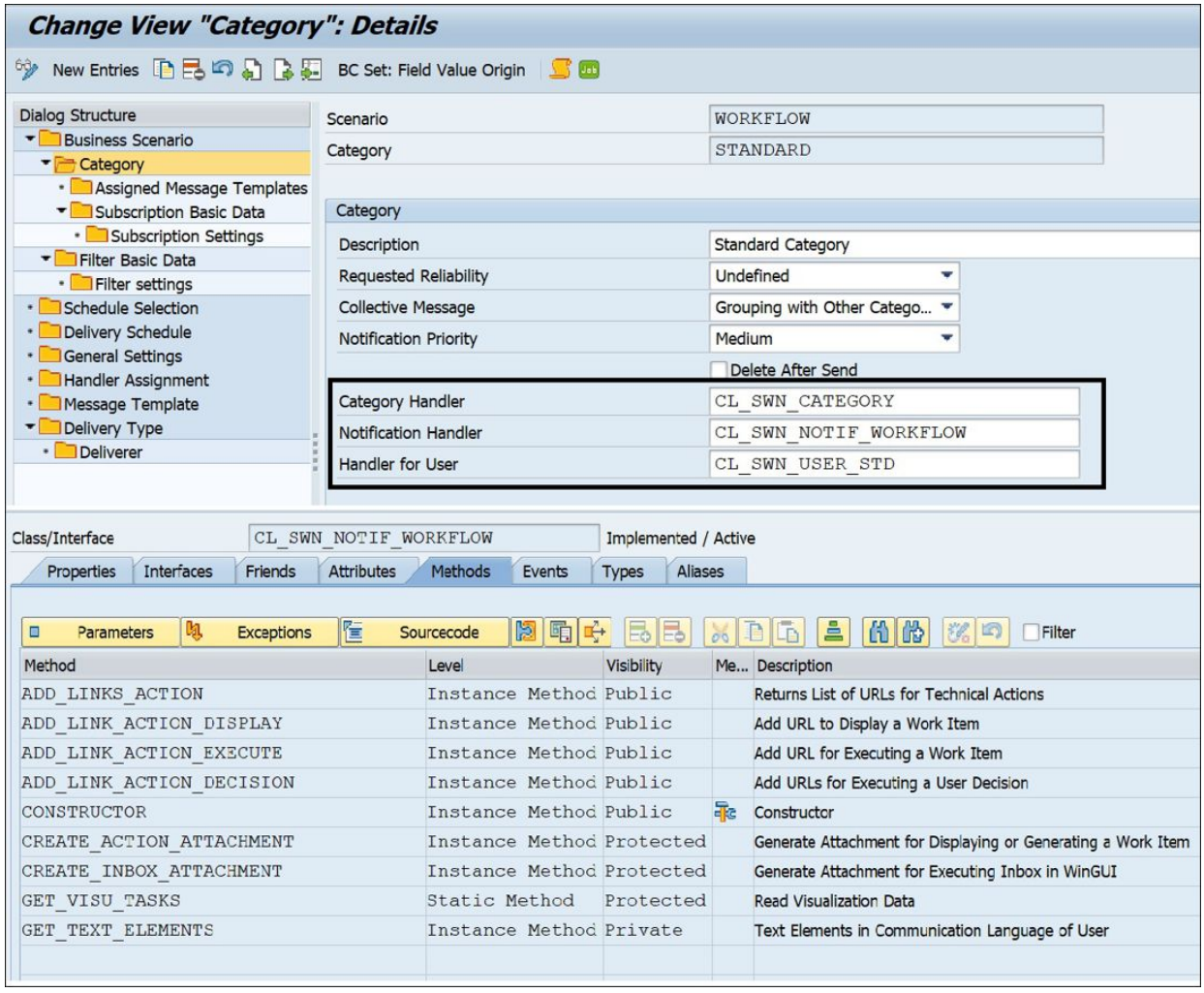


Figure 7.24 Handler Classes and the Methods of Class CL_SWN_NOTIF_WORKFLOW

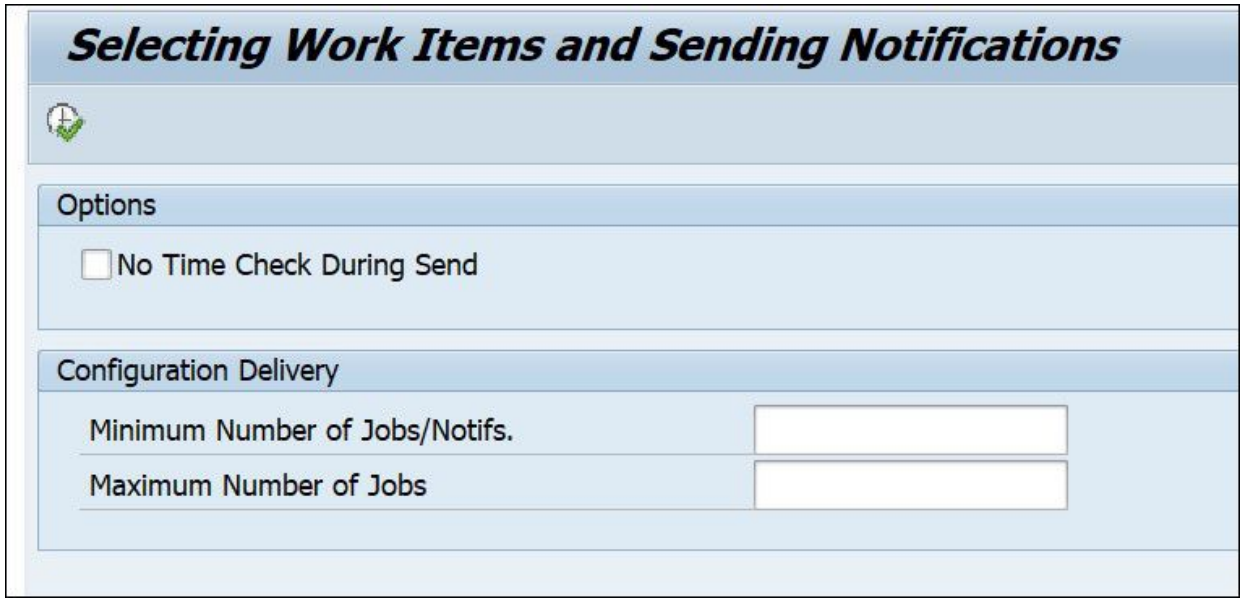


Figure 7.25 Selection Screen of Program SWN_SELSEN

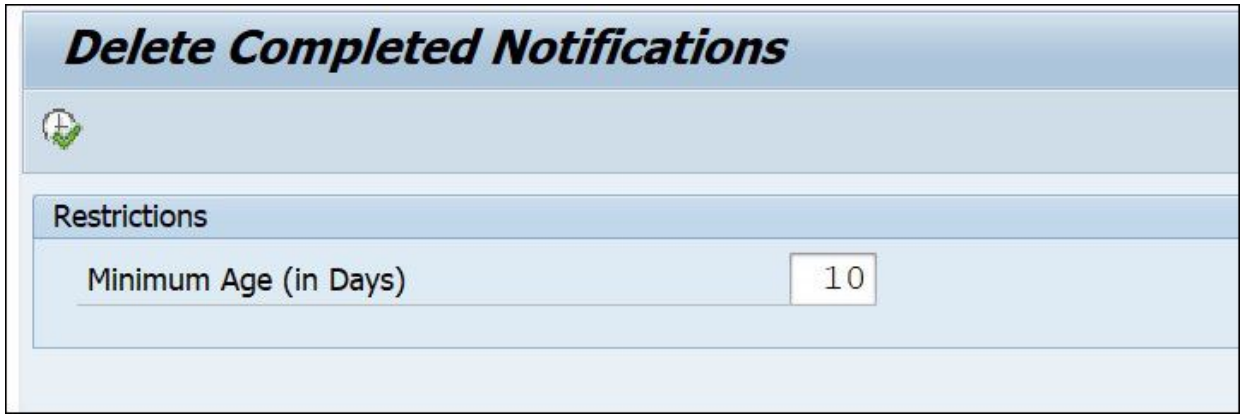


Figure 7.26 Selection Screen of Program RSWNNOTIFDEL

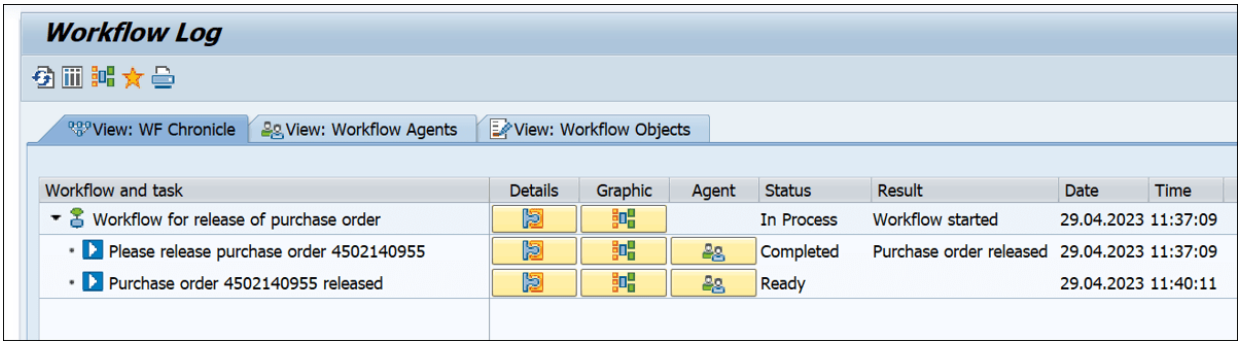


Figure 8.1 User View of a Workflow Log: WF Chronicle

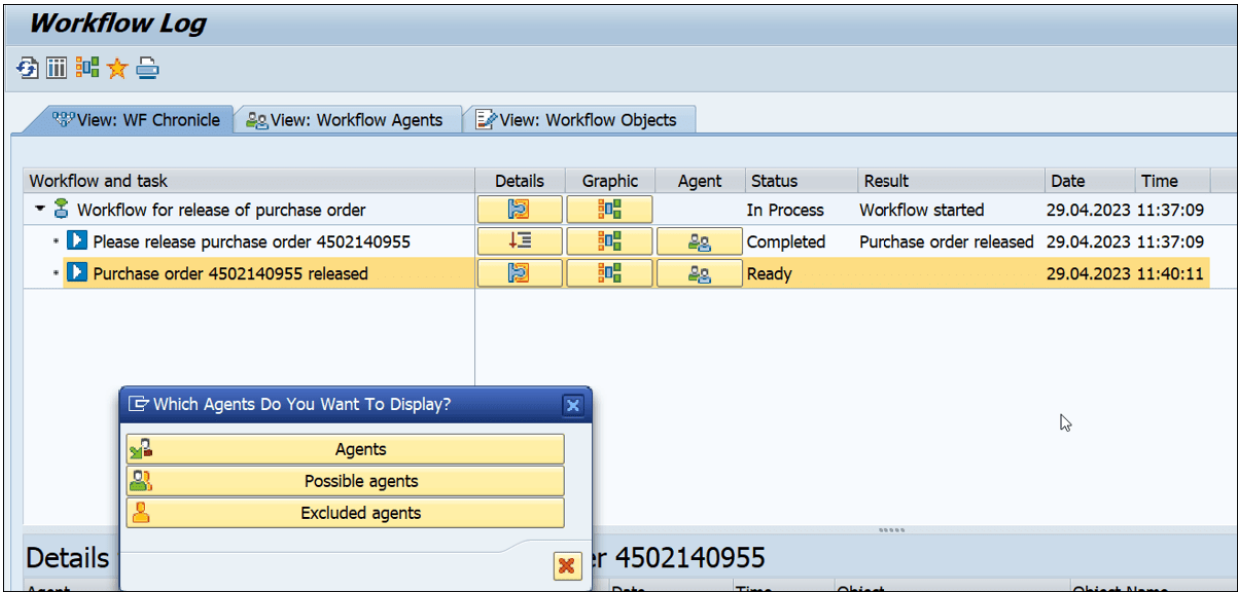


Figure 8.2 Agent View in the Workflow Log

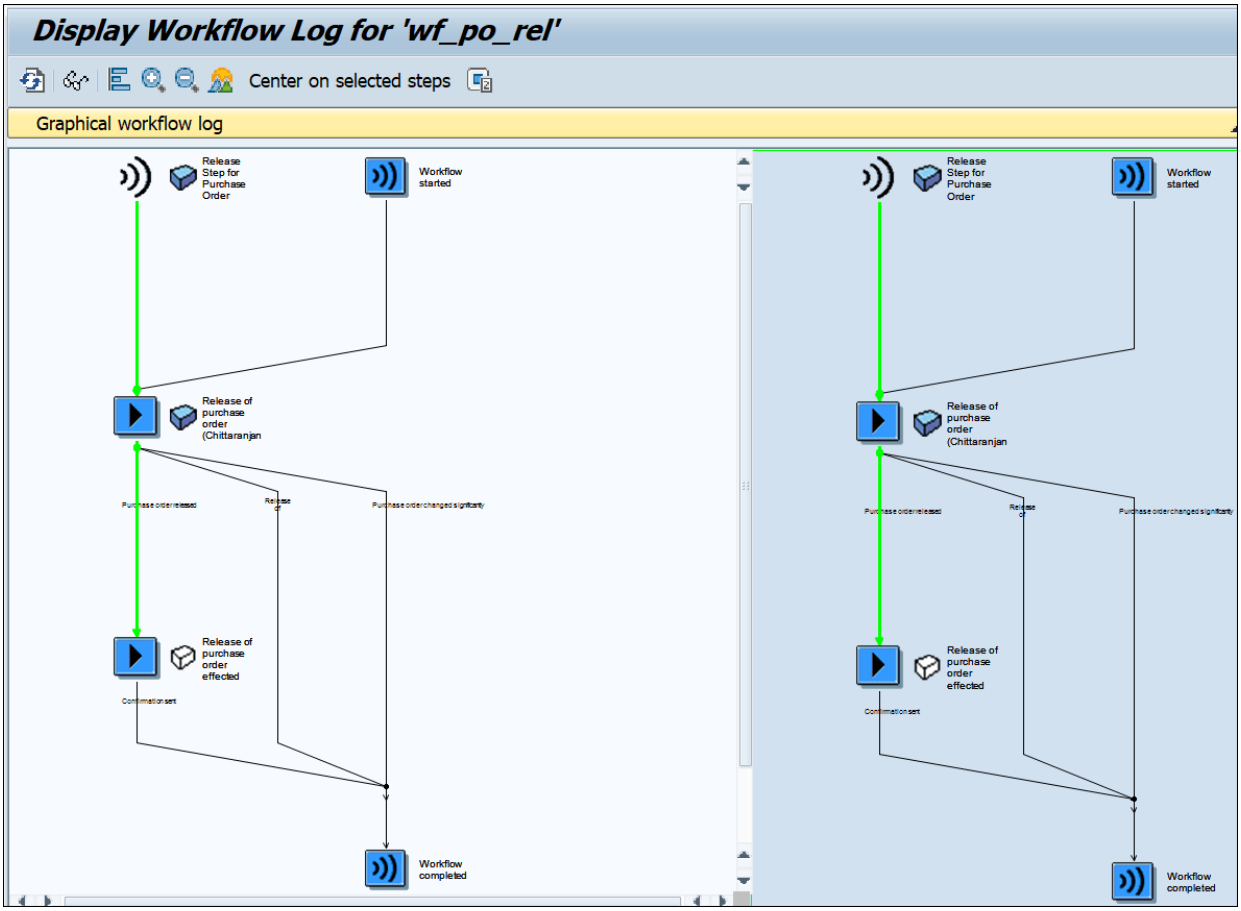


Figure 8.3 Workflow Log in Graphical View

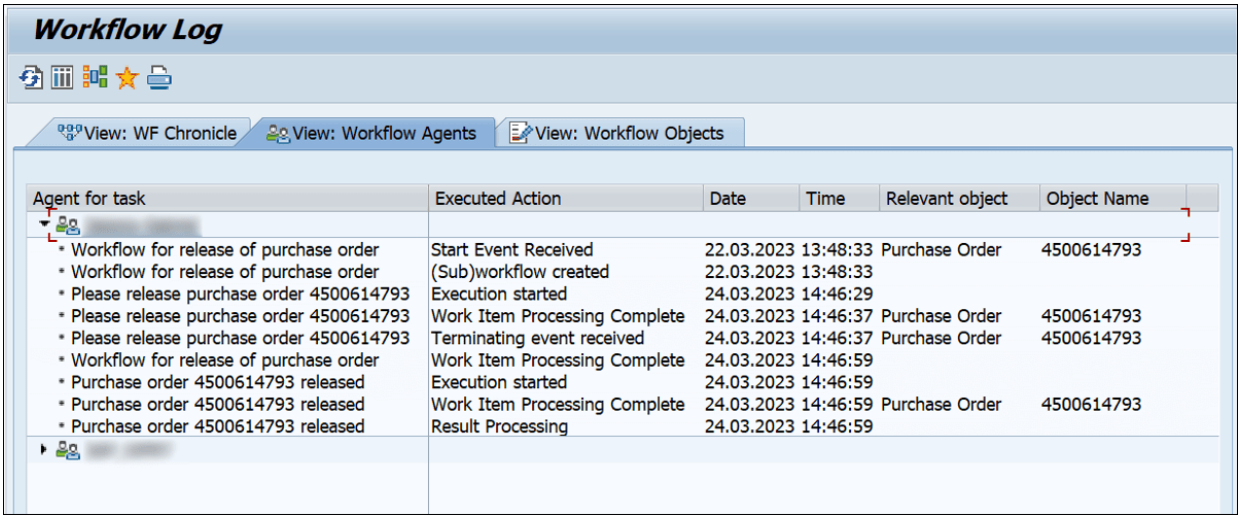


Figure 8.4 Workflow Agent View

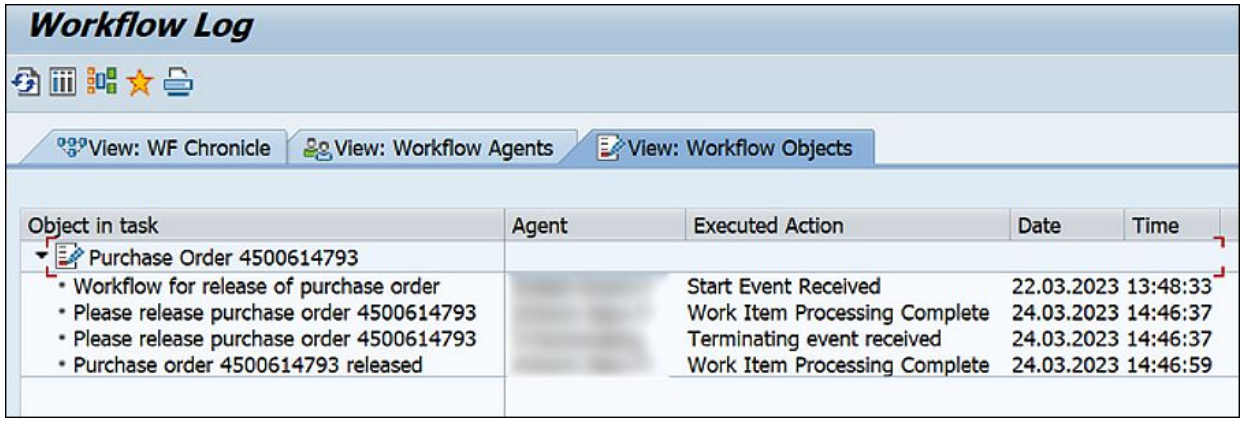


Figure 8.5 Workflow Object View in the Workflow Log

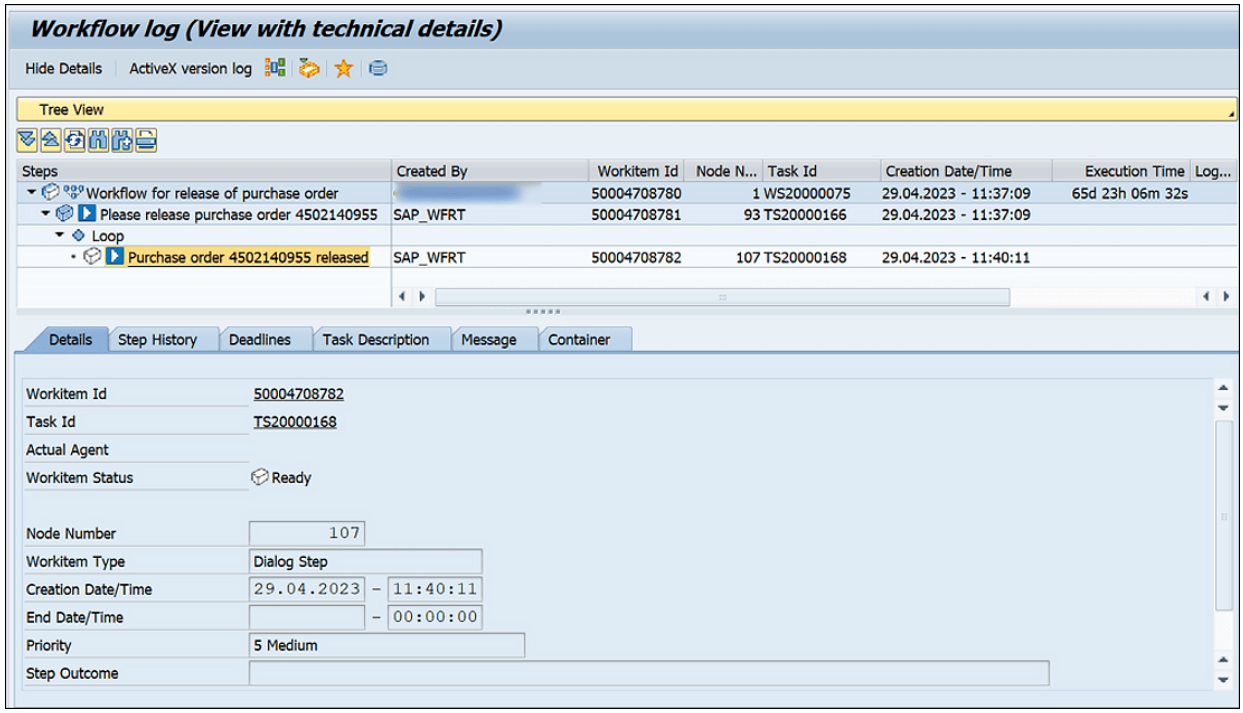


Figure 8.6 Technical Workflow Log View

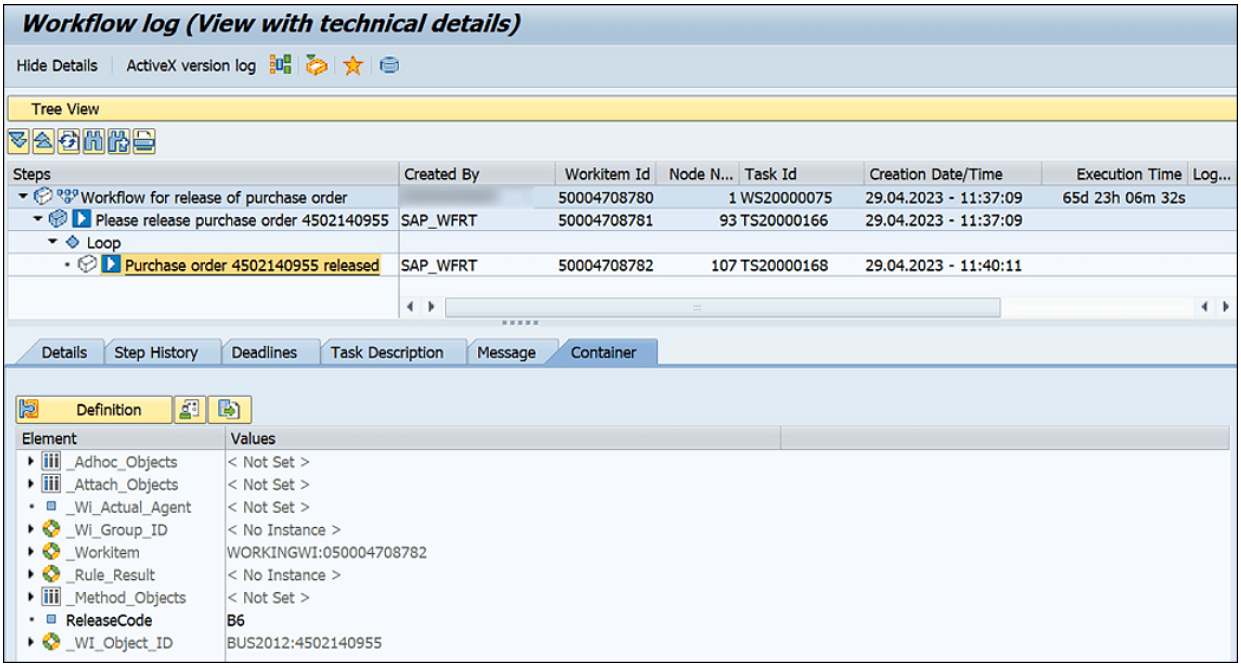


Figure 8.7 Technical Workflow Log: Container View

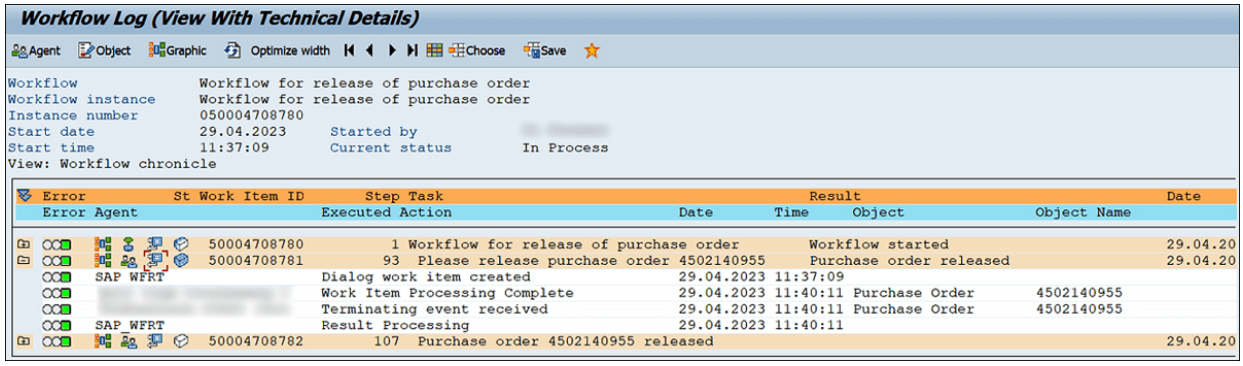
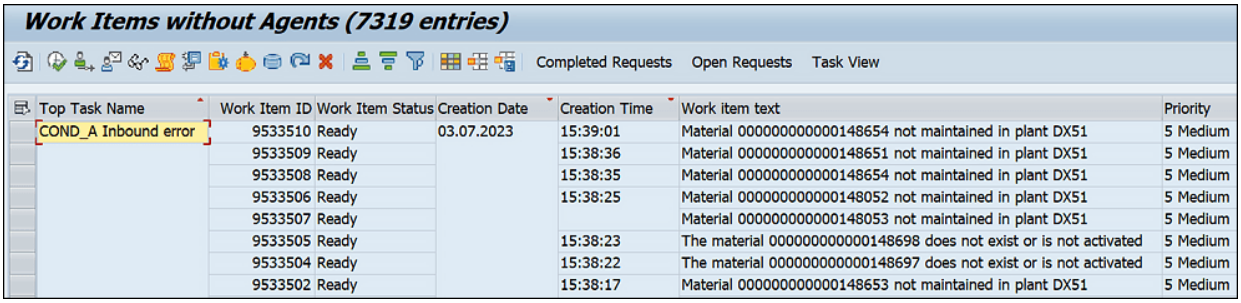


Figure 8.8 Workflow Log: Classic Technical View



Top Task Name	Work Item ID	Work Item Status	Creation Date	Creation Time	Work item text	Priority
COND_A Inbound error	9533510	Ready	03.07.2023	15:39:01	Material 000000000000148654 not maintained in plant DX51	5 Medium
	9533509	Ready		15:38:36	Material 000000000000148651 not maintained in plant DX51	5 Medium
	9533508	Ready		15:38:35	Material 000000000000148654 not maintained in plant DX51	5 Medium
	9533506	Ready		15:38:25	Material 000000000000148052 not maintained in plant DX51	5 Medium
	9533507	Ready			Material 000000000000148053 not maintained in plant DX51	5 Medium
	9533505	Ready		15:38:23	The material 000000000000148698 does not exist or is not activated	5 Medium
	9533504	Ready		15:38:22	The material 000000000000148697 does not exist or is not activated	5 Medium
	9533502	Ready		15:38:17	Material 000000000000148653 not maintained in plant DX51	5 Medium

Figure 8.9 Work Items without Agents

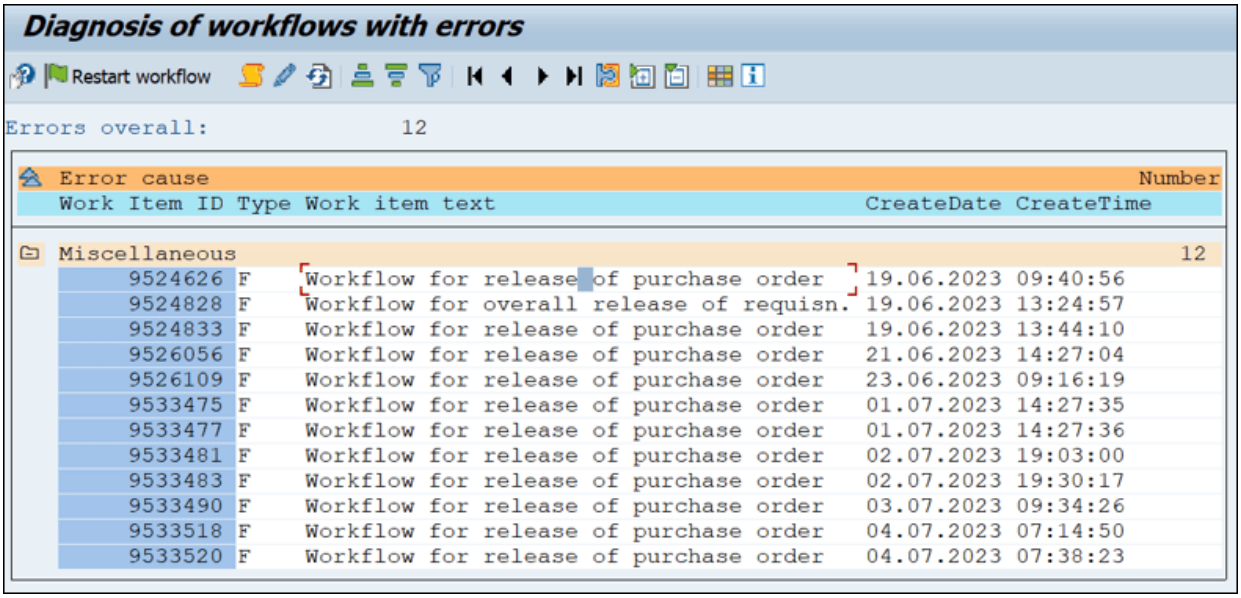


Figure 8.10 Diagnosis of Workflows with Errors

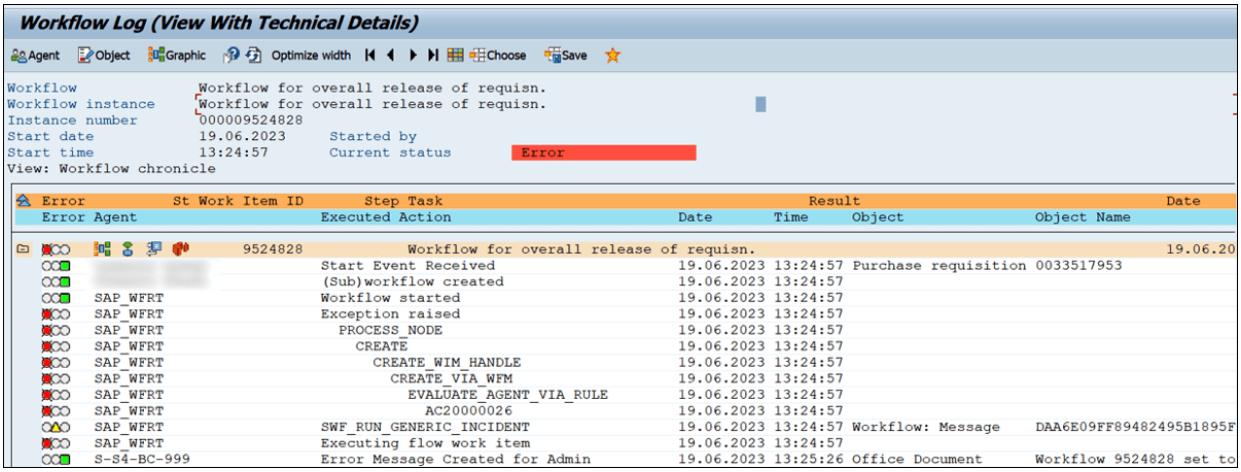


Figure 8.11 Agent Determination Error

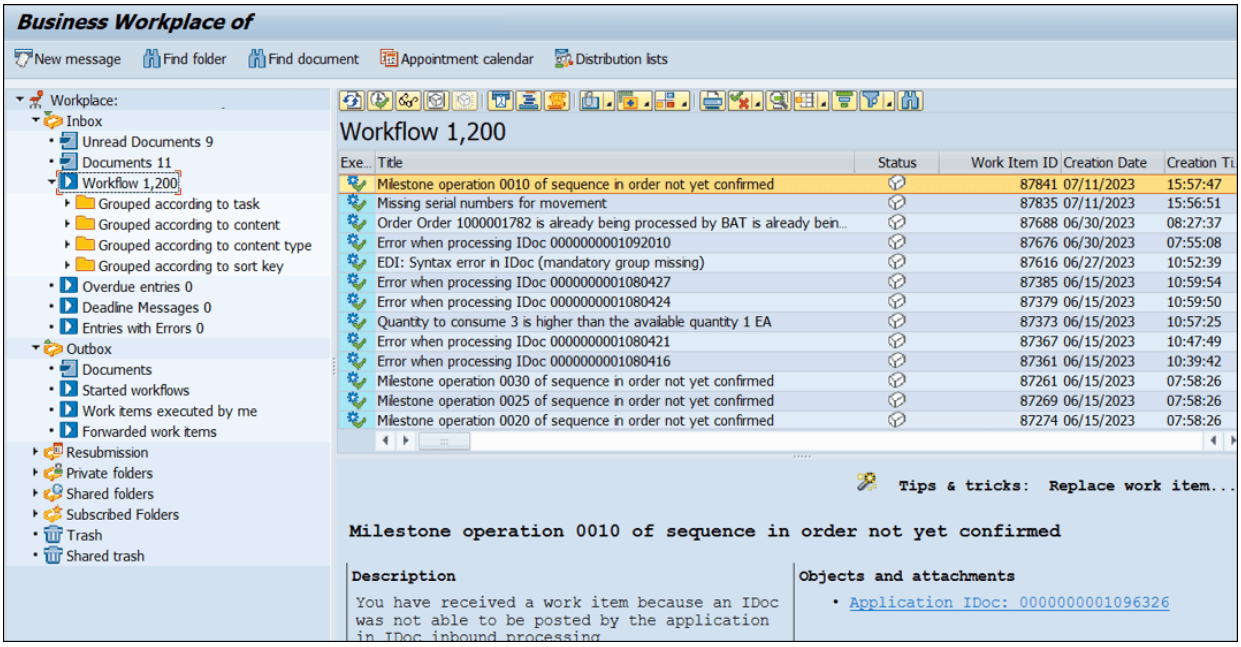


Figure 8.12 Business Workplace

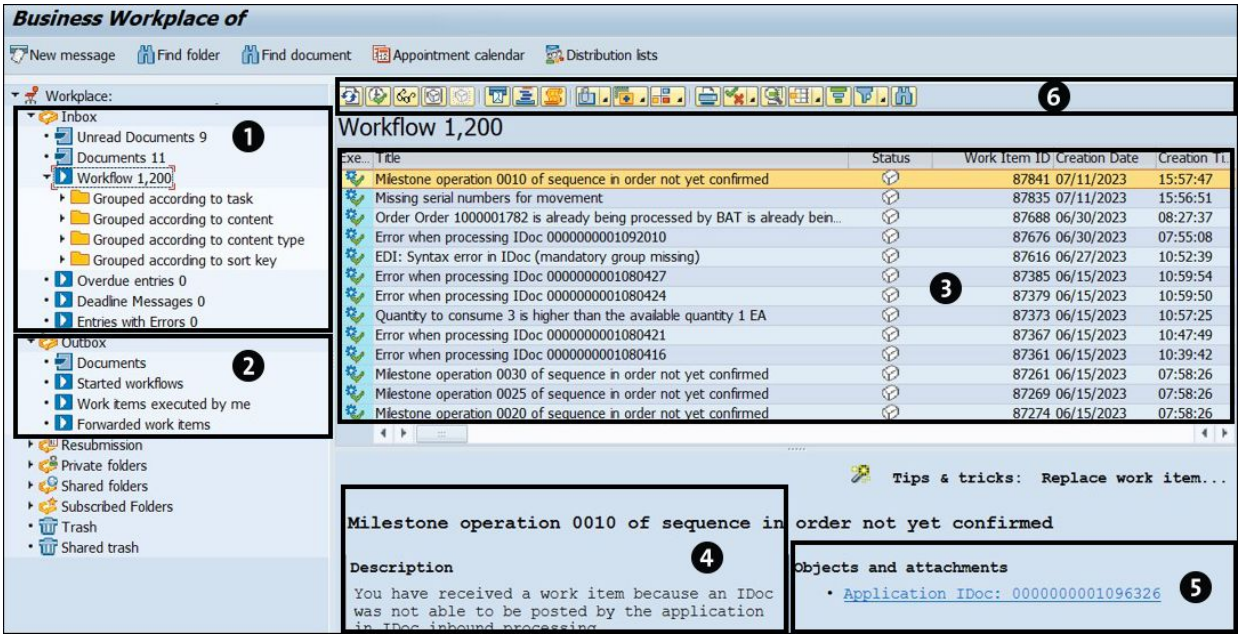


Figure 8.13 Business Workplace with Components

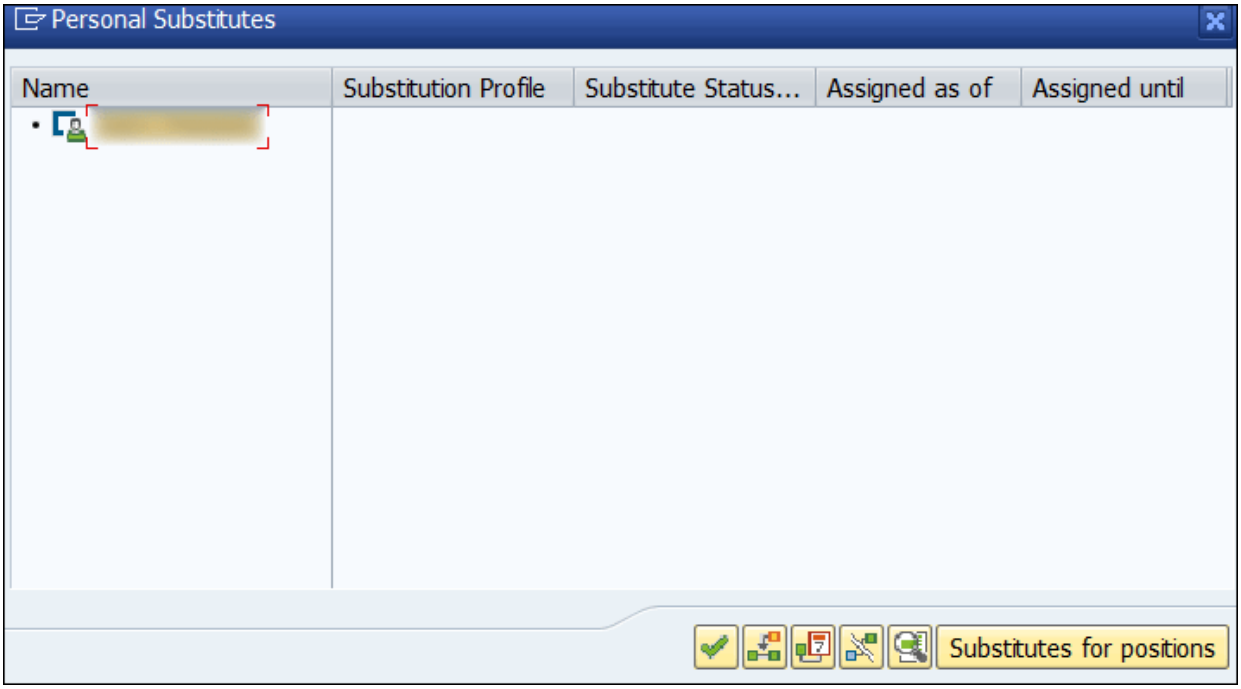


Figure 8.14 Personal Substitutes Creation Screen

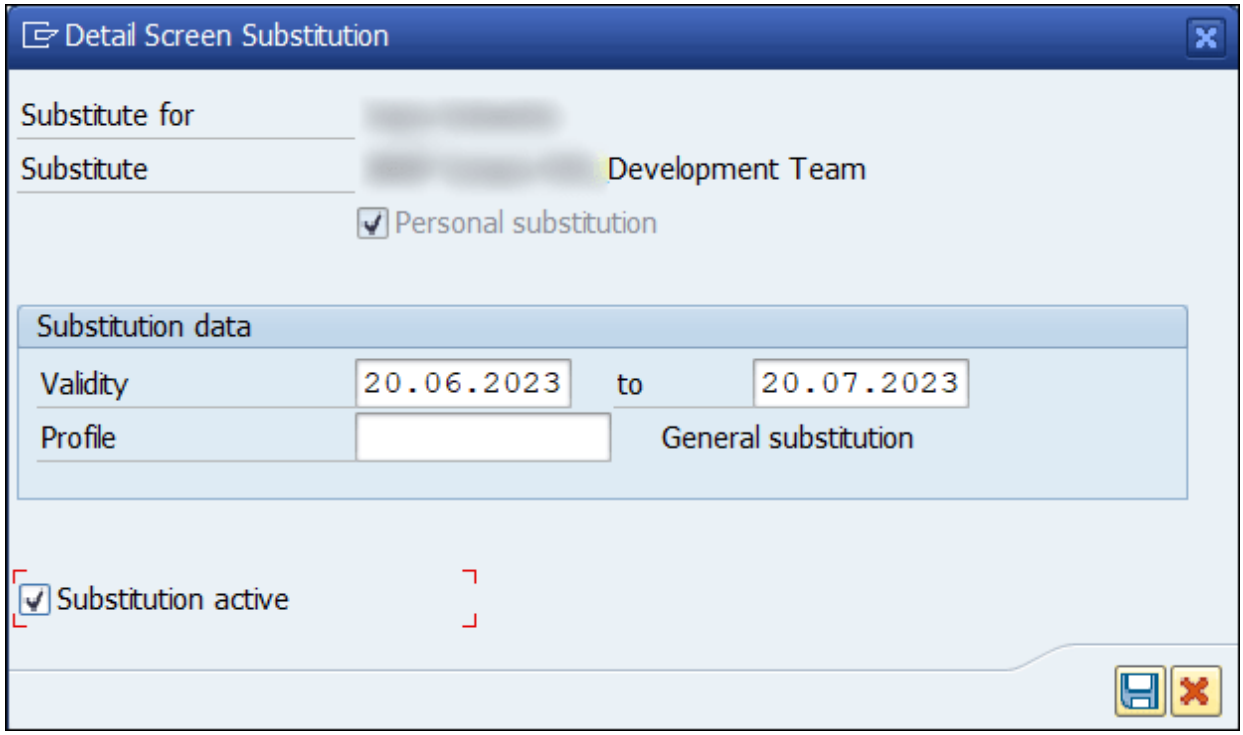


Figure 8.15 Substitute Detail Screen

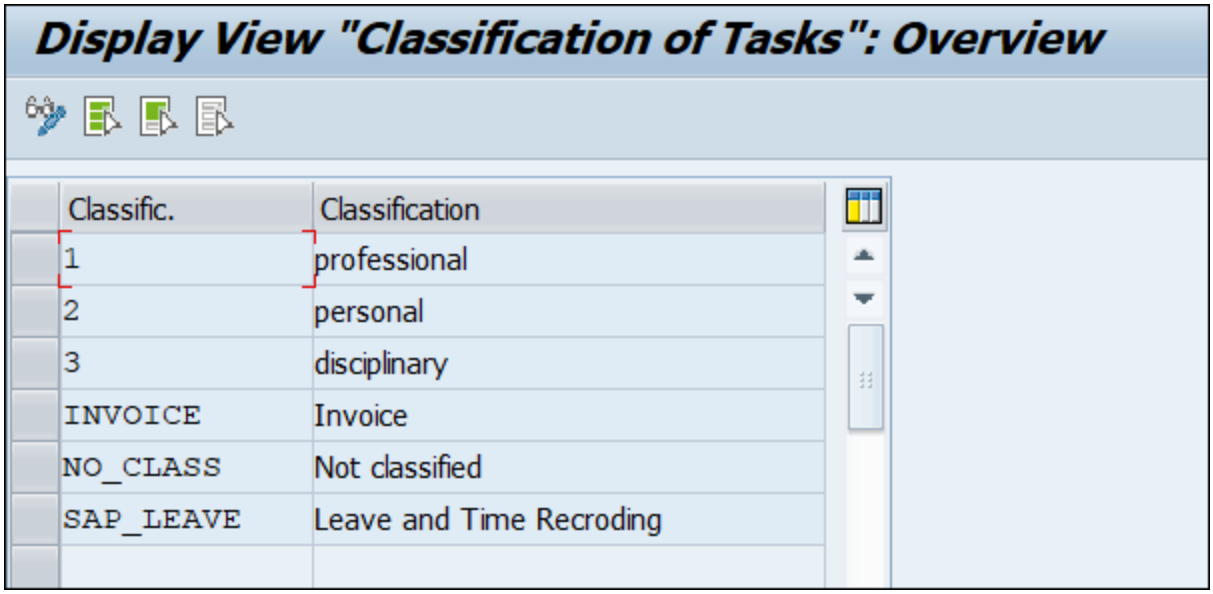


Figure 8.16 Classification of Tasks

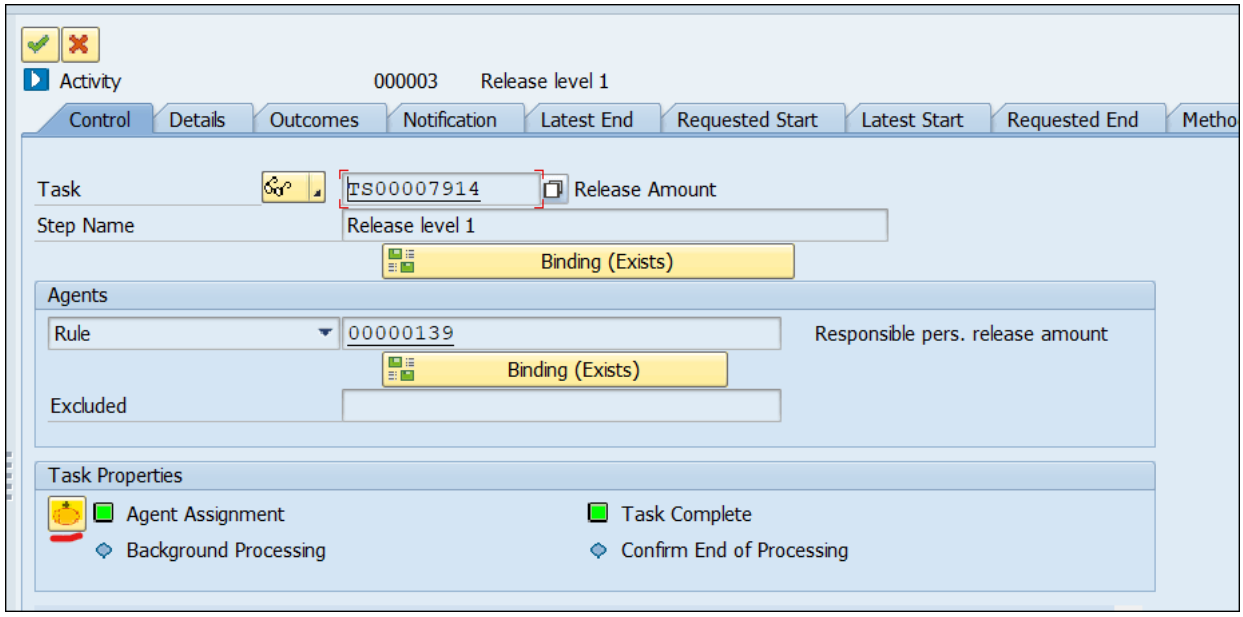


Figure 8.17 Maintain Task Classification in the Task Attribute Maintenance Screen

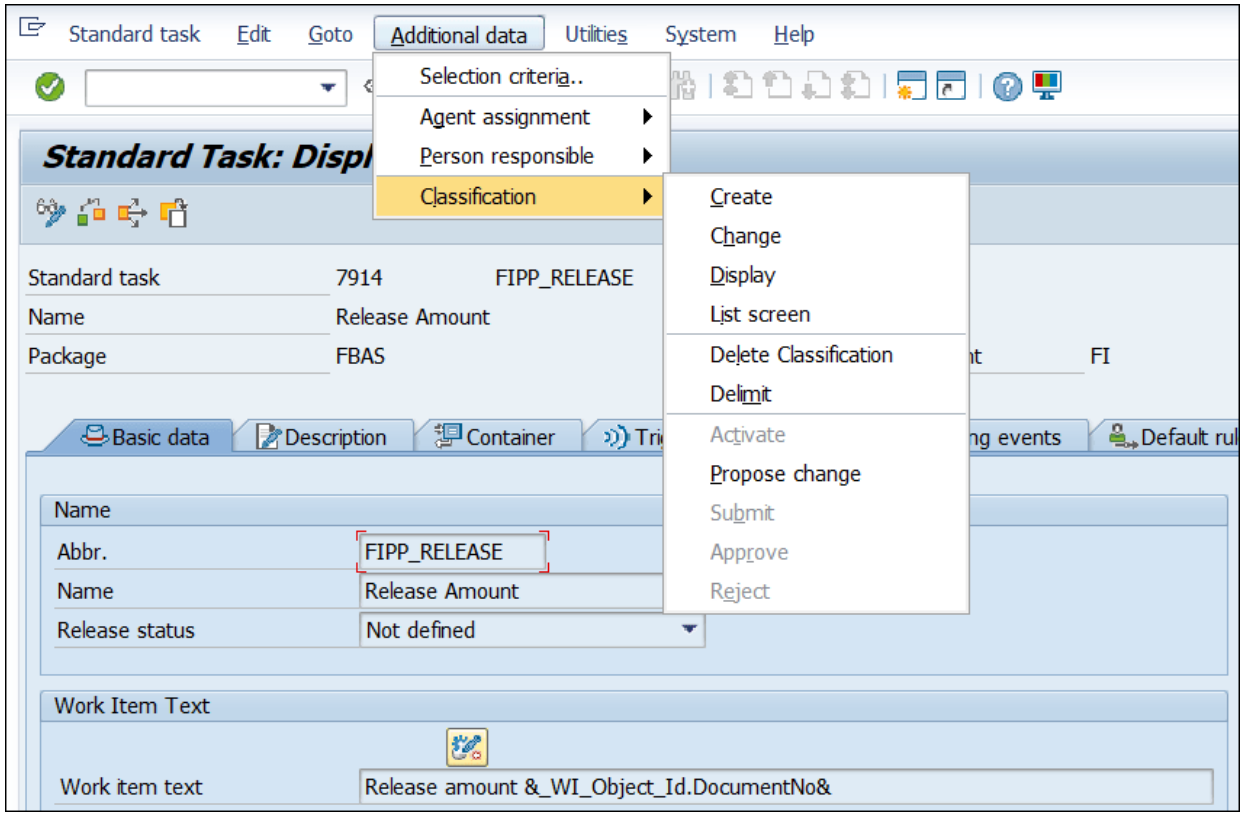


Figure 8.18 Create Classification from the Task Maintenance Screen

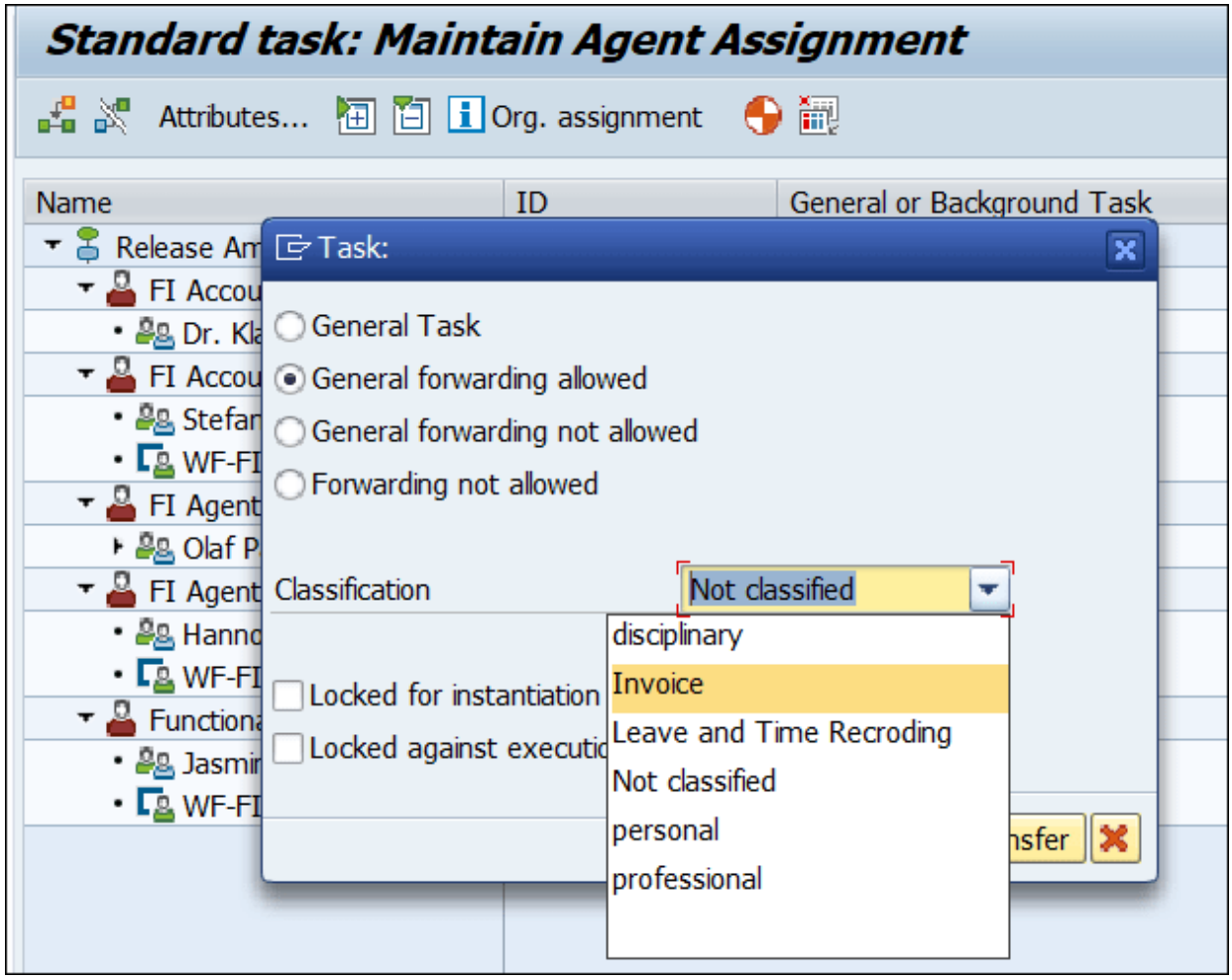


Figure 8.19 Maintenance of Task Classification

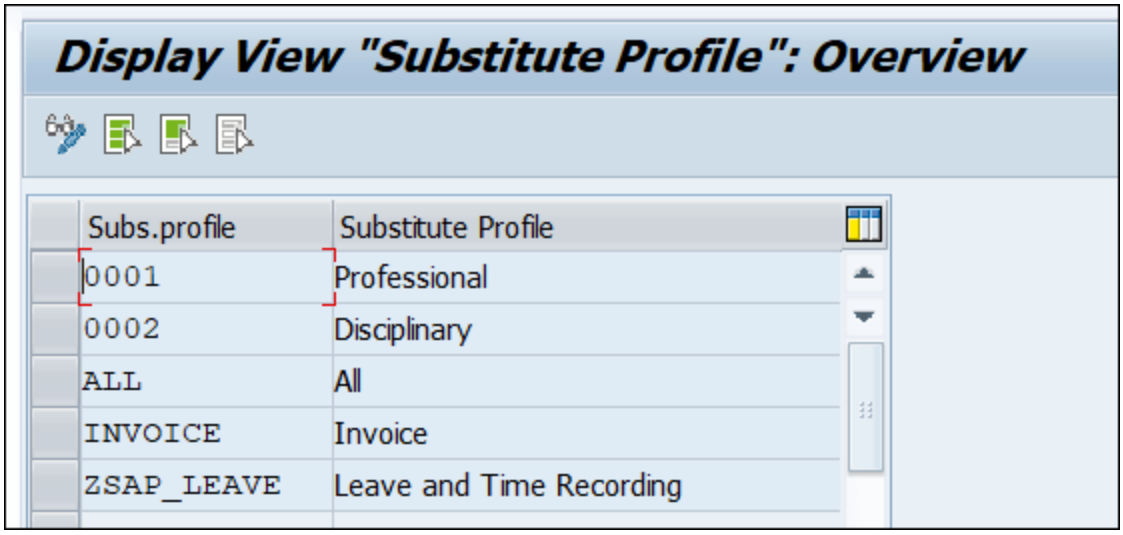
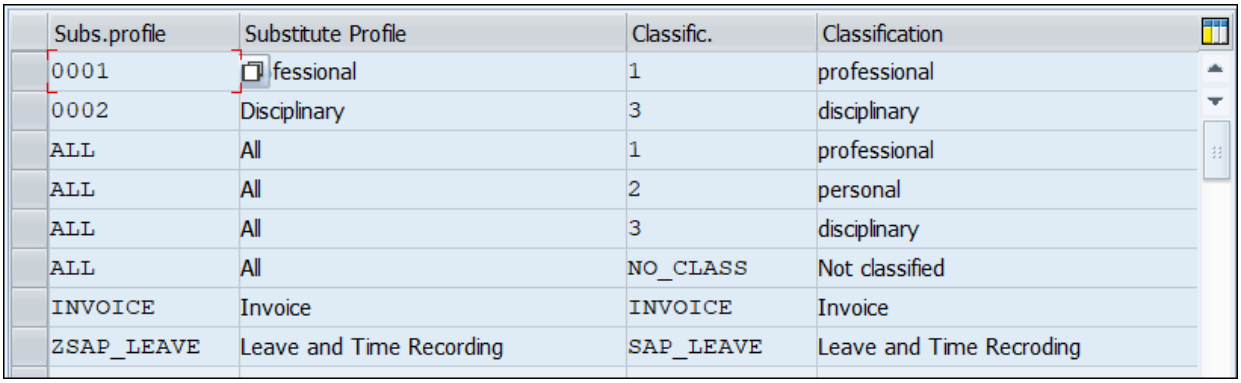


Figure 8.20 Substitute Profiles



Subs.profile	Substitute Profile	Classific.	Classification
0001	fessional	1	professional
0002	Disciplinary	3	disciplinary
ALL	All	1	professional
ALL	All	2	personal
ALL	All	3	disciplinary
ALL	All	NO_CLASS	Not classified
INVOICE	Invoice	INVOICE	Invoice
ZSAP_LEAVE	Leave and Time Recording	SAP_LEAVE	Leave and Time Recroding

Figure 8.21 Assign the Substitute Profile to Classifications

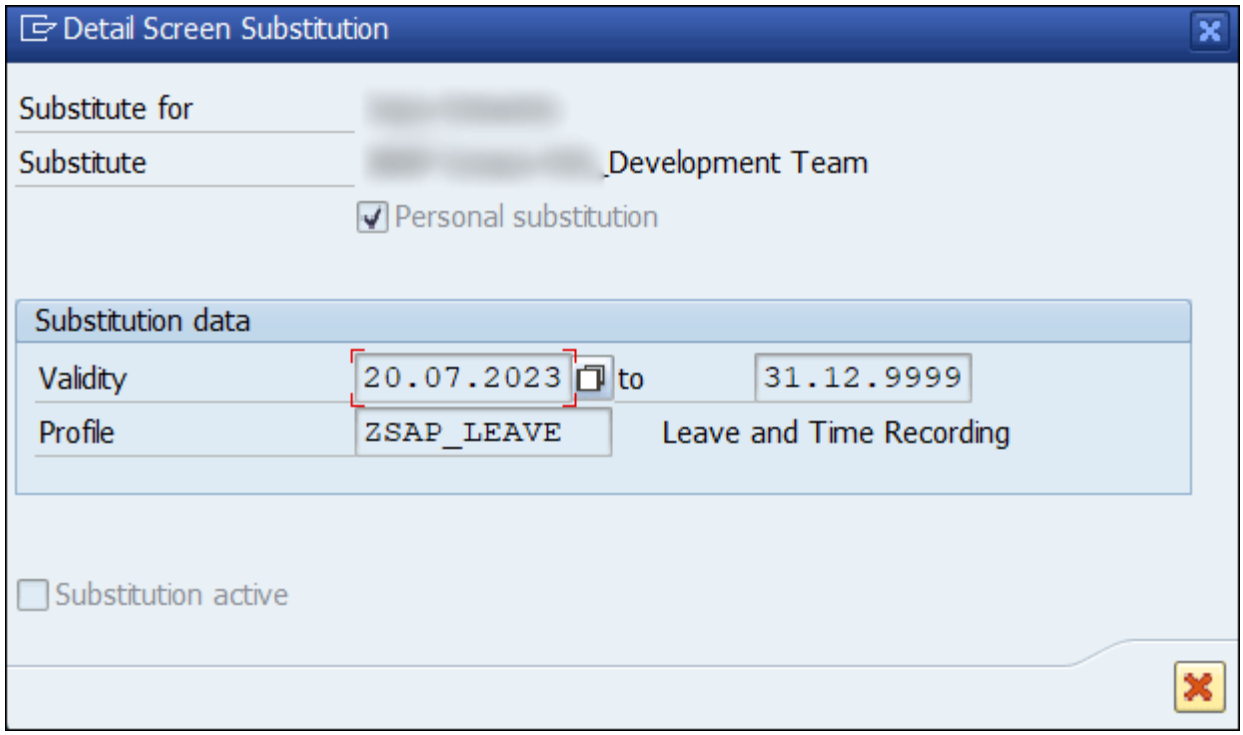


Figure 8.22 Substitute with Profile

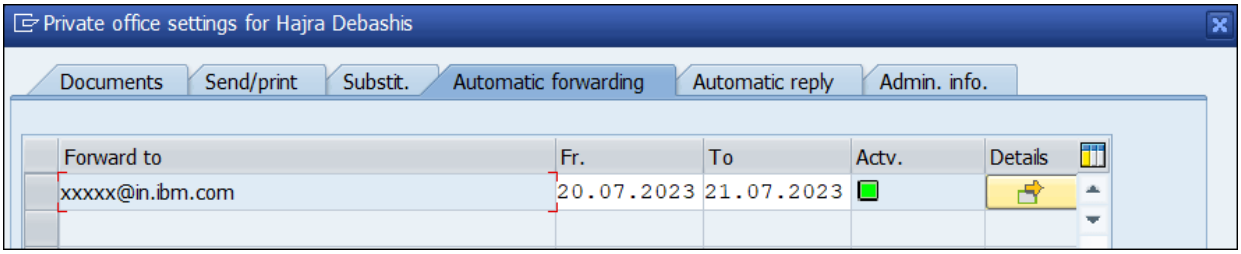


Figure 8.23 SAP Office Document Automatic Forwarding

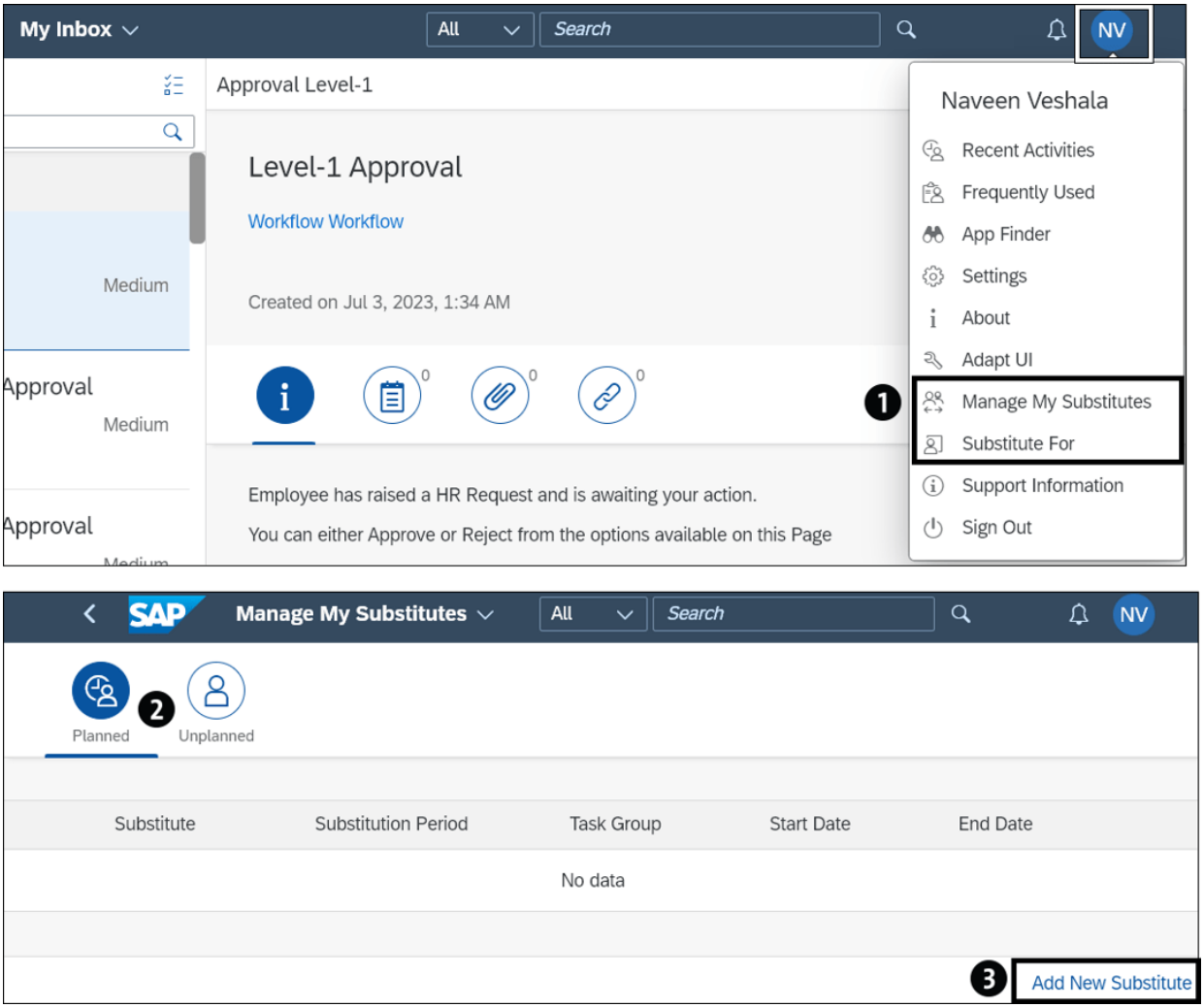


Figure 8.24 My Inbox Substitutes

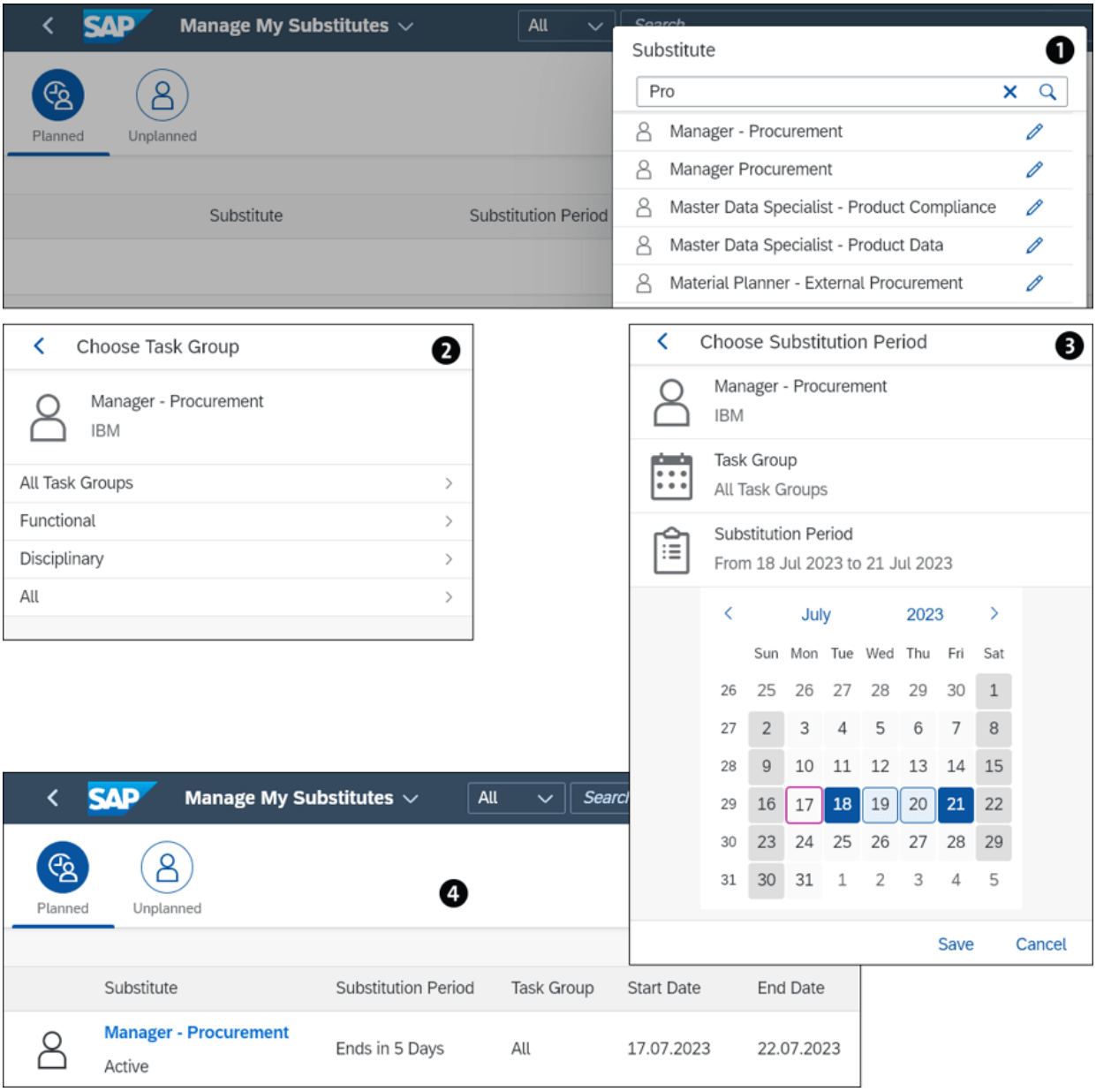


Figure 8.25 Add Planned Substitution

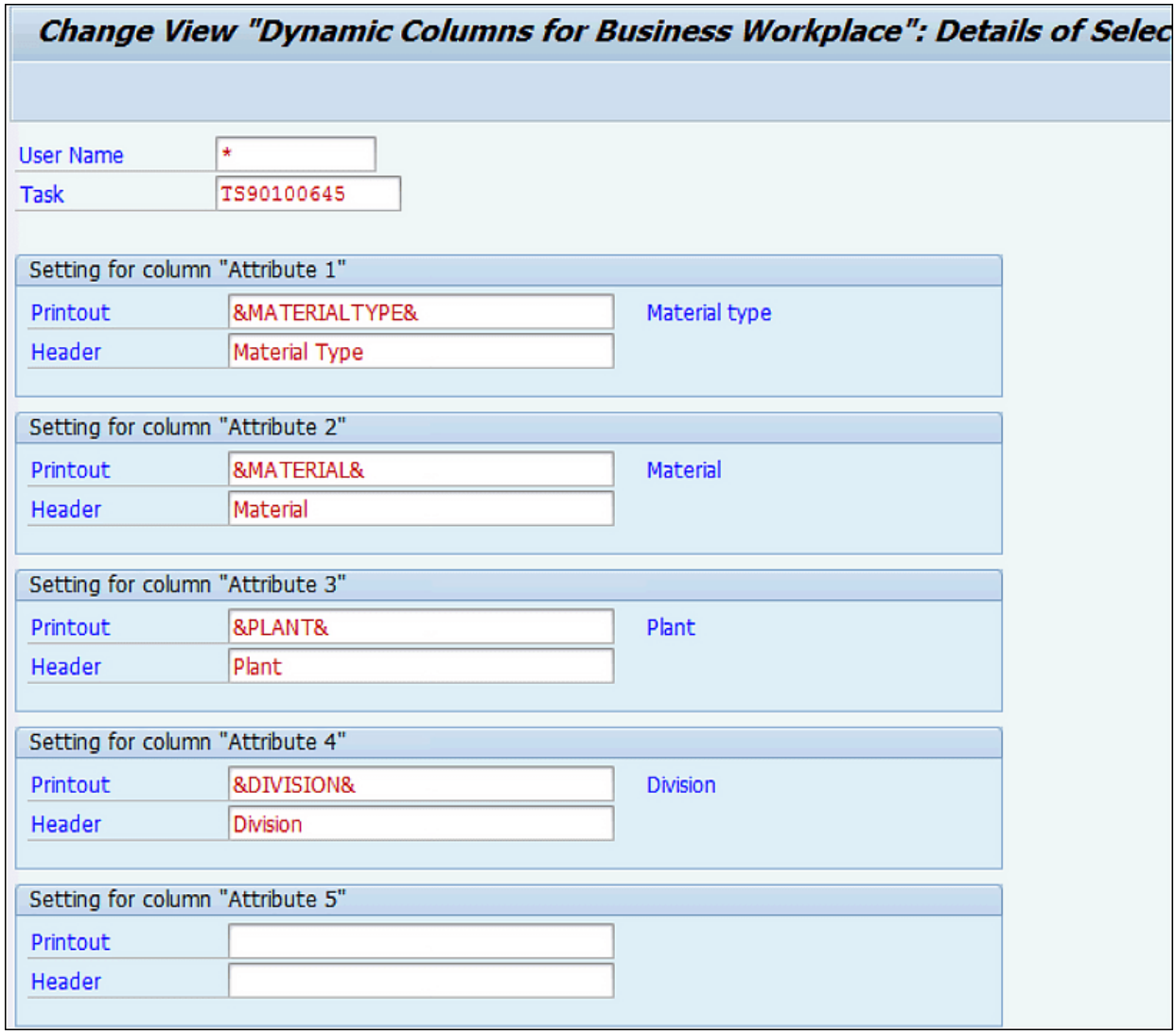


Figure 8.26 Dynamic Column Configuration for Business Workplace

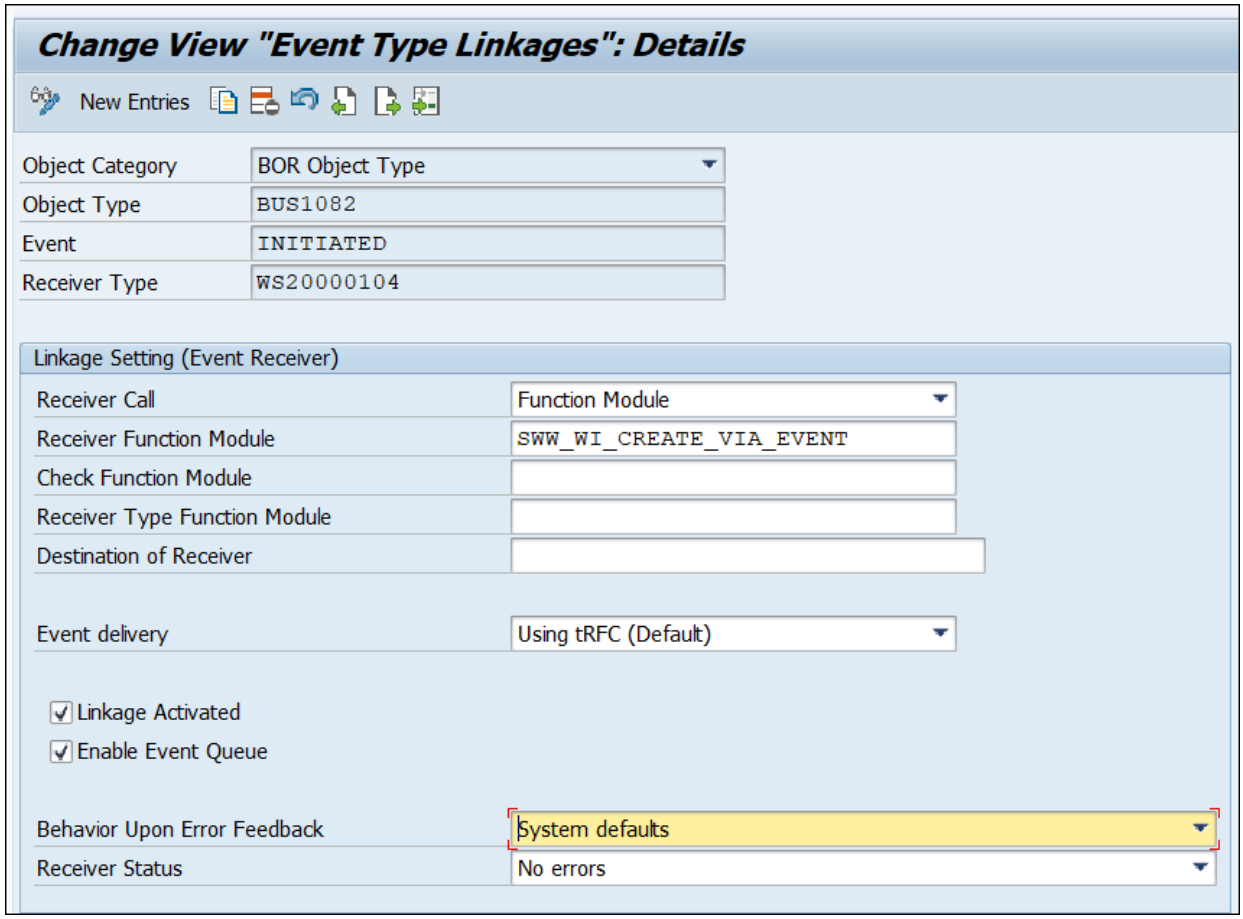


Figure 8.27 Event Linkage

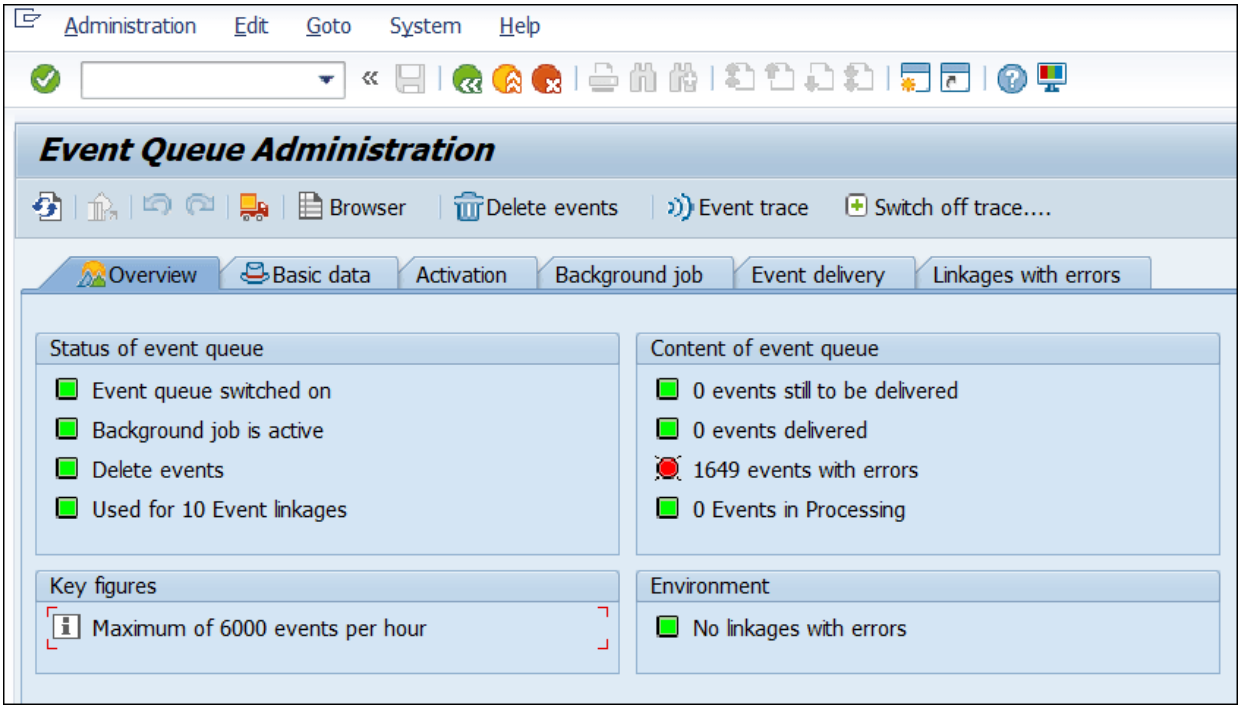


Figure 8.28 Event Queue Administration Overview

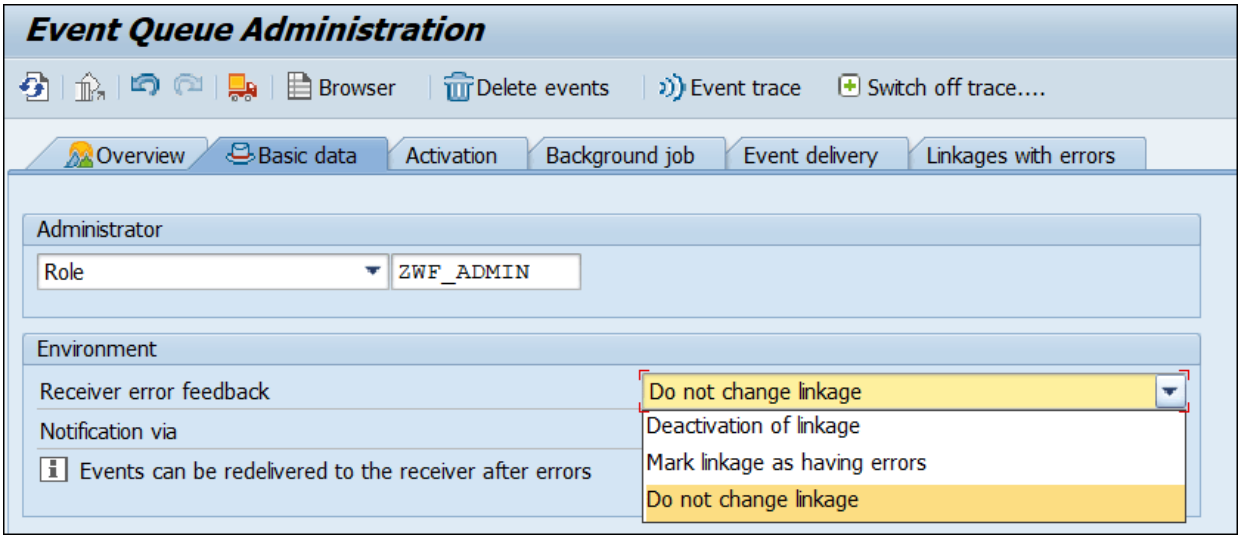


Figure 8.29 Event Queue Administration Basic Data

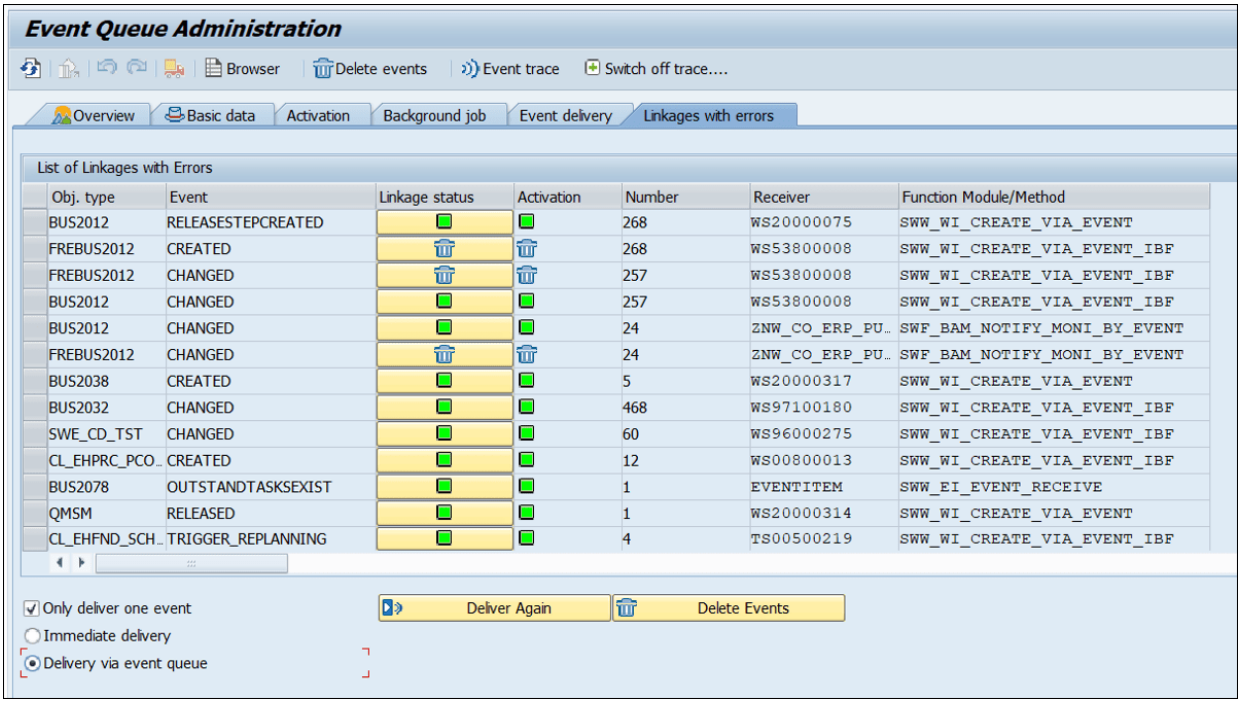


Figure 8.30 Event Queue Linkage with Errors

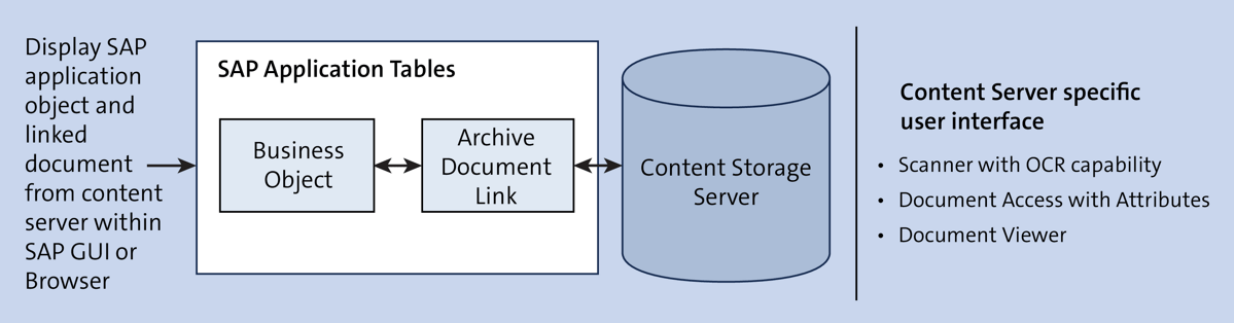


Figure 8.31 Content Server and ArchiveLink Overview

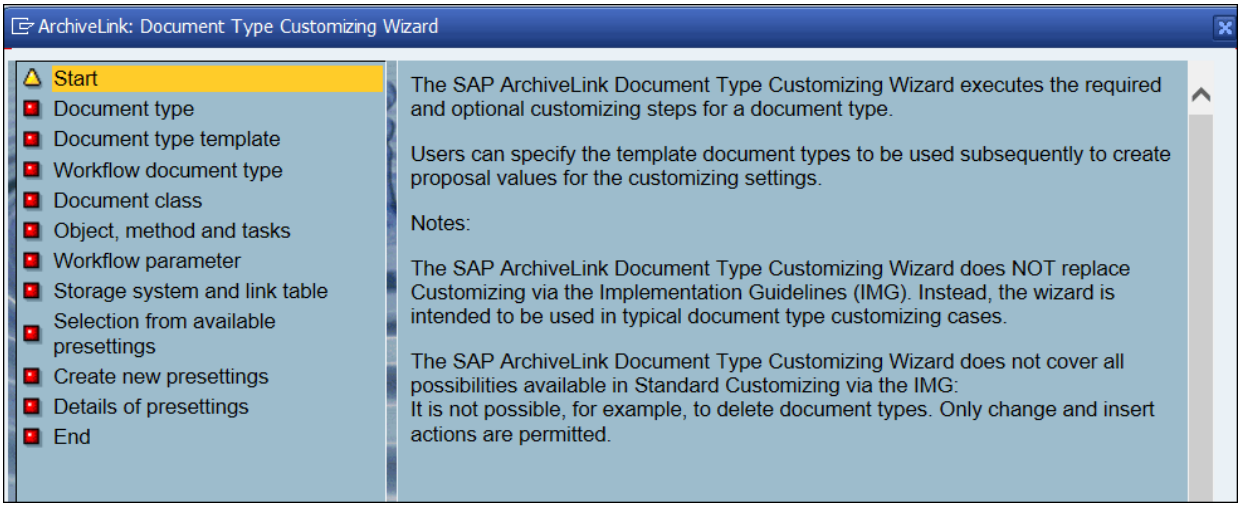


Figure 8.32 Document Type Customizing Wizard

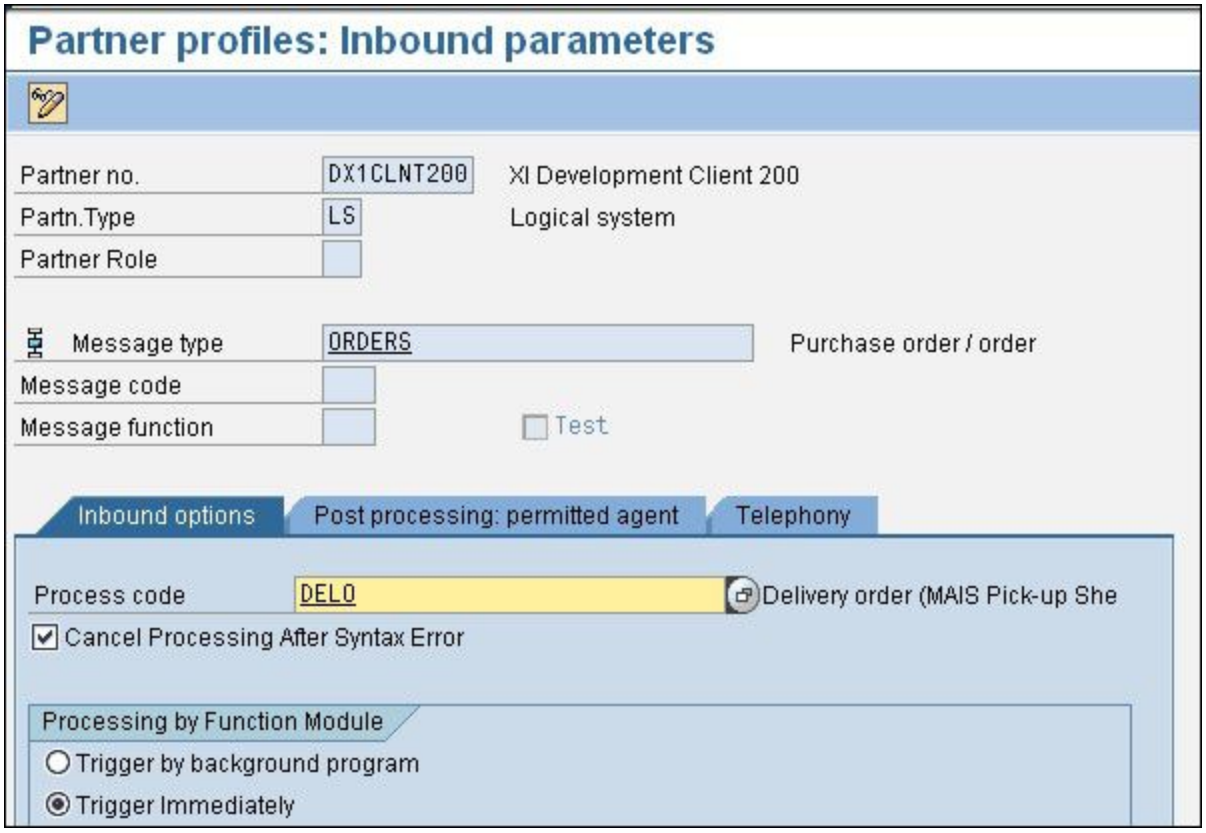


Figure 9.1 Partner Profile Configuration

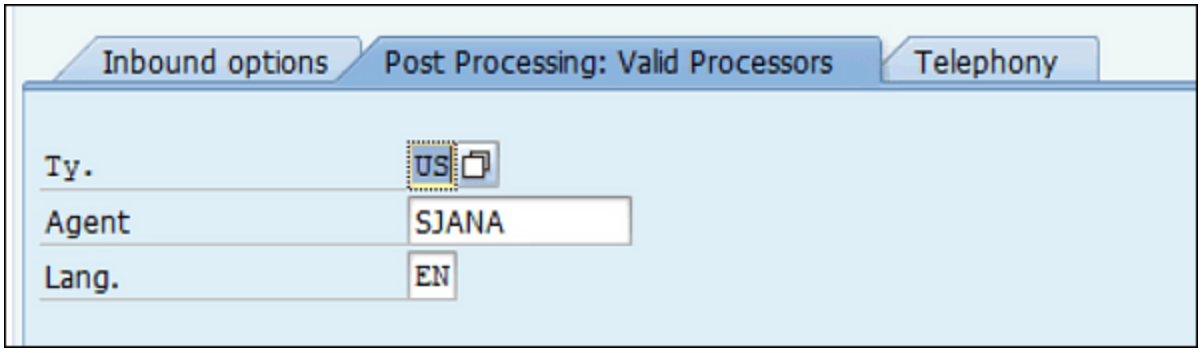


Figure 9.2 Post Processing Agent for IDoc

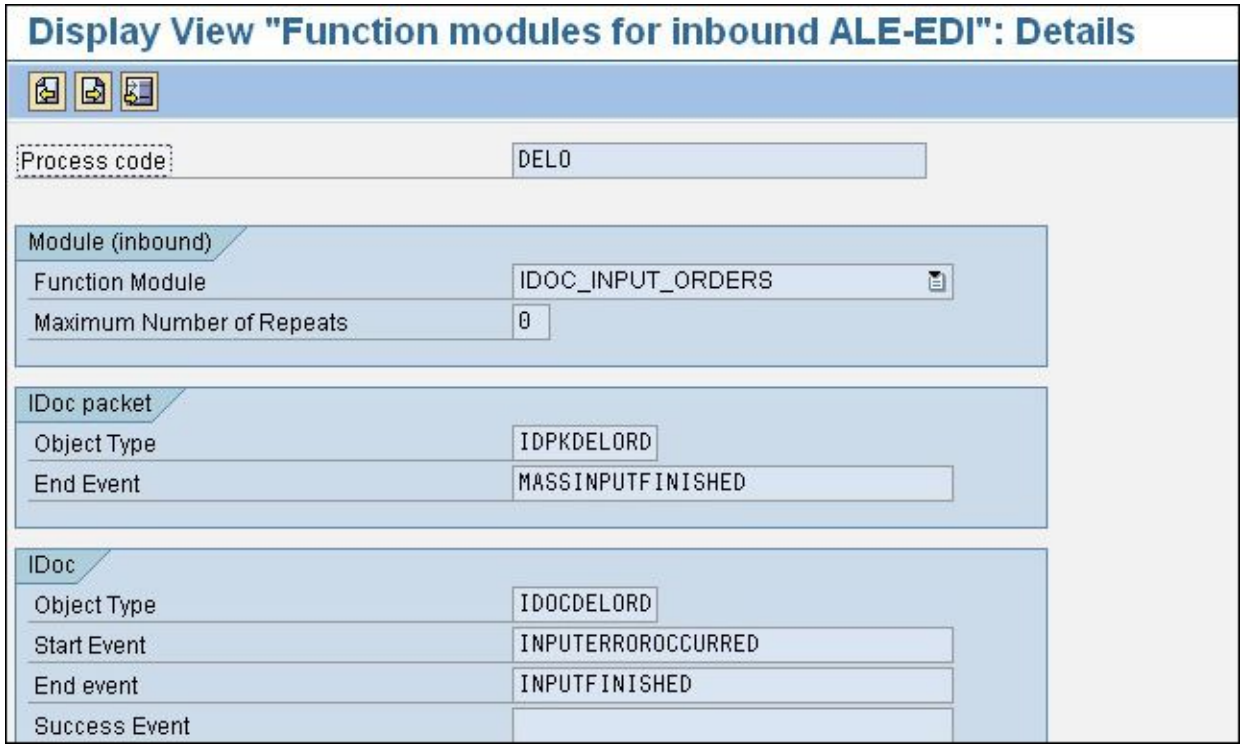


Figure 9.3 Process Code Configuration

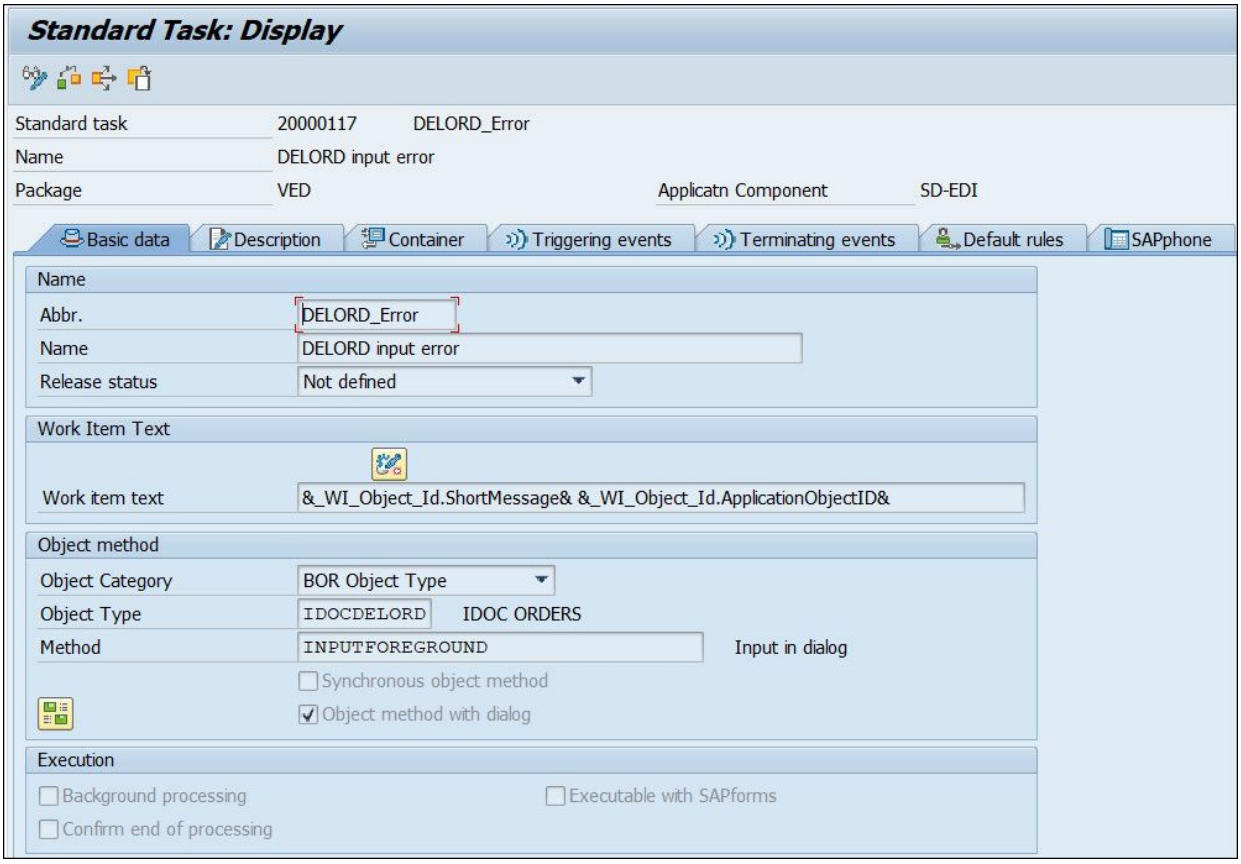


Figure 9.4 Standard Task to Process DELORD IDoc Errors

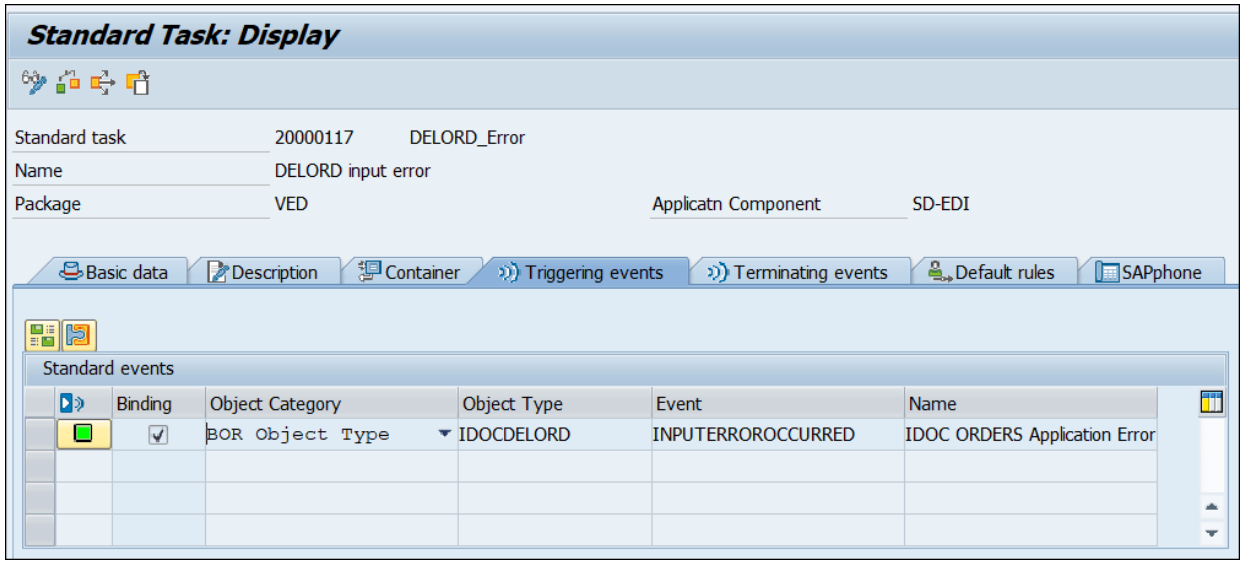


Figure 9.5 Triggering Event for IDoc Error Handling Task

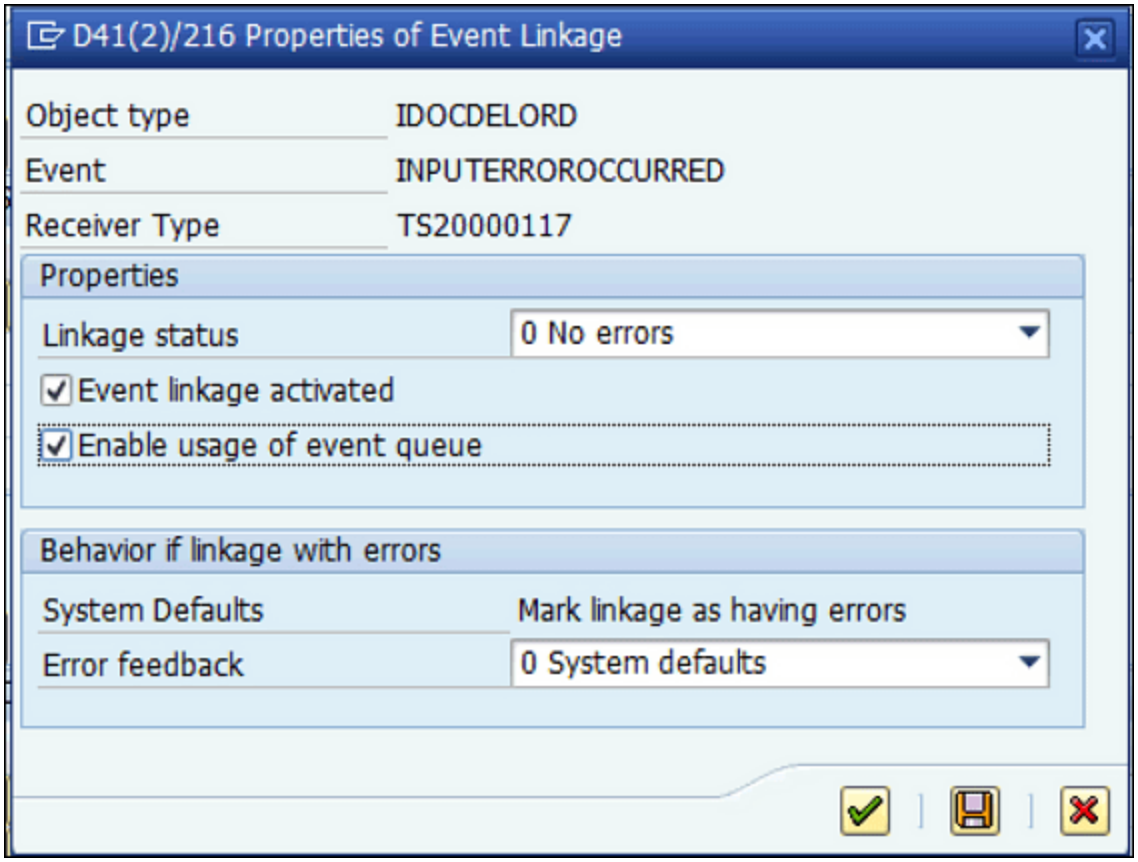


Figure 9.6 Activating Event Linkage for the IDoc Failure Processing Task

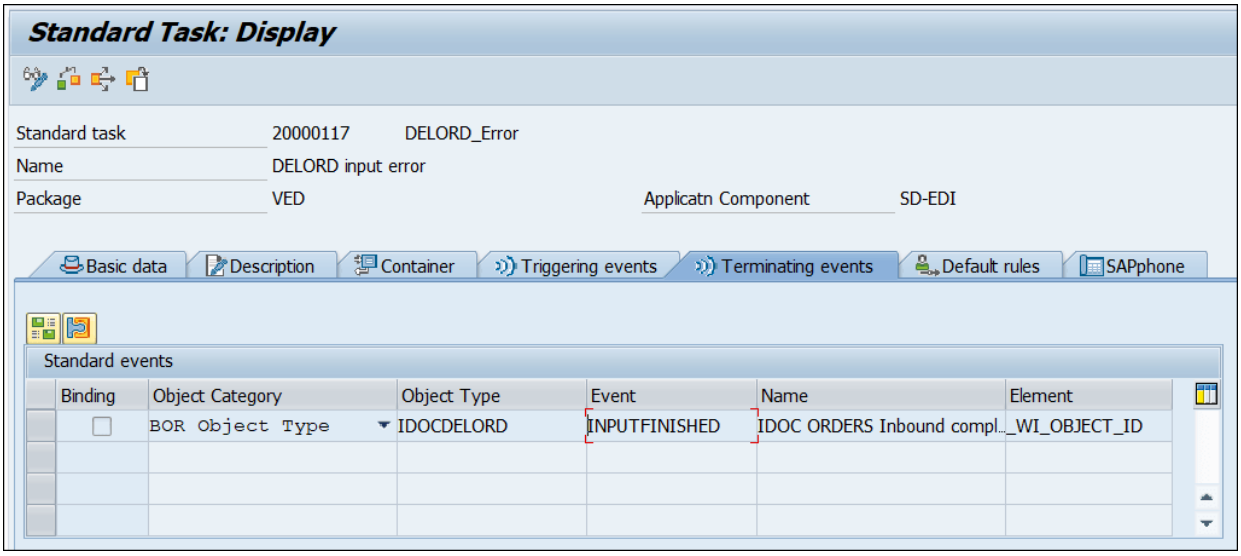


Figure 9.7 Terminating Event to Complete the Work Item

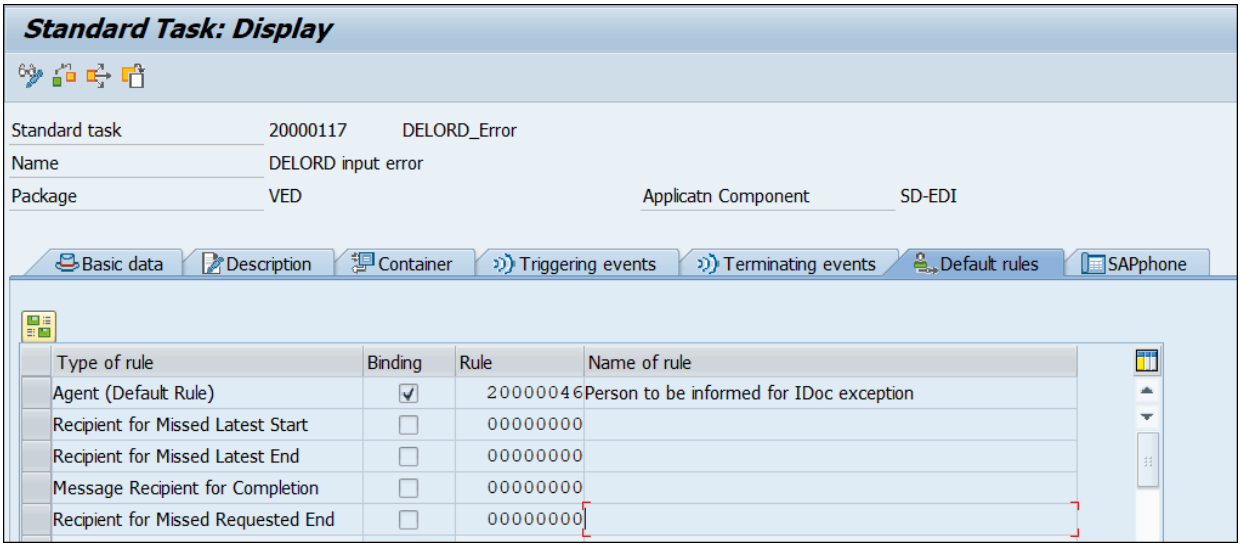


Figure 9.8 Default Rule for Work Item Agent to Process the Workflow Task

EVCODE	FUNCNAME	MAXRETRIES	EVENT_TYPE	MASS_EVENT	PACKETOBJT	IDOCOBJTYP	APPLOBJTYP	EVENT_END
IMAT	IDOC_INPUT_IMATIS	00	INPUTERROROCCURRED			IDOCWMMBXY	BUS2017	INPUTFINISHED
WMMB	L_IDOC_INPUT_WMMBXY	00	INPUTERROROCCURRED	MASSINPUTFINISHED	IDPKWMMBXY	IDOCWMMBXY	BUS2017	INPUTFINISHED
ZWMMB	Z_IDOC_INPUT_WMMBXY	00	INPUTERROROCCURRED	MASSINPUTFINISHED	IDPKWMMBXY	IDOCWMMBXY	BUS2017	INPUTFINISHED

Figure 9.9 Same IDoc Error-Handling Event Used for Multiple Process Codes

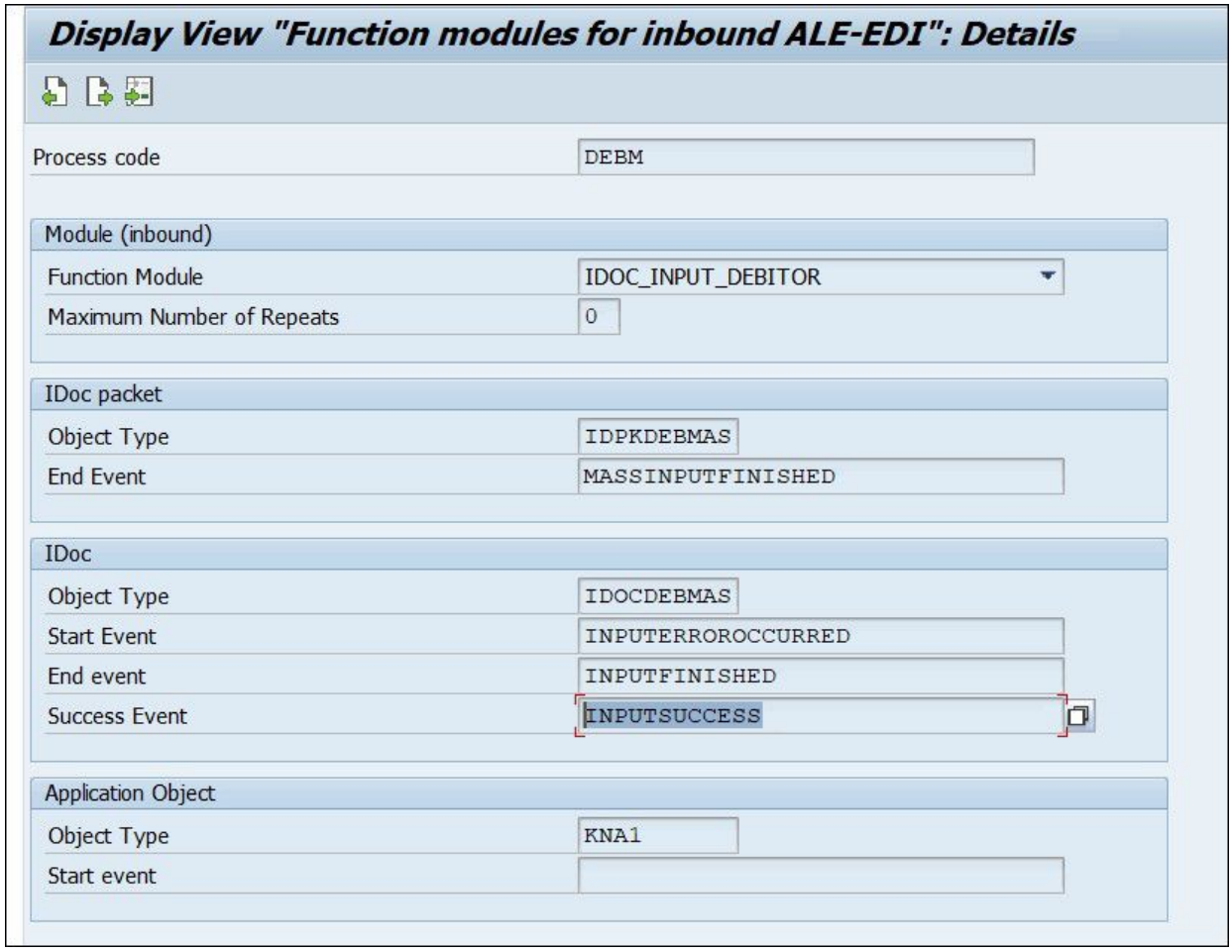


Figure 9.10 Success Event INPUTSUCCESS Configured for the Process Code

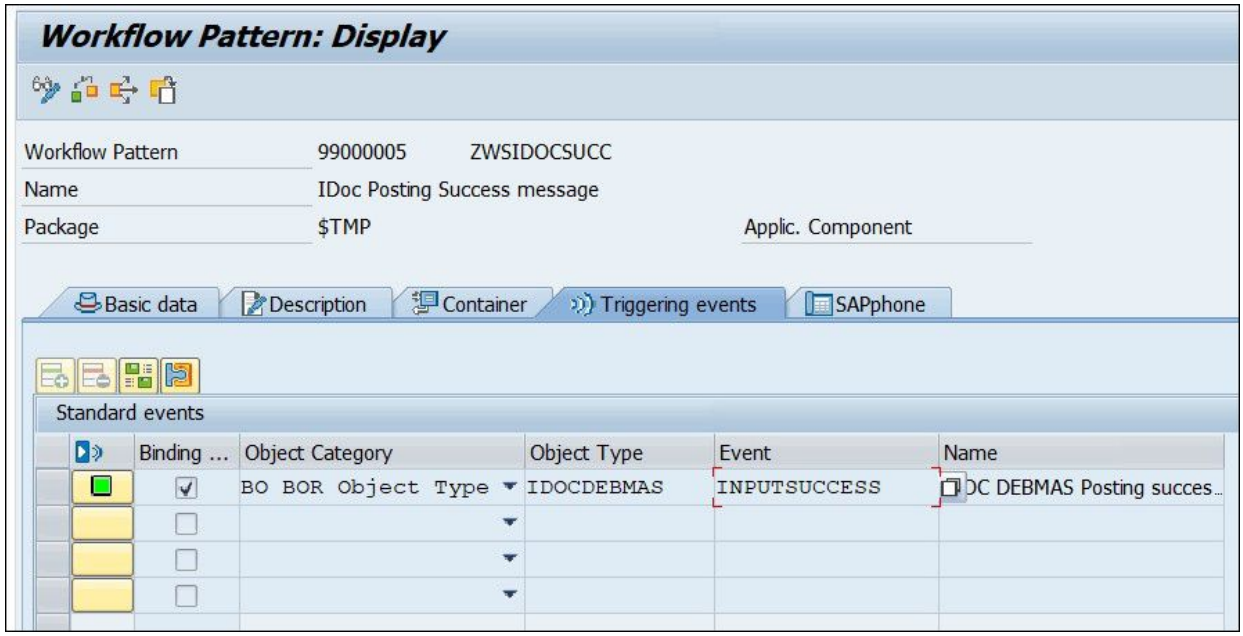


Figure 9.11 Event Linkage Entry with IDoc Object Type, Success Event, and Custom Workflow

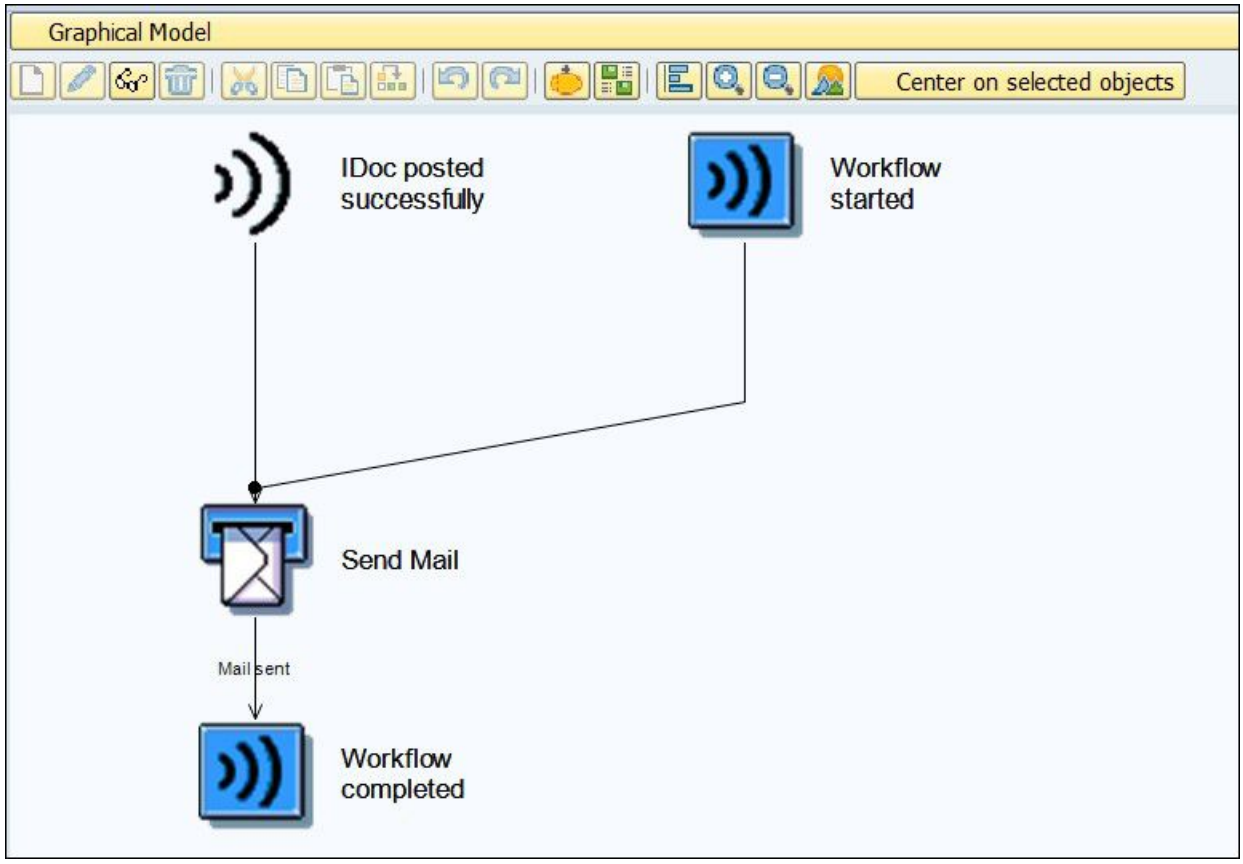


Figure 9.12 Workflow Definition for Successful Notification of IDoc Posting

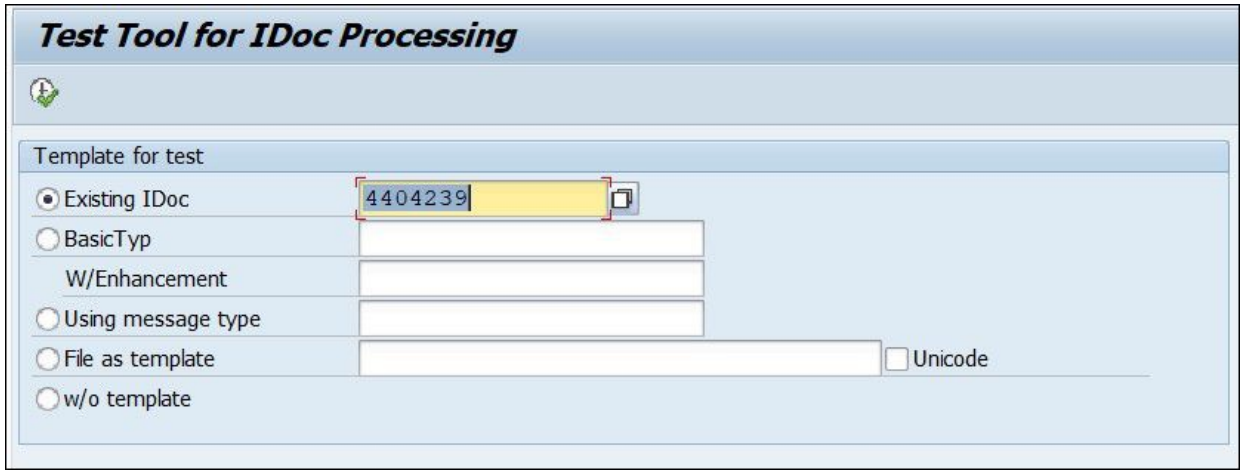


Figure 9.13 Creating an IDoc as a Copy from Transaction WE19

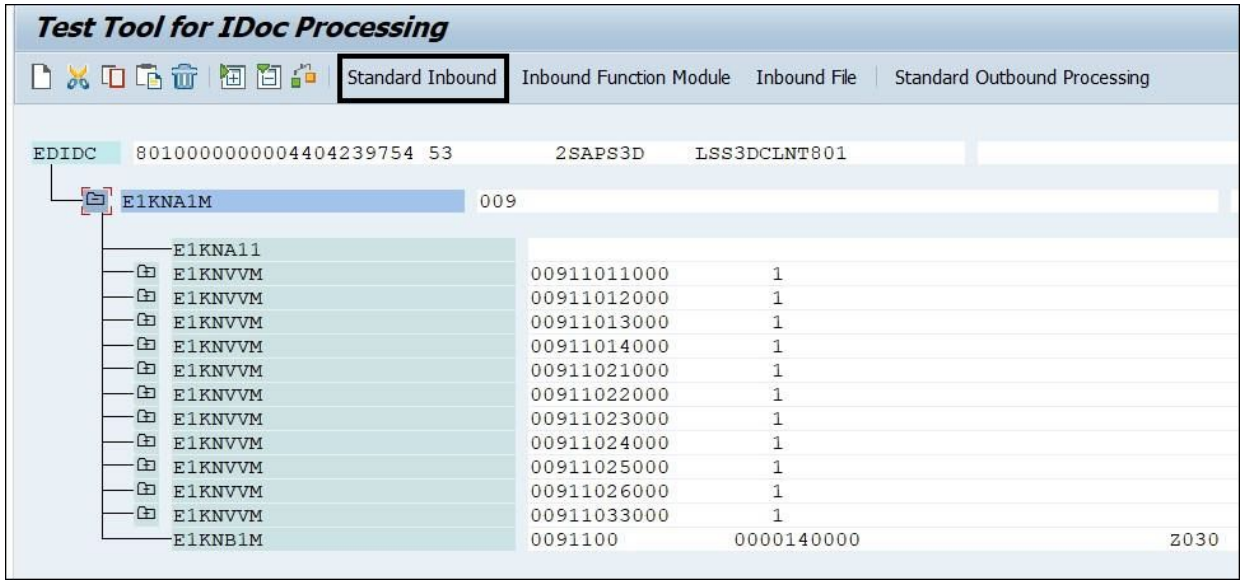


Figure 9.14 Posting an IDoc via Transaction WE19: IDoc Test Transaction

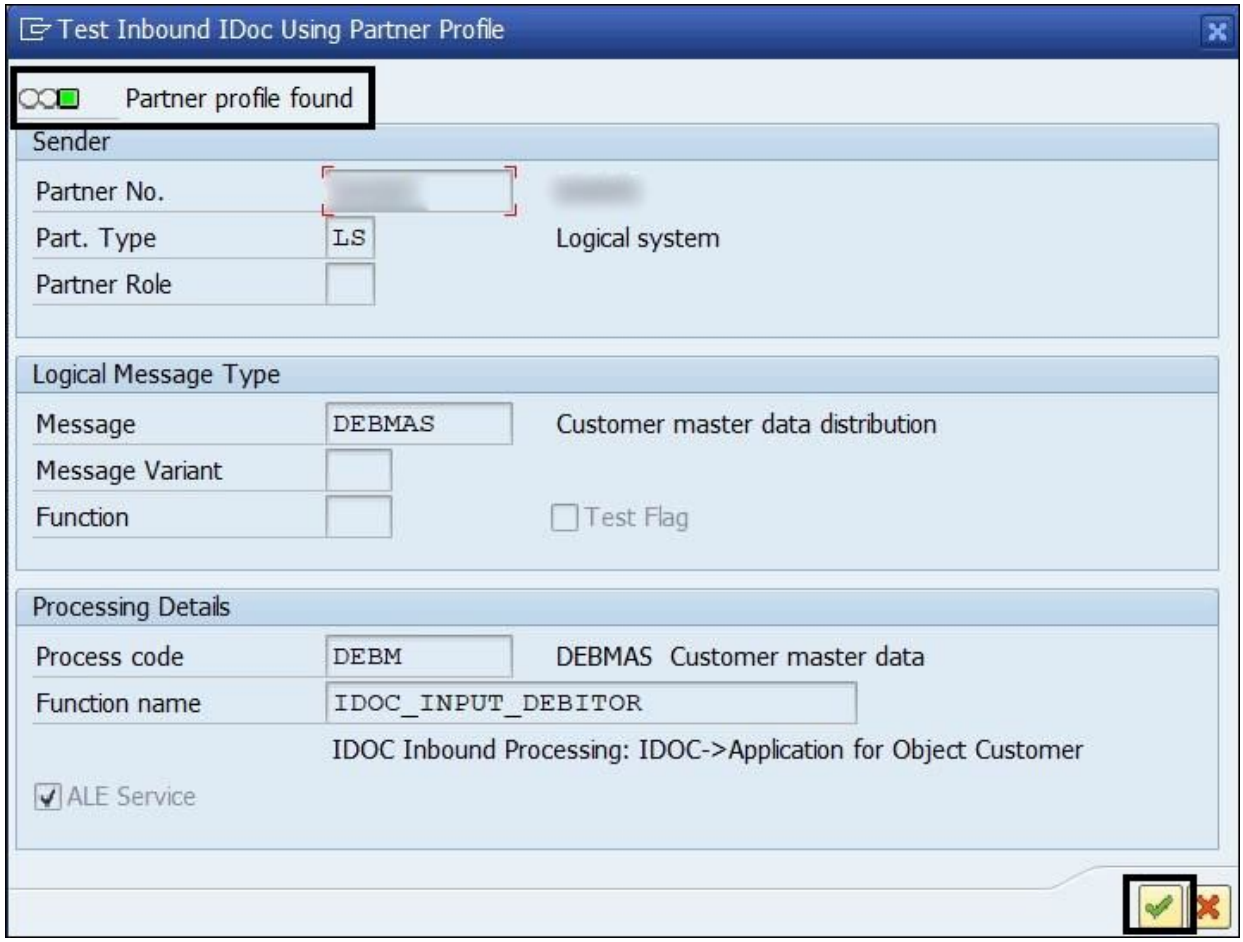


Figure 9.15 Partner Profile Confirmation Popup before IDoc Posting in Transaction WE19

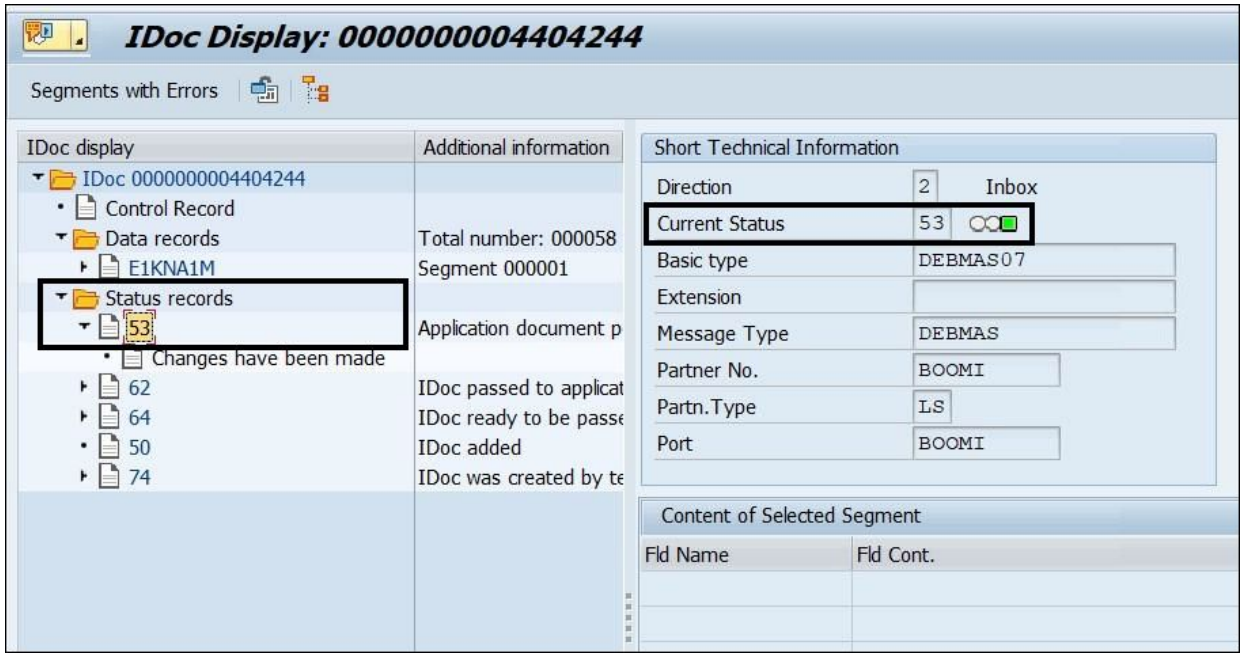


Figure 9.16 IDoc Status Check from Transaction WE02

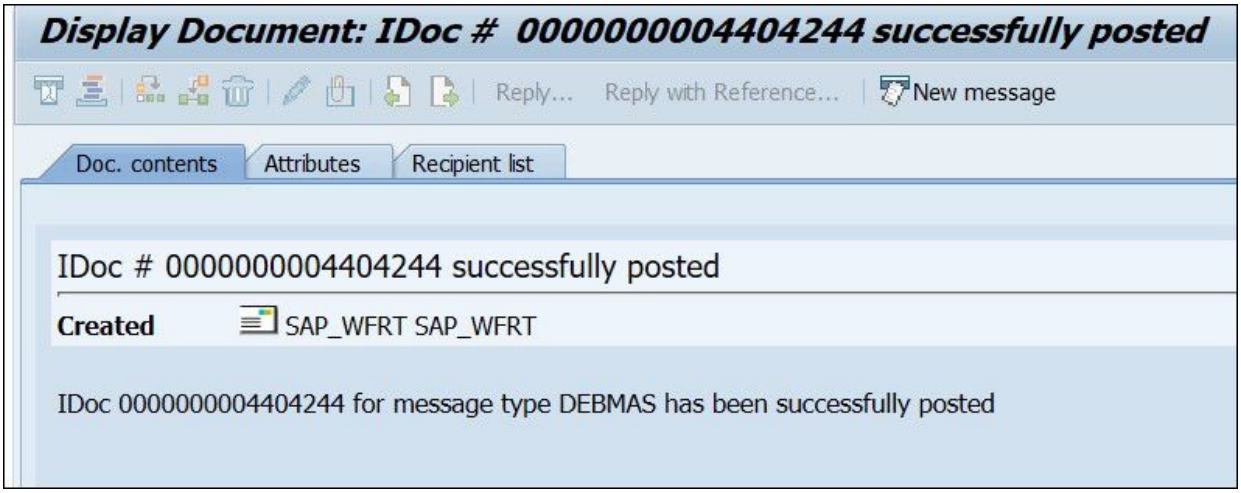


Figure 9.17 IDoc Posting Notification as Viewed from Transaction SOST

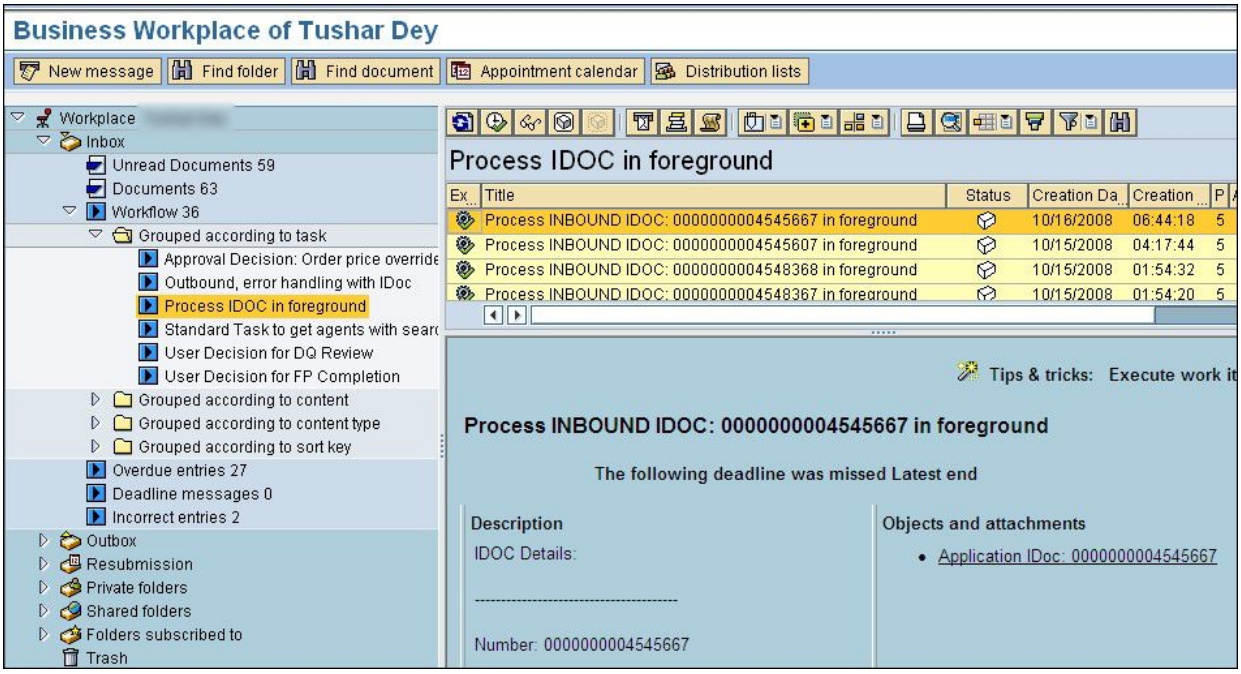


Figure 9.18 Work Item Created in the User Inbox to Process the IDoc in the Foreground

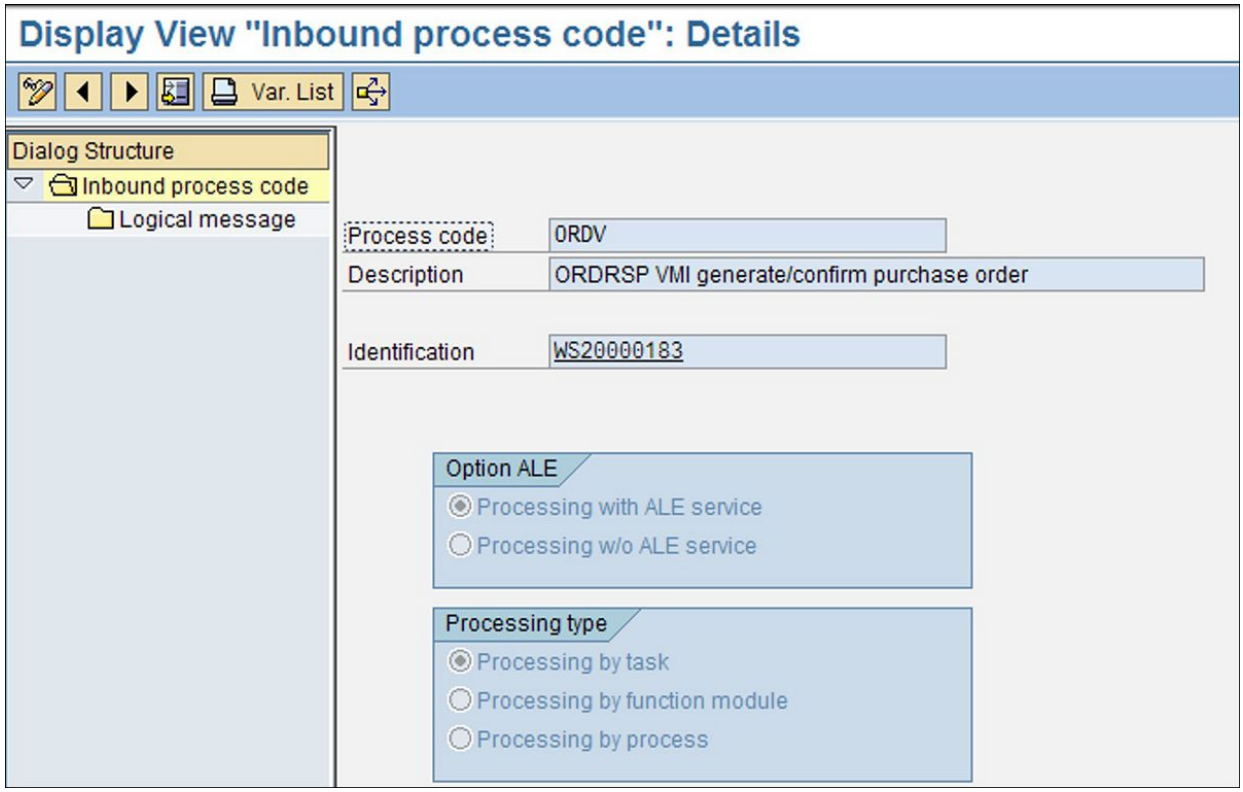


Figure 9.19 Settings to Process the IDoc with the Workflow

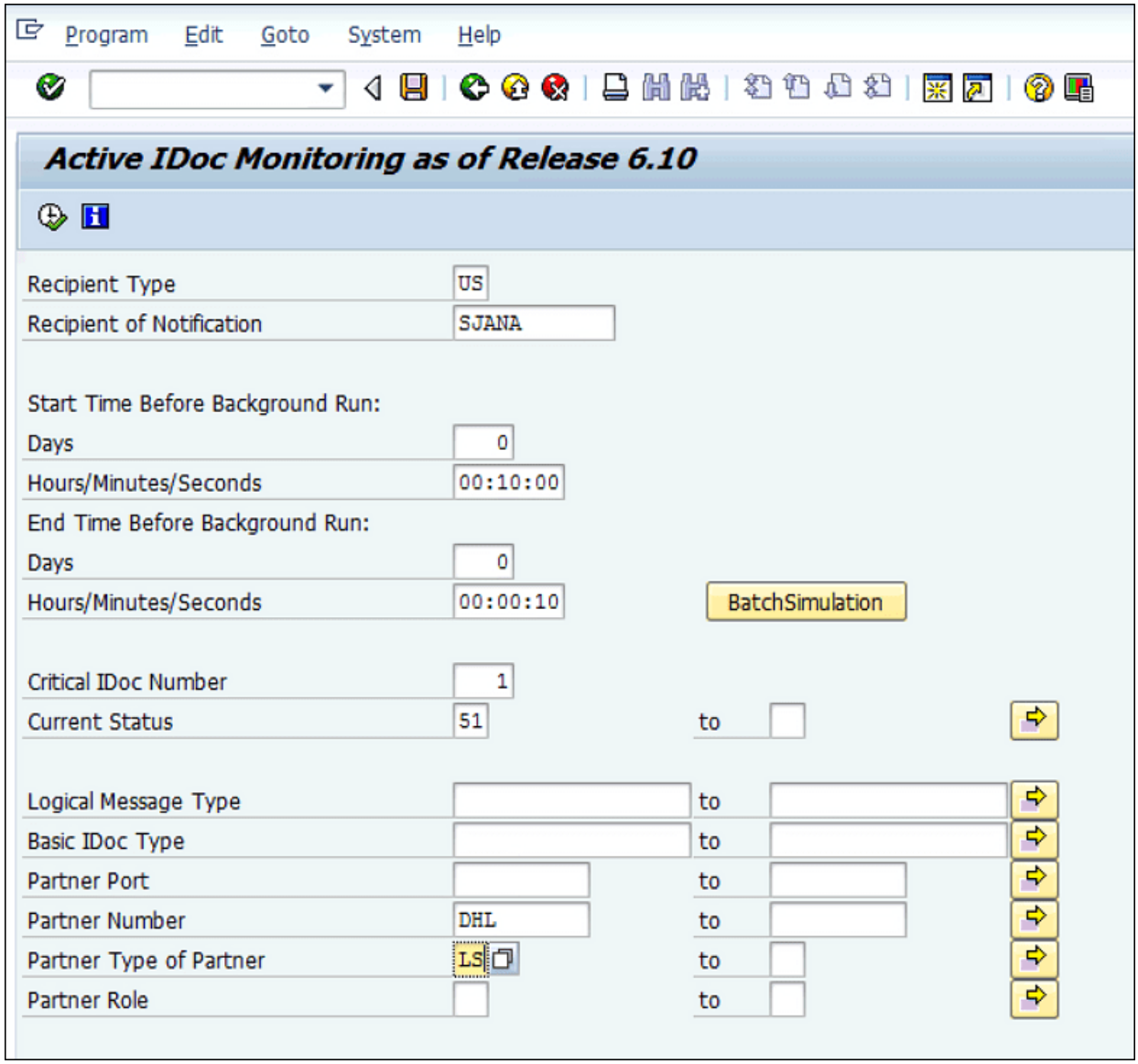


Figure 9.20 Selection Screen of the Active Monitoring Batch Job

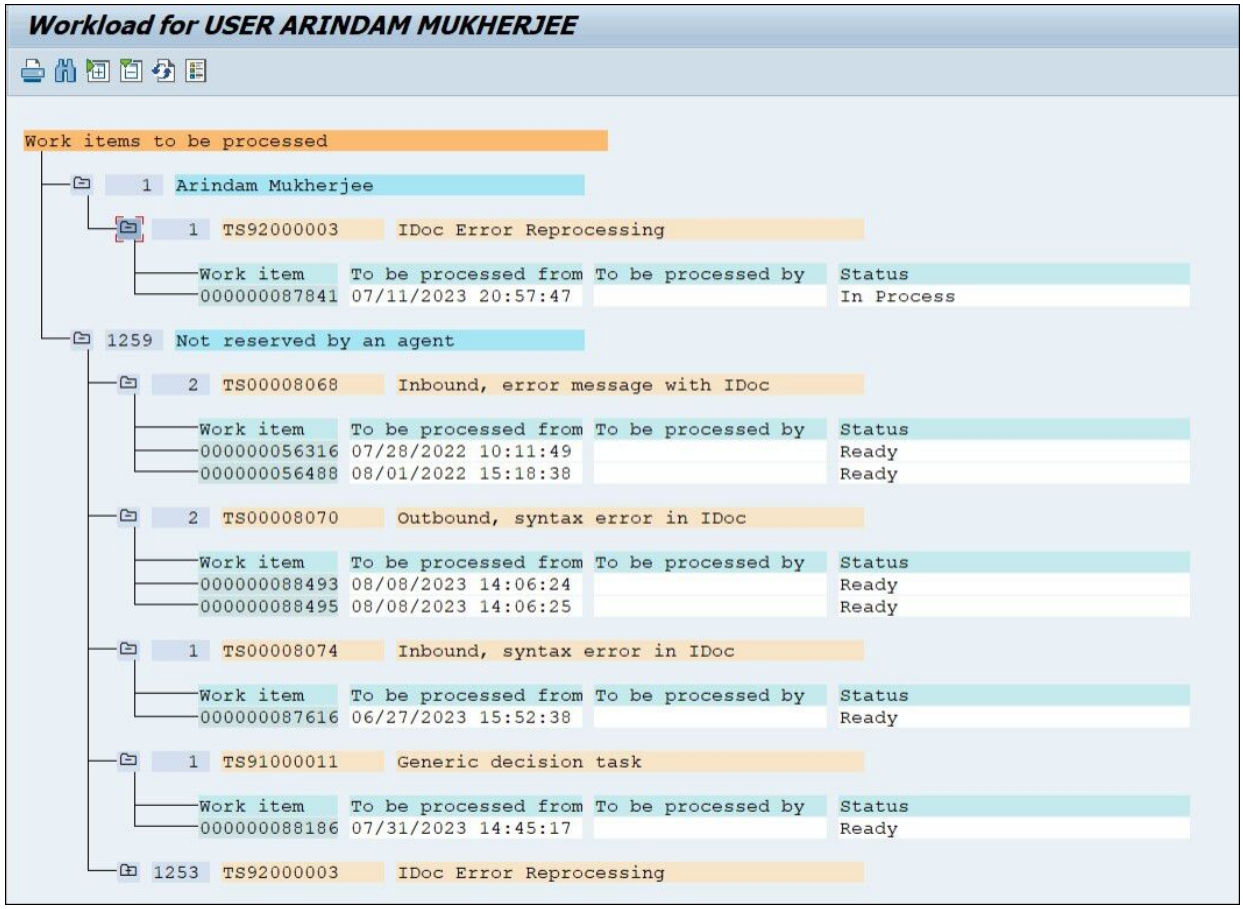
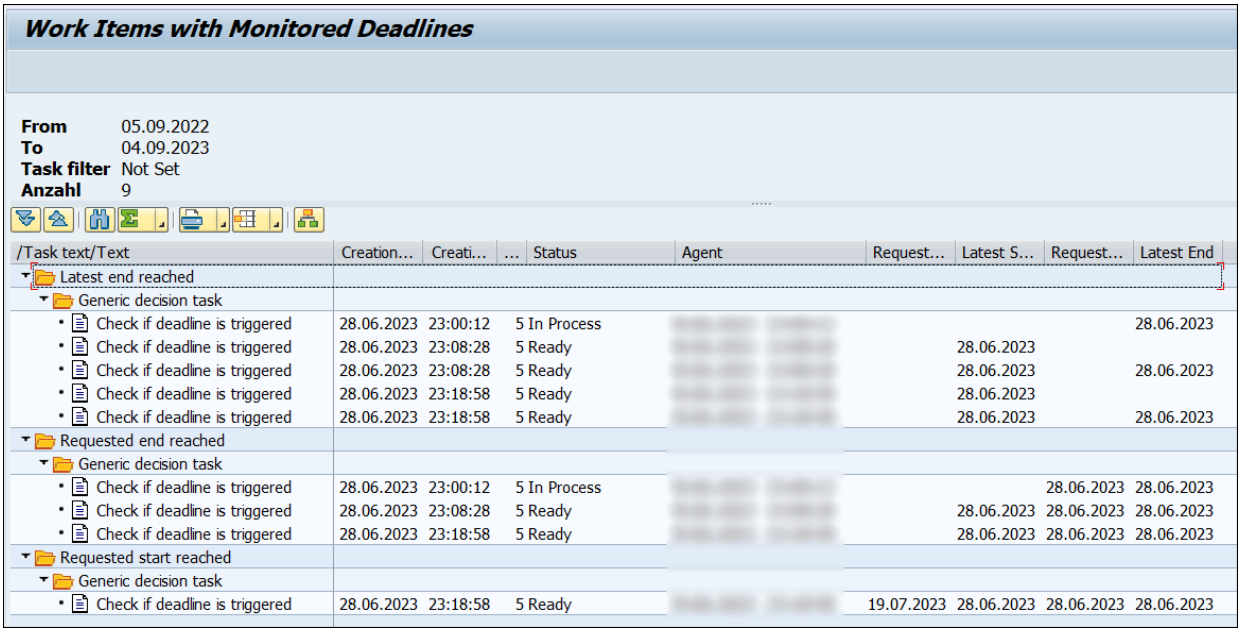


Figure 9.21 Workload Analysis (Transaction SWI5) Sample Report Output

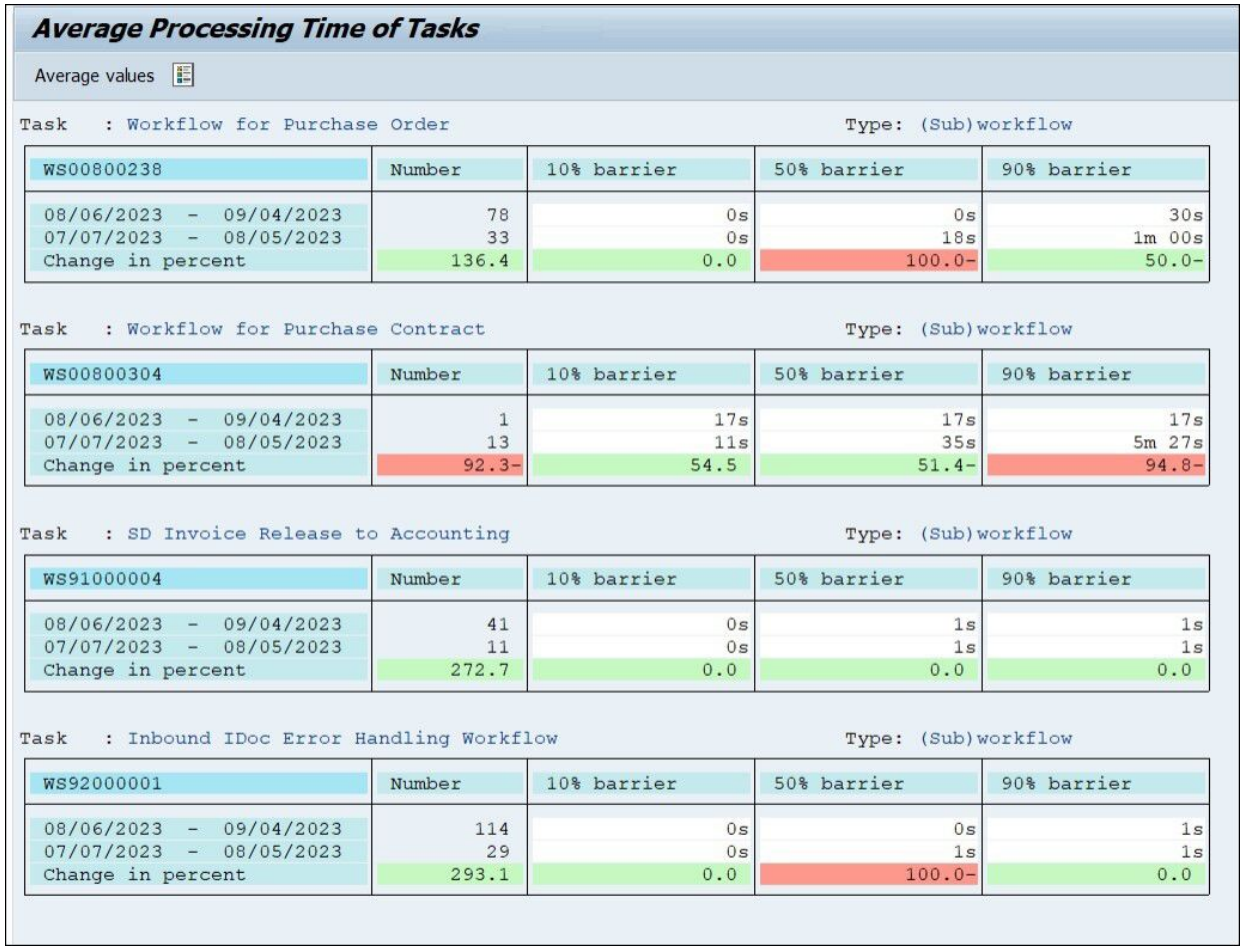


Work Items with Monitored Deadlines

From 05.09.2022
To 04.09.2023
Task filter Not Set
Anzahl 9

/Task text/Text	Creation...	Creati...	...	Status	Agent	Request...	Latest S...	Request...	Latest End
Latest end reached									
Generic decision task									
Check if deadline is triggered	28.06.2023	23:00:12		5 In Process					28.06.2023
Check if deadline is triggered	28.06.2023	23:08:28		5 Ready			28.06.2023		
Check if deadline is triggered	28.06.2023	23:08:28		5 Ready			28.06.2023		28.06.2023
Check if deadline is triggered	28.06.2023	23:18:58		5 Ready			28.06.2023		
Check if deadline is triggered	28.06.2023	23:18:58		5 Ready			28.06.2023		28.06.2023
Requested end reached									
Generic decision task									
Check if deadline is triggered	28.06.2023	23:00:12		5 In Process				28.06.2023	28.06.2023
Check if deadline is triggered	28.06.2023	23:08:28		5 Ready			28.06.2023	28.06.2023	28.06.2023
Check if deadline is triggered	28.06.2023	23:18:58		5 Ready			28.06.2023	28.06.2023	28.06.2023
Requested start reached									
Generic decision task									
Check if deadline is triggered	28.06.2023	23:18:58		5 Ready		19.07.2023	28.06.2023	28.06.2023	28.06.2023

Figure 9.22 Monitored Deadline (Transaction SWI2_DEAD) Sample Report Output



Average Processing Time of Tasks

Average values

Task : Workflow for Purchase Order — Type: (Sub)workflow

WS00800238	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	78	0s	0s	30s
07/07/2023 - 08/05/2023	33	0s	18s	1m 00s
Change in percent	136.4	0.0	100.0-	50.0-

Task : Workflow for Purchase Contract — Type: (Sub)workflow

WS00800304	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	1	17s	17s	17s
07/07/2023 - 08/05/2023	13	11s	35s	5m 27s
Change in percent	92.3-	54.5	51.4-	94.8-

Task : SD Invoice Release to Accounting — Type: (Sub)workflow

WS91000004	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	41	0s	1s	1s
07/07/2023 - 08/05/2023	11	0s	1s	1s
Change in percent	272.7	0.0	0.0	0.0

Task : Inbound IDoc Error Handling Workflow — Type: (Sub)workflow

WS92000001	Number	10% barrier	50% barrier	90% barrier
08/06/2023 - 09/04/2023	114	0s	0s	1s
07/07/2023 - 08/05/2023	29	0s	1s	1s
Change in percent	293.1	0.0	100.0-	0.0

Figure 9.23 Work Items by Processing Duration (Transaction SWI2_DURA) Sample Report Output

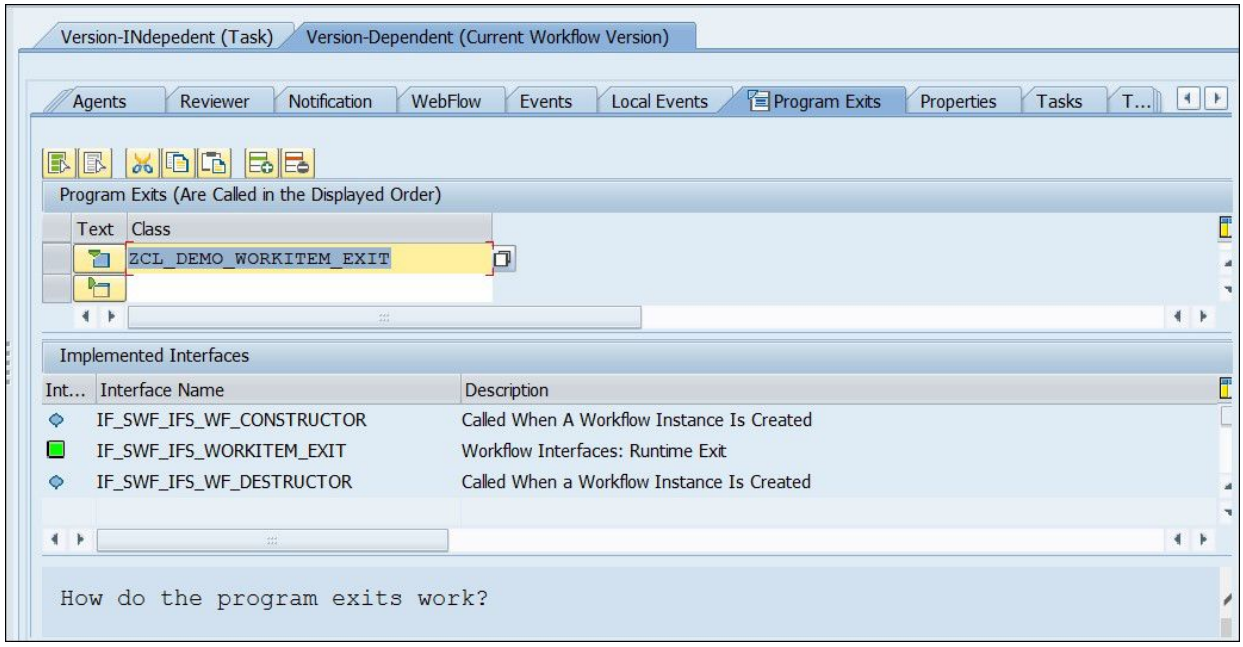


Figure 9.24 Maintenance of a Program Exit at the Workflow Header Level

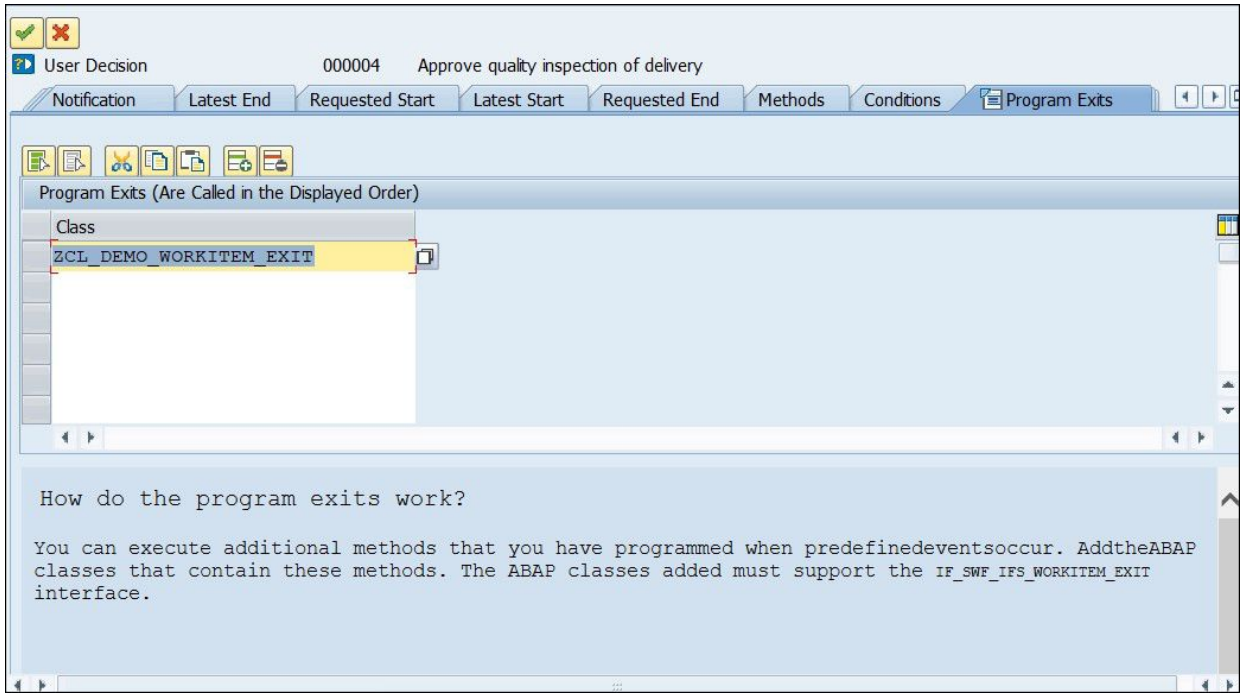


Figure 9.25 Maintenance of a Program Exit at the Workflow Step Level

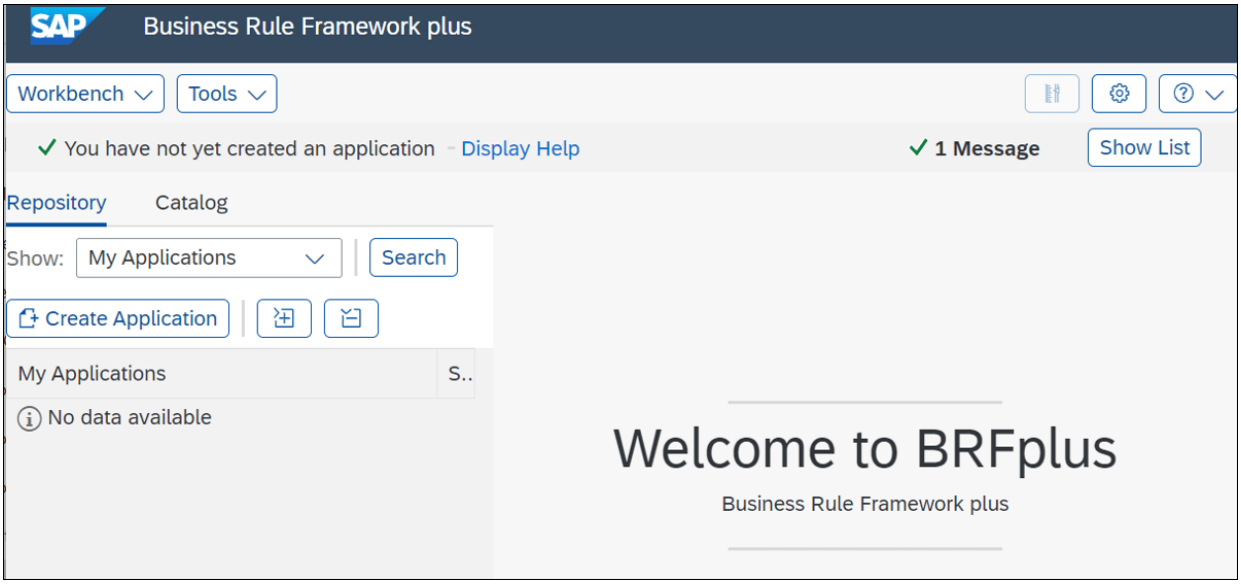


Figure 10.1 BRFplus Landing Page

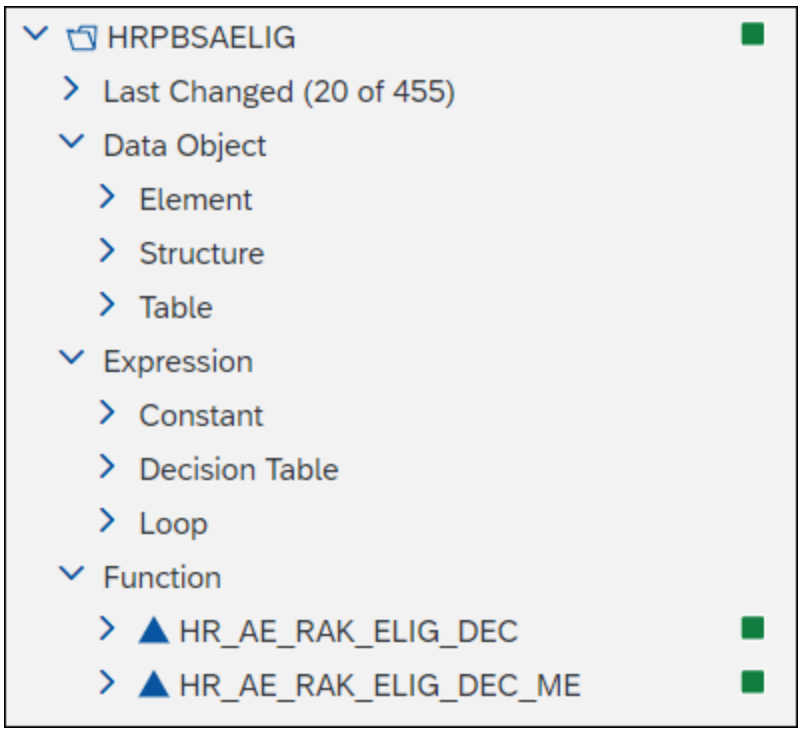


Figure 10.2 Typical BRFplus Application

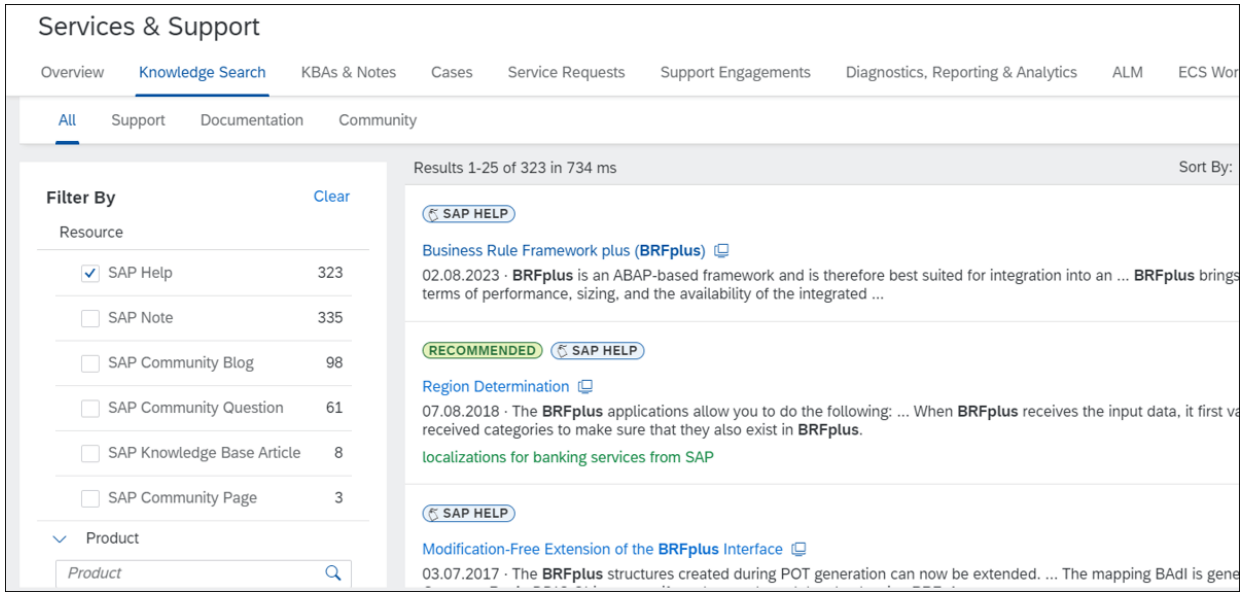


Figure 10.3 SAP Support Portal: Search Results Page

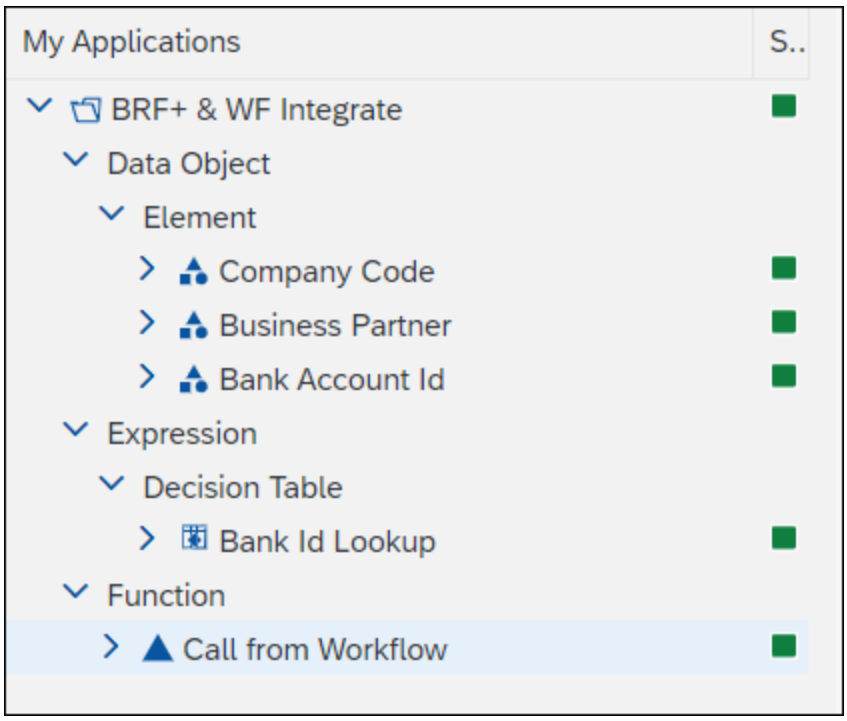


Figure 10.4 BRFplus Demo Scenario Application

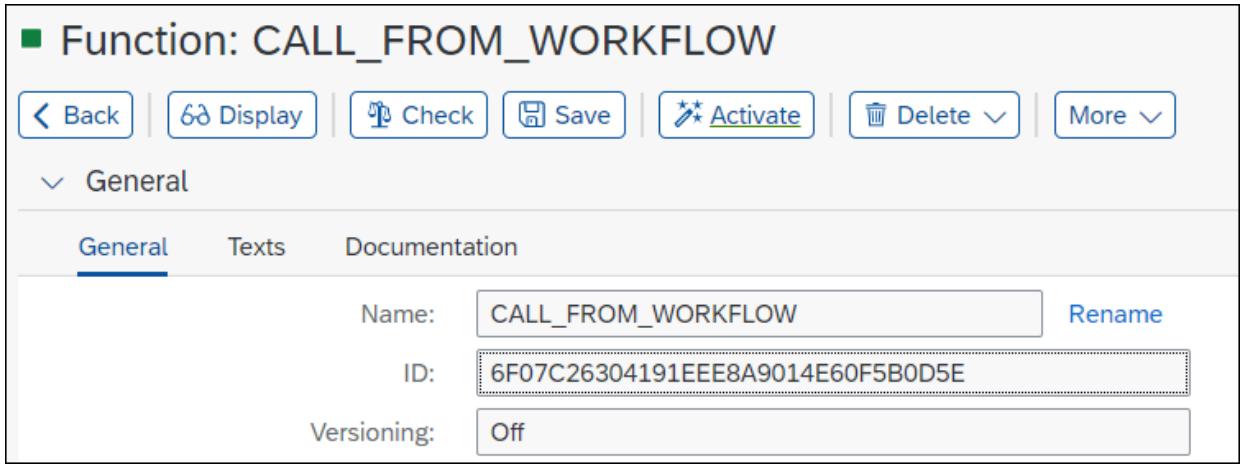


Figure 10.5 BRFplus Function ID

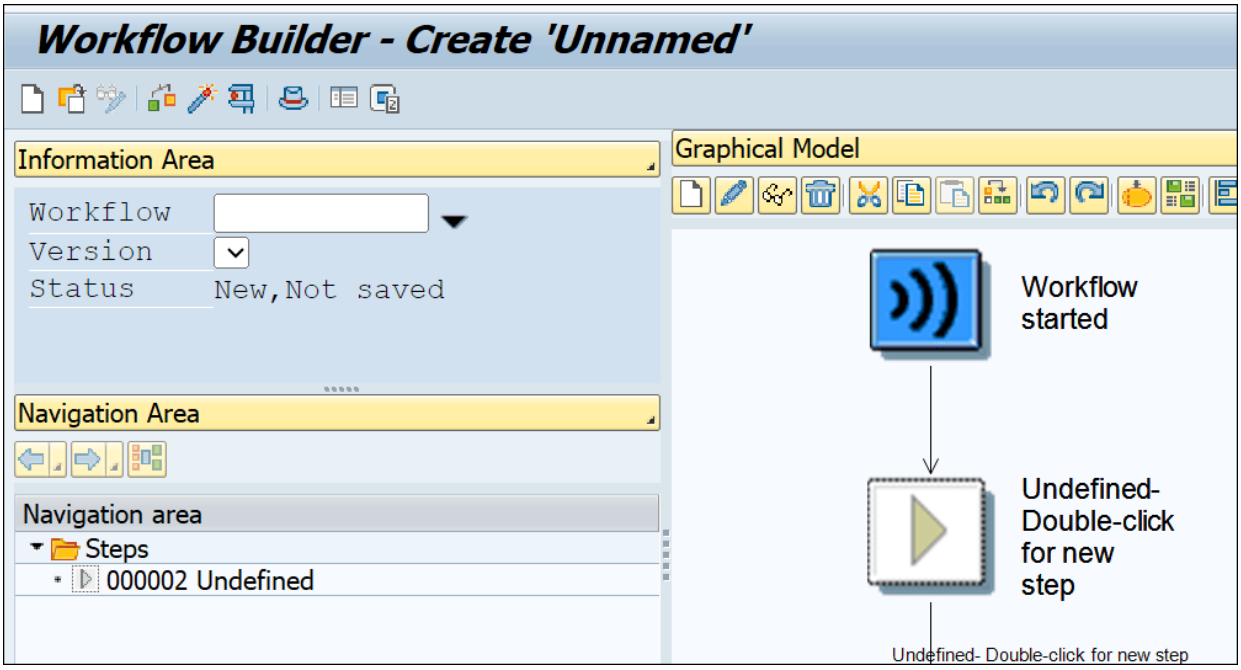


Figure 10.6 Transaction SWDD View

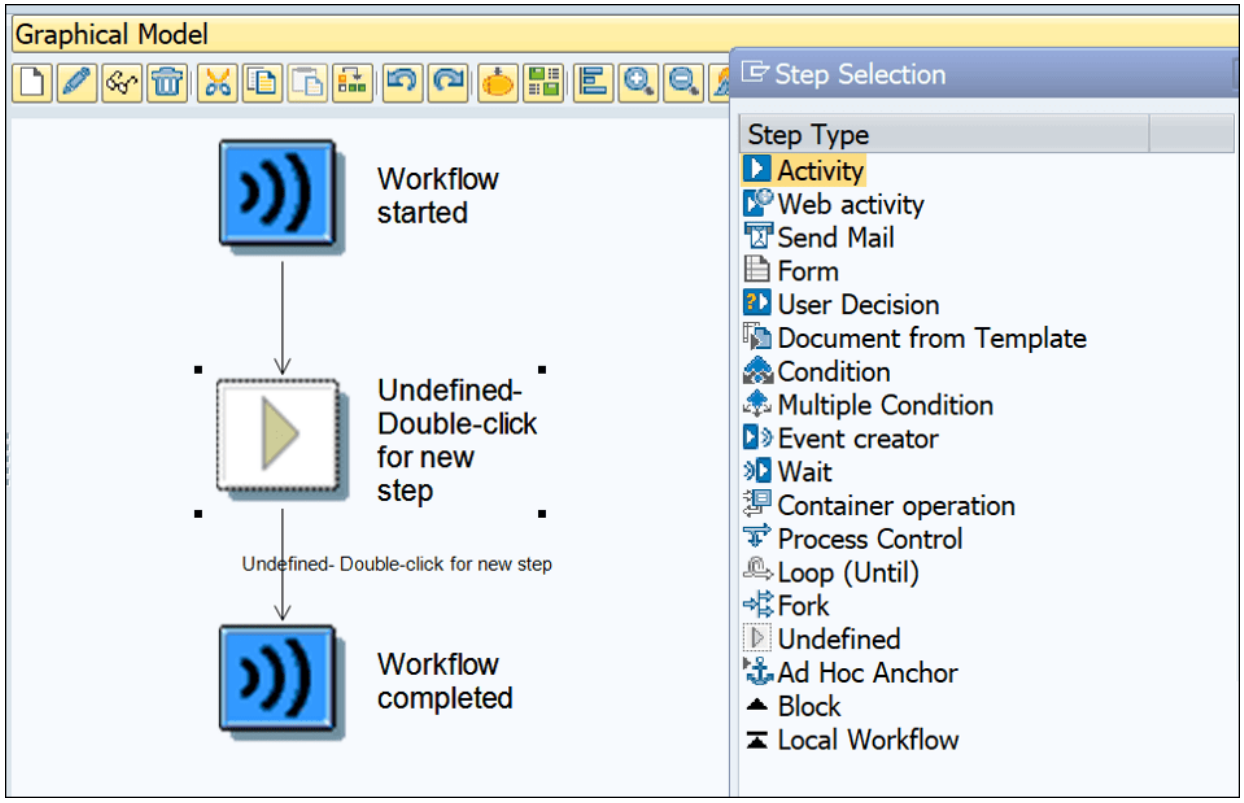


Figure 10.7 Create New Task for the Activity Type

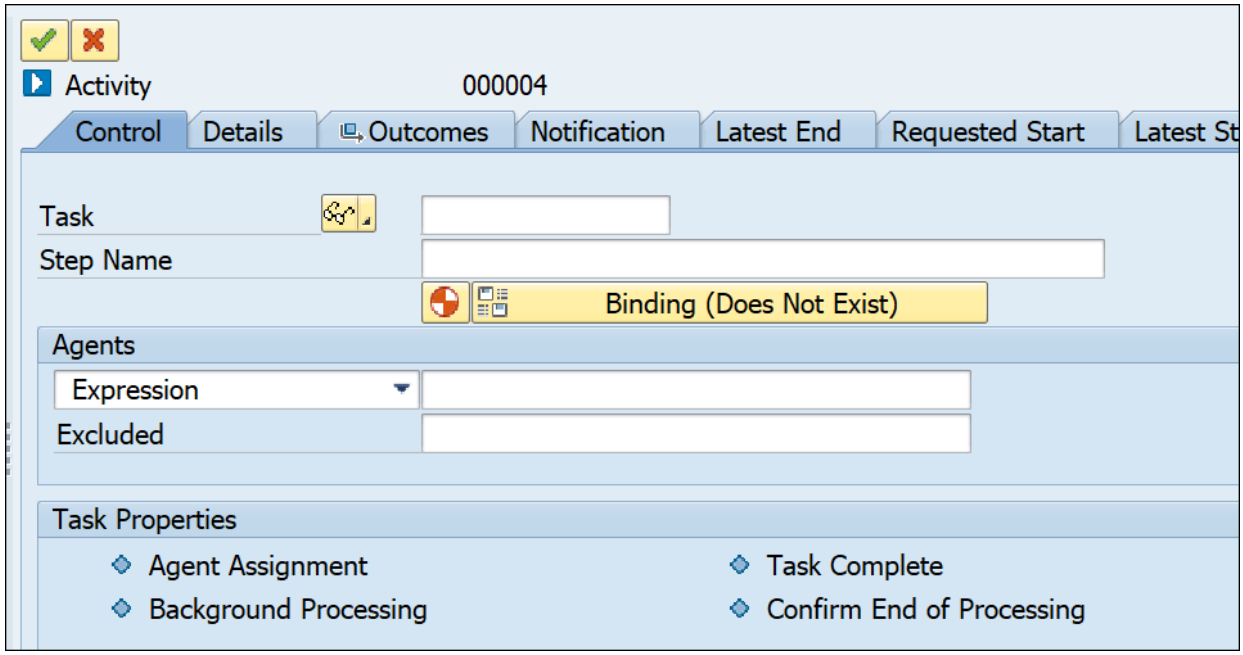


Figure 10.8 Activity Step

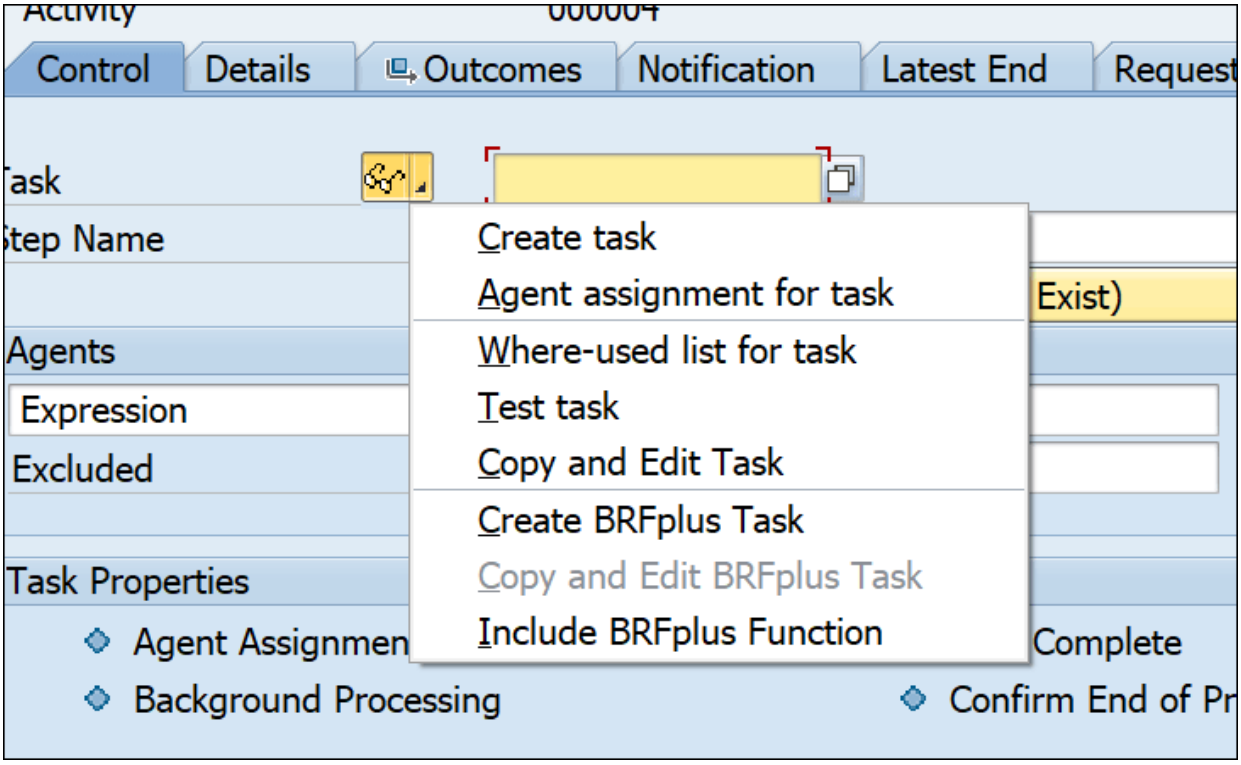


Figure 10.9 Include BRFplus Function

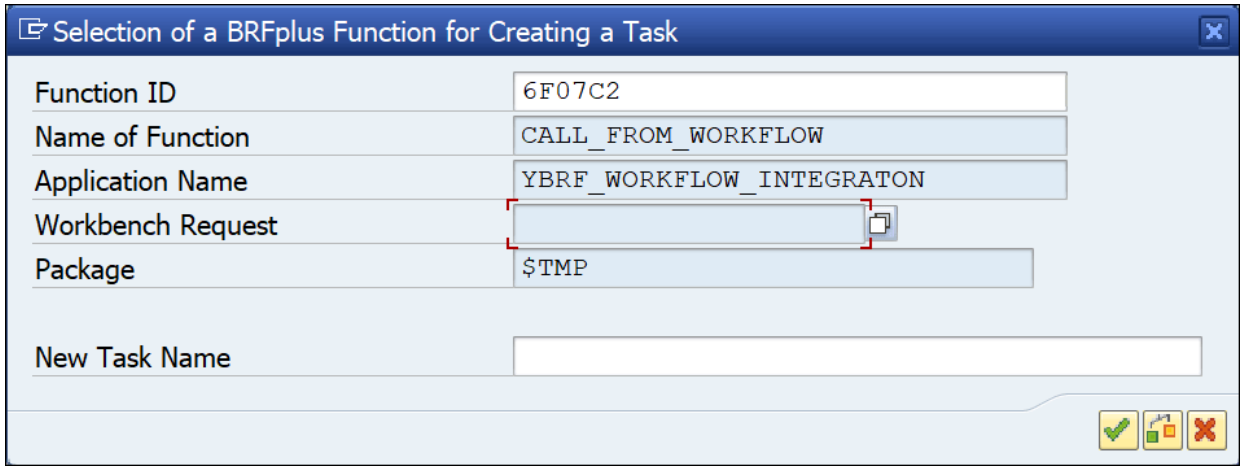


Figure 10.10 Include BRFplus Function ID

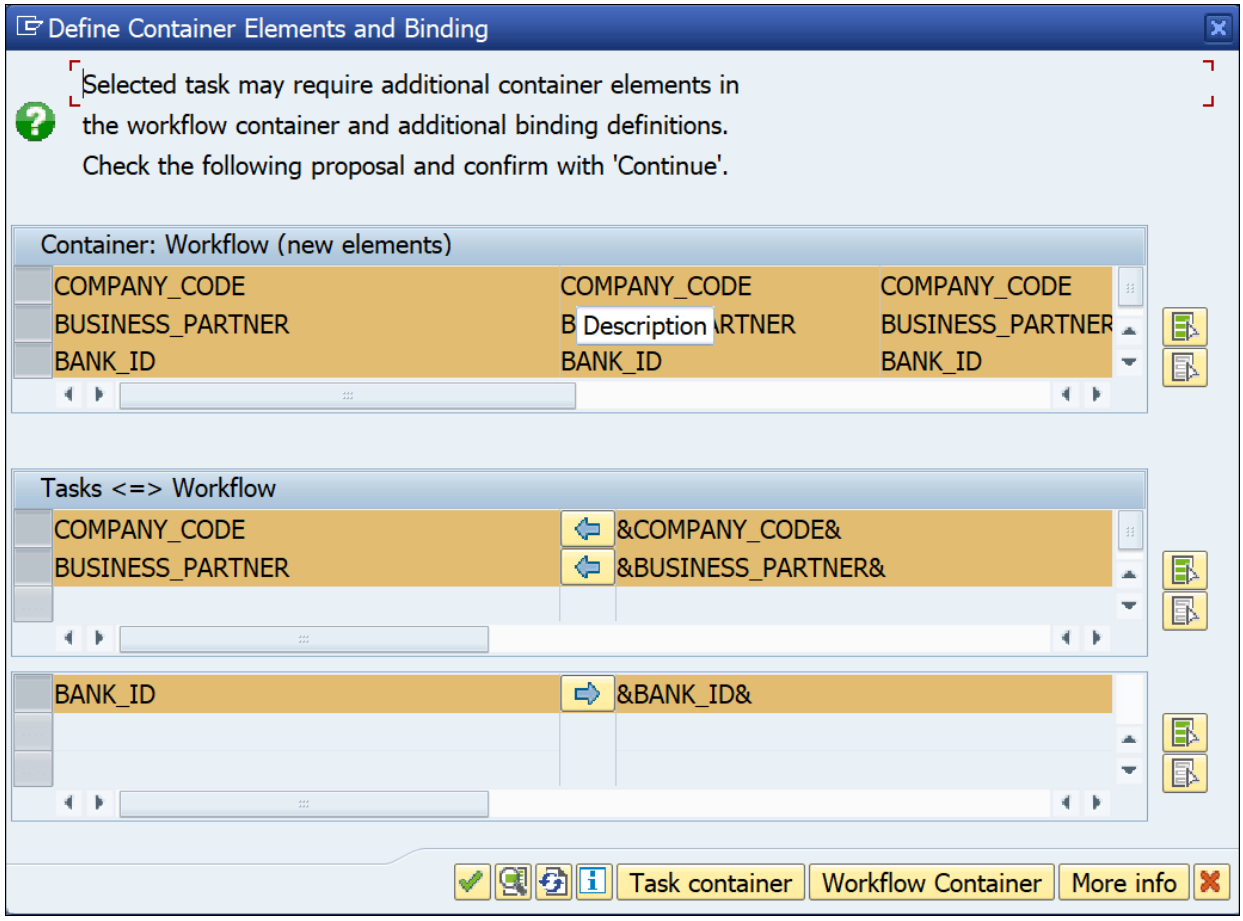


Figure 10.11 Mapping Proposal between BRFplus Function and Workflow Container Elements

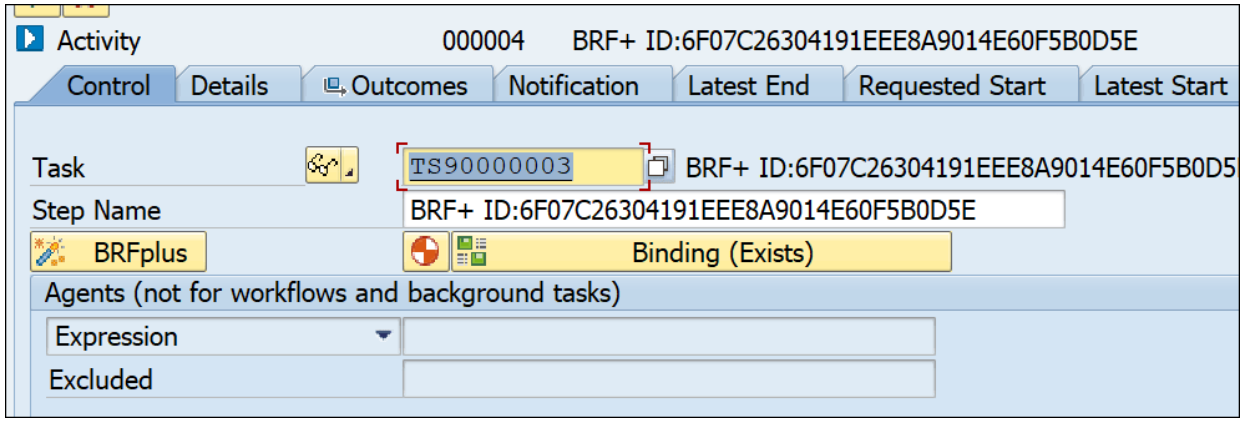


Figure 10.12 Task View after Attaching BRFplus Function

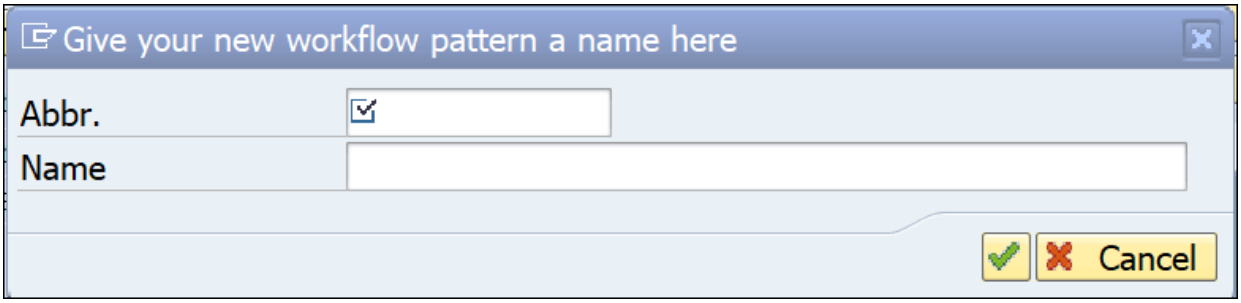


Figure 10.13 Name Your Workflow

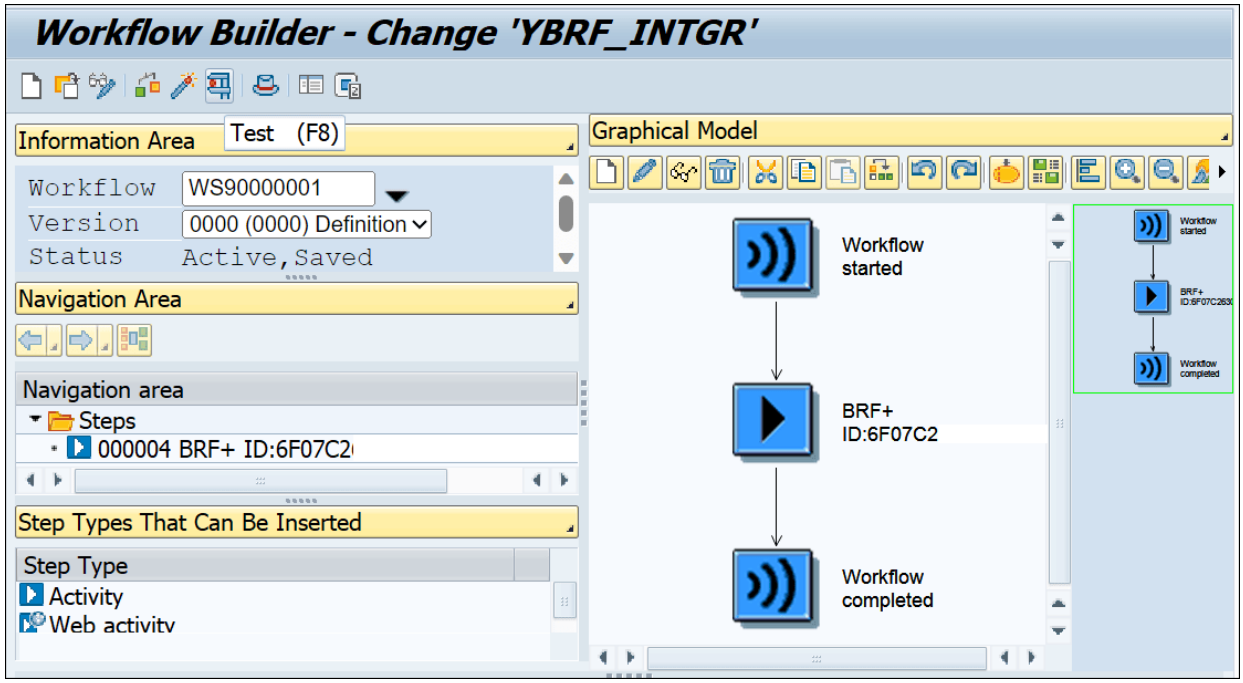
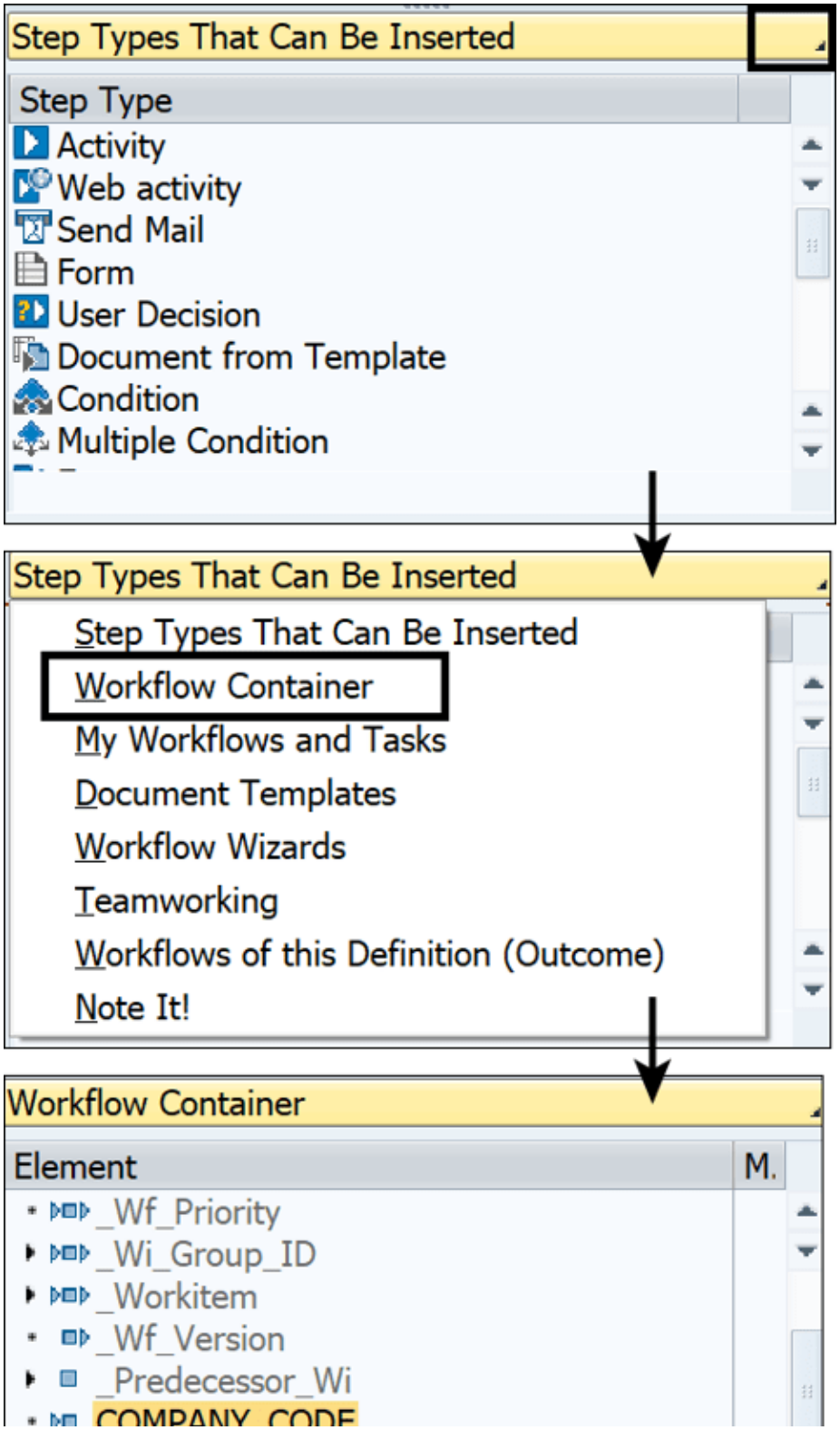


Figure 10.14 Workflow Builder View after Activation


Step Types That Can Be Inserted
Step Type
Activity
Web activity
Send Mail
Form
User Decision
Document from Template
Condition
Multiple Condition
Step Types That Can Be Inserted
Step Types That Can Be Inserted
Workflow Container
My Workflows and Tasks
Document Templates
Workflow Wizards
Teamworking
Workflows of this Definition (Outcome)
Note It!
Workflow Container
Element
M.
_Wf_Priority
_Wi_Group_ID
_Workitem
_Wf_Version
_Predecessor_Wi
COMPANY_CODE

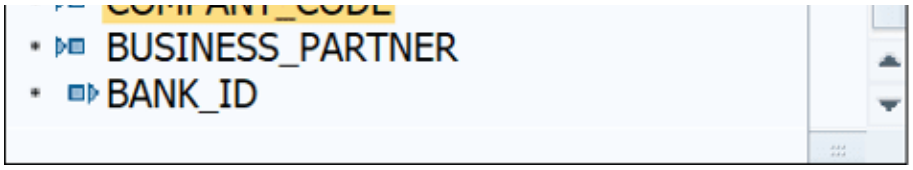


Figure 10.15 Steps to Get the Workflow Container View

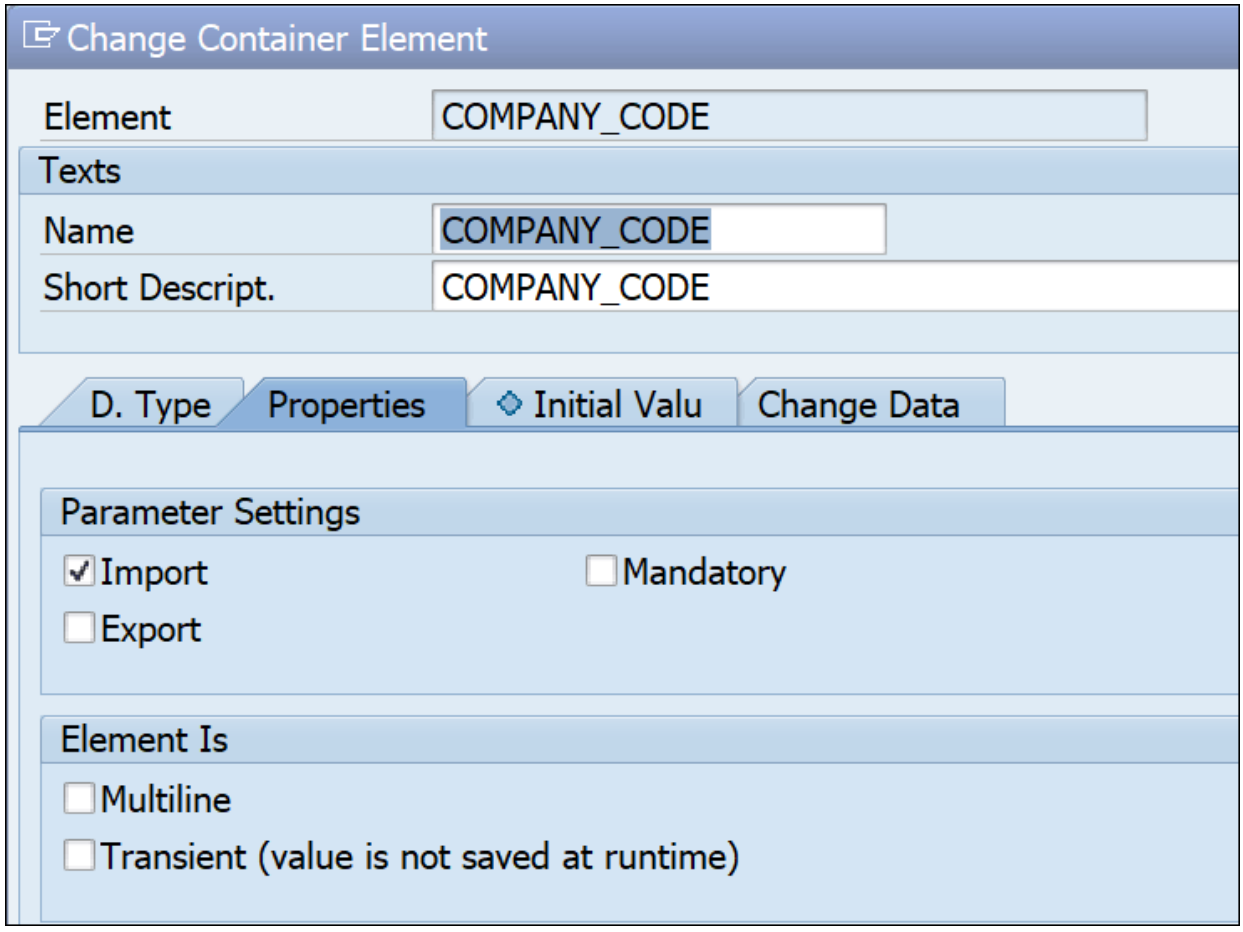


Figure 10.16 Change Container Element View

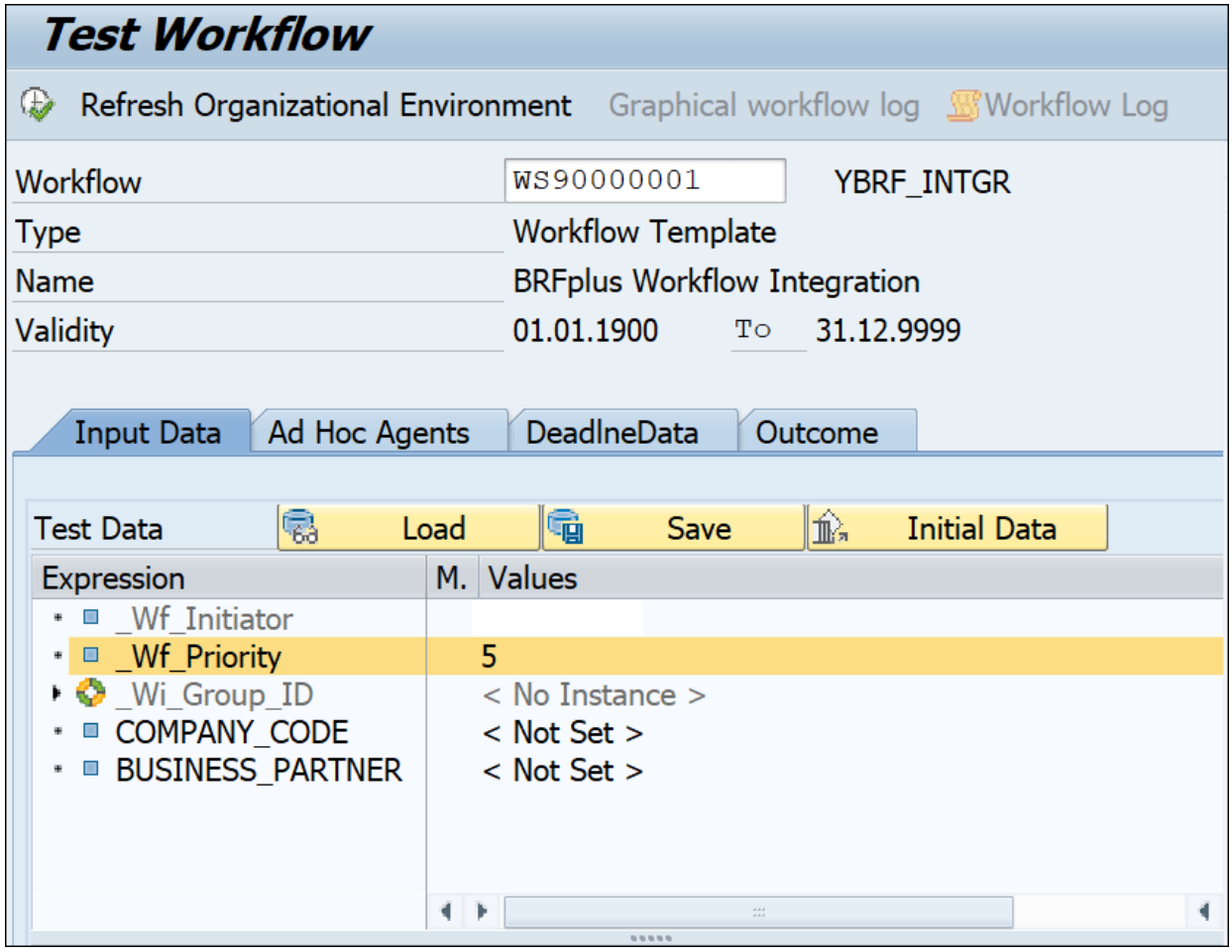


Figure 10.17 Test Workflow View

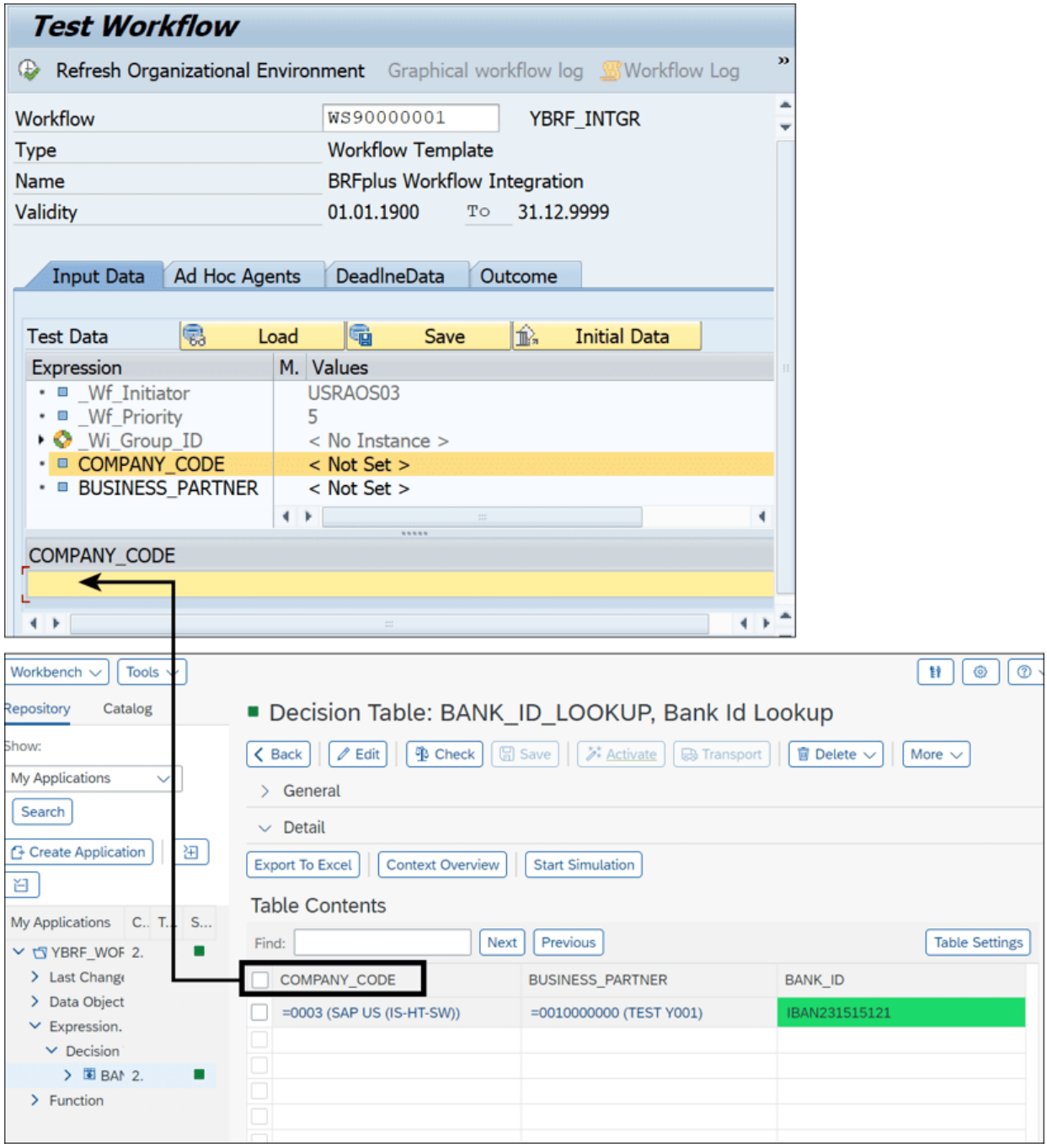


Figure 10.18 Mapping Values from the BRFplus Decision Table

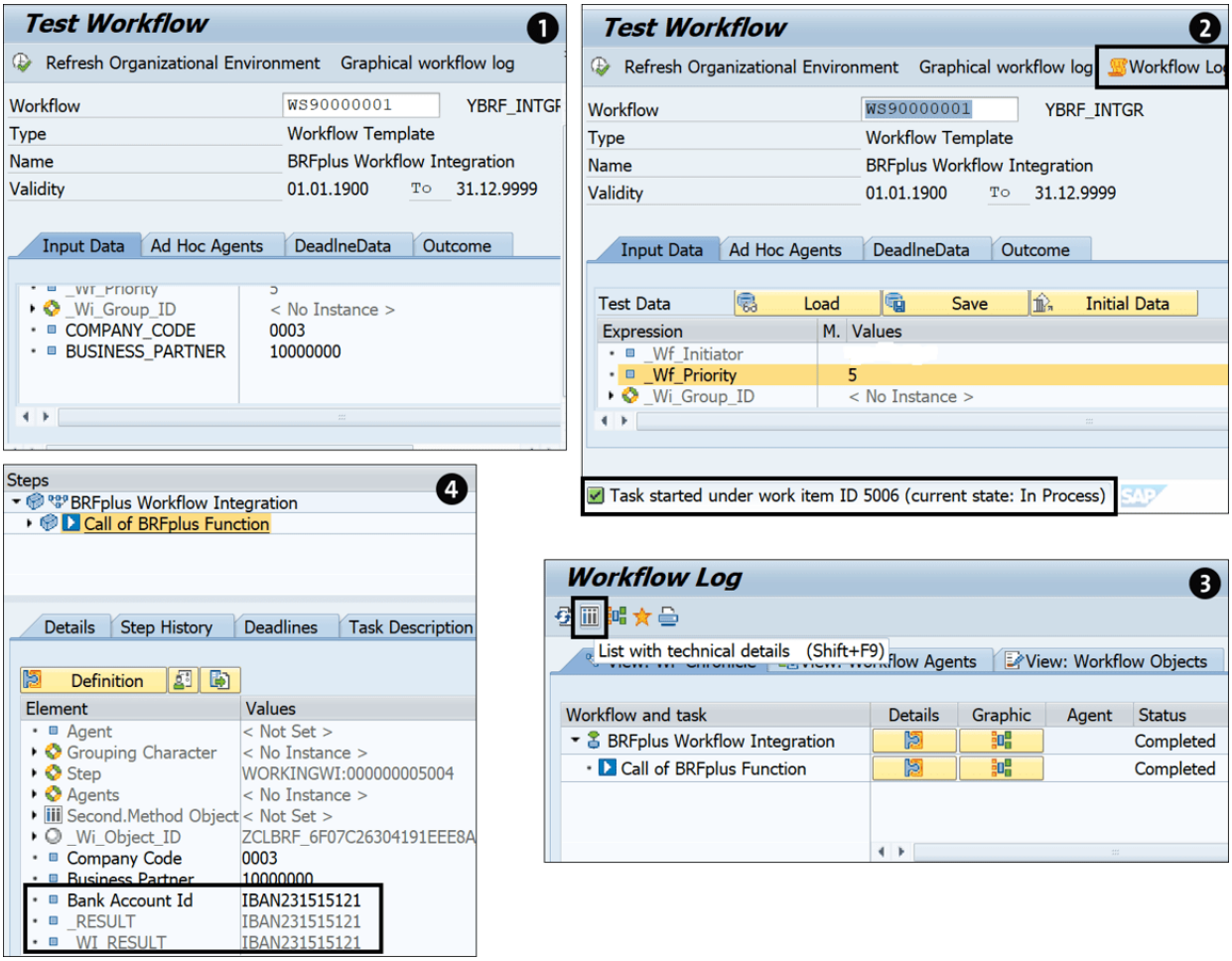


Figure 10.19 Sequence of Steps to Execute and Validate the Result

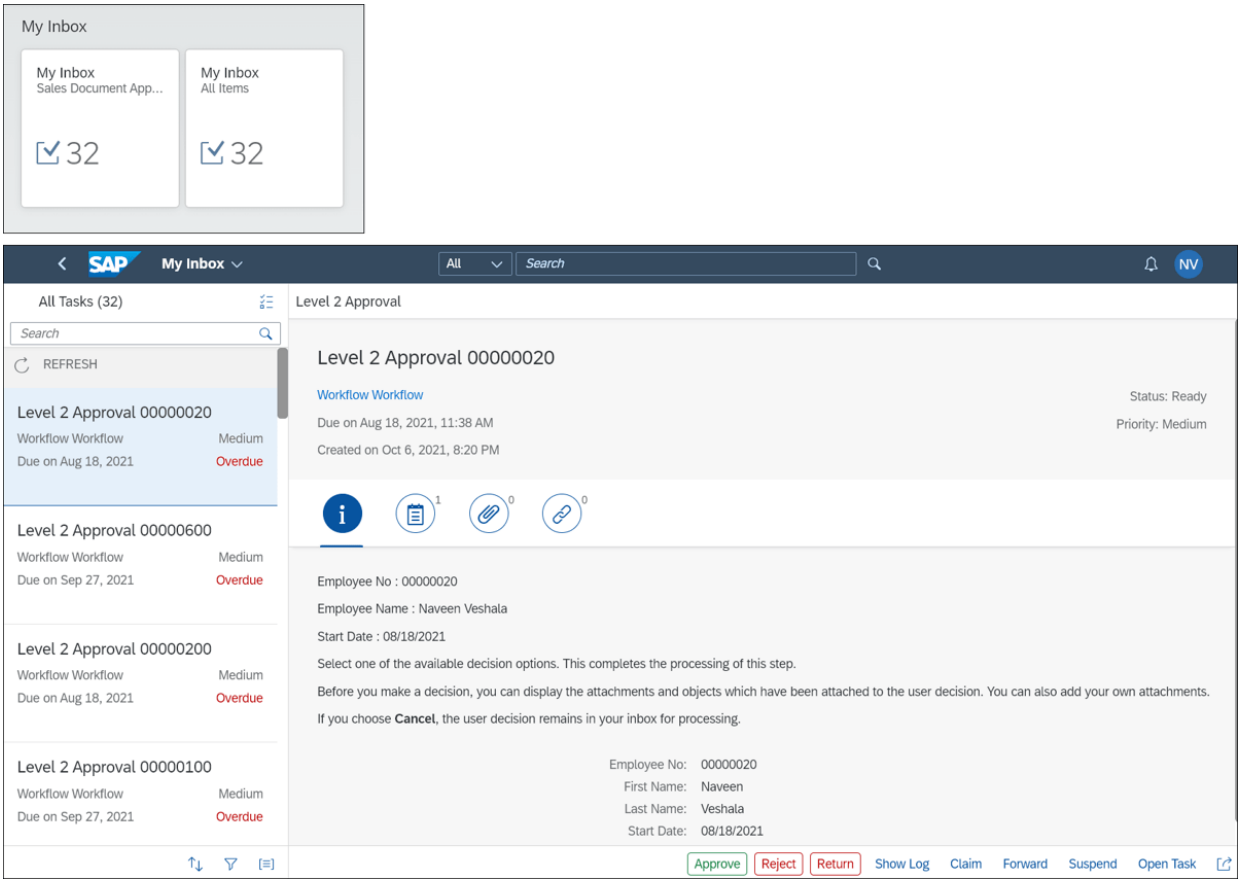


Figure 11.1 My Inbox App

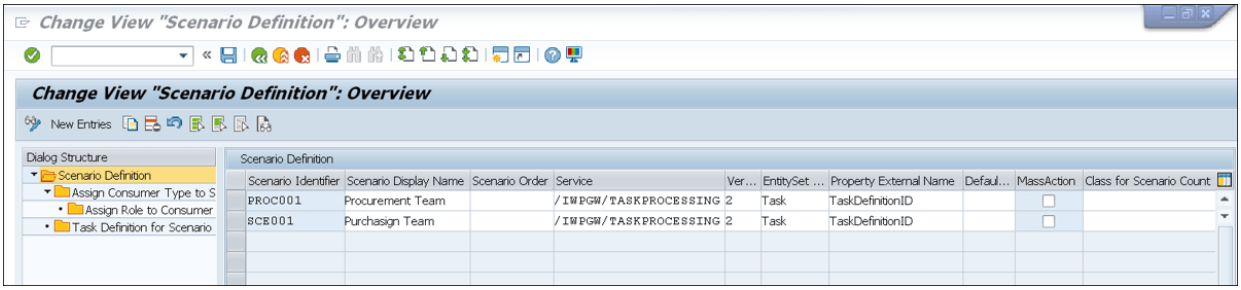


Figure 11.2 Scenario Definition

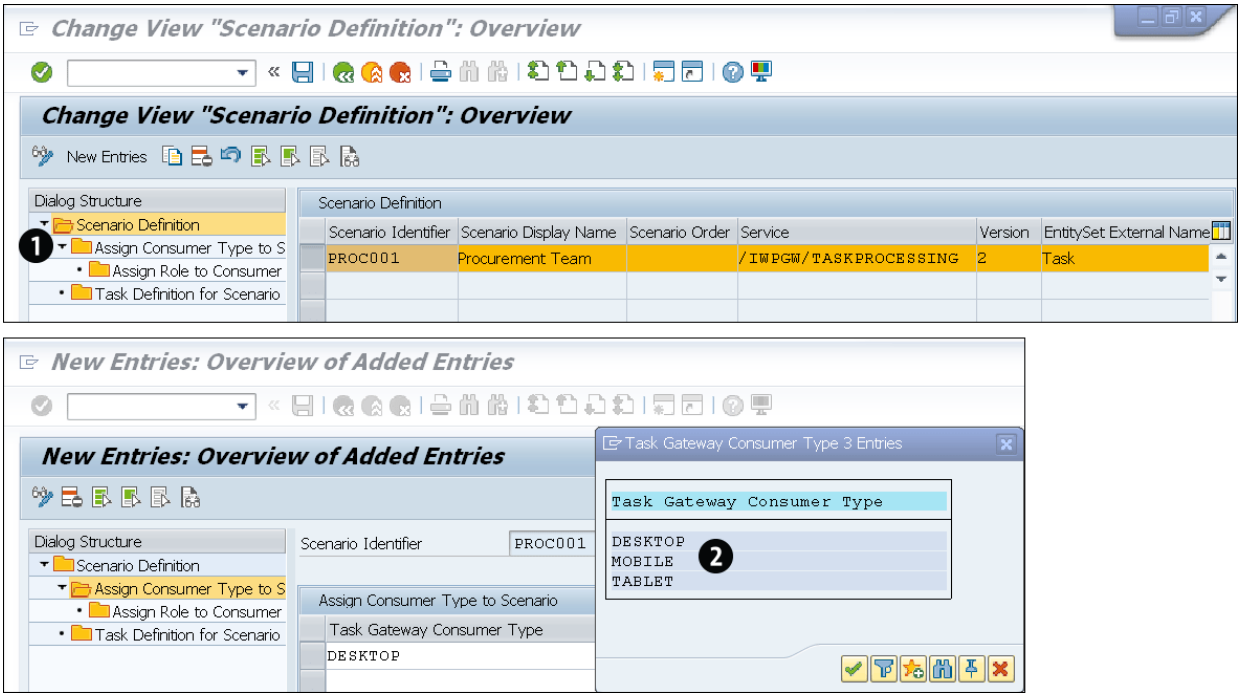


Figure 11.3 My Inbox Scenario ID Configuration

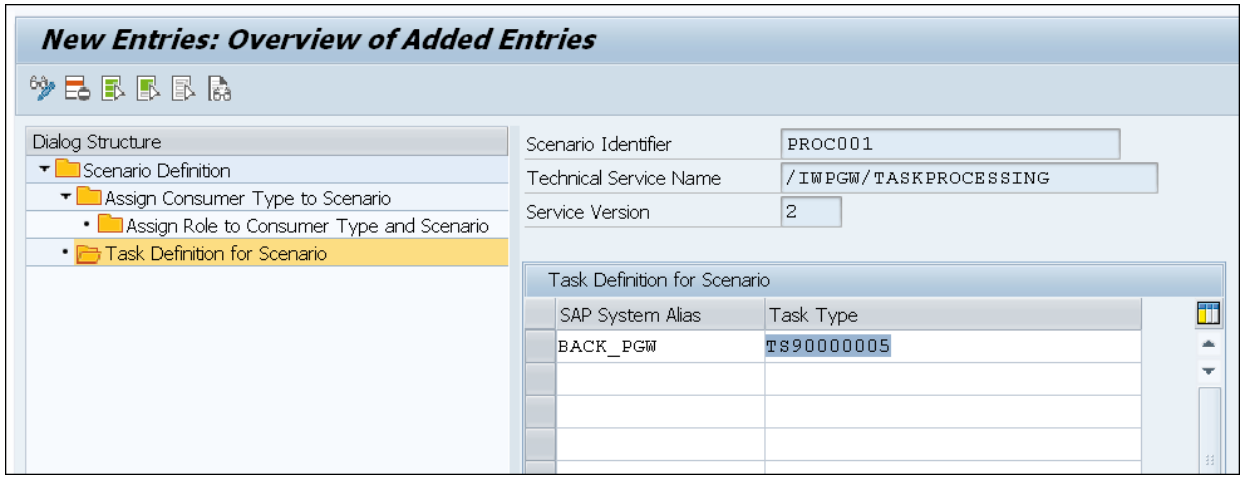


Figure 11.4 Assigning Task IDs to a Scenario ID

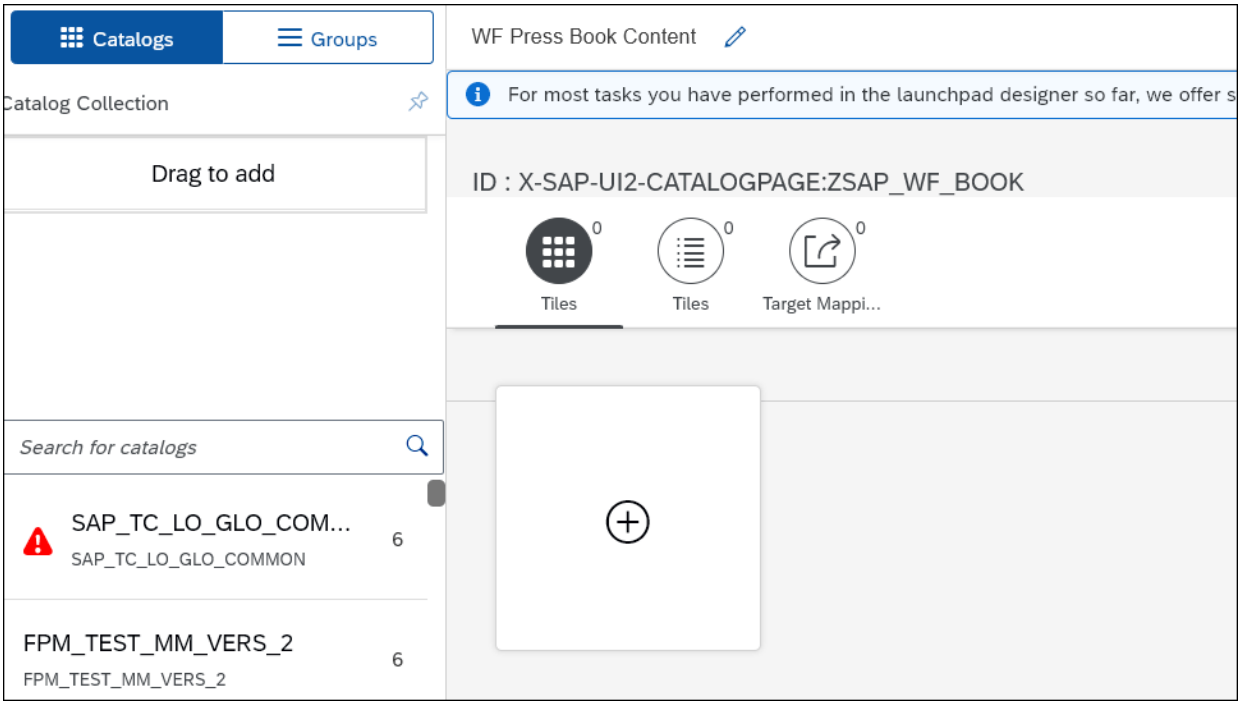


Figure 11.5 Creating the SAP Fiori Scenario Tile

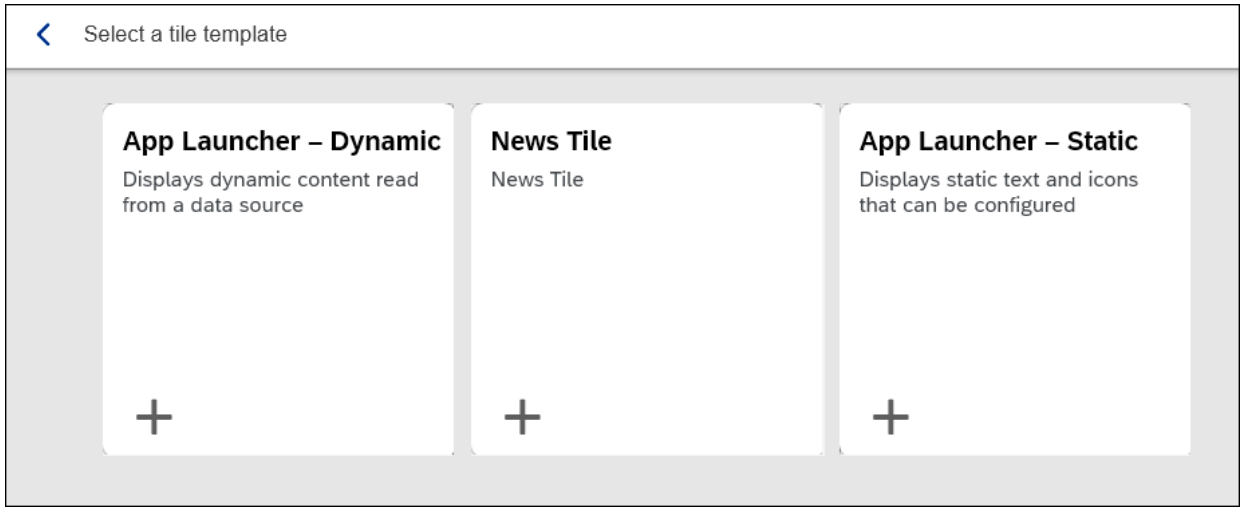


Figure 11.6 Scenario Dynamic Tile

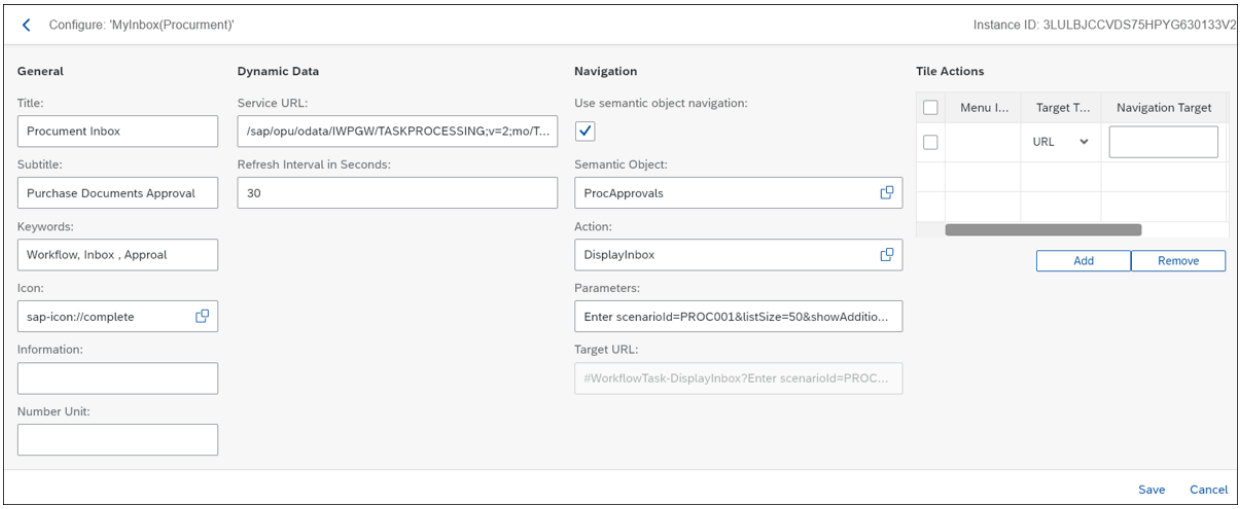


Figure 11.7 Configuring a Scenario-Specific Tile

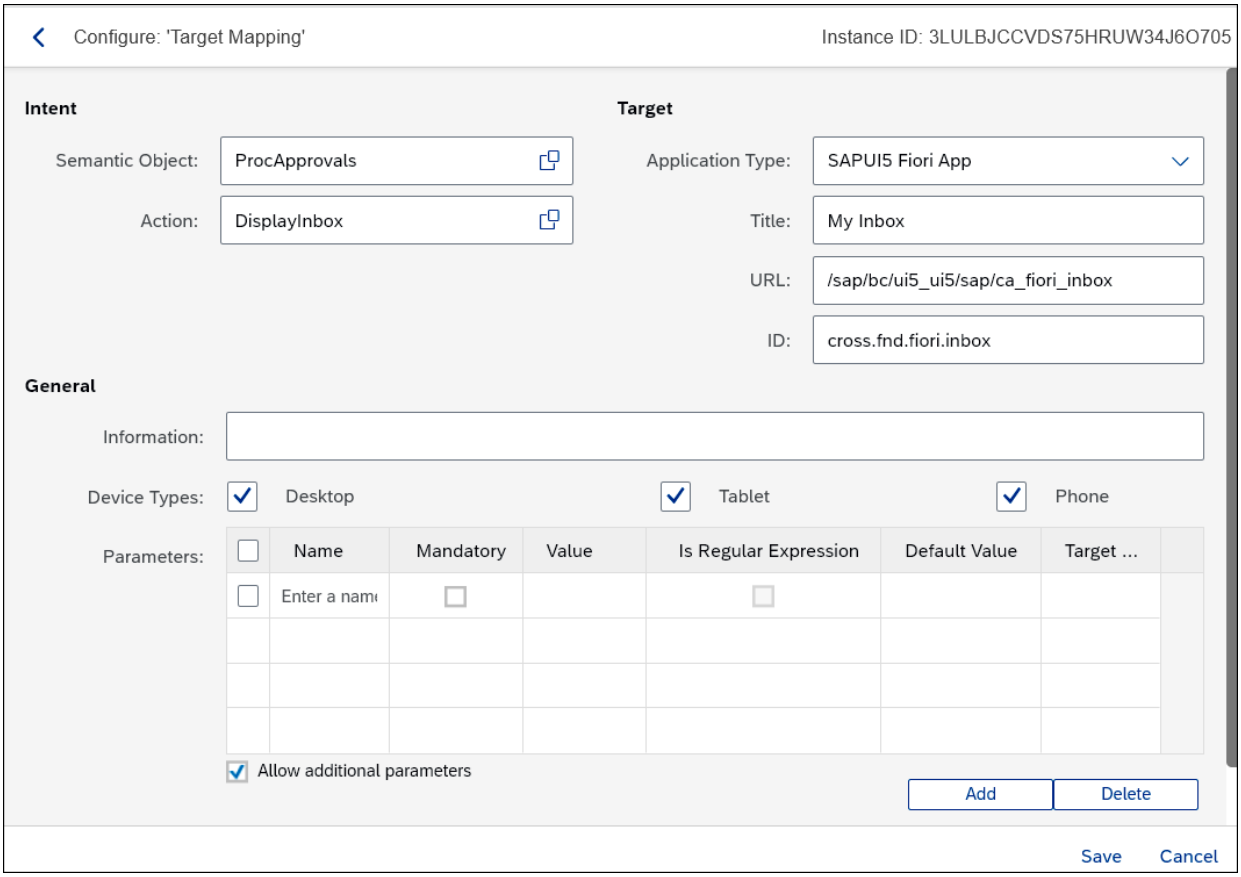


Figure 11.8 Target Mapping

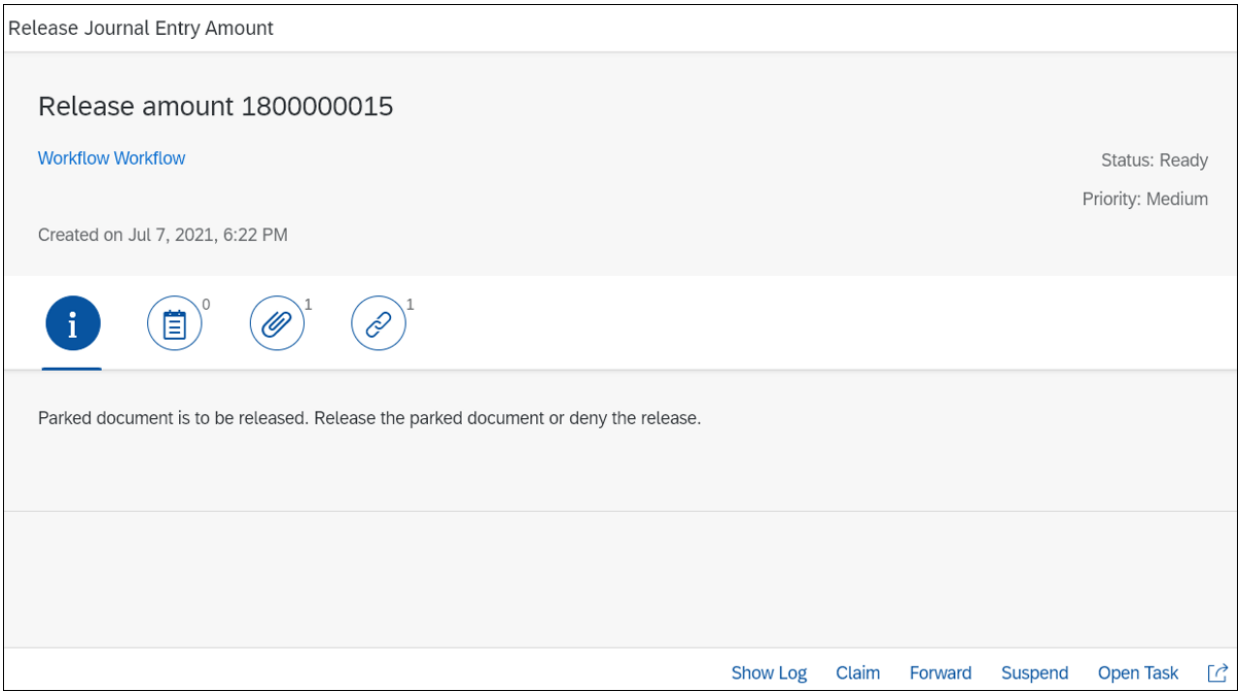


Figure 11.9 Additional Details

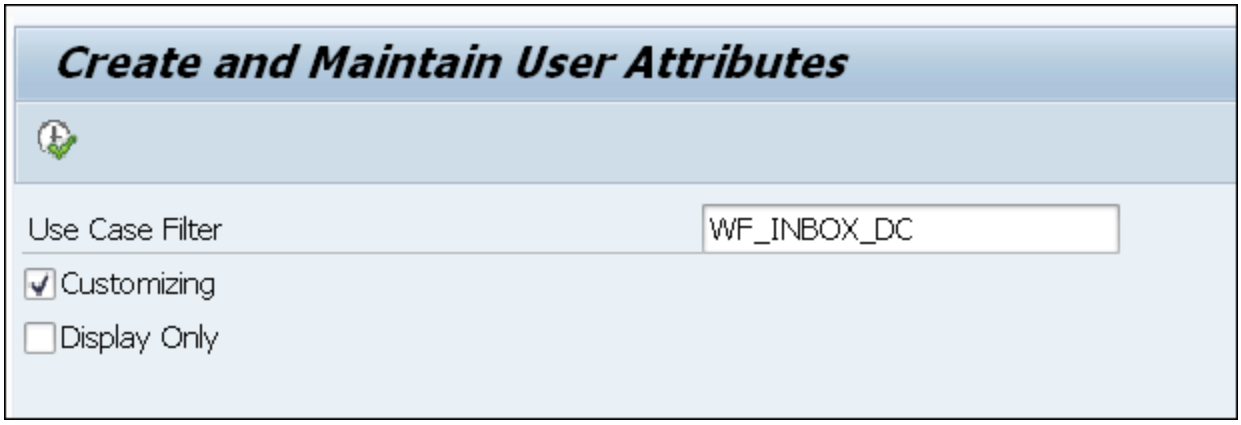
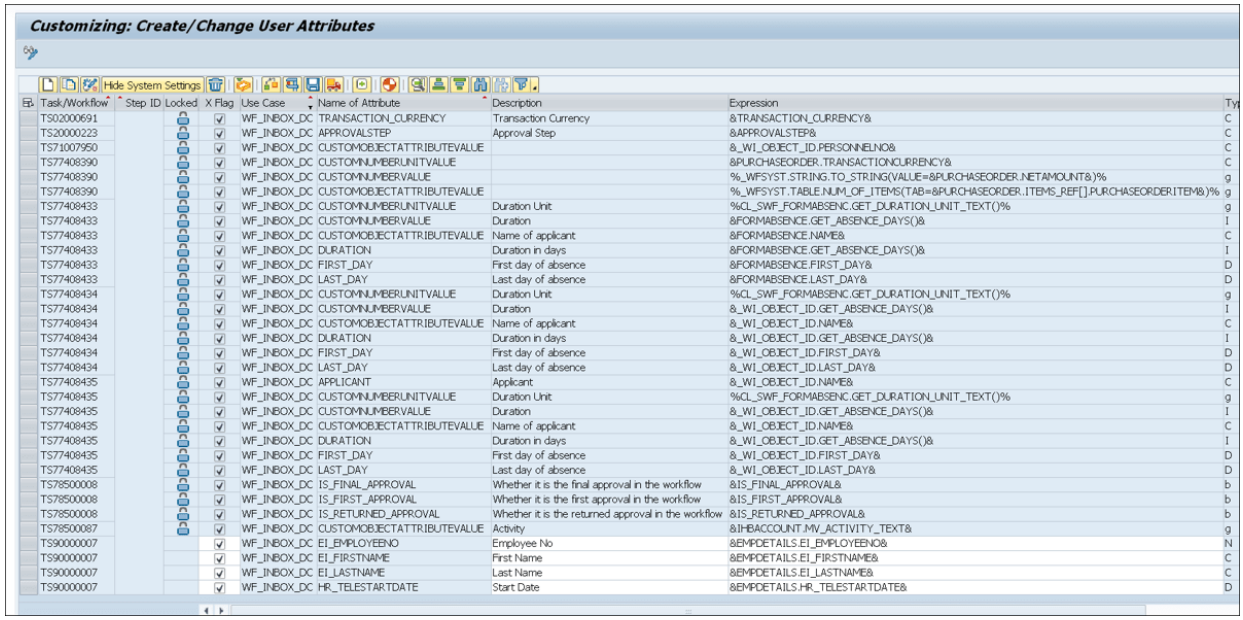


Figure 11.10 Transaction SWF_USER_ATTR



Task/Workflow	Step ID	Locked	X Flag	Use Case	Name of Attribute	Description	Expression	Typ
TS02000691			☑	WF_INBOX_DC	TRANSACTION_CURRENCY	Transaction Currency	&TRANSACTION_CURRENCY&	C
TS20000223			☑	WF_INBOX_DC	APPROVALSTEP	Approval Step	&APPROVALSTEP&	C
TS71007950			☑	WF_INBOX_DC	CUSTOMOBJECTATTRIBUTEVALUE		&_WI_OBJECT_ID.PERSONNELNO&	C
TS77408390			☑	WF_INBOX_DC	CUSTOMNUMBERUNITVALUE		&PURCHASEORDER..TRANSACTIONCURRENCY&	C
TS77408390			☑	WF_INBOX_DC	CUSTOMNUMBERVALUE		%_WFSYST.STRING.TO_STRING(VALUE=&PURCHASEORDER.NETAMOUNT&)%	g
TS77408390			☑	WF_INBOX_DC	CUSTOMOBJECTATTRIBUTEVALUE		%_WFSYST.TABLE.NUM_OF_ITEMS(TAB=&PURCHASEORDER..ITEMS_REF[].PURCHASEORDER.ITEM&)%	g
TS77408433			☑	WF_INBOX_DC	CUSTOMNUMBERUNITVALUE	Duration Unit	%CL_SWF_FORMABSENC.GET_DURATION_UNIT_TEXT()%	g
TS77408433			☑	WF_INBOX_DC	CUSTOMNUMBERVALUE	Duration	&FORMABSENCE.GET_ABSENCE_DAYS()&	I
TS77408433			☑	WF_INBOX_DC	CUSTOMOBJECTATTRIBUTEVALUE	Name of applicant	&FORMABSENCE.NAME&	C
TS77408433			☑	WF_INBOX_DC	DURATION	Duration in days	&FORMABSENCE.GET_ABSENCE_DAYS()&	I
TS77408433			☑	WF_INBOX_DC	FIRST_DAY	First day of absence	&FORMABSENCE.FIRST_DAY&	D
TS77408433			☑	WF_INBOX_DC	LAST_DAY	Last day of absence	&FORMABSENCE.LAST_DAY&	D
TS77408434			☑	WF_INBOX_DC	CUSTOMNUMBERUNITVALUE	Duration Unit	%CL_SWF_FORMABSENC.GET_DURATION_UNIT_TEXT()%	g
TS77408434			☑	WF_INBOX_DC	CUSTOMNUMBERVALUE	Duration	&_WI_OBJECT_ID.GET_ABSENCE_DAYS()&	I
TS77408434			☑	WF_INBOX_DC	CUSTOMOBJECTATTRIBUTEVALUE	Name of applicant	&_WI_OBJECT_ID.NAME&	C
TS77408434			☑	WF_INBOX_DC	DURATION	Duration in days	&_WI_OBJECT_ID.GET_ABSENCE_DAYS()&	I
TS77408434			☑	WF_INBOX_DC	FIRST_DAY	First day of absence	&_WI_OBJECT_ID.FIRST_DAY&	D
TS77408434			☑	WF_INBOX_DC	LAST_DAY	Last day of absence	&_WI_OBJECT_ID.LAST_DAY&	D
TS77408435			☑	WF_INBOX_DC	APPLICANT	Applicant	&_WI_OBJECT_ID.NAME&	C
TS77408435			☑	WF_INBOX_DC	CUSTOMNUMBERUNITVALUE	Duration Unit	%CL_SWF_FORMABSENC.GET_DURATION_UNIT_TEXT()%	g
TS77408435			☑	WF_INBOX_DC	CUSTOMNUMBERVALUE	Duration	&_WI_OBJECT_ID.GET_ABSENCE_DAYS()&	I
TS77408435			☑	WF_INBOX_DC	CUSTOMOBJECTATTRIBUTEVALUE	Name of applicant	&_WI_OBJECT_ID.NAME&	C
TS77408435			☑	WF_INBOX_DC	DURATION	Duration in days	&_WI_OBJECT_ID.GET_ABSENCE_DAYS()&	I
TS77408435			☑	WF_INBOX_DC	FIRST_DAY	First day of absence	&_WI_OBJECT_ID.FIRST_DAY&	D
TS77408435			☑	WF_INBOX_DC	LAST_DAY	Last day of absence	&_WI_OBJECT_ID.LAST_DAY&	D
TS78500008			☑	WF_INBOX_DC	IS_FINAL_APPROVAL	Whether it is the final approval in the workflow	&IS_FINAL_APPROVAL&	b
TS78500008			☑	WF_INBOX_DC	IS_FIRST_APPROVAL	Whether it is the first approval in the workflow	&IS_FIRST_APPROVAL&	b
TS78500008			☑	WF_INBOX_DC	IS_RETURNED_APPROVAL	Whether it is the returned approval in the workflow	&IS_RETURNED_APPROVAL&	b
TS78500087			☑	WF_INBOX_DC	CUSTOMOBJECTATTRIBUTEVALUE	Activity	&IHBACCOUNT.MV_ACTIVITY_TEXT&	g
TS90000007			☑	WF_INBOX_DC	EI_EMPLOYEENO	Employee No	&EMPDETAILS.EI_EMPLOYEENO&	N
TS90000007			☑	WF_INBOX_DC	EI_FIRSTNAME	First Name	&EMPDETAILS.EI_FIRSTNAME&	C
TS90000007			☑	WF_INBOX_DC	EI_LASTNAME	Last Name	&EMPDETAILS.EI_LASTNAME&	C
TS90000007			☑	WF_INBOX_DC	HR_TELESTARTDATE	Start Date	&EMPDETAILS.HR_TELESTARTDATE&	D

Figure 11.11 User Attributes Task Details

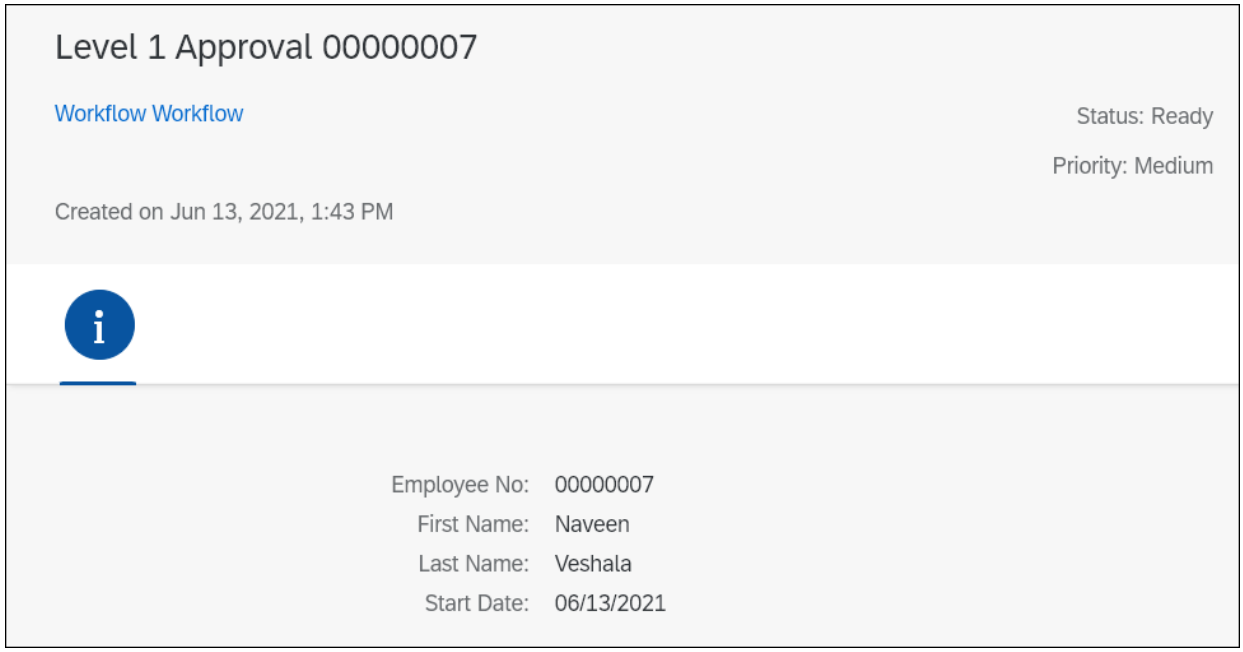


Figure 11.12 My Inbox: Additional Details

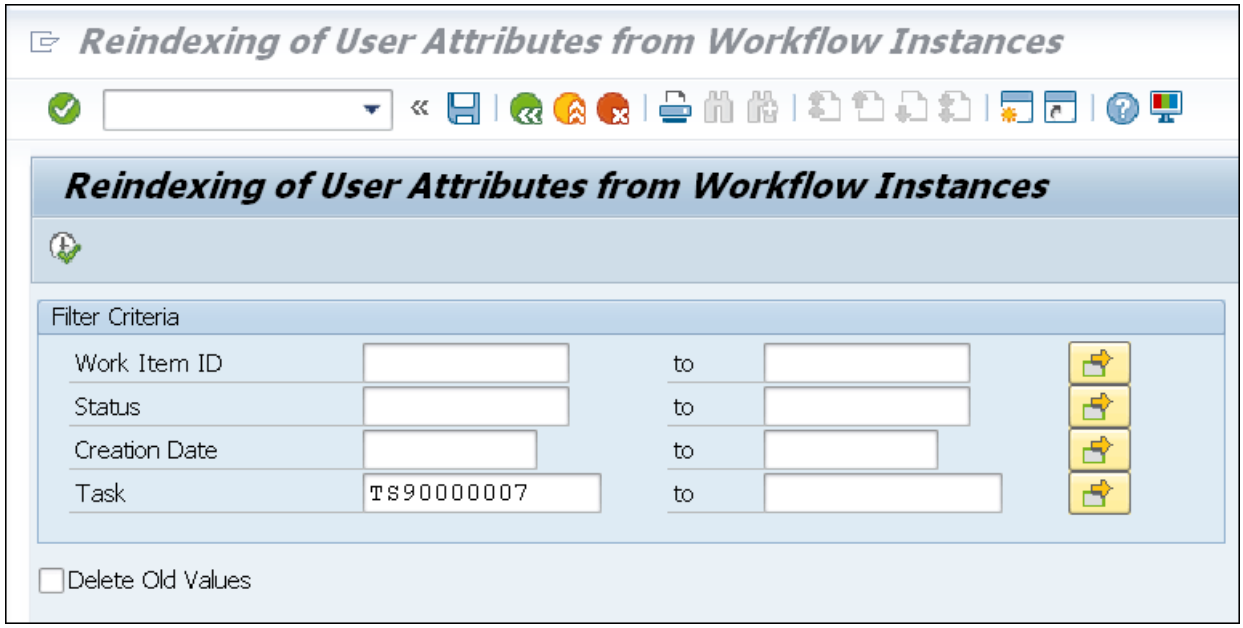


Figure 11.13 Reindexing User Attributes for Old Work Items

Figure 11.14 Updated User Attributes

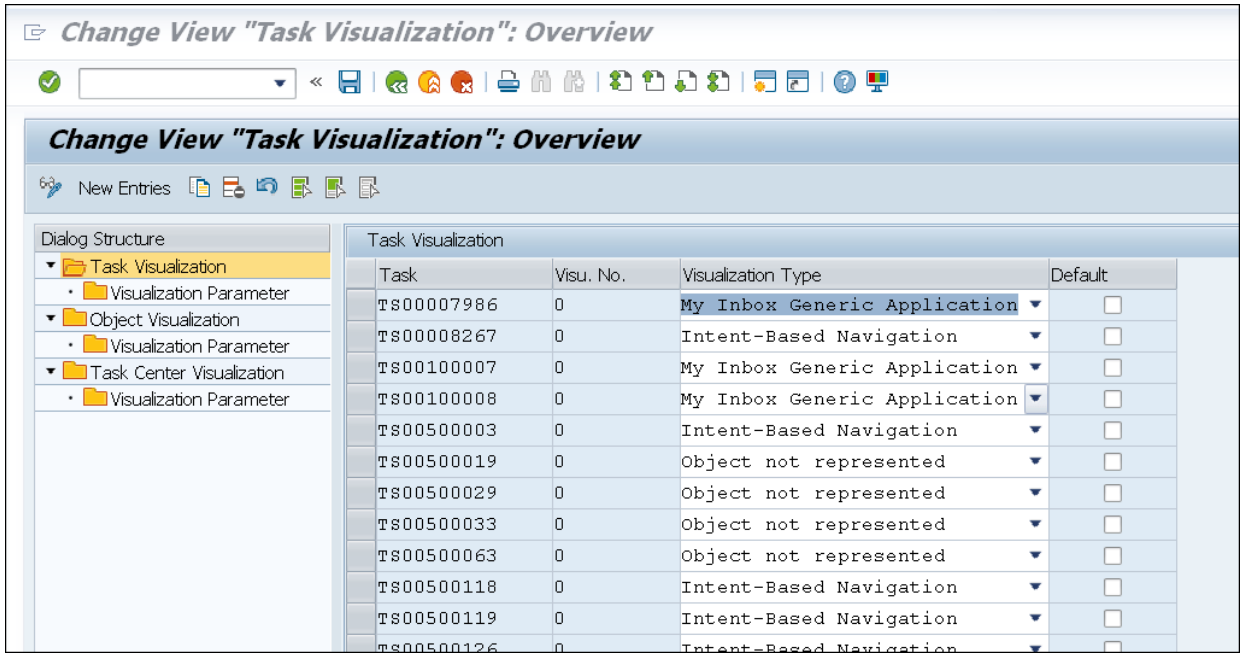


Figure 11.15 Transaction SWFVISU

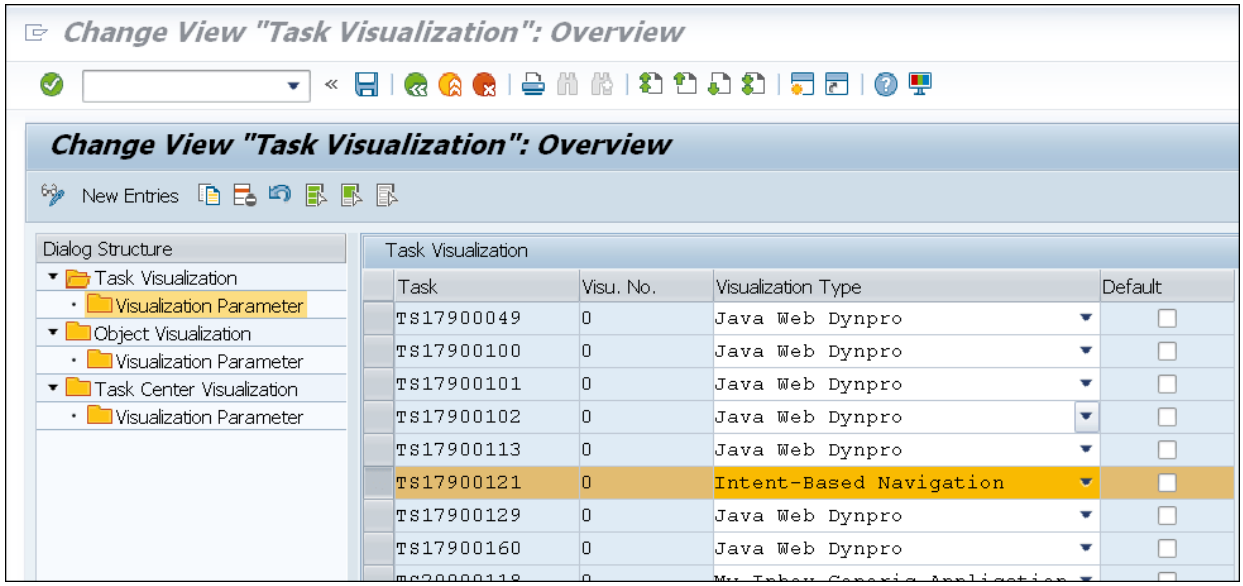


Figure 11.16 Visualization Parameter

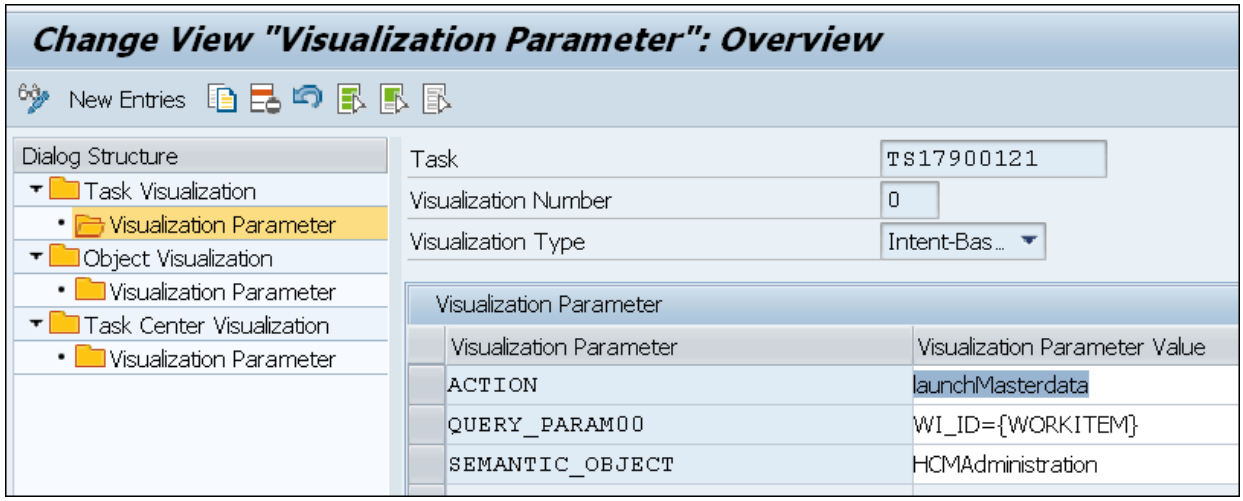


Figure 11.17 Visualization Parameters: Intent-Based Navigation

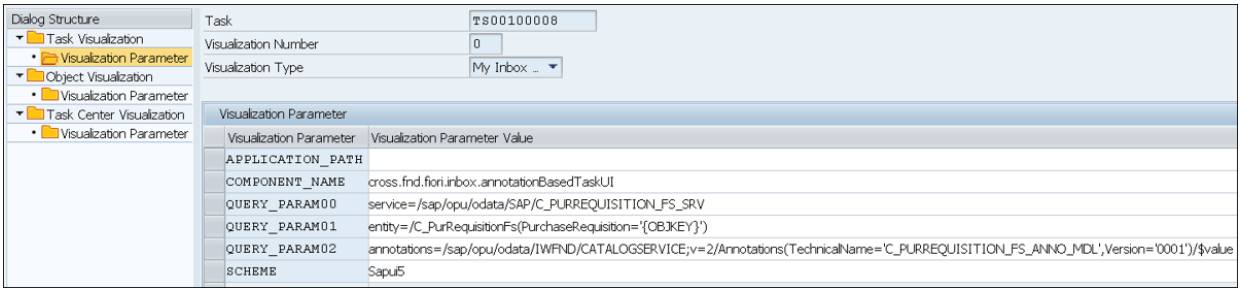


Figure 11.18 Visualization Parameters: My Inbox

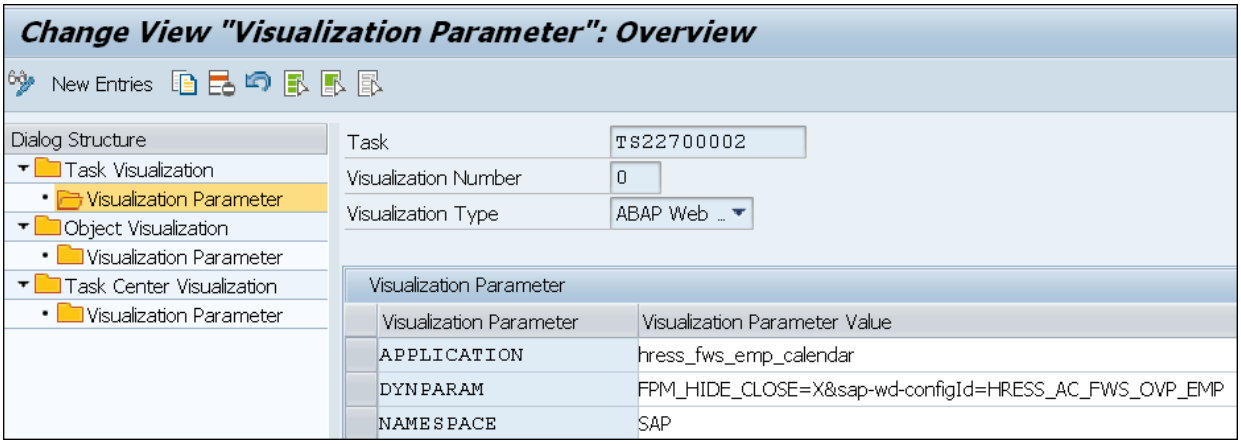


Figure 11.19 Web Dynpro Visualization Parameters

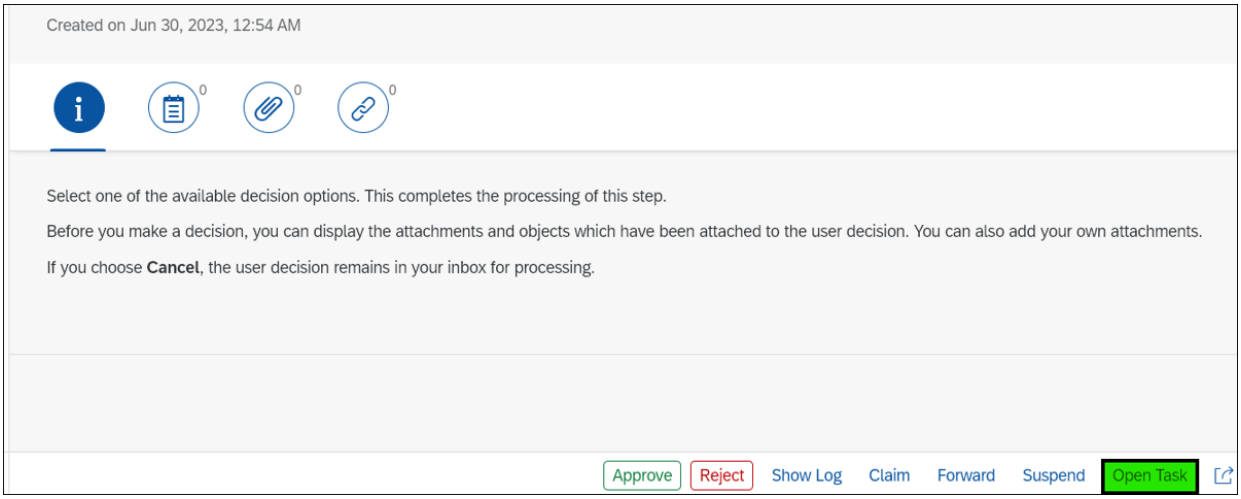


Figure 11.20 My Inbox Open Task

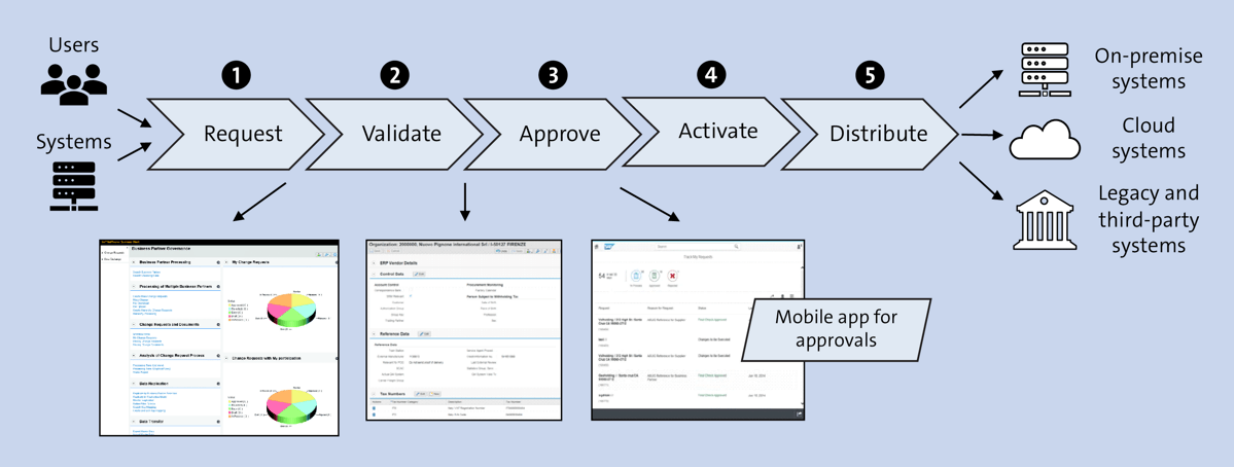


Figure 13.1 Typical Steps in a Change Request Approval Workflow Process in SAP MDG

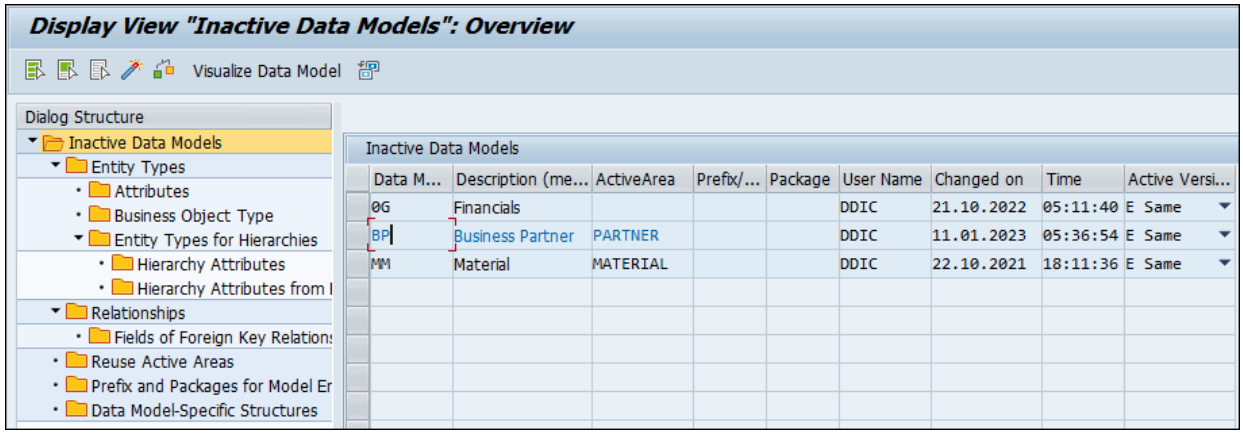


Figure 13.2 Standard Data Models Delivered by SAP

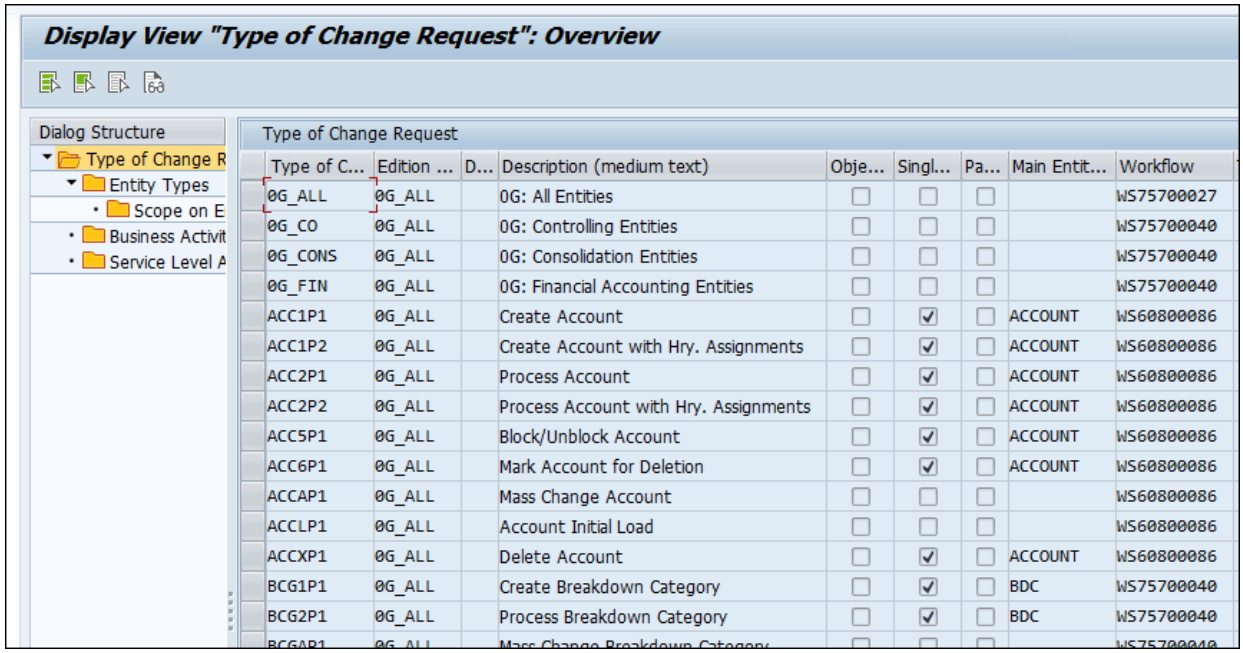


Figure 13.3 Standard Change Request Types Delivered by SAP

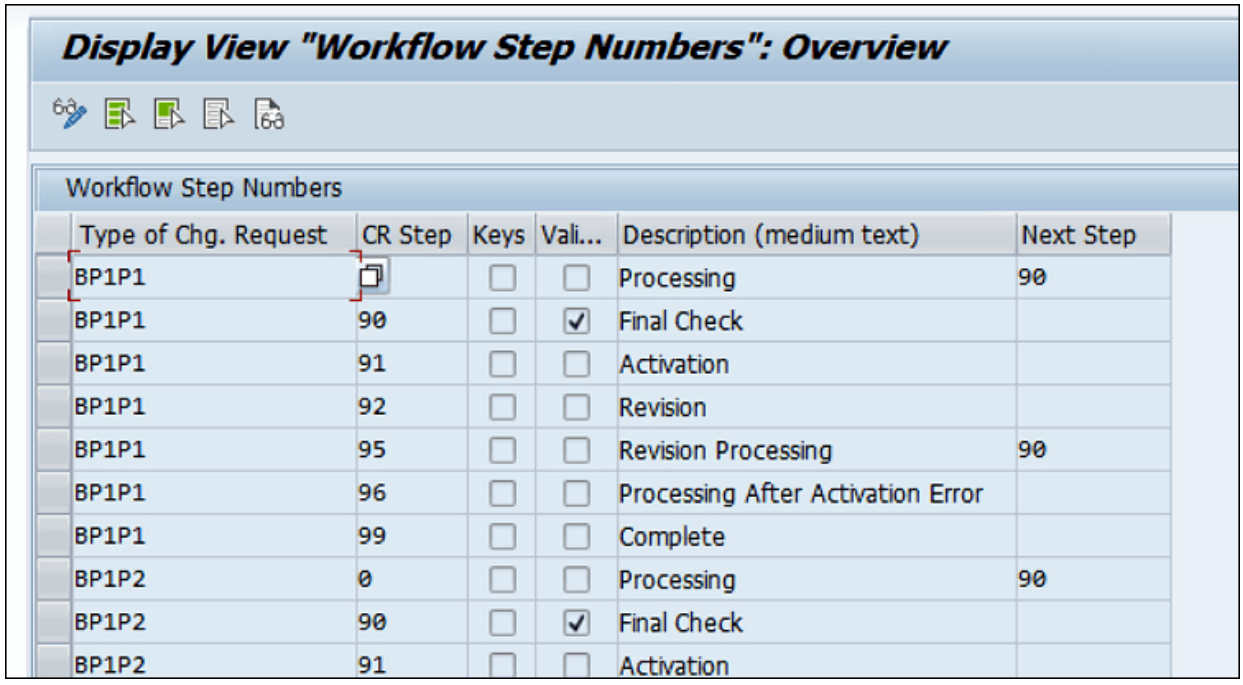


Figure 13.4 Change Request Steps for Change Request Types That Use Rule-Based Workflows

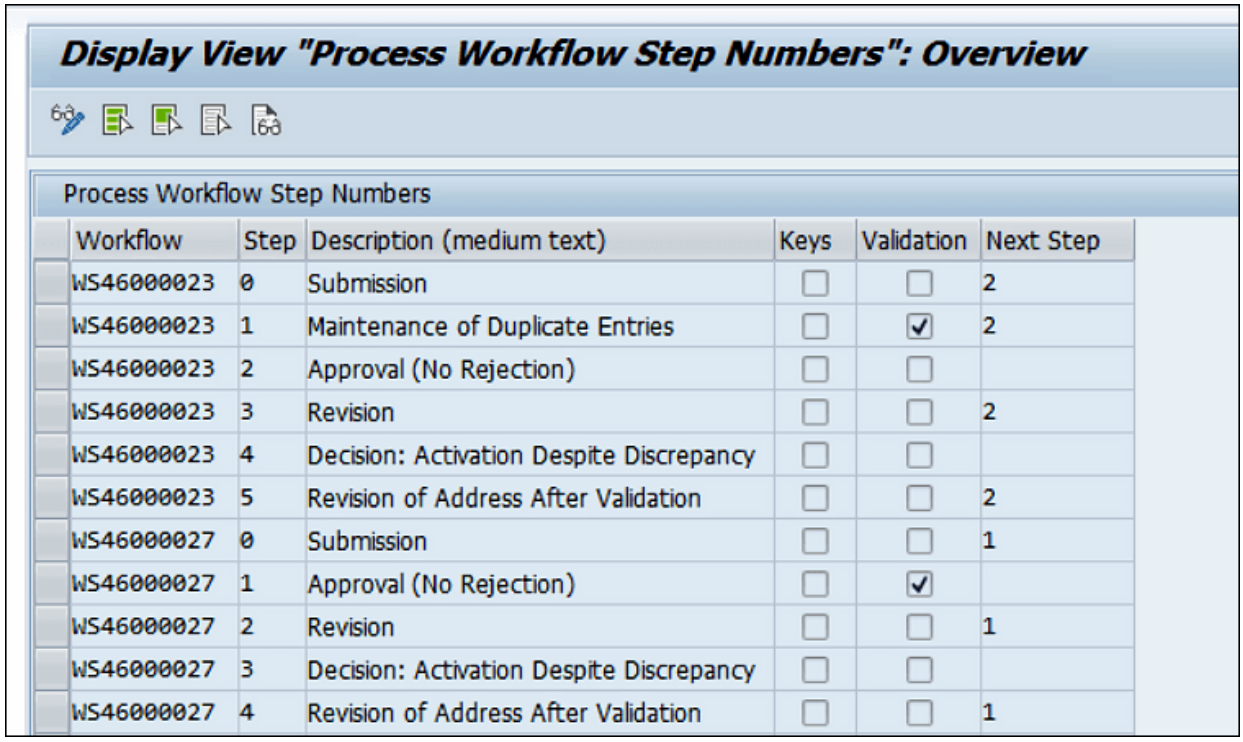
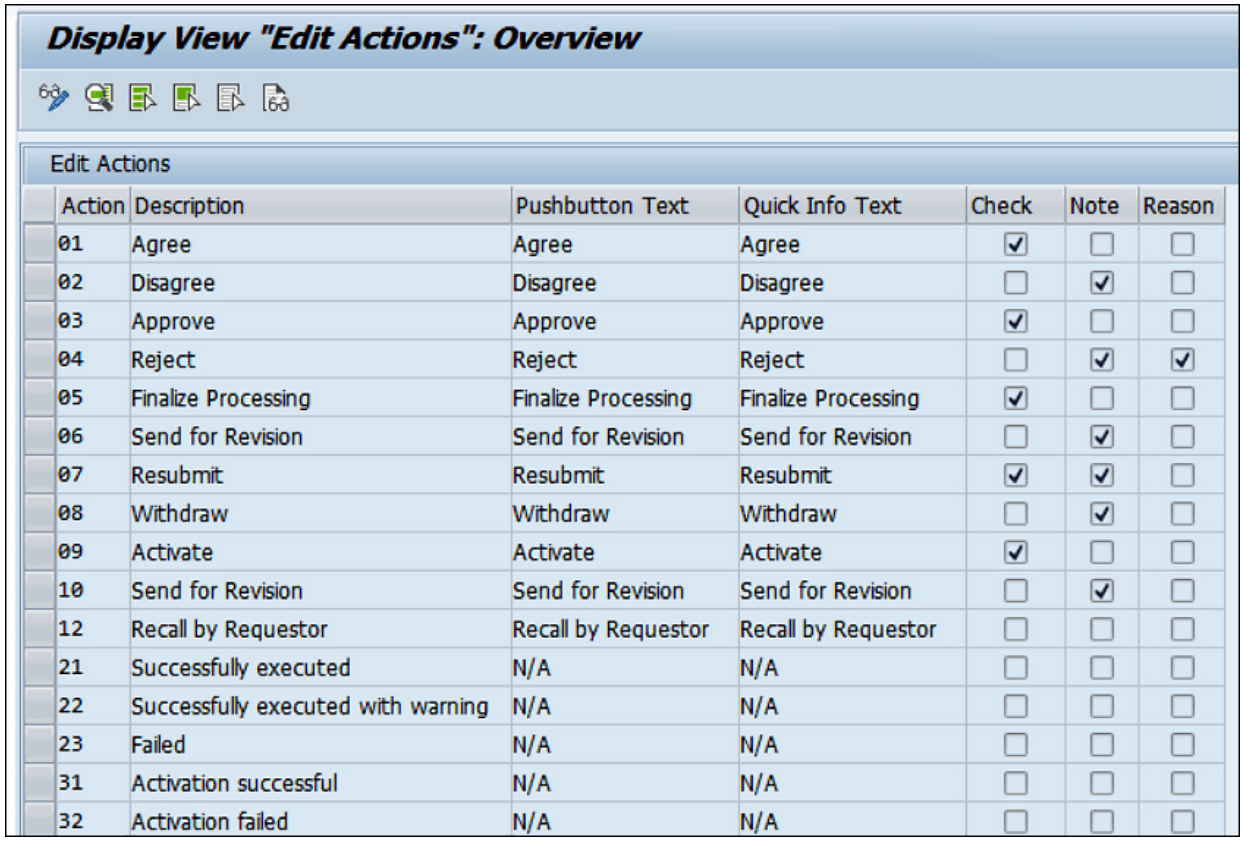


Figure 13.5 Change Request Steps Defined for Preconfigured Workflow Templates



Display View "Edit Actions": Overview

Edit Actions

Action	Description	Pushbutton Text	Quick Info Text	Check	Note	Reason
01	Agree	Agree	Agree	☑	☐	☐
02	Disagree	Disagree	Disagree	☐	☑	☐
03	Approve	Approve	Approve	☑	☐	☐
04	Reject	Reject	Reject	☐	☑	☑
05	Finalize Processing	Finalize Processing	Finalize Processing	☑	☐	☐
06	Send for Revision	Send for Revision	Send for Revision	☐	☑	☐
07	Resubmit	Resubmit	Resubmit	☑	☑	☐
08	Withdraw	Withdraw	Withdraw	☐	☑	☐
09	Activate	Activate	Activate	☑	☐	☐
10	Send for Revision	Send for Revision	Send for Revision	☐	☑	☐
12	Recall by Requestor	Recall by Requestor	Recall by Requestor	☐	☐	☐
21	Successfully executed	N/A	N/A	☐	☐	☐
22	Successfully executed with warning	N/A	N/A	☐	☐	☐
23	Failed	N/A	N/A	☐	☐	☐
31	Activation successful	N/A	N/A	☐	☐	☐
32	Activation failed	N/A	N/A	☐	☐	☐

Figure 13.6 Examples of Change Request Actions Delivered with the Standard System

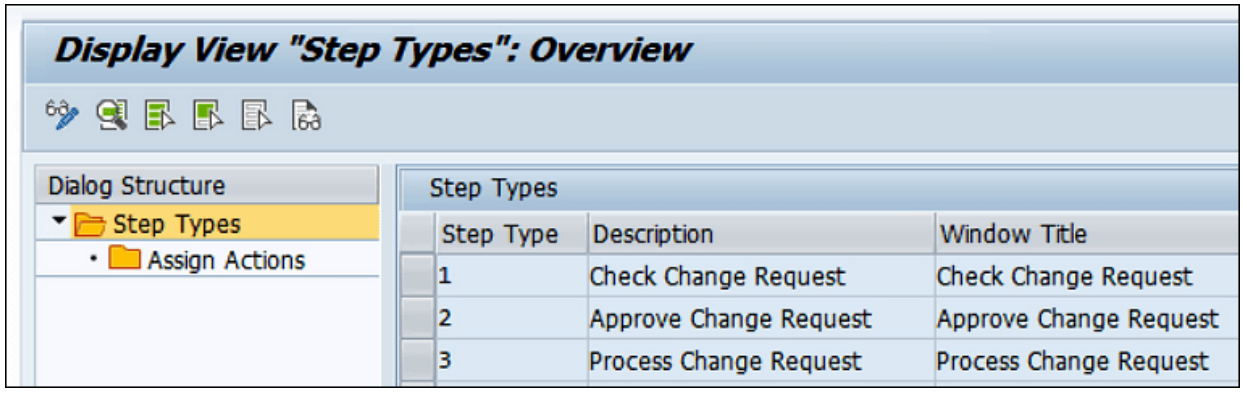


Figure 13.7 Change Request Step Types

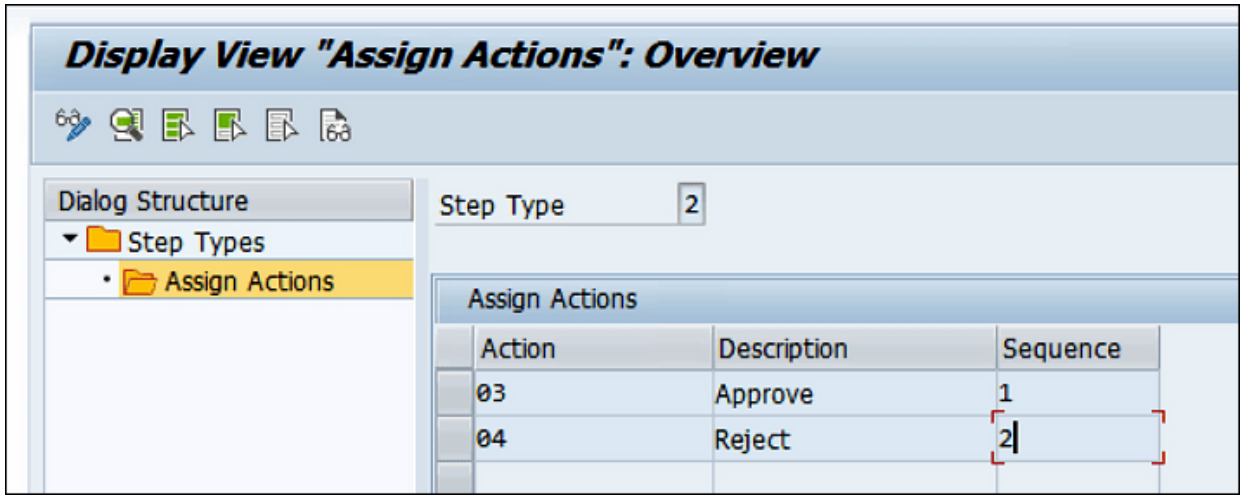


Figure 13.8 Actions Assigned to Change Request Step Type Approve Change Request

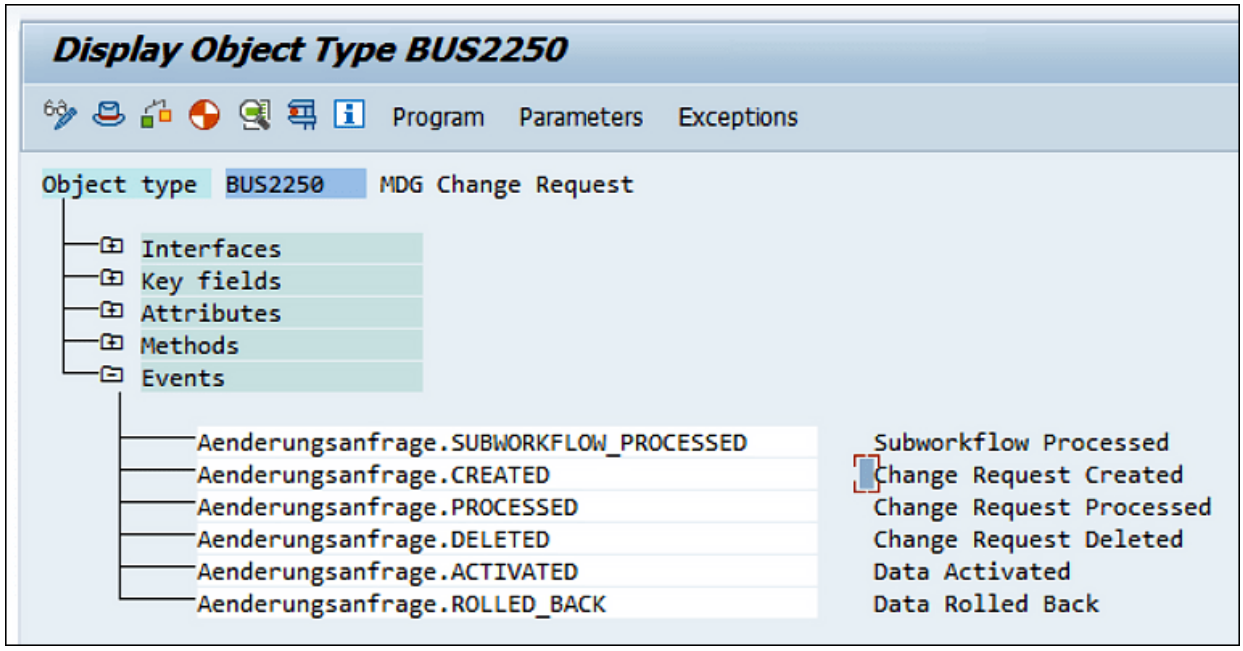


Figure 13.9 Business Object BUS2250 (SAP MDG Change Request)

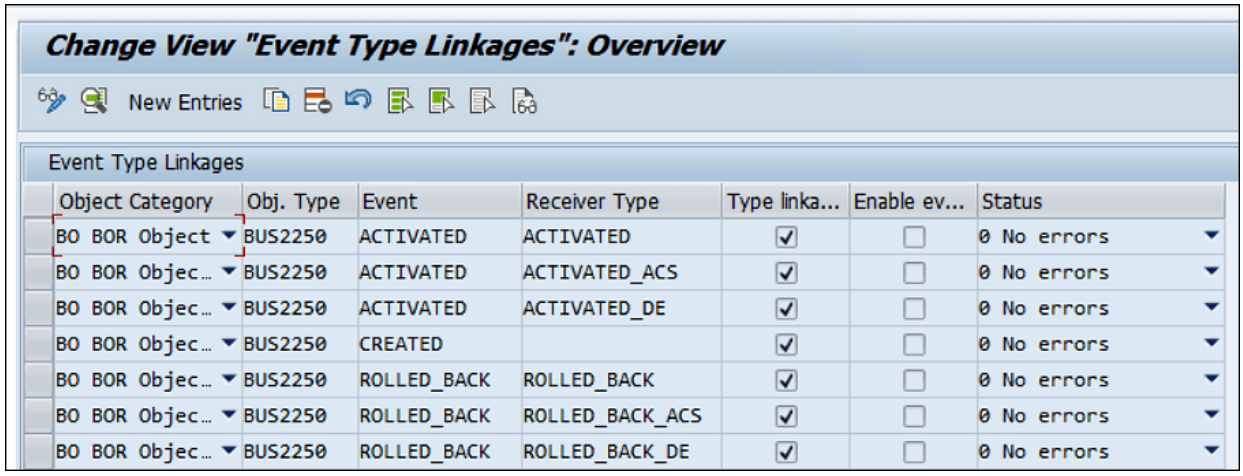


Figure 13.10 Event Type Linkages for Object BUS2250 (SAP MDG Change Request) Part 1

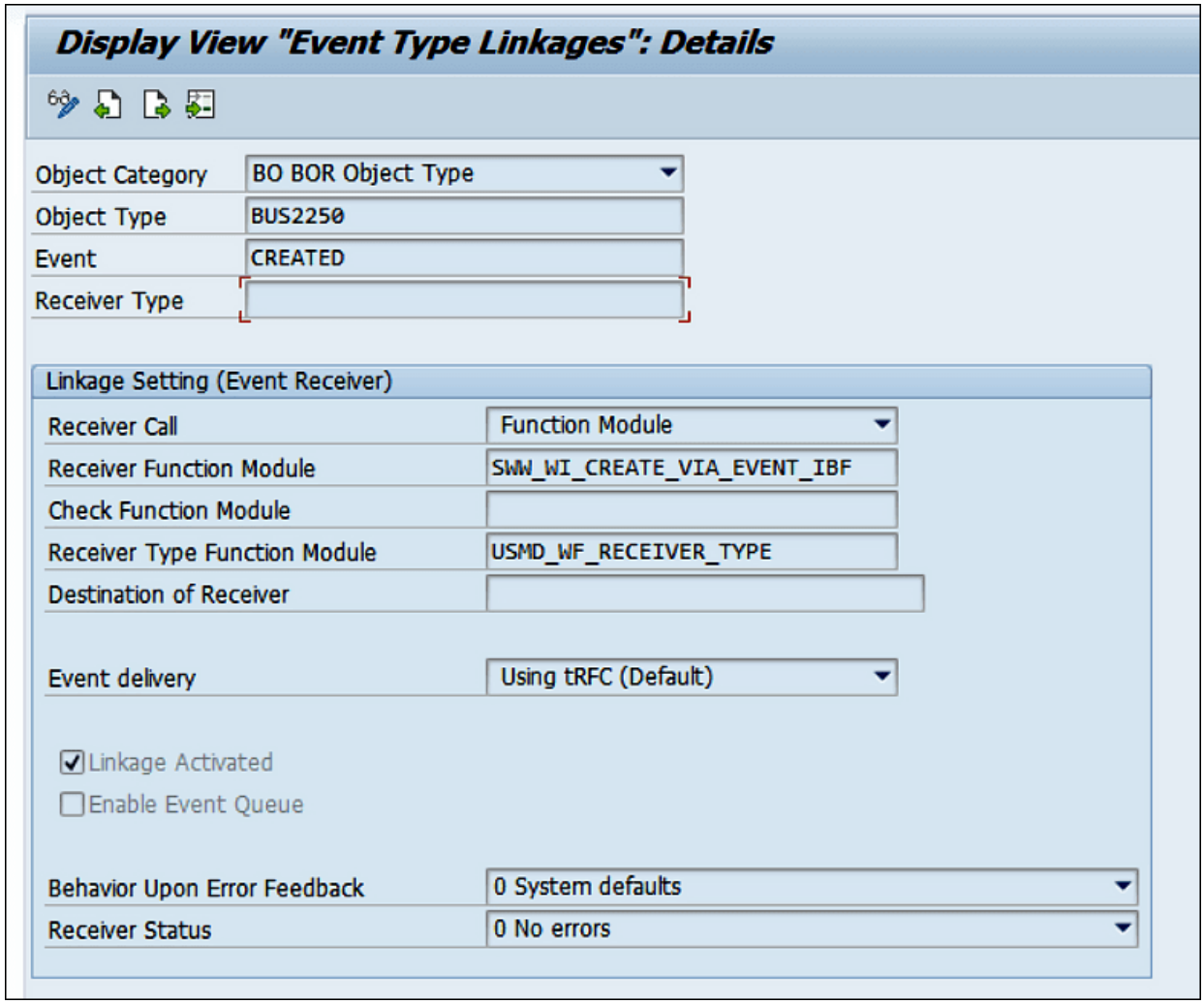


Figure 13.11 Event Type Linkages for Object BUS2250 (SAP MDG Change Request) Part 2

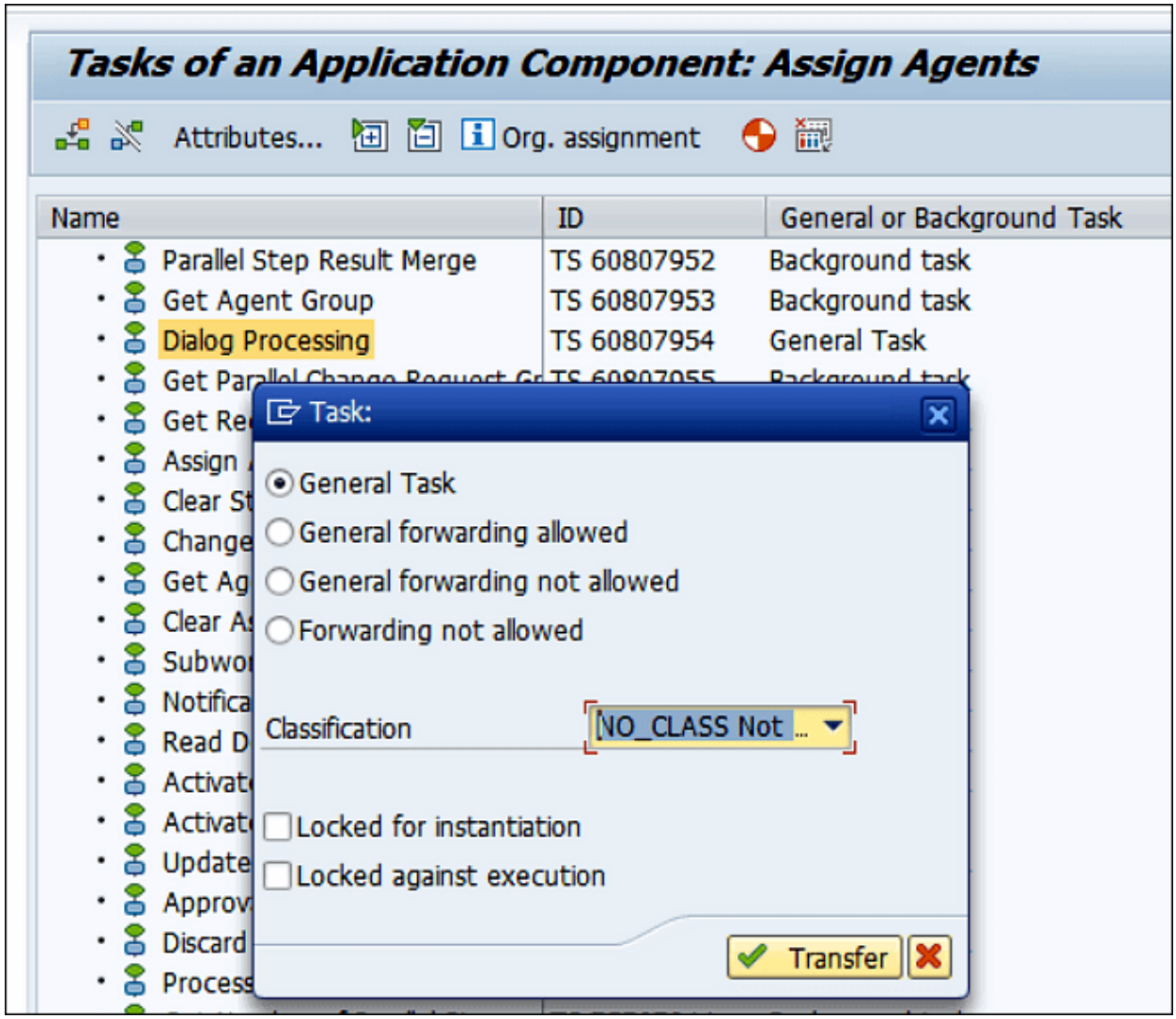


Figure 13.12 Categorizing Workflow Tasks as General Tasks

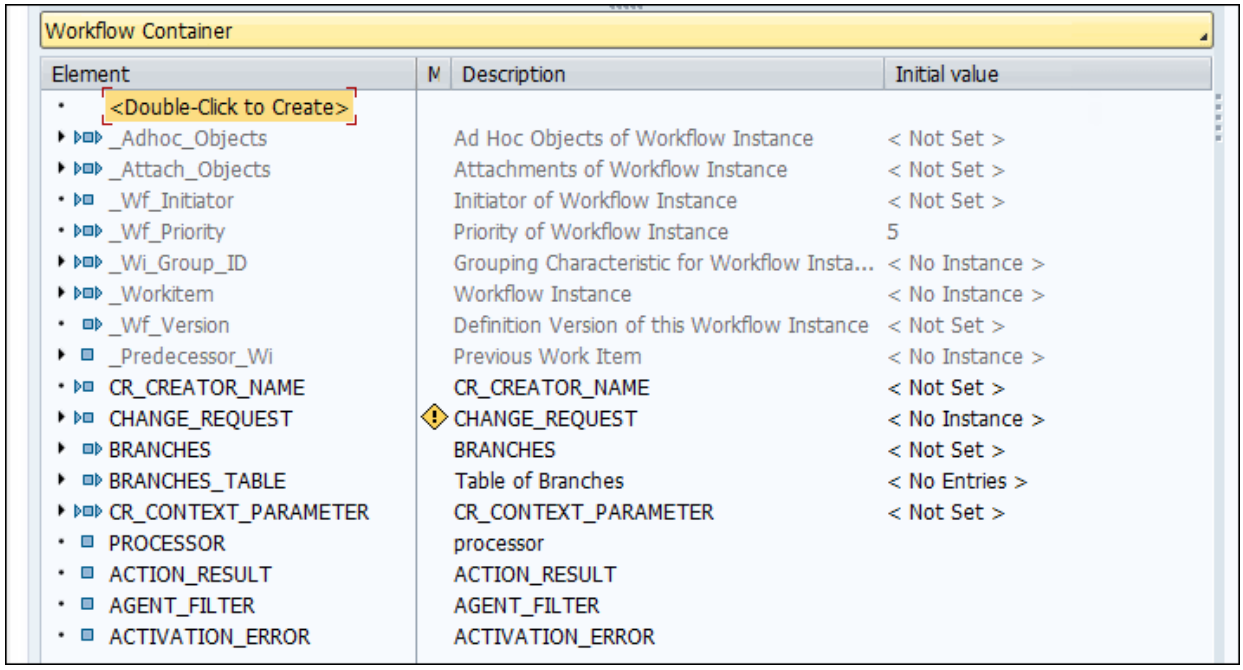


Figure 13.13 Typical Workflow Container Used by Standard Workflow Templates in SAP MDG

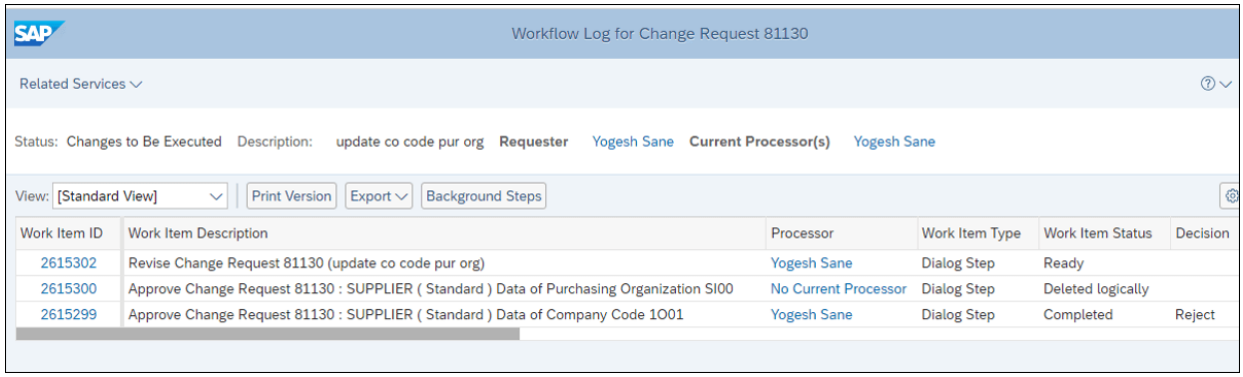


Figure 13.14 Workflow Log for an SAP MDG Change Request

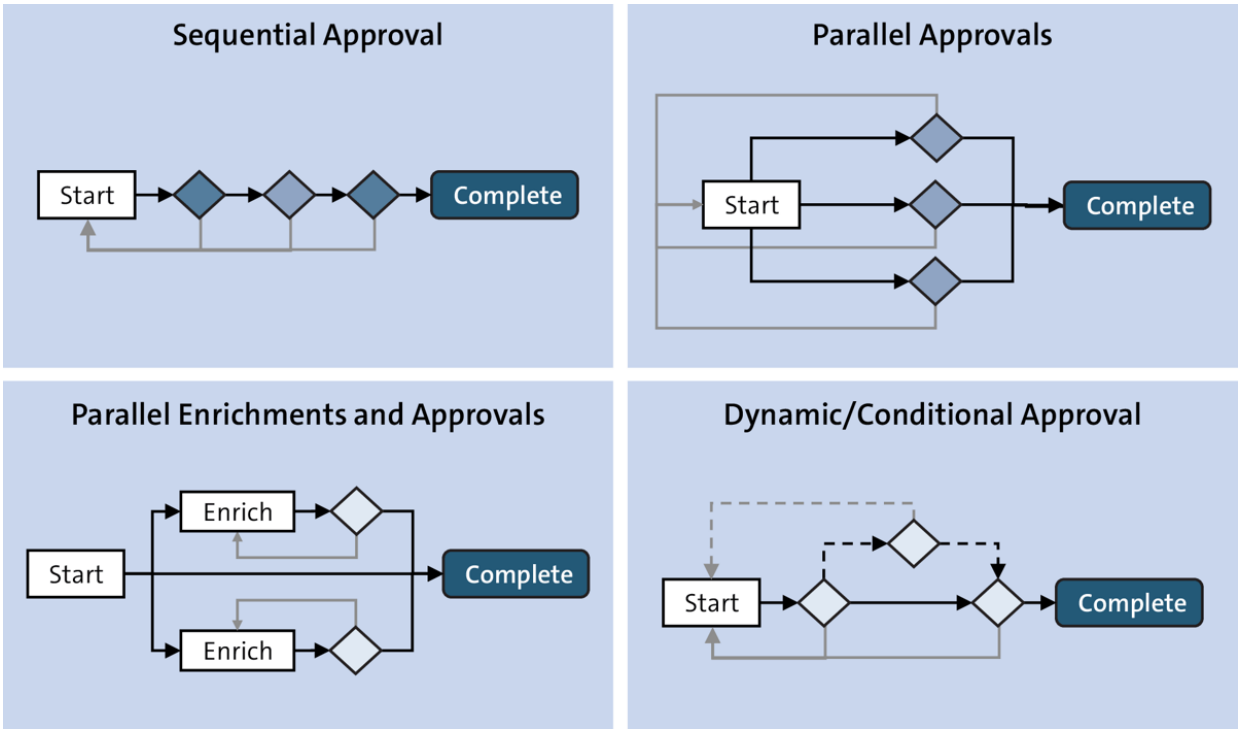


Figure 13.15 Examples of Process Patterns Built Using the Rule-Based Workflow Template

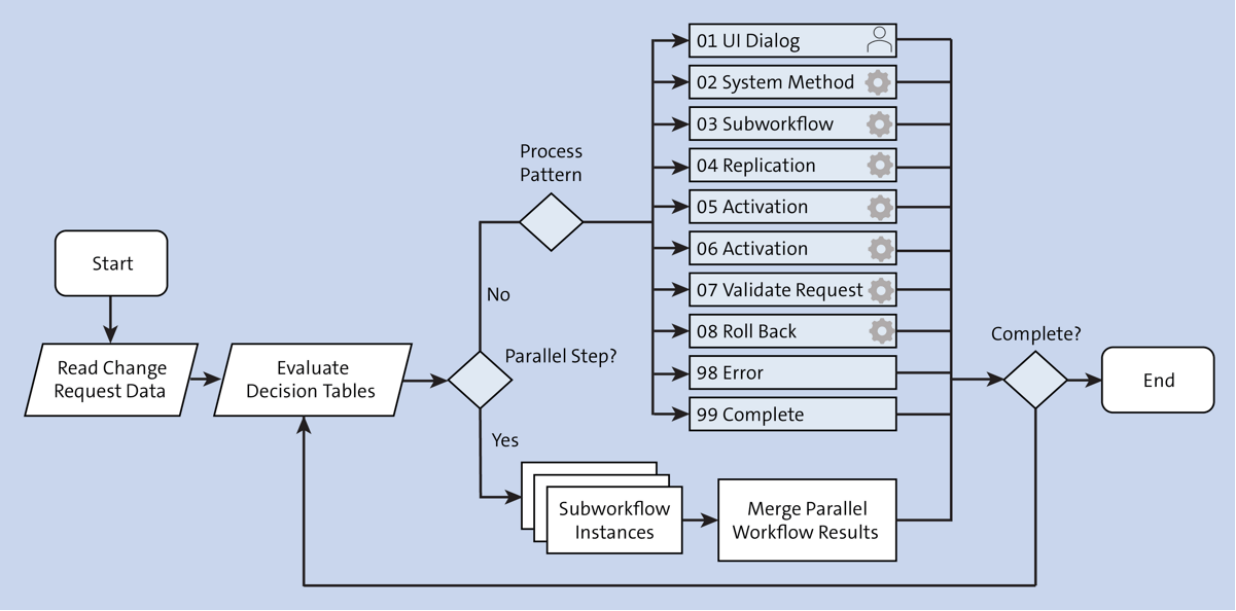


Figure 13.16 Process Diagram for Rule-Based Workflow Template WS60800086

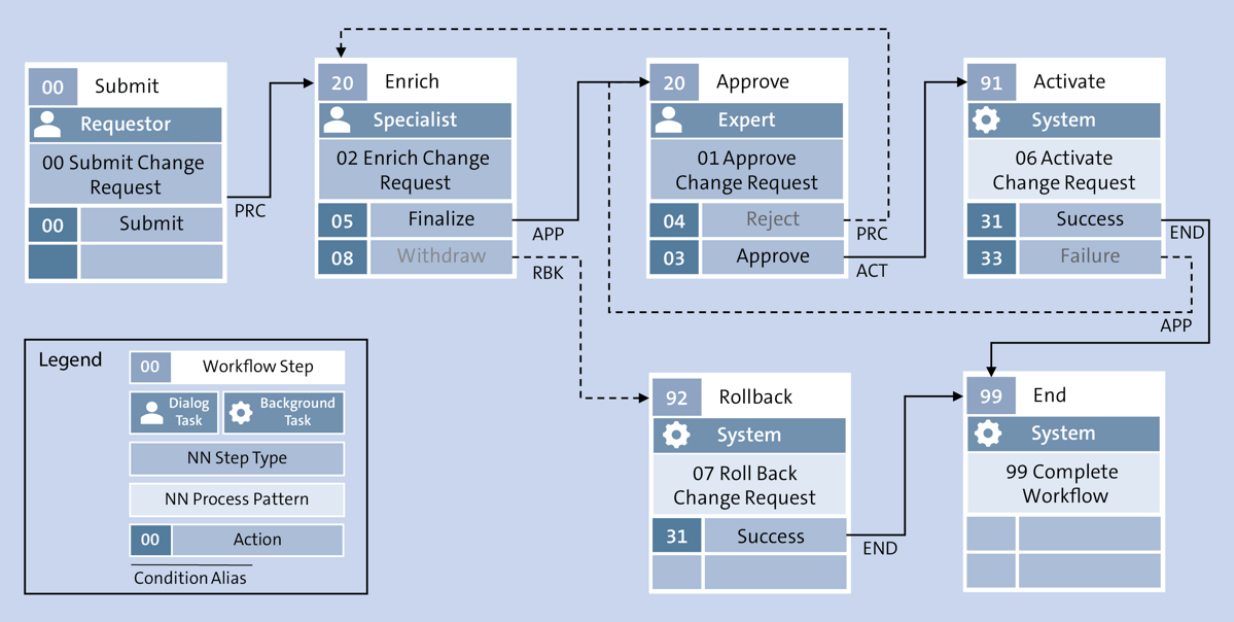


Figure 13.17 Outline of a Process Implemented Using the Rule-Based Workflow

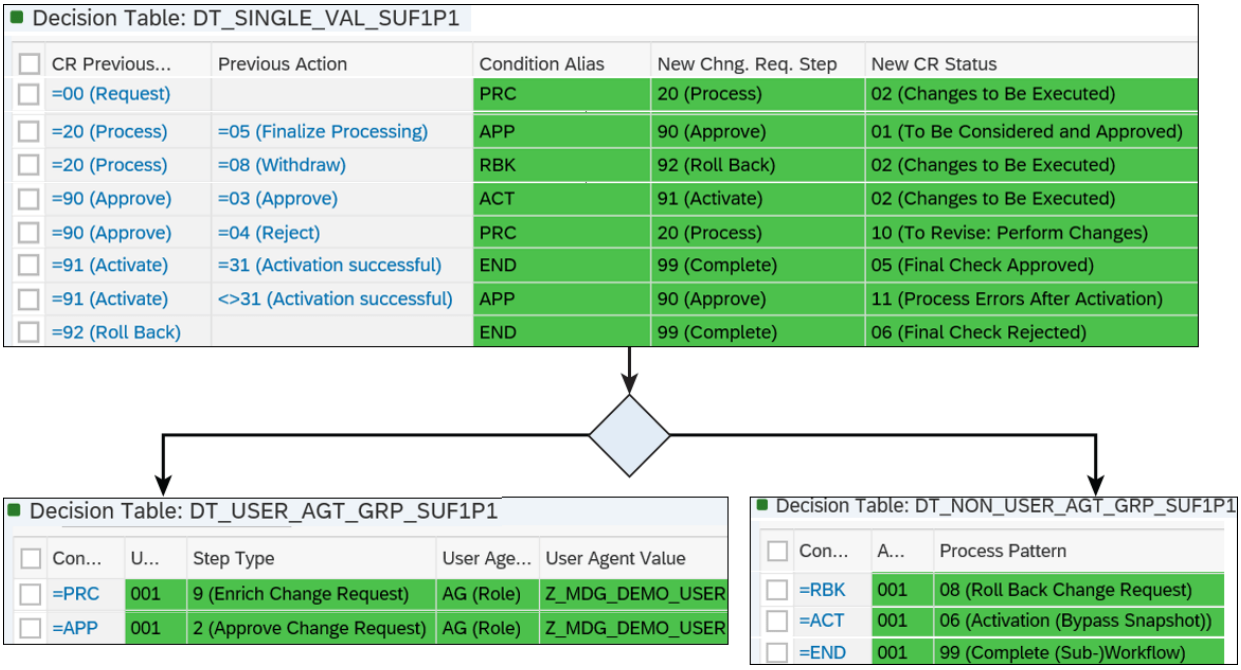


Figure 13.18 Decision Table Entries for the Preceding Process

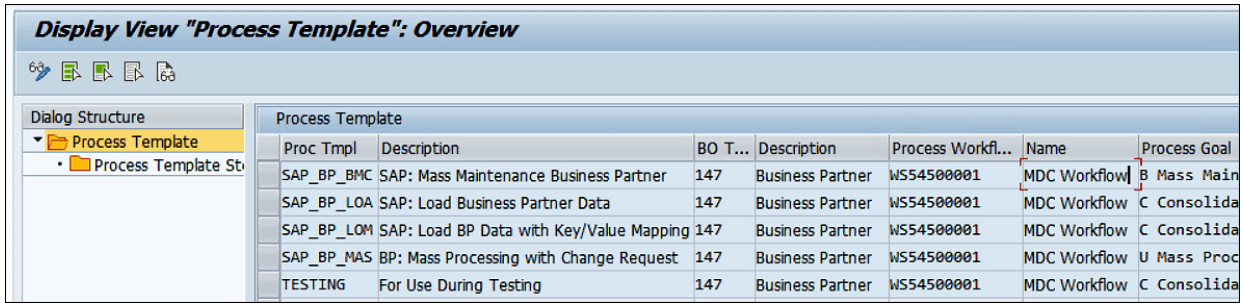


Figure 13.19 Process Templates Used in SAP MDG, Consolidation and Mass Processing

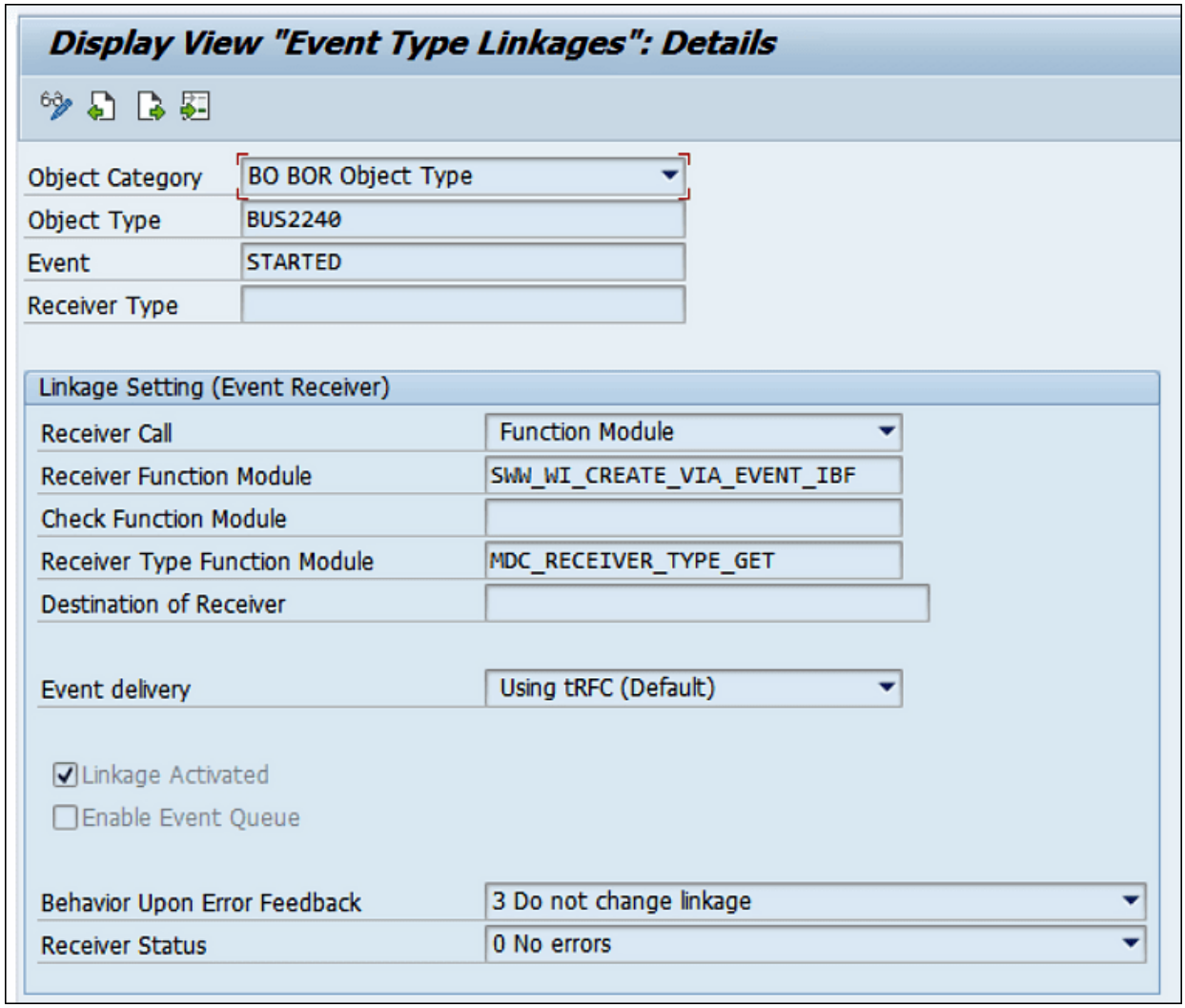


Figure 13.20 Event Type Linkage for the STARTED Event of Business Object BUS2240

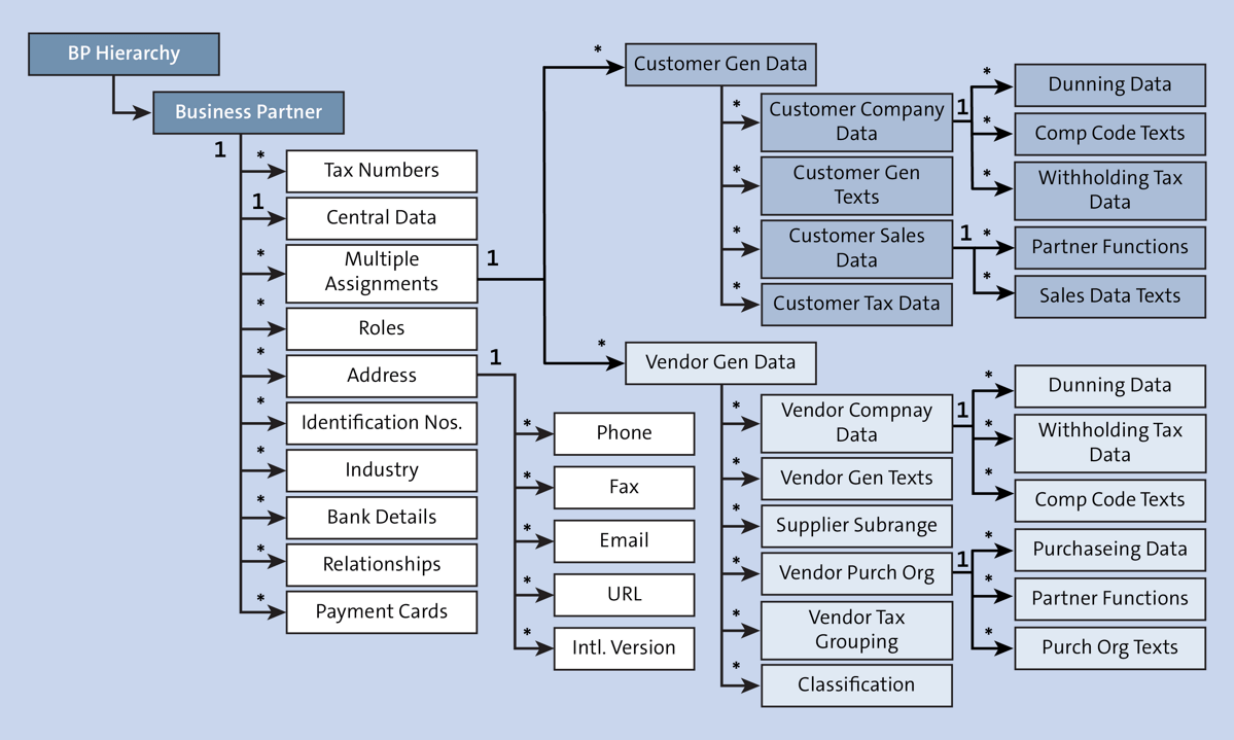


Figure 13.21 Business Partner Data Model Used in SAP MDG

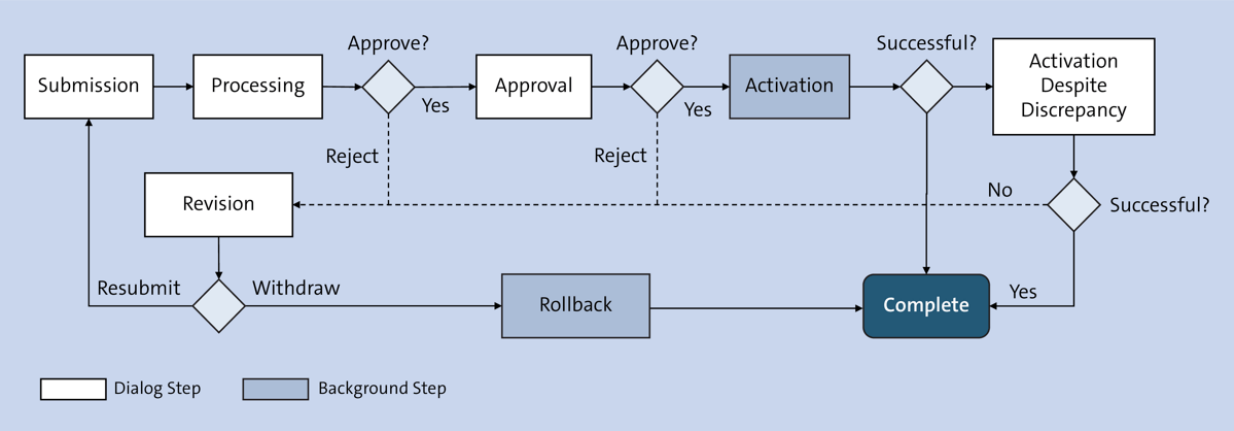


Figure 13.22 Workflow Process Diagram for WS60800095 and WS72100006

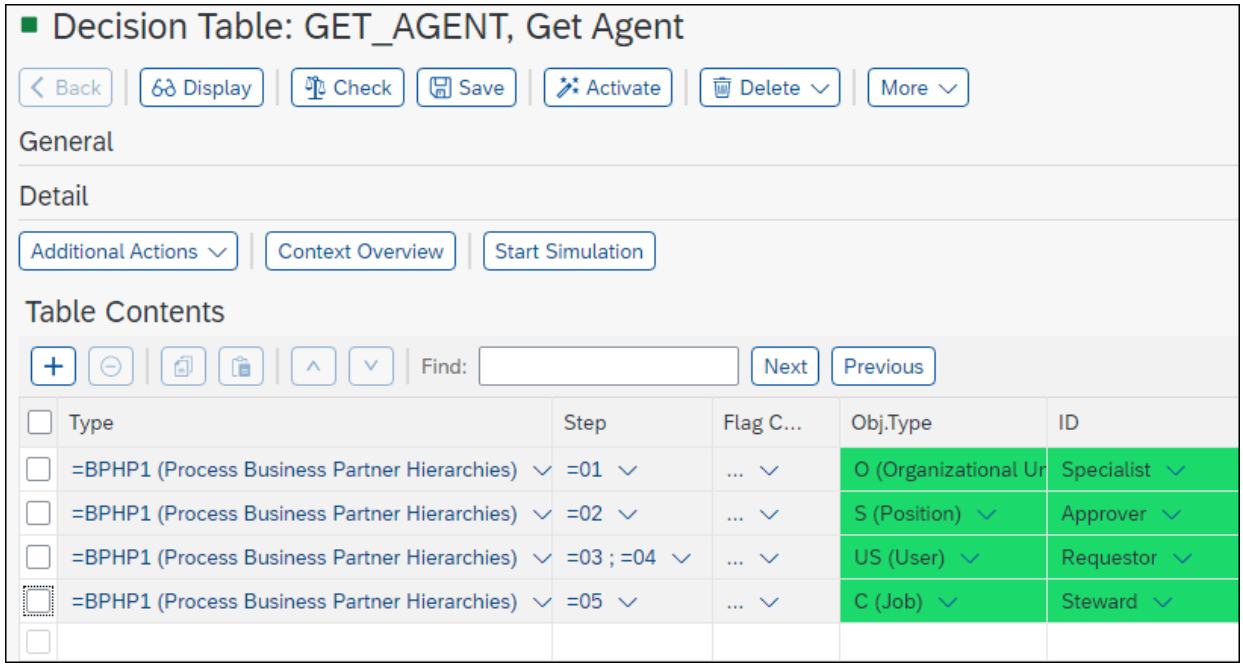


Figure 13.23 Example of Agent Determination Using a BRFPlus Decision Table

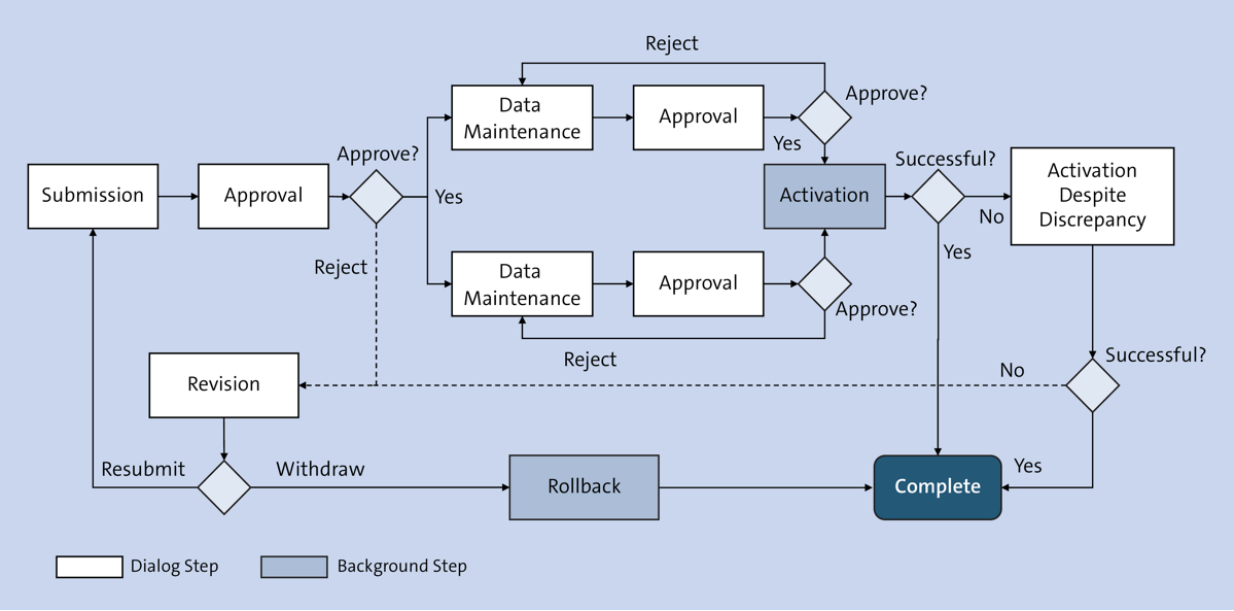


Figure 13.24 Process Diagram for Workflow Templates WS54400001 and WS54300005

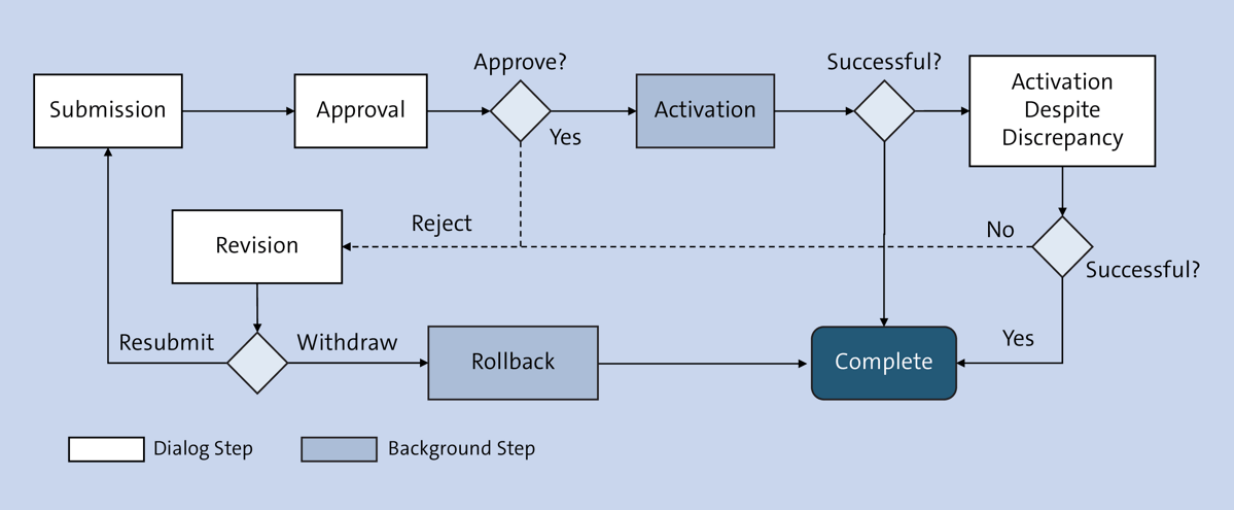


Figure 13.25 Workflow Process Diagram for Template WS54300003

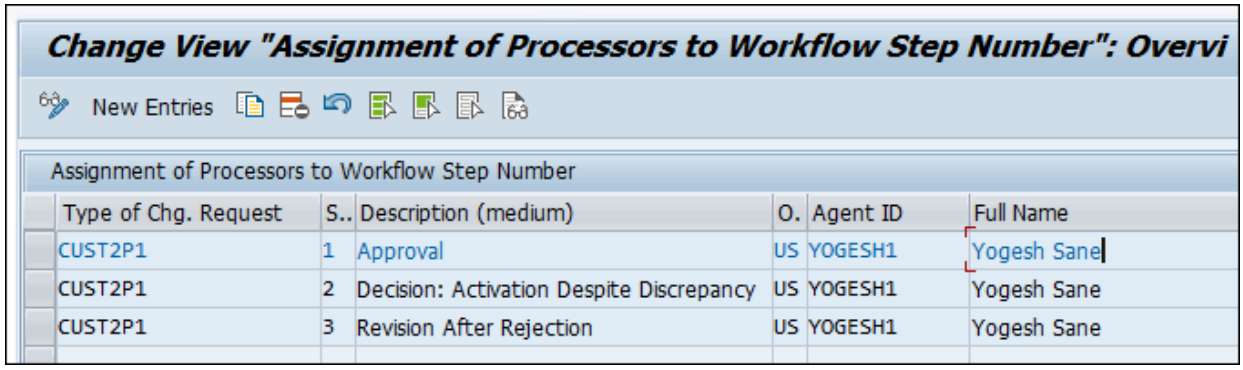


Figure 13.26 Agent Assignment for Workflow Template WS54300003

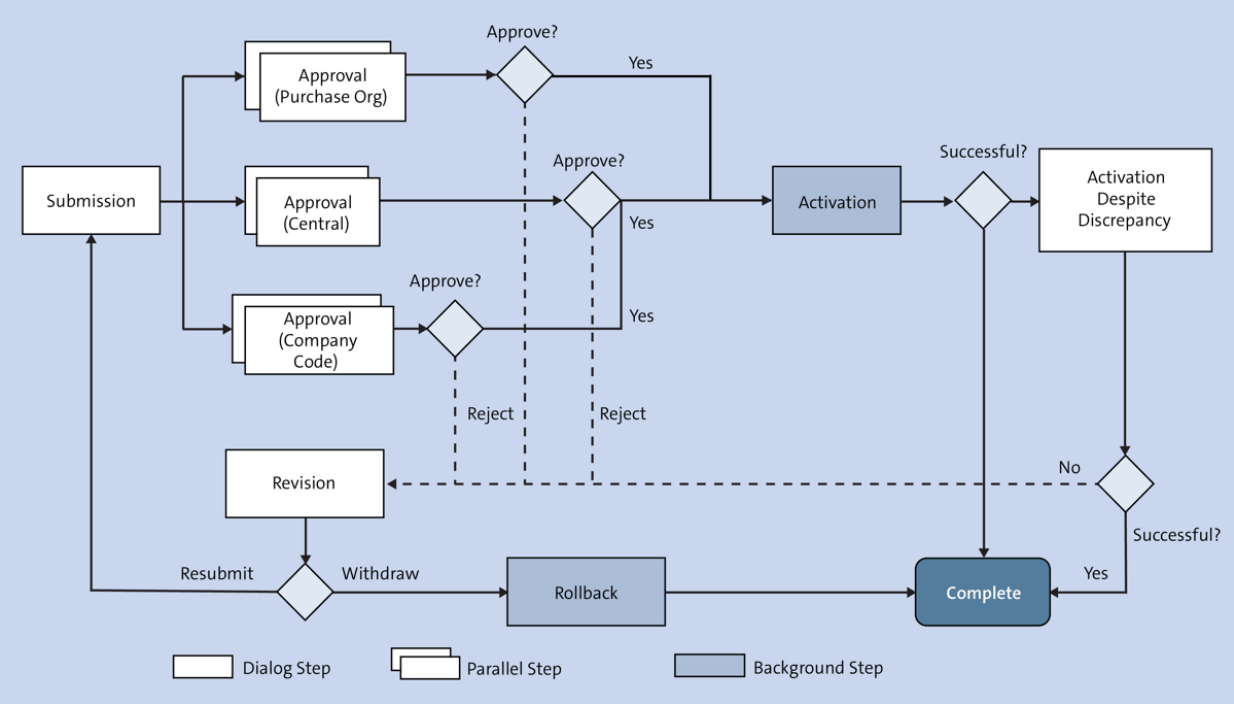


Figure 13.27 Workflow Process Diagram Template for WS54300007

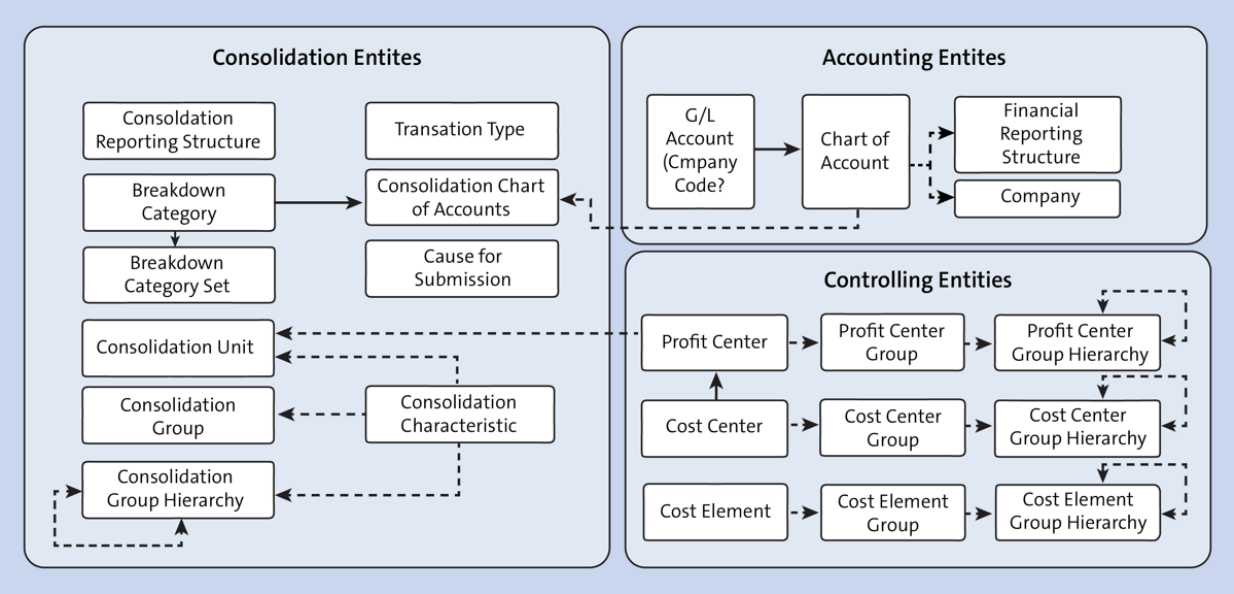


Figure 13.28 Entities and Relationships in the Finance Data Model in SAP MDG

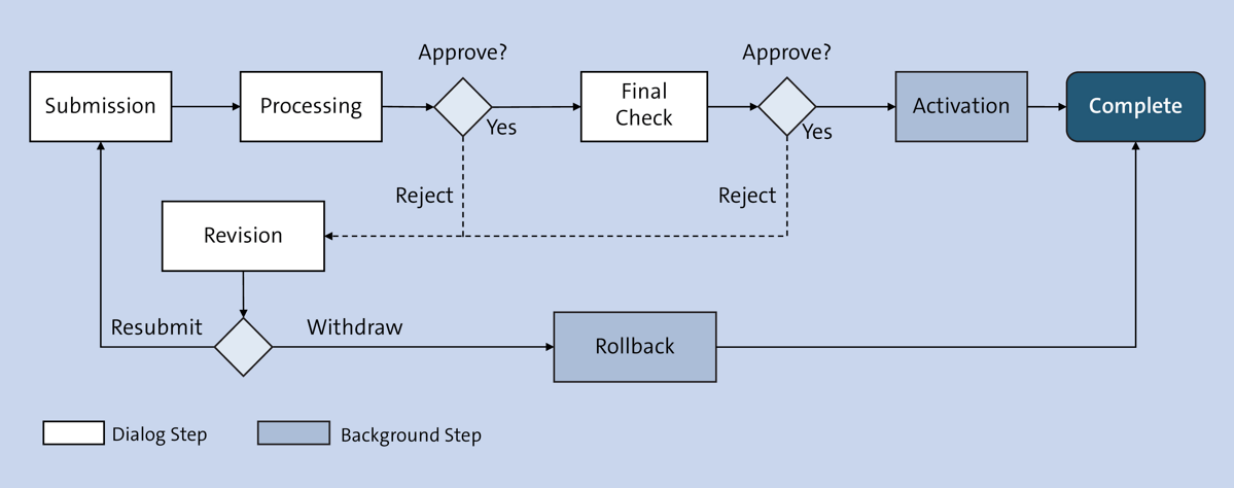


Figure 13.29 Workflow Process Diagram for Template WS75700040

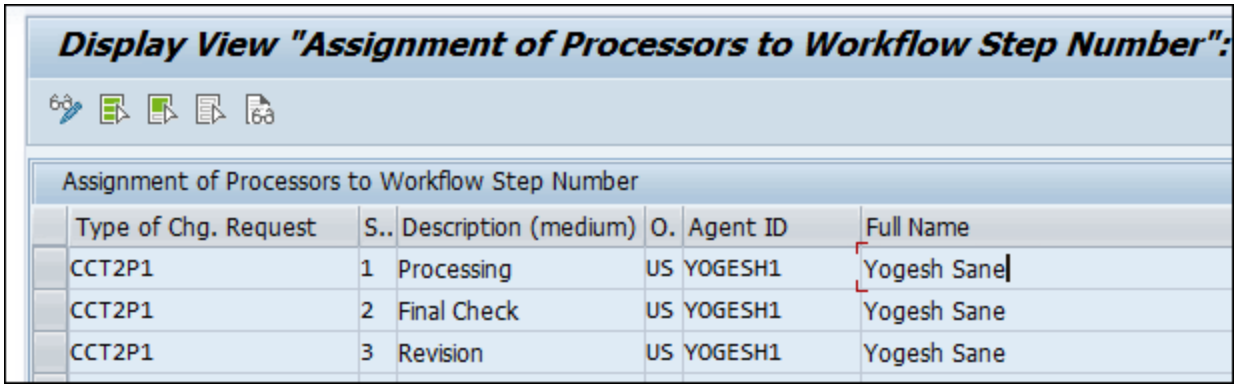


Figure 13.30 Agent Assignment for Workflow Template WS75700040

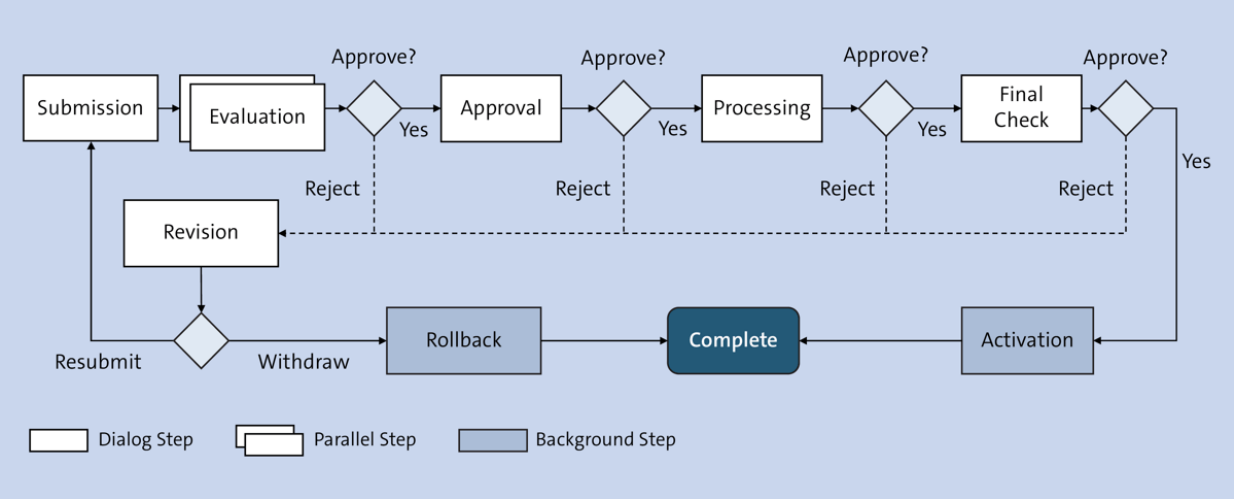


Figure 13.31 Process Diagram for Advanced Workflow Template WS75700027

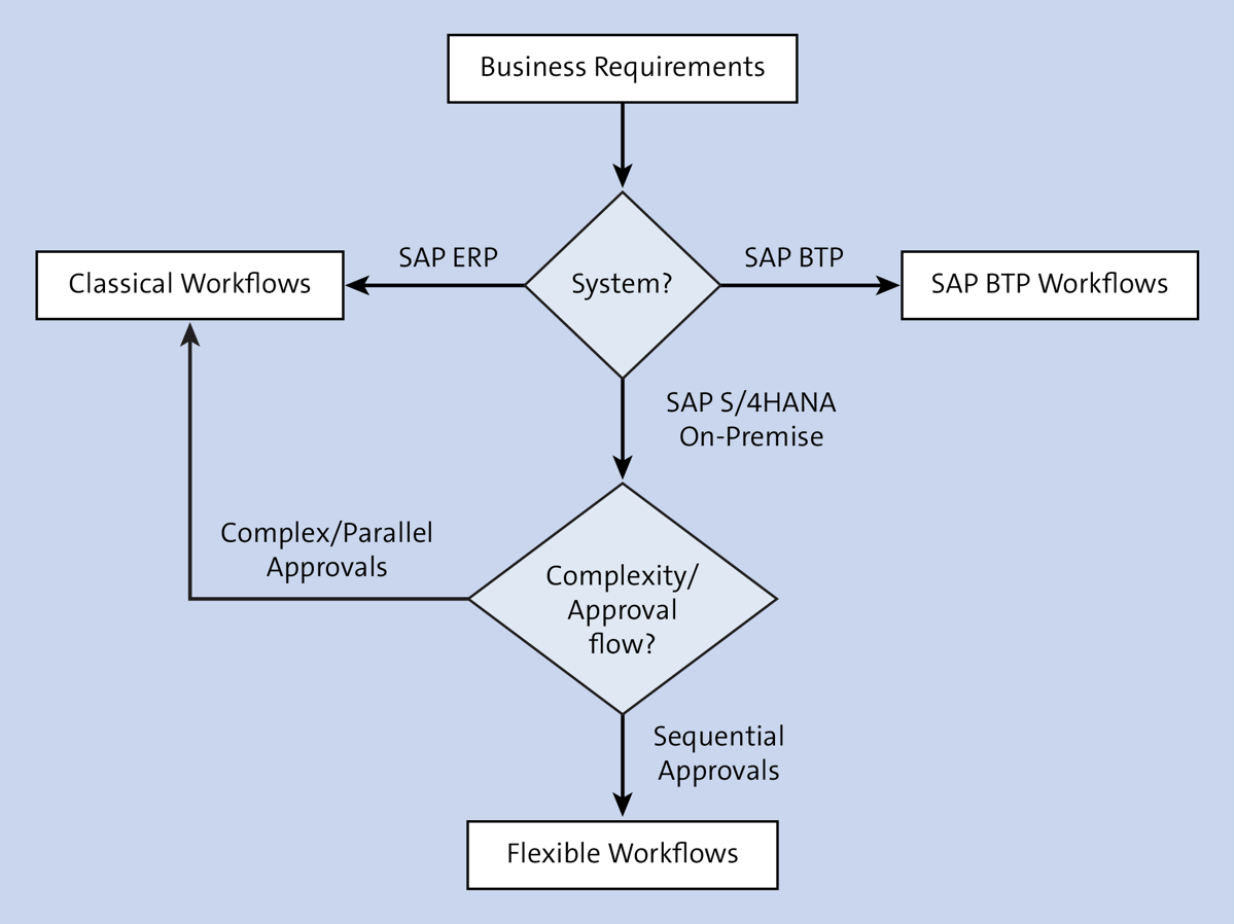


Figure 14.1 Choosing between Classical and Flexible Workflows

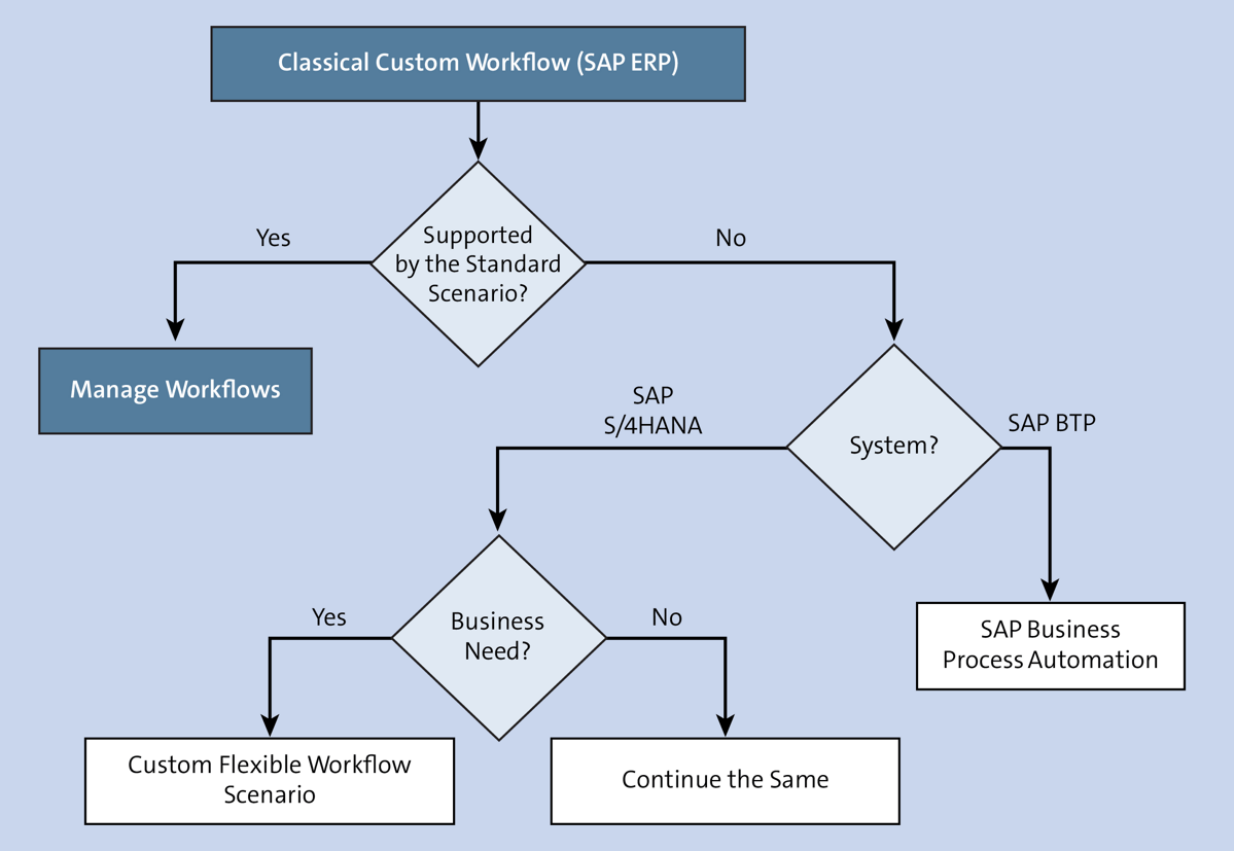


Figure 14.2 Migration Path

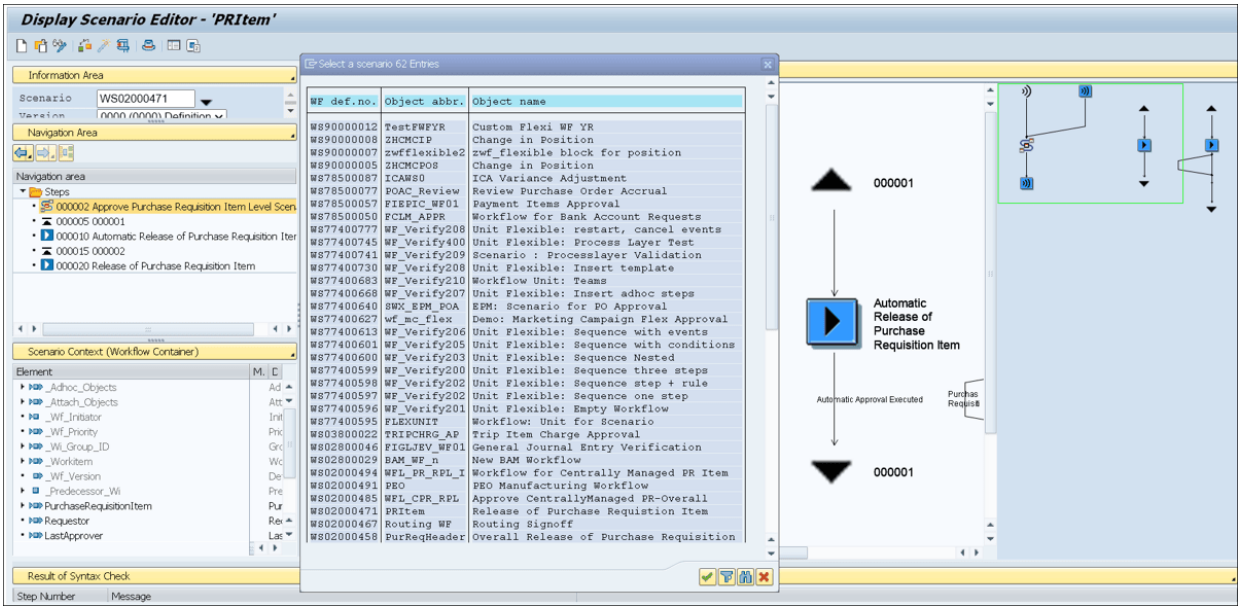


Figure 14.3 Transaction SWDD_SCENARIO

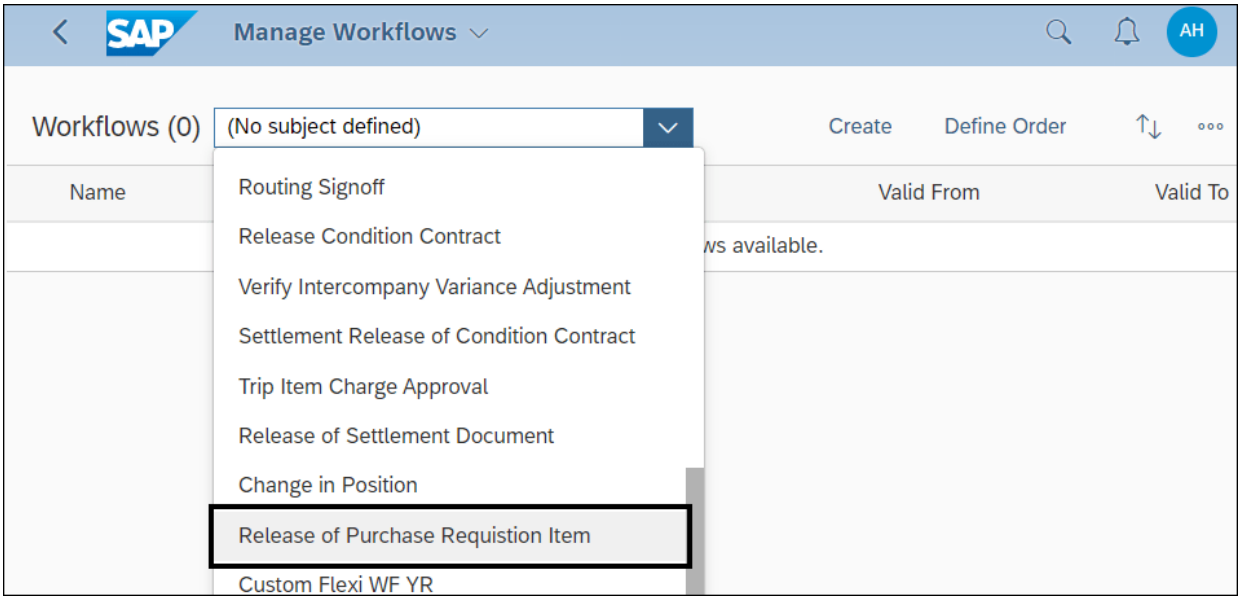


Figure 14.4 Manage Workflows App: Workflows List

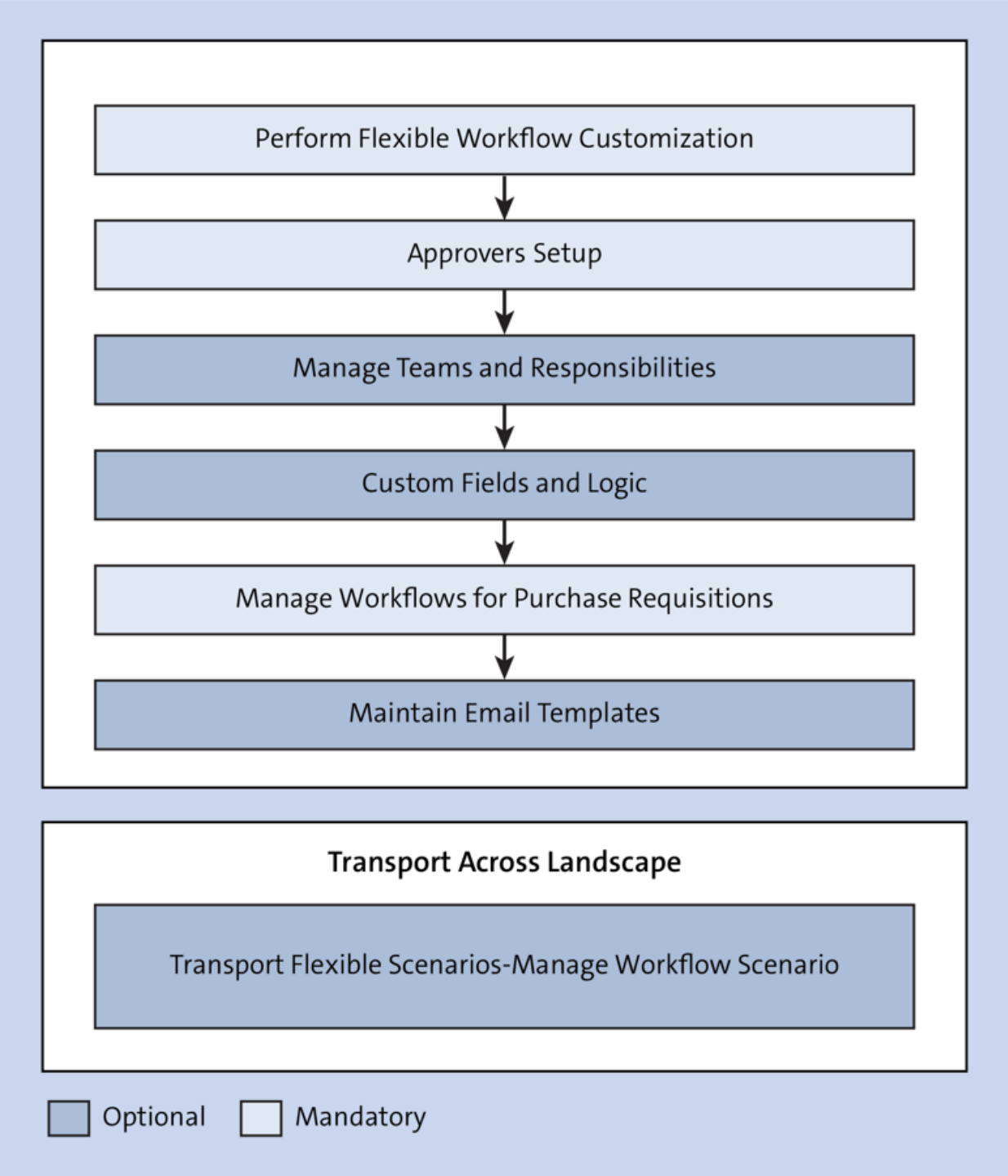


Figure 14.5 Scenario Activation Steps

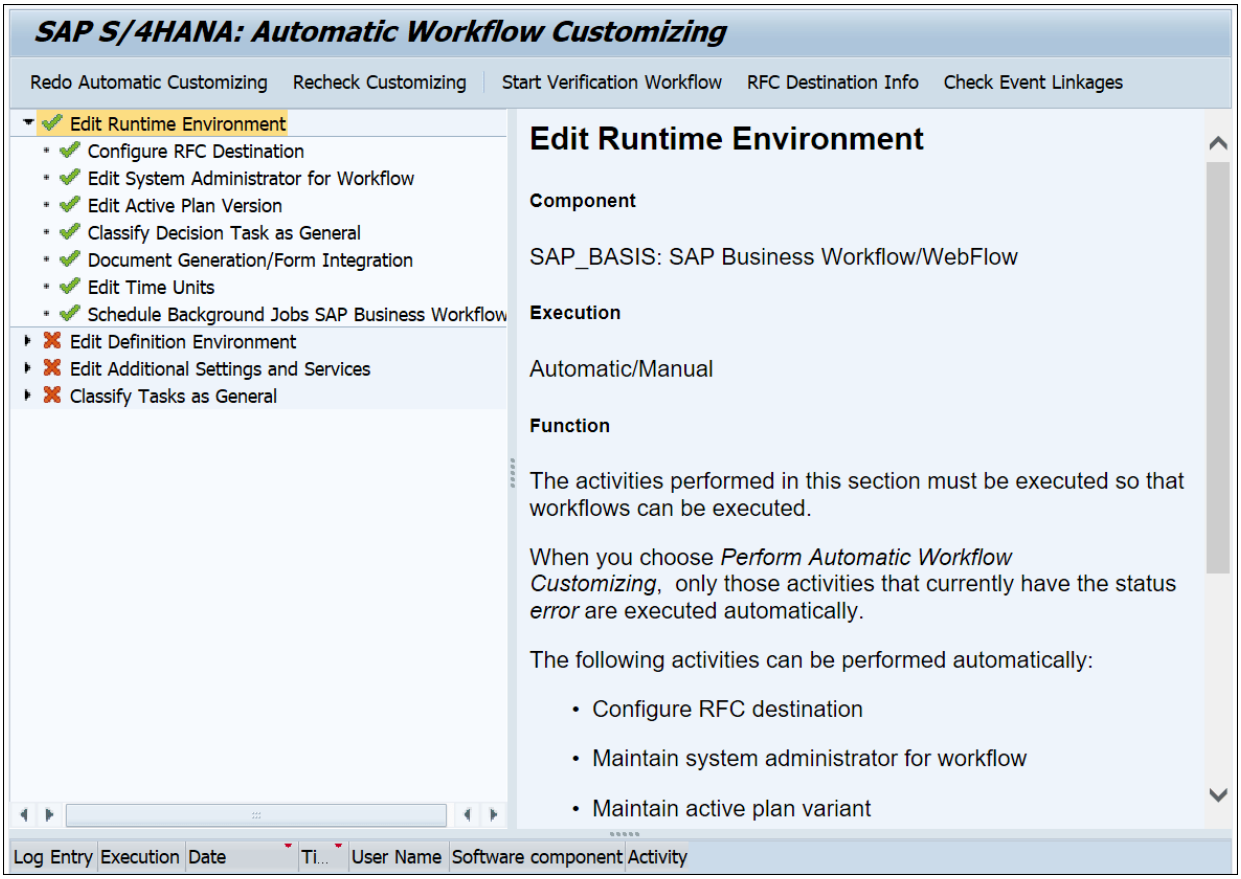


Figure 14.6 Transaction SWU3

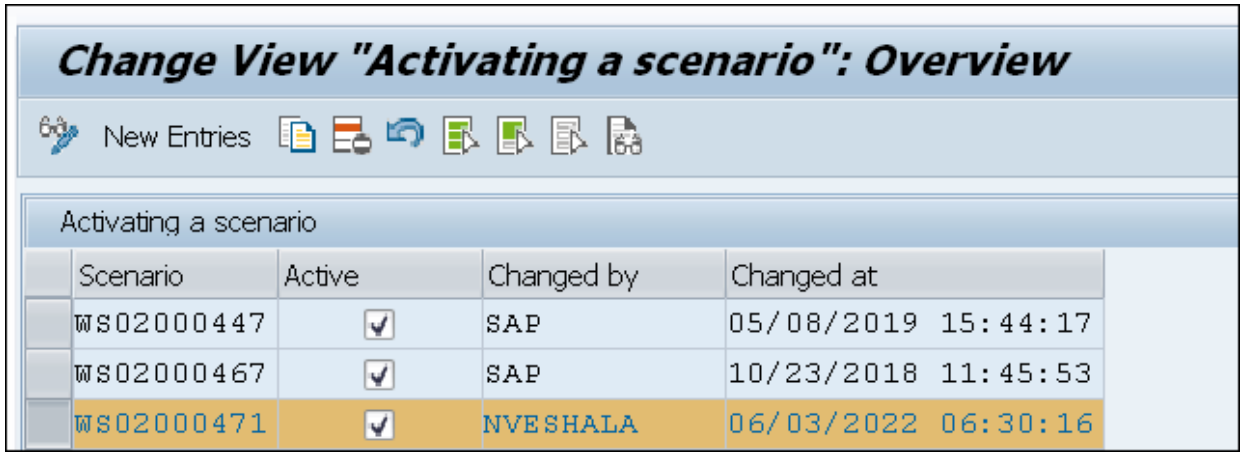
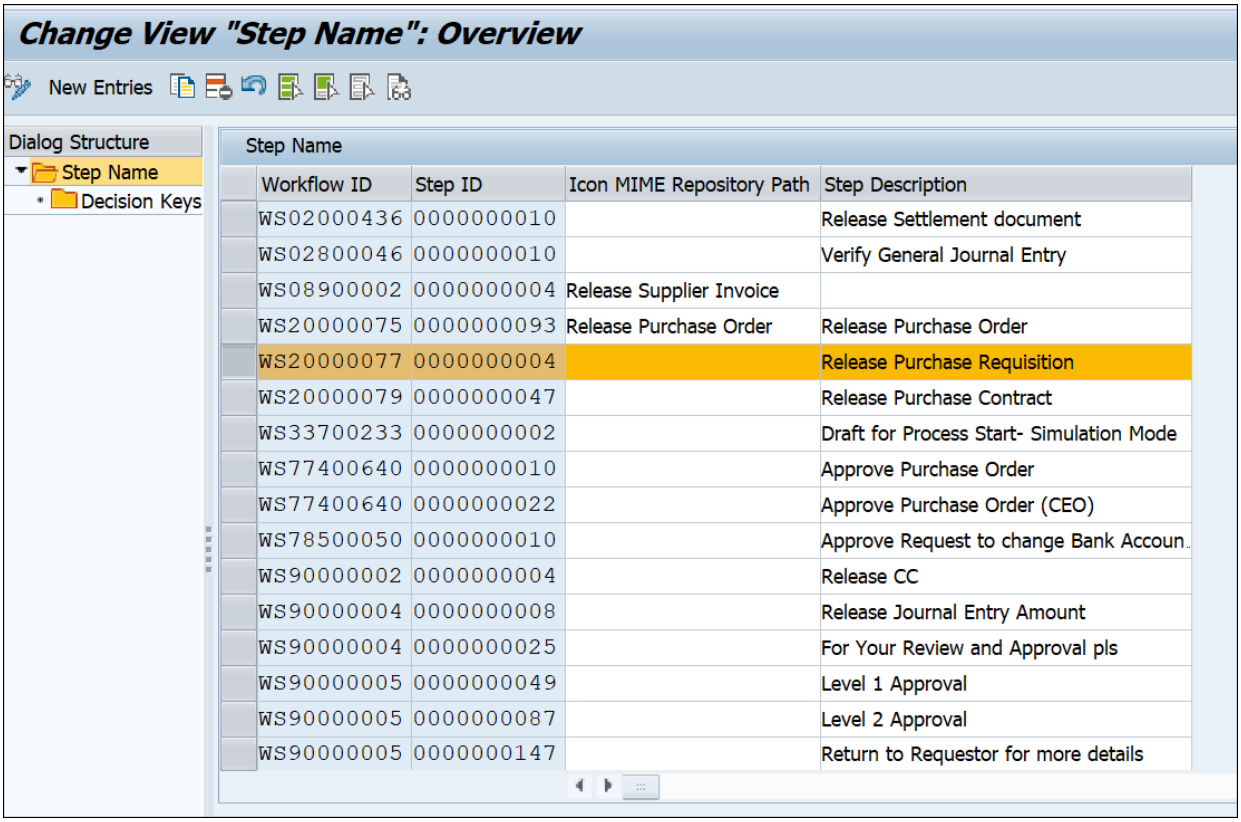


Figure 14.7 Activate the Flexible Scenario



Workflow ID	Step ID	Icon MIME Repository Path	Step Description
WS02000436	0000000010		Release Settlement document
WS02800046	0000000010		Verify General Journal Entry
WS08900002	0000000004	Release Supplier Invoice	
WS20000075	0000000093	Release Purchase Order	Release Purchase Order
WS20000077	0000000004		Release Purchase Requisition
WS20000079	0000000047		Release Purchase Contract
WS33700233	0000000002		Draft for Process Start- Simulation Mode
WS77400640	0000000010		Approve Purchase Order
WS77400640	0000000022		Approve Purchase Order (CEO)
WS78500050	0000000010		Approve Request to change Bank Accoun
WS90000002	0000000004		Release CC
WS90000004	0000000008		Release Journal Entry Amount
WS90000004	0000000025		For Your Review and Approval pls
WS90000005	0000000049		Level 1 Approval
WS90000005	0000000087		Level 2 Approval
WS90000005	0000000147		Return to Requestor for more details

Figure 14.8 My Inbox Settings

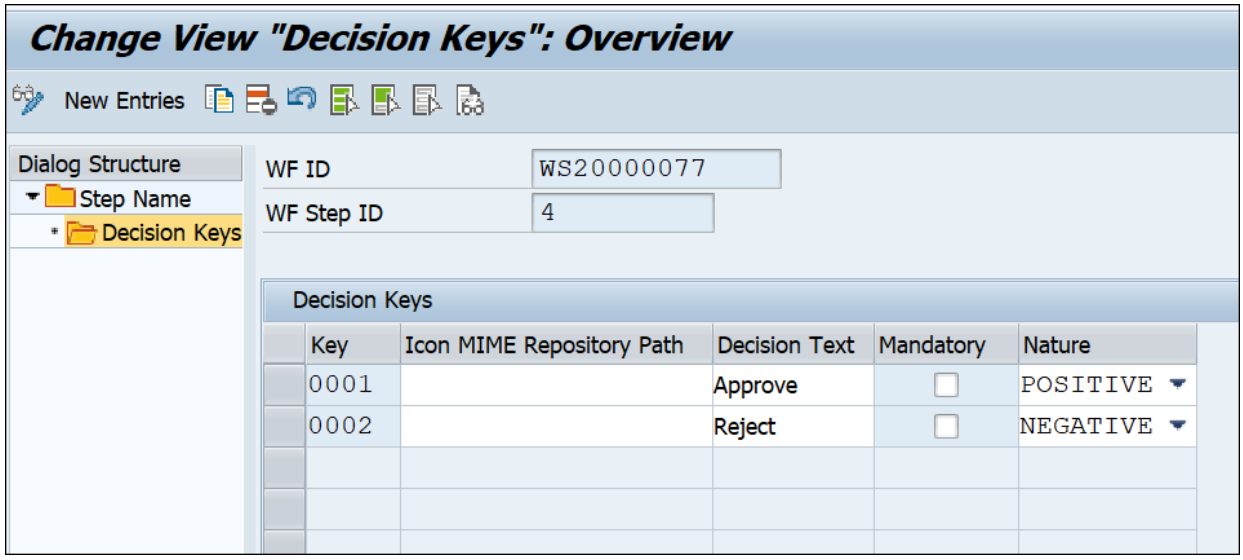


Figure 14.9 My Inbox: Task Decision Keys

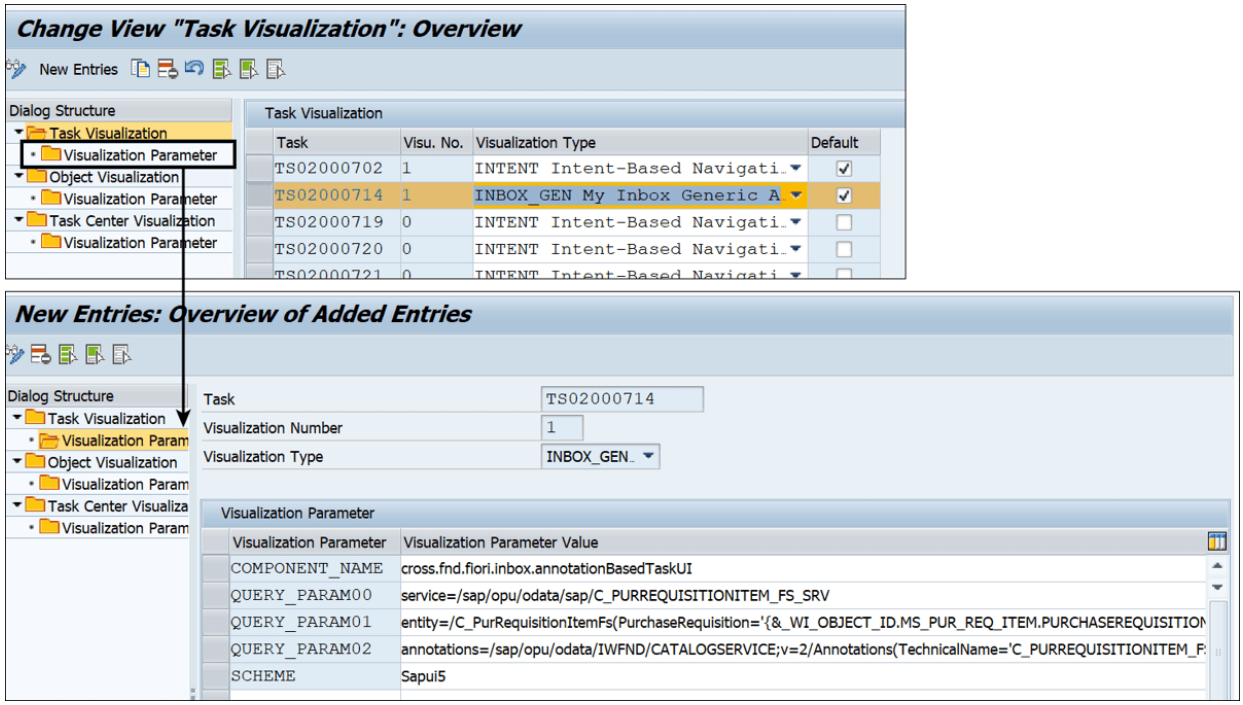


Figure 14.10 Transaction SWFVISU

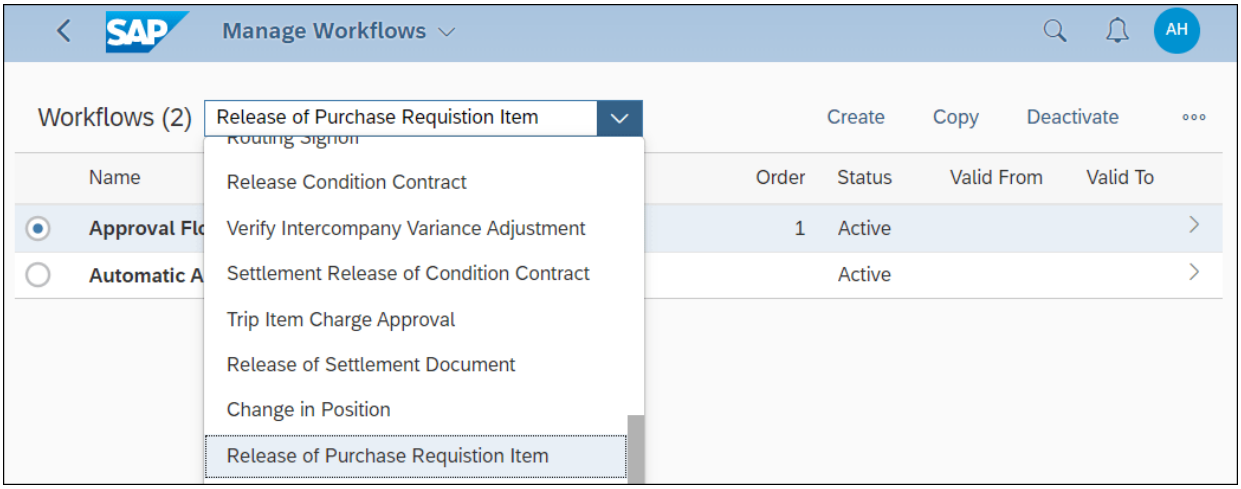


Figure 14.11 Manage Workflows Homepage View

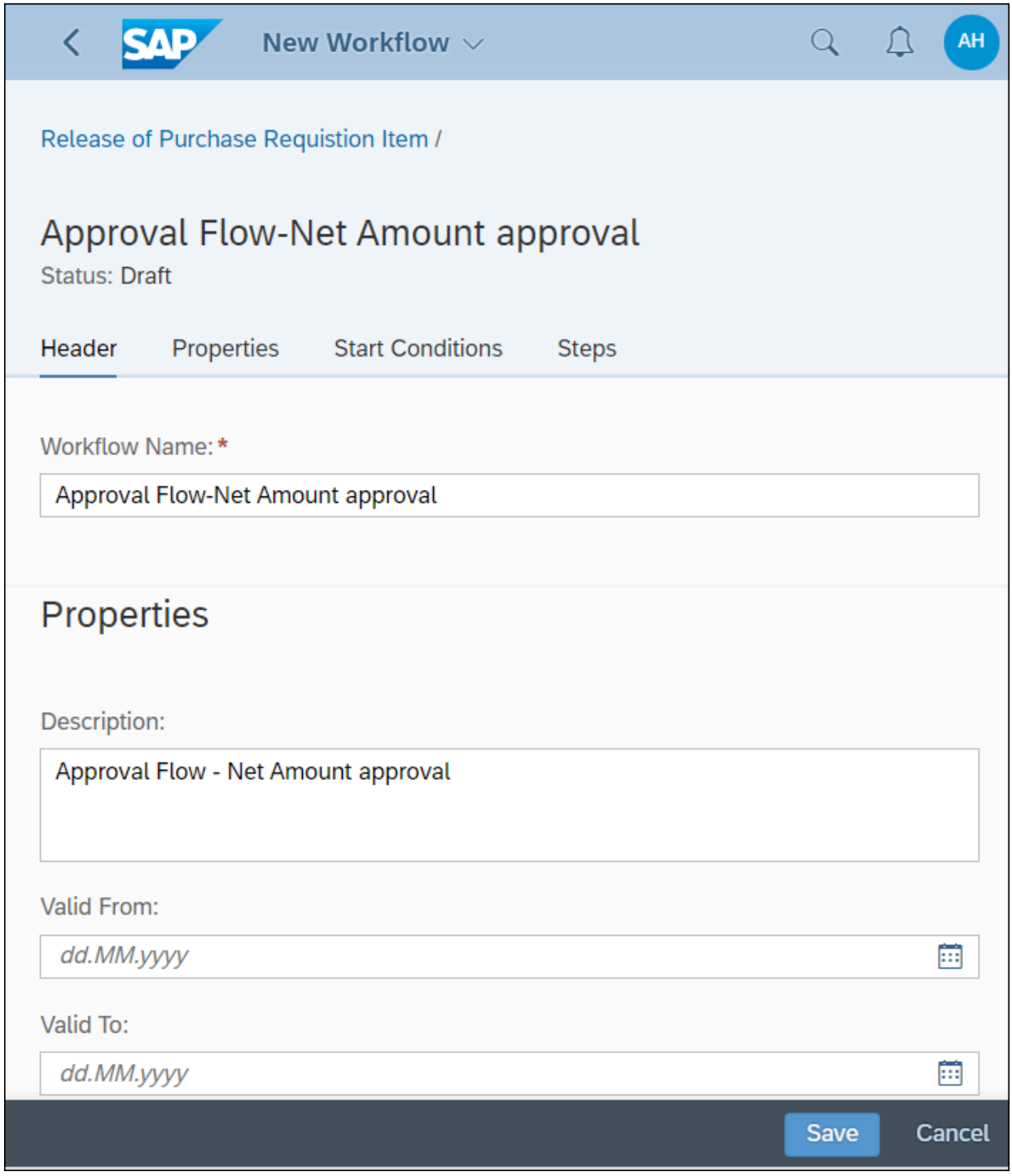


Figure 14.12 Manage Workflows: New Template

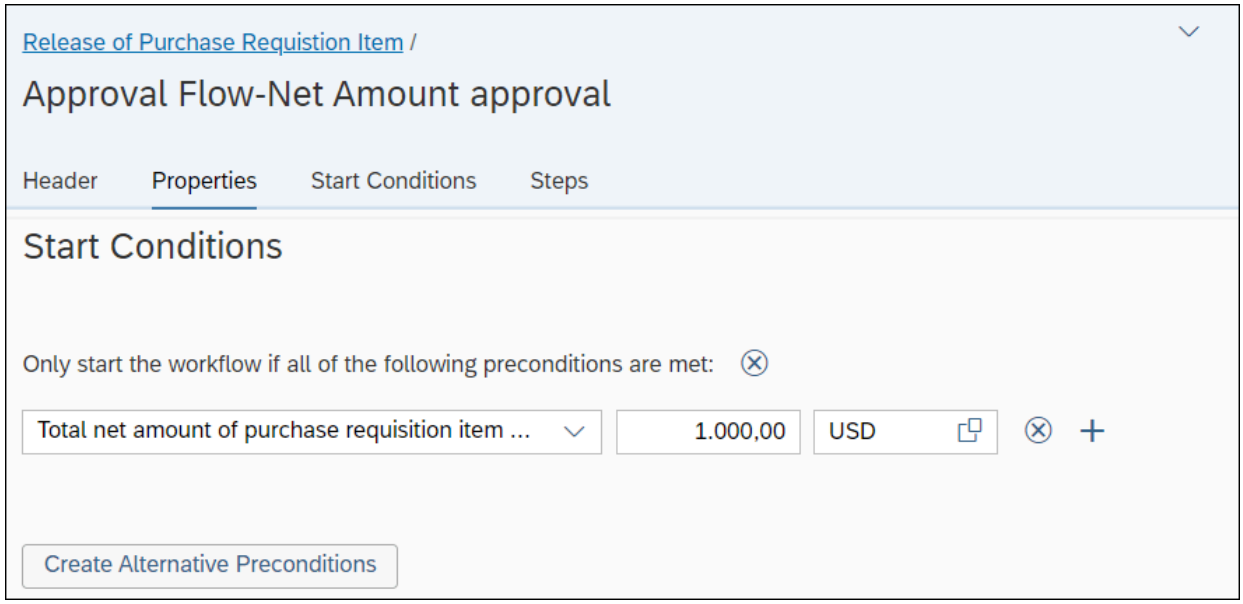


Figure 14.13 Manage Workflows: Start Conditions

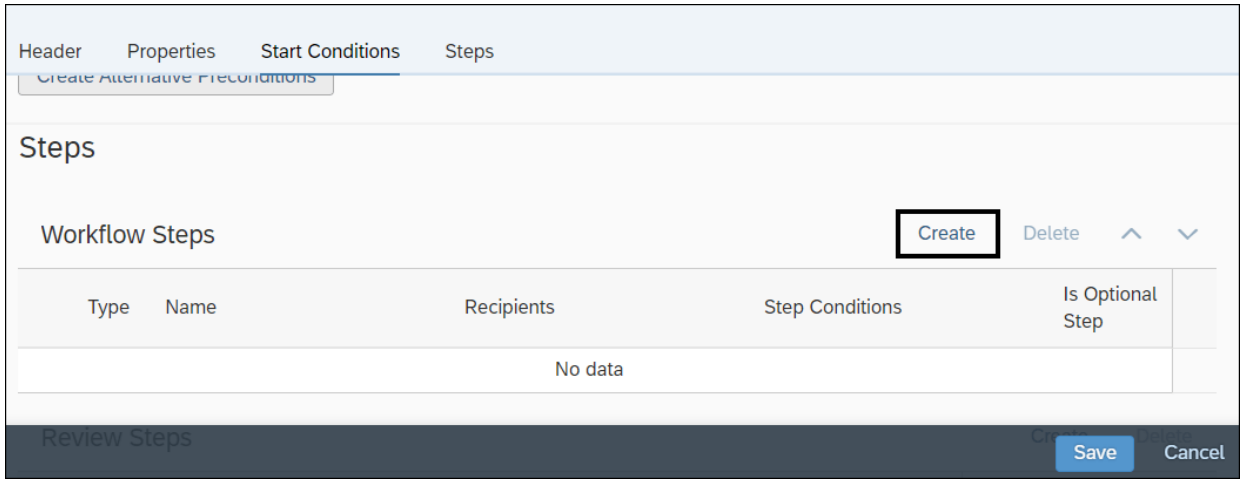


Figure 14.14 Manage Workflows: Add Steps

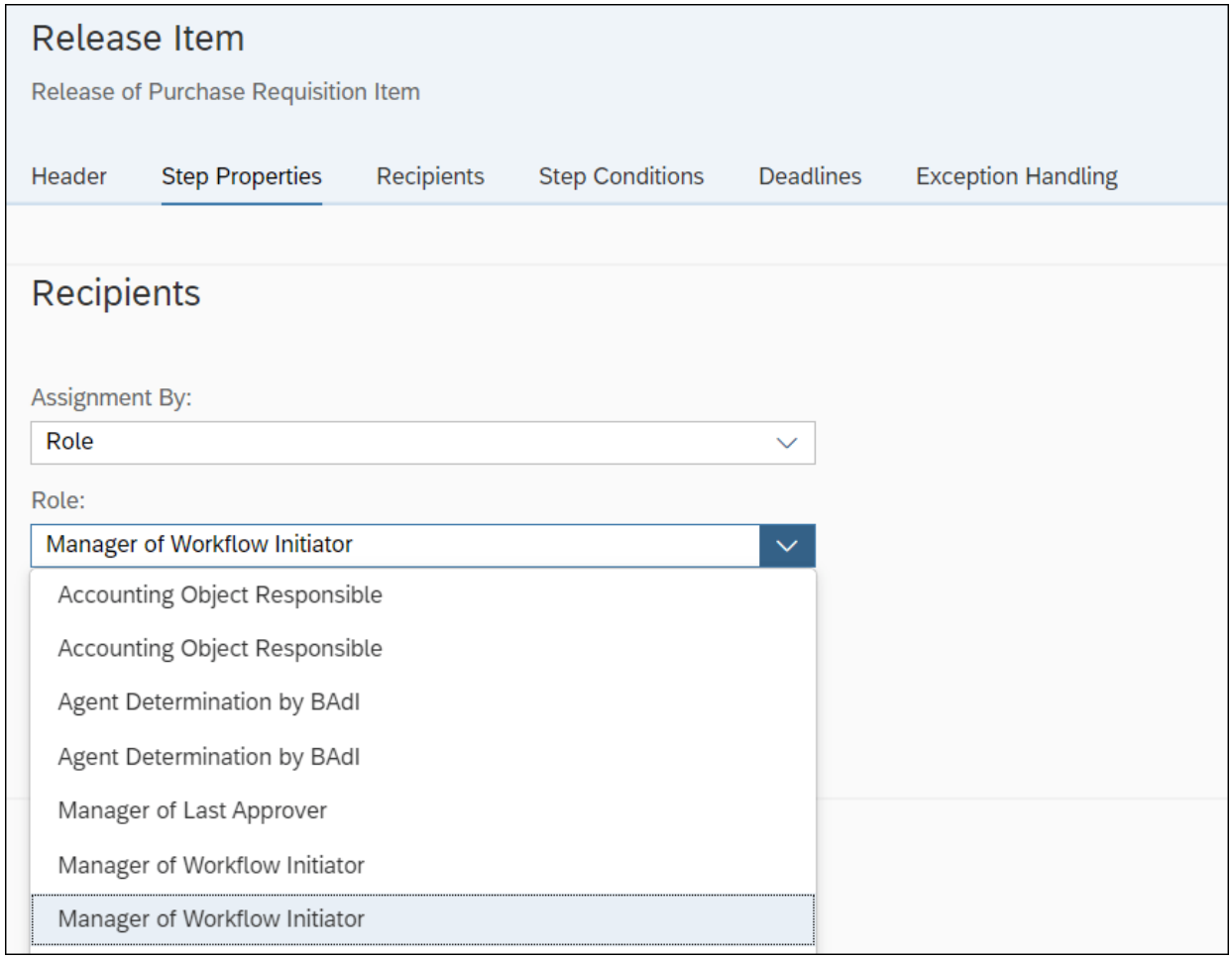


Figure 14.15 Manage Workflows: Step Details

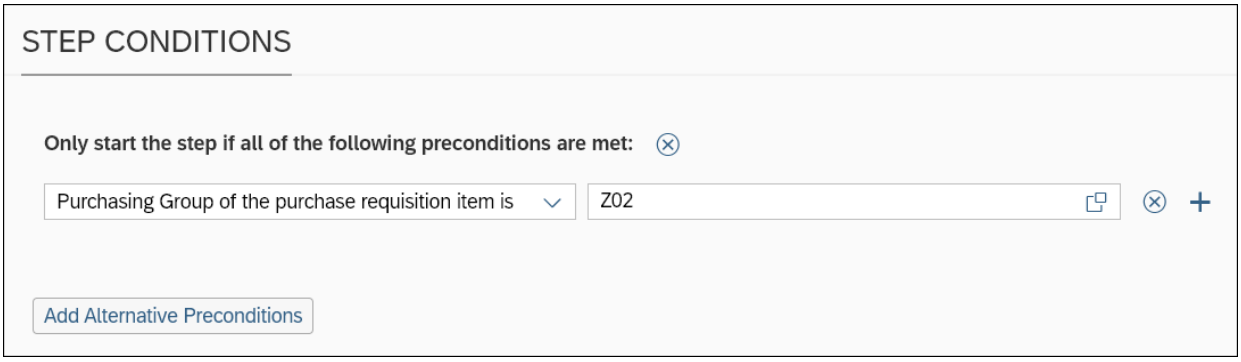


Figure 14.16 Manage Workflows: Step Conditions

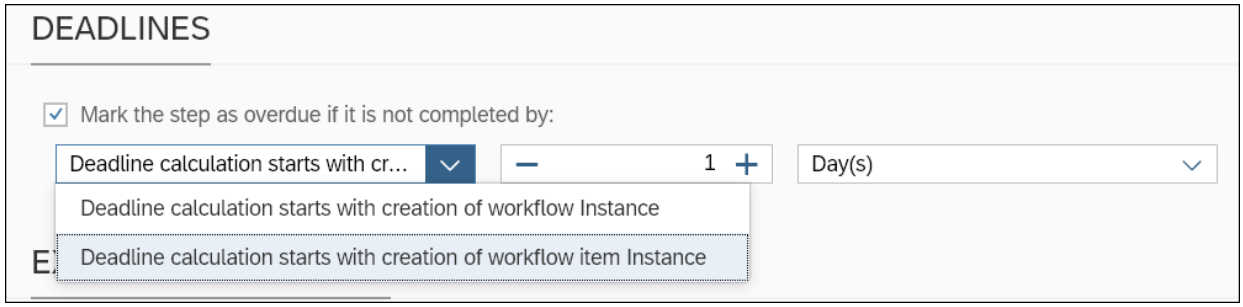


Figure 14.17 Manage Workflows: Deadlines

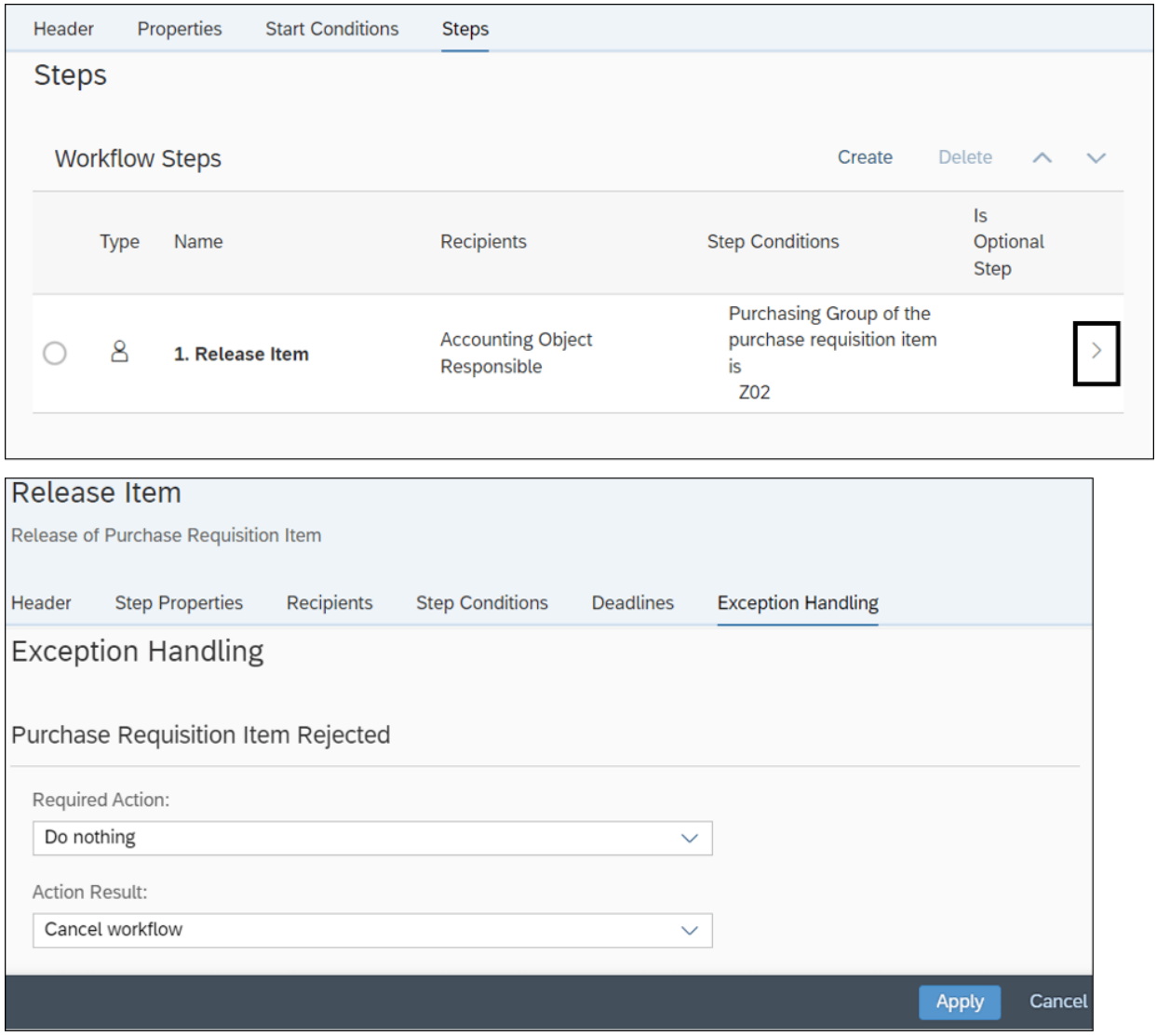


Figure 14.18 Manage Workflows: Exception Handling

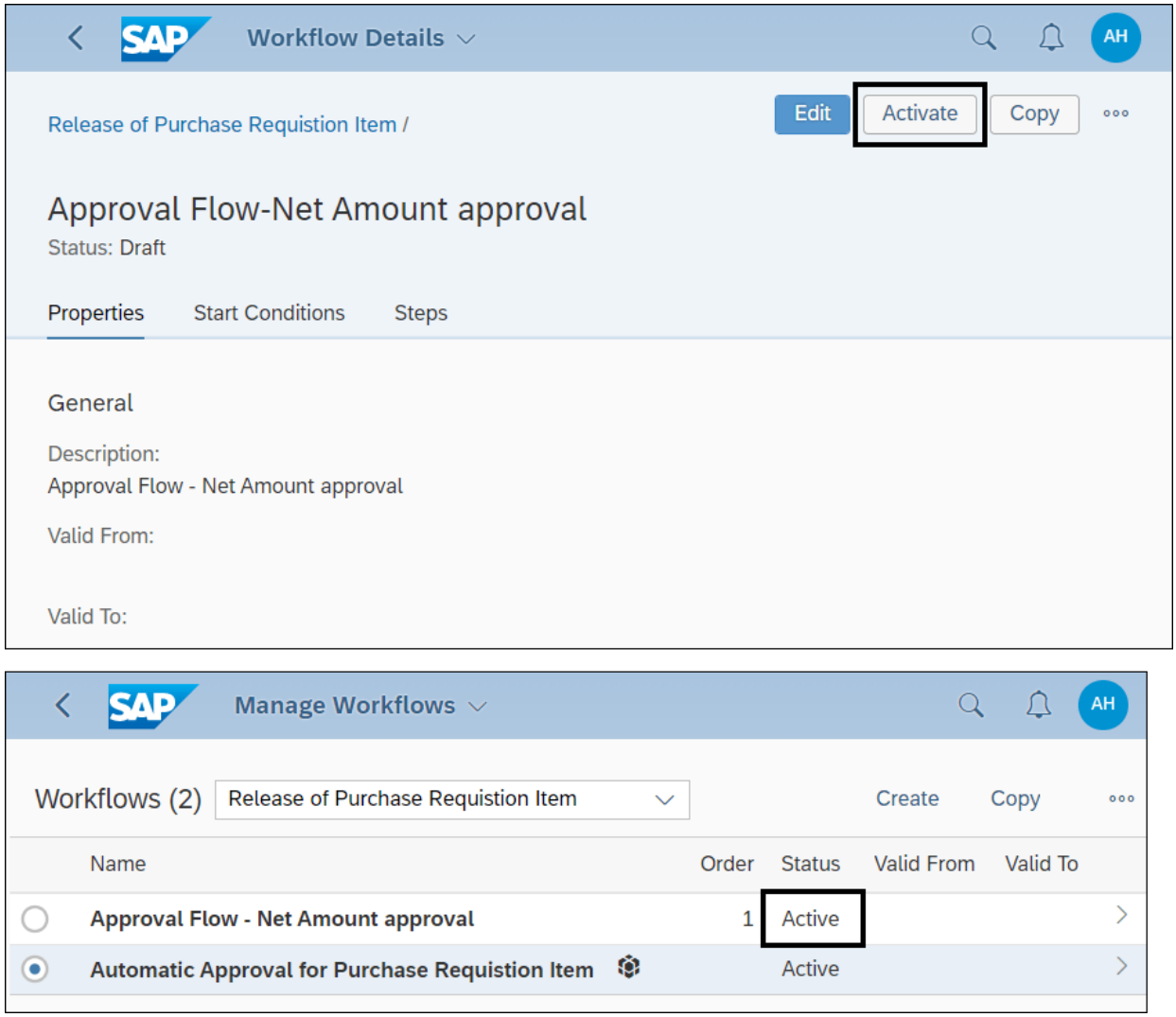


Figure 14.19 Manage Workflows: Activate the Workflow Template

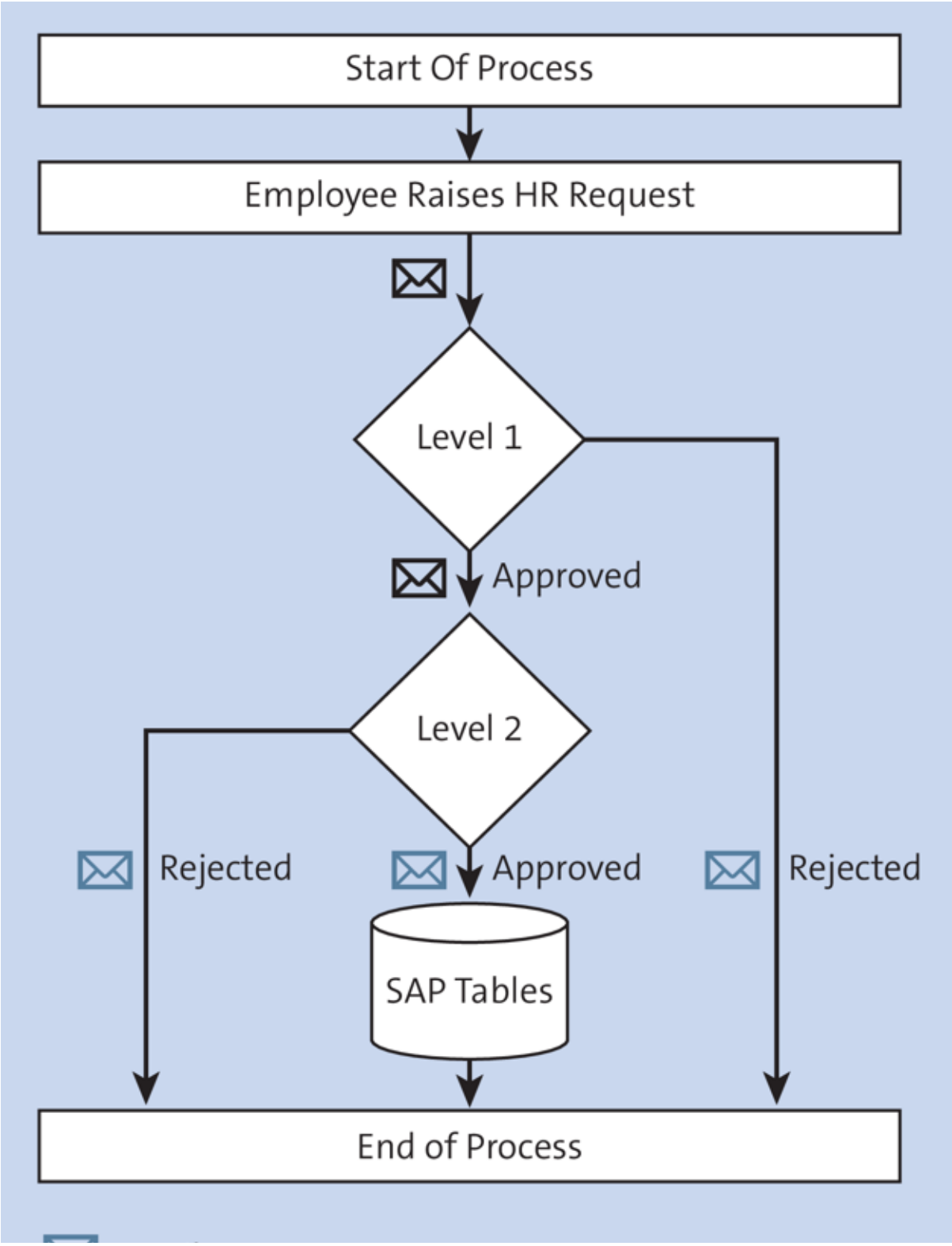

Start Of Process
Employee Raises HR Request
Level 1
Approved
Level 2
Rejected
Approved
Rejected
SAP Tables
End of Process

Figure 15.1 Flow Diagram for the Sample Use Case

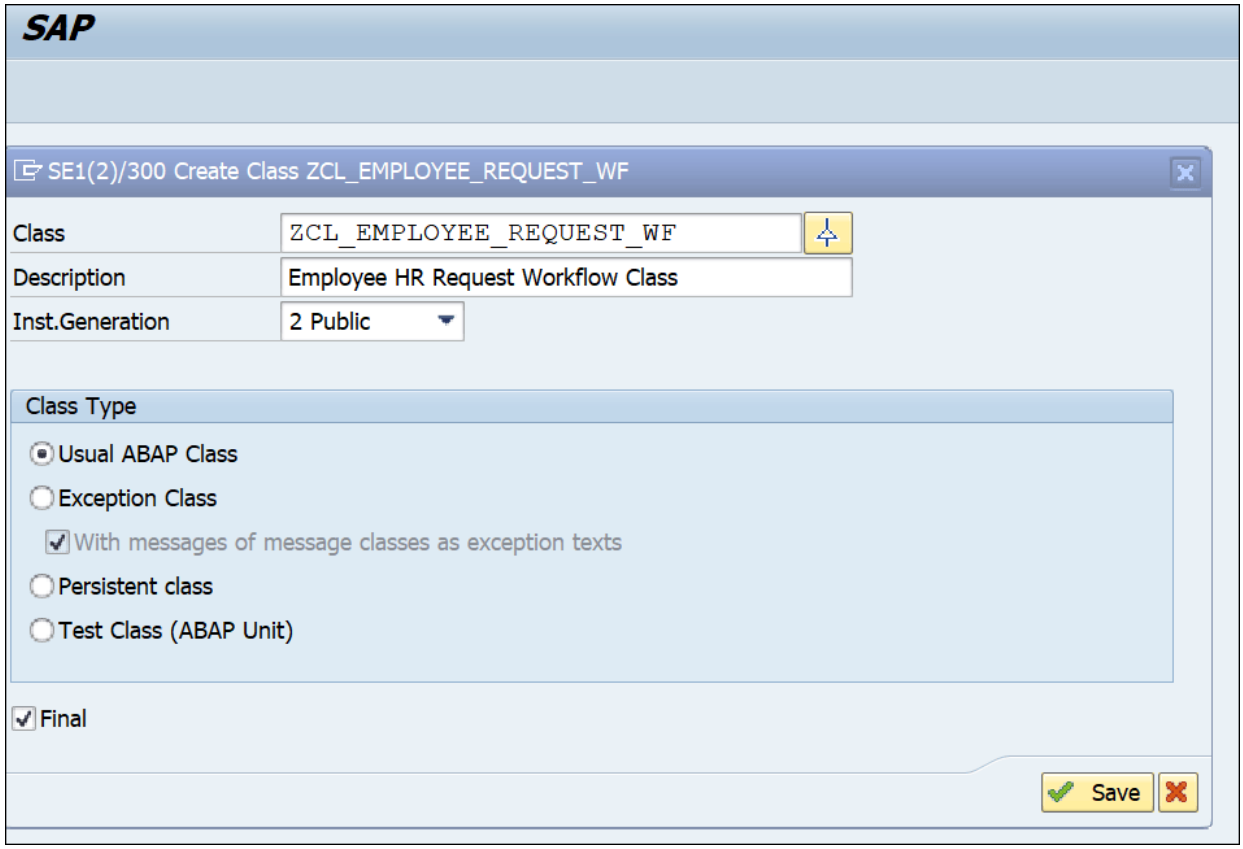


Figure 15.2 Custom Class for the Sample Use Case

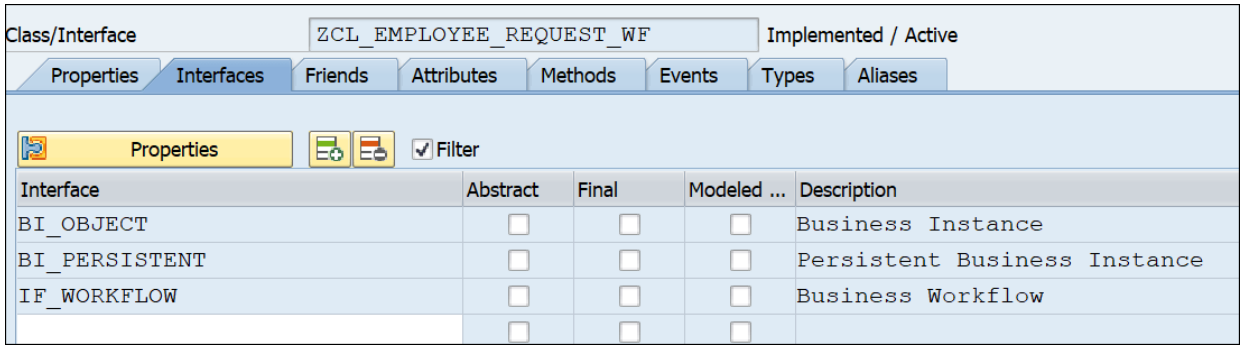


Figure 15.3 Interface for the Sample Use Case

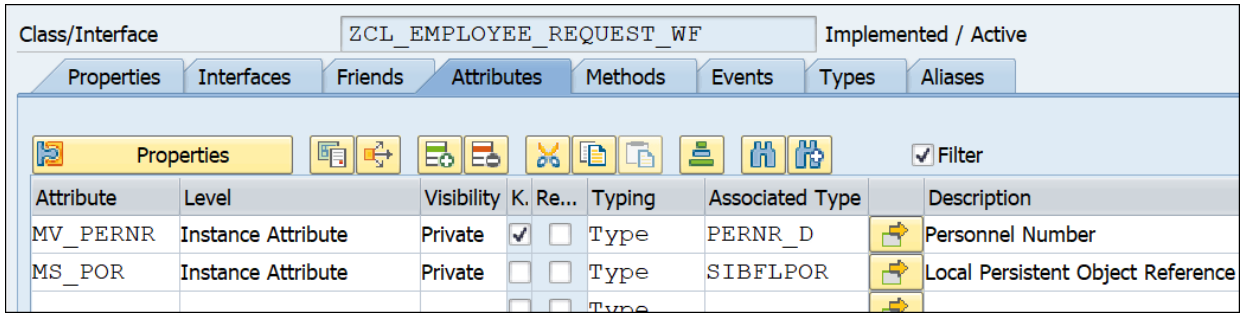


Figure 15.4 Attributes of the Workflow Class for the Sample Use Case

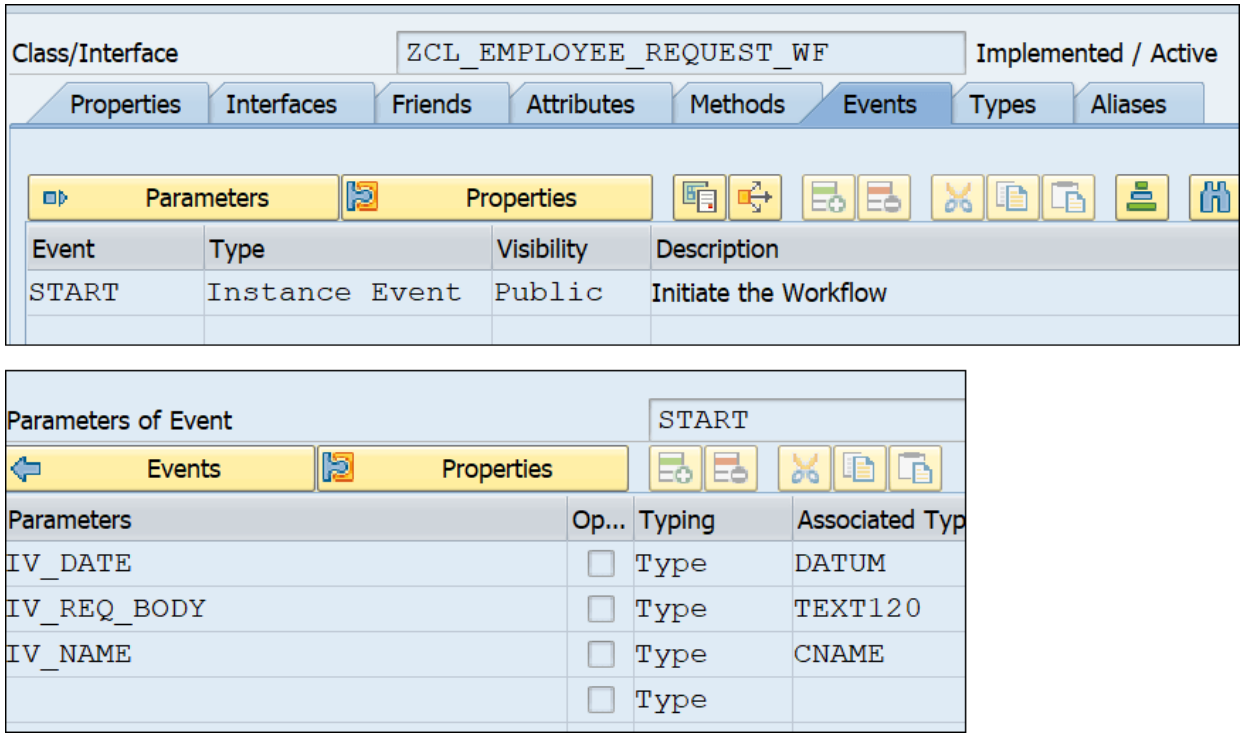


Figure 15.5 Event in the Custom Class for the Sample Use Case

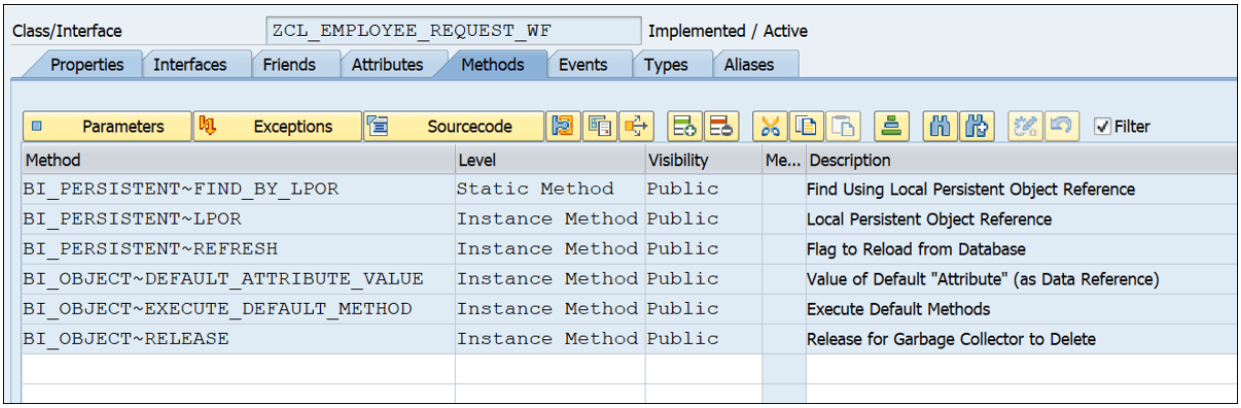


Figure 15.6 Methods of the Workflow Class of the Use Case

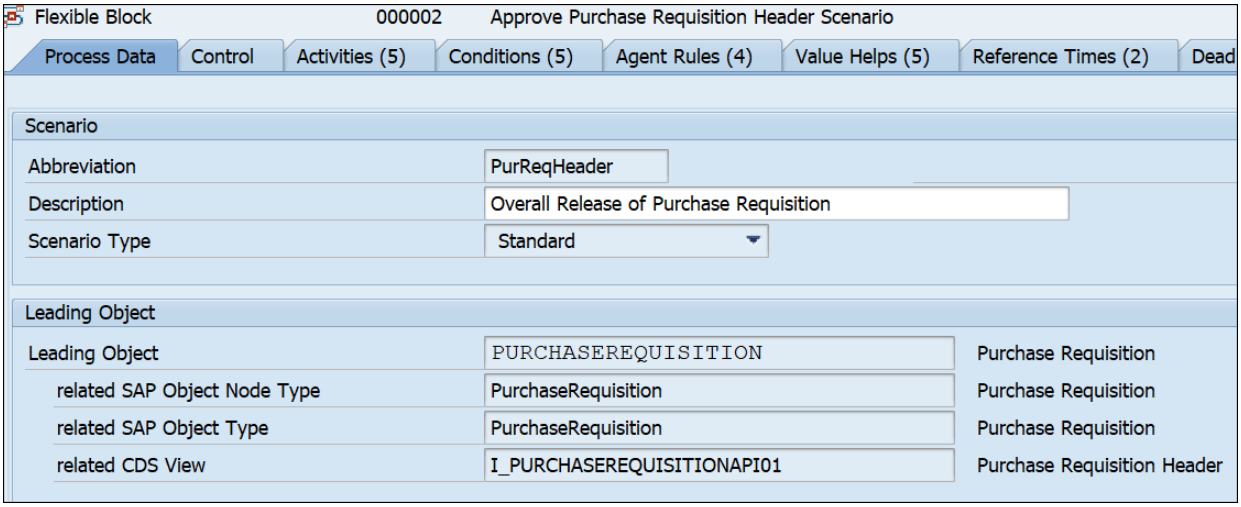


Figure 15.7 Leading Object and Related CDS View for Purchase Requisition

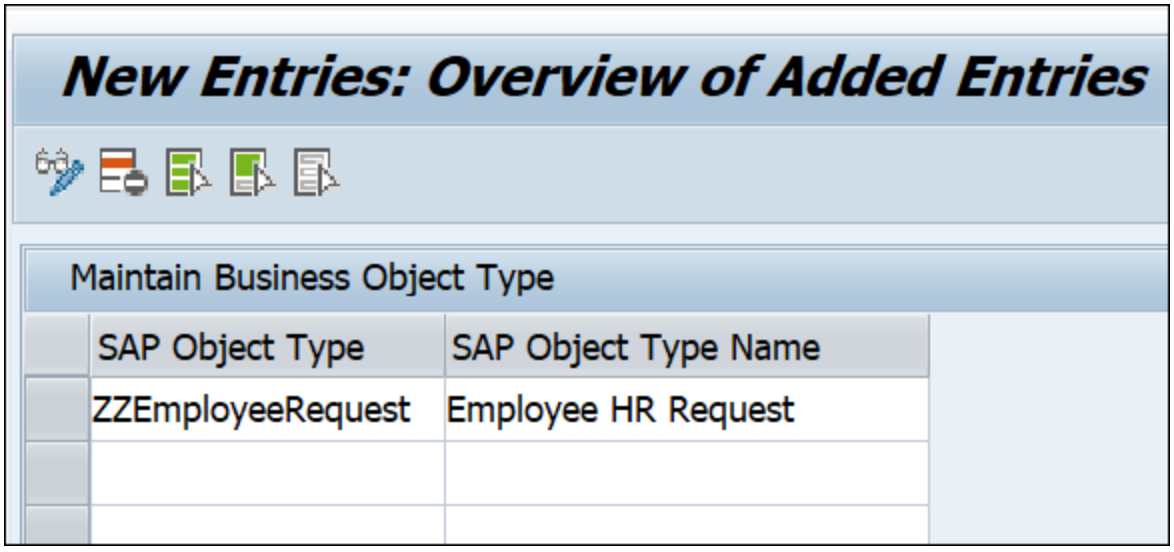


Figure 15.8 Maintain the Business Object Type

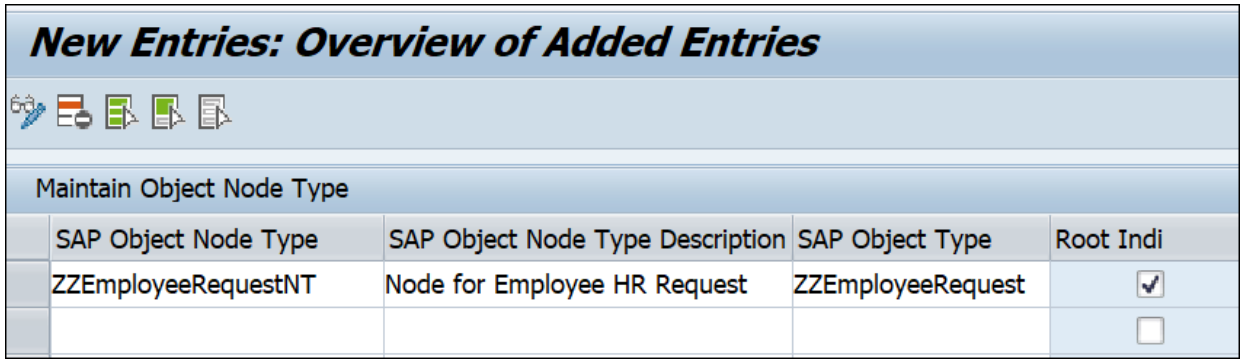


Figure 15.9 Maintain Business Object Node Type.

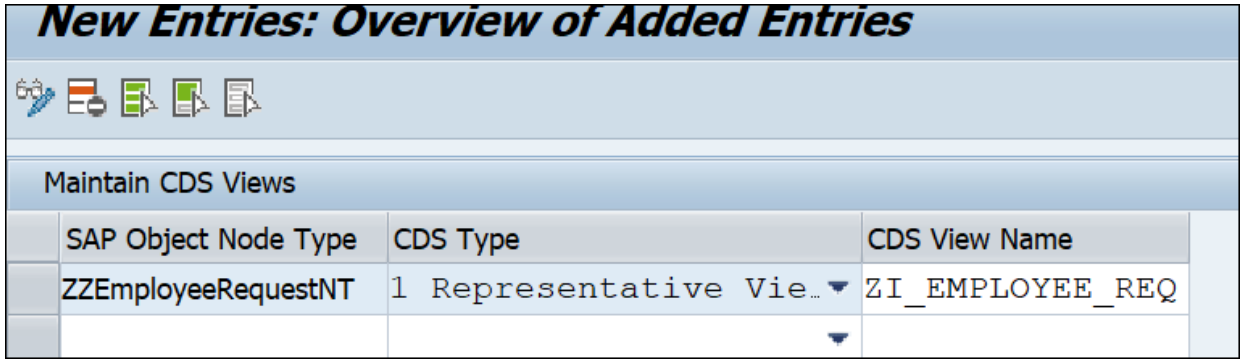


Figure 15.10 Maintain CDS Views

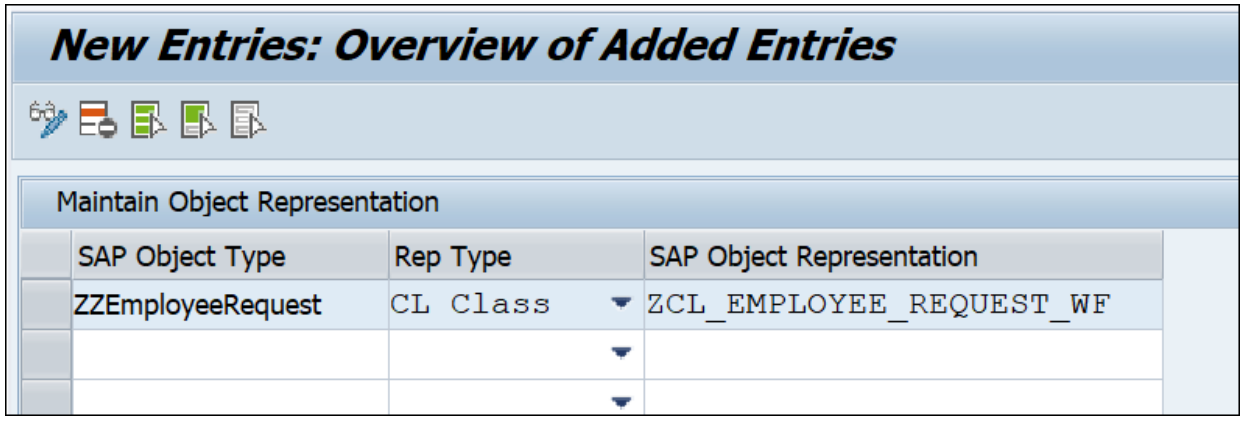


Figure 15.11 Maintain Object Representation

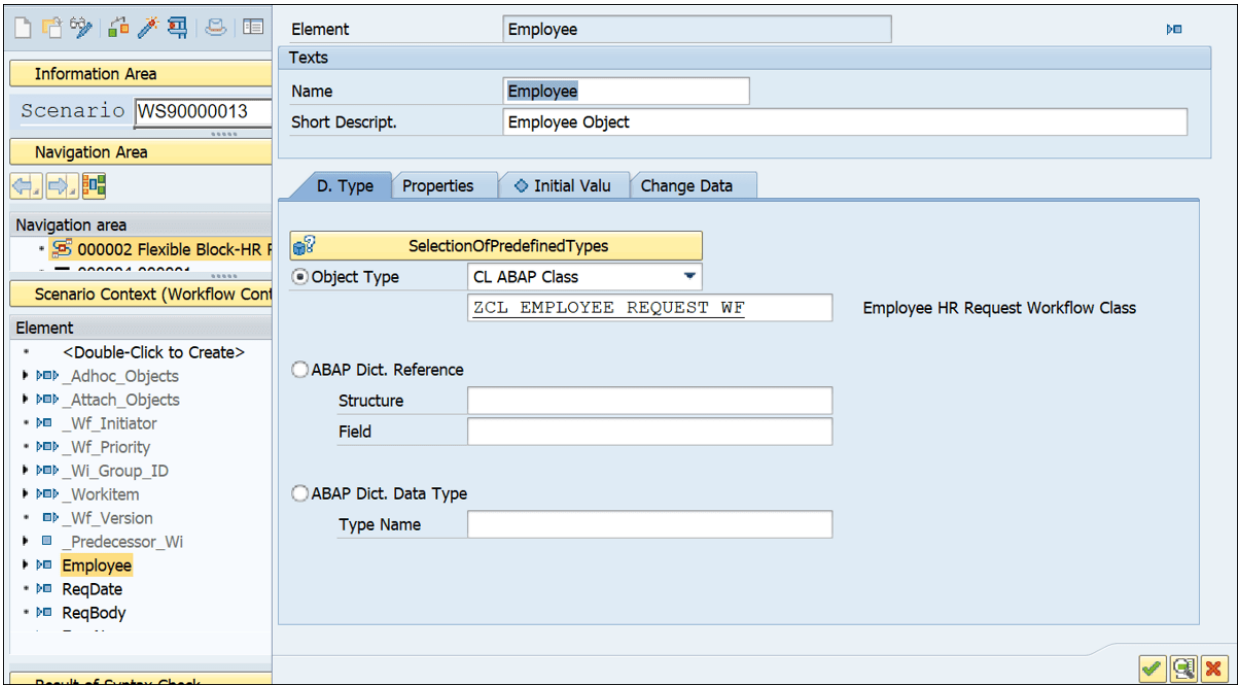


Figure 15.12 Scenario Context in the Workflow Builder

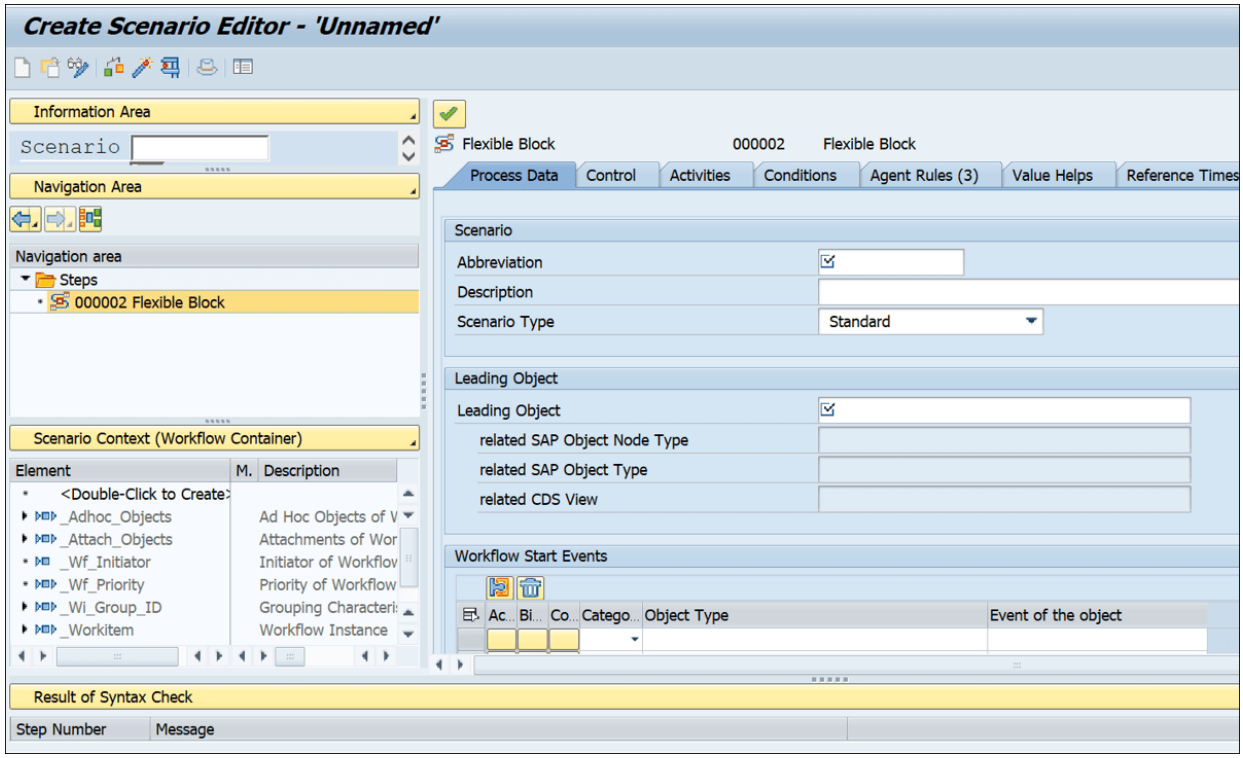


Figure 15.13 Creation of a New Workflow Scenario for the Sample Use Case

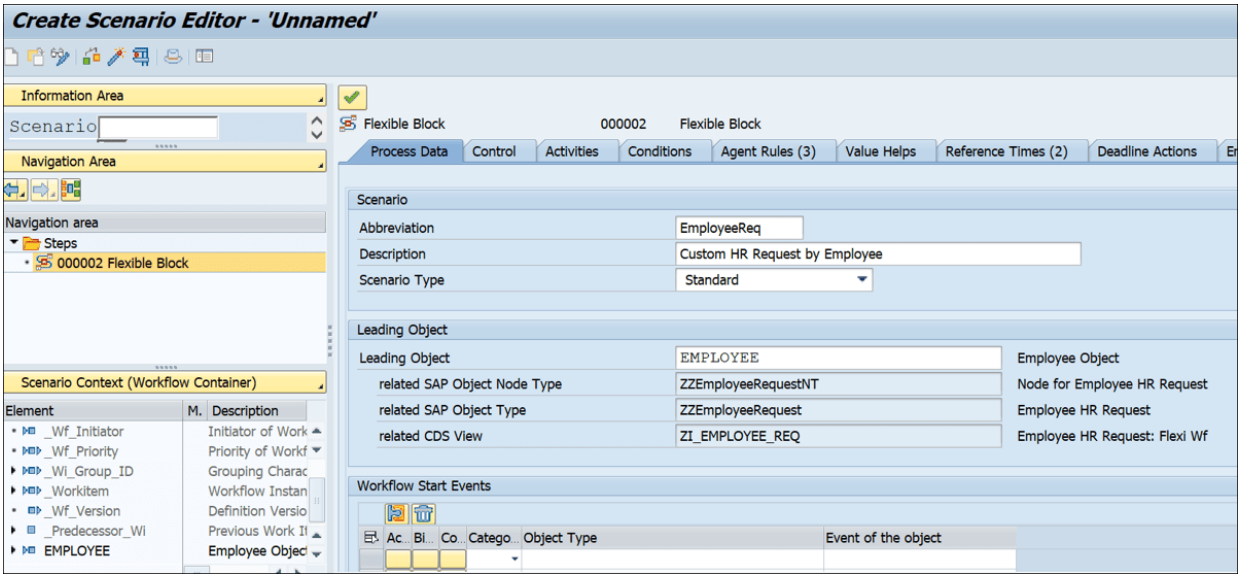


Figure 15.14 Process Data in a Flexible Block of the Sample Workflow Scenario

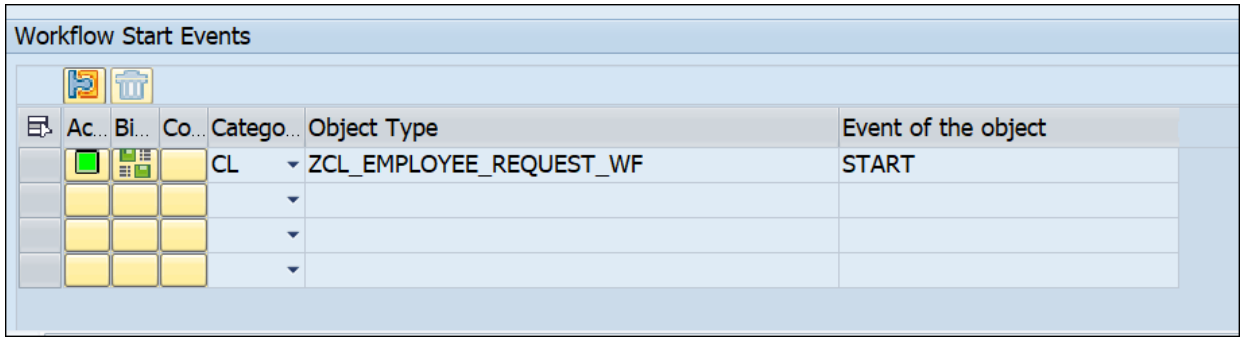


Figure 15.15 Start Event for the Custom Use Case

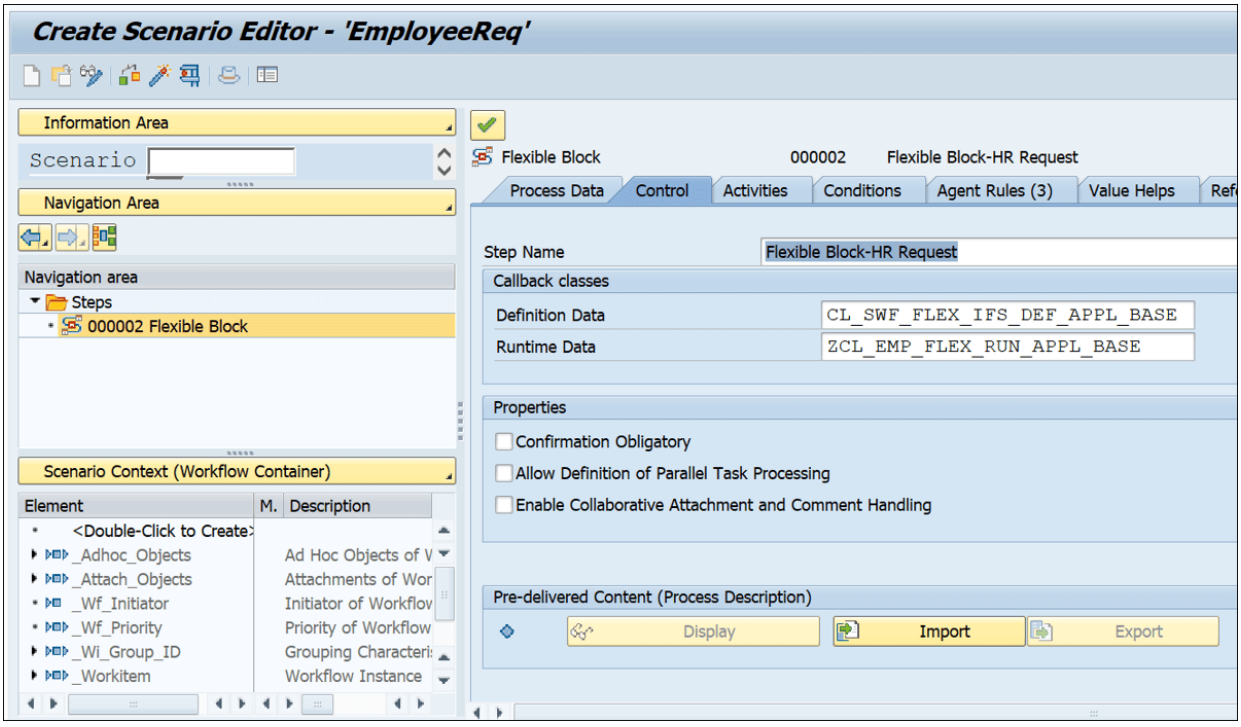


Figure 15.16 Control Tab of the Flexible Block

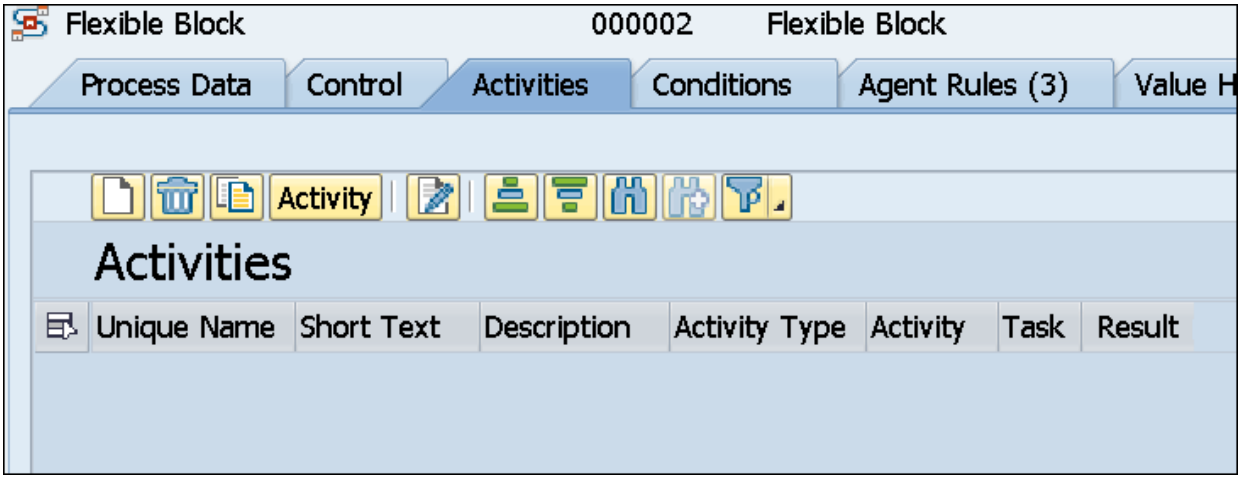


Figure 15.17 Activities in the Flexible Block of the Workflow Scenario

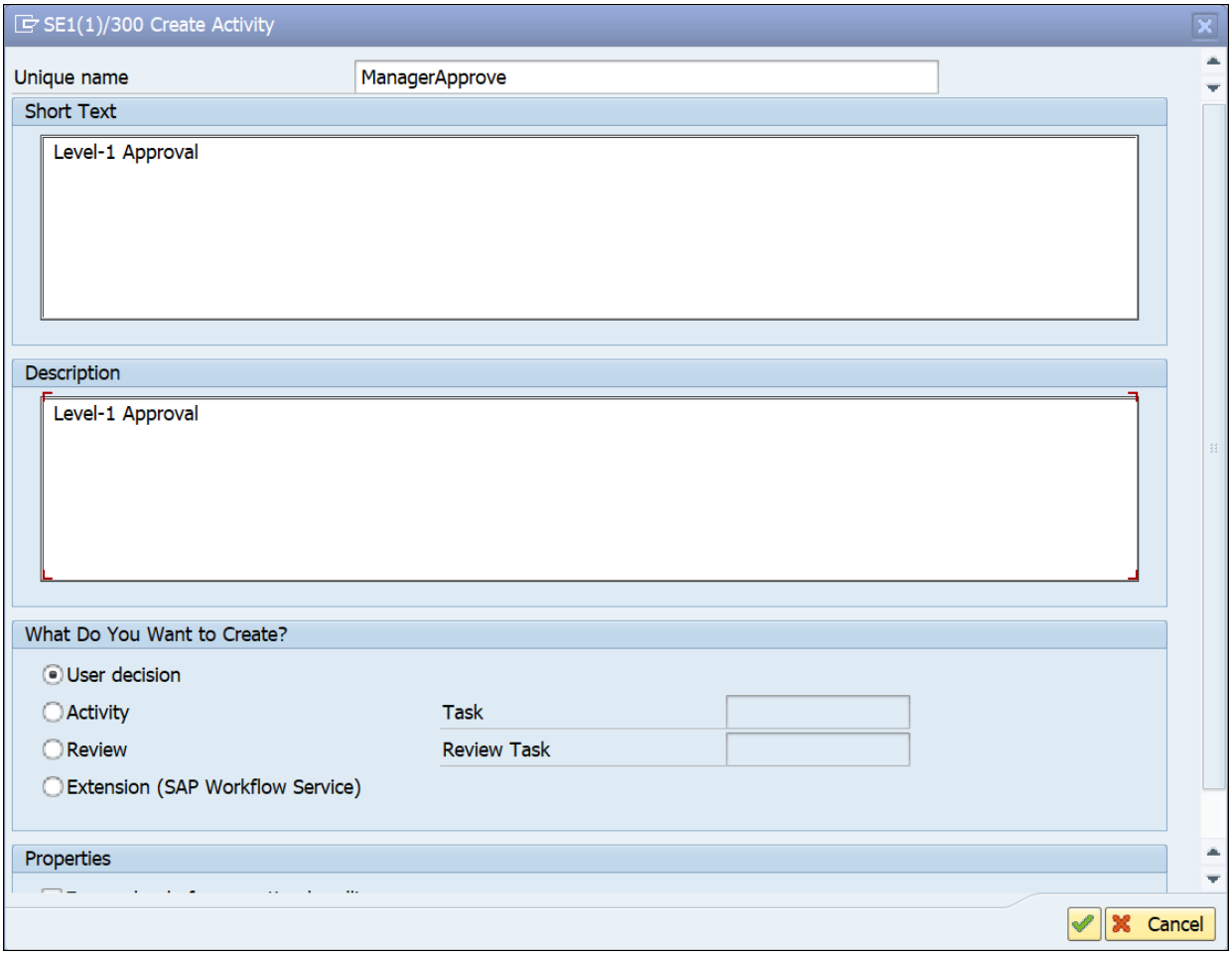


Figure 15.18 Create Activity Screen

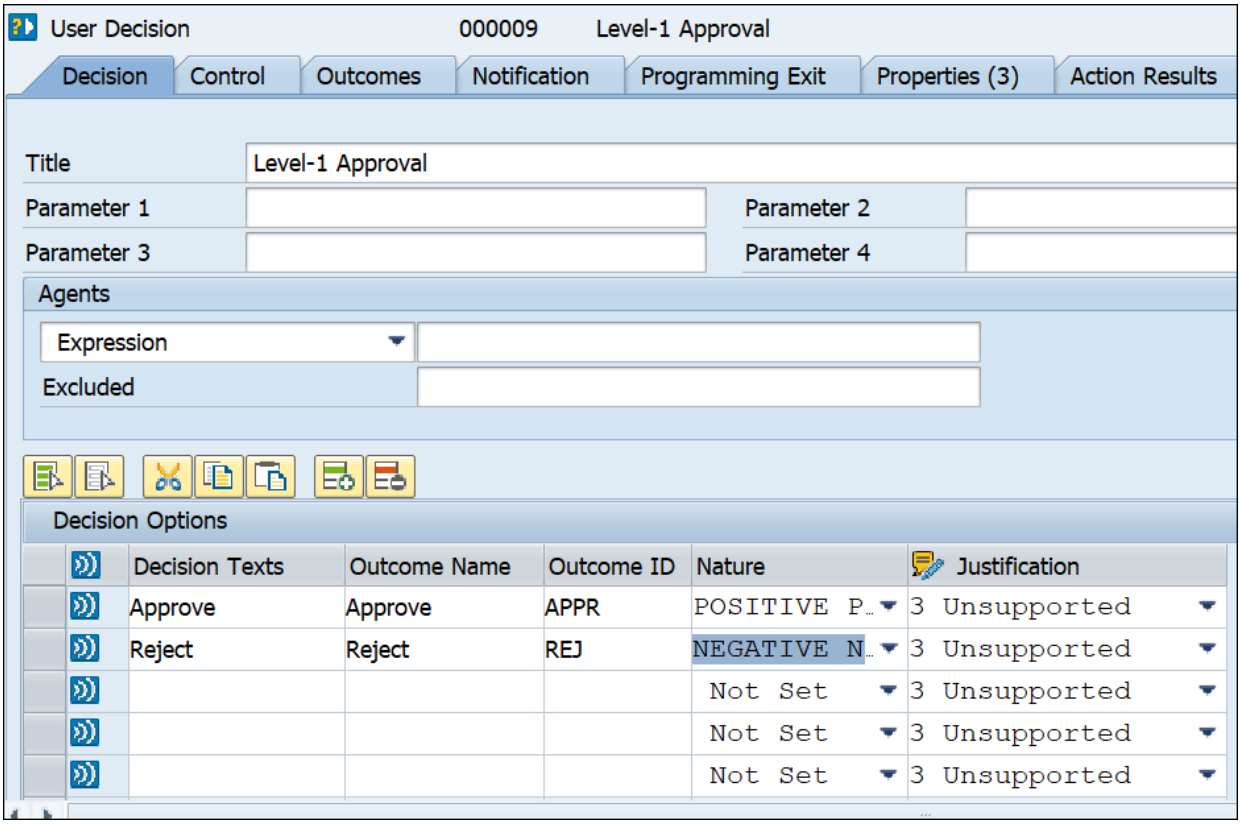


Figure 15.19 User Decision Activity for Line Manager (Level-1)

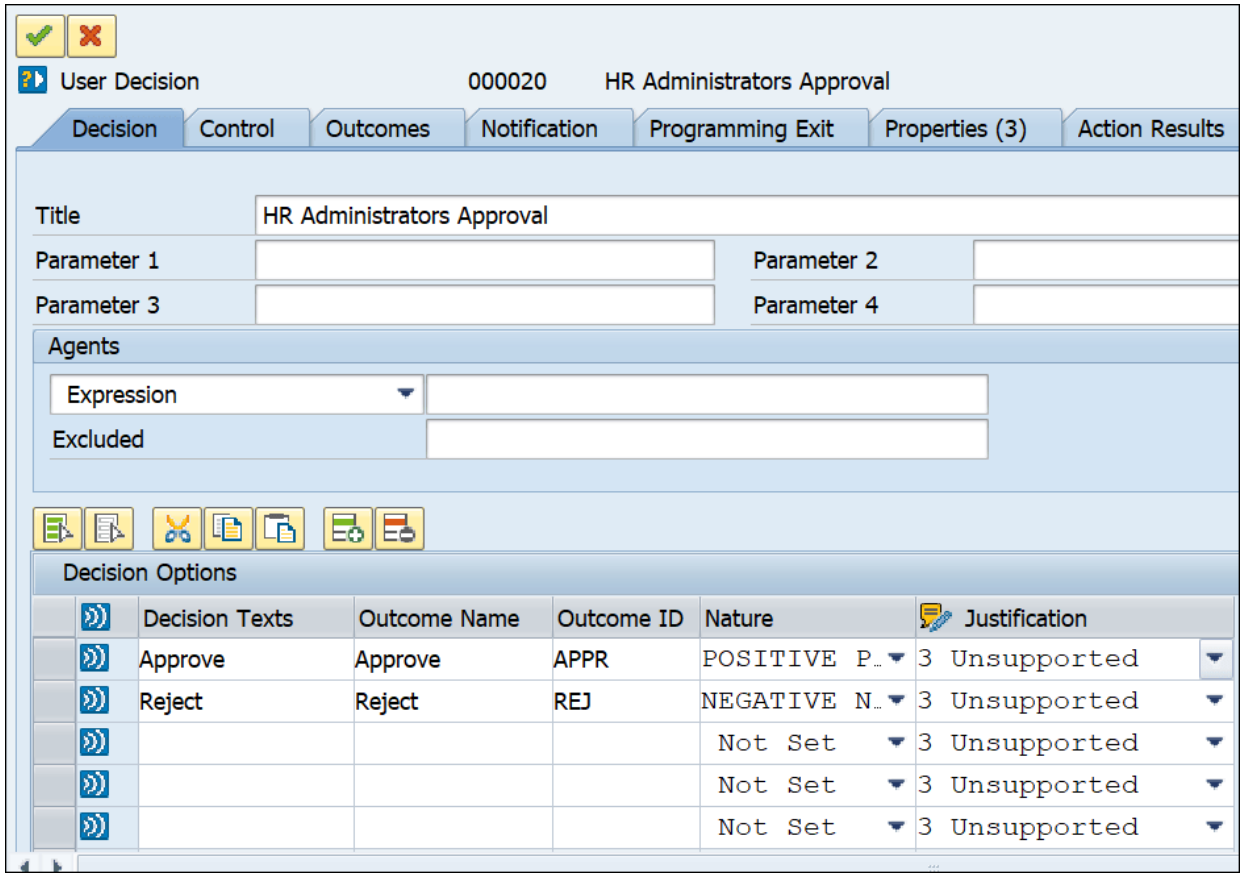


Figure 15.20 User Decision Activity for HR Administrator

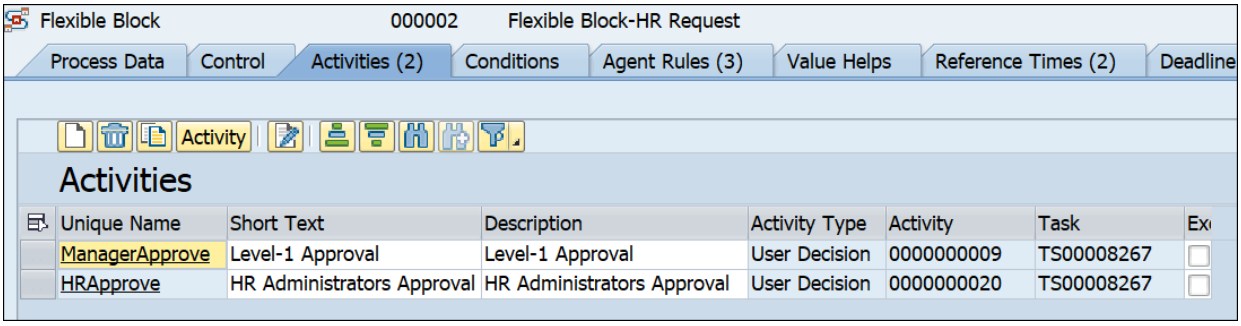


Figure 15.21 Activities Overview Screen in the Flexible Block

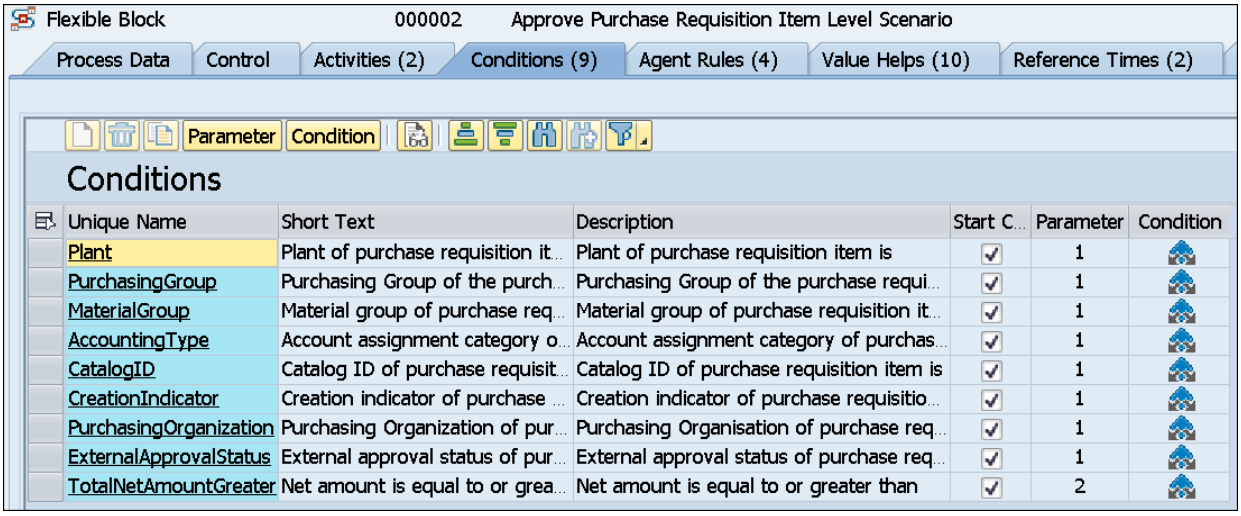


Figure 15.22 Conditions in the Approve Purchase Requisition Item Level Standard Workflow Scenario

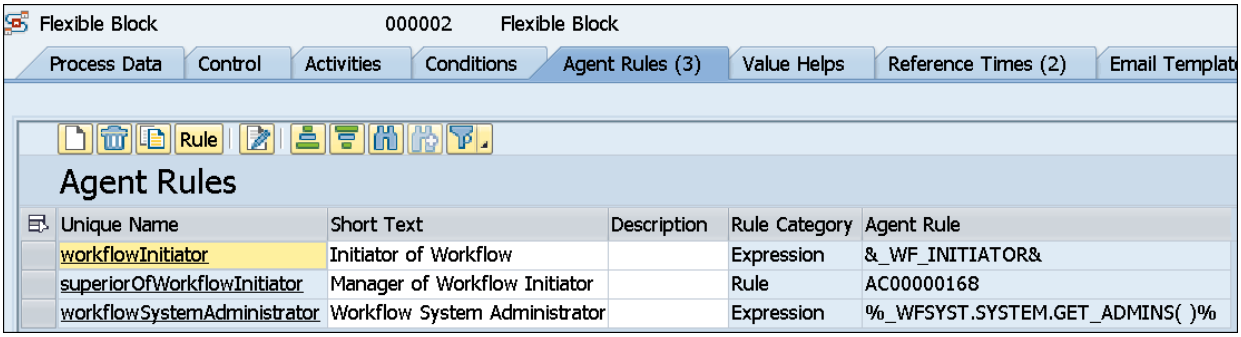


Figure 15.23 Standard Agent Rules in the Flexible Block Defaulted while Creating a Custom Scenario

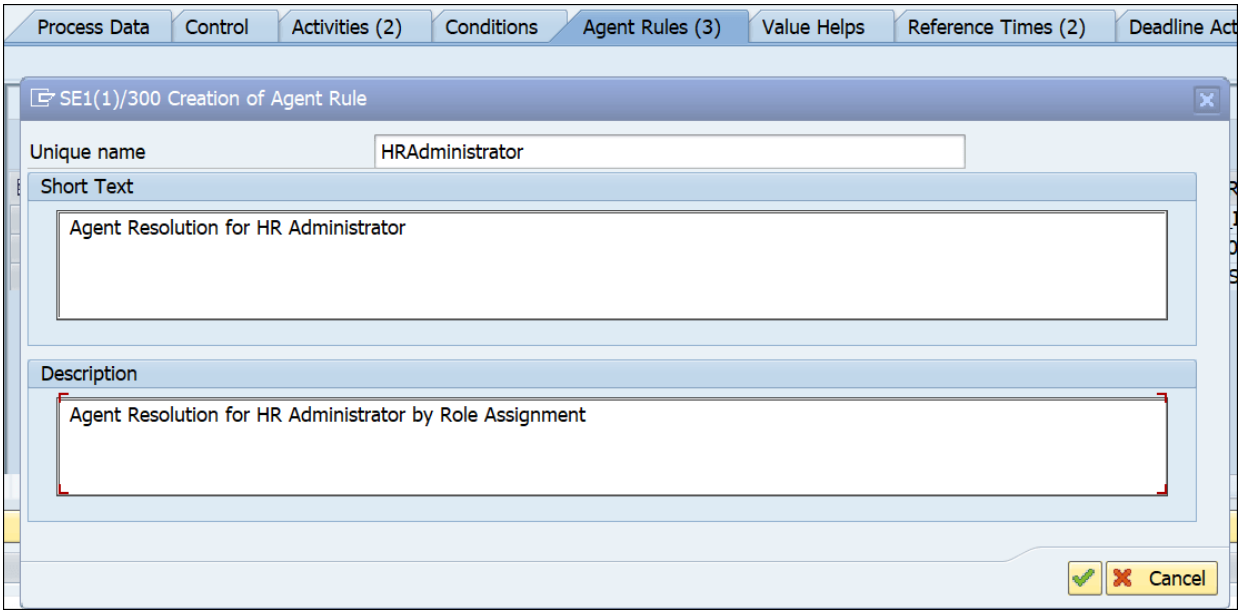


Figure 15.24 Create New Agent Screen

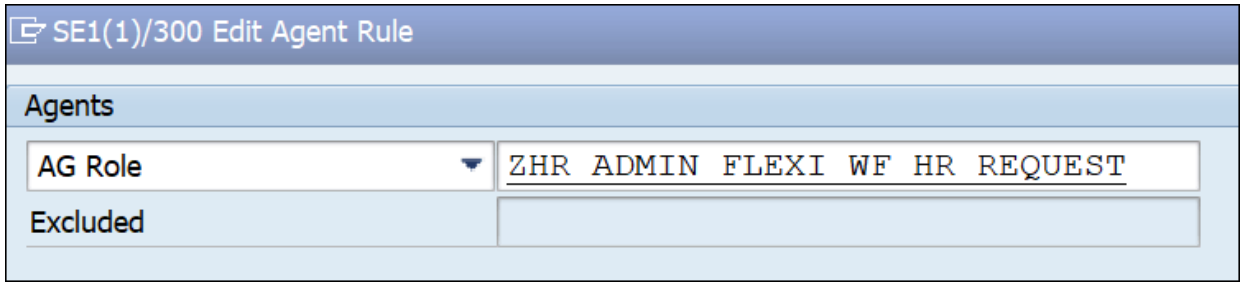


Figure 15.25 Role-Based Agent Rule Configuration

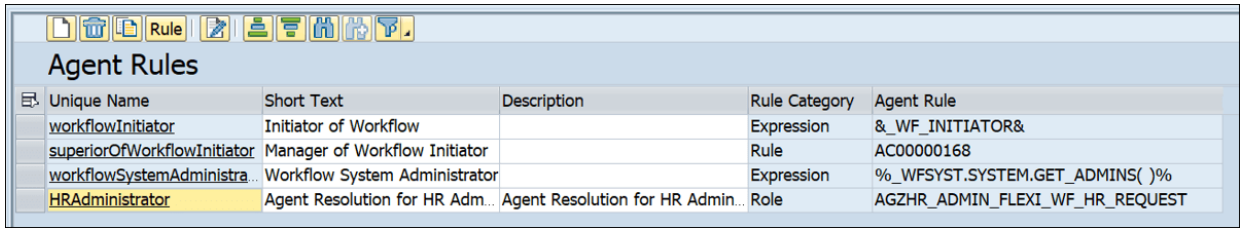


Figure 15.26 Agent Rules

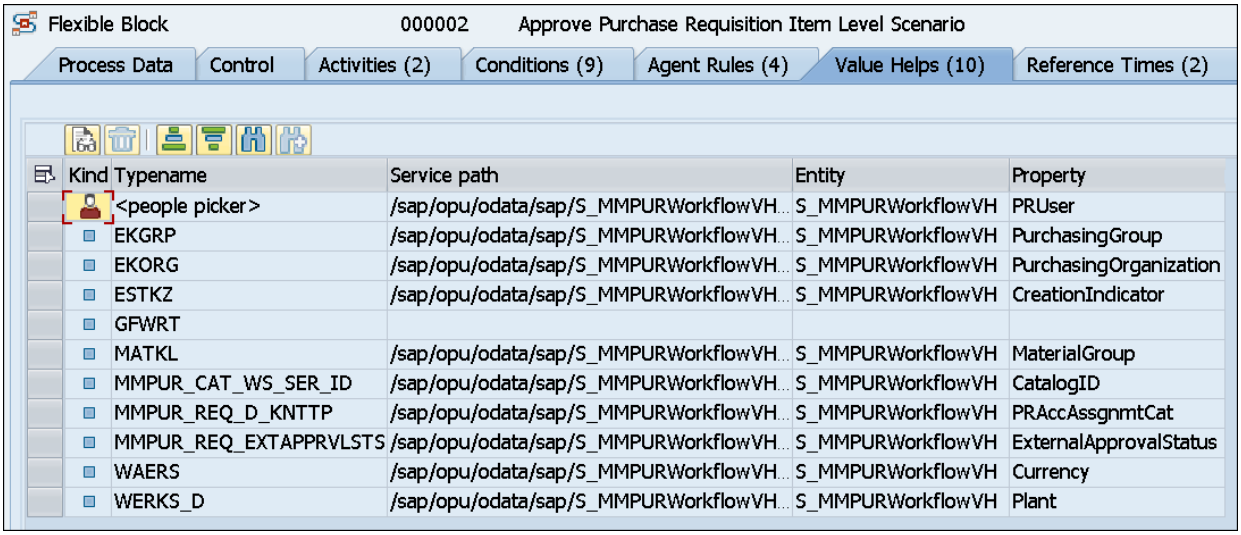


Figure 15.27 Value Helps: Approve Purchase Requisition Item Level Scenario

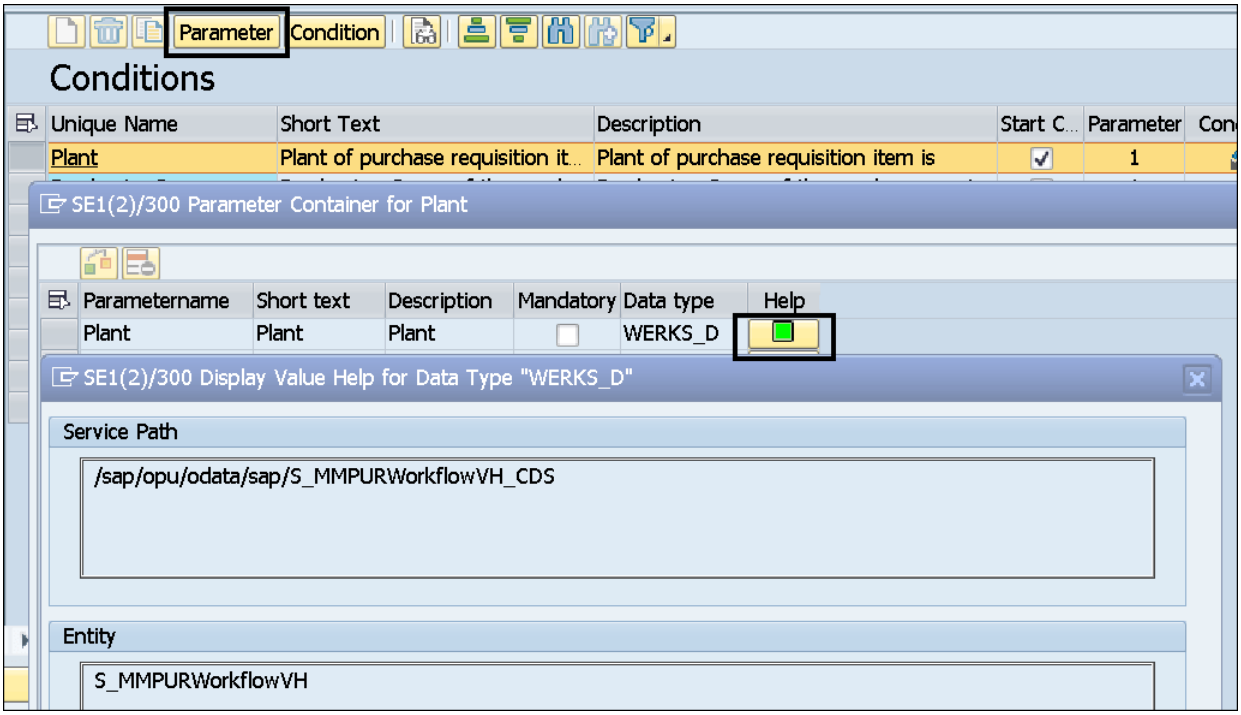


Figure 15.28 Value Help via Conditions

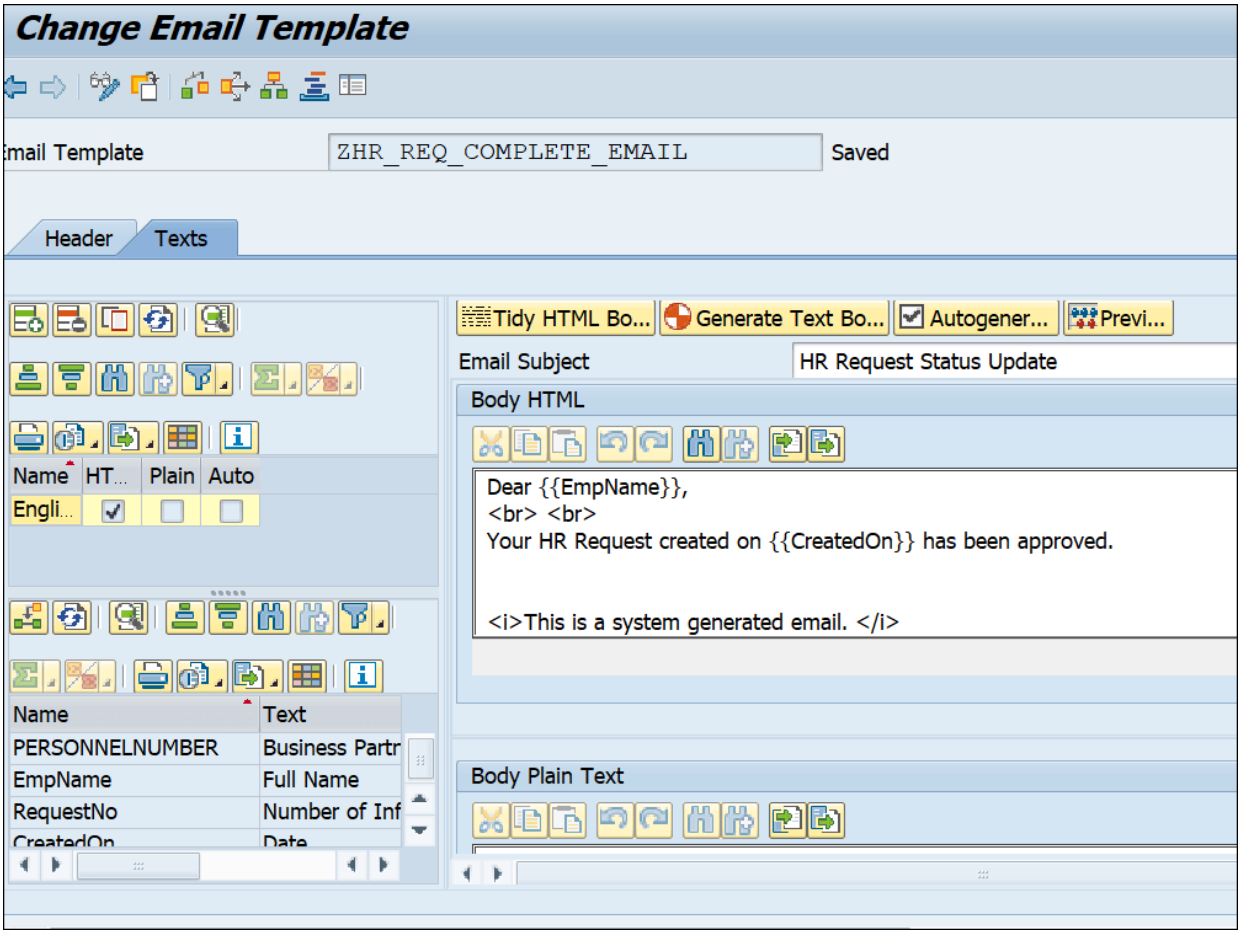


Figure 15.29 Email Templates Maintenance

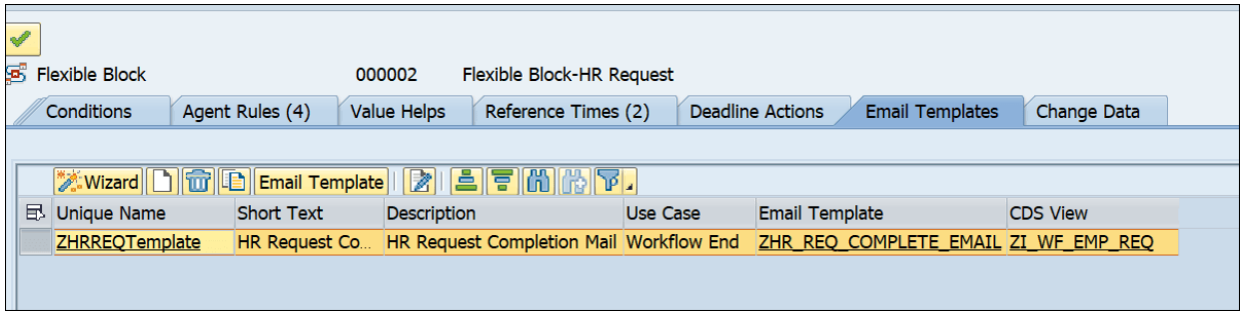


Figure 15.30 Email Templates in the Workflow Scenario

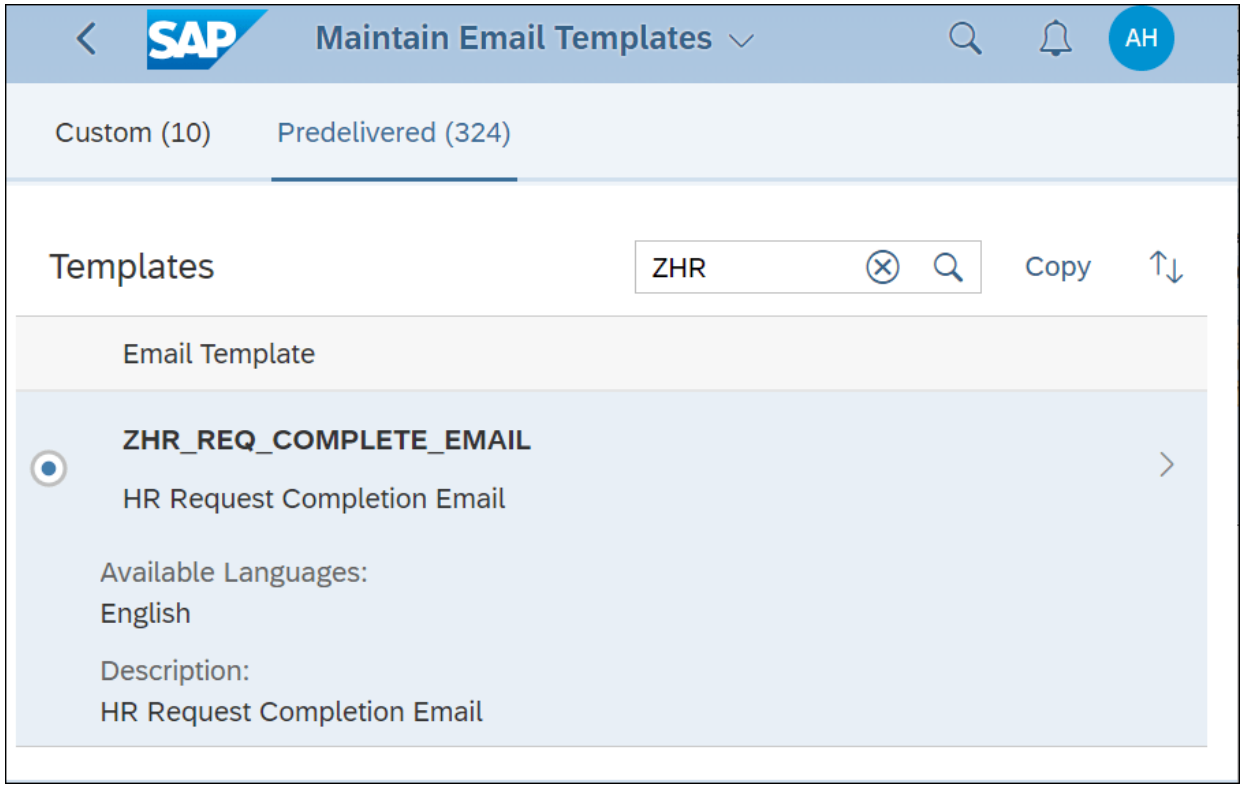


Figure 15.31 Email Template from Custom Workflow Scenario Definition

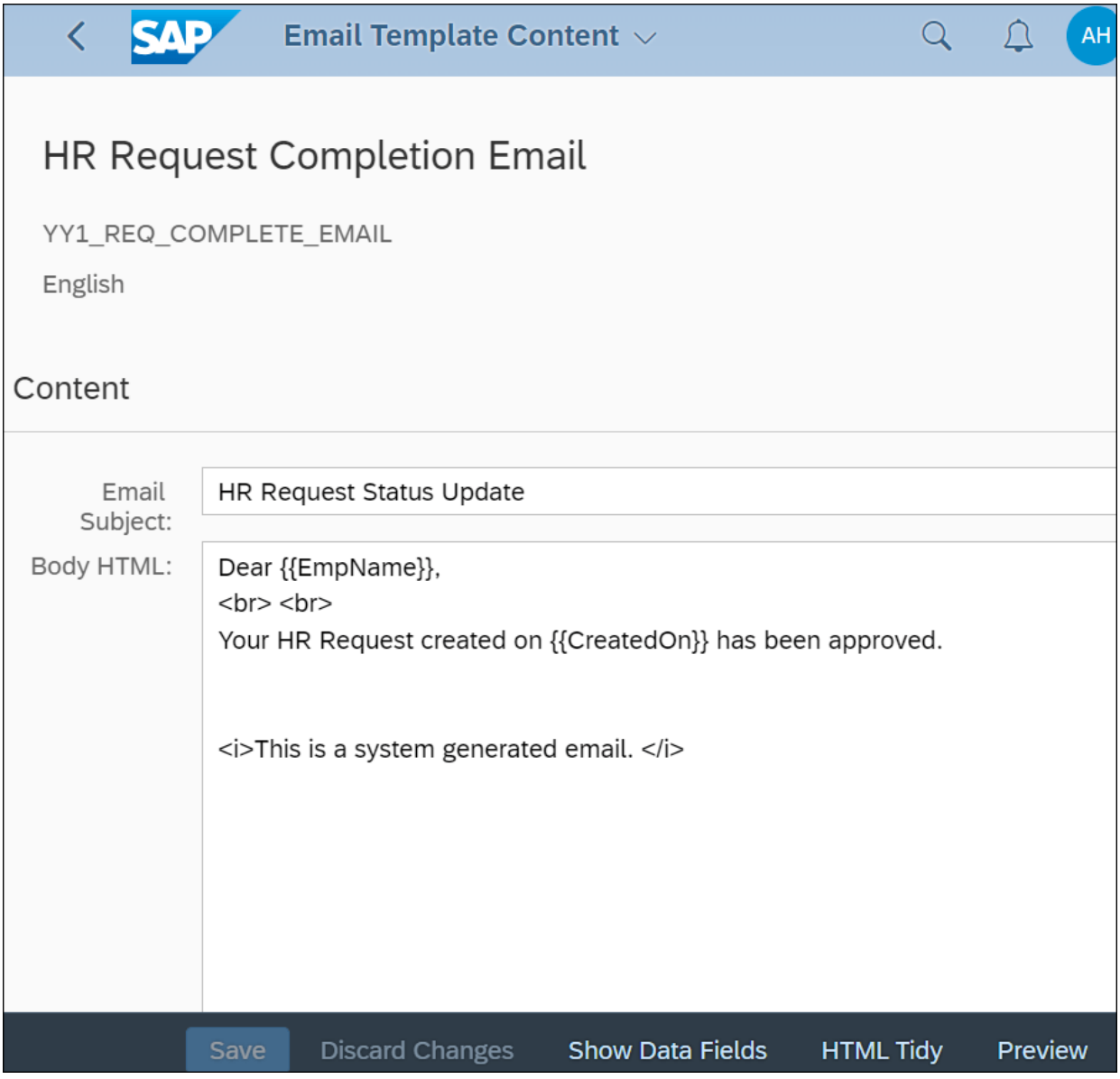


Figure 15.32 Copying Email Content in the Maintain Email Templates App

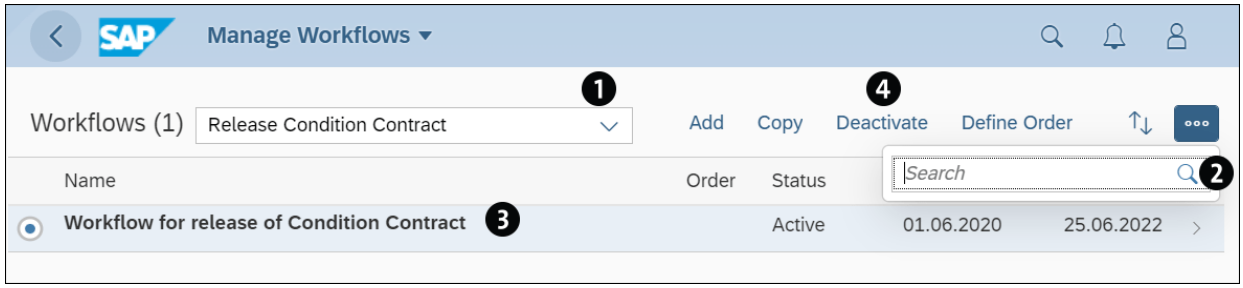


Figure 15.33 Manage Workflows App

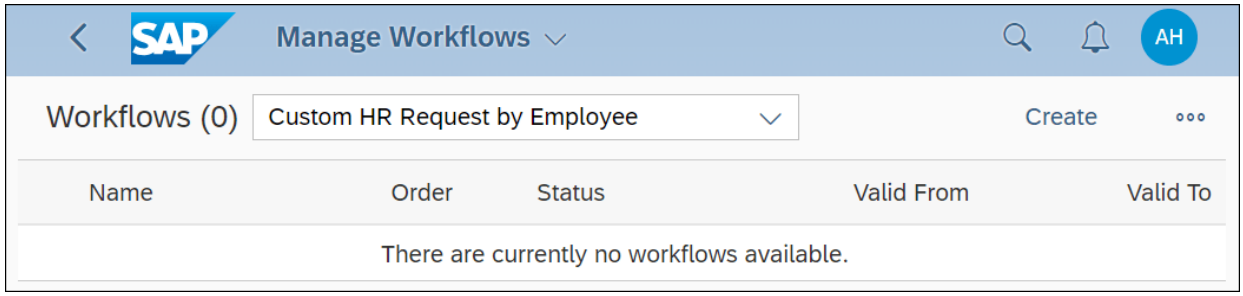


Figure 15.34 Create New Workflow

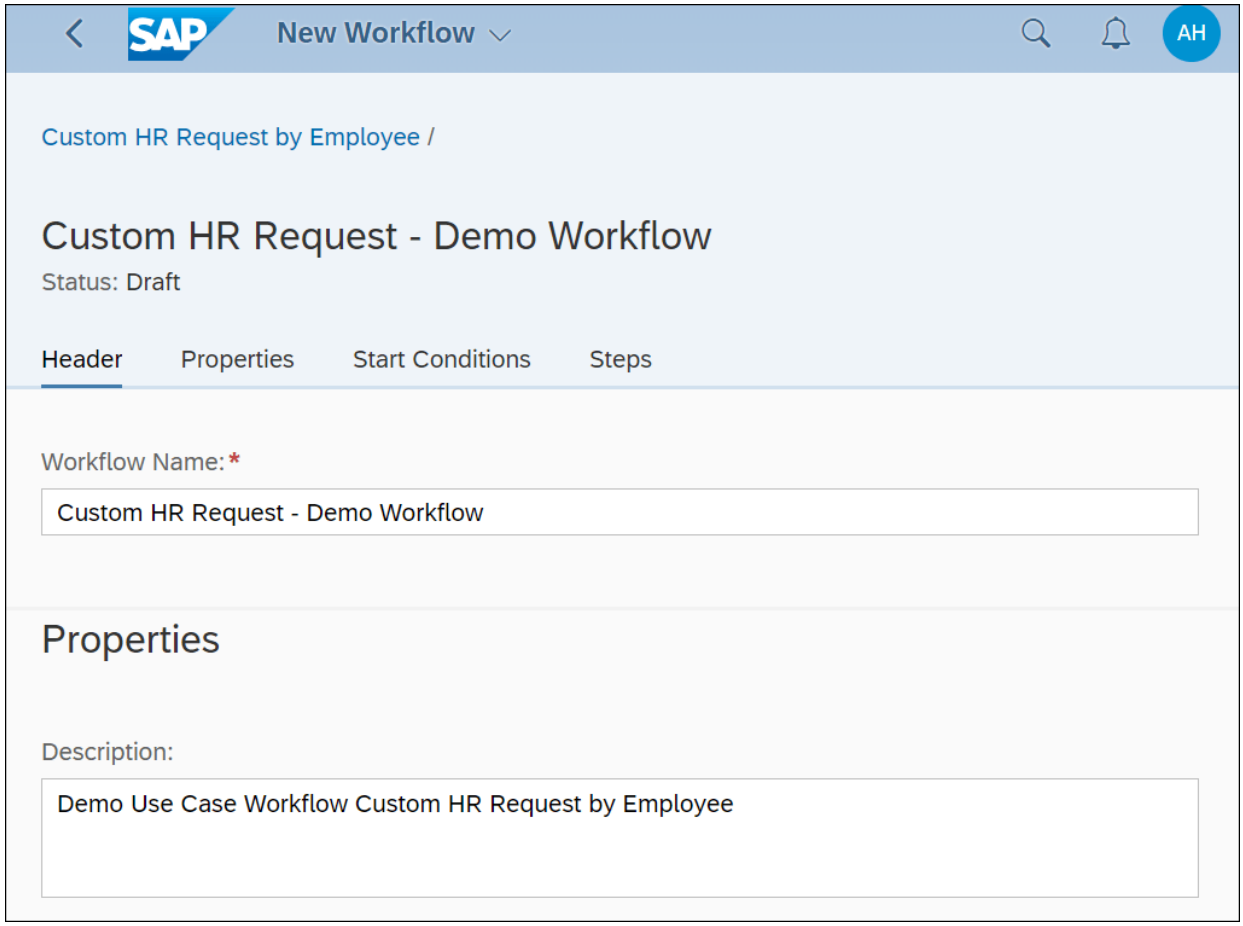


Figure 15.35 Properties of the Custom Flexible Workflow

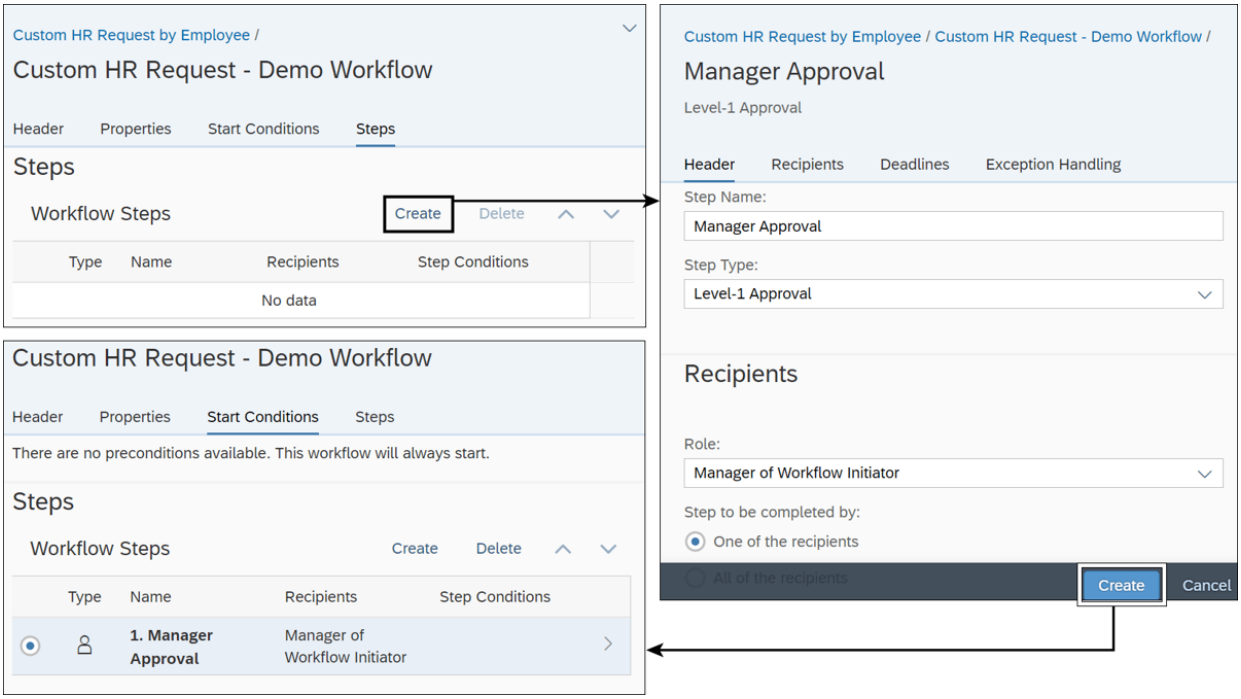


Figure 15.36 Manage Workflow: Create Steps for the Custom Use Case

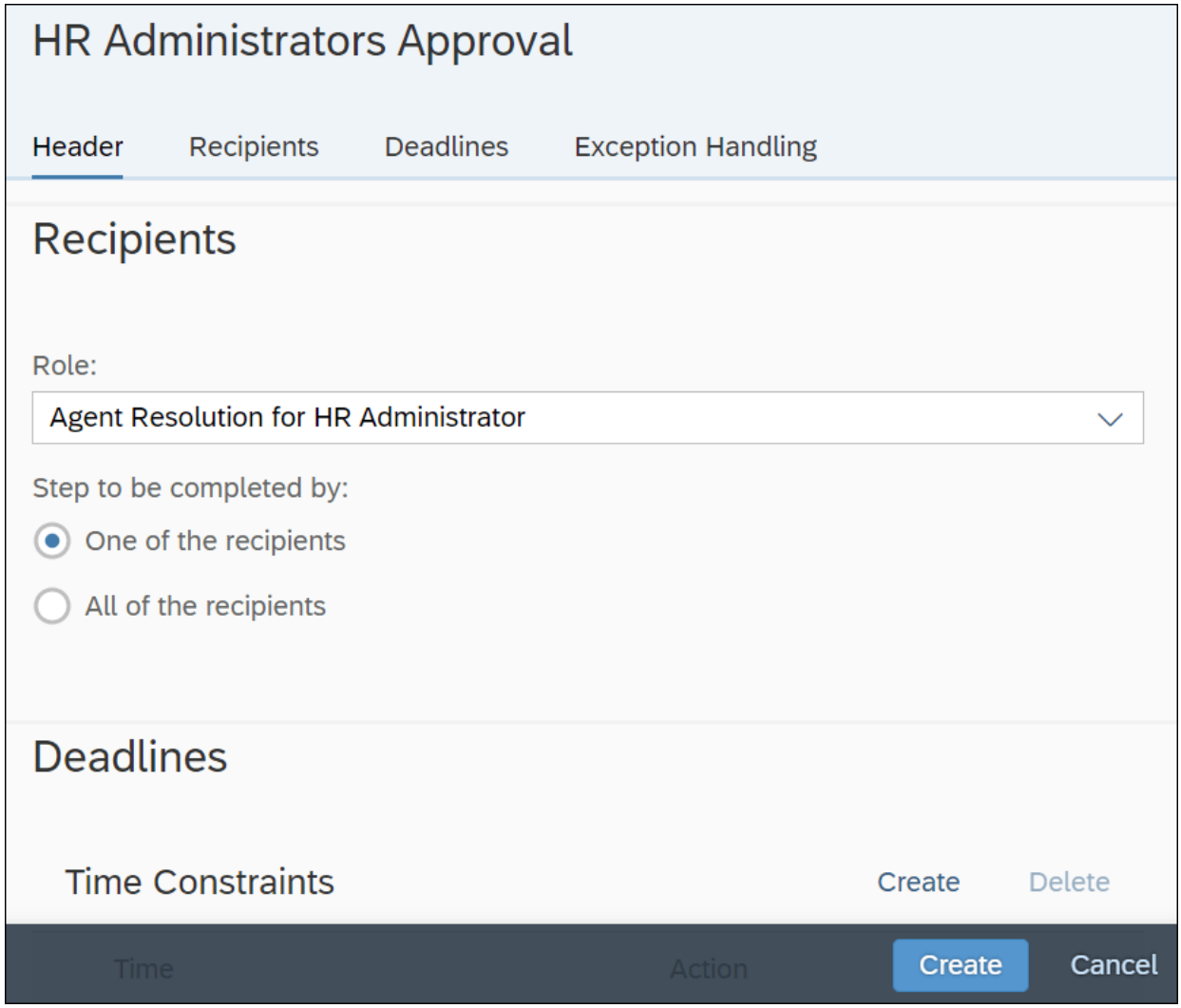


Figure 15.37 Agent Assignment in Flexible Workflow

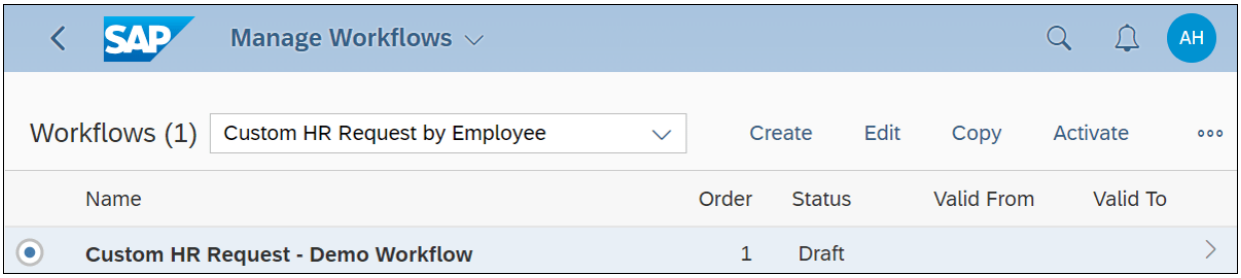


Figure 15.38 Activate the Draft Workflow of the Sample Use Case

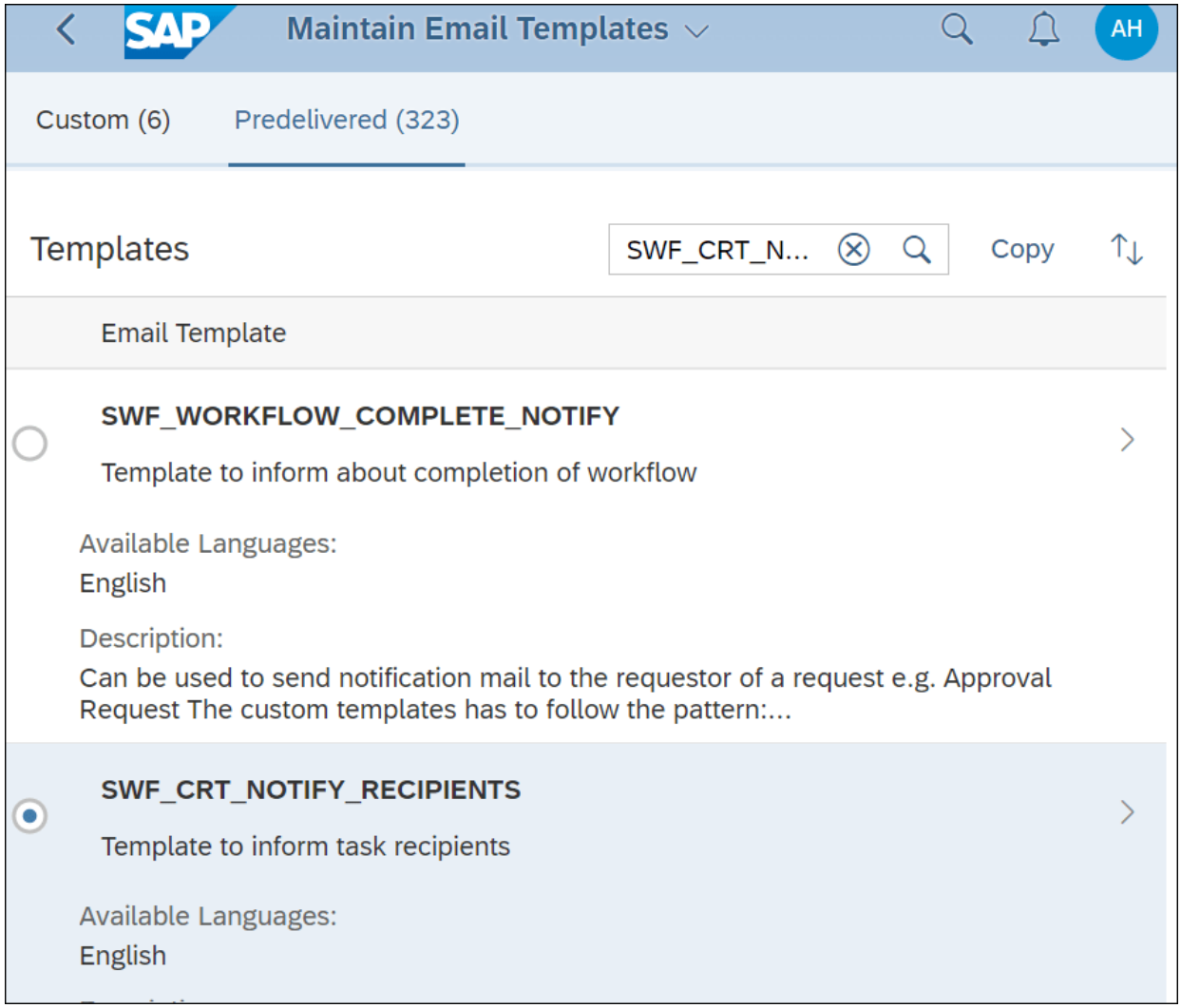


Figure 15.39 Copying the Standard Email Template

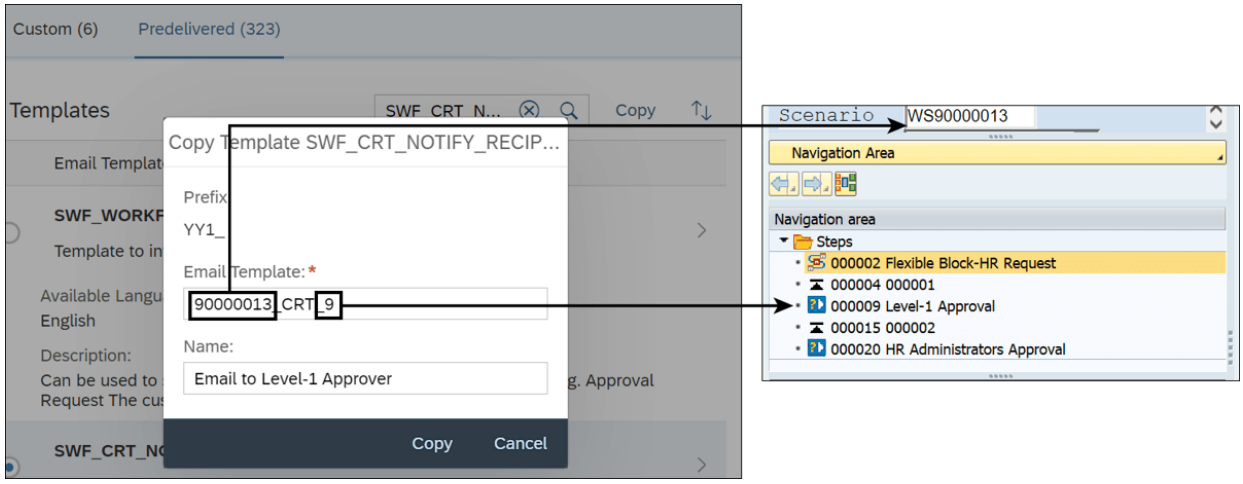


Figure 15.40 Naming the Custom Email Template

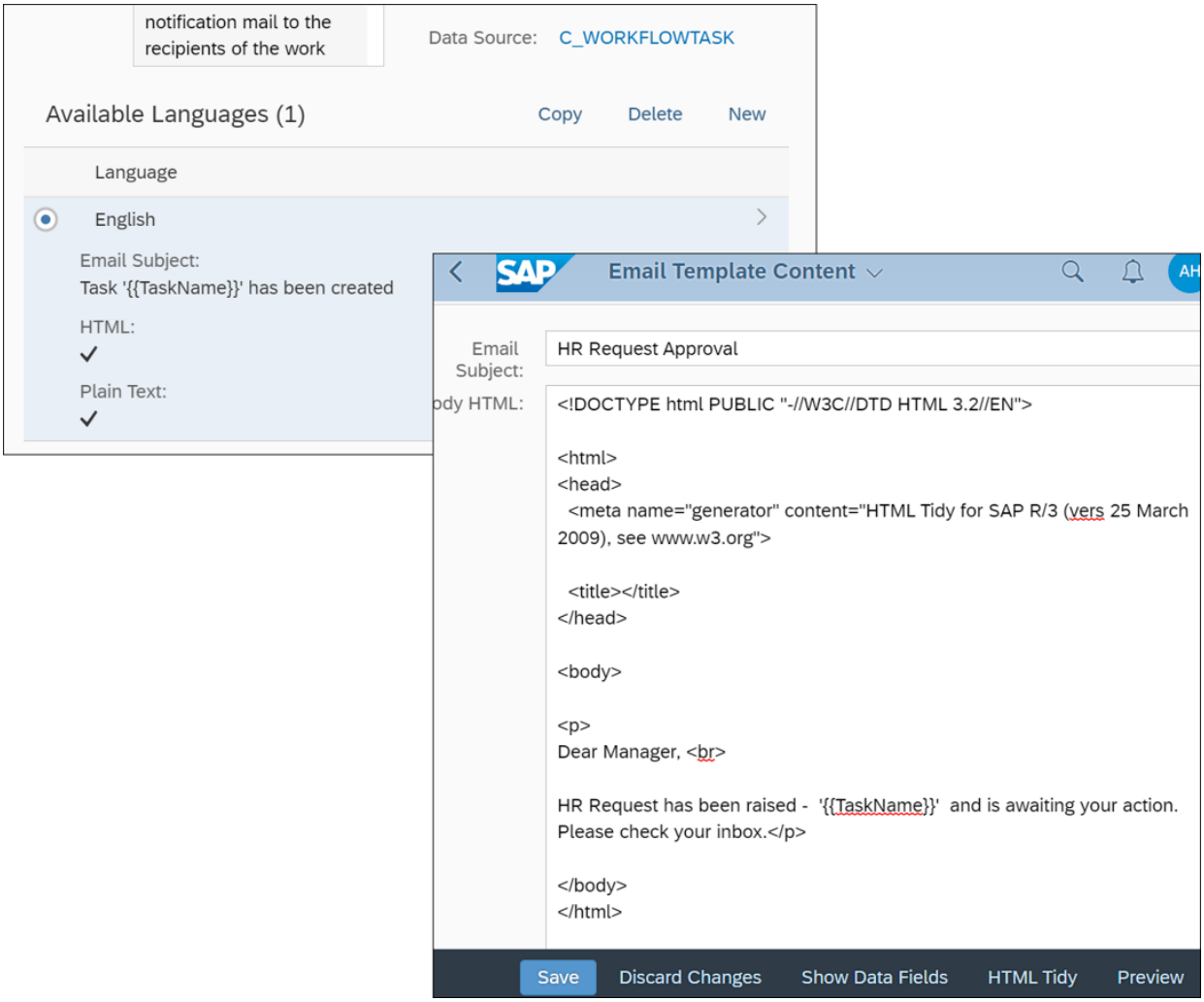


Figure 15.41 Email Templates Subject and Body Contents

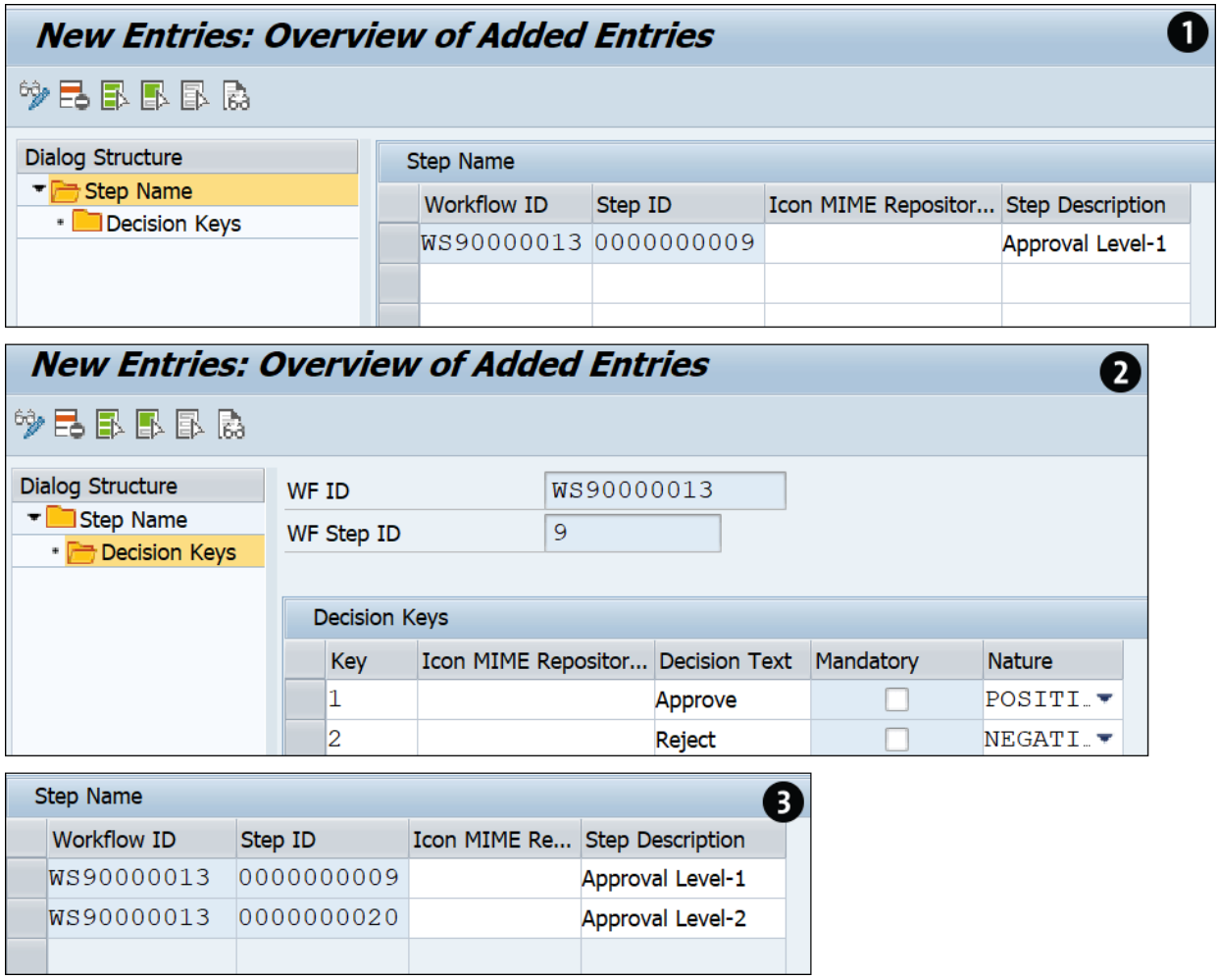


Figure 15.42 Define Steps for My Inbox Approval

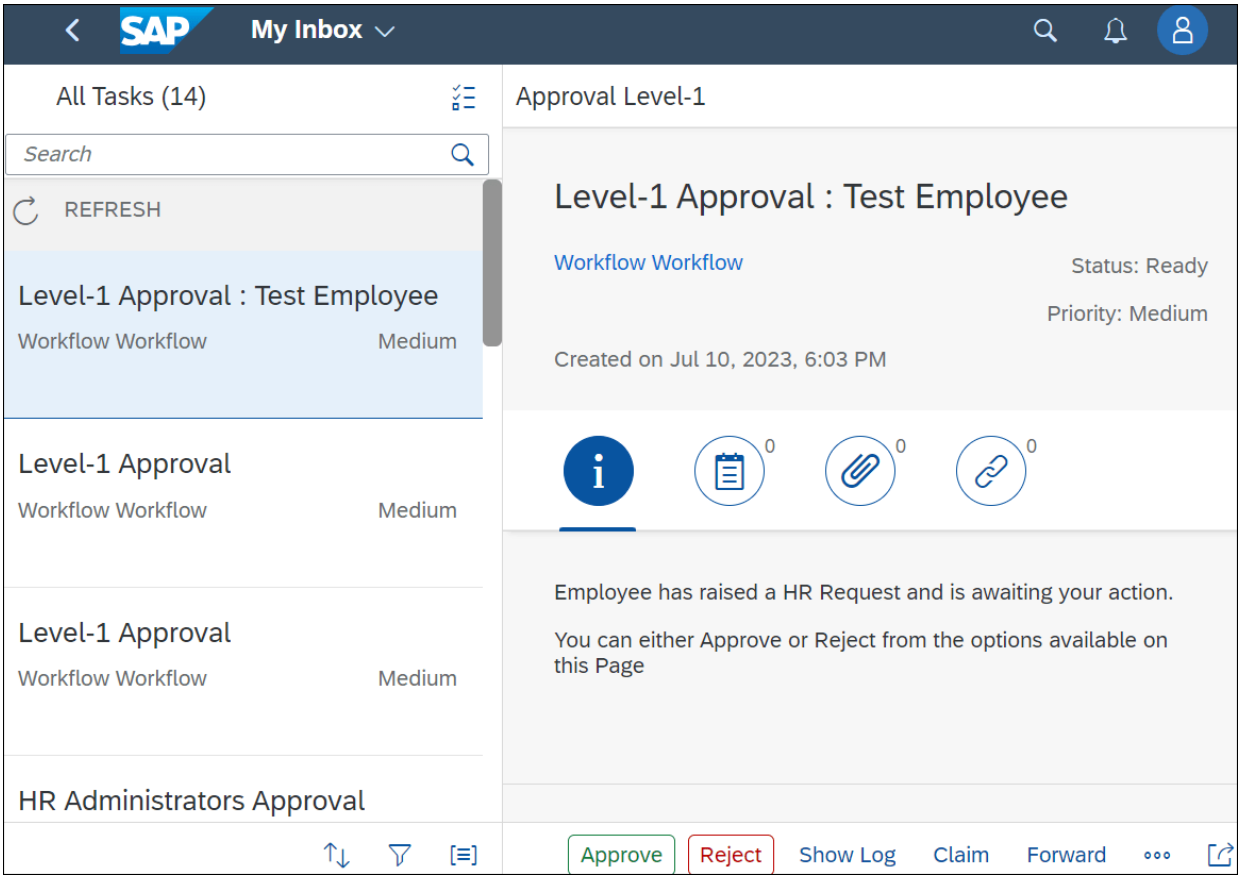


Figure 15.43 Work Item in the First Recipient's Inbox

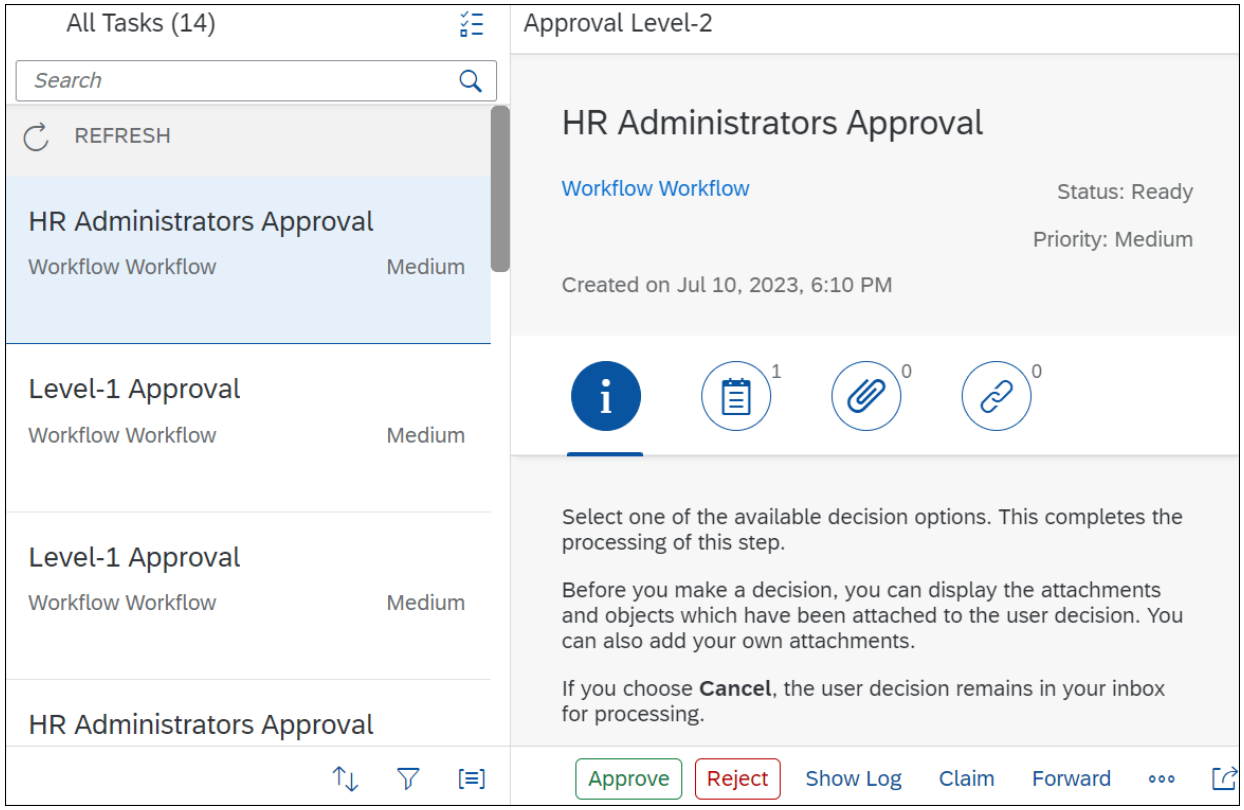


Figure 15.44 Work Item in Second Recipient's Inbox

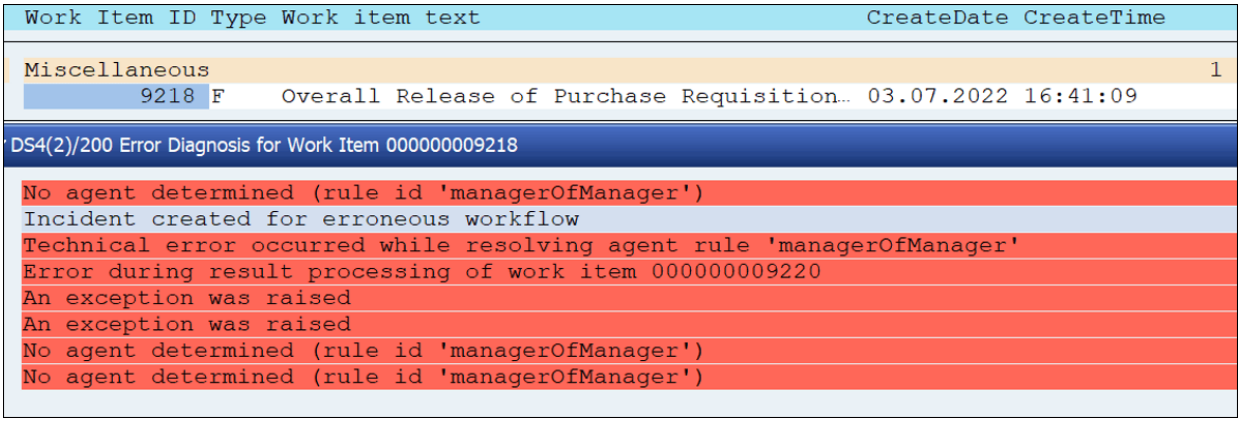


Figure 15.45 Agent Determination Error from Transaction SWI2_DIAG (Diagnosis of Workflows with Errors)

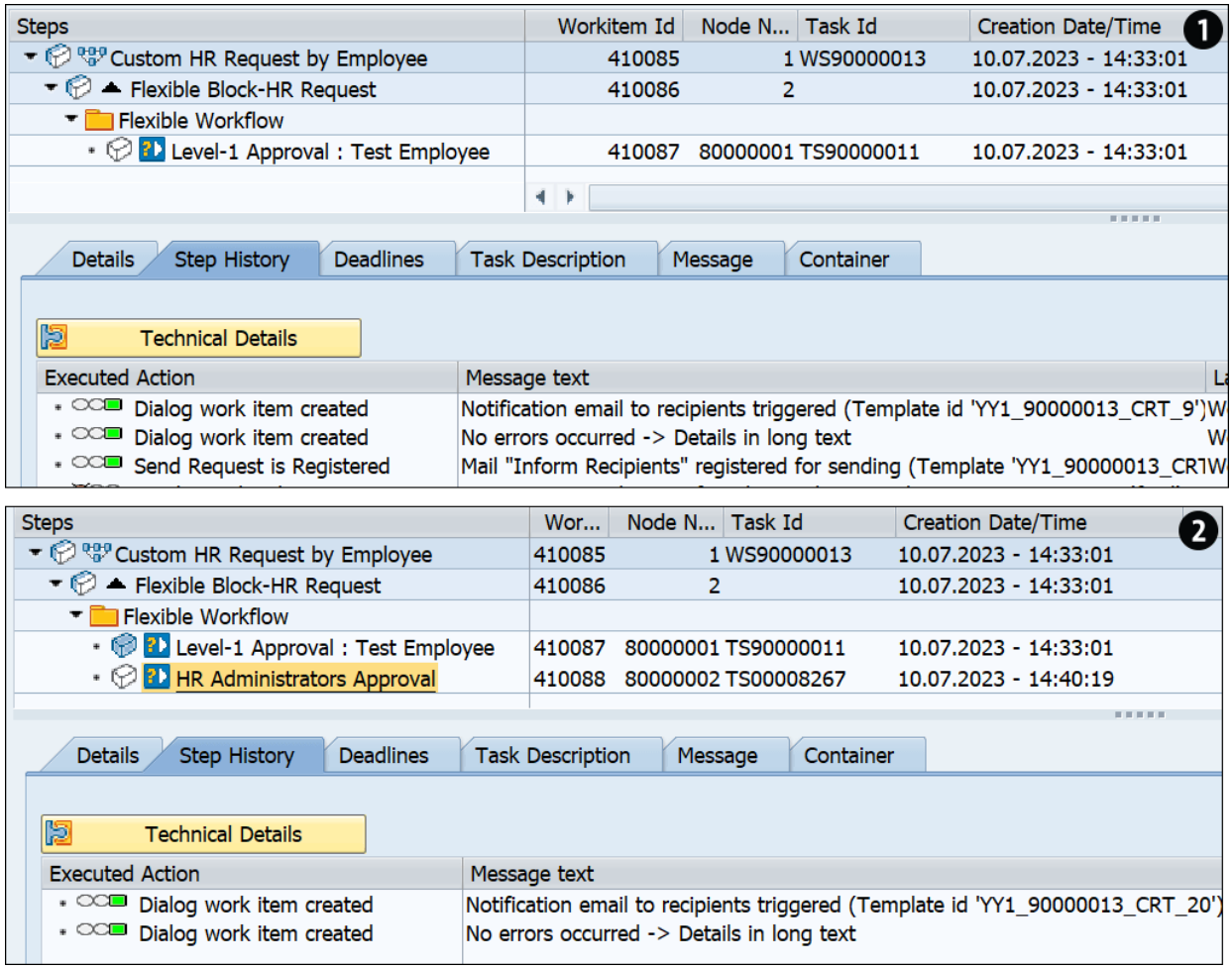


Figure 15.46 Workflow Technical Log for HR Request Use Case

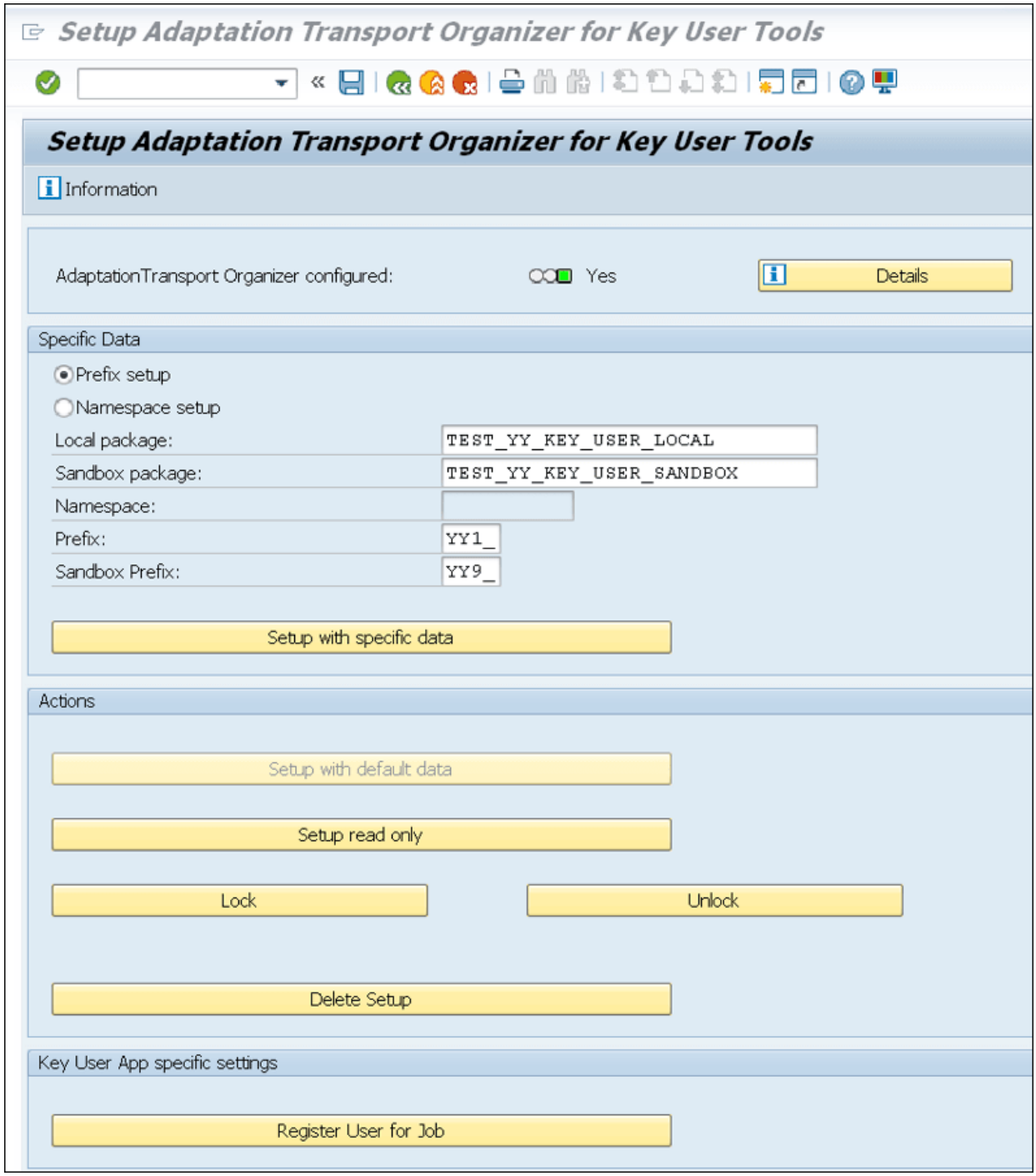


Figure 16.1 Transaction S_ATO_SETUP

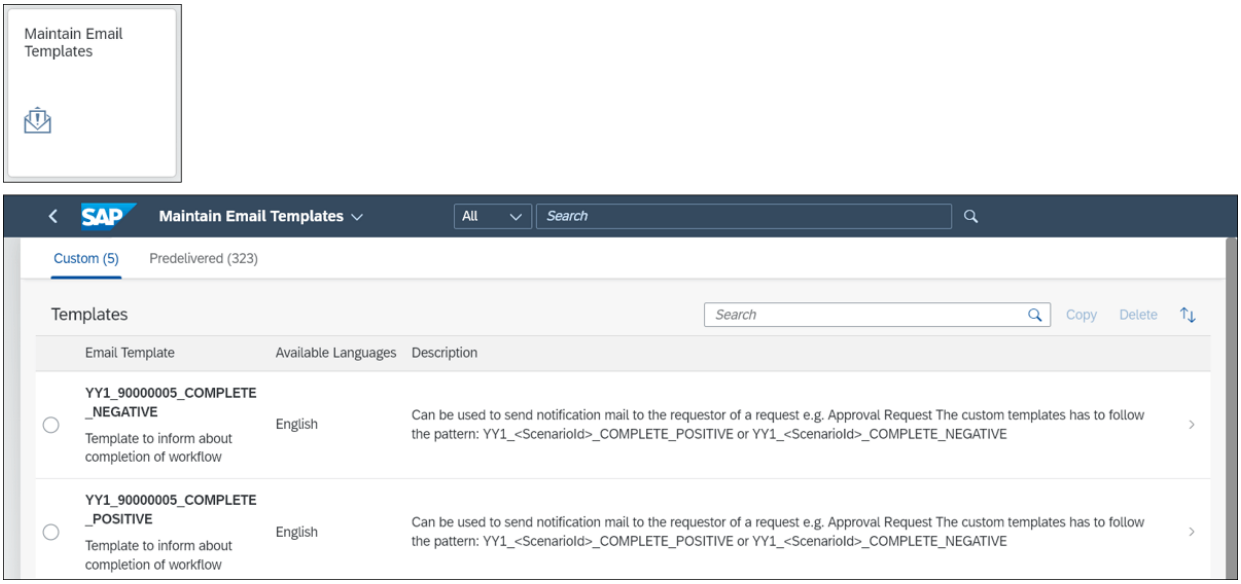


Figure 16.2 Maintain Email Templates App

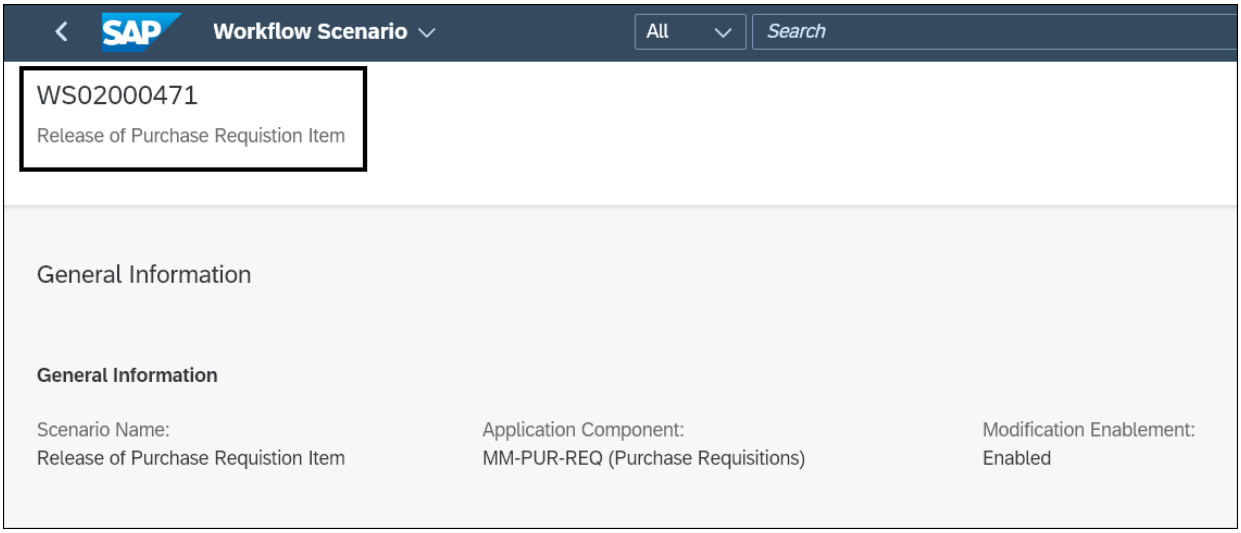


Figure 16.3 Scenario ID

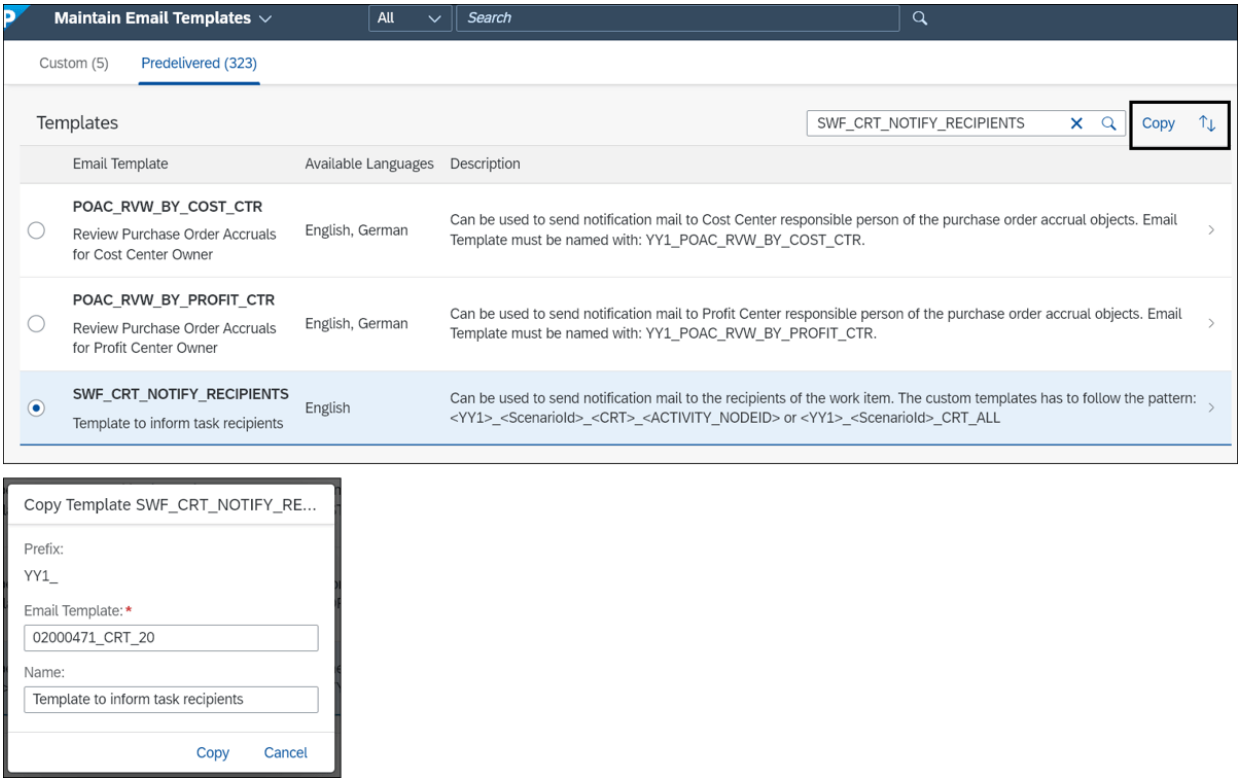


Figure 16.4 Copy Standard Email Template

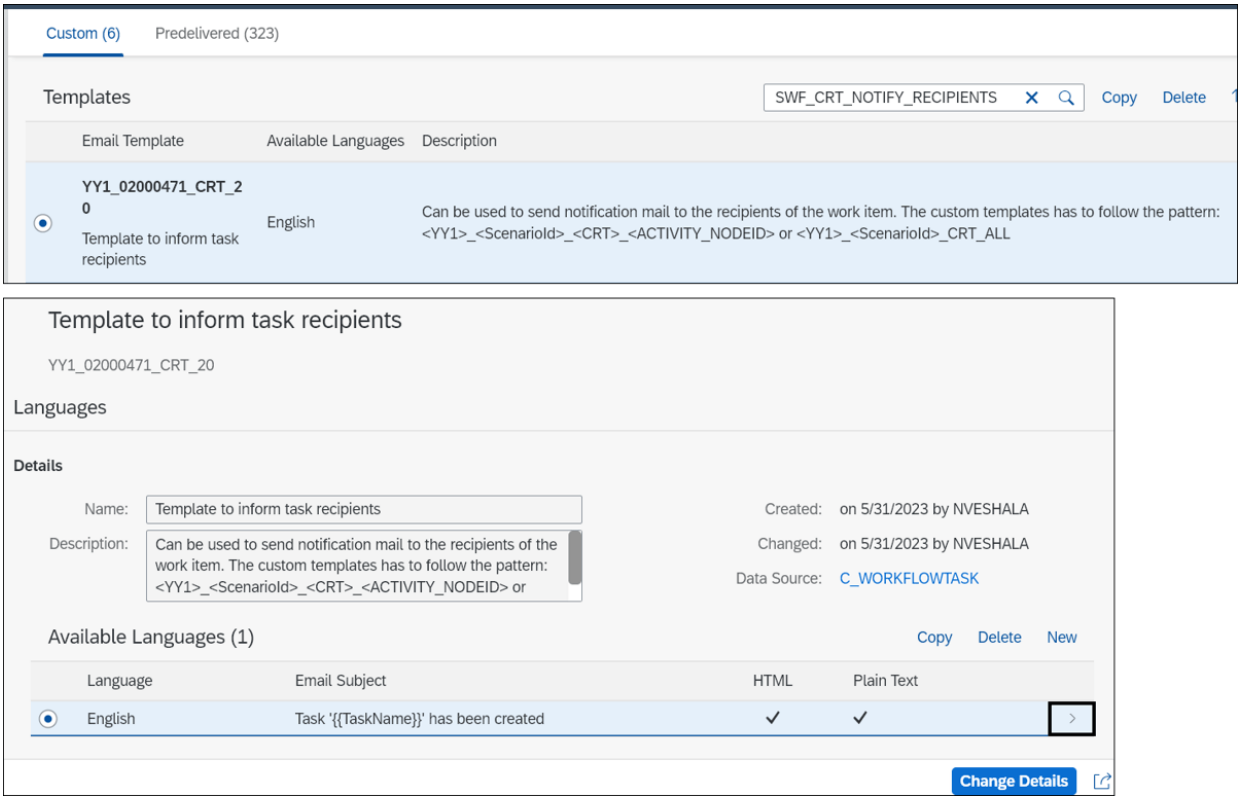


Figure 16.5 Email Template Details

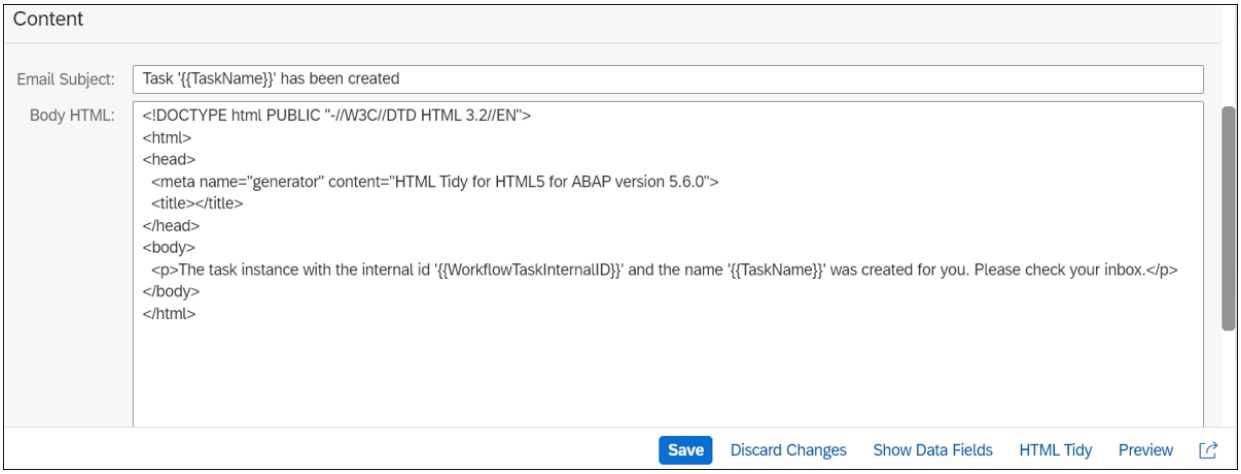


Figure 16.6 Email Content

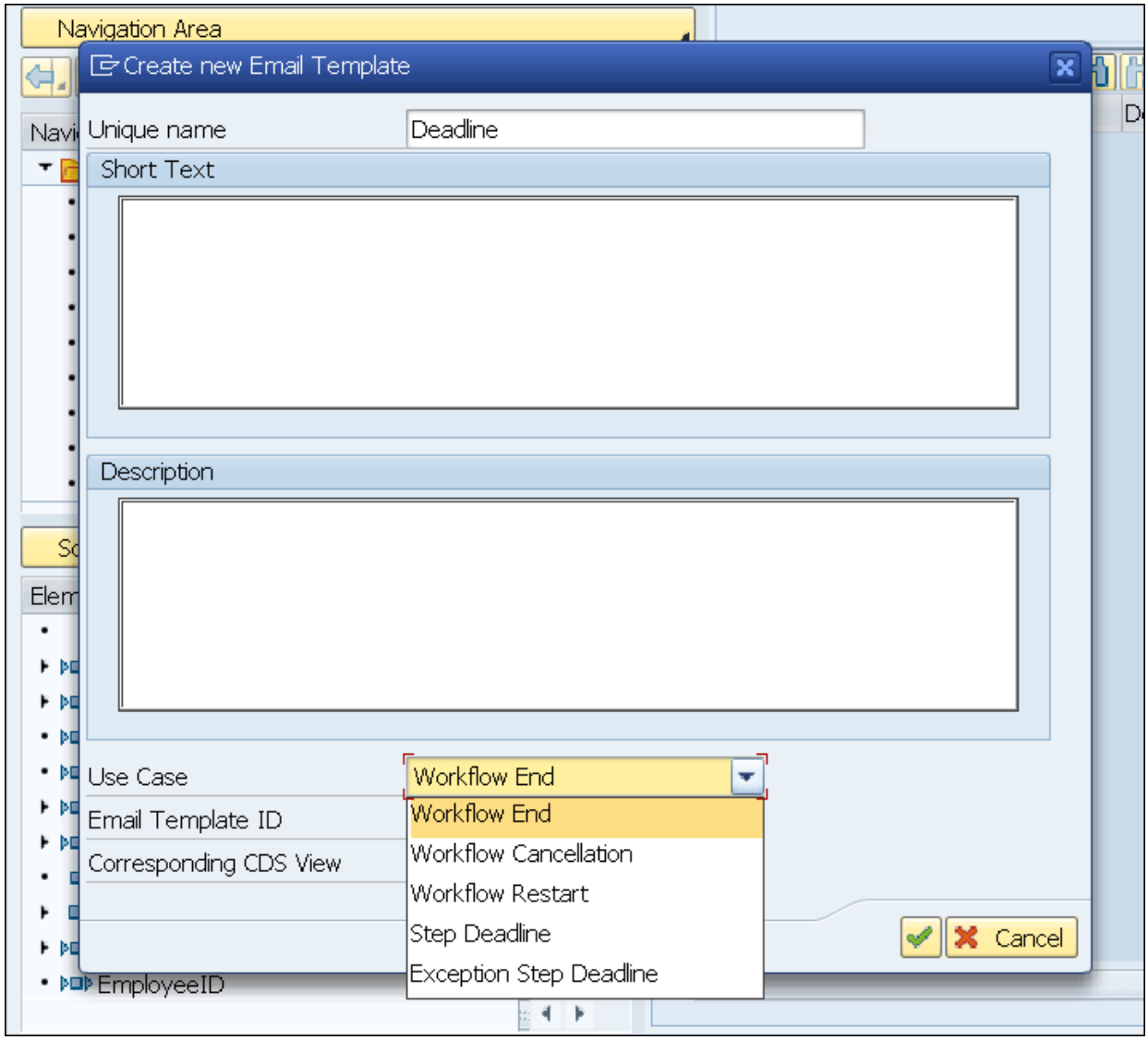


Figure 16.7 Email Notification Use Cases

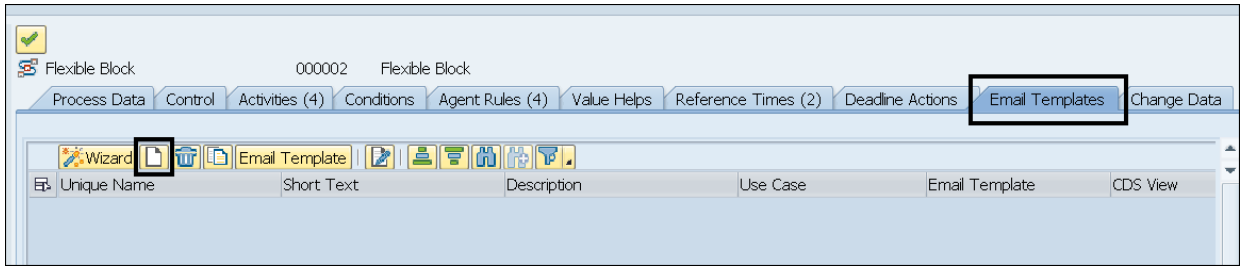


Figure 16.8 Deadline: Email Template

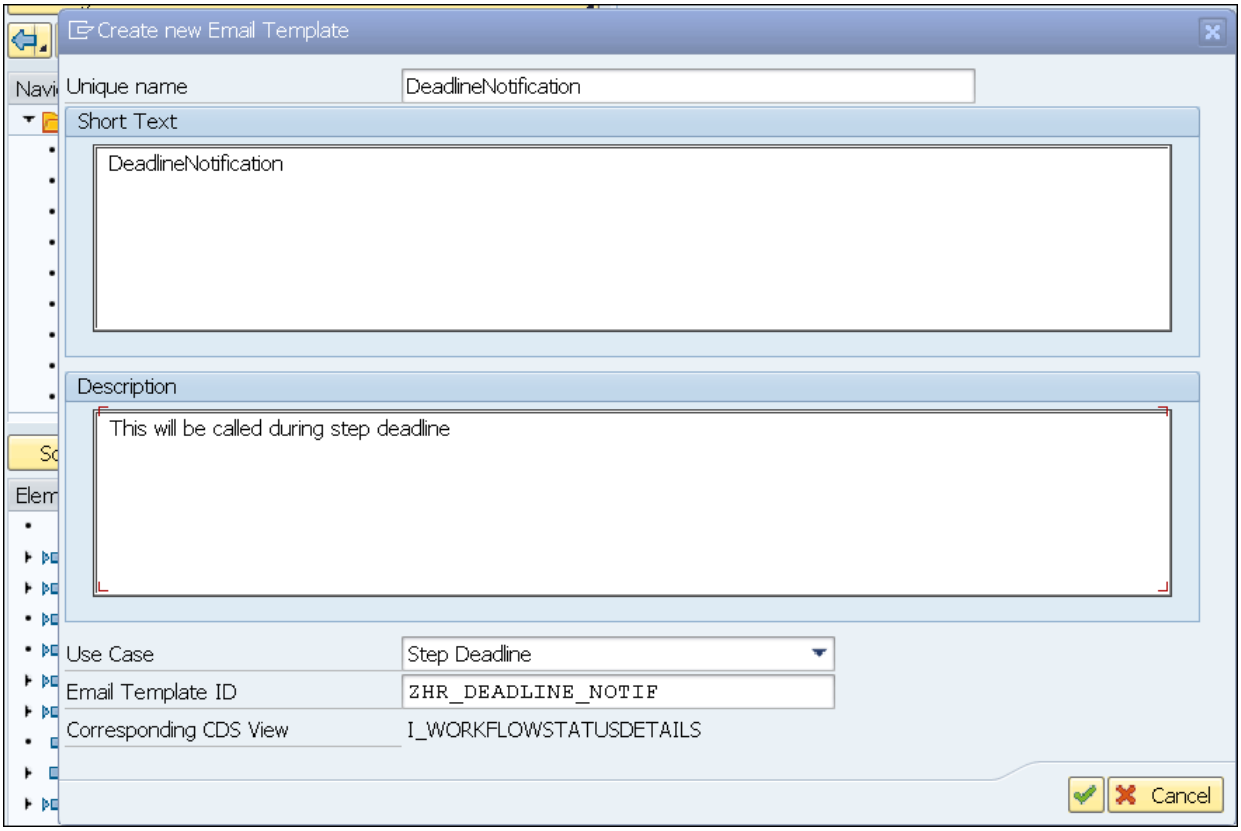


Figure 16.9 Deadline: Create New Email Template

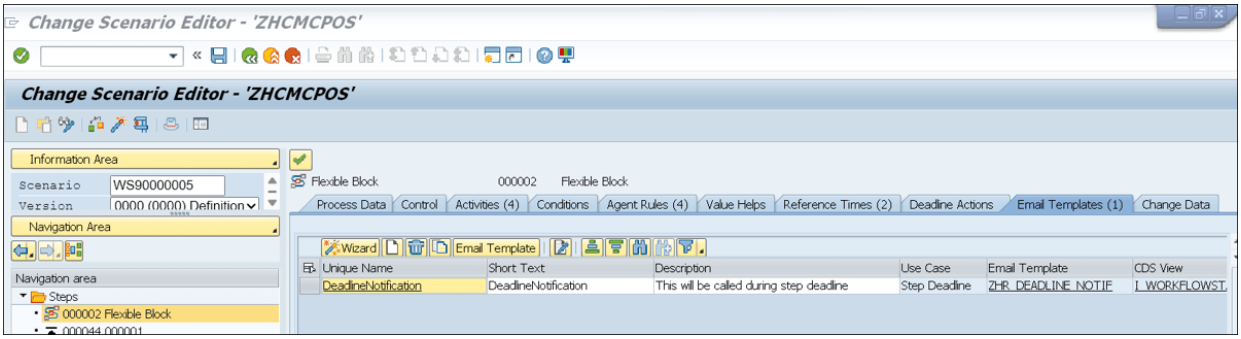


Figure 16.10 Scenario: Email Template

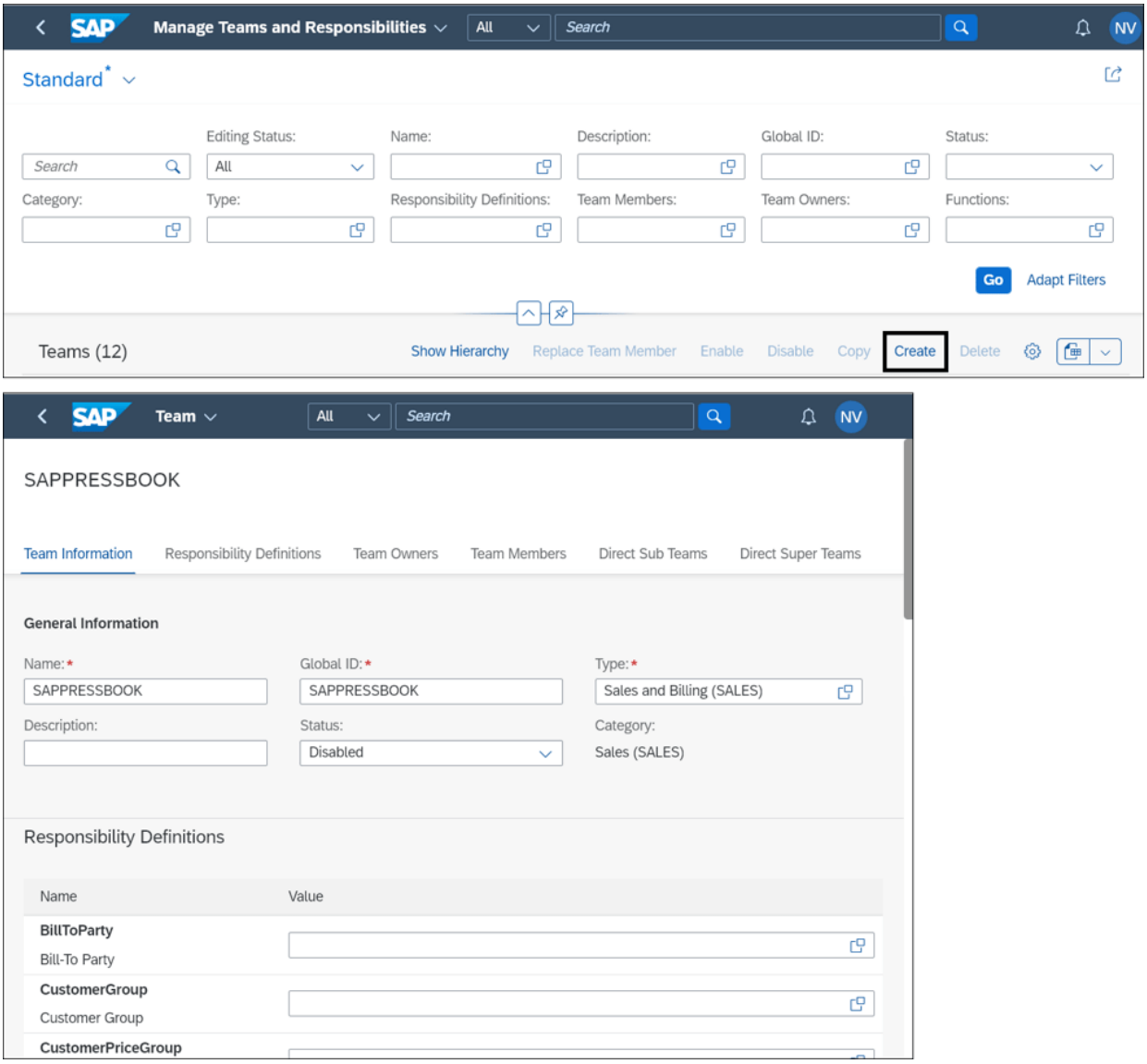


Figure 16.11 Manage Teams and Responsibilities

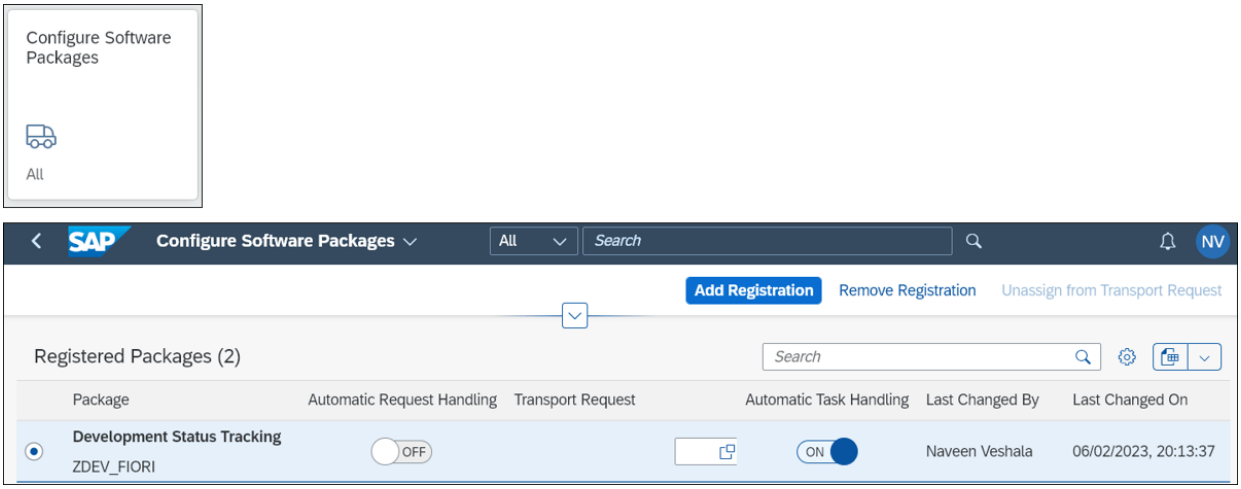


Figure 16.12 Configure Software Packages

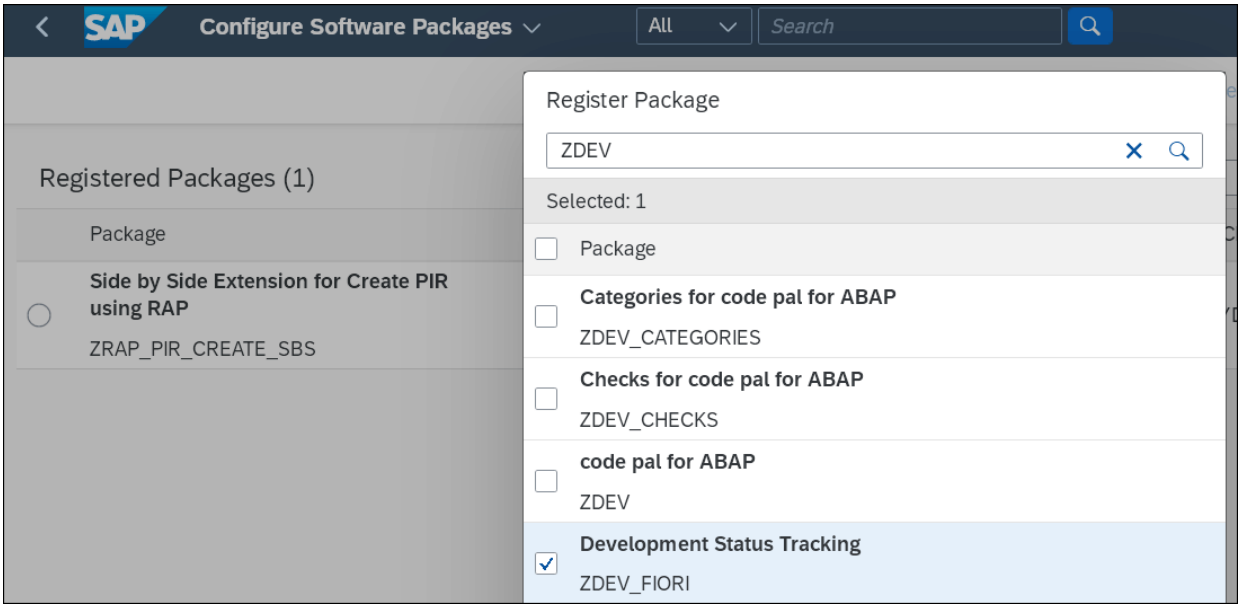


Figure 16.13 Register Package

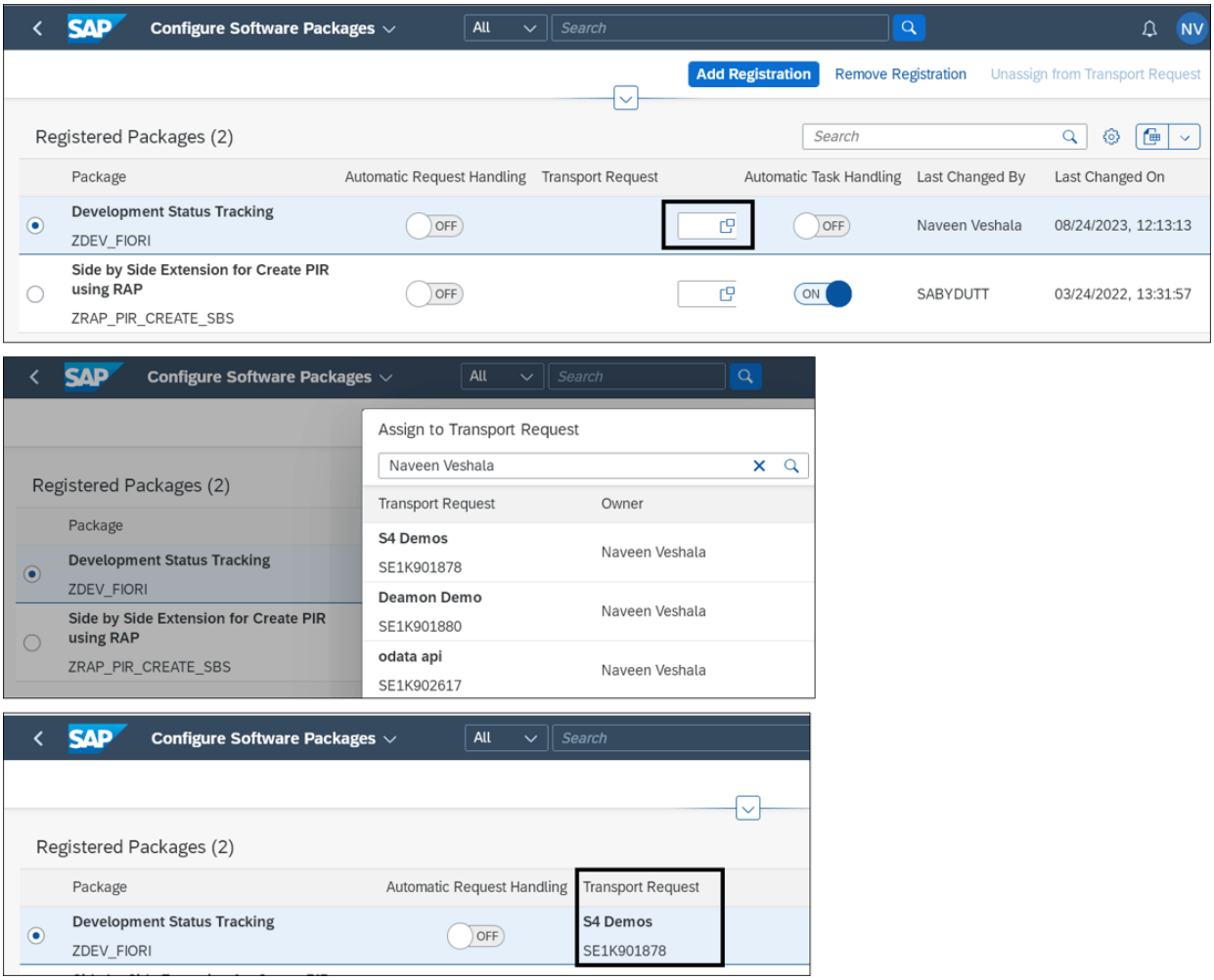


Figure 16.14 Assign the Transport to the Registered Package

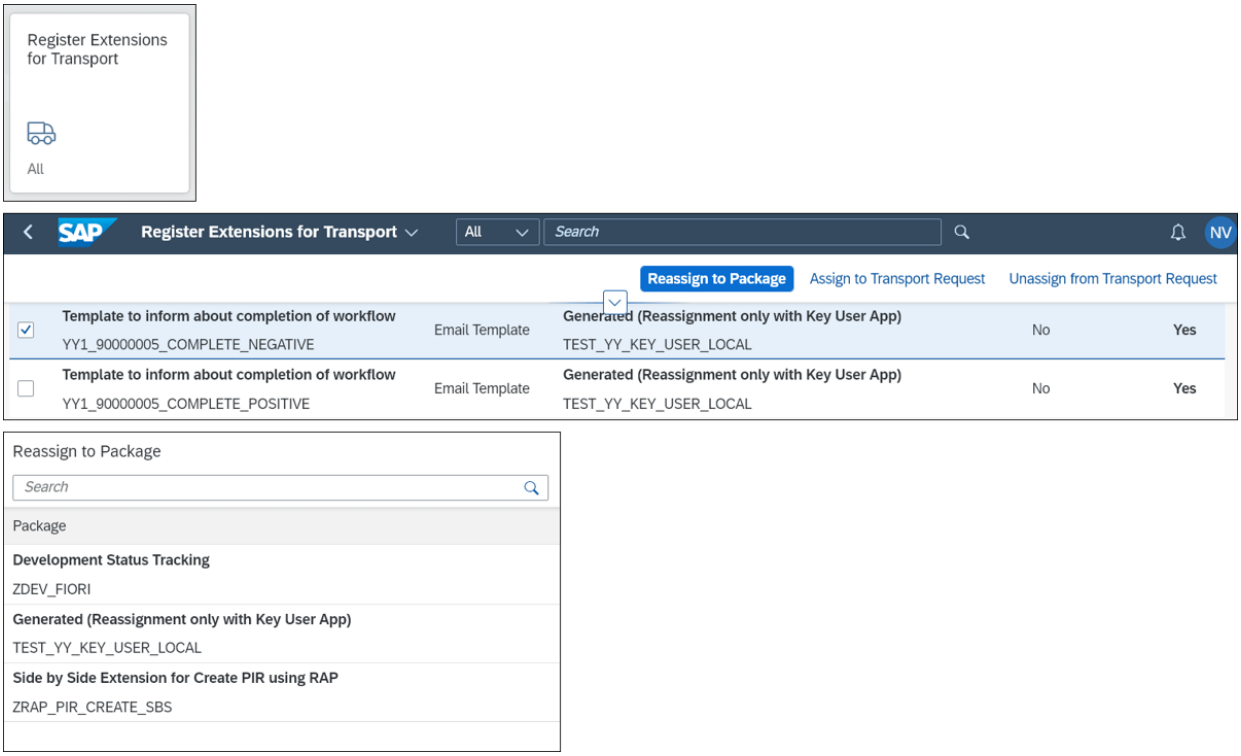


Figure 16.15 Register Extensions for Transports

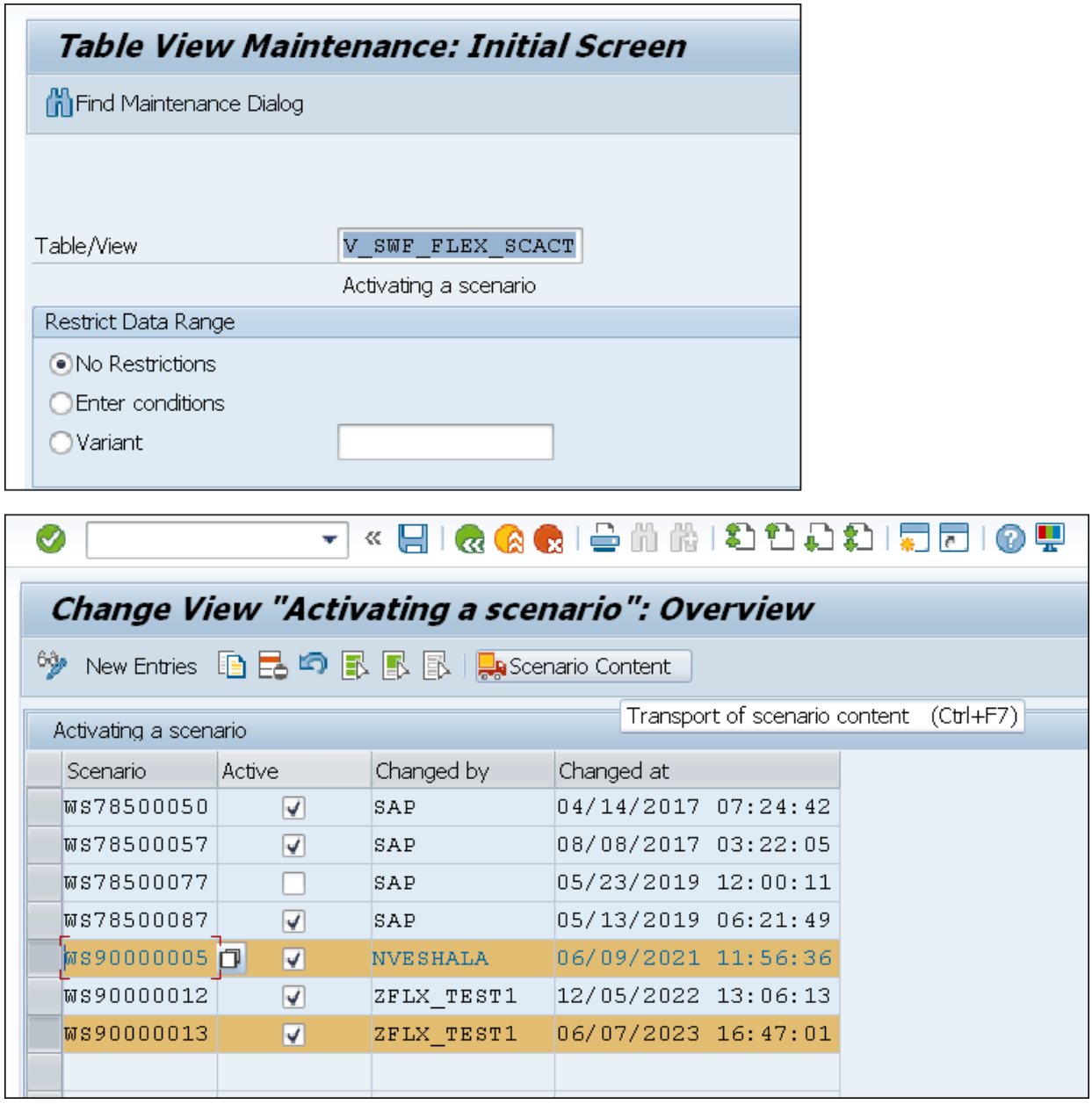


Figure 16.16 Transport Scenario

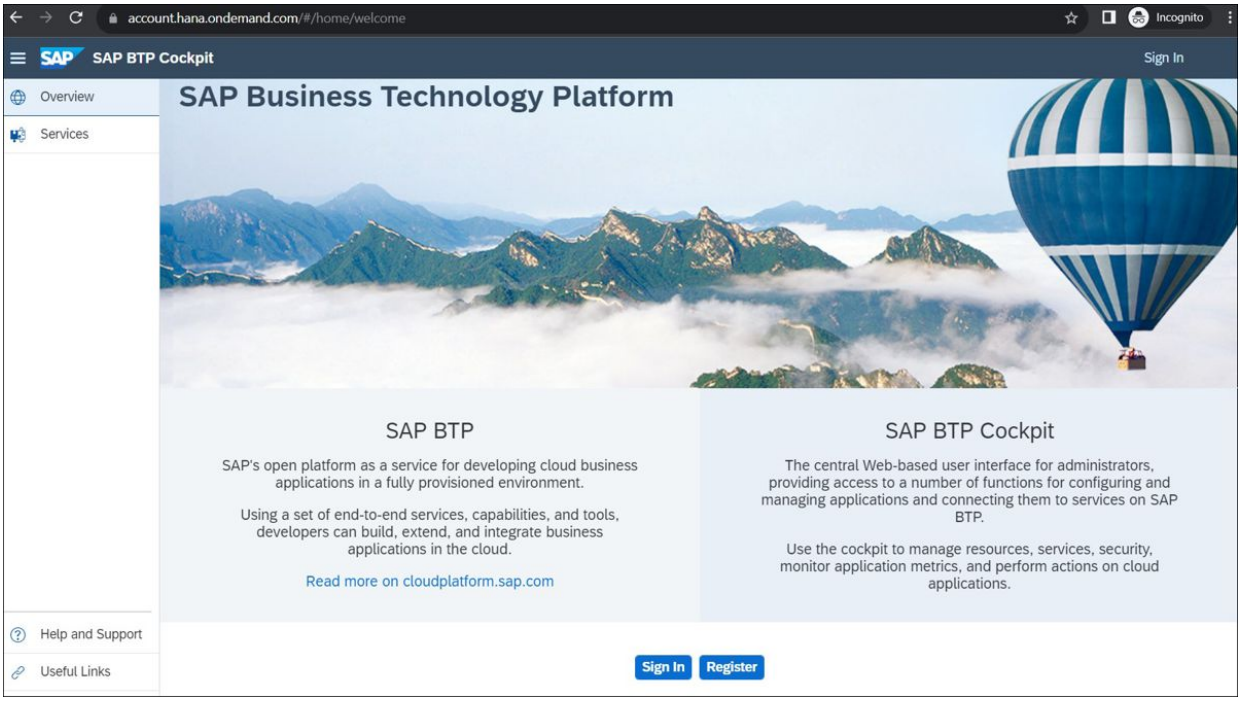


Figure 17.1 SAP BTP Cockpit Sign-In Page

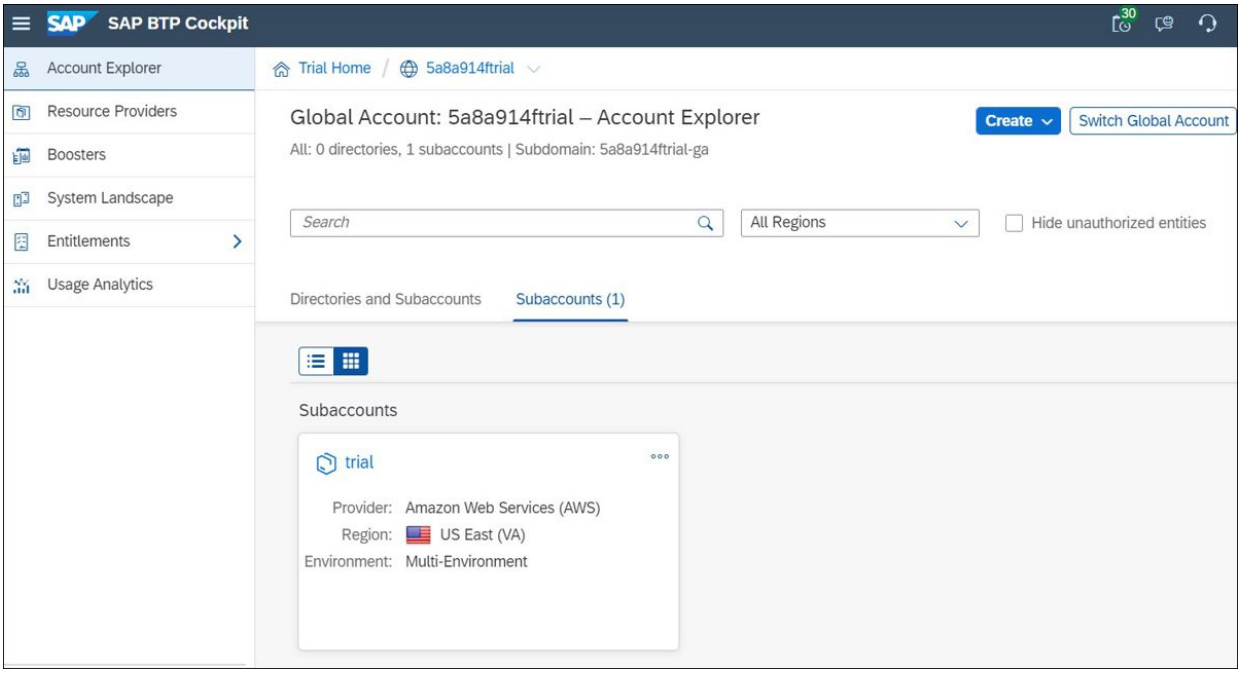


Figure 17.2 SAP BTP Cockpit after Login

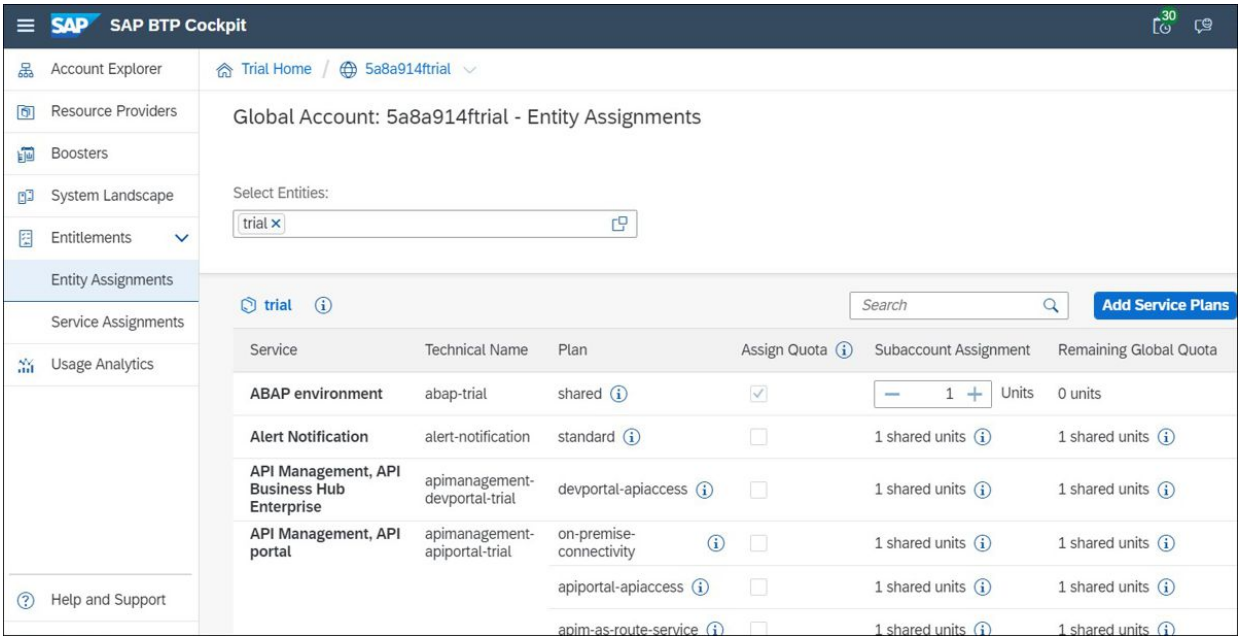


Figure 17.3 Add Service Plans

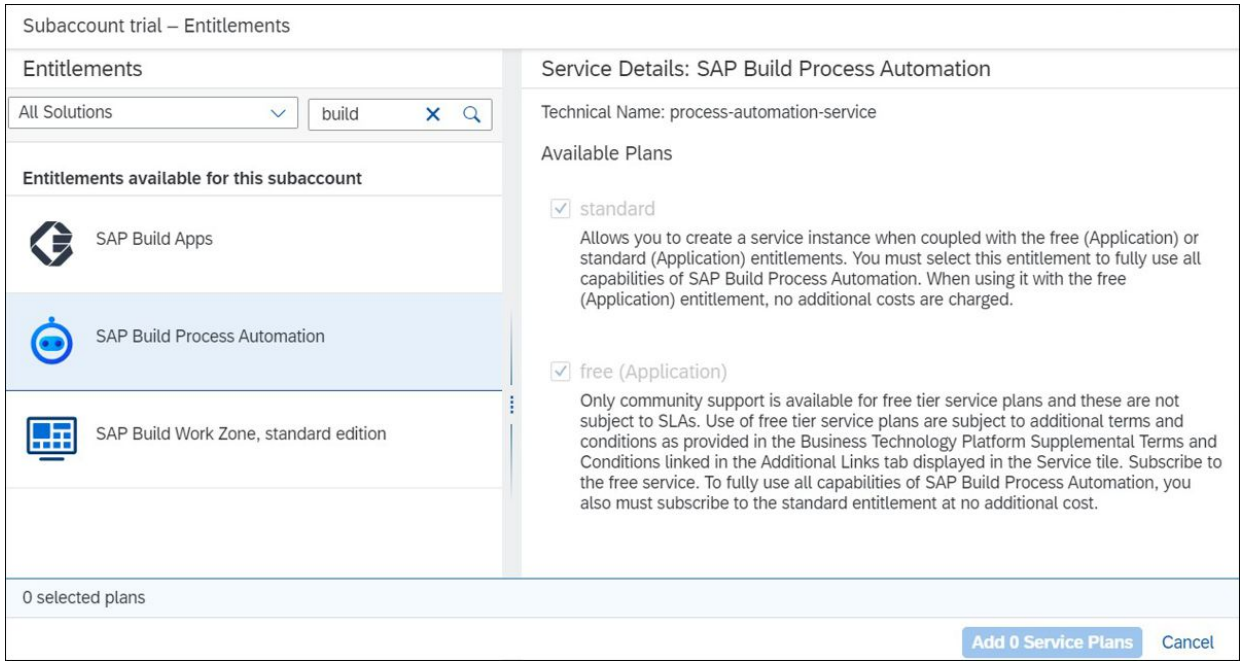


Figure 17.4 SAP Build Process Automation: Available Service Plans

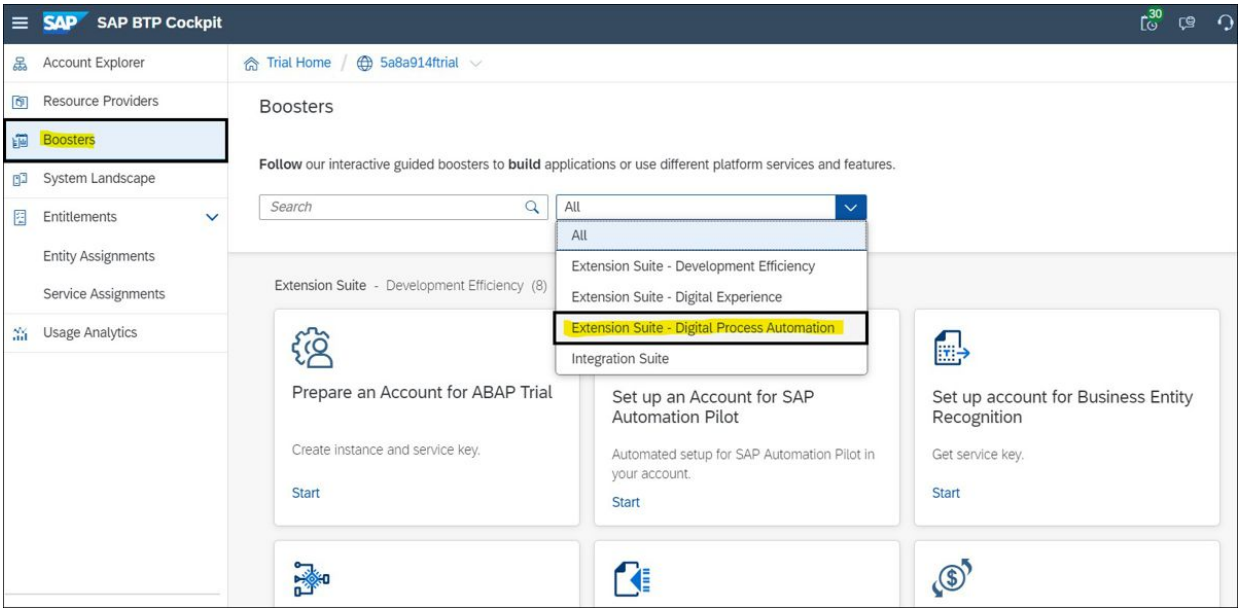


Figure 17.5 Start Service Configuration with the Booster Option

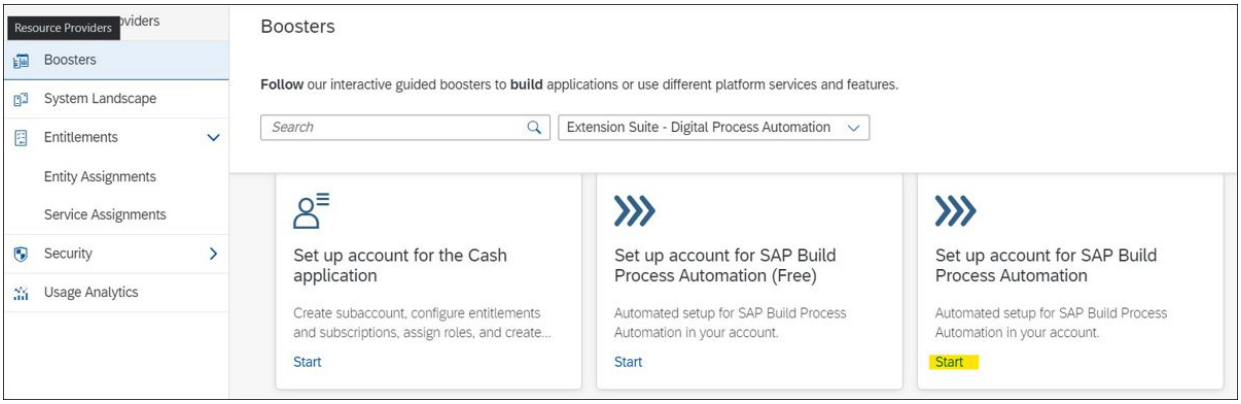


Figure 17.6 Start Booster Configuration of the Service

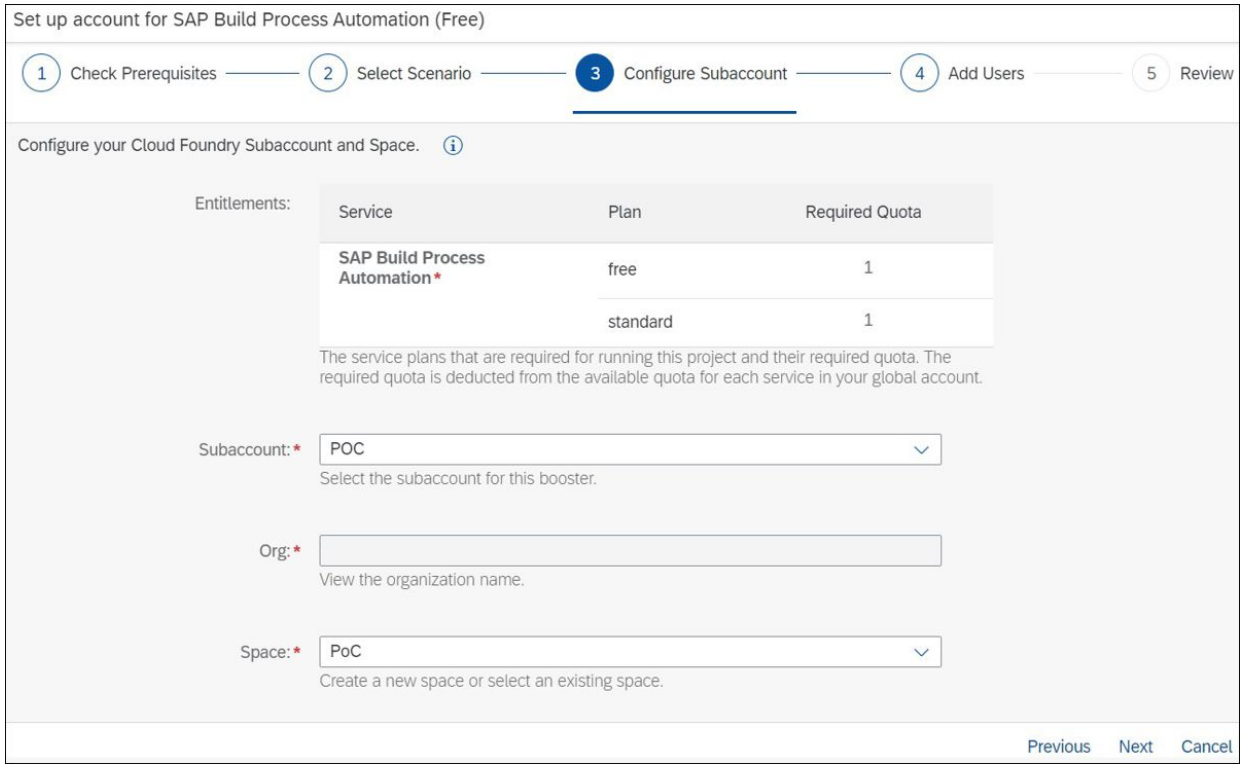


Figure 17.7 Configuration of the SAP Build Process Automation Service to a Selective Subaccount and Space

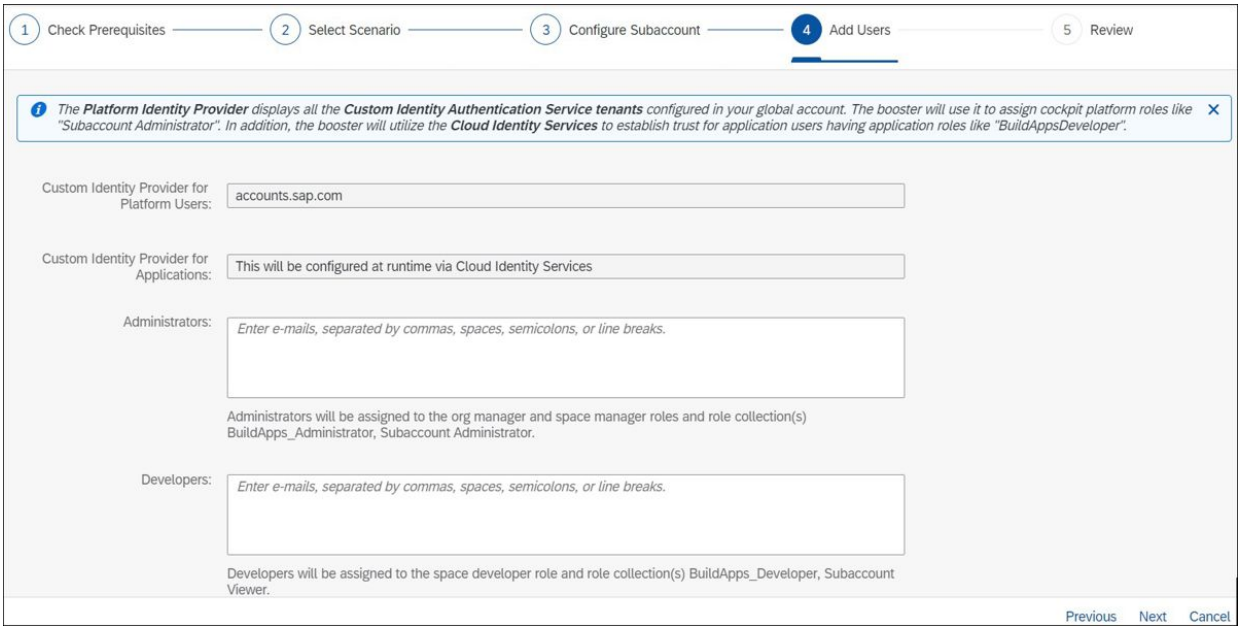


Figure 17.8 Identity Authentication Service Selection Screen

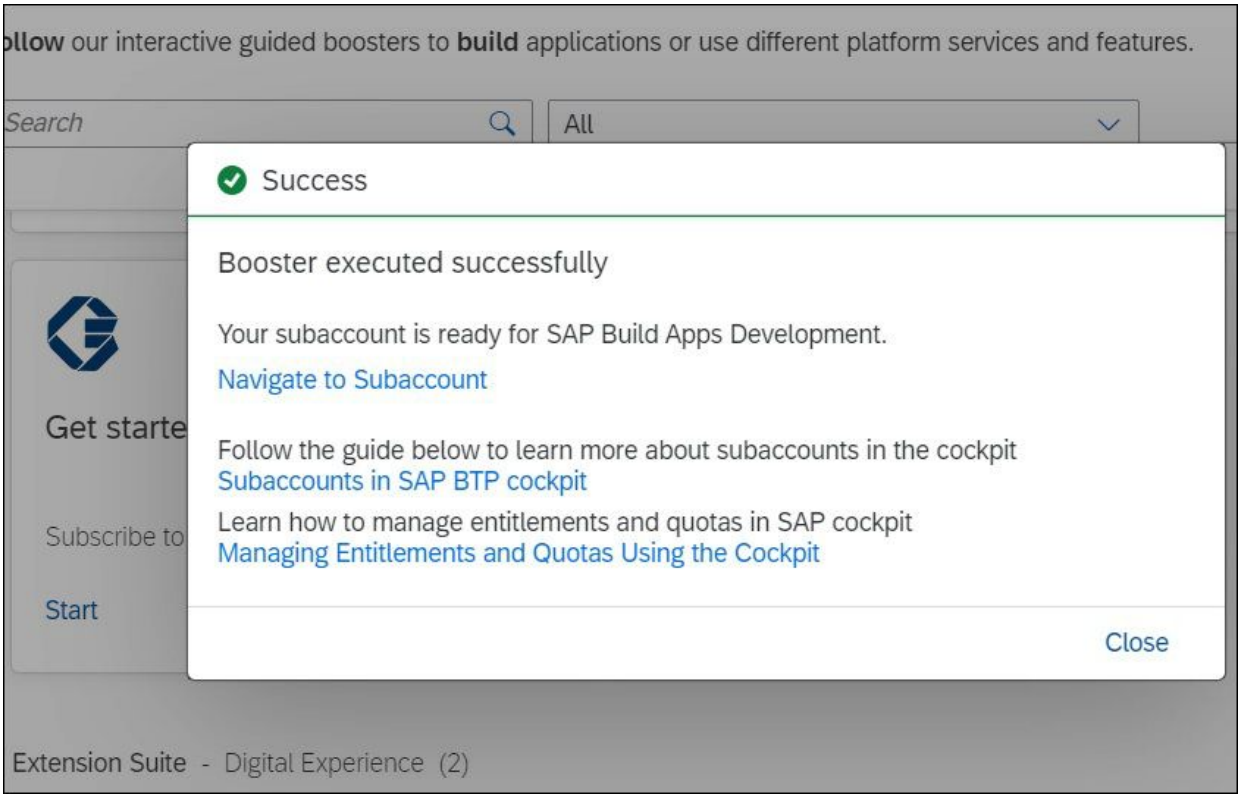


Figure 17.9 Booster Created the Service Successfully

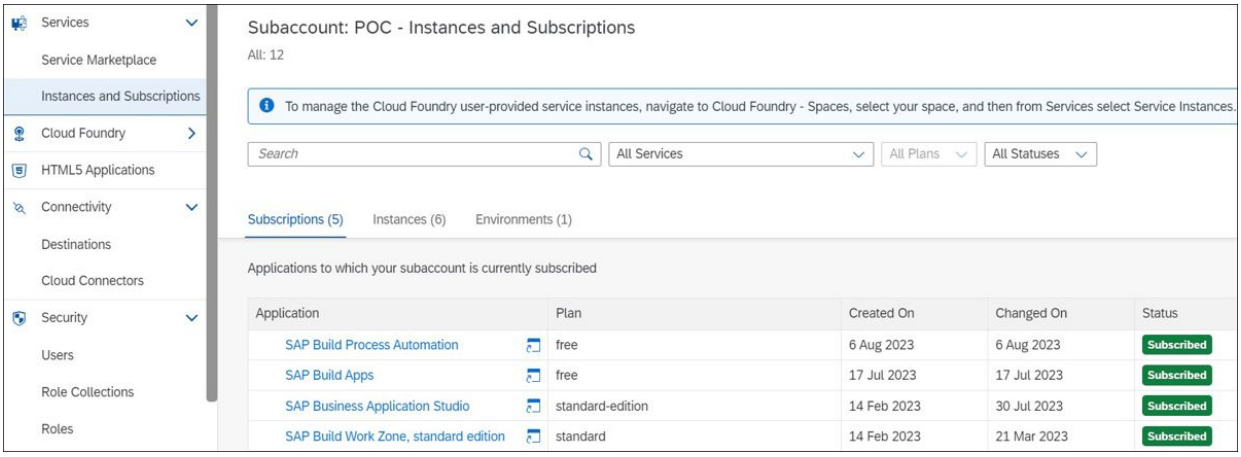


Figure 17.10 SAP Build Process Automation Service Created and Ready to Use

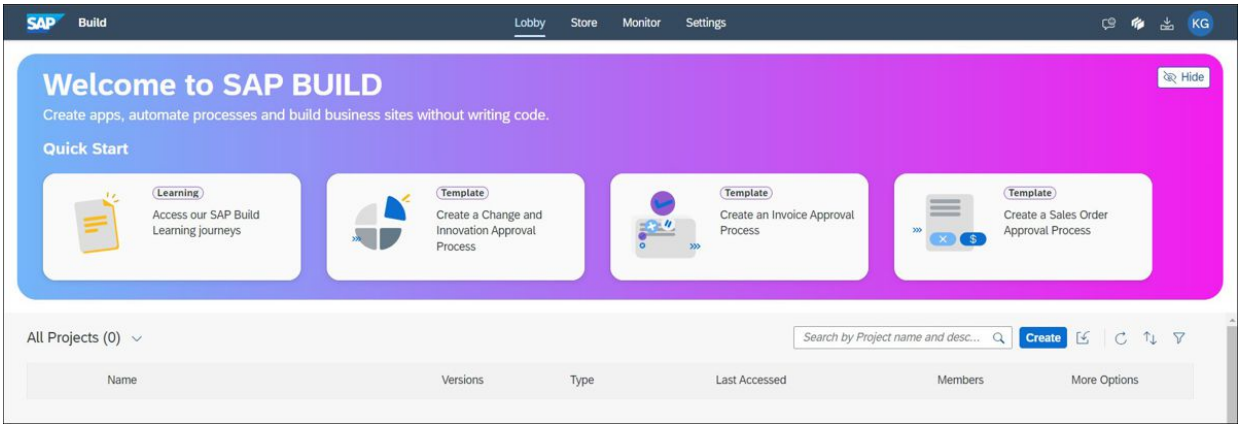


Figure 17.11 SAP Build Process Automation: Tool Landing Page after Login to Service

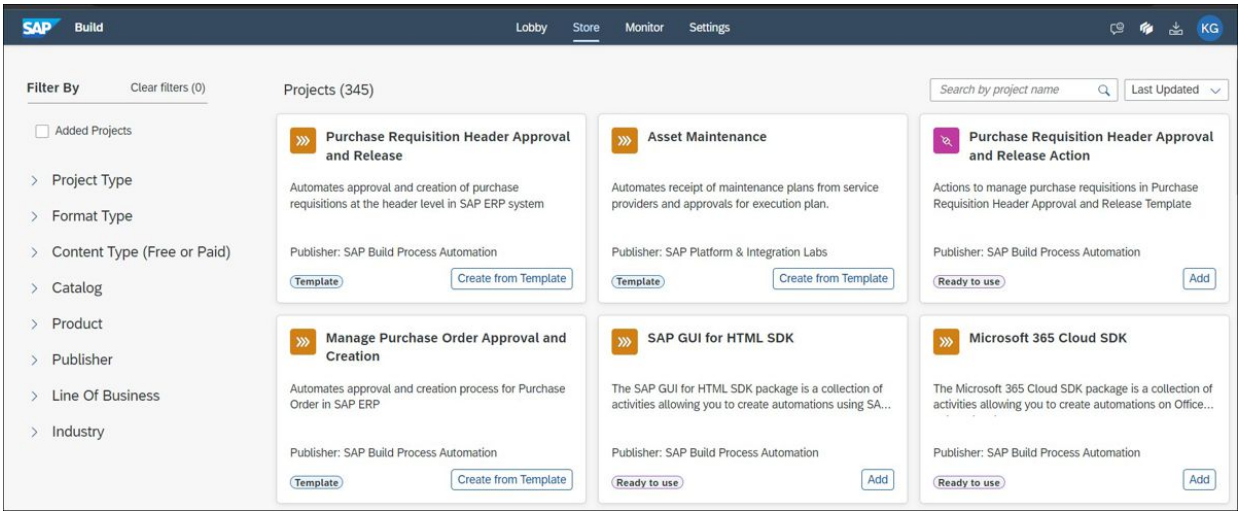


Figure 17.12 SAP Build Process Automation: Store

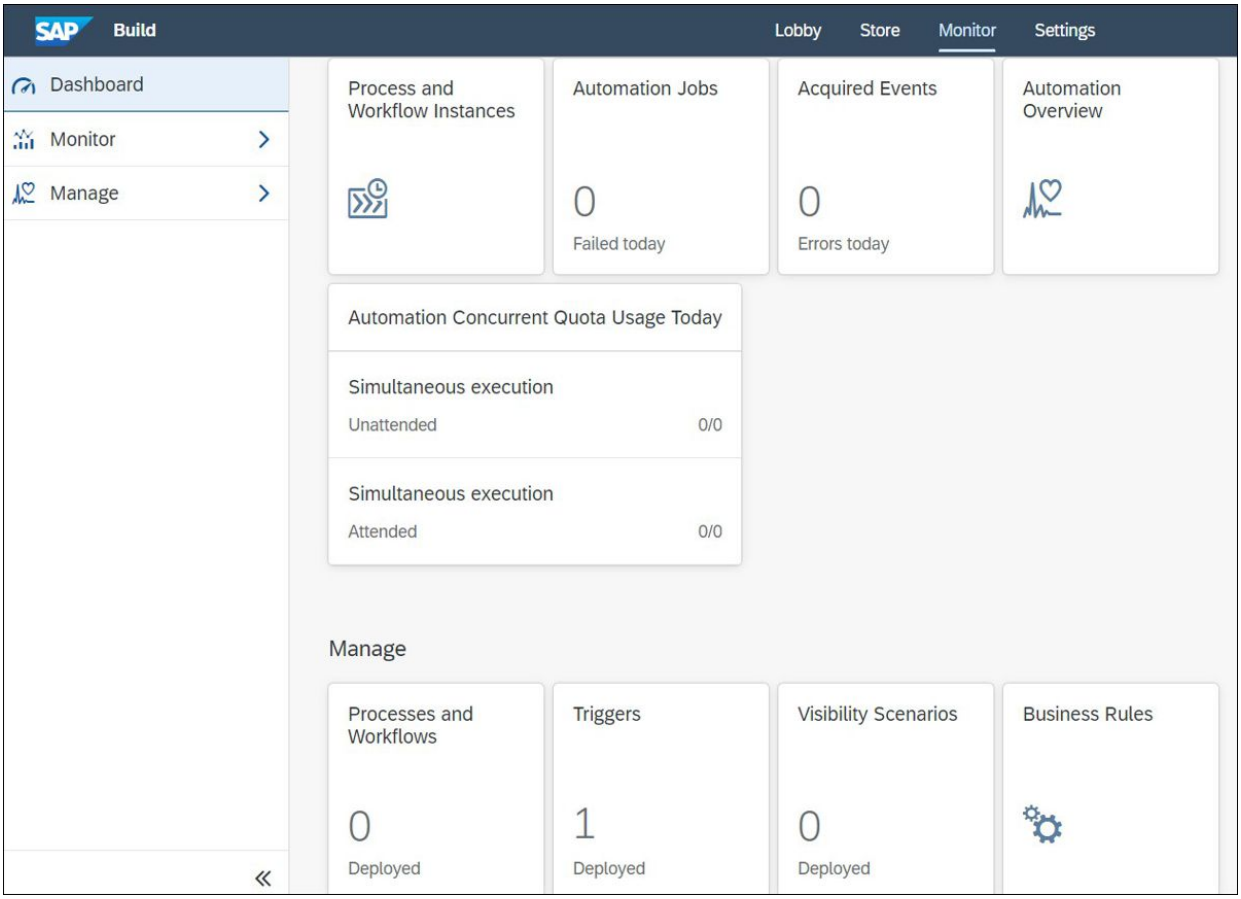


Figure 17.13 Monitor Workflow: Manage Business Rules and Process Visibility

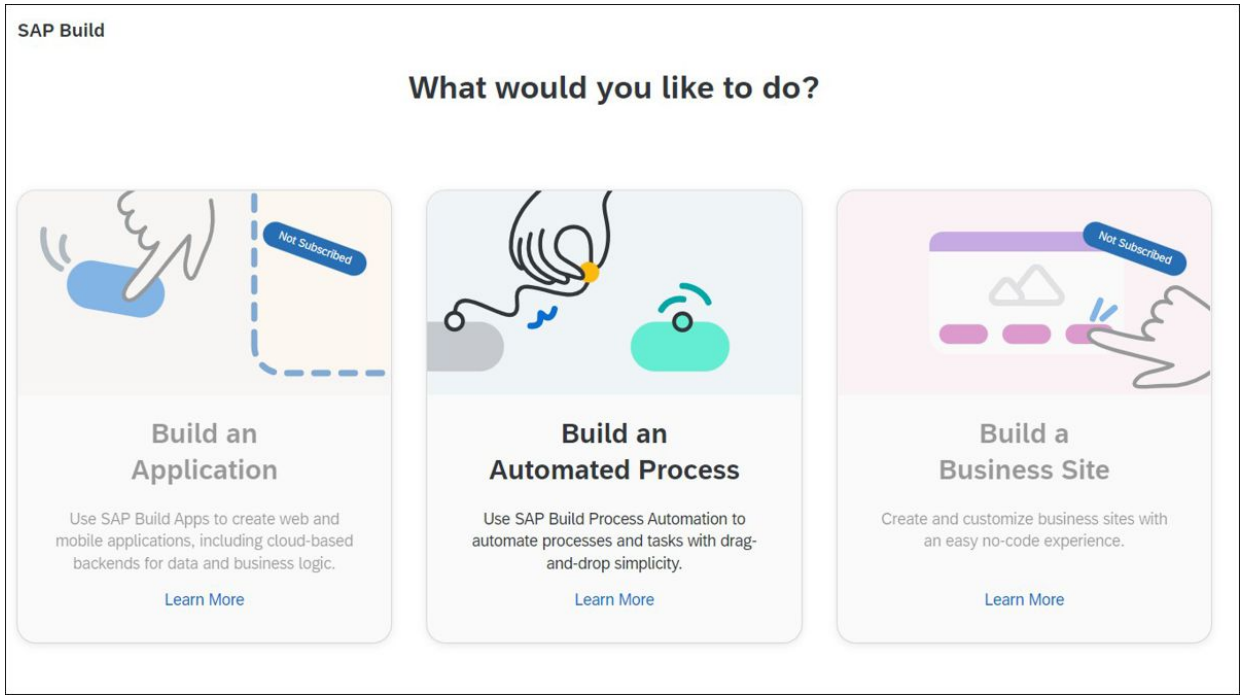


Figure 17.14 Select Build an Automated Process

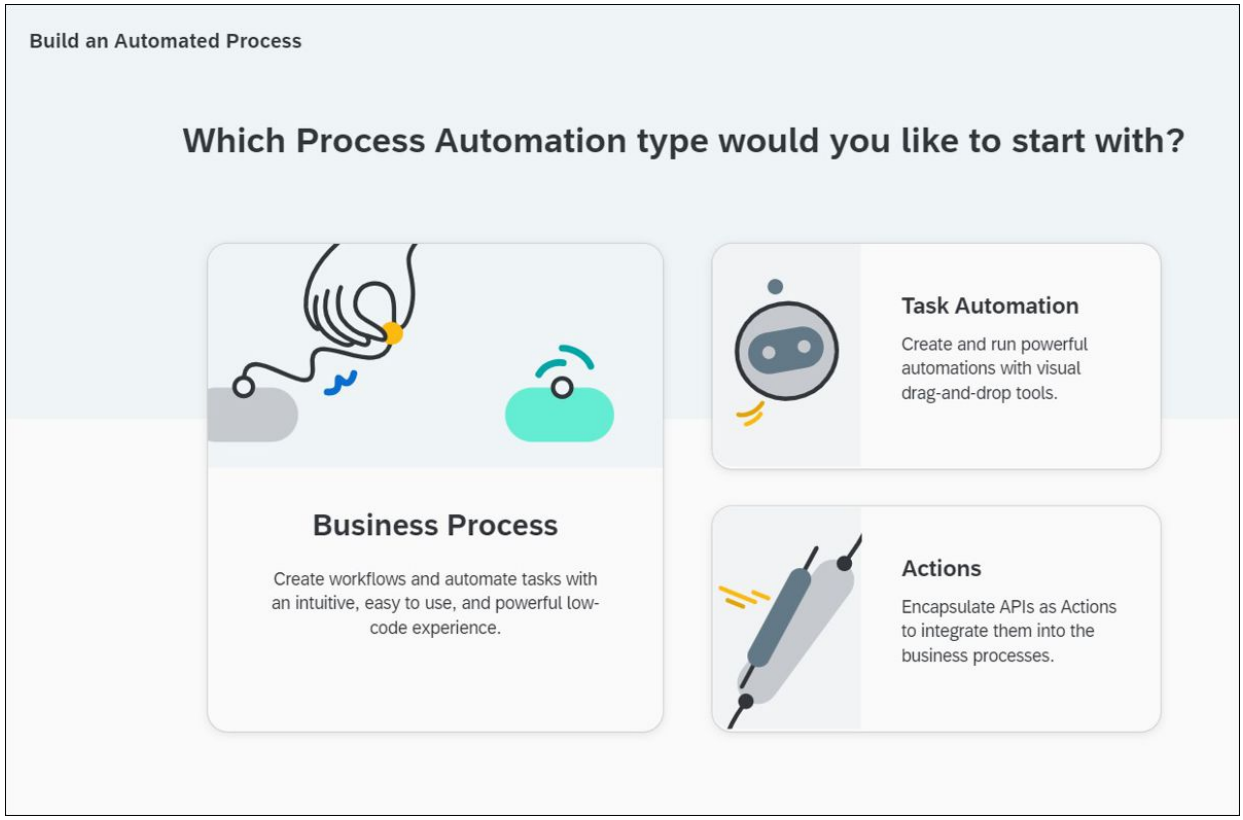


Figure 17.15 Select Business Process to Start a Workflow Project

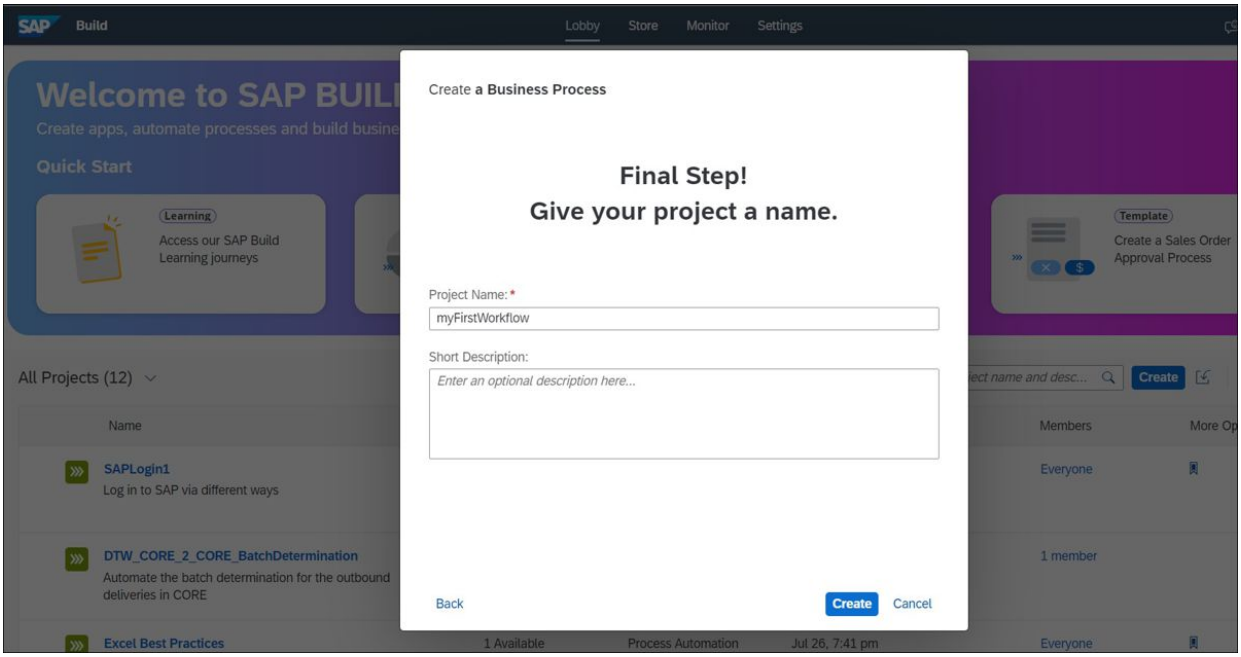


Figure 17.16 Name Your Project

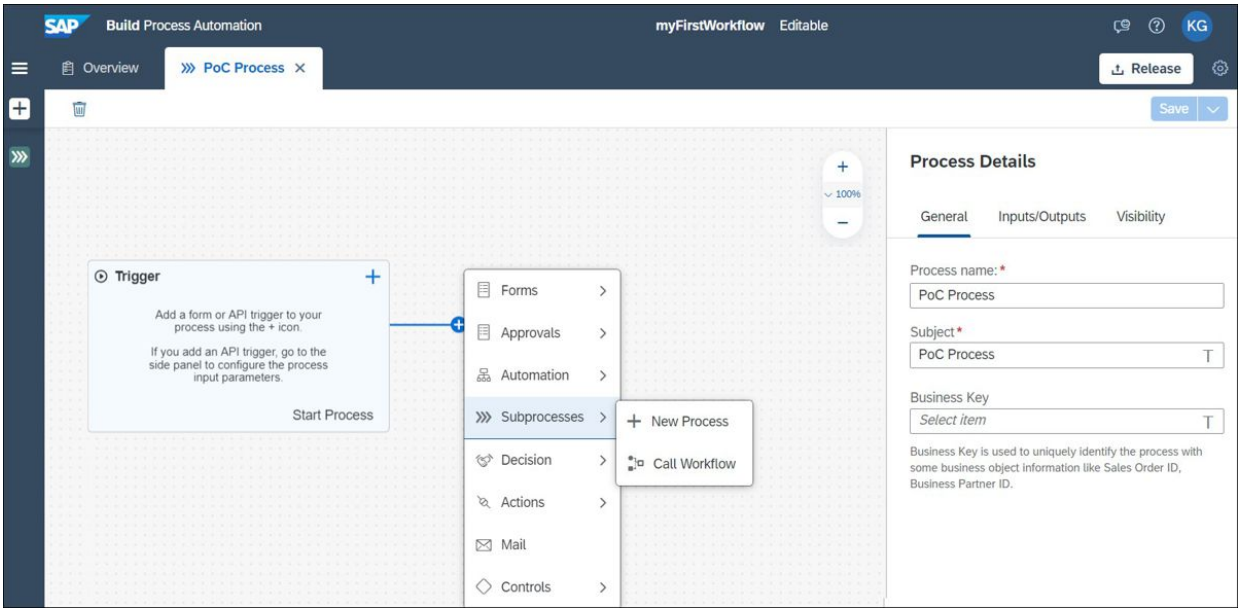


Figure 17.17 Workflow Editor to Start Creating the Workflow

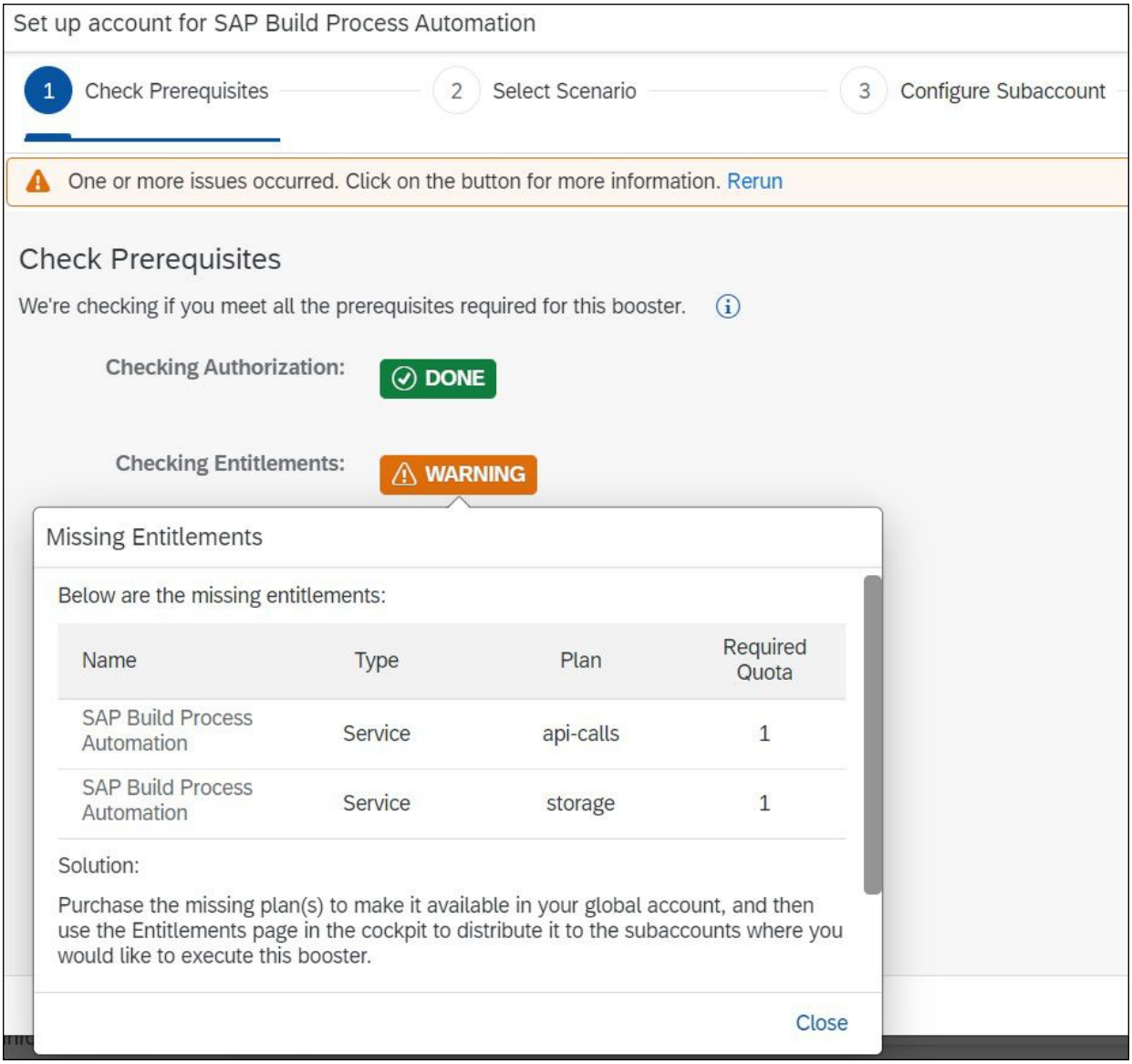


Figure 17.18 Issue during Booster Setup

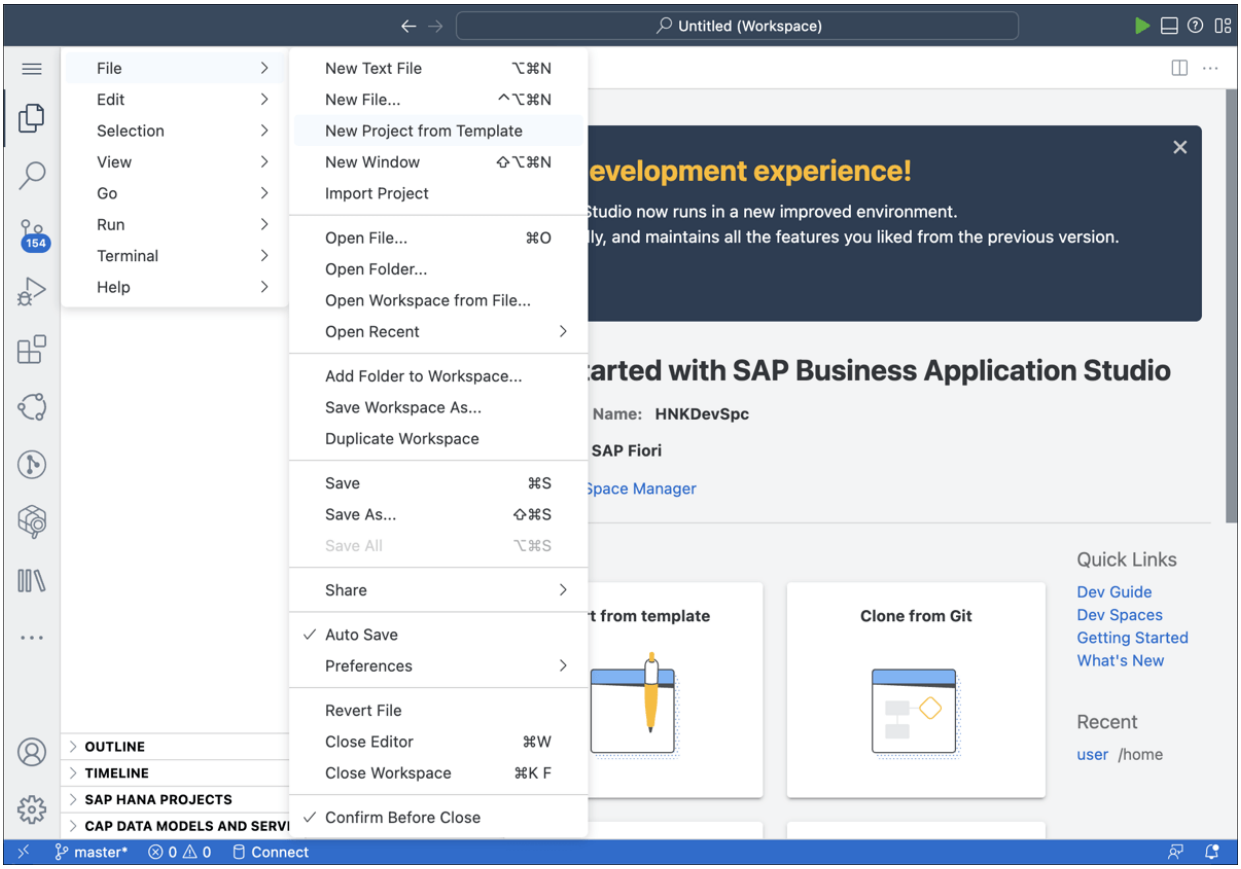


Figure 18.1 Create a Multi-Target Application Project in SAP Business Application Studio

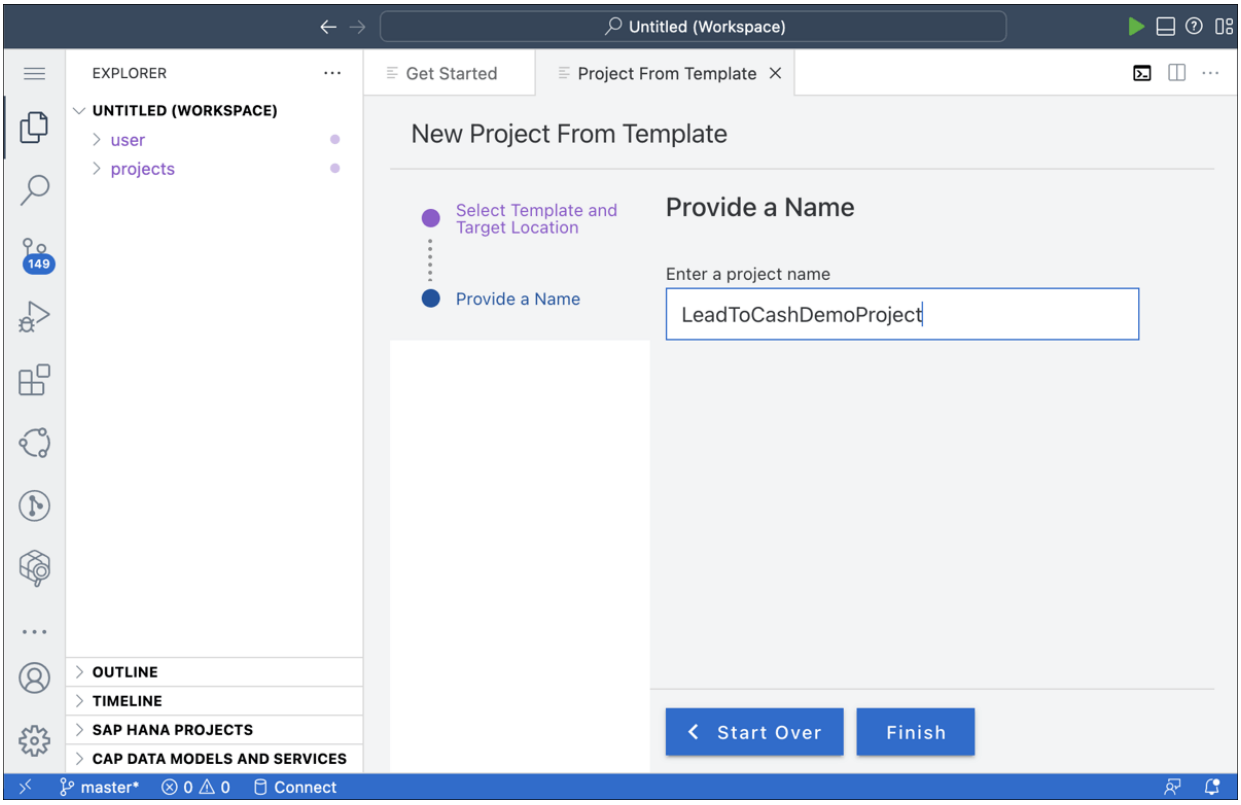


Figure 18.2 Create an MTA Project Folder

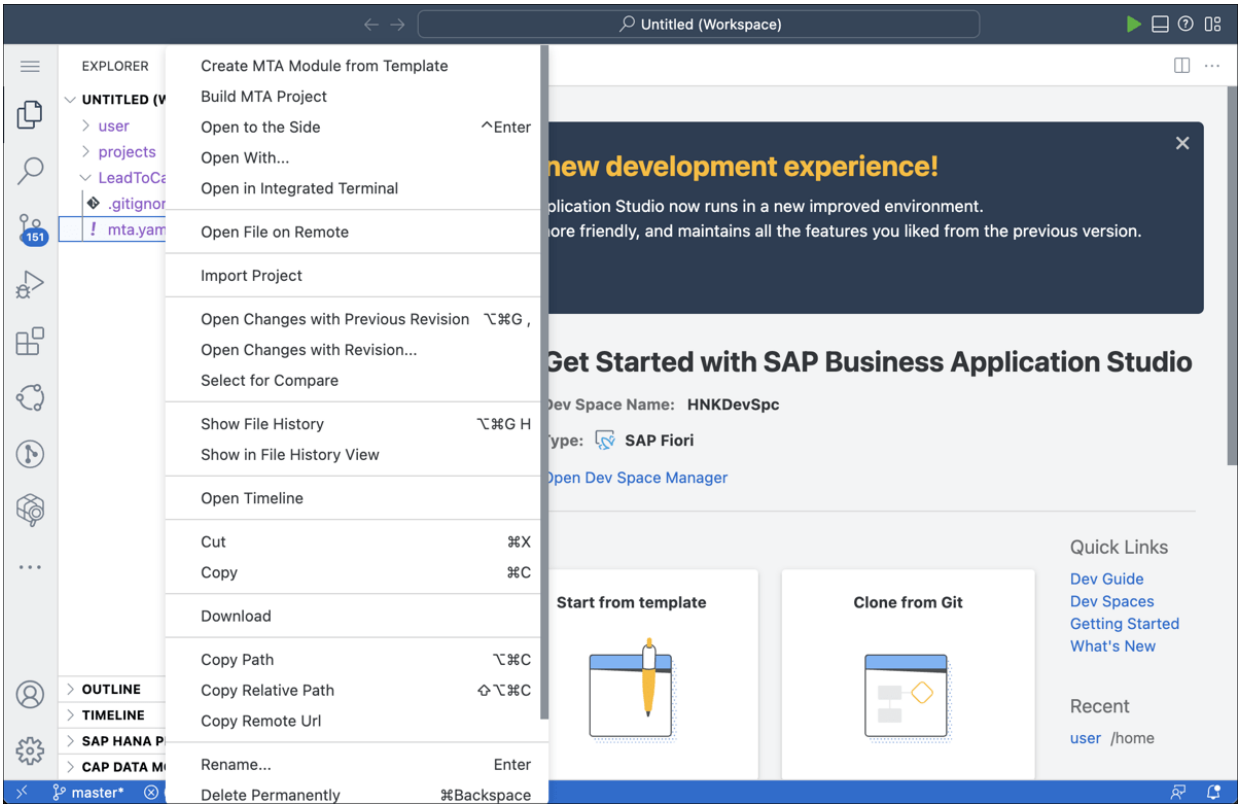


Figure 18.3 Create a Workflow Module (Part 1)

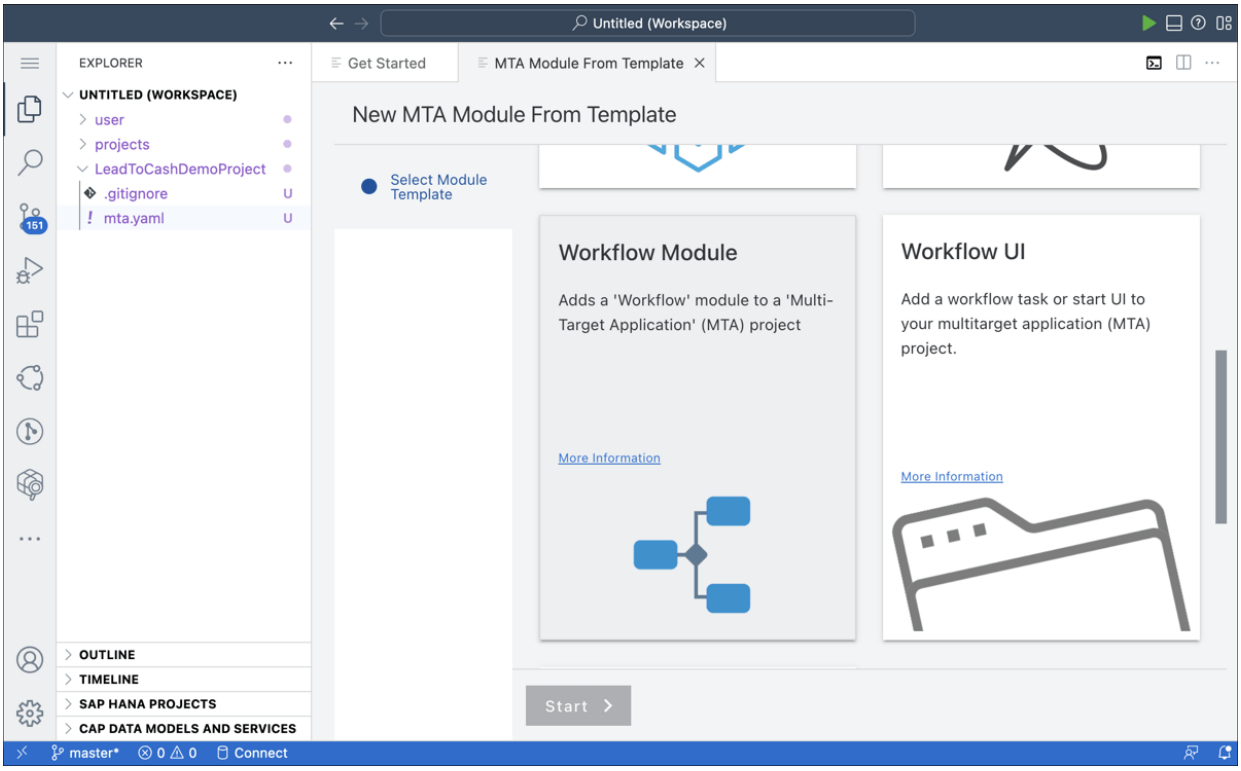


Figure 18.4 Create a Workflow Module (Part 2)

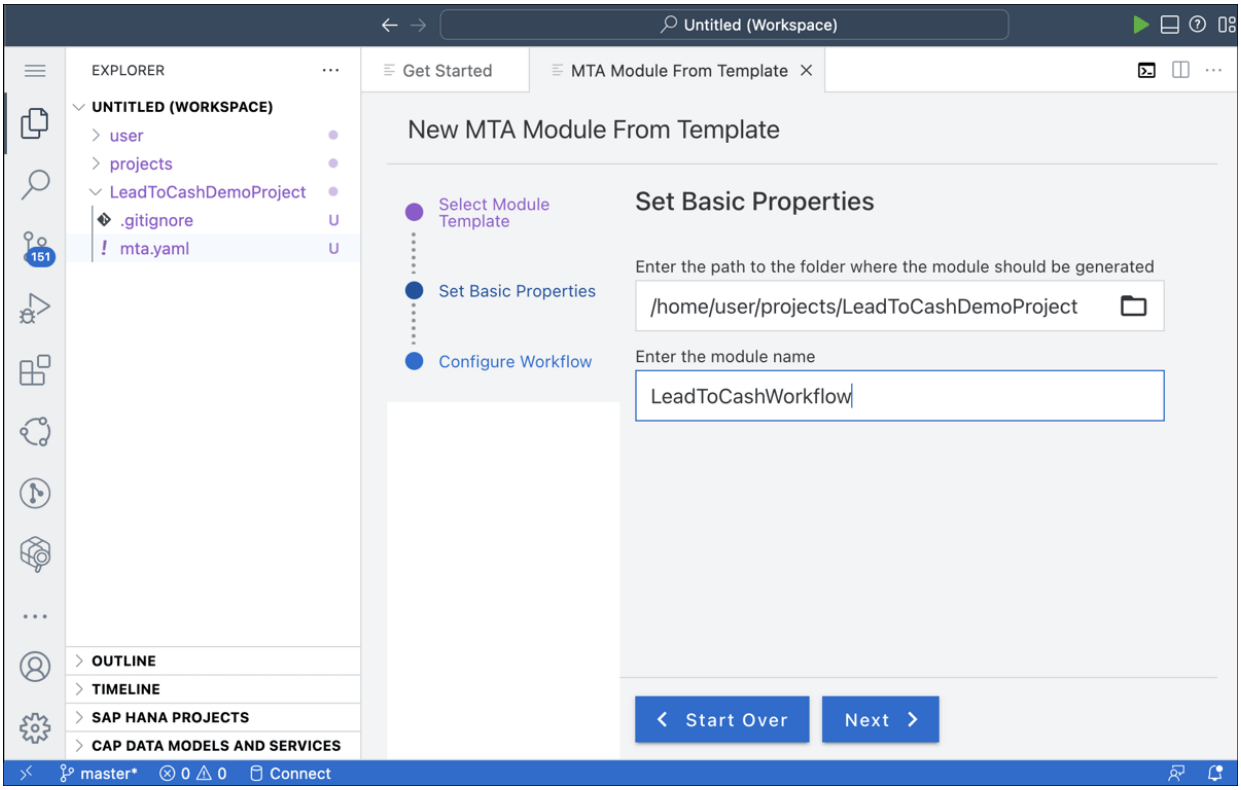


Figure 18.5 Name Your Workflow Module

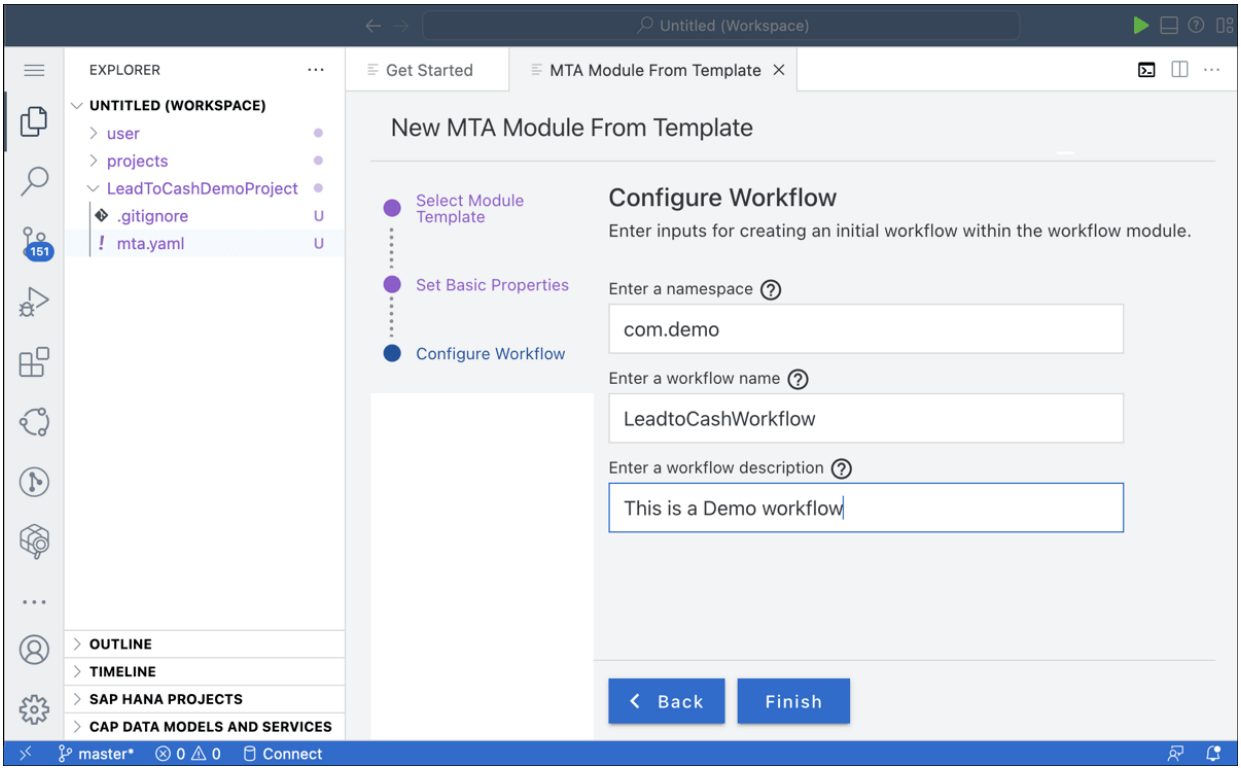


Figure 18.6 Provide a Namespace, Workflow Name, and Description

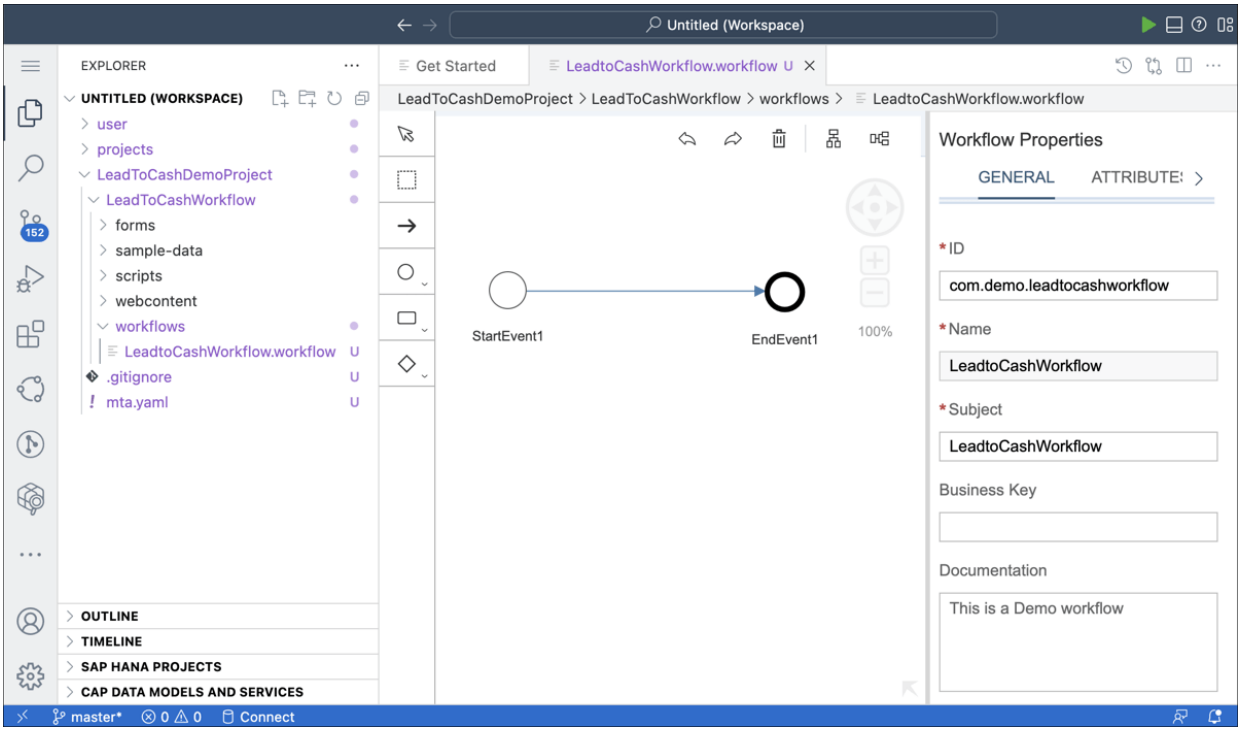


Figure 18.7 Workflow Editor

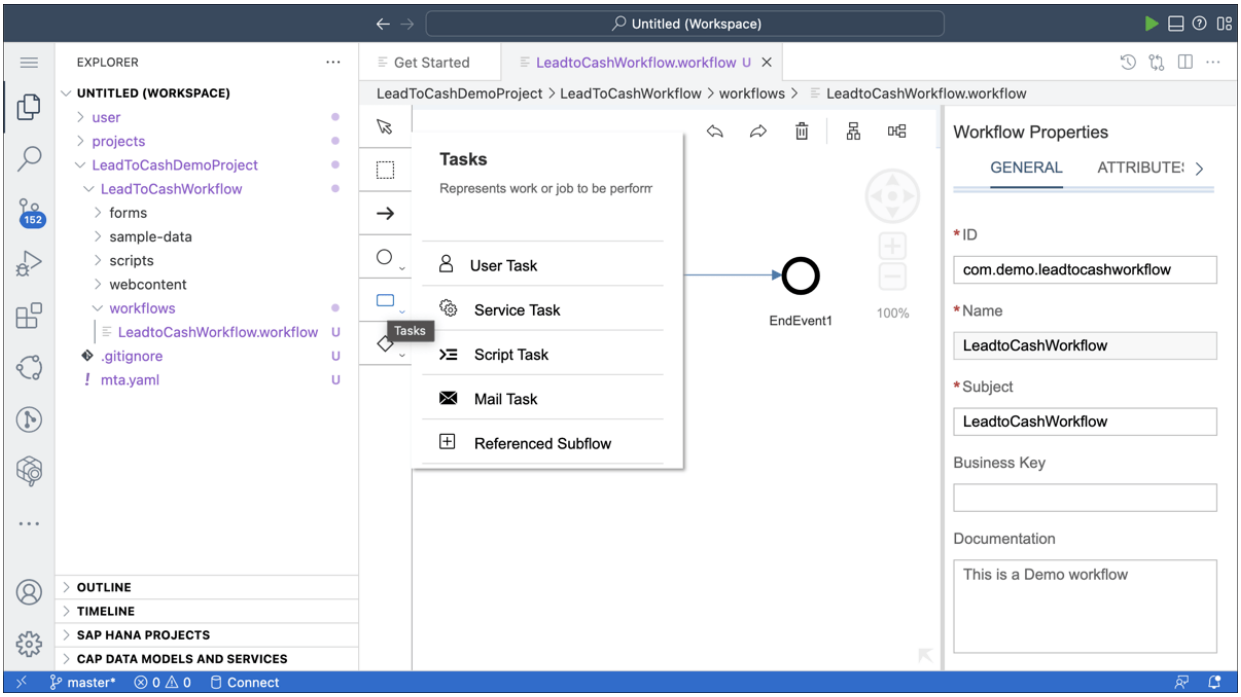


Figure 18.8 Create a Script Task

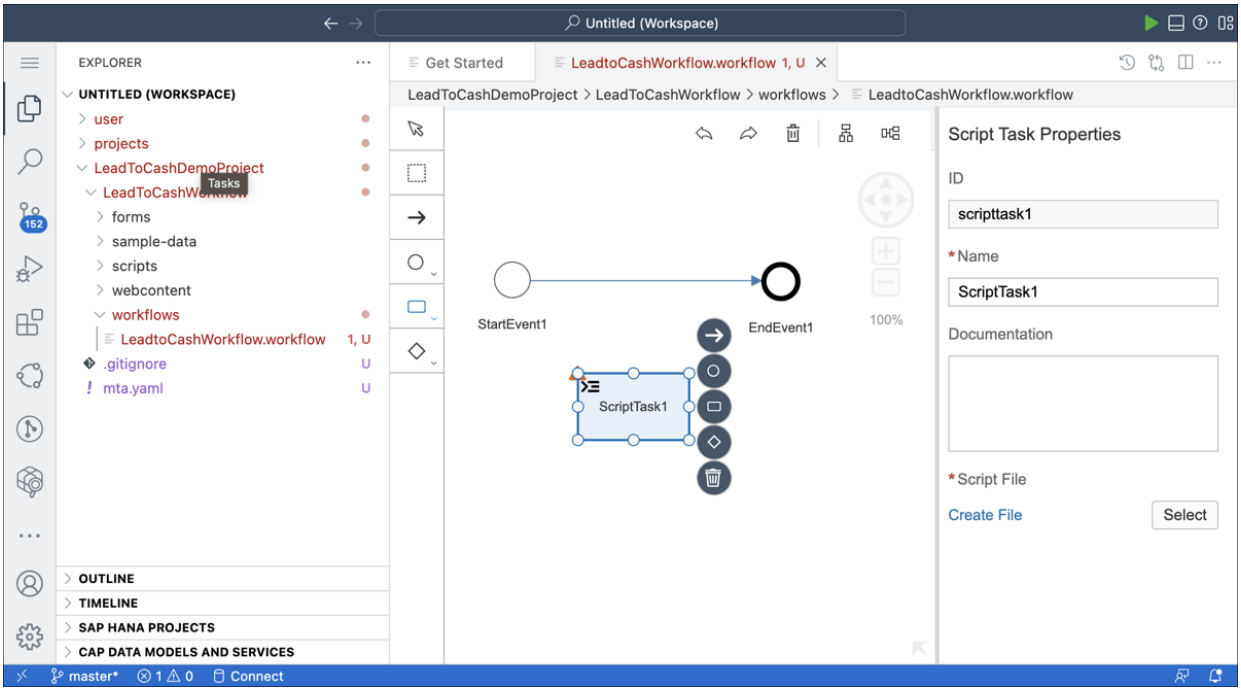


Figure 18.9 Script Task Created

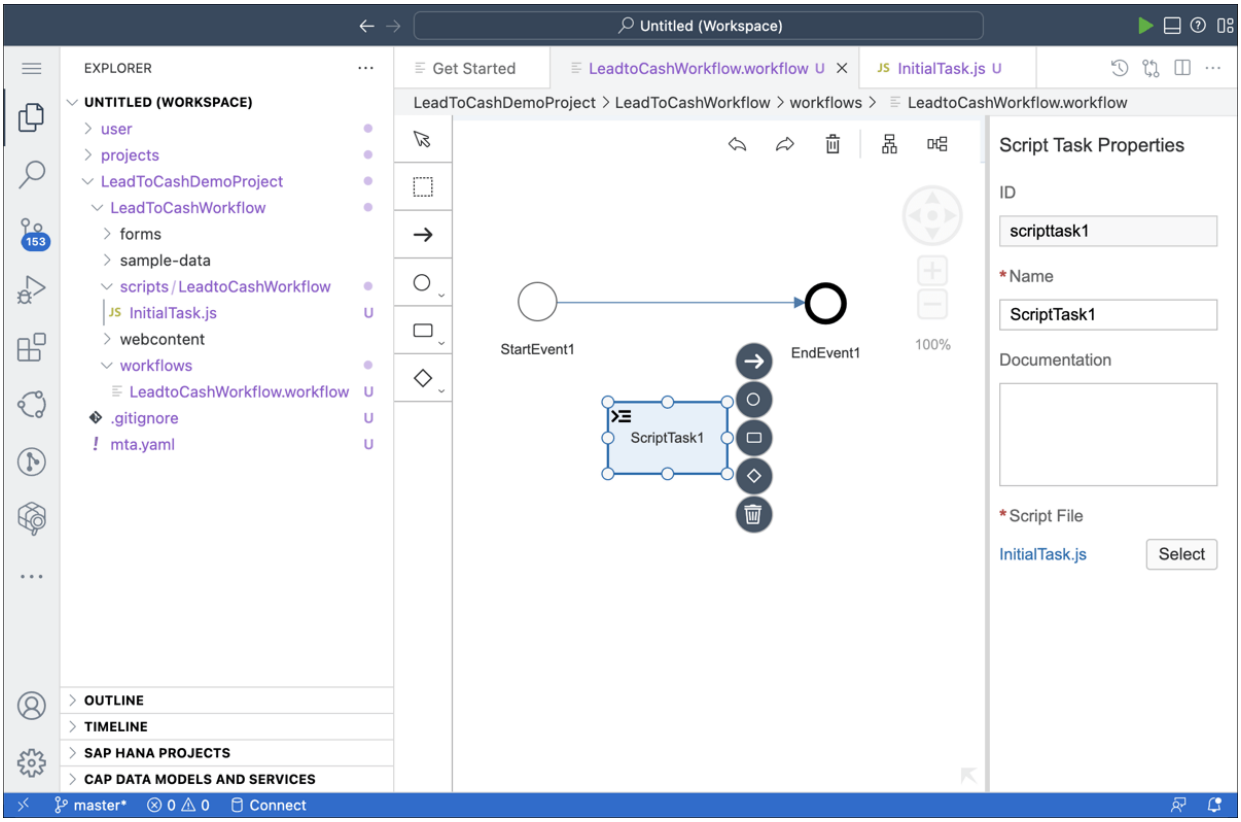


Figure 18.10 Creating a .js File for the Script Task

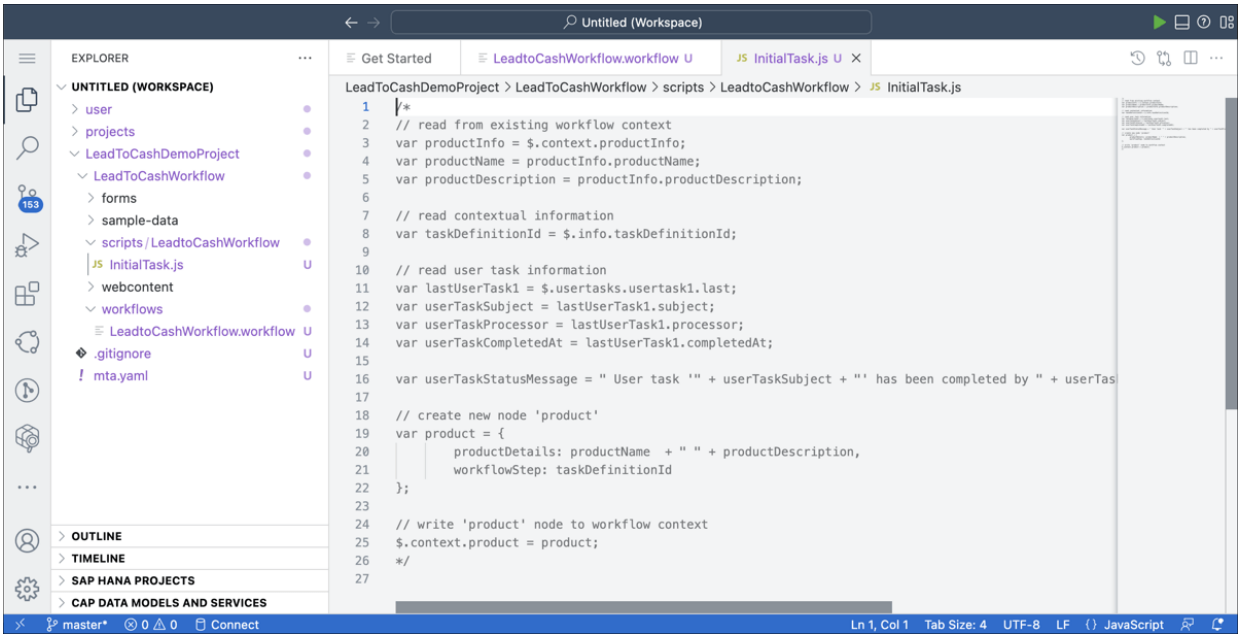


Figure 18.11 Write the Data Handling Logic in the .js File

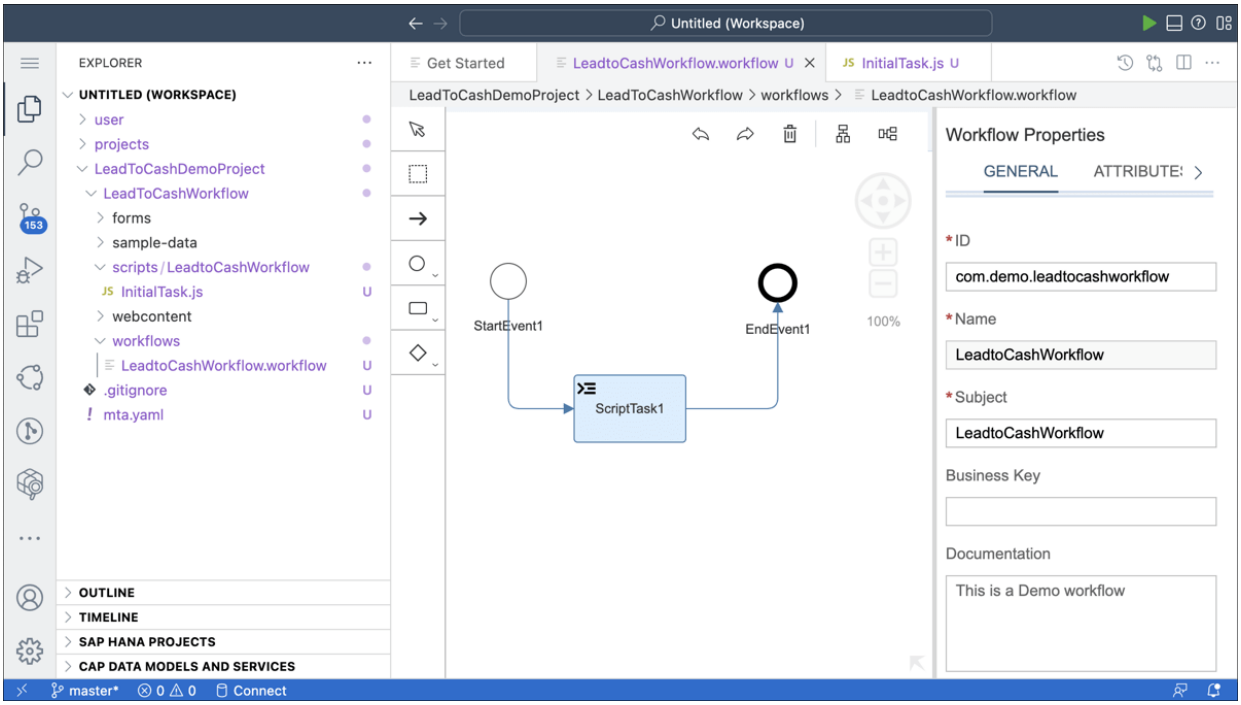


Figure 18.12 Workflow with a Script Task

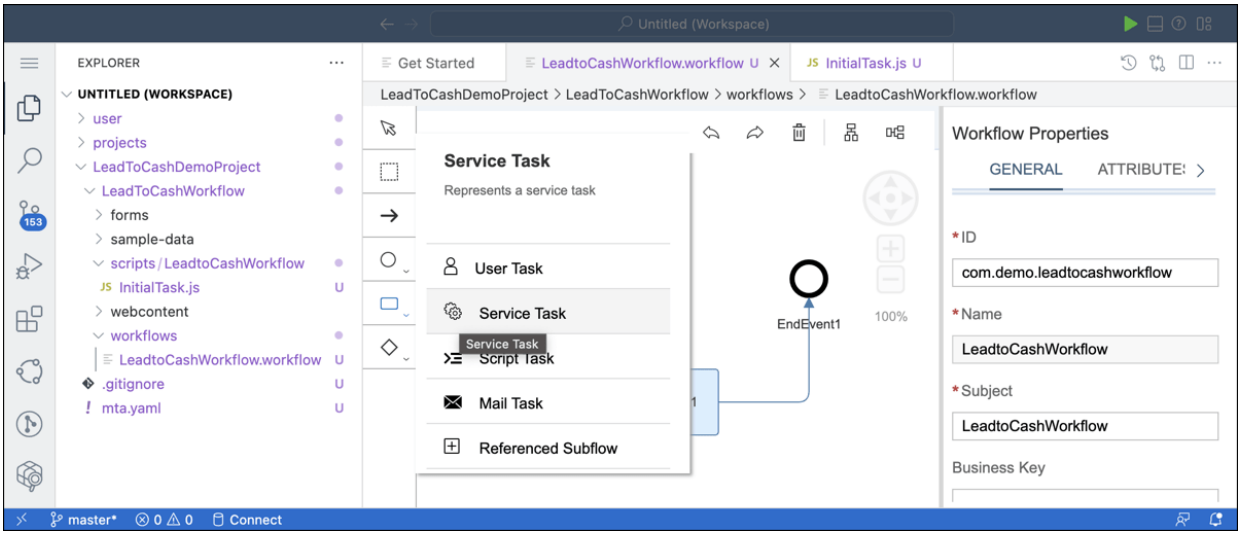


Figure 18.13 Create a Service Task

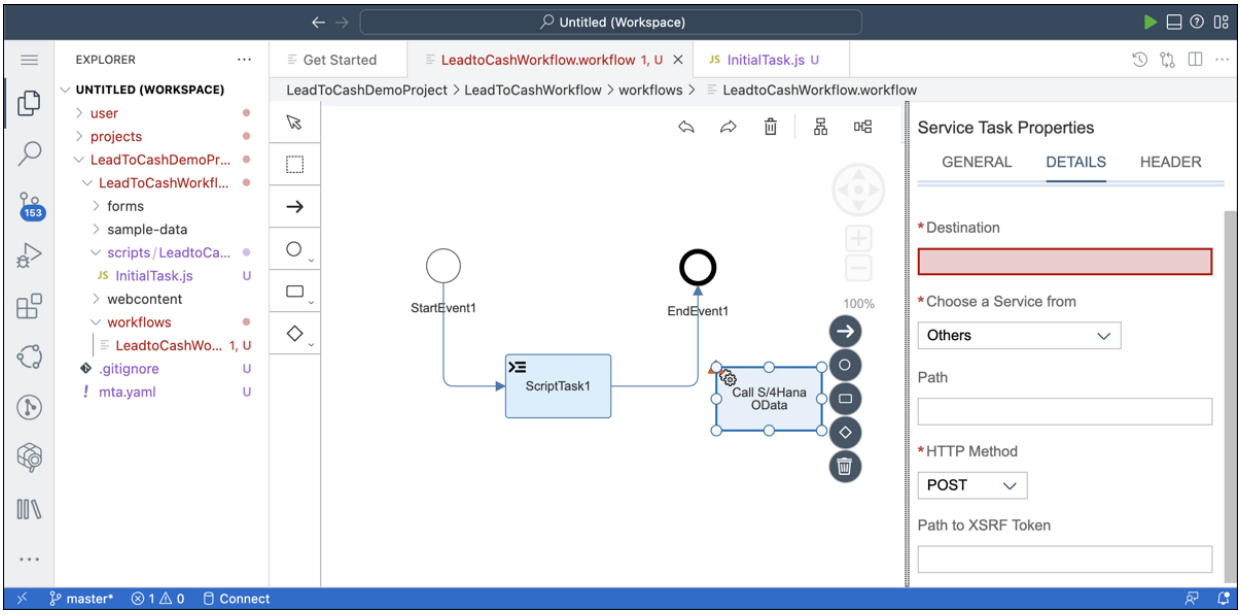


Figure 18.14 Configure the Service Task

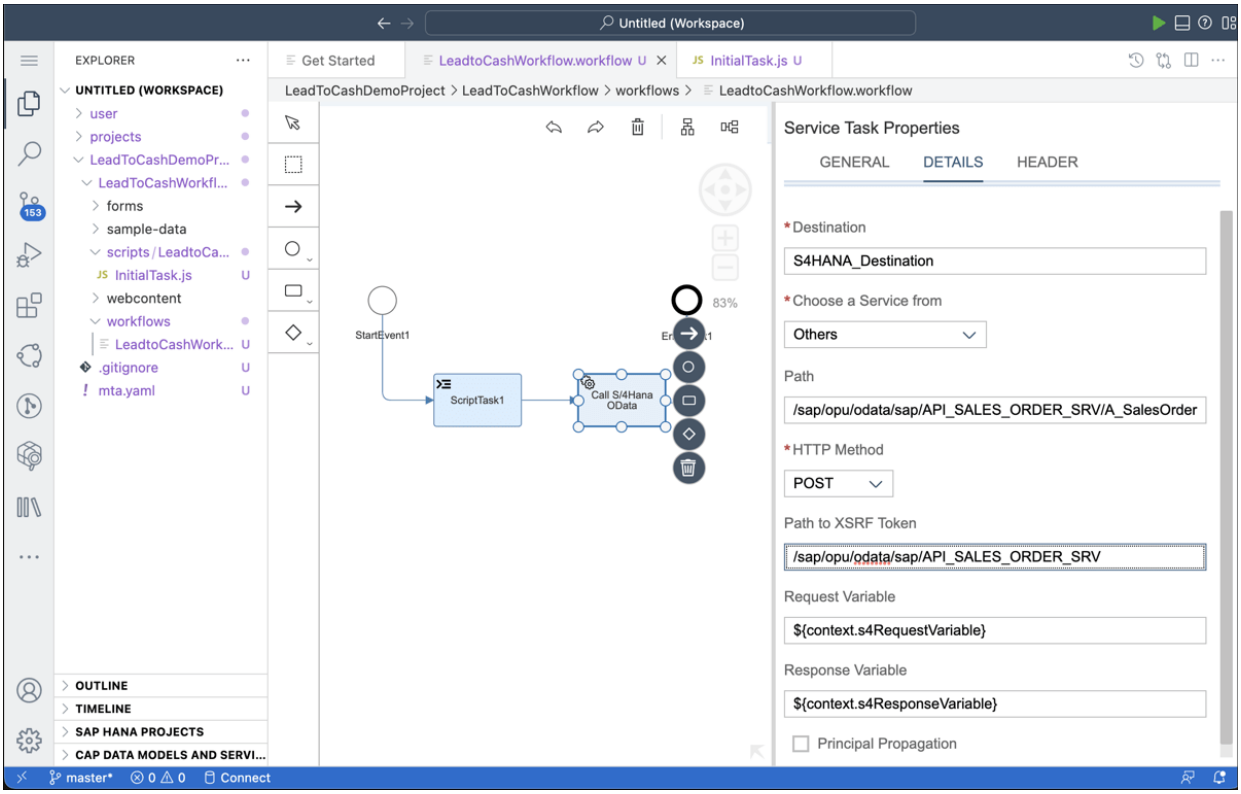


Figure 18.15 Configured Service Task

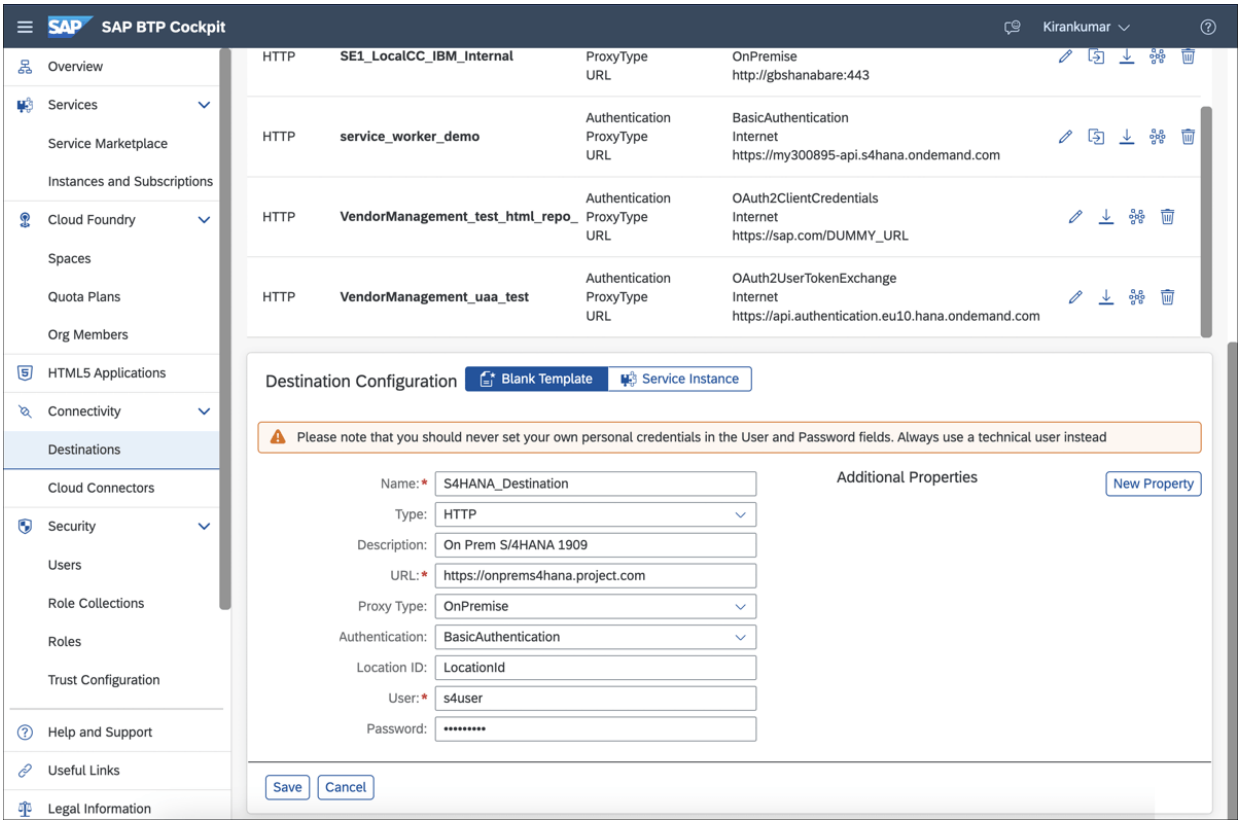


Figure 18.16 Creating a Destination

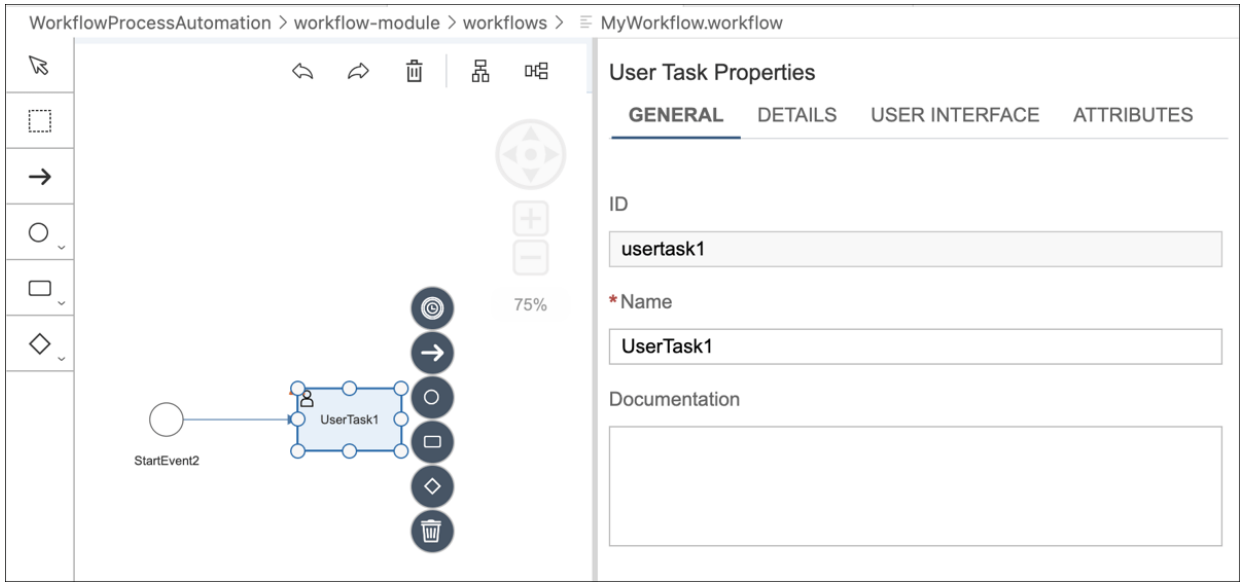


Figure 18.17 Create a User Task

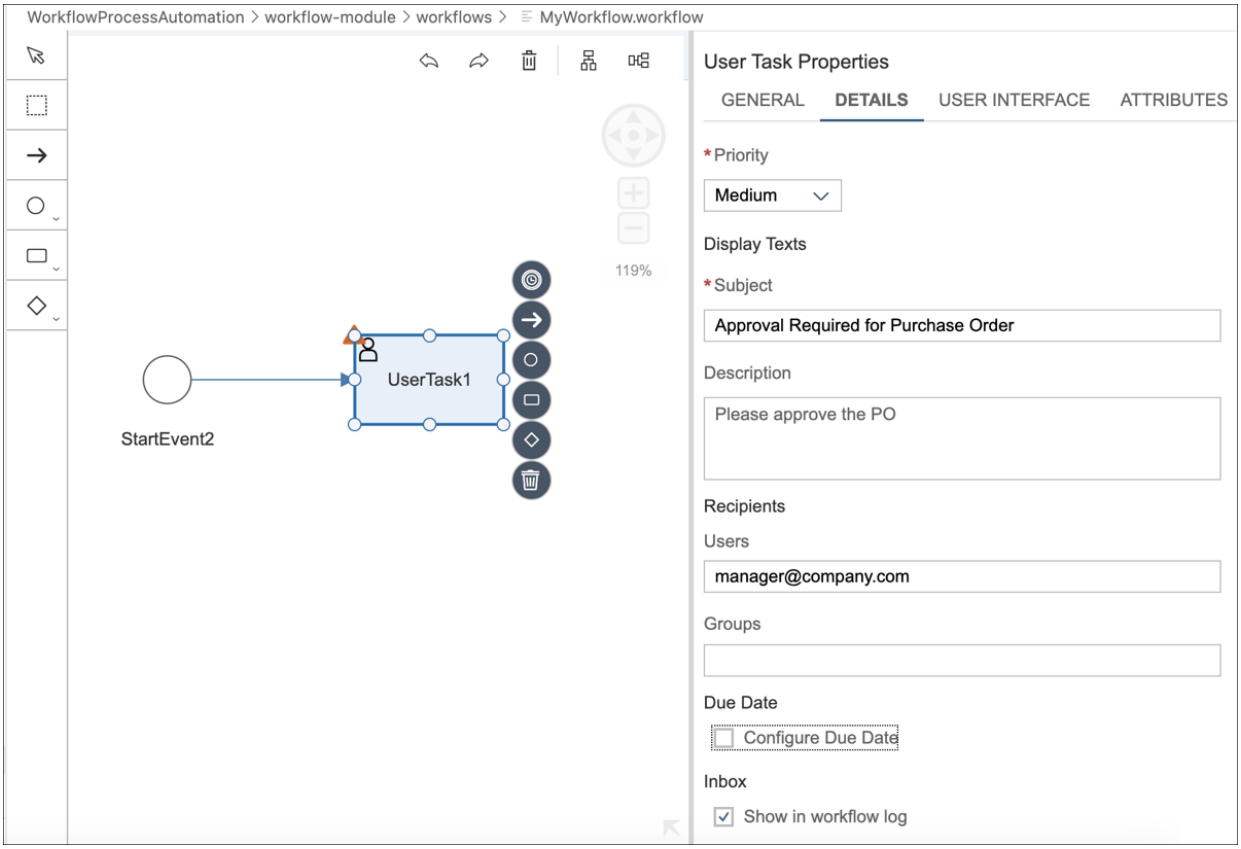


Figure 18.18 Configure Details Section of the User Task

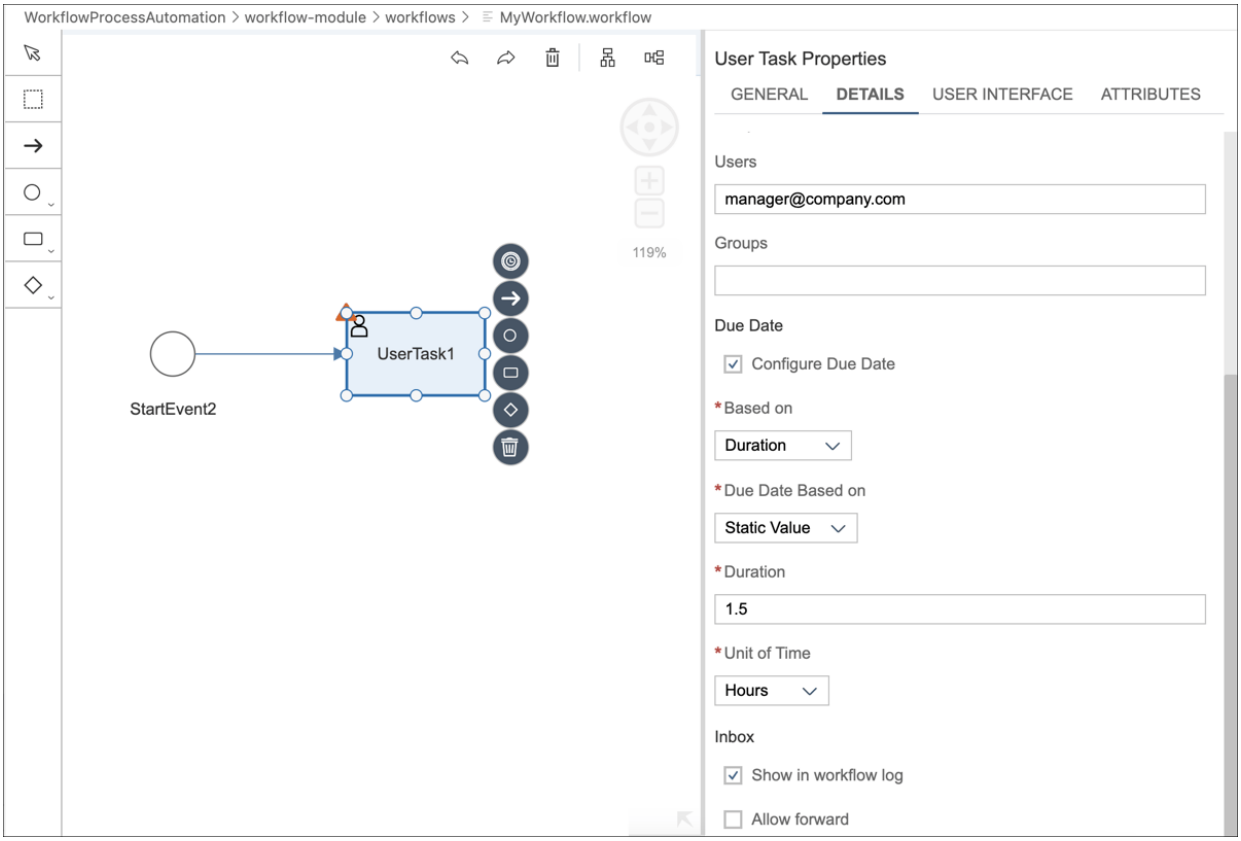


Figure 18.19 Set a Due Date for the User Task

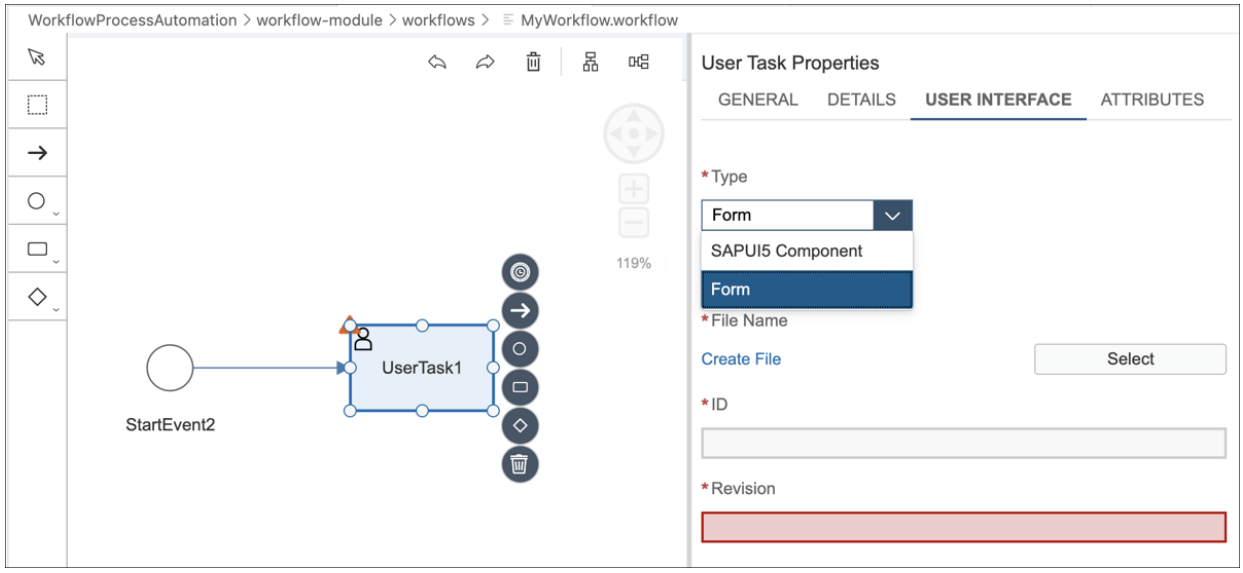


Figure 18.20 Types of Forms to Create for a User Interface

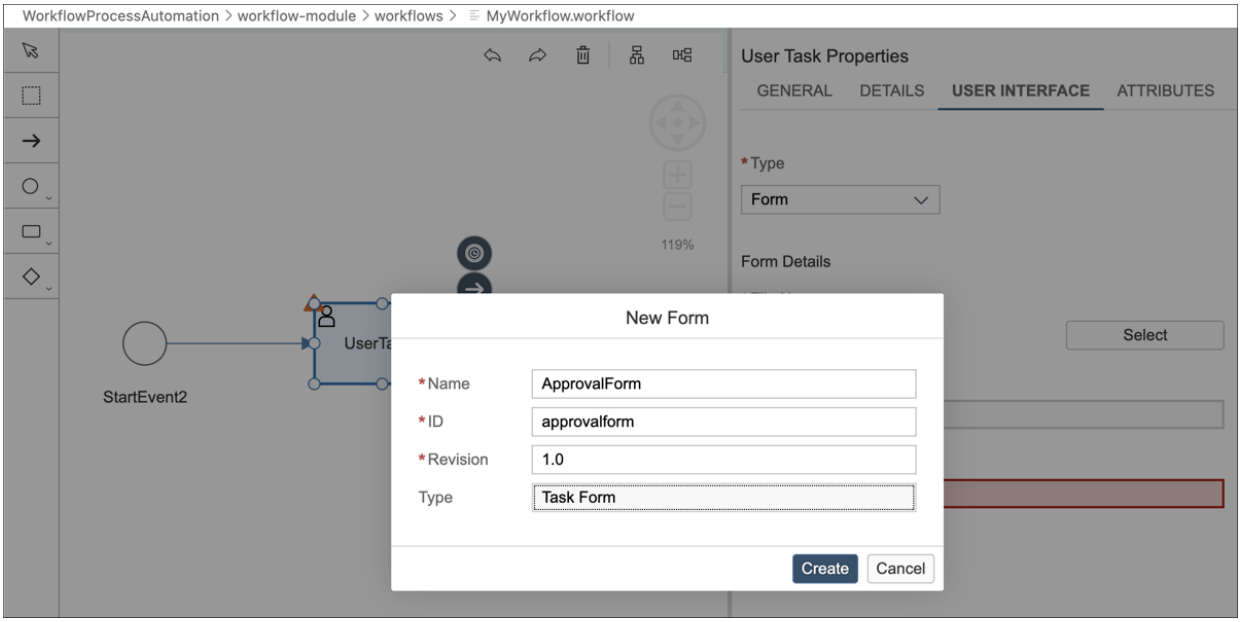


Figure 18.21 Create a New Form

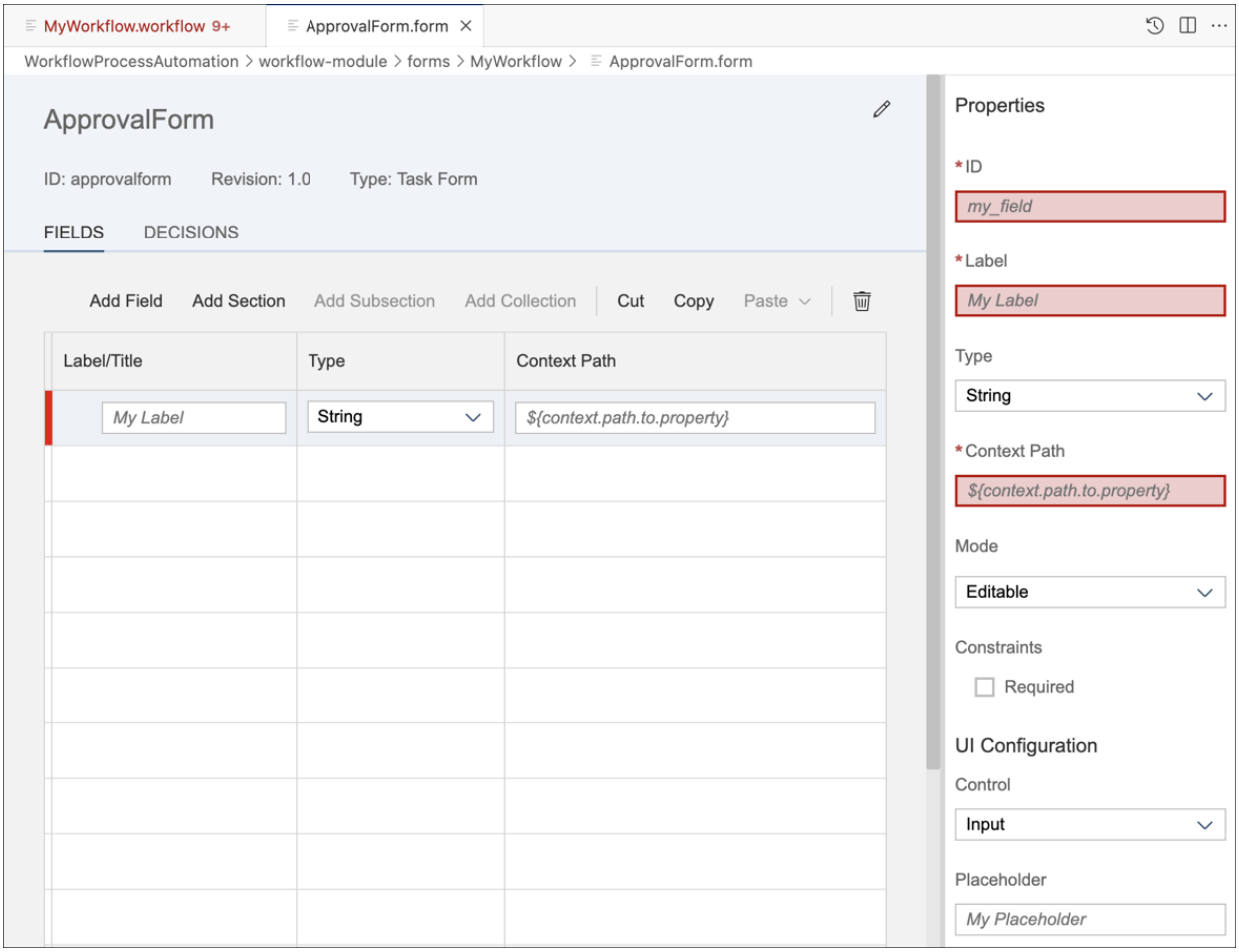


Figure 18.22 Editor Screen of the New Form

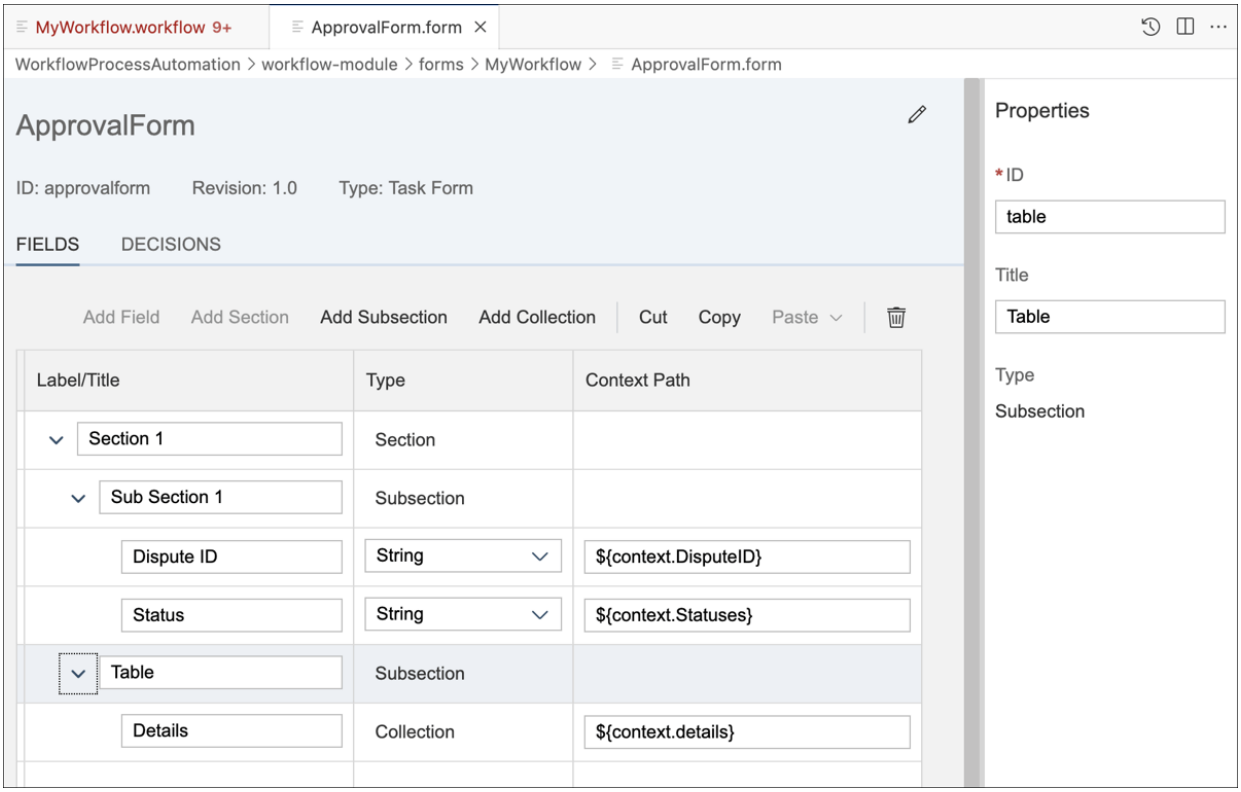


Figure 18.23 Adding Controls and Configuring the Form

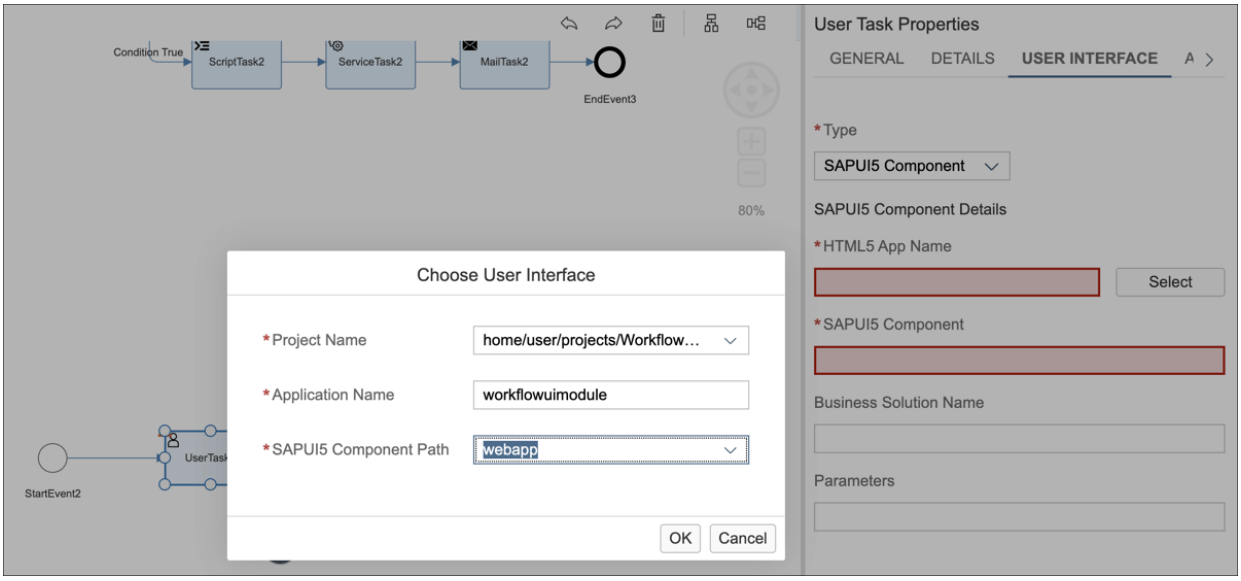


Figure 18.24 Adding an SAPUI5 Project as the UI for a User Task

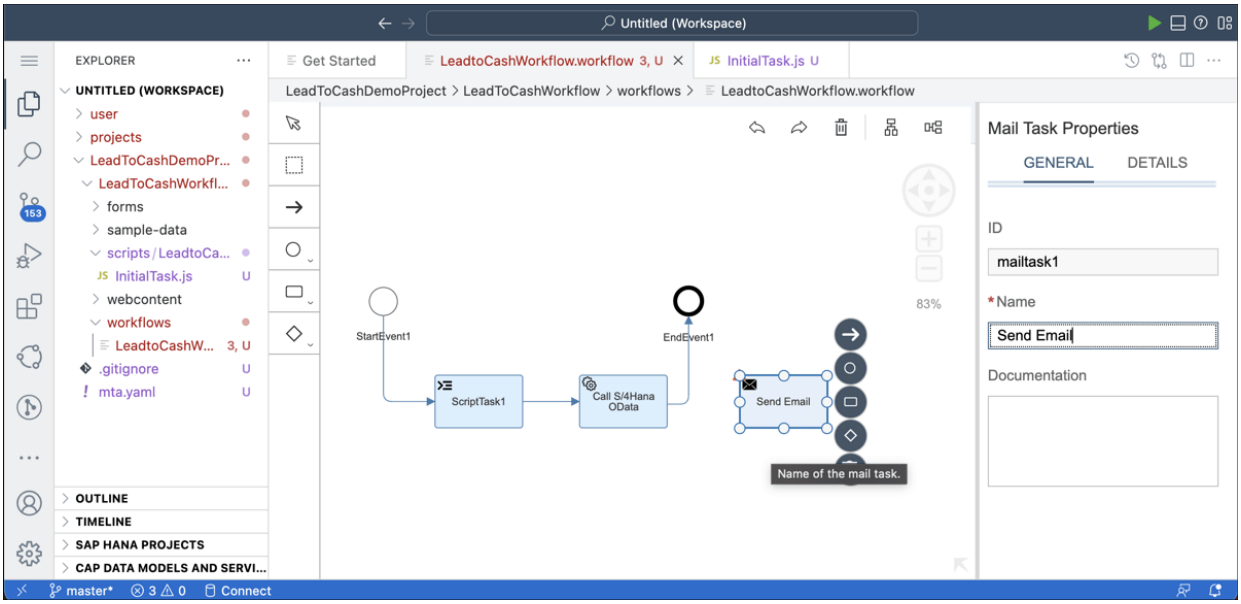


Figure 18.25 Create an Email Task

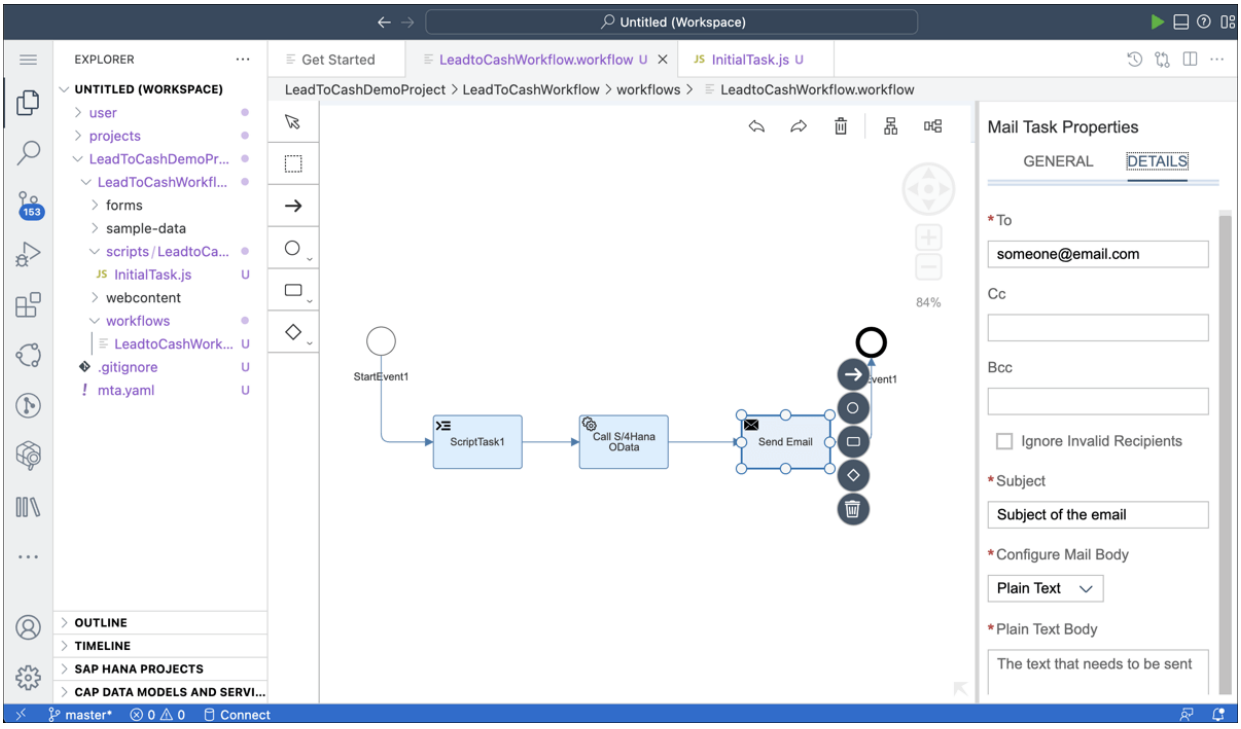


Figure 18.26 Configure the Email Task

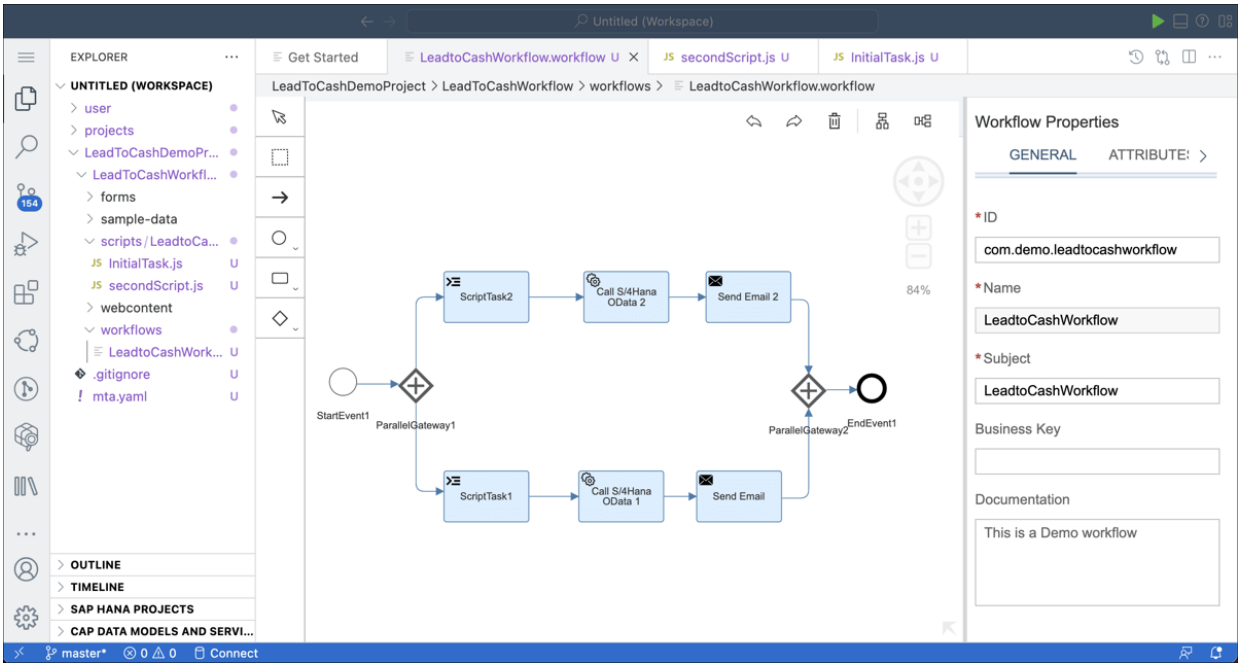


Figure 18.27 Configuring a Parallel Gateway

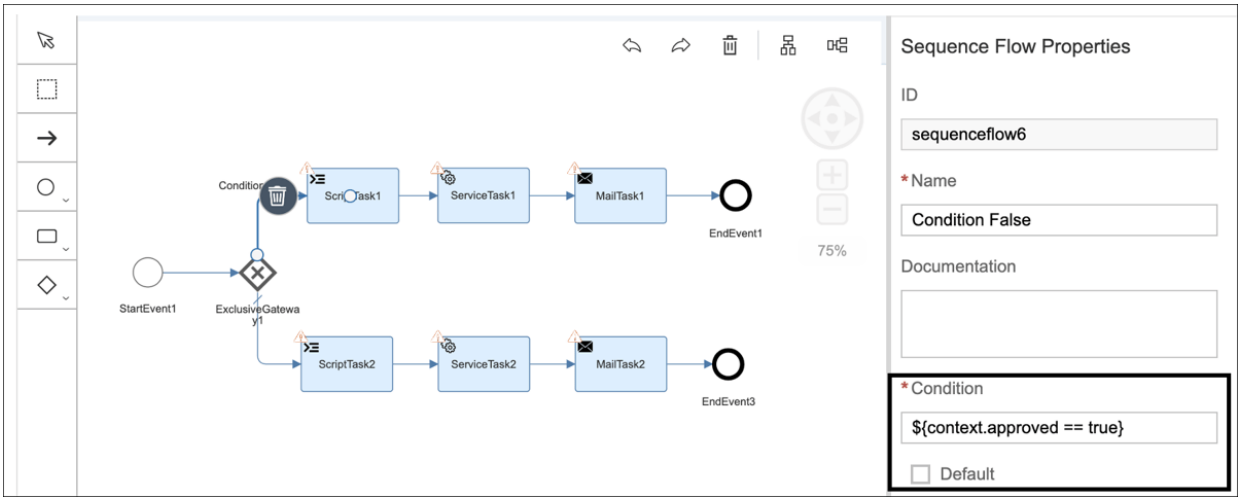


Figure 18.28 Configure the Sequence Condition for an Exclusive Gateway

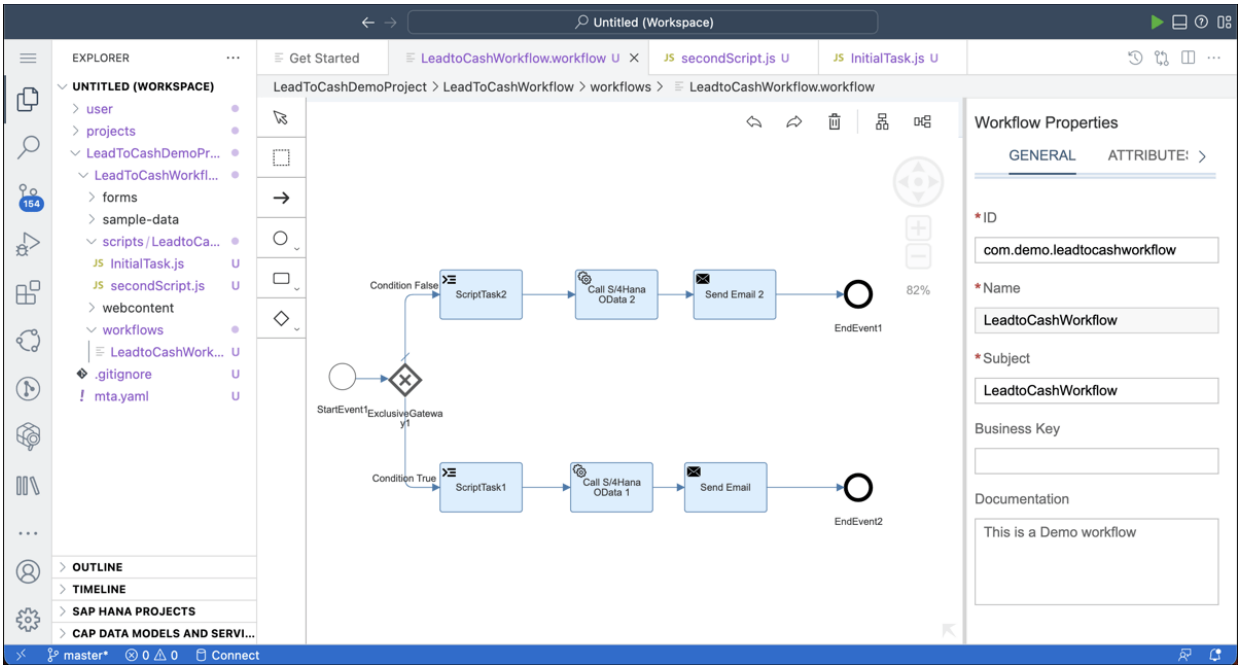


Figure 18.29 Configuring an Exclusive Gateway

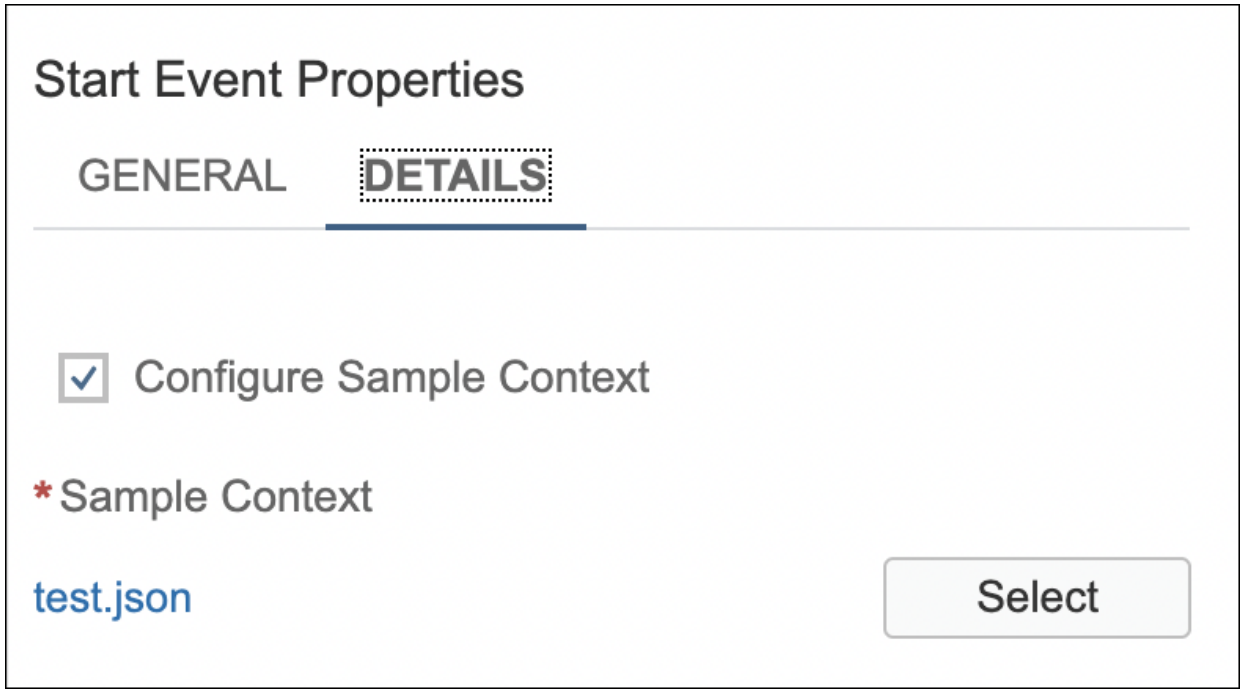


Figure 18.30 Setting a Sample Context for Start Event

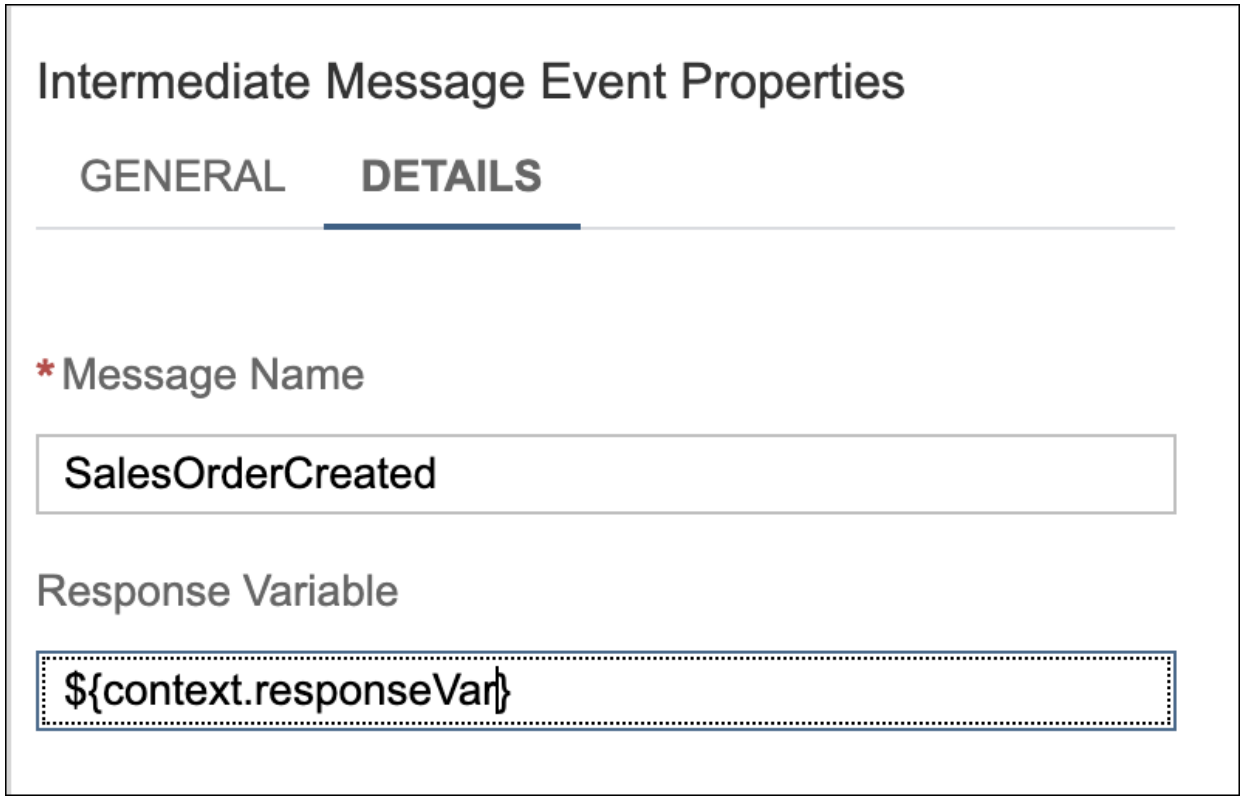


Figure 18.31 Configuring the Intermediate Event Message Properties

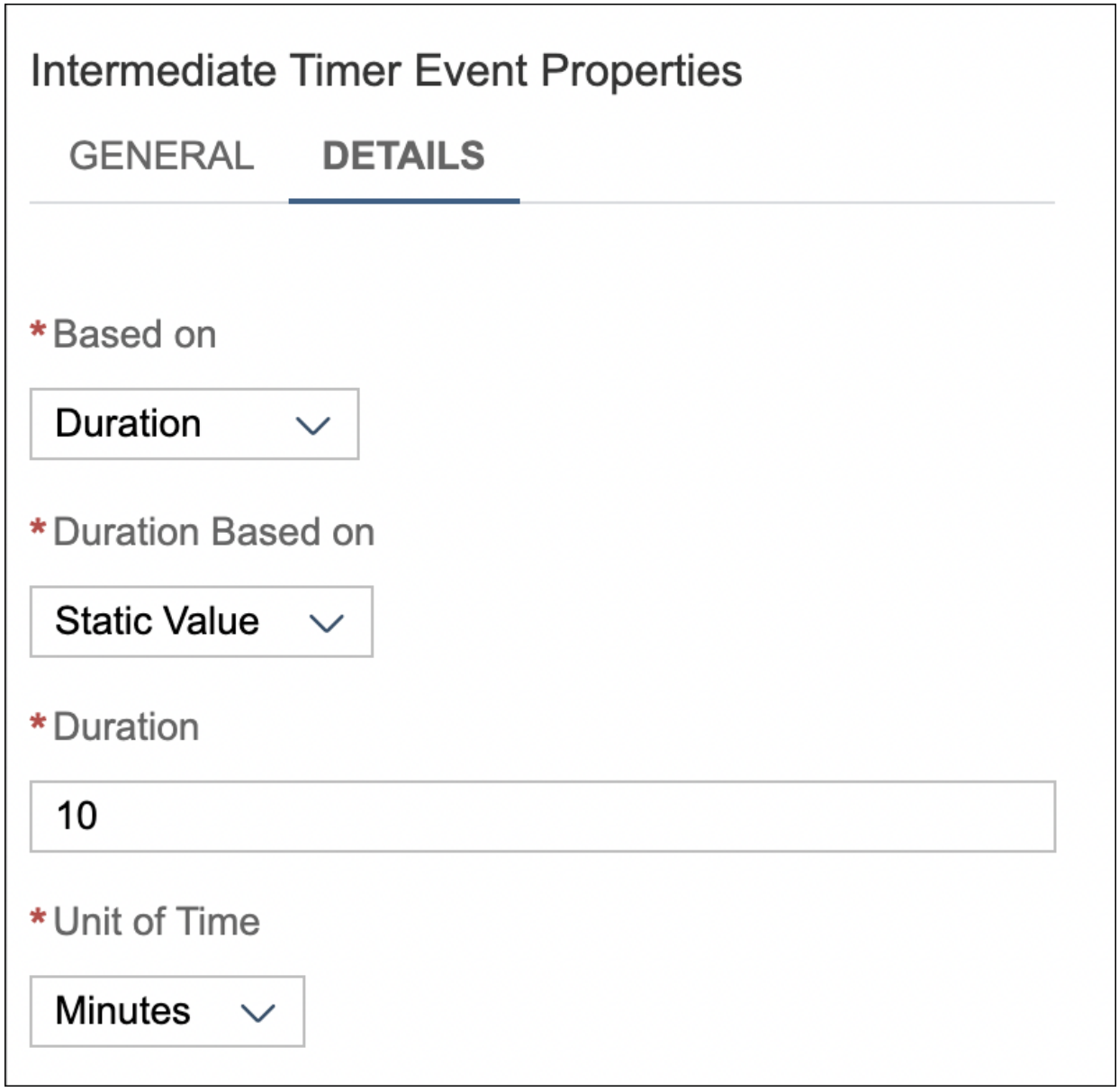


Figure 18.32 Configure the Intermediate Timer Event

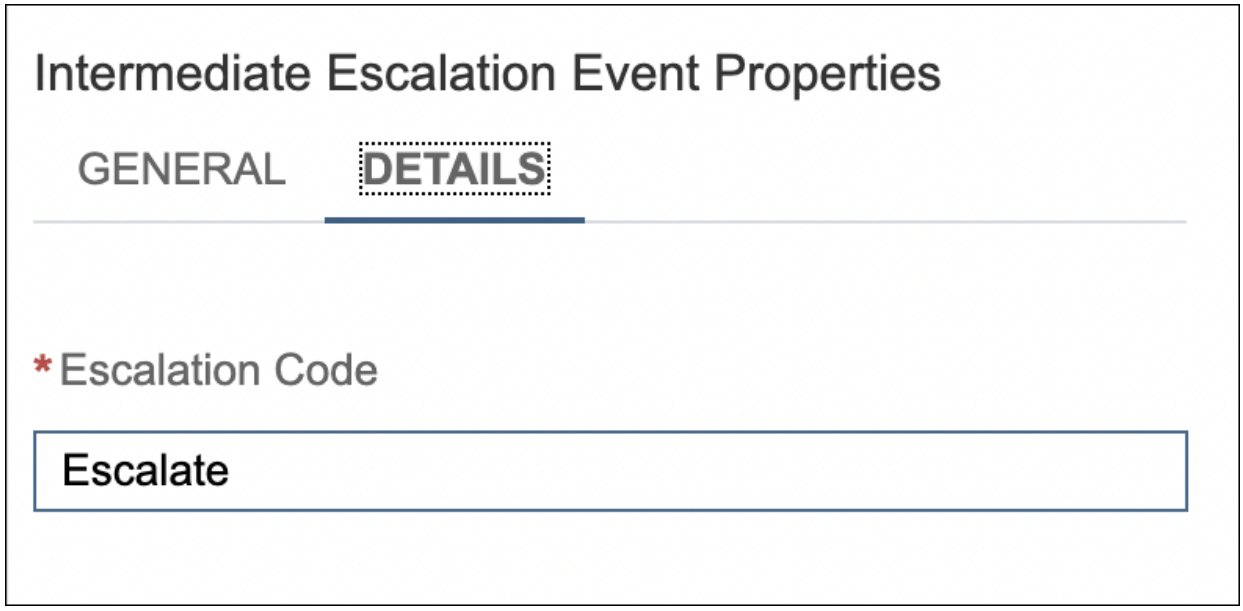


Figure 18.33 Configuration of an Intermediate Escalation Event

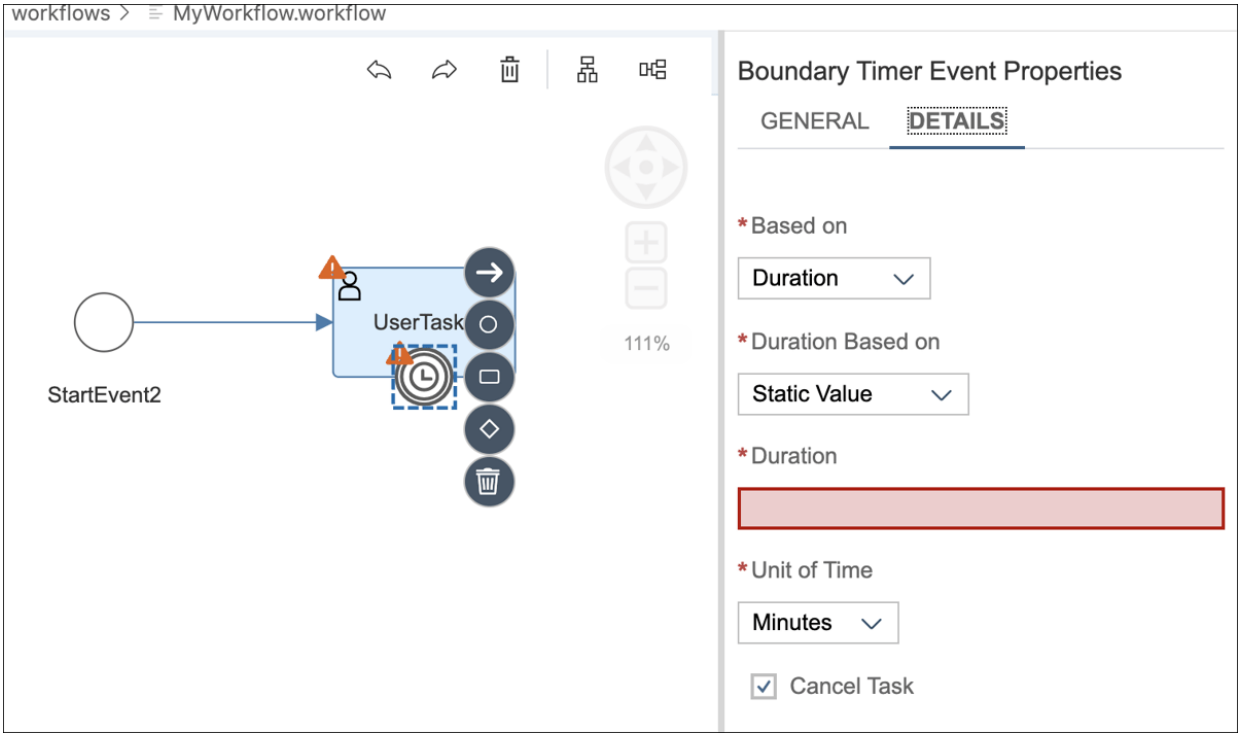


Figure 18.34 Add a Boundary Timer Event

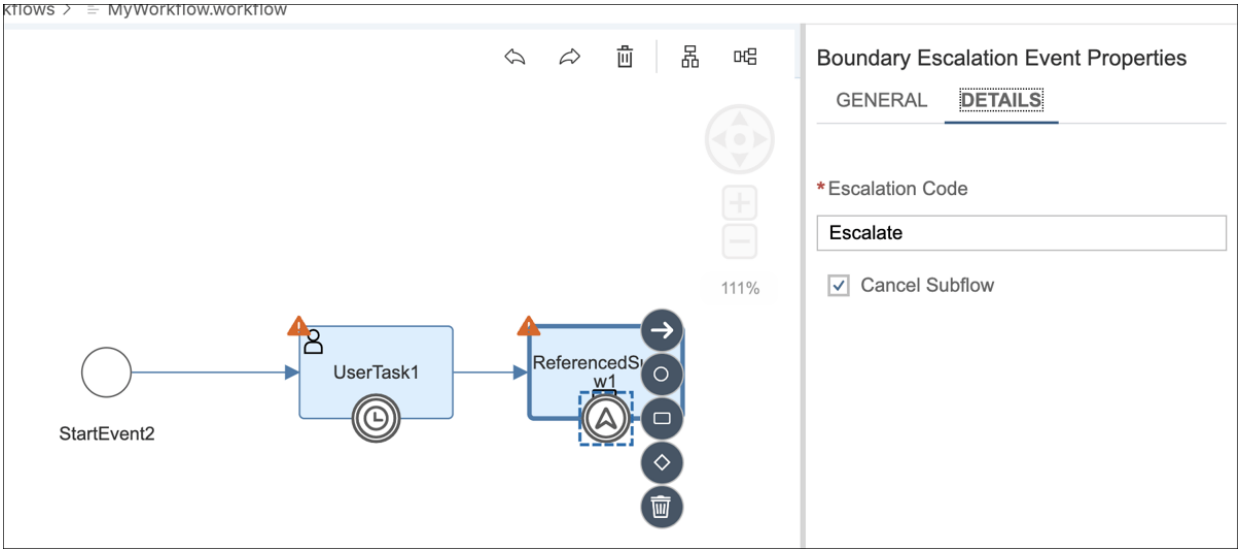


Figure 18.35 Configure a Boundary Escalation Event for a Referenced Subflow

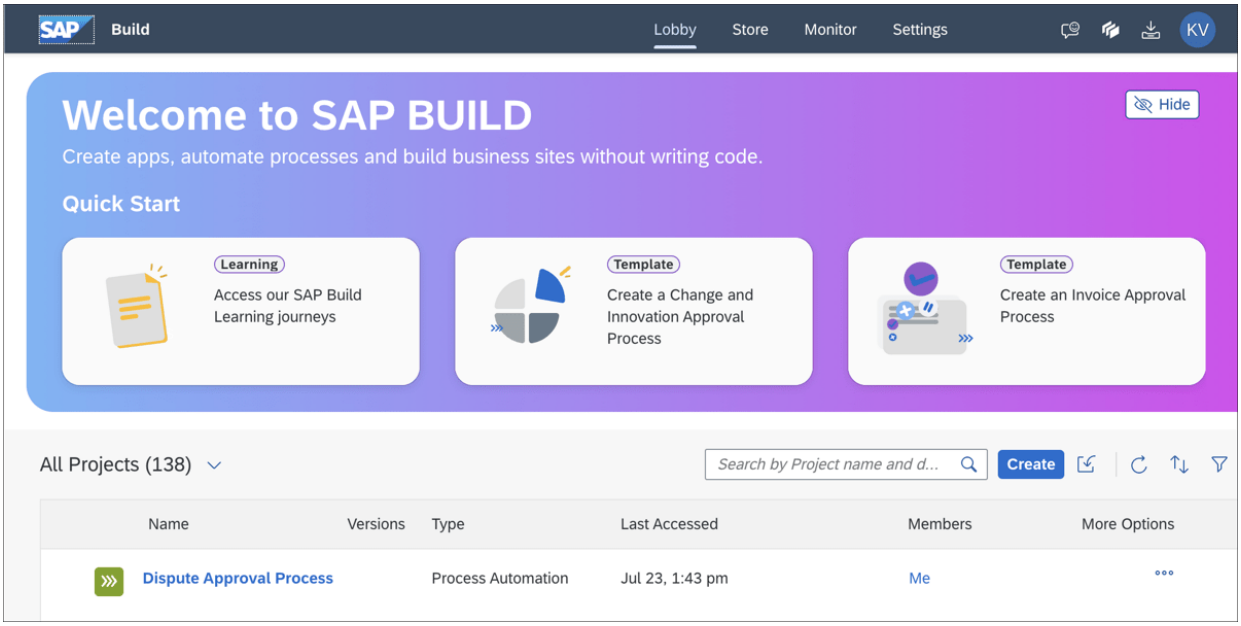


Figure 18.36 Lobby of the SAP Build Process Automation Cockpit

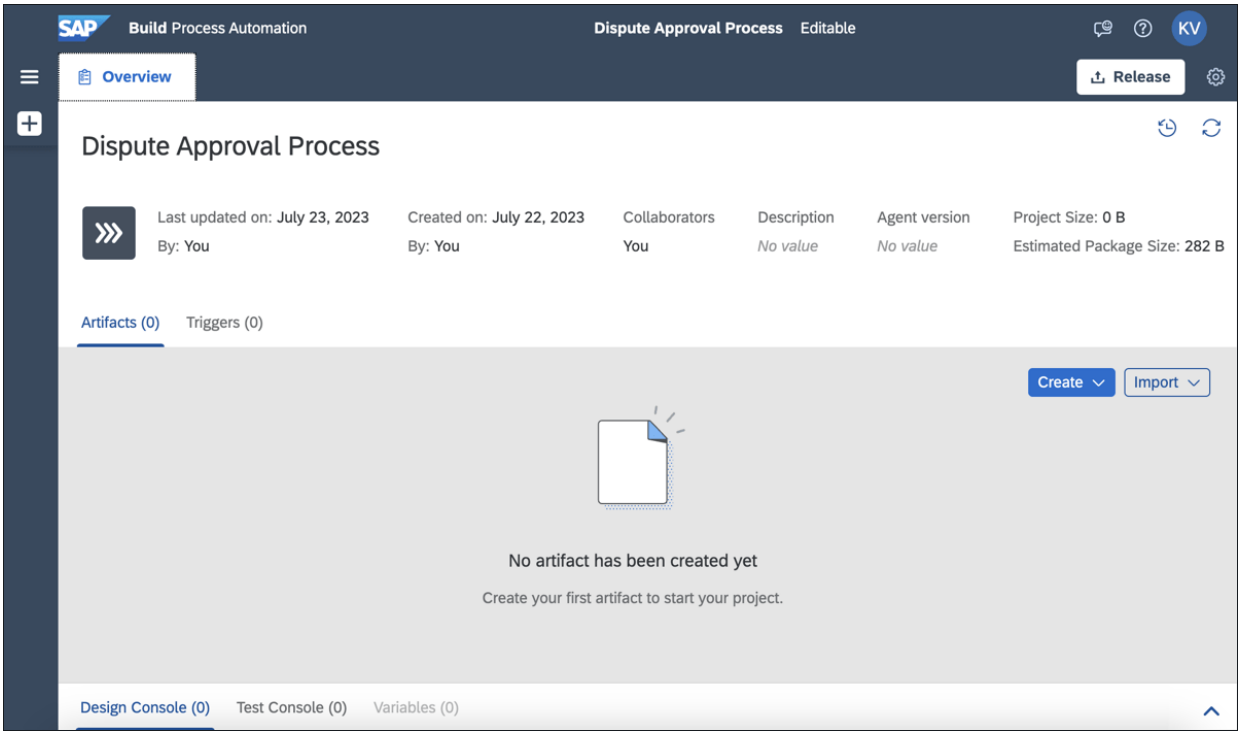


Figure 18.37 Landing Page of the Build Editor

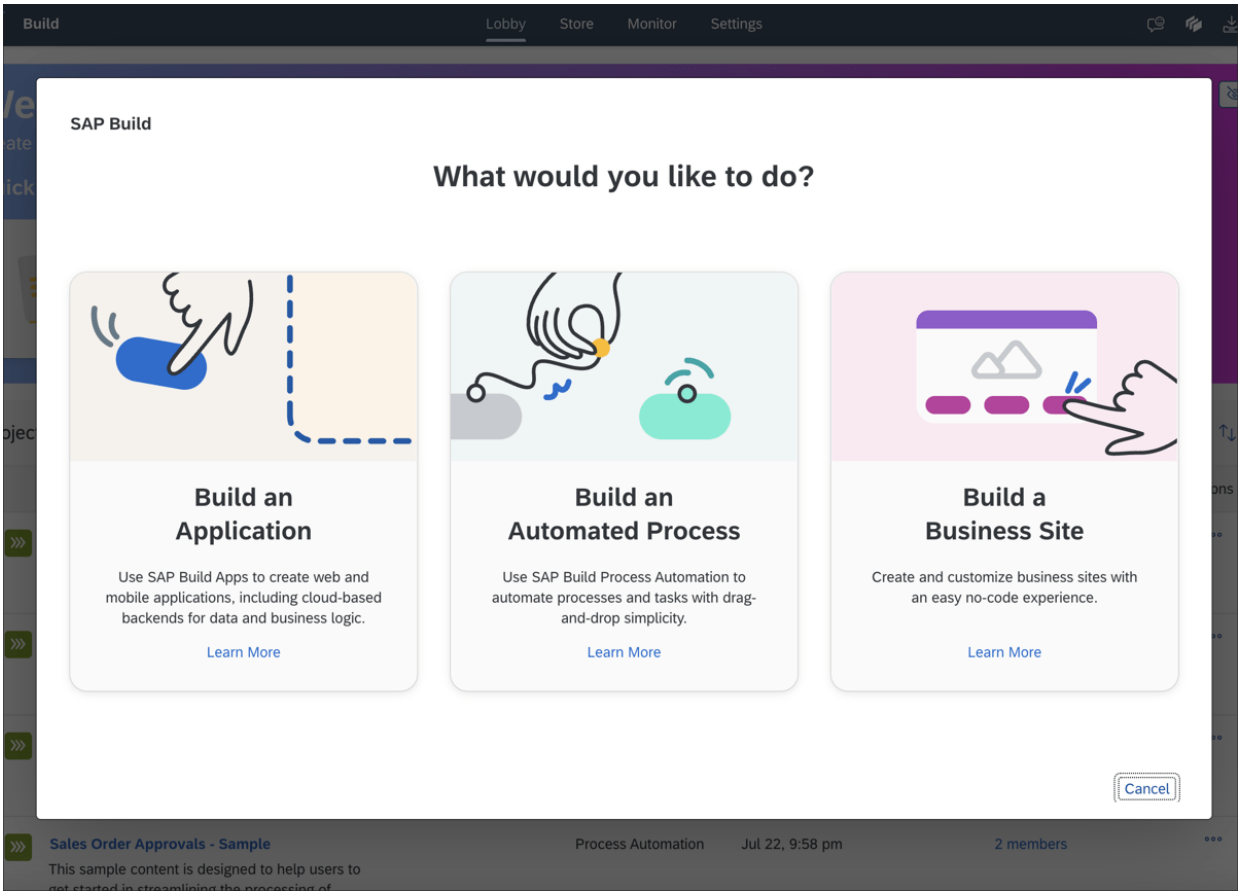


Figure 18.38 Selecting Build an Automation Process

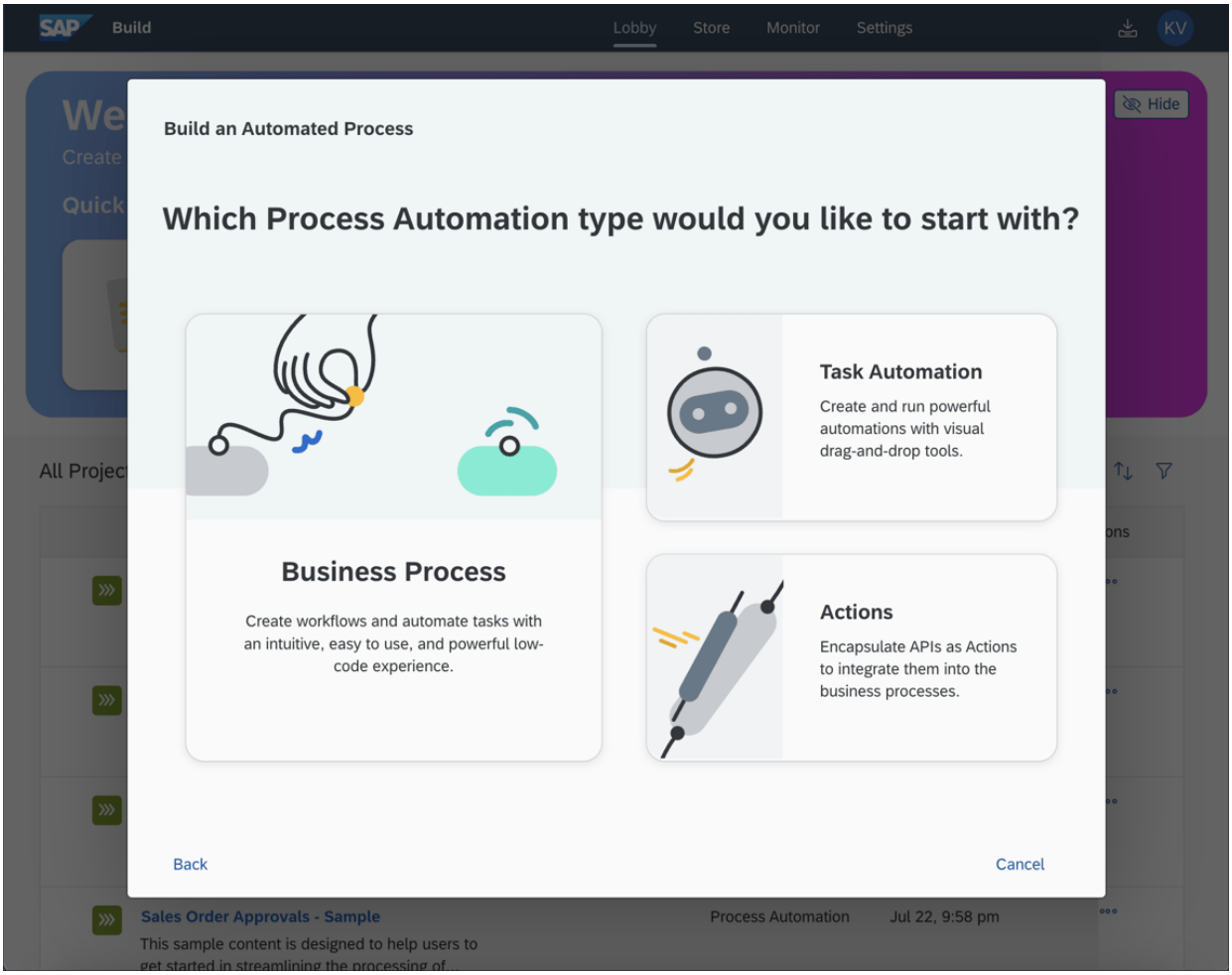


Figure 18.39 Selecting the Process Automation Type

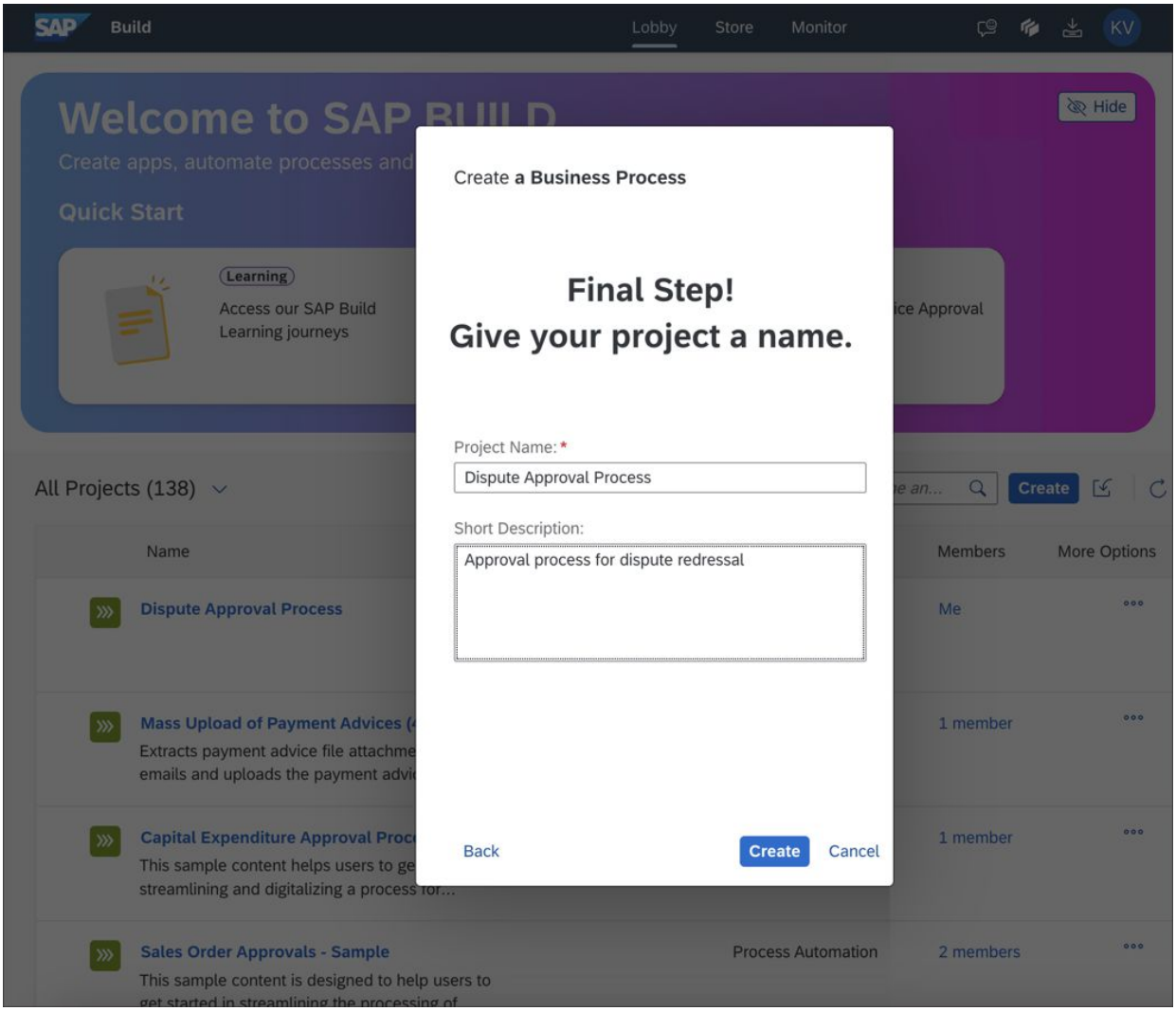


Figure 18.40 Naming the Project

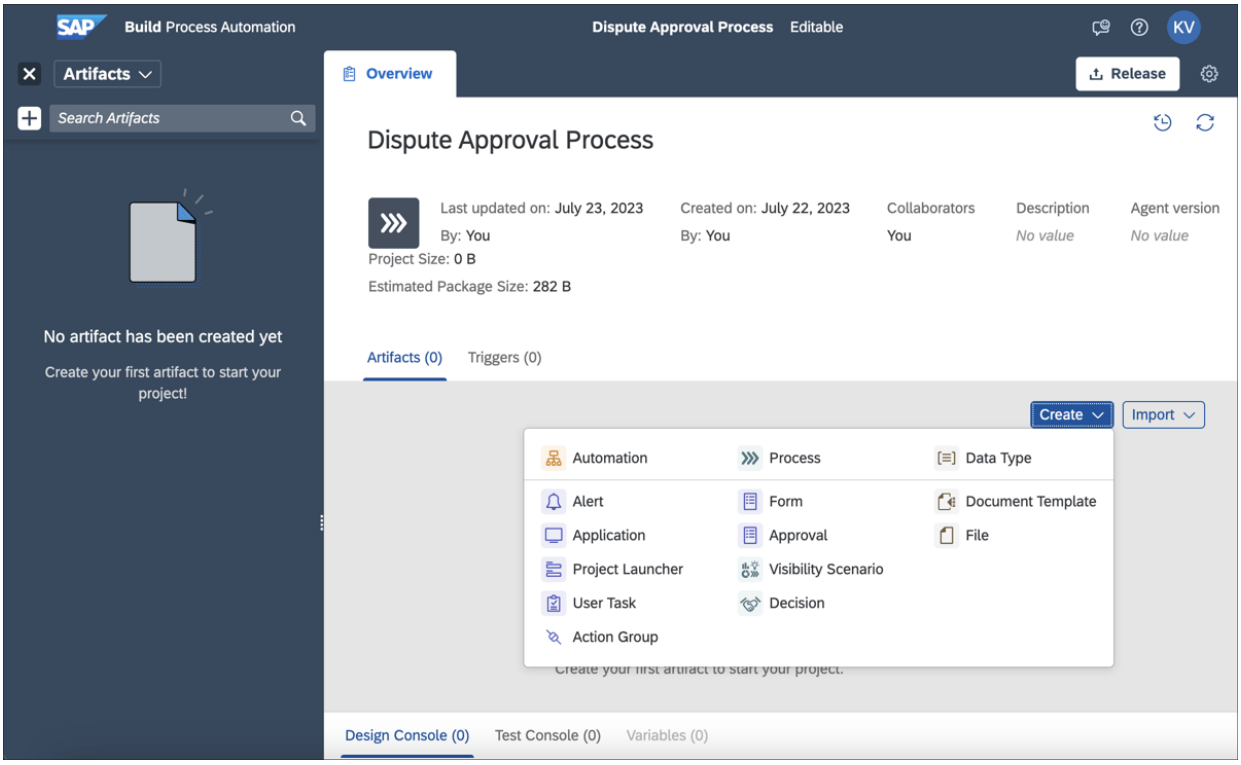


Figure 18.41 Create Your Process Artifacts

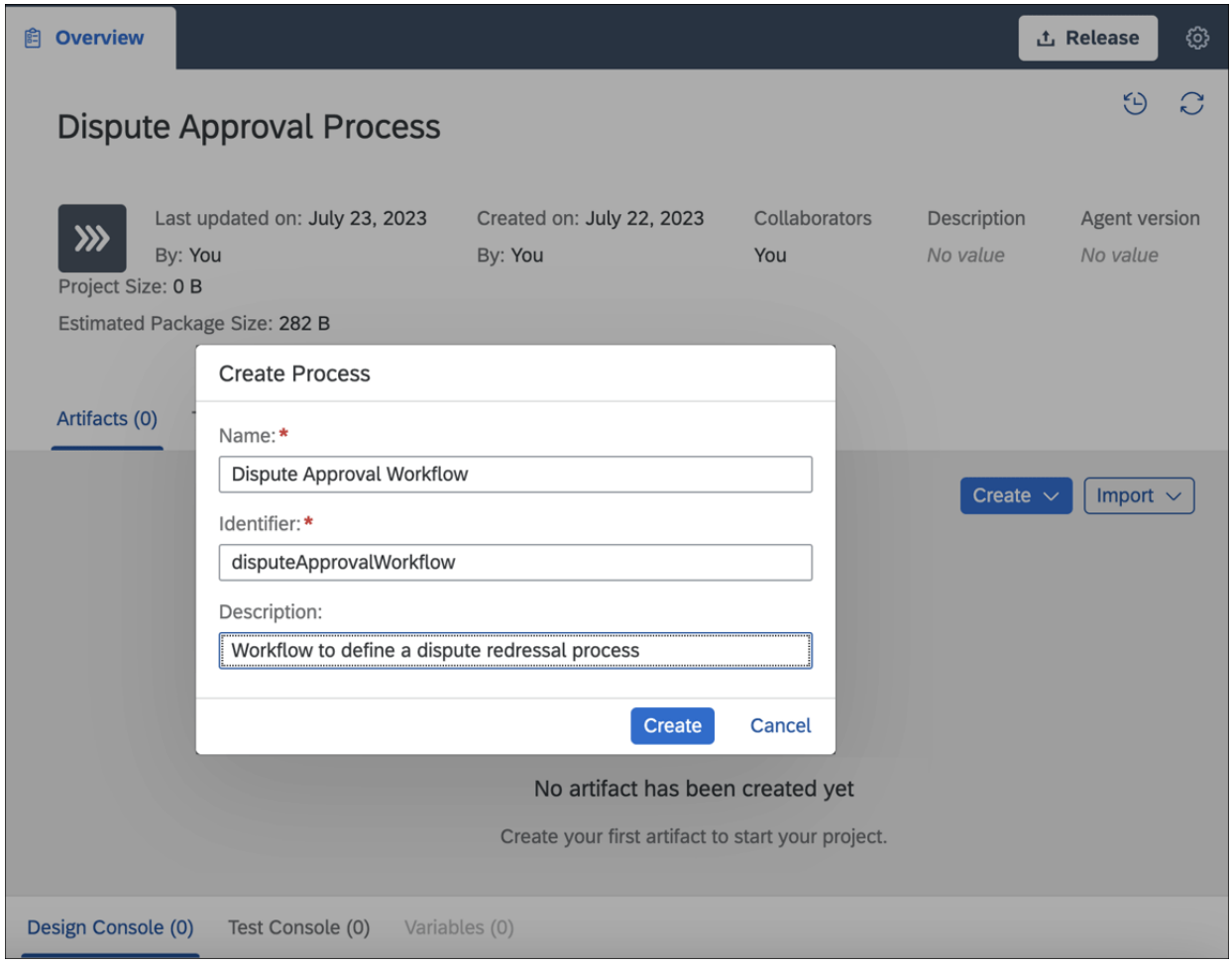


Figure 18.42 Creating a Process with a Meaningful Name

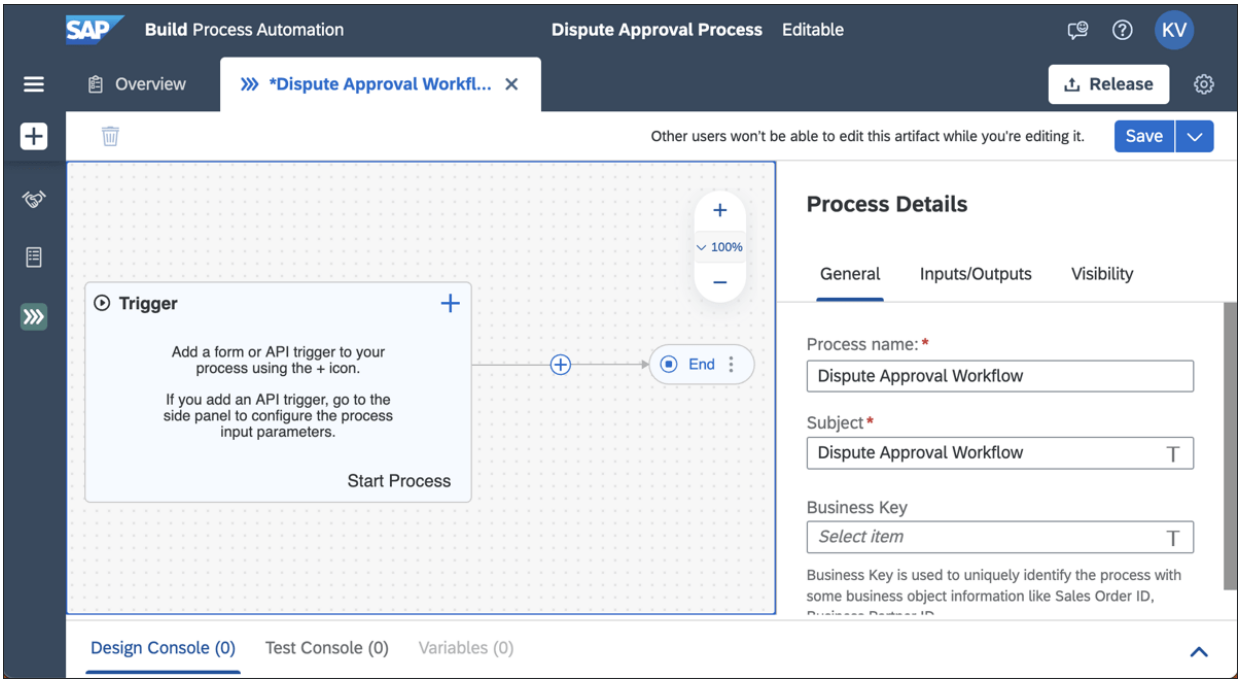


Figure 18.43 Process Builder with a Predefined Design Template

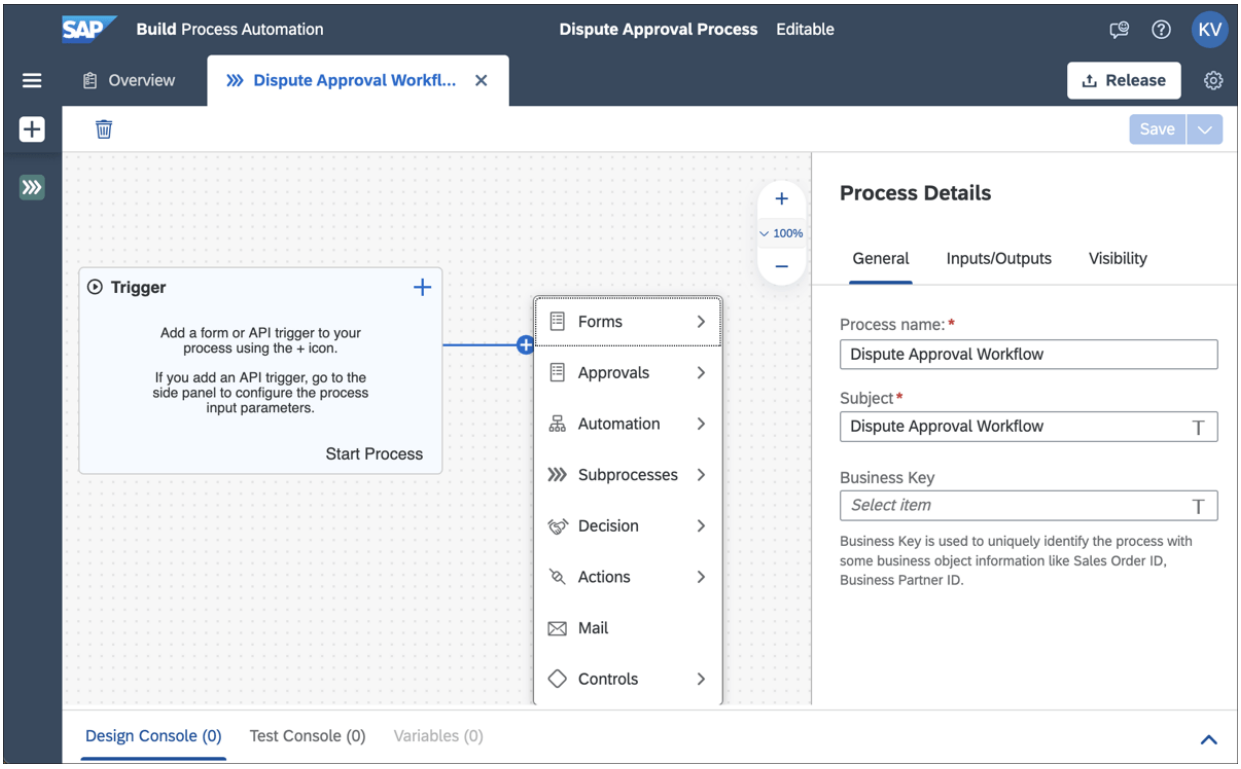


Figure 18.44 Artifacts Available in the Process Builder

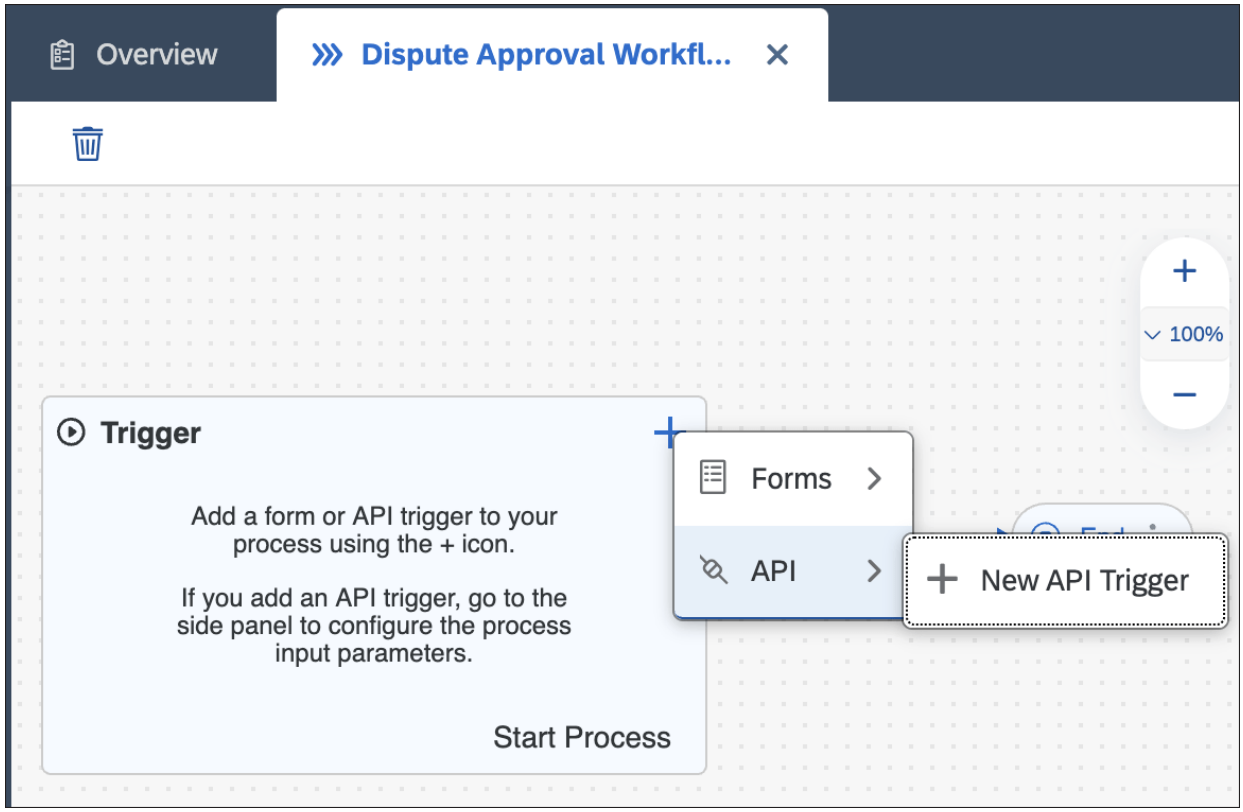


Figure 18.45 How to Use a Trigger to Start the Process

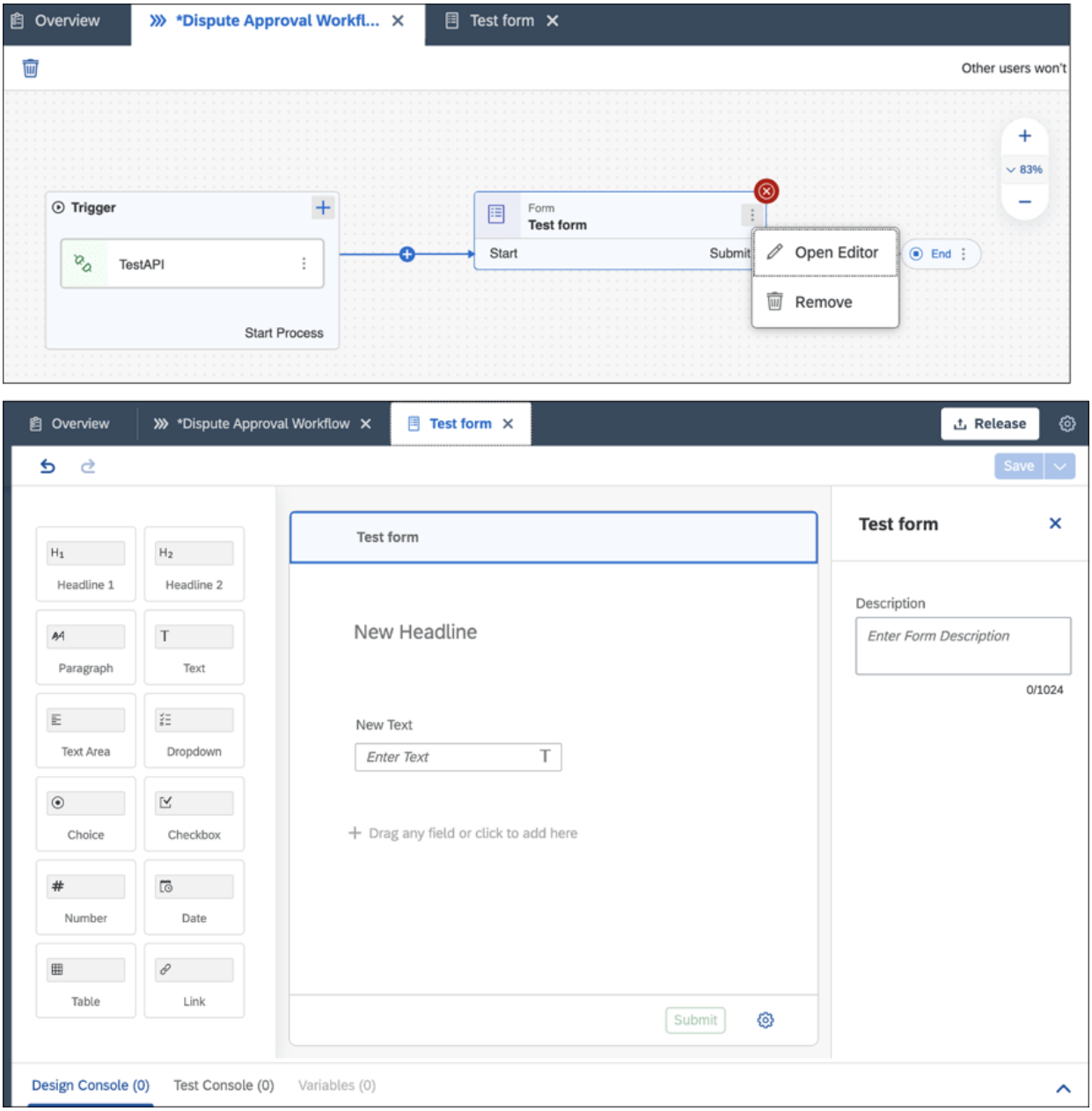


Figure 18.46 Creating a Simple Form

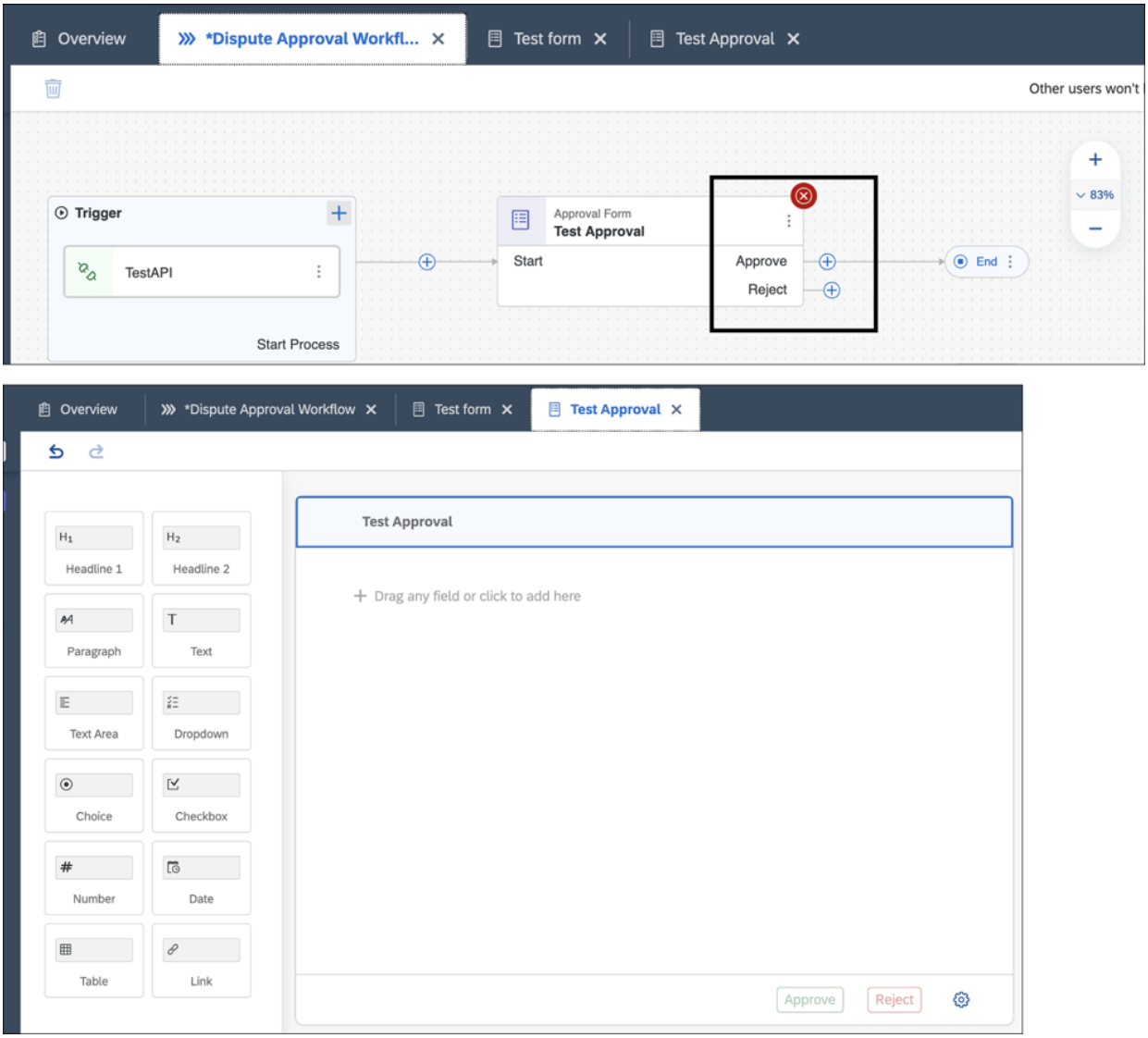


Figure 18.47 Create an Approval Form

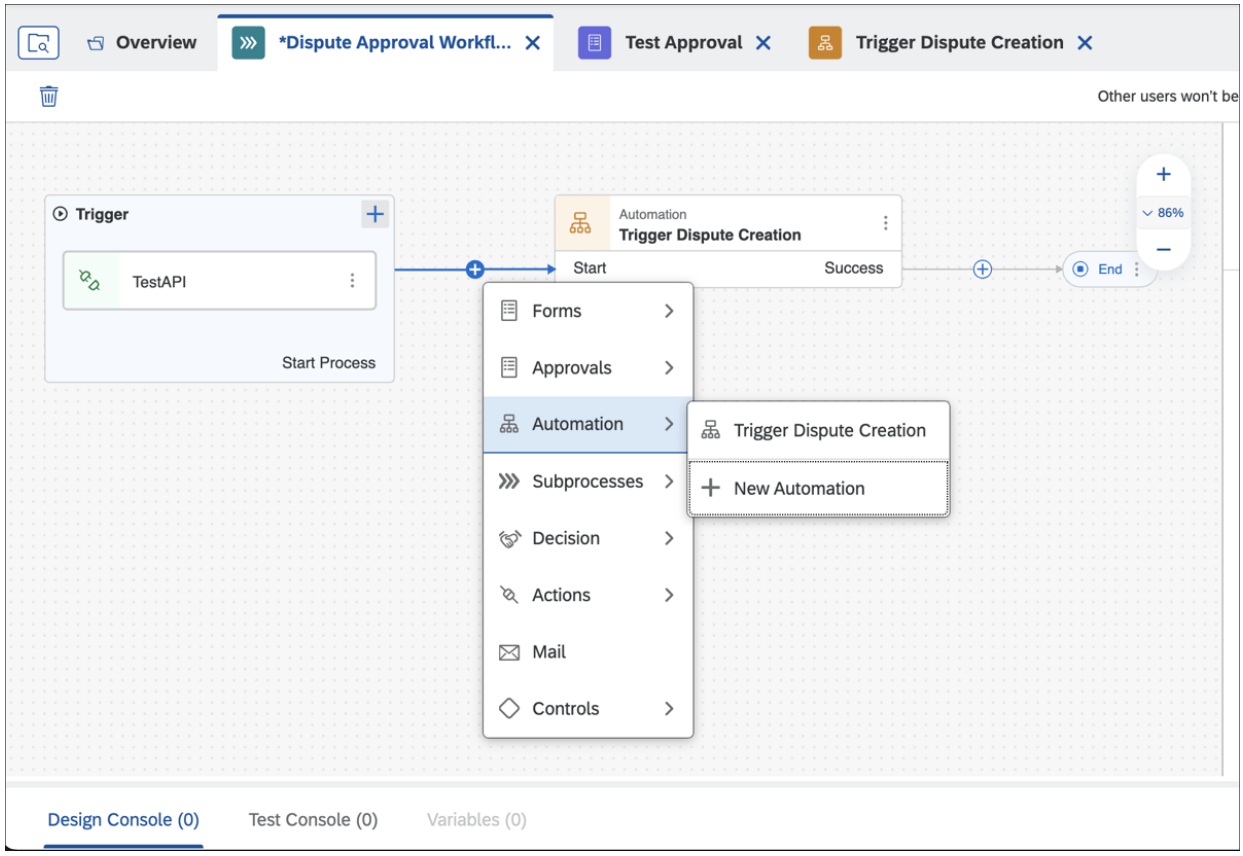


Figure 18.48 Adding an Automation Task in Your Process

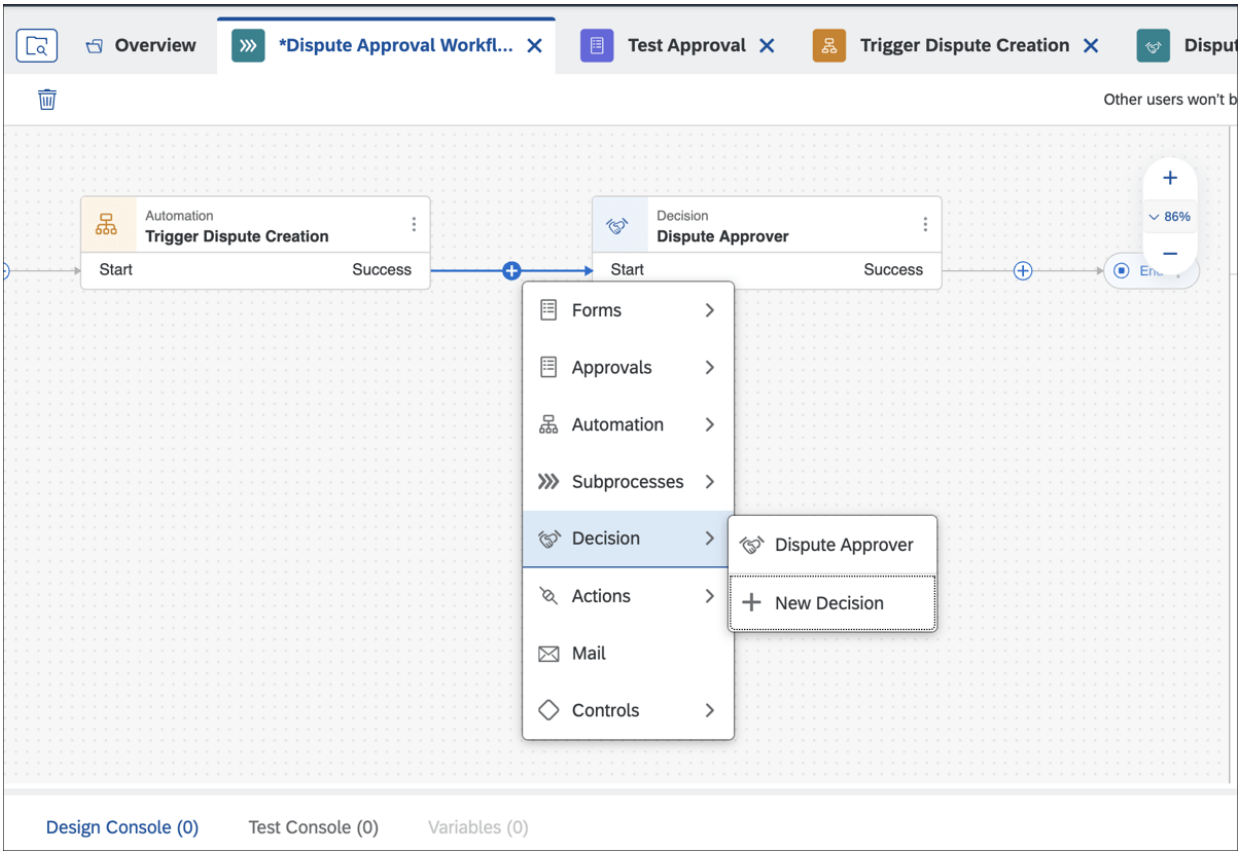


Figure 18.49 Creating a Decision Task

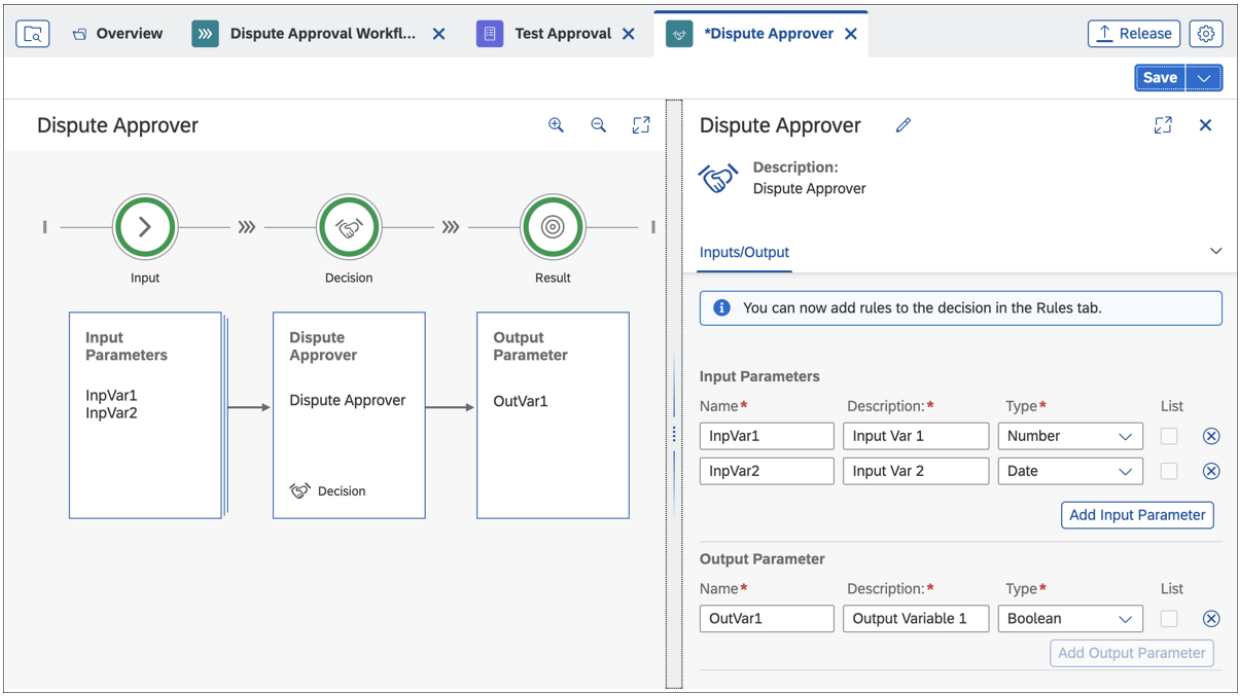


Figure 18.50 Creating and Editing the Decision Rules

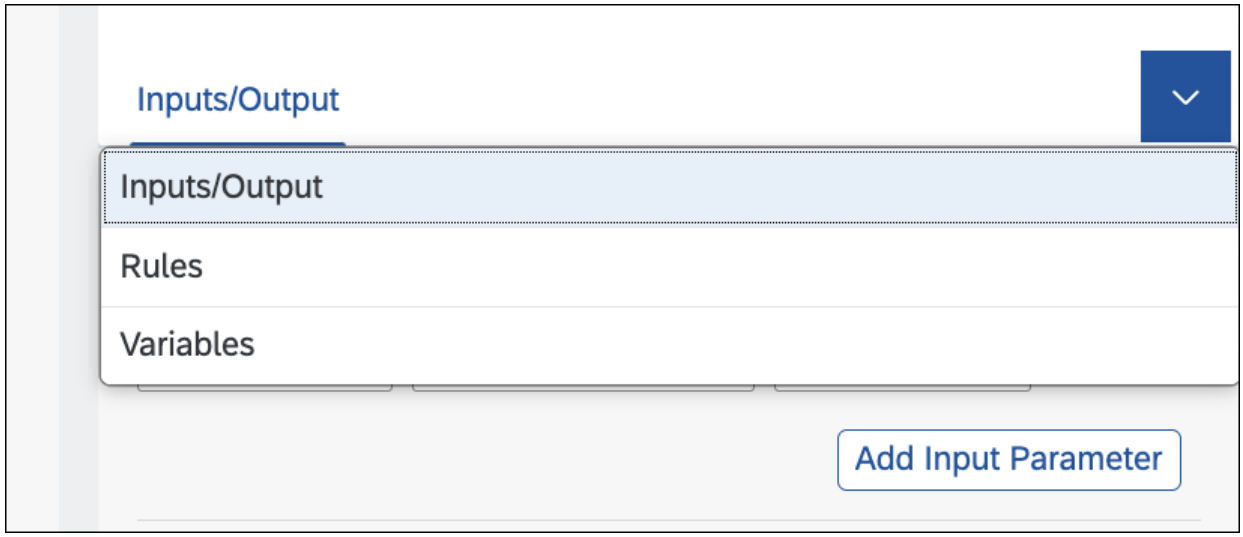
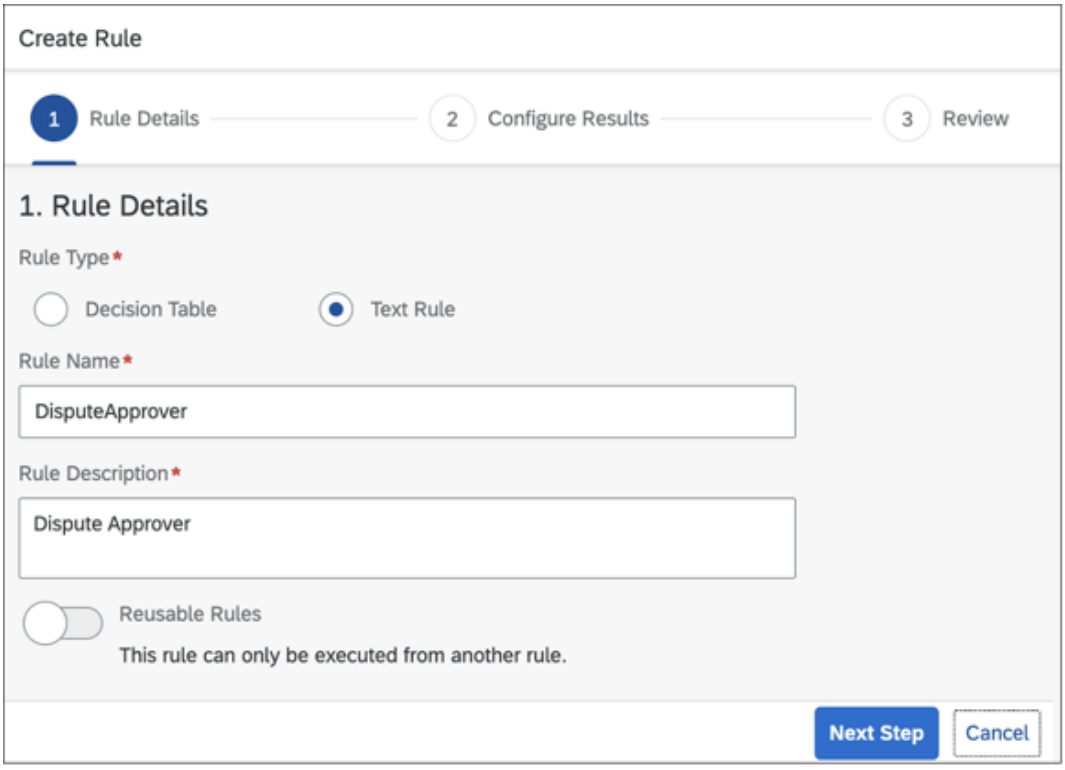


Figure 18.51 Navigating to the Rules Tab


Create Rule
1 Rule Details
2 Configure Results
3 Review
1. Rule Details
Rule Type*
Decision Table
Text Rule
Rule Name*
DisputeApprover
Rule Description*
Dispute Approver
Reusable Rules
This rule can only be executed from another rule.
Next Step
Cancel

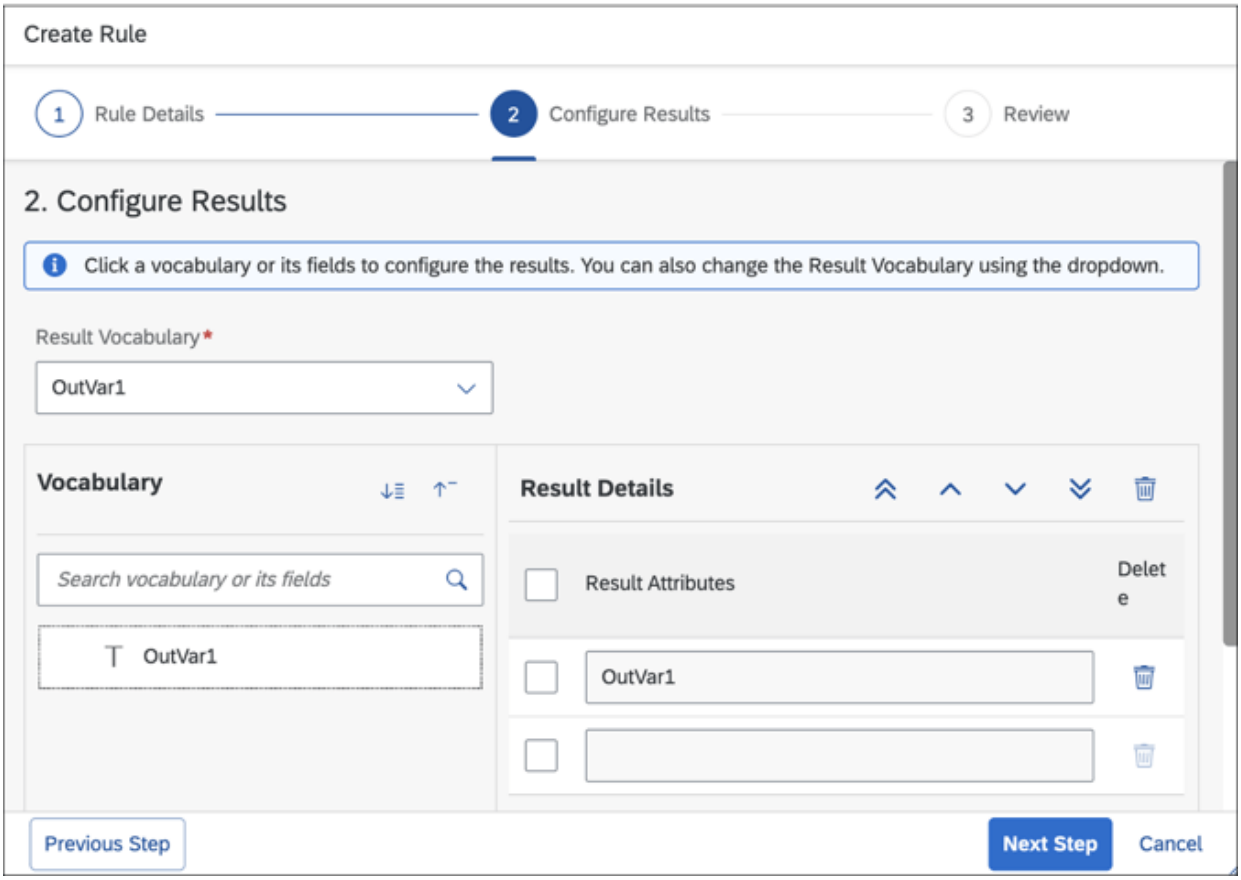

Create Rule
1 Rule Details
2 Configure Results
3 Review
2. Configure Results
Click a vocabulary or its fields to configure the results. You can also change the Result Vocabulary using the dropdown.
Result Vocabulary*
OutVar1
Vocabulary
Search vocabulary or its fields
OutVar1
Result Details
Result Attributes
Delet
e
OutVar1
Previous Step
Next Step
Cancel

Figure 18.52 Creating a Text Rule

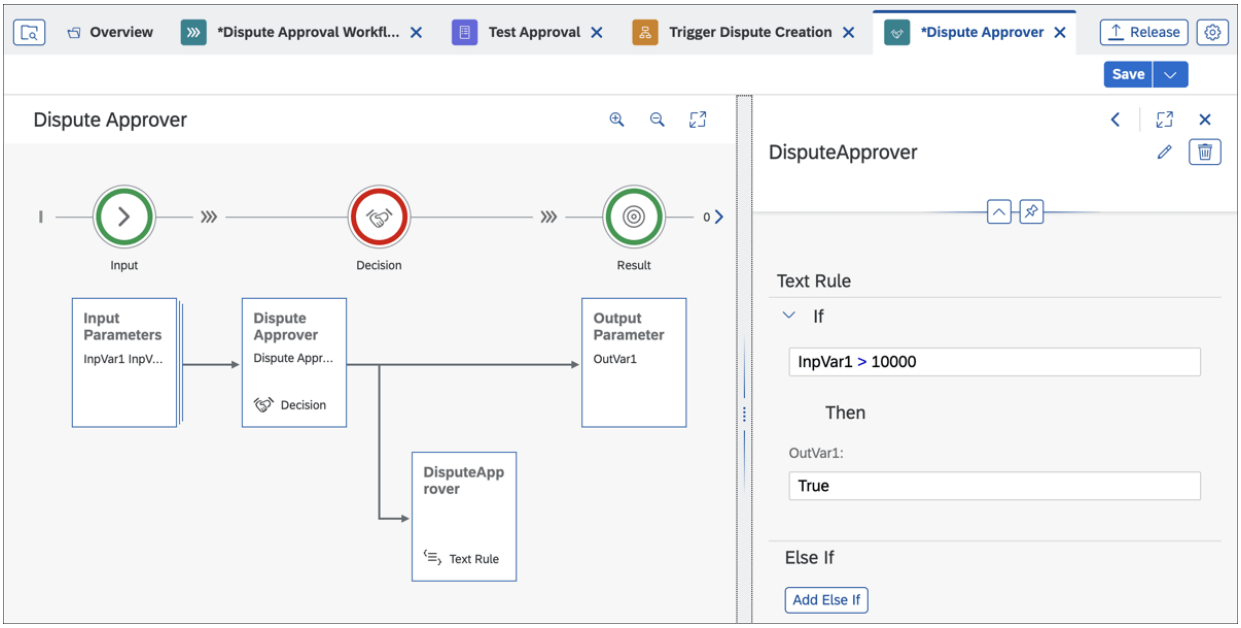


Figure 18.53 Creating a Rule Condition and Configuring the Output

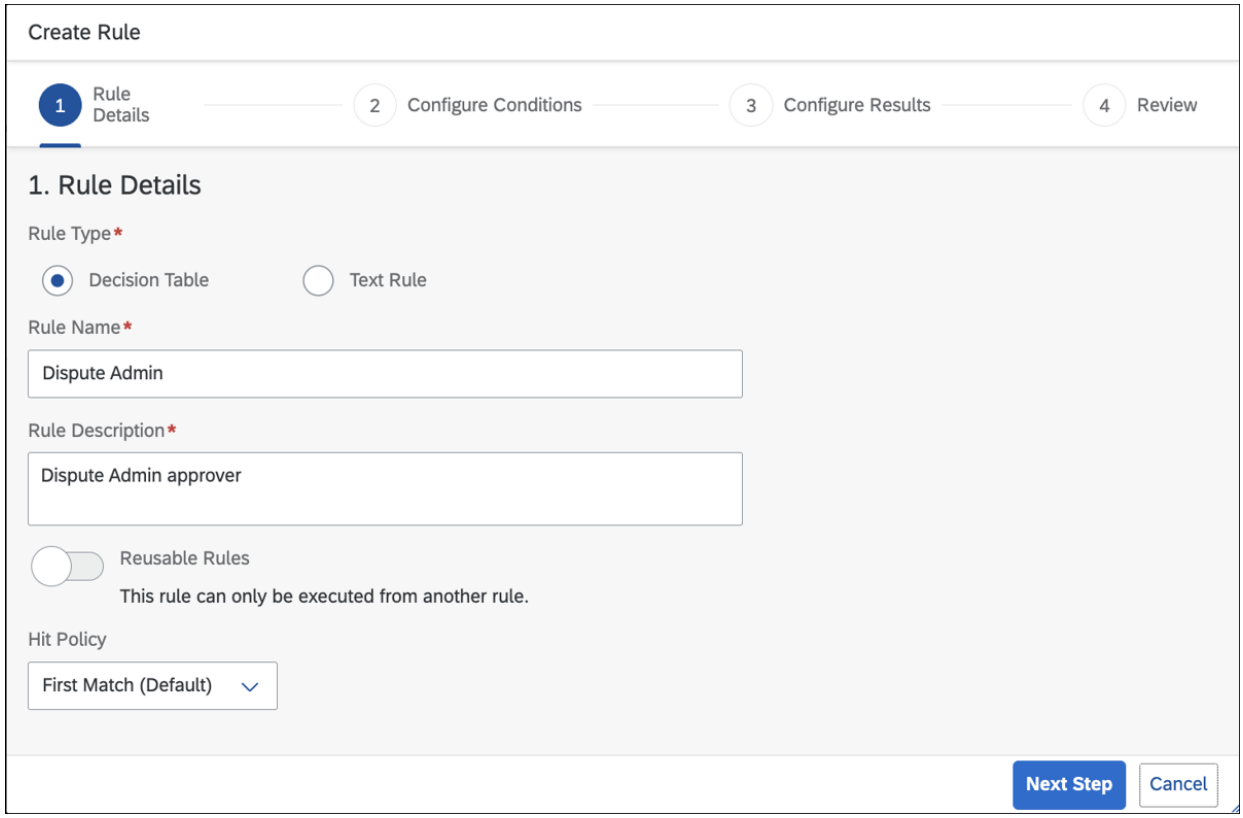


Figure 18.54 Creating a Name for the Decision Table Rule

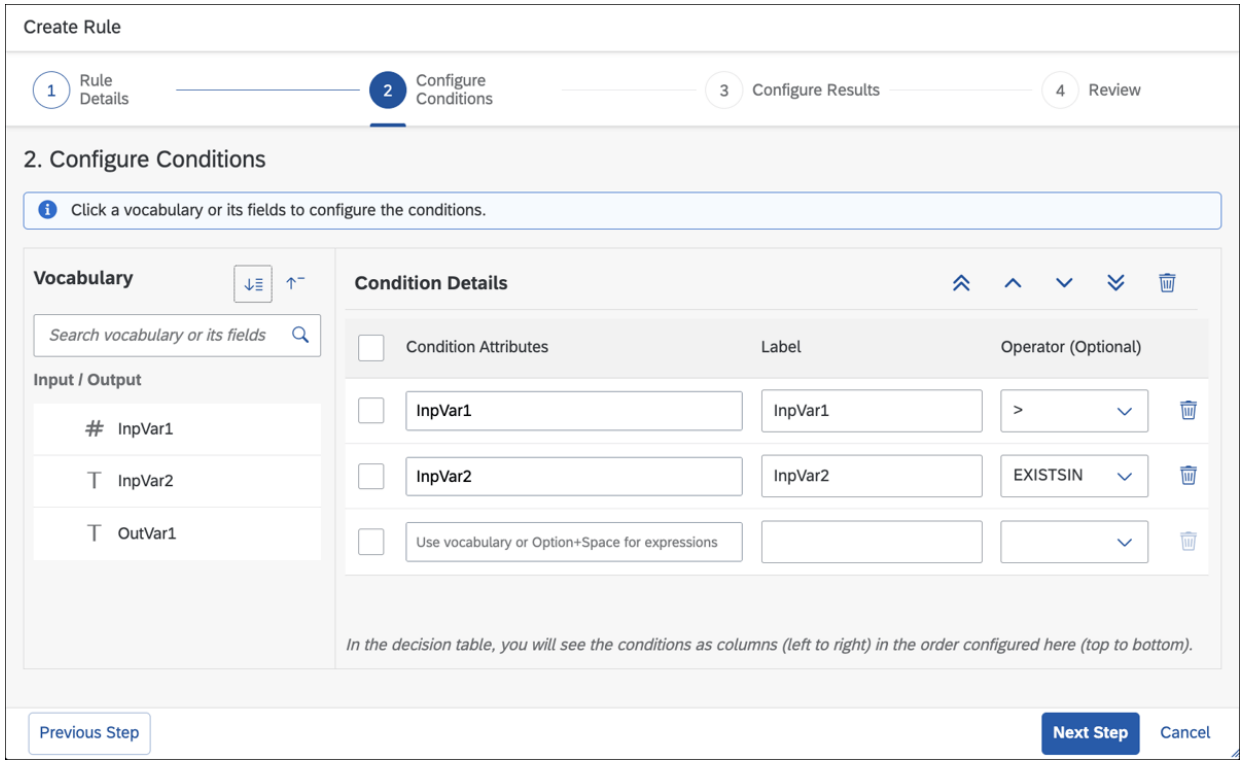


Figure 18.55 Configuring the Input Condition Attributes

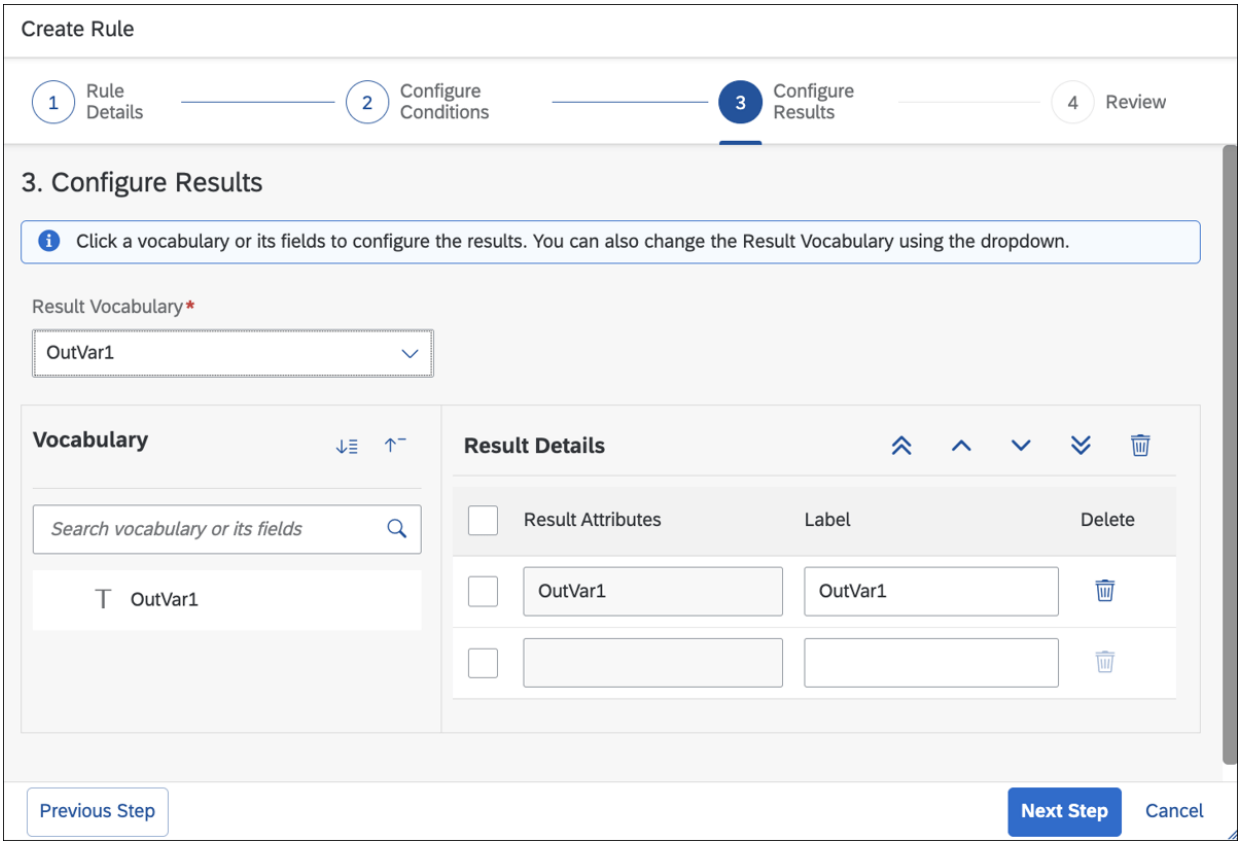


Figure 18.56 Configuring the Output Condition Attributes

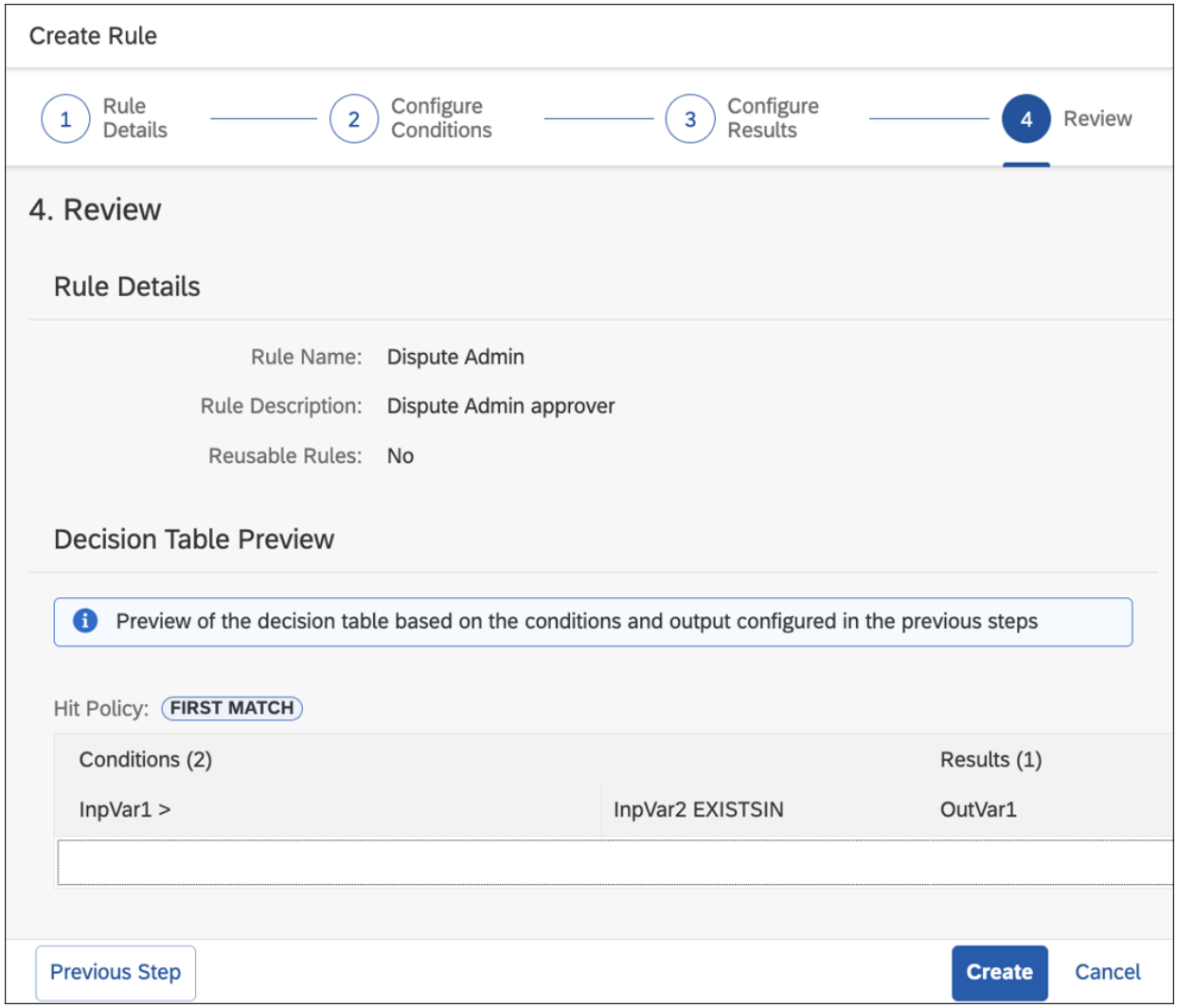


Figure 18.57 Reviewing and Creating the Rule

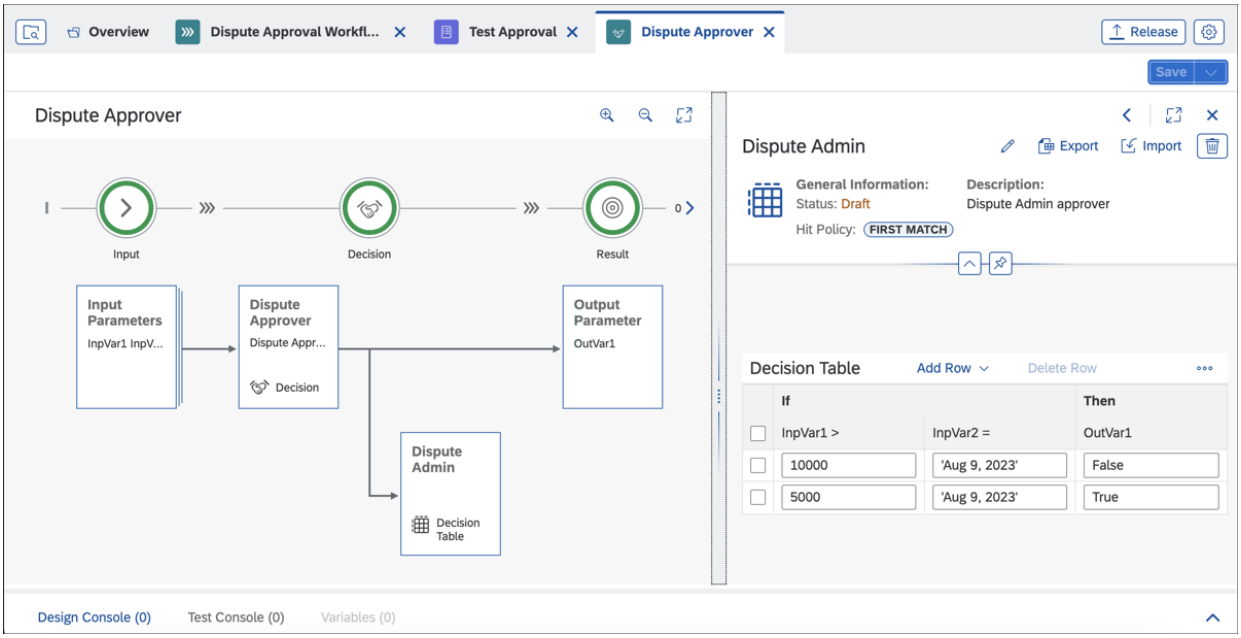


Figure 18.58 Configuring the Decision Table

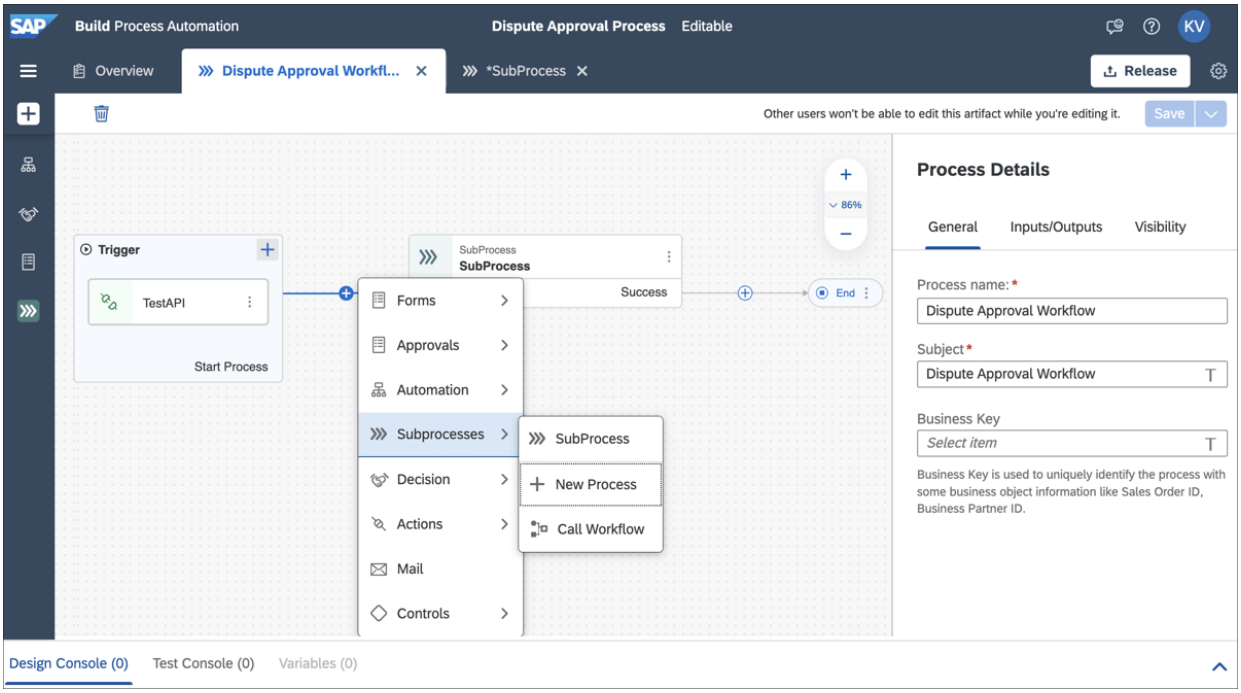


Figure 18.59 Adding a Subprocess

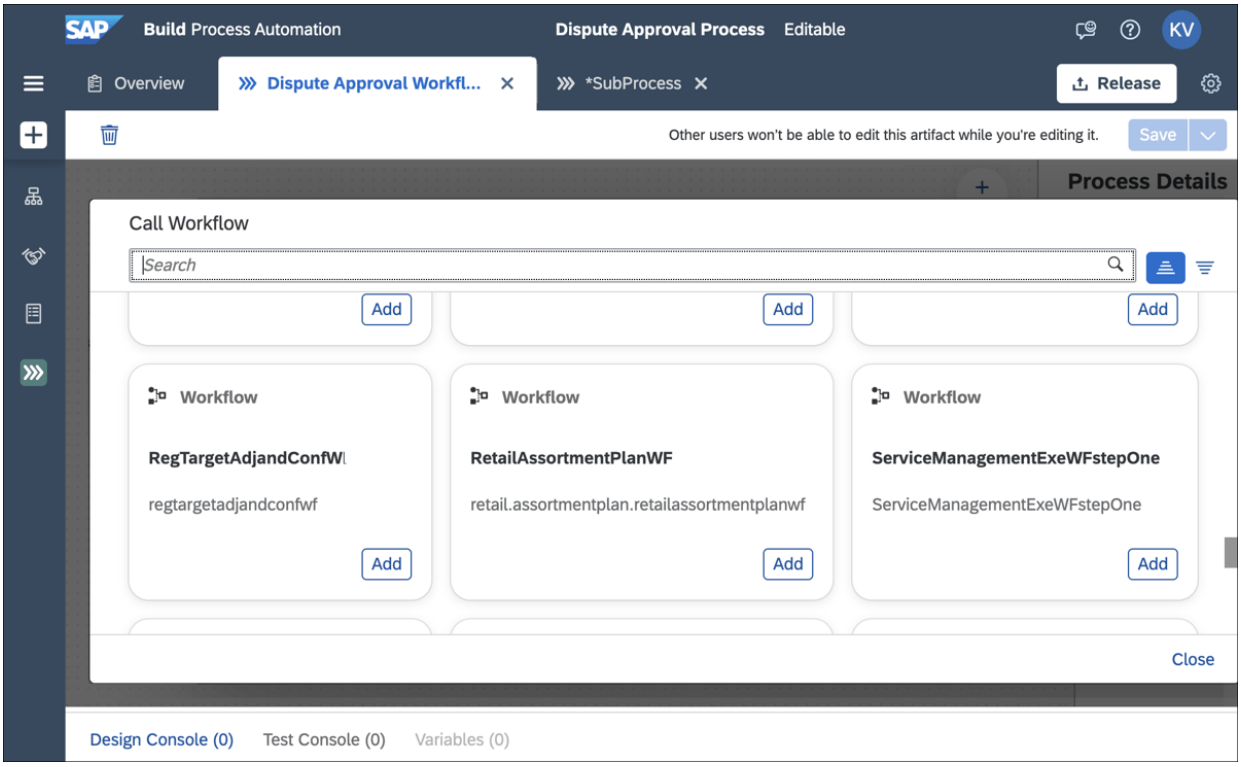


Figure 18.60 Adding an SAP Business Application Studio–Based Workflow as a Subprocess

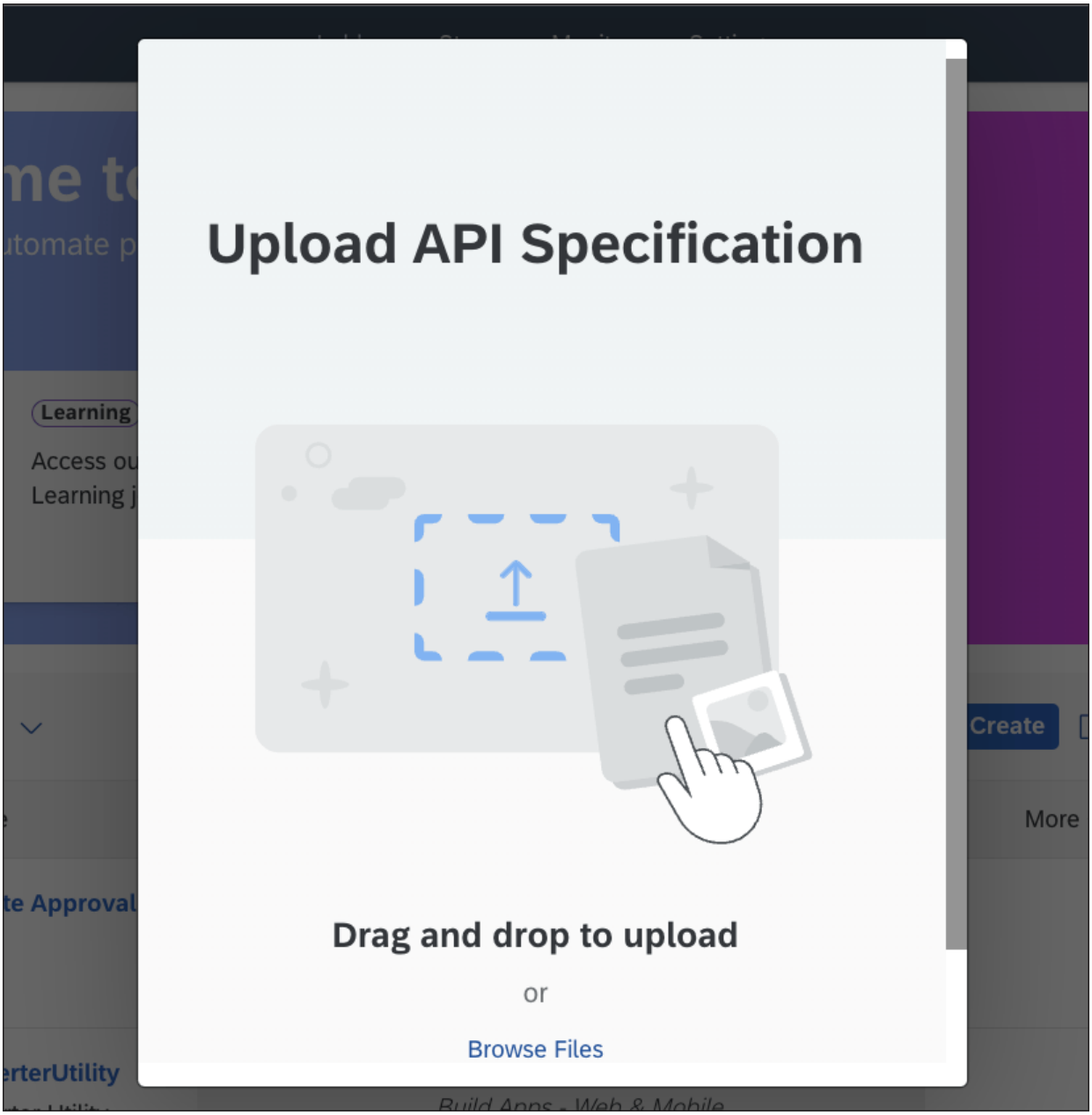


Figure 18.61 Uploading an API Specification

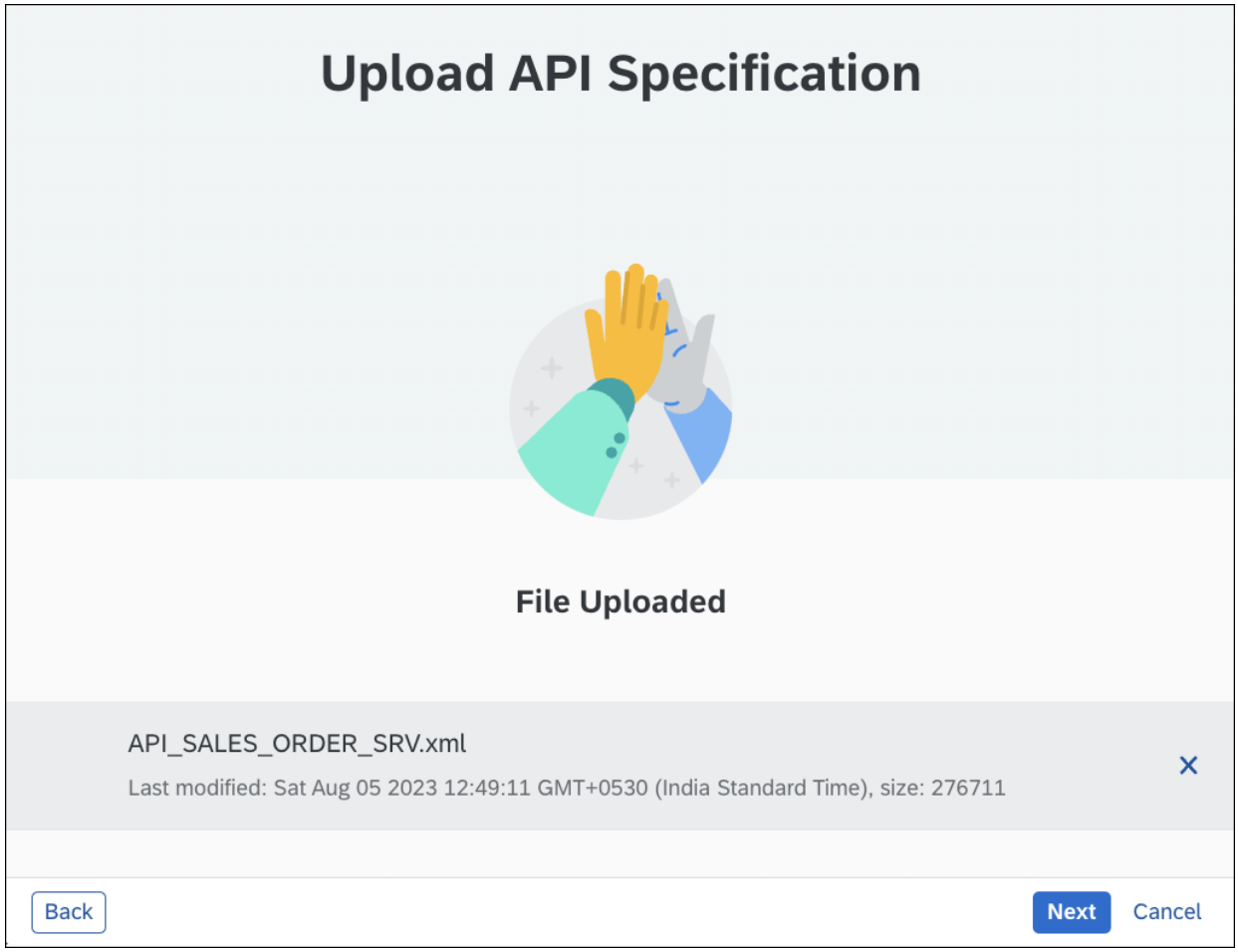


Figure 18.62 Uploading the API Specification

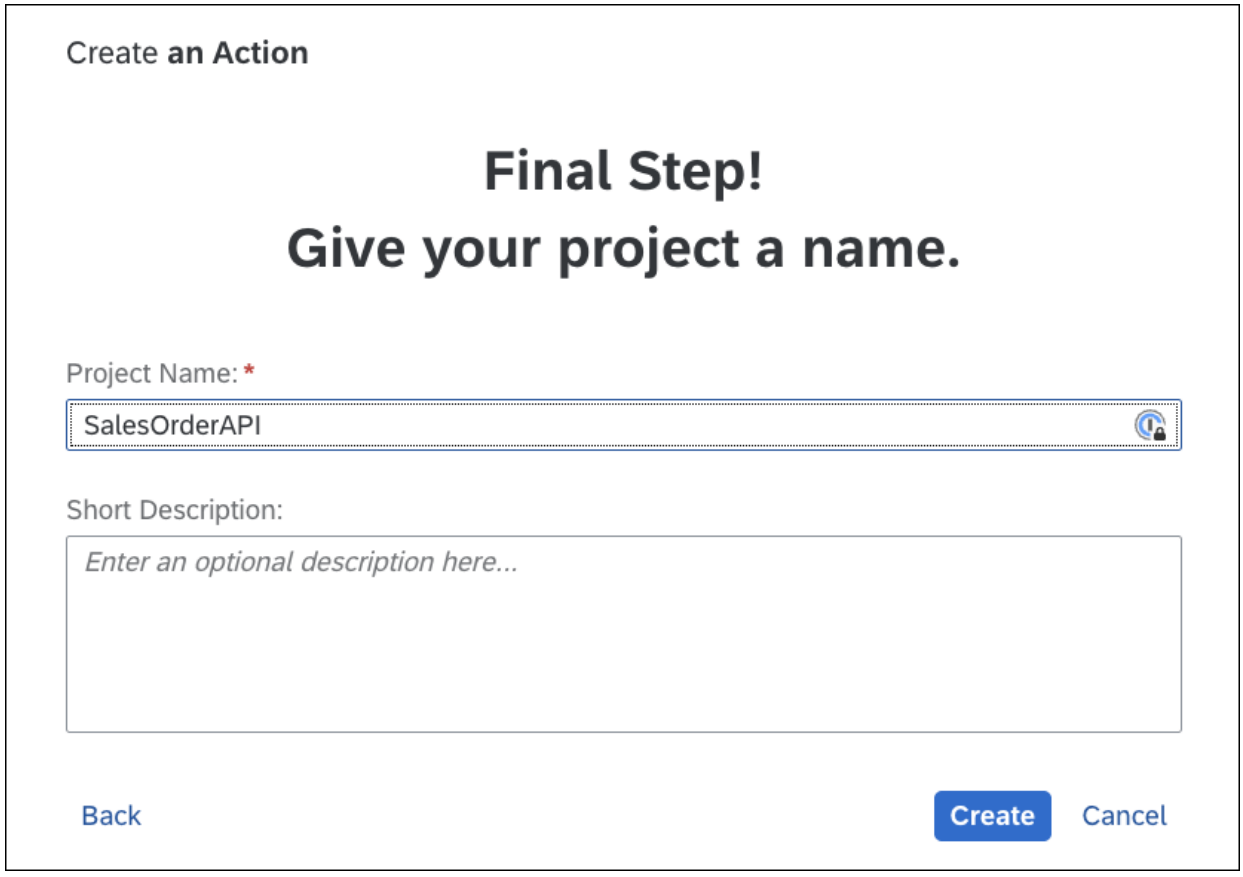


Figure 18.63 Naming the Action Project

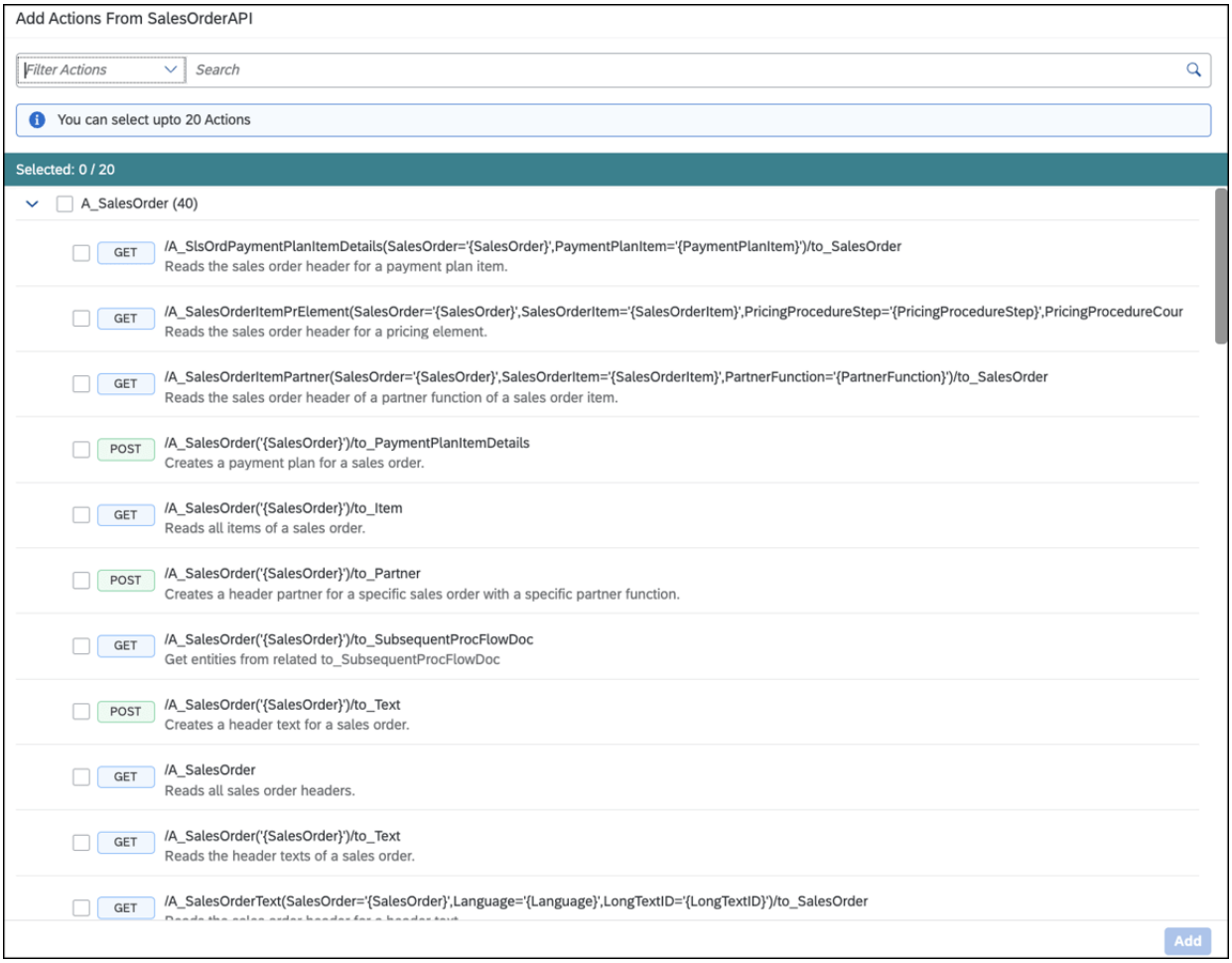


Figure 18.64 Selecting the Actions

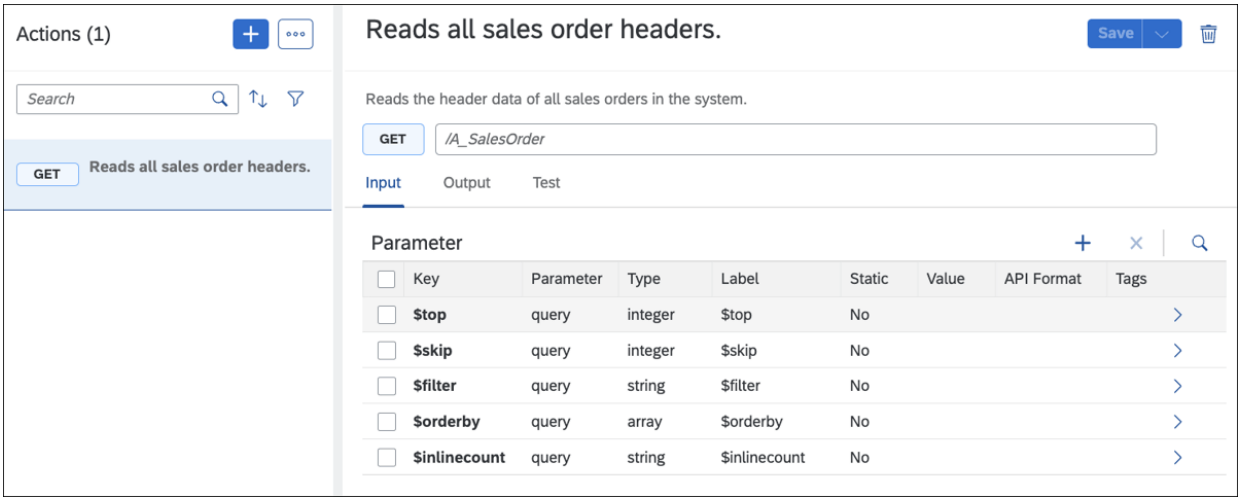


Figure 18.65 Configuring the Selected Actions

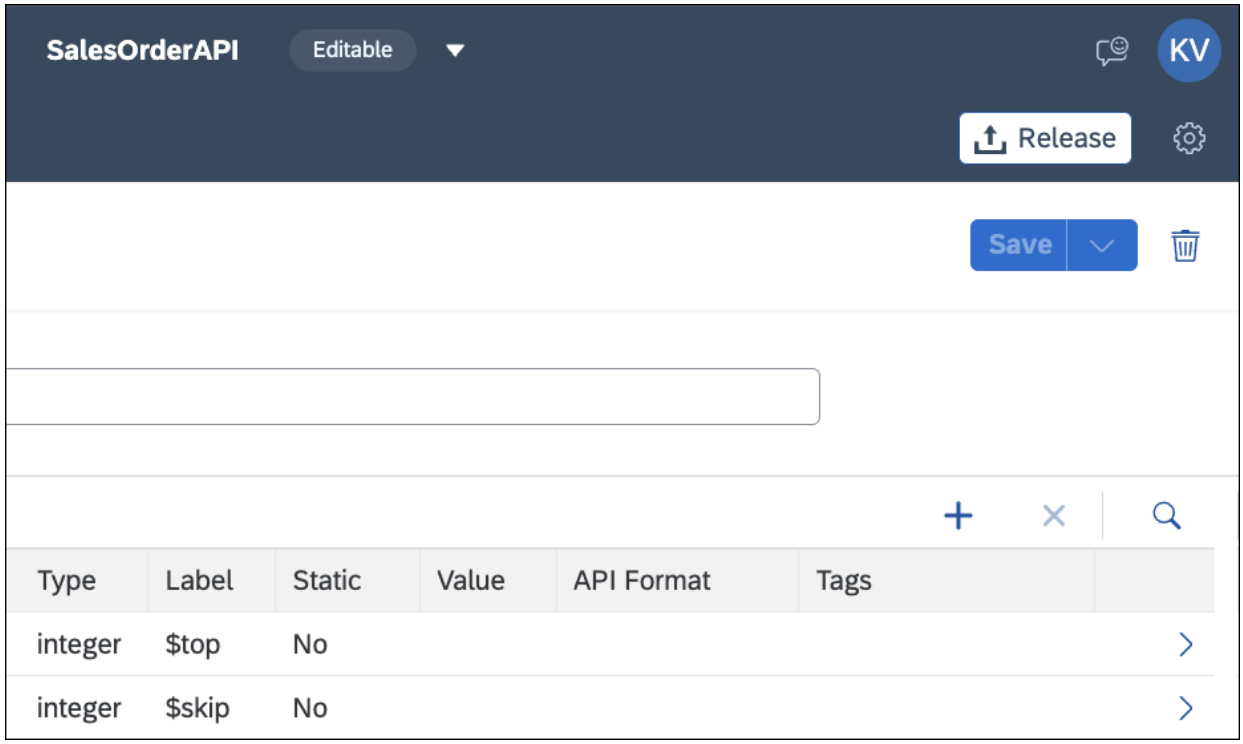


Figure 18.66 Releasing the Project

Figure 18.67 Publishing Your Action

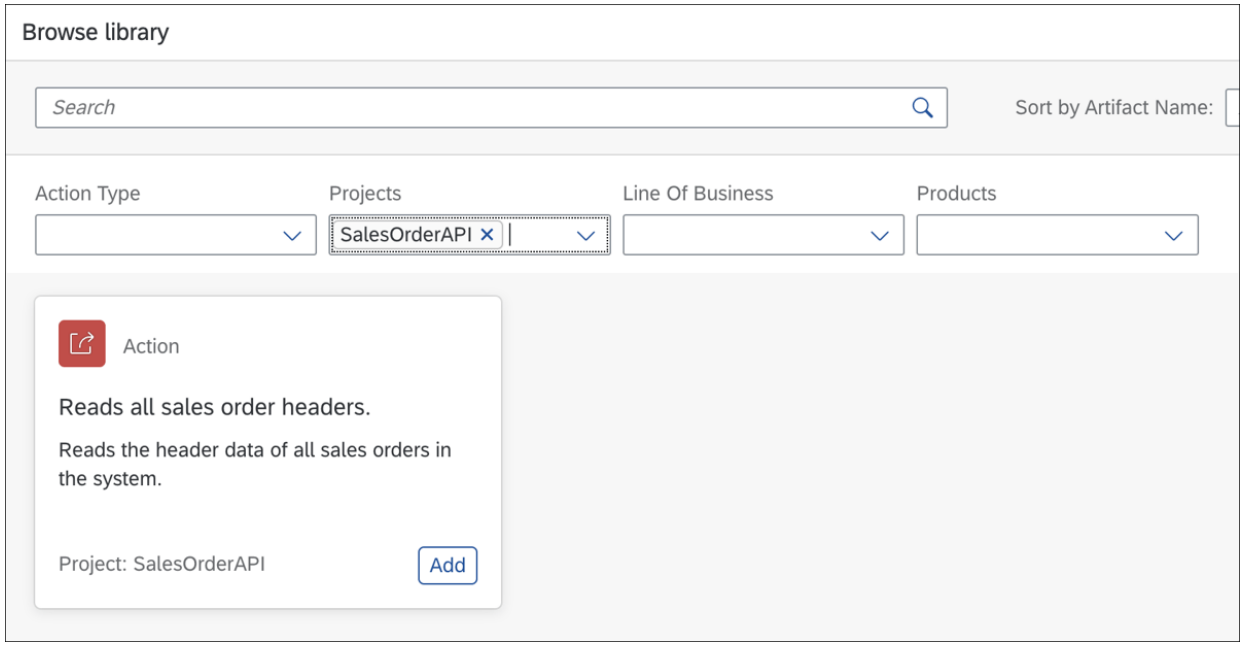


Figure 18.68 Selecting and Adding the API

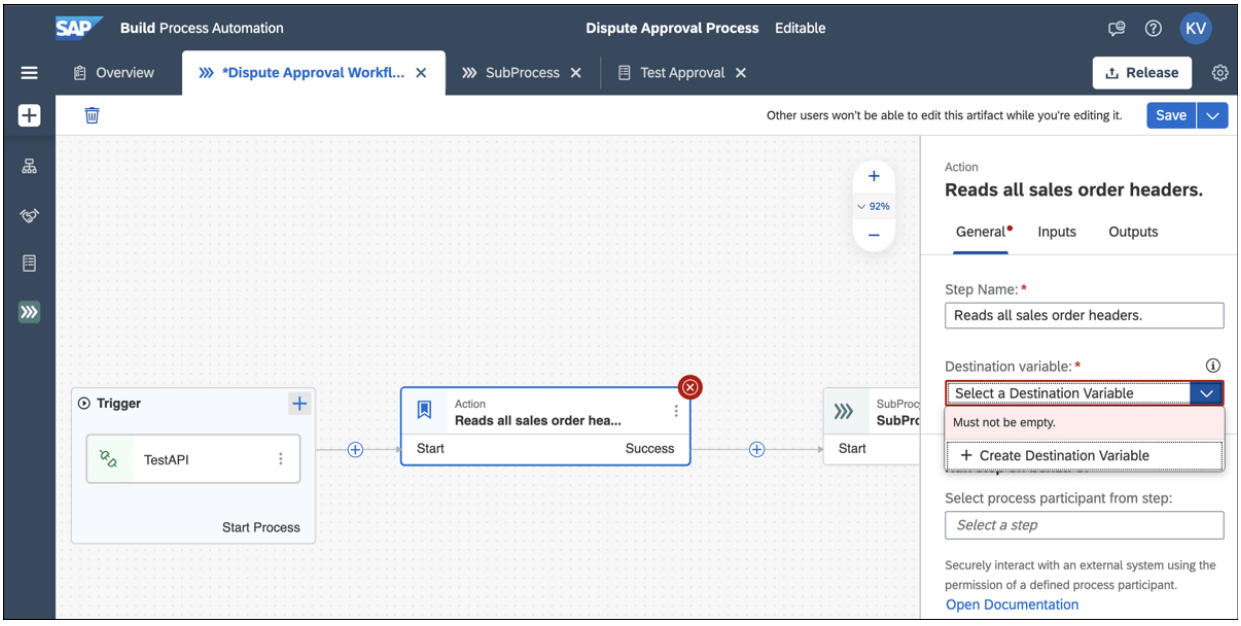


Figure 18.69 Adding the Action to Your Process

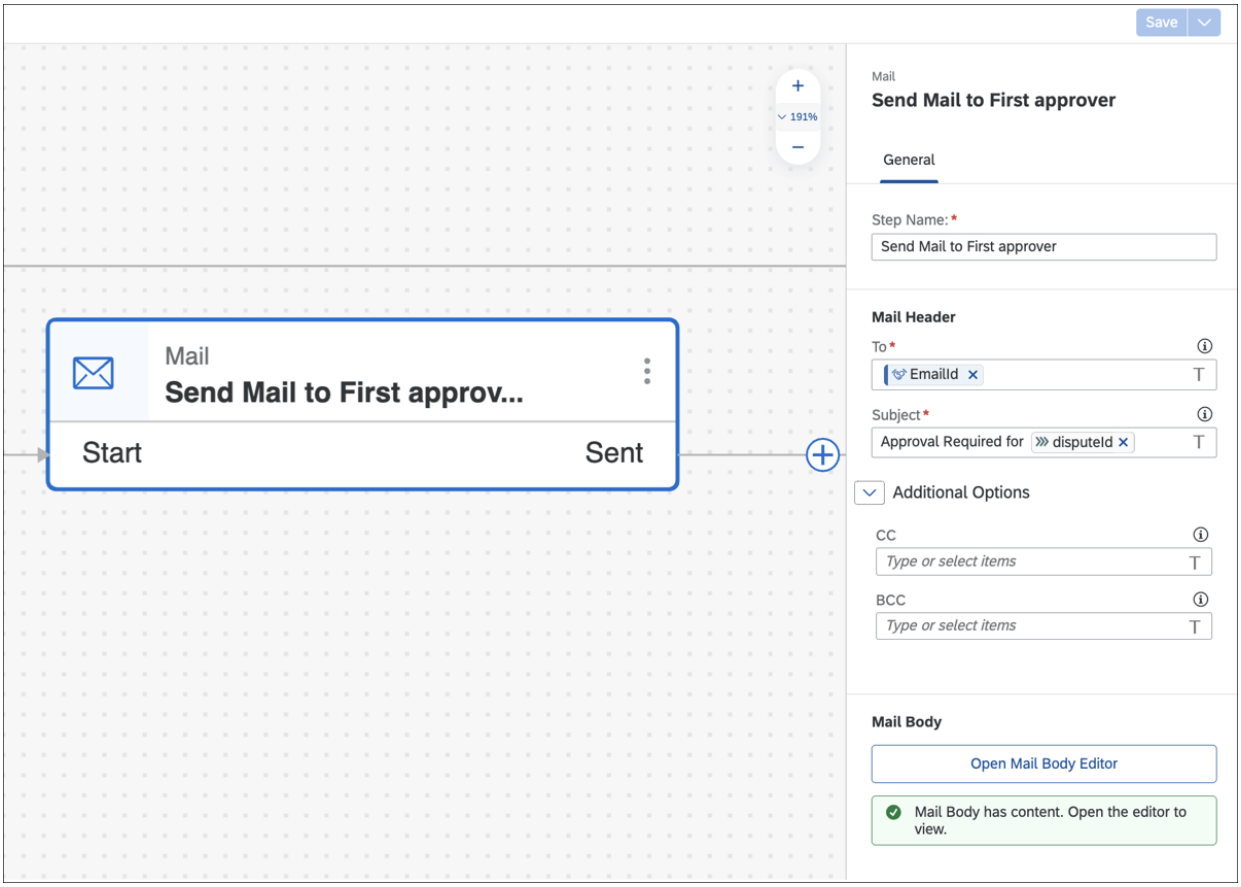


Figure 18.70 Creating a Mail Task

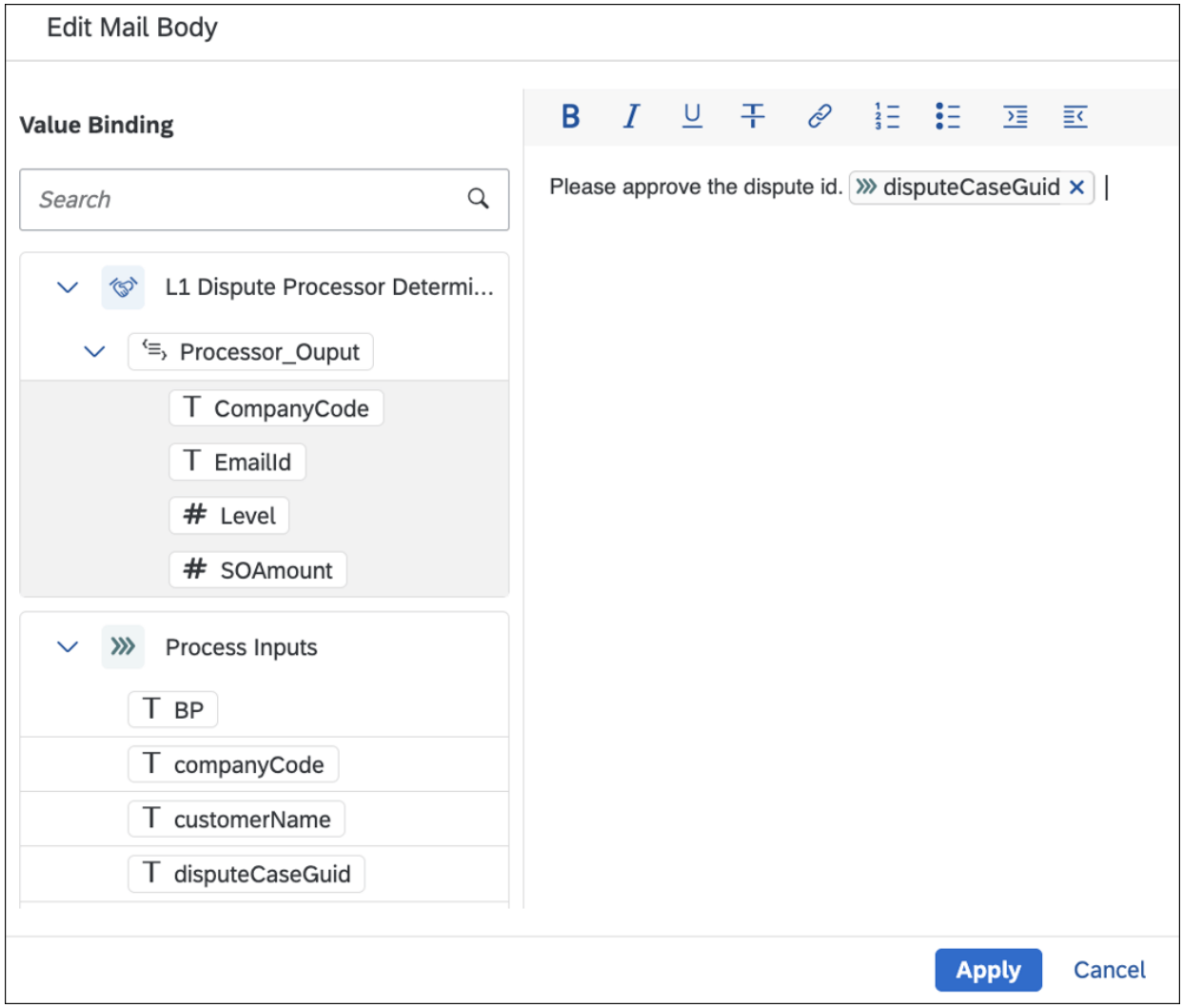


Figure 18.71 Editing the Mail Body

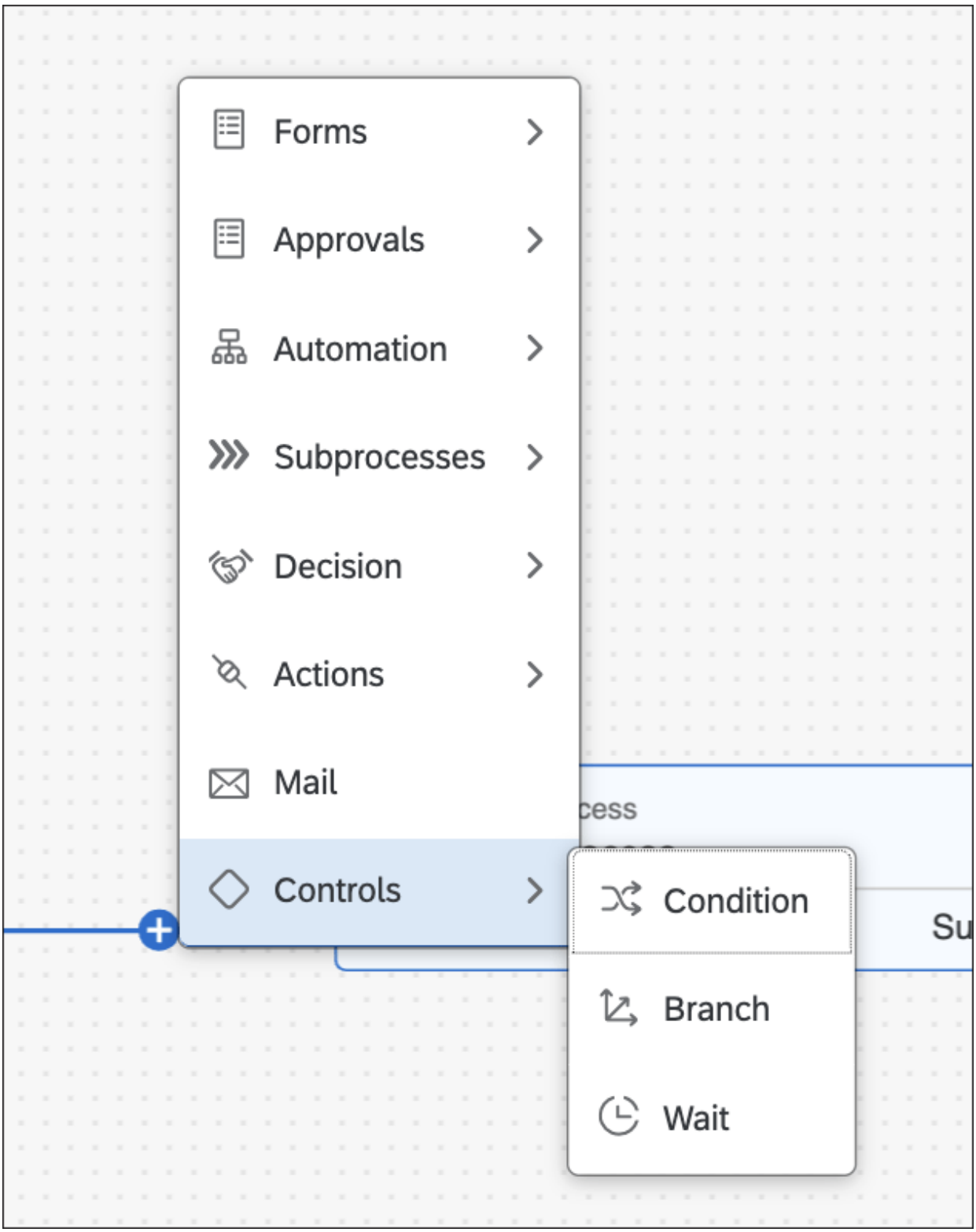


Figure 18.72 Controls in a Process

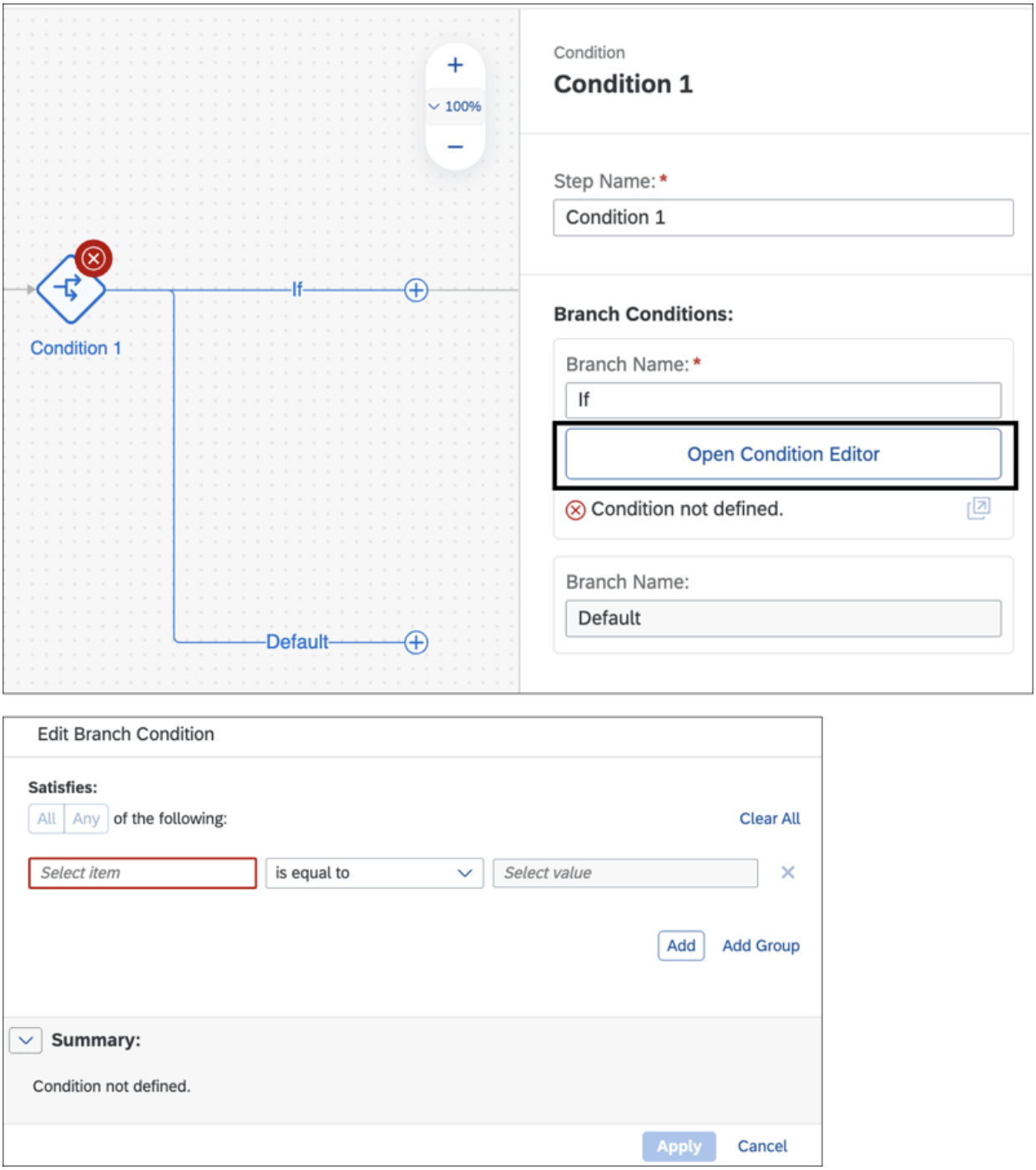


Figure 18.73 Condition Control

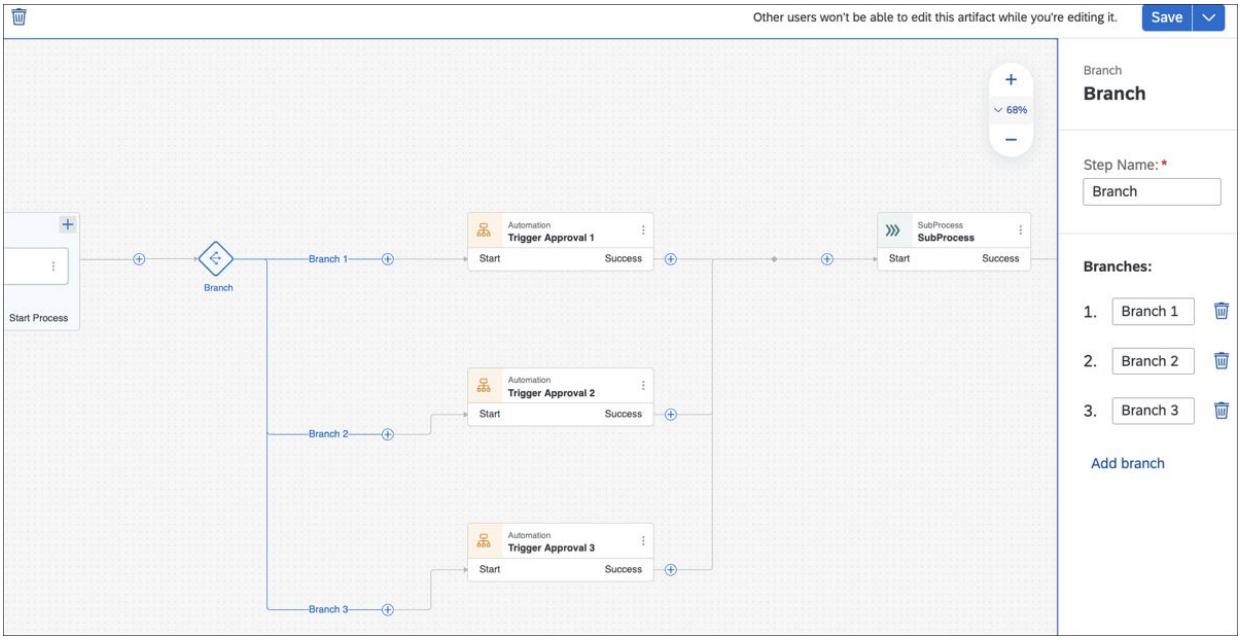


Figure 18.74 Branch Control

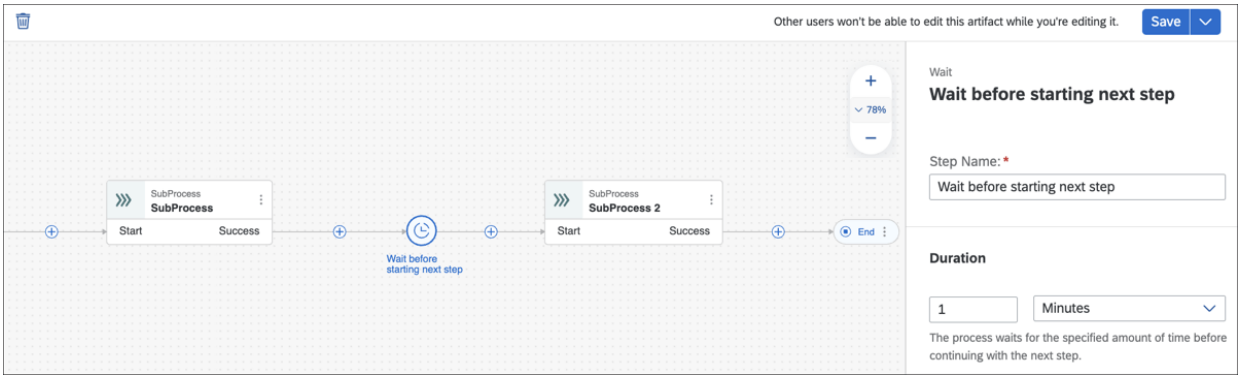


Figure 18.75 Wait Control

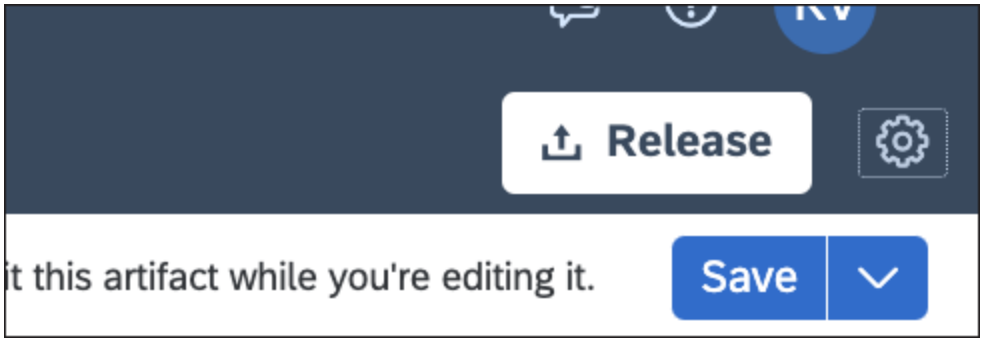


Figure 18.76 Releasing a Project

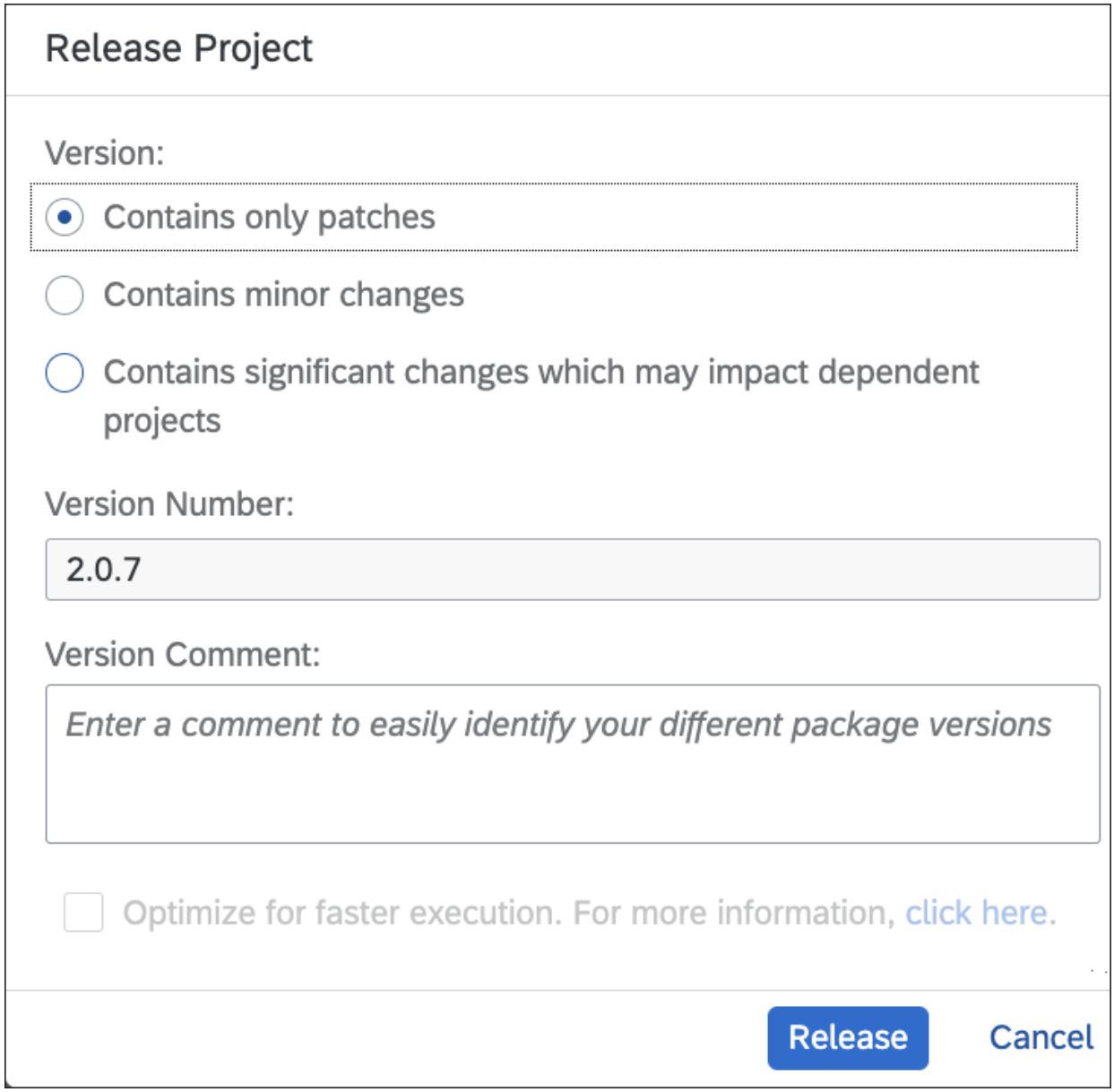


Figure 18.77 Versions When Releasing a Project

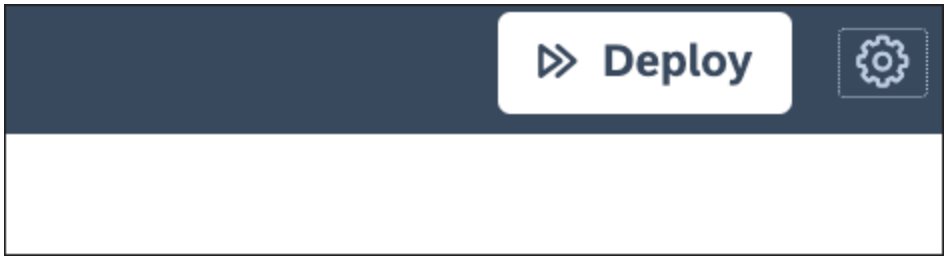


Figure 18.78 Deploy Button

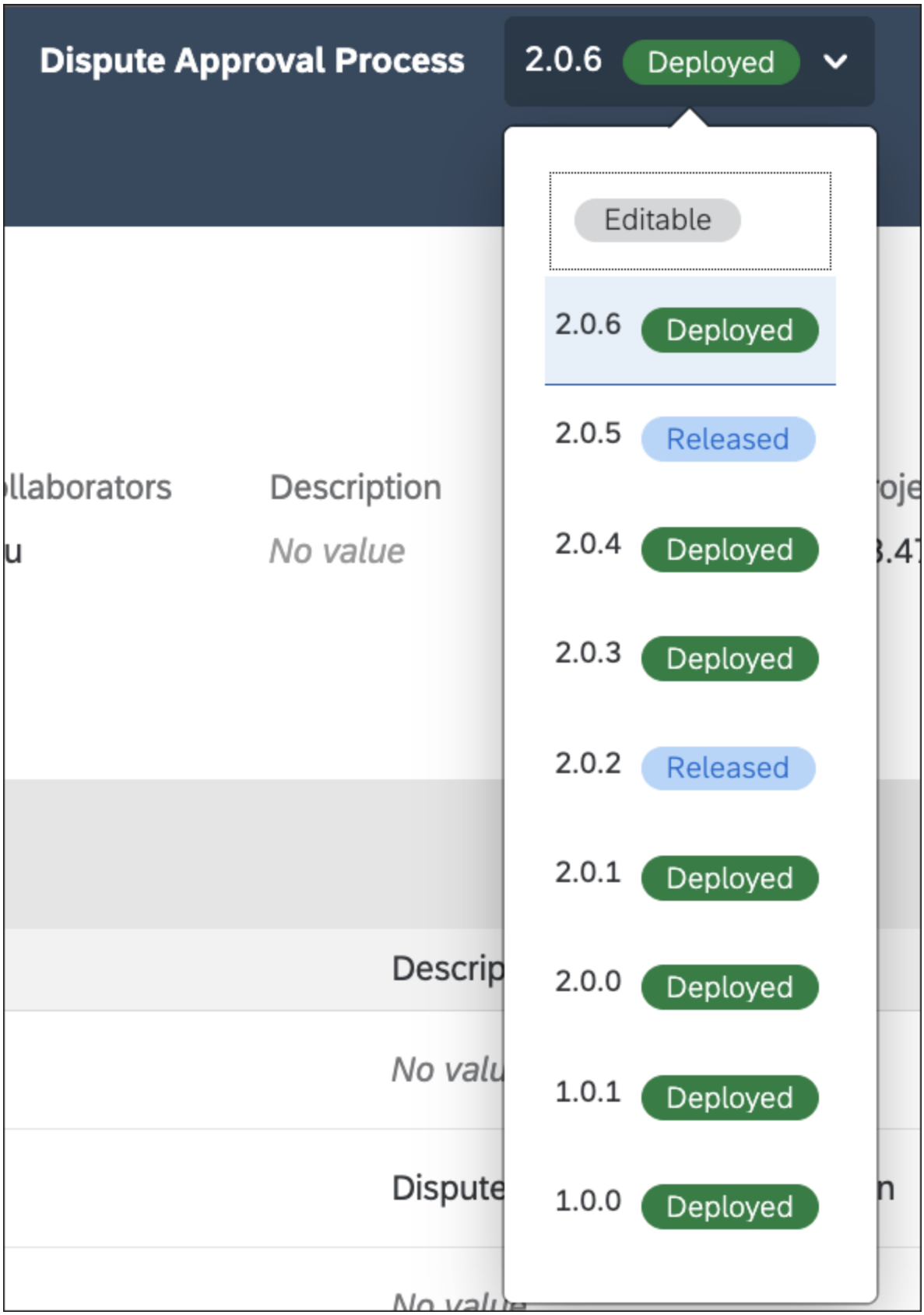

Dispute Approval Process
2.0.6
Deployed
Editable
2.0.6
Deployed
2.0.5
Released
2.0.4
Deployed
2.0.3
Deployed
2.0.2
Released
2.0.1
Deployed
2.0.0
Deployed
1.0.1
Deployed
1.0.0
Deployed
llaborators
Description
u
No value
Descrip
No valu
Dispute

Figure 18.79 Versions of the Project

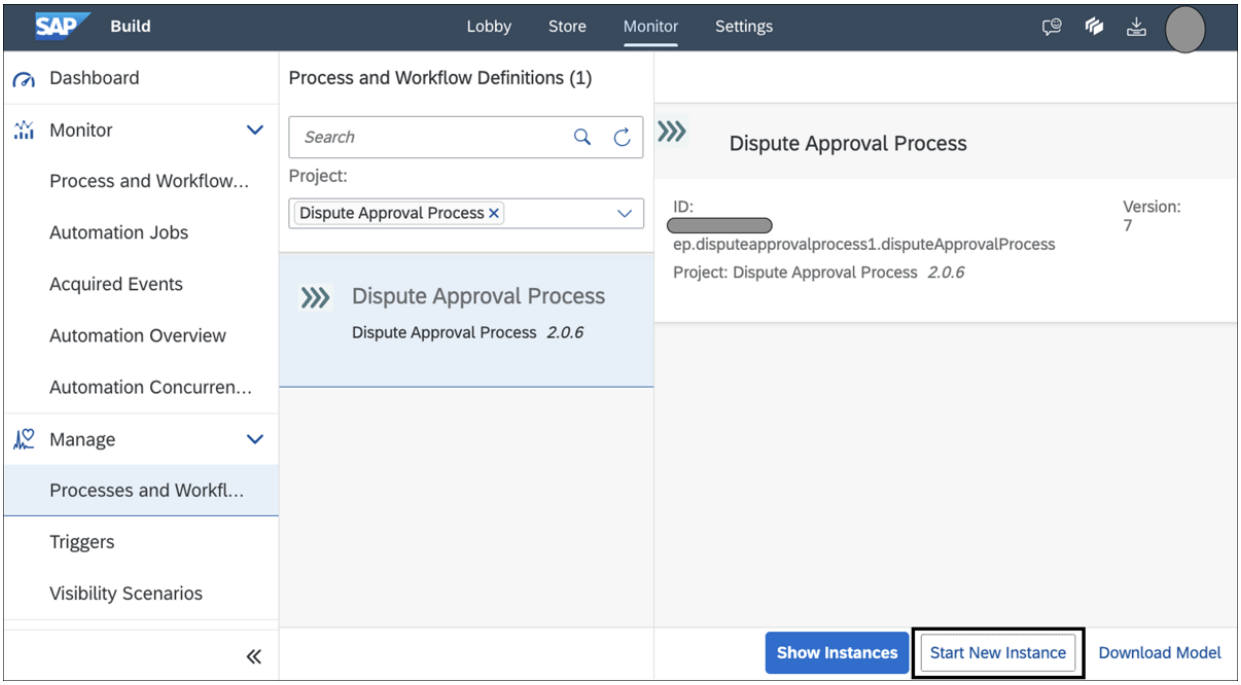


Figure 18.80 Testing or Starting a New Instance of the Deployed Process

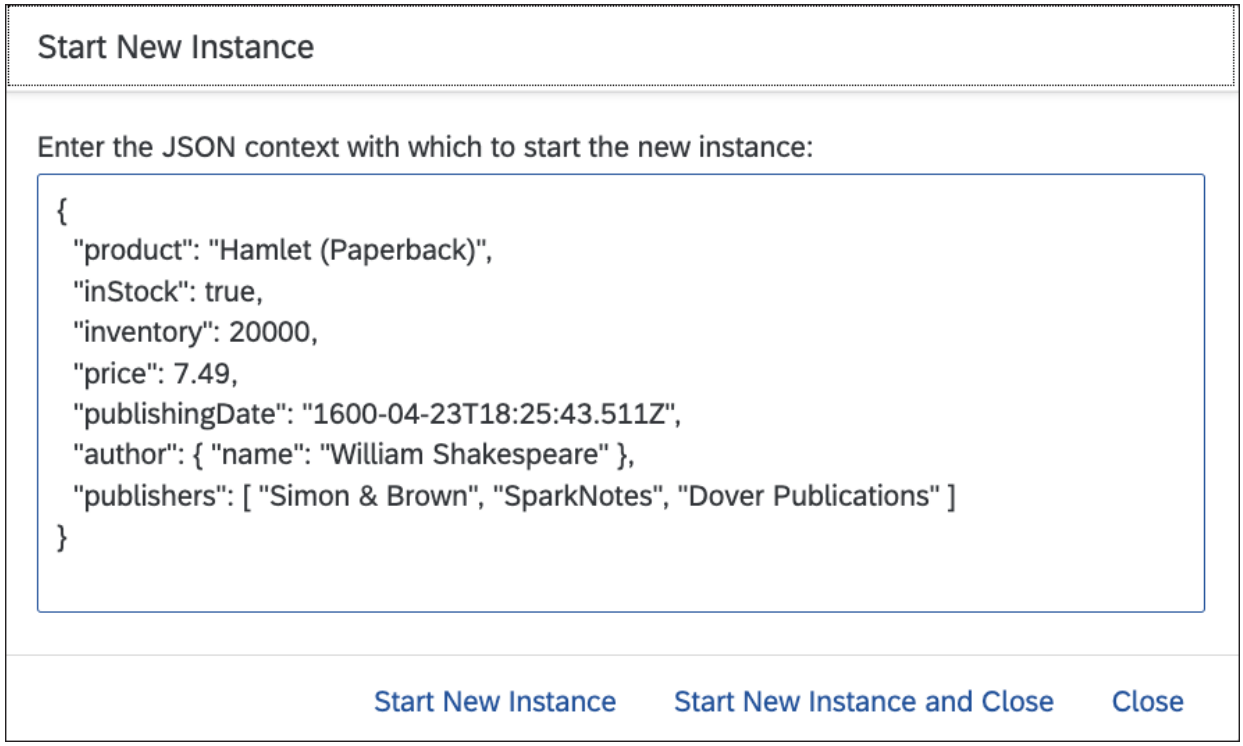


Figure 18.81 Start a New Instance after Entering a JSON Payload

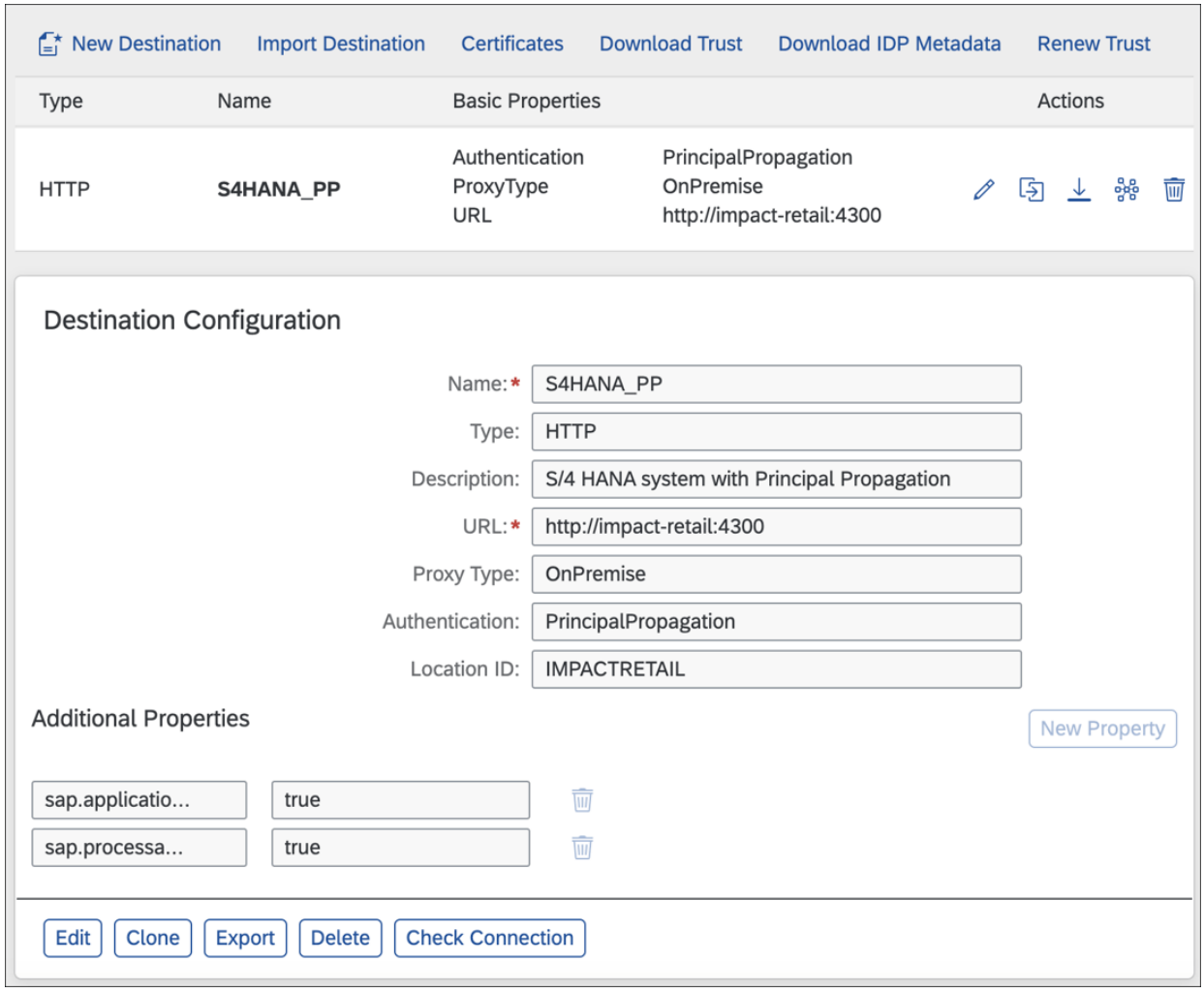


Figure 18.82 Additional Parameters for the Destination

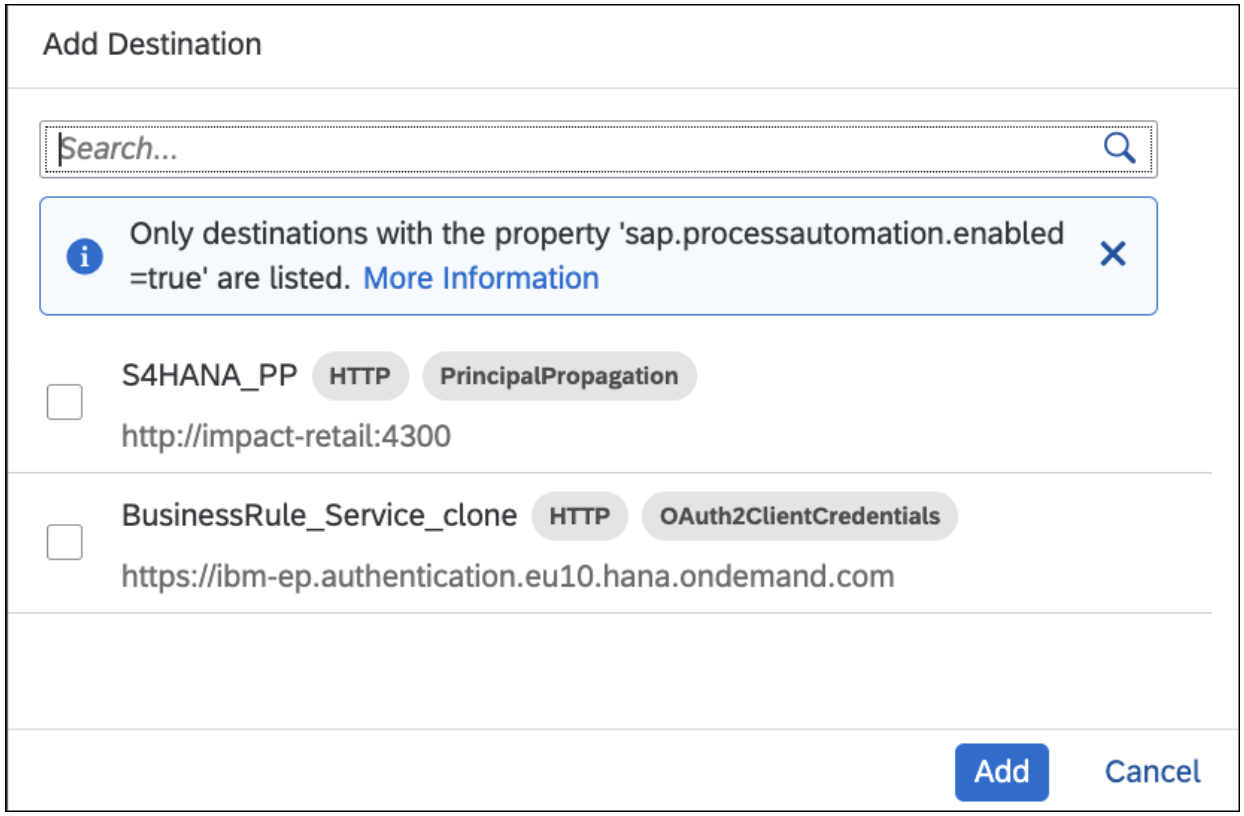


Figure 18.83 Adding the New Destination

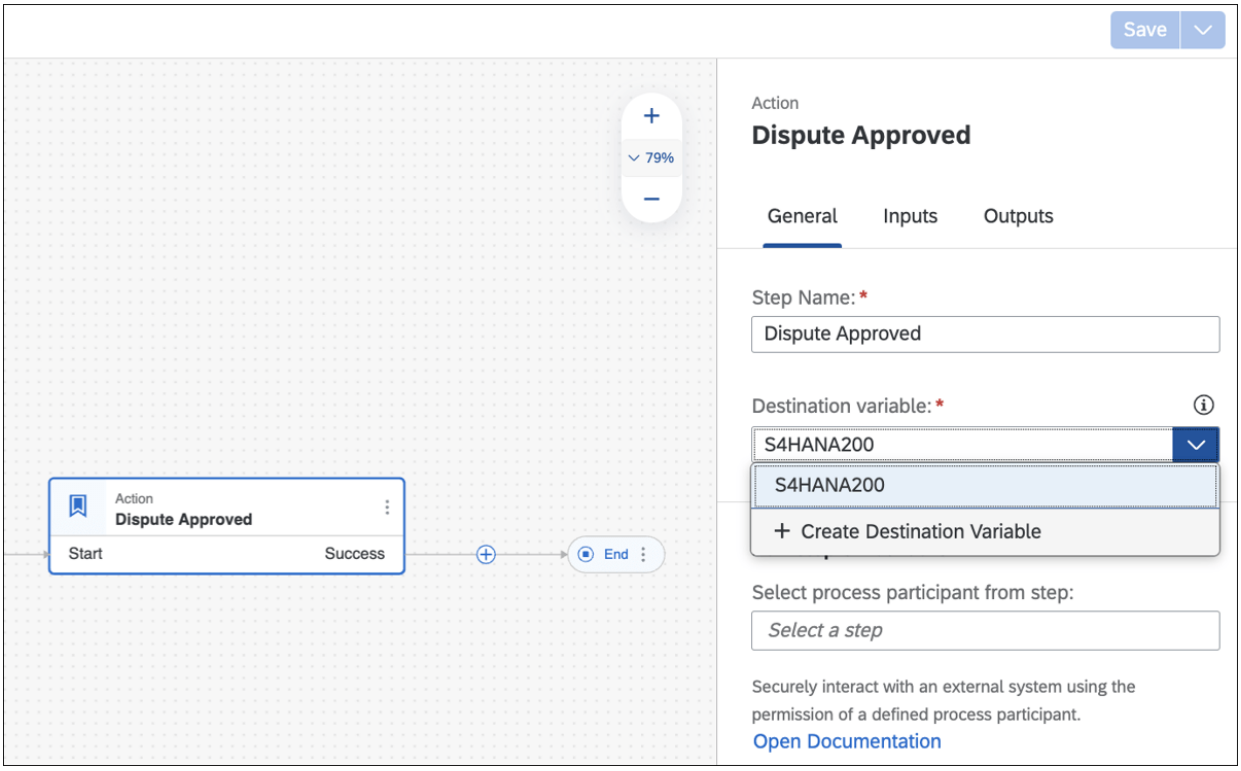


Figure 18.84 Adding a Destination Variable for Your Action Project

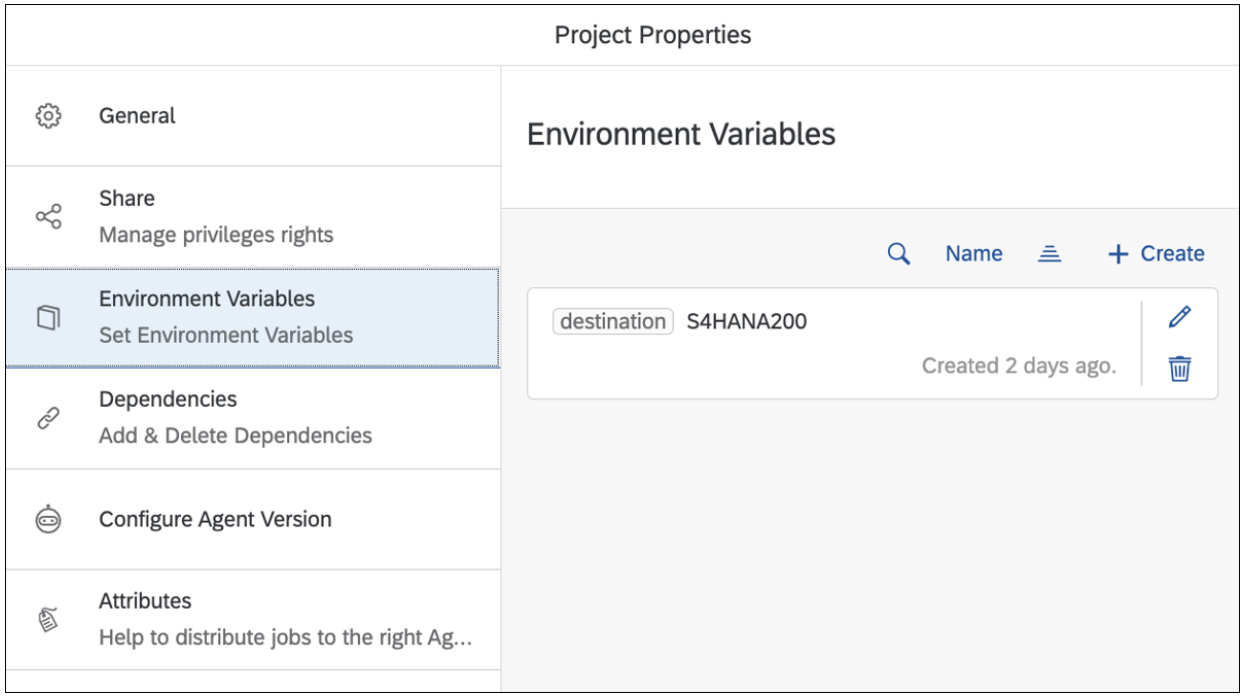


Figure 18.85 Adding Destination Environment Variable under Project Properties

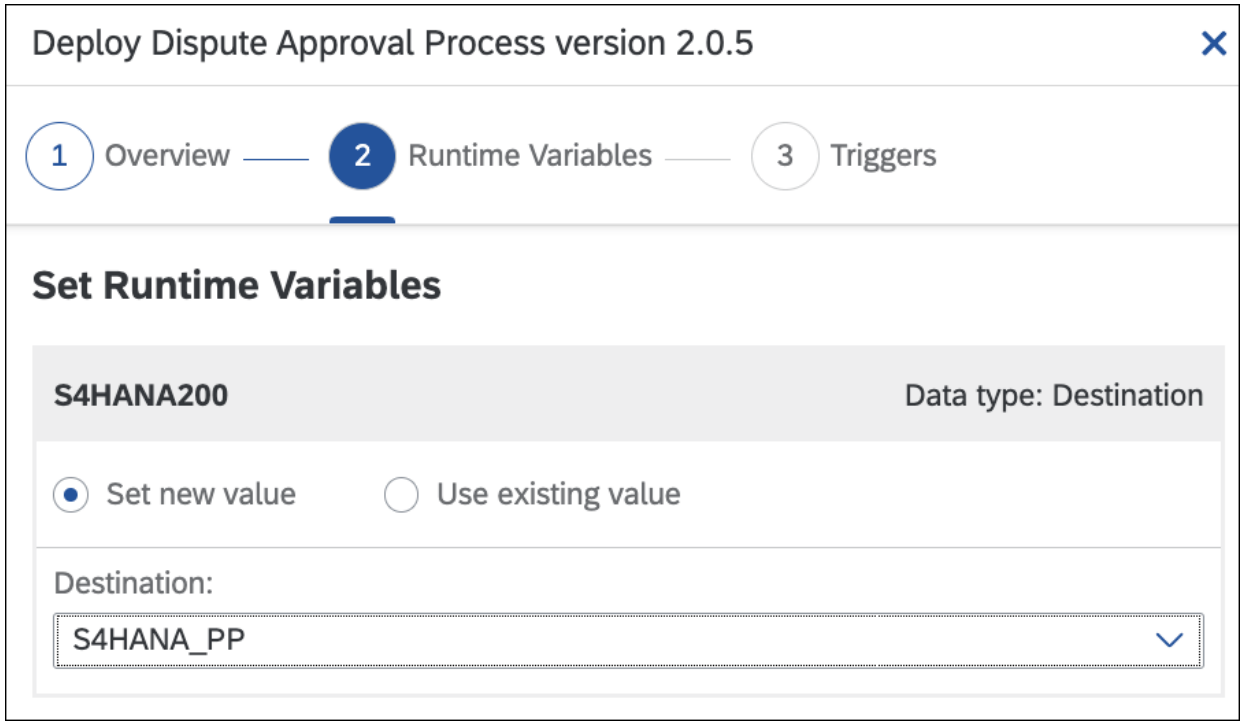


Figure 18.86 Mapping the Environment Variable to the Destination while Deploying

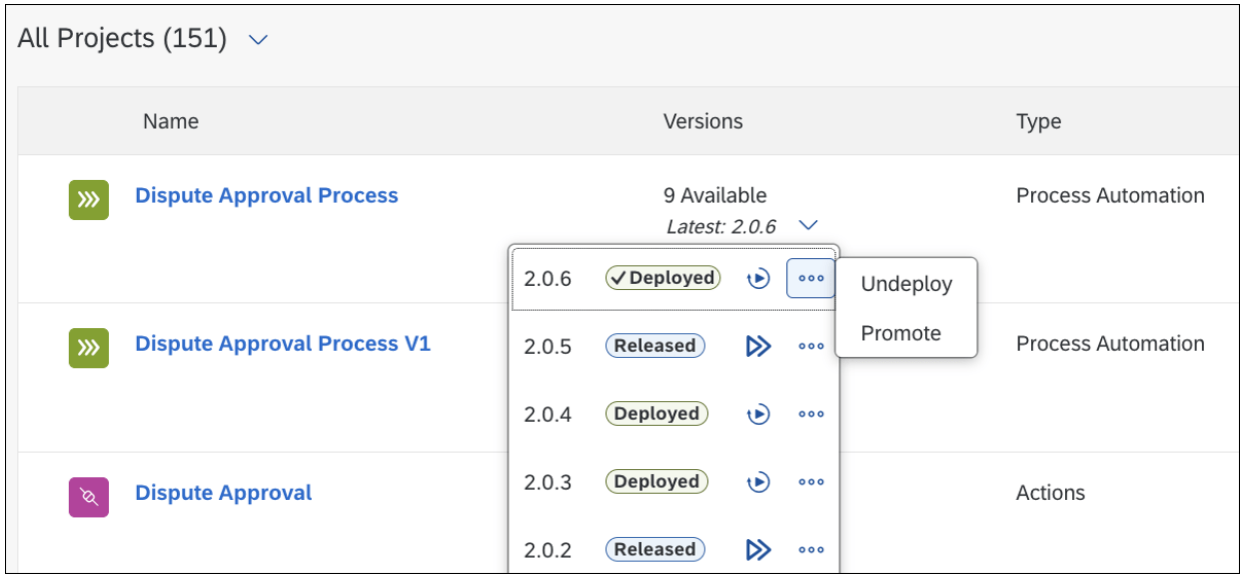


Figure 18.87 Transport Using the Promote Feature

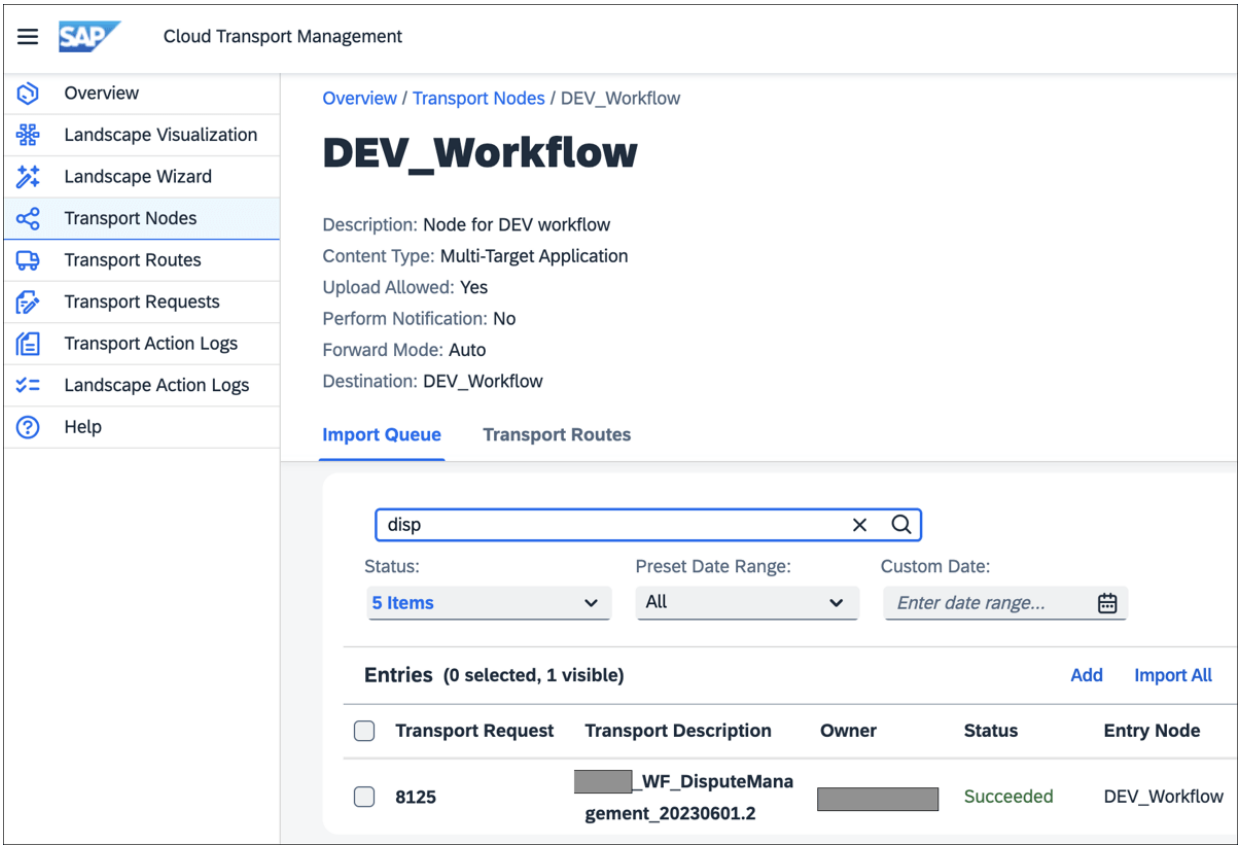


Figure 18.88 Transport Created in the Dev Node of SAP Cloud Transport Management System

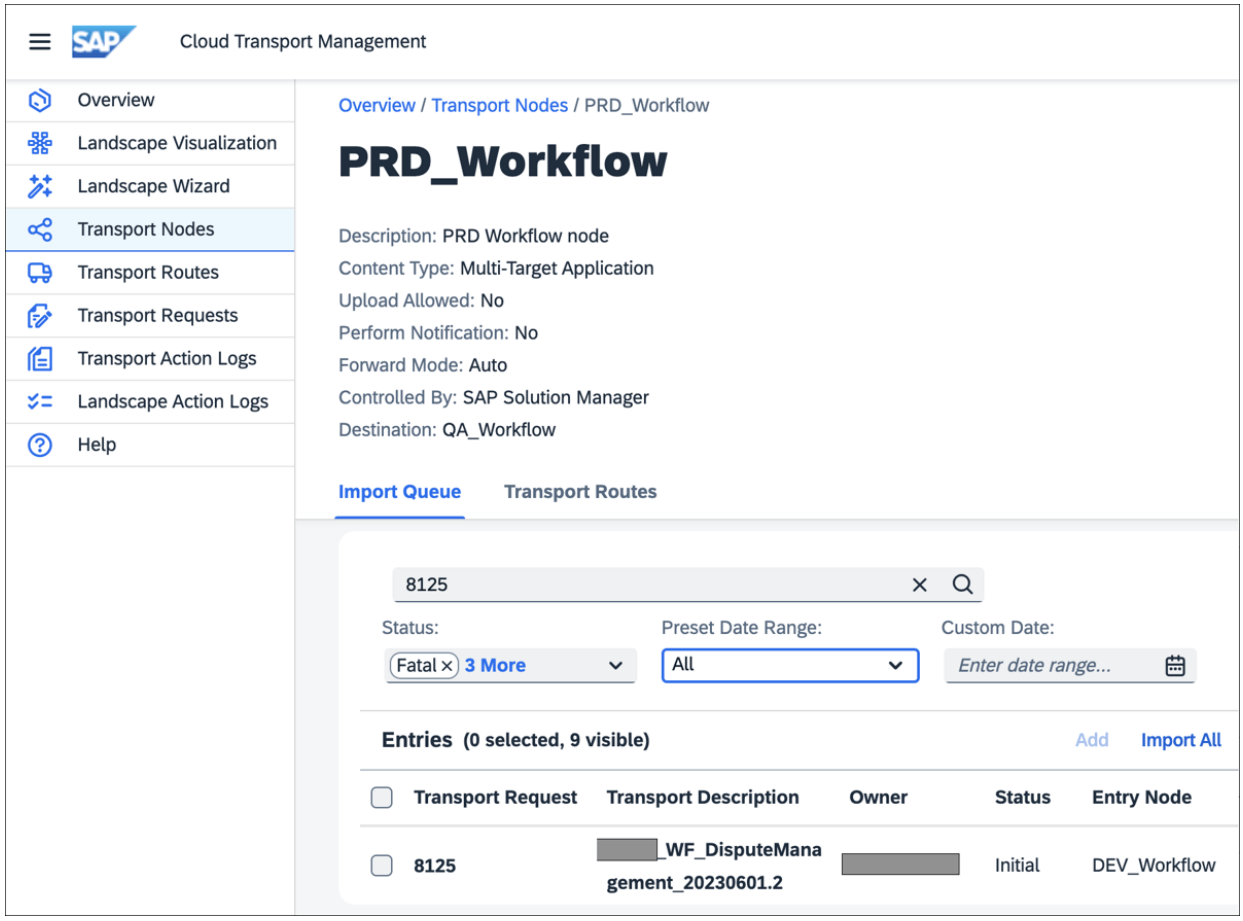


Figure 18.89 Transport Waiting in the Production Node to Be Imported

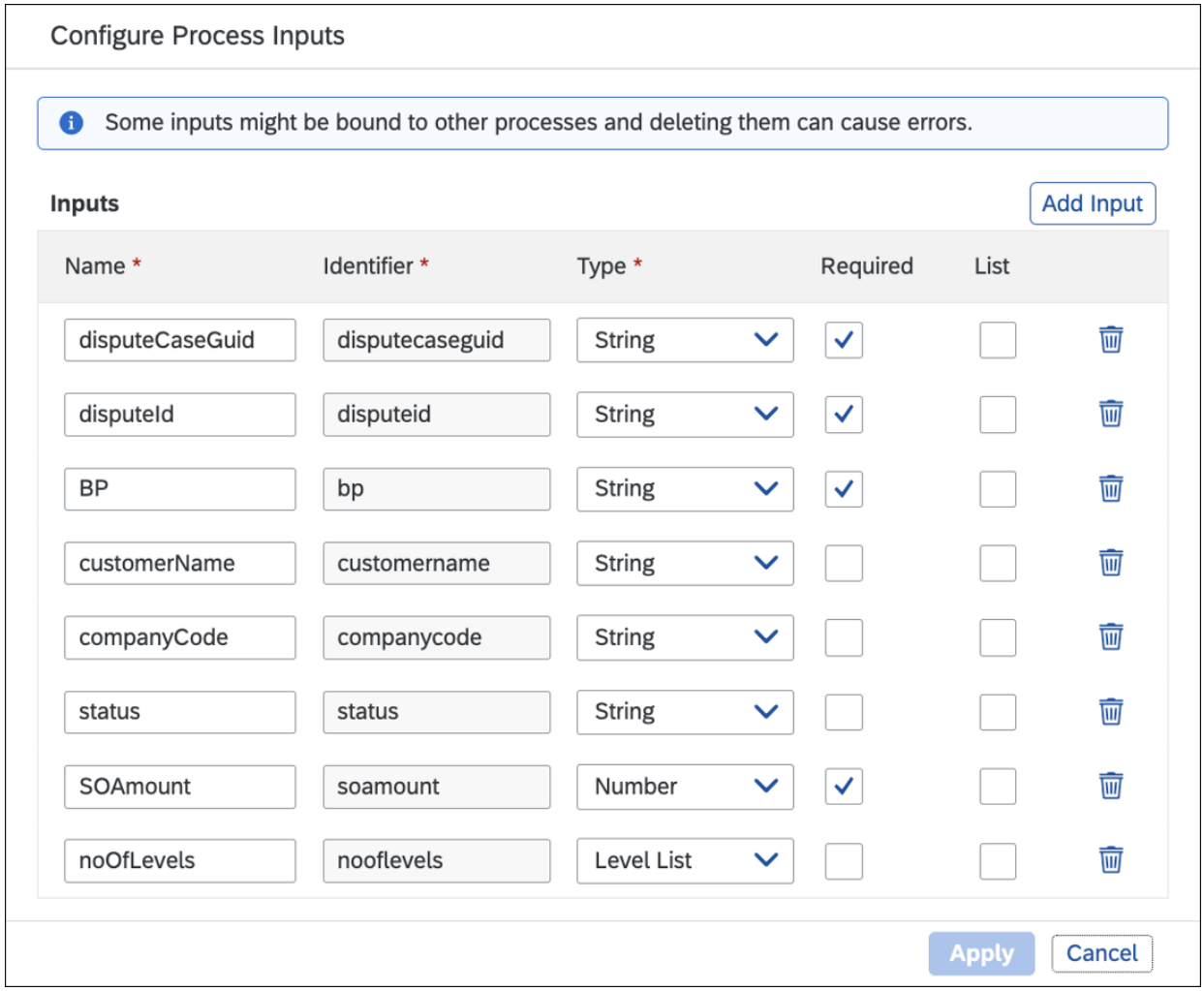


Figure 18.90 Adding Context Fields

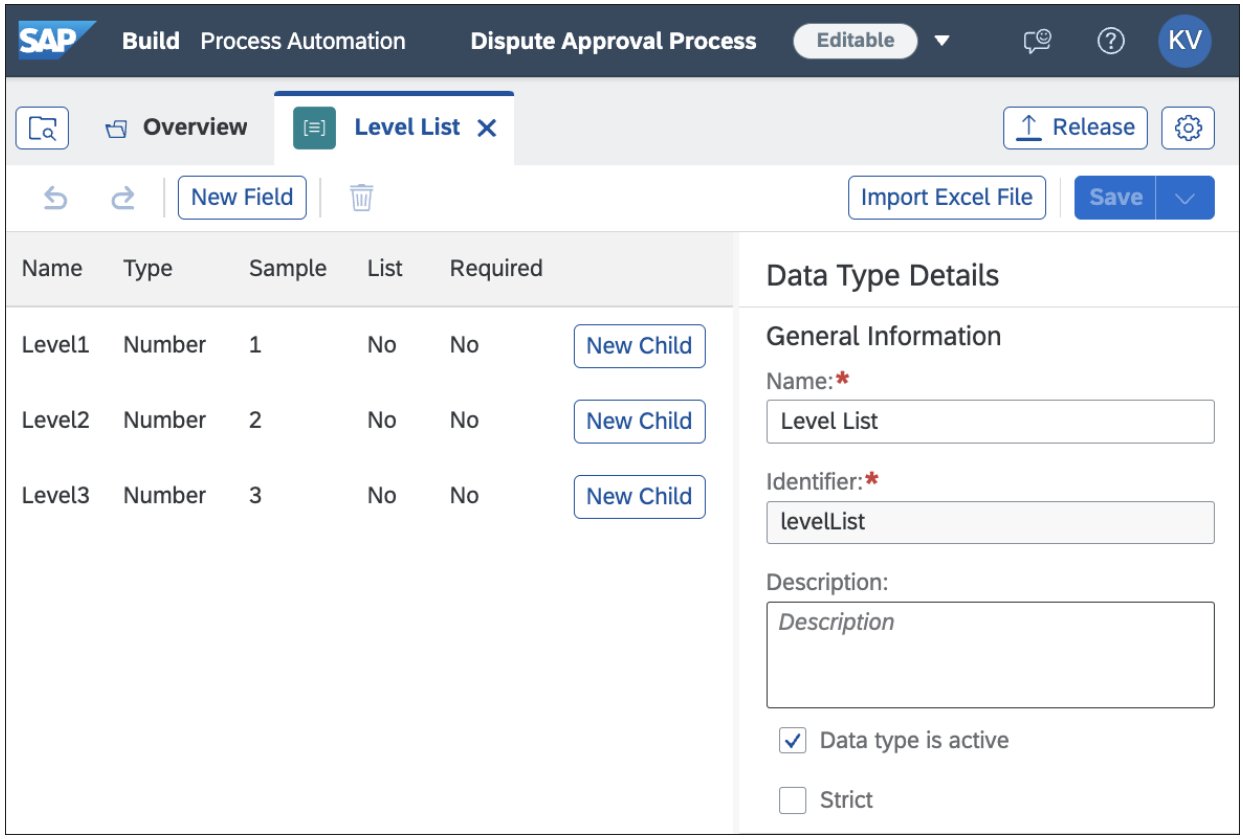


Figure 18.91 Creating a Custom Data Type

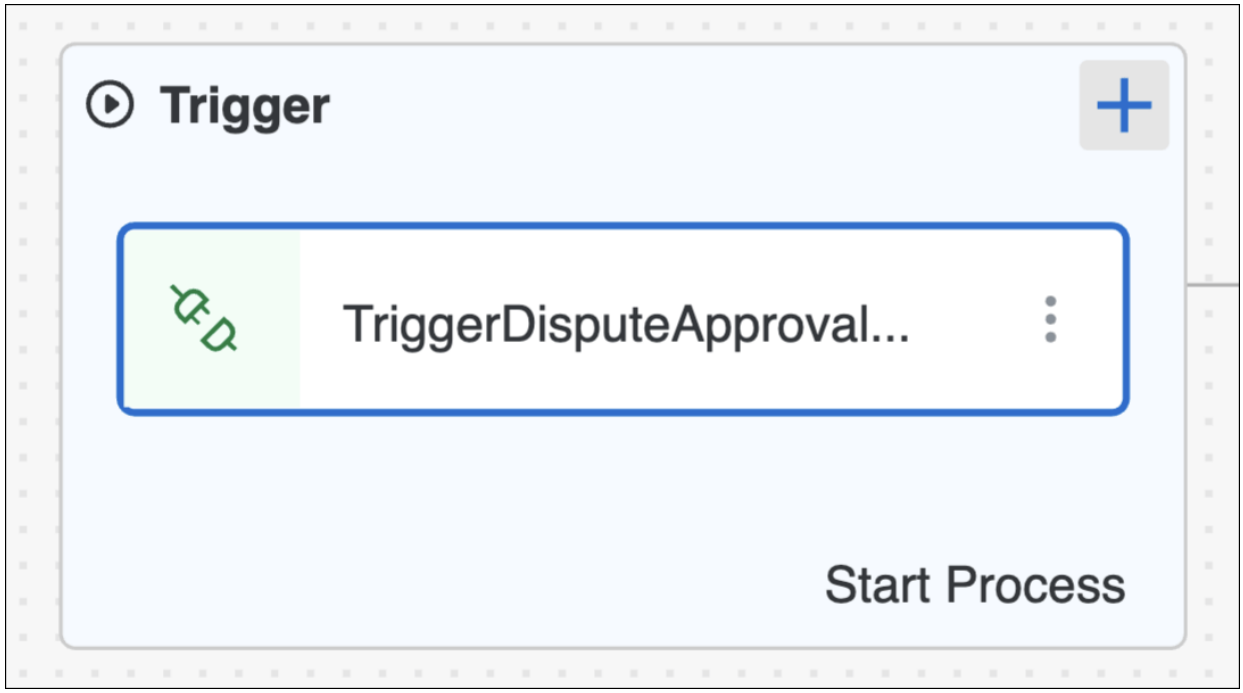


Figure 18.92 Configuring the Trigger

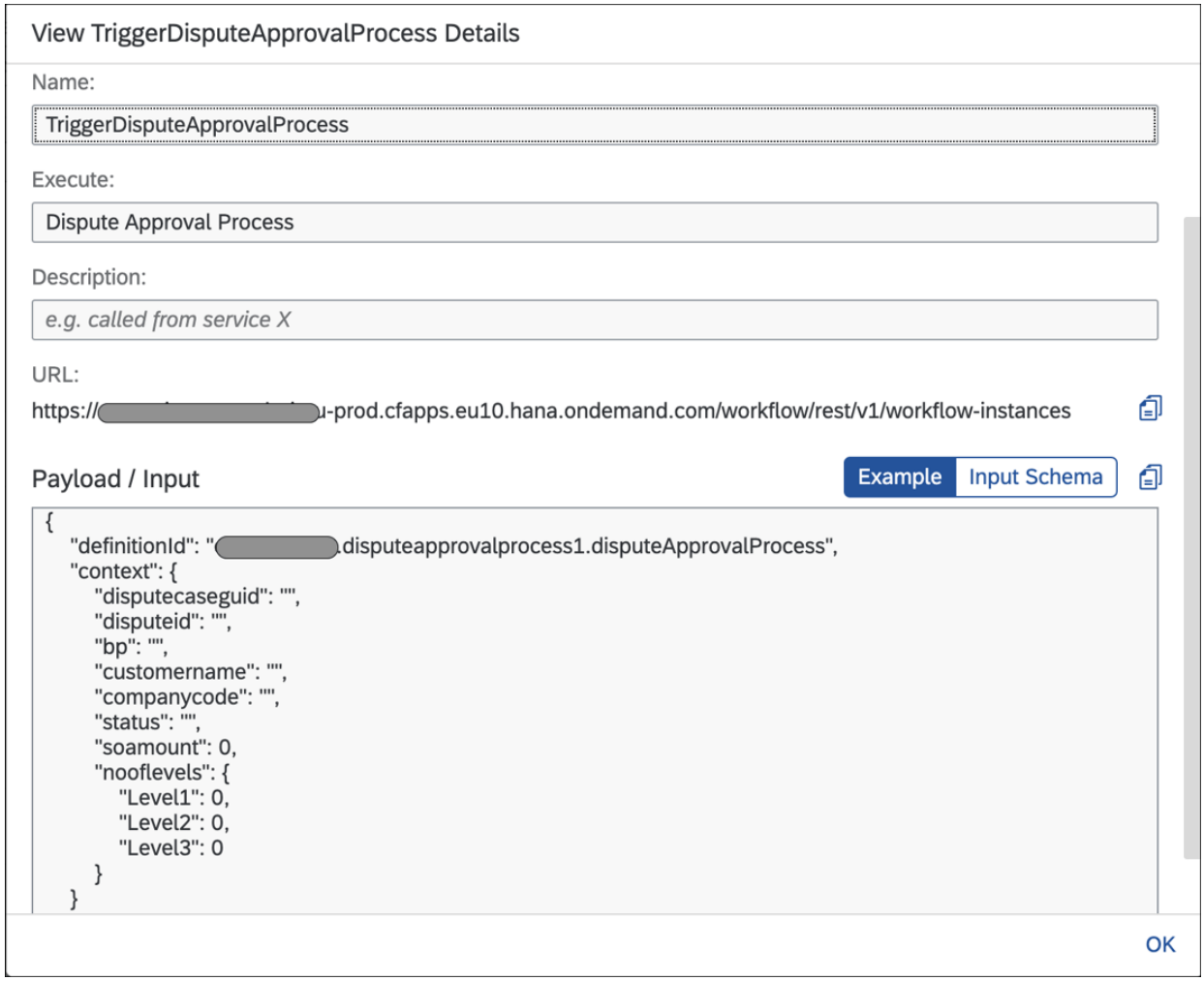


Figure 18.93 View Trigger Details

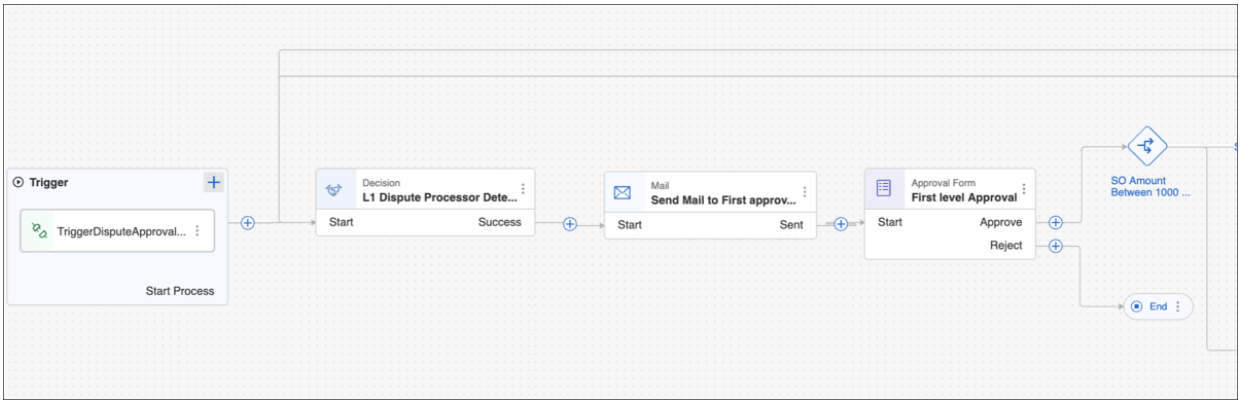


Figure 18.94 Initial Steps of the Process

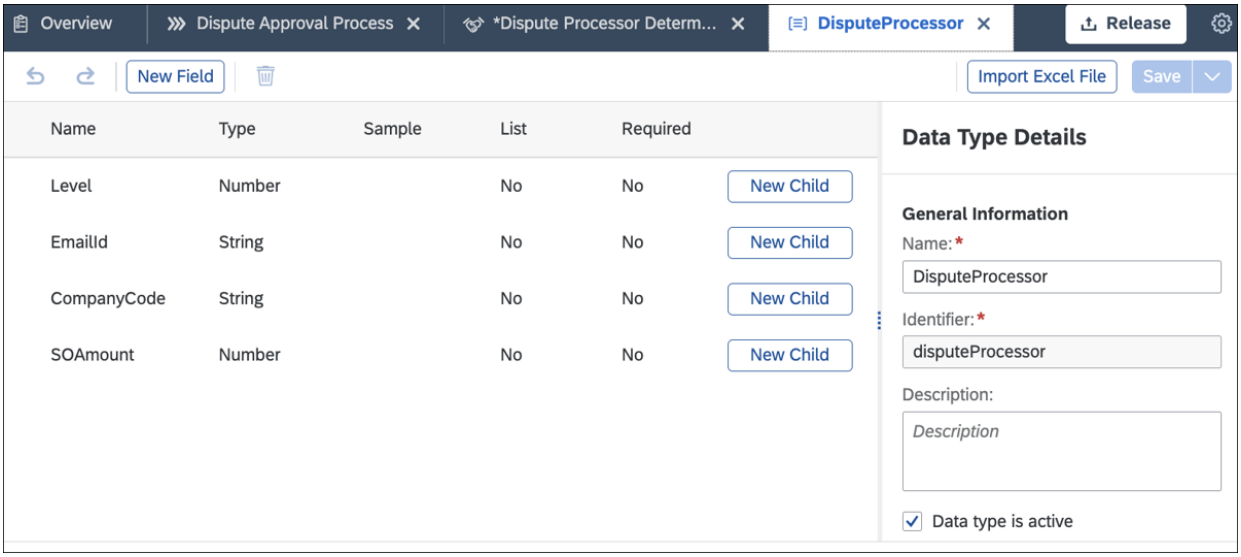


Figure 18.95 Creating a Custom Data Type

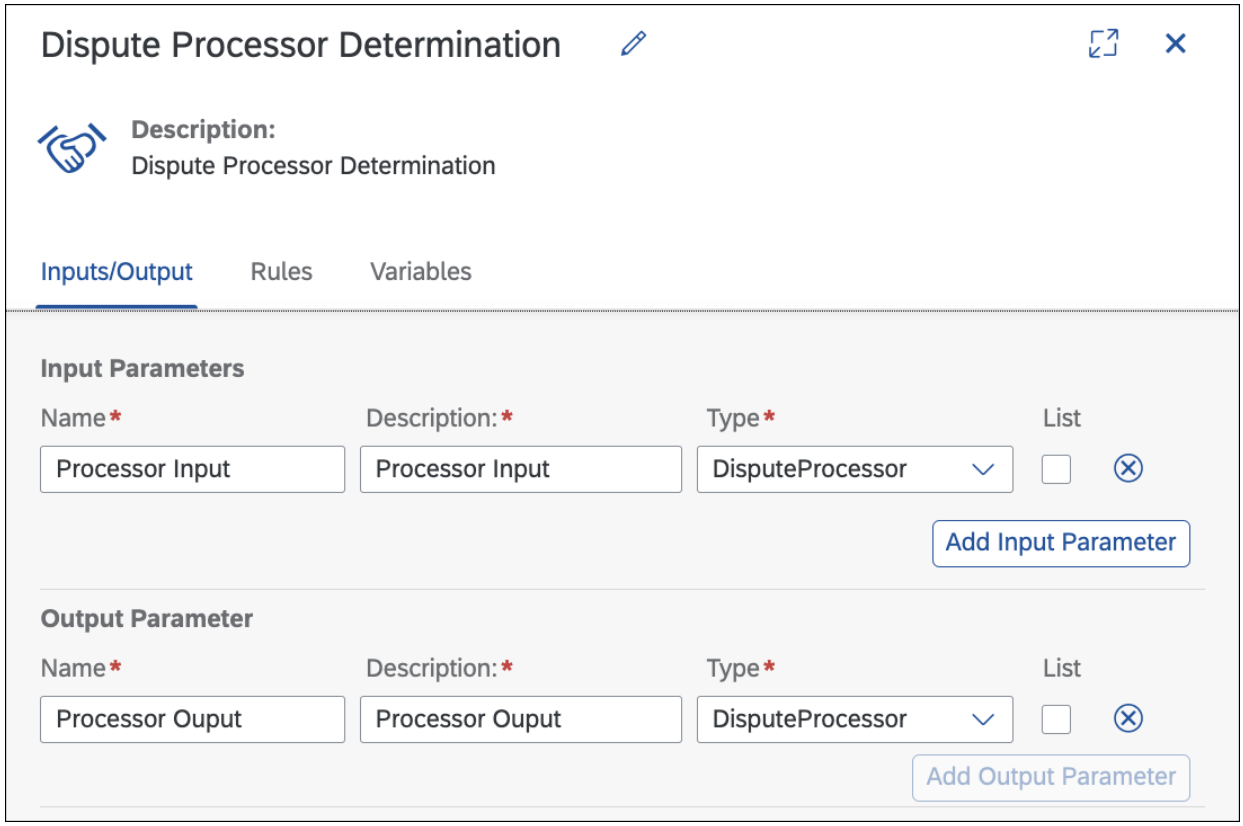


Figure 18.96 Setting the Decision Parameters

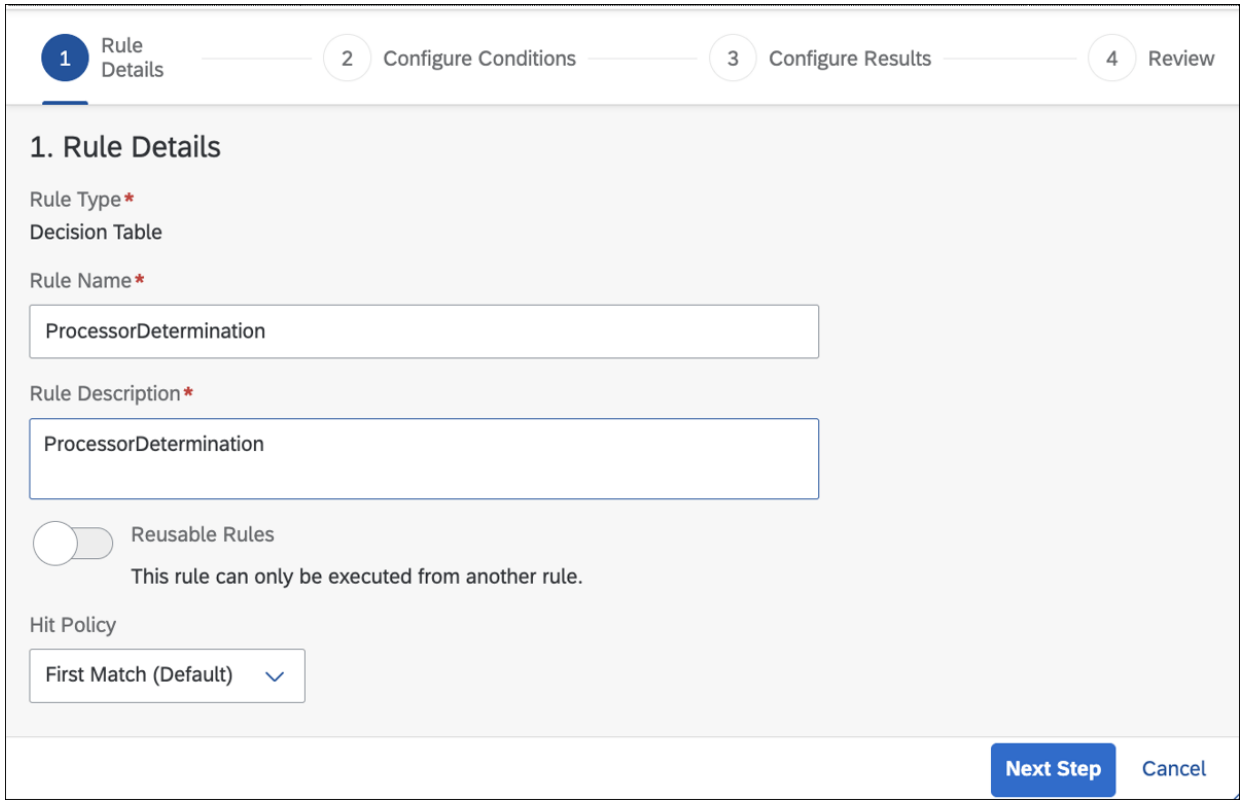


Figure 18.97 Rule Details

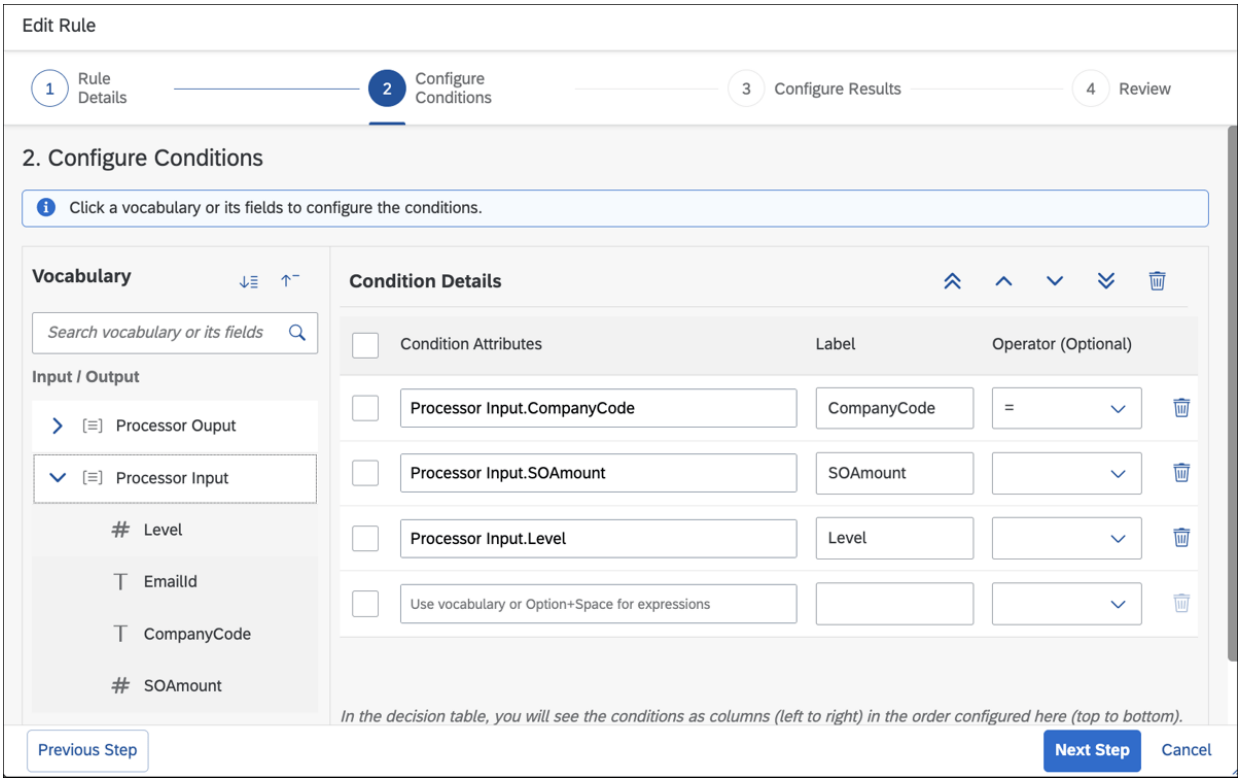


Figure 18.98 Configuring the Conditions

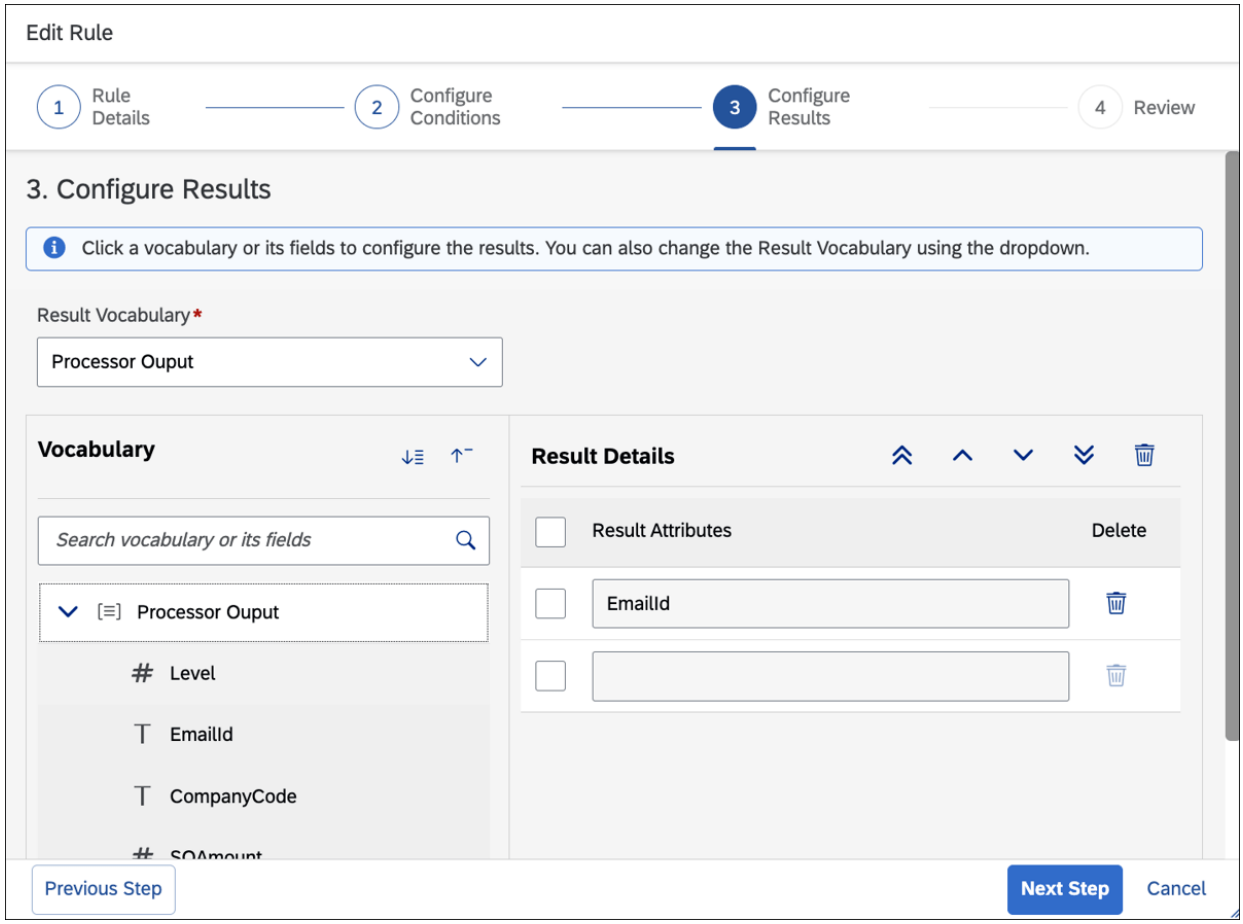


Figure 18.99 Configuring the Output Results

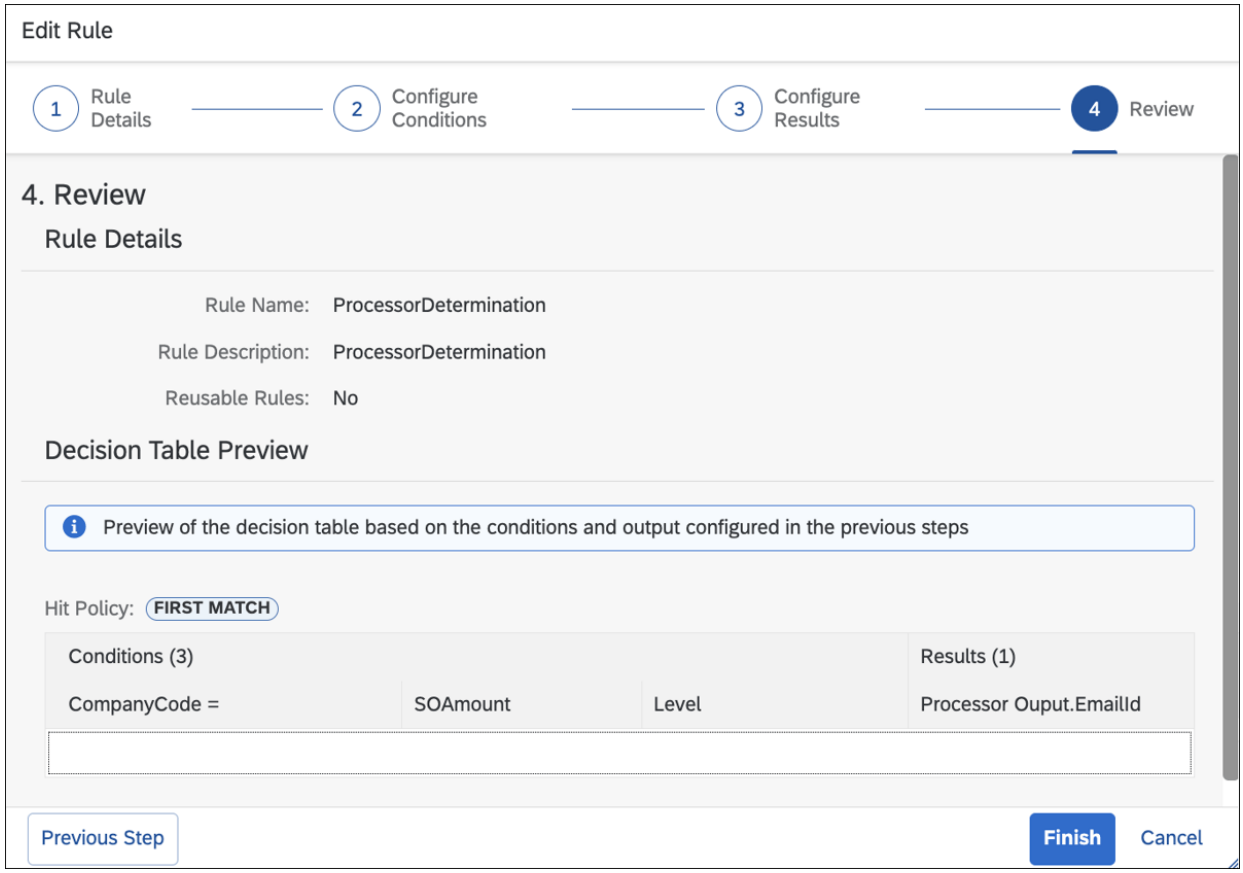


Figure 18.100 Reviewing the Rule

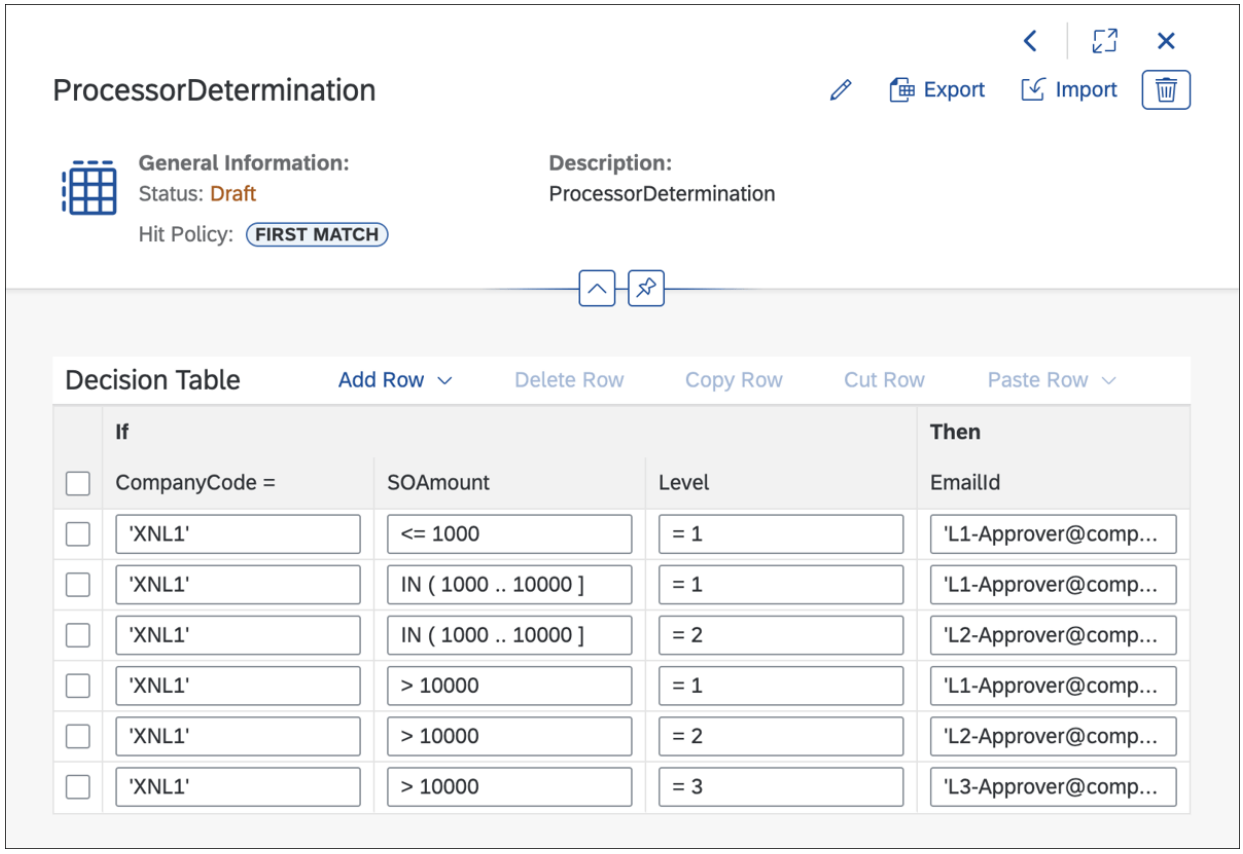


Figure 18.101 Decision Table

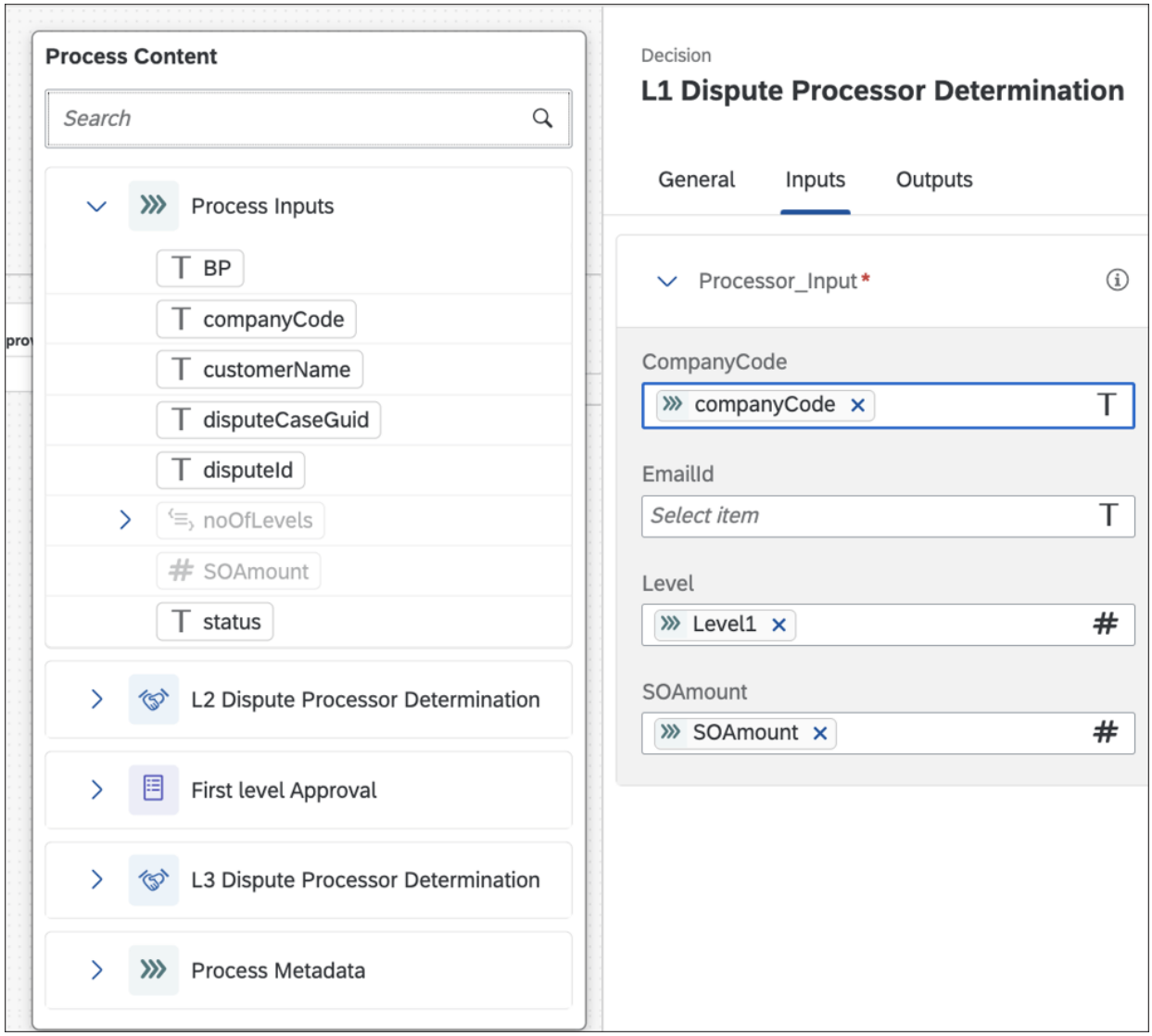


Figure 18.102 Decision Field Mapping

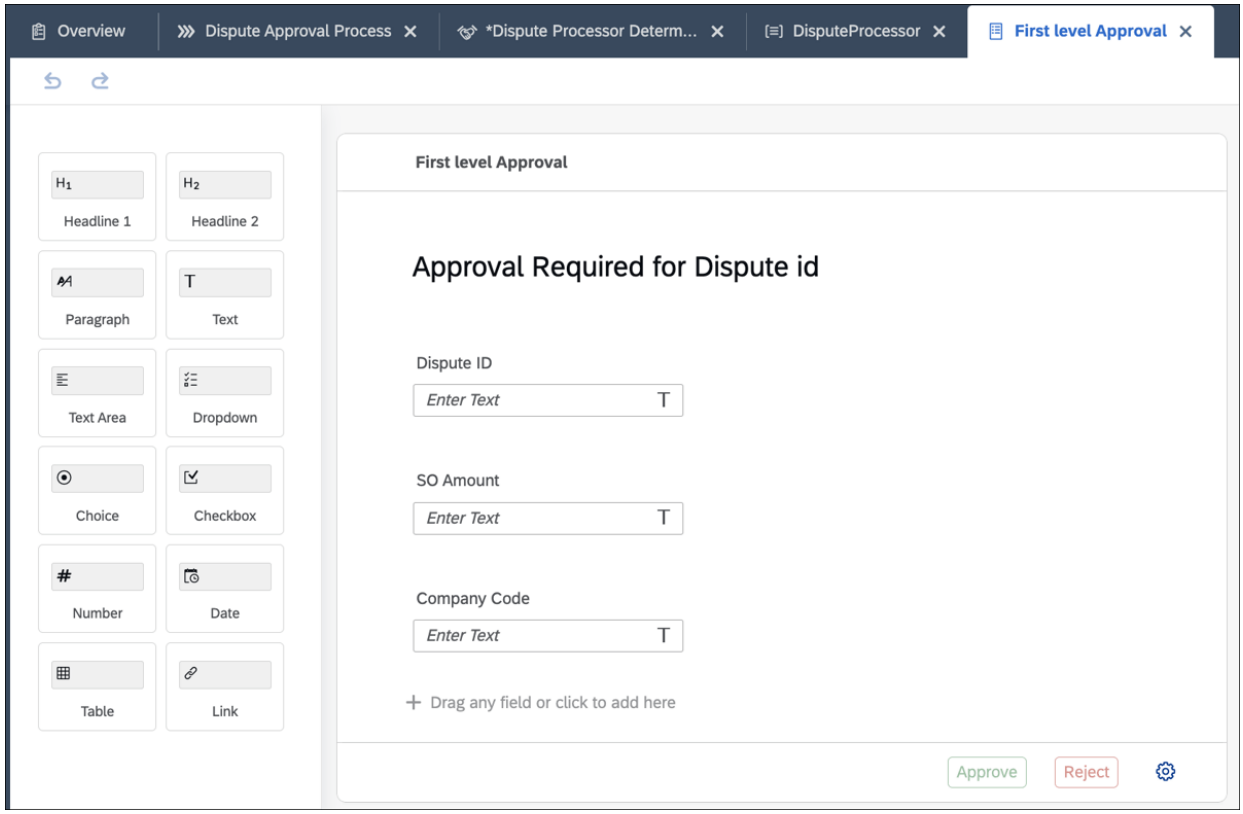


Figure 18.103 Create Approval Forms

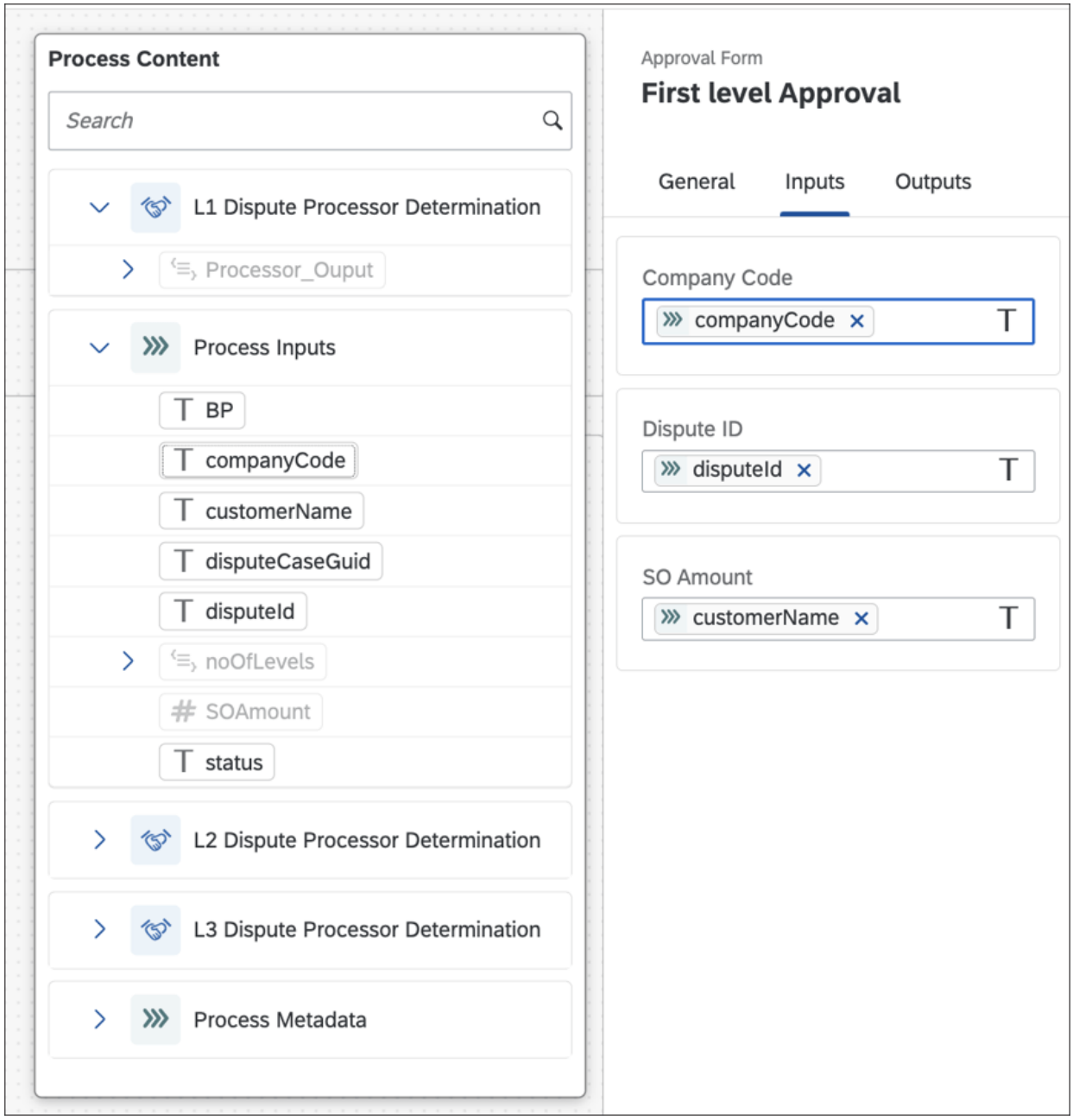


Figure 18.104 Form Field Data Mapping from the Context

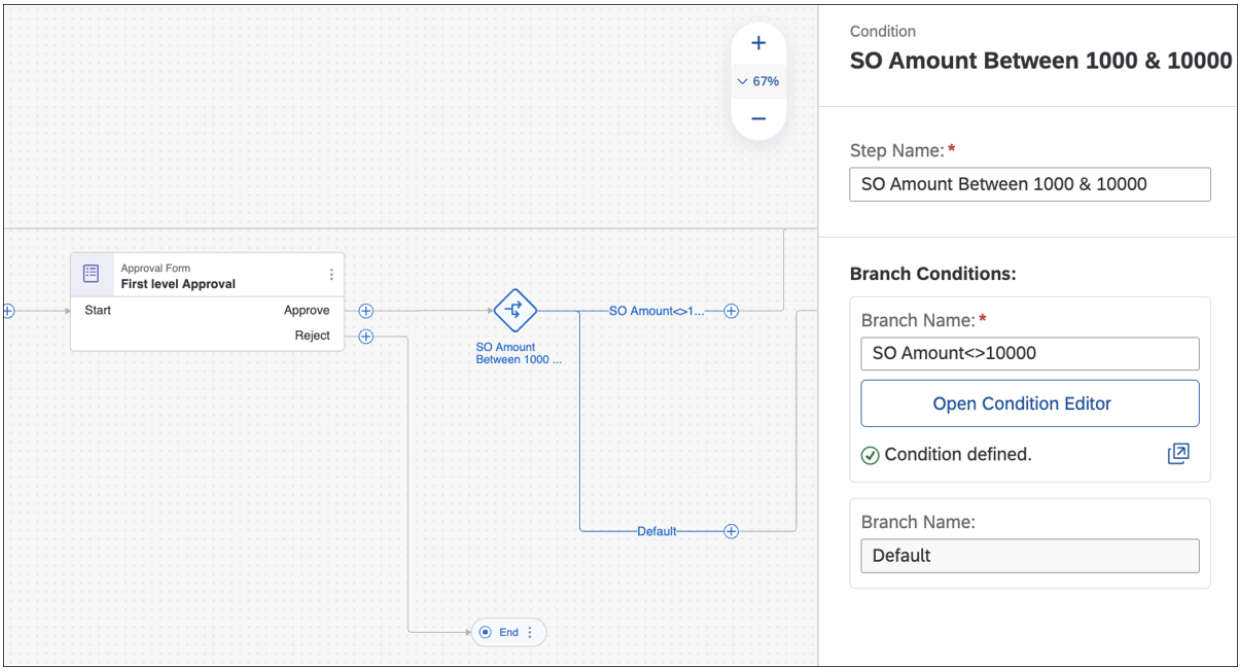


Figure 18.105 Creating a Condition

Figure 18.106 Adding Condition Criteria

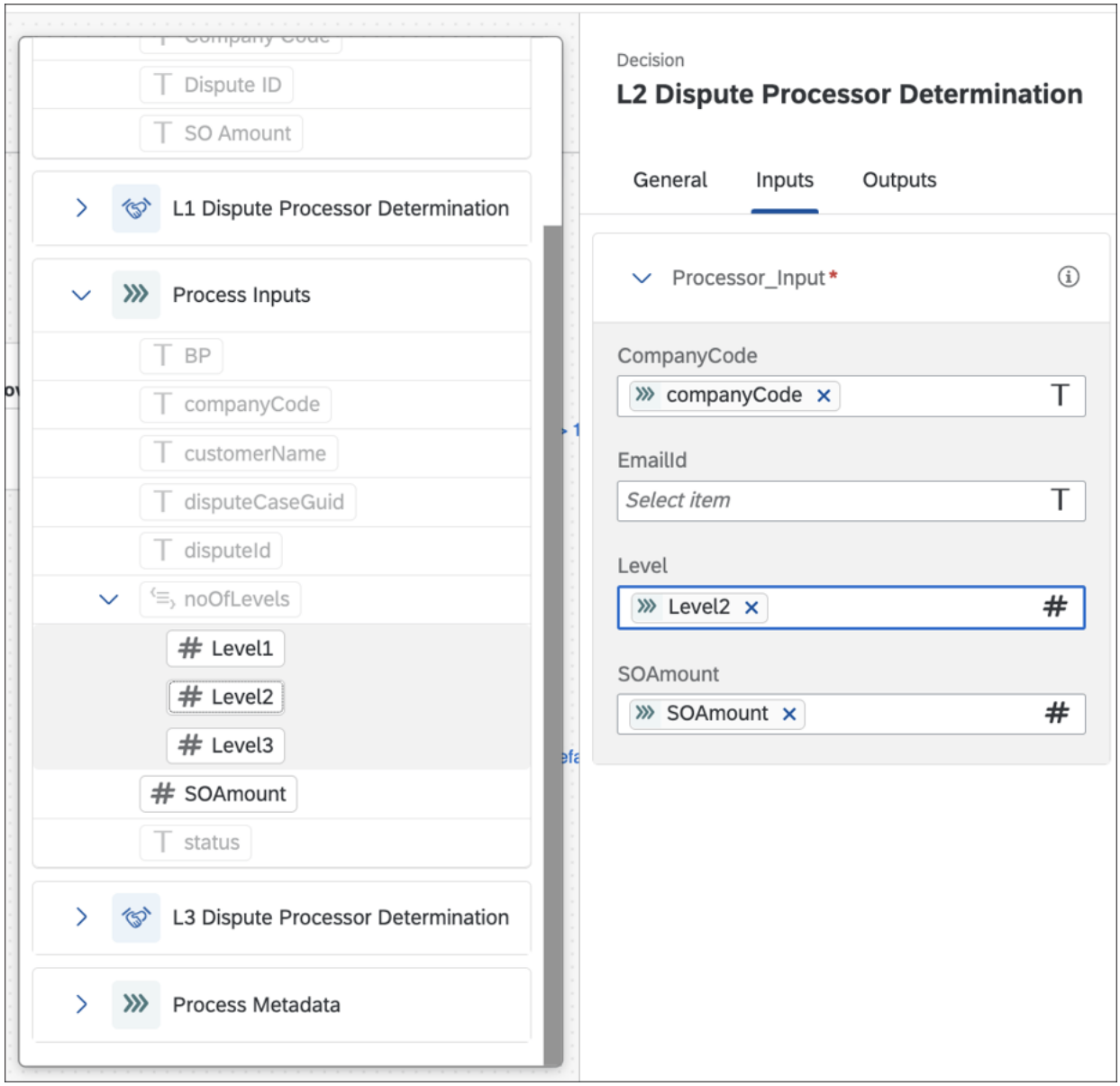


Figure 18.107 Second Decision Step

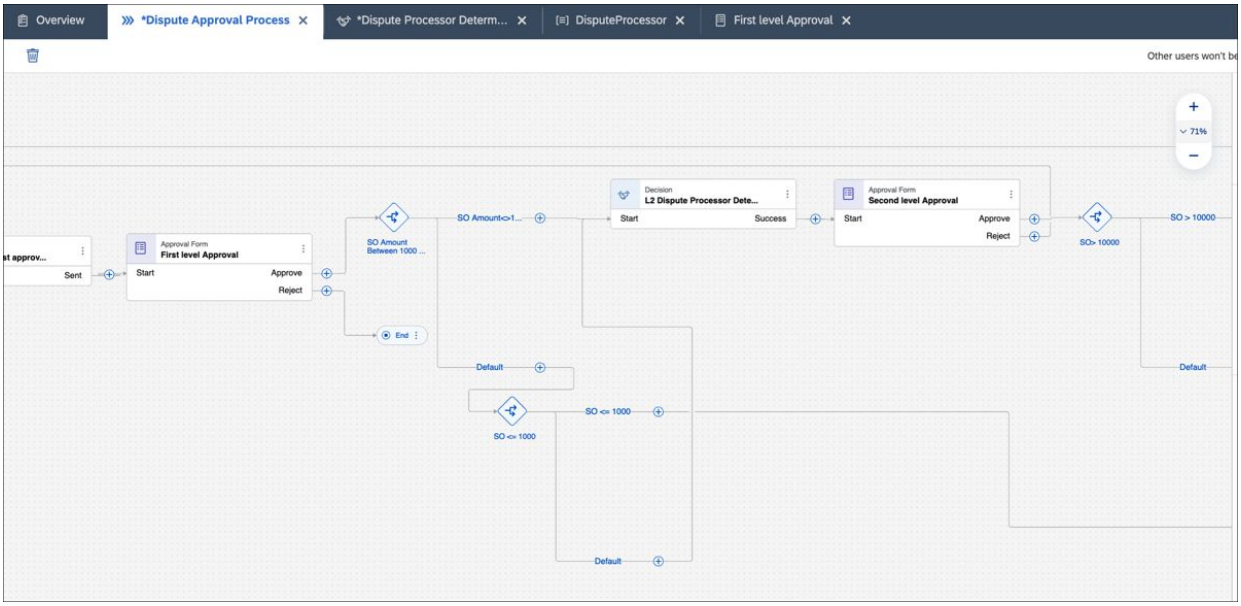


Figure 18.108 First Condition Check and Second-Level Approval Steps

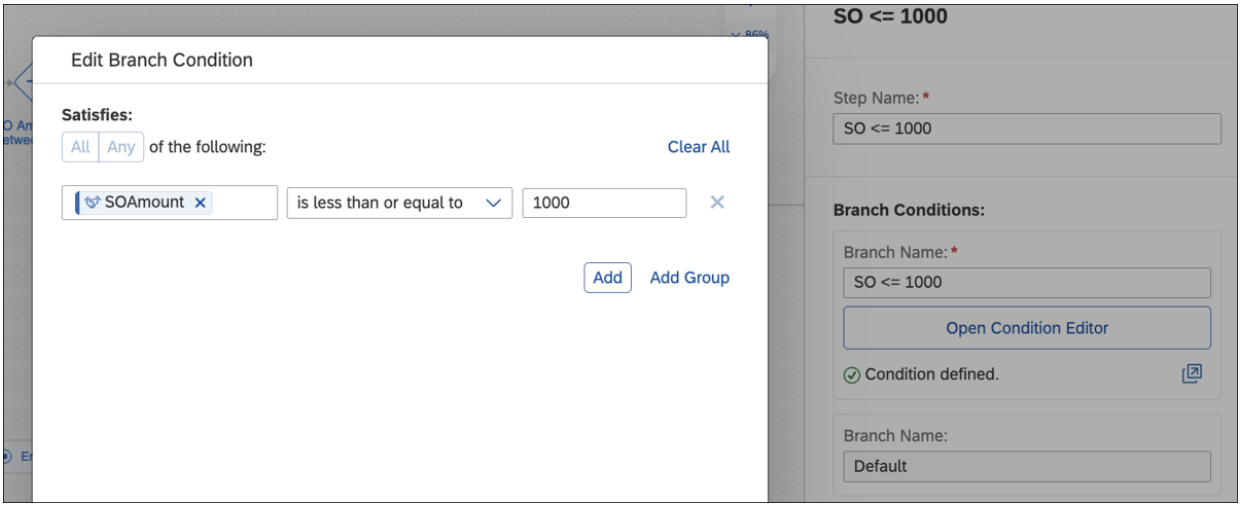


Figure 18.109 Second Condition Configuration

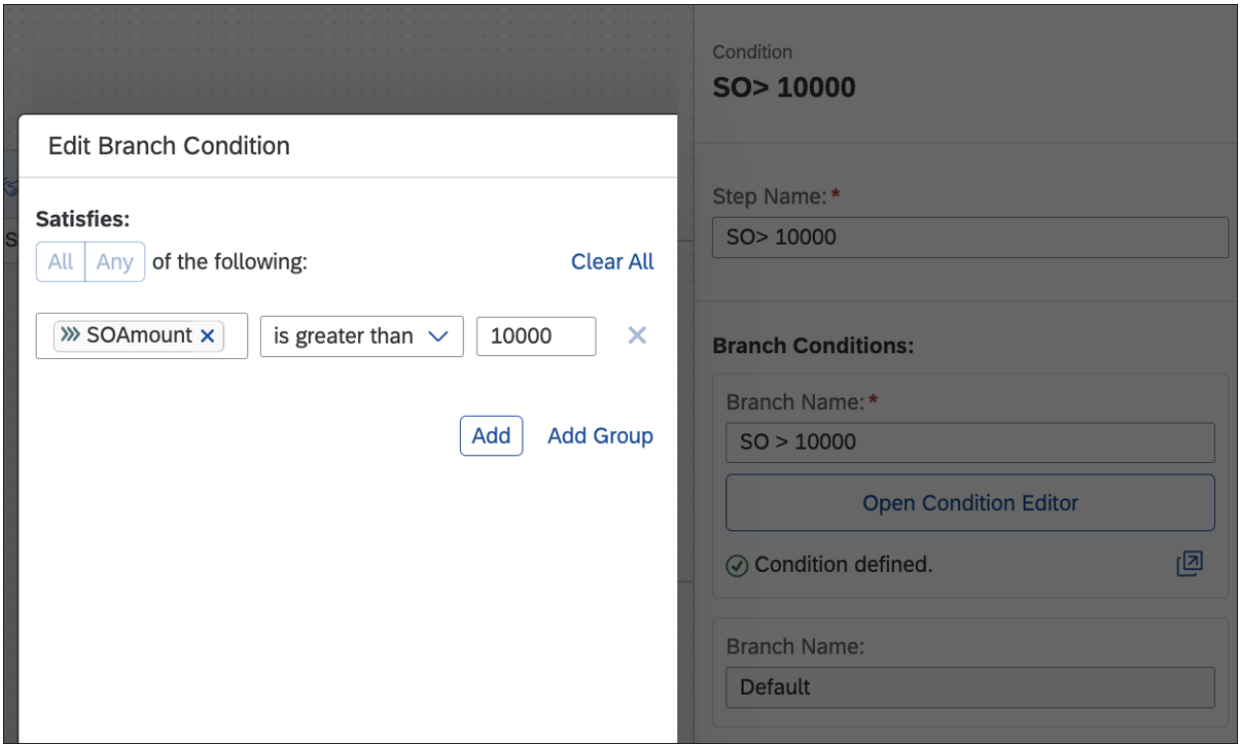


Figure 18.110 Third Condition Check

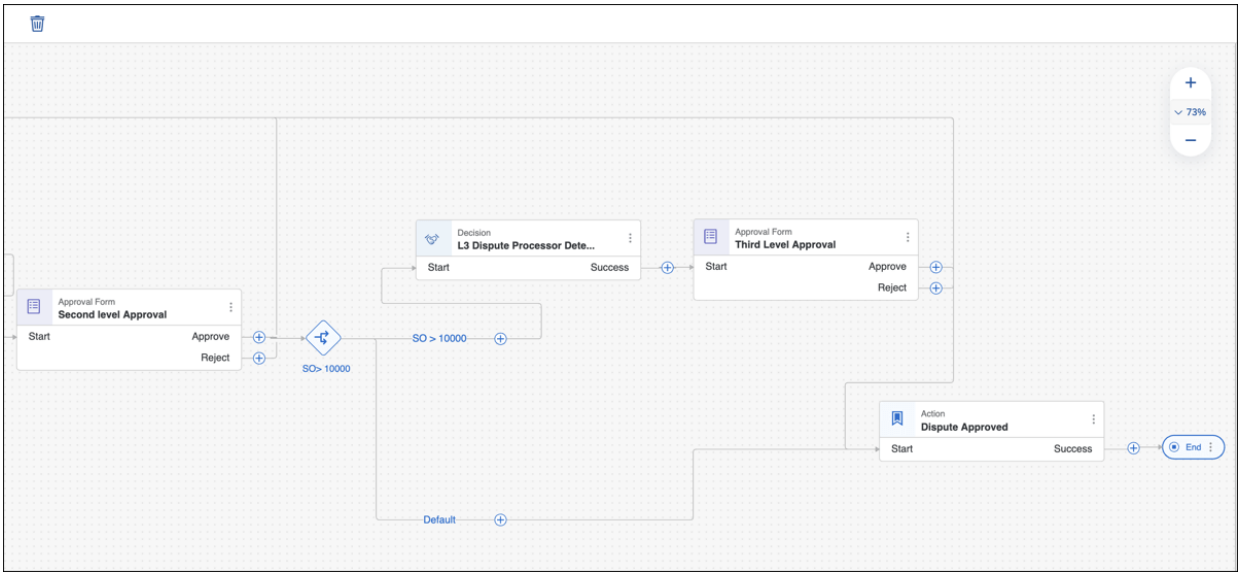


Figure 18.111 Third Condition Check and Action to Update the Dispute

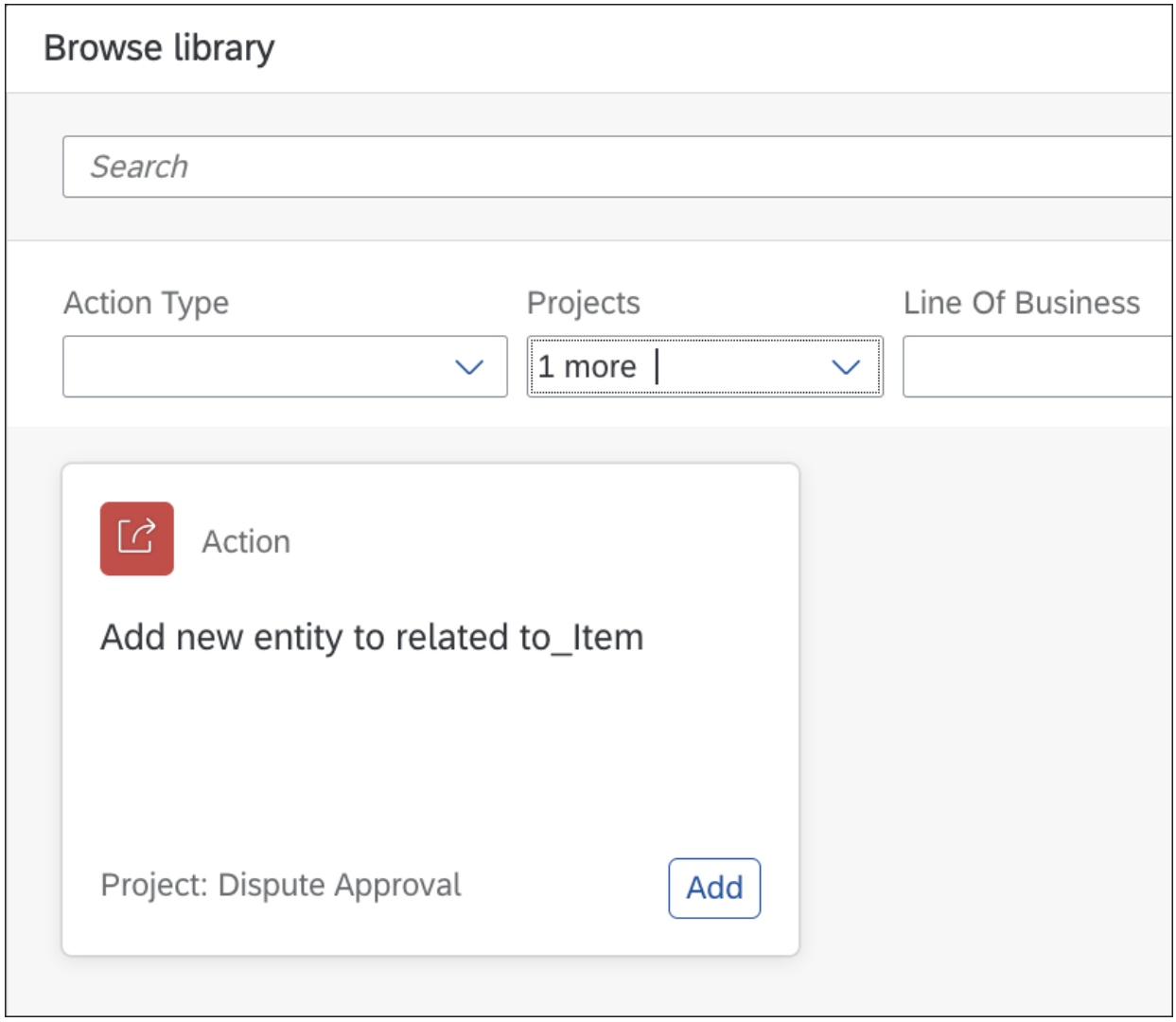


Figure 18.112 Add Dispute Update Action to the Process

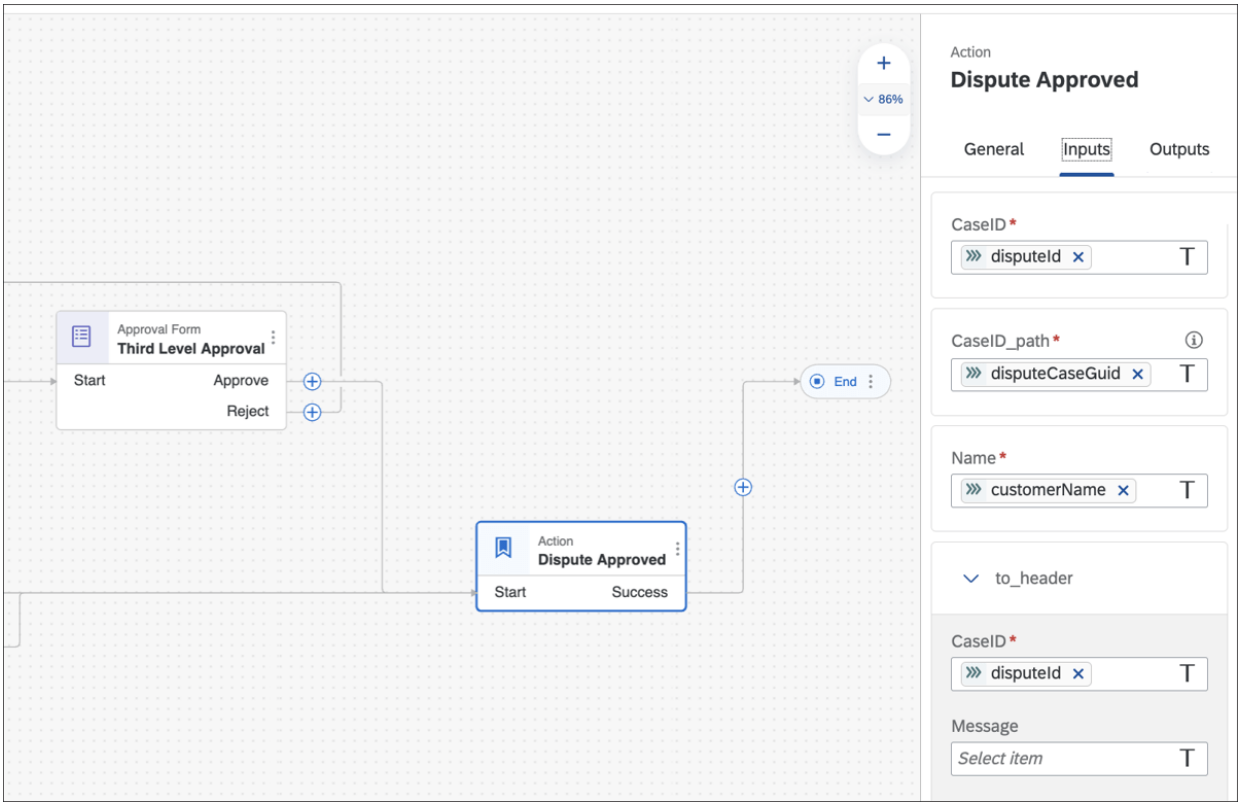


Figure 18.113 Setting the OData Parameters to Update the Dispute Case

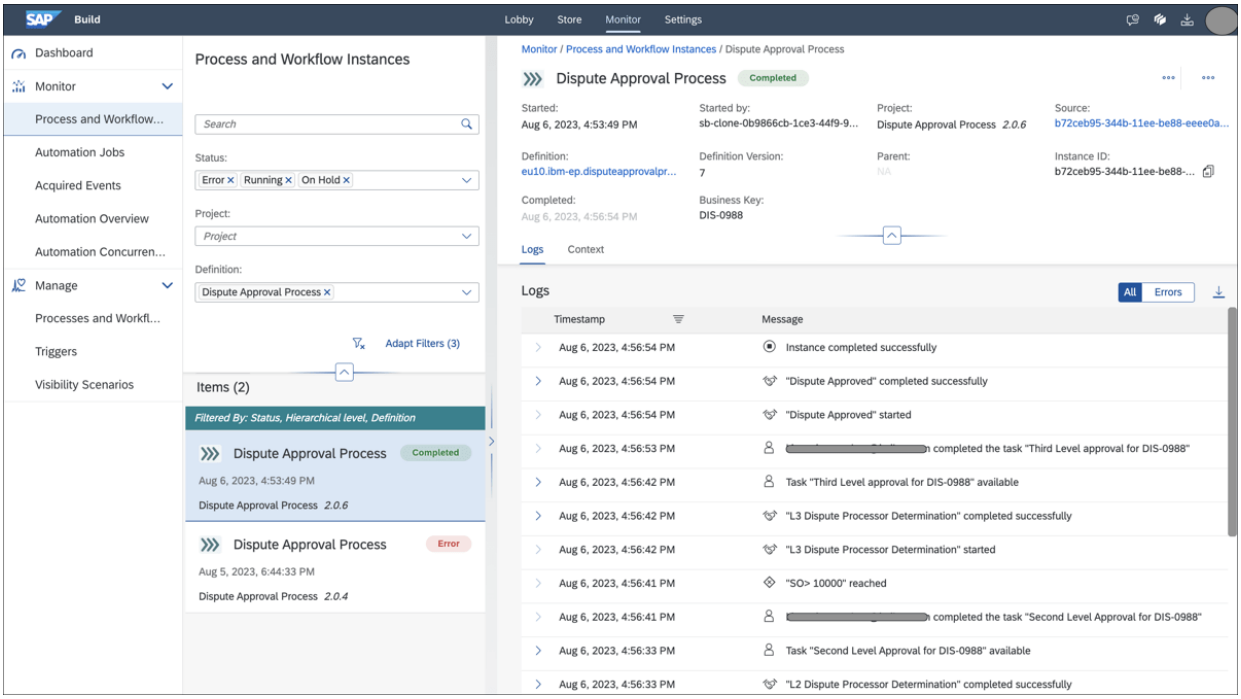


Figure 18.114 Process Log Showing Successful Execution of the Process

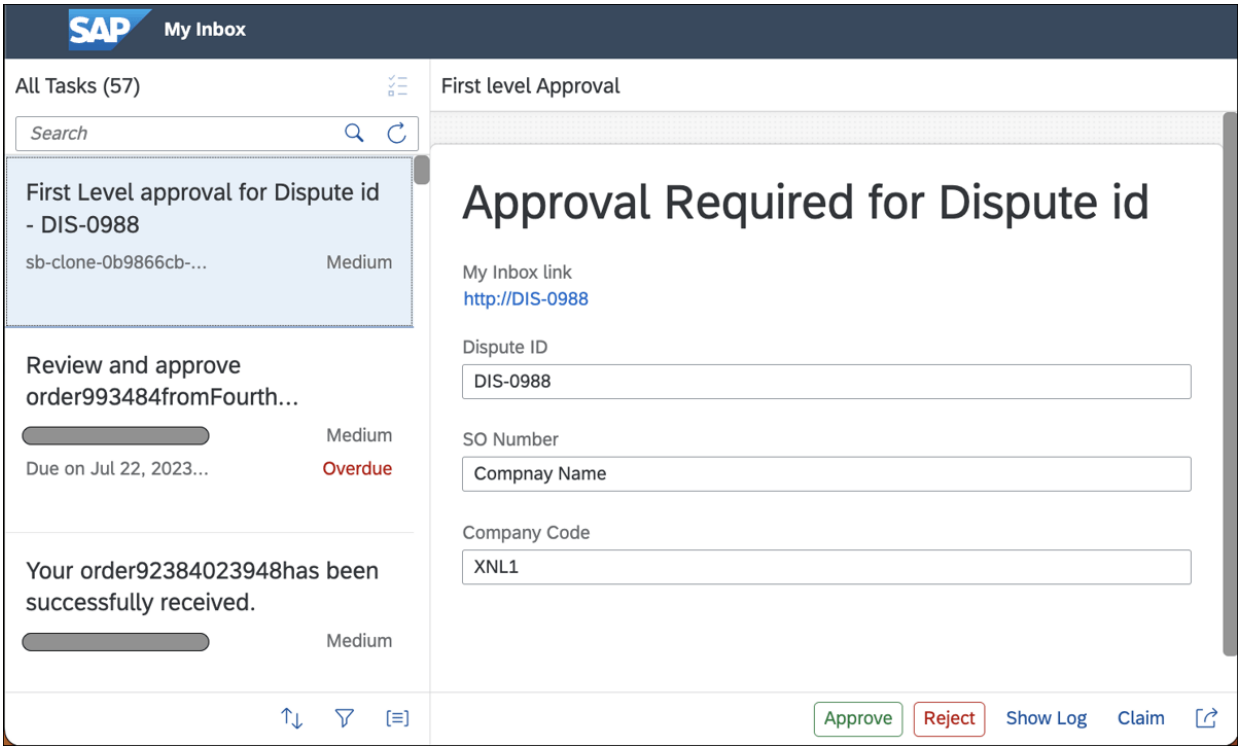
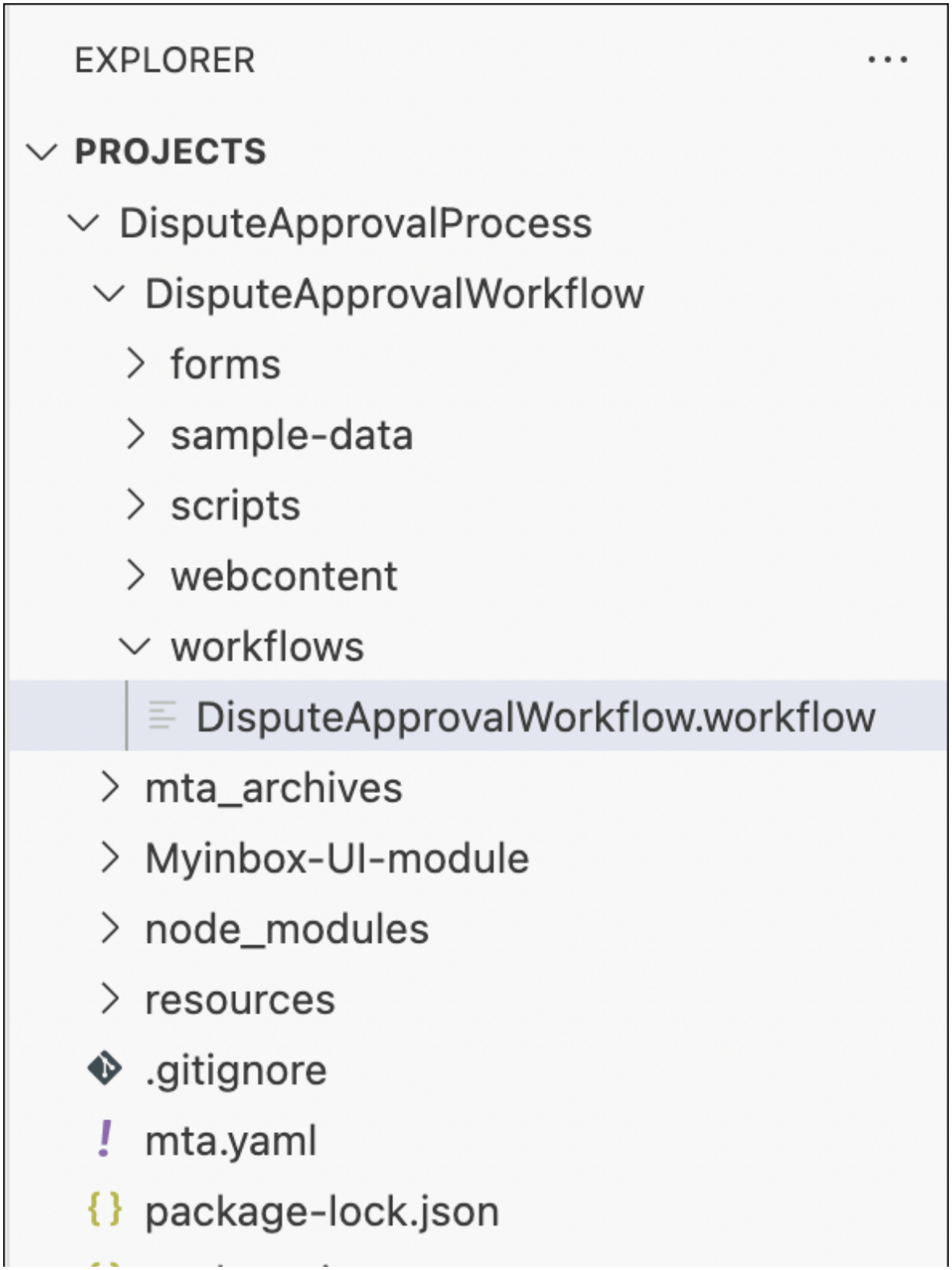


Figure 18.115 My Inbox Task


EXPLORER
PROJECTS
DisputeApprovalProcess
DisputeApprovalWorkflow
forms
sample-data
scripts
webcontent
workflows
DisputeApprovalWorkflow.workflow
mta_archives
Myinbox-UI-module
node_modules
resources
.gitignore
mta.yaml
package-lock.json

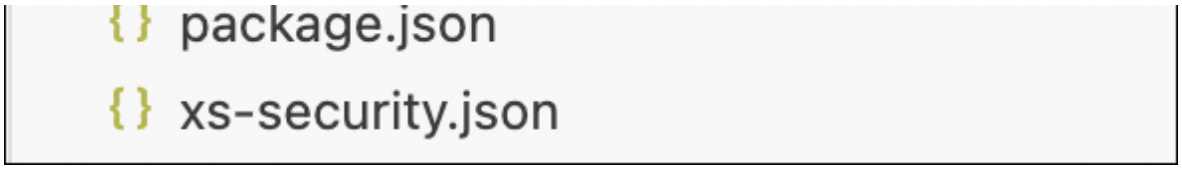


Figure 18.116 Creating a Workflow Project

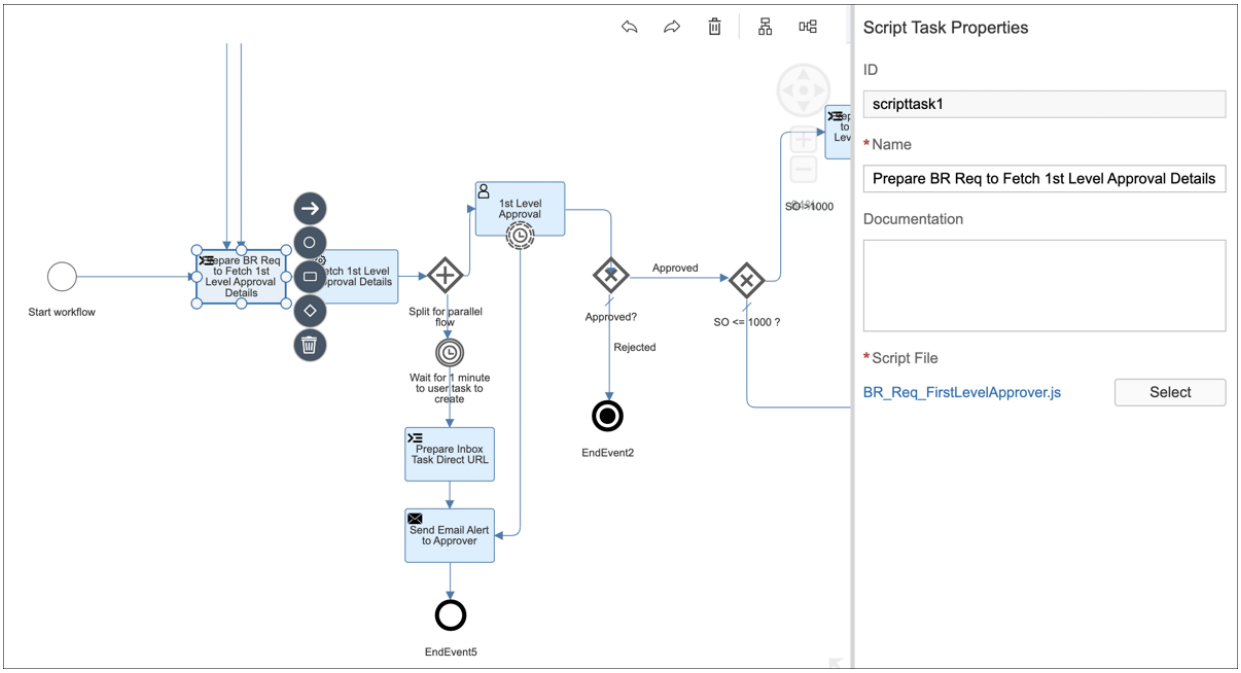


Figure 18.117 Creating a Script Task

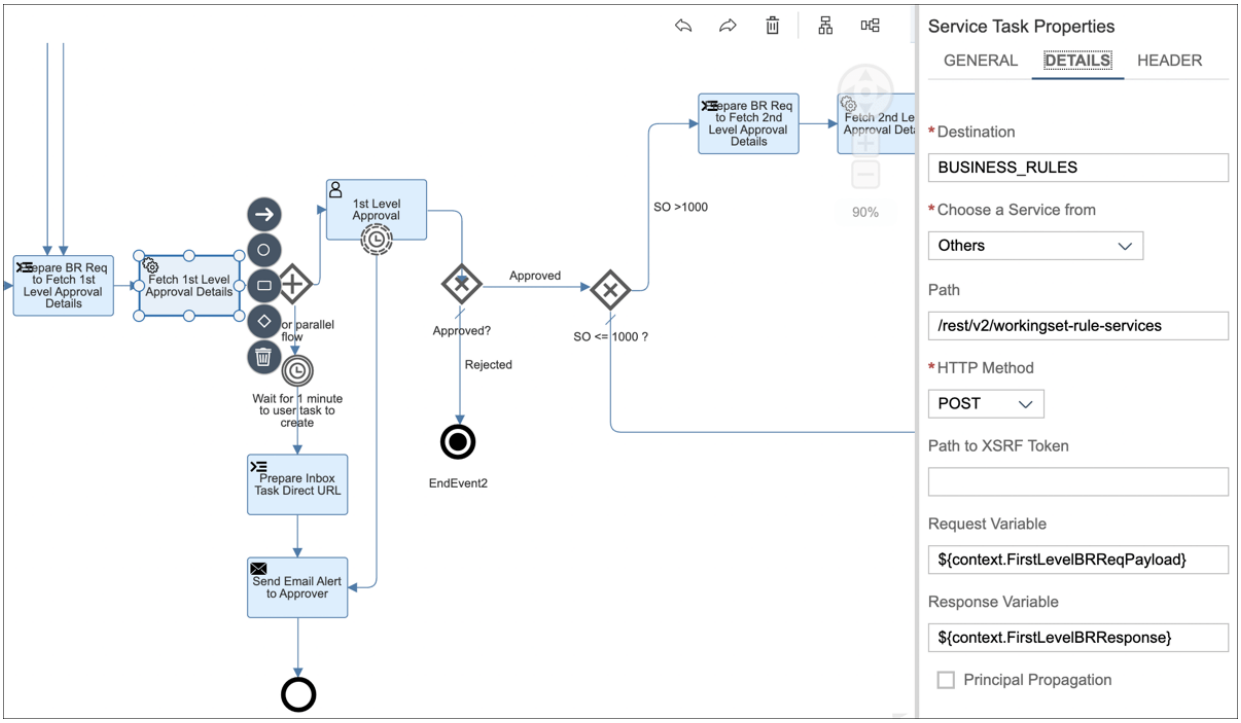
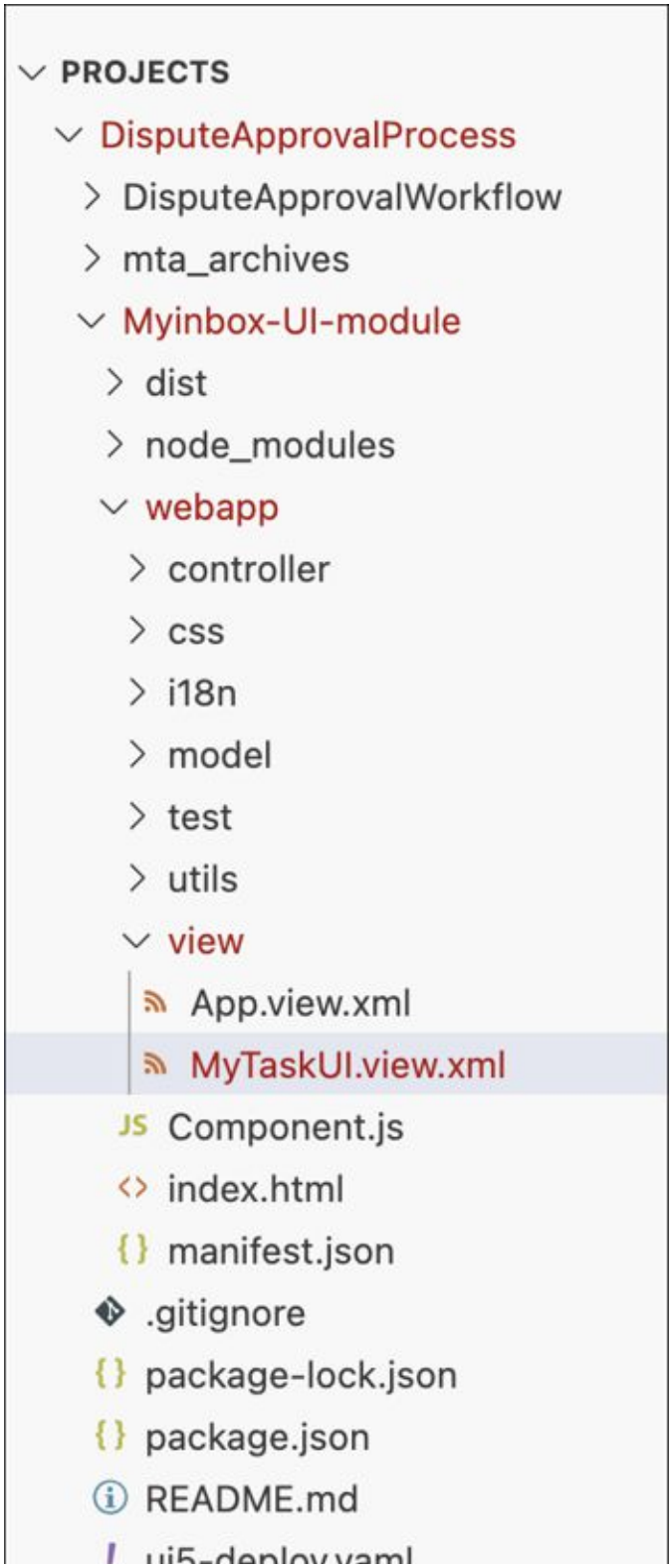


Figure 18.118 Creating a Service Task to Fetch the Business Rule


PROJECTS
DisputeApprovalProcess
DisputeApprovalWorkflow
mta_archives
Myinbox-UI-module
dist
node_modules
webapp
controller
css
i18n
model
test
utils
view
App.view.xml
MyTaskUI.view.xml
Component.js
index.html
manifest.json
.gitignore
package-lock.json
package.json
README.md

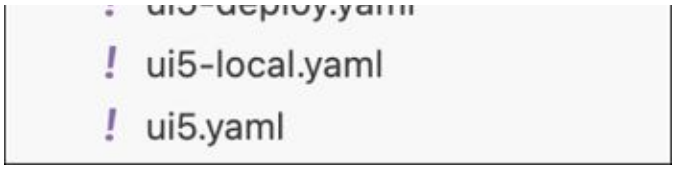


Figure 18.119 UI Module for the User Task

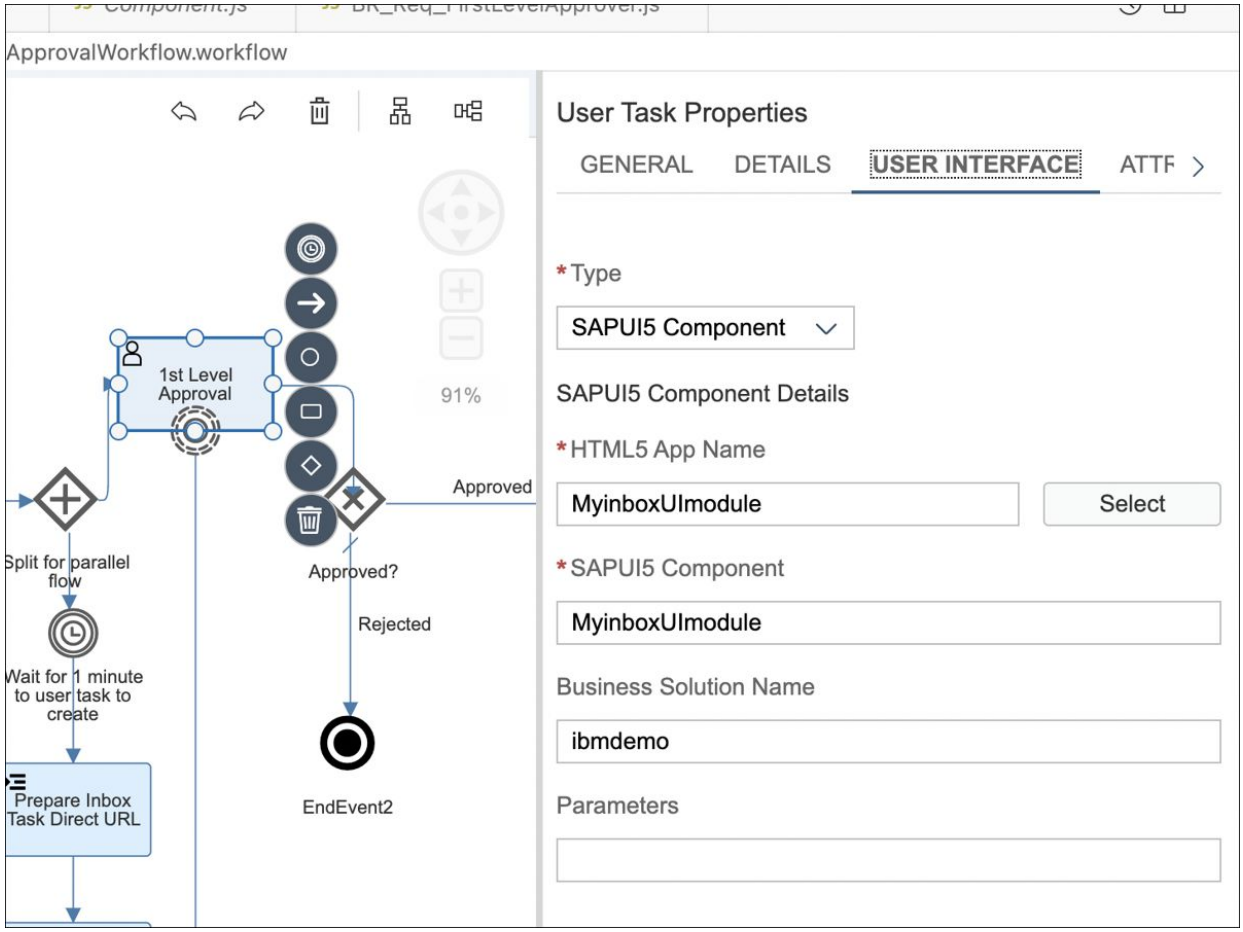


Figure 18.120 Adding the UI Module to the User Task

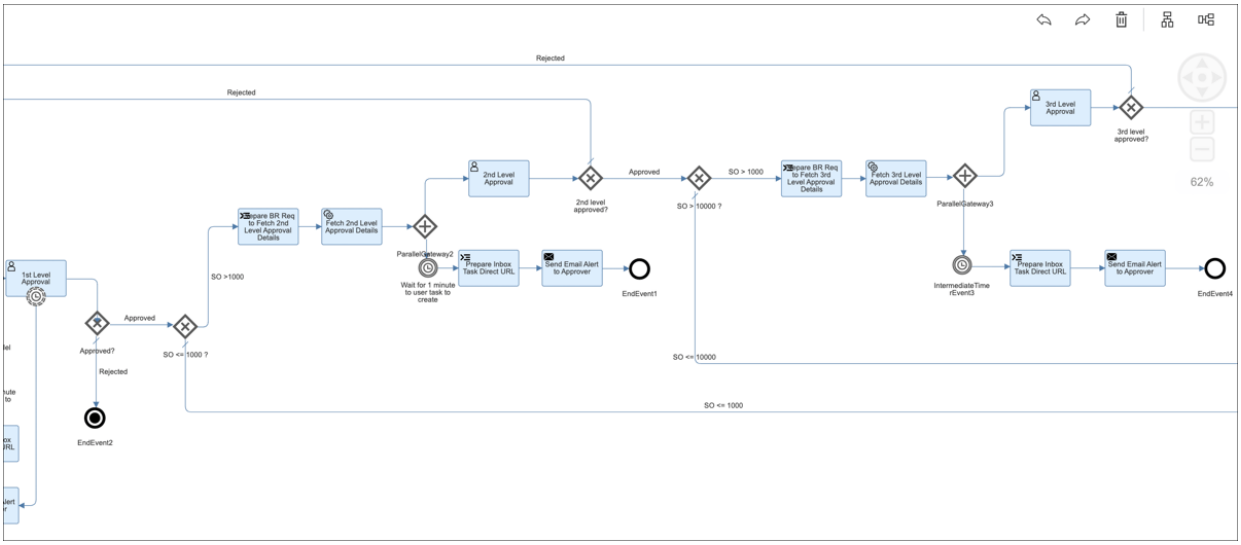


Figure 18.121 Remaining Flow (Part 1)

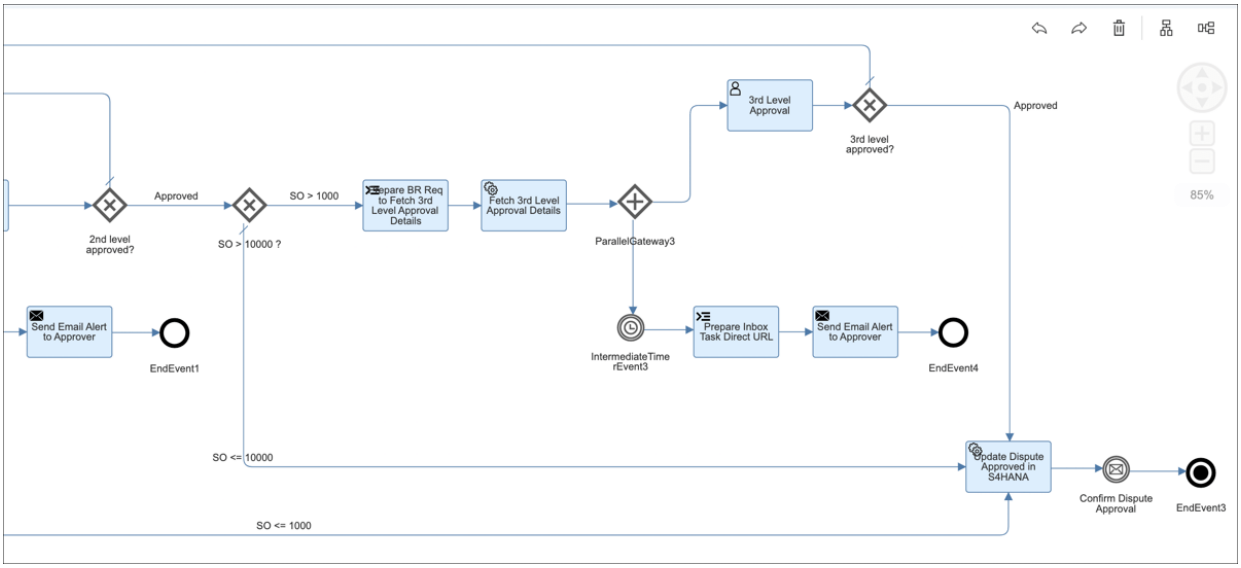


Figure 18.122 Remaining Flow (Part 2)

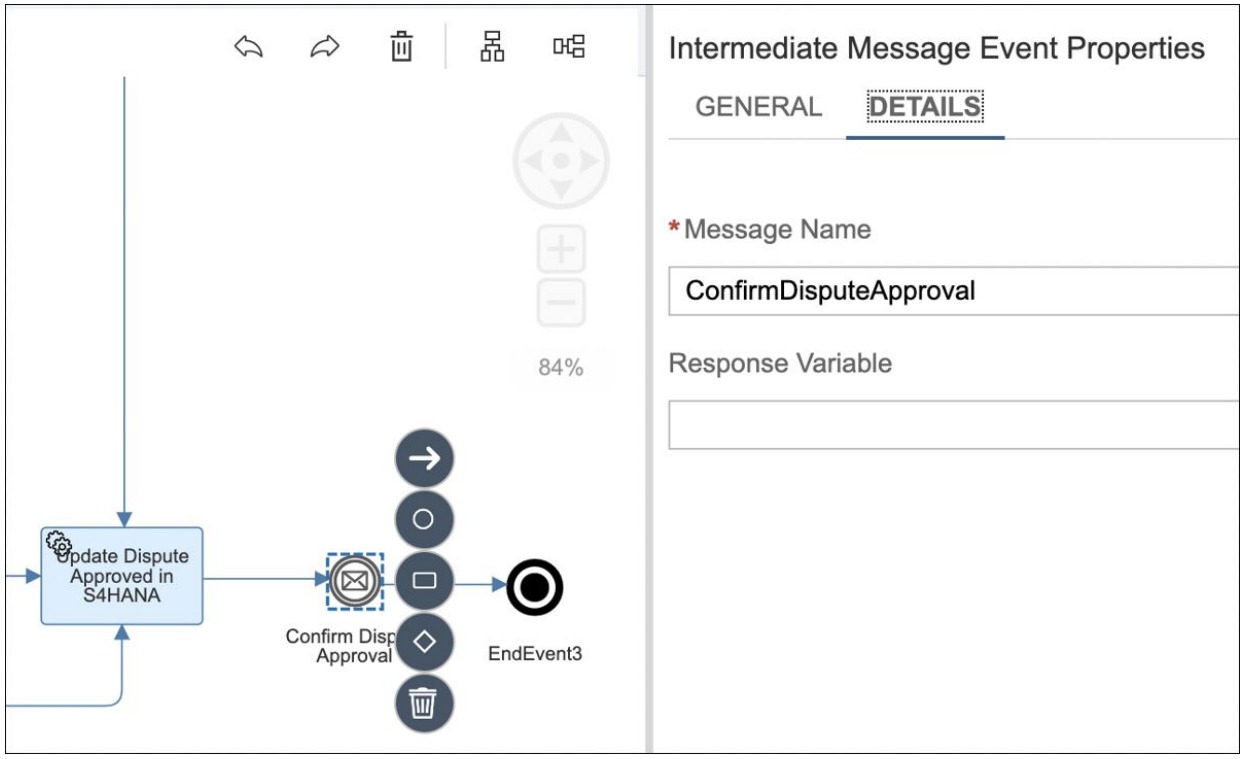


Figure 18.123 Intermediate Message Event That Waits for a Business Event

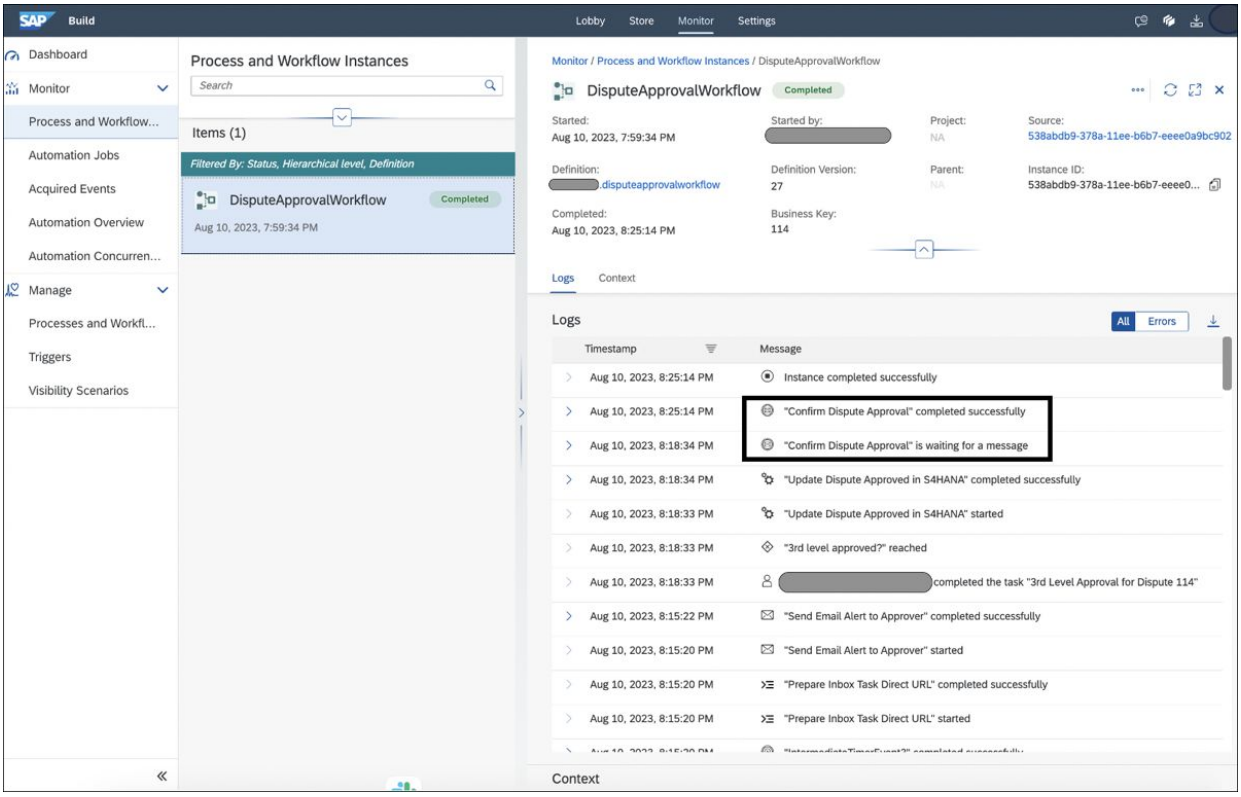


Figure 18.124 Execution Logs of the Workflow

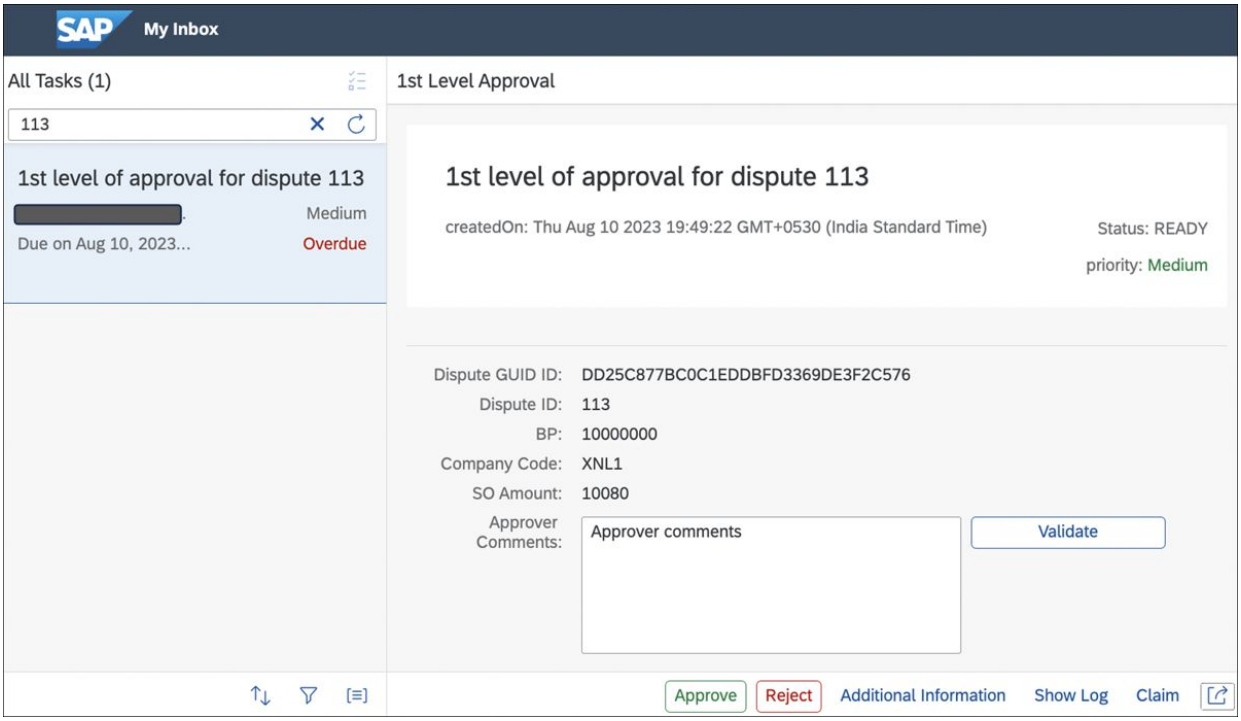


Figure 18.125 Custom My Inbox Screen

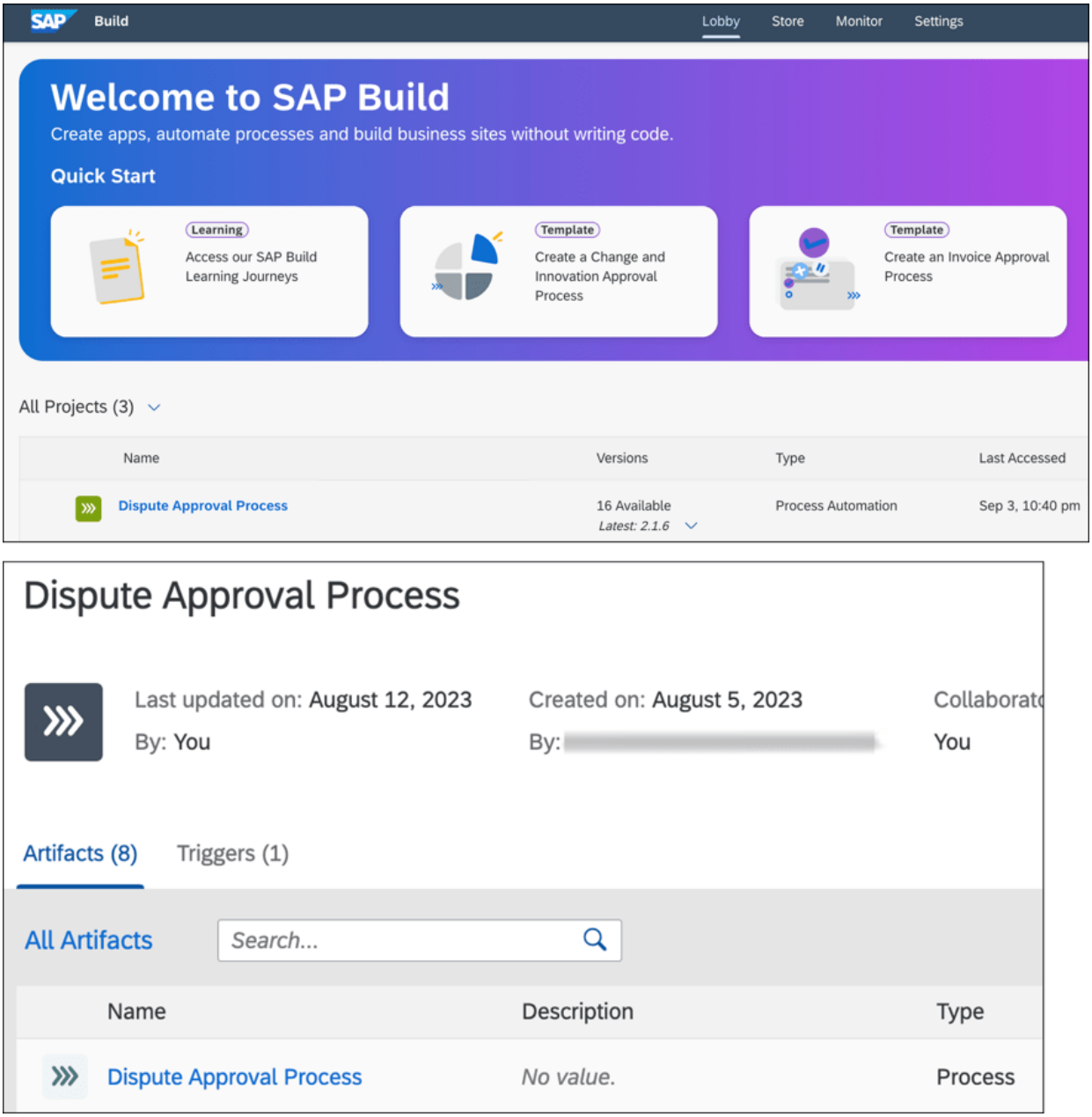


Figure 19.1 Opening the Process in Process Builder

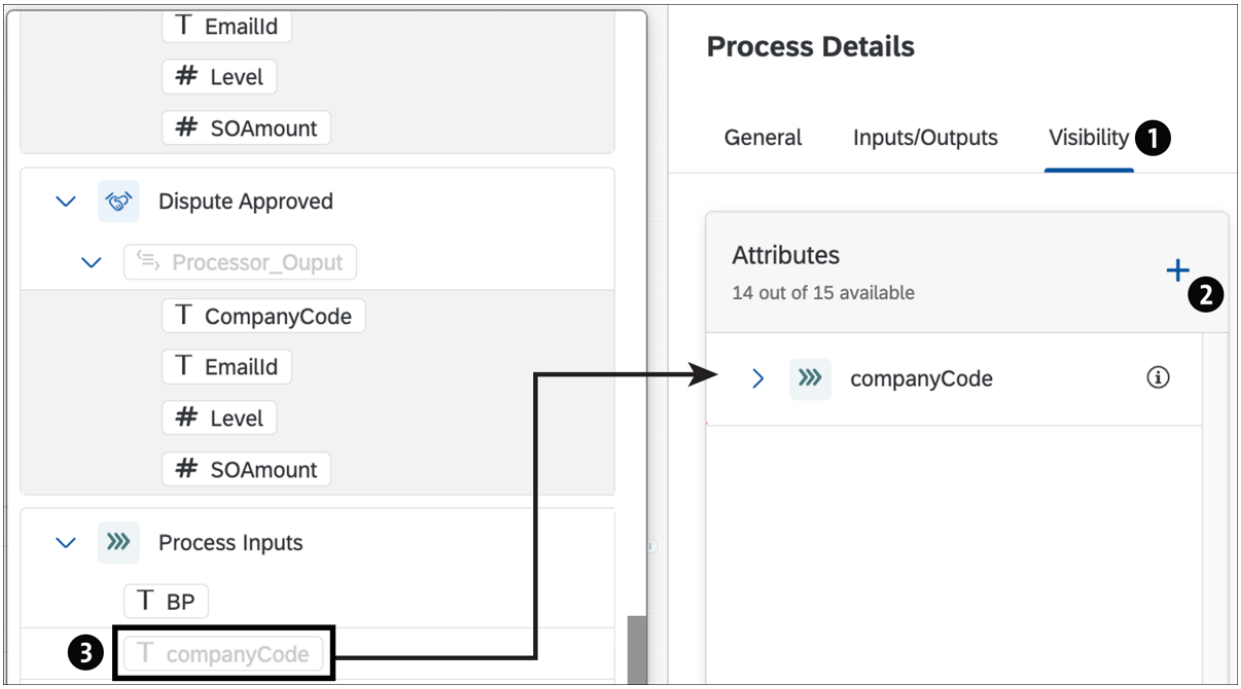


Figure 19.2 Adding Attributes to the Visibility Scenario

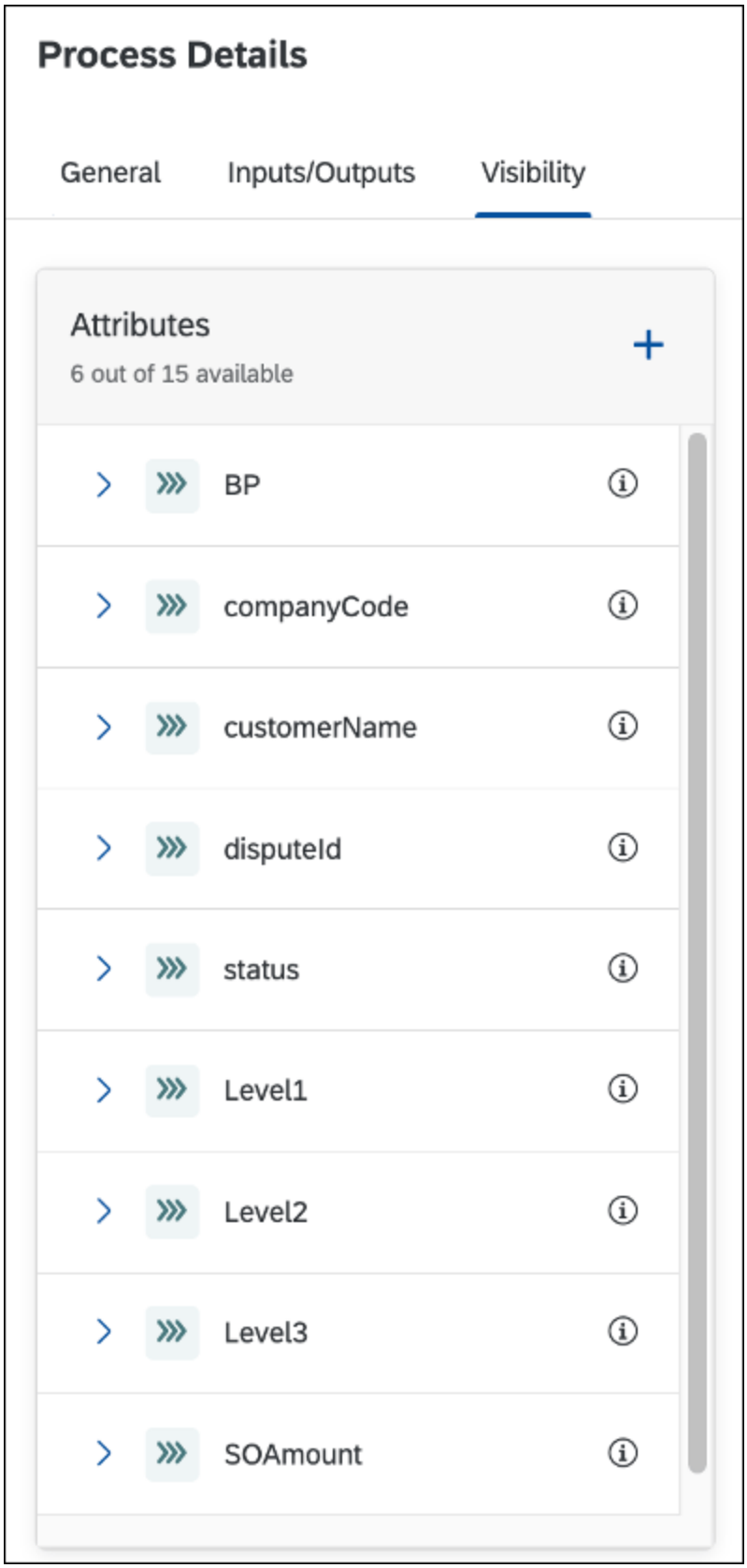

Process Details
General
Inputs/Outputs
Visibility
Attributes
6 out of 15 available
BP
companyCode
customerName
disputeId
status
Level1
Level2
Level3
SOAmount

Figure 19.3 Required Attributes Added

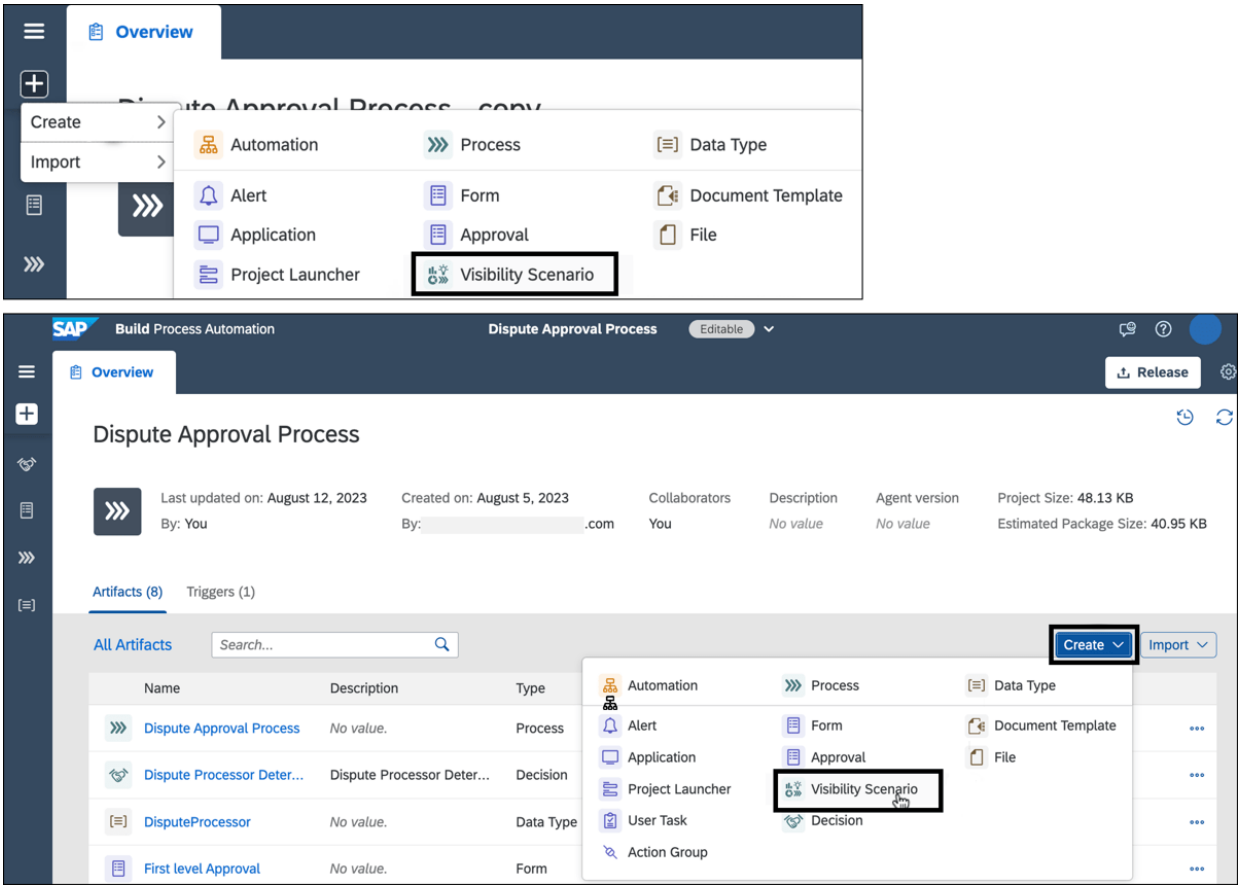


Figure 19.4 Creating the Visibility Scenario

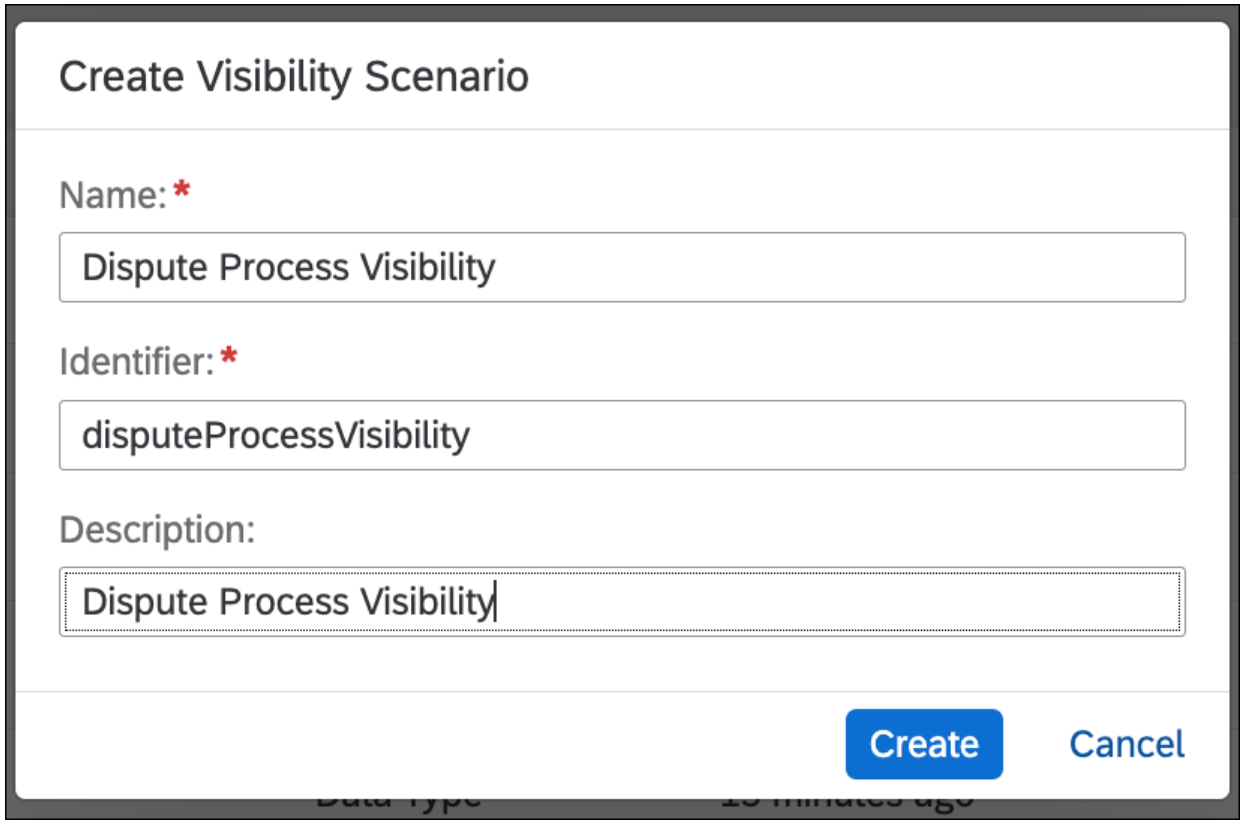


Figure 19.5 Creating the Visibility Scenario

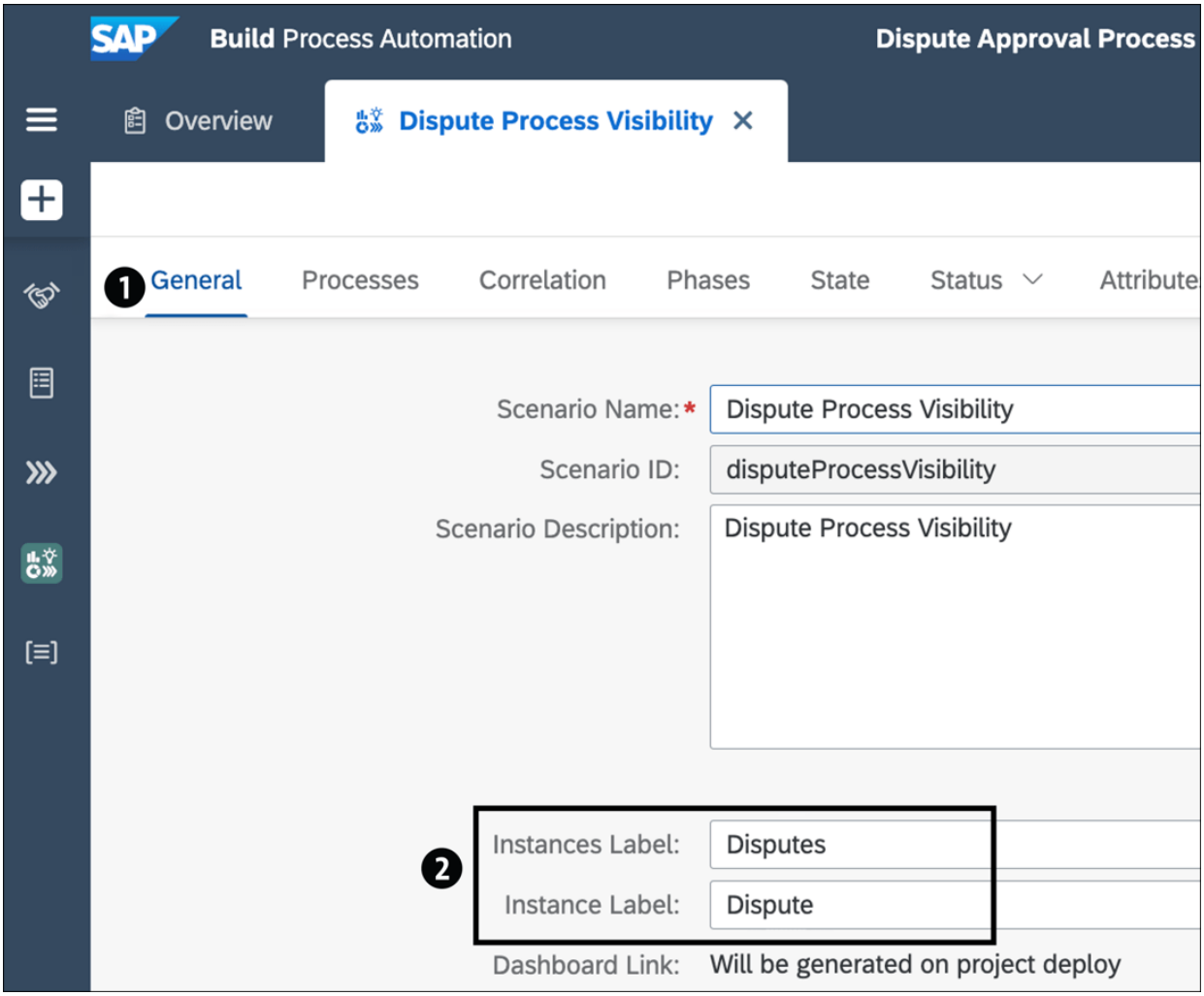


Figure 19.6 Configuring the Visibility Scenario: General Tab

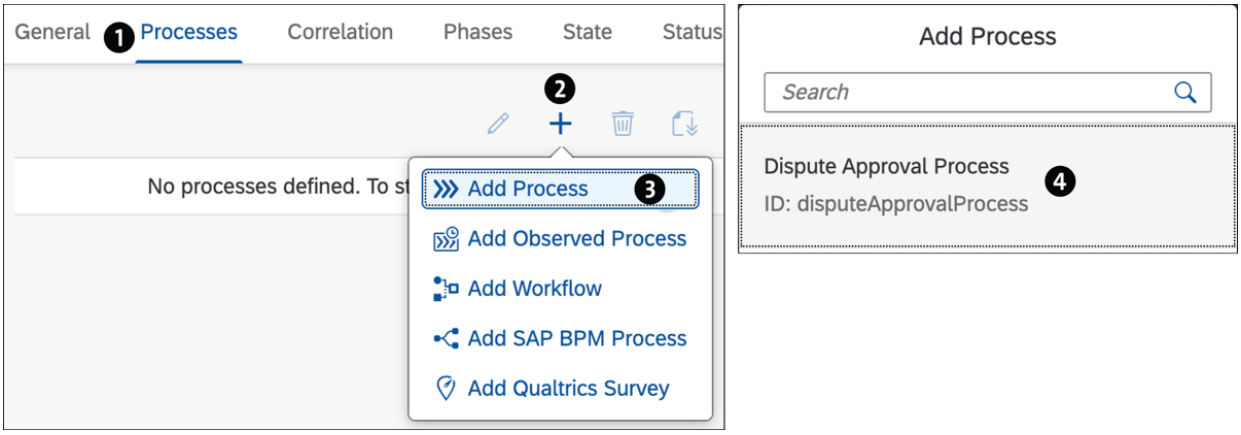


Figure 19.7 Adding the Process to the Visibility Scenario

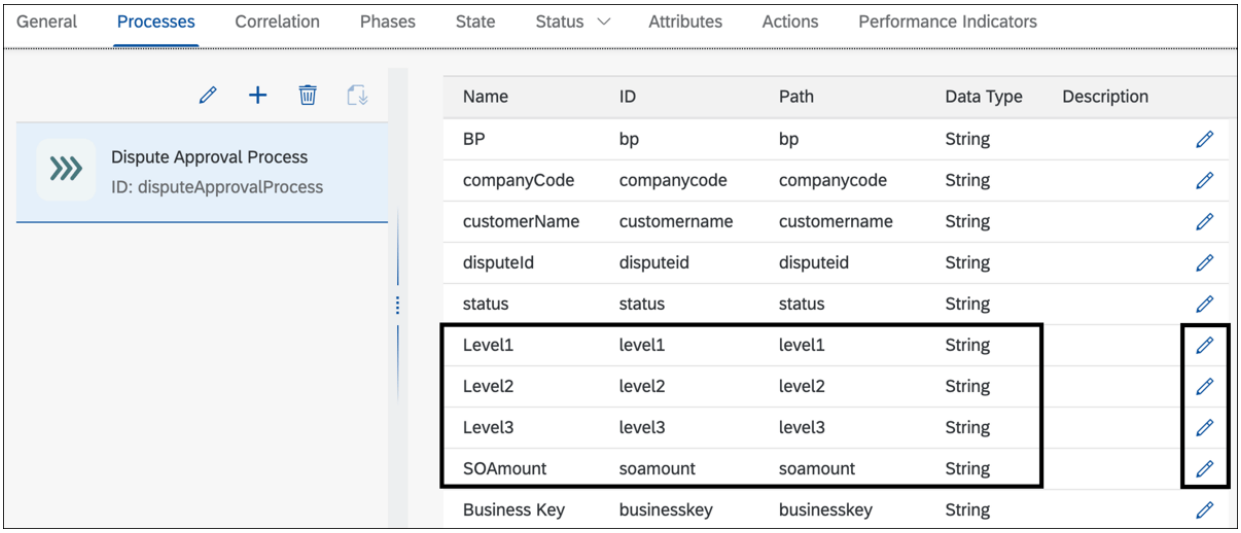


Figure 19.8 Events and Context Attributes Available in the Visibility Scenario

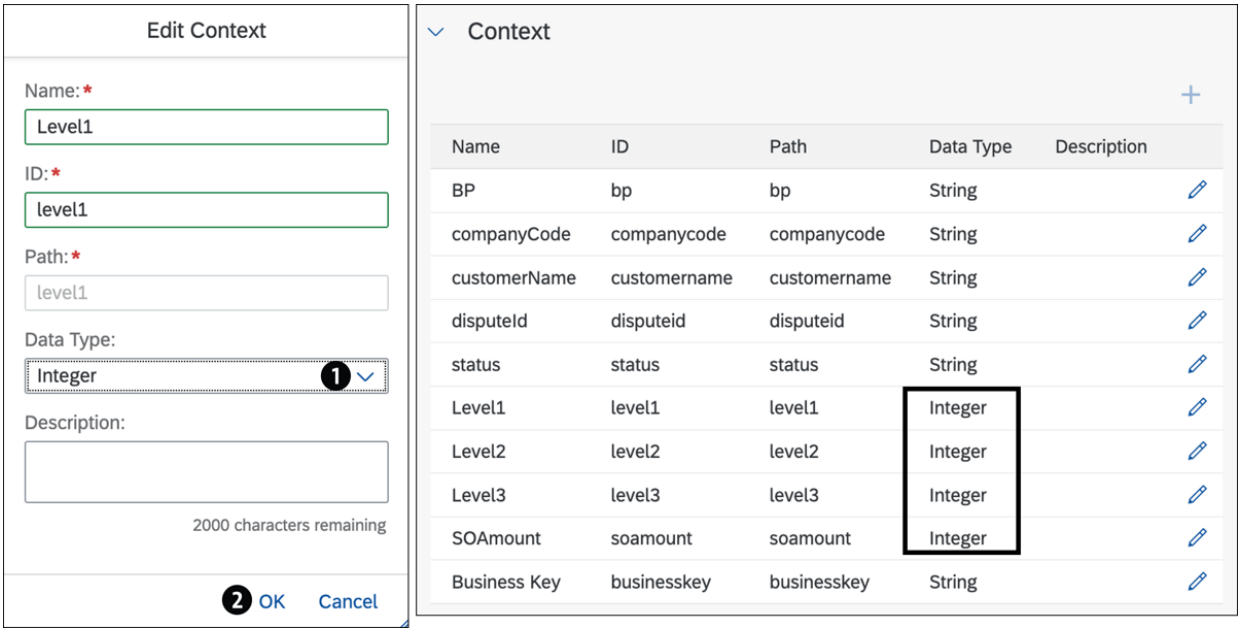


Figure 19.9 Correcting the Data Types of Context Attributes

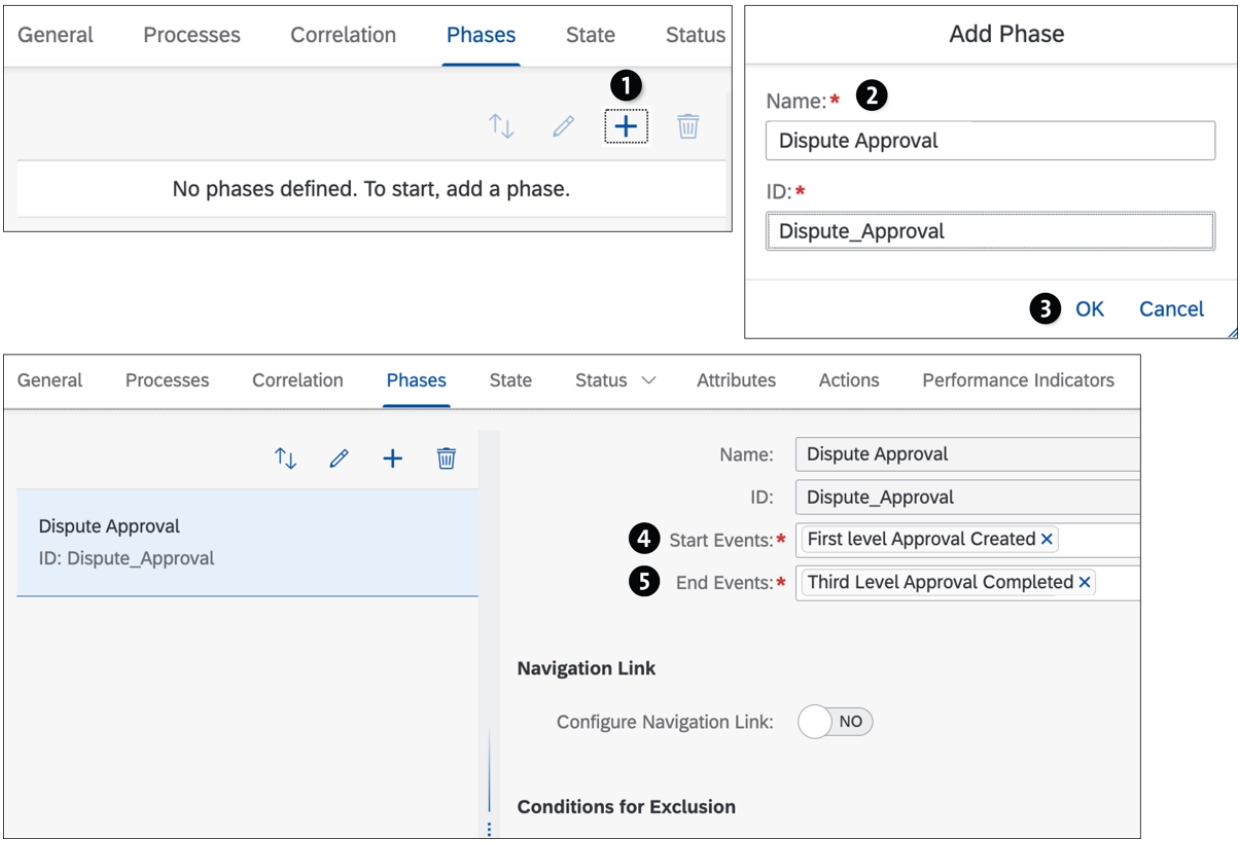


Figure 19.10 Configuring Process Phases

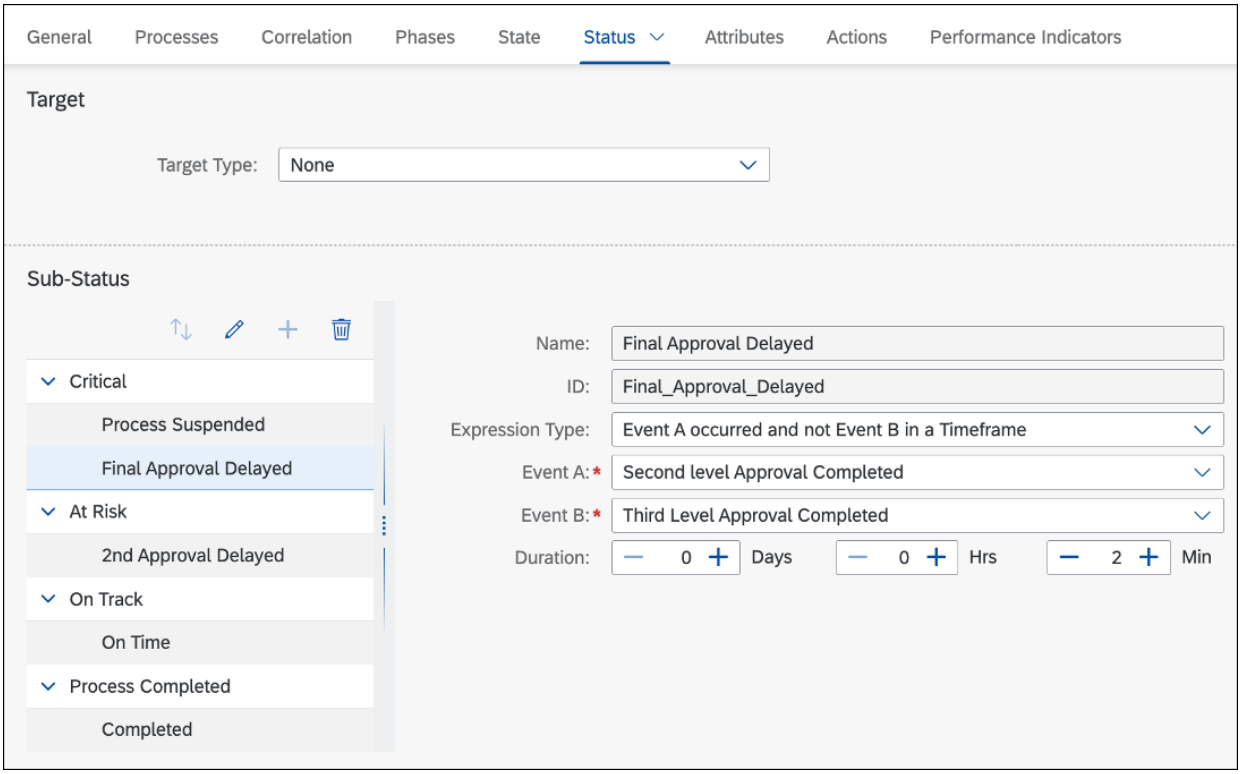


Figure 19.11 Configuring Process Phases

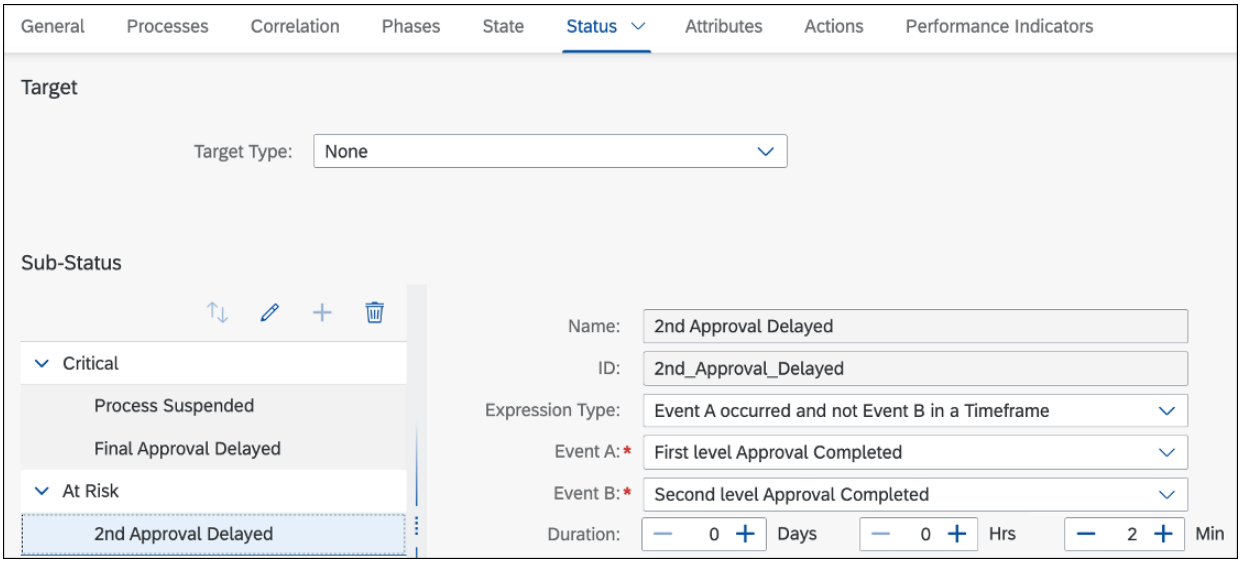


Figure 19.12 Defining More Sub-Statuses

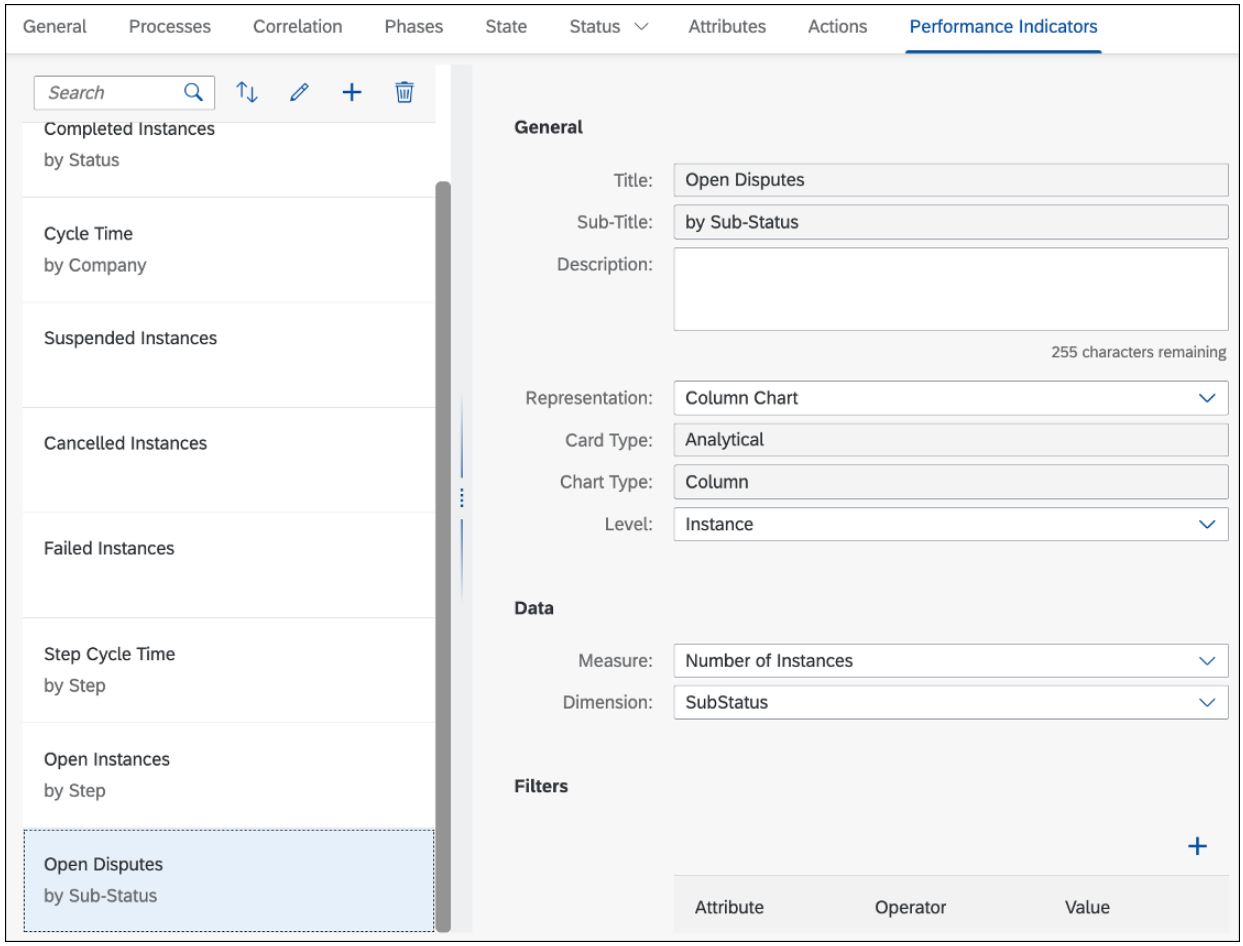


Figure 19.13 Configure Custom Performance Indicators

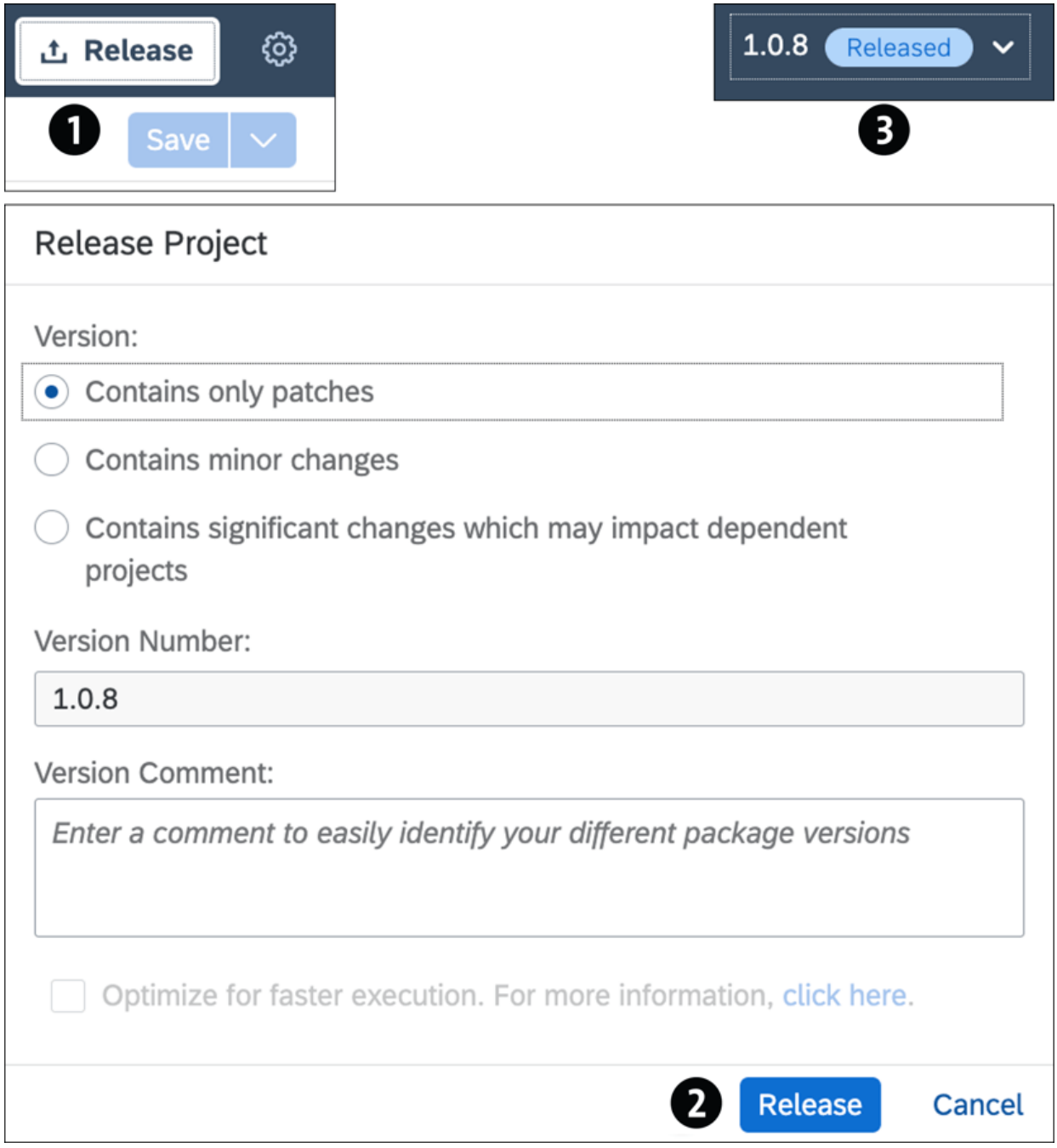


Figure 19.14 Releasing the SAP Build Process Automation Project

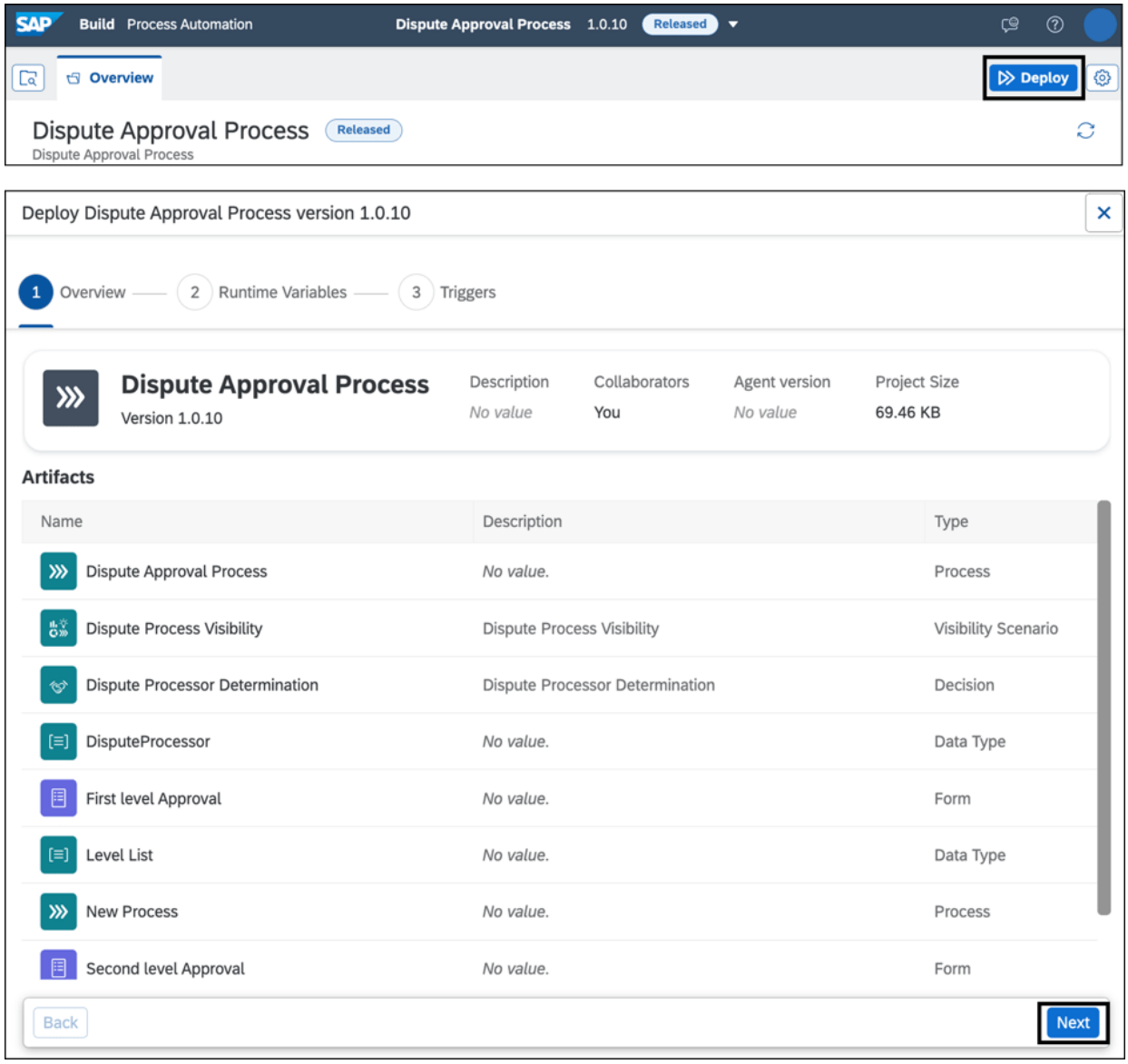


Figure 19.15 Deploy Project: Step 1 and Step 2

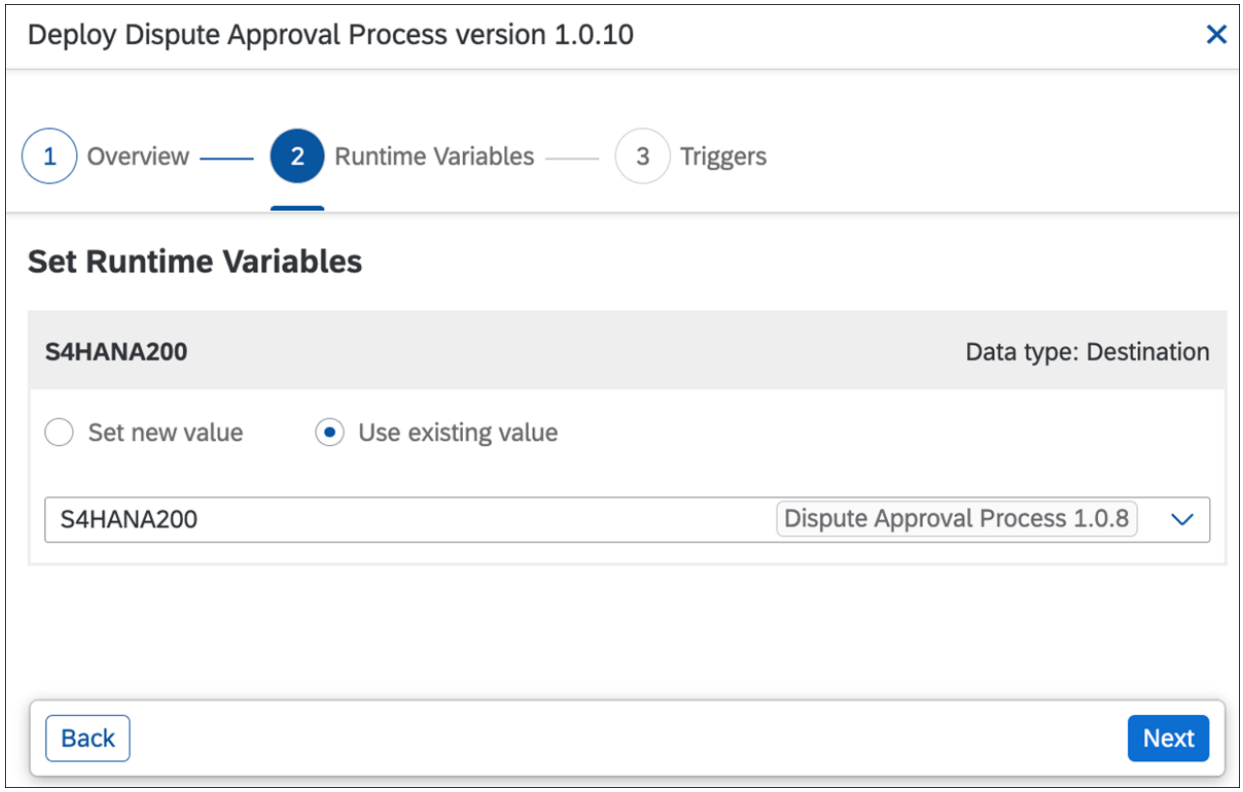


Figure 19.16 Deploy Project: Step 3

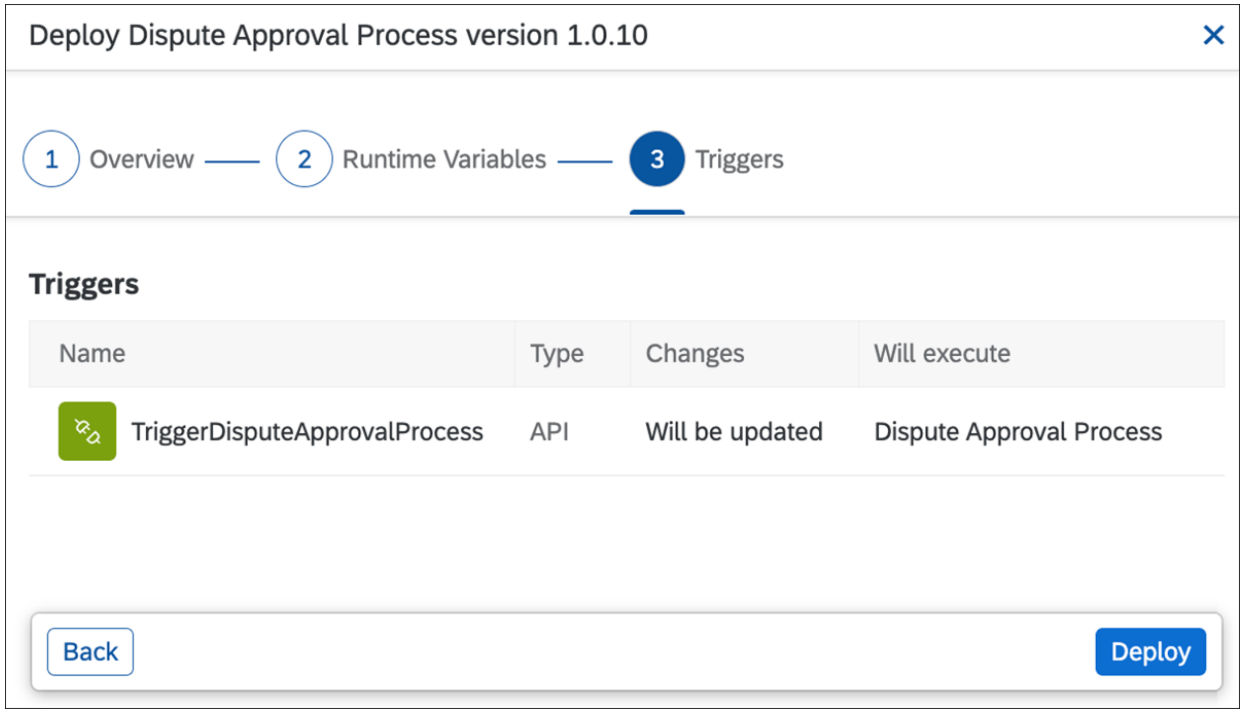


Figure 19.17 Deploy Project: The Final Step

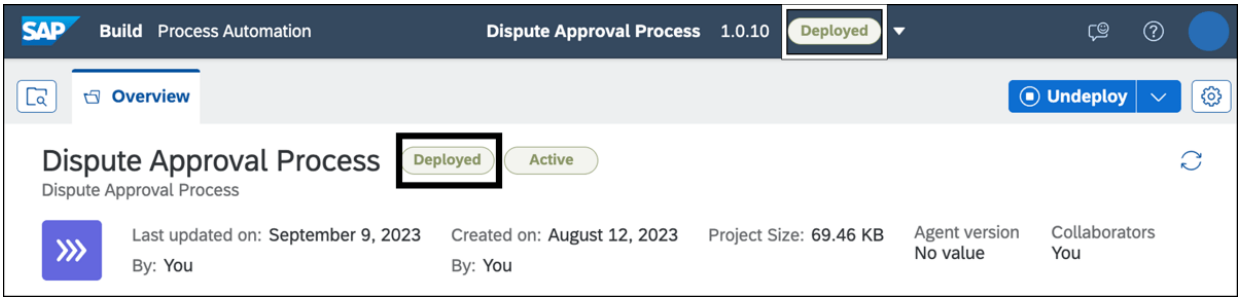


Figure 19.18 Project Status Post: Successful Deployment

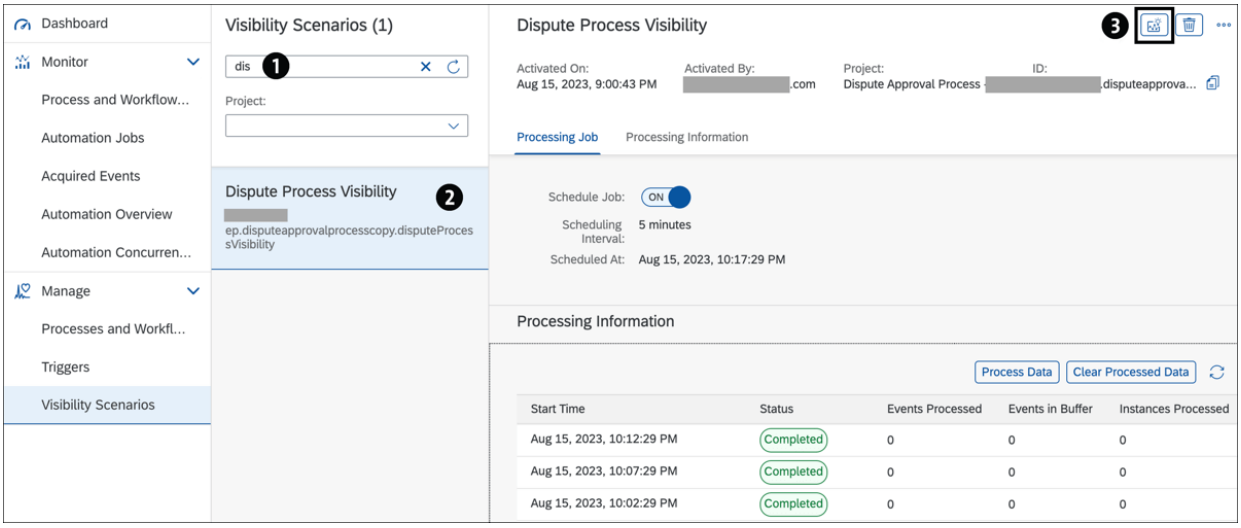


Figure 19.19 Launching the Visibility Dashboard: Steps 1 and 2

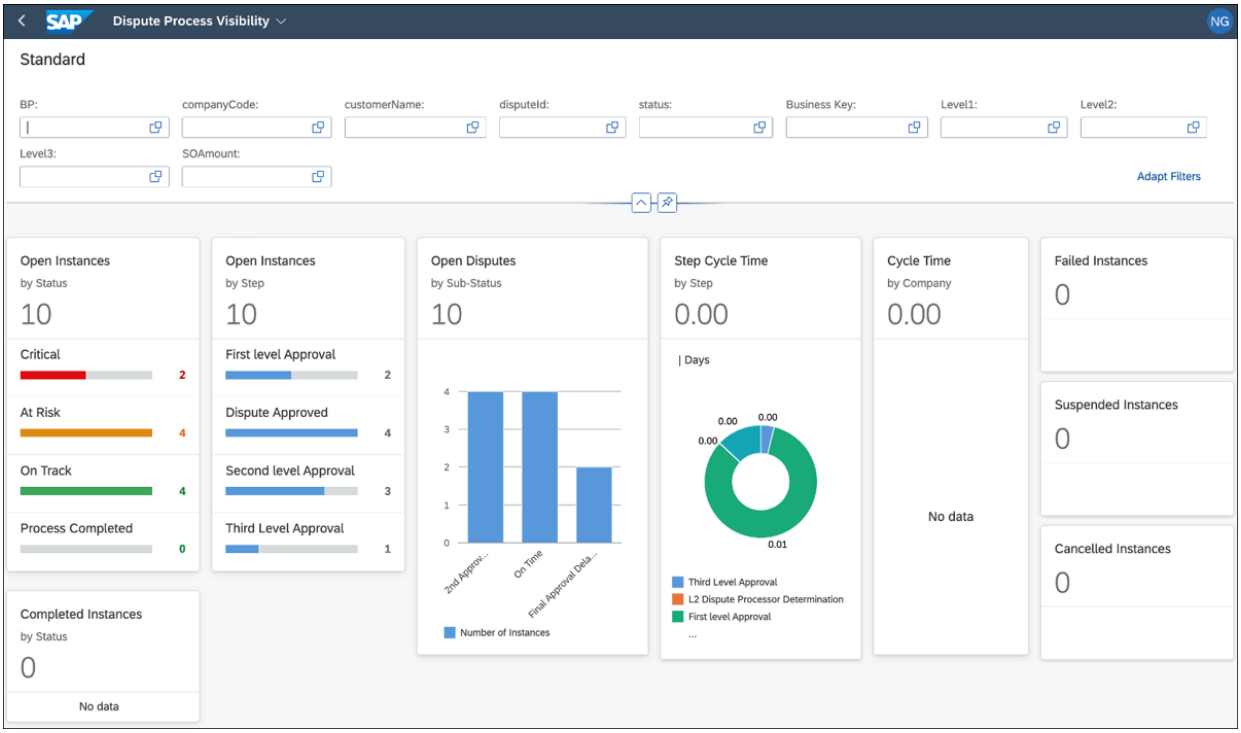


Figure 19.20 Process Visibility Scenario: Launched

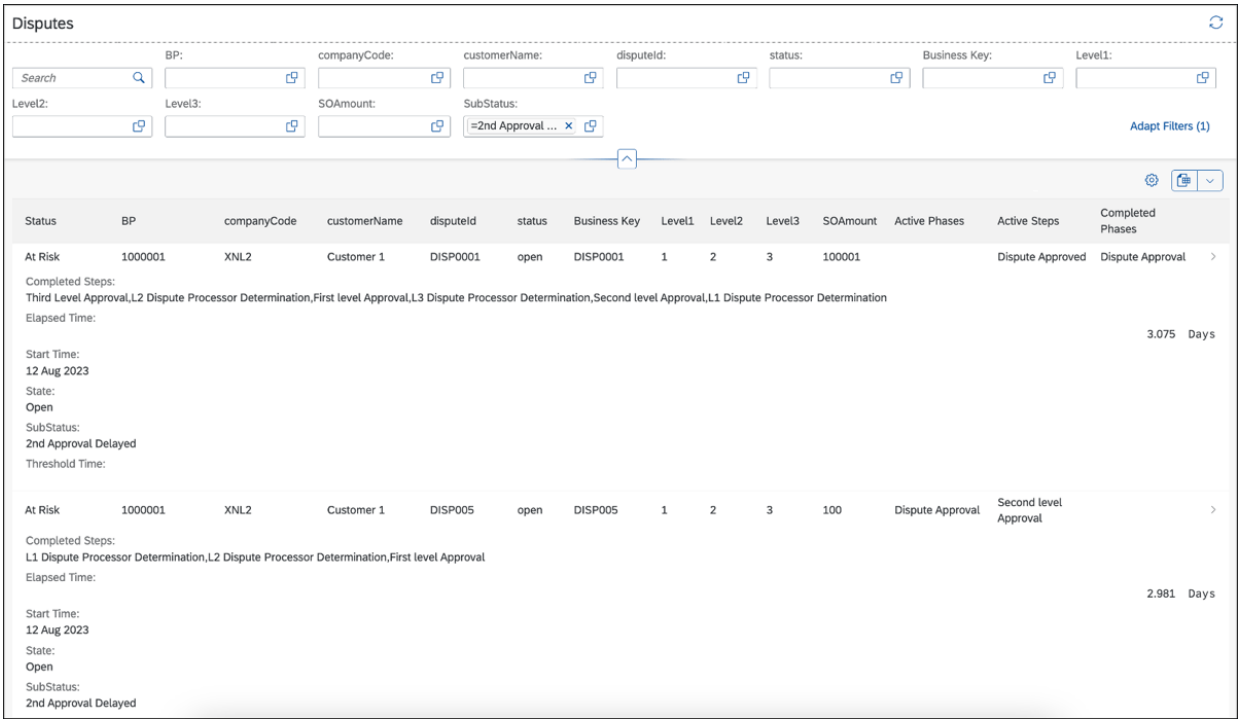


Figure 19.21 First-Level Drilldown Feature in the Process Visibility Dashboard

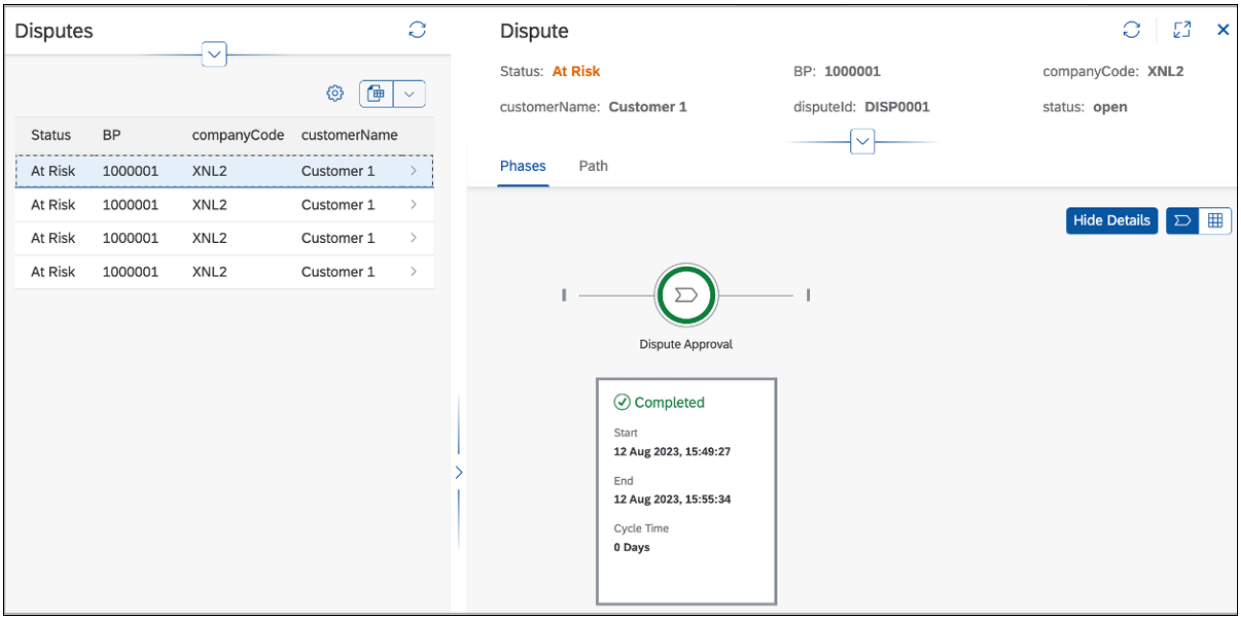


Figure 19.22 Second-level Drilldown Feature in the Process Visibility Dashboard

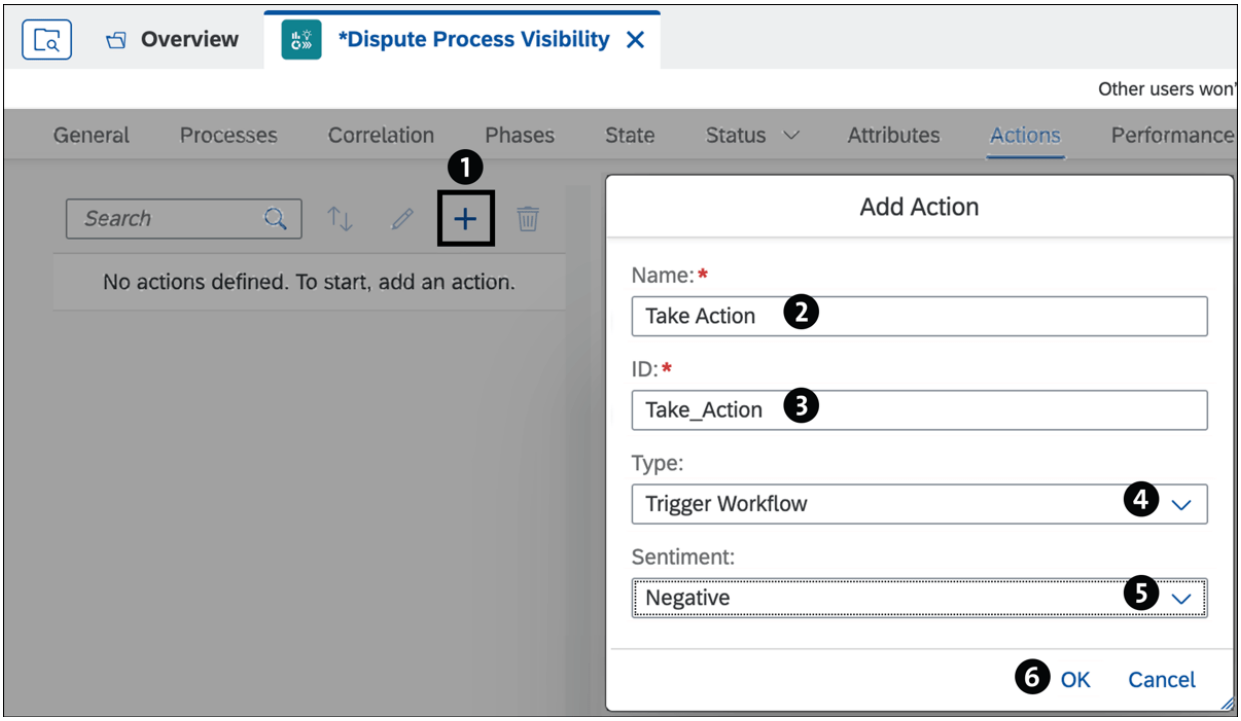


Figure 19.23 Adding Action into the Visibility Scenario

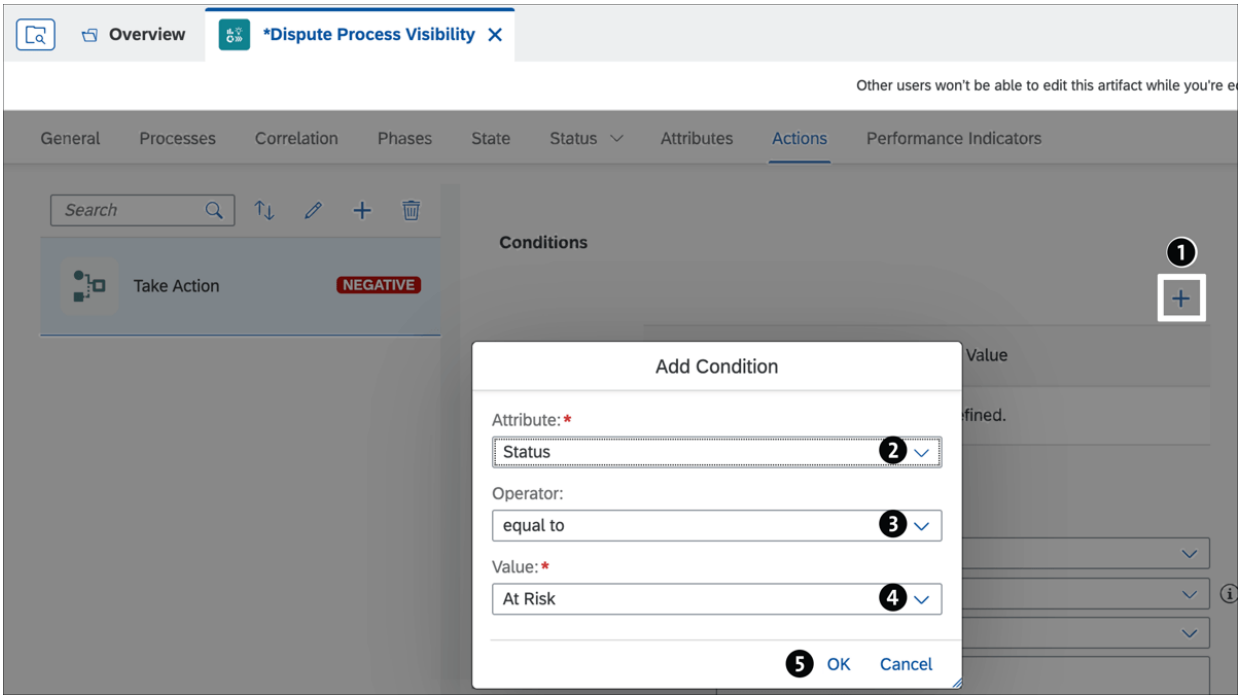


Figure 19.24 Adding a Condition for the Action

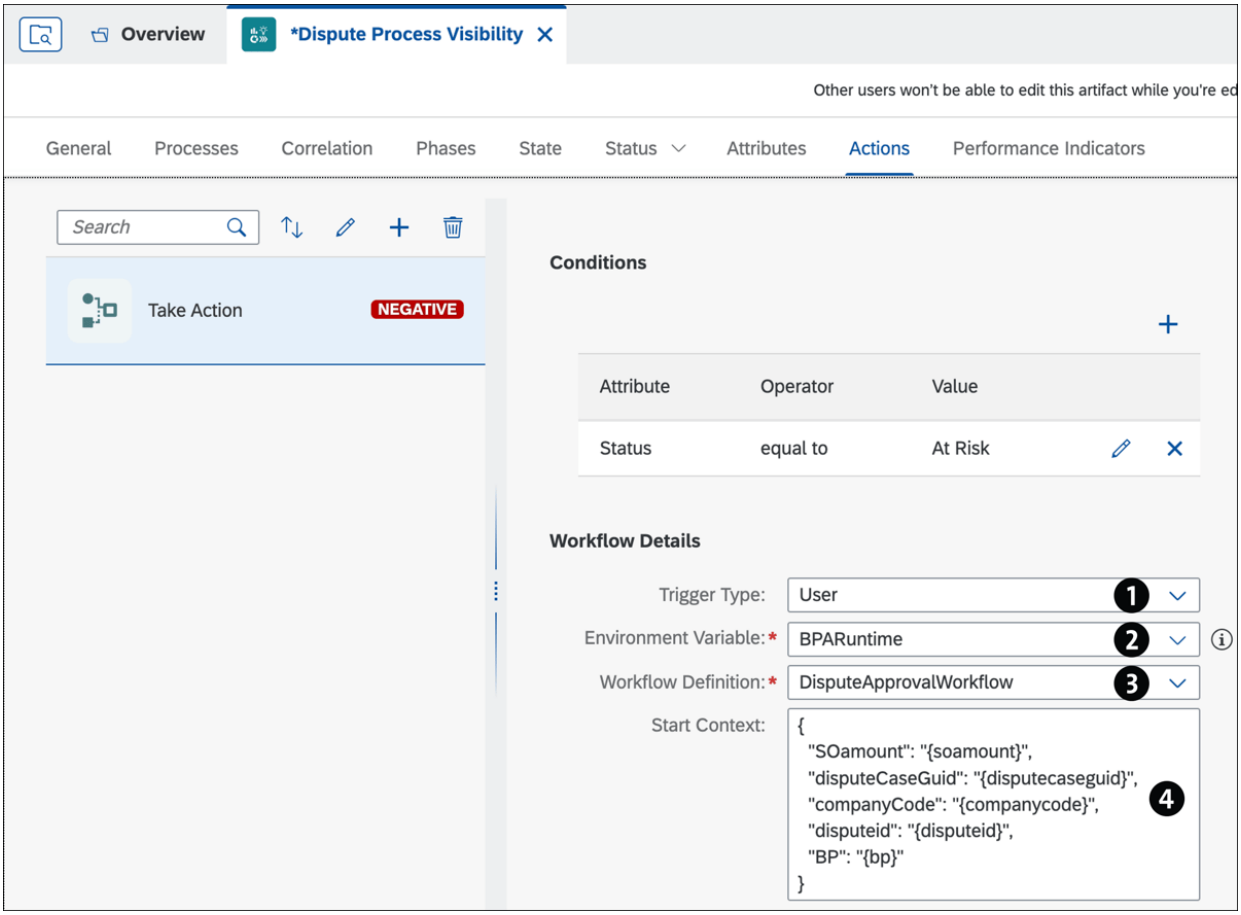


Figure 19.25 Add Action Workflow to Visibility

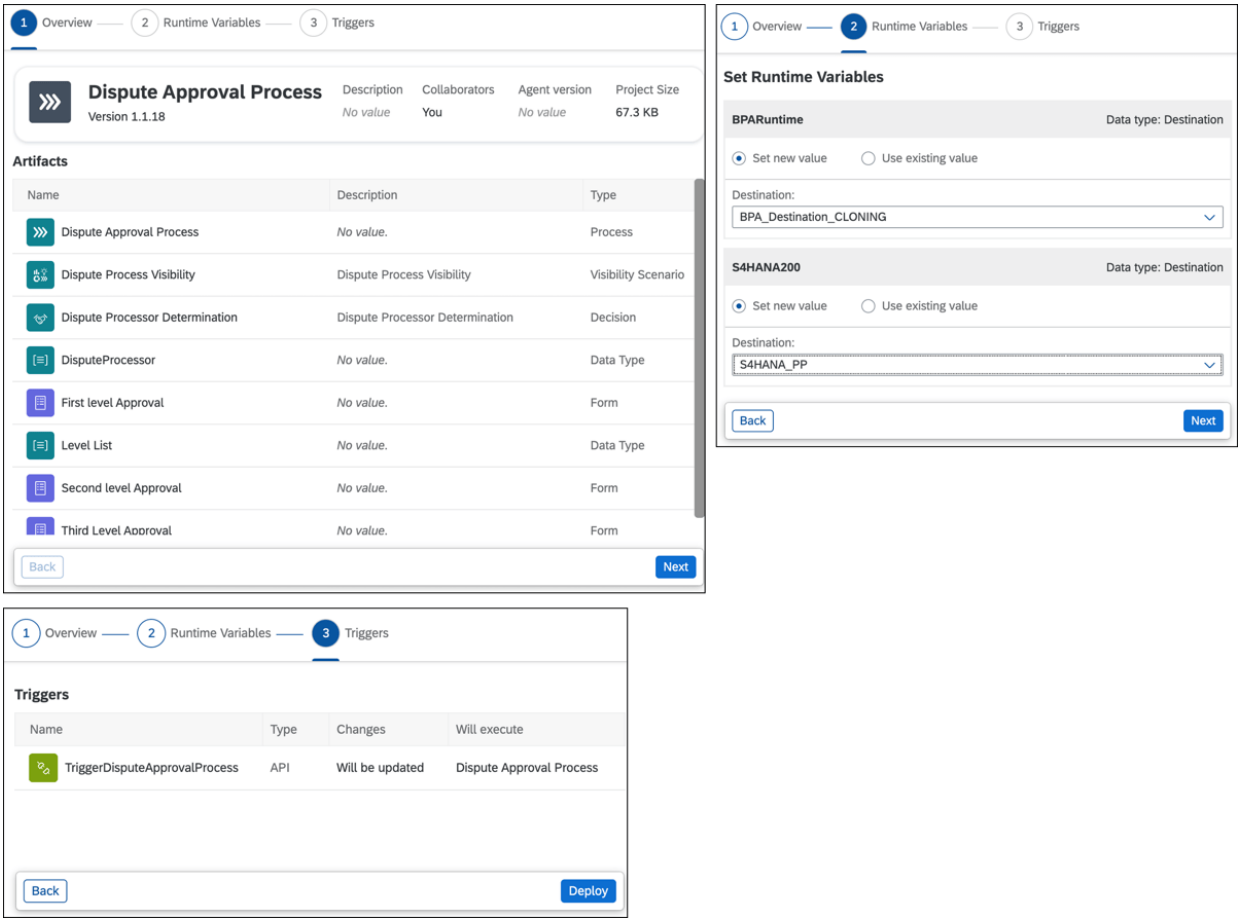


Figure 19.26 Deployment of the Project after Configuring the Action to Trigger Workflow

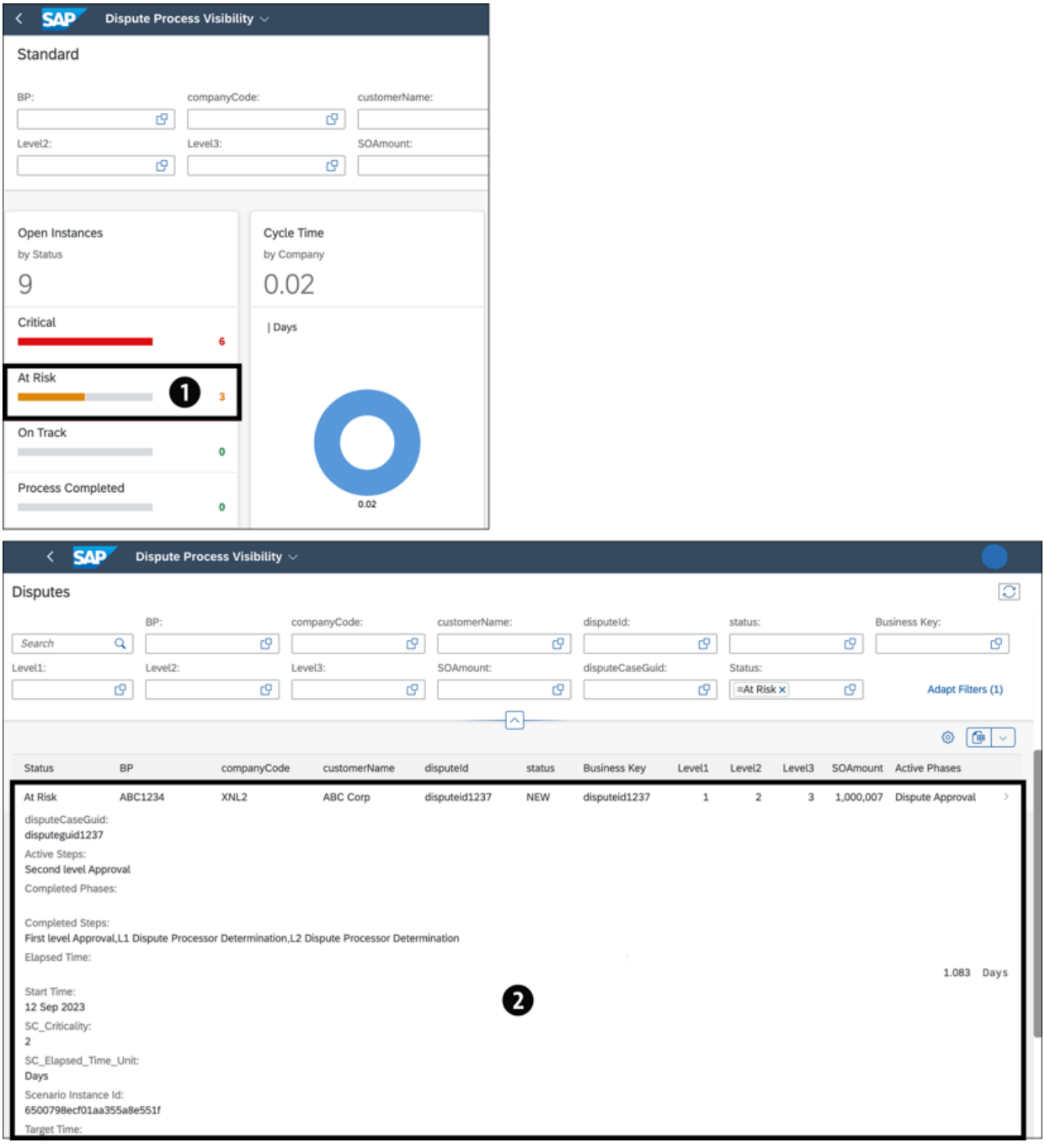


Figure 19.27 Testing Trigger Workflow Action: Steps 1 and 2

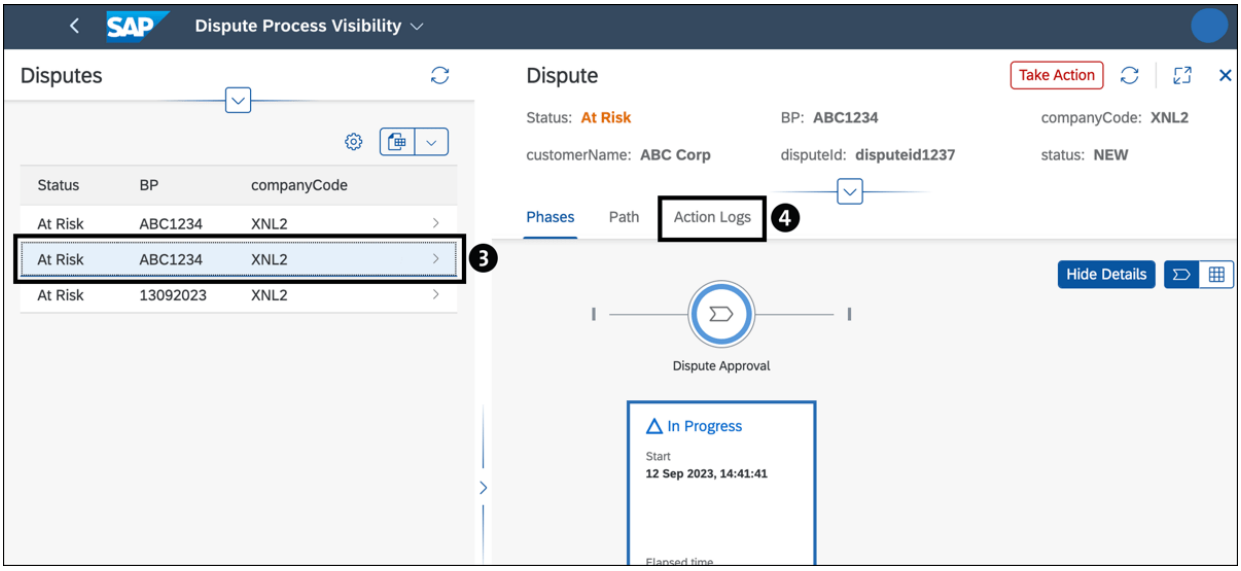


Figure 19.28 Testing Trigger Workflow Action: Steps 3 and 4

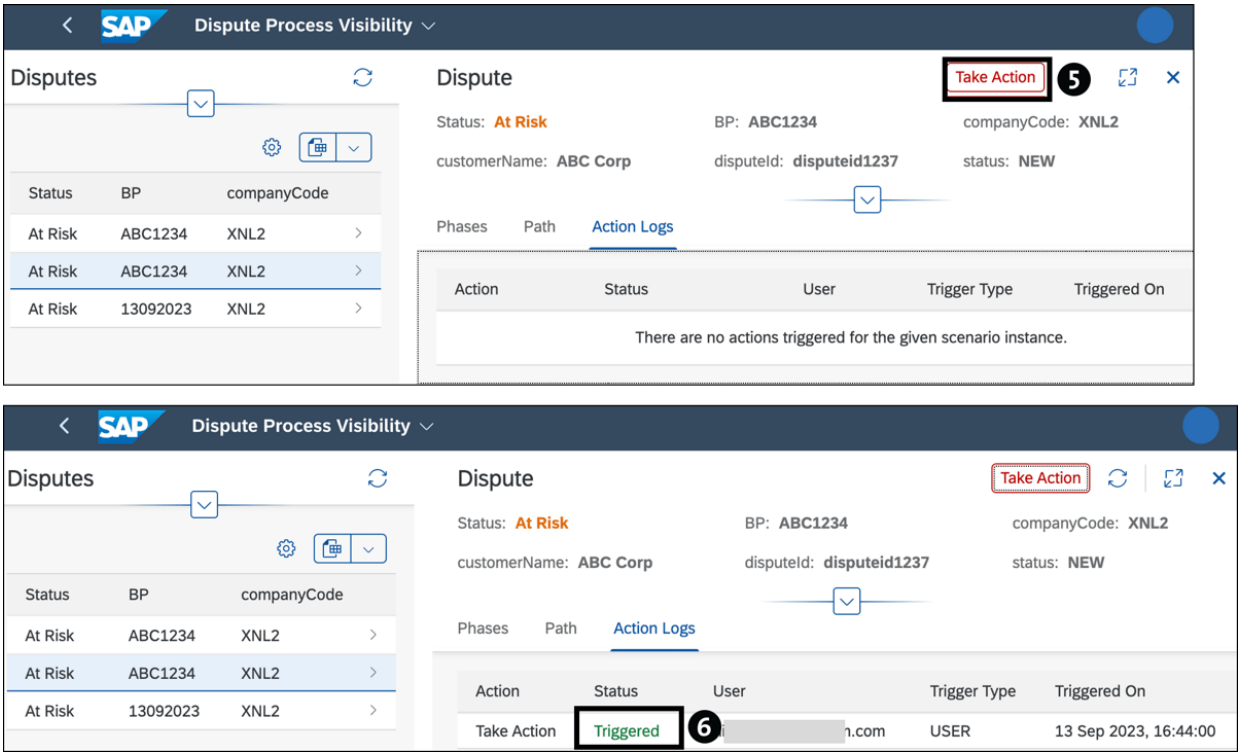


Figure 19.29 Testing Trigger Workflow Action: Steps 5 and 6

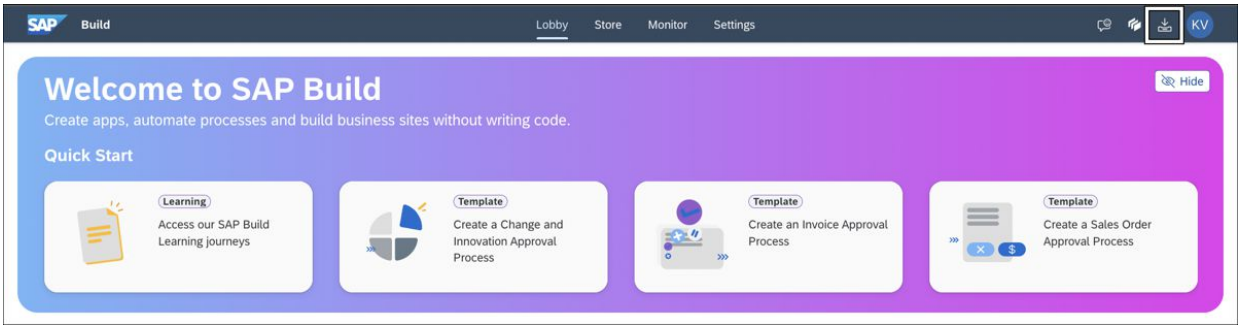


Figure 20.1 My Inbox Link in the Lobby of SAP Build Process Automation

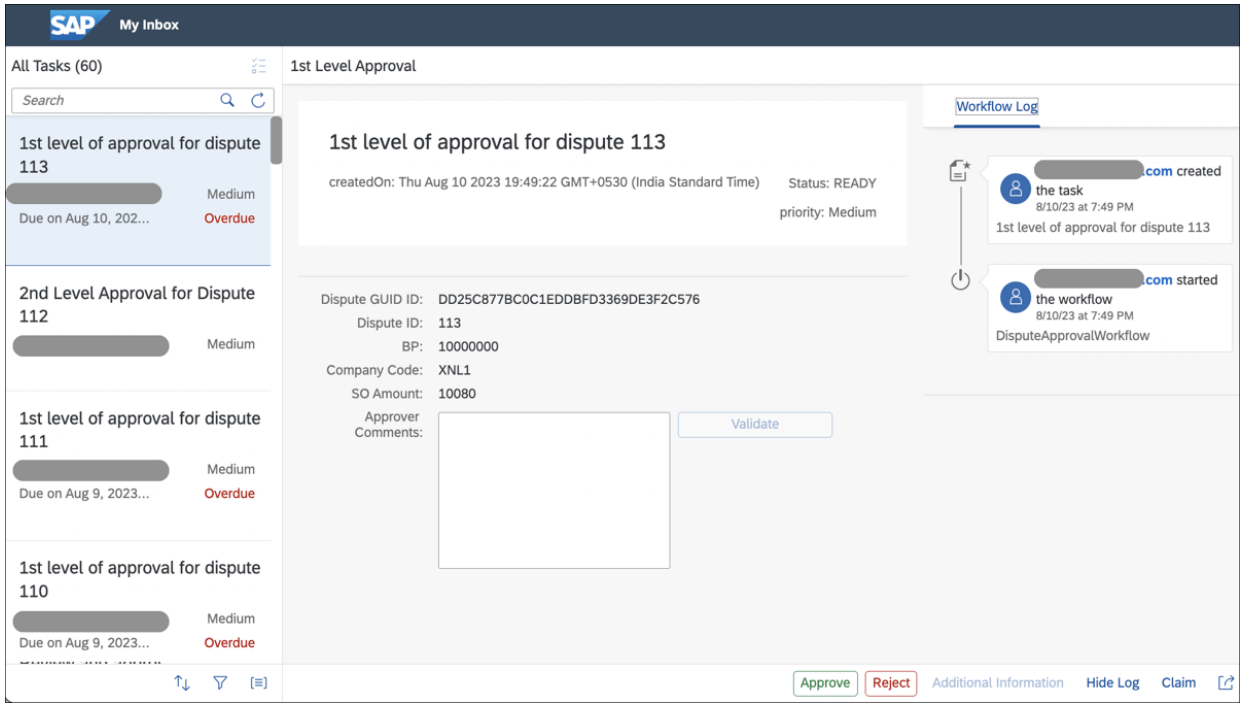


Figure 20.2 My Inbox in SAP BTP

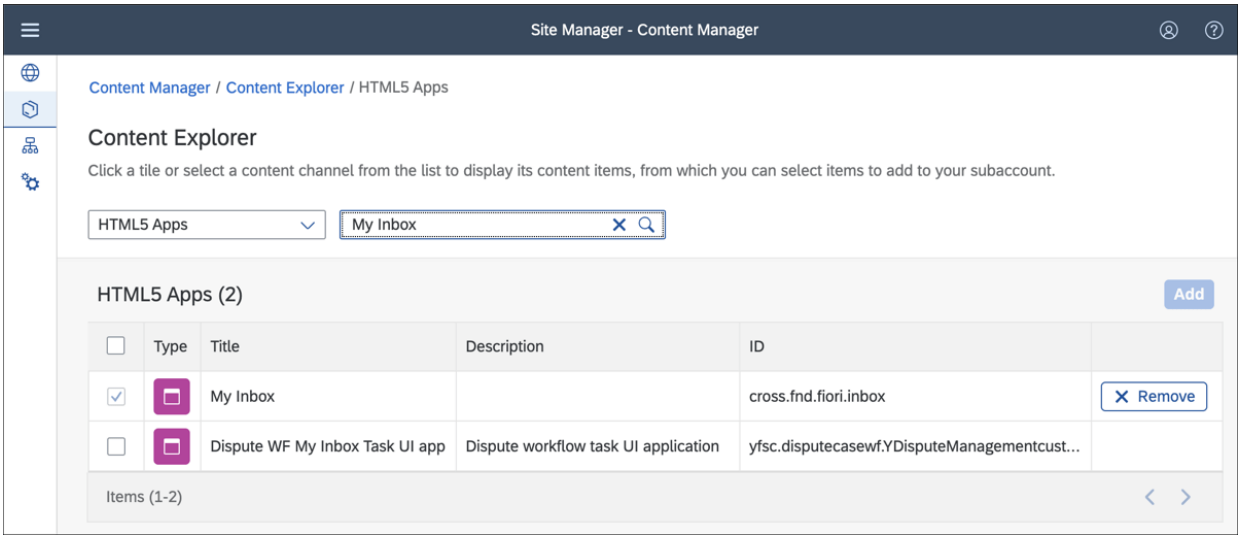


Figure 20.3 Adding the My Inbox App to the Contents of Your Site

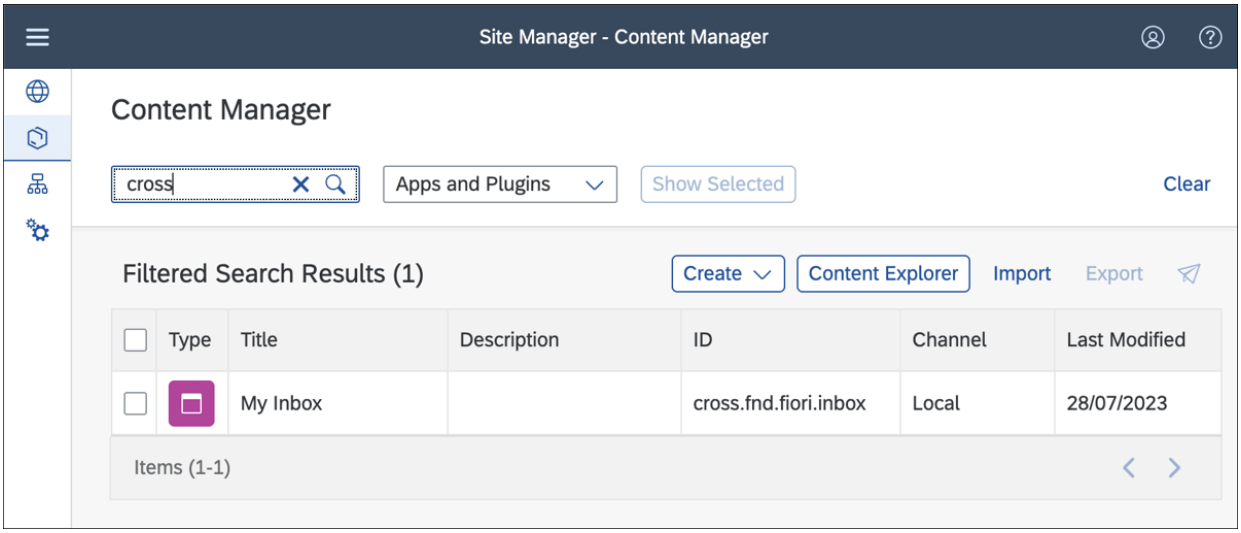


Figure 20.4 My Inbox App Now Available in Content Manager

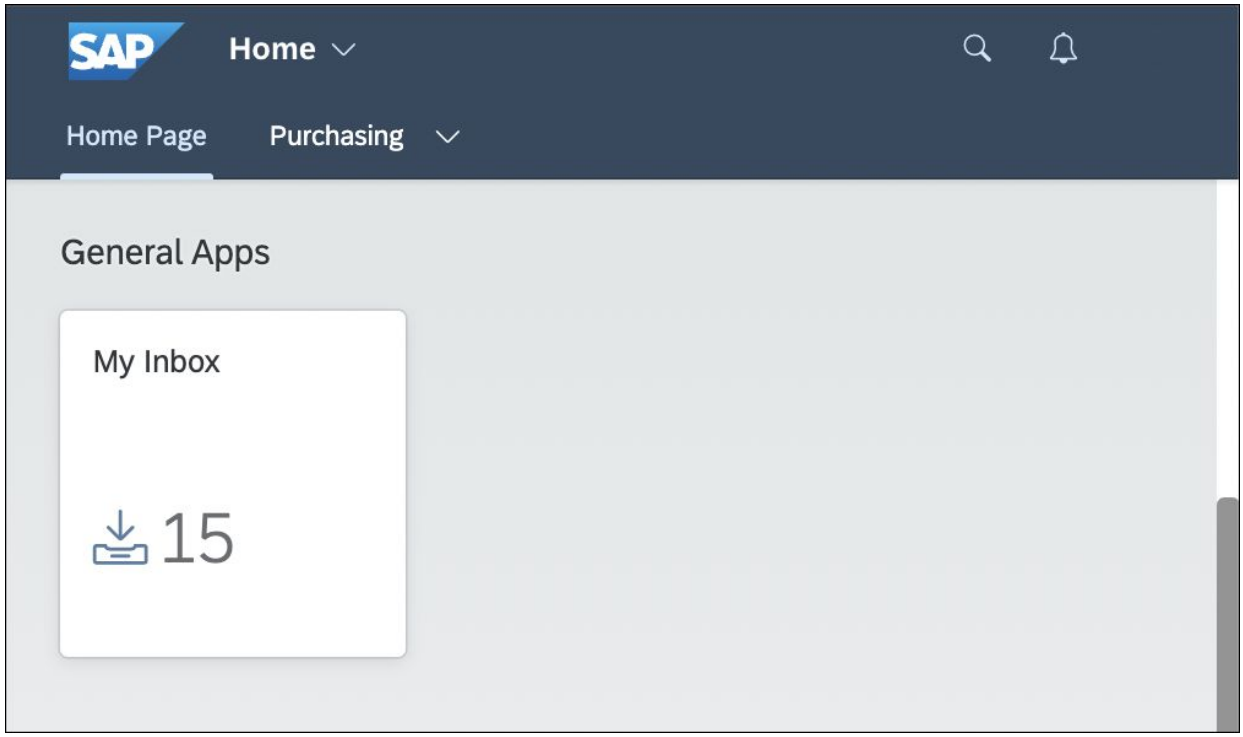


Figure 20.5 My Inbox Made Available in the Launchpad

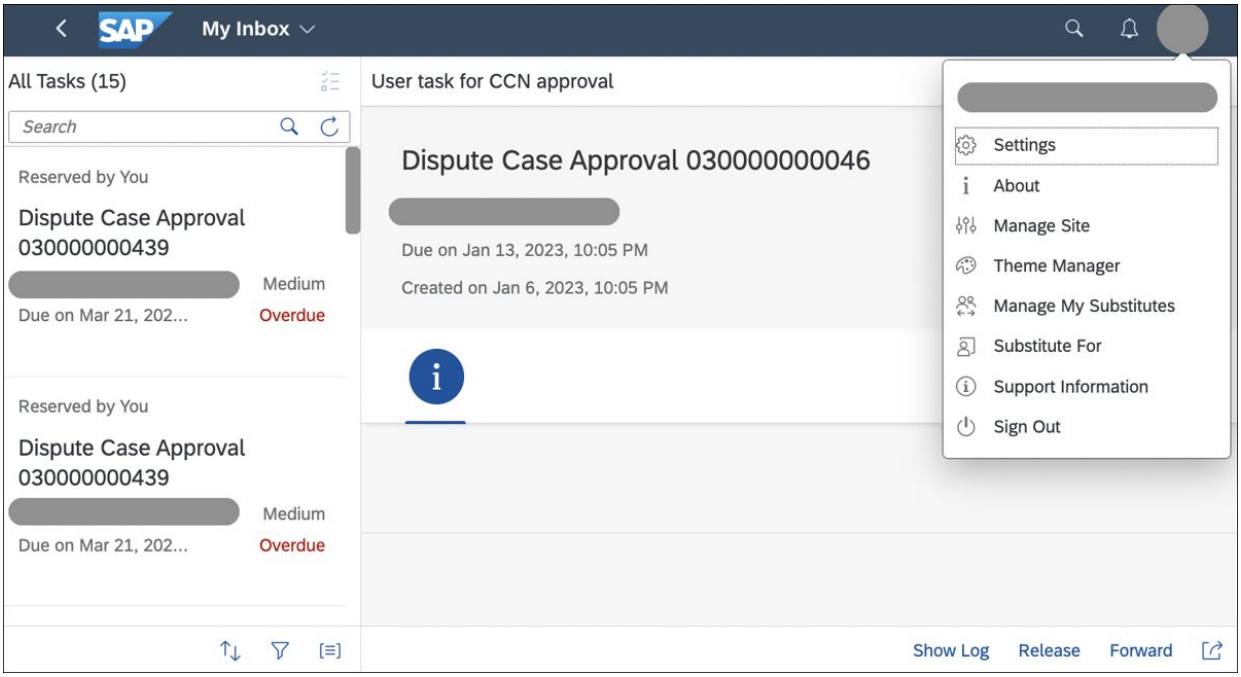


Figure 20.6 My Inbox User Action Menu Option: Manage My Substitutes

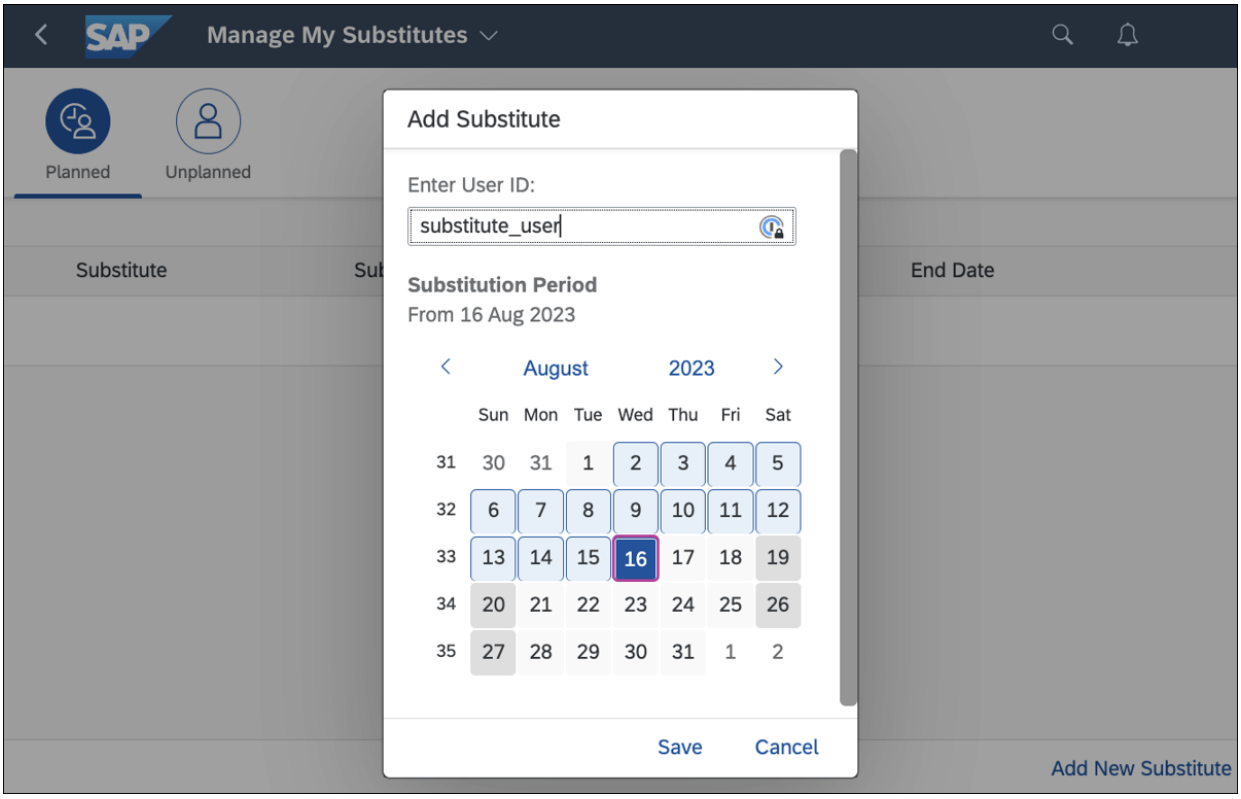


Figure 20.7 Adding a Substitute

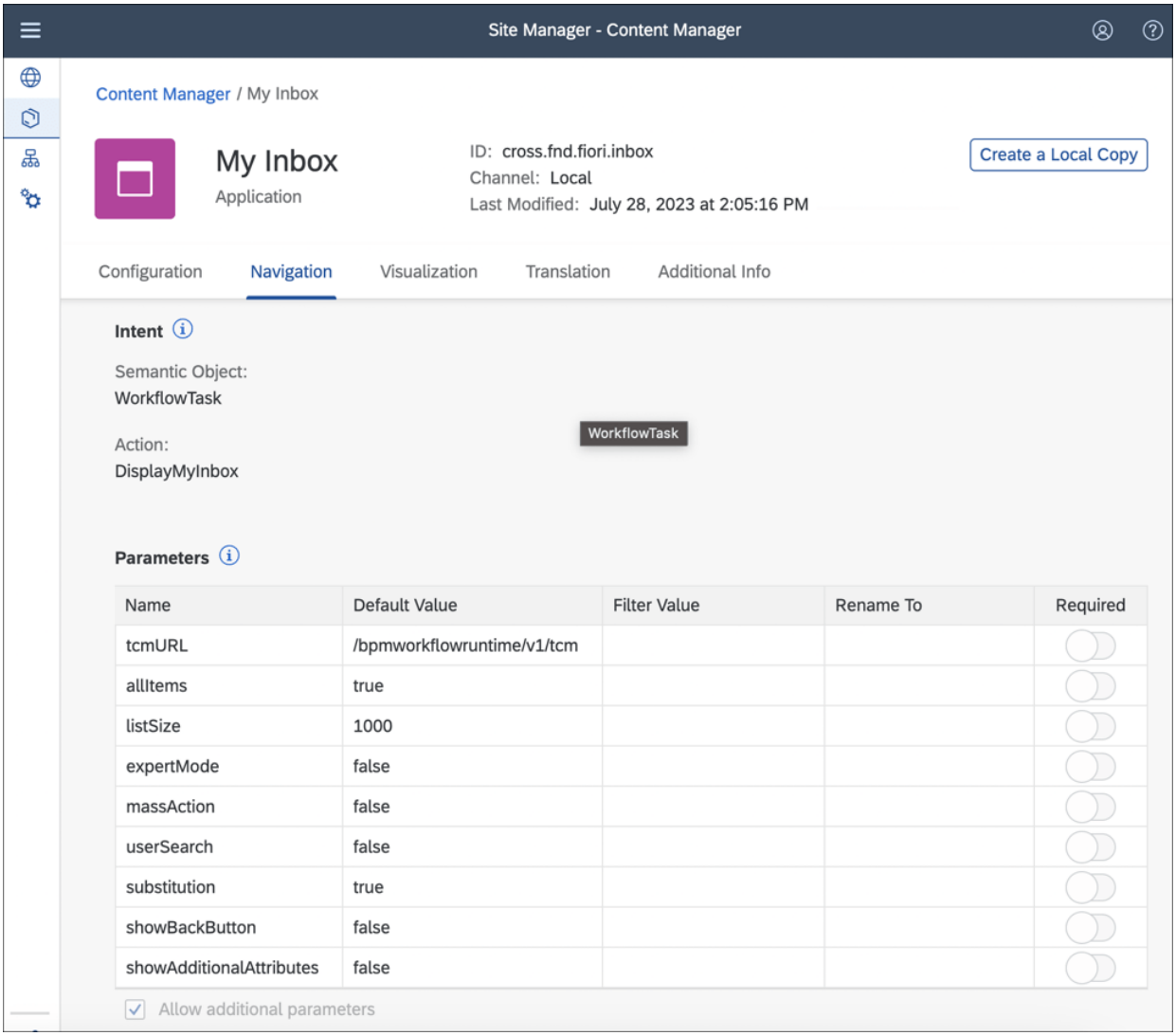


Figure 20.8 My Inbox Parameters

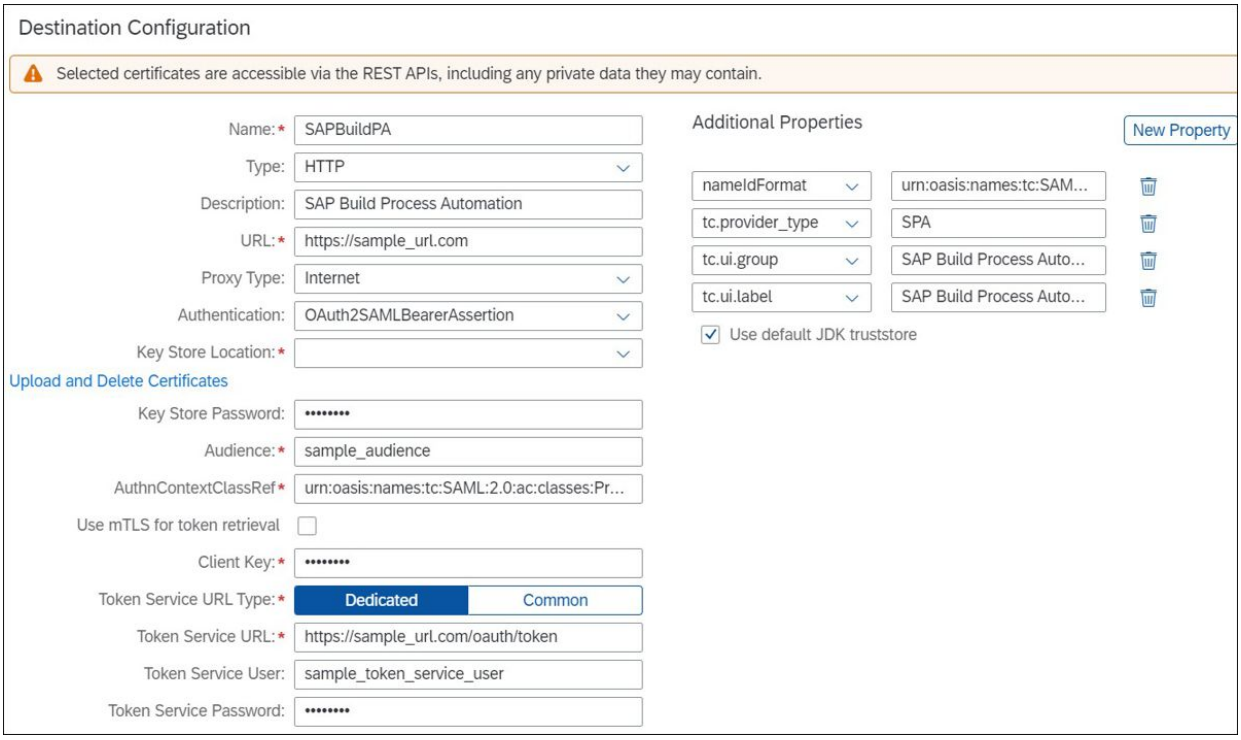


Figure 20.9 Destination Setup for SAP Build Process Automation with SAP Task Center