



Revenue Accounting and Reporting with SAP S/4HANA®

- › Configure SAP S/4HANA for classic and optimized revenue recognition
- › Calculate, post, and report on revenue with step-by-step instructions
- › Enrich your data through BRFplus structures, custom revenue accounting items, and more

Sreten Milosavljević
Swayam Prabha Shankara



Rheinwerk
Publishing

Sreten Milosavljević, Swayam Prabha Shankara

Revenue Accounting and Reporting with SAP S/4HANA®



Imprint

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

Editor Megan Fuerst

Acquisitions Editor Emily Nicholls

Copyeditor Julie McNamee

Cover Design Graham Geary

Photo Credit iStockphoto.com: 527727557/©

Gregory_DUBUS

Production E-Book Kyrsten Coleman

Typesetting E-Book Ill-satz, 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:
2023033462**

ISBN 978-1-4932-2434-0 (print)

ISBN 978-1-4932-2435-7 (e-book)

ISBN 978-1-4932-2436-4 (print and e-book)

© 2024 by Rheinwerk Publishing Inc., Boston (MA)

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

Notes on Usage **Table of Contents**

Preface

1 Introduction to Revenue Recognition

- 1.1 What Is Revenue Recognition?
- 1.2 Five-Step Model of IFRS 15
 - 1.2.1 Step 1: Identify the Contract
 - 1.2.2 Step 2: Identify Performance Obligations
 - 1.2.3 Step 3: Determine the Transactional Price
 - 1.2.4 Step 4: Allocate the Transactional Price
 - 1.2.5 Step 5: Recognize Revenue
- 1.3 Industry Impact
 - 1.3.1 Telecommunications
 - 1.3.2 Life Sciences
 - 1.3.3 Manufacturing
- 1.4 Revenue Recognition and SAP

- 1.4.1 Revenue Accounting and Reporting
- 1.4.2 Event-Based Revenue Recognition
- 1.5 Summary

2 Designing Your Revenue Recognition Landscape

2.1 The SAP Landscape and Revenue Recognition

- 2.1.1 SAP System Landscape
- 2.1.2 RAR in the SAP Landscape

2.2 Integration with Different SAP Components

- 2.2.1 Sales and Distribution
- 2.2.2 Customer Relationship Management
- 2.2.3 Billing
- 2.2.4 Non-SAP Systems

2.3 Revenue Recognition Data Model

- 2.3.1 Sender Components
- 2.3.2 Structures
- 2.3.3 Tables
- 2.3.4 Relationships
- 2.3.5 Extensions and Transports

2.4 Choosing Your Revenue Recognition Tool

2.4.1 RAR with SAP S/4HANA

2.4.2 RAR with SAP ERP

2.5 Summary

3 Configuring Inbound Processing (Classic and Optimized)

3.1 Setting Up Revenue Accounting Integration

3.2 Inbound Processing

3.2.1 RAI Classes in Classic Inbound Processing

3.2.2 RAI Classes in Optimized Inbound Processing

3.3 Extending RAI Classes

3.3.1 Defining Extensions

3.3.2 Populating Extensions

3.4 Summary

4 Revenue Accounting Items

4.1 Processing Revenue Accounting Items

4.1.1 Processing Methods

- 4.1.2 Parallel Processing Framework
- 4.1.3 Processing Order Items
- 4.1.4 Processing RAIs with a Predecessor
- 4.2 Managing Revenue Accounting Items
 - 4.2.1 Changing Items
 - 4.2.2 Exempting Items
- 4.3 Extending Transaction FARR_RAI_MON
- 4.4 Error Resolution While Creating Revenue Accounting Items
 - 4.4.1 Proactive and Reactive Measures
 - 4.4.2 Inflight Errors
 - 4.4.3 Data Validation Checks
 - 4.4.4 Resolving Errors without Transaction FARR_RAI_MON
- 4.5 Business Rules Framework Plus
 - 4.5.1 Applications and Structures for RAR Integration
 - 4.5.2 BRFplus Setup in RAR
- 4.6 Creating Custom Revenue Accounting Items
 - 4.6.1 Custom RAI Structure
 - 4.6.2 Custom RAI Content
 - 4.6.3 Singleton Classes
 - 4.6.4 Dynamic Processing Flow Controlled from Table
- 4.7 Summary

5 Contract Management

5.1 Setting Up Contract Management

- 5.1.1 Optimized versus Classic Contract Management
- 5.1.2 Accounting Principles
- 5.1.3 Presentation Methods
- 5.1.4 Calculation Methods for Local Currency
- 5.1.5 Contract Modifications
- 5.1.6 Cost Recognition
- 5.1.7 Contract Assets and Liabilities
- 5.1.8 Assign Company Codes, Number Ranges, and Contract Categories
- 5.1.9 Condition Types
- 5.1.10 Posting Periods

5.2 Setting Up Performance Obligations

- 5.2.1 Create Performance Obligation Types
- 5.2.2 Determine POB Types in BRFplus

5.3 Managing Performance Obligations and Event Types

- 5.3.1 Over Time Revenue Recognition
- 5.3.2 Point-in-Time Fulfillments

5.4 Modifying Contracts

- 5.4.1 Prospective or Retrospective
- 5.4.2 Contract Combination
- 5.4.3 Terminating Contracts
- 5.4.4 Contract Freeze

- 5.5 Handling Price Allocations
- 5.6 Summary

6 Revenue Posting and Reporting

- 6.1 Basics of Reporting and Calculations
 - 6.1.1 Posting Logic
 - 6.1.2 Customization for Posting of Revenue
 - 6.1.3 Table FARR_D_POSTING and Revenue Categories
- 6.2 Posting with ABC
 - 6.2.1 Transfer Revenue
 - 6.2.2 Calculate Contract Liabilities and Contract Assets
 - 6.2.3 Revenue Posting Run
- 6.3 Integrating with Profitability Analysis
- 6.4 Reporting
 - 6.4.1 SAP Fiori Applications
 - 6.4.2 New Styles of Reporting and CDS Views
- 6.5 Summary

7 Administration and Troubleshooting

7.1 Parallel Processing Framework and Performance Issues

- 7.1.1 What Is Parallel Processing?
- 7.1.2 Application Server Instances
- 7.1.3 Job Server Group
- 7.1.4 Custom Parallel Processing Framework
- 7.1.5 Modularization
- 7.1.6 Packaging

7.2 Reclassification

- 7.2.1 Creation and Structure of ZFARR_D Tables
- 7.2.2 Populating the Custom Posting Table
- 7.2.3 Extending the Standard Posting Program

7.3 Data Cleanup

- 7.3.1 Data Checks
- 7.3.2 Customizing
- 7.3.3 Business Add-In FARR_BADI_RAI2
- 7.3.4 Message Capturing by Simulating

7.4 Helpful Tips and Tricks

- 7.4.1 Navigator
- 7.4.2 Additional Information

7.5 Summary

8 Migrating to Optimized Contract Management

- 8.1 Business Case for Migration
- 8.2 Preparation Activities for Migration
 - 8.2.1 Contract Management Activation
 - 8.2.2 BRFplus Verification
 - 8.2.3 Business Add-In Modifications
 - 8.2.4 Process Changes
- 8.3 Performing the Migration
 - 8.3.1 Migration Activities
 - 8.3.2 Migration Errors
- 8.4 Post-Migration Cleanup
- 8.5 Summary

9 Event-Based Revenue Recognition

- 9.1 Solution Background
- 9.2 Sell-from-Stock Scenario
- 9.3 Sales-Oriented Scenario for Customer Projects
- 9.4 Revenue Recognition Keys

9.5 Summary

[Important Business Add-Ins](#)

[The Authors](#)

[Index](#)

Service Pages

Legal Notes

Preface

Revenue accounting and reporting (RAR) has been available in the market for almost a decade, but it's still considered as a new module within a niche area. However, in the past decade, we've seen a significant shift in how business is run. There's no company in the world that's not talking about digital transformation, and technology plays a significant part.

More and more companies are implementing a service-oriented business model, meaning that even traditional manufacturing companies are trying to operate as service providers. These changes are creating a need for compliance with the International Financial Reporting Standards (IFRS) 15 as a necessary reporting requirement and as a tool that enables companies to properly assess their results. All this being said, revenue management is now a critical area for projects where the whole organization is in focus.

Since the transition to IFRS 15, RAR has evolved in many ways in SAP S/4HANA (now on release 2022 as of the time of writing), which will be covered by this book. RAR has embraced technological advancements since its appearance on the market, and there are now many more business cases where RAR can or must be applied.

In this book, we've aimed to provide information for different types of readers. Our first target group consists of decision makers who need to start a project where RAR might be in focus. Decision makers need to feel confidence that despite the complexity, SAP offers a tool that can handle their business requirement with ease. We've also written for consultants who want to grow in the area of revenue recognition; even after a decade in the market, there's still a huge demand for good consultancy. Next is developers. Since RAR is a complex area, it will benefit when the newest techniques are used, so in this book we'll get developers up to speed and indicate some dos and don'ts. Last but not least, we've written this book for end users, with the goal of showcasing the look and feel of RAR, the reporting options, and the overall user experience.

How This Book Is Organized

We've organized this book to follow the flow of an RAR implementation project. We'll start with an explanation of the SAP standard and later we'll follow the process to set up RAR. To close the book, we'll introduce the migration to optimized contract management (OCM) and the new event-based revenue recognition (EBRR) solution.

We recommend reading this book sequentially from [Chapter 1](#) onward; however, if you prefer, you can directly go to any chapter and start reading about that topic. For example, if you're interested in learning about setting up contract management, you can start reading [Chapter 5](#)

without reading previous chapters. Let's review what is covered in each chapter of this book:

- **Chapter 1: Introduction to Revenue Recognition**

This chapter provides an overview of both IFRS 15 and Accounting Standards Codification (ASC) 606. We'll provide examples from the most impacted industries and explain the most common pitfalls during a project. Also, we'll showcase high-level design solutions that might prove useful in understanding how business scenarios specific to certain industries can be designed in RAR.

- **Chapter 2: Designing Your Revenue Recognition Landscape**

In this chapter, we'll start our deep dive into RAR. First, we'll explain how RAR fits into the overall system landscape (these decisions might be crucial in some industries). Second, we'll provide detailed information about the data model and structure of tables in RAR. This is important for both functional and technical consultants, so we recommend reading this chapter carefully.

- **Chapter 3: Configuring Inbound Processing (Classic and Optimized)**

Inbound processing is the first step to perform when setting up RAR, where we tell the system how to "talk" with other components of the SAP landscape. Here, we focus on both the functional and technical sides, and we also bring in the topic of optimized inbound processing (OIP), which is rather new and can be a solution for some common problems.

- **Chapter 4: Revenue Accounting Items**

RAIs and the usage of the RAI monitor are the focus of this

chapter. Here, you'll be guided step by step through proper use of the application for RAI processing. Also, we'll include technical details about how RAIs can be extended.

- **Chapter 5: Contract Management**

In this chapter, we review the necessary settings for proper creation of contracts in the RAR system. We'll provide step-by-step instructions so that even inexperienced readers can understand how setup is organized and what needs to be done to enable a simple flow.

- **Chapter 6: Revenue Posting and Reporting**

This chapter introduces integration between RAR and accounting. We'll explain in detail how this integration can be achieved and provide step-by-step guidance for the results to expect from this integration. Last but not least, in this chapter, we touch on reporting options. This area benefited the most from the SAP HANA engine.

- **Chapter 7: Administration and Troubleshooting**

In this chapter, we focus on the administration of the overall RAR solution and how to troubleshoot the most common problems. We also spend a significant portion of this chapter on an important technical topic within RAR: the parallel processing framework (PPF).

- **Chapter 8: Migrating to Optimized Contract Management**

OCM is a highlight of newer RAR versions. However, in addition to explaining the benefits of OCM, we'll also provide examples of where it might not be the best fit. So,

in this chapter, we aim to broaden your knowledge in this area so proper architectural decisions can be made.

- **Chapter 9: Event-Based Revenue Recognition**

This chapter is dedicated to the newest tool in the SAP portfolio for revenue management: EBRR. This chapter provides a high-level introduction to the solution and describes when it can be applied. We will not provide detailed information about EBRR's setup and use in this book.

Acknowledgments

We feel that after a while on the market, a tool such as RAR deserves an in-depth book. Through participation in many projects, we witnessed that RAR was being approached in the same way as always; however, there are new features that enable users to execute tasks much more efficiently than before. So, we took on the challenge to explain certain RAR topics in detail, and also to shed light on new subjects that would help in projects.

Writing this book was a tremendous effort: It took quite a few sleepless nights and weekends. So, our first thank you goes to our families, who had to sacrifice even more time so this book would become a reality. Also, I would like to mention our customers who were very kind in sharing their knowledge about IFRS 15 and ASC 606, which helped improve the quality of this publication, and also who were very considerate about work which was happening in parallel. Last but not least, thank you to the team at

Rheinwerk Publishing, namely Megan Fuerst, who guided and helped this book reach its audience the right way.

—*Sreten Milosavljević*

I want to thank my coauthor Sreten for giving me the opportunity to be a part of this book. He has a lot of experience in RAR and is a major contributor in this book, and I am very grateful for that. Writing this book was challenging in terms of time and effort; we are definitely thankful to our families and parents as they have been very supportive. As much as writing this book took effort, it was a very positive thing in my life.

—*Swayam Prabha Shankara*

Conclusion

We hope that reading this book will help readers become knowledgeable on the subject of RAR and the implementation effort. We also hope that developers will find it useful in applying new techniques for enhancement. Finally, we hope that end users will find this book useful in order to widen their knowledge and gain understanding of how to work with RAR.

1 Introduction to Revenue Recognition

Everyone knows that cash doesn't equal revenue. But how and when can we recognize and report revenue? In some cases, the answer to this question is still relatively simple. What we invest in customers is revenue to recognize, and when we issue an invoice, we can recognize that revenue. However, there are more complicated cases in which both amount and time of recognition can be a challenge. In this chapter, we'll explain the basics of revenue recognition.

Revenue is an accounting process that gains even more importance in a competing market. If a company operates in the automotive, services, or telecommunications space, it's important to report revenue correctly and consistently as the main measurement of its overall success.

In the past several years, there have been important, tectonic changes in how entities recognize revenue (some called these changes the “perfect storm in accounting” as they came together with International Financial Reporting Standards [IFRS] 9, 16, and 17). Now that the initial adjustment is over, it's a good moment to look back and see how the initial implementation went, what challenges we're

facing now half a decade after implementation, and where we go from here.

As the market-leading enterprise resource planning (ERP) provider, SAP has been providing tools for revenue recognition for a while now. In the early SAP R/3 and SAP ERP days, there were products for calculating accrued revenue that were tightly integrated with the Sales and Distribution (SD) module. With the latest versions of SAP ERP, the standalone SAP Revenue Accounting and Reporting solution came into light as a tool that will be used to fulfill the latest requirements from the International Accounting Standards (IAS) board.

Now, the *revenue accounting and reporting* (RAR) functionality has evolved further. In SAP S/4HANA, we have classic and optimized versions of RAR, which can be integrated not only with sales and distribution but also with new products such as SAP Billing and Revenue Innovation Management and even with external systems. The tool went through significant changes, resulting in several new added functionalities over the years and changes to already existing ones. RAR is becoming even more important with migration to SAP S/4HANA because it's a mandatory tool for any kind of revenue recognition. In addition, SAP provides additional products, such as event-based revenue recognition (EBRR), which can be used to solve some specific challenges.

In this chapter, we'll introduce key revenue recognition topics to lay the foundation for the rest of the book, including basic concepts, the IFRS five-step model, the impact of those standards on industries, and how SAP has

provided revenue recognition functionality to meet those industry needs.

1.1 What Is Revenue Recognition?

Revenue is one of the key measurements of business performance for every company. Having that in mind, companies need to push whatever qualifies as revenue to its limits. Examples of such activities are all around:

- A construction company collects the payments in advance, before the work is performed.
- An audit company charges clients on time and material basis, before the work is completed.
- A manufacturing company gets invoices paid before the goods reach the end customers.

In all these cases, at the heart of accrual accounting, there is a revenue recognition concept as a set of rules for how and when revenue can be recognized. In a simple scenario, a product is delivered to the customer who immediately pays for it. However, modern business is more complicated than that. We're facing situations where a company takes a long time to produce a product, but gets paid in the meantime. In that case, we can't make any more links between cash and revenue. In addition, competition pushed businesses to improve time to market, which in return, gave birth to bundles. Marketing departments are based on consumer preference reports creating new products that bundle different goods and services. These bundles also contain hidden discounts that aren't visible at first sight. For

example, you subscribe to the newest service offering from your telecommunications service provider (telco), and, in return, you get selected services, cloud space, and other goodies for free. All these items are in fact a discount given by your vendor to either retain you as a customer or as a reward for switching to them.

All these situations make the world of revenue recognition both complex and exciting. However, they aren't new, so the question is, what actually changed? Why was there a need to introduce IFRS 15 in the first place? If you ask audit companies this question, the answer is usually that a new model was needed to improve comparability between companies and across industries. IAS 18 also contains a set of rules and principles, but these were written quite broadly and gave a lot of freedom to companies to interpret them for specific situations.

Example: Impact of IFRS 15 Changes

A telco company is selling a device and service bundle. Because the company is new to the market, expensive devices are added to packages without charge to get consumers' attention.

In the old IAS 18 world, all the revenue paid by the consumer would be treated as service revenue, while devices would get nothing (because the consumer wasn't paying for them). When you compare this company with others, it would be difficult to differentiate if their service revenue was boosted by service quality or improvements in network equipment, for example, or simply because

they grabbed a lot of clients based on offering the newest device for free.

Here, IFRS 15 (or Accounting Standards Codification [ASC] 606) comes into the picture by introducing tools and measurements that make this kind of comparison easier. In our example, the introduction of the standalone selling price (SSP) forced companies to allocate some portion of the revenue to the device too, even if its sales price is zero.

IFRS 15 has already been in place for a while, with the first pilot companies adopting it five years ago. From January 1, 2018, IFRS 15 has been mandatory for all entities listed on any stock exchange. Currently, IFRS 15 is mandatory in 168 countries worldwide. Having said that, an entity that doesn't comply with IFRS 15 might find that attracting investors or sourcing credit lines is more difficult because it sends the message that the company's financial statements aren't comparable with others and can't be trusted. These facts reveal why revenue recognition should be the centerpiece in every company's digital transformation journey.

IFRS 15 and ASC 606

Often, you'll see ASC 606 and IFRS 15 used as synonyms. Indeed, being jointly developed by FASB and IASB, they are very similar, although not really the same. For example, regarding license renewal scenarios, ASC 606 lets you start revenue recognition only once the period of renewal starts, whereas IFRS 15 doesn't have such a limitation. If your customer decides to renew a license on

January 1, starting from January 7, IFRS 15 lets you recognize that revenue immediately, while ASC 606 requires recognition to start only on January 7. The impact of these different treatments needs to be considered before choosing an application.

1.2 Five-Step Model of IFRS 15

When looking at revenue coming from contracts, many of the revenue transactions are rather straightforward and simple. However, some can be highly complex and challenging to understand, such as contracts with multiple elements, agreements with milestone payments, or software arrangements. In all of these cases, it might be difficult to understand what entity committed to deliver, how much revenue should be recognized, and when revenue can be recognized.

To help users apply the standard, the IFRS board developed a five-step model, as shown in [Figure 1.1](#). We'll walk through each of the five steps in the following sections.

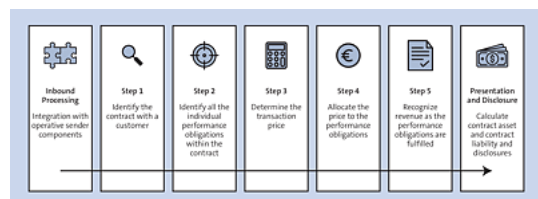


Figure 1.1 Five-Step Model of Revenue Recognition

1.2.1 Step 1: Identify the Contract

The first step looks like the easiest one: a *contract* is a document (where the standard doesn't apply is for written, oral, or common business practice), which defines the rights and responsibilities of contractual parties. However, in practice, this one might be more challenging. Imagine an entity that is selling goods and services to customers over a certain period of time, which could be over several years. Here, various discounts are applied to some goods, there are free goods and services provided, and so on. In parallel, vendors can have several other contracts for the same goods that are part of different deals. It gets more complicated if our customer isn't a real end customer, but we're directly delivering goods to the end user.

To properly identify contracts, it's essential to define at what level a deal is being negotiated and what discounts are being applied. In addition, it's not uncommon to establish a certain time frame in this logic. For example, if orders are being created with the same customer within a certain time frame, they need to be combined in one contract.

Changes of Contracts

Contract modification has a significant impact on system performance. In practice, it's not uncommon to see companies from the same line of business with big differences in contracts structure: in one case, there is a large number of smaller contracts, and in another, a small number of huge contracts. These situations will be discussed in later sections.

Maybe the most important step in identifying a contract is that it must represent a binding arrangement between two parties. There are two main options when it comes to identifying

a contract:

- **Contract-by-contract basis**

The contract-by-contract approach refers to identifying all separate contracts that represent separate deals with the customers.

- **Portfolio approach**

The portfolio approach refers to making a group of similar contracts that will be looked at on a higher level. The portfolio approach was tempting to organizations, especially in the early days, because it didn't require a deep dive into data and distinguishing all relevant information, which is required to properly apply the standard (this was particularly visible in telco companies). The portfolio approach does, however, come with its own set of challenges, but it won't be covered in this book.

In this activity, that is, identifying the contract, the first task for a business is to identify IFRS 15 as the standard that should be applied or determine if there is some other standard available for specified contracts. For example, leasing contracts are part of IFRS 16, and although IFRS 15 might be responsible for providing some data, contract creation and management belong to a standard that isn't part of this book. Even though no entity is completely excluded from revenue standards, the following transactions aren't part of IFRS 15/ASC 606:

- Leases (covered by IFRS 16)
- Contracts part of financial services—insurance (covered in IFRS 17/ASC 944)
- Guarantees
- Nonmonetary exchanges to enable future sales
- Credit card fees
- Gaming contracts

In addition, there are certain transactions that might end up in revenue postings and recognition but aren't part of IFRS 15, such as dividends, which should be clearly excluded and not mixed with regular revenue coming from contracts with customers.

The next challenge an entity might face is how to identify a customer. This problem is more visible if we have multiple parties involved in one contract.

Example: Identifying the Customer

Let's consider an example. A pharmaceutical company is selling different diagnostic instruments and reagents that are being used to perform different tests. These items are being sold through a distributor, but to the same chain of hospitals and delivered to different locations. With the same hospital chain, the entity has additional agreements that are discussed and agreed upon separately. In addition, the entity might use different distributors for deliveries and distribution to the hospital chain.

To differentiate the customer, it's necessary to look at which level of terms and conditions are being agreed upon. If terms are agreed with the hospital chain, then the chain should be the customer. All contracts that might result from the same agreement need to be combined under the same IFRS 15 contract.

To conclude, there is no out-of-the-box answer to the question of how a customer should be identified, which is why it's essential that the entity looks at each contract individually to make sure contracts, customers, and rules are applied correctly when contracts are combined.

Once we identify the customer, the next step is *enforceability*. In this activity, it's essential to establish if there is legal and binding enforceability of the contract with clear terms and conditions. Sometimes, contracts represent the simple purchase of products, and sometimes they represent a complex multiyear project to build a power plant. In both cases, there must be enforceability for both parties to define the contract.

Enforceability is one more area in which its meaning can vary from simple cases where the contract is within a single area of jurisdiction to more complex cases when contracts are spread over several countries in which different jurisdictions apply. In the latter case, it might be more challenging to determine enforceability. Complexity is increased further when parties enter into contract amendments and modifications by changing terms and conditions, applying discounts, or even changing the substance of the contract itself. Either way, the critical task is to determine that there are rights and obligations for both parties and enforceability of the contract. Only in those cases can we safely say that a document can be treated as a contract under IFRS 15.

Example: Enforceability

Let's consider another example. A vendor supplied a customer with goods prior to contract signing. The customer received the goods and started using them. Can this kind of contract be considered an IFRS 15 contract?

Here we need to verify the enforceability. The standard is open about *how* a contract is concluded, so it can be based on a verbal agreement. Therefore, if there is enough evidence that an agreement existed even without a signed contract, those arrangements can be considered contracts.

Let's look at an alternative scenario. A telco company is providing a free trial period during which the customer is using the service without any payment. After that period has expired, the customer can decide whether to enter into a long-term agreement. Can this transaction be considered a contract?

In this case, the answer is clearly no. During and after, the customer is free to simply walk out of the contract, so there is no enforceability. Because the document wasn't binding for both parties, it can't be considered an IFRS 15-relevant contract. This example can be particularly handy even in cases where we have long-term agreements between parties in which the customer is being supplied with goods over a period of time. If there is an option for a customer to stop using our services and thus suffer no legal consequence, that kind of agreement isn't enforceable and can't be considered an IFRS 15 contract.

In addition to enforceability, there is also the idea of *probability*. Before the agreement can be considered an IFRS 15 contract, an assessment has to be made to determine the probability of collecting the amount stated in the agreement. In IFRS 15, *probable* is

defined as “more likely to occur,” and the general rule is that more than 75% is considered probable.

When we’re talking about probability, we must understand price concessions because it’s very important to determine the impact of future price concessions on the entity’s ability to fulfill payment obligations. If it’s “highly unlikely” that payment is going to be made, there is probability that the agreement can’t be considered an IFRS 15 contract. However, if we only know that the amount to be collected can be less than agreed with the agreement remaining collectable, then we’re talking about *variable consideration*, which will be discussed in [Section 1.2.3](#) when we cover how to determine the transactional price.

To conclude, the determination of contracts is a particularly sensitive area that has a big impact on the IT side of providing data for IFRS 15 reporting. The number of agreements considered IFRS 15 contracts overall is having a direct impact on complexity, performance, and data requirements. Therefore, it’s worth spending time to have a clear idea of what should be identified as an IFRS 15 contract.

1.2.2 Step 2: Identify Performance Obligations

Performance obligations (POBs) represent a promise in a contract to deliver goods or services to a customer. The most important topic when identifying POBs is *distinction*: a good is treated as distinct if a customer can benefit from the good on its own. In addition, in practice, if an entity sells this good on its own, it can be treated as distinct.

Why is this important? In practice, companies often ask themselves whether some service, which is usually bundled, should be represented as a separate POB. Let’s take the case of warranties: When a customer buys some goods, in most cases, the customer has been offered some sort of guarantee that the sold item will work properly for a certain time frame. However, the customer must also be offered an extended warranty—adding periods in which a company guarantees that the product will work without malfunctions. So, the question companies ask is whether they should treat warranties as separate POBs or not. To answer this, we need to take a look at the company’s product catalog: if they offer a warranty as a separate product, then we can treat it as a separate POB. However, if it’s not sold separately, it needs to be bundled with the original product with proper impact on the transactional price and the SSP. So, in this case, the usual answer is that a standard warranty is treated as part of the original product, and any additional warranty that the customer opts for is a separate POB.

Activation Fees

One common case is the different fees charged to customers when buying some good or service. The company will simply charge a certain fee as a cost to the customer for mere activation of the service.

The answer in this case is very similar: if they can buy that product separately—it’s a separate POB. If not, then it needs to be bundled with the main POB that is being activated.

As mentioned earlier, distinction is an important activity when determining POBs. Each distinct good or service that entity provides is a separate POB. If it's not distinct, it's bundled with other nondistinct goods until a distinct bundle is created. If we look at our previous example, it's clear that any kind of fee the customer purchases (or is obliged to purchase) can't be treated as a separate POB because it isn't clear how the customer can benefit from it.

Once it's clear that the POB is capable of being distinct, we need to assess it as separately identifiable from other promises in the contract.

Let's walk through another example. A software company is entering into a contract with a customer to provide a customized SAP S/4HANA solution. Because the customer is working in a niche area, besides licenses, significant work is needed to provide a solution that fits the customer's request. Besides this, three years of support service are provided and one year of extended support for future rollouts. How many separate POBs should this contract have?

In this case, we're talking about three distinct POBs:

- SAP S/4HANA software
- Rollout support
- Regular service support

Even though the customer purchased two separate items (software license and customization service), the customer can't benefit from the license itself and needs a custom solution, so these two need to be bundled into one POB. Because rollout support is different from regular support, those two items need to be created as separate POBs.

The conclusion of this example is that proper POB identification is critical for proper revenue recognition under IFRS 15. Even the concept of POBs came only with IFRS 15; it's essential not only for proper revenue recognition in terms of amounts but also for timing of proper recognition. Again, besides clear business reasoning, there is also an impact on IT systems: the bundling and unbundling concept might require additional investments in the infrastructure.

1.2.3 Step 3: Determine the Transactional Price

The *transactional price* is the amount of funds the entity is expecting to get in exchange for transferring the promised goods or services. The entity allocates the transactional price to POBs to recognize revenue. As such, the transactional price is considered a net amount, meaning without any other amounts being transferred to third parties such as taxes.

As described, determining the transactional price can be very simple and straightforward, but there are some complex cases. In practice, complexities mainly come from one of the following groups:

- **Variable consideration**

In some cases, how much an entity can expect to get in exchange for goods and services isn't straightforward. If we're giving invoice discounts such as volume rebates, at the time when sales are made, we don't know if the customer will reach the volume needed to get the discount. When quantifying variable considerations, it's necessary to rely on

the probability that a certain event will occur. So, if the history with certain customers shows that, in most cases, they reach a threshold to achieve some discount, it's safe to say the same is probable and build it in the transactional price. Estimated probability isn't required, meaning an entity can use different considerations on a case-by-case basis.

- **Significant financing component**

Contracts do contain a significant financing component if payment from a customer happens much before or much after performance. This timing difference can benefit either the customer (if the vendor is financing the customer's purchase) or vendor (if the vendor is making payment before performance is fulfilled). If any of these cases occur, the entity needs to present its revenue recognized by fulfilling performance separately from revenue coming from interest.

Significant Financing Component or Time Value of Money

A significant financing component is often also called the *time value of money*. Time value of money is calculated if a contract has a significant financing component.

- **Noncash consideration**

Any consideration, even if not monetary, needs to be included when determining the transactional price. However, the key word in this case is *control*. If, for example, a company enters a steel-processing agreement where it needs to provide the service of making steel from iron, but it doesn't take control over the ore, the price of the ore can't be included in the transactional price. In this same example, if the vendor has control over the ore, the price of the ore (measured by fair value) needs to be included in the transactional price.

- **Consideration payable to customer**

These examples contain any amounts (cash or noncash) that the entity might pay to the customer in the form of incentive or rebate, which need to be considered when determining the transactional price.

To conclude, experience shows that even if determining the transactional price seems straightforward, there are often extremely difficult challenges. For instance, in the variable consideration example, it's necessary for IT and business teams to work together to establish feasible rules that can be fulfilled by the data in various systems.

1.2.4 Step 4: Allocate the Transactional Price

The next step is the *allocation* of the transactional price. Once the entity establishes the transactional price (considering all discounts and considerations), the same needs to be allocated to different POBs in the contract.

When determining how much of the transactional price needs to be allocated to POBs, the SSP must be consulted. The SSP is a key element of the IFRS 15/ASC 606 standard because it's used as a factor regarding which transactional price is allocated to POBs:

- *What* is allocated is determined by the transactional price.
- *How much* is allocated is determined by the SSP.

To put it simply, the SSP represents the value an entity can get for selling the same product or service without applying any hidden discounts to the customer for buying a bundle of products.

To understand the importance of the SSP, let's look at the example shown in [Table 5.14](#).

POB	Quantity	Transactional Price	SSP	Allocated Transactional Price	Allocated Transactional Price per Unit	Allocation Effect
POB 1	1,000	2,000.00	1,200.00	1,621.62	1.62	(378.38)
POB 2	1	-	500.00	675.68	675.68	675.68
POB 3	1	3,000.00	2,000.00	2,702.70	2,702.70	(297.30)
Total		(5,000.00)	(3,700.00)	(5,000.00)		

Table 5.14 Allocation of Transactional Price through the SSP

So, in this case, we have a contract with three different POBs. POB 1 has an agreed quantity of 1,000 pieces selling at 2,000, and the SSP is 1,200. POB 2 is being given to the customer for free, but its SSP is 500, which means that it's regularly sold for 500. POB 3 is sold at 3,000 units with a regular price of 2,000.

Here, we can see the main difference between IAS 18 and IFRS 15 revenue recognition. In the old standard, POB 1 would have been recognized by the amount it's billed: 2,000. However, this amount is adjusted by the ratio of the SSP for the specific POB to the total SSP for the contract, so it's getting 378.38 less revenue than before.

The opposite case occurs with POB 2 because it's given to the customer free; according to IAS 18, there wouldn't be any revenue to be recognized there. However, because IFRS 15 places importance on the value given to the customer by providing this product for free, revenue must be recognized there. Therefore, revenue is proportionally weighted and adjusted, so this POB is getting 675.68 units more.

Here, we can see the main difference between the old and new standards: the amount of revenue that is recognized is the same as always, which absolutely makes sense—you can't have more revenue than you're actually billing from the customer. But how much and where revenue is being allocated is completely different and is primarily determined by the SSP of each product and its ratio in all the POBs. This activity had a huge impact on certain industries.

Similar to elements discussed in these steps, determining the SSP can be straightforward and simple. When pricing is stable and clear, an entity can make a direct link between the costs of producing/procuring goods that are later sold, so determination of the SSP requires little to no effort. However, more often than not, you'll face different challenges in this area:

- The entity is selling the same product with a variety of different prices.

- There is no clear link between producing goods and their later sale.
- Goods sold are unique and don't have similar examples on the market.

In general, there are three possible ways to determine the SSP:

- **Cost-plus method**

This method is also known from the transfer pricing determination process, and it's based on determining the SSP as a sum of all costs related to production/procurement (manufacturing, labor, related costs) and adding some fixed margin on top. For example, a company is producing material X. The cost-plus method means that we take the material, labor, and common costs to come up with the costs of goods manufactured (COGM). To that, we can apply a fixed margin of 10% to get the SSP for product X. [Table 5.15](#) shows an example breakdown using the cost-plus method.

Direct Costs	
Material costs	1,000.00
Depreciation	300.00
Labor costs	400.00
Sum of Direct Costs	1,700.00
Indirect costs	100.00
Cost of goods	1,800.00
Fixed margin	10%
Total SSP	1,980.00

Table 5.15 Calculation of the SSP with the Cost-Plus Method

The cost-plus method is arguably the most popular way to determine the SSP, mainly due to its simplicity. However, this method has its own set of challenges. For example, an entity might have a case in which the same product can be sold to different customers with different margins (e.g., charging a higher margin to higher risk customers). In other scenarios, the method is very clear and fits entities that sell goods with no challenge to determine costs associated with production. In some other cases, that might not be the situation, such as companies where production is associated with R&D costs, so making a clear link between production costs and the SSP might be a challenging subject.

- **Market observable price**

In this method, we determine the SSP as the amount the customer would need to buy the product if they were to purchase it on the market. For example, a vendor is bundling routers a customer needs to use for their streaming service. Because the routers are being purchased from the supplier as part of some broader contract, we can't use the cost-plus method because costs can't be determined properly. In this case, we must look at the market to determine how much the customer should pay to obtain that router. Different criteria need to be considered such as position in the market, customer group (retail or wholesale), distribution channel, and so on. In this method of determining the SSP, a major challenge occurs when there is no similar product on the market to compare with to determine the SSP. In those cases, adjustments that need to be made might

result in providing a wrong or false SSP.

An example of the market observable price can be in industries where there is competition with the same or comparable products. In that case, a valid approach is to estimate the SSP by comparing the product with competing products.

- **Residual approach**

In this approach, the entity would deduct determined SSPs from the total transactional price, resulting in the SSP for the remaining POB. [Table 5.17](#) shows an example breakdown using the residual approach.

	SSP	Transactional Price
Bundle		500.00
POB 1	100.00	
POB 2	250.00	
POB 3	$500.00 - (250.00 + 100.00) = 150.00$	

Table 5.17 Calculation with the Residual Approach

In this example, we can see that the SSPs for the first two POBs are known and already determined. However, for POB 3, there is no clear method for how the SSP can be determined, so the entity applies the residual approach. From the total transactional price, the sum of already determined SSPs has been deducted. That result (residual) is the SSP for POB 3. Cases in which the residual approach makes sense are when there is high volatility in pricing (i.e., uncertain pricing), price interdependence when the SSP of one POB influences the SSP of another POB, or when there is a lack of data needed to determine the SSP. But it's important to mention that the residual approach is least favorable among others and should be applied only if there is no other way to determine the SSP of a product.

Besides determining the SSP, in practice, an entity might face more challenges when it comes to allocating the transactional price. One of these issues is discounts. *Discounts* are usually applied to certain items that belong to a bundle, but when it comes to IFRS 15, the standard is clear: even if a discount is applied to a certain item, the effect should be allocated to all POBs that belong to the contract.

To conclude, the determination of SSPs is crucial and possibly the most important activity when it comes to IFRS 15/ASC 606 compliance reporting. There are two aspects that need to be considered. First, businesses need to work out the preferred method for determining the SSP, as well as where and how it will be applied. Second, it's essential that this method is agreed upon with the IT team as well because it's crucial that data necessary for proper determination is available.

1.2.5 Step 5: Recognize Revenue

In this section, we'll elaborate on the last step, *revenue recognition*, which is when and how revenue can be recognized. Revenue can be recognized when a POB is satisfied, which happens when control over promised goods and services is being transferred to the customer. Transfer of control can happen as one of the following:

- **Point in time**

A certain event triggers revenue recognition fully or partially.

- **Over time**

A simple passage of time serves as a trigger for revenue recognition.

The most critical part of revenue recognition is determining when control has been transferred to the customer. The concept of *control transfer* is applicable for both goods and services; control is considered to be passed for services too even if they are being consumed instantly.

Control is considered to be passed if the customer has the ability to use and get substantial benefits from all or the remaining assets. For example, if a customer makes a prepayment, control isn't passed because the customer doesn't actually have control over the product. It's always advisable when evaluating control to look at it from the customer's perspective; this reduces the risk of recognizing revenue prematurely.

At the moment of contract inception, the entity must determine whether the POB will be satisfied at a point in time or over time. Application of over time revenue recognition isn't limited to services (which are usually considered) but also applies to some delivery or production of complex equipment or assets. In other words, it's important to look at terms and conditions of the contract to be able to clearly judge whether revenue should be recognized over time or at a point in time.

In general, for revenue to be recognized over time, one of the following three criteria needs to be met:

- The customer simultaneously receives and consumes benefits of a product.
- The seller performance significantly enhances the asset owned by the customer.
- The seller performance creates an asset that can't be used alternatively, and the seller has an enforceable right to receive payment for performance to date.

The first example is straightforward and is applicable for most of the services performed over time: The seller and customer are entering into contracts where the seller needs to provide a continuous service in a defined period of time. In that case, irrespective of payment terms, revenue needs to be recognized over time.

The second case is applicable for activities that are performed for a customer for enrichment or enhancement, but that are always owned by the customer. An example is when the seller receives a product from a customer that needs to be enhanced so it can be consumed by the customer. In this case, the seller only provides an enrichment service, and the asset itself never changes ownership. Essentially, this case isn't much different from the first one, so revenue can be recognized over time.

The third case is when a seller builds, for example, some asset that is made specifically according to a customer's specification and that can't have another purpose. The important part here is that the seller has an enforceable right for payment for work completed by that date. For example, say the seller entered a contract to build a specific asset exactly according to a customer's specification, and this asset can't fulfill any other need. If the seller has an enforceable right for payment done by the date (meaning if the customer cancels the contract along the way, the customer will still need to pay for work

done)—revenue can be recognized over time. If such a clause doesn't exist, then revenue needs to be recognized at a point in time.

Special over time revenue recognition cases are the percentage of completion (POC; i.e., measures of progress) scenarios. These cases are very important for some lines of business—such as construction, where satisfying performance happens over time, but depends on the percentage of work completed. The point of this revenue recognition method is that the amount of revenue being recognized fits the pattern that reflects the transfer of control of goods promised to the customer.

Methods for measuring progress can be grouped in the following two areas:

- **Output methods**

These methods measure revenue in correlation to value that has been transferred to the customer.

- **Input methods**

These methods measure revenue according to effort to satisfy a POB.

Point-in-time revenue recognition can be used if none of the criteria for over time recognition have been met. There aren't clear rules, but rather indications when point in time recognition can be used, and they are related to control: if the customer has control over the POB, the obligation has been satisfied, and revenue can be recognized.

In practice, different challenges might make determining when revenue can be recognized a bit less obvious. It's no surprise that those cases are related to ways in which it can be proved that the customer has control over goods. For example, say a customer is making a contract with the seller to buy certain goods. The seller is paying in advance, and there is a certain time period needed for goods to reach the client's warehouse. When can sellers recognize the revenue?

The answer is that it depends. When selling goods that require transportation to reach the customer, the contract specifies terms—called *incoterms*—regarding the responsibilities of the seller and buyer. If the incoterm used is, for example, Delivered at Place (DAP), the customer becomes the owner of the goods at the moment of delivery to their storage location. So, control hasn't been passed until goods reach their final destination, and revenue can't be recognized prior to that.

In conclusion, when the new standard was first published, the five-step model was fresh and required a lot of studying to be fully understood. Some areas, such as transactional price determination, weren't really new, and companies could adjust to it without a huge effort. Some other areas were different, and it required adjustments both to be understood and to be applied. After years of having the standard around, all these terms should now be known. However, it needs to be repeated that the way they are applied might be different depending on the scenario, so deciding how contracts will be created or the method behind the SSP determination is still of utmost importance.

Now, maybe more than ever, it's important that both business and IT teams work hand in hand and do their best to help each other. IT needs clear rules about what needs to be provided, and business teams need IT's help to find the data for use in satisfying requirements.

1.3 Industry Impact

The introduction of IFRS 15 meant significant change in terms of how revenue is recognized for almost all entities operating and presenting financial statements. However, the impact varies and is highly dependent on the industry in which an entity operates.

As we can see in [Figure 1.2](#), the impact might be small to none for industries where invoice-driven revenue recognition is the norm. For example, in the retail industry, a customer purchases a product, pays for it, and leaves. In general, revenue recognition for those cases is very straightforward.

On the other end of the spectrum, some industries have arrangements with multiple elements (bundles) and with that, the structure and number of bundles becomes even more complex (e.g., telecommunications). In addition, some industries have complex contract structures with multiple customers and multilevel discounts where arrangements last a long time. These industries are heavily impacted by IFRS 15 changes, primarily because to achieve correct revenue recognition, unbundling POBs needs to happen. The second reason is that here we're talking about industries with a massive number of transactions, and, besides unbundling, these rules need to be applied automatically.

And, of course, there are industries with a complex product structure, such as professional services or software companies where there is a mix of products where revenue

needs to be recognized both over time and at a point in time.

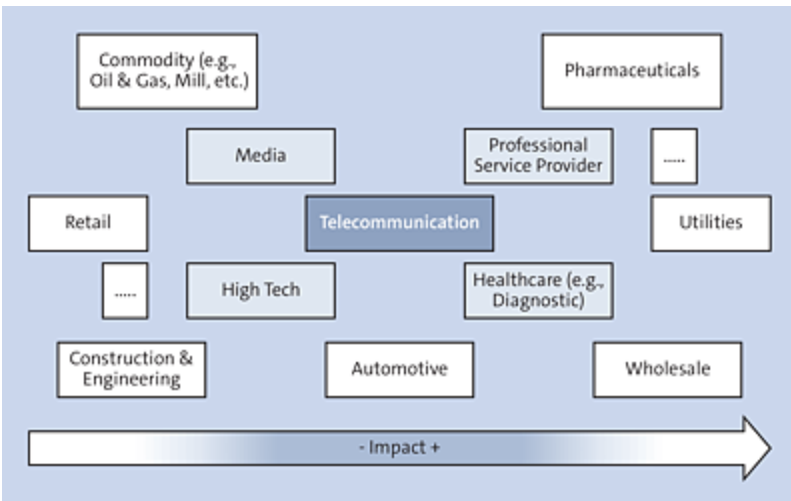


Figure 1.2 Industry Impact of the IFRS 15 Implementation

In the following sections, we'll explore the impact of IFRS 15 changes on different industries with specific attention on those with the biggest challenges. This information will be useful for you whether you're trying to solve some specific scenarios from the mentioned industries or are just starting by setting the landscape.

1.3.1 Telecommunications

Telecommunications has always been one of the most fast-paced industries. Operating in a highly volatile environment surrounded by constant technological changes, government regulations, and frequent industry disruptors, companies were always forced to innovate to survive—from landline to mobile and from voice to data focus. So, what changed recently?

Today, telco companies are experiencing the perfect storm. First, there's technological pressure that makes it difficult to keep the pace: 5G is slowly becoming a standard, and there is a whole bag of technology advances that need to be used to stay relevant in constantly competing markets (blockchain and artificial intelligence [AI], to name a few). Second, government pressure isn't decreasing; on the contrary, lists of standards and regulations that companies must fulfill are growing every day (data privacy, to name one). So, companies need to invest in both infrastructure and develop new processes and standards to adhere to those regulations. And, third, there are completely new kids on the block—internet companies—acting as disruptors.

In only five years, the market share between internet companies and classic telco operators completely changed (see <http://s-prs.co/v570003> for a detailed breakdown). Eight years ago, telcos were still key players. They looked down from above at all the companies operating on the same playground, but saw them as necessities, not competition.

However, in just five years, the picture was completely turned upside down. Now telco operators seem like followers to internet companies, which are in turn called giants. In addition, there is a completely new set of competitors now operating in the internet space and increasing this pressure even further.

So, what is the issue here? Well, none of the growth of internet companies would be possible without the infrastructure provided by telco operators. In return, however, the telco operators didn't gain much. Plus, internet

companies started entering areas that were historically telco services, such as voice.

Don't forget, this occurred before the COVID-19 pandemic. COVID-19 created the need for jobs and homes to integrate seamlessly. So, it's to be expected that telecom operators today are under more pressure than ever before.

This situation was a call for action. Many operators started digital transformation journeys that resulted in an overall decrease in costs. Overall, the shift in business model also required investments in improving customer experience and improving time to market for new products. All these changes in the business model triggered changes in reporting.

We'll look into a few key ways that the telecommunications industry is managing this new landscape, as well as the IFRS 15 requirements, in the following sections.

Integration and Data Volume

The main issue that requires attention when working with telecom operators is integration with different systems. Historically, telcos started as providers of fixed line telephony; with the development of mobile services, they merged with mobile providers.

In addition, telco products are becoming more and more complex. In the beginning, it was simple; customers were paying for services offered through cable telephony. Later, mobile business brought additional complexity by combining devices (mobile phones, routers, etc.) with service. At the end, with the expansion of the internet, we got products

based around data, such as internet services and broadband networks. Therefore, customers now expect one bill that contains everything, even possibly within a single line.

Convergence in Telecommunication Companies

Convergence in the telecommunications industry means that eventually all voice and data services will belong to a single data stream. There are a few factors pushing for convergence, but most important is competition from new companies. The first example of convergent products was the merging of data and voice into a single revenue stream. Later, we might see new products in addition to voice and data that will include buying digital products or subscriptions to streaming services—all bundled in one.

To support such a business model, information systems of telco operators became very complex. On one hand, you have different customer relationship management (CRM) (often handling different business streams), billing, and charging systems. On the other hand, it's required that they all work together to provide data for proper IFRS 15 reporting.

[Figure 1.3](#) shows a very simplified view of systems that are involved in regular business operations in any telecom provider. Multiply these for mobile and fixed telephony, as well as involve convergent products, and the complexity becomes clear.

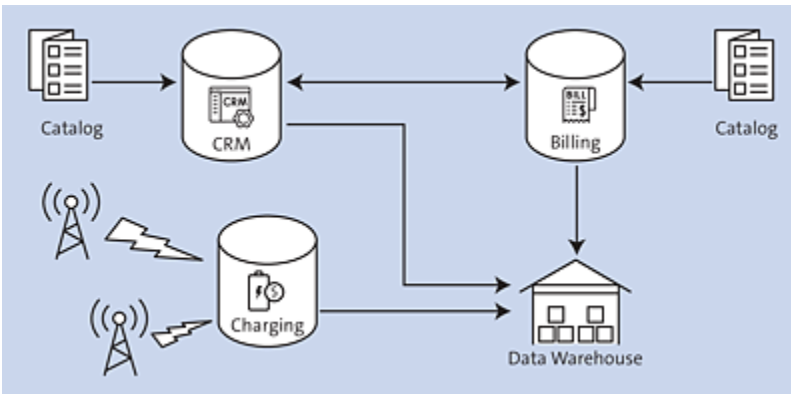


Figure 1.3 Landscape of Business Support Systems in the Telco Industry

Business Support Systems

Business support systems (BSS) are the components that a telco company uses to run its business operations for customers. Together with operations support systems (OSS), they are used to support various end-to-end telecommunication services. BSS systems usually consist of a product catalog, CRM system (which focuses on customer and partner management), revenue management system (often called billing), and order management system.

Integration is key when it comes to IFRS 15 reporting in any telco company. It's necessary to identify all data points needed to satisfy all five steps of the IFRS 15 model. However, more often than not, you'll find that some data doesn't exist or the data requires multiple sources to be created/compiled. In those cases, it's very useful to have a tool that will be placed between the source systems and the revenue accounting engine to prepare data later used to satisfy the five-step model in IFRS 15. This kind of tool is usually called a data hub, and it refers to an extract,

transform, load (ETL) product that's used to transform and enrich data before it's loaded for IFRS 15 calculation.

What Is ETL?

In a nutshell, ETL tools are used when there is a need to enrich or transform data before it's loaded into its final source. Both extraction and transformation are multistep processes where data is retrieved from external sources, cleaned, and transformed into the needed format. ETL software typically runs automatically on the basis of scheduled jobs, but there are more and more tools that can run in real time.

Among the many ETL products on the market is SAP Data Services. It comes with plug-ins that make it possible to work with most data sources used.

This kind of setup enables a significant reduction of risk when it comes to transforming data coming from different sources. Keep in mind that even basic data needed for fulfilling IFRS 15 requirements might be scattered in different data sources and require significant time to compile so that it can be used for allocation calculation.

This problem becomes even bigger and clearer when it's compared with the fact that the necessary data isn't usually ready before month end (billing runs are done once or, in the best case, a few times a month) and that the team preparing IFRS 15 reporting has limited time to prepare. Bringing an ETL tool into the picture also makes sense when used on top of an SAP HANA database.

Contracts and Performance Obligations

As mentioned in the previous section, one of the biggest challenges in the telco line of business is data volume, which is a determining factor on whether an entity will be able to successfully perform tasks for IFRS 15 reporting. Data volume should also be a considering factor on how contracts will be created and how granular POBs will be created in contracts. In the earlier days of RAR, it was strongly recommended that a certain number of POBs shouldn't be exceeded for the tool to work properly.

[Figure 1.4](#) shows a simple telco contract with just three POBs: one for the device that will be recognized as point-in-time revenue, one for the service that is going over time, and one for a one-time installation fee.

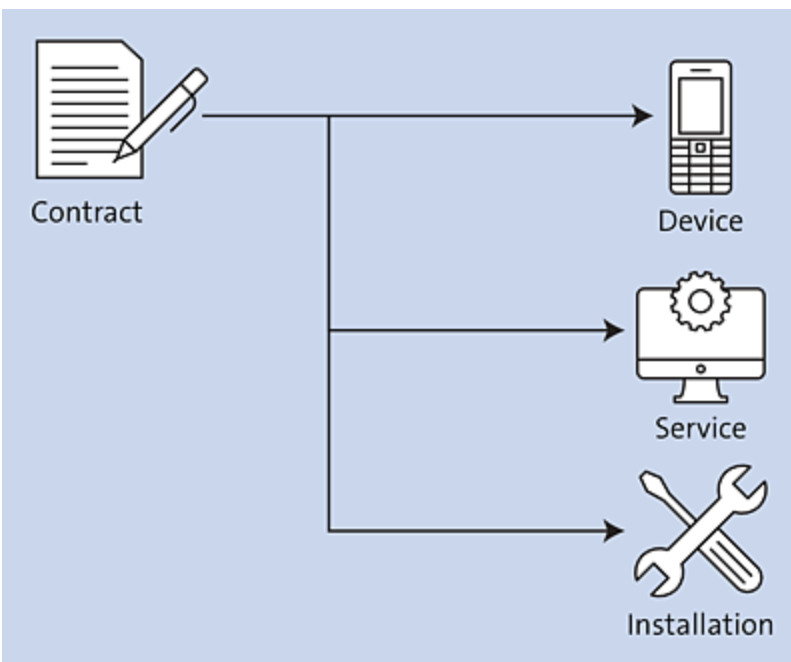


Figure 1.4 Example of a Simple Telco Contract

This simplified method makes data that is loaded for IFRS 15 calculation lightweight so that RAR is being kept very lean,

which results in give good processing time.

On the flip side, there might be a need for using additional analytical tools for detailed reporting on a more granular level, as shown in [Figure 1.5](#).

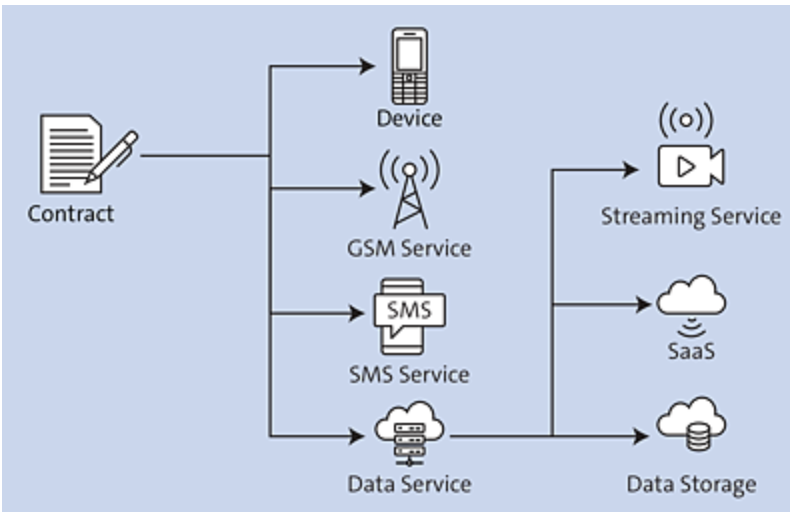


Figure 1.5 Example of an Expanded Telco Contract

In this case, we're creating POBs on a more granular level. We're splitting the data service being provided to the customer into the following:

- Streaming service provided to the customer
- Software as a service (SaaS; e.g., Microsoft 365 subscription)
- Data storage on the cloud

Remember, if we can sell these services separately to the customer and the customer is clearly benefiting from them, these can be created as separate POBs according to step 2 of the revenue recognition model (refer to [Figure 1.1](#)). So, in this case, the same contract has seven POBs instead of three (and, if we include an installation fee, that makes

eight POBs). This multiplies the process by almost 3, and if it's then multiplied by several million contracts (which isn't uncommon in telco companies), it's clear that data processing time will definitely increase. Of course, this is then multiplied by the number of jobs that need to run to fetch the data from sources, and so on.

The last point to mention here is related to contract modifications. Remember, a contract isn't a document carved in stone: customers will often come asking for additional services, prolongation, additional devices, and so on. So, the number of contracts processed every month will increase as well for the number of contract modifications that need to be processed.

Note

Whether a service is being charged or not doesn't determine its creation as a POB. Telco companies often bundle products such as streaming services or cloud space, which aren't separately charged to customers. However, they do have a certain value, thus they can't be simply skipped from either creation as POB or including them as part of the service (thereby increasing its SSP).

Prepaid or Not Prepaid

In the telco industry, you'll find two major types of contracts (especially in the Global System for Mobile Communications [GSM] market): prepaid and postpaid. *Prepaid contracts* or "pay-as-you-go" are contracts for telco services where the customer pays for credit and later consumes it as different

telco services (voice, data, SMS). Once the credit is consumed, the continuing services of outgoing calls, sending messages, or accessing data are denied.

Postpaid contracts are the opposite: the customer consumes a service and pays for it at the month end. As the fight for the market became more and more fierce, companies were trying to get more consumers and make them loyal. To do so, they focused on the postpaid market and created bundles that included devices and different free items, just to bind the consumer to a longer contract period. When we talk about IFRS 15's impact on telco companies, the postpaid contract is a classic example.

However, the prepaid contract had an evolution of its own. In the beginning, you simply paid a credit and used it for any service. At the month end, the credit would either simply disappear or would be rolled to the next period. But in the modern telco environment, even prepaid contracts are becoming more complex: they also include freebies, special packages that favor one type of service, loyalty points, and so on.

So, should prepaid contracts be considered in an IFRS 15 project? The answer is yes. However, how and to what extent depends on the specific case. If a company is doing a majority of its business in the postpaid world, it would make sense that the prepaid market is treated with the portfolio approach. In that case, the market will be sliced into several typical contract types, and it will have a portfolio created for each.

In other cases, if the prepaid market is a major part of the total telco services sales and many different products are

included, it would make sense to consider the contract-by-contract approach. Similar to postpaid contracts, the company would need to identify data points that are needed for creation of contracts, POBs, and all relevant data needed for proper revenue recognition. Here's the first challenge: data sources for the prepaid and postpaid market are often not the same. Postpaid cases are focused on CRM and billing systems as the main sources to identify the customer, products, and prices. In prepaid cases, data coming directly from the network often needs analyzing, which means a much larger data volume to work with. In those cases, the ETL tool approach used for data cleansing is becoming a must, rather than just a recommended approach.

In conclusion, for the prepaid scenario, the business and IT departments need to work together to determine how requirements are going to be fulfilled.

Additional Complexities

Complexities in the telco space also appear in other areas that are worthy of mention. Let's walk through them:

- **Time value of money**

As mentioned before, in cases where a customer pays significantly before or after goods or services are provided, we're talking about a significant financing component, or the contract has the time value of money. An example in telco is the sale of devices on installments. Customers often get a device and pay for it in installments. This process fits the description because the user gets goods immediately and pays for them over a specific period of time.

However, the economic climate was such that interest rates were very low and the financing impact was minimal. But changes are happening because interest rates are increasing, and the impact of the financing component might become significant again. Therefore, it's recommended that the entity reevaluates if there is a change in impact of the financing component and then goes on to implementation if necessary.

- **Variable consideration**

In terms of IFRS 15, under variable consideration, we include all future discounts or rebates, incentives, or returns that are highly probable to occur under validity of contract. In terms of the telco line of business, variable considerations are all future changes that might influence the transactional price. In other words, if the company expects that some part of the transactional price agreed upon by the customer might need to be returned, it will need to recognize the liability for the very same amount. The key word here is *highly probable*, meaning the company will need to assess how likely are events to occur that might influence consideration from the customer.

An example of variable consideration is if a company is selling its services to different customer groups, and some of these groups are at more risk than others for its receivables to become nonpayable. In those cases, the company would need to assess that probability and adjust the transactional price accordingly.

Once more, it's essential that IT and business teams work together in finding a proper solution. To make such an

assessment, data would need to be available and given to the business so a proper solution could be found.

- **Loyalty points**

Let's assume that our telco company has a loyalty program that gives 1 loyalty point for every \$100 worth of service used. This point can be redeemed for the purchase of different services, and 1 point is worth \$0.1. Based on analysis, we expect that 90% of loyalty points are redeemed and total sales of services within the period is \$100,000.

First, we need to calculate the SSP of loyalty points: $\$100,000 \times 0.1 \times 0.9 = \$9,000$. In the total transactional price of the contract, now we need to include the value of loyalty points: $100,000 + 9,000 = \$109,000$. Now, we can calculate the allocation:

- Loyalty allocation: $100,000 \times (9,000/109,000) = \8.257
- Service allocation: $100,000 \times (100,000/109,000) = \91.743

This very simple case shows us the impact of loyalty points on total revenue allocation. In addition, data analysis is needed to set up proper rules to provide results that fit reality. Here, we need to estimate the redeemed rate of loyalty points and the relationship between goods or services and points given to the customer. More detailed analysis would potentially give different results: for example, if customers are using points to buy goods and not services, then allocation might be linked to event-based revenue and not time-based revenue.

1.3.2 Life Sciences

The life sciences industry sector comprises companies that operate in the research, development, and manufacturing of pharmaceuticals, biotech medicines, medical devices, and other areas that aim to improve the lives of living beings.

Medical device companies work on the development of medical devices (e.g., instruments, machines, software, or a combination of these) used to diagnose, prevent, and treat different medical conditions.

When we're talking about the diagnostics sector, there are two major areas of diagnostics:

- **In vivo diagnostics**

In vivo diagnostics is a type of diagnostics where tests are performed on the body itself rather than on isolated parts of it. Examples of in vivo diagnostics are different types of scanners or imagery devices that can diagnose without taking any samples from the body.

- **In vitro diagnostics**

In vitro diagnostics are tests that can diagnose conditions and infections based on samples taken from the body. Contrary to in vivo tests, in vitro tests are performed in the laboratory based on samples taken from the patient. When talking about diagnostics, they include a variety of products, starting from the instruments to the reagents that are needed to compile the results of testing or a diagnosis.

Processes in diagnostics companies are usually about bundles. One bundle will include *instruments* (used to get analysis results), *reagents* (used to perform analysis), and

services (installation, maintenance, extended warranties, etc.), as shown in [Figure 1.6](#).

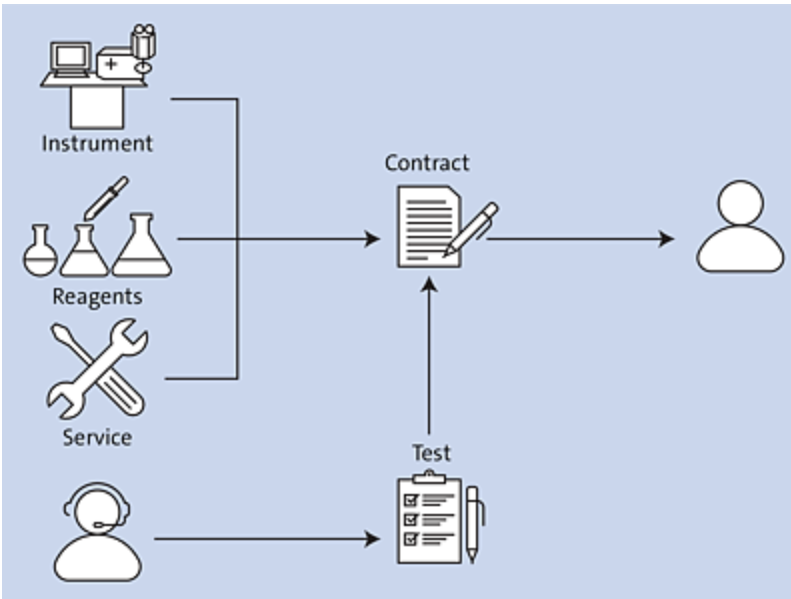


Figure 1.6 Process in the Diagnostics Industry

Usually, purchases that the customer makes are bundled in the contract together. One contract for a medical device usually includes an instrument, different reagents that are needed for that instrument to operate, and services for installation of that device. In addition, it's possible that the contract includes additional services such as ongoing maintenance or warranty, call center assistance, and so on.

Contracts usually have a duration, but it's not uncommon that a company has perpetual agreements with the customer. Perpetual contracts, though not uncommon or wrong, are an additional complexity in this industry because it's difficult to identify clear rights and obligations in those cases.

In practice, we see two different types of contracts:

- **Capital sales contracts**

In these contracts, the customer buys goods and then owns them. The instrument is delivered and installed to the customer premises, and once it's activated, the customer can start using it.

- **Lease contracts**

In these contracts, the instrument isn't owned by the customer but remains owned by the seller. In those cases, goods usually stay in some sort of consignment stock and are never owned by the customer. Here, the customer is using the benefits of the instrument over a period of time—usually equal to the duration of the contract.

Reagents are also goods needed to operate instruments. Medical devices are able to perform analysis and deliver diagnoses by using these reagents. Reagents are sold or given for free depending on the nature of the deal. Usually, if a customer leases an instrument through a lease contract, the price of the instrument is represented through reagents. This is also true the other way around: if an instrument is bought as a capital sale, reagents are sold with a discount.

However, the most common way of billing the customer is through a number of tests. The customer uses both the instrument and reagents to perform tests, and the company charges the customer for the number of tests they run. At month end, the number of tests run is reported, and the bill is sent to the customer.

Therefore, it's clear that the main challenge when implementing IFRS 15 in diagnostics companies is process complexity. Even in this most basic example, we can see that the process is complex with many potential variations.

Therefore, it's essential that the company performs proper analysis of its processes and standardizes them as much as possible across different subsidiaries. Each process variation during a project can be another problem requiring special attention to be solved.

In the following sections, we'll walk through how the life sciences industry handles IFRS 15 requirements with specific examples.

Identification of Contracts and Customers

Contract identification might turn out to be easier said than done for diagnostics companies. Often there are multiple parties involved: distributors, chains of hospitals, and hospitals to which instruments are actually being delivered. Additional complexity can appear if there are multiple parties involved in research and development of the device itself.

In addition, options are often provided to the customer regarding canceling part of or the whole agreement, different discounts given, right of return, or other provisions. These options can really change the sense of the agreement, so it's crucial to look at all amendments that are part of the original contract to draw a proper accounting conclusion.

Again, one of the implications to consider is the IT impact. If a contract is set on a very high level—meaning it will contain a large number of POBs—the natural implication is that the same contract will go through a number of modifications during its lifetime. Modifications will always

trigger recalculation of the whole contract, which might yield unexpected results.

Identification of the SSP and Transactional Prices

As we clarified at the beginning, the SSP is the price an entity could get for its product if the product was sold without bundling it with some other products. We already mentioned ways the SSP can be determined, but all of them have additional complexities once applied to the diagnostics industry. Because diagnostics companies are often global, applying the standard cost-plus method might not be that straightforward.

In the example shown in [Figure 1.7](#), the company is operating in different regions and applying different margins per region to cost of goods sold (COGS) to arrive at the regular price for the customer. For example, in the Asia-Pacific (APAC) region, because there is fierce competition, the company decided to apply a lower margin to its product to improve its position in the marketplace.



Figure 1.7 Different Margins in the Regions

In this case, it's important to evaluate first whether the cost-plus method is most suitable for determining the SSP of a

product. But again, there may be no similar product on the market, so the market price can't be used, or some other complexities may make determination of the SSP even more difficult.

Either way, the only answer to this question is that the company aligns with its audit company on time and resolves this question before adoption of IFRS 15 starts. As mentioned before, it's essential to align with the IT team to evaluate what data is available so it can support the selected method.

Leasing Process and Integration with IFRS 15

As mentioned previously, one very common way of selling in the diagnostics industry is to lease medical instruments rather than sell them to customers. In this case, two separate issues appear.

The first problem is the complexity of the process. [Figure 1.8](#) shows an example of how goods can be transferred between different storage locations when performing lease sales.

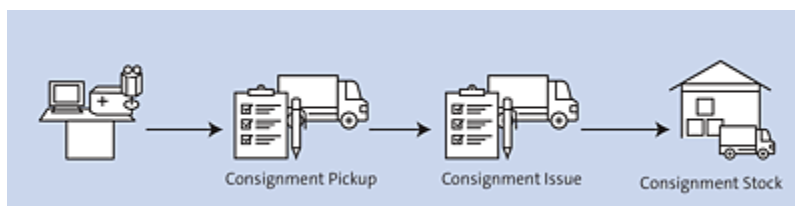


Figure 1.8 Process of Lease Sales of an Instrument

You can see that the first process is that we need to perform goods issue from stock. Because goods are being transferred to a different storage location, we need to create a sales order first with a consignment pickup document.

Based on this document, the goods movement is performed. After that, once the instrument is ready for the customer, we're creating one more sales order to represent issue from consignment, which will be followed by one more goods movement to a storage location that represents the instrument is actually located at the customer premises.

This process is rather complicated because, in terms of IFRS 15, the first question is when do we consider that control over goods is transferred to the customer so revenue can start to be recognized? Most likely, that would be the last step, which is the moment of transfer to the customer location, but because there is a whole chain of documents prior to that step, it's a challenge to identify the exact moment revenue recognition is triggered.

The second challenge is billing: Based on which document is billing actually created? Is it in the last order, or can billing be created in some other document? Remember that this process can be covered in multiple documents—separate orders for the instrument, for services, and for reagents—and all of them are bundled in one IFRS 15 contract. Therefore, a proper document for billing needs to be identified as well.

Because this is a lease contract too, the company also needs to identify the fair value of an instrument. Again, it's not uncommon that the instrument itself is given for free, so the entity would need to make sure the lease value is represented as the fair value of the product. In addition, when both operating leases and financing leases are in play, these need to be integrated with the fixed assets register.

In all of these cases, it's clear that the process for instrument sales is often very complex. It's essential that besides unifying the processes as much as possible, the entity has all the right tools in place to represent them in their information system so proper accounting treatment can occur.

1.3.3 Manufacturing

IFRS 15 also has a huge impact on contracts in manufacturing industries. The main issues for this industry are as follows:

- **Definition of POBs**

The manufacturing industry now has new POBs that all will need to be accounted for separately. It's not always easy and straightforward to define POBs as separate obligations, especially with bundles.

- **Recognizing revenue**

In the classic manufacturing industry process, once goods were delivered, revenue was recognized. However, as mentioned before, IFRS 15 enforces a rule of control: revenue can be recognized only at the moment the customer actually controls the product. This also goes together with cost recognition because costs need to be recognized at the same period as revenue.

- **Contract combination**

The practice of combining contracts didn't exist in the past as it is now with IFRS 15.

All of these issues (and more) require that special attention is paid once the IFRS 15 adoption project is in place. As

usual, we're talking about whether revenue can be recognized at a point in time or needs to be recognized over time. To provide proper accounting treatment, we need to look at each particular contract. If, for example, a company is making a specific product that can be used by this customer and the company has the enforceable right to be paid for it, we're talking about *over time revenue recognition*.

Over time recognition comes with its own set of challenges. Revenue often can't be recognized in a linear fashion; thus, progress or POC needs to be calculated. In IFRS 15, there are two accepted methods:

- **Input costs method**

In this method, we're recognizing revenue based on the costs incurred such as resources, machine hours, labor, and so on. For example, in [Table 5.22](#), we're working on a project where we evaluated total costs to be \$100,000 and billing price to be \$150,000.

	Costs (\$)	Revenue (\$)
Total	100,000.00	150,000.00
Up to Date	35,000.00	52,500.00

Table 5.22 Calculation of Revenue Recognition Based on the Input Method

Here, revenue is recognized proportionally based on costs incurred. So, because we have 35% of costs, we're recognizing 35% of the revenue.

- **Output costs method**

Opposite to the input method, the output method is based

on measuring the outputs of the process. For the same example shown in [Table 5.22](#), if we agreed to produce 1,000 pieces for \$150 each, and so far, we've delivered 300 pieces, the revenue calculation would look like [Table 1.5](#).

	Quantity (\$)	Revenue (\$)
Total	1,000.00	150,000.00
Up to Date	300.00	45,000.00

Table 1.5 Calculation of Revenue Recognition Based on the Output Method

Regarding point-in-time revenue recognition, it also has its fair share of challenges. The main problem is the proof of delivery process.

As mentioned earlier in the chapter, incoterms are rules defined by the International Chamber of Commerce (ICC) that are used to define obligations and rights between the seller and buyer. Incoterms define how the costs and risks are allocated between parties in the contract.

Most illustrative in our case are the following incoterms:

- **Ex Works (EXW)**

The customer becomes an owner of goods. once they are produced. The customer bears all the risks from the place of production to the final destination.

- **Delivery at Place (DAP)**

DAP means the buyer becomes the owner of the goods only once those goods are delivered at their location.

Different incoterms influence ownership over goods (for a complete illustration, see <http://s-prs.co/v570002>). In a simple case, ownership is passed when goods leave the warehouse, whereas in some others, it's when the customer receives them. This can play a significant role, especially in cases where the location of the customer and vendor are far apart.

The challenge for proof of delivery is clear: depending on the incoterm selected, the revenue will be recognized in a completely different time compared to the invoice issue and goods issue process. In addition, costs related to revenue need to be recognized in the same period as revenue.

To solve this problem, we need to include incoterms used while deciding when revenue is actually going to be recognized. If we're using an incoterm which states that even after invoicing and delivery, risks and benefits still aren't transferred to the customer, we'll need to have both revenue coming from the invoice and COGS coming from delivery to be deferred. Once we get proof or confirmation that risks are transferred to the customer, we'll reverse that deferral and post it to recognized costs and revenues.

The previous examples show that although revenue recognition is probably simpler in manufacturing than the diagnostics or telco industries, complexity also exists in the manufacturing industries. It's important to perform proper processes and contract analysis to determine the most suitable and correct accounting treatment of revenue recognition.

1.4 Revenue Recognition and SAP

As mentioned at the beginning of this chapter, revenue recognition tools have been available in SAP for a while. Prior to IFRS 15, the tool for performing revenue recognition processes was also called revenue recognition in SAP ERP, but it was placed directly under SD as part of the SD-BIL-RR component. This solution was aimed at solving the main issues of timing of billing and revenue recognition before the introduction of IFRS 15. For that purpose, in SD-RR (i.e., revenue recognition), we had the following main methods for how revenue should be recognized:

- Revenue recognition at the same as billing
- Time-related revenue recognition (revenue recognition between a set of dates)
- Service-related revenue recognition (revenue is recognized based on some specific event)
- Credit/debit memo with reference to a preceding document
- SAP for Media-specific method related to service revenue recognition, but with more of a focus on royalties, for example

The setup of SD-RR was straightforward and because all data would be taken from SD; it was embedded in the standard SD setup, especially when considering the accounting impact.

However, with the introduction of IFRS 15, the concept of SD-RR was abandoned, and a new tool was introduced to meet the requirements from the new standard—SAP Revenue Accounting and Reporting. This new tool was designed to not only solve challenges that came with the new standard but also make significant improvements to the old solution:

- The order/invoice processing and revenue recognition processes were decoupled. This approach alone brought significant flexibility in environments where order management and billing processes are natively in non-SAP environments (e.g., refer to the telco example).
- The second improvement came directly from the first one: SAP Revenue Accounting and Reporting is much easier to integrate with external systems.
- Native support for multiple reporting standards (IFRS/ASC 606/GAAP) was included.
- It was built around the five-step model of IFRS 15. This means that allocation was introduced as a concept because it didn't exist in the old revenue recognition solution.

The functionality of the standalone SAP Revenue Accounting and Reporting solution has been included in SAP's latest ERP suite, SAP S/4HANA. We'll refer to it as RAR throughout this book.

However, SAP now positions two solutions in the market:

- RAR with classic and optimized contract management (CCM and OCM)

- Event-based revenue recognition (EBRR)

Although this book is based on RAR as the primary tool for revenue recognition, we'll discuss both of the preceding solutions at a high level to shed light on which solution is most appropriate for certain business cases.

1.4.1 Revenue Accounting and Reporting

RAR is the main tool for revenue recognition in both SAP S/4HANA and SAP ERP to satisfy IFRS 15/ASC 606 requirements. There was no single product available that would comprehensively deal with revenue reporting prior to RAR. When integrated with sales and distribution, SD-RR was the go-to solution, but mainly when companies needed to differentiate the timing of revenue recognition. Results analysis was a solution for calculating revenue recognition in complex project environments. As a solution, results analysis has been available on the market since mid-2014; it went through several iterations of changes to become ready to support customers moving to IFRS 15:

- SAP Revenue Accounting and Reporting 1.1 came with solutions that had full integration with sales and distribution. The Adapter Reuse Layer (ARL) concept was introduced together with the split between time-based revenue recognition and EBRR.
- SAP Revenue Accounting and Reporting 1.2 was briefly on the market and was quickly upgraded with the next versions. This version brought improvements in terms of account assignment options and integration with cost object controlling.

- SAP Revenue Accounting and Reporting 1.3 is the latest solution available for implementation. Further development is being done through support packs (SPs); at the moment, SP 15 is available. With different SPs, many additional features were introduced, such as new fulfillment types (proof of delivery, drop shipments, call-off orders), but the main advance was in terms of stability and an error correction capability.

As of SAP S/4HANA version 1809, RAR became an integral part of the SAP HANA foundation, which also includes software component version REVREC. Changes between the 1.3 version and the one on the SAP HANA foundation were significant in terms of additional features and improvements to the code, but the functionality itself remained mainly the same. The exception was the introduction of OCM and optimized inbound processing (OIP; with version 1909). Although some people are calling SAP HANA version 1.4 or even RAR 2.0, this naming convention is colloquial, not official from SAP.

NUMC to CHAR

One very important change that came with versions available for SAP S/4HANA is the change of data type for POB ID. Basically, until SAP HANA, the data type for POBs (FARR_D_POB-POB) was numerical. Due to performance improvements, this was changed to characters. No additional migration considerations are needed except to revise any custom code that might relate to the old data type. Details are given in SAP Note 2672794.

Figure 1.9 shows the high-level architecture of RAR.

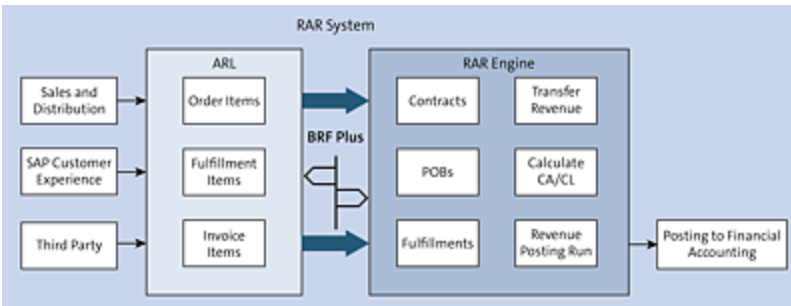


Figure 1.9 RAR Architecture

As already mentioned, RAR is a flexible tool that can easily be integrated with different systems used as data sources. Native integration for RAR is available with some SAP products such as sales and distribution, SAP Billing and Revenue Innovation Management, and SAP Customer Relationship Management (SAP CRM). All these systems are used as a source for getting data which is later grouped into three categories: order items, fulfillment items, and invoices.

Order items will have all the information needed for contract creation:

- Basic data such as customer, start/end date of contract, and so on
- Conditions used to determine both the transactional price and the SSP, which will be used to allocate the transactional price

RFC Integration

To integrate with sales and distribution and other components, RAR uses the remote function call (RFC)

functionality as part of the inbound processing configuration (see [Chapter 3](#)). For proper integration, there are two options: either defining and using proper logical systems, or using logical destination NONE only after implementing SAP Note 2957643.

In addition to SAP tools, there is an option to integrate with third-party solutions. The requirements are the same as with SAP solutions, but special attention needs to be paid regarding data quality. The data coming to RAR must be in the same format and quality as if it was coming from an SAP solution.

Once data is sent from the source system, it reaches the ARL. However, once received, it's not used immediately to create contracts and POBs. Instead, data is used to create revenue accounting items (RAIs), which are used as placeholders for data before the data can be used further. The reason for this is simple: once a contract is created, many of the standard RAR data tables are updated, and a revenue schedule is generated. In total, we're talking about potentially more than a dozen tables being updated, and, for time-based POBs, potentially many lines being created for the revenue schedule. Not to mention that if the contract needs to be combined with some already existing contract, the modification process is also triggered. All of these reasons and more led SAP to this RAI approach, where an initial data quality check is performed before contracts can be created. Here, SAP offers different statuses that should enable even more control over data quality.

Once data is processed in ARL, it's checked against rules in Business Rules Framework plus (BRFplus).

Business Rules Framework Plus

BRFplus is a tool used to define rules without needing to write ABAP code. It's not a new tool and was first available with SAP S/4HANA release 1610. For example, if you think about writing validations or substitutions in finance, more often than not, you need to invite your ABAP programmer to write some code, followed by the whole substitution needing to be regenerated, which requires special transport methods. The idea of BRFplus is that the same can be achieved without writing a line of code by just using decision tables. Usage of BRFplus is gaining traction recently with the move to BRFplus for configuring output management in sales and distribution.

BRFplus comes with different applications that are used to do the following:

- Determine POBs and POB types
- Determine rules for changes in POB statuses
- Perform account determination

RAR comes with predelivered applications that are dependent on the type of integration the customer is performing (SAP Billing and Revenue Innovation Management, sales and distribution, or third party). To use these applications, they should be copied to the customer's namespace and adjusted to fit the customer's specific needs.

Once RAIs are processed, the main RAR objects are created or updated—depending on the RAI type.

If the order is created in sales and distribution and processed as an equivalent RAI type, a result contract and corresponding POBs are created. If we processed a fulfillment item or an invoice, tables maintaining such values will be updated together with tables maintaining data for revenue calculation.

So, we have contracts created/updated and that means we're ready for the month-end closing process. In RAR, that refers to running *ABC programs*. There are three programs (explained in detail in [Chapter 6, Section 6.2](#)) that need to be run so the results of RAR processing can be seen in financial accounting.

Running ABC Programs

Customers often have questions about how often programs for calculating and posting revenues should be run. Of course, if reporting of revenue needs to be made more often, then ABC programs should be run more often because data in finance will land only once and be calculated by RAR. Some customers are happy with running the programs once a month, but others opt for running more often. Month-end closing in RAR isn't possible if contracts are in error (without moving them to the next period). So, running ABC programs once a month may not provide enough time for solving potential issues. On the other hand, running ABC programs in extremely short intervals can cause deadlocks between programs. So, each customer is unique (it's not the same if contracts are made primarily of event-based or time-based POBs) and needs to determine the best option while considering

the potential problems that come from running programs too often or not often enough.

Optimized Contract Management

As mentioned, RAR as a tool has been on the market for close to 10 years. Its flexibility and performance were really tested during the introduction of IFRS 15. For example, it was the first tool in SAP ERP that could be used for integration with external systems to perform revenue recognition. This caused a situation where the design of the solution needed to be adapted to fit new, changed requirements.

SAP S/4HANA brought improvements to RAR in two phases:

- With release 1909, *optimized contract management (OCM)* was introduced
- With release 2020, *optimized inbound processing (OIP)* was introduced

Both improvements are optional, meaning that the customer isn't required to use them. This was done mainly to reduce the risk for customers who are already using RAR and have a significant number of enhancements.

As mentioned before, both OCM and OIP were primarily oriented toward solving different technical and performance issues noted by customers while using RAR.

The following main improvements came with OCM:

- **Day-based contract modifications**
In classic contract management (CCM), there is complex

logic between the nominator/denominator that was used both for time-based and event-based POBs. In OCM, the system works with days for time-based POBs, allowing more precise calculation of modifications and proper triggering of prospective instead of retrospective modifications.

- **Contract termination or impairment calculation**
One of the biggest pain points in CCM occurs when the contract is terminated. In those cases, everything that was a contract asset should be reposted to profit and loss (P&L). However, SAP didn't come up with a solution for this business scenario mainly due to the different variations of the process, which would need a lot of custom logic to be handled properly. In OCM, terminations are standard functionality.
- **Contract acquisition costs at the contract level**
In OCM, RAR supports recognition of acquisition costs as an asset and amortization of those costs to fit revenue recognition rules.

The following main improvements came with OIP:

- **No batch processing for RAIs**
This was the main improvement that came with OIP. In classic inbound processing, you needed to either schedule a job or run Transaction FARR_RAI_MON to process RAIs. This often caused deadlocks and errors that would leave RAIs in error. In OIP, there is an option to process RAIs in real time and avoid this problem.
- **Redesigned and optimized technical architecture of tables**
In classic inbound processing, tables and application

programming interfaces (APIs) would be dynamically created depending on the structure and options the user activates. For example, if the user adds profitability analysis as an integration with RAR, characteristics would be added to table FARR_D_POB. In addition, /1RA tables would get generated once RAI classes were activated. In OIP, these activities aren't needed anymore: RAR comes with a static set of tables that fit their defined purpose.

However, the introduction of OCM and OIP has a few limitations. For example, some functionalities that existed before aren't available anymore, and a few business add-ins (BAIs) that were available are replaced with new ones. This information is especially handy if the user is planning to upgrade from classic to optimized versions and considering inbound processing as well. In addition, when it comes to integration with the Project System module for project management, this can only be done with CCM. More details will be discussed in the following chapters.

1.4.2 Event-Based Revenue Recognition

With EBRR, costs and revenues are posted as they occur, meaning they are immediately matched and posted. EBRR is integrated with the general ledger in real time, meaning revenues are posted and can be found in the income statement and margin analysis (*margin analysis* is the new version of account-based profitability analysis). In reality, this means that unlike RAR, in EBRR, information about recognized revenue is available as soon as the revenue is posted. There are no jobs to be started or batch jobs to be run: revenue is posted and can be reported.

EBRR is focused on solving revenue reporting issues that are built around different scenarios. When published, EBRR was available only on the cloud, without options to perform allocation of revenue and no parallel reporting capabilities (thus the name *event-based*). However, with release 2022, the tool was made more advanced and comprehensive to fulfill complete IFRS 15/ASC 606 requirements. If companies have scenarios that are compatible with standard SAP delivery, EBRR can be seriously considered as the choice for tracking and recognizing revenue. For more information about EBRR, see [Chapter 9](#).

1.5 Summary

In this chapter, we explained the complexity of the revenue recognition processes. The introduction of IFRS 15 required businesses to provide much more data to have a correct revenue recognition process. This chapter also covered the impact on specific industries, such as telco and manufacturing. In some cases, such as telco or high tech, IFRS 15 is becoming even more important than previous revenue reporting because now revenue is spread to different components of the bundle, which wasn't the case in the past. In addition, the standard now requires customers to disclose any hidden discount, which might be applied to the contract. Before IFRS 15, if a customer was given some device or a service for free, the revenue recognition process was very simple: revenue equals zero. However, IFRS 15 rules brought a completely new set of requirements, and these complexities are in each of the five steps needed for applying IFRS 15, starting from identifying contract and POBs, determining the exact transactional price, and the new kid on the block—determining the SSP. All these things are putting additional pressure on businesses to provide the proper data needed.

The aim of this chapter was also to bring revenue recognition to the attention of IT as well. Being IFRS 15 compliant doesn't mean just implementing rules and following the five steps. It also means providing data and executing processes in a way that those five steps can be followed. Sometimes, that requires big changes in IT

landscapes, infrastructure, and so on. Therefore, IT is experiencing complexity as well from IFRS 15 enablement. It's necessary that this complexity is understood as soon as possible and that the project is executed with it in mind. The only way an IFRS 15 implementation can be successful is if the IT and business teams are working together toward the same goal.

With this foundation in mind, we'll further explore revenue recognition landscapes with SAP in the next chapter.

2 Designing Your Revenue Recognition Landscape

The design of the overall landscape each time a new SAP product is introduced is the most important step when starting an implementation. It requires a vision of not only what is needed at the moment but also what's required for the future as well. Some of the mistakes made at this step are very difficult to correct afterward, especially in current cloud-oriented environments.

In this chapter, we'll discuss how an organization can incorporate revenue recognition solutions into their overall SAP landscape. We'll start with an overview of the typical landscape and how revenue accounting and reporting (RAR) fits in, and then we'll explain how RAR shares data with different SAP components. We'll follow with an explanation of the data model for RAR, and a comparison of the classic and optimized versions of RAR with SAP S/4HANA.

2.1 The SAP Landscape and Revenue Recognition

To get started with RAR, you need an understanding of its place in the broader SAP landscape. We'll break down a typical landscape in the following sections and explain how RAR fits in.

2.1.1 SAP System Landscape

Let's start with the basics: an SAP system is an installation of SAP products. The classic SAP landscape is usually seen as a three-tier landscape identified with a system ID (SID). Most basic setups look something like the example shown in [Figure 2.1](#).

There are three systems representing the different activities being done in them. In the development system, users are performing initial configurations and developments. Those will be moved to the quality or test system, which, by definition, is owned by the business users. Once changes and customizations are tested and confirmed, they can move to production. In the production system, configured applications and developments are used in daily work. Each time a new change or adjustment is required, the same flow must be followed.

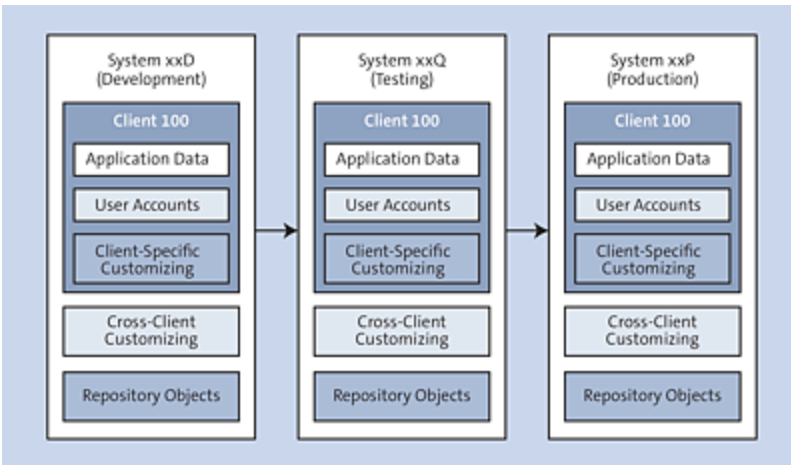


Figure 2.1 Standard SAP Landscape

Data in systems is organized by clients. By definition, a *client* is a self-contained organizational and technical unit. It's somewhere between being part of the business organizational structure (because it sits on the top of each organizational element) and technical (because all data in one system is being organized on the client level). On each client, user accounts are created and application data is set (both transactional and master data). Customizing can be either client dependent (when the customized application is within a single client) or cross-client (when the customized application is spread across several clients). However, on the system level, the repository objects are organized, such as tables, classes, programs, and other technical elements. In other words, the table is a cross-client object, and client identification occurs in the first column.

One of the first questions that comes up is how many clients you need to have. SAP has only two prerequisites: there must be client 000, which is used as the baseline client for language installations, for instance, and client 066, which is

used by SAP when a login to the system is needed. Everything else is left to the customer.

In the development system, users often create one client called the master, golden, or clean client, in which only customizing is done without creating transactional data and with a minimum of master data. Developers also create one more client, called the sandbox client, where initial testing of customizations and developments will be created. However, this setup can be more complex depending on the size of the teams and differences in activities. It's not uncommon for the sandbox client to be separated as a system to provide a safer environment where repository objects are separated, but this would require integration in the transport route and usually some workaround integration.

A similar situation occurs with a quality client where, depending on activities, more clients can be created. For instance, if both testing activities and data migration testing are being done, it makes sense that these are separated into two clients. If the customer is running maintenance of the system (business as usual [BAU]) activities and projects in parallel, this could be separated by the clients on the quality system. However, if we're talking about major transformation projects, it's not uncommon to have separated landscapes (a set of complexities that isn't covered in this book). Production is the only system where it's most common to have only one client.

This simple setup isn't often seen in current SAP environments. Users are always trying to keep landscapes as lean and simple as possible, but some complexities from

businesses they operate are very difficult to avoid. Think about an organization running different lines of business where each is operating with certain levels of specific needs. In those situations, it can be a good idea to keep those systems separate to minimize risks for development overlap and later regression testing and downtimes.

Examples include the following:

- Pharma
- Manufacturing
- Diagnostics
- Human resources (not a separate line of business, but kept as a separate system due to regulations)

We're already dealing with 12 separate systems (each being multiplied with development, quality, and production environments).

But let's not forget regions. Centralization is always a key topic when designing a landscape: On what level can we centralize data? Again, complexity comes from the business: a few decades back, being a global organization was an exception, not a rule. Since then, the world became smaller, and to be a global organization isn't uncommon. So, in a project for such an organization when planning the initial design, the same question arises: Should we aim for one central system or many separate systems? Both options have pros and cons (which again won't be discussed in detail here), but it's important to mention the complex impact on the overall landscape. Consider the split landscape shown in [Figure 2.2](#).

You can see that now, instead of three systems, there are nine. Now, imagine this scenario together with a system split per line of business. If these two are combined, the landscape can easily grow to 36 systems in total where some global consolidation system would be needed at the top.

Let's not forget that we're operating in a cloud world, where more and more companies are going with options to host applications on a cloud hyperscaler. This brings obvious benefits such as lower maintenance and infrastructure costs. The company can focus on their core business activities, while someone else can take care of the infrastructure, security, and bringing systems to the latest patch levels.

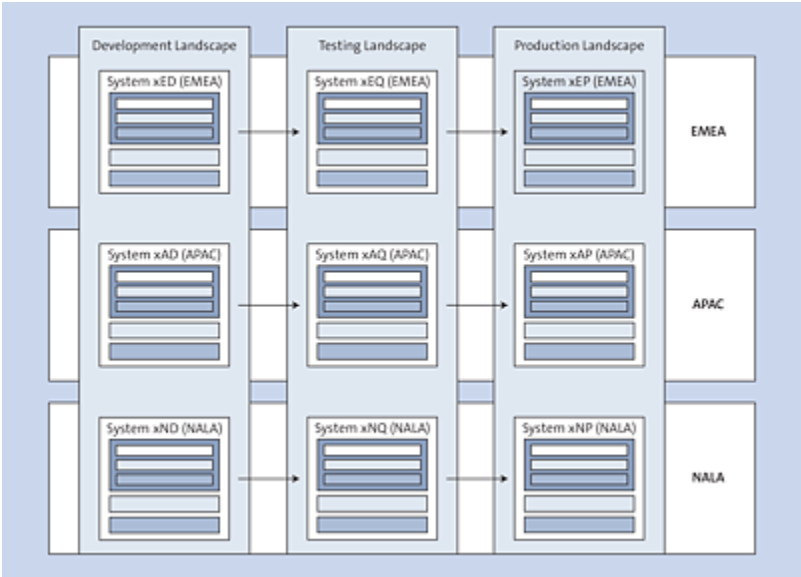


Figure 2.2 Landscape with Regional Split

But this shows even more how important it is to pay attention to how the system should be designed. The relationship between the company that is managing our application and its data and the user is precisely defined

and determines how many systems should be hosted and at what performance level. Any change to that can lead to a process that is both costly and time-consuming.

2.1.2 RAR in the SAP Landscape

So, how does revenue recognition fit into the picture? There are many options for how RAR can be plugged into an existing landscape. Keep in mind that International Financial Reporting Standards (IFRS) 15 is a global standard, and, if a company is implementing it, it needs to be done on a global level. By that alone, at least part of the question is answered. But some questions remain: Are you going to implement RAR as a standalone component or as part of an existing instance? Which version of RAR are you going to use? What kind of sizing should be done to ensure proper performance? All these questions will be answered in this section.

We already have a landscape in which our system is operating. The next task is to find the right place to fit RAR into it. There are two main options for how RAR can be placed into the landscape:

- **Implementing RAR as an add-on package on an existing SAP instance**

In this case, as shown in [Figure 2.3](#), we'll implement RAR as an add-on on an already existing SAP ERP or SAP S/4HANA instance. Depending on the version of SAP you're running, that will be an embedded module (from SAP S/4HANA version 1809 onward) or a separate add-on that needs to be activated to work.

Because all the modules will be in one place, integration provides a main benefit. For example, if a user is running integration with sales and distribution, revenue accounting items (RAIs) will be created as soon as documents are saved in it (this is valid for orders and fulfillment events equally), and once RAIs are processed, a contract will be created in the same system where the original item was created.

After running the posting program, the resulting finance and controlling documents are created automatically. Users can easily establish the document flow by using standard transactions and functions in both RAR and sales and distribution.

However, these benefits come with a price. To install or upgrade the RAR version, there might be prerequisites that need to be met by the main system. Therefore, an upgrade of RAR might trigger an upgrade of the whole instance, which might not be an easy task in there are a number of customized applications.

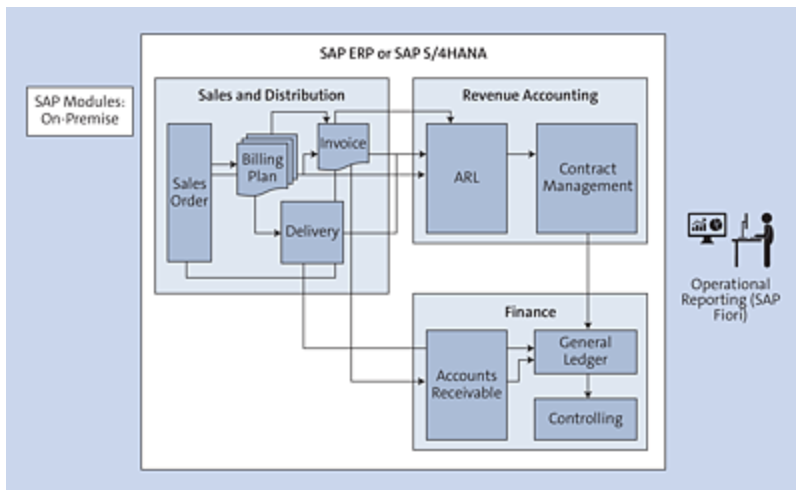


Figure 2.3 Landscape with the Revenue Accounting Add-On

- **Implementing RAR on a separate instance (sidecar approach)**

A *sidecar approach* means that RAR will be implemented on a separate instance (see [Figure 2.4](#)), which will be used primarily as a calculation engine for IFRS 15 revenue. It can be integrated with external systems via various techniques, and data will be fed first to the Adapter Reuse Layer (ARL) from where it will be processed in contracts. Once the posting run is performed, financial documents will be collected and transferred to the main instance (which again can be either SAP ERP or SAP S/4HANA). In the RAR instance, you still need to do some work to make it fully usable. The organizational structure that is needed for RAR needs to be created (company codes, profit centers, cost centers, etc.), and a certain number of accounts required exclusively for RAR need to be created. We don't need a full chart of accounts, but only ones that RAR requires (contract asset, liability, revenue, cost adjustments, etc.). Between the two systems, an Application Link Enabling (ALE) link is required, which belongs to a standard setup of distributed systems. You'll reap some benefits if you're working in an environment where the source of transactions required for IFRS 15 calculations is *not* an SAP system. Let's walk through a few examples:

- As a first example, say you're integrating with an external customer relationship management (CRM) system that is used to raise orders or a system that is used exclusively for billing. Such situations usually mean that you don't even need to synchronize master data for customers or materials because operative

systems are their main source. In those cases, using separate instances has major benefits.

- A second example is when the data volume is huge. If you need to import hundreds of thousands of orders per month together with fulfillment events and invoices, and then we load them to our main ERP system, it could become an overhead.
- A third example could be something we mentioned as a limitation factor in an integrated environment: the system version. In a sidecar approach, the user is much more flexible in keeping system versions aligned.

Obviously, there are some drawbacks to this approach. The main one is the additional investment in infrastructure; however, because it's only a one-time cost, it might not be the deciding factor for a user to choose not to go with a distributed environment. The second drawback is synchronization: all common master data between systems must be kept in sync. In addition, additional attention needs to be paid if profitability analysis is in scope, and you plan to use a distributed environment. This is because a profitability segment will be created in one environment, and the numbering must be consistent compared to the system where the profitability analysis will be used.

To conclude, whether to go with one or another approach depends on many factors, and both approaches have their own benefits and costs. Usually, when the source system is sales and distribution for orders and fulfillments, clients choose the add-on (embedded) approach because the

benefits of integration are clear. Others might think about the costs versus the flexibility of the sidecar approach.

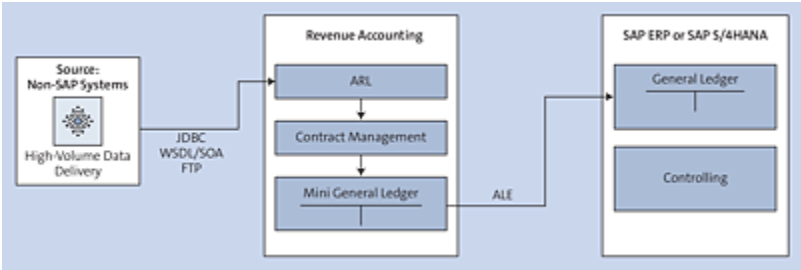


Figure 2.4 Sidecar Approach

2.2 Integration with Different SAP Components

Revenue recognition represents the central piece in every company's reporting. Therefore, RAR needs to fulfill a list of requirements to be able to successfully fulfill that role. In complex IT environments, it's not uncommon to have a heterogenous landscape with different systems that are providing data used for revenue reporting. That is why it's essential for RAR to connect with a variety of those systems.

Overall, the architecture of RAR provides that kind of flexibility, but on other hand, SAP, being an integrated system, provides the benefits of such integration. In the following sections, we'll discuss in detail how such integration can be achieved inside and outside of common SAP environments.

2.2.1 Sales and Distribution

The sales and distribution functionality has been natively integrated with RAR since its first delivery. One of the ideas for introducing RAR as the preferred solution for revenue accounting was to be a successor of Transaction VF44, the legacy tool for revenue recognition in SAP. As RAR, Transaction VF44 was natively integrated with sales and distribution.

If we look at [Figure 2.5](#), displaying the old (Transaction VF44) revenue recognition solution, we can see that the

level of integration was very high.

In revenue recognition, revenue wasn't posted to revenue but to the deferred revenue or unbilled receivable (depending on the previous balance of deferred revenue and invoicing). After that, based on specific, predefined events, the revenue recognition process and reposting from deferred revenue or unbilled receivables occurred. In addition, over time, revenue recognition was possible, and it was decoupled from the billing plan.

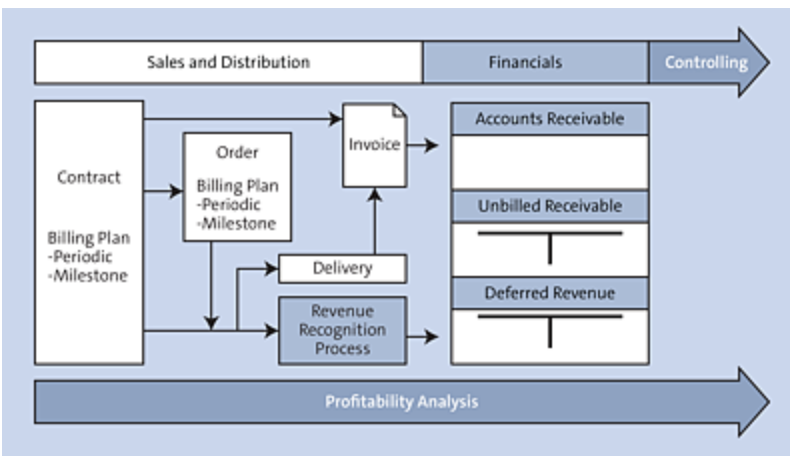


Figure 2.5 Transaction VF44 Processes

Transaction VF44 was retired for the following reasons:

- **No allocation**
There was no possibility for allocation of transactional price. In the old revenue recognition solution, one of the main requirements of IFRS 15 wasn't possible: allocation of the transactional price based on the standalone selling price (SSP).
- **No support for multiple accounting principles**
Sales and distribution revenue recognition didn't support multiple accounting principles. Results would always be

posted to the ledger group blank, meaning the same amount to the same ledgers.

- **No cost recognition**

One of the basic requirements—revenue and costs should be recognized in the same period—wasn't supported because sales and distribution revenue recognition didn't cover that requirement.

- **No integration**

Because Transaction VF44 was so tightly integrated with sales and distribution, it wasn't possible to integrate it with external applications in a feasible way.

All of these were reasons why RAR was built from scratch to fulfill mainly IFRS 15 requests but also to fill in the gaps from the old solution. [Figure 2.6](#) shows how integration between sales and distribution and RAR works: it's a direct connection in which sales orders and all subsequent events are sent straight to RAR.

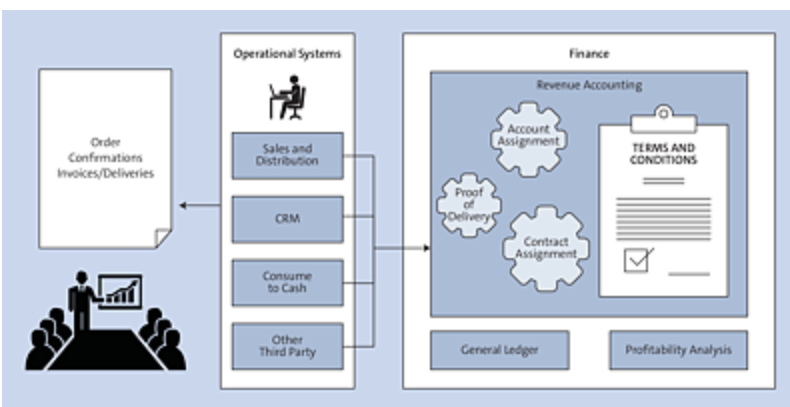


Figure 2.6 RAR Integrated with Sales and Distribution

Although RAR is still natively integrated with sales and distribution, it also can be integrated with different sources

of data. In addition, RAR provides features to fulfill the five steps of IFRS 15:

- The contract is identified by the data transfer of the sales document (sales order or sales contract). In addition, RAR comes with predefined options to perform contract combinations and with extensions if custom rules for contract combinations need to be performed.
- Performance obligations (POBs) are identified from sales document items. It also supports compound and bill of materials (BOM) POBs to reflect complex structures in sales and distribution.
- Different condition types are passed from sales and distribution to perform allocation. An allocation engine was developed to calculate and spread allocated values to different POBs.
- Parallel accounting comes natively with RAR: different accounting principles can be defined and assigned to different ledger groups.

When it comes to the approach, technically even with sales and distribution integration, users can choose between the integrated or sidecar approaches for their RAR landscape. In those cases, RAR would be implemented on a system with sales and distribution (either as an add-on or natively in SAP S/4HANA) and integrated with the system running finance. However, in practice, this is very rare. Usually, in sales and distribution environments, RAR utilizes its strengths from integration and runs on the same platform.

2.2.2 Customer Relationship Management

SAP Customer Relationship Management (SAP CRM) (or SAP S/4HANA Service in the SAP S/4HANA environment) consists of various components that allow you to integrate CRM with other SAP and non-SAP modules, internet, mobile devices (e.g., smartphones and tablets), and enterprise portals. In the center, it has an SAP CRM server that includes the following subcomponents:

- CRM enterprise functions
- CRM middleware

Then, there are adapters to communicate with handheld devices and the internet. The SAP system is used as the backend, SAP BusinessObjects Business Intelligence is used for analytical reporting, and SAP CRM is used to enhance the capabilities of CRM.

You can also see the SAP CRM architecture and all its listed key components in [Figure 2.7](#).

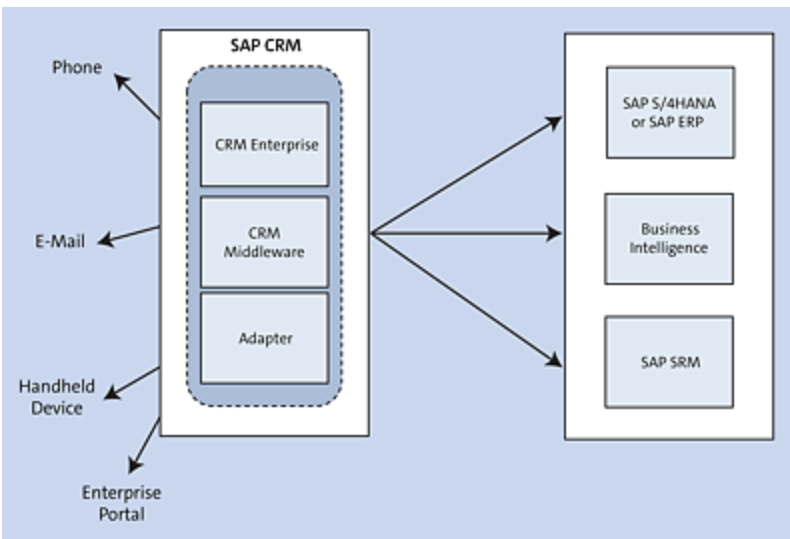


Figure 2.7 SAP CRM Architecture

The main features that we use in SAP S/4HANA Service are service contracts, service orders, and service notifications. All of these documents are transferred to sales and distribution and later to RAR.

Sender component CRM service (also called CRS) and sales and distribution are provided by SAP for integration with revenue accounting. Once the CRS contract is created and released, the billing plan is created in sales and distribution through which billing will be done. In this setup, an SSP is created and maintained through BRFplus rules in RAR.

Sender component CRS uses sender component CS01 (CRM service contract item) and SDII for billing item (sales document invoice item). For integration to be set, customization settings need to be followed, which we'll discuss in detail in the upcoming chapters.

Once RAI classes are set, the next step is to set up the sender component. Here, the user needs to connect document item types as the source with the sender component and assign a logical system name if necessary. SAP provides the following standard components: CRS (for CRM service) and SD (for sales and distribution) with links to SD01 and SDII RAI classes. The technical names are set by SAP to ensure that the system automatically provides the required settings for each class when it's created.

The RAI class CS01 defines the technical characteristics of the order items, and the class SD03 defines those of the invoice items of sales and distribution billing.

2.2.3 Billing

SAP Billing and Revenue Innovation Management is an SAP tool that targets customers with high-volume consumption businesses. [Figure 2.8](#) shows a breakdown of the core components that make up the solution.

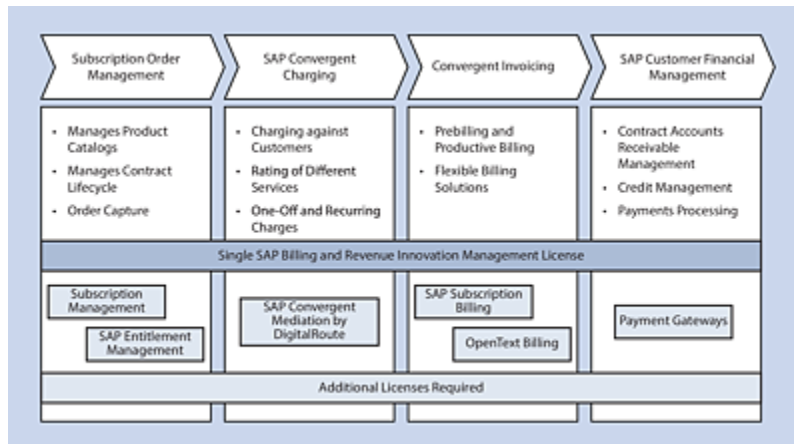


Figure 2.8 SAP Billing and Revenue Innovation Management Architecture

SAP Billing and Revenue Innovation Management consists of several products bundled together to make a single solution for all high-volume businesses:

- **Subscription order management**

Subscription order management is the first step when it comes to working with SAP Billing and Revenue Innovation Management. It's used to capture orders, quotations, or any other documents that represent the beginning of the sales process. SAP CRM is the main backbone of subscription order management. To perform its main task, it uses product and customer catalogues. The first step of subscription order management is creating the order that is later passed to SAP Convergent Charging for rating and charging activities.

- **SAP Convergent Charging**

SAP Convergent Charging is used to determine the

amount to be charged to the customer by using two main components: charging and rating. Charging is used to determine the customer account against which a charge should be applied, and rating relies on a mechanism to determine the amount to apply against the customer. The rating is based on complex mechanisms that are used to determine one-off, recurring charges, limits, discounts, and so on.

- **Convergent invoicing**

Once the amount to be billed is calculated, information is passed to convergent invoicing. SAP Convergent Charging passes billable items that should be consumed by convergent invoicing. All data is being used to produce billing data, and convergent invoicing has many features such as pre-billing, summarized bills, and so on. Unlike SAP Convergent Charging, which is a separate application, convergent invoicing is part of the SAP S/4HANA stack.

- **SAP Customer Financial Management**

SAP Customer Financial Management's main component is contract accounting, which represents a subledger that is specifically designed to accommodate the needs of a sector with a huge number of customers. It offers many features such as multiple accounting principles reporting, tax reporting, detailed information on the customer level, and so on.

There are many additional features that can be used together with SAP Billing and Revenue Innovation Management, but which are separately licensed. Some of them are subscription management and SAP Subscription Billing, which are aimed mainly at companies running

businesses that rely on recurring charges; SAP Convergent Mediation by DigitalRoute, which is used when data needs to be transformed between sources and SAP systems; and payment gateways, which offer different adapters to communicate with PayPal, Swift, and so on.

SAP Billing and Revenue Innovation Management has gone through transformations in the past several years. Since SAP S/4HANA release 1909, subscription order management and contract accounting have been integrated into SAP S/4HANA, which makes integration simpler.

If we look at [Figure 2.9](#), there are two integration points between RAR and SAP Billing and Revenue Innovation Management. The first one is data that is needed to capture the order, which will be represented as a contract in RAR. Then, we need to retrieve the data, which is used for invoicing, called billable items.

From an architecture point of view, the main question is how to organize the landscape when SAP Billing and Revenue Innovation Management is in place. From [Figure 2.9](#), we can see that subscription order management and contract accounting being integrated into SAP S/4HANA provides a lot of benefits from being integrated within the same system as RAR. However, note that SAP Billing and Revenue Innovation Management is put in place for high-volume businesses, so the data that will be processed in it is already significant.

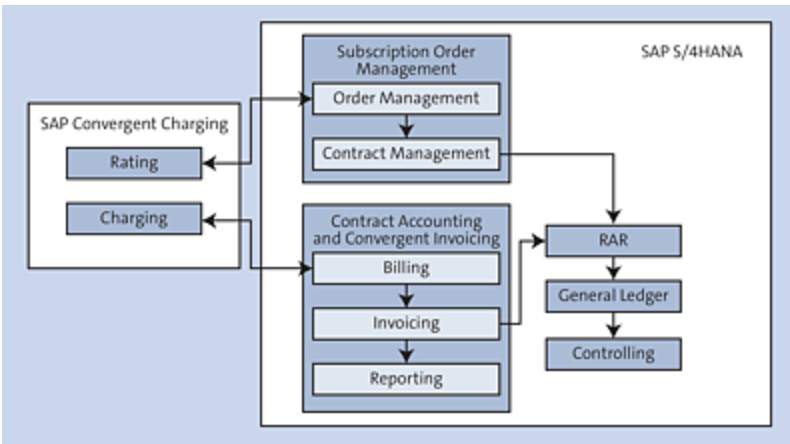


Figure 2.9 SAP Billing and Revenue Innovation Management: Integration Points

In addition, it's worth mentioning that some companies only use parts of SAP Billing and Revenue Innovation Management rather than the whole suite (e.g., convergent invoicing and subscription order management, without SAP Convergent Charging, or subscription order management and sales and distribution billing). That also needs to be taken into account while deciding between keeping integration on a high level (meaning installing RAR on the same system) or moving to a distributed landscape.

2.2.4 Non-SAP Systems

So, what happens if you have a landscape that isn't purely SAP? Depending on the industry, this option isn't so uncommon. Think about telco companies: for legacy reasons, systems used for billing and even more for charging are linked to equipment that is delivered. All major providers of telco equipment have their own systems for billing and charging.

In those environments, it's to be expected that companies will have a lot of different systems that will be source systems for a tool such as RAR. The first question is, should we go with the sidecar approach, or a single instance that will include both the ERP system and the RAR data? This question was already covered in previous sections, but it's worth mentioning that this decision depends primarily on the data volume: if we expect to have significant amount of data coming from source systems, then the sidecar approach is worth considering.

But there is also a second question that might be important. Companies with multiple systems used for different purposes usually have developed them on different platforms. This means that data structures from those systems will be different, and, consequently, RAR won't have a single, unique data source. So naturally this data would need to be transformed before it can come to RAR. The simple solution can be that RAR performs data transformation too. As will be covered in detail, the ARL staging area of RAR already consists of rules and error management procedures that can be applied on data to prepare it for RAR processing. In addition, different statuses of RAIs can be applied to manage data quality.

So, is this enough? The answer is no. A landscape in these kinds of environments is very complex, and any kind of reprocessing and data correcting requires considerable time. Imagine if RAR throws some of the common errors; the user would have an option to either correct the errors manually or maybe update BRFplus rules for those contracts to be processed. All of these activities take considerable time, and

in the month-end closing process, for example, will also add complexity that is difficult to resolve.

The solution to this problem is implementing one more layer that is responsible for providing data in the exact format RAR requires, plus implementing business rules that make sure data from source systems is properly transformed. These are called extract, transform, load (ETL) tools.

[Figure 2.10](#) shows that before data comes to RAR, it needs to pass through a transformation layer. In this layer, you define rules that should perform transformation of the data into the required format that RAR can read (often called creation of data points). Only when the data is ready can loading into RAR start.

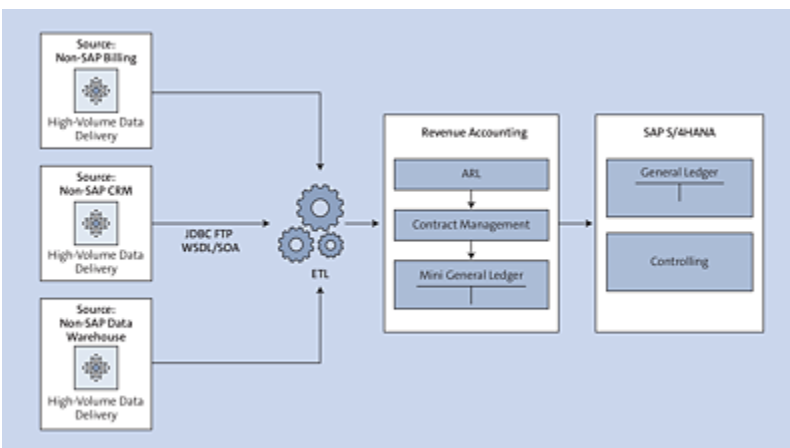


Figure 2.10 RAR Landscape with ETL

If you decide to go with ETL as the tool to transform your data, this tool needs to be connected with source systems and have data loaded into it. The connection between systems can be made via different techniques and depends on the tools available and processing requirements. Most ETL tools on the market support all major databases (Oracle, MS SQL, DB2, SAP HANA) and main connection options (Java

Database Connectivity [JDBC], Open Database Connectivity [ODBC], File Transfer Protocol [FTP]).

Once data is in the ETL tool, there must be business rules that will be used for transformation. These rules can be simple, such as concatenating or adding different strings and filtering data, or these rules can be very complex, such as creating SQL scripts of several hundred lines. The complexity of rules will depend on the diversity of source systems and on the quantity of data.

Example: Calculating the SSP

As mentioned before, the SSP is paramount in IFRS 15 calculations. In some cases, it's easy and straightforward where businesses decide to apply very simple solutions, such as the cost-plus method or list-price method as practically expedient. However, in other cases, calculating the SSP can be very complex. Let's take a telco company as an example and SMS as a service they provide. If we apply a simple approach such as cost-plus, we sum up all costs that are related to providing this service. But these costs are the same as those needed to provide voice or data services. So, applying a simple cost-plus method might lead to an SSP for the SMS service that would cause very high revenue to be allocated to it, which is contrary to how business is evolving.

As a practical expedient, businesses are allowed to consider the percentage of usage also when calculating the SSP. So, you need to compare such calculated costs with how much SMS was used from packages on average and compute the SSP that way. However, to do this, you

would need information not only from billing and CRM systems but also from charging. Charging systems create call detail record (CDR) files with volumes that easily reach terabytes in size on a monthly level (depending on the number of contracts). So, to calculate the SSP, the ETL tool needs to be optimized and sized properly.

Once data is calculated, it's then sent to RAR, which again uses the ARL as a staging area to perform technical correctness of data received. When completed, the data will be sent to contract management and later posted.

In the market, there are plenty of tools for ETL processing. For simple transformations, even SAP middleware connectors can be used (e.g., SAP Process Integration or SAP Process Orchestration), but if you need to perform more complex calculations, you should consider tools that are more suitable for ETL activities.

This setup can evolve further. Often customers realize the power of the collected data and instead of using it only for ETL purposes, they build data lakes and put ETL on top, creating *data hubs*. The SAP HANA database provides a performance level that can help you maximize performance and usage of your data.

2.3 Revenue Recognition Data Model

Data is the foundation for any system. Along with huge data comes the challenges of accessibility, storage, and processing. The same data is available, but the smartness of the system and the software is the key to meeting the requirements of the mandates. Mandates differ depending on requirements and countries/regions; when it comes to RAR, we're talking about IFRS 15 and ASC 606 mandates specifically, as covered in [Chapter 1](#). Data needs to be processed and presented in the format that is expected by these mandates. Huge businesses have huge amounts of data, and the big task for them is to be compliant with the mandates to run their business smoothly and manage their data smartly. Data needs to be converted to the language of business. We need to understand the data that is available and transform it into a form that is accepted by the system. In the previous section, we discussed ETL and the data transformation that must be done before data is sent to RAR. Once the data is received from any source of integration, it's saved in the ARL and processed. Storing the data to be accessed correctly is the prime concern. This is where structures, tables, and formats of data storage come in. Understanding the technical objects that are available and the relationship between them leads us to understand the system better and provides easy access to data. Let's explore how it's done in RAR.

Organizing data is important for easy access, and easy access is important for program design and flow, configuration, and testing. Understanding the organized format leads to a better understanding of the component and that's what gives functional or technical consultants an upper hand in any given SAP module. The configuration details will be explained in [Chapter 3](#). In this section, we focus on the technical output of some configuration steps. Configurations such as RAI class creation and generation result in the creation of some technical objects. So, this section is more focused on the technical details and will be more relevant for a developer. In general, it's important to have exposure to these technical objects, as well as their structure and uses. We also focus here on the tables that are available in RAR and most importantly the table relationships to recall the optimized way to access data. In this section, we'll mostly focus on classic inbound processing. Optimized inbound processing (OIP) is discussed in detail in [Chapter 3](#). A core understanding of classic inbound processing is required to get a clear picture of OIP.

Optimized Inbound Processing

OIP is the simplified version of classic inbound processing. Some of the concepts in this section are applicable to both classic inbound processing and OIP. The difference is that OIP offers a fixed database model (based on basic and sender component-specific information). This means that structures, application programming interfaces (APIs), and runtime working structures previously generated under classic inbound processing are now available as predefined fixed structures, APIs, and runtime working

structures with OIP. With OIP, RAIs are created, and revenue accounting contracts are created from them.

Data flows from different sources. For RAR, the data sources are shown in [Figure 2.11](#).

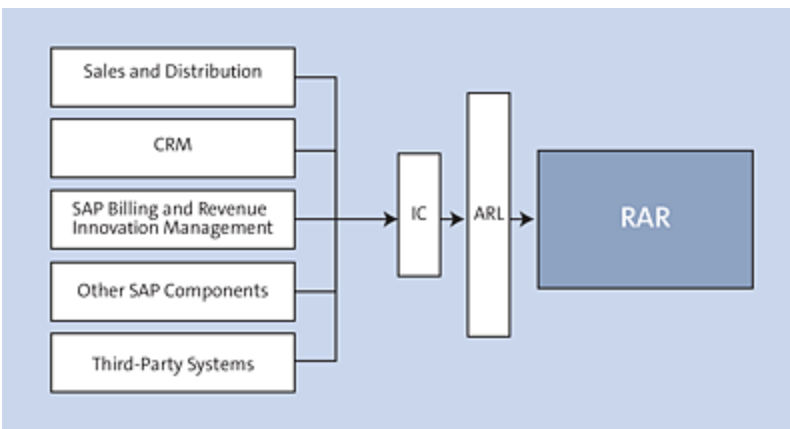


Figure 2.11 Data from Sources to RAR

You can see that there are multiple sources, called sender components. The integration component needs to create the RAIs and send them to revenue accounting through the ARL. Once RAIs are created, they need to be processed to create/update revenue accounting contracts and POBs. The source applications are defined as sender components.

Revenue recognition-related transactions result in the creation of RAIs, which are like packets of information. Each RAI belongs to a class that will determine its technical definition and technical properties. There are two item (or record) types: main and condition. The main and condition records are related as 1 and 0 to N relations. The main item or record type contains the most basic data, and a condition item can't exist without the main item. Condition items contain amounts and account-related information.

RAIs have three possible statuses:

- **Raw**
The initial status of the RAI when it's first created.
- **Processable**
The raw RAIs when transferred.
- **Processed**
The final status of RAIs.

There are two additional statuses: **Raw - Exempted** and **Processable - Exempted**. RAIs with these two statuses are no longer considered for further processing steps and are removed from the processing flow. Exempted items can be restored when the settings of the selected exemption reason allow restoration, for example, when the error is resolved. An exemption history is created for each exempted and restored RAI.

2.3.1 Sender Components

Because we spoke about the sender component at the start of this section, let's look at the basic configuration related to the sender component. The details of each step of the configuration are discussed in [Chapter 3](#); here we're considering only the technically relevant steps to show the technical objects or results that are relevant. Follow these steps to configure the sender components:

1. Define the sender components by executing Transaction FARR_IMG and following menu path **Revenue Accounting • Inbound Processing (Classic) •**

Revenue Accounting Item Management • Define Sender Components.

This sender component is the sending system, which can be the SAP CRM system, sales and distribution system, and others. **CA**, **CRS**, and **SD** are the standard sender components provided by SAP, as shown in [Figure 2.12](#). If you have a third-party or external system sending data, you can add it by clicking on **New Entries** and entering “ZX” in the **Send. Comp** field. Then, enter the **Sender Component Description** as “Third-Party System”.

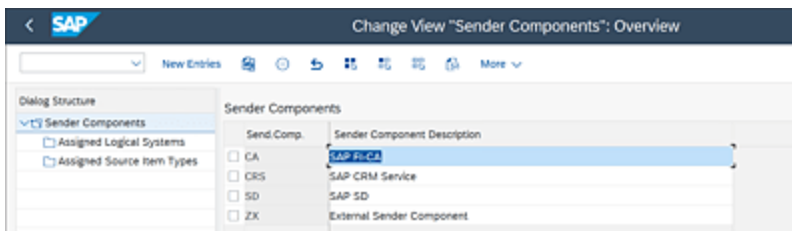


Figure 2.12 Sender Components Screen

2. The sender component is assigned to a logical system. You first need to define the logical system by executing Transaction FARR_IMG and following menu path **Revenue Accounting • Inbound Processing • Revenue Accounting Item Management • Define Logical Systems** to open the **Logical Systems: Overview** page, as shown in [Figure 2.13](#).

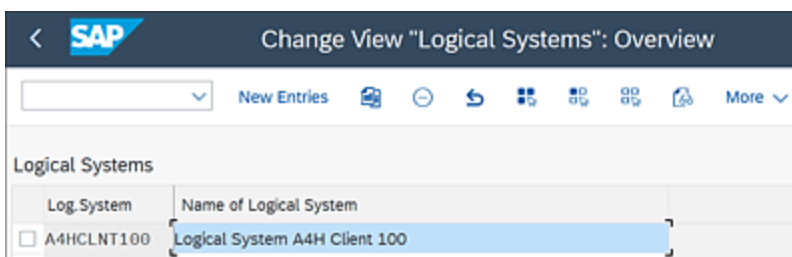


Figure 2.13 Define Logical Systems

3. You assign the logical system to the sender component by executing Transaction FARR_IMG and following menu path **Revenue Accounting • Inbound Processing (Classic) • Revenue Accounting Item Management • Define Sender Components**.
4. Select the sender component, and double-click on **Assigned Logical Systems**. In the screen that appears, click on **New Entries**, and select the required logical system, as shown in [Figure 2.14](#).

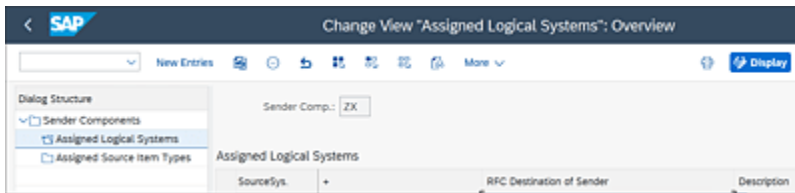


Figure 2.14 Assign Logical Systems

5. The sender component information is complete when we define the source item types, which are the orders, fulfillments, and invoices for a contract that flow from source applications. To define them, execute Transaction FARR_IMG, and go to menu path **Revenue Accounting • Inbound Processing (Classic) • Revenue Accounting Item Management • Source Document Item Types**. SAP provides a list of source item types that correspond to order, fulfillments, and invoices for the specific sender components, as shown in [Figure 2.15](#), for example, **CAOI**, which is class type **Order Item** for provider contracts. If you want to define a custom source item type for an external sender component, click on **New Entries**, and add the required custom source item type.

SrcitmType	Class Type	Item Type Description
<input type="checkbox"/> SDFI	Fulfillment Item	▼ Fulfillment Item
<input type="checkbox"/> SDIG	Order Item	▼ Invoice generated POB
<input type="checkbox"/> SDII	Invoice Item	▼ Invoice Item
<input type="checkbox"/> SDOI	Order Item	▼ Order Item
<input type="checkbox"/> SDPI	Invoice Item	▼ Billing Plan Item
<input type="checkbox"/> ZACI	Order Item	▼ External Order Item Type
<input type="checkbox"/> ZAI	Invoice Item	▼ External Invoice Item Type

Figure 2.15 Source Item Types for Order, Fulfillment, and Invoice

- The source item types are assigned to the sender component by executing Transaction FARR_IMG and following menu path **Revenue Accounting • Inbound Processing (Classic) • Revenue Accounting Item Management • Define Sender Components**. Select the required sender component, and double-click on **Assigned Source Item Types** as shown in [Figure 2.16](#).

Send.Comp.	Sender Component Description
<input type="checkbox"/> CA	SAP FI-CA
<input type="checkbox"/> CRCS	SAP CRM Service
<input type="checkbox"/> SD	SAP SD
<input checked="" type="checkbox"/> ZX	External Sender Component

Figure 2.16 Assign Source Item Type to Sender Component

- When the screen shown in [Figure 2.17](#) appears, click on **New Entries**. Then, select **SrcitmType**, and press F4 to get the list of options. Select the required source item types for your sender component. You can select the source item type for order, fulfillment, and invoice from here.

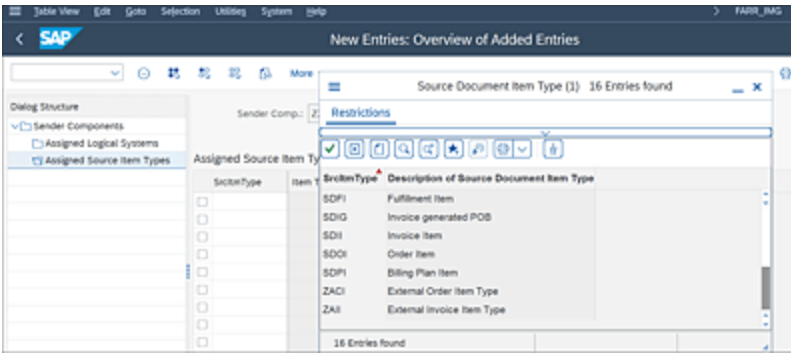


Figure 2.17 Assign Source Item Type to Sender Component (Cont.)

You've now prepared the sender component and logical system details and defined the source item types.

2.3.2 Structures

Structures are data objects with components of any data type that are saved in sequence in the memory. The data type of a structure is a structured type, or a structure defined in the ABAP Dictionary. *Type* and *sequence* are the important terms here: the type for a programmer helps plan the memory, and the declarations and the sequence are crucial for data access. In the following sections, we'll explore the structures that are a result of the RAR configuration.

Interface Components

Interface components are part of the ARL. This is basically a collection of structures for extracting the information from various SAP components. Interface components contain the structures of all the fields that are available for main items, condition items, and various sender component-specific structures. For example, if the sender component is sales

and distribution, then we have sales and distribution-specific structures (**SD**) and profitability analysis-related structures (**COPA**), as shown in [Figure 2.18](#).

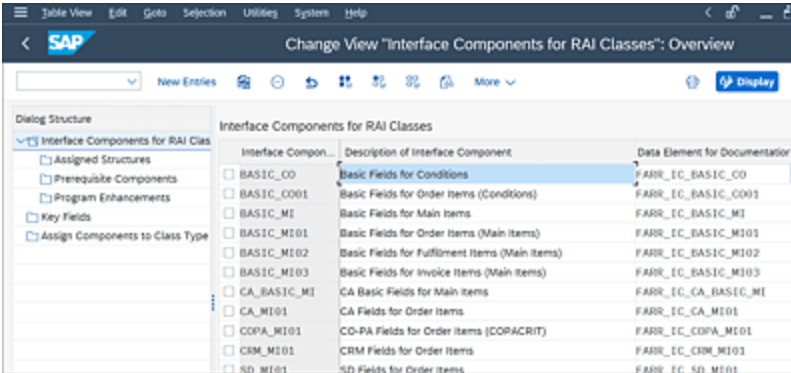


Figure 2.18 Interface Components

As an example, let's take a closer look at interface component **BASIC_MI**. This forms the basis for the RAI **Main Item** fields and contains the most important data for the RAIs such as the header ID and others. [Figure 2.19](#) shows a snapshot of the **BASIC_MI** fields.

Interface Component: **BASIC_MI** Basic Fields for Main Items

Status: Raw Main Item

Fid Disp.: MI Main Item

Field Name	Data element	Data Type	Lngh	Short Description
HEADER_ID	FARR_HEADER_ID	C Data Type	20	Header ID of Source Document for Revenue Accounting Item
ITEM_ID	FARR_ITEM_ID	CHAR	15	Item ID of Source Document for Revenue Accounting Item
KEYPP	FARR_KEYPP	NUMC	3	Subarea for Parallelization
RAIC_TYPE	FARR_RAIC_TYPE	CHAR	2	Revenue Accounting Item Class Type
RAIC	FARR_RAIC	CHAR	4	Revenue Accounting Item Class
STATUS	FARR_RAI_STATUS	CHAR	1	Status of Revenue Accounting Item
EXCHIST	FARR_RAI_EXCHIST	CHAR	1	History Record Exists for Exemption
CHIST	FARR_CHIST	CHAR	1	History Exists
BUKRS	BUKRS	CHAR	4	Company Code
QUANTITY	FARR_QUANTITY	QUAN	18	Quantity
QUANTITY_UNIT	FARR_QUANTITY_UNIT	UNIT	3	Unit of Measure
INITIAL_LOAD	FARR_INITIAL_LOAD	CHAR	1	Initial Load
LOG_HANDLE	BALLOGHNDL	CHAR	22	Application Log: Log Handle
MIG_PACKAGE	FARR_MIG_PACKAGE	CHAR	4	Migration Package ID
CREA_USER	FARR_CREA_USER	CHAR	12	User who created the revenue accounting item
CREA_TMSTP_UTC	FARR_CREA_TMSTP_UTC	DEC	15	Time the revenue accounting item was created

Figure 2.19 Fields Available in Interface Component BASIC_MI

To check the details of the field structure of interface components, go to Transaction **FARR_IMG**, and follow menu path **Revenue Accounting • Inbound Processing**

(Classic) • Maintain Revenue Accounting Item Class. You'll arrive at the screen shown in [Figure 2.20](#).

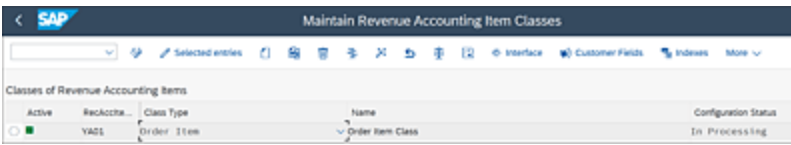


Figure 2.20 Revenue Accounting Item Class Screen

Here, click the **Interface** button to see the screen shown in [Figure 2.21](#).

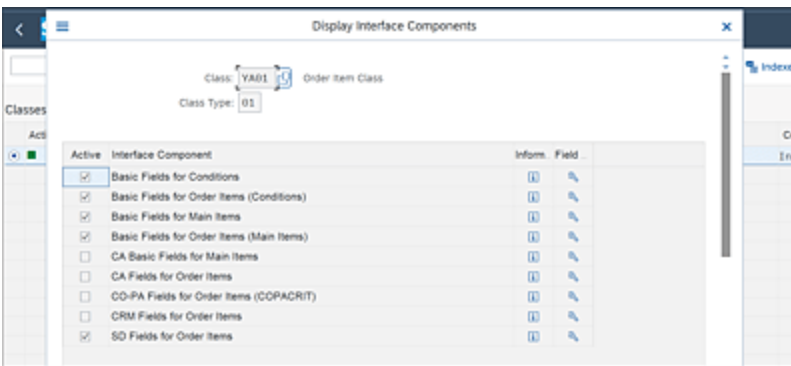


Figure 2.21 Screen to Select Interface Components

Select the **Basic Fields for Main Items** interface component, and click on the **Field** icon.

When the screen shown in [Figure 2.22](#) appears, you can choose the status of the RAI under **Status Selection**. You can select any one of the statuses such as **Raw**, **Raw - Exempted**, **Processable**, **Processable - Exempted**, and **Processed**. Then under **Field Display**, you see the default **Main Item** or **Condition** radio buttons. Depending on the interface component that you've selected, if you choose **Main Item**, then the main item is defaulted, and the same thing is true for **Condition** and condition items. You can also choose to display only the key fields by selecting the **Key Fields** checkbox; if you don't select that checkbox, all fields

are displayed. When finished, click **OK**. As shown in [Figure 2.22](#), **Basic Fields for Main Items** is selected with the status set to **Raw** and the field display set to **Main Items** without checking the **Key Fields** checkbox (refer to [Figure 2.19](#)).

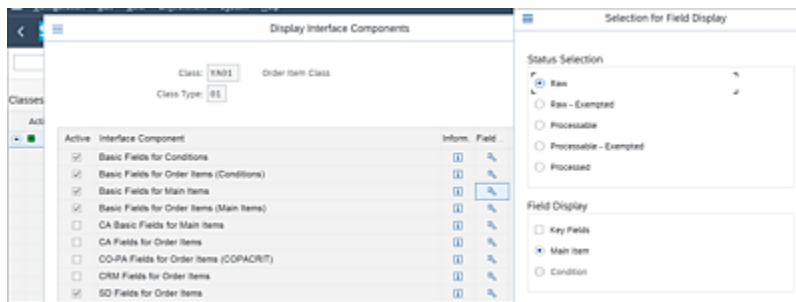


Figure 2.22 Navigating to the Field Details of the Interface Component

RAI Classes and Class Types

The classes for RAIs demonstrate the technical properties of each RAI. RAI classes can be created by selecting the appropriate interface components (discussed in the previous section) and can also be enhanced by adding customer-specific fields. We'll discuss adding customer-specific fields in detail in the coming sections.

RAI classes determine technical attributes related to RAIs such as the following:

- The database tables for storing the RAIs (the set of tables are created as part of RAI class generation, as we'll discuss in [Section 2.3.3](#); these tables carry the name of the RAI class)
- Function modules for processing and saving RAIs
- Structure of RAIs, which is determined by your choice of interface components and customer fields

The *RAI class type* tells you if it's of type order, fulfillment, or invoice. These are the only three possible types. Even if you choose a standard class type or create a custom class type, it will belong to one of these three types. You need to define the class type when you maintain the RAI class.

It's very important to understand the class type before you select the interface components. As for the order type, you need to select interface components that are related to orders. For invoice, you need to select the invoice class type, as you have to further set whether the record type is a main or condition record.

Let's walk through the class types in the system, as listed in [Table 2.1](#).

Class Type	Description
01	Order items
02	Fulfillment items
03	Invoice items

Table 2.1 Class Types

In general, the three class types indicate the following:

- **Order items**

These are used to store data that will be used to create or update contracts. These RAIs contain data needed for contract creation such as duration, derivation POBs, and contract modification or combination. Besides basic data for contracts (created as main items), there is a section with data that contains values represented as conditions.

One main item can have multiple condition items depending on your setup.

- **Fulfillment items**

These are used when fulfillments are being passed to RAR that are usually represented as goods issue.

- **Invoice items**

These are created once the invoice is entered and passed to the system. It contains the main item section with the predecessor item to which the invoice is related and condition items that only have price conditions and not statistical conditions (e.g., SSP). Invoice items are created for regular invoices, credit/debit memos, and planned items for the billing plan.

The RAI classes have different names to represent the operational source. The technical names are preset by SAP because it's important that the system knows the basic setting that needs to be provided for each class. For example, if the class for the sales and distribution integration is called SD01, then "SD" stands for sales and distribution and "01" stands for order type; similarly, SD02 is for fulfillment. All the item classes are listed in [Table 2.2](#).

Integration Source	Class	Relevant Type
Sales and distribution	SD01	Order
	SD02	Fulfillment
	SD03	Invoice
SAP Billing and Revenue Innovation	CA01	Order

Integration Source	Class	Relevant Type
Management	CA02	Fulfillment
	CA03	Invoice
SAP CRM or SAP S/4HANA Service	CS01	Order
	CS03	Invoice
Third-party or external systems (custom item class)	Z*01	Order
	Y*01	Order

Table 2.2 Item Classes by Source

The RAI class is created by selecting the relevant interface components and adding the customer-specific fields. We've already discussed the interface components in the previous section, and now we'll focus on adding the customer fields.

The configuration steps for creating a class are discussed in [Chapter 3](#), so here we'll demonstrate just the basic steps:

1. Go to Transaction FARR_IMG and follow menu path **Revenue Accounting • Inbound Processing (Classic) • Maintain Revenue Accounting Item Classes**. Custom class YA01 for order type will be created in this example.
2. Click the **Create** button, and enter the following details, as shown in [Figure 2.23](#):
 - **Rev. Acc. Itm Class**: Enter "YA01" for this example.
 - **Name**: Provide a suitable description, which is "Order Item Class", for this example.

- **Class Type:** Choose **01** to create an order item class.

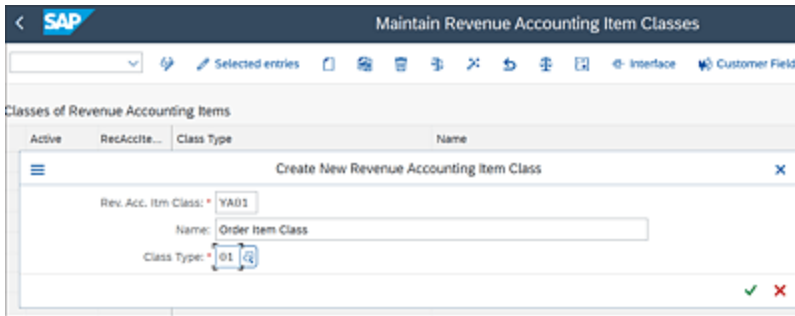


Figure 2.23 Custom Order Item Class Creation

3. Click the green checkmark to proceed further to the next step to select the interface components.
4. Click the **Interface** button, and the popup shown in [Figure 2.24](#) will appear. Check the boxes under the **Active** column for the interface components you'd like to select.

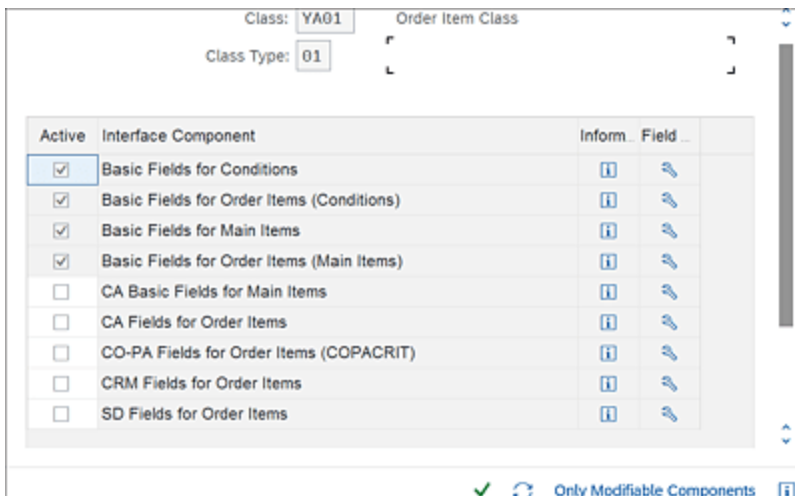


Figure 2.24 Select the Interface Component for YA01

5. There are some default interface components that are provided for each class type. If required, you can add more by selecting them and then clicking on the green

checkmark to confirm the selection of interface components.

Customer Fields and Append Structures to the Class

Now, let's add custom fields to the class. You can click the **Customer Fields** button during the class creation to add additional custom fields. In RAR, data is planned based on where the field is needed. There are certain fields that you might need only for a contract, only during processing, and so on. This categorization is based on the extensibility concept of the Easy Enhancement Workbench (EEW), which uses extension include structures. This helps the architect plan and add the required fields per the requirement to provide easy and efficient field extensibility. These include structures are then included in all relevant tables and internal structures.

There are fields that you might need in contracts but not in reporting, just like there are fields necessary for processing that aren't required in contracts. This helps get rid of redundant data, optimize performance, and limit the data volume, as typically huge amounts of data are being handled.

So, before you click that button, you need to add the fields to the following extension include structures:

- **INCL_EEW_FARR_ARL**

This structure is for fields only used in RAI processing. These fields are needed to define rules for contract combinations or contract composition in BRFplus. They

only extend the RAI tables, which we'll discuss in [Section 2.3.3](#).

- **INCL_EEW_FARR_POB**

This structure is for fields also used in revenue accounting contracts. When you need fields at the POB level, you add them to this structure. Fields added here are available in BRFplus rules and the POB table.

- **INCL_EEW_FARR_REP**

This structure is for fields also used in reporting. This is a huge section, and the fields that you add here are available on a broader spectrum such as RAI tables, POB tables, posting tables, and BRFplus rules. The fields added here can also be passed to general ledger documents.

- **INCL_EEW_FARR_CONTRACT**

This structure is for fields on the contract header level.

Now that you know about these four structures, you need to decide where the fields need to go and add them in those respective structures.

As an example, let's look at how you can add an additional field as an extension to INCL_EEW_FARR_POB:

1. From Transaction SE11, display the include **INCL_EEW_FARR_POB**, and then click the **Append Structure** button. A popup appears, as shown in [Figure 2.25](#).

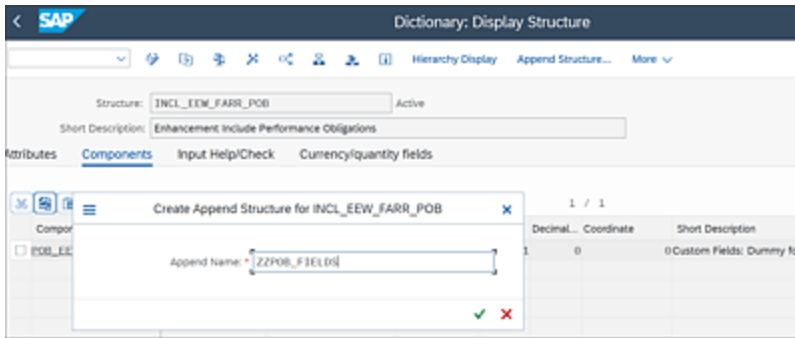


Figure 2.25 Creating an Append Structure for INCL_EEW_FARR_POB

2. Enter the new append structure name in the **Append Name** field (for this example, “ZZPOB_FIELDS”), and click on the green checkmark. Two fields—**ZZPOB_CUR** and **ZZBPOB_STATUS**—will be added to the append structure along with a suitable description, as shown in [Figure 2.26](#). Note that the data element and domains were already created previously using Transaction SE11.

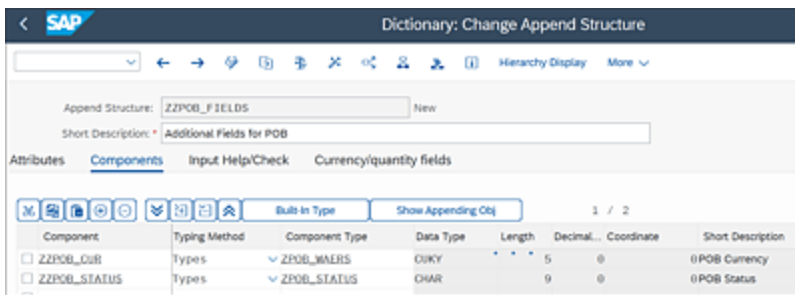


Figure 2.26 Custom Fields Added to the Append Structure

3. Click the **Activate** icon in the menu or press **Ctrl+F3** to save and activate append structure **ZZPOB_FIELDS**.
4. Once activated, you can go back to structure **INCL_EEW_FARR_POB**. The final structure is shown in [Figure 2.27](#).
5. Click the **Activate** icon in the menu or press **Ctrl+F3** to activate structure **INCL_EEW_FARR_POB**. Upon

activation of the structure, you can go back to table FARR_D_POB in Transaction SE11 to see that the two fields are now available, as shown in [Figure 2.28](#). For more information on tables, see [Section 2.3.3](#).

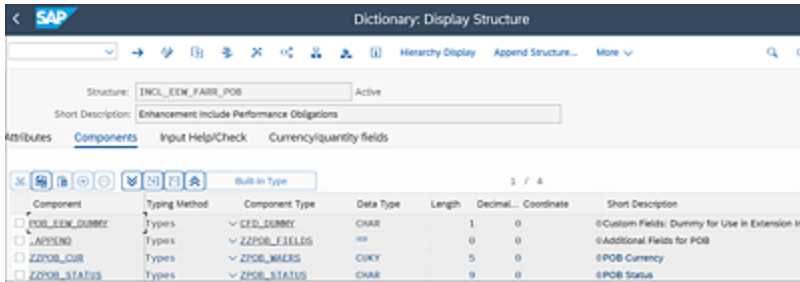


Figure 2.27 Structure INCL_EEW_FARR_POB with Additional Custom Fields

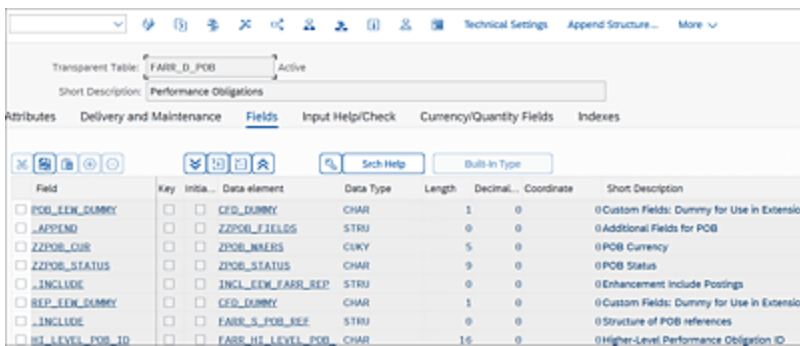


Figure 2.28 Table FARR_D_POB with the New Additional Custom Fields

6. Similarly, let's add another a new field called **ZZPOB_DESC** for the additional POB description to the structure for reporting INCL_EEW_FARR_REP following the same steps. After adding the new field, the structure **INCL_EEW_FARR_REP** looks like [Figure 2.29](#).

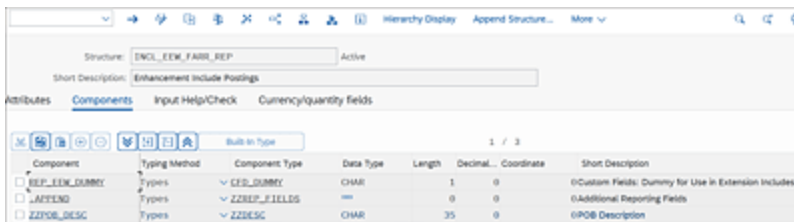


Figure 2.29 Structure INCL_EEW_FARR_REP with Additional Fields

Now let's go back to completing the class creation screen by following these steps:

1. Execute Transaction FARR_IMG and follow menu path **Revenue Accounting • Inbound Processing (Classic) • Maintain Revenue Accounting Item Classes**.
2. Select the class for this example, **YA01**, and then click the **Customer Fields** button.

All the fields that are added to any of the structures (INCL_EEW_FARR_ARL, INCL_EEW_FARR_POB, INCL_EEW_FARR_REP, and INCL_EEW_FARR_CONTRACT) are now available in the dropdown when you use the **F4** help in [Figure 2.30](#) for choosing the customer fields. The newly added fields won't be here unless they are added to the structures. Here you also have the option to choose if those fields are to be available for raw or processable/processed statuses.

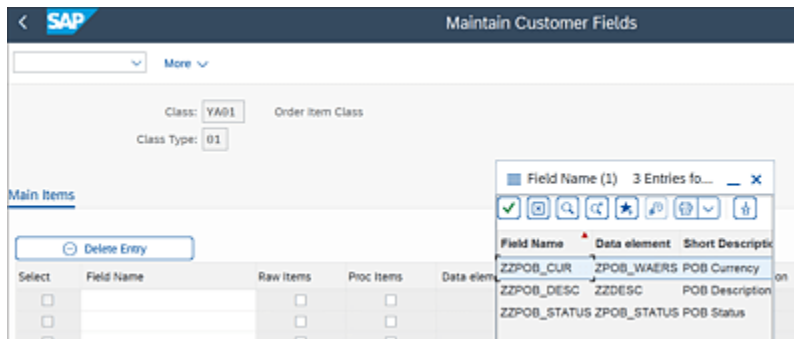
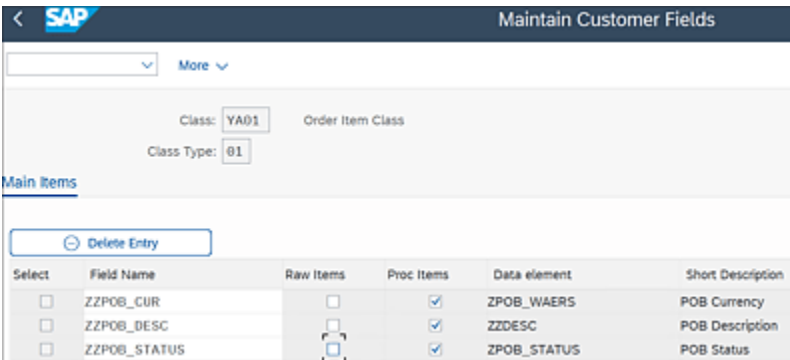


Figure 2.30 Customer Fields for Addition

3. Now add the additional fields, and choose the appropriate checkboxes for availability of those fields in **Raw Items** and/or processable and processed items (**Proc Items**). You can even choose that they be

available for all three statuses by ticking the checkboxes, as shown in [Figure 2.31](#).



Select	Field Name	Raw Items	Proc Items	Data element	Short Description
<input type="checkbox"/>	ZZPOB_CUR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPOB_WAERS	POB Currency
<input type="checkbox"/>	ZZPOB_DESC	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZZDESC	POB Description
<input type="checkbox"/>	ZZPOB_STATUS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPOB_STATUS	POB Status

Figure 2.31 Customer Fields Selected for Class YA01

Ideally, instead of directly adding to structure INCL_EEW_FARR_POB, you can create a group of custom fields that you want to add and create each group as different append structures. This way, you can create multiple append structures for different categories of fields. Finally, you just add these structures to INCL_EEW_FARR_POB. This improves modularity and reusability. To elaborate, let's say there is a set of fields that are required for POB reporting such as the description of POB status and other descriptions. You can add them as an append structure ZZPOB_DISP, and this structure can then be directly used in INCL_EEW_FARR_REP if required for reporting purposes.

Consider the following fields that will be added:

- ZZPOB_CUR
- ZZPOB_STATUS
- ZZPOB_BUNDLE
- ZZPOB_DESC
- ZZPOB_STATU_DESC

This list can be divided into two sets as ZZPOB_MAIN and ZZPOB_DESC, as shown in [Table 2.3](#).

Append Structure ZZPOB_MAIN	Append Structure ZZPOB_DESC
ZZPOB_CUR	ZZPOB_DESC
ZZPOB_STATUS	ZZPOB_STATU_DESC
ZZPOB_BUNDLE	

Table 2.3 Append Structure Sets

To structure INCL_EEW_FARR_POB, you can add both append structures: ZZPOB_MAIN (which carried the essential processing data) and ZZPOB_DESC (description and other texts). Now for reporting, if you need to add fields ZZPOB_DESC and ZZPOB_STATUS_DESC, then you could simply add the append structure ZZPOB_DESC to structure INCL_EEW_FARR_REP.

This way, you can segregate the fields and create structures. These structures are then reused to add them to any of the four structures.

2.3.3 Tables

Tables that are created during the generation of RAI classes and the standard tables that are most commonly used in RAR are discussed in this section. We'll also explain how to add indexes and how to view the table structures.

Generated Results

The RAI class is activated by clicking on the **Activate** option from the **Configuration** menu, as shown in [Figure 2.32](#).

Once the class is active, you need to generate it by executing Transaction FARR_IMG and following menu path **Revenue Accounting • Inbound Processing (Classic) • Generate Interfaces for Revenue Accounting Item Classes**.

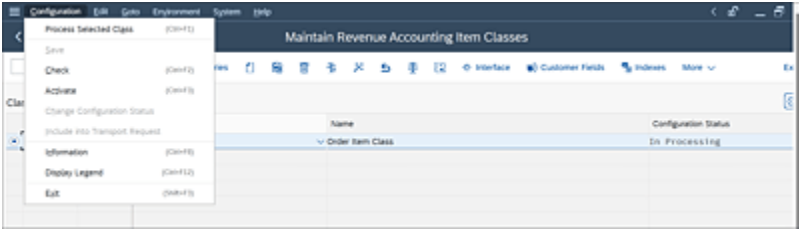


Figure 2.32 Class YA01 Activation

There, you select the class to generate, which is **YA01** in this example, and then click on **Generate**. The screen shown in [Figure 2.33](#) will appear.

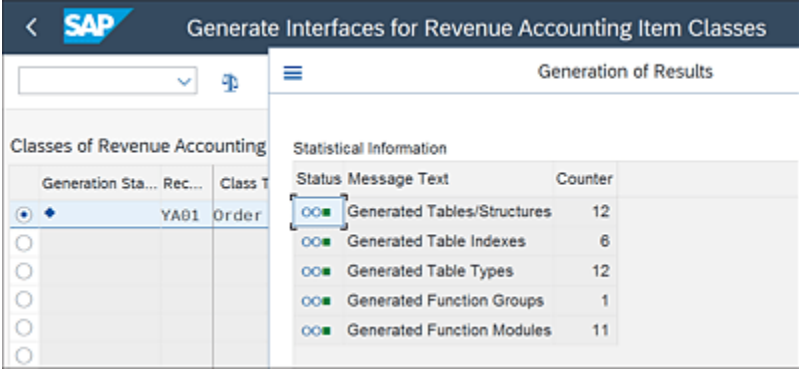


Figure 2.33 Generation of Class YA01

The interfaces are generated for RAI classes, so now what? There are numerous objects created, activated, and ready for you to use post generation. You can see a list by clicking the **Display Log** button. We've listed them categorically for ease of understanding in [Table 2.4](#).

Function Group	/1RA/YA01
Function modules	<ul style="list-style-type: none"><li data-bbox="607 317 1101 352">• /1RA/YA01_RAI_CREATE_API<li data-bbox="607 394 1045 430">• /1RA/YA01_RAI0_INSERT<li data-bbox="607 472 1045 508">• /1RA/YA01_RAI2_INSERT<li data-bbox="607 550 1045 585">• /1RA/YA01_RAI3_INSERT<li data-bbox="607 627 1045 663">• /1RA/YA01_RAI4_INSERT<li data-bbox="607 705 1045 741">• /1RA/YA01_RAI0_UPDATE<li data-bbox="607 783 1045 819">• /1RA/YA01_RAI1_UPDATE<li data-bbox="607 861 1045 896">• /1RA/YA01_RAI2_UPDATE<li data-bbox="607 938 1045 974">• /1RA/YA01_RAI3_UPDATE<li data-bbox="607 1016 1045 1052">• /1RA/YA01_RAI4_UPDATE

Function Group	/1RA/YA01
Table types	<ul style="list-style-type: none"> • /1RA/0YA010MI_TAB • /1RA/1YA010MI_API_TAB • /1RA/0YA011MI_TAB • /1RA/0YA012MI_TAB • /1RA/0YA013MI_TAB • /1RA/0YA014MI_TAB • /1RA/0YA010CO_TAB • /1RA/1YA010CO_API_TAB • /1RA/0YA011CO_TAB • /1RA/0YA012CO_TAB • /1RA/0YA013CO_TAB • /1RA/0YA014CO_TAB
Table index	<ul style="list-style-type: none"> • /1RA/0YA010MI 001 • /1RA/0YA010MI 002 • /1RA/0YA012MI 001 • /1RA/0YA012MI 002 • /1RA/0YA014MI 001 • /1RA/0YA014MI 002

Table 2.4 List of Generated Objects

Now we start with the tables that will be created after the generation of the RAI classes. With the generation of the classes, the created tables store the RAIs in various statuses. Each status of the RAIs of an RAI class has two database tables created: one for the main item (MI) and one for the condition item (CO). [Table 2.5](#) lists the important RAI tables that were created along with their status descriptions.

Tables Created	Description
/1RA/0YA010MI	Main items - Raw
/1RA/1YA010MI_API	Class YA01 structure type API - main items
/1RA/0YA011MI	Main items - Exempt Raw
/1RA/0YA012MI	Main items - Processable
/1RA/0YA013MI	Main items - Exempt Processable
/1RA/0YA014MI	Main items - Processed
/1RA/0YA010CO	Condition items - Raw
/1RA/1YA010CO_API	Class YA01 structure type API - condition items
/1RA/0YA011CO	Condition items - Exempt Raw
/1RA/0YA012CO	Condition items - Processable
/1RA/0YA013CO	Condition items - Exempt Processable
/1RA/0YA014CO	Condition items - Processed

Table 2.5 Important RAI Tables

Let's understand what the tables are stored with, their attributes, and their structure, as well as how it matches with what we've configured. The table names are automatically created by the system, and they follow a particular convention: "/1RA" is common throughout all the tables and then follows "/0" or "/1", and then the class name. In our case, it's "YA01" for the order. Similarly, the RAI class for fulfillment would be "YA02" and for invoice would be "YA03". Then, the statuses are used as follows: "0" for **Raw**, "1" for **Exempt Raw**, "2" for **Processable**, "3" for **Exempt Processable**, and "4" for **Processed**. The final letters are "MI" for main items and "CO" for condition items. This forms the core of RAR tables; of course, we have a lot more tables to discuss in this section going forward.

Using Transaction FARR_RAI_MON, you can see the RAIs in all statuses. This is a very powerful utility that lets you display, test, transfer, exempt, restore, and process the RAIs. We'll investigate some details next.

Raw Items: OMI and OCO

The main items for raw belonging to order, fulfillment, and invoice classes are stored in the tables ending with /1RA/0****0MI, as shown in [Figure 2.34](#) under the **Fields** tab. Similarly, the condition items are stored in the tables with naming convention /1RA/0****0CO, as shown in [Figure 2.35](#).

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		Client
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCCO	CHAR	3	0		Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCLS	CHAR	10	0		Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCTY	CHAR	4	0		Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCID	CHAR	35	0		Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_TMSIME	DEC	15	0		Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> KEYYP	<input type="checkbox"/>	<input type="checkbox"/>	FARR_KEYYP	NUMC	3	0		Subarea for Parallelization
<input type="checkbox"/> BAIC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_BAIC_TYPE	CHAR	2	0		Revenue Accounting Item Class Type
<input type="checkbox"/> BAIC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_BAIC	CHAR	4	0		Revenue Accounting Item Class

Figure 2.34 Raw Items Main Records: /1RA/0YA010MI

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		Client
<input type="checkbox"/> KEYYP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_KEYYP	NUMC	3	0		Subarea for Parallelization
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCCO	CHAR	3	0		Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCLS	CHAR	10	0		Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCTY	CHAR	4	0		Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_SRCID	CHAR	35	0		Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_BAL_TMSIME	DEC	15	0		Timestamp UTC for Revenue Accounting
<input type="checkbox"/> CONDITION_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KSCHA	CHAR	4	0		Condition Type

Figure 2.35 Raw Items Condition Records: /1RA/0YA010CO

Under the **Attributes** tab for these tables, you can see that they are all created as \$TMP or local objects. They are all available only after the classes are generated. So, when the configuration is transported to the next system, such as the test system and then the production system, the RAI classes are created and generated in each system, and then the tables are created in the system.

As we know, the first status of the RAI is the **Raw** status. To further process the RAIs and create contracts and POBs, you need to transfer the RAIs to the **Processable** status in one of two ways:

- **Mass transfer**
If you have a huge number of RAI items to be transferred,

then go to Transaction FARR_RAI_TRANS (Transfer Revenue Accounting Items) or program RFARR_RAI_PP_TRANS_START. The key fields for mass transfer are shown in [Figure 2.36](#): the RAI class (**Rev. Acc. Itm Class**) and **Sender Component**.

The screenshot shows the SAP 'Transfer Revenue Accounting Items' dialog box. It is divided into two main sections: 'Selection Data' and 'Technical Parameters'.
In the 'Selection Data' section, there are five rows of fields, each with a 'to:' field and a copy icon to its right:
1. 'Rev. Acc. Itm Class': [input field] to: [input field]
2. 'Sender Component': [input field] to: [input field]
3. 'Source Item Logical System': [input field] to: [input field]
4. 'Header ID': [input field] to: [input field]
5. 'Company Code': [input field] to: [input field]
There is also a 'Further Selections Exist:' checkbox at the bottom of this section.
The 'Technical Parameters' section contains:
- 'Number of Intervals': [input field with value 2]
- 'Block Size For Mass Selection': [input field with value 1,000]
- 'Simulation Mode':
- 'Dialog Mode':
- 'Synchronous Call':

Figure 2.36 Transfer Revenue Accounting Items

You can also provide the **Header ID** if you're running it for a specific header ID, but because this is for mass selection, you generally wouldn't provide any restricting inputs. **Number of Intervals** determines the number of jobs for the transfer and usually is defaulted to **2**, but you can change it per your requirement. The **Block Size For Mass Selection** will determine the package size for mass processing.

You can execute in the background by following menu path **Program • Execute in Background**, as shown in [Figure 2.37](#). Alternatively, depending on the volume of data, you can run it in the foreground via Transaction FARR_RAI_TRANS by clicking on the **Execute** button or

pressing **Ctrl+F8** to transfer the RAIs matching the selection from the status **Raw** to the status **Processable**.

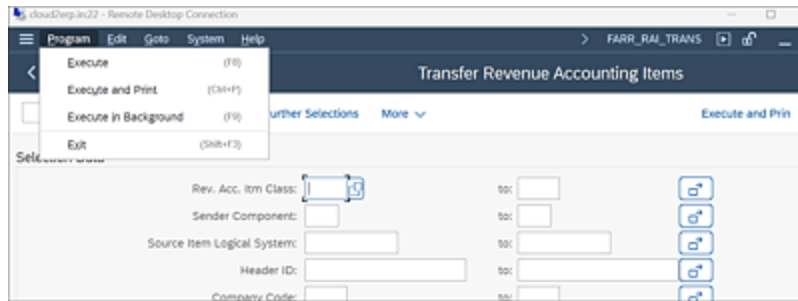


Figure 2.37 Transaction FARR_RAI_TRANS Executing in the Background

- **Selected RAIs transfer or dialog transfer**

Alternatively, you can call Transaction FARR_RAI_MON and provide the selection that is specific for your requirements, as shown in [Figure 2.38](#).

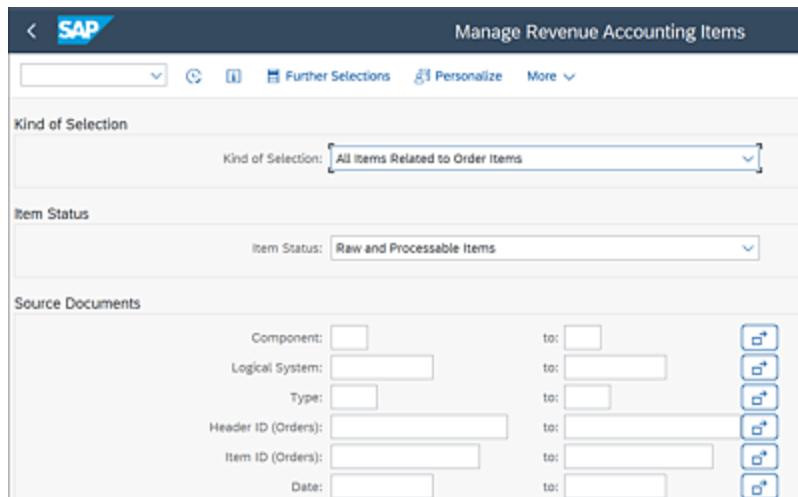


Figure 2.38 Transaction FARR_RAI_MON for Transferring of RAIs from Raw to Processable

You can choose **All Items Related to Order Items** or **All Items Related to Invoice Items**, or you can provide the **Header ID** for which you want the RAIs to be selected. Click on **Execute**, and you'll be prompted with a list of items that are available in **Raw** status. Then, from the list

shown in [Figure 2.39](#), choose the RAIs you want to transfer, and click the **Transfer** button that is provided. This way, you choose the specific items that you want to transfer to **Processable** status. We'll talk about Transaction FARR_RAI_MON in detail in later sections.

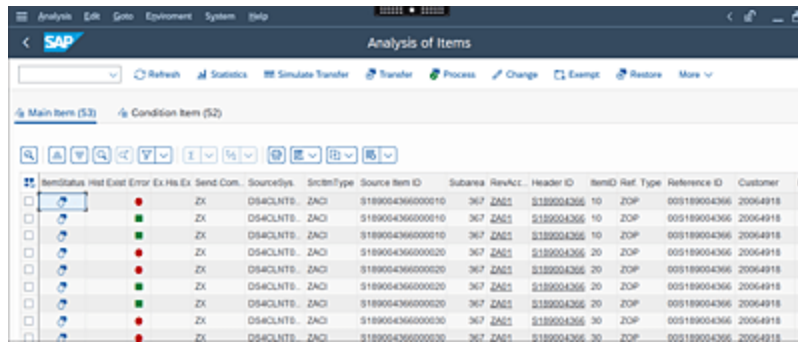


Figure 2.39 Choosing the RAIs to Transfer in Transaction FARR_RAI_MON

Processable Items: 2MI and 2CO

The main processable items belonging to order, fulfillment, and invoice classes are stored in the tables ending with /1RA/0***2MI, as shown in [Figure 2.40](#). Similarly, the condition items are stored in the tables with naming convention /1RA/0***2CO, as shown in [Figure 2.41](#).

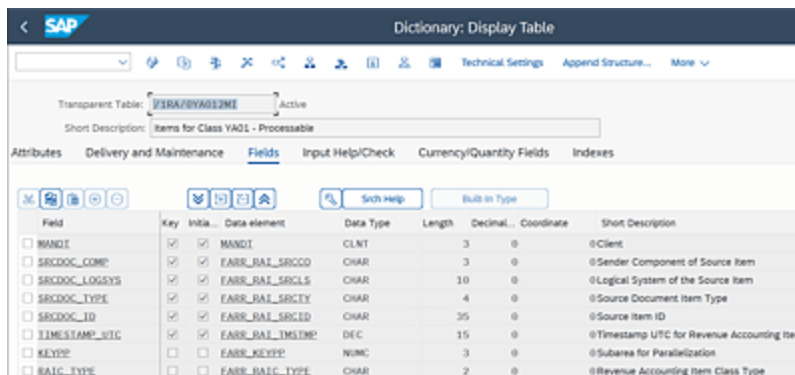


Figure 2.40 Processable Items Main Records: Table /1RA/0YA012MI

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		@Client
<input type="checkbox"/> KEYYP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_KEYYP	NUMC	3	0		@Subarea for Parallelization
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAL_SRCDO	CHAR	3	0		@Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAL_SRCLS	CHAR	10	0		@Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAL_SRCCTY	CHAR	4	0		@Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAL_SRCID	CHAR	35	0		@Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAL_TMSIMP	DEC	15	0		@Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> CONDITION_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KSOBA	CHAR	4	0		@Condition Type
<input type="checkbox"/> PL_ACCOUNT	<input type="checkbox"/>	<input type="checkbox"/>	FARR_PL_ACCOUNT	CHAR	10	0		@Profit and Loss Account

Figure 2.41 Processable Items Condition Records: Table /1RA/0YA012CO

The data from **Processable** status is transferred to the **Processed** status in two ways, similar to the previous section:

- **Mass process**

If you have a huge number of RAI items to be processed, then go to Transaction FARR_RAI_PROC (Process Revenue Accounting Items) or program RFARR_RAI_PP_PROC_START. Provide the selection criteria as required (see [Figure 2.42](#)), and then execute. Again, you can choose whether to run in the background or foreground by following the same steps described in the previous section.

Process Revenue Accounting Items

Selection Data

Rev. Acc. Item Class: to:

Sender Component: to:

Source Item Logical System: to:

Header ID: to:

Reference Type: to:

Reference ID: to:

Company Code: to:

Further Selections Exist:

Technical Parameters

Number of Intervals:

Block Size For Mass Selection:

Dialog Mode:

Synchronous Call:

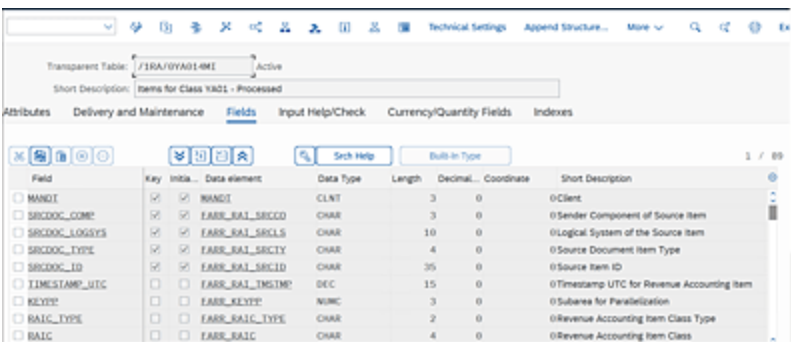
Figure 2.42 Process RAIs

The system processes the items matching the selection from the status **Processable** to the status **Processed**.

- **Selected RAIs processing or dialog processing**
You can call Transaction FARR_RAI_MON and provide the selection that is specific for your requirement or choose from the provided list and then click on the **Execute** button that is provided. This way, you can choose the specific items that you want to transfer from **Processable** to **Processed** status.

Processed Items: 4MI and 4CO

The main processed items belonging to order, fulfillment, and invoice classes are stored in the tables ending with /1RA/0****4MI, as shown in [Figure 2.43](#). Similarly, the condition items are stored in the tables with naming convention /1RA/0****4CO, as shown in [Figure 2.44](#). This is the final status, and the RAIs here are final as well. Successfully processed RAIs will create/update revenue accounting contracts and POBs based on rules set in BRFplus.



Field	Key	Intra...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		Client
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCDO	CHAR	3	0		Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCLS	CHAR	10	0		Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCTY	CHAR	4	0		Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAI_TMSIMP	DEC	15	0		Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> KEYSI	<input type="checkbox"/>	<input type="checkbox"/>	FARR_KEYTYP	NUMC	3	0		Subarea for Parallelization
<input type="checkbox"/> BAIC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_BAIC_TYPER	CHAR	2	0		Revenue Accounting Item Class Type
<input type="checkbox"/> BAIC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_BAIC	CHAR	4	0		Revenue Accounting Item Class

Figure 2.43 Processed Items Main Records: Table /1RA/0YA014MI

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		Client
<input type="checkbox"/> KEYPP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_KEYPP	NARC	3	0		Subarea for Parallelization
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCDO	CHAR	3	0		Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCLS	CHAR	10	0		Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCITY	CHAR	4	0		Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		Source Item ID
<input type="checkbox"/> CONDITION_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KSCDA	CHAR	4	0		Condition Type
<input type="checkbox"/> TIMESTAMP_SEC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAI_TSTMP	DEC	15	0		Timestamp UTC for Revenue Accounting Item

Figure 2.44 Processed Items Condition Records: Table /1RA/0YA014CO

Note that the preceding activities have all been shown for the order class, which in this case is YA01. The steps would be the same for fulfilment classes and invoice classes, and the same objects would be created. The only difference is that the class name would vary based on the fulfillment class or invoice class.

Standard Tables

In addition to the tables that are generated with the creation of RAI classes, there are many standard tables that are related to RAR. There is a huge list of important tables related to RAR, and we’ve listed the core tables in the [Table 2.6](#).

Table Name	Description
FARR_D_ACCRRUN	Accrual Run Table
FARR_D_ATTACH	SDOK: Table for Document Contents (Import/Export)
FARR_D_BIZ_RECON	Business Reconciliation Table
FARR_D_CATCHUP	Revenue Catch-Up

Table Name	Description
FARR_D_CHECK_STA	Data Consistency Check Selection Run Status
FARR_D_CHG_MIG	Change Type Migration Table
FARR_D_CHG_TYPE	Table of Change Type
FARR_D_COLOGFLD	Changed Fields of RAR, Database Changes
FARR_D_COLOGHDR	Header Entries of DB Changes
FARR_D_COLOGITM	Logs of Item DB Changes
FARR_D_COLOGTEC	Technical Table
FARR_D_CONS	RA - Consistency Check/Contract Data
FARR_D_CONT_ERR	Log Messages of Contract Errors
FARR_D_CONTR_D	Draft Table for Revenue Accounting Contract
FARR_D_CONTRACT	Contracts
FARR_D_C00BJNR	CO Object Number Table for Migration
FARR_D_COST_CO	Processed Cost RAs - Condition Items
FARR_D_COST_MI	Processed Cost RAs - Main Items
FARR_D_DEFERRAL	Deferrals
FARR_D_DEFITEM	Deferral Items
FARR_D_DELDEFITM	Deletion Update for Deferral Item (Memorized for BI)

Table Name	Description
FARR_D_DPP_SORTC	DPP EoP: Information on Customer Retention
FARR_D_DPP_SORTP	DPP EoP: Information on Bus. Partner Retention
FARR_D_EV_CONTR	Events That Occurred for Contracts
FARR_D_FLFMT_MI	Processed Fulfillment RAIs - Main Item
FARR_D_FREEZ_IDX	Freeze Period Index Table for AIF Bulk Handling
FARR_D_FRZ_C_IDX	Freeze Period by Contract Index Table for AIF Bulk Handling
FARR_D_FULFILLMT	Fulfillments
FARR_D_FX2_EXPL	Data for Revenue Exchange Rate Explanation
FARR_D_INB_CO	Postponed RAIs - Condition Items
FARR_D_INB_COPA	Postponed RAIs - COPA Data
FARR_D_INB_MI	Postponed RAIs - Main Items
FARR_D_INV_CO	Processed Invoice RAIs - Condition Items
FARR_D_INV_FX_ED	Distributed Invoice and Exchange Rate Difference
FARR_D_INV_MI	Processed Invoice RAIs - Main Items
FARR_D_INVOICE	Invoice Entries

Table Name	Description
FARR_D_ITEM_PROC	Inbound Processing: Item Proc. Status for Account Principle
FARR_D_JOB	Accrual Run in Background Job
FARR_D_KEYPP_ENQ	Table Is Used Only by ENQUEUE-Object EFKK_KEYPP
FARR_D_MAPPING	Mapping Table to Map Source Document to POBs of RA-Contracts
FARR_D_MAPPING_D	Draft Table for RA Mapping Operational Document
FARR_D_MAPPING_F	Mapping Table for Fulfillment
FARR_D_MAPPING_I	Mapping Table for Invoices
FARR_D_MAPPING_M	Mapping of Manual POBs between Accounting Principles
FARR_D_MNL_CHG	Manual Changes of Performance Obligations
FARR_D_NOTES	Table for Farr Notes
FARR_D_OBJKEYS	Data Consistency Check Object Keys (New)
FARR_D_OBJKEYS_E	Data Consistency Check Object Keys with Errors
FARR_D_ORD_C0	Processed Order RAIs - Condition Items
FARR_D_ORD_MI	Processed Order RAIs - Main Items

Table Name	Description
FARR_D_PA0BJMAP	Profitability Segment Hashing
FARR_D_POB	Performance Obligations
FARR_D_POB_CORRT	Correct Performance Obligations before Contract Turns to Err
FARR_D_POB_CTYPE	Performance Obligation Change Type
FARR_D_POB_D	Draft Table for RA Performance Obligation
FARR_D_POB_FRZ	Freeze Periods of Performance Obligations
FARR_D_POB_HIS	History of POB/Contract Structure Changes
FARR_D_POSTING	Postings
FARR_D_POSTING_C	Detailed Postings from Cross-Period Aggregation
FARR_D_POSTING_P	Detailed Postings from Inner-Period Aggregation
FARR_D_POSTING_S	Postings (Shadow Table)
FARR_D_PP_LOCK	Revenue Accounting: Locked RAIs per Mass Run
FARR_D_PP_LOCKED	Obsolete
FARR_D_PP_PACK	Package-Related Parameters for Parallel Processing
FARR_D_PP_UNITS	Unit Related Parameters for Parallel Processing

Table Name	Description
FARR_D_PPND_REV	Postponed Revenue and Cost Recognition Items
FARR_D_PRODDEL	Productive Cleanup: Deletion Log for Order Item Header IDs
FARR_D_RAI_CH	Change Sequence of Changed Items
FARR_D_RAI_DELH	History Table for Deletion of Exempted Items
FARR_D_RAI_HIST	History Table for Revenue Accounting Item Exemptions
FARR_D_RAI_LOG	Log of Data Storage of RAIC Items
FARR_D_RAI2_ERR	RAI2 Processing Inconsistencies
FARR_D_RAI2_PROC	Acct. Principles RAI2 Was Successfully Processed For
FARR_D_RECKEY_S	Reconciliation Keys (Shadow Table)
FARR_D_RECON_ERR	History Table for Reconciliation between Logistics and RA
FARR_D_RECON_HIS	History Table for Reconciliation between Logistics and RA
FARR_D_RECON_KEY	Reconciliation Keys
FARR_D_RVS_RUNID	Run ID Reverse History
FARR_D_SHIFT_HIS	Error Contract Shift History

Table 2.6 List of RAR Tables

We'd like to provide some insight on the main tables used extensively in most of our developments, as listed in [Table 2.7](#). These tables are basic building blocks of RAR and the relationships between them will be explained in [Section 2.3.4](#). We consider these tables as some of the most important standard tables in RAR.

Tables	Description
FARR_C_BUKRS	Company Code Relate
FARR_C_CLOSE	Revenue Accounting Period Close
FARR_D_CONTRACT	Contracts
FARR_D_DEFERRAL	Deferrals
FARR_D_DEFITEM	Deferral Item
FARR_D_INVOICE	Invoice Entries
FARR_D_MAPPING	Mapping Table to Map Source Document to POBs of Results Analysis Contracts
FARR_D_MAPPING_D	Draft Table for Results Analysis Mapping Operational Document
FARR_D_POB	Performance Obligations
FARR_D_POSTING	Postings
FARR_D_RECON_KEY	Reconciliation Keys

Table 2.7 List of Very Important and Frequently Used RAR Tables

Adding Indexes to the Tables

As we all know, indexes are created for faster access to the data records of tables. Some indexes are created by the system and provided by default to the raw, raw exempted, processable, processable exempted, and processed tables. You can even create additional indexes to the tables per your requirement to make data access faster. However, keep in mind that creating a lot of indexes on the huge tables isn't a good idea because that will put an additional load on the database. Because the system creates the indexes on the RAR tables, you can add any fields from the standard fields or even include the custom fields as part of the fields in an index.

As they are the custom indexes, the naming should be in the customer namespace. This is done during the RAI class creation.

To create a table index, follow these steps:

1. Execute Transaction FARR_IMG, and follow menu path **Revenue Accounting • Inbound Processing (Classic) • Maintain Revenue Accounting Item Classes**.
2. Select the class that you want to create the indexes. In this case, the class will be **YA01**.
3. Click on **Indexes**, and the screen shown in [Figure 2.45](#) appears.

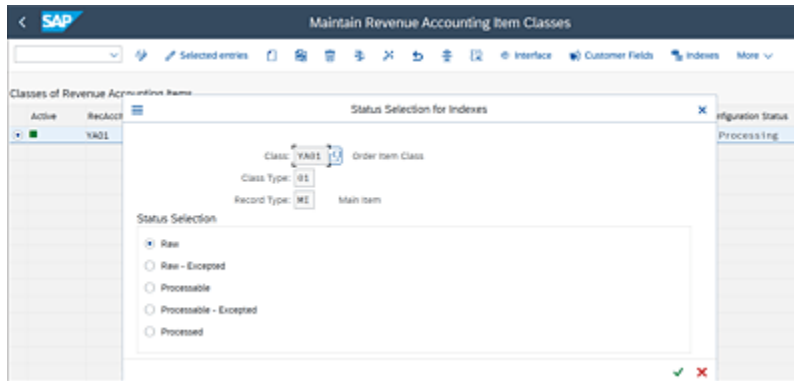


Figure 2.45 Creating Indexes on RAI Tables

4. You can only select the MI tables or the table for main records. Under **Status Selection**, select any of the statuses for the selected class:

- **Raw**
- **Raw - Exempted**
- **Processable**
- **Processable - Exempted**
- **Processed**

5. Click the green checkmark. If you've selected to create an index on the **Raw** table, then [Figure 2.46](#) appears where you can click on the **Create Index** button and create an index with the desired fields selected and then click on the green checkmark. You can see that there are already two primary indexes existing in the system: **001** and **002**. The new index must be created in the customer namespace starting with Y or Z.

The custom fields are also available for index creation.

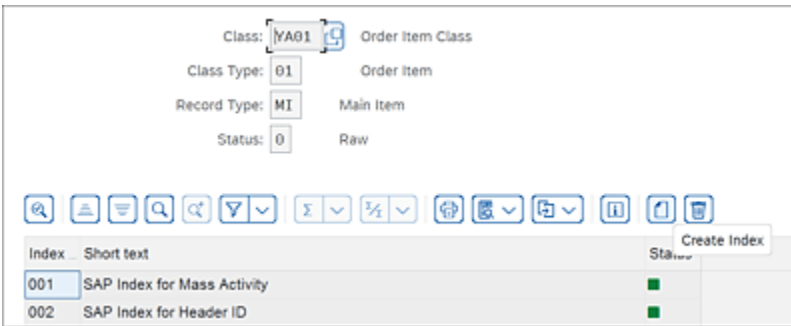


Figure 2.46 Creating the Index

View the Structure

You can view the table structure from the RAI class creation screen. Execute Transaction FARR_IMG, and follow menu path **Revenue Accounting • Inbound Processing (Classic) • Maintain Revenue Accounting Item Classes**. Select the class that you want to view the table structures for; in this case, **YA01**. Once the class is selected, choose **More • Display Table Structure** to view the table structure of the desired table, as shown in [Figure 2.47](#).

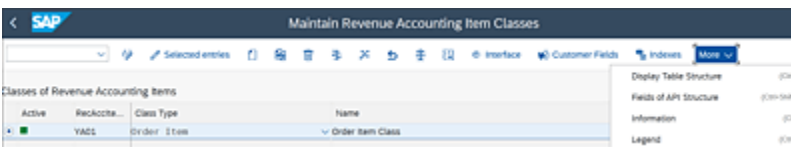


Figure 2.47 Display Table Structure

You can change the status and accordingly view the fields that are available by clicking **Other Status**, as shown in [Figure 2.48](#). You can see the structure of different tables of main or condition items in **Raw**, **Processable**, **Processed**, **Raw - Exempted**, and **Processable - Exempted** statuses.

Class: YA01 Order Item Class
 Class Type: 01 Order Item
 Status: 2 Processable

Main Items Conditions

Field Name	Tabl.	K. C.	Data element	Domain	A. Data Type	No. of ...
MANDT	1	<input checked="" type="checkbox"/>	MANDT		C CLNT	3
SRCDOC_COMP	2	<input type="checkbox"/>	FARR_RAI_SRCCD		C CHAR	3
SRCDOC_LOGSYS	3	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS		C CHAR	10
SRCDOC_TYPE	4	<input checked="" type="checkbox"/>	FARR_RAI_SRCTY		C CHAR	4
SRCDOC_ID	5	<input checked="" type="checkbox"/>	FARR_RAI_SRCID		C CHAR	35
TIMESTAMP_UTC	6	<input checked="" type="checkbox"/>	FARR_RAI_TMSTMP		P DEC	15
KEYPP	7	<input type="checkbox"/>	FARR_KEYPP		N NUMC	3
RAIC_TYPE	8	<input type="checkbox"/>	FARR_RAIC_TYPE		C CHAR	2
RAIC	9	<input type="checkbox"/>	FARR_RAIC		C CHAR	4

Figure 2.48 Display Table Structure

As in the earlier section, where we added the custom fields, we marked those customer fields to be available only in processable and processed items so we can see that the raw items don't have those custom fields included; instead, they are available only in the processable and processed items. The custom fields missing in the **Raw** status main table are shown in [Figure 2.49](#), and those available in the **Processable** table for main records are shown in [Figure 2.50](#). The custom fields are also marked in [Figure 2.50](#): **ZZPOB_CUR**, **ZZPOB_DESC**, and **ZZPOB_STATUS**.

Field Name	Tabl..	K. C.	Data element
VBELV	80	<input type="checkbox"/>	VBELV
VBTYP	81	<input type="checkbox"/>	FARR_VBTYPL
VTWEG	82	<input type="checkbox"/>	VTWEG
WERKS	83	<input type="checkbox"/>	WERKS_EXT

Figure 2.49 Custom Fields Missing in the Raw Table for Main Records

Field Name	Tabl..	K. C.	Data element	Domain
VBELV	81	<input type="checkbox"/>	VBELV	
VBTYP	82	<input type="checkbox"/>	FARR_VBTYPL	
VTWEG	83	<input type="checkbox"/>	VTWEG	
WERKS	84	<input type="checkbox"/>	WERKS_EXT	
ZZPOB_CUR	85	<input checked="" type="checkbox"/>	ZPOB_WAERS	WAERS
ZZPOB_DESC	86	<input checked="" type="checkbox"/>	ZZDESC	CHAR35
ZZPOB_STATUS	87	<input checked="" type="checkbox"/>	ZPOB_STATUS	CHAR9

Figure 2.50 Custom Fields Available in the Processable Status Table for Main Records

2.3.4 Relationships

Retrieval of data depends on the relationship between the tables. When we need to get data in our custom programs, or during the BAdI implementation, the data will have to be pulled from various different tables by connecting the tables with key fields. Sometimes, the key fields aren't connected,

but they are connected through the other fields in the database table. There will be multiple tables involved in building the necessary data for our programs and reports. Generally, functional consultants help with the data relationship model or with the list of tables that we need to access and the relation between the tables. As we work in any SAP module, we start to understand the table relations by experience. In this section, we're focusing on table relationships and the internal hierarchy that exists in the RAR system between the data that is available in the **Raw**, **Processable**, and **Processed** statuses and the other RAR tables.

[Figure 2.51](#) shows a graphical depiction of hierarchies and relationships in RAR. The topmost of the data hierarchy is the header ID that heads the chain. Under each header ID, there can be multiple contracts. Each contract will have a set of POBs that again have different types in them. The POBs can have multiple orders under them, which are identified by the key fields SRCDOC_COMP, SRCDOC_LOGSYS, SRCDOC_TYPE, and SRCDOC_ID, and then are stored in the order item tables (/1RA/0**010MI, /1RA/0**012MI, and /1RA/0**014MI under normal cases with no exemptions). We're not looking at fulfillments here.

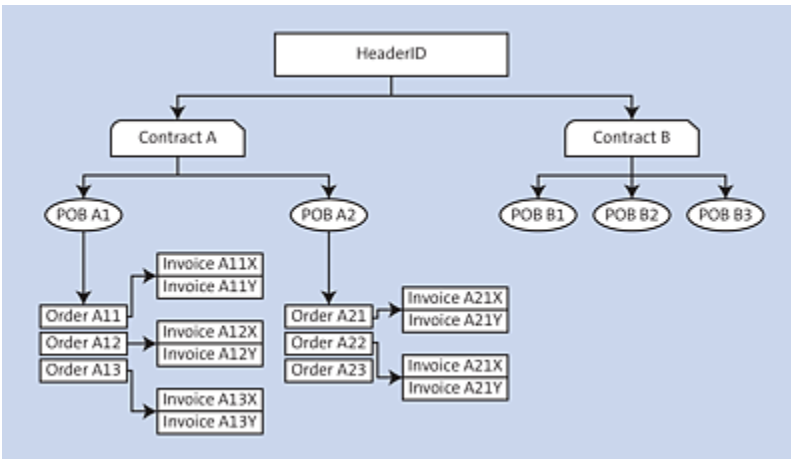


Figure 2.51 Hierarchies and Relationships

Each of these orders can have multiple invoices. Invoices are identified by the key fields SRCDOC_COMP, SRCDOC_LOGSYS, SRCDOC_TYPE, and SRCDOC_ID, and then are stored in the invoice item tables (/1RA/0**030MI, /1RA/0**032MI, and /1RA/0**034MI under normal cases with no exemptions).

Understanding the relationship between the tables is crucial for designing the data-fetching logic. During knowledge-transfer sessions that we've experienced in previous projects, we saw developers who tried to make changes to existing RAR custom programs and found it difficult to understand the hierarchical flow of data in RAR. If you're new to RAR, don't worry, we'll discuss some examples next that should help you understand how to connect tables the right way. [Figure 2.51](#) shows the header ID at the top and then the contracts and the POB. Then you can see that the RAIs can be in the **Raw**, **Processable**, and **Processed** statuses. Each of the orders can have one or more invoices associated to them. We'll discuss some sample cases to demonstrate the relationships. Important note: The

following examples assume there is no fulfillment class created. We are using the order class and invoice class for these examples. The next example helps us with data retrieval for specific cases when we have a certain input provided on the selection screen, for example, and we have to get order or invoice RAIs.

Case 1: Get the Processed Invoice RAIs for a Header ID and Specific Source Document ID (Processed Order RAIs)

In this case, say we have a custom program. From the selection screen of the program, we get the header ID and source document IDs of the order in **Processed** status, and we have to fetch the processed invoice items.

Select SRCDOC_COMP, SRCDOC_LOGSYS, SRCDOC_TYPE, SRCDOC_ID, and HEADER_ID from the processed order items (in our case, the table name will be /1RA/0YA014MI) based on the HEADER_ID and the specific SRCDOC_ID on the selection screen.

Now we need to fetch the processed invoices for these SRCDOC_IDs from the processed invoice items (table /1RA/0YA034MI). The processed invoice item table also has SRCDOC_ID, but it's not the same as the processed order items. We have to pass the order item's SRCDOC_ID along with SRCDOC_COMP, SRCDOC_LOGSYS, and SRCDOC_TYPE to the processed invoice items (table /1RA/0YA034MI) as ORIGDOC_ID along with other fields (ORIGDOC_LOGSYS, ORIGDOC_TYPE, and ORIGDOC_I) being mapped, as shown in [Figure 2.52](#).

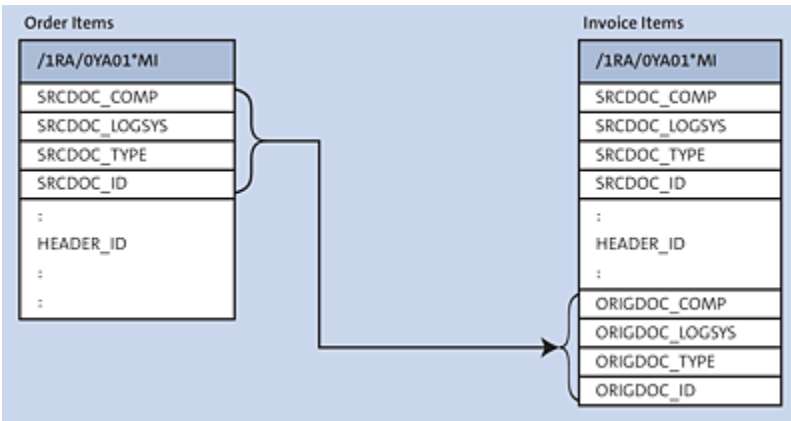


Figure 2.52 Orders to Invoice Relationship

Case 2: Get Processed Invoice RAIs for the Range of POB IDs

Consider a case where you have to develop a program for which the user has provided the POB ID on the selection screen, and then you have to pull all the processed invoice items for that particular POB ID or for a range of POB IDs. To do this, you have to first validate the POB_ID or the range of POB_IDs from the selection screen with table FARR_D_POB, which is a POB master table.

Now you need to get the SRCDOC_ID of the orders, for which you need to go mapping table FARR_D_MAPPING, which will provide the mapping between POB and SRCDOC_ID of orders only. Table FARR_D_MAPPING's SRCDOC_ID maps with orders and not with invoices.

Now collect order SRCDOC_ID along with SRCDOC_COMP, SRCDOC_LOGSYS, and SRCDOC_TYPE, followed by pulling the invoices for the retrieved orders.

You have to pass order SRCDOC_ID, along with SRCDOC_COMP, SRCDOC_LOGSYS, and SRCDOC_TYPE to the

processed invoice items (table /1RA/0YA034MI) as ORIGDOC_ID along with other fields (ORIGDOC_LOGSYS, ORIGDOC_TYPE, and ORIGDOC_ID), as shown in [Figure 2.53](#).

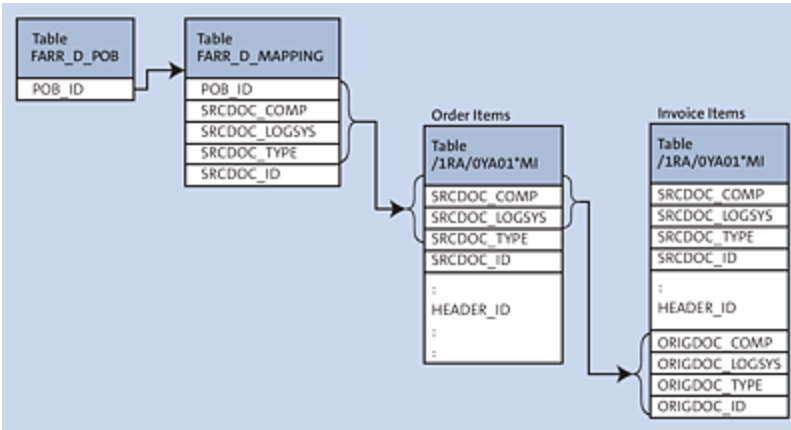


Figure 2.53 POB_ID to Processed Invoice RAIs

Case 3: Get Processed Invoice RAIs for a Contract or Range of Contracts

Consider a case where you've been provided with the contract ID, and you have to retrieve the processed invoices items. They may also have filtering based on POB_IDs. So now you have to start building the table relationship from contract to processed invoice RAIs.

The first step is to validate the CONTRACT_ID or the range of CONTRACT_IDs, which is on the selection screen with contract master table FARR_D_CONTRACT.

Because the contracts are linked to SRCDOC_ID of the orders through the POB_IDs, you need to build a link between CONTRACT_ID and POB_ID and then to SRCDOC_ID (orders).

You now go to table FARR_D_MAPPING table with the CONTRACT_ID. Table FARR_D_MAPPING always comes to the rescue when you need to build the relationships. Use the existing primary index on CONTRACT_ID for performance optimization and get all the POB_IDs for all contracts. You will also have to select the order SRCDOC_ID, and SRCDOC_COMP, SRCDOC_LOGSYS, and SRCDOC_TYPE from table FARR_D_MAPPING.

Now that you have order SRCDOC_IDs, along with SRCDOC_COMP, SRCDOC_LOGSYS, and SRCDOC_TYPE, you can pull the invoices for the retrieved orders. To do so, you have to pass order SRCDOC_ID along with SRCDOC_COMP, SRCDOC_LOGSYS, and SRCDOC_TYPE to the processed invoice items (table /1RA/0YA034MI) as ORIGDOC_ID along with other fields (ORIGDOC_LOGSYS, ORIGDOC_TYPE, and ORIGDOC_ID), as shown in [Figure 2.54](#).

If there is a condition provided for restricting the selection to specific POB IDs, then you can filter the data by deleting the other POB IDs that aren't required with a delete statement on the internal table with conditions.

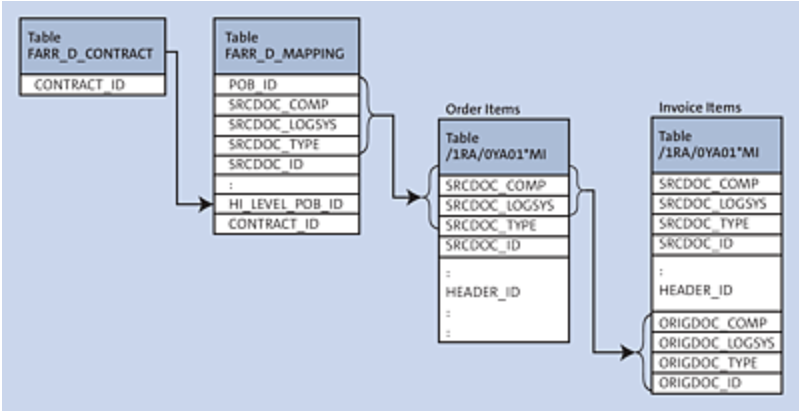


Figure 2.54 Contracts to Processed Invoice RAIs

2.3.5 Extensions and Transports

As part of extensions, we'll look at additional topics related to changing and processing data. We'll also discuss the transport-related issues that you might face when moving table changes across systems. The transport issues aren't specific to RAR, but we have encountered them in a past RAR project and felt it would be useful to include.

Transaction FARR_RAI_MON

Transaction FARR_RAI_MON is where you can see the RAIs in all statuses. This is an amazing utility that lets you display, test, transfer, exempt, restore, and process the RAIs, as well as modify the field contents of RAIs that are configured to be modifiable.

In the **Kind of Selection** dropdown, as shown in [Figure 2.55](#), you can choose to display items from orders, fulfillment, invoices, or all of them.

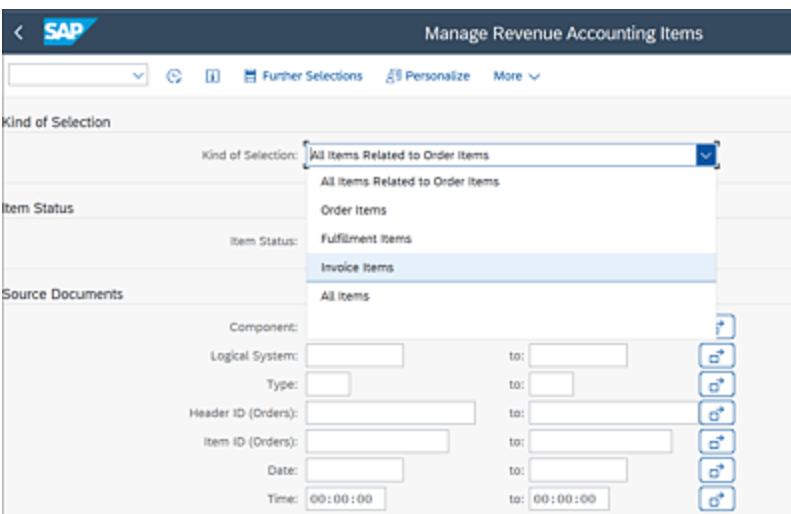
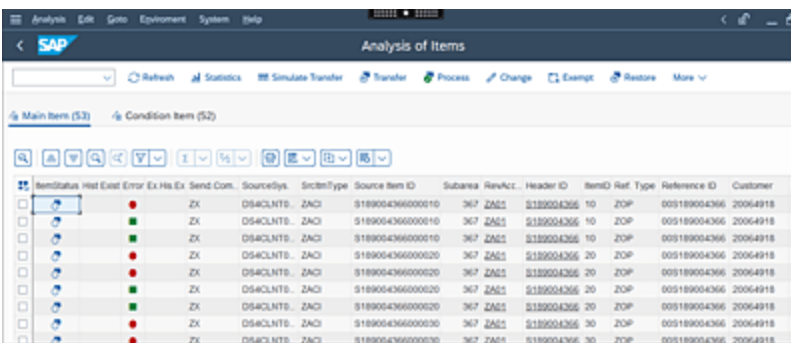


Figure 2.55 Transaction FARR_RAI_MON Screen for Selection

Additionally, you can add the **Item Status** in your selection criteria. Under the **Source Documents** section of the selection screen, you can filter your selection by passing values such as the sender **Component**, **Logical System**, **Type**, **Header ID (Orders)**, **Item ID (Orders)**, and other selections.

Click the **Execute** icon to get the output shown in [Figure 2.56](#). You can switch between the **Main Item** and **Condition Item** tabs on this very user-friendly screen. Note that this isn't for mass processing and is only for dialog and limited data processing.

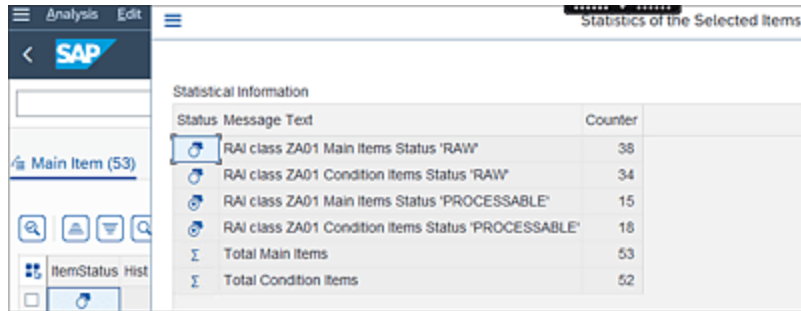


Item Status	Item ID	Source	Reference ID	Customer
ZX	DS4CLNTE	ZACI	S189004366000010	20064918
ZX	DS4CLNTE	ZACI	S189004366000010	20064918
ZX	DS4CLNTE	ZACI	S189004366000010	20064918
ZX	DS4CLNTE	ZACI	S189004366000020	20064918
ZX	DS4CLNTE	ZACI	S189004366000020	20064918
ZX	DS4CLNTE	ZACI	S189004366000020	20064918
ZX	DS4CLNTE	ZACI	S189004366000020	20064918
ZX	DS4CLNTE	ZACI	S189004366000030	20064918
ZX	DS4CLNTE	ZACI	S189004366000030	20064918
ZX	DS4CLNTE	ZACI	S189004366000030	20064918

Figure 2.56 List of Items Available for Transfer, Process, Exempt, and Restore in Transaction FARR_RAI_MON

Here, you can select the following buttons:

- **Refresh**
On clicking the **Refresh** button, all the database selections are performed again, and the latest data is displayed from database.
- **Statistics**
This button opens the screen shown in [Figure 2.57](#), which summarizes the RAIs available in different statuses and gives a count of the same.



Status Message Text	Counter
RAI class ZA01 Main Items Status 'RAW'	38
RAI class ZA01 Condition Items Status 'RAW'	34
RAI class ZA01 Main Items Status 'PROCESSABLE'	15
RAI class ZA01 Condition Items Status 'PROCESSABLE'	18
Total Main Items	53
Total Condition Items	52

Figure 2.57 Statistics as in Transaction FARR_RAI_MON

- **Simulate Transfer**

You can simulate the transfer of RAIs that are in statuses from **Raw** to **Processable**. Because it's in simulation mode, you can't commit to the database, but you can see the feasibility or error.

- **Transfer**

This button is for the actual transfer of raw RAIs to processable RAIs by committing to the database.

- **Process**

The processable RAIs are processed. Contract creation happens at this stage.

- **Change**

In the configuration, there is a provision to set some fields as modifiable. Execute Transaction FARR_IMG, and follow menu path **Revenue Accounting • Inbound Processing (Classic) • Define Modifiable Fields for Revenue Accounting Items**. You can choose the fields that you want to modify by entering them in the screen shown in [Figure 2.58](#).



Figure 2.58 Define Modifiable Fields for Revenue Accounting Items

You can modify the content of the fields listed in the preceding configuration using Transaction FARR_RAI_MON. In our example, the field **PRCTR** is listed as modifiable, so in Transaction FARR_RAI_MON, when you select an RAI and click on **Change**, you can see that the **Profit Center** field (corresponding to **PRCTR**) is editable and others aren't, as shown in [Figure 2.59](#).

Item ID	Subarea	RevAcc.	Header ID	ItemID	Ref. Type	Reference ID	Customer	Partner	CoCd	Profit Center	ProcTime	Final Time	Final.	Targ.
34366000010	367	ZA01	S199004366	10	ZOP	00S199004366	20064918		1010	CH64212701	0	0		
34366000010	367	ZA01	S199004366	10	ZOP	00S199004366	20064918		1010	CH64212701	0	0		
34366000010	367	ZA01	S199004366	10	ZOP	00S199004366	20064918		1010	CH64212701	0	0		

Figure 2.59 Modifiable Fields in RAI in Transaction FARR_RAI_MON

- **Exempt**
RAIs that aren't processed further can be exempted.
- **Restore**
RAIs that are exempted can be restored and processed further.

Custom Tables Creation and Conversion during Transports

RAR has a huge number of tables, but still there will be business requirements that call for additional custom tables

to be created. It's best to minimize the creation of additional custom tables when possible because it adds a maintenance task, increases the number of deliverables to the client, and puts an additional load on the database. However, there will be situations where you can't avoid it.

You can create custom tables using Transaction SE11, but this section isn't actually about how to create custom tables; rather, it's about a specific scenario. Generally, as part of every project's naming convention, there will be project- or client-specific internal agreement or conventions for naming the custom tables and other technical objects. In one of our particular projects, for example, they had the table naming convention of table /ABCD/ORA_0001, with each table incremented to the next number until finally we reached some with two digits, for example, table /ABCD/ORA_0025. The project went live, and all the objects successfully made it to the production system. During support, there was a bug, and some changes were made to table /ABCD/ORA_0031; specifically, there were some field type changes that triggered table conversion. The table changes were captured in a transport request and moved successfully to the production system. During the next phase of the project, we had additional changes to existing objects, and there was a change to table /ABCD/ORA_0021. Note that these changes also had field type changes to existing fields and additional fields being added. The changes were saved in a transport request and moved to the test system, and it was successful with no errors.

As part of the project test cycles during final testing stages, one of the test environments was copied from the production system. The transport request with the changes

to table /ABCD/ORACLE_0021 was now moved to the new test environment, which was a production copy. When the transport request was moved to the system, it resulted in an error, and it was unclear why the transport request carrying the table changes resulted in an error in the test environment (which is a copy of the production system) while the same transport request was moved successfully to the other test environment, which isn't a production copy.

The target table data was crucial, and there were millions of records saved in it, so we couldn't risk moving that transport request to production. After analyzing this issue in detail, we found that the problem was in the table conversion that gets triggered during the transport of transport requests to production and nonproduction systems.

Let's discuss what it means when we say that the transport request carrying changes to any table triggers table conversion. If the ABAP Dictionary definition is changed by a transport request or has been changed, the database structure of the table is adjusted to the change in the ABAP Dictionary during activation. The database structure of a table can be adjusted to its changed ABAP Dictionary definition in three ways:

- **By deleting and recreating the database table**
This is when there is no data in the table. The existing table is deleted in the database and recreated.
- **By changing the database catalog (ALTER TABLE)**
If there is data in the table, this specific step is called the alter table as it tries to change the structure of the table.

- **By converting the table**

If the structural change isn't possible with ALTER TABLE, then the table is converted. So, when there is data in the table, and ALTER TABLE can't be executed by the database system, the table conversion is triggered. In this case, there is a change in table structure, field length, field types, and so on.

The conversion process includes specific steps in the system when the conversion of table is triggered when the transport requests are moved to the next systems. The following changes are all part of standard SAP:

1. **Lock the table**

Locks the table from any further changes. The table lock ensures that there are no table updates, new entries, or modifications on this table until this activity is complete.

2. **Rename the table**

You can rename the table or change the name of the table to a new temporary table name. The system renames the table in the database, and all indexes on the table are deleted. Let's call the table in action table TAB. The temporary table that will be created for table TAB (in our case, the table name is /ABCD/ORA_0021) is thus table QCMTAB (in our case, the table name is QCM/ABCD/ORA_00) The 15-character limitation leads to the truncated name. If a table named QCM/ABCD/ORA_00 already exists in the database (e.g., from a previous conversion attempt), this table is deleted before the table is renamed (didn't happen in our case).

3. **Activate the table**

The next step is the activation of the new version of the

table in the ABAP Dictionary. The system creates table TAB (table /ABCD/ORR_0021) in the database with its new structure under the name table QCM8TAB (table QCM8/ABCD/ORR_0). In addition, the system creates the primary index of the table in the database. The structure of database table QCM8TAB (table QCM8/ABCD/ORR_0) thus corresponds to the structure of table TAB (table /ABCD/ORR_0021) in the ABAP Dictionary after this step.

4. **Reload the data**

The data is copied from table QCMTAB (table QCM/ABCD/ORR_00) to table QCM8TAB (table QCM8/ABCD/ORR_0) (with the ABAP command MOVE-CORRESPONDING). After this step, the data is present in both tables, requiring additional space.

5. **Delete the QCM table**

The data in table QCMTAB (table QCM/ABCD/ORR_00) is no longer required at the end of the conversion. The table is deleted if all records could be copied from table QCMTAB into table QCM8TAB.

6. **Rename the table and create secondary indexes**

Table QCM8TAB is renamed table TAB. The system recreates the secondary indexes for the table defined in the ABAP Dictionary in the database. The system also creates the views on the table that were deleted in the first step again in the database.

7. **Unlock the lock**

The lock set on the table is released. The table is now unlocked.

In our example, basically, we changed table /ABCD/ORR_0021, and FIELDX was changed from NUMC2 to NUMC4. When this

change moved to the test system, which isn't a copy from production, it worked fine. But when the same changes moved to the test environment, which is a production copy, we could not activate the table. The dump was thrown by the system as shown in [Figure 2.60](#).

Here's what happened: During the support phase of the project, a different table named /ABCD/ORA_0031 was changed and moved to production. This transport request triggered a conversion, and the QCM table was created and named table QCM/ABCD/ORA_00. This was left in the system without being deleted.

Now the transport request carrying our main concerned table named /ABCD/ORA_0021 is moved to the test system, which is a copy of the production system. It triggers the conversion process and now tries to create the QCM table with the name QCM/ABCD/ORA_00 (as the name of the table differs only in two characters).

11:42:19	ED Log Text
🔍	Request: convert Table /ORACLE/ORA_0021 (11/18/03 22:16:19)
🔍	Process: mnrslap001_20_27082
🔍	Test activation of Table /ORACLE/ORA_0021
🔍	Test activation of Table /ORACLE/ORA_0021 successful
🔍	⚠️ A restart log already exist for table /ORACLE/ORA_0021
🔍	Conversion of table /ORACLE/ORA_0021 was restarted
🔍	Table /ORACLE/ORA_0021 is converted using shadow field updates
🔍	The conversion is continued at step 2
🔍	Type of conversion: T -> T
<hr/>	
🔍	Beginning of Step /ORACLE/ORA_0021-STEP2 (11/18/03 22:16:55)
🔍	Renaming of table /ORACLE/ORA_0021 to QCM /ORACLE/ORA_00 on the database
🔍	❌ TABLE QCM /ORACLE/ORA_00 already exist
🔍	❌ Renaming of table /ORACLE/ORA_0021 to QCM /ORACLE/ORA_00 failed

Figure 2.60 Issue Log

This is now a duplicate table. The system doesn't allow its creation, saying that the QCM table already exists and a restart log for the table is already there in the system—but it's actually for the first table /ABCD/ORA_0031.

This confusion was created due to the following reasons:

- The temporary tables had to be deleted in production as well as in the test systems, which were created as a copy from production.
- The 15-character limit for naming a table leads to the creation of table QCMTAB with truncated names (table QCM/ABCD/ORA_00).
- When the naming conventions in the projects are followed as table /ABCD/ORA_0041 and table /ABCD/ORA_0021. Then, during conversion creation of table QCMTAB, their names will change to QCM/ABCD/ORA_00 for both tables. Due to the limitation of 15 characters, this results in duplicate tables, although technically they are different tables.

How did we resolve the issue? We can see the temporary tables that are created during table conversion in Transaction SE17. When we checked Transaction SE17 in the test system, which isn't a copy of the production system, we found that there were no temporary tables in there. But, when we checked Transaction SE17 in the other test system, which is a copy of the production system, we found temporary table QCM/ABCD/ORA_00 already existing there, which was shown in the log when the table activation failed. SAP doesn't allow another table with the same name to be created, regardless of whether it's a dictionary table, temporary conversion tables, or QCM tables.

We asked the Basis team to delete all the temporary tables in the test environment, which is a production copy, and then tried activating the table—it was successfully activated.

The temporary tables can be deleted from Transaction SE14, as shown in [Figure 2.61](#), under the right authorization and approvals. You should also validate the impact of the deletion, but be careful when doing so. To delete temporary objects, go to Transaction SE14, and select the option for **Tables** (see [Figure 2.61](#)).

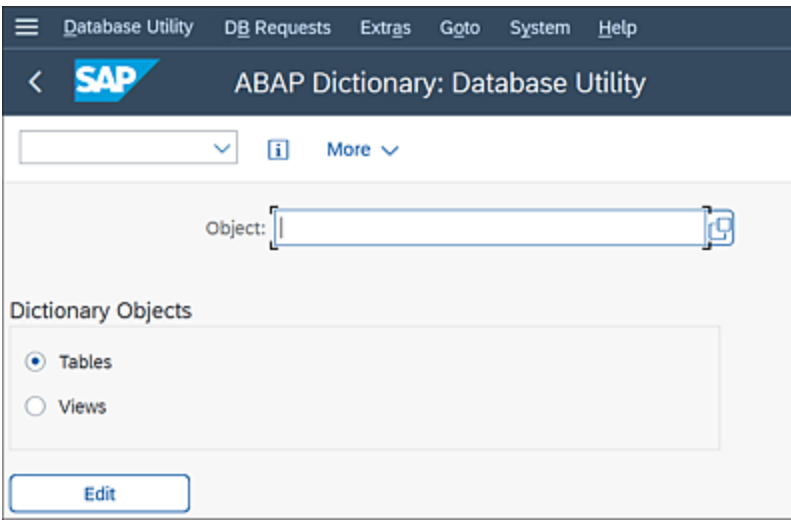


Figure 2.61 Transaction SE14 for Deleting QCM Tables

Select **Invalid Temporary Table** from the **Extras** menu, as shown in [Figure 2.62](#).

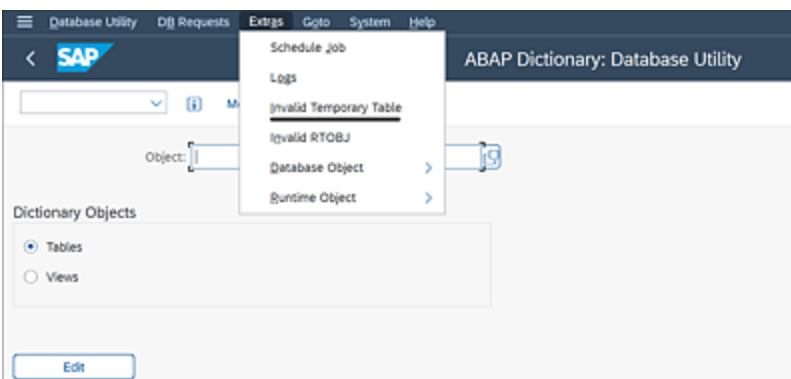


Figure 2.62 Select Invalid Temporary Tables

You'll see a list of temporary tables, as shown in [Figure 2.63](#). But because this a development system, it doesn't show

any tables here. However, in actual systems, you can see the temporary tables if there are any. Then, you can select the table that you want to delete and click on the **Delete Selected** button as highlighted here to delete the temporary table.

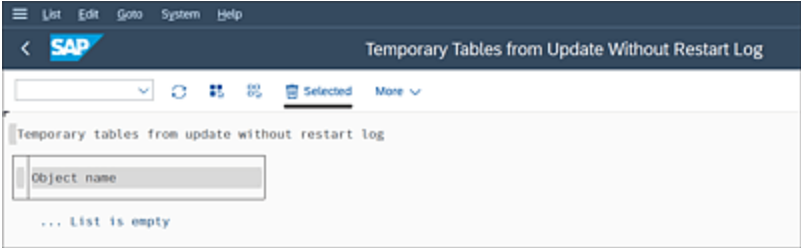


Figure 2.63 Temporary Tables Display and Deletion

2.4 Choosing Your Revenue Recognition Tool

As of the time of writing, there are two main options when it comes to selecting a RAR tool, and it mainly depends on whether you're using an older SAP ERP system or SAP S/4HANA. For SAP S/4HANA, if you're on version 1809 or later, RAR will come integrated with the system, whereas for older SAP S/4HANA users, the same rules as for SAP ERP apply. So, for SAP S/4HANA users, the decision comes down to which features of RAR they will use (will they activate optimized contract management [OCM] and optimized inbound processing [OIP] or not?), while for SAP ERP users, the decision might be more complicated, and they would need to look at feature packs and decide which one best suits their needs.

In the following sections, we'll give a detailed overview of which features come with which versions so you can make an appropriate decision.

2.4.1 RAR with SAP S/4HANA

As of SAP S/4HANA 1809, the former SAP Revenue Accounting and Reporting add-on has become an embedded part of SAP S/4HANA. This relates to product version SAP Revenue Accounting, including software component version REVREC. The RAR functionality still needs to be integrated into operational components that send order and billing information to revenue accounting. Since SAP S/4HANA

1809, the following operational components, or products, support integration with RAR:

- Sales and distribution
- SAP Billing and Revenue Innovation Management
- SAP CRM
- SAP S/4HANA Service
- External sender components

For the integration with sales and distribution, the integration functionality previously deployed through the software component SAP Sales and Distribution Integration with SAP Revenue Accounting and Reporting 1.0 (SAP SALES INTEGR SAP RAR 1.0) has also been added to the SAP S/4HANA 1809 stack and the SAP S/4HANA releases that followed.

Starting from SAP S/4HANA release 1909, customers can use an optimized version of the revenue accounting contract management. In SAP S/4HANA 2020, SAP added an optimized version of inbound processing. *Optimized contract management (OCM)* focuses on performance optimizations and additional features, such as day-based contract modifications. *Optimized inbound processing (OIP)* allows for real-time processing of operational documents. The Customizing for OCM and OIP can be found under **Revenue Accounting • Revenue Accounting Contracts and Revenue Accounting • Inbound Processing**.

Note

For new implementations, SAP recommends considering the optimized SAP S/4HANA capabilities of contract management and inbound processing, as future investments will mainly focus on these new capabilities.

Since offering the optimized versions, the existing contract management as known under SAP Revenue Accounting and Reporting 1.3 and SAP S/4HANA is now referred to as contract management (classic; or classic contract management [CCM]). The inbound processing will be referred to as inbound processing (classic). In Customizing, you can find the settings for CCM and inbound processing (classic) under **Revenue Accounting • Revenue Accounting Contracts (Classic) and Revenue Accounting • Inbound Processing (Classic)**.

Starting with SAP S/4HANA release 2021, customers can use the direct posting feature for posting management. *Direct posting* offers closer integration with the Universal Journal by posting directly into the Universal Journal without the revenue accounting subledger and smaller posting granularity due to posting by revenue accounting contract.

Starting with SAP S/4HANA release 2021, the parallel processing framework (PPF) used by program FARR_REV_TRANSFER (Transaction FARR_REV_TRANSFER) and program FARR_CONTRACT_LIABILITY (Transaction FARR_LIABILITY_CALC) has been adapted to the new SAP S/4HANA PPF (Transaction SHDB PFW). The old PPF based on the banking service is still used by other RAR programs such as RAR posting (Transaction FARR_REVENUE_POSTING), RAI transfer (Transaction FARR_RAI_TRANS), and RAI processing (Transaction FARR_RAI_PROC).

As the new Transaction SHDB parallelization framework (PFW) uses a different log mechanism than the old PPF, the job logs of Transaction FARR_REV_TRANSFER and Transaction FARR_LIABILITY_CALC are no longer available in Transaction BANK_PP_MONITOR. Using Transaction SHDB PFW, you can use report SHDB_PFW_SUPPORT to export the monitor information of a given instance. The application logs generated by those programs aren't changed. You can still view the application logs using Transactions SLG1 or SLGD.

In addition to these changes, there was a significant rework of the predelivered SAP Fiori apps and analytical reports. Users who used (or evaluated) are aware that in version 1.3 of SAP Revenue Accounting and Reporting, SAP delivered only a few reports called sample reports that had been developed using Web Dynpro technology. With SAP S/4HANA, the design of reports has changed, they are developed with SAP Fiori technology, and a substantial number of reports was added to the standard library.

When coming to SAP S/4HANA, the most important point is that RAR is an integral part of the SAP S/4HANA stack, which means that RAR is the main tool for revenue recognition; users can't rely on Transaction VF44 anymore. In addition, the main decision that needs to be made is whether to use OCM or CCM and inbound processing (depending on their version of SAP S/4HANA). Here, the decision is individual and despite the fact that optimized versions will be developed in the future, there are still limitations in using them (we'll discuss this in [Chapter 5](#)). An even stronger case for keeping the classic functionality is if the customer migrates from SAP ERP with SAP Revenue Accounting and Recognition to RAR in SAP S/4HANA. The impact on

enhancements and developments done in the classic environment can be considerable and even lead to a complete reimplementing of RAR.

2.4.2 RAR with SAP ERP

RAR is delivered on SAP ERP as an add-on component called SAP Revenue Accounting 130 (REVREC 130). The minimum system requirement for this add-on to be implemented is SAP ERP enhancement package (EHP) 5. Similar to SAP ERP, RAR is delivered as an add-on component on SAP S/4HANA on versions up to release 1709.

Additional versions of RAR were called support packs (SPs) until version 03, where the naming was changed to feature pack (FP) to version 07. Later, naming was changed back to support pack on which the current new versions are delivered (version 16 was the latest at time of writing).

The initial SP was mainly used when upgrading from versions 1.1 or 1.2. The first standalone version issued for the standalone use of RAR was version SP03 in August 2017. In this version, several program corrections were issued related to integration with results analysis and the now retired Hybris billing. The main change implemented was related to the contract modification process and enforcing the change of estimates instead of the prospective change for POBs, which were value relevant, and event type CI (customer invoice). This correction was done mainly for technical reasons while computing the remaining SSP. Another major correction with SP03 was related to how database updates are done. It had been noticed in previous

versions that data inconsistency can occur due to different triggers for database updates. This was solved by performing only one COMMIT once all processes in the ARL are completed. This is the main reason customers are strongly recommended to go to this SP.

Feature pack 04 (FP04) was issued in December 2017, and was the first to use the FP name. The FP is very similar to the SP, but it also contains nonmandatory and nondisruptive deliveries. The most important feature that came with this FP is the ability to do customized logic for calculating contract assets and contract liabilities (CA/CL). The default is that CA/CL is calculated on the contract level, but the option to calculate it on the POB level is also available. SAP Note 2560937 was issued with a description and information related to calculating CA/CL on the POB level. Besides this option, a BAdI to perform custom calculation of CA/CL was delivered and can be implemented by the user.

FP05 was delivered March 2018 based on customer requests. The main focus was on migration and transition topics, plus performance improvement when needing to process millions of contracts with many POBs. With this FP, for the first time, there were recommendations on the number of POBs per contract (see SAP Note 2551667) and items in BRFplus. In addition, the following features were introduced:

- Reprocessing contracts and account determination
- Calculate and post exchange rate differences for the fixed rate method
- Detailed inflight and data validation checks

- Improved flexibility for the contract modifications process
- Navigation from a financial accounting document to a RAR contract

FP06 came in July 2018, and was published with main corrections in the integration area with sales and distribution. Processes that were covered include the intercompany sales process, which came as an option in Customizing; the drop shipment process; and reconciling operational data with RAR.

The following additional functionalities were enabled as well:

- Enabling change documents for account assignments
- Allowing POB cancellation of a compound structure separately
- POB cancellation and conflict management
- BAdI: Log POB Data
- Additional contract shift scenarios
- Overfulfillment during returns processing
- Applying contract change to earliest open period
- Improving the remaining SSP calculation flexibility
- Improving the remaining amount calculation formula for allocation to difference condition types
- Performing SSP range validation during contract modification

FP07 was delivered in November 2018, and it represents a major step forward in terms of added functionalities. The most important features are new fulfillment types: call-off

order, proof of delivery, and goods in transit. Beside these, flexibility is added to existing processes in contract management such as allowing a negative allocated price, allowing one POB in compound to be nondistinct, and so on.

FP08 and FP09 were mainly issued to cover program bugs and improve overall stability of the solution. No major new functionalities were delivered.

SP10 was delivered in September 2019, and besides overall program corrections, it contained a few new features that might become useful to customers:

- **BRFplus trace tool**

This development is used to visualize the full decision path of the BRFplus function calls during creation of a POB from source RAI data. The BRFplus trace can be used by customers to explain their own customization as well as a support engineer during problem analysis. Note that this trace is only available for BRFplus usage in inbound processing.

- **RAI monitor enhanced view**

A new selection criterion has been added to Transaction FARR_RAI_MON that allows you to display partially processed RAIs or processed and partially processed RAIs together.

SP11 and SP12 again mainly referred to program corrections and improvements.

Next, SP13 was delivered in November 2020, with new features and program corrections:

- **New functionality to manage migration**

This new functionality provides a guided procedure for migrating legacy data to RAR with a special focus on migrating data from sales and distribution revenue recognition. While leveraging the existing migration programs, it streamlines and safeguards the execution of this critical process.

- **Performance improvements for processing large contracts**

There are certain limitations for running SAP Revenue Accounting and Reporting 1.3 with contracts of very large sizes. These limitations are described in SAP Notes 2616387 and 2551667. This can often lead to timeouts when performing basic tasks, such as opening the contracts in the SAP GUI or combining them. Within SP13, certain technical changes have been introduced to improve performance to allow users to process larger contracts.

- **Allow condition types with negative values**

RAR provides the functionality to check whether a condition type of a particular POB has the same +/- sign. If the sign is different, an error message is raised. This functionality was introduced to prevent errors during a posting run in case a customer was using cost-based or combined profitability analysis. For certain custom processes (e.g., managing rounding differences), this could cause an issue. SAP now delivers functionality to allow configuration to specify which condition types can have both +/- signs for the same POB.

SP14, 15, and 16, which were delivered up to October 2022, brought upgrades of existing programs and minor features.

What we can see is that RAR was evolving in terms of new features mainly up to deadline for IFRS 15 adoption. After that, new features were coming to fill gaps that hadn't been addressed previously. But mainly, program improvements and dealing with performance issues were constantly improved, which continues today.

For new users, it's very much recommended to go straight to the latest version. However, existing users need to carefully evaluate which version to upgrade to. Here, going to the latest version needs to be carefully considered, and a decision should be made based on the number and type of custom developments the customer might have. Programs are constantly evolving, and possible upgrades might have an impact on existing developments. So before deciding which version to upgrade, users need to verify changes to programs beside checking features that are coming with the new version.

2.5 Summary

Despite the fact that RAR is a relatively new tool, it offers a variety of options for integration. This gives a certain level of flexibility to users so they can select the most optimal path while designing their system landscape. On the other hand, RAR is a very robust solution if used with natively integrated modules such as sales and distribution. It provides an opportunity for users to focus on solving issues related to business processes.

However, the downside of being a new solution is that RAR is still evolving. That is why it's very important to carefully evaluate which version will be implemented in your landscape. This becomes even more crucial when deciding whether you'll opt for optimized or classic inbound processing and contract management.

In RAR, maybe more than in other SAP modules, it's essential to be familiar with data structures and understand how the solution runs under the hood. That's why we also focused on presenting table structures and data models for RAR in this chapter. All this should improve your awareness before landscape and architectural decisions are made.

With this foundation in mind, we'll move on to the step-by-step configuration for inbound processing in the next chapter.

3 Configuring Inbound Processing (Classic and Optimized)

Configuring inbound processing is the starting point in any revenue accounting and reporting (RAR) implementation. We'll explain how RAR and components with source data communicate with each other via two options: classic and optimized inbound processing (OIP).

Documents in RAR are always created as the result of documents created in either other SAP components (sales and distribution, customer relationship management [CRM], contract accounting) or as a result of loading data from external systems. Before data reaches the revenue accounting engine, RAR has a staging layer that needs to be passed called the Adapter Reuse Layer (ARL).

We discussed the architecture of the ARL in [Chapter 2](#). To review, when data is created in an operational application, it's sent to ARL for processing. Data is created as a temporary item, a revenue accounting item (RAI), which contains all needed information for contract and performance obligation (POB) creation in revenue accounting.

When creating RAR data, it needs to pass through a few checks:

- Revenue accounting configuration defines how RAIs will be processed.
- BRFplus rules are checked against information sent so the system knows which POBs need to be created, which fulfillment types to use, and so on. These rules are covered in detail in [Chapter 5](#) when we discuss contract management.

Once data passes ARL, we know what contract and POBs should be created. In addition, depending on the type of RAI sent to the results analysis engine, different modules will be involved in processing the data.

Not all data creates the same type of RAI. The system will create different RAIs depending on whether you're sending order data (which is needed to create or update contract/POBs) or whether you're sending data related to fulfillments (e.g., invoices or delivery). At the top, RAIs are created with different levels of detail (e.g., cost conditions) or a specific business process is used (e.g., SDPI [SD planned items] or planned RAIs for billing plans).

Once data comes to the ARL for processing and the RAI gets created, it will have one of the following statuses, as discussed in [Chapter 2, Section 2.3.1](#):

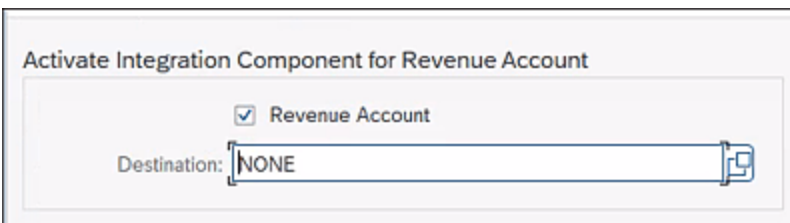
- **Raw** status is used when the user needs to preprocess data before the RAR contract can be created. RAIs will be created in this status if basic data checks are unsuccessful.

- **Processable** status means that RAIs are ready to be transformed into POBs and contracts.

Now, let's dive into the detailed configuration instructions for setting up inbound processing in SAP S/4HANA. We'll start with setting up the RAR integration with sales and distribution, and then we'll walk through setting up RAI classes for both optimized and classic inbound processing. To close, we'll explain how to extend RAI classes by defining and populating extensions.

3.1 Setting Up Revenue Accounting Integration

The first step in activating RAR is enabling integration between RAR and sender components. When integrating with sales and distribution, follow IMG menu path **Sales and Distribution • Revenue Accounting and Reporting • Integrate with Revenue Accounting and Reporting**. You'll arrive at the screen shown in [Figure 3.1](#), which is used to enable integration by selecting the **Revenue Account** checkbox. In the **Destination** field, you can enter the logical system if RAR is operating on a different instance than the rest of your ERP system. If they are on same instance, you can enter "NONE".



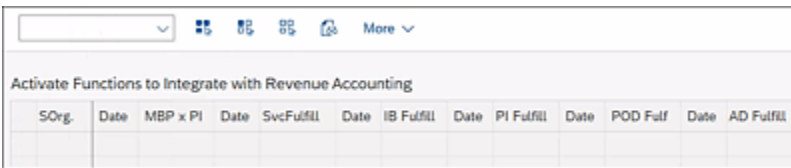
Activate Integration Component for Revenue Account

Revenue Account

Destination: NONE

Figure 3.1 Enabling Integration between Sales and Distribution and RAR

Click the **Execute** button, and then the next step is to enter whether additional functionalities are to be used by RAR in integration with sales and distribution in the **Activate Functions to Integrate with Revenue Accounting**, as shown in [Figure 3.2](#).



SOrg.	Date	MBP x PI	Date	SvcFulfill	Date	IB Fulfill	Date	PI Fulfill	Date	POD Fulf	Date	AD Fulfill

Figure 3.2 Integration with Additional Functions

Click the **Change** button to enable editing the fields. Here, you can enable the usage of additional fulfillment events, as follows:

- **MBP x PI**

If this milestone billing plan integration functionality is activated, you'll get SDPI (planned items) generated once you save the sales document that contains the billing plan.

- **SvcFulfill**

Settings in the **SvcFulfill** column are related to the system creating fulfillment entries for orders that contain nonstockable materials (type DIEN). This section usage needs to be thought through due to the decision regarding how services need to be treated in RAR: if they are consumed equally to billing, they need to be maintained as time-based POBs, so this setting isn't necessary. However, if services are managed in service contracts and are used by service orders (or corresponding documents coming from the customer service module), this setting might be needed.

- **IB Fulfill**

For intercompany processes, the intercompany invoice is sent to be used as a fulfillment trigger.

- **PI Fulfill**

The purchase invoice can be used as a trigger for revenue recognition when you're working with drop shipment scenarios (also called subcontracting). In this case, the company making the sale doesn't hold the stock of goods, so the goods are delivered by a third-party vendor without coming to the seller's warehouse first. The vendor invoice can be treated as a fulfillment trigger.

- **POD Fulf**

Proof of delivery is a document which shows that goods are now owned by the buyer. It can be used when incoterms are used to show that the buyer becomes the owner of the goods only when those goods arrive at their location.

- **AD Fulfill**

This fulfillment type relies on the **Customer Acceptance Date** field on the sales order, which allows us to recognize revenue only once the customer accepts delivery.

The settings are made per sales organization, which increases system flexibility. In addition, you can make some default entries by leaving the setting blank. You also can control when items will be created by providing specific dates in the **Date** fields (meaning that before that date, the functionality wouldn't be executed).

The next step is actually setting up how integration between RAR and sales and distribution will be done. You're deciding which item to integrate with RAR and how. This setup needs to be accessed through Transaction SPRO by following menu path **Sales and Distribution • Revenue Accounting and Reporting • Maintain Revenue Accounting Item Settings** to arrive at the screen shown in [Figure 3.3](#). Click on the **Change** button to enter settings.

SOrg.	SaTy	ItCa	Type	Package ID	
<input type="checkbox"/>	AU40	ZCRM	ZG2N	Credit/Debit Memos with reference to...	
<input type="checkbox"/>	AU40	ZDRM	ZL2N	Credit/Debit Memos with reference t...	
<input type="checkbox"/>	AU40	ZLV	L2N	Credit/Debit Memos with reference t...	
<input type="checkbox"/>	AU40	ZMRO	ZLMP	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZLMT	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZLPW	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZMDS	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZMMF	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZMMS	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZMSH	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZPMF	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZPMT	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	AU40	ZMRO	ZRRL	Relevant for Revenue Accounting	▼
<input type="checkbox"/>	ID40	ZCRM	ZG2N	Credit/Debit Memos with reference t...	
<input type="checkbox"/>	ID40	ZDRM	ZL2N	Credit/Debit Memos with reference t...	

Figure 3.3 Integration between Sales Documents and RAR

On the sales organization level, you need to define which sales document types and item categories will be integrated with RAR and which won't. The first three columns are reserved for sales organization/document type/item category type, which need to be linked with RAR. The values in the **Type** column can vary as they define what kind of integration is done:

- **Relevant for Revenue Accounting**
This means that if the user enters a combination of sales

document and item category, that combination will be sent to ARL for processing and creation of contracts/POBs.

- **Not Relevant for Revenue Accounting**

This option is the opposite: the combination won't be sent to RAR.

- **Credit/Debit Memos with reference to predecessor**

This is used when the user enters a document that should just increase or decrease the transactional price. The credit memo is a document with which you want to subsequently change the agreed price with the customer (due to some extraordinary circumstances or additional negotiations). Note that if the mentioned combination is used, it must have a predecessor—either an order or an invoice—because a credit/debit memo without a reference isn't supported.

Other options will be used in specific processes that are covered in other chapters (e.g., call-off order functionality).

Should All Sales Documents Be Maintained as RAR Relevant?

As in other areas, the answer to this question depends on the exact business case and how customer plans to utilize RAR. One clear benefit can be that RAR becomes the single point for complete revenue reporting for an organization. All information regarding revenue would be in FARR_D* tables, so standard apps for reporting (e.g., disaggregation of revenue or contract assets/contract liabilities (CA/CL) movements) can be used. In addition,

any kind of additional custom reports would be easier to deliver because data would be found in a single place.

However, additional complexity accompanies this approach. For example, if we want to integrate quantity contracts with RAR, what would that mean? The nature of such a contract is that the customer is unsure how many goods will be delivered, so we could argue that step 1 from IFRS 15 isn't being fulfilled because there is no enforceability. So, each time a customer decides to get some goods, a sales order would have to be created based on which delivery and invoicing occur. In terms of RAR, this would make use of the call-off order functionality (when sales orders are created on the basis of a sales contract). But this would mean additional overhead on business users who would need to monitor RARs creation and processing. In addition, standard RAR demands would still need to be followed.

To conclude, there are clear benefits to using RAR for all sales documents, but it's not mandated. Customers might opt to use RAR only for documents with clear contractual obligation, and that approach is also valid. The decision mainly depends on the internal capacity to handle the additional workload that would come with RAR.

Sometimes, you need to change items that are usually relevant for RAR to not relevant. For this to work, the standard solution is to create separate item categories, but because this typically isn't possible (usually to avoid giving extra tasks to the sales team), you can try to do so with enhancement by using business add-in (BAdI)

FARRIC_BADI_ORDER.

To do so, execute Transaction SE18. For the BAdI's **Interface** field, enter "IF_FARRIC_ORDER", and for the **Implementing Class** field, enter "ZRCLFI_FARRIC_BADI_ORDER", as shown in [Figure 3.4](#).

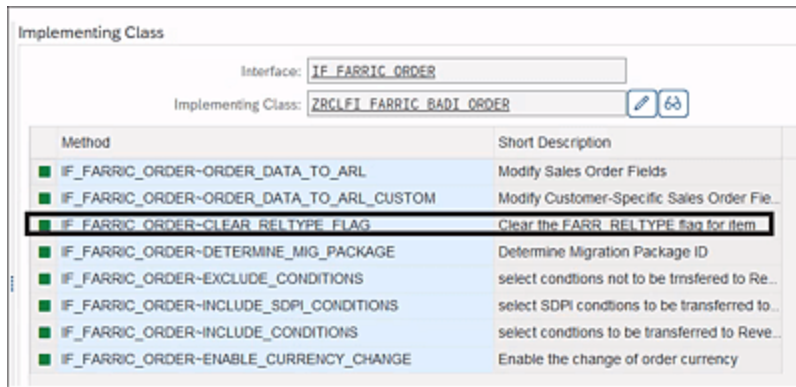


Figure 3.4 Clearing of Relevancy Flag

A method is available where you can enter logic based on that criterion's relevancy flag needing to be removed (**IF_FARRIC_ORDER~CLEAR_RELTYPE_FLAG**).

In addition, if you're using this method, be aware that relevancy can be moved only in one direction: from relevant to nonrelevant. In other words, you can't make an item relevant if setup was done in the opposite direction. In this BAdI, you also have multiple methods that might help if data manipulation is required between the order and RAR contract, before the ARL is reached.

Reconciliation between Sales and Distribution and RAR

If any of these methods are used, special attention needs to be paid to reconciliation topics. Changing data based on some criteria that aren't entered in the standard setup

will cause differences between sales and distribution and RAR. In some cases, this can be overlooked (when the modification is very strict and controllable), but, in others, it can represent an issue.

The second point is related to the contract lifecycle: unlike other documents in finance, RAR contracts are often changed, so if you decide to use this BAdI, you also need to make sure that process will work if there are contract modifications.

An example of the effect of reconciliation can be found when you want to set a threshold for items not being relevant for RAR (e.g., only contracts where the contractual value is greater than \$100,000 are relevant). In this case, it wouldn't be optimal to create a separate item category type, but one option would be to use the mentioned BAdI. You must be careful in those cases that, for example, the sales order can't be combined with another in a single RAR contract.

3.2 Inbound Processing

As mentioned in the previous section and [Chapter 2, Section 2.3](#), the RAI class determines technical aspects of the RAI item that need to be created:

- Database tables in which the system will store RAIs depending on their status and type
- Functional modules that will process RAIs
- Function modules that will save RAIs
- Custom fields that users might add to an RAI class

RAI classes are stored in three different types: 01 for order items, 02 for fulfillment items, and 03 for invoices. For more details, see [Chapter 2, Section 2.3.2](#).

Behind this setup, there is a list of interfaces that are used when integrated with proper data sources and in specified cases (see [Table 3.1](#)).

Interface Component	Description	Data Element
BASIC_C0	Basic fields for conditions	FARR_IC_BASIC_C0
BASIC_C001	Basic fields for order items (conditions)	FARR_IC_BASIC_C001
BASIC_MI	Basic fields for main items	FARR_IC_BASIC_MI

Interface Component	Description	Data Element
BASIC_MI01	Basic fields for order items	FARR_IC_BASIC_MI01
BASIC_MI02	Basic fields for fulfillment items	FARR_IC_BASIC_MI02
BASIC_MI03	Basic fields for invoice items	FARR_IC_BASIC_MI03
CA_BASIC_MI	CA basic fields for main items	FARR_IC_CA_BASIC_MI
CA_MI01	CA fields for main items	FARR_IC_CA_MI
COPA_MI01	Structure COPACRIT	FARR_IC_COPA_MI
CRM_MI01	CRM fields for order items	FARR_IC_CRM_MI
SD_MI01	Sales and distribution fields for order items	FARR_IC_SD01_MI

Table 3.1 ARL Interface Components

Which components will be active depends on what kind of integration is needed: CA components are used if the integration point is with contract accounting, CRM if you're integrating with customer relationship management, and SD if the integration point is sales and distribution. The COPA structure will be used if you're using controlling and it's inheriting all characteristics defined in profitability/margin analysis.

With the basics covered, we'll explain how to set up RAI classes in both classic and optimized inbound processing (OIP) in the following sections.

3.2.1 RAI Classes in Classic Inbound Processing

Once you determine the integration type between operational applications and RAR, the first step is to set up ARL for inbound processing. This is done through configuration of RAI classes, which we'll explain in the following sections.

Interface Components

For the classic environment, go to Transaction FARR_IMG, and follow menu path **Revenue Accounting • Inbound Processing • Revenue Accounting Items**. You'll arrive at the screen shown in [Figure 3.5](#), which shows the available interface components.

SAP provides preconfigured interface components that you can use immediately. These include basic components, which are mandatory for each record type, as well as other optional components that you can add to a class. Note that you can't create your own customer-specific interface components.

Dialog Structure		Interface Components for RAI Classes		
	Interface Compo...	Description of Interface Component	Data Element for Documentation	
<input type="checkbox"/>	BASIC_CO	Basic Fields for Conditions	FARR_IC_BASIC_CO	
<input type="checkbox"/>	BASIC_CO01	Basic Fields for Order Items (Conditions)	FARR_IC_BASIC_CO01	
<input type="checkbox"/>	BASIC_MI	Basic Fields for Main Items	FARR_IC_BASIC_MI	
<input type="checkbox"/>	BASIC_MI01	Basic Fields for Order Items (Main Items)	FARR_IC_BASIC_MI01	
<input type="checkbox"/>	BASIC_MI02	Basic Fields for Fulfillment Items (Main Items)	FARR_IC_BASIC_MI02	
<input type="checkbox"/>	BASIC_MI03	Basic Fields for Invoice Items (Main Items)	FARR_IC_BASIC_MI03	
<input type="checkbox"/>	CA_BASIC_MI	CA Basic Fields for Main Items	FARR_IC_CA_BASIC_MI	
<input type="checkbox"/>	CA_MI01	CA Fields for Order Items	FARR_IC_CA_MI01	
<input type="checkbox"/>	COPA_MI01	CO-PA Fields for Order Items (COPACRIT)	FARR_IC_COPA_MI01	
<input type="checkbox"/>	CRM_MI01	CRM Fields for Order Items	FARR_IC_CRM_MI01	
<input type="checkbox"/>	SD_MI01	SD Fields for Order Items	FARR_IC_SD_MI01	

Figure 3.5 Interface Components

After making your selection in the **Interface Compo...** column (**SD_MI01**, in our example), the following settings in the **Dialog Structure** define an interface component:

- **Assigned Structures**

The assignment of structures (see [Figure 3.6](#)) depends on the record type and the status (for details, see [Chapter 2, Section 2.3](#)). For entries that don't belong to the **Main Item** record type, you can specify that these entries are only conditionally active by selecting the **Condit. Active** checkbox.

Component: SD_MI01		Description: SD Fields for Order Items		
Assigned Structures				
Rec. Type	Status	Structure	Condit. Active	
<input type="checkbox"/> Main Item	▼ All Statuses	▼ FARR_S_ICMI01_SD	<input type="checkbox"/>	
<input type="checkbox"/> Main Item	▼ Raw	▼ FARR_S_ICMI01_S00	<input type="checkbox"/>	
<input type="checkbox"/> Main Item	▼ Raw - Exempted	▼ FARR_S_ICMI01_S00	<input type="checkbox"/>	

Figure 3.6 Assigned Structures

- **Prerequisite Components**

This setting indicates which interface components are prerequisites. Prerequisite components get activated automatically when the current component is used in the RAI class.

- **Program Enhancements**

Program enhancements can be defined as shown in

[Figure 3.7](#). The following program enhancements are available:

- **Enrich Raw Items:** In this case, additional information needs to be added to RAI items created in **Raw** status.
- **Enrich Processable Items:** This is similar to the previous item, but items are created in **Processable** status.
- **Final Check Before Saving Processable Items:** In this case, you need to perform an additional check at the last moment before data is saved in the database.



Figure 3.7 Enhancement of Data Method

These enhancements are realized using methods of class CL_FARR_RAI_IFCOMP.

One example of using this BAdI and method ENRICH is when you need to determine the customer for an IFRS 15 contract. We'll discuss this problem in [Chapter 5](#): one of the crucial steps in creating a contract is to define the customer, which can't always be picked up from the specific partner function in the sales document (default is PAYER).

So, in this case, you have a master contract to which all follow-on documents belong if they are members of a bundle. In this document, you have a customer defined with the proper partner function. You want to make sure that irrespective of the customer in the sales contracts/sales

orders, you're always picking the customer defined in the master contract.

To achieve this, you need to implement the BAdI with specific code for fetching the proper customer and using it in RAR documents, as shown in [Listing 3.1](#). Implementation will be done by running Transaction SE18 and selecting BAdI FARR_BADI_RAI2 and method ENRICH.

```
SELECT vbeln,
       kunnr
FROM vbak
INTO TABLE @DATA(lt_vbak)
FOR ALL ENTRIES IN @ct_rai2_mi
WHERE vbeln = @ct_rai2_mi-wwmct .
IF sy-subrc IS INITIAL.
  SORT lt_vbak BY kunnr.
  lv_flag = abap_true.
  EXPORT lv_flag FROM lv_flag TO MEMORY ID 'FLAG'.
ENDIF.
READ TABLE ct_rai2_mi ASSIGNING FIELD-SYMBOL(<ls_mi>) INDEX 1.
IF sy-subrc IS INITIAL.
  lv_wwmct = <ls_mi>-wwmct.
  EXPORT lv_wwmct FROM lv_wwmct TO MEMORY ID 'WWMCT'.
ENDIF.

"Taking data to temporary internal table
DATA(lt_rai2_tmp) = ct_rai2_mi.

CLEAR lv_vbeln.
LOOP AT lt_rai2_tmp ASSIGNING FIELD-SYMBOL(<ls_rai2_mi>).
  IF <ls_rai2_mi>-raic = lc_raic.
    <ls_rai2_mi>-due_date = <ls_rai2_mi>-posting_date.
  ENDIF.
  READ TABLE lt_vbak ASSIGNING FIELD-SYMBOL(<ls_vbak>) WITH KEY
                                vbeln = <ls_rai2_mi>-wwmct.
  IF sy-subrc IS INITIAL.
    <ls_rai2_mi>-kunnr = <ls_vbak>-kunnr.
  ENDIF.
ENDLOOP.
ct_rai2_mi = CORRESPONDING #( lt_rai2_tmp MAPPING kunnr = kunnr
                              due_date = due_date ) .

UNASSIGN <ls_rai2_mi>.
CLEAR: lv_vbeln.
FREE : lt_rai2_tmp.
ENDMETHOD.
ENDIF.
```

Listing 3.1 Example of Changing Partner Type Determination for Customer in RAR

RAI Class Configuration and Activation

The next step in inbound processing configuration is to set up RAI classes. SAP doesn't supply any RAI classes, but there are predefined class names for certain components.

You need to define the classes to satisfy your technical requirements. The settings you make here influence the appearance of the database tables, as well as the interfaces for the data transfer.

To set up RAI classes, you need to execute three steps ([Figure 3.8](#)):

1. Set up the interfaces.
2. Add customer fields.
3. Add database indexes.

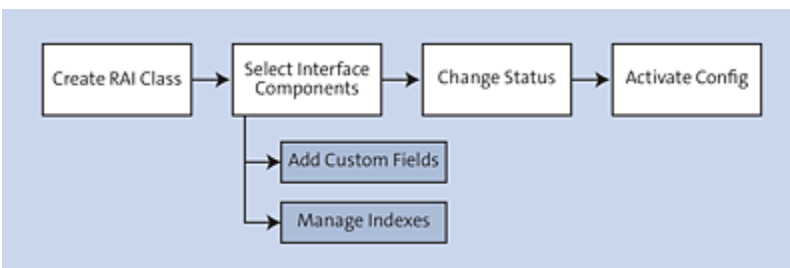


Figure 3.8 Steps for Creation of RAI Classes

You can maintain RAI classes in Transaction FARR_RAI_CONF. [Figure 3.9](#) shows the setup screen, where you can access the **Interface**, **Customer Fields**, and **Indexes** buttons. More information about these options was provided in [Chapter 2, Section 2.3](#).

Active	RecAcct...	Class Type	Name	Configuration Status	Generatn Trgt System
<input type="checkbox"/>	SD01	Order Item	PG RAI Class - Order Item	Transportable	<input checked="" type="checkbox"/>
<input type="checkbox"/>	SD02	Fulfillment Item	PG RAI Class - Fulfillment Item	Transportable	<input checked="" type="checkbox"/>
<input type="checkbox"/>	SD03	Invoice Item	PG RAI Class - Invoice Item	Transportable	<input checked="" type="checkbox"/>

Figure 3.9 Setup of RAI Classes

Once you define an RAI class, you need to select the **Class Type**. As we’ve discussed, there are three types available: **Order Item**, **Fulfillment Item**, or **Invoice Item**.

The next step is to define whether the class is transportable or not under the **Configuration Status** column. [Table 3.2](#) shows the activities that are available based on the configuration status.

Status	Changes?	Transport
In Processing	No constraints	No
Transportable	No constraints	Yes
Released as Productive	<ul style="list-style-type: none"> • Additional interface components • Additional customer fields • Indexes 	Yes

Table 3.2 Transport Statuses

You need to consider these before the status is set or changed because not all changes are possible once a certain status is reached.

After creation, the class has an **In Processing** configuration status. You can change a class with this status as you like. If you activate the class, the system doesn't include it in a transport order. The system is only able to include the class in a transport request if you've set the status to **Transportable**. You can also make as many changes as you like in this status.

When you set the configuration status to **Released as Productive**, you can only make the following compatible changes:

- Select additional interface components that aren't yet active.
- Select additional customer fields that aren't yet active.

By selecting the **Generatn Trgt System** option, the RAI class will be automatically generated in the target system. This specifies that when you transport the RAI class, it's automatically generated in the target system by the after-import method `FARR_RAIC_GEN_AFTER_IMP`. To set the indicator, add the class to a transport request using the **Activate Configuration** function (a small matchstick icon, not shown). In the dialog that appears, confirm that you want the class to be generated automatically in the target system.

You can only transport classes that have the configuration status **Transportable** or **Released as Productive**. When you transport the configuration of a class, the Customizing includes `CI_FARR_S_*` and `CI_FARR_S_*_ALL` must already be present in the target system containing the required fields.

When you activate the configuration, the system checks the completeness of the work structures for RAIs that are used in the processing programs. If necessary, it then automatically adds the fields of the class that aren't yet contained in the work structures. The fields are added in the related Customizing include CI_FARR_S*_ALL.

Once the class is activated in the target system, you can either delete all items in Transaction FARR_RAI_MON during generation or leave it as is.

While setting up interface components, you also can activate specific interfaces. By selecting an RAI class (**SD01**, in our example) and the **Interface** button, you'll arrive at the screen shown in [Figure 3.10](#). Here, you can add or remove any additional interface components by selecting or deselecting the **Active** checkboxes based on your requirements.

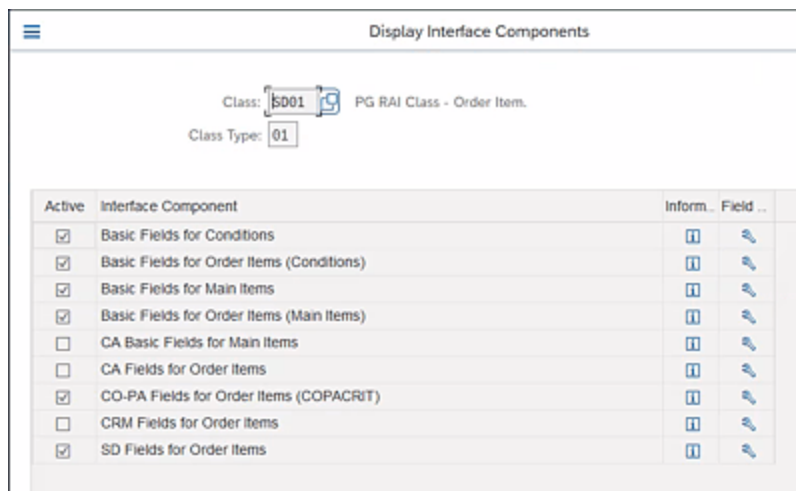


Figure 3.10 Interface Component Selection

Once this is done, SAP will automatically generate interfaces that will enable linking to specific applications and data transfer.

Before RAI classes can be used, they must be activated. The activation option is available only if the user has proper authorization in the system. It's good practice to verify whether a class has any issues or inconsistencies before it can be used. You can do that by performing an activation check.

With the **Perform Consistency Check for Configuration** icon at the top of the screen (see [Figure 3.11](#)), you can check a selected RAI class. When you call this transaction, the system only checks the generation status based on the entries made in maintenance for the class. The check focuses on changes and additions to the following items:

- Interface components
- Customer fields
- Indexes

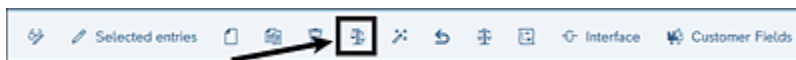


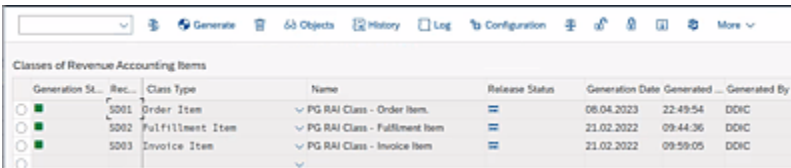
Figure 3.11 Perform Consistency Check

If you explicitly check a class, the system performs detailed comparisons, which focus on changes that affect the generated function modules.

When you choose **Activate Configuration**, the system checks the completeness of the work structures for RAIs used in the processing programs. The system can then automatically add fields of the classes that aren't already contained in the work structures. These fields are added in the related customer includes of the work structures.

Interface Generation

Before classes can be used, you need to generate interfaces. Execute Transaction FARR_IMG, and navigate to **Inbound Processing • Revenue Accounting Items • Generate Interfaces for Item Classes**. For interfaces to be generated, the screen must look like [Figure 3.12](#).



Generation St.	Rec.	Class Type	Name	Release Status	Generation Date	Generated...	Generated By
5001		Order Item	PG RAI Class - Order Item		08.04.2023	22:49:54	DDIC
5002		Fulfillment Item	PG RAI Class - Fulfillment Item		21.02.2022	09:44:36	DDIC
5003		Invoice Item	PG RAI Class - Invoice Item		21.02.2022	09:59:05	DDIC

Figure 3.12 Generation of Interfaces Option

The following options are available at the top of the interface generation screen:

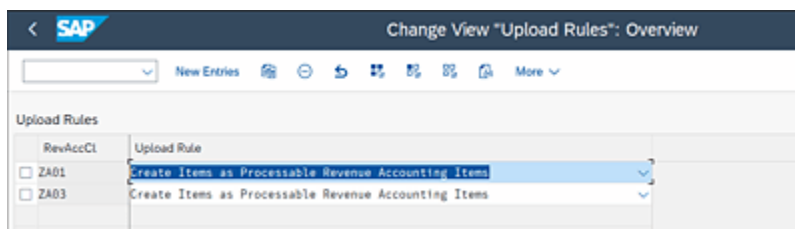
- **Generation check**
The system compares configuration with interface components, indexes, and customer fields. Once the check is completed, a list of inconsistencies can be displayed.
- **Delete generated object**
If an object isn't set as productive, it can be deleted.
- **Comparison of configuration with active version**
The system will try to find any differences between the current configuration in the system and the last active version.
- **Release and lock for use**
Once setup is completed, it can be released and made ready for productive usage.

After you generate interfaces, all related methods, classes, and needed data objects are generated. The system will write all changes to the log to maintain the history. Sometimes, a class becomes inactive, so it's useful to check its status after this activity by running the same process over again (status must be green after activation).

However, if you selected that automatic generation should be performed after transport (via the **Generatn Trgt System** option discussed previously), this activity isn't needed.

Once generation occurs, the system will generate a separate RFC function module in the background for each RAI class. This will later enable data transfer between the ARL and the contract management engine.

To define upload rules, follow menu path **Revenue Accounting • Inbound Processing • Revenue Accounting Items • Assign Upload Rules**. As shown in [Figure 3.13](#), you're defining how items will be created once they are pushed from the source application to the ARL. You have two options: create the RAI as a raw item or create the RAI as a processable item. If the item is created in **Raw** status, you'll need to transport it to **Processable** status before it can be passed to the revenue accounting engine. If the item is created as **Processable**, as in our example, it can be processed straight after creation.



RevAccCl	Upload Rule
<input type="checkbox"/> ZA01	Create Items as Processable Revenue Accounting Items
<input type="checkbox"/> ZA03	Create Items as Processable Revenue Accounting Items

Figure 3.13 Upload Rules

The last step while creating RAI classes is to verify what fields will be available for modification once the RAI is created. This step is optional but can be useful when data requires modification before being processed by the ARL. However, in other cases, it might not be advisable to modify data because that would cause inconsistency between the source application and RAR contract that is created as a result.

3.2.2 RAI Classes in Optimized Inbound Processing

OIP offers a fixed database model (based on basic plus sender component-specific information). This means that structures, application programming interfaces (APIs), and runtime working structures previously generated under the classic inbound processing are now available as predefined fixed structures, APIs, and runtime working structures, respectively. The predefined fixed structure, APIs, and runtime working structures are provided for the following sender components:

- Sales and distribution (SAP)
- Contract accounting (SAP)
- SAP CRM or SAP S/4HANA Service
- External third-party sender components

In [Figure 3.14](#), you can see a high-level picture of OIP. Here, it's clear that RAIs are still coming to the processing engine in a similar way as in classic inbound processing. The

system also works in a very similar way in respect to integrations: source data can originate in sales and distribution, SAP Billing and Revenue Innovation Management, or external applications. Once data is processed, results will be saved directly in the Universal Journal.

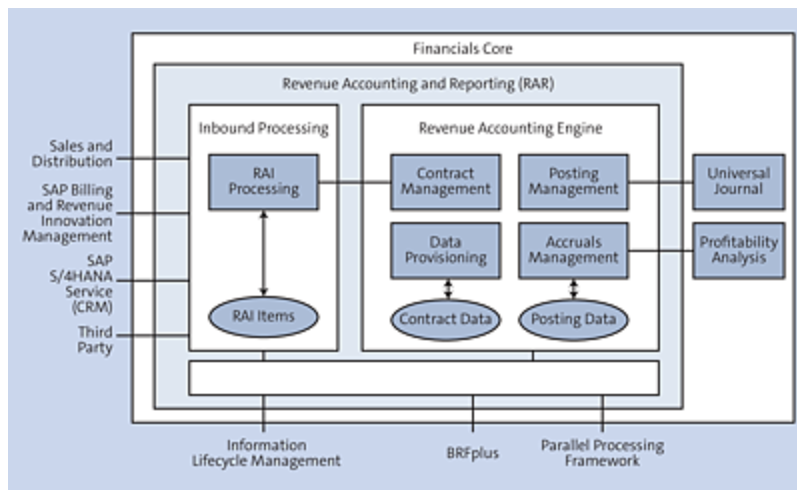


Figure 3.14 Architecture of Optimized Inbound Processing

To avoid batch job processing that is present in classic inbound processing, OIP processes RAIs in real time (no mass/batch processing required), based on the inbound processing APIs. This means the background jobs to transfer and process RAIs are no longer required. If an error occurs during processing, postponed RAIs are created and stored in a specified table.

Note

For integration with SAP Billing and Revenue Innovation Management, you're still required to execute job FP_RAI_TRANSF in SAP Billing and Revenue Innovation Management. This program creates the RAI structures that

are then immediately processed from revenue accounting using OIP. In contrast to using the classic inbound processing, you save the processing of RAIs in revenue accounting through a batch job.

OIP is based on static database tables. Successfully processed RAIs are saved in corresponding database tables, as shown in [Table 3.3](#). They are transformed into revenue contracts with POBs and are used to update the corresponding contract with fulfillments and with cost and invoice data.

Table	Description
Processed RAIs	
FARR_D_ORD_MI	Processed Order RAIs - Main Items
FARR_D_ORD_CO	Processed Order RAIs - Condition Items
FARR_D_FULFILLMT	Processed Fulfillment RAIs
FARR_D_INV_MI	Processed Invoice RAIs - Main Items
FARR_D_INV_CO	Processed Invoice RAIs - Condition Items
FARR_D_COST_MI	Processed Cost RAIs - Main Items
FARR_D_COST_CO	Processed Cost RAIs - Condition Items
Partially Processed RAIs	
FARR_D_ITEM_PROC	Partially Processed Data
Postponed RAIs	
FARR_D_INB_MI	Postponed RAIs - Main Items Table

Table	Description
FARR_D_INB_CO	Postponed RAIs - Condition Items Table

Table 3.3 Database Tables

In the following sections, we'll explain how to activate OIP and then how RAIs are processed and postponed.

Activation of Optimized Inbound Processing

By default, in on-premise SAP S/4HANA, OIP isn't activated (note that it's activated by default in SAP S/4HANA Cloud, public edition). To activate it, you must create an implementation for BAdI `IF_FARR_BADI_DET_N_IP_VERSION` (Determination of Inbound Processing Version) in Transaction SE18, as shown in [Figure 3.15](#).

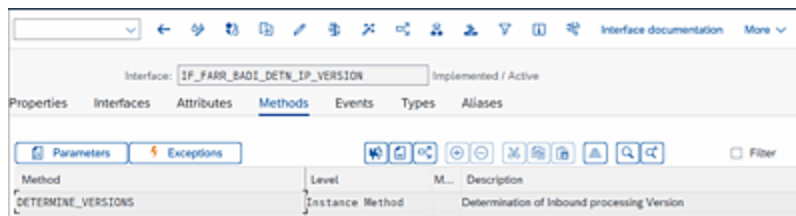


Figure 3.15 BAdI `IF_FARR_BADI_DET_N_IP_VERSION`

Here, you can see interface **`IF_FARR_BADI_DET_N_IP_VERSION`** and method **`DETERMINE_VERSIONS`**, which need to be implemented. This method has two parameters: **`ITS_ORDER_ITEM`** and **`CTS_SRCDOC_VERSION`**, as shown in [Figure 3.16](#).

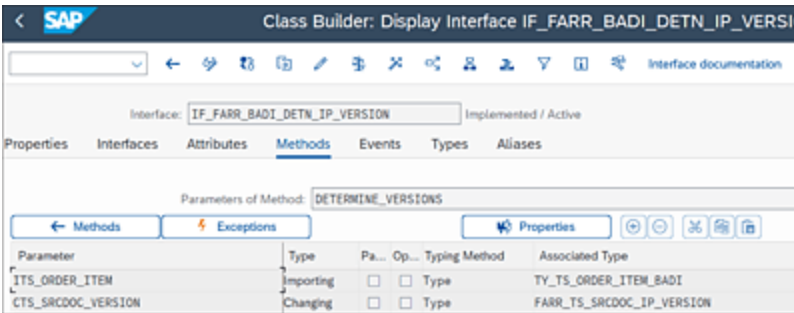
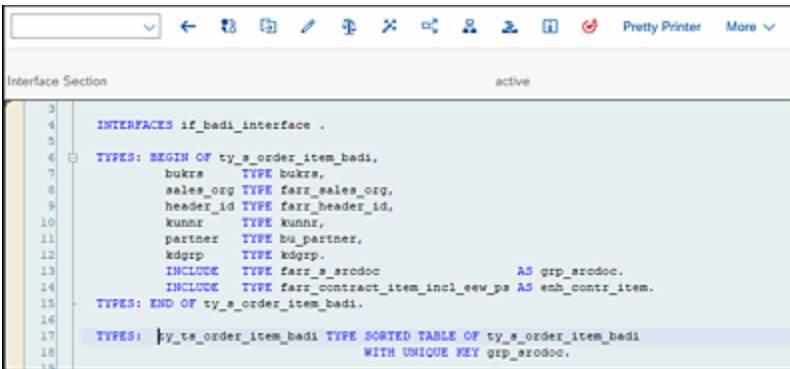


Figure 3.16 Method DETERMINE_VERSIONS

Once you double-click on the method, you'll see the available parameters, as shown in [Figure 3.17](#).

Following are the parameters that can be used to determine the usage of inbound processing:

- Sender component of source item
- Logical system of the source item
- Source document item type
- Source item ID
- Company code
- Sales organization
- Customer
- Business partner
- Customer group
- Customer fields (provided by the extension include of the contract items)



```
3
4 INTERFACES if_badi_interface .
5
6 TYPES: BEGIN OF ty_s_order_item_badi,
7         buksr      TYPE buksr,
8         sales_org  TYPE farr_sales_org,
9         header_id  TYPE farr_header_id,
10        kunnr      TYPE kunnr,
11        partner    TYPE bu_partner,
12        kdgrp      TYPE kdgrp,
13        INCLUDE    TYPE farr_s_ardoc AS grp_ardoc,
14        INCLUDE    TYPE farr_contract_item_incl_saw_ps AS enh_contr_item.
15 TYPES: END OF ty_s_order_item_badi.
16
17 TYPES: ty_order_item_badi TYPE SORTED TABLE OF ty_s_order_item_badi
18        WITH UNIQUE KEY grp_ardoc.
19
20
```

Figure 3.17 Parameters of Method DETERMINE_VERSIONS

To receive the data in the form of order, fulfillment, cost, and invoice RAIs in the RAR system from operational systems, inbound processing needs to have an API for a transaction or RAI type, which is the first substantial difference when comparing to classic inbound processing. After the BAdI is implemented, the API is automatically activated.

The integration component embedded in the operational system is responsible for collecting documents relevant to revenue accounting and securing their transfer to the RAR via the APIs for revenue accounting. Inbound processing receives data in the form of RAIs. After being processed by inbound processing, these RAIs are passed over to contract management for revenue recognition.

Besides the internal APIs, which are used for standard integration with other SAP products, inbound processing offers remote-enabled function modules. These function modules should be used for integration with operational systems without standard integration with RAR. For instance, if you need to integrate with external components used for order or fulfillments creation, these APIs should be implemented.

Classic or Optimized?

OIP and classic inbound processing can be used at the same time. The determination of which version to use for inbound processing is based on the source document item and how it was sent from the sender component. This means operational document item A in the sender component can create a revenue contract using OIP while another operation document B item from the same sender component can create a revenue contract using classic inbound processing.

For example, using the BAdI `FARR_DETERMINE_IP_VERSION`, you can decide that the operational documents with sales organization 0001 and order type OR use OIP, whereas operational documents in other sales organizations or other order types use classic inbound processing.

In addition, in contrast to classic inbound processing, if an external sender component is used, you don't need to configure any customizing or generate structures/interfaces. You can directly use the following function modules:

- `FARR_INBOUND_ORDER_API`
- `FARR_INBOUND_INVOICE_API`
- `FARR_INBOUND_FLFMT_API`
- `FARR_INBOUND_COST_API`

RAI Processing

RAIs are created when transactions relevant to revenue and cost recognition are received from operational systems. RAIs contain source information about such transactions, and this data is used to create or update revenue accounting contracts and POBs in RAR. RAIs received from sender systems are processed by the system in the following order:

1. Validation of key fields
2. Enrichments
3. Checks
4. Determination of additional attributes using BRFplus
5. Contract management

RAI statuses include **Processed**, **Partially Processed**, or **Postponed**. Once an RAI has been processed in all these steps without any errors, then it will acquire the status **Processed**. If an RAI that is relevant for more than one accounting principle is processed successfully for only a subset of the accounting principles, it acquires the status **Partially Processed**. If processing fails in any of these steps, the RAI acquires the status **Postponed**.

With inbound processing, you can process return order RAIs. Return order RAIs must refer to their predecessor order RAIs, and they usually contain negative quantities and amounts. After going to the RAI monitor (Transaction FARR_RAI_MON), reference to predecessor order RAIs must be documented by the operational system by filling in the following fields:

- **Predecessor Item Sender Component**

This is the sender component of the item that is the

parent to the item getting reversed (for sales and distribution integration, the value will be **SD**).

- **Logical System of the Predecessor Item**
This is the logical system that is defined in inbound processing.
- **Predecessor Item Type**
This is the type of the parent item. In most cases, the value will be **SDOI**.
- **Predecessor Item ID**
This the ID of the item that is the predecessor.

When inbound processing receives a return order item, the order item isn't treated as a separate and independent item. Return order items with their predecessor order item make up the predecessor document chain. Whenever inbound processing receives a return order item or a predecessor item, it internally builds the whole predecessor document chain from all relevant order items. These order items are then aggregated to produce one POB per accounting principle. The aggregation is applied to quantities and amounts. Predecessor items can't refer to other predecessor items because multiple layer hierarchies of return order RAIs aren't supported.

For the system to be able to match predecessor order items with return order items, some prerequisites must be fulfilled. The following must be the same in the predecessor order item and the return order item:

- Company code
- Transaction currency

- Units of measure
- POB category
- Value relevance flag

The aggregated quantity and amount must not be negative after all the items are aggregated. You can cancel return orders created on an existing sales order at any time. When the cancellation is issued on a return order from the sender component, the aggregation is triggered in inbound processing with the cancellation request. During this aggregation, the quantity and amount declared on the goods receipt of the return order isn't considered.

When the deletion flag is set for a return order item, the final date is set, or the RAI is value-relevant, these items aren't considered for the aggregation of quantities and the recalculation of amounts for the predecessor order item. The quantity unit isn't checked for return order items.

When the predecessor document and the related return order items are fully invoiced, the **Final Invoice** indicator is set in contract management on the POB of a contract of a related sales order. For more information about contract management, see [Chapter 5](#).

Data received from sender systems is processed in inbound processing by the system doing the following:

- Technical validations
- Checks
- Enrichments

If any issues occur during the technical validations, the RAIs are rejected, and processing of these RAIs isn't postponed. A `GENERAL_FAULT` exception is triggered with an error message by the APIs in inbound processing.

Postponed Items

For some errors during enrichments or checks, processing of the corresponding RAIs is interrupted, and these RAIs aren't processed further (e.g., because the RAIs are linked to a revenue accounting contract that is locked). Further, if an error occurs during processing in contract management, the corresponding RAIs are considered postponed items or partially processed items. Consequently, these RAIs are moved to the database tables for postponed items to be processed later with one of these options:

- With the Manage Revenue Accounting Items app (SAP Fiori)
- With Transaction `FARR_REPR_PPRAI` (report `FARR_REPROCESS_POSTPONED_RAII`)

Postponed RAIs are always stored with a postponement reason. The following postponement reasons can be used to specify which postponed items are selected for processing:

- **Locking error (L)**
RAI processing failed to lock the corresponding revenue accounting contract. Consequently, the RAI is saved as a postponed RAI and must be processed again.
- **Missing original document (M)**
The fulfillment or invoice RAIs that have failed to be processed before the initial order RAI is processed are

saved as postponed RAIs and must be processed again manually.

- **Persistent error (blank)**

This is used for any other undefined errors.

- **Failed assertion (X)**

If a serious error occurs during the processing of RAIs, processing is stopped completely, and all received RAIs are postponed.

- **Assumed invoice (A)**

Invoice items that are considered assumed invoices are saved as postponed RAIs until the posting date is reached, once the posting date is greater than or equal to the system date. The postponed invoice items must be processed using the application job for processing RAIs to convert them into real invoices, which are then sent to contract management. Finally, these invoice items result in postings.

If multiple postponement reasons apply for a postponed RAI, the priority of the postponement reasons—as defined previously—is applied.

If there are multiple postponed items relating to the same source document item, only the latest one is selected for processing, and the others are ignored. Postponed items are also ignored when a new RAI related to the same source document item is received. Postponed items are deleted only when an RAI related to the same source document item has been processed successfully for all available accounting principles.

When reprocessing items, the technical validation of postponed items is skipped because such items were already validated before being added to the postponed items table.

Postponed items are reprocessed in the following sequence:

1. Orders
2. Fulfillments
3. Costs
4. Invoices

3.3 Extending RAI Classes

The system that you're configuring could be integrated with any third-party system, SAP Billing and Revenue Innovation Management, SAP CRM, or sales and distribution. RAIs are provided with a set of standard fields that are relevant for integration and can pass the relevant information from the operational document to RAR. For example, when you have the sales order that you created in sales and distribution, the information from this document can be passed to RAR with the help of the standard fields provided as part of the RAR-sales and distribution integration and can be used for further processing. The fields that are already part of standard SAP are generally sufficient, but there could be cases where you need additional fields to be added to the RAI structure. This is precisely when you can add customer fields to RAIs. If you enhance RAIs, the interface for creating RAIs is enhanced. In addition, the database tables that store RAIs are enhanced.

To extend the RAI structure, you need to go to Transaction FARR_RAI_CONF, and choose the RAI class. You need to understand the business requirement clearly and look at the additional functionalities or extensions that you need to add to the standard fields to meet the business requirement. In addition, you'll need to list all the custom fields and finalize what data or part of data is processed and passed to the new custom fields that are added. Once you have the list, you need to understand the visibility, or where exactly the new custom fields will be needed, based on how you group

them. The grouping is made more systematic by the concept of the Easy Enhancement Workbench (EEW), which clearly defines four different includes. The details are given in the following sections.

3.3.1 Defining Extensions

When it comes to defining extensions in RAR, it follows the concept of the EEW, which means that SAP provides structures that can take additional fields, and these structures are attached to tables. So when you need to add a field, you'll add it in a structure, and the field will automatically be visible in all relevant components.

There are specific includes that are used to add the customer fields. The technical details of how to add to an include are given in this section, and the exact steps needed to add the custom fields to the RAI structure is explained in [Chapter 2, Section 2.3.2](#). In that section, we've included screenshots to demonstrate how you can achieve this technically, explained a few custom fields, and shown how the new added fields will be available during the RAI class configuration.

When deciding which include to add the customer fields to, consider the following categories:

- Fields required during RAI processing
- Fields required in revenue accounting contracts at the POB level
- Fields required in reporting

- Fields required in the revenue accounting contract at the header level

Depending on these categories, each field needs to be added to one of the following extension include structures (refer to [Chapter 2, Section 2.3.2](#), where we've explained this with an example):

- **INCL_EEW_FARR_ARL**
Fields only used in RAI processing.
- **INCL_EEW_FARR_POB**
Fields also used in revenue accounting contracts.
- **INCL_EEW_FARR_REP**
Fields also used in reporting.
- **INCL_EEW_FARR_CONTRACT**
Fields on the contract header level.

The custom fields are added to the include structures by appending to these structures. Once the structures are enhanced with the new custom fields, these fields will be available for you during RAI class configurations, as shown in [Figure 3.18](#).

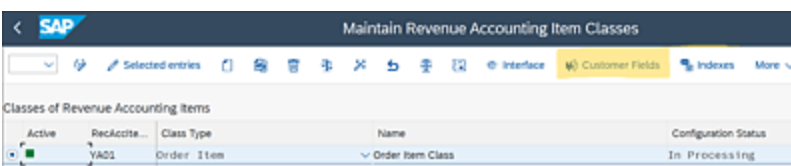


Figure 3.18 Adding Customer Fields to the RAI Class

Once you click on the **Customer Fields** button in [Figure 3.18](#), you see the screen shown in [Figure 3.19](#) for adding the desired custom fields. You're also provided with the option of choosing the right status (**Raw**, **Processable**,

or **Processed**), as to when the custom fields should be available in the RAI.

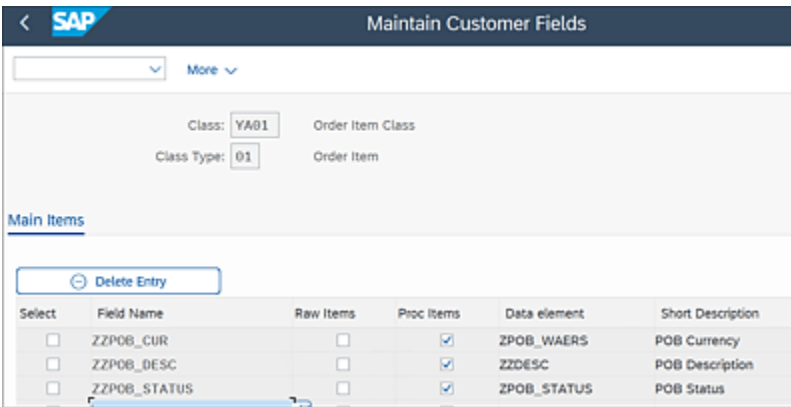


Figure 3.19 Choosing the Custom Fields

3.3.2 Populating Extensions

The previous section helped you understand how to extend the RAI structure and also how to control the visibility of the newly added fields at different **Raw**, **Processable**, and **Processed** statuses of the RAIs. With the addition of the custom fields, you need to add the custom logic to validate, enhance, or process the custom fields. However, standard SAP has already provided class `CL_FARR_RAI_IFCOMP` for validating the standard fields, and this class has the standard validations on the standard fields. Every business has its own unique requirements, which leads to adding custom fields. These fields have to be validated, and there are also requirements where you'll need additional validation, processing, or enhancing on standard fields apart from what is already being done in class `CL_FARR_RAI_IFCOMP`. To meet all these requirements, enhancement spot `FARR_ARL` includes four BAdIs you can use. Refer to [Chapter 4, Section 4.6](#), for complete details.

3.4 Summary

In this chapter, we explained the initial setup that needs to be done when configuring RAR: inbound processing. You learned about the differences between classic and optimized inbound processing, as well as some experiences that could come in handy when resolving the most common issues and problems.

In the next chapter, we'll discuss the next step: What happens with RAIs once they get created, and how do we turn RAIs into contracts and POBs?

4 Revenue Accounting Items

Revenue accounting items (RAIs) are sent from the source system (SAP or non-SAP) ready to be processed by revenue accounting and reporting (RAR) to create contracts, perform contract modifications, or perform fulfillments in a contract. In this chapter, we'll explain RAIs in detail and provide more information from a user perspective regarding processing and some basic error handling.

RAIs are integral to the RAR functionality in SAP S/4HANA, impacting everything from performance obligations (POBs) to contract management. We'll begin our discussion in this chapter with a walkthrough of RAI processing, before diving into RAI management, extensibility, and error handling. Next, we'll cover the usage of Business Rules Framework plus (BRFplus) for RAI integration with adjacent processes. To close, we'll provide instructions to create custom RAIs.

4.1 Processing Revenue Accounting Items

As mentioned in earlier chapters, RAIs come from external systems to the Adapter Reuse Layer (ARL) and are ready for processing. The ARL is part of RAR, which is the first stop for data sent from source systems. The ARL also performs data quality checks and uses BRFplus rules to transform data to structures that will represent contracts, POBs, and revenue schedules. Let's review the high-level design of the ARL, as shown in [Figure 4.1](#).

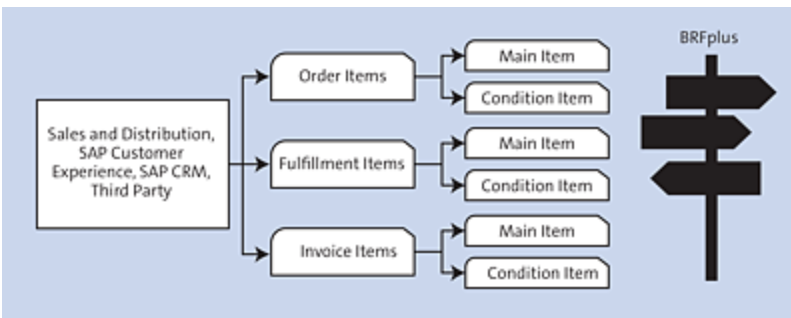


Figure 4.1 ARL Design

Once data is sent from operational systems via the standard RAR interface, it reaches the ARL. In the ARL, all data items are represented as RAIs, which are just data received and ready to be processed once it passes basic data quality checks.

All the items can be split into two categories:

- **Order items**

These items are used to perform basic contract operations: either contract creation or contract modification. If you're working with the sales and distribution interface, a sales order or sales contract will be usually represented as an order item. If you're working with external systems, it's usually data coming from customer relationship management (CRM) or order

management systems. The important thing is that order items don't have a predecessor to be created.

- **Items with predecessor**

When it comes to processing fulfillments or invoices, it's essential that you know to which order item this activity refers. For example, if there is a goods issue as a fulfillment event, you need to know to which order and item this fulfillment refers.

In the following sections, we'll explain processing for both RAI categories. First, let's walk through the available methods for processing that are relevant for both categories.

4.1.1 Processing Methods

All items created as RAIs have two categories: main items and condition items. [Figure 4.2](#) shows the RAI monitor (accessed via Transaction FARR_RAI_MON) with a split on main and condition items.

POB/FR. Contract	ItemStatus	Mat Exist	Error	Ex	His	Ex	Send Com.	SourceSys	SrcItem	Type	Source Item ID	Subarea	RevAcc.	Header ID	ItemID	Ref. Type	Reference ID	Customer	
								SO			SOO	0099000671000010	815	SO01	99000021	10	SOO	0099000671	100008400
								SO			SOO	0099000672000010	580	SO01	99000022	10	SOO	0099000672	100008400
								SO			SOO	0099000673000010	968	SO01	99000023	10	SOO	0099000673	100008400

Figure 4.2 Main and Condition Items in Transaction FARR_RAI_MON

Main items have information that is needed for identifying or creating IFRS 15 contracts. In the main items, you'll find customer, reference to document (source of RAI creation), or all dates relevant and needed for contract creation or fulfillment update, for example. Condition items contain values relevant for contracts. For order items, you'll find at

least two items: condition representing transactional price and condition representing standalone selling price (SSP). For fulfillment items, there can be one item (if you're having proof of concept [POC] fulfillment) or none; for invoice items, you can have one or several items depending on whether discounts are represented in RAR or not.

Irrespective of the RAI type, there are several ways to process RAIs. We'll discuss them in the next sections.

Manual Processing

The first option is manual processing: we'll approach this activity through Transaction FARR_RAI_MON, which is usually the first stop for all RAR users. Once you run the transaction, a screen like [Figure 4.3](#) will appear.

The screenshot shows the SAP FARR_RAI_MON transaction screen with the following sections and fields:

- Kind of Selection:** A dropdown menu set to "All Items Related to Order Items".
- Item Status:** A dropdown menu set to "03 Raw and Processable Items".
- Source Documents:** A grid of input fields for Component, Logical System, Type, Header ID (Orders), Item ID (Orders), Date, and Time, each with a corresponding "to:" field and a selection icon.
- Master Data:** A grid of input fields for Company Code, Business Partner Number, and Customer, each with a corresponding "to:" field and a selection icon.
- Revenue Accounting Item Classes:** A grid of input fields for Class and "to:" field with a selection icon.
- Further Attributes:** A grid of input fields for Reference Type, Reference ID, Error Status, and Revenue Accounting Item ID, each with a corresponding "to:" field and a selection icon. A checkbox for "Further Selections Exist:" is also present.

Figure 4.3 Transaction FARR_RAI_MON: Initial Screen

You first need to select which items you want to see from the **Kind of Selection** dropdown:

- **Order Items**

The system will show only items that are used to create or update contract/POB data in RAR.

- **Invoice/Fulfillment**

Invoice/fulfillment are items that are made as successor items. An important thing to have in mind is that the **Header ID** field controls the document number that needs to be processed. If you select **Invoice/Fulfillment**, you need to enter the invoice number or number of the post goods issue (PGI) document.

- **All Items Related to Order Items**

Opposite to the previous option, entering the number of the order and the system will display all items related to that order, including invoices and fulfillments, if any.

- **All Items**

The system displays all items relevant to the mentioned **Header ID**.

In general, the best option to select here is **All Items Related to Order Items** because it allows you to enter just the order number and search based on that number. This option ensures that all invoices and fulfillments will be displayed together with the order.

The next option is related to **Item Status**. RAIs can change into a total of three statuses before and after being transferred to RAR:

- **01: Processable**
This means the item is ready to be processed into contracts/POBs.
- **02: Processed**
Once an item gets processed, it's stored in a separate table and changes to the **Processed** status.
- **03: Raw and Processable Items**
Raw is an optional status, and the system can be configured to create items in this status that aren't yet ready to be processed. So, in this case, data checks are performed before the item even reaches the **Processable** status.

Once you save a document that is then replicated as an RAI, some items are created as raw and some as processable. The system understands the processable item as the item that is completed and ready to be processed into the POB/fulfillment event/invoice. If this isn't the case, it means the item didn't pass all consistency checks and can't be processed further. However, RAR differentiates the level of issues we might have, and the following rules apply:

- If an item has some error in any key field, the item won't be created at all.
- If an item has an error, but not in key fields, it will be created as a raw RAI.
- If an item doesn't have any errors, it will be created as a processable RAI.

Raw Status and When to Use It

Whether to use **Raw** status or go straight to **Processable** status is a common question. As usual, there is no answer to suit all customers, but the following needs to be considered: If items are initially created as raw, we're clearly adding one more step for users to execute before they can finish processing RAIs (RAI first needs to be transferred, and only after that, can it be processed in RAR). On the other hand, we're enforcing one more level of data check before the item is created. Now, the real question is whether we need this extra data consistency check. If we're using integration with sales and distribution, this most likely isn't needed because any data error would need to be corrected in the document we're trying to process. However, if we're integrating with an external system, it's highly likely that the error couldn't be corrected in the source system, so it would make sense to have **Raw** status items before they get into **Processable** status. But if there is a system doing data consistency checks before items are sent to RAR, using an additional status might not be necessary. So, before making a decision about using an extra status, all these things need to be considered.

In the **Source Documents** section, you can make all the additional selections to pinpoint the exact items requested to be displayed:

- **Component**

Represents the system to which RAR is being integrated and which will be used as the source for displaying data. Possible entries here are a result of your definition of

sender components in inbound processing, as discussed in [Chapter 3, Section 3.2](#).

- **Logical System**

Specifies the logical definition of the system that is the source for RAIs you need to process. The source item's logical system, component, type, and ID constitute the link to the source item (e.g., a sales and distribution order item) that the RAI (e.g., a sales and distribution return order) relates to.

- **Header ID (Orders)**

Reference to the original document that you try to find. This field will have different meanings depending on what was entered in **Kind of Selection**: if you select **Order Items** or **All Items Related to Order**, then the order number needs to be entered here, whereas if the **Invoice /Fulfillment** item is selected, the invoice of the PGI document number needs to be entered here.

Options in the **Master Data** section give more ways to limit data that will be displayed. For example, you can enter a **Customer** number or **Business Partner Number** to fetch the exact records you need.

Sometimes, you may need to reprocess items that are already in the error. You can select those items in the **Further Attributes** section using the **Error Status** field, as shown in [Figure 4.4](#).

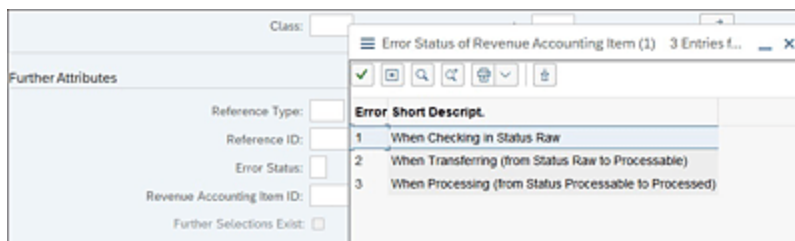
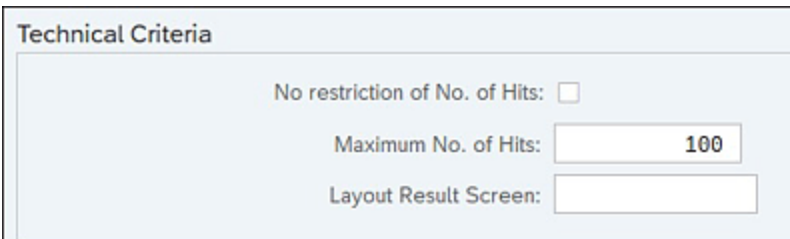


Figure 4.4 Error Status Display

As already mentioned, before being transferred to RAR, items can be in error status, which is further split into when the error actually appeared. There are in total three different statuses being triggered in different points of the RAI lifecycle, which can be used to filter RAIs further that need to be displayed. At the bottom of the screen under the **Technical Criteria** section, the technical details can be found, as shown in [Figure 4.5](#).



Technical Criteria

No restriction of No. of Hits:

Maximum No. of Hits:

Layout Result Screen:

Figure 4.5 Technical Criteria for Display

Sometimes, even if you limit the selection criteria, the number of items to be displayed is too high. Because this activity can be time-consuming, by default, SAP limits the maximum number of hits to 100 main items. That can be overridden by either entering some other number in the **Maximum No. of Hits** field or simply selecting that no restriction of number of hits should be applied (**No restriction of No. of Hits**).

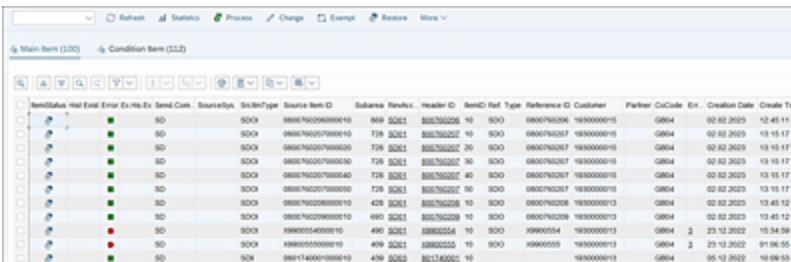
Further Selections

At the top of the initial screen of Transaction FARR_RAI_MON, you can see the **Further Selections** button, which you can use to access all fields in the main item structure that can be used as selection criteria. This

option becomes particularly useful if you've introduced some custom fields that could be used to filter the RAI display. We'll discuss this in more detail in [Section 4.3](#).

Once items are selected, click the **Execute** icon, and the system will display entries that fit the selections made, as shown in [Figure 4.6](#).

In the first column, **ItemStatus**, the system will display raw, processable, or processed items depending on your selection criterion. The next important column is **Error** status. Here, the system uses a simple traffic light system to display the proper status of the RAI: no errors equals a green light, and error equals a red light. If there is an item with an error, you can click on the item to go to the log where you'll see what caused the error while processing the RAI.



ItemStatus	Has Error	Ex-Has Ex.	Send Com.	Source Sys.	SubNetType	Source Item ID	Subarea	Rev/cls.	Header ID	ItemID	Ref. Type	Reference ID	Customer	Partner	CuCode	Err.	Creation Date	Create Tim
SO				SOO	SOO	00000000000010	000	SOO1	00000000	10	SOO	00000000	'0000000'0	GR04		02.02.2025	12:45:11	
SO				SOO	SOO	00000000000010	718	SOO1	00000000	10	SOO	00000000	'0000000'0	GR04		02.02.2025	13:15:17	
SO				SOO	SOO	00000000000020	718	SOO1	00000000	20	SOO	00000000	'0000000'0	GR04		02.02.2025	13:15:17	
SO				SOO	SOO	00000000000030	718	SOO1	00000000	30	SOO	00000000	'0000000'0	GR04		02.02.2025	13:15:17	
SO				SOO	SOO	00000000000040	718	SOO1	00000000	40	SOO	00000000	'0000000'0	GR04		02.02.2025	13:15:17	
SO				SOO	SOO	00000000000050	718	SOO1	00000000	50	SOO	00000000	'0000000'0	GR04		02.02.2025	13:15:17	
SO				SOO	SOO	00000000000010	428	SOO1	00000000	10	SOO	00000000	'0000000'0	GR04		02.02.2025	13:45:12	
SO				SOO	SOO	00000000000010	490	SOO1	00000000	10	SOO	00000000	'0000000'0	GR04		02.02.2025	13:45:12	
SO				X99000	X99000	00000000000010	490	SOO1	00000000	10	SOO	X9900000	'0000000'0	GR04	2	25.12.2022	15:34:59	
SO				X99000	X99000	00000000000010	490	SOO1	00000000	10	SOO	X9900000	'0000000'0	GR04	2	25.12.2022	01:06:55	
SO				SOO	SOO	00000000000010	478	SOO1	00000000	10	SOO	00000000	'0000000'0	GR04		05.12.2022	16:09:55	

Figure 4.6 Transaction FARR_RAI_MON: Results

For the results just shown, you can see what error is behind the item that wasn't processed, as shown in [Figure 4.7](#). In our example, you can see that error **C01** was raised after processing our RAI.

Date/Time/User	Number	External ID	Object text	Subobject Text
02.02.2023 01:33:59 JS4753	7	JS475320230202013...	Revenue Accounting	Processing of Rev
● Problem class Important	4			
▲ Problem class Medium	2			
■ Problem class Additional Information	1			

Ty_	Message Text	Ltxt
■	Start of processing of order item SDOI09900554000010 for accounting principle IFRS	
●	Inflight check: Error C01/Contract 100870 found before save to database	
●	C01: Allocation error found for contract 100870	
▲	Performance obligation type of performance obligation 5000000000000004 is missing	
▲	Name of leading performance obligation 5000000000000004 is missing	
●	Required field 'Fulfillment Type' of performance obligation 5000000000000004 is empty.	
●	Required field 'Event Type' of performance obligation 5000000000000004 is empty.	

Figure 4.7 Transaction FARR_RAI_MON: Error

The same results can be observed if you access Transaction SLG1 for displaying system logs.

In addition, you need to understand when this error occurred, which can be checked by looking at the results table (refer to [Figure 4.6](#)) and column **Err...** on the right. Here, you can see numbers 1-3, which fit to numbers that you can enter on the initial screen and that represent the type of error as due to creation of item, change of status, or processing.

Also in the results table, you can click on the values in the **Header ID** field to go directly to the document that was used to create this RAI item. However, this functionality works only when integrating with sales and distribution.

A special explanation is needed for the **Subarea** column (refer to [Figure 4.6](#)). When we say *subareas* in RAR, it's the KEYPP field. The KEYPP field is used for the parallel processing framework (PPF). For more information, see [Section 4.1.2](#).

The last two columns in the results table are reserved for the date (**Creation Date**) and time (**Create Time**) when the item was created. This data can't be found in the table

that stores the main and condition items, but it's being decoded from the **Timestamp** field. This concept deserves additional explanation. As mentioned before, RAIs belong to the ARL, which is the integration layer between the external component and RAR. In practice, this means that an item can be created in an external component, then multiple changes can happen before the items get processed in the ARL and RAR data gets updated. All of these items will be created as separate RAIs with different timestamps, but while processing them, only the last item will be processed. This approach sometimes can create issues.

Example of Issue with Latest Timestamp

Let's say a user created a sales order with an item that needs to create a time-based POB, and there is a contract with a start and end date. After saving, an RAI was created, but it wasn't processed. Then, the user decided that the item needed to be rejected, so the rejection reason was used in the sales order, which was again saved. Now, in Transaction FARR_RAI_MON, there are two items, but the item with the rejection reason will have the latest timestamp and so is the only item processed. However, once the user sets a rejection reason in the sales document, this automatically triggers the population of the **Finalization Date** field with the current date. Therefore, the system tries to create a new POB, which needs to be created as terminated. This isn't possible, and it will throw an error while processing.

The solution for this exact challenge is in process organization: once created, the RAI needs to be

processed. Once that has happened, then the item can be processed with the rejection reason. If the process were executed in such an order, the issue wouldn't appear because the POB would be created and then terminated. In other words, the system wouldn't try to create terminated items.

Now, when the list contains all items based on your selections, you need to perform some activities. Possible options are displayed at the top of the screen, as shown in [Figure 4.8](#).



Figure 4.8 Transaction FARR_RAI_MON: Options

Let's walk through a few of the available buttons:

- **Refresh**

This button's usage is the same as in other applications in SAP: if there was some change in the external application that is used to create RAIs, this change will be displayed by clicking this button.

- **Statistics**

When this button is selected, the system will display a breakdown of all RAIs that fit the selection made on the first screen of Transaction FARR_RAI_MON, as shown in [Figure 4.9](#).

Status	Message Text	Counter
🔍	RAI class SD01 Main Items Status 'PROCESSABLE'	10
🔍	RAI class SD01 Condition Items Status 'PROCESSABLE'	20
🔍	RAI class SD03 Main Items Status 'PROCESSABLE'	90
🔍	RAI class SD03 Condition Items Status 'PROCESSABLE'	92
Σ	Total Main Items	100
Σ	Total Condition Items	112

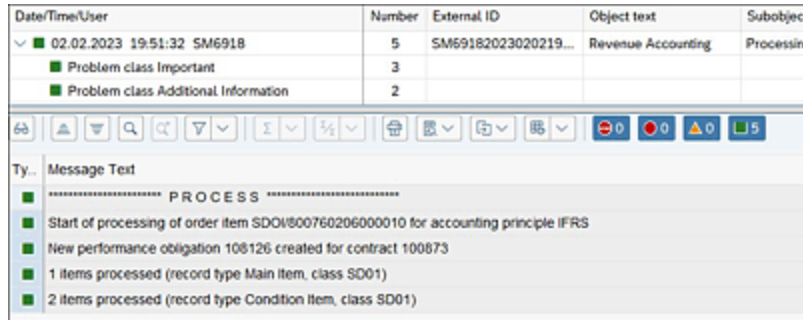
Figure 4.9 Transaction FARR_RAI_MON: Statistics Display

- **Process**

This button processes RAIs, that is, moves them from status **Processable** to status **Processed**. Before you can click this button, you must select items in the table that need to be processed. The important thing here is that if you have multiple items that belong to the same **Header ID** and select only one of them, all will be processed if they belong to the same source document type. The reason for this is that revenue allocation occurs on the contract level, so it's essential that all items belonging to the same contract (represented by **Header ID**) get processed.

Once items are being processed, the system will issue a message about the results of the processing. These results (irrespective of whether they are errors or successes) are written in the system log. In the bottom of the message screen, you can use the **Display Log** icon to display the log, which contains useful information about what was created/updated (or for an error, information about what issue occurred), as shown in [Figure 4.10](#). This

functionality is useful when it comes to either troubleshooting or further analysis. In the background, the system is transferring data between processable and processed tables.



Date/Time/User	Number	External ID	Object text	Subobject
02.02.2023 19:51:32 SM6918	5	SM69182023020219...	Revenue Accounting	Processing
Problem class Important	3			
Problem class Additional Information	2			

Message Text

***** PROCESS *****

- Start of processing of order item SDO/800760206000010 for accounting principle IFRS
- New performance obligation 100126 created for contract 100873
- 1 items processed (record type Main item, class SD01)
- 2 items processed (record type Condition item, class SD01)

Figure 4.10 Transaction FARR_RAI_MON: Processing Results

We'll discuss two more key buttons, **Change** and **Exempt**, in [Section 4.2](#) during our coverage of RAI management.

Automatic Processing

Often, the number of RAIs being created is substantial, and running Transaction FARR_RAI_MON can become a technical challenge. Another possible problem is that the process of calculating revenue needs to be performed in an orderly manner, meaning that processing RAIs needs to be executed prior to calculating results, so we get accurate calculated revenue. This is particularly relevant for companies that are often going through contract modification processes.

All of this means that processing RAIs manually might not be the most optimal solution. To perform RAI processing in an automated way, SAP delivers programs that might help

in organizing the most streamlined processes of RAI transformation into contracts and POBs.

SAP delivers two transactions that can be run in the background to get contracts created/updated or any other items successfully processed: Transaction FARR_RAI_TRANS and Transaction FARR_RAI_PROC. After running these transactions sequentially, you'll use the monitor (Transaction FARR_RAI_MON) to complete processing and arrive at a contract.

Let's dig a little deeper. Transaction FARR_RAI_TRANS is used to transfer items from the **Raw** status to the **Processable** status. When the program is run, you can see the selections, as shown in [Figure 4.11](#).

Selection data is very similar to the selection data in Transaction FARR_RAI_MON, and here you can limit **Selection Data** based on item class (**Rev. Acc. Itm Class**), **Sender Component**, and **Header ID**, which represent the exact items we need processed.

The screenshot shows a dialog box with three main sections:

- Selection Data:** Contains fields for 'Rev. Acc. Itm Class', 'Sender Component', 'Source Item Logical System', 'Header ID', and 'Company Code'. Each field has a 'to:' field and a selection icon. A 'Further Selections Exist' checkbox is at the bottom.
- Technical Parameters:** Contains 'Number of Intervals' (set to 2), 'Block Size For Mass Selection' (set to 1.000), 'Simulation Mode' (checked), 'Dialog Mode' (unchecked), and 'Synchronous Call' (unchecked).
- Settings for Application Log:** Contains 'External ID' (text field) and 'Problem class' (dropdown menu set to '2 Important').

Figure 4.11 Transaction FARR_RAI_TRANS

If you have multiple item classes to be processed, it's important to know the order of items and how they are processed. Using sales and distribution as an example, the program transfers RAIs in the following order of source document types:

1. Order items without predecessor (SDOI)
2. Order items with predecessor (SDOI)
3. Fulfillment items (SDFI)
4. Planned invoice items (SDPI)
5. Invoice items (SDII)

Irrespective of which items are available, these five steps will be always executed.

Two more options (technical parameters) are important to understand. **Block Size For Mass Selection** determines the number of subareas that are used for processing. For example, an entry of **1,000** there means that all RAIs will be assigned to those 1,000 subareas. Which subarea an item is assigned to is determined based on the **Header ID** and RAI source document type (all RAIs with the same source document and **Header ID** will belong to the same subarea). For more information on subarea determination, see [Section 4.1.2](#).

Next is the **Number of Intervals** that will run concurrently to process those RAIs. This number depends on the technical capabilities of the system; however, it should be clear that the locking mechanism is activated in that case—

each item assigned to the same subarea is locked for processing.

In addition, there is also a setting named **Synchronous Call**, which causes the results of the running application to be written in the system log and can be accessed by Transaction SLG1. At the bottom of the screen, under **Settings for Application Log**, you have options for how detailed this log should be. If you don't select **Synchronous Call**, the batch monitor will be displayed after running the application.

When you're done filling in the key fields, click the **Execute** button or **Schedule** to schedule it as a job from the menu in the top-left corner.

Next, Transaction FARR_RAI_PROC moves items from the **Processable** status to the **Processed** status. [Figure 4.12](#) shows how the transaction looks and which options are available.

Selection Data	
Rev. Acc. Item Class:	to: [] []
Sender Component:	to: [] []
Source Item Logical System:	to: [] []
Header ID:	to: [] []
Reference Type:	to: [] []
Reference ID:	to: [] []
Company Code:	to: [] []
Further Selections Exist:	<input type="checkbox"/>

Technical Parameters	
Number of Intervals:	[2]
Block Size For Mass Selection:	[1,000]
Dialog Mode:	<input type="checkbox"/>
Synchronous Call:	<input type="checkbox"/>

Settings for Application Log	
External ID:	[]
Problem Class:	[2 Important]

Figure 4.12 Transaction FARR_RAI_PROC

As is visible from the screen, this transaction has the same options as Transaction FARR_RAI_TRANS. The main and most important difference is technical: while Transaction FARR_RAI_TRANS is used to transfer items from status 0 to 2 (**Raw** to **Processable**), Transaction FARR_RAI_PROC is processing items and moving them from status 2 to 4 (**Processable** to **Processed**).

So, you might be asking the following question: If you're not using **Raw** status, do you need to also schedule Transaction FARR_RAI_TRANS? The answer is yes. The reason was mentioned at the beginning of this chapter: RAIs will be created if basic data checks are passed, so they will be created in **Raw** status as ready to be processed.

The process of sending data to create an RAR contract ends with Transaction FARR_RAI_MON, which, in this case, serves as an error correction tool: you can run it to verify how many items ended in error, what kind of errors are present, and how they can be resolved before new RAIs are created. So, to have a proper picture of the processed and remaining items in error, ideally, you should schedule Transaction FARR_RAI_TRANS before Transaction FARR_RAI_PROC and then run Transaction FARR_RAI_MON.

4.1.2 Parallel Processing Framework

A key functionality to highlight is the use of the parallel processing framework (PPF) to create subareas, as we've touched upon in previous sections. The RAIs are grouped on specific areas and assigned the same KEYPP value, and then the packages are created. The packages are then

distributed to jobs. The number of parallel jobs that can be created is again dependant on configuration and is determined by your Basis team. The creation of parallel jobs is sensitive as the packages must be grouped in a way that the locking is effective. The technical attributes of the KEYPP field are shown in [Figure 4.13](#).

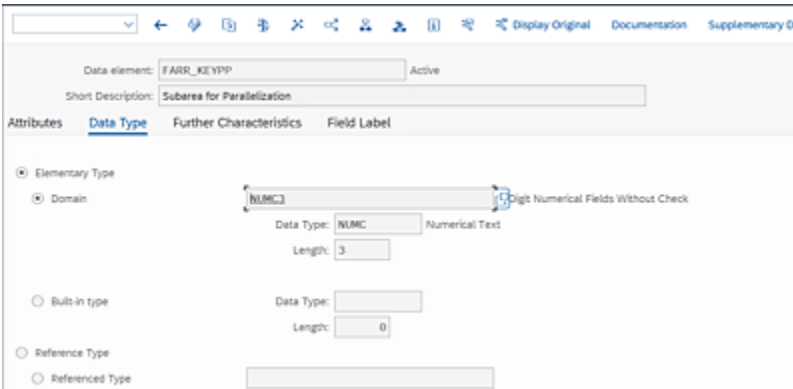


Figure 4.13 KEYPP Field Technical Description

The KEYPP field can have values from 000 to 999. In RAR, we generally must deal with huge volumes of data. The PPF has been designed to speed up the processing by dividing the data into packages, and the packages are passed on to parallel jobs or child jobs that run in parallel to save time. When defining the packages, you must consider the data locking and data grouping. To group the data into packages, the KEYPP field is used.

Lock object EFARR_KEYPPBUKRS is used during the ENQUEUE process of locking the KEYPP fields for processing, as shown in [Figure 4.14](#).

W	Lock parameter	Table	Field
<input checked="" type="checkbox"/>	MANDT	FARR_S_KEYPP_BUKRS_ENQ	MANDT
<input checked="" type="checkbox"/>	KEYPP	FARR_S_KEYPP_BUKRS_ENQ	KEYPP
<input checked="" type="checkbox"/>	BUKRS	FARR_S_KEYPP_BUKRS_ENQ	BUKRS
<input checked="" type="checkbox"/>	OBJPP	FARR_S_KEYPP_BUKRS_ENQ	OBJPP

Figure 4.14 KEYPP Lock Object

Most of the RAI mass processing activities can be done in parallel. The overall workload (the RAIs to be processed for the specified selection criteria) is split into intervals based on field KEYPP. There are 1,000 possible different KEYPPs (000–999). If you start the RAI processing for 100 intervals, 10 KEYPPs will be assigned per interval (10 x 100 = 1.000).

The parallel processing runs are currently supported for the following:

- RAI transfer (Transaction FARR_RAI_TRANS)
- RAI processing (Transaction FARR_RAI_PROC)

The usual number of intervals should be 3–5 times higher than the number of parallel jobs (e.g., 30–50 intervals for 10 parallel processes). The number of parallel processes may be restricted by Basis configuration.

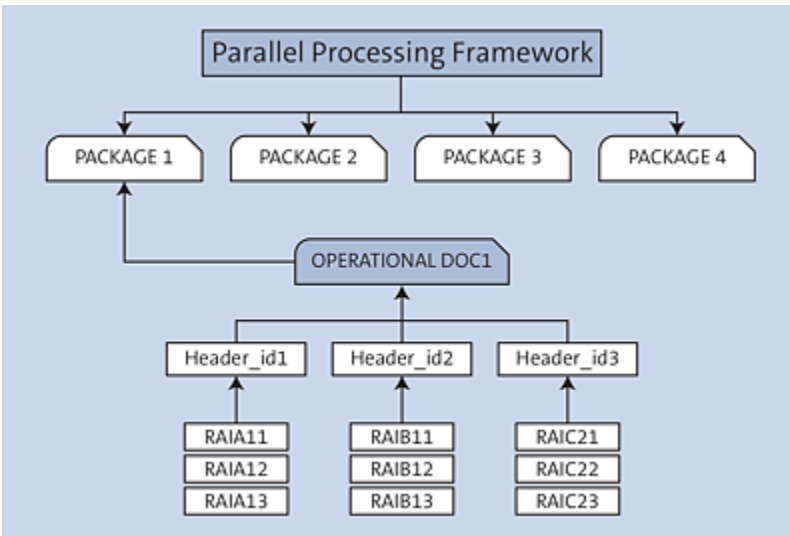


Figure 4.15 Parallel Processing Model

KEYPP will be assigned based on the same references, for example, all the header IDs belonging to a certain operational document or order will be assigned the same KEYPP, and then they will all be grouped in the same package for parallel processing. So, all the RAIs belonging to the same header ID and the header IDs belonging to the same operational document are packed in a single package, as shown in [Figure 4.15](#). KEYPP is also heavily used in contract accounting and SAP Billing and Revenue Innovation Management.

4.1.3 Processing Order Items

Order items are the starting point for RAR contract creation or modification of existing contracts whenever there is an initial creation of RAR contracts or subsequent changes. Definition of the order item class is done in inbound processing (see [Chapter 3, Section 3.2](#)). Here, we'll focus on the instructions for processing order RAI classes.

In our example, we're creating one RAR contract based on a sales and distribution order. Once we populate the order with needed data (customer, condition items, duration, etc.), we save it and get its number, which we'll use in Transaction FARR_RAI_MON while searching for the order. It's important to mention that only complete documents will be transferred to RAR; if you get a message that a document is incomplete, but you can still save it, such a document won't be transferred.

After entering the sales order number in the **Header ID (Orders)** field and clicking **Execute**, we'll see the result shown in [Figure 4.16](#).

ItemStatus	Error	Subcontract Number	Header ID	Source Item ID	Customer	Contract Date	Contract Time	Quantity
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1
Processable	■	4071 0000	0000000000	0000000000	0000000000	12.02.2020	08:45:00	1

Figure 4.16 Processing Order Items in Transaction FARR_RAI_MON

The first two columns are empty, which means that these RAIs are coming for the first time in Transaction FARR_RAI_MON so the contract and POB number aren't yet assigned. The **ItemStatus** column and the green square in the **Error** column tell us that the item is in **Processable** status and was created without any errors, meaning it's ready for processing. After columns of information about the data in the sales order, the date and time when the item was created are shown.

We also can see **SDPI** items in the source item type (**SrcItemType**) column, which means that billing of this sales

order is to be done on a regular basis, so the order has planned items representing these future billings.

Once we process the data by clicking the **Execute** button, we'll get information about what contract was created, as shown in [Figure 4.17](#).

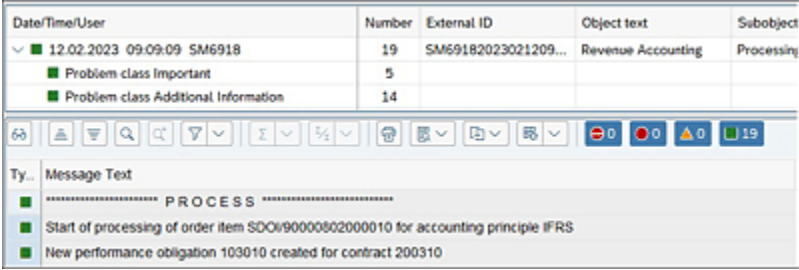


Figure 4.17 Contract Created in Transaction FARR_RAI_MON

Now, as the next step, we'll have our sales team make some changes. Let's assume they perform two types of changes:

- 1. They will change some text in the order.
- 2. They will change the standalone selling price (SSP).

In addition, it's important to mention that these two changes are to be performed at two separate moments in time, but no additional Transaction FARR_RAI_MON processing occurs in the meantime.

So, once we run Transaction FARR_RAI_MON, we'll see that two items are there, as shown in [Figure 4.18](#).

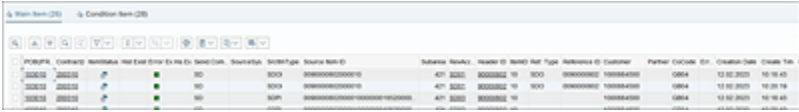


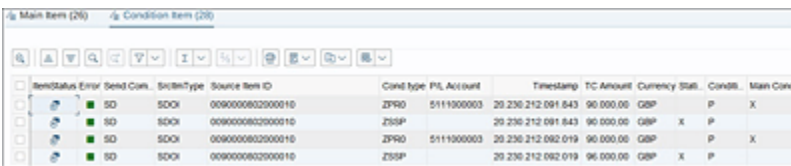
Figure 4.18 Changes in FARR_RAI_MON

Now, the contract and POB number are populated in the first two columns, which tells us that these RAIs represent a

contract modification. It's possible to click on the **Contract** number to go straight to the contract in the respective SAP Fiori app or SAP Business Client program.

To verify that the SSP change was transferred, we can select the **Condition Item** tab, as shown in [Figure 4.19](#).

Here, we see four conditions: each change produced two condition entries. One condition represents a pricing condition (**ZPRO**) with a **P/L Account** assignment. This condition is marked as the main condition type (**Main Cond.**). The SSP is a statistical condition type (**ZSSP**), so the **Main Cond.** checkbox remains empty.



Item Status	Error	Send Cond.	SubItem Type	Source Item ID	Cond. type	P/L Account	Timestamp	TC Amount	Currency	Stat.	CondB.	Main Cond.
			SO	0090000802000010	ZPRO	5111000003	20.230.212.091.843	90.000.00	GBP		P	X
			SO	0090000802000010	ZSSP		20.230.212.091.843	90.000.00	GBP	X	P	
			SO	0090000802000010	ZPRO	5111000003	20.230.212.092.019	90.000.00	GBP		P	X
			SO	0090000802000010	ZSSP		20.230.212.092.019	96.000.00	GBP	X	P	

Figure 4.19 Transaction FARR_RAI_MON: Conditions

By looking at these items, we see that an item with a later timestamp is an SSP with the amount 96,000.00, which means this is the item entered as last (highest timestamp).

Now we go with processing RAIs. The important thing is in the last two columns where these two items are created with different timestamps. The question is, do we need to only select the item we want to process? The answer is no because the system will automatically select only the last item and delete the rest.

There's one important point to mention regarding conditions. In order items, the order always passes at least two items: one is the main item, which represents the transactional price, and the other one is the statistical item,

which represents the SSP. There can be specific situations, but this one is the most basic one. If you're considering integration with sales and distribution, a pricing procedure is in place that determines the pricing condition to be translated as the transaction price. [Figure 4.20](#) shows what the pricing procedure assigned to a sales document looks like. All condition types appear on the lefthand side with their assigned values. On the right side, the condition value is price multiplied per quantity, which represents the value that will be passed to RAR.

L.	CnTy	Name	Amount	Crcy	per	UoM	Condition Value	Curr.	Status
	ZPR0	PG: List Price.	1,00	GBP		1PC		1,00	GBP
	ZKUM	Cumulation cond-Stat	1,00	GBP				1,00	GBP
		Gross Value	1,00	GBP		1PC		1,00	GBP
		Net Value of Item	1,00	GBP		1PC		1,00	GBP
	ZSSP		1,00	GBP		1PC		1,00	GBP
		Net Value 2	1,00	GBP		1PC		1,00	GBP
	ZTXD	Tax Service - Doc Lv	0,000	%				0,00	GBP
	ZTXE	TaxSvc call - doc lv	0,000	%				0,00	GBP
	ZWST	Output Tax	0,000	%				0,00	GBP
		Total	1,00	GBP		1PC		1,00	GBP

Figure 4.20 Pricing Procedure

Here, the condition type (**CnTy**) represents price **ZPR0**. Once we save document, it will create RAIs that are ready for processing, as shown in [Figure 4.21](#).

Once items get processed, an RAR contract is created with all the corresponding POBs.

ItemStatus	Hist	Exist	Error	Send Com.	SourceSys	SrcItemType	Source Item ID	Cond.type	PiL Account
✓				SD		SDOI	0090000847000010	ZPR0	5111000003
✓				SD		SDOI	0090000847000010	ZSSP	

Figure 4.21 Condition Items in Transaction FARR_RAI_MON

Now, let's assume that during invoicing, there is a change in the condition type. This isn't uncommon because often

companies give some discounts at the moment of invoicing. [Figure 4.22](#) shows that the condition type changed between the order and the invoice (instead of ZPRO, we get **ZPRG**). This situation more often occurs in the case of credit/debit memos when users want to change the value of the debit memo, and there is a specific condition type only for manual pricing.

The screenshot shows a SAP Pricing Elements table with the following data:

Condition Type	Name	Amount	Crcy	per	UoM	Condition Value	Curr.	Status	NumCCo
ZPRG	PG Global List Price	15,00	GBP		1PC		15,00 GBP		1
ZKUM	Cumulation cond-Stat	12,00	GBP				7,20 GBP		0
	Gross Value	15,00	GBP		1PC		15,00 GBP		1
	Net Value of Item	15,00	GBP		1PC		15,00 GBP		1
ZSSP		12,00	GBP		1PC		7,20 GBP		1
	Net Value 2	7,20	GBP		1PC		7,20 GBP		1
ZTXD	Tax Service - Doc Lv	0,000	N				0,00 GBP		0
ZTXE	TaxSvc call - doc lv	0,000	N				0,00 GBP		0
ZWST	Output Tax	0,000	N				0,00 GBP		0
	Total	15,00	GBP		1PC		15,00 GBP		1
	Standard - USA With	15,00	GBP		1PC		15,00 GBP		1

Figure 4.22 Changed Condition Type during Invoicing

Because there is a standard integration between sales and distribution and RAR, you'll get an RAI ready for processing where now we see condition type **ZPRG** instead of **ZPRO**. However, RAR won't allow such an item to be processed due to the difference in the main condition type in the order (defined while creating the contract) and the invoice items (defined during invoicing).

4.1.4 Processing RAIs with a Predecessor

In terms of process chain dependency, all RAIs can be divided into those created with a predecessor and those created without a predecessor. In the previous section, we described processing RAIs without predecessors, which are

used to create or change RAR contracts. When it comes to RAIs with predecessors, there are multiple RAI types that are created depending on the fulfillment type of an RAI, with the exception of planned items (created as a result of a billing plan) and invoice items (created as a result of invoicing). In other words, fulfillment items have several different types depending on the type of fulfillment expected, while invoice items are always the result of invoicing only.

When you need to process an RAI item with predecessor, the best method is to select the option in Transaction FARR_RAI_MON for processing **All Items Related to Order Items**, as shown in [Figure 4.23](#). By that, you're ensuring that the system will pick up all items related to an order irrespective of whether they are fulfillment or invoice items.

The screenshot shows the selection screen for transaction FARR_RAI_MON. At the top, there are menu options: Execute, Get variant..., Program Documentation, Further Selections, Personalize, and More. Below this, the 'Kind of Selection' section has a dropdown menu set to 'All Items Related to Order Items'. The 'Item Status' section has a dropdown menu set to '05 Raw and Processable Items'. The 'Source Documents' section contains several input fields: Component, Logical System, Type, Header ID (Orders) (with the value 41007138), Item ID (Orders), Date, and Time (with the value 00:00:00). Each field is followed by a 'to:' field and a magnifying glass icon. At the bottom left, there is a 'Master Data' label.

Figure 4.23 Transaction FARR_RAI_MON with Order Number

For example, if you just enter an order number (**41007138**), in the result table, the system will display all items related to this order number, as shown in [Figure 4.24](#). Here, you can see, based on the source item type (**SrcItmTy**), that it

picked up all fulfillments and invoices created based on this order that can be processed.

Note that in the **Original Item ID** field (not shown), the system stores the order item, which is the main source (predecessor) for the follow-up item.

A few things should be kept in mind when processing follow-up items related to the original item. Sometimes, the follow-up item is created before the original item is processed. In this case, the follow-up items won't be processed before the original item is processed without an error. In such cases, you'll receive an error, as shown in [Figure 4.25](#).

ItemID	Ref	Ty	Header ID	RevAcc	Subarea	Source Item ID	SourceS
1190			70000118	SDQ2	012	0070000118001190	SDFI
1200			70000118	SDQ2	012	0070000118001200	SDFI
1210			70000118	SDQ2	012	0070000118001210	SDFI
1220			70000118	SDQ2	012	0070000118001220	SDFI
2			9110000084	SDQ3	962	9110000084000002	SDI
3			9110000084	SDQ3	962	9110000084000003	SDI
4			9110000084	SDQ3	962	9110000084000004	SDI
1190			6110000011	SDQ3	012	6110000011001190	SDI
1200			6110000011	SDQ3	012	6110000011001200	SDI
1210			6110000011	SDQ3	012	6110000011001210	SDI
1220			6110000011	SDQ3	012	6110000011001220	SDI
1190			9110000086	SDQ3	012	9110000086001190	SDI
1200			9110000086	SDQ3	012	9110000086001200	SDI
1210			9110000086	SDQ3	012	9110000086001210	SDI
1220			9110000086	SDQ3	012	9110000086001220	SDI

Figure 4.24 Results for Items with Predecessors

■	Start of processing of invoice item SDI/9110000286000001 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDI/9110000286000001)
■	Start of processing of invoice item SDI/9110000285000001 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDI/9110000285000001)
■	Start of processing of invoice item SDI/6110000022000020 for accounting principle IFRS
■	Start of processing of invoice item SDI/9110000600000004 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDI/9110000600000004)
■	Start of processing of invoice item SDI/9110000600000003 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDI/9110000600000003)

Figure 4.25 Error for Invoice Processing and POB Determination

In this case, the system is notifying you that the predecessor item for the mentioned invoice couldn't be

found. There can be different reasons for this, but all are related to process design and its execution. In addition, credit/debit memos without reference can't be processed.

The second issue that often arises is that a change was made to the order and RAIs were created before the invoice was processed. This means the order item to which the invoice relates already exists as unprocessed in Transaction FARR_RAI_MON with an older timestamp than the one with invoice. This potentially could cause an inconsistency because the RAR engine isn't aware of the type of change in the order item. To prevent this, the ARL won't allow such an RAI to be processed. It will produce an error, as shown in [Figure 4.26](#).

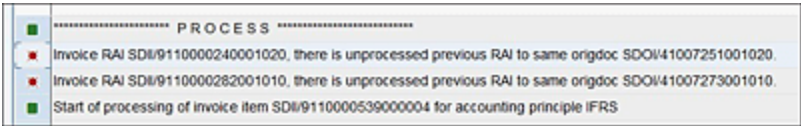


Figure 4.26 Invoice Error Due to an Unprocessed Order RAI

In addition, RAR doesn't support the change of condition types between invoice and order items. Often users use different condition types in a credit memo (which is also an invoice item) than they do in an order. RAR won't let such items be processed and will throw an error about condition types inconsistency between the order and invoice item. These kinds of situations should be resolved during process design.

4.2 Managing Revenue Accounting Items

Along with processing RAIs, there is often a need to do some other activities with RAIs before, during, or after processing. These activities can be split into either changing items after they are created in the ARL or removing them from the list for processing.

Despite RAR being a tool with high automation (meaning that data usually isn't created in RAR itself, but rather is sent from external applications), sometimes there is a need to manipulate the data before it's sent for processing. These options are enabled to ensure that once data reaches RAR, it's of the highest possible quality. But these options come with offsets as well, and you need to understand what effect using these options can have on the overall system and data integrity. In the following sections, we'll explain in detail the changing and exempting of data options that are to be processed in RAR.

4.2.1 Changing Items

You should now have a basic understanding of how RAR is structured: contracts and POBs are always created based on data that is sourced in some other application, which can be either SAP or non-SAP based. Before data can be stored in RAR, it needs to pass staging in the ARL.

So, what happens if some data needs a change or correction? In most cases, you need to fix the data in the source system where it appeared. There are two reasons for that:

- The aim always should be to maintain consistency between systems. In some cases, this requirement is strongly recommended (e.g., integration with sales and distribution where there are reports available to detect these kinds of issues), but in some others, it's not that strong.
- Data correction can be a tedious task. You may not always know what can and should be changed to bypass some problem and why.

However, the reality is that not all data has the same importance (e.g., changing the SSP of an item isn't the same as changing a description field), and it's not always straightforward and possible to correct data. For example, you could import data in RAR from an external CRM system where two systems have different timelines for closing. So, when you notice that some correction needs to be made, it's no longer possible to create a correction in the external system.

All of this led to the ability to manually perform changes in RAIs. To activate such an option, you need to customize which RAI fields are changeable. You can access this functionality by following menu path **Revenue Accounting • Inbound Processing • Revenue Accounting Items • Define Modifiable Fields for Revenue Accounting Items.**

You first need to define which fields aren't available for change. This information can be found in structure FARR_S_RAI2_MI_FIX for processable main items via Transaction SE11. Similarly, you can see which items aren't available for changing for condition items or items in **Raw** status. Now, once you know that the field can be changed, you can access this activity to get to the screen shown in [Figure 4.27](#).

RevAccCl	Rec. Type	Status	Field Name	FieldAttr
SD01	All Record Types	All Statuses	QUANTITY	
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		
	All Record Types	All Statuses		

Figure 4.27 Changeable Fields for RAIs

The first thing you need to define is to which RAI class this change will be used in the **RevAccCl** field: you can define fields that will be changed on order items, fulfillment items, or invoice items. Then, the next step is about record types (**Rec. Type** field) and whether you want to change data in main items, condition items, or both (**All Record Types**). After you define which **Status** the data can be in to be changed (**Raw**, **Processable**, or **All Statuses**), you define which field needs to be changed in the **Field Name** column. In the last column, **FieldAttr**, you need to define whether the field will be visible to the user or only displayed without the ability to change. In this example, the **QUANTITY** field was made changeable during processing in **All Statuses**.

Once setup is complete, when running Transaction FARR_RAI_MON, you'll be able to use the **Change** button, as shown in [Figure 4.28](#).

By selecting **Change** and specifying an item, a separate screen will appear in which you can change data that was previously customized as changeable.

The customization for changing items is completed. However, the recommendation is to avoid making manual changes to RAIs. The best way to keep the system consistent is to correct data in the source system in which it was created in the first place. However, if you feel that manual intervention over RAIs is unavoidable, extra caution should be paid when selecting the data that can be changed.

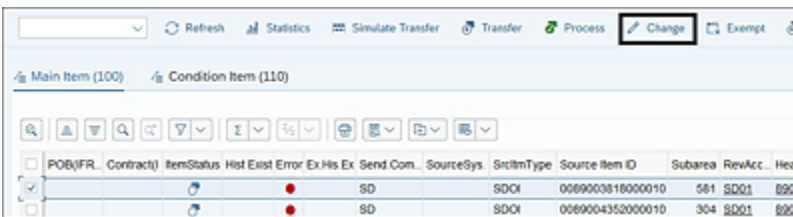


Figure 4.28 Change Option in Transaction FARR_RAI_MON

4.2.2 Exempting Items

Exemption of items is a process when, for some reason, an item needs to be excluded from processing in Transaction FARR_RAI_MON. Usually this refers to items being corrupted in terms of data quality, which can't be fixed in the source system. If not exempted, these items stay in Transaction FARR_RAI_MON (and in the processable table), consuming table space and making item processing items.

In addition, by leaving corrupted items in the processable table, you can keep complete contracts from processing. For example, if you have item 20 from a sales order with an error, the whole order won't be processed unless the item is exempted. For classic inbound processing, the table structure for exempted items looks like [Figure 4.29](#).

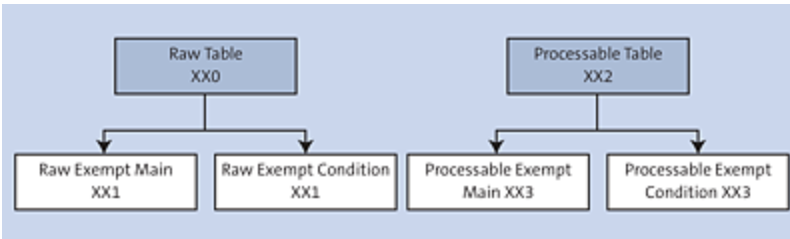


Figure 4.29 Exempted Items Table Structure

Exempted items are kept in separate tables both for main and condition items. In other words, the table structure for exempted RAIs mimics the table structure for raw and processable items. The naming convention is that tables for exempted items have a 1 or 3 after the name of the RAI class. For example, for integration with sales and distribution, the name of the RAI class for order items is SD01, so tables generated for them are as follows:

- Table /1RA/SD012MI for main items
- Table /1RA/SD012C0 for main items

For these RAI classes, the corresponding tables for exempted items are generated:

- Table /1RA/SD013MI for exempted main items
- Table /1RA/SD013C0 for exempted condition items

If you're using raw data in processing and have three RAI classes (order, fulfillment, and invoice items), then in total

you'll have 12 tables related to exempted items: two tables for main and condition items, multiplied by one set for raw and processable, and multiplied by the number of RAI classes.

Tables for exempted items are almost the same as those for processable or raw items. The difference is in the fields giving information about when the exemption occurred and by whom. In the **Exemption History** shown in [Figure 4.30](#), it's clear which user performed the exemption and when. This information is useful for audit requirements.

Exemption History			EXCHIST
Exemption Date (UTC)	15.02.2023	20230215	EXCDATE
Time of Exemption (UTC)	12:36:41	123641	EXCTIME
Exemption Reason	EX	EX	EXCREASON
User for Exemption	SM091B	SM091B	EXCUSNAM

Figure 4.30 Exemption Items Fields

For optimized inbound processing (OIP), the table structure is significantly changed. Instead of dynamic generation of tables and APIs, in OIP, the system works with static, predefined tables. In addition, the process of exemption is replaced by postponing.

Caution

A shift in tables structures is one of the reasons customers need to be extra careful before deciding which option to adopt when either implementing RAR for the first time or deciding about upgrading to a newer version of RAR.

Before the process of exemption can be used, you need to perform certain customizing tasks in RAI management while setting up inbound processing. Follow menu path

FARR_IMG • Inbound Processing • Revenue

Accounting Item Management. When this area for customizing appears, fill in the following fields (see [Figure 4.31](#)):

- **ExempRsn**

First, you need to define the exemption reason with this two-digit code. If there is a specific logic for exempting items, you might have multiple reasons; otherwise, one is enough.

- **Exemption Reason**

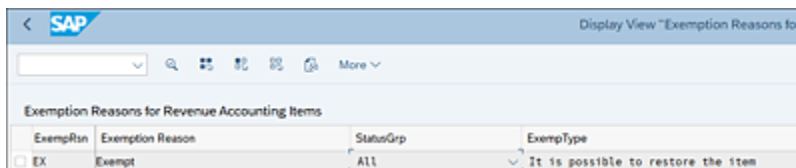
You can enter text explaining the reason for exemption.

- **StatusGrp**

You can define for which items you're creating exempted items: just for processable, just for raw, or for all.

- **ExempType**

Exemption type tells the system whether there will be an option to restore an item that was exempted once. This again depends on the process. If the reason for exemption is data issues, you can either fix the data in the source system or recreate it, in which case, it might be a good option not to restore the item that was exempted. Alternatively, the process might be that data will be corrected while in the exempt table. In that case, defining a restore reason is a good option.

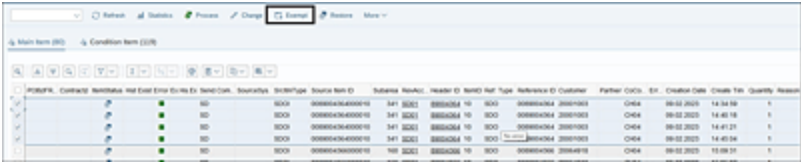


ExempRsn	Exemption Reason	StatusGrp	ExempType
<input type="checkbox"/> EX	Exempt	All	It is possible to restore the item

Figure 4.31 Defining the Exemption Reason

The option to define restoration reasons can be found just below the menu path for exemption (**Define Restoration Reasons for RAI Items**), and setup is the same as for exempt reasons except that **ExempType** doesn't need to be defined.

Once you set an exemption reason, it's ready to be used. Run Transaction FARR_RAI_MON, and enter the header ID that needs to be processed. In the list shown in [Figure 4.32](#), you can see that the **Exempt** button is available.



Item ID	Source Item ID	Subline	Header ID	Ref ID	Ref Type	Reference ID	Customer	Partner	CXCD	EX	Creation Date	Crash Trn	Quantity	Reason
1000	0000000000000000	100	00000000	100	00000000	00000000	00000000	00000000	00000000		09-02-2025	14:30:50	1	
1000	0000000000000000	100	00000000	100	00000000	00000000	00000000	00000000	00000000		09-02-2025	14:40:18	1	
1000	0000000000000000	100	00000000	100	00000000	00000000	00000000	00000000	00000000		09-02-2025	14:41:21	1	
1000	0000000000000000	100	00000000	100	00000000	00000000	00000000	00000000	00000000		09-02-2025	14:41:24	1	
1000	0000000000000000	100	00000000	100	00000000	00000000	00000000	00000000	00000000		09-02-2025	15:00:51	1	

Figure 4.32 Exemption in Transaction FARR_RAI_MON

Click the **Exempt** button to see exemption reasons in the **Exempt Processable Item** popup, as shown in [Figure 4.33](#). Select your exemption reason (**EX Exempt**, in our example), and click the green checkmark.

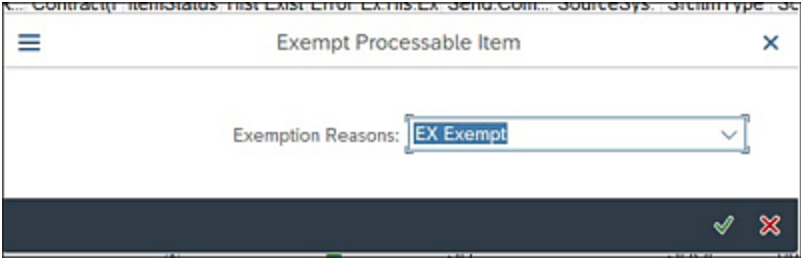


Figure 4.33 Selecting the Exempt Reason

In Transaction FARR_RAI_MON, you can see which items are exempt by choosing **07 Exempted Items** in the **Item Status** dropdown, as shown in [Figure 4.34](#).

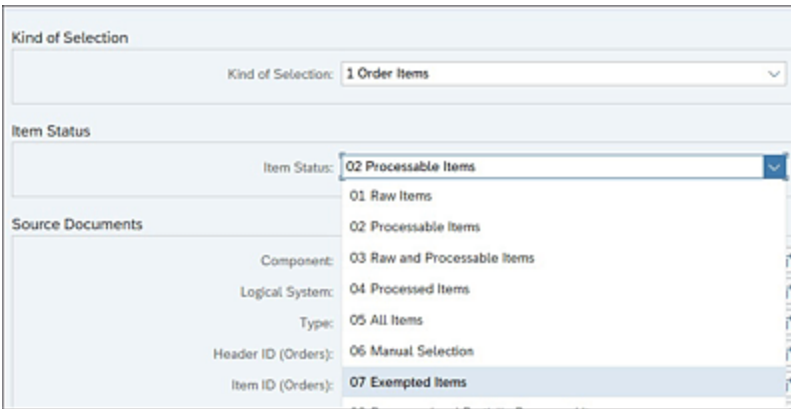


Figure 4.34 Exempted Items in Transaction FARR_RAI_MON

In the list shown in [Figure 4.35](#), you can see all the items being exempted. In addition, the option for restoring an item is available (the **Restore** button), which can be used to transfer items back to the processable table.

POB(FRS) Contract	Item Status	HM	Exist	Error	Ex	HS	Ex	Send	Comp	Source	Source Type	Source Item ID	Subarea	Reason Code	Header ID	Item ID	Exempt Date	Reason	Exempted Date
								SD		SDCR	SDCR	0090004364000010	341	SD01	88054364	10	15.02.2023	EX	SM4915
								SD		SDCR	SDCR	0090004364000010	341	SD01	88054364	10	15.02.2023	EX	SM4915
								SD		SDCR	SDCR	0090004364000010	341	SD01	88054364	10	15.02.2023	EX	SM4915
								SD		SDCR	SDCR	0090004364000010	341	SD01	88054364	10	15.02.2023	EX	SM4915
504027	220402							SD		SDCR	SDCR	0090000820000010	009	SD01	90000820	10	15.02.2023	EX	SM4915
504027	220402							SD		SDCR	SDCR	0090000820000010	009	SD01	90000820	10	15.02.2023	EX	SM4915
504027	220402							SD		SDCR	SDCR	0090000820000010	009	SD01	90000820	10	15.02.2023	EX	SM4915
504012	220412							SD		SDCR	SDCR	0090000820000010	503	SD01	90000820	10	15.02.2023	EX	SM4915

Figure 4.35 Exempted Items List in Transaction FARR_RAI_MON

4.3 Extending Transaction FARR_RAI_MON

The RAI monitor is the main tool used for data transfer between the ARL and RAR database tables. The standard program contains most of the options to search and filter items that need to be processed or analyzed further. However, you can extend some of the standard options without the need for additional programming.

In the upper part of the screen in Transaction FARR_RAI_MON, you'll find the option to further enhance selection parameters (**Further Selections**), as shown in [Figure 4.36](#).

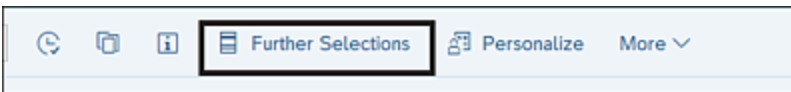


Figure 4.36 Further Selections in Transaction FARR_RAI_MON

By selecting this option, you can see all the fields that are part of structures assigned to standard RAI structures, as shown in [Figure 4.37](#). All fields here can be used as selection parameters when you want to see only data that is relevant for processing. On the left-hand side are all the fields in the structure for the main items. In our example, we've selected **Channel partner**, as you can see under the **Dynamic selections** screen on the right-hand side.

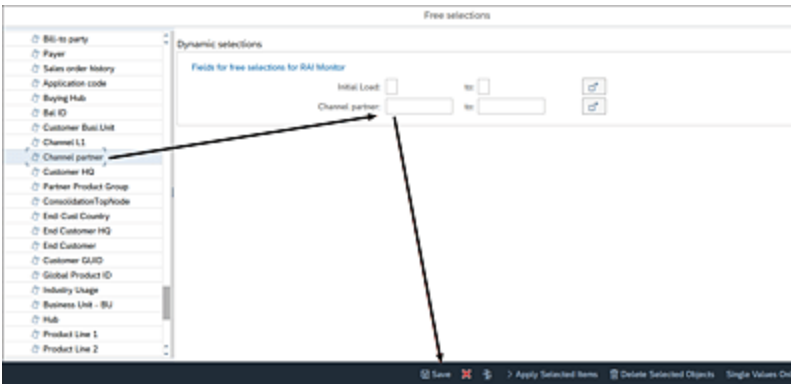


Figure 4.37 Data Selection for Further Options

Once additional data has been selected, click the **Save** button, and the **Further Selections Exist** checkbox will be selected in the **Further Attributes** section (refer to [Figure 4.3](#)).

Use Case for Additional Selection

It's not uncommon that one RAR contract is created as a combination of different documents that also can have different sources. For example, one RAR contract is created as a combination of sales contract, sales order, and service order. Here, to make the necessary links between them, we often use separate, custom fields. So, to ensure that in one shot all documents are being taken into consideration, a good idea is to extend the search in Transaction FARR_RAI_MON with that special custom field.

4.4 Error Resolution While Creating Revenue Accounting Items

As mentioned in previous chapters, data quality is one of the prerequisites for RAR to perform. When thinking about how to ensure that data is able to be processed by RAR, it's worth mentioning that RAR is much more integrated with external systems than plain finance. In other words, the old paradigm that operations and finance need to integrate at the moment of account determination isn't the case with RAR.

Due to the nature of IFRS 15, RAR represents processes in sales operations that are copied in finance. So that is why implementation of RAR requires a whole different level of understanding for finance consultants about SAP Billing and Revenue Innovation Management and sales and distribution. The same applies to sales and distribution consultants: it's not enough just to have a high-level understanding of accounting and controlling (if any). Now, people involved in the sales process need to understand from the very beginning the financial implications of decisions made very early in the process.

All this being said, if these recommendations are followed, the data coming to RAR as RAIs should be of sufficient quality. However, it's safe to assume that even with the most rigid rules, some errors will appear in the RAI monitor requiring both *proactive* and *reactive* measures. We'll discuss both proactive and reactive measures next, and then explain a few error handling techniques.

4.4.1 Proactive and Reactive Measures

Proactive measures are checks put in place so the item will be verified before it comes to RAR. [Figure 4.38](#) represents one way of preventing errors from happening.

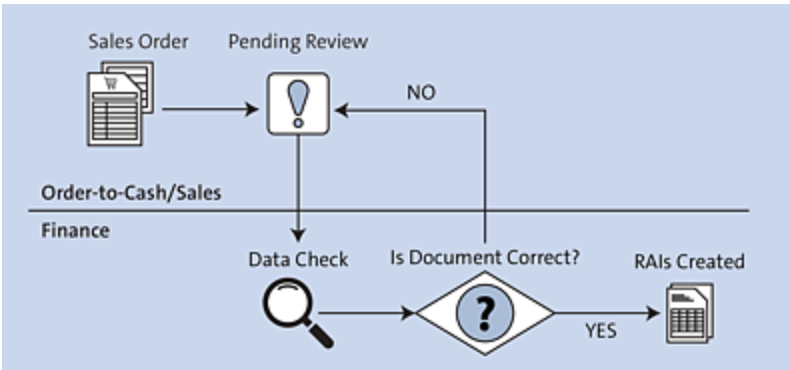


Figure 4.38 Statuses While Processing RAIs

The process starts in the sales department by creating the sales document. This document can be a sales order, contract, or even service order. Once data has been saved, it will have an assigned status that review is needed (**Pending Review** status), which is the finance team's responsibility. The finance team will check data that is needed for RAI creation such as the following:

- SSP existence/correctness
- Contract duration
- Item type correctness
- Some custom rules that companies might have implemented in their processes

As a result of the finance team check, the document may be sent back to the sales team to complete it/correct it if errors are found. The other result may be that the finance team

determines that document is in proper quality and decides to set the status to **Approved**, which would trigger creation of RAIs.

Of course, this process can be extended in many ways. For example, a company might opt for making some kind of workflow: there might be more statuses representing readiness for processing (document might be initially entered and not yet be processed until created), there might be routing between different departments based on statuses, and so on. But the main point remains—let only correct data come to RAR.

The main benefit of such a process is that the data coming as an RAI is the quality needed for successful processing and contract creation. This part can't be emphasized enough—as mentioned, the logic on which revenue recognition works according to IFRS 15 is completely different than before. Here, revenue recognition happens based on triggers, and sometimes these triggers are automatic and something you have no influence on (e.g., time-based POBs). So, error fixing is much harder or even impossible if an issue is noticed after the error recognition process has already kicked in. So, the main goal of the setting statuses process is preventing such issues from happening.

There are some drawbacks to the process. In [Figure 4.38](#), you can see that the finance team is expected to be familiar with applications that aren't strictly from the finance domain, such as checking the sales document and change log, document flow, and so on. The second point isn't a limitation as such, but needs consideration: Such a process

is very suitable for organizations that have a revenue recognition process as part of shared service processes. In that case, it would be enough to train one group of people who could perform such controls and checks. Usually, IFRS 15 is a global policy for reporting, so making the process organized in the form of a shared service center would make sense.

Now, let’s discuss reactive measures. Not all errors can be caught before they reach the ARL: Even in environments where most of errors would be caught before they reach the ARL, some will come through. When they come to the table for transfer or processing (depending on whether raw items are included or not), those items will appear in Transaction FARR_RAI_MON for processing. When you open the transaction, all items with an error will have a red dot in the **Error** column, as show in [Figure 4.39](#).

To see what the error type is, click an item and see what message is. As shown in [Figure 4.40](#), the system will display all information needed for you to recognize the root cause and correct the problem.

Status	Has Error	Ex. Hls	Ex. Send Com.	Source/Obj.	DocType	Source Item ID	Subarea	Service	Header ID	ItemID	Ref. Type	Reference ID	Customer	Partner	CoCo	Er.	Creation Date	Create Tim
	●			SO	SOO	888760259000010	526	SOO	888760259	10	SOO	888760259	1200000344	AL01	3	18.02.2023	22:21:38	
	●			SO	SOO	888760270000010	436	SOO	888760270	20	SOO	888760270	1000005296	AL02	2	22.02.2023	10:14:57	
	●			SO	SOO	888760270000020	436	SOO	888760270	20	SOO	888760270	1000005296	AL02	2	22.02.2023	07:27:54	
	●			SO	SOO	888760290000010	617	SOO	888760290	10	SOO	888760290	1000000801	AL02	2	25.02.2023	10:11:36	
	●			SO	SOO	888760290000010	938	SOO	888760290	10	SOO	888760290	1200000803	AL03	3	22.02.2023	09:51:11	
	●			SO	SOO	888760290000020	938	SOO	888760290	20	SOO	888760290	1200000803	AL03	3	22.02.2023	09:51:11	
	●			SO	SOO	888760297000010	271	SOO	888760297	10	SOO	888760297	1200000803	AL03	3	22.02.2023	09:49:36	
	●			SO	SOO	888760297000020	271	SOO	888760297	20	SOO	888760297	1200000803	AL03	3	22.02.2023	09:49:36	

Figure 4.39 Errors in Transaction FARR_RAI_MON

Ty.	Message Text	Ltxt
●	Start of processing of order item SDO/800760259000010 for accounting principle IFRS	
●	Inflight check: Error C01/Contract 102008 found before save to database	
●	C01: Allocation error found for contract 102008	
●	Start date of time-based performance obligation \$0000000000000001 is empty	
●	Either a duration or an end date must be specified for perf. obligat. \$0000000000000001	

Figure 4.40 Details of Error in Transaction FARR_RAI_MON

You can see what errors were thrown before data could be saved to the database. The first error you can see is that SAP has thrown an inflight check error. These checks are proactive checks that SAP makes before data is saved in tables in order not to cause inconsistencies. In other words, inflight check errors will appear so that data inconsistency doesn't appear at all. Data validation checks will appear before data is written permanently to database. Both of these checks will be covered in detail next.

In total, there are 26 types of errors that can appear, and all of them are a threat to data integrity. All errors are split into two categories:

- C errors represent inflight checks.
- E errors represent data consistency checks that are performed before data is written in tables.

C and E Errors

The list of errors that can be issued isn't final and can be updated by SAP to include more of them. This means that more issues are being discovered and classified, and this issue list can grow with time.

4.4.2 Inflight Errors

Inflight errors are introduced as standard functionality in RAR to give more information about the root cause of

erroneous items that might appear as a result of processing. The inflight check functionality is shown in [Figure 4.41](#).

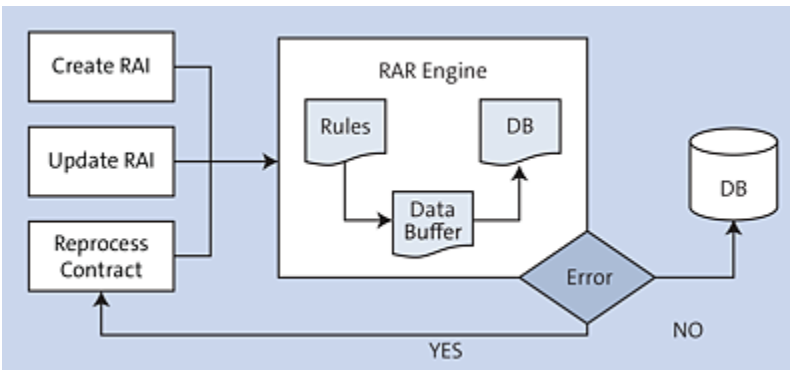


Figure 4.41 Inflight Error Logic

As mentioned, this error check sits in the RAR engine itself and is triggered each time there is some change coming from any user activity, which is either creating new RAIs, updating existing data, or reprocessing contracts, which is an activity that runs directly on a contract when it's already created. When such an activity is performed, data is kept in a buffer and not saved to the database yet, and all checks are performed over that buffer. The database commit is executed only after all errors are cleared.

Not all activities will trigger inflight error checks, however. If invoice items are processed that don't lead to price reallocation or running ABC programs, these activities won't trigger inflight error checks. This was done on purpose because the idea of error checking is to avoid data inconsistency without stopping business-related activities such as the month-end closing process.

Inflight error checks can be further extended. SAP created a BAdI called Extended Checks before Saving to Database

where you can opt to implement your own logic by using subclass CL_FARR_DATA_EXTENDED_CHECK.

Once an inflight error occurs, in most situations, the best option is to fix the problem where it occurred (in the source system) and trigger recreation of RAIs again. Another option is to fix the data in the RAI table if proper customization is made. However, this option should be used only as a last resort because it would mean there is a difference between data in the source system and RAR.

But let's look at our example error shown previously in [Figure 4.40](#). In the first line, the system reported its C01 error for a contract that already exists. So, the RAI is modifying the existing contract, which is triggering reallocation.

In short, the C01 error means that if a change coming from an RAI sent by the source system is saved in the database, it will create an inconsistency in the balance of the allocation effect. For all POBs in a contract, the total allocation effect must be zero. For example, a calculation that would throw an error as described previously can be represented as shown in [Table 4.1](#).

POB_ID	Transactional Price	Allocated Price	Allocation Effect
POB A	1,000.00	1,200.00	+200
POB B	2,000.00	1,900.00	-100
Total	3,000.00	3,100.00	+100

Table 4.1 Allocation Effect

In our case, we have a new RAI that can't be used to create a POB, but if the transaction price were created, it would create an inconsistency with the allocated price.

In other words, looking simply at the C01 error, it won't always give you a clear place where the issue occurred and what to do to solve it. It will show more about what kind of problem could be caused if the error isn't solved.

In the third line shown previously in [Figure 4.40](#), you can see all the necessary information about where and why the error actually occurred. The system is saying that the start date of item \$000000000001 is missing. First, now you're clear what caused an error: the POB affected is actually a time-based POB missing its start date, which is preventing the POB from being created. The second thing you can see from the message is that the system is trying to create a new POB. So, if you're integrating with sales and distribution in this case, you can look at a new sales document item, which is newly added in the sales document where the start/end date is missing and resolve it there. Once that is done, a new POB with a later timestamp is created that you can process, and this error would be resolved.

4.4.3 Data Validation Checks

Unlike the inflight errors functionality, data validation checks are issued after the database update is executed to validate data once the data reaches tables in RAR. There are currently 21 error categories that can be issued once data is validated, and, unlike inflight errors, data validation checks start with prefix E.

The purpose of using data validation checks is different from inflight errors. We mentioned that inflight error checks don't prevent the business part of contract management, even if errors are detected. So even if some C errors are raised, contracts will still be posted and revenue will be recognized if any. The purpose of E errors, on the other hand, is to protect you from submitting potentially erroneous results to revenue reporting. Therefore, contracts with E errors won't be processed further.

The root cause of errors coming from data validation checks can vary from errors coming from ARL data processing, wrong BRFPplus rules, or even mistakes in different BAdI implementations. However, there are situations where data validation checks can't be of help. These are different errors caused by user actions (e.g., wrong revenue suspension), no end-to-end reconciliation (between sender component and contracts), compounding POBs issues, and so on.

Data validation checks are performed in two steps:

1. **Consistency check**

This activity is performed by running Transaction FARR_CONTR_CHECK, where the system runs all 21 checks and writes results to table FARR_D_CONS.

2. **Consistency monitor**

This activity reads entries in table FARR_D_CONS and displays results to the user.

When you execute Transaction FARR_CONTR_CHECK, the screen shown in [Figure 4.42](#) is displayed.

The screenshot shows a configuration window for Transaction FARR_CONTR_CHECK. It is organized into three main sections:

- Selection Data:** Contains 'Accounting Principle' (input field with '1'), 'Company Code' (input field), 'Excluding Completed Contracts' (checkbox), and two 'to:' fields (input fields).
- Technical Parameters:** Contains 'Block Size For Mass Selection' (input field with '1,000'), 'Dialog Mode' (checkbox), and 'Synchronous Call' (checkbox).
- Settings for Application Log:** Contains 'External ID' (input field) and 'Problem class' (dropdown menu showing '2 Important').

Figure 4.42 Transaction FARR_CONTR_CHECK

Here, you need to specify **Accounting Principle**, **Company Code**, **Dialog Mode** (i.e., running the application in foreground or background), and **Problem class** (i.e., level of log details). In addition, it's a good idea to schedule and run this program in regular intervals. In those cases, you should pay attention to which company code is included in the run because only that company will be refreshed.

Once the program is run by clicking the **Execute** icon, data is entered in table FARR_D_CONS and read with Transaction FARR_CONTR_MON, as shown in [Figure 4.43](#).

Besides the standard fields of **Accounting Principle**, **Company Code**, and **Revenue Accounting Contract** (i.e., range of contracts) that are optional, the most important setting is in the **Processing** section. If you select **Read data from error table**, you're reading entries previously created by Transaction FARR_CONTR_CHECK. If you select **Read data online**, the system reads the contract management tables and executes checks online. This option makes running Transaction FARR_CONTR_CHECK obsolete. However, it should be taken into consideration that

executing checks online takes significant time and can cause memory dumps even in the case of a moderate number of contracts. The recommended method is using Transaction FARR_CONTR_CHECK to fill the error table with contracts that have errors, and then run the monitor to display them. After the errors are corrected, the results are written to the database.

The screenshot shows a web-based form for the FARR_CONTR_MON transaction. It is divided into three main sections:

- Contract:** Contains input fields for Accounting Principle, Company Code, and Revenue Accounting Contract. To the right, there are three 'to:' labels with corresponding input fields and small square icons.
- Processing:** Contains three radio button options: 'Read data online' (selected), 'Read data from error table', and 'Save result to error table'.
- Technical Parameter:** Contains a text input field for 'Max. batch size of Contracts' with the value '1,000' and a label 'Max. no. of POBs:' below it.

Figure 4.43 Transaction FARR_CONTR_MON

When you're done, click the **Execute** button. The result of Transaction FARR_CONTR_MON is shown in [Figure 4.44](#) and can be split into three sections.

Exception	Contn	POB	CoCode	Acc Princ	Post Lab	Post Level	Post Lab Lead/Line	U-C	Cstat	Leading	rgnr	Lnt	Exc	Assoc	BOM	Root	Comp	FullType	Event	Type	vs. R	
000004	101176	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000004	101177	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000004	101178	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000004	101184	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000004	101183	CH04	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				X											NA	X
000004	101184	CH04	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				X											NA	X
000075	101195	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000076	101196	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000077	101187	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000077	101198	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000078	101199	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D											NA	
000079	102000	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D											NA	X
000080	102001	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000080	102002	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										X	
000081	102003	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										NA	X
000082	102004	0804	IFIG		Contract Liability/Contract Asset	Post at Performance Obligation Level				D	E										NA	X

Figure 4.44 Error Monitor Results

In the first part, you see information about the POB that was checked and the liability posting level. In addition, on the right side, you can see fulfillment information with compounding information, if any. When you scroll to the right, you'll see which checks were performed successfully and which have errors, as shown in [Figure 4.45](#).

Exception Code	POB	Start/End	Liability Posting Level	Start/End	Created on	Created By	E01	E02	E03	E04	E05	E06	E07	E08	E09	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E20	E21
	200075	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	200076	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	200077	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	200078	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	200079	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007A	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007B	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007C	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007D	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007E	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007F	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007G	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007H	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007I	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007J	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	20007K	0.00	0.00	0.00	20.02.2020	17.02.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 4.45 Errors Found While Processing Consistency Check

Note

You can find PDF versions of [Figure 4.44](#) and [Figure 4.45](#) available for download at sap-press.com/5700 under the **Product supplements** section.

In this case, POB number **200077** has failed data validation check **E11**. If you read the documentation about data validation errors, you'll find that E11 represents errors related to calculation of contract liability if the option of calculation on the contract but posting on the POB level is used. So, to determine the reason for the error, you need to compare values in the columns for planned liability versus posted liability. It's possible that running program B was skipped or there's some other problem to be investigated further. But depending on the impact on reporting, the decision might simply be to exclude the contract from processing or to fix a problem in the current month.

4.4.4 Resolving Errors without Transaction FARR_RAI_MON

In certain cases, organizations have thousands of RAIs being created on a daily or weekly basis, and often in such situations, Transaction FARR_RAI_MON can't be used as the main tool for error reporting/issues resolution. So, what should you do in such cases?

The first step is to check how many items have an error in the processable table. Run Transaction SE16N over the processable order RAI table (table /IRA/0SD012MI), for example (most problems will be with order items). Here, you need to filter the **Error** column, as shown in [Figure 4.46](#).

In the popup that appears after you click **Error**, select value **3** because it represents processing data moving from the **Processable** to **Processed** status. Options **1** and **2** are related to checks performed in which the item is either saved in the **Raw** status or moved to the **Processable** status, respectively. So, these statuses will be used only when the **Raw** status is set. The system will provide the total number of data items in error. You can further limit the date to get a list that only shows items that failed while processing at a specific time.

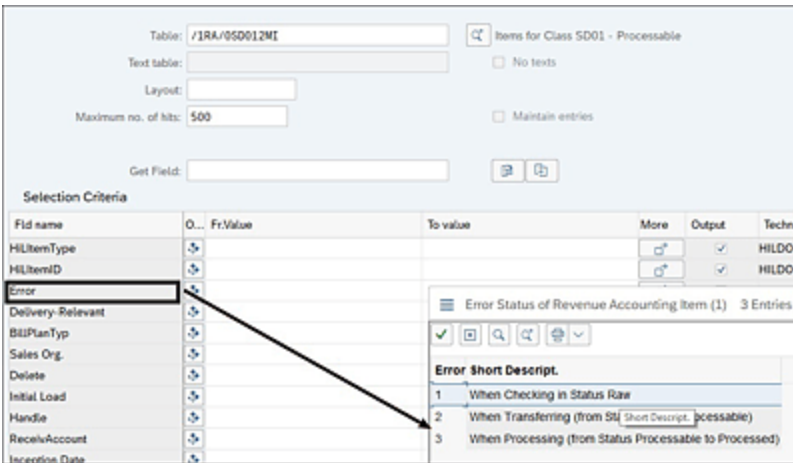


Figure 4.46 Errors in Processable Table

Now, you have a list of items that are in error, which caused them not to be processed. The next step is to look at the log. Each processing of RAIs is saved in the log with a different level of information, and you can check that information further. Run Transaction SLG1, as shown in [Figure 4.47](#).

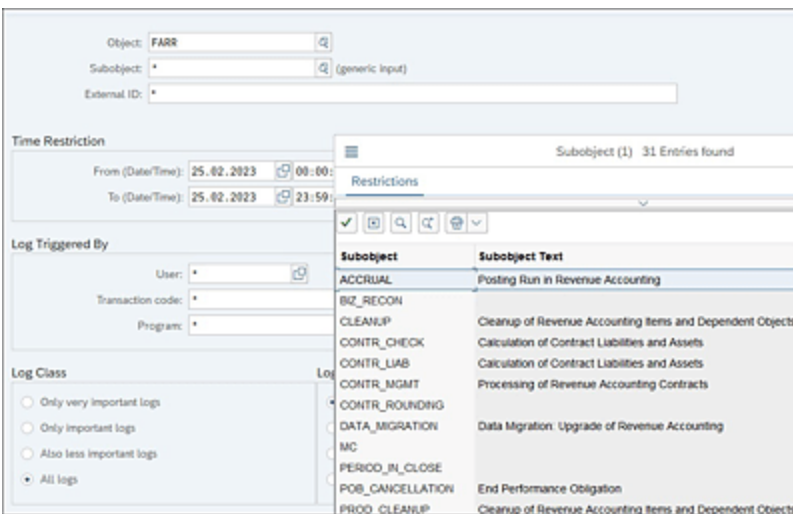


Figure 4.47 Transaction SLG1

In the **Object** field, you'll enter "FARR", which causes a list of possible entries to appear in the **Subject** column. In this case, it can be either **CONTR_MGMT** if you're searching

for errors during processing of contracts, or **RAI_PROCESS** (not shown) if you're searching for errors during RAI processing. Once you execute the transaction, you'll arrive at the screen shown in [Figure 4.48](#).

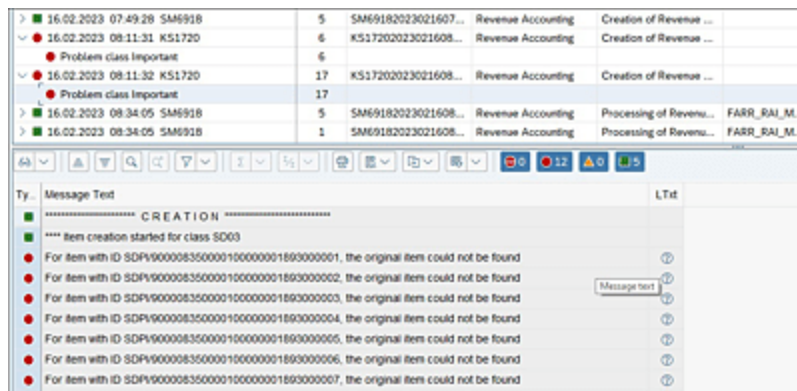


Figure 4.48 List of Errors in Transaction SLG1

You can see that the information in Transaction SLG1 is very similar to the information in Transaction FARR_RAI_MON.

The conclusion on error management related to RAIs is that unverified data coming to the ARL should be avoided as much as possible in the first place. Different proactive error management approaches can be taken: either by implementing logic in applications that create orders or by implementing layers that will be used to clean data before it reaches the ARL.

Once data is in the ARL, standard mechanisms are triggered to protect data integrity in RAR itself. You can extend these checks with available methods, or you can create a custom error management module before data is passed further.

All of this emphasizes the key message: data quality is crucial for RAR to fulfill its main purpose as an engine to calculate IFRS 15 revenue.

4.5 Business Rules Framework Plus

Business Rules Framework plus (BRFplus) is an SAP tool that is used to reduce complexity when it comes to defining business rules. Rules that are generated by BRFplus can be easily incorporated in different applications used in SAP.

Think, for example, about validations or substitutions in which you define rules once that you can later reuse across different applications. Compare that with the old methods where you needed to define rules based on an application area, and the number of areas determined the number of rules.

BRFplus isn't related to SAP HANA nor is it that new, but with newer releases of SAP S/4HANA, it's getting more and more attention. The overall BRFplus architecture is shown in [Figure 4.49](#).

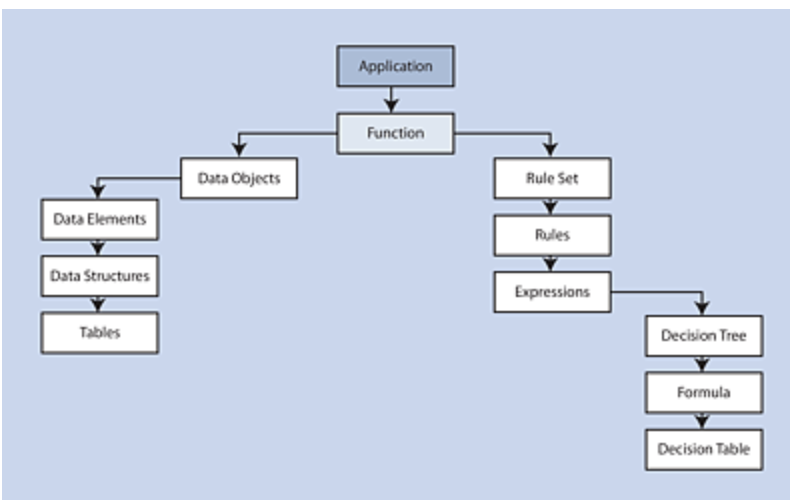


Figure 4.49 BRFplus Architecture

Let's walk through each part of the BRFplus architecture:

- **Application**

The application is a simple container in BRFplus for other BRFplus objects. It's classified highest in the hierarchy, and you can create as many applications as needed.

There are three types of applications:

- System applications
- Master data applications
- Customizing applications

The main difference is in transports: system and customizing applications are always transportable, whereas master data applications are created locally.

- **Function**

This is the rule interface and behaves as a link between application code and BRFplus code. Function has a context and a result. `Context` is an import parameter when an application is being called, and `Result` is the return of its execution.

- **Rule set**

Rule set is nothing but a collection of rules that will be executed for a specific business case. It's an entry point for tasks to be executed. The rule object is a technical representation of a simple business rule to be applied on a specific object. Rules have to be assigned to the rule set; they can't be executed as standalone rules.

- **Expressions**

Expressions make up the computational power of BRFplus where each contains a logical formula that needs to be executed. There is a predefined number of expression

types, and BRFplus is enhanced each time with a new set of expression types.

- **Decision tables**

Decision tables belong to a catalog within each BRFplus application. It's a crucial part because it holds all rules based on which the derivations of target values are determined.

With this architecture in mind, we'll explore BRFplus in the context of RAR in the following sections. We'll start with the available applications and structures, and then explain how to set up BRFplus and relevant extensions.

4.5.1 Applications and Structures for RAR Integration

SAP delivers different BRFplus template applications that are integrated with RAR. Applications delivered are to be used for separate functions:

- Integration with sender components
- Account determination
- Determination of POB status

To access delivered applications, run Transaction BRFPLUS. You'll arrive at the screen shown in [Figure 4.50](#).

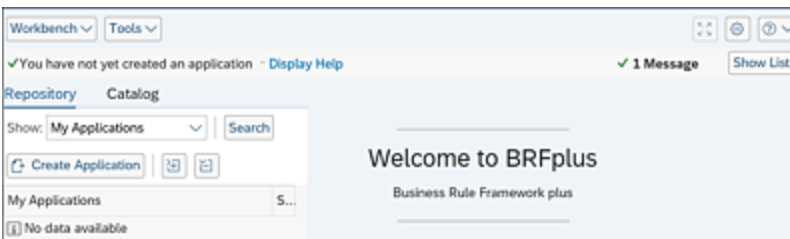


Figure 4.50 Transaction BRFPLUS

To find the application that you need, you can perform a search on the left part of the screen. Here, you can see all the information about the application selected. Click the **Search** button in the **Repository** section.

BRFplus Expert Mode

It's a good idea when working on development of BRFplus application to be in expert mode. By default, the transaction is running in **Simple** mode, which can be changed. In the top right corner, select the **Personalize** button (not shown), and a further list of options will appear. On the left part of the window, a selection named **User** mode will appear. Select **Expert** from the dropdown list. Once this is done, you'll see technical names instead of descriptions and a few more features that are helpful when it comes to the development of applications.

The **Search** screen shown in [Figure 4.51](#) will appear. To find the RAR applications required, enter "FARR*" in the **Application** area, which will display all standard applications for RAR. Click **Ok**.

Figure 4.51 BRFplus Application Selection

The first step in the process is to select which applications are needed. To work with the ARL, the first needed application is one used for processing data and creating RAR contracts and POBs. Here, you'll select applications based on the type of integration used:

- FARR_AP_CA_PROCESS_TEMPLATE for integration with SAP Billing and Revenue Innovation Management
- FARR_AP_SD_PROCESS_TEMPLATE for integration with sales and distribution
- FARR_AP_CRM_PROCESS_TEMPLATE for integration with SAP CRM
- FARR_AP_PROCESS_TEMPLATE for integration with external non-SAP components

Once you select an application that is proper for integration in a specific case, it's highly recommended to copy it to the new application in a specific namespace. In addition, be sure to also select the **Contained Objects** option when copying the application. This option will move all

corresponding objects to the target application too, so no object will be missed.

Important Note

Special attention should be paid to changes and transporting of changes in BRFplus. When creating an application, you need to assign a transport request to which changes will be saved. All BRFplus applications for RAR are of type system, which means they can't be maintained in production directly, but they can be maintained on the quality assurance system. Situations where applications are maintained in separate systems should be avoided. Because each application has its own unique ID, maintenance in different systems would cause transport to become impossible. The aim should be to maintain applications in one system and move changes by a regular transport route.

Once you've copied the necessary application, you can maintain it, which means you're maintaining the decision tables. Decision tables contain a set of rules or values that represent the configuration according to which decisions are made.

The next step is to configure these rules in the decision table. On the left-hand side, under the **Expression** menu, you can find all the decision tables that need maintenance. All columns available can be split into two sections: (1) input columns that represent input parameters for decisions and (2) output columns that represent results. Input columns are marked with gray, and output columns are marked in green.

First, you need to maintain the decision tables needed for integration with sender components. There is an order in which decision tables functions are executed, and their descriptions follow:

- **DT_PROCESS_COMPOUND**
Applied for no bill of materials (BOM) items and contains a set of rules for how compounding will be performed.
- **DT_PROCESS_BOM**
Contains members of BOM to be managed as distinct or nondistinct POBs.
- **DT_PROCESS_POB**
Main table that contains rules for how POB is determined, including type, fulfillment type, deferral method, start and end dates, and so on.
- **DT_PROCESS_POB_ADD**
Contains links to implicit POBs.
- **DT_PROCESS_SSP**
Used for SSP determination. SSPs can be determined in two ways, either sent by sender component or determined in BRFplus. If needed, the SSP can be entered in this decision table together with tolerance limits and calculation type.
- **DT_PROCESS_DEFERAL**
Used for special condition types such as right of return (ROR) to determine deferrals.
- **DT_PROCESS_HEADER**
Used to determine contract header attributes such as description and contract category.

Processing SSPs in BRFplus

As stated, a standard decision table in BRFplus is table DT_PROCESS_SSP, which is used for determining the SSP for the POB. However, there are limitations on how this feature should be applied. A huge amount of data in BRFplus can seriously hamper system performance, and there is a limit of around 10,000 entries, which should be respected. Regarding SSPs, there are other features, such as validity period, which often are needed by the customer (available in condition records maintenance) and which could easily reach that limit even for a moderate number of materials. So, using BRFplus as a main source for SSP maintenance should be used only as a last resort and only when following the limitations such an approach can impose.

Maintenance of decision tables is simple and intuitive: you need to enter criteria based on which target values are being determined. For POB determination, for example, you'll enter material and/or document type and line-item category as input parameters, and you'll enter the POB type being determined as the target.

Once decision tables for POB determination are maintained, the next application that requires maintenance is FARR_ACC_DETERMINE_TEMPLATE. The approach is similar, and this template application needs to be copied to the customer namespace where changes will be performed.

The following decision tables are available for setup:

- **FARR_ACCT_DETERMINE_DT_CORR**
Used for revenue adjustment for allocated revenue with posting category RV in table FARR_D_POSTING when running program A.
- **FARR_ACCT_DETERMINE_DT_CORR_A**
Used the same way, just for linked POBs.
- **FARR_ACCT_DETERMINE_DT_CT_AST**
Used for postings of contract assets with category CA in table FARR_D_POSTING while running program B.
- **FARR_ACCT_DETERMINE_DT_CT_LIB**
Used for postings of contract liability with category CL in table FARR_D_POSTING while running program B.
- **FARR_ACCT_DETERMINE_DT_DCOGS**
Used for posting cost deferrals in scenarios where cost recognition is used with category CJ in table FARR_D_POSTING.
- **FARR_ACCT_DETERMINE_DF_REV**
Used for posting deferred revenue with posting category DR in table FARR_D_POSTING.
- **FARR_ACCT_DETERMINE_DT_RADJ**
Used for posting receivable adjustments with category RA in table FARR_D_POSTING after running program A.
- **FARR_ACCT_DETERMINE_DT_RC_CST**
Used for recognized costs in a cost deferral scenario with category CO in table FARR_D_POSTING.
- **FARR_ACCT_DETERMINE_DT_RC_CST**
Used for recognized revenue with category RA in table FARR_D_POSTING.

- **FARR_ACCT_DETERMINE_DT_ROR**
Used for ROR posting. It's represented with a separate condition type and not a posting category.
- **FARR_ACCT_DETERMINE_DT_UB_REC**
Used for unbilled receivables with category UR in table FARR_D_POSTING.

Besides these, if optimized contract management (OCM) is used, you can also customize table FARR_ACCT_DETERMINE_DT_ASST_IM for impairment postings by posting category AI in table FARR_D_POSTING. Impairment is the result of termination: it represents a balance of either the contract asset or contract liability that is being moved later to profit and loss (P&L).

To maintain BRFplus tables, you click the + button to add new entries. The same approach is needed for all decision tables: input parameters are marked in gray, and target (or export) parameters are marked in green. You need to select from the list of possible entries, which will depend on the data element assigned to the table. Once entries are added, the table needs to be saved and later activated.

Not all tables require maintenance for all business scenarios. For example, if only one accounting principle is used with the contract assets/contract liabilities (CA/CL) calculation (see [Chapter 5, Section 5.1.3](#)), then maintenance of unbilled receivables and deferred revenue tables isn't necessary. Similarly, if there is no cost recognition process, then the table for cost deferral can be left empty.

Once all tables have been maintained, rules are configured (see [Chapter 5, Section 5.2.2](#) for more information on creating rules for POBs). To complete integration with RAR, application assignments need to be performed, which we'll discuss next.

4.5.2 BRFplus Setup in RAR

Once tables have been maintained, you need to make assignments to RAR by using Transaction FARR_IMG and going to **Revenue Accounting • Inbound Processing • Revenue Accounting Item Management • Assign BRFplus Applications to Revenue Accounting Item Classes**.

Once the transaction is run, you need to assign the application that was customized in the previous step to appropriate RAI classes. This step is only necessary for order item classes.

The next step is assignment of the other two applications: one for POBs and one for postings/account determination. The application for POB statuses is optional, but for postings and account determination, it's mandatory. However, these steps can be found in [Chapter 5](#). You execute the process by accessing Transaction FARR_IMG and going to **Revenue Accounting Contracts • Assign BRF+ Application to Revenue Accounting Processes**. As shown in [Figure 4.52](#), you need to select which application is needed for which process. There are two processes available that can be selected depending on your needs: **AD Account Determination** and **PS Performance Obligation Status**.

BRF+ Rule Configuration	
BRF+ Related Process	BRF Application
<input type="checkbox"/> AD Account Determination	▼ FARR_ACC_DETERMINE_PG1
<input type="checkbox"/> PS Performance Obligation Status	▼ FARR_POB_STATUS_PG1

Figure 4.52 Posting Application Assignment

Now, to change some entries in decision tables, the most common method is via Microsoft Excel. BRFplus comes with an embedded Microsoft Excel download/upload functionality so you can easily download all entries from the decision table, quickly make updates, and then upload again. The template fits the structure of the decision table, making it simple to share among different people and upload back to BRFplus without any additional adjustments. Every decision table in the application has this option available.

You also have additional options that make maintenance of BRFplus decision tables easier. In the menu at the top of the screen, you have the option of defining which BRFplus decision table can be maintained using the simplified BRFplus user interface (UI).

The next step is to assign specific decision tables to be maintained by the simplified UI. Navigate to **ID for Decision Table UI** in the dialog structure, as shown in [Figure 4.53](#). Once you select this option, there will be a link between tables used for customizing and the BRFplus application they belong to.

Dialog Structure		ID for Decision Table UI		
▼ ID for Decision Table UI		Decision Table ID Used for Customizing	BRF Application	Decision Table Name
▶ Related Decision Table ID		<input type="checkbox"/> PROCESS_BOM	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_BOM
		<input type="checkbox"/> PROCESS_COMPOUND	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_COMPOUND
		<input type="checkbox"/> PROCESS_DEFERRAL	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_DEFERRAL
		<input type="checkbox"/> PROCESS_HEADER	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_HEADER
		<input type="checkbox"/> PROCESS_POB	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_POB
		<input type="checkbox"/> PROCESS_POB_ADD	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_POB_ADD
		<input type="checkbox"/> PROCESS_SSP	FARR_AP_SQ_PROCESS_PG1	DT_PROCESS_SSP

Figure 4.53 Assignment of BRFplus Applications to the Simplified GUI

Once you select the decision table in [Figure 4.53](#), it can be maintained with the simplified GUI, as shown in [Figure 4.54](#). The main (right-hand) part of the simplified GUI shows which application is selected and which decision tables are created in the application. Once you select the needed table decision table, maintenance will begin.

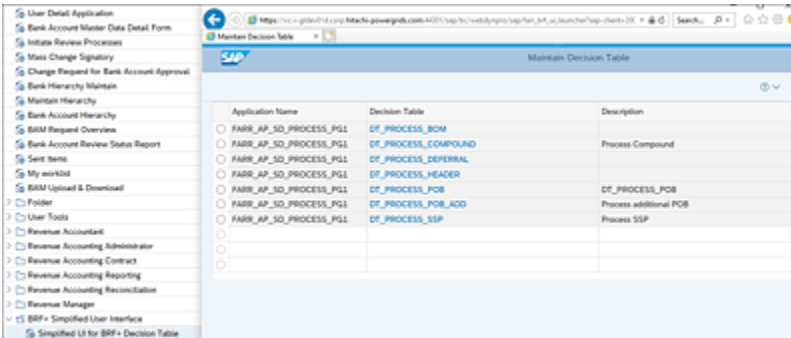


Figure 4.54 Simplified GUI

The simplified UI option enables you to quickly change data required for POB determination of, for example, the SSP. However, you should be very careful regarding maintenance. As previously mentioned, maintenance of RAR BRFplus data is an activity that should be taken with extra caution. Settings made here can have multiple impacts, and the recommendation is to always respect the transport route for changing anything in decision tables.

4.6 Creating Custom Revenue Accounting Items

We've already discussed the creation of RAIs in previous sections. The RAIs that are created all adhere to the configurations and the rules defined. There are several instances when you need to validate, manipulate, delete, enhance, or perform a lot of other things on RAIs. The custom fields that you add need to be filled or validated and that can be achieved here. So, customizing the RAIs and controlling the further processing of RAIs is the main discussion of this section.

The RAIs that are created in RAR are dependent on the information that is sent from the source system. The source systems are called the sender component, as discussed in [Chapter 2, Section 2.3.1](#). The data that is sent is converted to RAIs. It's very important for the data to be right to create correct RAIs.

With data, there are always issues, so keep the following in mind:

- The data can't be assumed to be right always, so you need to validate it.
- The data can't always be in the expected format, so you need to check the data format.
- The number of characters that is expected for a field may be exceeded, so you need to check that.

- There will be some fields that need to be populated based on the values of the other fields within the same structure.
- Data needs to be enriched and prepopulated with some default values.

A lot of data handling needs to be done here before the RAIs are created and also during the various statuses of RAIs. In the following sections, we'll explore the option to check for data correctness, as well as which technical objects (e.g., BAdIs, classes, and methods) are available from SAP. In addition, we'll discuss the generic or the dynamic way these objects can be coded or developed.

4.6.1 Custom RAI Structure

Customizing RAIs is tailoring an RAI by structure and/or content (prepopulating or manipulating the data in the RAIs). The structure of an RAI is determined by the following components, as shown in [Figure 4.55](#):

- **Standard fields**
There is a standard set of predefined fields for RAIs.
- **Status of RAI**
States whether the RAI is in **Raw**, **Processable**, or **Processed** status.
- **Interface components**
You can choose what interface components to include in the RAI structure (see [Chapter 2, Section 2.3.2](#)). This includes the components for sales and distribution, profitability analysis, and more.

- **Customer fields**

Adding the custom fields is explained in detail in [Chapter 2, Section 2.3](#). The extension include structures, such as INCL_EEW_FARR_ARL, INCL_EEW_FARR_POB, INCL_EEW_FARR_REP, and INCL_EEW_FARR_CONTRACT, have to be extended.

You can customize the RAI structure by selecting the required interface component and also via customer fields.

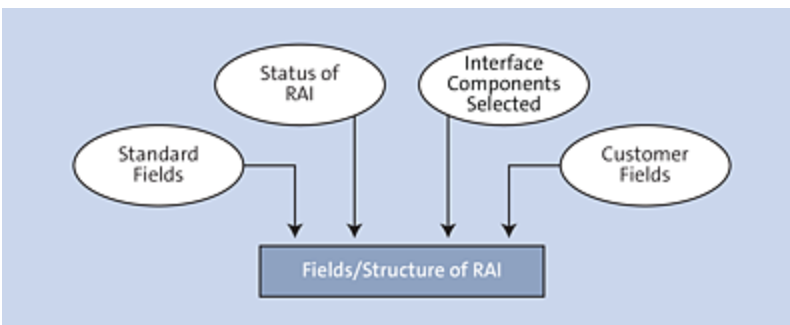


Figure 4.55 RAI Structure

4.6.2 Custom RAI Content

When we say *content* here, we're referring to the data in the RAI. Customization in the content means validating, modifying, adding default values to, and changing the format or type of the data; prepopulating the values by validating other fields; and many such data-related operations. As you know, some of this can be achieved via configuration, but most of this can be achieved by code, resulting in the question of where all this coding is done. We'll walk through different options in the following sections.

Let's consider an example where we are customizing the content of an RAI based on values of the other fields in the RAI. Say we have a case related to the field `SERVICE_TYPE`, which is a standard field in the RAI structure available in all RAI statuses. Per our requirement, this particular field in the RAI is allowed a set of values from a custom table. In standard SAP, the service type is allowed to be blank, so the RAI will be created in the raw status with blank `SERVICE_TYPE`. Now, we have to write the code where we can select the right `SERVICE_TYPE` from the custom table based on other fields like the company code, customer number, partner, and other conditions in the RAI. To achieve this, we need to write code at the point where the raw RAIs are being transferred to the processable status.

Similarly, we might want to change the content of some fields in the RAI when it's in the processable status and is being moved to processed. To achieve all this and more, SAP provides a list of BADIs that we'll explore in this section. These BADIs also add validations and enrichment to the customer fields; even standard fields can be enriched and validated, apart from the standard built-in validations.

Standard Approach

SAP has provided a standard class called `CL_FARR_RAI_IFCOMP` for validating the standard fields. This class is called at various stages and has a set of methods that are called for during the RAI creation and for various interface component validations. There are also methods that are called during the transfer of RAIs and during the processing of RAIs. All

the necessary standard validations and enrichment happens in the methods of the class, which are listed in [Table 4.2](#).

Methods	Description
RAI0_ENRICH_BASIC_C0	Enrich raw data basic condition item fields
RAI0_ENRICH_BASIC_MI	Enrich raw data basic main item fields
RAI0_ENRICH_BASIC_MI01	Enrich raw data order main item fields
RAI0_ENRICH_BASIC_MI03	Enrich raw data invoice main item fields
RAI2_CHECK_BASIC_C0	Final check before saving processable conditions
RAI2_CHECK_BASIC_C001	Final check before saving processable order conditions
RAI2_CHECK_BASIC_MI	Final check before saving processable main items
RAI2_CHECK_BASIC_MI01	Final check before saving processable order items
RAI2_CHECK_BASIC_MI02	Final check before saving processable fulfillment items
RAI2_CHECK_BASIC_MI03	Final check before saving processable invoice items
RAI2_CHECK_SD_MI01	Final check before saving processable SD order items

Methods	Description
RAI2_ENRICH_BASIC_C001	Enrich raw data basic condition item fields
RAI2_ENRICH_BASIC_MI	Enrich processable basic main item fields
RAI2_ENRICH_BASIC_MI01	Enrich processable order main item fields
RAI2_ENRICH_BASIC_MI02	Enrich processable basic main item fields
RAI2_ENRICH_BASIC_MI03	Enrich processable basic main item fields
GET_CURRENCIES_COMP_CODE	Get currency codes of company code
ADD_MESSAGE_TO_TABLE	Add message to table
CHECK_CURRENCY_CODES_EQUAL	Check currency code between main and condition items
CHECK_ORIGDOC	Check fields of original document
CHECK_ORIGDOC_WAERS	Check transaction currency of original document
CHECK_PA0BJNR	Check if PA0BJNR is filled
CHECK_PINVDOC	Check fields of planned invoice item
GET_CURRENCIES	Get local currency values

Methods	Description
SWITCH_CURRENCIES	Switch transaction currency if different from order
GET_DATA_FROM_ORDER	Get values from order item
GET_MAPPING	Get reference and contract from the mapping table
GET_MAPPING_TAB	Get mapping table for source documents
GET_RAI_MI_TAB	Get RAI table for source documents
GET_REFERENCE_FROM_MAPPING	Get reference from mapping table
GET_REFERENCE_FROM_ORDER	Get reference from order item
GET_REFERENCE_FROM_PREDOC	Get reference from predecessor document
SET_RAI0_KEYPP	Set subarea for parallelization
SET_RAI2_KEYPP	Set subarea for parallelization
CALCULATE_EXCHANGE_RATE	Calculate exchange rate for currency conversion

Table 4.2 Methods of Class CL_FARR_RAI_IFCOMP

These methods have various validations, checks, and enrichments for standard fields, even the methods to calculate the exchange rate and methods for copying the KEYPP field between main records and condition records. This

is the standard provided by SAP. Some of the methods here are triggered based on the configuration and the interface components that we selected for our RAI class.

Class CL_FARR_RAI_IFCOMP is for validating the standard fields. Now when you have to validate, check, enrich, or use default values in the custom fields or even the standard fields, this can be done in enhancement spot FARR_ARL along with the provided BAdIs, which we'll discuss next.

Custom Approach: Signature and Implementation Model

There are a lot of projects that require customer fields to be added to the RAI structure. The customer fields are needed because each business has unique requirements that call for customization. So, for storing their business-specific requirements, we add customer fields to the RAI structure (adding customer fields is discussed in detail in [Chapter 2, Section 2.3](#), where ZZPOB_CUR, ZZPOB_STATUS, and ZZPOB_DESC were added in a step-by-step demonstration).

There are cases where you also need customer fields to be validated, enriched, and prepopulated with default values. In the standard class mentioned in the previous section, CL_FARR_RAI_IFCOMP, the standard fields will be taken care of. So, for customer fields, SAP has provided enhancement spot FARR_ARL. In addition, you can also add validations or enrichments on standard fields; it's not just limited to customer fields.

To get started, go to Transaction SE18, and enter the **Enhancement Spot** as "FARR_ARL", as shown in

Figure 4.56.

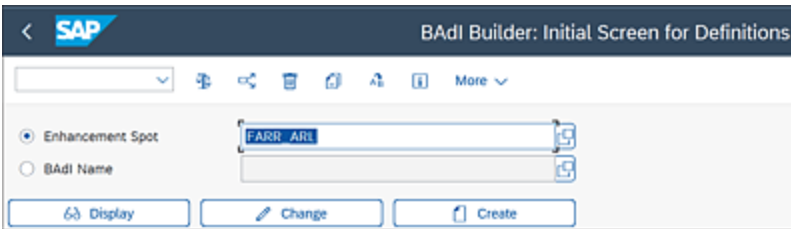


Figure 4.56 Enhancement Spot FARR_ARL

Click on **Display** to arrive at the screen shown in Figure 4.57.

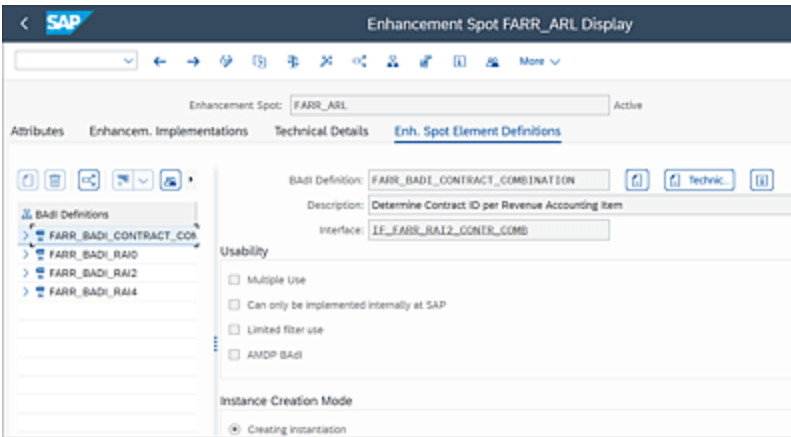


Figure 4.57 BAdI Definition of FARR_ARL

In FARR_ARL, there are four BAdI definitions of which only three are of concern for us now, as shown in Table 4.3. The BAdI for contract combination will be discussed in another section.

BAdI Name	Description	When It's Called
FARR_BADI_RAIO	Creation of raw RAIs	Before the raw items are saved to the database

BAdI Name	Description	When It's Called
FARR_BADI_RAI2	Creation of processable RAIs	Before the processable items are saved to the database
FARR_BADI_RAI4	Creation of processed RAIs	Before the processed items are saved to the database

Table 4.3 BAdIs for RAR

In this section, we'll discuss the BAdIs in detail, including details about their signature and implementation.

Business Add-In: FARR_BADI_RAIO

The first BAdI, FARR_BADI_RAIO, is triggered during the creation of custom raw RAIs. This one is triggered or called before the RAIs are created in the raw state and saved to database, so you can add any logic or changes/validations required to the standard or the custom fields even before the raw RAIs are saved to the database. The changes or the logic that you apply here will be done to the RAIs before they hit tables /1RA/0**010MI and /1RA/0**010C0, as shown [Figure 4.58](#).

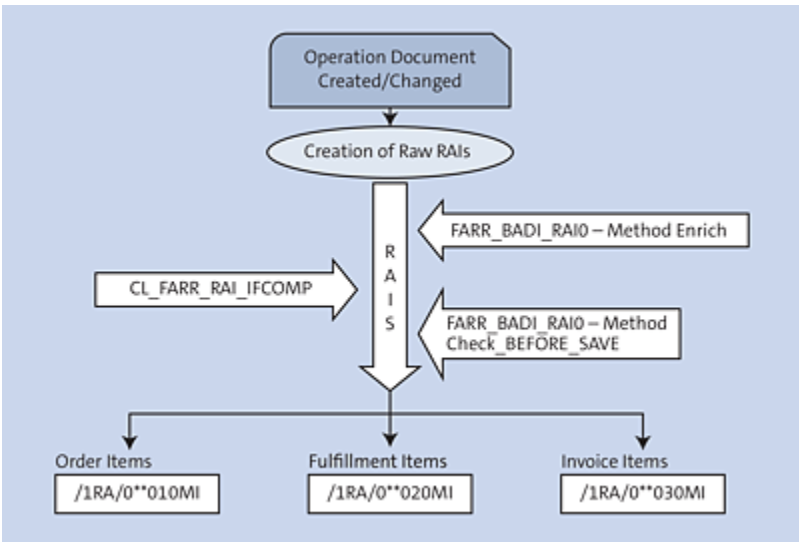


Figure 4.58 FARR_BADI_RAIO Triggered during RAI Creation

You can see that the ENRICH method is called first, then the class CL_FARR_RAI_IFCOMP is called, and finally the CHECK_BEFORE_SAVE method.

Now let's explore the methods and the signatures of the methods to understand what structures are available. Returning to the screen shown in [Figure 4.57](#), double-click on BAdI definition **FARR_BADI_RAIO**, and then double-click on interface **IF_FARR_BADI_RAIO**. Then, you can see the methods available in the interface, as shown in [Figure 4.59](#).

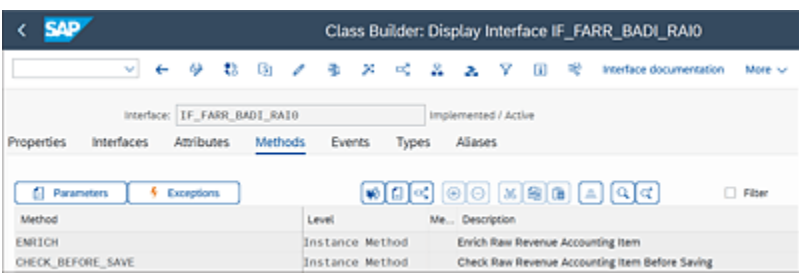


Figure 4.59 Methods in Interface IF_FARR_BADI_RAIO

As you can see, interface **IF_FARR_BADI_RAIO** has two methods: **ENRICH** and **CHECK_BEFORE_SAVE**. Double-

click on **ENRICH** to see the details of its parameters, as shown in [Figure 4.60](#).

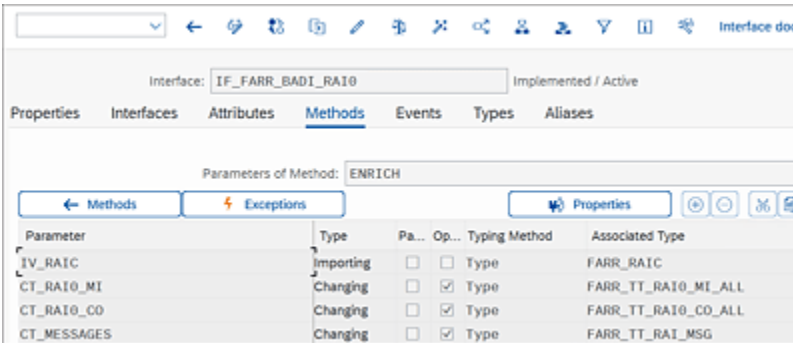


Figure 4.60 FARR_BADI_RAI0: ENRICH Method

The **ENRICH** method is executed before the SAP-delivered enrichment functionality for each interface component is executed. This method is called even before the RAIs are saved to the database or during the creation of the RAIs in **Raw** status. So, you can set any default values to any fields (custom or standard), and you can write code to select data from the database table and populate it in any of the RAI fields (custom or standard). You can also perform validations, throw error messages, and collect them in changing parameter **CT_MESSAGES**, as shown in [Figure 4.60](#).

The standard check and enrichments are done in class CL_FARR_RAI_IFCOMP. In addition, if you need to add further checks, enrichment, and validation, you can do so here in this BAdI. The BAdI has one **Importing** parameter and three **Changing** parameters. Importing parameter **IV_RAIC** is of type **FARR_RAIC**, which refers to the RAI class. You must pass the class you've configured in Transaction FARR_IMG under the defined RAI class, for example, SD01, SD03, and so on.

Next, we have the three changing parameters, which allow the data content to be changed. This is where you can change, enrich, or validate the field values. Let's take a quick look at each, as shown in [Figure 4.60](#):

- **CT_RAIO_MI**

This will have the main item of the RAI data. The structure of the parameter is of type **FARR_S_RAIO_MI_ALL**, which you can view in Transaction SE11 by giving the structure name in **DATA TYPE**. As multiple RAIs will be passed to the method, it's a table referring to the table type of **FARR_TT_RAIO_MI_ALL**.

- **CT_RAIO_CO**

This will have the condition items of the RAI data. It will carry multiple records of condition items, so it's a table of table type **FARR_TT_RAIO_CO_ALL** with the structure of the table of type **FARR_S_RAIO_MI_ALL**.

- **CT_MESSAGES**

This is of table type **FARR_TT_RAI_MSG**, which is a type of work structure of **FARR_S_RAI_MSG**.

If there are any errors in the RAIs, then changing parameter **CT_MESSAGES** is updated with the key field values of the RAI. It also has the attribute of the message structure to pass the message details. The RAIs with errors will be captured here and won't be moved further.

The **CHECK_BEFORE_SAVE** method is executed before raw RAIs (RAI0) are saved to the database. This is another method for adding changes just before you save the final call and before saving all the RAIs created in the **Raw** status to the database.

This BAdI has the same importing and changing parameters, as shown in [Figure 4.61](#).

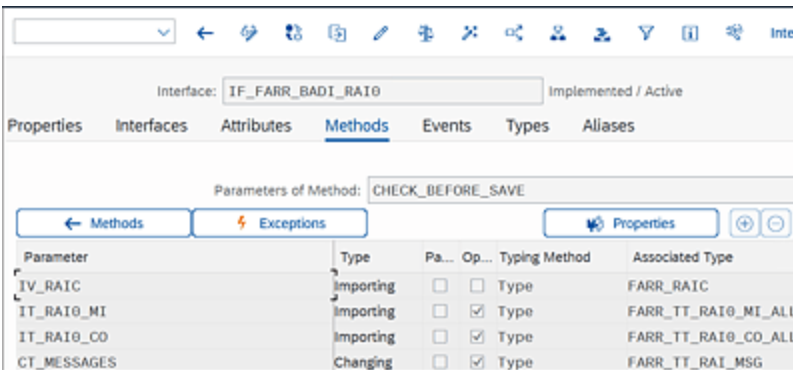


Figure 4.61 BAdI FARR_BADI_RAI0: CHECK_BEFORE_SAVE Method

Business Add-In: FARR_BADI_RAI2

The raw RAIs that don't have any errors are then transferred to the **Processable** status. You will need some validations or enrichments at this stage. So, this BAdI, FARR_BADI_RAI2, is used for adding checks, validations, or enrichments to raw RAIs that are being transferred to the **Processable** status.

This BAdI is triggered or called before the RAIs are transferred to the **Processable** status and saved to the database, so you can add any logic or changes/validations that you need to add to the standard or the custom fields of the raw RAIs that are being transferred. The changes or the logic that you apply here will be done to the RAIs before they hit tables /1RA/0**012MI and /1RA/0**012C0, as shown in [Figure 4.62](#).

From Transaction FARR_RAI_MON, you can select RAIs in **Raw** status and then debug the transfer of RAIs by clicking the **Transfer** button. During a mass transfer of RAIs, you

call program RFARR_RAI_PP_TRANS_START. In both cases, the BAdI is called.

There are also two methods in this interface: ENRICH and CHECK_BEFORE_SAVE. ENRICH is called before the raw RAIs are moved to the status **Processable** and are saved to the database, that is, during the transfer of the RAIs.

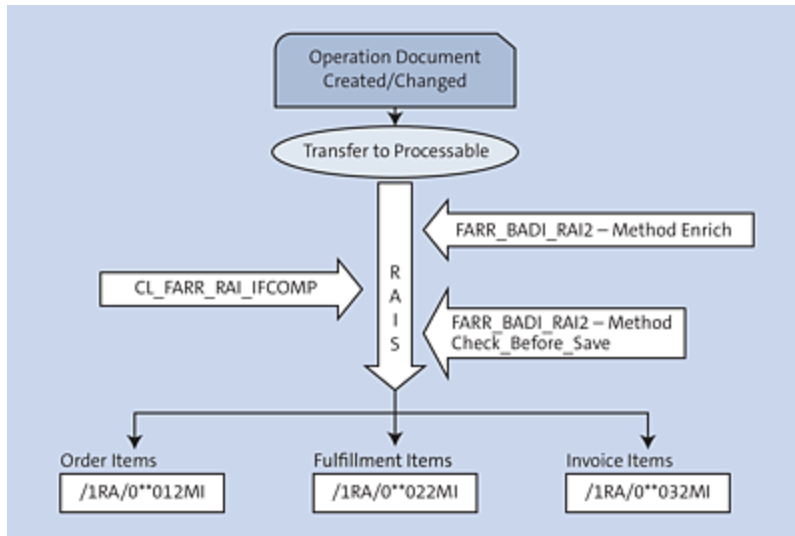


Figure 4.62 BAdI FARR_BADI_RAI2: Triggered during RAI Transfer

You can set default values to any fields (standard or custom) and write code to perform database selects for populating certain fields based on conditions. You can also write code to populate certain RAI fields based on the values of other RAI fields, or by writing database selects to fetch the value of certain RAI fields that are based on the conditions of other RAI fields. Finally, you can perform validations, throw error messages, and collect them in changing parameter **CT_MESSAGES**, as shown in [Figure 4.63](#).

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC
CT_RAI2_MI	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL
CT_RAI2_CO	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI_MSG

Figure 4.63 BAdI FARR_BADI_RAI2: ENRICH Method

The standard check and enrichments are done in class CL_FARR_RAI_IFCOMP. If you need to add further checks, enrichments, and validations, you can do so in this BAdI. There is an enrichment functionality for each interface component provided by SAP, which will be called after the ENRICH method is called.

The BAdI has one importing parameter and three changing parameters. The importing parameter is **IV_RAIC**, which is of type **FARR_RAIC** and refers to the RAI class. You have to pass the class configured in Transaction FARR_IMG under the defined RAI class, for example, SD01, SD03, and so on.

Then, you have three changing parameters, which, as you know, allow the data content to be changed. This is where you can change, enrich, or validate the field values. The changing parameters shown earlier in [Figure 4.63](#) are described here:

- **CT_RAI2_MI**

This will have the main item of the RAI data. The structure of the parameter is of type **FARR_S_RAI2_MI_ALL**, which you can view in Transaction SE11 by giving the structure name in **DATA TYPE**. As multiple RAIs will be passed to

the method, it's a table referring to the table type of **FARR_TT_RAI2_MI_ALL**.

- **CT_RAI2_CO**

This will have the condition items of the RAI data. It will carry multiple records of condition items, so it's a table of table type **FARR_TT_RAI2_CO_ALL** with the structure of the table of type **FARR_S_RAI2_MI_ALL**.

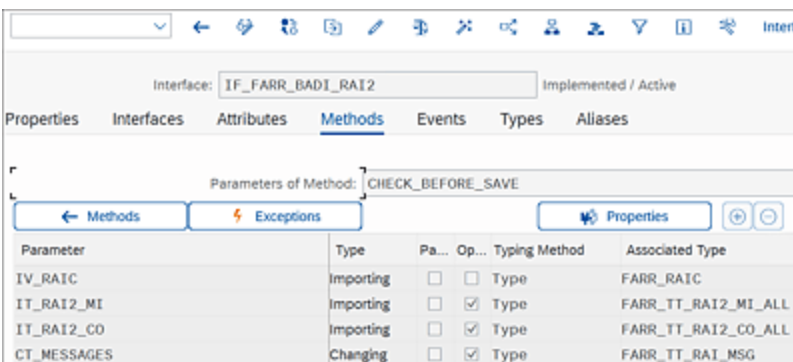
- **CT_MESSAGES**

This is of table type **FARR_TT_RAI_MSG**, which is a type of the work structure of **FARR_S_RAI_MSG**.

If there are any errors in the RAIs, then changing parameter **CT_MESSAGES** is updated with the key field values of the RAI. It also has the attribute of the message structure to pass the message details. The RAIs with errors will be captured here and won't be moved on further.

The **CHECK_BEFORE_SAVE** method is executed before processable RAIs are saved to the database. This is another method for adding changes just before you save to the database for all the RAIs.

This BAdI has the same importing and changing parameters, as shown in [Figure 4.64](#).



The screenshot shows the SAP IDE interface for the BAdI IF_FARR_BADI_RAI2. The 'Methods' tab is selected, and the 'CHECK_BEFORE_SAVE' method is highlighted. Below the method name, there is a table listing the parameters of the method. The table has columns for Parameter, Type, Pa... (Parameter), Op... (Optional), Typing Method, and Associated Type.

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC
IT_RAI2_MI	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL
IT_RAI2_CO	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI_MSG

Figure 4.64 BAdI FARR_BADI_RAI2: CHECK_BEFORE_SAVE Method

Business Add-In: FARR_BADI_RAI4

This one is triggered or called before the RAIs are processed and moved to the **Processed** status and saved to the database, so you can add any logic or changes/validations to the standard or custom fields. The changes or the logic that you apply here will be done to the RAIs before they hit tables /1RA/0**014MI and /1RA/0**014C0, as shown in [Figure 4.65](#).

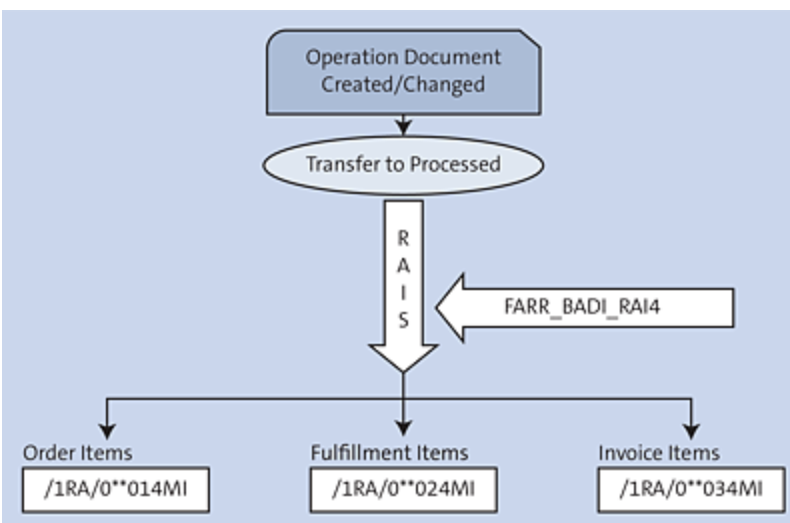


Figure 4.65 BAdI FARR_BADI_RAI4: Triggered during the RAI Process

Returning to the screen shown previously in [Figure 4.57](#), double-click on BAdI definition **FARR_BADI_RAI4**, and then double-click on interface **IF_FARR_BADI_RAI4**. Then, you can see the methods available in the interface, as shown in [Figure 4.66](#).

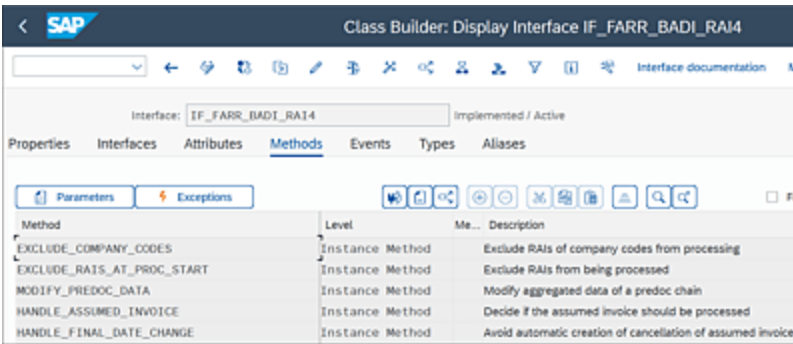


Figure 4.66 Interface IF_FARR_BADI_RAI4 Methods

This BAdI has a few different methods to discuss:

- **EXCLUDE_COMPANY_CODES**

This method, as shown in [Figure 4.67](#), is called when the RAIs are being moved to the **Processed** status and before they are saved to the database. This method is called mainly for company code checks. If there are some company codes that you may want to exclude, then you can implement this method in BAdI FARR_BADI_RAI4.

For mass Transactions FARR_RAI_PROC (Process Revenue Accounting Items), FARR_RAI_PROC_LOAD (Initial Load Process Revenue Accounting Items) used during the migration and initial load processing, and FARR_RAI_PROC_NEWACP (Reprocess Revenue Accounting Items for New Accounting Principle), this method is executed before the RAIs are selected from the database. RAIs that belong to one of the listed company codes provided in exporting parameter ET_EXCLUDED_COMPANY_CODE aren't selected for processing.

If RAI processing is started from the RAI monitor (Transaction FARR_RAI_MON), the method is executed at the beginning of the RAI processing.

The method has one importing parameter, IV_PROCESSING_MODE, which is of type FARR_PROCESSING_MODE

(processing mode includes processing, initial load, transition). To importing parameter IV_PROCESSING_MODE, you have the option to provide three possible values as input:

- Space: For normal processing.
- **1**: For initial load processing.
- **2**: For new accounting principle during reprocessing.

The method has one exporting parameter called ET_EXCLUDED_COMPANY_CODE. This will save the list of company codes that have to be excluded during processing. The structure of this exporting parameter is FARR_S_COMPANY_CODE_MSG. This exporting parameter will store multiple values and has to be of type table. The table type is FARR_TT_COMPANY_CODE_MSG, which includes a standard messaging structure and the company code to be excluded.

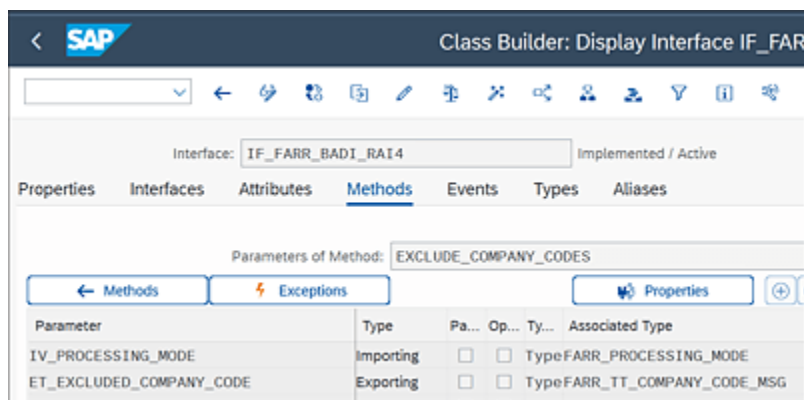


Figure 4.67 EXCLUDE_COMPANY_CODES Method

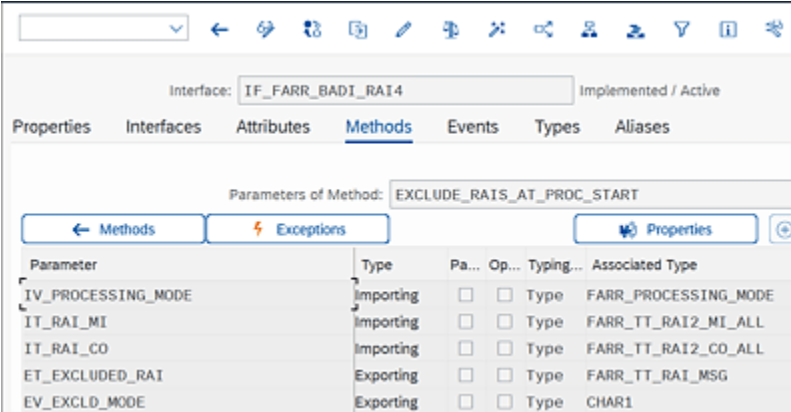
- **EXCLUDE_RAIS_AT_PROC_START**

This method, as shown in [Figure 4.68](#), is executed at the beginning of processing, initial load processing, or reprocessing for a new accounting principle. It's basically like making an exclusion list of RAIs that should be

excluded and not processed. The reason for exclusion could be something like the validations that are done on RAIs; in this case, the RAIs failing the validation are all grouped together in the exclusion list.

The importing parameters are IV_PROCESSING_MODE, IT_RAI_MI, and IT_RAI_CO. Parameters, IT_RAI_MI and IT_RAI_CO contain all revenue accounting main items and conditions that are selected for processing and aren't locked by other processes.

The exporting parameters in the BAdI implementation are ET_EXCLUDED_RAI and EV_EXCLD_MODE. By using exporting parameter ET_EXCLUDED_RAI, you can define RAIs that should not be processed. All RAIs listed in ET_EXCLUDED_RAI aren't processed.



The screenshot shows the SAP ABAP IDE interface for the method EXCLUDE_RAIS_AT_PROC_START. The interface includes a toolbar at the top, a navigation pane with tabs for Properties, Interfaces, Attributes, Methods, Events, Types, and Aliases, and a main area displaying the parameters of the method. The parameters are listed in a table with columns for Parameter, Type, Pa..., Op..., Typing..., and Associated Type.

Parameter	Type	Pa...	Op...	Typing...	Associated Type
IV_PROCESSING_MODE	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_PROCESSING_MODE
IT_RAI_MI	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL
IT_RAI_CO	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL
ET_EXCLUDED_RAI	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI_MSG
EV_EXCLD_MODE	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	CHAR1

Figure 4.68 EXCLUDE_RAIS_AT_PROC_START Method

- **MODIFY_PREDOC_DATA**

This method is executed for revenue accounting order items during processing, initial load processing, and reprocessing for a new accounting principle. This is triggered when an RAI belonging to the predecessor chain is processed. [Figure 4.69](#) shows the parameters of the method.

Importing parameters IT_RAI_MI_PREDOC_CHAIN and IT_RAI_CO_PREDOC_CHAIN contain all revenue accounting main items and conditions in the **Processable** and **Processed** statuses that have been used for aggregation during predecessor handling.

Importing parameters IS_RAW_POB_AGGREGATED and IT_RAW_POB_CO_AGGREGATED contain the data of the raw POB that resulted from the aggregation of the RAIs.

In the BAdI implementation, it's possible to change those fields of the raw POB that are defined in structure FARR_S_PREDOC_CHANGE_MI and hand them over using exporting parameter ES_RAW_POB_AGGREGATED.

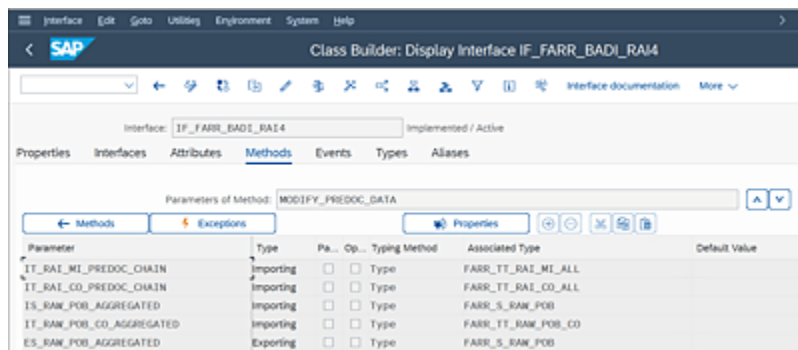


Figure 4.69 MODIFY_PREDOC_DATA Method

- **HANDLE_ASSUMED_INVOICE**

This method, as shown in [Figure 4.70](#), is executed at the beginning of processing. This is basically used for invoice processing and has the following parameters:

- Importing parameter IV_RAW_INV of type FARR_S_RAW_POB: This will carry the key fields to identify the invoice RAIs.
- Returning parameter RV_PROCESSABLE of type XFELD: This flag allows you to return whether the invoices are to be processed or not.

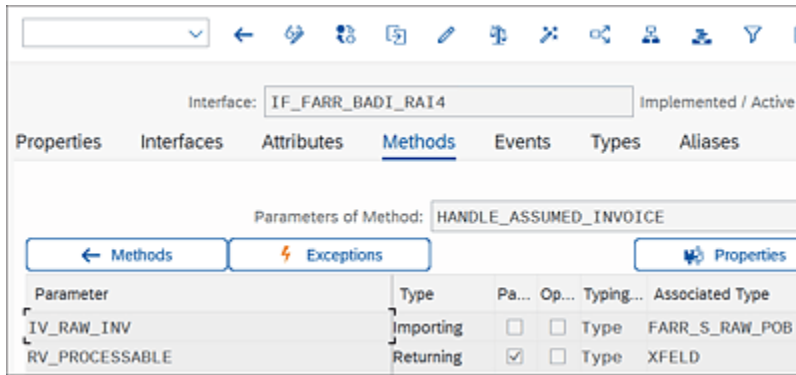


Figure 4.70 HANDLE_ASSUMED_INVOICE Method

- **HANDLE_FINAL_DATE_CHANGE**

This method, as shown in [Figure 4.71](#), is used to avoid automatic creation of cancellation of assumed invoice. This method has a single returning parameter RV_OWN_LOGIC of type XFELD. The parameter name includes “own logic” to convey that you could design the returning parameters per your own requirements and base further processing on the returning parameters as well.

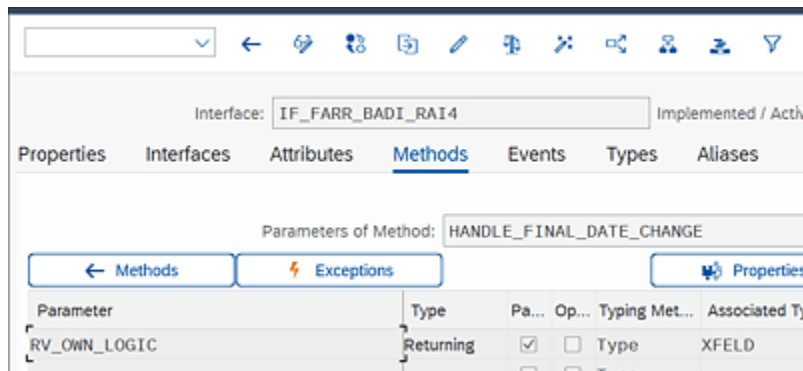


Figure 4.71 HANDLE_FINAL_DATE_CHANGE Method

We’ve now shown the possible objects available for bringing in validation on customer fields and standard fields. It’s important to have the data in the right format and ensure that it’s correct, which can be done using these methods.

4.6.3 Singleton Classes

In this section, we'll briefly introduce singleton classes. We'll explain how to create a singleton class with a step-by-step approach, and then we'll show you how to instantiate and use a singleton class.

Before we move further, let's first discuss why we're talking about singleton classes here and how this was used in a particular scenario in one of our projects. In one of our previous projects, we had implemented BAdI `FARR_BADI_RAI2` and had to do a lot of validation before the raw RAIs were transferred to the **Processable** status. In particular, we had a lot of enrichments, validations, and checks to do in method `ENRICH`. The most important concern in that project was the huge volume of data and the future rollouts that would add to the volume of data. It was crucial for us to develop code that was performance optimized, keeping the number of database accesses to a minimum, and with no redundant code pieces.

So, the design was centered around optimizing the performance. The standard validations weren't enough as there were two systems or sources of information for the same POB, which weren't in sync. Because every third-party source system is different, the standard data validations aren't sufficient. There were a lot of custom fields added as the requirements were complex and couldn't be addressed just with the standard fields. Even those custom fields had a lot of validation, setting of default values, and various checks to perform on the custom fields, as well as enrichments of the standard fields and custom fields. This

required us to implement BAdI FARR_BADI_RAI2 and specifically method ENRICH to address the preceding concern.

As part of our design, we had to develop two custom tables for the dynamic design that we had planned. One of the tables to be designed was called the flow determination table, ZRAR_FLOW, and the other one was for validation rules maintenance, ZRAR_VALID_RULE.

The data in table ZRAR_FLOW will be like a dynamic flow control table, and table ZRAR_VALID_RULE will have a set of validation rules maintained for processing the RAIs. The entries in these tables will have the same set of entries for all the headers of a given company code.

Consider a case where you're transferring 100,000 entries, as shown in [Figure 4.72](#). When method ENRICH is designed, it will have a SELECT statement to select data from table ZRAR_FLOW and another SELECT statement to fetch data from table ZRAR_VALID_RULE. Because there are 10,000 HeaderIDs, the SELECT statements will be called 10,000 times for select data on table ZRAR_FLOW and 10,000 times for select data on ZRAR_VALID_RULE. Both SELECT statements will have the same data for all the header IDs because the RAI transfer is being done for one company code. This means that the same data will be pulled again and again 20,000 times. Because our target here is to reduce the number of redundant database selects, we could just select the data once during the entire processing and save the data somewhere to reuse it instead of having to select it 20,000 times. This is when we came up with the singleton approach.

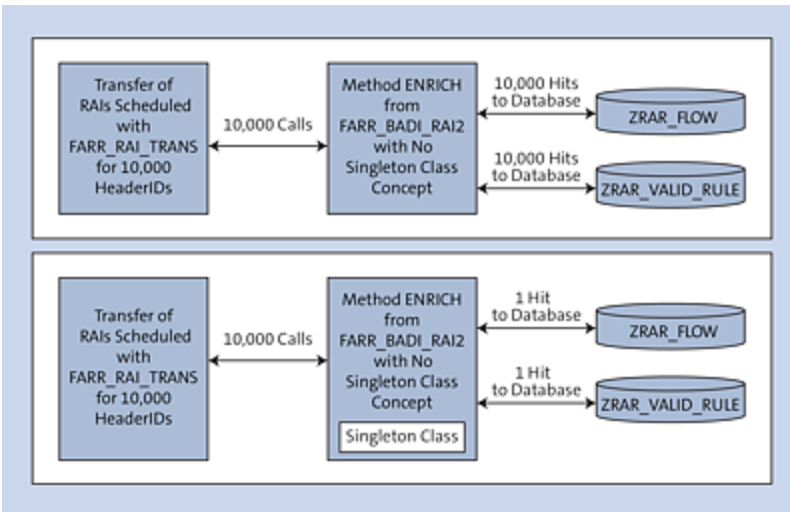


Figure 4.72 Singleton Class in ENRICH: FARR_BADI_RAI2

The data from tables ZRAR_FLOW and ZRAR_VALID_RULE will be collected only once in the singleton class, which will be instantiated only once during the entire cycle of mass transfer for thousands of header IDs in the singleton class to avoid a lot of database calls. With that example under our belt, we'll now discuss the singleton class and how to create it.

Singleton classes are different from static classes. A class that only contains static components and no instance components is referred to as a static class. A global static class is loaded once with its class pool into the current internal session. Like every ABAP program, it can't be explicitly deleted from the session. The static methods (declared using CLASS-METHODS) of a class can't be redefined in subclasses.

A singleton is a design pattern where the class has the task of creating objects. The class ensures that only one object exists for every internal session that is made available to consumers.

The difference between static classes and singleton classes is shown in [Table 4.4](#).

Singleton Class	Static Class
This class contains static components and instance components as well.	This class contains static components and no instance components.
This class can have at the most one instance.	This class contains only static methods and attributes, so it doesn't require instantiation.
From a memory consumption perspective, this class gives you control over when to instantiate.	From a memory consumption perspective, there is no way to explicitly free up the memory space occupied by static classes.
Objects are created only from within the class.	Objects can be created from outside the class.

Table 4.4 Difference between Static Classes and Singleton Classes

Let's dive into creating a singleton class. Go to Transaction SE24, enter a class name per the naming conventions used in the project, and click on the **Create** button. As in our case, let's call the class "ZCL_RAR_MEMLOAD", as shown in [Figure 4.73](#).

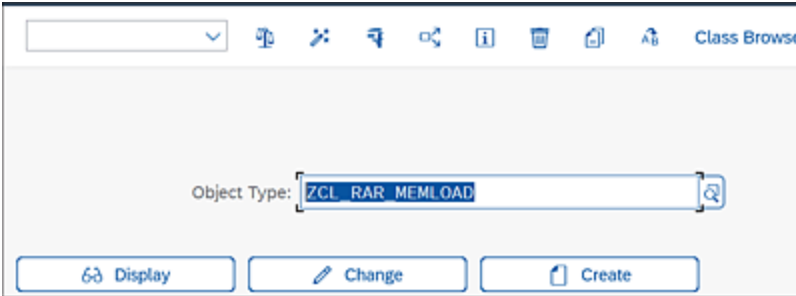


Figure 4.73 Creating a Singleton Class

After you click on **Create**, the screen shown in [Figure 4.74](#) appears, where you have to enter an appropriate **Description**.

Check the **Final** flag. All other details can stay as the default entries. It's important to make the class **Final** because you can't allow inheritance on this class.

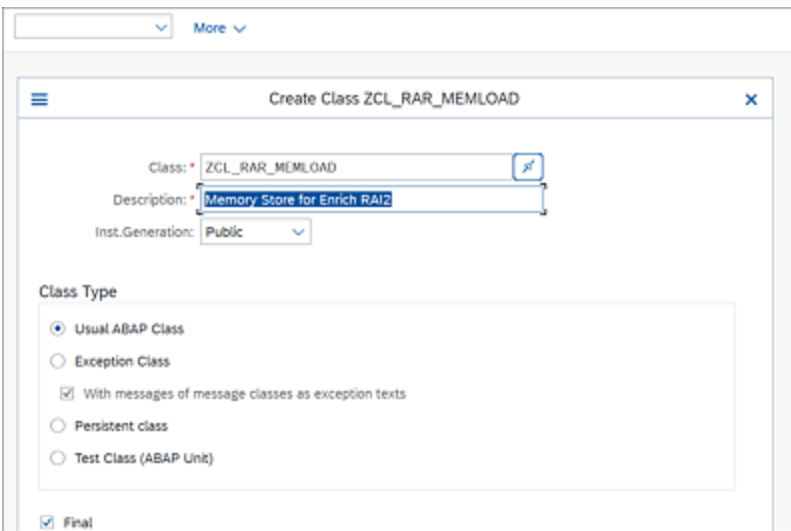


Figure 4.74 Singleton Class

Click on **Save**, and on the next screen, select the appropriate package and save in an appropriate transport request. The class is created and activated as shown in [Figure 4.75](#).

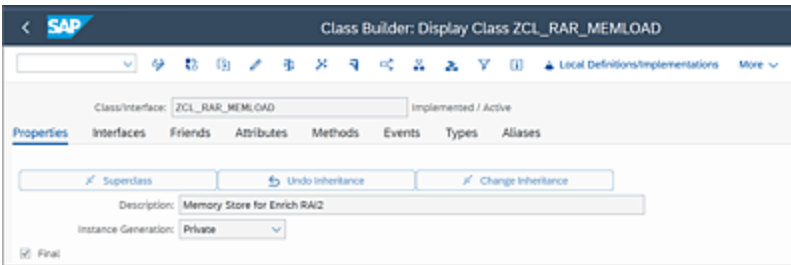


Figure 4.75 Singleton Class Creation

Now you can start creating the types and method for the class. First, you need to create a private static attribute of the type of reference to the same class, as shown in [Figure 4.76](#). Go to the **Attributes** tab of the class, and enter the name of the **Attribute** as “GO_INSTANCE”. Set **Level** to **Static Attribute**, and then enter **Private** under **Visibility**. Most important is to enter **Type Ref To** under **Typing**, and the **Associated Type** should be the same class name, which, in this case, is **ZCL_RAR_MEMLOAD**.

You next need to create another attribute by following almost the same steps in [Figure 4.76](#). This will be a global table **GT_ZRARC_FLOW_MGMT** to store the data from table ZRAR_FLOW. The associated type in this case will be **ZRAR_FLOW_TT**, which is a table type for the table.

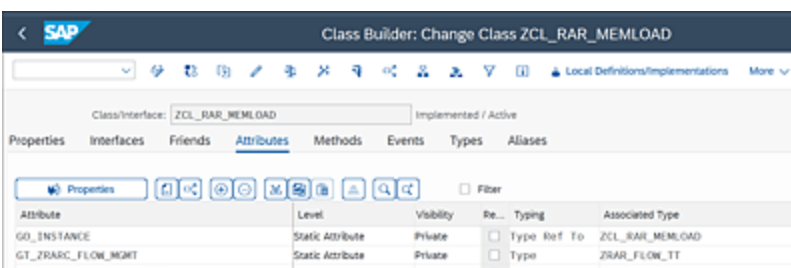
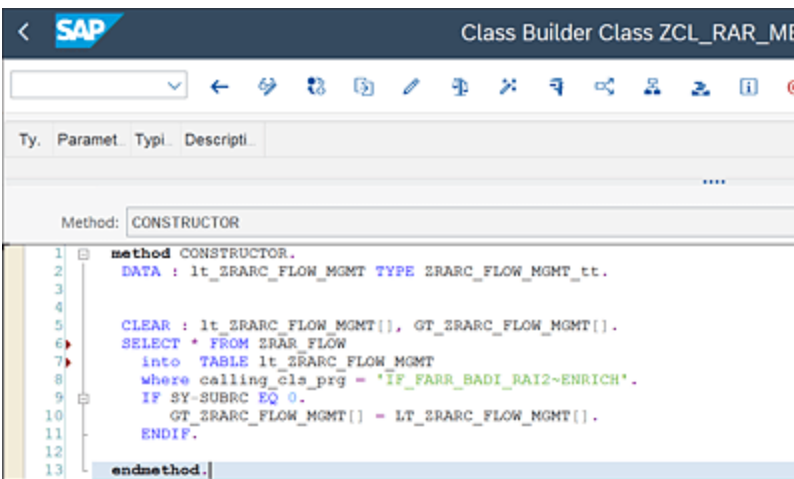


Figure 4.76 Singleton Class Attributes

Next, go to the **Methods** tab of the class, and create two methods. The first one is the constructor, which will have the SELECT statement to the dynamic flow table ZRAR_FLOW.

Enter the **Method** name as “CONSTRUCTOR”, enter **Instance Method** under **Level**, enter **Private** under **Visibility**, and enter the description as “constructor” or any description you like.

Double-click on the method CONSTRUCTOR, and then you can use the editor to enter the code. Enter the code for selecting data from table ZRAR_FLOW and store it in the global internal table, as shown in the sample code in [Figure 4.77](#). Of course, you could add your own additional logic and filters to the code.



```
1 method CONSTRUCTOR.
2   DATA : lt_ZRAR_FLOW_MGMT TYPE ZRAR_FLOW_MGMT_tt.
3
4
5   CLEAR : lt_ZRAR_FLOW_MGMT[], gt_ZRAR_FLOW_MGMT[].
6   SELECT * FROM ZRAR_FLOW
7     INTO TABLE lt_ZRAR_FLOW_MGMT
8     WHERE calling_cls_prg = 'IF_FARR_BADI_RAI2-ENRICH'.
9   IF SY-SUBRC EQ 0.
10    gt_ZRAR_FLOW_MGMT[] = lt_ZRAR_FLOW_MGMT[].
11  ENDIF.
12
13 endmethod.
```

Figure 4.77 Example of a Constructor of Singleton Class

We now have to create the second method GET_INSTANCE. Follow the same steps as for the previous method, but set the **Visibility** to **Public**, and enter **Static Method** under **Level**. Then double-click on the **GET_INSTANCE** method to get into the editor and enter the code. Enter the code to create an instance of the class, as shown in [Figure 4.78](#). The sample code is provided, but you could add any additional logic to the code. This particular piece of code focuses on the fact that this will be the only place where you can create an object of the class.

Ty.	Parameter	Typing	Description
	value(RO_INSTANCE)	TYPE REF TO ZCL_RAR_MEMLOAD	Memory Store for Enrich RAI2


```

Method: GET_INSTANCE active
1 method GET_INSTANCE.
2
3   if go_instance is NOT bound.
4     CREATE OBJECT go_instance.
5   endif.
6   ro_instance = go_instance.
7   endmethod.

```

Figure 4.78 Singleton Class Instance Method

Save and activate the class, and the class is ready for instantiating and for use. The two methods in the class will look like [Figure 4.79](#).

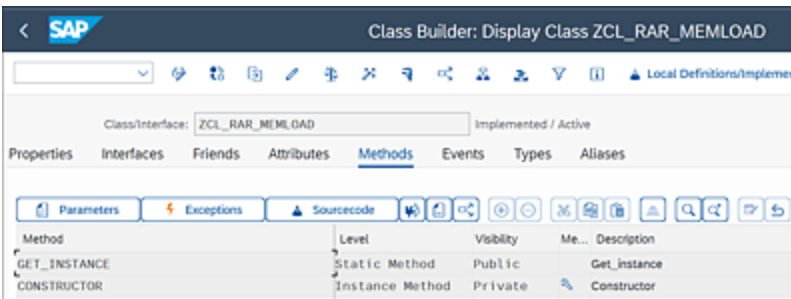


Figure 4.79 Two Methods of the Sample Singleton Class

Now this singleton class will be called inside method ENRICH for BAdI FARR_BADI_RAI2 to further explain the way this singleton class will be called.

You create a global object called GO_OBJECT that will be of the type of class ZCL_RAR_MEMLOAD. Note that ZCL_RAR_MEMLOAD is the singleton class that you created. Then, in ENRICH, the following happens, as shown in [Figure 4.80](#):

1. During the mass transfer, the loop starts off with the first header_id. During the processing of the first header_id, method ENRICH from BAdI FARR_BADI_RAI2 will be called. In the method, instantiation of the singleton class will happen, and object GO_OBJECT will be created for

singleton class ZCL_RAR_MEMLOAD from method GET_INSTANCE of the class.

2. The constructor will be called, and the SELECT statement for table ZRAR_FLOW will be executed. Global table GT_ZRARC_FLOW_MGMT will be populated with the data from flow table ZRAR_FLOW.
3. Using the object GO_OBJECT and the global internal table GT_ZRARC_FLOW_MGMT in the singleton class, which will have the data from the flow table, you'll populate the global internal table in method ENRICH. The global internal table in ENRICH is also called GT_RARC_FLOW_MGMT, which will also be populated with the flow table data and will be available in the BAdI FARR_BADI_RAI2 in the ENRICH method.
4. In the loop for the next header_id, object go_object is checked if it's bound; then it won't be created again, and the SELECT statement isn't executed. However, GT_ZRARC_FLOW_MGMT in ENRICH will be populated by using the object reference, as you can see in [Figure 4.80](#). This is a way of reducing database accesses.
5. The singleton design pattern ensures that only one instance of the class is created. The class contains its own constructor, which will fetch the data from the database only once, and the method to create the instance. In that method, logic is placed that ensures the creation of only one instance.

```
IF go_object IS NOT BOUND.  
  CALL METHOD zclrar_arl_mem_load=>get_instance  
    RECEIVING  
      ro_instance = go_object.  
  
ENDIF.  
CLEAR : gt_datval[], gt_precedance[].  
IF go_object IS BOUND.  
  gt_ZRARC_FLOW_MGMT[] = go_object->gt_ZRARC_FLOW_MGMT.
```

Figure 4.80 Singleton Class

This stays active during full processing, and you can access the necessary data.

4.6.4 Dynamic Processing Flow Controlled from Table

At times, users expect flexibility in customization. Even the coding standards expect that there should not be any hardcoding in the code because we always want it to be flexible. It would be nice if we could make the code configurable, like plug and play, but that would be inflexible and, as a result, too difficult to implement. We've seen cases where we have requirements for different company codes within the same program. Here, we want to demonstrate a technical approach or a programming technique that will enable us to develop a dynamic program which will help us control the calls to classes, methods, or performs based on the input values or other variables. The programs will control the processing blocks as plug-and-play services. This is the level of flexibility that we want to discuss.

Let's consider a case where we have a requirement that a set of activities are to be performed for each company code. From Transaction SE24, let's create class ZCLRAR_BADIRAI2, which will have all the validations and enrichment developed as methods. Then, we'll create a method that will drive the sequence of calls to different methods based on the requirements for the company code. If the company code = 1000, then the method to be called is PROCESS_1000;

inside PROCESS_1000, we'll have calls to more methods specific to company code 1000. Similarly, if the company code = 2000, then the method to be called is PROCESS_2000. Now the decision to call method PROCESS_1000 or PROCESS_2000 in normal programming will have the block of code shown in [Listing 4.1](#).

```
If company code = 1000
  Call method process_1000.
Elseif company code = 2000
  call method process_2000.
Elseif .....
```

Listing 4.1 Normal Programming Coding Block

What if instead of code with IF conditions, we could have a table that will control the flow, where we don't have to hardcode the value of the company code in the code? This is where we need the flow control table. The diagrammatic representation explains the entire concept as shown in [Figure 4.81](#).

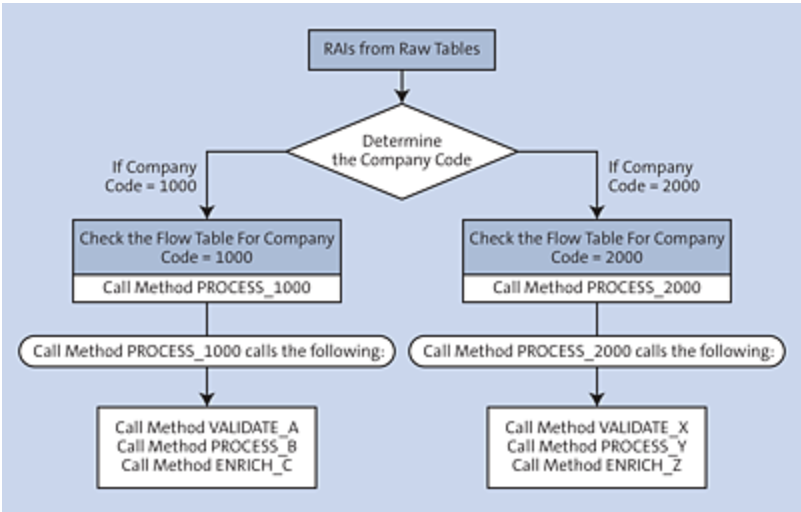


Figure 4.81 Dynamic Flow Determination at Runtime Based on Company Code

We've tried to focus on the important components of this design. We need the following:

- Flow control table
- Switch methods
- Allow methods

Let's explore each of the components in detail.

Flow Control Table

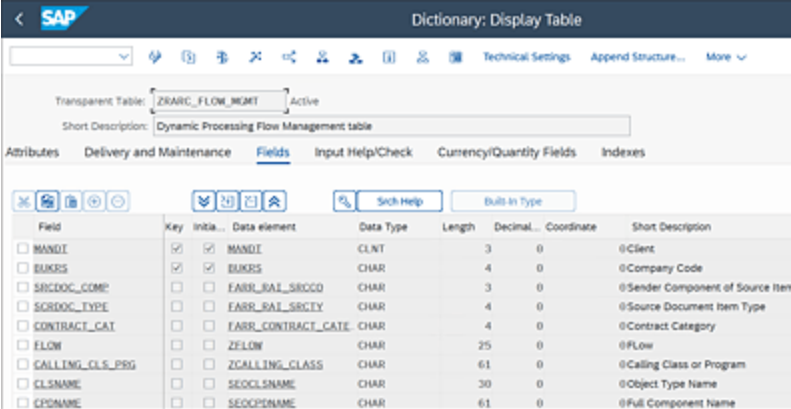
We're familiar with the creation of a table using Transaction SE11. In our example, we've designed the flow control table ZRARC_FLOW_MGMT, as shown in [Figure 4.82](#). The table has the following key fields as listed in the **Fields** tab:

- **BUKRS**
This will have the specific company code that we want the customization for.
- **SRCDOC_COMP**
This is the identifier of the sender system and helps us form primary key combinations.
- **SRCDOC_TYPE**
This will help us identify the RAI's document type: order, fulfillment, or invoice.
- **CONTRACT_CAT**
This is for the contract details and will be specific to each project.
- **FLOW**
This has the values as **Allow** or **Switch**. **Allow** basically is like an IF condition to proceed with the execution of the

method. **Switch** is like an option to choose which method to execute.

- **CALLING_CLS_PRG**

This the name of the method or the program that has the code.



Transparent Table: ZRARC_FLOW_MGMT Active
Short Description: Dynamic Processing Flow Management table

Attributes Delivery and Maintenance Fields Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		@Client
<input type="checkbox"/> BUKRS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	BUKRS	CHAR	4	0		@Company Code
<input type="checkbox"/> SINCDOC_COMP	<input type="checkbox"/>	<input type="checkbox"/>	FARR_BAI_SIBCCO	CHAR	3	0		@Sender Component of Source Item
<input type="checkbox"/> SINCDOC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_BAI_SIBCEY	CHAR	4	0		@Source Document Item Type
<input type="checkbox"/> CONTRACT_CAT	<input type="checkbox"/>	<input type="checkbox"/>	FARR_CONTRACT_CATE	CHAR	4	0		@Contract Category
<input type="checkbox"/> FLOW	<input type="checkbox"/>	<input type="checkbox"/>	ZFLOW	CHAR	25	0		@Flow
<input type="checkbox"/> CALLING_CLS_PRG	<input type="checkbox"/>	<input type="checkbox"/>	ZCALLING_CLASS	CHAR	61	0		@Calling Class or Program
<input type="checkbox"/> CLSNAME	<input type="checkbox"/>	<input type="checkbox"/>	SEOCCLNAME	CHAR	30	0		@Object Type Name
<input type="checkbox"/> CPDNAME	<input type="checkbox"/>	<input type="checkbox"/>	SEOCPCNAME	CHAR	61	0		@Full Component Name

Figure 4.82 Flow Table

We need to create a table maintenance generator for table ZRARC_FLOW_MGMT. Creation of a table maintenance generator through Transaction SE11 is an essential ABAP skill that is beyond the scope of this book.

Once the table maintenance generator is created, we need to go to Transaction SM30 and enter the table name as “ZRARC_FLOW_MGMT”. Then the flow table needs to be populated with the details of the company code, source document component, flow, and calling program, as shown in [Figure 4.83](#).

CoCd	Send Comp.	Src.	Contr.	Flow	calling cls prg	Class/Interface	Interface Component
<input type="checkbox"/> 1000	ZX	*	*	SWITCH	IF_FARR_BADI_RAI2+ENRICH	ZCLRAR_BADIRAI2	PROCESS_1000
<input type="checkbox"/> 2000	ZA	*	*	SWITCH	IF_FARR_BADI_RAI2+ENRICH	ZCLRAR_BADIRAI2	PROCESS_2000
<input type="checkbox"/> 1000	ZX	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	VALIDATE_A
<input type="checkbox"/> 1000	ZX	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	PROCESS_B
<input type="checkbox"/> 1000	ZX	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	ENRICH_C
<input type="checkbox"/> 2000	ZA	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	VALIDATE_X
<input type="checkbox"/> 2000	ZA	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	PROCESS_Y
<input type="checkbox"/> 2000	ZA	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	ENRICH_Z

Figure 4.83 Flow Table Entries

In these table entries, we have company code, sender component, source document type, and contract category because our example is based on RAR. If it's for any other module, you can define your set of fields for determining the flow.

Switch Method

As we previously mentioned, calling method PROCESS_1000 or method PROCESS_2000 is dependent on the value of the company code. In general, if the call to methods is dependent on the input data or based on the input value, those methods will be called the switch methods. The call to the method switches to the corresponding method per what is maintained in the flow table. This is called the switch method.

If we look at our example, the call to method PROCESS_1000 or PROCESS_2000 is based on the company code value, so those methods are called switch methods in our case. The **Flow** column in the table shown in [Figure 4.83](#) shows the **SWITCH** value accordingly. We'll provide the sample code to implement the logic next. This code can be used along with the flow table to dynamically call the methods from a

particular class based on the input value. The sample is from the ENRICH method in BAdI RAI2.

Additionally, if we add another company code 3000 in the future, then we can enhance the design by adding a new method PROCESS_3000 in class ZCLRAR_BAdIRAI2 and calling the necessary methods inside it. Note that we won't have to make any changes to method ENRICH of BAdI RAI2. Instead, we just add an entry to the flow table for company code 3000 with the **Flow** value as **SWITCH**, and the method to be called is PROCESS_3000.

Let's take a look at the sample code. In [Figure 4.84](#), you can see that there is a SELECT statement that will get the data from the flow table matching the company code (bukrs) if the value of company code is 1000, Flow is set to SWITCH, and the calling program is IF_FARR_BADI_RAI2~ENRICH.

```
1) METHOD if_farr_badi_rai2-enrich.
2)   DATA: lv_meth          TYPE string,
3)         lv_class        TYPE string,
4)         lt_ptab         TYPE abap_parmbind_tab,
5)         lt_etab         TYPE abap_excpcbind_tab,
6)         lt_rainmsg       TYPE farr_tt_rai_msg,
7)         lt_zrarc_flow_mgmt TYPE zrarc_flow_mgmt_tt.
8)   DATA: exc_ref         TYPE REF TO cx_sy_dyn_call_error.
9)
10)  READ TABLE ct_rai2_mi INTO DATA(ls_mi_determine) INDEX 1.
11)  IF sy-subrc EQ 0.
12)    SELECT * FROM zrarc_flow_mgmt
13)    INTO TABLE lt_zrarc_flow_mgmt
14)    WHERE bukrs = ls_mi_determine-bukrs
15)    and FLOW = 'SWITCH'
16)    AND calling_cls_prg = 'IF_FARR_BADI_RAI2-ENRICH'.
17)  IF sy-subrc EQ 0.
18)    SORT lt_zrarc_flow_mgmt ASCENDING.
19)  ENDIF.
20)
21)  lt_ptab = VALUE #( ( name = 'IV_RAIC'
22)                    kind = cl_abap_objectdescr=>exporting
23)                    value = REF #( iv_raic ) )
24)                  ( name = 'CT_RAI2_MI'
25)                    kind = cl_abap_objectdescr=>changing
26)                    value = REF #( ct_rai2_mi ) )
27)                  ( name = 'CT_RAI2_CO'
28)                    kind = cl_abap_objectdescr=>changing
29)                    value = REF #( ct_rai2_co ) )
30)                  ( name = 'CT_MESSAGES'
31)                    kind = cl_abap_objectdescr=>changing
32)                    value = REF #( lt_rainmsg ) ).
33)  lt_etab = VALUE #( ( name = 'OTHERS' value = 4 ) ).
```

Figure 4.84 Dynamic Call of Methods

Then, the SELECT statement will get the entry from table ZRARC_FLOW_MGMT, as shown in [Figure 4.85](#).

CoCs	Send Comp.	Sec...	Contr...	Flow	calling cls prg	Class/Interface	Interface Component
1000	ZX	*	*	SWITCH	IF FARR_BADI_RA12=ENRICH	ZCLRAR_BADIRA12	PROCESS_1000

Figure 4.85 Switch Method Entry

The remaining part of the code shown in [Figure 4.86](#) is the syntax for the dynamic call. The code has the class name coming from table ZRARC_FLOW_MGMT and also the subsequent method name. They are passed in dynamic format, making the call generic. We just need to make sure that we follow the syntax along with the parameter as shown.

```

34 READ TABLE lt_zrarc_flow_mgmt INTO DATA(lt_zrarc_flow_mgmt) WITH KEY bukrs = ls_mi_determine-bukrs
35                                zrodoc_comp = ls_mi_determine-zrodoc_comp
36                                rloc = 'ENRICH'
37 IF sy-subrc EQ 0.
38     SWI.
39     CALL METHOD (lt_zrarc_flow_mgmt->classname) => (lt_zrarc_flow_mgmt->optname)
40     PARAMETER-TABLE
41     lt_pstab
42     EXCEPTION-TABLE
43     lt_etab.
44 IF lt_rainag[] IS NOT INITIAL.
45     APPEND LINES OF lt_rainag TO et_messages.
46     ENDF.
47 CATCH ok_sy_dyn_call_error INTO exc_ref.
48 MESSAGE exc_ref->get_text( ) TYPE 'I'.
49 ENDF.
50 ENDF.
51 ENDF.
52 ENDMETHOD.

```

Figure 4.86 Dynamic Call of Methods

Allow Methods

Allow methods have the value as ALLOW in the flow table. Refer to [Figure 4.81](#): when the company code equals 1000, then method PROCESS_1000 will be called. Method PROCESS_1000 will in turn call methods VALIDATE_A, PROCESS_B, and ENRICH_C.

During some point in the project, if the third party sending data will take care of validations, then we may have to remove method VALIDATE_A in the code. Then, the developer will be asked to comment the call to VALIDATE_A. If we could control this through the table rather than having to ask the

developer, that would occur through the ALLOW method concept of the flow table. To achieve this, we have to remove the entry from the flow table for method VALIDATE_A for company code 1000. Every method should have the code shown in [Listing 4.2](#) to check if the current method is allowed to be executed.

```
Call Method PROCESS_1000
Read entry from GT_ZRARC_FLOW_MGMT with company code = incoming company code
  FLOW = 'ALLOW'
                                CALLING_CLS_PRG = 'ZCLRAR_BAdIRAI2'
                                CLSNAME = 'ZCLRAR_BAdIRAI2'
                                CPDNAME = 'VALIDATE_A'.
    Check sy-subrc eq 0.
  Logic of the methods
EndMethod.
```

Listing 4.2 Sample Code for ALLOW Methods

In [Listing 4.2](#), you can see that there is a READ statement trying to fetch the entry for method VALIDATE_A from the flow table for company code 1000. Because it has been deleted, the sy-subrc that carries the result of the READ statement will have a value not equal to 0. This means the READ statement failed, as we've deleted the entry from the table for VALIDATE_A. The check statement will fail, and the code for VALIDATE_A won't be executed. This way, we exclude the execution of VALIDATE_A through the table.

We can control the calls to methods from the table this way without having to change the code, which is what we meant by “configuring the code.” The processing of methods will be controlled like a plug-and-play device. This is a demonstration of dynamic and flexible ways of achieving the required functionality through code. For the purposes of this book, we've focused on a specific RAR example, but this

type of coding isn't just limited to RAR and can be extended to any module.

4.7 Summary

As repeated many times in this book, the quality of results produced by RAR directly and fully depends on the quality of data submitted for processing. That is one of the reasons why SAP built architecture with the ARL as a staging area for both error handling and data transformation before it can reach RAR.

In addition, the BRFplus tool stores all business rules needed for both creation of RAR-relevant data and further processing and posting processes.

Both structures that are part of the ARL and BRFplus are made extensible and can be adapted to fit exact business requirements by the users. Whether you need to include some column in a process for POB determination or custom fields from a sales order in your RAI structures, these activities are achievable and can be done in a straightforward way. This once came with a cost. Instead of maintaining transparent tables, now users can learn how to use new tools such as BRFplus.

The RAIs that are created in the system are based on the data that is being sent from the sending system. But we need to tailor the RAIs per our requirements by prepopulating and validating. We explored the options that are available and how best to use them to meet the requirements. Ultimately, we want our contracts to be created correctly.

Once inbound processing is configured, we're ready to start working on contract management setup. So, let's move on to the next chapter where we're exploring contract configuration, creation, modification, and more.

5 Contract Management

This chapter gives detailed, step-by-step instructions for how to set up a key area of revenue accounting and reporting (RAR): contract management. We'll explain how certain settings can influence the overall process and what needs to be considered before certain decisions are made. In addition, we'll discuss common problems with contract management.

In this chapter, we'll explain how to configure contract management with instructions that are applicable to both classic contract management (CCM) and optimized contract management (OCM). We'll start by providing the general configuration steps for contracts and performance obligations (POBs). Then, we'll explain how to manage POBs and events, followed by walking through the contract modification process. To close, we'll discuss price allocation handling.

5.1 Setting Up Contract Management

After setting up inbound processing, the next step is setting up contract management. Similar to inbound processing, you must decide whether to go with OCM or CCM.

5.1.1 Optimized versus Classic Contract Management

OCM is the recommended solution because it represents the place where all future developments will be made. In addition, it has been significantly reworked and improved when compared to CCM. It has a set of features that weren't available in CCM, such as the following:

- Day-based contract modifications
- Freeze periods for time-based POBs
- Early termination of contracts
- Improved SAP Fiori apps and reporting based on core data services (CDS)

Certain limitations have been covered since SAP S/4HANA release 2021:

- Intercompany billing as fulfillment event
- Drop shipment as fulfillment event

- Manual price allocation with spreading
- Suspend revenue

Important Note

Results analysis still only works with CCM as of SAP S/4HANA release 2022.

When we say improved SAP Fiori apps, besides improvements that are visible as in better user experience apps, two very important apps were added that can make everyday work easier for end users: the Revenue Explanation app and Change History Display app. These two are particularly important because they help in an area that can be very time-consuming: determining the root cause of changes in revenue allocation for POBs.

To sum up, determining the way to set up contract management is one of the most important decisions for new RAR users on SAP S/4HANA. The option chosen depends mainly on the type of business scenario that needs to be covered. In some cases, users are bound to the classic version (e.g., in the professional services segment, where revenue needs to be calculated as a percentage of completion [POC]). In other cases, OCM offers many features that are very useful in most cases and can lower costs of potential developments (e.g., new event types).

When you start the configuration of revenue accounting by running Transaction FARR_IMG, if you're running a system on SAP S/4HANA after version 1709, you'll see the screen shown in [Figure 5.1](#).

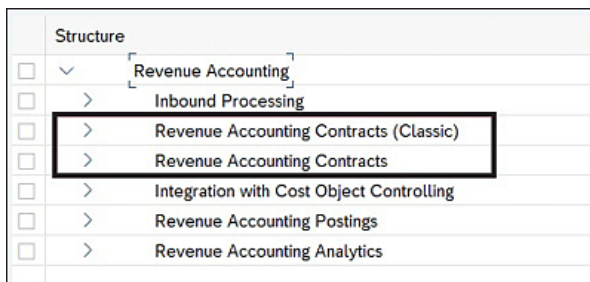


Figure 5.1 Configuration of Contract Management

If opting for CCM, select the **Revenue Accounting Contracts (Classic)** node; otherwise, you'll use OCM. Only one step is needed when determining whether to go with OCM or CCM. In the node for **Revenue Accounting Contracts**, there is one node extra named **Set Contract Management for Contract Categories**. Once you define the contract categories ([Section 5.1.8](#)), you'll go to that node and select what type of contract management will be used. As shown in [Figure 5.2](#), you can select the checkboxes under **Contract With CM Instead of CM Classic**

to indicate contract categories that are using OCM. Leave these checkboxes unselected if you want to use CCM.

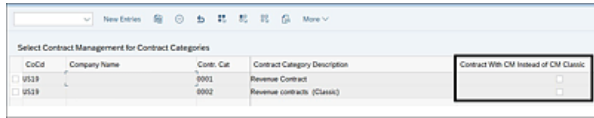


Figure 5.2 Selection of Contract Management Type

Technically, when you're using OCM, you'll see that contracts created in table FARR_D_CONTRACT have the **RAR version code** field populated with **X**, as shown in [Figure 5.3](#).

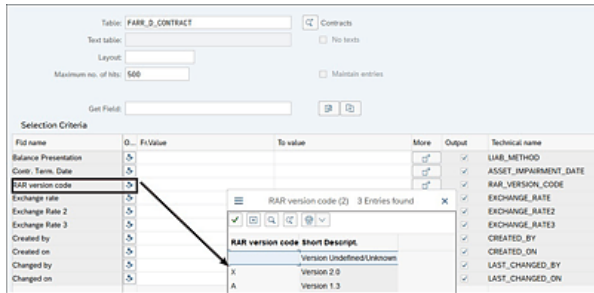


Figure 5.3 Table FARR_D_CONTRACT with Values for Optimized Contract Management

Table FARR_D_MAPPING will also have the version indicated in the same field. The system isn't stopping users from having multiple categories with different versions of contract management; this can be especially useful when it comes to migration, which gives a clear picture on which contracts are created under which approach (classic or optimized).

Note

Both options have the same nodes, so there is no difference in configuration steps that we'll discuss in this chapter. However, we'll use OCM in our example because that's the latest functionality from SAP.

OCM supports only the actual exchange rate method. The reason for this is that the focus for further product development has been shifted to the actual method due to increasing demand from customers.

You might wonder if you need to change the setup in the accounting principle. The answer is no. The first reason is that you can have contracts in both CCM and OCM active at the same time. This situation isn't so common with new, greenfield implementations, but it can occur with migrated environments. Second, the contract category setup is separate with no exchange rate method setting. So, if you select OCM as the preferred method for contract management, the actual

exchange rate method will be used, irrespective of the accounting principle settings.

5.1.2 Accounting Principles

This brings us to the first step in setting up contracts after selecting OCM in contract categories: setting up accounting principles. In this section, the following entries need to be made before contracts are created. Before we start explaining different settings, note that the settings are the same irrespective of whether we're talking about OCM or CCM. The only difference is that OCM is working only with actual exchange rate settings, so if you use OCM and make an entry for fixed rate in the accounting principle definition, that rate will be ignored, and the actual rate will be used instead.

To configure accounting principles, navigate to Transaction FARR_IMG, and follow menu path **Revenue Accounting Contracts • Configure Accounting Principle-Specific Settings**. The first setting you need to make is how many accounting principles you need. RAR supports reporting according to different accounting principles (e.g., International Financial Reporting Standards [IFRS] and local Generally Accepted Accounting Principles [GAAP]). In this case, if you need two reporting principles, you'll need to make two entries in this table: **IFRS** and **LG** (i.e., local GAAP). As a result, RAR will create two RAR contracts each time a document is saved from the source application.

In this case, there are parallel reporting requirements, so the setting looks like [Figure 5.4](#).



AccP	Name of Accounting Principle	Presentat
IFRS	International Financial Reporting Standards	2 Contract Liability/Contract Asset
LG	Local GAAP	1 Unbilled Receivable/Deferred Revenue

Figure 5.4 Setup of Parallel Accounting Principles

You see that for one accounting principle, **2 Contract Liability/Contract Asset** is used as the presentation, and, for another, **1 Unbilled Receivable/Deferred Revenue** is used. This kind of setup can be used in cases where revenue split (allocation) is needed for global reporting, but, for local reporting, the entity only requires presentation of deferred revenue. This kind of setup is common in multicountry scenarios.

As mentioned before, the impact of such a setting is that two RAR contracts are created each time a document from the source system is saved. This will be clearly visible in table FARR_D_MAPPING where you'll see two RAR contracts assigned for each source document, **30030824** and **300308099** (see [Figure 5.5](#)).

Search in Table: **FARL_D_MAPPING** Mapping Table to map source document to POBs of RA-Contracts

Number of hits: **6**

Runtime: **0** Maximum no. of hits: **100**

Insert Columns:

POB	Send Comp	Source Sys	SrcItem	Type	Source Item ID	POB Contract	Acct Ref	Type	Reference ID	RevAccCl	Header ID	SoftDelete	Arch.Date
71628	SD	SDOI	0041011458000020			300008024	IFRS	SDO	0041011458	SD01	41011458		
71629	SD	SDOI	0041011458000030			300008024	IFRS	SDO	0041011458	SD01	41011458		
71630	SD	SDOI	0041011458000040			300008024	IFRS	SDO	0041011458	SD01	41011458		
75347	SD	SDOI	0041011458000020			300008099	LG	SDO	0041011458	SD01	41011458		
75348	SD	SDOI	0041011458000030			300008099	LG	SDO	0041011458	SD01	41011458		
75349	SD	SDOI	0041011458000040			300008099	LG	SDO	0041011458	SD01	41011458		

Figure 5.5 Multiple Contracts Created for One Sales Order

Parallel Accounting Principles

If the entity needs to meet parallel reporting requirements, setting up additional accounting principles isn't the only activity that needs to be done. You can split the setup into two main areas: setup needed in RAR and setup needed in finance.

Before accounting principles can be used, they need to be defined in finance. You'll find the setting by following menu path **Parallel Accounting • Accounting Principles and Ledgers • Defined Accounting Principle**. Once the accounting principle is created, it can be used in RAR. Before setting it up, it's important to repeat that for each document, two contracts will be created, which means in all RAR-related tables, there will be two entries instead of one. Besides the impact on data space required by such a setup (only in high-volume environments), developments that potentially will use data from RAR tables are also impacted in that it's necessary to specify whether the source of information for the contract created is for IFRS or local GAAP.

In addition to this setting, there is an impact on settings needed for ledger groups so financial documents can be posted in the general ledger. Once an accounting principle is defined, it's necessary that one is assigned to the ledger group. Before it's assigned to the ledger group, you need to make sure that needed ledgers will be assigned to the ledger group. In other words, you define the accounting principle and assign it to ledger groups, which need to contain all ledgers to which posting needs to be made.

The last step is maintenance of BRFPplus tables. In both account and POB determination applications, you need to set up decision tables. These tables have accounting principle as one of the input parameters. This setup is done once; however, keep in mind that any future maintenance of the new account assignment objects or new POB types need to be done for both accounting principles.

To recap, parallel reporting is a standard feature of RAR that enables an entity to report at the same time both for global and local purposes. Before you opt for this feature, keep in mind that additional effort will be needed: potential impact

on developments, increased volume (because RAR will create two contracts for each operational document entered), and any ongoing maintenance that will need to include both accounting principles.

5.1.3 Presentation Methods

The next setting is one of the most important things that needs to be done when setting up RAR: choosing the presentation level. As you saw in [Figure 5.4](#), under the **Present.** column, the system has two options available: **1 Unbilled Receivable/Deferred Revenue**, which we'll refer to as UR/DR, and **2 Contract Liability/Contract Asset**, which we'll refer to as CA/CL. To explain this, we need to look at the definition of these terms by IFRS 15. According to the definition in IFRS 15, a contract asset becomes receivable at the moment when the entity's right to receive consideration becomes unconditional. So, the question is, when does the right to receive consideration become unconditional? The corresponding term in the old revenue recognition process, unbilled receivable, didn't have such a limitation: unbilled receivable is recognized revenue for which the entity still didn't send an invoice. Here, you can see that the main difference is the statement about unconditionality from the IFRS standard. In SAP, this difference is implemented in presentation methods: CA/CL will treat receivable as becoming unconditional at the moment the invoice becomes due, whereas UR/DR will look only at the invoice posting date as relevant.

We'll discuss the details around both presentation methods in the following sections.

Contract Assets/Contract Liabilities

Let's consider an example of the CA/CL presentation method with the following details:

- Contract: Selling goods where trigger for fulfillment is issuing of invoice
- POB 1: Event based; total allocated revenue of \$1,200
- Invoice: At month end; due date of 30 days net

At the end of the month, because this is a time-based POB, you'll need to recognize the appropriate revenue amount, which is allocated revenue/12 = \$100. The posting will look like [Figure 5.6](#).

RAR uses temporary accounts to balance posting of revenue to be recognized. The meaning of this account is similar to the meaning of the bank clearing or goods receipt/goods issue accounts used commonly in other SAP-related processes. The balance of this account can be a debit (if revenue is recognized,

but invoice is issued that isn't due) and, in that case, should be reported under assets. Or, the balance on the account can be a credit (if an invoice is issued that is still not due, but revenue isn't recognized) and, in that case, should be reported under liabilities.

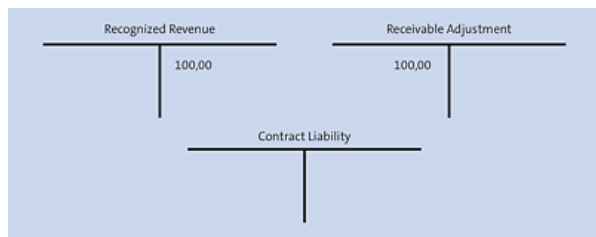


Figure 5.6 Recognized Revenue with the CA/CL Presentation Method

Once the invoice is due, only at that moment is the contract liability recorded, as shown in [Figure 5.7](#).

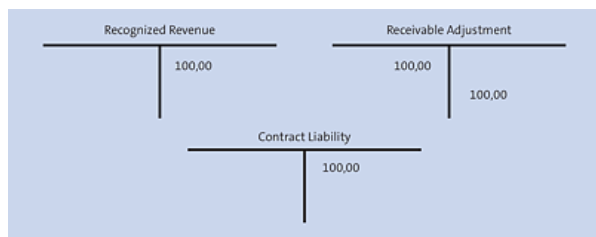


Figure 5.7 Posting of Contract Liability in the CA/CL Presentation Method

So once the invoice is due, the receivable adjustment account will be cleared against the contract liability. By using a reference to the due date in the invoice, consideration becomes unconditional and can be represented as a contract liability, thus it can be properly reported.

Unbilled Receivable/Deferred Revenue

When using UR/DR as the presentation method, deferred revenue simply represents a difference between the revenue recognized amount and what was billed to the customer. In this specific case, there is no timing difference between the moment when you recognized revenue (month end) and when you issued the invoice (also month end), so there is no deferred revenue posting happening at all. The posting is equal to the posting coming from billing the customer.

To make the difference clear between UR/DR and CA/CL, you need to repeat the key term—*unconditionality*—once more. Once an invoice satisfies all conditions to be treated as unconditional, it can be presented in the balance sheet as a contract asset or contract liability. In RAR, this unconditionality is determined by the due date value being filled in on an invoice.

Note

Neither Accounting Standards Codification (ASC) 606 or IFRS 15 mandates the usage of the terms *contract asset* and *contract liability*. Entities may use alternative descriptions if they provide sufficient information to distinguish those contracts where rights to receive consideration from the customer are conditional (contract assets) from those where the right to receive the consideration is unconditional (receivables), and whether an entity receives consideration from its customer or has the unconditional right to receive consideration in advance of performance (contract liability). However, a good practice is to adapt to new terms to avoid any confusion when comparing to the old reporting standard (this becomes especially important if an entity is using parallel reporting where CA/CL is leading and UR/DR is done for local reporting).

However, the situation which often happens in projects is that the entity can't (or doesn't need) to use the due date as a trigger for when a receivable becomes unconditional. An example of this situation is if a client has some guarantees that would make invoice payment certain even if the due date isn't reached. Or, simply using specific payment terms would make the due date field too confusing to be used as a criterion for recognizing contract assets. Even in that case, using UR/DR as the presentation method might not be suitable for two reasons:

- There are differences in calculation of UR/DR comparing to CA/CL (we'll cover these later in the text).
- If a customer has parallel reporting requirements and is using UR/DR for local reporting, the same method can't be assigned in two accounting principles due to system limitations.

In those cases, the best solution is to implement an enhancement so that the due date in the RAI is replaced with the invoice posting date, for example. For this enhancement, the best business add-ins (BAdIs) to use are `FARR_BADI_RAI2` (if you're not using the **Raw** status) or `FARR_BADI_RAI0` (if you're using **Raw** status). Run Transaction SE18, and select the proper BAdI in the **BAdI Name** field to be implemented (**FARR_BADI_RAI2**, in this case), as shown in [Figure 5.8](#).

Before you create an implementation, you first must create an implementation class that contains interface `IF_FARR_BADI_RAI2`. The most important method from this BAdI is method `ENRICH`, which is used to change data between the sender component and Adapter Reuse Layer (ARL). Here, you can introduce various checks or changes of data before they are processed in the ARL. However, there is a list of fields that can't be changed by this method, which can be found in structures `FARR_S_RAI2_MI_FIX` and `FARR_S_RAI2_CO_FIX`.

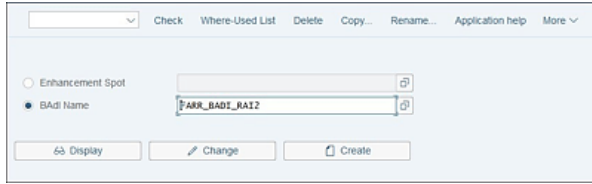


Figure 5.8 BAdI to Be Implemented

So, the first thing to do is create an implementation class and, within it, the method that you're going to use, as shown in [Figure 5.9](#).



Figure 5.9 Implementing Class with Methods to Be Used

Now, you need to write the logic in method `ENRICH`. The first thing to do is limit the logic to invoice items that will be created. So, for integration with sales and distribution, limit it to revenue accounting item (RAI) class `SD03`. The second part of the logic is to change the due date to be equal to the posting date, which is provided by the invoice, thereby bypassing the standard behavior of the CA/CL presentation method. Once this is done, the code should look like [Listing 5.1](#).

```

CONSTANTS : lc_raic TYPE FARR_RAIC VALUE 'SD03'.
DATA: ltr_header_id TYPE RANGE OF vbeln,
      lv_vbeln      TYPE vbeln,

"Taking data to temporary internal table
DATA(lt_rai2_tmp) = ct_rai2_mi.

CLEAR lv_vbeln.
LOOP AT lt_rai2_tmp ASSIGNING FIELD-SYMBOL(<ls_rai2_mi>).
  IF <ls_rai2_mi>-raic = lc_raic.
    <ls_rai2_mi>-due_date = <ls_rai2_mi>-posting_date.
  ENDIF.
  IF sy-subrc IS INITIAL.
    <ls_rai2_mi>-kunnr = <ls_vbak>-kunnr.
  ENDIF.
ENDLOOP.
ct_rai2_mi = CORRESPONDING #( lt_rai2_tmp MAPPING kunnr = kunnr
                             due_date = due_date ) .

UNASSIGN <ls_rai2_mi>.
CLEAR: lv_vbeln.
FREE : lt_rai2_tmp.
ENDMETHOD.

```

Listing 5.1 Code for Bypassing the Due Date

Once you activate this code by clicking the **Activate** button, you'll see the screen shown in [Figure 5.10](#), which displays that RAIs for invoice items have a due date equal to the posting date, enabling even the CA/CL presentation method to work based on some other criterion and not based on the due date.

Source Sys	Due Date	Posting Date	SrcmType	Source Item ID	Subarea	RevAccCl	Header ID	ItemID	Ref	Type	Ref ID	Init	Load
	01.10.2022	01.10.2022	SDI	080026029000010	518	SD03	800260202	10					
	01.10.2022	01.10.2022	SDI	080026030000010	176	SD03	800260300	10					
	01.10.2022	01.10.2022	SDI	0800260301000010	099	SD03	800260301	10					

Figure 5.10 Due Date Updated with Posting Date

Note

The example given is related to customers with classic inbound processing implemented. If you're using optimized inbound processing (OIP), the logic is different: data in RAR is created straight from source documents, bypassing the ARL. As mentioned, in OIP, there is no creation of application programming interfaces (APIs), tables, and classes depending on the source selected: all of these objects are predefined, meaning that a fixed database model is used, rather than a dynamically created one. Structures for sales and distribution, contract accounting, SAP Customer Relationship Management (SAP CRM), and third-party integration are all predefined.

Calculations

It's important to know how CA/CL is calculated, as well as the differences between UR/DR and CA/CL calculation. First, we need to note the formulas used to calculate CA/CL and UR/DR:

• CA/CL

- *Contract liability* = $MAX \{(Payment\ due - Fulfilled\ revenue), 0\}$
- *Contract asset* = $MAX \{(Fulfilled\ revenue - Receivable), 0\}$
- *Receivable* = $MAX \{Billable\ amount, Invoice\ due\ amount\}$
- *Billable amount* = $Original\ amount \times Fulfillment\ revenue \div Total\ allocated\ revenue$

• UR/DR

- *Unbilled revenue per performance obligation* = $MAX \{(Recognized\ revenue - Invoiced\ amount), 0\}$
- *Deferred revenue per performance obligation* = $MAX \{(Invoiced\ amount - Recognized\ revenue), 0\}$

Let's take the following example, starting with a CA/CL calculation. For simplicity, we'll exclude allocation, and there will be only one event-based POB:

- Revenue to be recognized: 48,608.13
- Invoiced to customer: 80,424.21, where due date = invoice date

According to the formula, you first need to calculate the billable amount:

- Billable amount = $80,424.21 \times 48,608.13 / \div 48,608.13 = 80,424.21$
- Receivable = $\text{MAX} \{80,424.21, 80,424.21\} = 80,424.21$
- Contract asset = $\text{MAX} \{(48,608.13 - 80,424.21), 0\} = 0$
- Contract liability = $\text{MAX} \{(80,424.21 - 48,608.13), 0\} = 31,816.08$

You can see that our billable amount is equal to the amount we’re invoicing, which again leads to calculating the same amount as receivable. When we put that information in the formula for contract asset, we get 0 because when we deduct receivable from fulfilled revenue, we get a value that is below 0, and the formula will return 0 in that case. Because the same amount is invoiced and due, the formula for contract liability will give us the difference between invoiced and recognized revenue as the contract liability.

To see how the contract liability is calculated, go to the table overview by running Transaction SE16/SE16n/SE11, and enter table “FARR_D_POSTING”. You’ll arrive at the screen shown in [Figure 5.11](#).

CoCd	Acc.	Reconcil. Key	POB	CnTy	Cate	D/C	Posting GUID	Year	Peri.	I TC Amount	Cnry
AU02	IFR.	20220120000101	119078	CL	H	000D3A2EDBC91E0DB3F54F2873BA3AA0	20.	12		31.816.08-	AUD
				CL						31.816.08-	AUD
AU02	IFR.	20220120000101	119078	ZFR0	K	000D3A2EDBC91E0DB3F53FA1D008B9F8	20.	12		80.424.21	AUD
				K						80.424.21	AUD
AU02	IFR.	20220120000101	119078	RA	S	000D3A2EDBC91E0DB3F54F2873BA3AA0	20.	12		31.816.08	AUD
AU02	IFR.	20220120000101	119078	ZFR0	H	000D3A2EDBC91E0DB3F53FA1D008B9F8	20.	12		80.424.21-	AUD
AU02	IFR.	20220120000101	119078	ZFR0	S	000D3A2EDBC91E0DB3F54C2F785FA85	20.	12		48.608.13	AUD
				RA						0.00	AUD
AU02	IFR.	20220120000101	119078	ZFR0	H	000D3A2EDBC91E0DB3F54C2F785FA85	20.	12		48.608.13-	AUD
				RV						0.00	AUD
										48.608.13-	AUD
										0.00	AUD

Figure 5.11 Contract Liability Calculation

Now, if we use same data to calculate UR/DR, we get the following results:

- Unbilled revenue per POB = $\text{MAX} \{(48,608.13 - 80,424.21), 0\} = 0$
- Deferred revenue per POB = $\text{MAX} \{(80,424.21 - 48,608.21), 0\} = 31,816.08$

Unbilled receivable returns 0, as expected: we billed more than the revenue we’re recognizing, so there is no unbilled part. The balance in this case is 31,816.08 as deferred revenue.

From this example, you could conclude that CA/CL and UR/DR would give the same results if you set the invoice date to be equal to the due date. However, that conclusion would be wrong. To analyze it further, we would need to look at additional example:

- Revenue to be recognized: 48,608.13
- Billing amount: 20,106.05

For simplicity, we’ll again consider billing amount = due amount, a single POB contract, and no allocation. Let’s take the preceding figures and put them into a UR/DR calculation:

- Unbilled revenue per POB = $\text{MAX} \{(48.608.13 - 20.106.05), 0\} = 28.508.02$
- Deferred revenue per POB = $\text{MAX} \{(20.106.05 - 48.608.13), 0\} = 0$

These results are as expected: Because we invoiced less than we're recognizing as revenue, we'll need to report that amount as an asset - which in this case is unbilled receivable - and this amount can be reported as such. Again, for the same reasons, we don't have any revenue to defer, which is exactly what our formula is giving us as a result.

Let's see what will be result of running same example with CA/CL method:

- Billable amount = $48,608.13 \times 48,608.13 / 48,608.13 = 48,608.13$
- Receivable = $\text{Max} \{48,608.13, 20,106.05\} = 48,608.13$
- Contract asset = $\text{Max} \{(48,608.13 - 48,608.13), 0\} = 0$
- Contract liability = $\text{Max} \{(20,106.05 - 48,608.13), 0\} = 0$

This is exactly the same as what you can see as a posting in RAR, as shown in [Figure 5.12](#).

CoCd	Acc.	Reconcl. Key	POB	CnTy	Cate.	D/C	Posting GUID	Year	Peri.	TG Amount
AU02	IFR	20220120000101	119079	ZPRD	IC	S	00003A2EDBC91EDD0B3F57100BF54FB99	20..	12	20,106.05
					IC					20,106.05
AU02	IFR	20220120000101	119079	ZPRD	RA	H	00003A2EDBC91EDD0B3F57100BF54FB99	20..	12	20,106.05-
AU02	IFR	20220120000101	119079	ZPRD	RA	S	00003A2EDBC91EDD0B3F57C1293ECBC3F	20..	12	48,608.13
					RA					28,508.02
AU02	IFR	20220120000101	119079	ZPRD	RV	H	00003A2EDBC91EDD0B3F57C1293ECBC3F	20..	12	48,608.13-
					RV					48,608.13-
										0.00

Figure 5.12 Contract Asset Calculation

The billable amount is the amount that that can be billed. Now, there might be a difference between billed amount and billable amount. If, for example, by billing plan, you can bill \$1,000, but you actually bill only \$800, the first one is considered the billable amount and the latter the billed amount.

The billable amount is equal to the revenue that can be recognized, which is why both receivable and billable amounts are the same. However, when you put them into the formula for contract assets, you see that amount becomes zero. The reason for this is that according to the concept of *conservatism* (all probable losses are recorded when they are discovered, while gains can only be registered when they are fully realized), you're still not registering that receivable as a contract asset.

CA/CL versus UR/DR

This example was brought up because organizations need to be very clear in what they expect from the calculation engine regarding both methods of revenue recognition, as well as how balance sheet postings will be made. In our example, we wanted to illustrate how (based on exact cases) amounts given by

both UR/DR and CA/CL can be the same, but in another case, using a very similar business scenario, they are dramatically different.

Therefore, we strongly recommend that business scenarios are created and verified with an auditor before the implementation of RAR. By doing so, you can avoid making expensive and lengthy changes on an already set system.

5.1.4 Calculation Methods for Local Currency

The next step is to choose a calculation method for local currency. In the accounting principle settings, to the right of the **Present.** column, you can find the **LC Calc. Methods** column. The system supports two main approaches regarding contracts in foreign currency, which need to be translated:

- **Fixed exchange rate method**

The revenue in local currencies is recognized at a fixed rate when the first event comes for the whole contract.

- **Actual exchange rate method**

The revenue accounting contract has a deferred revenue or contract liability balance, and revenue is recognized at the average rate of historical liability; if the revenue contract has no liability balance, revenue will be recognized at the spot rate when you execute the Transfer Revenue program (program A).

Note that if you select OCM, the actual exchange rate method for calculation will be applied irrespective of the selection made.

When revenue accounting creates a contract, it specifies a company code currency or transaction currency, which is typically a document currency of the operational document such as a sales order. The transaction currency is used to allocate the transaction price to POBs, recognize revenues, and post revenues to the general ledger.

In a foreign currency contract, the contract currency is different from the company code currency. When transferring revenue postings to the general ledger, revenue accounting supports multiple currencies that are predefined for the company code as local currencies. For each company code, you can define a maximum of three parallel currencies:

- The first local currency, which is also the company code currency, is defined in the company code master data.
- The second and third local currencies are defined in the additional local currencies data of the company code using Transaction OB22.

By default, revenue accounting calculates the amount in transaction currency and the first local currency (company code currency) and generates financial

accounting documents accordingly. However, you need to decide whether the second and third local currencies will be calculated in revenue accounting through the following steps:

1. Define the second and third local currencies in Transaction OB22.
2. Exclude additional currencies by following menu path **Revenue Accounting • Revenue Accounting Contracts • Exclude Local Currency from Calculation**. If you're using multiple currencies in the system (e.g., multiple local currencies or group currencies), those will be excluded from foreign exchange (FX) calculations.

As a requirement to use contracts in foreign currency, you need to make two settings. First is setting which condition type will be used for FX differences in table FARR_D_POSTING. Follow menu path **Revenue Accounting • Revenue Accounting Contracts • Condition Types • Define Reserved Condition Types**. In [Figure 5.13](#), you can see different cases that need to have reserved condition types assigned for allocation difference, right of return (ROR), exchange rate difference, and—in the case of OCM—early termination. The prerequisite for this activity is the creation of condition types, and, in this transaction, you can assign them per your requirements.

Reserved Condition Types	
Allocation Difference:	CORR
Right-of-Return Revenue Adjustment:	RROR
Right-of-Return Cost Adjustment:	CROR
Exchange Rate Difference:	EXRD
Early Termination:	

Figure 5.13 Reserved Condition Types

The second setting that needs to be done is account assignment. For FX rate differences account assignment, there is no separate setting in RAR nor BRFPplus decision tables; rather, RAR uses standard settings from SAP S/4HANA. So, you'd need to go to make necessary settings together with other general ledger accounts that are relevant for calculation of FX differences, which is part of standard SAP S/4HANA and beyond the scope of this book.

Once the contract has exchange rate differences, you'll see them on the contract level as a separate item in table FARR_D_POSTING with posting category **ED**, as shown in [Figure 5.14](#).

DocC	Acc. Reconcil Key	POB	City	Calc	D/C	Posting GUD	Year	Per.	I	TC Amount	City	LC Amount	LC Curr	Second LC	LC Cur
A.	IFR_2022010000102	100041	EXRD	ED	H	00003A4425061EEDCA8D28F0231286C03	20.	10		0.00	AUD	0.00	AUD	1,225.00	USD
															0.00
AU02	IFR_2022090000101	100041	2P90	IC	S	00003A2A80E81EED0F8E89F98120F4C4	20.	8		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022090000102	100041	2P90	S	S	00003A2A80E81EED0F8E89F98120F4C4	20.	9		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022010000102	100041	2P90	S	S	00003A4425061EEDCA8D28F0231286C03	20.	10		100,000.00	AUD	100,000.00	AUD	69,517.00	USD
															200,000.00
AU02	IFR_2022090000101	100041	2P90	RA	H	00003A2A80E81EED0F8E89F98120F4C4	20.	8		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022090000101	100041	2P90	S	S	00003A2C8C3EED0F8E89F98120F4C4	20.	9		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022090000102	100041	2P90	H	H	00003A2A80E81EED0F8E89F98120F4C4	20.	9		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022010000101	100041	2P90	S	S	00003A2E08C3EED0F8E89F98120F4C4	20.	10		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022010000102	100041	2P90	S	S	00003A4425061EEDCA8D28F0231286C03	20.	10		100,000.00	AUD	100,000.00	AUD	69,517.00	USD
AU02	IFR_2022010000102	100041	2P90	H	H	00003A4425061EEDCA8D28F0231286C03	20.	10		100,000.00	AUD	100,000.00	AUD	69,517.00	USD
AU02	IFR_2022010000101	100041	2P90	S	S	00003A80E3E1EEDCA8D28F0231286C03	20.	11		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
															0.00
AU02	IFR_2022090000101	100041	2P90	RV	H	00003A2E08C3EED0F8E89F98120F4C4	20.	9		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022010000101	100041	2P90	H	H	00003A2E08C3EED0F8E89F98120F4C4	20.	10		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
AU02	IFR_2022010000101	100041	2P90	H	H	00003A80E3E1EEDCA8D28F0231286C03	20.	11		100,000.00	AUD	100,000.00	AUD	68,292.00	USD
															200,000.00
															0.00

Figure 5.14 Contract with FX Differences

Now, when it comes to the actual exchange rate method, it's a good idea to look at the formula to understand fully the results that SAP will give.

Revenue accounting handles the historical rate as a weighted average rate. It's calculated by the following formula:

$$\text{Weighted average rate} = \frac{\text{Sum of historical amounts in local currency}}{\text{Sum of historical amounts in transaction currency}}$$

Now let's take an example, as shown in [Table 5.1](#).

Date	FX Rate	Transaction Currency Amount	Local Currency Amount
01.01	1.15	1,000.00 EUR	1,150.00 USD
01.02	1.25	2,000.00 EUR	2,500.00 USD
01.03	1.30	3,000.00 EUR	3,900.00 USD

Table 5.1 Example of Different Exchange Rates

Applying the preceding formula, you get the following:

$$\text{Weighted average rate} = \frac{(1,150.00 + 2,500.00 + 3,900.00)}{(1,000.00 + 2,000.00 + 3,000.00)} = \frac{7,550.00}{6,000.00} = 1.2583$$

The Transfer Revenue program calculates a foreign currency into local currencies as follows:

- If the cumulative invoiced amount or invoice due amount exceeds recognized revenue up to the execution of this program, then the revenue will be recognized by using the average exchange rate of the exceeding part of the cumulative invoice (invoice due amount).
- If the cumulative recognized revenue exceeds the cumulative invoice (invoice due amount) up to the execution of this program, then the exceeding part of revenue will be recognized using the exchange rate of the current date in this period when the program is executed. If the period you specify is in the past, then the last day of the period is used to determine the exchange rate.

The Transfer Revenue program is covered in detail in [Chapter 6, Section 6.2.1](#).

5.1.5 Contract Modifications

The next setting is related to contract modifications. According to IFRS 15 Revenue from Contracts with Customers, a contract modification is a change in the scope or price (or both) of a contract that is approved by the parties to the contract. When the system performs contract modification, the modification only applies to partially fulfilled POBs. We'll cover modification scenarios in detail in [Section 5.4](#). If a customer is expecting to get contracts that will be modified, it's important to make the appropriate setting in the accounting principle to enable them by selecting the **Enable Contract Modification** checkbox.

5.1.6 Cost Recognition

The **Cost Recognition** checkbox is a setting that enables cost recognition in addition to revenue recognition. Once a contract quantity has been executed, canceled, or settled, expenses or revenue related to the quantity must be recognized in the financial accounts of the firm. According to the matching principle of revenue accounting, costs of goods sold (COGS) must be posted at the same time as the corresponding revenue. We'll talk about cost recognition when we cover POB settings and corresponding event types in [Section 5.3](#).

5.1.7 Contract Assets and Liabilities

The last two settings in the accounting principle are reserved for additional setup related to calculation of CA/CL (refer to [Section 5.1.3](#)). The first setting is related to calculation of contract assets on the POB or contract level, and the second setting is related to how the calculated amount will be netted, as shown in [Figure 5.15](#).



Figure 5.15 CA/CL on the Performance Obligations Level

When the system uses calculation of CA/CL on the POB level, the formula used is slightly different:

- $Contract\ liability = Max \{(Payment\ due - Fulfilled\ revenue), 0\}$
- $Contract\ asset = Max \{(Fulfilled\ revenue - Receivable), 0\}$
- $Receivable = Max \{Billable\ amount, Invoice\ due\ amount\}$
- $Billable\ amount = Original\ amount \times Fulfillment\ revenue \div Total\ allocated\ revenue$

For our example, as shown in [Table 5.2](#), we'll use a more complicated case with multiple POBs and allocation.

POB Type	POBs	Transactional Price	SSP	Allocated	Quantity	Per Piece
Reagent	118940	632.24	166.54	33.1	2	16.66
Reagent	118941	2,179.98	856.47	171.31	3	57.10
Reagent	118942	295.60	75.64	15.13	4	3.78
Reagent	118943	369.50	95.00	19.00	5	3.80
Reagent	118944	1,798.50	654.54	130.92	6	21.82
Instrument	118945	0	24,528.30	4,906.15	1	204.42
Total		5,275.82	26,376.49	5,275.82		

Table 5.2 Example for CA/CL on POB Level Calculation

We have five different reagents and one instrument (see [Chapter 1](#) for more information on these terms). Based on the quantity of reagents, you also have an amount calculated of allocated revenue per piece. The instrument is leased, and the duration is 24 months (meaning the instrument is a time-based POB).

In this case, we're billing just reagents, and billing is done as follows:

- 118940 1 pc: 316.12
- 118941 2 pcs: 1,453.32
- 118944 3 pcs: 899.25

To evaluate how CA/CL is calculated, you need to go through every step of the process:

1. Processing of fulfillment RAIs

This step doesn't cause any change in table FARR_D_POSTING, so no additional explanation is needed. These fulfilments are later the source for revenue that will be recognized.

2. Processing of invoice RAIs

We're getting reversal of invoicing (IC posting category) and as offset account results analysis posting category, as shown in [Figure 5.16](#). Both postings are done with condition type (ZP01), which was used as a condition type in sales order.

After processing invoice RAIs, all amounts which came from billing are summed up to zero (invoice revenue and accounts receivable is zero).

IC	118940	ZP01	316,20	0	Billing Revenue	118940	ZP01	316,12
	118941	ZP01	1453,32			118941	ZP01	-1453,32
	118944	ZP01	899,25			118942	ZP01	-899,25
RA	118940	ZP01	-316,12		Receivable	118940		316,12
	118941	ZP01	-1453,32			118941		1453,32
	118944	ZP01	-899,25			118942		899,25

	Billable Amount	Revenue Recognized	Due	CL	CA
118940	316,20	16,66	316,12	299,46	0,00
118941	1453,32	114,21	1453,32	1339,11	0,00
118942	0,00	0,00	0	0,00	0,00
118943	0,00	0,00	0	0,00	0,00
118944	899,25	65,46	899,25	833,79	0,00
118945	0,00	408,85	0	0,00	408,85
SUM	2668,77	605,17	2668,69	2472,36	408,85

Figure 5.16 Posting after Processing of Invoice RAIs

3. Running a revenue transfer (program A)

The Revenue Transfer program calculates revenue that should be recognized (based on fulfillment events, either goods issue or time) and posts it against the results analysis account. The results analysis account serves as a clearing account for recognized revenue. It's posted on the condition type level.

4. Calculate CA/CL on the POB level

The calculation of CA/CL is done on the POB level through several steps, as shown in [Table 5.3](#):

- Calculate billable amount for each POB
- Calculate invoiced due amount (in our case = invoiced amount)
- Calculate the contract liability with the following formula:
Contract liability = MAX ((Invoiced amount - Fulfilled revenue), 0)
- Calculate the contract asset with the following formula:
Contract asset = MAX ((Fulfilled revenue - Billable), 0)

POBs	Billable Amount	Recognized Revenue	Due	CL	CA
118940	316.20	16.66	316.12	299.46	0.00
118941	1,453.32	114.21	1,453.32	1,339.11	0.00
118942	0.00	0.00	0	0.00	0.00
118943	0.00	0.00	0	0.00	0.00
118944	899.25	65.46	899.25	833.79	0.00
118945	0.00	408.85	0	0.00	408.85
Total	2,668.77	605.17	2,668.69	2,472.36	408.85

Table 5.3 Results of Calculating CA/CL on the POB Level

Here you can see that we're getting results that are going in different directions: POBs that were billed are in the contract liability, whereas the POB that isn't invoiced is in the contract asset. This makes perfect sense because that POB (instrument) has revenue allocation but isn't invoiced. So, in this step, in the moment before the calculation is finished, you see that some POBs are in CL status, and one is in CA status.

5. CL netting

According to IFRS 15, contracts can have either a contract asset or contract liability. Therefore, it makes perfect sense to net the amount on the contract level but have only contract liability or contract asset. Again, that is done in several steps:

- Calculate total CA/CL on the contract level.
- Spread it to POBs according to their standalone selling price (SSP) ratio, which is calculated automatically by comparing the total SSP with the SSP of each POB.

In this case, the balance is on the CL side, with $2,472.36 - 408.85 = 2,063.51$. Spreading will be done as shown in [Table 5.4](#).

POBs	SSP	SSP Ratio	Total CL	CL on the POB Level
118940	166.54	0.63		13.03
118941	856.47	3.25		67.00
118942	75.64	0.29		5.92
118943	95.00	0.36		7.43
118944	654.54	2.48		51.21
118945	24,528.30	92.99		1,918.92
Total	26,376.49	100.00	2,063.51	

Table 5.4 Calculation of CL on the POB Level

Because the total result on the contract level was contract liability, portions of it were being assigned to all the POBs. So, the calculation of CA/CL on the POB level can lead sometimes to unexpected results: here you see that the POB, which you would expect to have contract asset (revenue without billing), actually has CA calculated in one of the steps and therefore ends with its own portion of contract liability.

CA/CL on the Contract Level versus the POB Level

We wanted to illustrate in this section the implications if you choose the POB level over calculation on the contract level. Calculation on the contract level is much easier because it's more predictable in the sense of verifying results against some external engine or calculation tool. However, there are some valid reasons that still might push an organization in the direction of calculating CA/CL on the POB level. There might be a requirement to also present the balance sheet side on the POB level type (disaggregation of CA/CL in the same way as in revenue) or provide some additional account assignments on the balance sheet too (e.g., a work breakdown structure [WBS] element that wouldn't be available otherwise).

Taking this into consideration, an organization should be clear why it's opting for presentation on the POB level before it makes such a choice so they'll know what to expect as the result of the run and avoid future expenses related to changing the presentation level.

After completing the setup of the presentation level of CA/CL, you're done with the accounting principle settings.

5.1.8 Assign Company Codes, Number Ranges, and Contract Categories

Next, you need to assign company codes to the accounting principle. You can access the **Supported Company Codes per Accounting Principle** activity below the creation of accounting principle activity, as shown in [Figure 5.17](#).

Depending on reporting requirements, the company code (**CoCd**) can be assigned to one or two accounting principles (if parallel reporting is needed). To use the company code productively, **Status** has to be set to **PO Productive**. Other statuses are used for migration purposes. To set the status to be PO, it has to first be saved as migration. Statuses can be set back (from productive to migration), which can help with deletion of faulty data.

Supported Company Codes per Accounting Principle									
AccP	CoCd	Transf.Dat	Status	Adopt Date	Src.Acc.Py	Est.Acc.Py	End.D.Usng.	App	Est.DP
<input type="checkbox"/>	IFRS	A002	30.06.2022	PO Productive	01.07.2022	IFRS			
<input type="checkbox"/>	IFRS	0804	31.10.2020	PO Productive	01.11.2020	IFRS			
<input type="checkbox"/>	IFRS	FR02	31.10.2022	PO Productive	01.11.2022	IFRS			
<input type="checkbox"/>	IFRS	0804	31.10.2022	PO Productive	01.11.2022	IFRS			
<input type="checkbox"/>	IFRS	ID02	30.06.2022	PO Productive	01.11.2022	IFRS			
<input type="checkbox"/>	IFRS	J002	30.06.2022	PO Productive	01.11.2022	IFRS			
<input type="checkbox"/>	IFRS	NL03	31.10.2022	PO Productive	01.11.2022	IFRS			
<input type="checkbox"/>	IFRS	US19	31.10.2020	PO Productive	01.11.2020	IFRS			
<input type="checkbox"/>	IFRS	Z002	30.06.2022	PO Productive	01.11.2022	IFRS			
<input type="checkbox"/>	IFRS	Z003	30.06.2022	PO Productive	01.11.2022	IFRS			

Figure 5.17 Assignment of Company Codes

It's also important is to set the number ranges for the contracts, POBs, and run IDs. This is done in **Revenue Accounting • Revenue Accounting Contracts • Number Ranges**. In theory, you can choose between external and internal numbering. External numbering is when a number needs to be provided to the

system manually or from an external system, and internal numbering is when a number gets determined and assigned by the system itself (internally). While external numbering might be appealing, especially in situations of integration with external systems, this way of numbering brings a lot of challenges and, in some cases, can't be used.

When starting this setup, as shown in [Figure 5.18](#), select the **Change Intervals** button (with the pencil), and then create the number range (two-digit identifier). The from-to number range is then assigned.

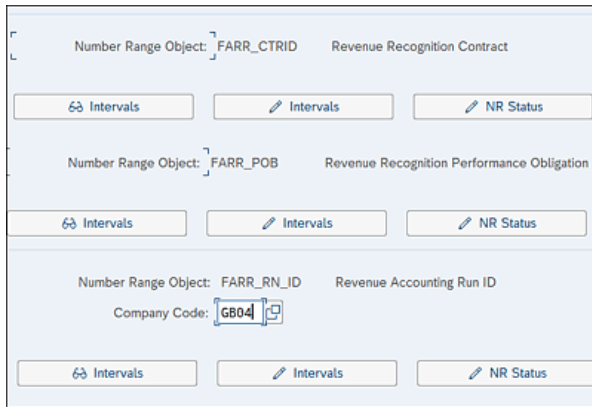


Figure 5.18 Setup of Numbering Objects

The system uses three different number range objects that need to be maintained. Once you enter the maintenance screen shown in [Figure 5.19](#) by selecting the **Maintain** option, you need to enter numbers from and to (**From No.** and **To Number**), which is very similar to setting the numbering in any other area of finance. In our example, you can see that values are maintained as internal and see the current value for the first number range.

No	From No.	To Number	NR Status	Ext
01	00000000100000	00000000199999	102399	<input type="checkbox"/>
02	00000000200000	00000000299999	0	<input type="checkbox"/>
03	00000000700000	00000000799999	0	<input type="checkbox"/>
04	00000000600000	00000000699999	0	<input type="checkbox"/>
05	00000000300000	00000000399999	0	<input type="checkbox"/>

Figure 5.19 Number Ranges for Contracts and POBs

Run IDs are reference numbers that the system uses to track background jobs created for revenue postings. Numbering of run IDs is used for creating new revenue posting jobs (run IDs). You can define only one number range for run IDs and must use an ID of 01. The system doesn't use number ranges other than 01. Run IDs are set in the context of individual company codes, and each run ID is unique in the scope of a company code.

When creating a run ID, if the system doesn't find number range 01, it automatically creates a number range 01 that spans from 00000001 to 99999999. If you've defined other number ranges that overlap the system default

range, the system can't create the default number range, and no new POBs can be created.

Once you set up the number ranges, you need to set up the contract categories. You can define contract categories in the **Contr. Cat** field and assign number ranges to them in the **No. Range** field (see [Figure 5.20](#)). Each contract category is associated with a number range that is used for creating new contracts of that category.

Customizing of Contract Category		
Contr. Cat	Description	No. Range
<input type="checkbox"/> 0001	Revenue Contract	01
<input type="checkbox"/> 0002	Revenue contracts (Classic)	02

Figure 5.20 Contract Categories

Once the contract category is set, you need to assign it to the company code (as described in [Section 5.1.1](#)) and decide whether to use CCM or OCM.

5.1.9 Condition Types

To access data properly, you need to define reserved condition types for RAR only. To do so, navigate to **Revenue Accounting • Revenue Accounting Contracts • Condition Types**.

Important Note

Condition type definition comes from the sales and distribution system, but settings there play no role in RAR.

Once you access this activity, the following groups are available and can be maintained:

- **Allocation effect**

This condition type carries differentials that result from price allocation. When allocating transaction prices for a contract, the system aggregates all pricing conditions and then allocates the total amount among the POBs in the contract. The system then represents the allocated prices on a differential basis. Without changing the original pricing conditions, the system adds a special condition type that carries the differential resulting from the allocation. For example, if a POB with an original price of EUR 15 is allocated EUR 20, the system represents the allocation result as an original price of EUR 15 and an allocation effect of EUR 5.

- **ROR revenue adjustment**

This condition type carries the amount of revenue that must be deferred to be recognized due to a ROR held by the customer.

- **ROR cost adjustment**

This condition type carries the amount of cost that must be deferred to be recognized due to a ROR held by the customer.

- **Early termination expense (only in case of OCM)**

This condition type carries the balance amount of all related assets and liabilities such as receivables, CA/CL, or UR/DR if a contract was terminated early.

5.1.10 Posting Periods

Once all setup is completed, you can make periods open for processing. The starting period must be an open posting period as defined in the **Open and Close Posting Periods** setting in Customizing for financial accounting. However, depending on the message control Customizing, you can control whether violation of this requirement is disallowed or only triggers a warning message. The revenue accounting period that you try to open or close can't be earlier than the latest transition period (the period that includes the legacy data transfer date). The reconciliation keys for the current and previous revenue accounting periods must be closed, and all contract liabilities and assets must be calculated and posted.

Accountants can open or close an accounting period for revenue accounting. Alternatively, when accountants perform revenue postings, they can choose to close the revenue accounting period for which the postings are performed.

There isn't an option for setting the status to **Closed** in Customizing. If you want to set a period to **Closed**, you need to set the status to **Open** from the next period. For example, if you want to set period 4 to **Closed** for fiscal year 2023, you need to open the period from period 5.2023.

To access this setting (see [Figure 5.21](#)), you can search for the maintenance view of table FARR_C_CLOSE; run Transaction FARR_IMG and follow menu path **Revenue Accounting Contracts • Open and Close Revenue Accounting Periods**, or run Transaction FARR_PERIOD_IN_CLOSING.

Revenue Accounting Period Close					
CoCd	AccP	Fr. FYear	Fr. Period	Status	
<input type="checkbox"/>	AU02	IFRS	2022	11	Open
<input type="checkbox"/>	CO02	IFRS	2020	11	Open
<input type="checkbox"/>	FR02	IFRS	2022	11	Open
<input type="checkbox"/>	GB04	IFRS	2022	11	Open
<input type="checkbox"/>	ID02	IFRS	2022	11	Open
<input type="checkbox"/>	JP02	IFRS	2022	11	Open
<input type="checkbox"/>	NL03	IFRS	2022	11	Open
<input type="checkbox"/>	US19	IFRS	2020	11	Open
<input type="checkbox"/>	ZA02	IFRS	2022	11	Open
<input type="checkbox"/>	ZA03	IFRS	2022	11	Open

Figure 5.21 Periods Management in RAR

The following statuses are available:

- **Open**

All transactions can generate reconciliation keys in the current period. This refers to transfer revenue, contract liability calculation, and reconciliation keys coming from new transactions.

- **In Closing**

New transactions will be posted in the new period, but running ABC programs still can be done for the “old” period.

- **Closed**

Period is closed for any changes, and all changes must be done in the new period.

5.2 Setting Up Performance Obligations

Let's start from the beginning: What is a POB? For a definition, you need to look at ASC 606 or IFRS 15. As defined in IFRS 15, paragraph 22:

“A performance obligation is a promise to provide a distinct good or service or a series of distinct goods or services as defined by the revenue standard. At contract inception, an entity shall assess the goods or services promised in a contract with a customer and shall identify as a performance obligation each promise to transfer to the customer either:

- *A good or service (or a bundle of goods or services) that is distinct.*
- *A series of distinct goods or services that are substantially the same and that have the same pattern of transfer to the customer.”*

Here, you already see some very important points: POBs need to be defined already at the moment of contract inception, and POBs must be distinct. The point about defining POBs at the moment of contract inception is clear—when you sign the contract with the customer, POBs are what's in the contract, goods, or services the customer is buying. This is also the first watchpoint for you: you need to take a detailed look at documents such as framework agreements or quantity/value contracts to determine if they are relevant for IFRS 15 or, in this particular case, whether you can identify POBs at this stage. In general, the answer is that the contract needs to have a clear definition of not only enforceability but also about what the

customer is buying if you want to say it's relevant as an IFRS 15 contract.

Regarding treating POB as distinct, paragraph 26 of IFRS 15 reads, "Goods and services that aren't distinct are bundled with other goods or services in the contract until a bundle of goods or services that is distinct is created." The bundle of goods or services in that case is a single POB. Essentially, if goods and services are bundled to create one deliverable to the customer, components are treated as nondistinct POBs because the customer can't benefit from them separately. In that case, they can be bundled into one POB, which will be included as a POB in a contract.

Let's consider an example that is commonly used in organizations whose line of business is project delivery (software and system integration companies). Here, you deliver services to customers, for example, in software development, consulting, extended maintenance, or managed services, together with some hardware such as computers, data centers, and so on. In addition, there might be some spare parts and even some additional costs (e.g., traveling) included as part of the project.

In this case, it's questionable whether the customer benefits from buying each of these items separately because the customer signed a contract related to delivery of the whole project, not just parts of it. So according to statements from the IFRS 15 standard, each separate POB (service, hardware, and spare parts) would be treated as nondistinct POBs that are bundled in one distinct POB. This bundle would be later treated as a unique item, which would need to be presented as a compound POB.

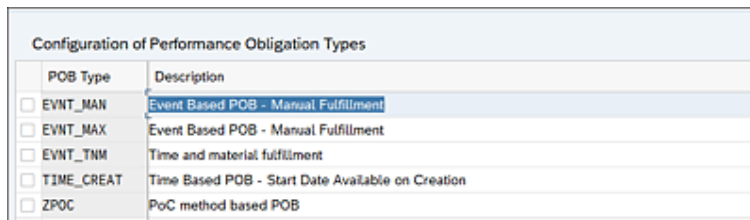
If items in a contract with your customer aren't distinct items, they can't be posted separately for revenue recognition. These

items must be combined with other nondistinct items to form a distinct POB. The POB that results from this combination is called a *compound POB*, and the entire structure is called a *compound structure*.

In the following sections, we'll explain how to set up POBs. We'll begin with the basic creation of POB types.

5.2.1 Create Performance Obligation Types

Let's first see what kind of customizing is needed to create POBs. The first step is to define POB types in Transaction FARR_IMG by following path **Revenue Accounting Contracts** • **Define Performance Obligation Types**. Once you select this option, the screen shown in [Figure 5.22](#) will appear. We'll discuss the different POB type options in more detail in [Section 5.3](#).



POB Type	Description
<input type="checkbox"/> EVNT_MAN	Event Based POB - Manual Fulfillment
<input type="checkbox"/> EVNT_MAX	Event Based POB - Manual Fulfillment
<input type="checkbox"/> EVNT_TNM	Time and material fulfillment
<input type="checkbox"/> TIME_CREAT	Time Based POB - Start Date Available on Creation
<input type="checkbox"/> ZPOC	PoC method based POB

Figure 5.22 POB Types Definition Initial Screen

Here you need to enter data by clicking either the **New Entries** or **Copy** buttons. If you're creating new POB types, you'll arrive at the screen shown in [Figure 5.23](#). In this example, we selected POB type **EVNT_MAN** and clicked the **Copy** button to create a new one.

Figure 5.23 POB Types Entering Details

The screen for creating POB types is split into three subscreens: **General Data**, **Fulfillment Data**, and **Allocation Data**. In the **General Data** screen, you define the POB name in the **Perf.Obligat.Name** field. This data can be very useful as an additional revenue reporting dimension.

Usage of POB Type and POB Name

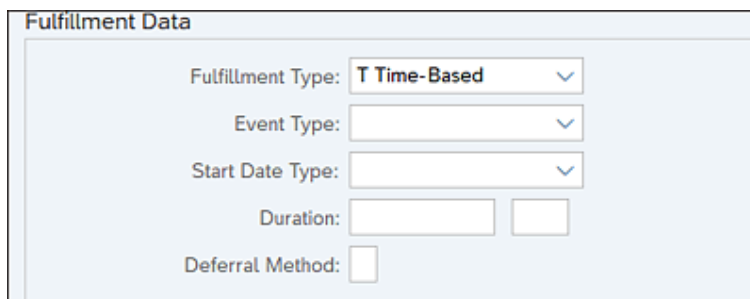
Companies usually need to perform revenue reporting on different levels. For example, the first level comprises main revenue groups for reporting (e.g., device sales, service sales), and the next level can be the detailed level where the first one is further broken down (e.g., main devices, peripherals, support services, maintenance services, etc.). The classic approach is to create it as separate general ledger account and thus separate POB types; however, that would make the number of POBs large and therefore hard to manage.

One alternative is to have major groups created as POB types and lower levels as POB names. This enables a leaner structure of POBs, fewer general ledger accounts, and still detailed reporting. The only difference you need to have in mind depending on whether you're working on some older or newer SAP S/4HANA versions of RAR: Newer versions use new SAP Fiori apps where all that needs to be done is adding new

fields and CDS views pick them up automatically as dimensions in revenue reports. In older versions, you need to work with extension of table FARR_D_POSTING to get proper fields available. See [Chapter 6](#) for more information on reporting.

Besides defining the POB name, there's also the option to select the **No Cost Recognition** checkbox. This setting needs to be used in accordance with the setting on the accounting principle level: if you set that cost recognition is possible, it makes sense to control on the POB type whether cost recognition is allowed or not through this checkbox. In addition, it also applies the other way around: if you choose that cost recognition won't be allowed, it makes sense not to change the value of this field.

The next section, **Fulfillment Data**, is about determining the fulfillment type of the POB and fulfillment trigger for event-based POBs. The system provides several standard types for fulfillment, or you can create new ones if necessary. If you select that the **Fulfillment Type** is **T Time-Based**, the screen will look like [Figure 5.24](#).



The screenshot shows a form titled "Fulfillment Data" with the following fields:

- Fulfillment Type: T Time-Based (dropdown menu)
- Event Type: (dropdown menu)
- Start Date Type: (dropdown menu)
- Duration: (two input fields)
- Deferral Method: (checkbox)

Figure 5.24 Time-Based POB Type

Details about different fulfillment and event types will be covered in [Section 5.3](#).

In the **Allocation Data** section, you define how the POB will be processed with the allocation engine. The first step is to determine the relevancy of POBs on allocation. If the **Excl. from**

Alloc. checkbox isn't selected, the allocated price of the POB is always brought over from the conditions that are specified in the corresponding operational item. No additional value is allocated to or taken away from this POB.

Let's look at examples of allocation if the POB is and isn't excluded from allocation. In [Table 5.5](#), both POBs are included in allocation, which means that revenue will be spread to all of them according to the provided SSP. Here, you see that because the SSP of both POBs is equal, they will take the same portion of the revenue even if their transactional prices are different.

POB	Transactional Price	SSP	Allocation Percentage	Allocation Amount
POB A	1,000	2,000	50%	1,500
POB B	2,000	2,000	50%	1,500
Total	3,000	4,000		

Table 5.5 Allocation Example

In our next example shown in [Table 5.6](#), there's one POB excluded from allocation. The effect of excluding it from allocation is the same as providing SSP = 0. This means its allocation percentage will be 0, which means the allocated revenue was also zero. However, the total revenue allocated to the first POB is equal to the amount included in both POBs. In other words, if the POB is excluded from allocation, any transactional price given to this POB will still be included in the total contract transactional price, but POB being excluded from allocation won't carry any revenue for itself.

POB	Transactional Price	SSP	Allocation Percentage	Allocated Amount
POB A	1,000	2,000	100%	3,000
POB B	2,000	0	0%	0
Total	3,000	2,000		

Table 5.6 Excluded from Allocation Example

Excluded from Allocation: Practical Usage

When do you use these POBs? The easiest way to explain this is the example of any kind of fee that the customer might pay but for which the customer isn't getting any benefit. For example, activation fees for mobile contracts are one-time fees that don't represent any specific service the customer uses and can be represented as a POB excluded from allocation (there are cases when this is bundled together with a basic service, but that's not the question here).

Some goods or services that the customer is paying for and using, but for which there is no contractual obligation, is another example. Because they are part of the contract itself, you need to include them, but not in allocation because there is no obligation to use them.

Let's return to the POB type settings shown previously in [Figure 5.23](#). The **Residual** option is used when you want to apply the residual calculation method of SSP for a particular POB. Again, it would be useful to look at the IFRS 15 standard about the residual method to be applied.

The residual approach involves deducting from the total transaction price the sum of the estimated SSPs of other goods and services in the contract to estimate the SSP for the remaining goods or services. Use of a residual approach to estimate the SSP is permitted in certain circumstances. A residual approach should only be used when the reporting entity sells the same good or service to different customers for a broad range of prices, making them highly variable, or when the reporting entity hasn't yet established a price for a good or service because it hasn't been previously sold. This might be more common for sales of intellectual property or other intangible assets than for sales of tangible goods or services.

The circumstances where the residual approach can be used are intentionally limited. Management should consider whether another method provides a reasonable estimate of the SSP before using the residual approach.

As you can see, situations when the residual approach might be used are extremely limited, but it can be useful. First, let's check the prerequisites for using the residual approach:

- Out of all the POBs in a contract, only one can be set as residual.
- No SSP should be defined for that residual POB.
- SSPs must be defined for all other POBs.
- If the sum of SSPs is higher than the allocatable amount, that POB will get $SSP = 0$. This situation should be always well thought through because the business scenario in which the residual approach is used isn't valid in that case ($SSP >$ transactional price).

Let's check the residual method through an example where a company sells a bundle product for 1,000 EUR, as shown in [Table 5.7](#).

POB	Sales Order Item	SSP	Quantity	Allocated Amount
POB 1	Product A	10 EUR	20	$10 \times 20 = 200$ EUR
POB 2	Product B	20 EUR	30	$20 \times 30 = 600$ EUR
POB 3	Product C	None	1	$1,000 - (200 + 600) = 200$ EUR

Table 5.7 Residual Method Calculation

You can see that the SSP (and therefore allocation too) assigned to a POB will be calculated as a difference between the total allocated price and allocated price assigned to other POBs from the contract. As mentioned, if this method is used, the POB needs to be properly marked with the **Residual** checkbox, but more importantly, the entity should check whether they have the business case for this method of SSP determination.

When can you use this residual approach? In the telco industry, it's not so uncommon to sell premium phone numbers. In those cases, the customer is paying extra just to have a number that is simple or that has some meaning to the client. In those cases, if you're bundling service and device with this premium number, you'll get a total transactional price much higher than the SSP of items that are known (service and device). However, the premium number doesn't have a specific SSP, so the residual approach provides the SSP to the number by the difference between the total transactional price and the sum of SSPs given to the device and service.

In some rare cases, you can define the SSP already on the POB type level because POB types usually are defined on some

higher level than materials, for example. In addition, materials are so different that a single SSP wouldn't really be correct. If this is the case, you can use the fields available in [Figure 5.23](#), shown earlier.

What deserves special attention is the **SSP Tolerance** field. Tolerance defines how much the transaction price can deviate from its SSP before triggering a price allocation. Sometimes, the transaction prices of your sold items aren't exactly the same as, but are very close to, their SSPs. In this case, your items are sold almost at fair value, and therefore you don't want to perform price allocation on this contract. The SSP tolerance allows you to specify a range, instead of a single value, of SSP. The system performs price allocation only when at least one of the items in the contract is sold at a price beyond its specified price range.

There are two fields to define tolerances: one defines the tolerance in amount (**SSP Tolerance**), and one defines the tolerance in percentage (**SSP Tol. Perc.**).

In the example shown in [Table 5.8](#), you see that for POB 2 and POB 3, price is within the specified range, but for POB 1, it's outside the range. This means that allocation will be performed on this contract.

POB	Transaction Price	SSP	SSP Tolerance	SSP Tolerance Percentage	Price within Specified Range?
POB 1	17 EUR	20 EUR	2 EUR	N/A	No
POB 2	20 EUR	20 EUR	2 EUR	N/A	Yes

POB	Transaction Price	SSP	SSP Tolerance	SSP Tolerance Percentage	Price within Specified Range?
POB 3	32 EUR	30 EUR	3 EUR	N/A	Yes

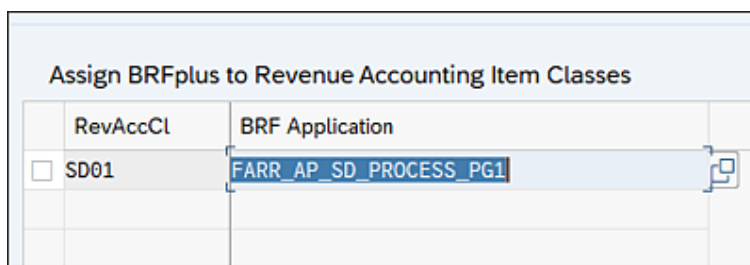
Table 5.8 SSP Tolerances

Once these settings are filled in, click the **Execute** button to complete the basic setup of POB types.

5.2.2 Determine POB Types in BRFplus

Now, let's move on to the next step: determination of POBs based on certain rules. This determination is done in BRFplus, and you first need to define which applications will be used. In this case, applications are being assigned to item classes without a predecessor (meaning no additional setup is needed for fulfillment or invoice items).

Execute Transaction FARR_IMG, and follow menu path **Inbound Processing • Revenue Accounting Item Management • Assign BRFplus Applications to Revenue Accounting Item Classes**. You'll arrive at the screen shown in [Figure 5.25](#), where you can assign the **BRF Application** that will be used for integration between the integration component and RAR.



RevAccCl	BRF Application
<input type="checkbox"/> SD01	FARR_AP_SD_PROCESS_PG1

Figure 5.25 BRFplus Assignment of Application

SAP delivers the following BRFplus application templates:

- **FARR_AP_SD_PROCESS_TEMPLATE**
For RAI classes used for integration with sales and distribution.
- **FARR_AP_CRM_PROCESS_TEMPLATE**
For RAI classes used for the CRM service integration.
- **FARR_AP_CA_PROCESS_TEMPLATE**
For RAI classes used for the integration with contract accounts receivable and payable.
- **FARR_AP_PROCESS_TEMPLATE**
For RAI classes used for the integration with any other sender component.

Before assigning the application, you first need to copy the template application into the customer one and maintain decision tables. In this case, we'll focus on sales and distribution integration, but the process doesn't differ compared to other integrations.

Warning

There is one generic input structure maintained per RAI class for accounting principle-independent and accounting principle-dependent functions. This generic input structure contains certain accounting principle-dependent attributes that aren't filled when the BRFplus functions that are accounting principle-independent are executed (e.g., change indicator, controlling object number for results analysis integration, integration type with controlling results analysis, and the accounting principle itself). Including these listed attributes, which are accounting principle dependent, into the ruleset of BRFplus functions that are executed more than once per accounting principle will either have no effect on the

result data or even lead to unexpected results data of the BRFplus function.

You can access BRFplus by executing Transaction BRFPLUS via the screen shown in [Figure 5.26](#).

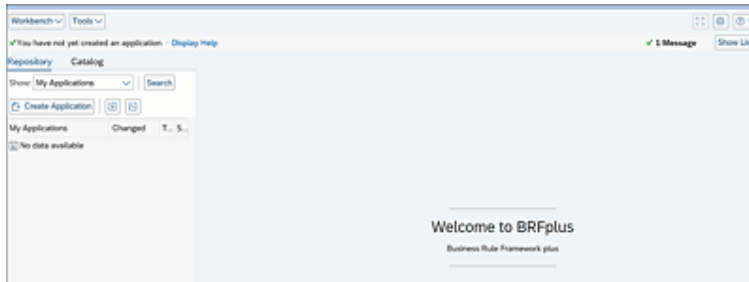


Figure 5.26 BRFplus Initial Screen

On the left side, the **Repository** appears where you'll perform all your selections. Before you do so, it's highly recommended to change to expert mode in BRFplus. To do so, select the **Personalize** option in upper-right part of the screen, and the screen shown in [Figure 5.27](#) will appear.

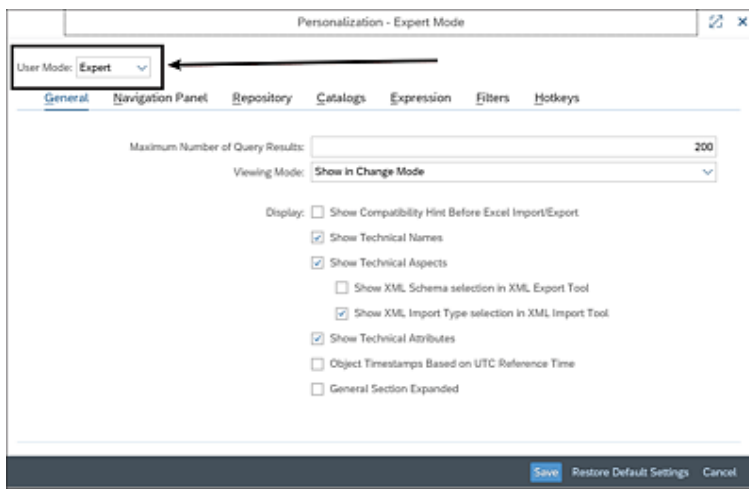


Figure 5.27 Expert Mode Selected

Basically, **Simple** mode won't allow you to do some additional operations (and you won't see technical details of decision tables), and **Expert** mode will allow you to perform all the

available operations. You can see the difference between the modes by working on both modes.

Now you can select the application. On the left-hand side, click on the **Search** button near the **My Applications** dropdown list (refer to [Figure 5.26](#)). The screen shown in [Figure 5.28](#) will pop up.

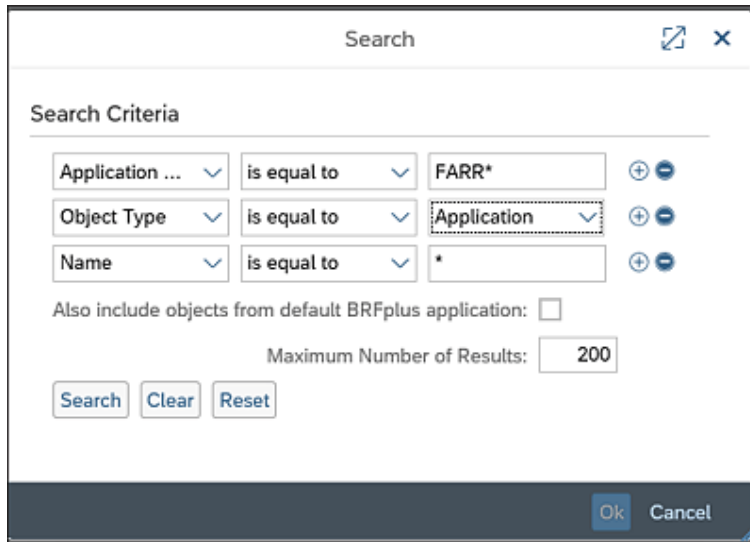


Figure 5.28 BRFplus Selection

Enter “FARR*” (or it can be Z*, Y*, or some customer namespace if selected) for **Application**. The important thing is to select **Application** for **Object Type** before you press . Once done, you can click the **Search** button, and the system lists all applications that fit your selection, as shown in [Figure 5.29](#). Here you can see the date of the last change, the transport status, and whether the document is in active status or needs to be activated.

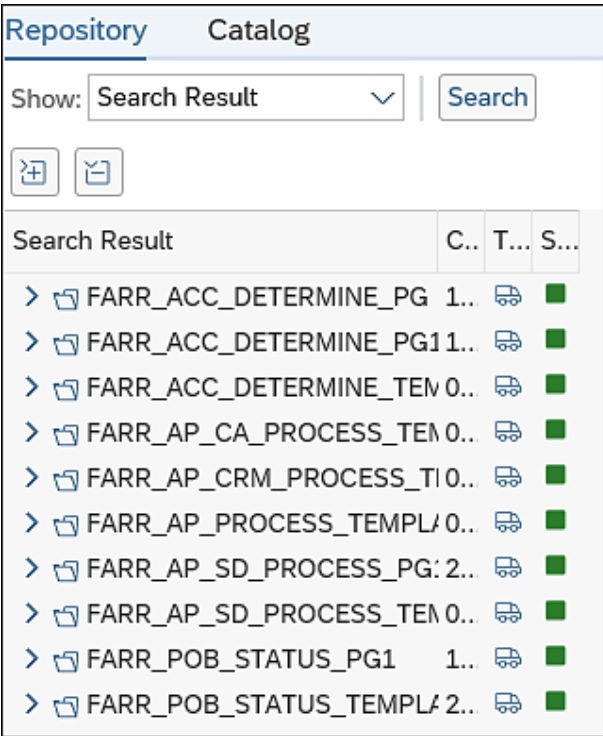


Figure 5.29 BRFplus Applications

The next step is that you open the application you need to modify and find the decision table, which is needed for POB type determination. To find the decision table, go to **Expression • Decision Table** folder, as shown in [Figure 5.30](#). For POB type determination, you search for table **DT_PROCESS_POB**. Click on the central part of the screen to see how POB type determination is done. Gray columns are used as input parameters, and green are result columns showing successful selection.

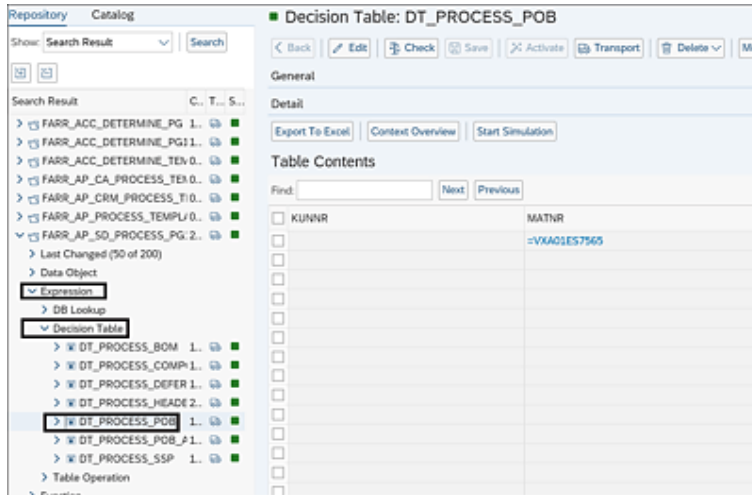


Figure 5.30 BRFplus Decision Table

To make new entries, there are two main methods. The first is manually creating entries, where you must make the table editable by clicking the **Edit** button, as shown in [Figure 5.31](#). The table will become editable, and you can enter, copy, or change all entries inside. As an example, we added new material that will be used as source material for POB type determination.

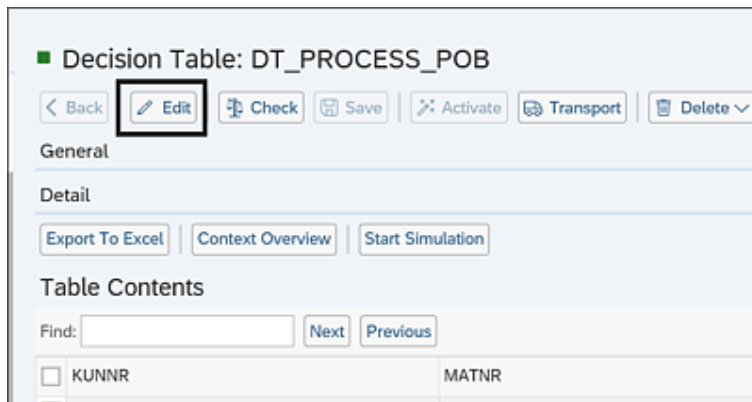


Figure 5.31 Editing a Decision Table

However, manual editing isn't the best case if you have many entries to be made. In those situations, you might look at the mass option, which is done as an integration with Microsoft Excel. You'll see an option to **Export to Excel** for each decision

table. Once you click it, the system will give you a table in Microsoft Excel format, as shown in [Figure 5.32](#).

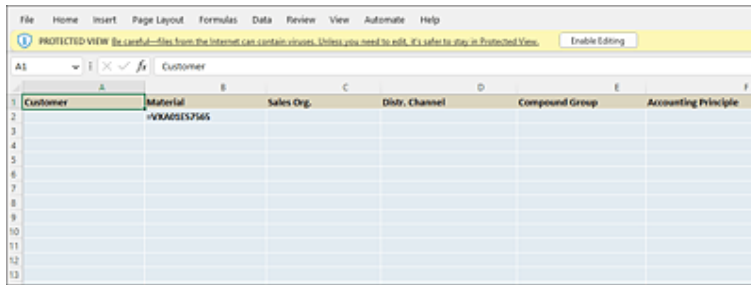


Figure 5.32 Export to Microsoft Excel from BRFplus

Once data is in Microsoft Excel, it can be modified en masse and saved. As an example, you could deliver this Microsoft Excel file as a template to populate new materials determination. When the changes are complete, you can upload the same Microsoft Excel file back to BRFplus by clicking the **Upload** button.

When you make a change, it's always good to check whether it does what you intended it to do. For this kind of check, click the **Simulate** option, and the system will ask for input parameters. When finished, click **Execute** to get results to verify. In the example shown in [Figure 5.33](#), before you load the Microsoft Excel file, make sure derivation will work as expected. Populate the data in the screen (in our example, update the **Simulation Data** with the new material [MATNR], and click the **Execute and Display Processing Steps** button for the simulation to run).

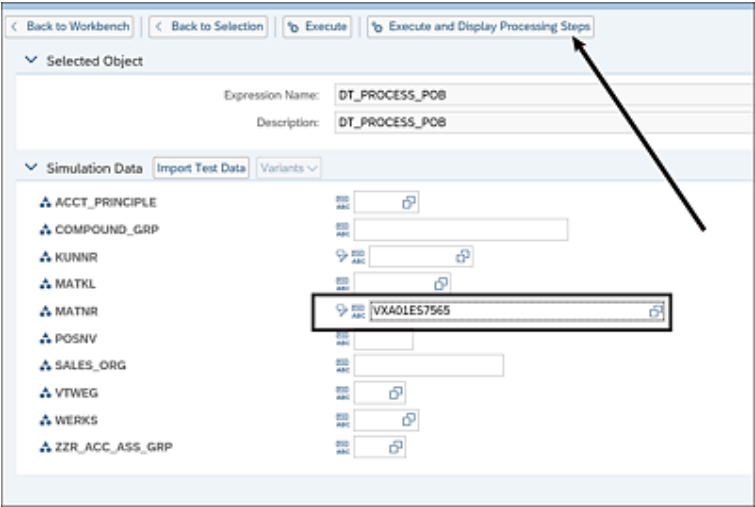


Figure 5.33 Execution of Simulation

After simulation, you can check the results, as shown in [Figure 5.34](#). Here you can see that we're getting the POB type with a specific name as a result, which is a time-based POB without any specifics defined further. This functionality of BRFplus is widely used to limit efforts when it comes to testing new changes.

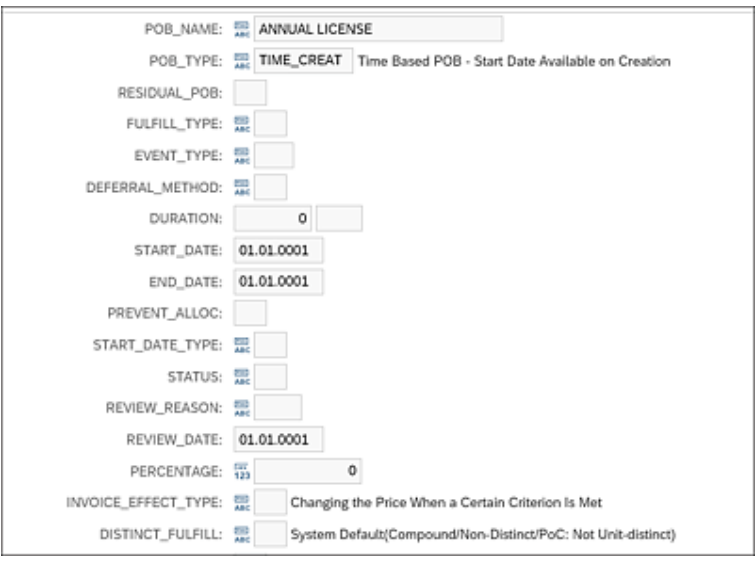


Figure 5.34 Results of Simulation

Once you're happy with the testing and changes, you need to activate the application by clicking the **Activate** button and

transport it by clicking **Transport Application**.

BRFplus Transport

Be very careful when transporting BRFplus applications. Similar to any other ABAP object, applications can be created as local, so users might think that maintaining them separately on each system is a quick way to finish setup. However, each application has a unique identifier, and maintenance in different systems would make further transport impossible. Therefore, the only valid approach is to have BRFplus maintained on one master client and transport it further.

In addition, when maintaining BRFplus, you might notice that input columns don't fit our needs. For example, in integration with sales and distribution, the default application is to have the POB type determined based on the material number or material group. Materials might not be most convenient because each time you create a new material, the decision table needs to be maintained. However, for some clients, determination of POB types can't be done on the level of material group, or, for example, you want to have another criterion to determine POB names.

To enable this, it's very easy to add additional columns as selection parameters and make determinations based on them. However, because you usually need some columns that aren't available by standard, the first step is to add the element you want to make available as a selection. For this to happen, you need to make sure that the structure used by the application contains that field.

After running Transaction BRFPLUS and choosing the correct application, you can navigate to **Data Object • Structure**, as

shown in [Figure 5.35](#). Here, select the structure behind the decision table in which you need to see changes.

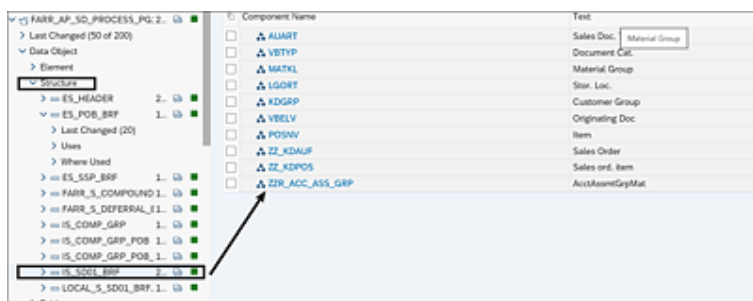


Figure 5.35 Adding a Field to the Structure

Because you're working with sales and distribution integration, the structure you need to edit is **IS_SD01_BRF**. Open it in edit mode, and add the needed field (**ZZ_R_ACC_ASS_GRP**, in our example). Once you have the structure ready, the next step is to make the field available in the decision table. Go to the needed decision table (table **DT_PROCESS_POB** in our example), and make sure that your table is opened in **Edit** mode. In the general view, select the **Table Settings** button, as shown in [Figure 5.36](#).

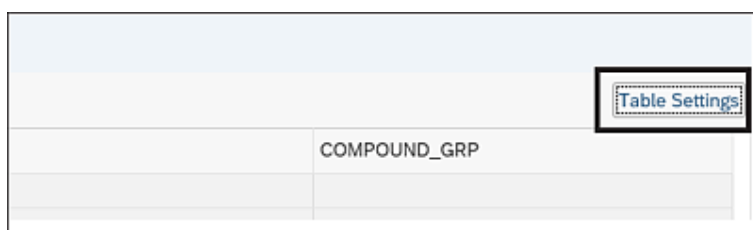


Figure 5.36 Table Settings for BRFplus

After that, the system will show you a list of already selected items in columns, as shown in [Figure 5.37](#). You can add new columns by clicking the **Add Column** button in table settings.

A new field has now been added as a selection, so you can use the account assignment group from sales and distribution (VBAP-KTGRM) for selection of POBs. In general, adding new columns in the selection is a very elegant way of getting what you need

from the system—and this is done with minimal knowledge of ABAP programming.

Result Data Object: ES_POB_BRF

Table Check Settings

Overlap Check Settings: Application Default

Completeness Check Settings: Application Default

List of Columns

Condition Columns

Column Name	Text	Mandatory Input	Column Accessibility
<input type="radio"/> KUNNR	KUNNR	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> MATNR	MATNR	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> SALES_ORG	SALES_ORG	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> VTWEG	VTWEG	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> COMPOUND_GRP	COMPOUND_GRP	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> ACCT_PRINCIPLE	ACCT_PRINCIPLE	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> WERKS	WERKS	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> MATKL	MATKL	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/> POSNV	POSNV	<input type="checkbox"/>	Full Access (Changes Allowe
<input checked="" type="radio"/> ZZR_ACC_ASS_GRP	ZZR_ACC_ASS_GRP	<input type="checkbox"/>	Full Access (Changes Allowe

Result Columns

Figure 5.37 New Field as Selection Criterion

When you look at applications for processing RAIs, you’ll notice that several more decision tables exist, as shown in [Figure 5.38](#). This is because the application is used for processing all data needed for the creation of contracts and the creation of POBs with all needed details.

- √ FARR_AP_SD_PROCESS_PG: 2..
- > Last Changed (50 of 200)
- > Data Object
- √ Expression
 - > DB Lookup
 - √ Decision Table
 - > DT_PROCESS_BOM 1..
 - > DT_PROCESS_COMPI 1..
 - > DT_PROCESS_DEFER 1..
 - > DT_PROCESS_HEADE 2..
 - > DT_PROCESS_POB 1..
 - > DT_PROCESS_POB_A 1..
 - > DT_PROCESS_SSP 1..
 - > Table Operation

Figure 5.38 Decision Tables Used in Contract and POB Creation

Following is list of a few tables and what they are used for:

- **Table DT_PROCESS_BOM**

Bill of materials (BOM) is a term that appears both in sales and production processes. A sales BOM is used for sales documents where the parent item is listed as a sales item, not an inventory item. When the parent is selected in the sales document, all the children appear as subitems automatically. For example, a company is selling a computer that consists of a tower, keyboard, monitor, mouse, and some other items. On the sales order, you want to show only Computer X, while for inventory, you need to make sure all components are selected. For this to work, we'll create a BOM where the header will be the computer and the components will be the subitems. In RAR, a distinct BOM structure can consist of multiple levels, and there is no restriction on the number of low-level POBs. Costs are always recognized on the lowest POB level.

When you look at table `FARR_D_POB`, you can recognize members of the BOM with the **Is Part of a BOM** field selected; **Composition** shows **Distinct**; and **Root POB in BOM** shows the POB ID of the high-level POB. SSPs are required for low-level POBs, except those that are defined as residual allocation or exclude from allocation.

In this decision table, you'll find all the information needed for the creation of BOM POBs in RAR. Note that BOM structures aren't supported in OCM. If a customer uses BOMs, they would need to look at either the option of creating all POBs as distinct or creating a compound group (a hierarchical list of POBs) instead.

- **Table DT_PROCESS_COMPOUND**

This application is used when defining a group is needed. The functionality is very similar to a BOM with having one header item and nondistinct POBs but with one difference: compound

POBs can use only time-based or POC as methods for fulfillment. If the customer requires some other method of fulfillment, BAdI FARR_BADI_COMPOUND_FULFILLMENT can be used in CCM to adjust fulfillment to fit the customer's needs.

- **Table DT_PROCESS_SSP**

If you want to maintain SSPs as a form of price list, this decision table can be used. Here, customers can identify parameters based on which SSP will be retrieved for a particular combination of material/sales organization/customer/and so on. Besides doing this, it's also possible to use this decision table as a safety point: if an SSP isn't being sent by a sender application, the SSP can be retrieved from here. However, if you're thinking to use BRFplus as a main source from where SSPs will be retrieved, there are certain limitations related to the number of expected entries in this table. In practice, we saw situations where customers used this decision table as a main source for SSP retrieval, and the number of entries grew with time (common situation if validity dates are introduced). There is a technical limitation that will cause performance issues if the number of rows in this table is more than 10,000. In those cases, think about either using some other option for SSP determination or stepping away from BRFplus as a tool for keeping them.

- **Table DT_PROCESS_POB_ADD**

This table is used to link POBs to be defined. Linked POBs are implicit POBs that aren't stated directly in the sales order, but it's understood by parties that they will be delivered. For example, if a customer purchases equipment, it's understood that they will get a warranty with it too. These additional POBs can be defined as linked.

5.3 Managing Performance Obligations and Event Types

Once you define POB types, you need to look into details about fulfillment and event types. By definition, all POBs can be grouped into three categories when it comes to their fulfillment type:

- **Time-based POBs**

POBs whose revenue is recognized over a period of time.

- **Event-based POBs**

POBs whose revenue is recognized at a specific point in time.

- **Percentage of completion (POC) POBs**

This is a special category of POB. Whereas there's usually a specific event or simple time passage that triggers revenue recognition, for a POC POB, RAR fully relies on the percentage calculated by results analysis and applies it as fulfillment.

To understand what needs to be configured, you need to look at the standard again to see how revenue can be recognized.

To clarify that POBs can be satisfied at a point in time, certain criteria need to be met. The first is that revenue can't be recognized over time, and the second is that the customer actually took control over an asset. Other indicators are as follows:

- Customer presented right to payment
- Customer has legal title over POB
- Customer is in physical possession
- Customer has risks and benefits over ownership of goods
- Customer accepted the asset

The preceding are actually indicators, not criteria, which means that the company needs to assess them to determine if the customer accepted control over an asset.

Here you see that revenue recognition is done at one point in time when one or more of those indicators are met. These events are represented as different *event types* when you define a POB type as event based.

Another type of recognition is time based (or over time). One of the most critical decisions for revenue recognition is the determination of whether revenue will be recognized over time because this decision influences how revenue will be reported for the future. In this case, however, there's a set of criteria where if just one of them is met, revenue can be recognized over time:

- Customer receives benefits and consumes them at the same time. Examples for this criterion fulfill most of the services that are delivered during a fixed amount of time.
- Vendor is making enhancements over an asset the customer controls while it's being enhanced. For example, there is a contract to make improvements to a product during a fixed period during which the customer is using an asset.

- Vendor performance doesn't create an asset with alternative use, and the vendor has the enforceable right to payment for performance completed to date.

What you can see here is that the entity needs to go with over time revenue recognition if the customer receives and consumes POBs. This would suit entities operating in service industries where the customer subscribes and consumes service over a certain period of time. In addition, the option to recognize revenue over time would fit different POC methods because time isn't the main criterion in those cases; instead, it's the ways you measure progress of POB usage and transfer of control.

Deciding how revenue will be recognized is one of the most important tasks and thus needs to be looked at in detail because differences between IAS 18/ASC 605 and IFRS 15/ASC 606 can be significant in terms of how revenue is treated and recognized.

We'll explain how to manage time-based POBs in the over time revenue recognition process and event-based POBs in the point-in-time revenue recognition process in the following sections.

5.3.1 Over Time Revenue Recognition

In this section, we'll look into the details of how revenue is recognized over time, or the effect of setting POBs to be time based. Let's look at a classic example of a bundle that contains one device and service, as shown in [Table 5.9](#). The device will be recognized as point-in-time revenue, and the service will be defined as a time-based POB because the customer can consume it over a certain period of time.

POB	Transaction Price	SSP	Duration	Allocated Revenue	Recognized	Monthly
Device	1	500		413.97	413.97	
Service	2,400	2,400	12	1,987.03	165.59	165.59

Table 5.9 Time-Based Revenue Recognition

In this case, you see that the device is being given to the customer almost for free to sell the service to the customer. The device is heavily discounted, so the customer signs a contract with obligation to use the service. Because customers will simultaneously consume and pay for this service, criteria are met for it to be recognized as over time revenue. So, the monthly revenue that will be recognized is equal to 165.59. If billing the customer monthly in equal amounts, the revenue that will be recognized is less than the amount billed to the customer (200.00 compared to 165.59). This will be discussed in detail in [Section 5.5](#). Month by month, the revenue getting recognized will look like [Table 5.10](#).

Jan.	Feb.	March	April	May	June	July	Aug.	Sept.	Oct.	Nov.
165.59	165.59	165.59	165.59	165.59	165.59	165.59	165.59	165.59	165.59	165.59

Table 5.10 Revenue Schedule for Time-Based Revenue

Now if you want to compare this to the old standard, you see that there are significant differences. In this case, we are recognizing the device as 413.97, which is more than the 1

amount that is billed and would be recognized based on IAS 18. On the other hand, service will be recognized month by month with an amount that is less than what is being billed.

We already mentioned that defining the POB types would need to be done in two steps:

1. Define the POB type in Transaction FARR_IMG.
2. Define how the POB type will be determined in BRPlus decision table DT_PROCESS_POB.

In both cases, you need to look at several key settings, which we'll describe next, followed by the resulting revenue schedule.

Time-Based POB Settings

Let's start with creating a POB type (refer to [Section 5.2.1](#)). In the **Fulfillment Data** section of the screen, as shown in [Figure 5.39](#), select **T Time-Based** in the **Fulfillment Type** field for a time-based POB.

The screenshot shows a 'Fulfillment Data' form with the following fields:

- Fulfillment Type: T Time-Based (dropdown menu)
- Event Type: (empty dropdown menu)
- Start Date Type: Available on Creation (dropdown menu)
- Duration: (two empty input boxes)
- Deferral Method: 1 (input field)

Figure 5.39 Time-Based POB Settings

Because you're creating a time-based POB type, the **Event Type** field needs to remain blank.

Let's turn our attention to the **Deferral Method** field. *Deferral methods* indicate how the fulfillment of the POB is distributed over the specified period of time and is relevant only to POBs with a time-based fulfillment type.

If you select the field, you'll see a list of available methods, as shown in [Figure 5.40](#).

Def. Meth.	Description
1	Linear Distribution, Day-Specific, 365/366 Basis
2	Linear Distribution, Day-Specific, 360 Basis
3	Linear Distribution, Day-Specific, 360 Basis, with Rounding Adjustment
4	Linear Distribution, Day-Specific, 365/366 Basis, with Rounding in the Last Period
F	Recognition in First Period
L	Recognition in Last Period
S	Linear Distribution, Period-Specific

Figure 5.40 Deferral Methods Available

Deferral method **1** is evenly distributing revenue over the number of days in the duration. Therefore, the revenue or cost recognized for each accounting period is in proportion to the number of days that fall in that accounting period. When you use this deferral method, the total number of days is calculated at 365 or 366 days per year, depending on whether it's a leap year. Our previous example would look like [Table 5.11](#).

Jan.	Feb.	March	April	May	June	July	Aug.	Sept.	Oct.	Nov.
168.76	152.43	168.76	163.32	168.76	163.32	168.76	168.76	163.32	168.76	163.32

Table 5.11 Revenue Schedule with Deferral Method 1

You can see that most revenue will go to months with 31 days and the least to February. Different calculations will be used if the year in which revenue is to be recognized is a leap year (February has 29 days and year 366).

Deferral method **2** represents the linear distribution on a basis of 360 days. The fulfillment of the POB is evenly distributed over the number of days in the duration. Therefore, the revenue or cost recognized for each accounting period is in proportion to the number of days that fall in that accounting period.

The difference between methods **2** and **3** is in how rounding will be treated. The normal situation is that contracts don't always start on the first day of the period. Therefore, even if you're splitting it equally, some differences can arise. These deferral methods force differences to be put into either the first or last period, respectively.

With deferral method **S**, the fulfillment of the POB is distributed over the number of accounting periods in the duration, regardless of the number of days that fall in each period. When you use this deferral method, the total number of days is calculated at 360 days per year, with 30 days per accounting period. If the duration isn't aligned with the accounting periods, then the following applies:

- If the part that falls in the first period is less than the entire period, the period is also recognized as having 30 days.
- If the part that falls in the last period is less than the entire period, nothing is distributed to the last period.

Using this method, only one fulfillment per period can be created, so if there is a prospective split, then a fulfillment with the same event date as the effective date of the prospective split is generated for the period when the prospective split took place.

In our scenario, using deferral method **S**, you would get 12 fulfillments created once you create the POB, where the deferral method would act in a similar way as method **2**. However, let's assume you go into contract modification with an effective date of January 25th. Instead of creating additional fulfillments, fulfillment for the month of January would be adjusted with an effective date January 25th and a quantity of 30. With method **S**, if the duration is less than a full month, fulfillment calculating 30 days would be created irrespective of the duration of the POB within a month.

Choosing Deferral Methods

Many discussions often arise when choosing a deferral method. The choice often comes down to simple and predictable deferral (methods 2 and 3) or accuracy (method 1). Although the final decision belongs to the user, there are two questions to guide the way:

- **How is billing done?**

If billing is done equally every month irrespective of the number of days, you should try to keep the deferral method the same. If not, there might be differences between billing and revenue recognized due to different amounts billed and recognized that aren't always wanted.

- **How is planning carried out?**

Normally, the planning process is done in advance in every organization. So how do you spread your planned revenue: according to the number of days or equally? It makes sense to keep planning and actual values coming in the same way.

If you're not happy with standard deferral methods, there is an option for creating custom methods and applying your own logic in it. BAdI FARR_BADI_DEFERRAL_METHOD_V2 (Handling of Deferral Methods on Performance Obligations) allows you to develop your own deferral methods. You can find this BAdI in Customizing for **Revenue Accounting under Revenue Accounting • Revenue Accounting • Contracts • Business Add-Ins**.

Returning to the POB type creation (refer to [Figure 5.39](#)), the next setting is the **Start Date Type** option, which determines how you want to treat the start date of the POB that is created, as shown in [Figure 5.41](#).

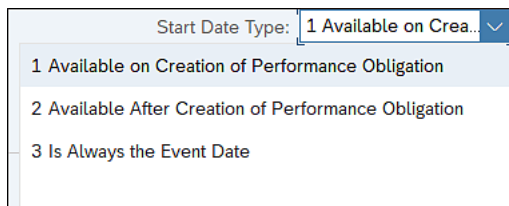


Figure 5.41 Start Date Options When Creating Time-Based POB

The start date indicates when revenue recognition calculation will start for time-based POBs. That is why it's important to understand options and compare them with data available in source systems so proper revenue calculation can occur:

- **1 Available on Creation of Performance Obligation**
The start date has to be provided once the POB is created. If not, the system will issue an error message.
- **2 Available After Creation of Performance Obligation**
The start date can be missing at the moment of creation of the POB and be specified later.
- **3 Is Always the Event Date**
Revenue won't be recognized until a certain event occurs. This can be useful when dealing with linked POBs with different fulfilment types on leading and not leading levels.

For this example, choose option **1 Available on Creation of Performance Obligation**. Regarding how dates are extracted, in the case of integration with sales and distribution, these are usually retrieved from the billing plan, which is normally used to specify how billing will be done. [Figure 5.42](#) shows the **Billing plan** that you'll receive, where you can see the contract **Start date** and contract **End date**.

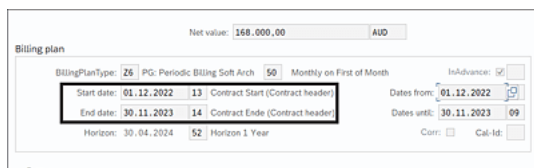


Figure 5.42 Billing Plan Dates

These dates are represented in table FPLA with fields FPLA-BEDAT and FPLA-ENDAT.

Once you have time-based POB, it's worth mentioning how revenue will be recognized. This will be done by using the following formula:

- *Remaining revenue to be recognized up to the period (Remaining revenue) = Contractual price of the POB - Revenue already recognized before this period*
- *Remaining days to be fulfilled up to the period (Remaining days) = Total days of the POB - Days already fulfilled before this period*
- *Revenue recognized of the period = Days to be fulfilled in the period × Remaining revenue ÷ Remaining days*

Understanding this calculation is key when it comes to contract modifications and results from RAR.

Revenue Schedule

When you create a time-based POB, the Contract Search app will show you the revenue schedule that represents how much revenue was to be recognized until now and how much will be recognized until the end of the contract.

In the SAP Fiori launchpad, once you click on the app and select a contract, first, you see totals, as shown in [Figure 5.43](#).

Item	Performance Obligation	Contractual Price	Standalone Selling Price	Allocated Amount	Allocation Effect	Price Status	Invoiced Amount	Quantity Fulfillment Progress	Effective Quantity	Fulfillment Type
203643	REAGAP (00)	0.00 EUR	250,000.00 EUR	300,000.00 EUR	300,000.00 EUR		0.00 EUR	0%	0.0000	PC (Time-Based)
203642	SERVICE (00)	120,000.00 EUR	0.00 EUR	0.00 EUR	120,000.00 EUR		10,000.00 EUR	8.33%	300,000 DAY	T (Time-Based)
203644	SERVICE (00)	240,000.00 EUR	0.00 EUR	0.00 EUR	240,000.00 EUR		20,000.00 EUR	8.33%	300,000 DAY	T (Time-Based)

Figure 5.43 Revenue in a Contract Based on Time

You can see that all POBs included in the contract are displayed in the report. You'll recognize time-based POBs based on the **Fulfillment Type** that is entered on the right. For time-based POBs, there will always be calculated how much of revenue is due to be recognized in percentages, which is displayed in the **Quantity Fulfillment Progress** column.

If you want to see details on time-based revenue, you need to select the **Revenue Schedule** option at the top of the table, and month-by-month calculations will be displayed, as shown in [Figure 5.44](#).

Item	Performance	Accounting Pk	Performance	Allocated A.	Effective Quan.	Quantity	Fulfillment	Contribution Pct.	Revenue	Uninvoiced	Period Revenue	Pricing Price	Invoiced A.
203643	REAGAP (00)			300,000.00	30	30	100%	0.00	0.00	0.00	0.00	0.00	0.00
203642	SERVICE (00)			120,000.00	300	30	10%	0.00	0.00	0.00	0.00	10,000.00	10,000.00
203644	SERVICE (00)			240,000.00	300	30	10%	0.00	0.00	0.00	0.00	20,000.00	20,000.00

Figure 5.44 Revenue Schedule

You can see that POB **203643** is time based and thus has revenue scheduled. In the **Status** column, you can check the current status of revenue that needs to be recognized in that period. If you see a square dot, it means that revenue is due for recognition and that

- 20230060000101 for the 15 days in June
- 20230070000101 for the complete month of July
- 20230080000101 for the complete month of August
- 20230090000101 for the remaining days in September

- **CnTy**

There will be entries for all the condition types (**CnTy**) of the POB.

- **Defer. Cat.**

The deferral category for time-based revenue forecasts will generally be **MA**. There are deferral categories for returns as well. To check the possible values for deferral categories, double-click on data element **FARR_DEFERRAL_CATEGORY**, check domain **FARR_DEFERRAL_CATEGORY**, and click on the value range. A screen appears that will show you the possible values for deferral category (see [Figure 5.46](#) and [Figure 5.47](#)).

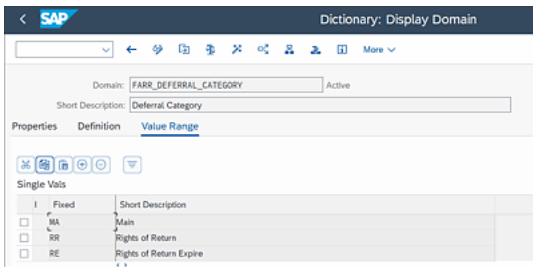


Figure 5.46 Deferral Categories

Deferral Category	Short Description
MA	Main
RR	Rights of Return
RE	Rights of Return Expire
TV	Time Value of Money

Figure 5.47 Category Descriptions

- **FulfillType**

Fulfillment type will give you the details of the fulfillment type, such as if it's time based (**T**), event based (**E**), POC (**O**), and no fulfillment (**N**). You can see the **Fulfillment Type** field in the table in [Figure 5.48](#).

<input type="checkbox"/>	FULLFILL_TYPE	<input type="checkbox"/>	FARR_FULFILL_TYPE	CHAR	1	0	0	Fulfillment Type
<input type="checkbox"/>	COMPANY_CODE	<input type="checkbox"/>	BUKRS	CHAR	4	0	0	Company Code
<input type="checkbox"/>	ACCT_PRINCIPLE	<input type="checkbox"/>	ACCOUNTING_PRINCIPLE	CHAR	4	0	0	Accounting Principle
<input type="checkbox"/>	AMOUNT_CURR	<input type="checkbox"/>	WAKERS	CURY	5	0	0	Currency Key
<input type="checkbox"/>	QUANTITY_INLI	<input type="checkbox"/>	FARR_QUANTITY_INLI	UNIT	3	0	0	Unit of Measure
<input type="checkbox"/>	SOURCE_ACCT	<input type="checkbox"/>	FARR_SOURCE_ACCT	CHAR	10	0	0	Source Account Number
<input type="checkbox"/>	TARGET_ACCT	<input type="checkbox"/>	FARR_TARGET_ACCT	CHAR	10	0	0	Target Account Number
<input type="checkbox"/>	STATISTIC	<input type="checkbox"/>	KSTAT	CHAR	1	0	0	Condition is used for statistics

Figure 5.48 Fulfillment Types

- **Src A/c No. and Target A/c**

The source account and target account store the account-related details and are used during the execution of program C, which is the posting program that creates the accounting document.

- **Category**

This will have the value of either **P** or **C**, which is the indicator of price or cost items, respectively.

- **Spec_Indicator**

The deferral item special indicator can be set as follows:

- **P**: Main price
- **C**: Main cost
- **D**: Allocation difference
- **F**: Free face value
- **'**: Normal entry

There are also amount fields for storing the calculated amount, as shown in [Figure 5.49](#):

- **DOC_AMT_CUMULATE** has the allocated amount of each POB that was valid with reconciliation key.
- **REV_AMT_DELTA** has the amount that will be posted and calculated. In other words, this field provides the amount, which needs to be posted as the delta between recognized revenue and IFRS 15 revenue.

<input type="checkbox"/> DOC_AMT_DELTA	<input type="checkbox"/> FARR_DEFITEM_DOC_A_CURR	23	2	0 Delta Amount of a Cond. for a POB in a Posting Period
<input type="checkbox"/> DOC_QTY_DELTA	<input type="checkbox"/> FARR_DEFITEM_DOC_Q_CURR	18	6	0 Delta Quantity for a Perf. Obligation in a Posting Period
<input type="checkbox"/> DOC_AMT_CUMULATE	<input type="checkbox"/> FARR_DEFITEM_DOC_A_CURR	23	2	0 Effective Value of a Condition Type for a Perf. Obligation
<input type="checkbox"/> DOC_QTY_CUMULATE	<input type="checkbox"/> FARR_DEFITEM_DOC_Q_CURR	18	6	0 Effective Quantity of a Performance Obligation
<input type="checkbox"/> REV_AMT_CATCHUP	<input type="checkbox"/> FARR_DEFITEM_REV_A_CURR	23	2	0 Amount of retrospective Revenue Catchup
<input type="checkbox"/> REV_AMT_DELTA	<input type="checkbox"/> FARR_DEFITEM_REV_A_CURR	23	2	0 Amount for Posting
<input type="checkbox"/> REV_QTY_DELTA	<input type="checkbox"/> FARR_DEFITEM_REV_Q_CURR	18	6	0 Quantity for Posting
<input type="checkbox"/> REV_QTY_DEF_DELTA	<input type="checkbox"/> FARR_DEFITEM_REV_Q_CURR	18	6	0 Revenue Quantity Difference
<input type="checkbox"/> REV_QTY_NO_DELTA	<input type="checkbox"/> FARR_DEFITEM_REV_Q_CURR	18	6	0 Revenue Quantity (non-deduct)

Figure 5.49 Revenue Amount

Let's walk through an example of REV_AMT_DELTA, as shown in [Table 5.12](#). If there is a case where the user needs to pay an amount of \$50 per month, the amount received from the customer is \$200, and the contract is for a period of four months.

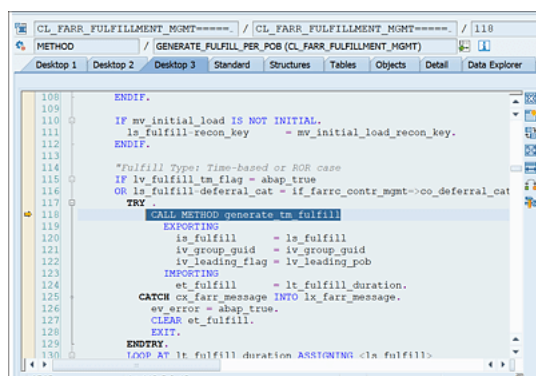
POB ID	Reconciliation Key	Start Date	End Date	REV_AMT_DELTA
1001	20230010000010	01.01.2023	31.01.2023	50.00
1001	20230020000010	01.02.2023	28.02.2023	50.00
1001	20230030000010	01.03.2023	31.03.2023	50.00
1001	20230040000010	01.04.2023	30.04.2023	50.00

Table 5.12 Sample of FARR_D_DEFITEM

This is the sample of data in table FARR_D_DEFITEM and its value after subtotaling based on condition types and the reconciliation key. The calculation isn't as simple as it looks. We've taken a very simple case to understand and to show what table FARR_D_DEFITEM stores. The start date is the key for calculating the duration of the contract. The start_date marks the first date of the contract, and the first reconciliation key will be created for the period in which the start_date starts. The end date marks the end of the contract period, and the difference between the start date and end date is the duration of the contract.

The REV_AMT_DELTA is calculated by using these fields. For demonstration purposes, we've highlighted the exact point in the debugger screen where the calculation of REV_AMT_DELTA

happened, as shown in [Figure 5.50](#).



```
108      ENDIF.  
109  
110      IF mv_initial_load IS NOT INITIAL.  
111         lv_fulfill_recon_key = mv_initial_load_recon_key.  
112      ENDIF.  
113  
114      *Fulfill Type: Time-based or ROF case  
115      IF lv_fulfill_tm_flag = abap_true  
116         OR lv_fulfill_deferral_cat = if_farro_contr_mgmt->co_deferral_cat  
117         TRY.  
118             <--->Alt METHOD generate ts fulfill  
119             EXPORTING  
120                 is_fulfill = lv_fulfill  
121                 iv_group_guid = iv_group_guid  
122                 iv_leading_flag = iv_leading_pob  
123             IMPORTING  
124                 et_fulfill = lt_fulfill_duration.  
125             CATCH cx_farr_message INTO lx_farr_message.  
126                 et_error = abap_true.  
127             CLEAR et_fulfill.  
128             EXIT.  
129         ENDTRY.  
130         LOOP AT lt_fulfill duration ASSIGNING cln_fulfill
```

Figure 5.50 Calculation of REV_AMT_DELTA

Table FARR_D_DEFITEM: What to Expect

As explained previously, this table not only contains the revenue schedule for current contract values, but also keeps the schedule for the status at each point in time. This is done based on reconciliation keys: each time the user reprocesses contract or performs contract modification, a new reconciliation key will be assigned to the POB, and table FARR_D_DEFITEM will be rebuilt. You can also see that all condition types used in the contract are on a separate line. Now, imagine that you have two time-based POBs with a duration of 24 months while using only two condition types: one for the main price and one for the SSP. Only for that contract, this table will have $24 \times 2 = 48$ lines for each reconciliation key. For 24 months, if the contract is reprocessed or modified 10 times, the number of entries will be 480. This can become a bottleneck if the user has contracts with a very long duration (some users even have indefinite contracts) and/or uses a lot of contract combinations. That is why it's recommended to think about on what level contracts will be combined and what should be simplified in the duration for contracts with a very long time period.

5.3.2 Point-in-Time Fulfillments

When revenue will be recognized at a point in time, a trigger needs to happen first. In POB types, there is a list of indicators used to conclude that the party which purchased an asset has de facto control over that asset—a key indicator for revenue recognition.

For example, physical possession of an asset typically gives the holder the ability to direct the use of and obtain benefits from that asset and is therefore an indicator of which party controls the asset. However, physical possession doesn't determine which party has control on its own. As an example, a publisher sends a book to a reseller but doesn't give the reseller the right to resell it for a few more weeks to ensure that the same date of sales start across all the resellers. In this case, it's arguable whether the reseller has control over the books, even if they are in their possession.

In addition, in some cases, the option of *acceptance* can appear. A customer acceptance clause provides protection to a customer by allowing the customer to either cancel a contract or force a seller to take corrective action if goods or services don't meet the

requirements in the contract. Judgment can be required to determine when control of a good or service transfers if a contract includes a customer acceptance clause.

Customer acceptance that is only a formality doesn't affect the assessment of whether control has transferred. The acceptance clause might not be a formality if the product being shipped is unique, as there is no history to rely upon. An acceptance clause that relates primarily to subjective specifications isn't likely a formality because the reporting entity can't ensure the specifications are met prior to shipment.

All of these cases make it particularly important for an entity to assess when actual events can trigger revenue recognition to occur.

Let's revisit our POB type creation screen, as shown in [Figure 5.51](#). POBs that should be recognized as point in time are represented as **E Event-Based** POB types with different options for the **Event Type** field.

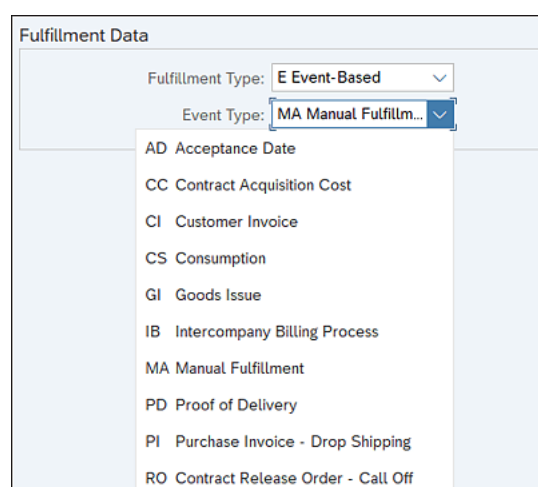


Figure 5.51 Event Types for POB Fulfillments

In the following sections, we'll look at the most important of these event types.

Goods Issue Fulfillment

By choosing **GI (Goods Issue)**, you're indicating to the system that the trigger for revenue recognition can be a pure issue of goods from stock. If you look at a standard sales and distribution use case, it can be represented by a simple diagram, as shown in [Figure 5.52](#).



Figure 5.52 Sales and Distribution Process

This sales process starts with the sales order in which the customer, kind of goods purchased, prices (transactional price and SSP), and all additional details are defined that might be needed to specify the relationship between seller and customer (discounts,

different partner functions defining where to ship, who will pay, terms of payments, etc.). Based on this sales order, you'll create a delivery. Delivery itself can be done in many different ways (including with warehouse management [WM] or without, separate pick and pack processes, etc.), but, essentially, goods are moved from stock to the customer. Based on this delivery, an invoice is issued, which is the main document for customers indicating that the sales process is complete.

Now, this is a very simplified example that can be more complicated if it includes down payments, pro forma invoices, or some documents that might be predecessors for sales orders. The accounting impact comes after issuing goods where stock is credited, and, COGS is as an offset account, or in the case of services, a cost of sales (CoS) account. Once an invoice is issued, there's a post to accounts receivable—which goes to the balance sheet and revenue—that impacts profit and loss (P&L) and when compared with COGS represents a profit.

However, once you introduce goods issue as the fulfillment event (see [Figure 5.53](#)), you no longer need to wait for billing to recognize revenue; revenue can be posted at the moment of sending goods to the customer.

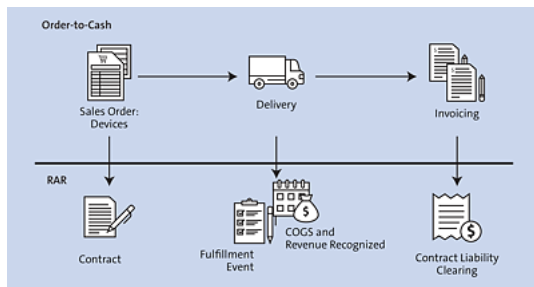


Figure 5.53 Goods Issue as Fulfillment Event

In RAR, contracts are created at the moment of sales order creation because that document proves the contractual relationship between customer and seller. At the moment of delivery, revenue is recognized and the receivable adjustment will be used as the offset account. At the same moment, COGS will be recognized, which is also one of the control mechanisms that revenue and costs fall into the same period.

Once the invoice is issued, the billing amount will be reversed and posted against the receivable adjustment account. This ensures that there aren't multiple documents posting revenue, and that only goods issue is used as the trigger for revenue recognition.

Now if we look at RAR in detail, the first step when integration will occur is when the user creates a sales order. In [Figure 5.54](#), we created one order that contains 1,200 PCs of material **CD994563**. From the prerequisite BRFplus settings, we set that fulfillment for this material is **GI** (goods issue; refer to [Figure 5.51](#)). We don't need to specify anything else because pricing will be determined automatically—meaning that the system will retrieve both the transactional price and SSP based on the maintained conditions in the system.

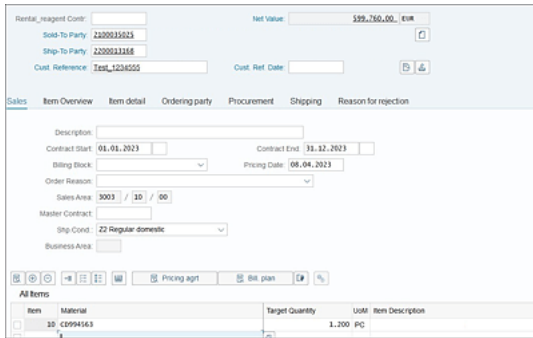


Figure 5.54 Sales Order Created

Once you save the document, you need to go to Transaction FARR_RAI_MON to see the created RAI. In the selection screen, enter just the sales order number, and the system will display created and unprocessed RAIs (if you select statuses **Raw** and **Processable**). The resulting screen shown in [Figure 5.55](#) displays basic information for the RAI, which is retrieved from the sales order, such as a **Customer**, **Reference ID**, and so on.

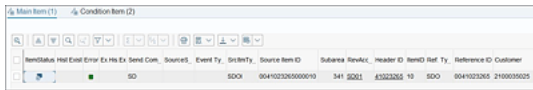


Figure 5.55 SDOI RAI Created

Two condition items are also created. Switch to the **Condition Item** tab to show the screen in [Figure 5.56](#).

One condition type is for the transactional price (**ZP01**) and that item is marked as the main so the **P/L Account** is identified. The second item is for the SSP (**ZSSP**), and this item is created as statistical. If BRFPplus settings mean that the SSP amount for the item is 0 (or the item is excluded from allocation), the total SSP on the contract level must be <> 0.

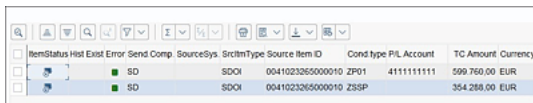


Figure 5.56 Condition Types Created

The next step is processing these RAIs (see [Chapter 4, Section 4.1](#), for details). Once performed, you'll get the messages shown in [Figure 5.57](#).

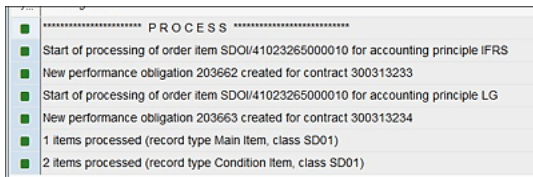


Figure 5.57 Processing of RAIs

The system is informing you that both POBs and contracts are created. In this case, parallel accounting is in use, which means the local ledger is parallel to IFRS, so two contracts with separate POBs will be created.

Technically, at this point, the system is moving processable RAIs to the processed table (table /1RA/SD014MI and /1RA/SD014CO, in this case) and populating all tables that are relevant for contract management. These are tables that begin with FARR_D, and the most relevant that are populated now are table FARR_D_MAPPING (showing linking between source document and POBs created), table FARR_D_CONTRACT (contract definition), and table FARR_D_POB (definition of POBs). In addition, the system will perform allocation if needed but no postings are made up to this point.

By running the Contract Search app, you can see the RAR contracts created by processing the sales order. As shown in [Figure 5.58](#), you need to enter the company code and accounting principle, and all contracts resulting from processing RAIs will be created. As highlighted, the **Operational Document** field represents the link between the document that was entered in the operational application and caused creation of the RAR contract (in this example, the sales order from sales and distribution).

If you select the **Comprehensive View** option, you can see details of the contract such as prices, event type, and how much you invoiced/recognized until now. In this case, as shown in [Figure 5.59](#), you see that **Duration** is **0**, which is as expected because you created an event-based POB, and that the current **Fulfilled Progress** is also **0**, which is again OK because there were no goods delivered to the customer yet.

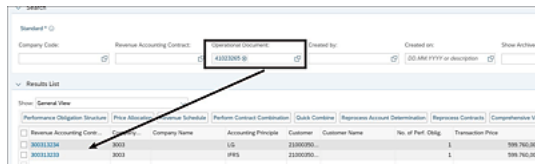


Figure 5.58 Contracts Created

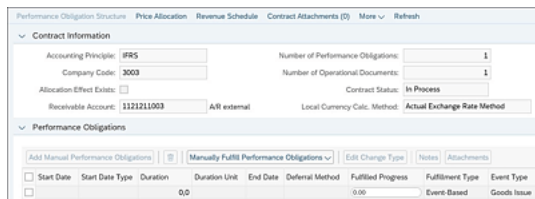


Figure 5.59 Details of the Contract

Now you can execute the next step, which is goods delivery. As mentioned, this can be done in multiple ways, but once post goods issue (PGI) is executed, you'll see RAIs created once more in Transaction FARR_RAI_MON, as shown in [Figure 5.60](#). Again, you're not changing any selection criteria. By using the sales order number as **Header ID**, the system will show you all the dependent RAIs created.

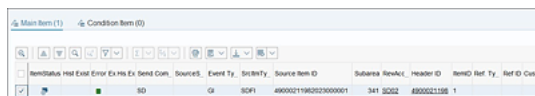


Figure 5.60 Fulfillment Process

Now you see that an item with only one main item was created with the event type (**Event Ty...**) set as **GI** and the source item type (**SrcItemTy...**) as **SDFI**. Fulfillment items don't have conditions, and, in this case, they are used just to indicate what quantity is delivered

to the customer, meaning how much revenue you can recognize. In this case, we delivered 250 pieces compared to 1,200 in the sales order.

Once RAI for fulfillment is processed, the item is moved from table /1RA/SD02MI to processed RAI table /1RA/SD04MI, and data tables updates are performed—most importantly for table FARR_D_FULFILLMNT, as shown in [Figure 5.61](#).

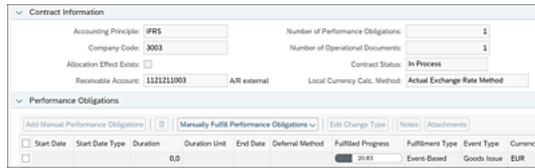


Figure 5.61 Contract with Fulfillment

This is now represented as fulfillment progress, as you see that these 250 pieces are represented as **20.83** percent under **Fulfilled Progress**. Again, the system didn't post anything, and this will stay as is until you run posting programs.

The last step in this case is issuing an invoice: you're billing the customer. Similarly, as in previous cases, an invoice RAI will be created in table FARR_RAI_MON, as shown in [Figure 5.62](#).

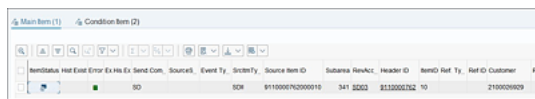


Figure 5.62 Invoice RAI

Similar to the order case, this item has condition items, which makes perfect sense, because you need to know if you invoiced more or less to customer. In this case, a discount is given to the customer, and multiple condition types are available (each discount will get a separate condition type). Once this item is processed, the item is moved from table /1RA/0SD032MI to processed table /1RA/0SD034MI.

With the last step executed, you can go to the Contract Search app and check the contract, as shown in [Figure 5.63](#).

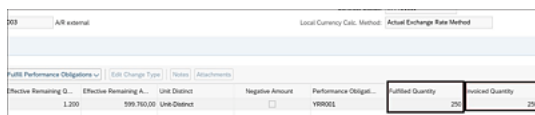


Figure 5.63 Contract Search after Invoicing

Now the system is showing the populated columns of **Fulfilled Quantity** and **Invoiced Quantity**. These values are retrieved from the invoice, which was processed as SDII RAI. Again, before running ABC programs, no postings will occur.

Customer Invoice Fulfillment

Next, let's choose **CI (Customer Invoice)** as our event type (refer to [Figure 5.51](#)). Let's say you have a customer who walks into a shop to buy a phone and a service. Once he selects a plan and a device, he signs a contract with a mobile provider. As a result, he is given an invoice that he pays at the same time. The process flow is depicted in [Figure 5.64](#).

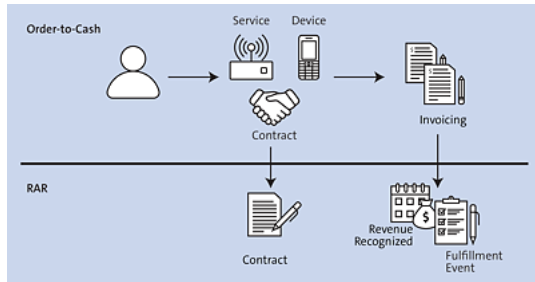


Figure 5.64 Customer Invoice as a Fulfillment Event

Now, we discussed time-based fulfillment in [Section 5.3.1](#), and it's clear that the service will be recognized over time, but what about the device? Again, it's a good idea to look at the IFRS 15 standard. Paragraph 31 of IFRS 15 states: "An entity shall recognize revenue when the entity satisfies the POB by transferring a promised good or service (i.e., an asset) to a customer. An asset is transferred when the customer obtains control of that asset." This means that the key word in recognizing revenue is "transfer of control" of the promised asset.

Paragraph 33 of IFRS 15 states the following:

"Control of an asset refers to the ability to direct the use of, and obtain substantially all of the remaining benefits from, the asset. Control includes the ability to prevent other entities from directing the use of, and obtaining the benefits from, an asset. The benefits of an asset are the potential cash flows (inflows or savings in outflows) that can be obtained directly or indirectly in many ways."

In other words, the customer also can benefit from having control over the asset.

So, in this case, you see that once the customer signs a contract, he can start using his phone, has a physical possession, and is presented with an obligation to pay for the device at the moment of receipt of the phone. All of these are clear indicators that revenue can be recognized at that moment.

In this case, once you create a contract, you'll get an order item in RAR that when processed creates a contract. At the same time, an invoice item will be created to represent both fulfillment and invoicing of the POB. So, in this case, all the events occur simultaneously.

Telco Example

Our previous description is a classic postpaid example of bundles. You can see that here we skipped providing the SDFI item to the system to do revenue recognition by fulfillment, but went to invoice as the only event triggering it. Be careful if this example fits the process you're establishing: if for the device to function some additional time must pass or some installment is pending, then the conclusion that the customer took control over the asset needs to be looked at more closely. In that specific case, providing fulfillment events that aren't invoicing might be an additional challenge.

Proof of Delivery Fulfillment

Let's move on to selecting **PD (Proof of Delivery)** as the **Event Type** (refer to [Figure 5.51](#)). This fulfillment event is very similar to the goods issue fulfillment event. In this case, you're delivering goods to the customer, but, based on contractual terms, the customer gets control over goods only at the moment when goods arrive at the warehouse, as shown in [Figure 5.65](#).

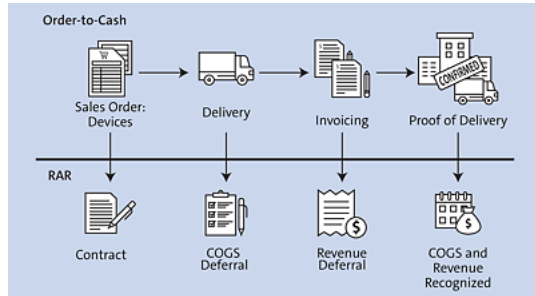


Figure 5.65 Proof of Delivery Process

You're delivering goods, but because the customer still hasn't taken control over the goods, you can't recognize revenue. Consequently, based on the principle that both revenue and costs related to that revenue need to be recognized in the same period, you can't recognize COGS too and need to defer it (for this to work, cost recognition needs to be enabled on the accounting principle and POB type levels). You're also issuing invoices, and revenue from the invoice will be reversed until you don't get a document from the customer of the logistics service provider (LSP) confirming that the customer took control over goods, that is, proof of delivery. In that moment, you can reverse all deferrals and recognize revenue and cost, respectively.

SAP follows this process to handle a proof of delivery:

- The customers need to be marked as a proof of delivery relevant (in the **Shipping** section for sales data in the sales order).
- The item category to be used needs to be marked as a proof of delivery relevant.
- In BRFplus, you need to set the POB type as valid for a proof of delivery: fulfillment type needs to be event based, and event type is **PD (Proof of Delivery)**.

Once the seller receives a proof of delivery from the LSP, it will be entered by Transaction VLPOD, and fulfillment can occur. This process has many limitations, but there are two that are most important: (1) You can't perform invoicing until a proof of delivery is received, so a pro forma invoice is sent for the customer to be able to pay or for goods to be transferred over the border if there is an export process involved; and (2) you can handle logistic process using different LSPs where some of them might send a proof of delivery, while others might not. If a proof of delivery isn't sent by the LSP, you need to go through the enhancement process to determine or estimate the actual delivery date.

Percentage of Completion Event

Next, let's select event type **POC**. This fulfillment type is used when revenue needs to be recognized based on the progress of the project. The percentage of completion (POC) method falls in line with IFRS 15, which indicates that revenue from POBs recognized over a period of time should be based on the POC. The method recognizes revenues and expenses

in proportion to the completeness of the contracted project. It's commonly measured through the cost-to-cost method, which uses the following formula:

$$\text{Percentage of completion (POC)} = \text{Costs incurred until date} \div \text{Total estimated costs}$$

For example, if you estimate that your total costs to execute one project will be 100,000 and until today 15,000 has been incurred, POC is calculated as 15%.

POC as a fulfillment event works when you need to transfer an amount calculated for revenue to be recognized to RAR from cost object controlling. The first important thing to mention is that RAR itself doesn't calculate anything; instead, it's taking amounts calculated from results analysis, which is represented by the results analysis key assigned to the cost object (either a WBS element or an internal order), as shown in [Figure 5.66](#).

Results analysis is working with planned costs and revenues that are posted to cost objects for which you want to calculate POC. The results of calculation with results analysis keys are work in progress (WIP) CoS, valuated revenue, and a few more different categories. Once the calculation is done, settlement is performed to send these values to controlling receivers (profitability analysis segments or sales orders).

There are two different scenarios when it comes to integration between RAR and results analysis:

- Perform results analysis that will transfer POC to RAR. In this case, WIP, CoS, and valuated revenues are calculated and posted, which can be identified by business transaction KABG in table COSB.
- Perform a posting run in RAR. The actual adjustment will be posted in RAR and automatically update valuated revenue for RA. These line items can be identified with business transaction KABE.

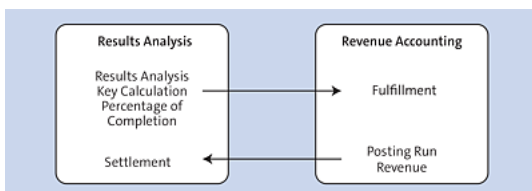


Figure 5.66 Percentage of Completion Method

A revenue-based valuation method (see [Figure 5.67](#)) is used in results analysis, as follows:

- Perform a posting run in revenue accounting. This will post actual revenue adjustments in controlling.
- Perform the results analysis. All actual revenues, that is, standard invoices and revenue adjustments from revenue accounting, are considered for the valuated revenues. All valuated revenues are posted. You can identify each line item by business transaction KABG.
- Perform a settlement.

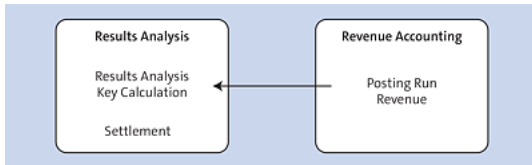


Figure 5.67 Revenue Method

RAR manages only revenue and revenue postings to financial accounting, while results analysis is still managing costs, WIP, reserves, and postings to profitability analysis. For posting to profitability analysis to occur, a proper version needs to be selected.

Depending on the business scenario, results analysis can be performed for a sales order, WBS element, or internal order as a cost object. RAR supports two integration scenarios for results analysis:

- **Cost-based**

In the cost-based scenario, you're comparing actual costs with plan costs and coming to the POC.

- **Revenue-based**

This method, unlike the previous one, is looking at revenues: planned revenue (which in this case is equal to sales order value) needs to be compared to actual revenue (which is equal to billing) and represents POC.

The results analysis method controls which formula is used to calculate the results analysis data for the cost objects (sales document, project, and internal order). The results analysis key will be part of the WBS element where POC is calculated. Setup can be found by running Transaction OKG3 to arrive at the screen shown in [Figure 5.68](#). By clicking the **Results Analysis Method** field, you can see details about the results analysis key that is defined and how values are calculated.

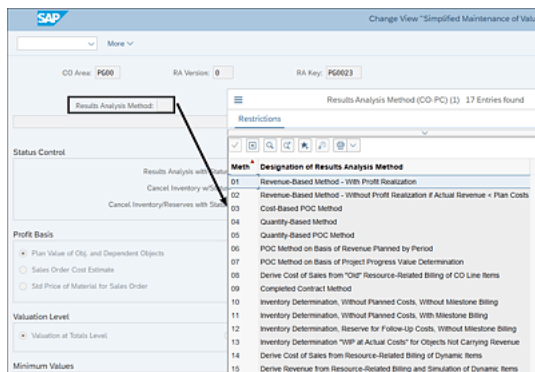


Figure 5.68 Maintenance of the Results Analysis Key

Let's take a closer look at the underlying calculations for three key methods:

- **Method 03**

Cost-based results analysis (results analysis method **03**) will calculate costs and revenue as follows:

$$POC = \text{Actual costs} \div \text{Planned costs}$$

$$\text{Calculated revenue} = POC \times \text{Planned revenue}$$

$$\text{Calculated costs} = POC \times \text{Planned costs}$$

- **Method 01**

Revenue-based results analysis (results analysis method **01**) will calculate costs and revenue as follows:

$$POC = \text{Actual revenues} \div \text{Planned revenues}$$

$$\text{Calculated revenue} = POC \times \text{Planned revenue}$$

- **Method 09**

Completed contract method (results analysis method 09) will calculate costs and revenue as follows:

$$\text{Calculated costs} = POC \times \text{Planned costs}$$

Until the contract is completed (status **TECO**), the project won't settle the costs and revenues. When it comes to setup, **POB Type** needs to be defined as POC relevant (see [Figure 5.69](#)).

The screenshot shows the configuration for a POB Type. The POB Type is 'ZPOC' and the Description is 'PoC method based POB'. Under the 'General Data' section, the 'Perf.Obligat.Nam' is 'POC BASED POB' and the 'No Cost Recognition' checkbox is unchecked. Under the 'Fulfillment Data' section, the 'Fulfillment Type' is 'O Percentage of Co...' and the 'Event Type' is empty.

Figure 5.69 Definition of the POB Type

The next step is to perform integration between results analysis and revenue accounting by executing Transaction FARR_IMG and going to menu path **Revenue Accounting • Integration with Cost Object Controlling**.

In project-related business processes, it's common that planning is performed in several versions before it's finalized. The same goes for currencies: a company can have multiple currency types in which it's working (local, group, and so on). In this step, it's necessary to indicate which version and currency type will be used for integration with RAR.

To access that Customizing step, there is no Customizing node, but you need to access the table where these values are kept. To do so, execute Transaction SM30 and enter view V_TKKA_RR_AC. In this view, as shown in [Figure 5.70](#), select **New Entries**, and maintain all company codes (**CoCd**) with versions (**RAVn**), controlling area (**COAr**), and currency types (**CC**) to be considered.

RA Customizing for RR: POC Currency, ACC Principle					
	COAr	CoCd	RAVn	AccP	CC
<input type="checkbox"/>	JP01	JP2	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	AU02	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	FR02	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	GB04	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	ID02	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	NL02	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	NL03	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	US19	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	ZA02	0	IFRS	10 Company code currency
<input type="checkbox"/>	PG00	ZA03	0	IFRS	10 Company code currency

Figure 5.70 Customizing Results Analysis: Version Relevancy

The next step to maintain is the integration of results analysis keys and RAR; you'll approach that in a similar way by maintaining view V_TKKA_RR_ME. Once the view is entered, as shown in [Figure 5.71](#), you need to link the different results analysis keys and how they integrate with RAR. In the integration method (**IntMeth**) dropdown, select option **1 PoC-Based Integration**, which means the cost method is used with the calculation of the RAR POC. In addition, option **3** means revenue method.

RA Customizing: RR-Relevant RA Keys, RA Methods					
	COAr	CoCd	RAVn	RA Key	IntMeth
<input type="checkbox"/>	JP01	JP2	0	PG0001	1 PoC-Based Integration
<input type="checkbox"/>	JP01	JP02	0	PG0024	1 PoC-Based Integration
<input type="checkbox"/>	JP01	ZA02	0	PG0001	1 PoC-Based Integration
<input type="checkbox"/>	JP01	ZA02	0	PG0024	1 PoC-Based Integration
<input type="checkbox"/>	PG00	AU02	0	PG0001	1 PoC-Based Integration
<input type="checkbox"/>	PG00	AU02	0	PG0023	1 PoC-Based Integration
<input type="checkbox"/>	PG00	AU02	0	PG0024	1 PoC-Based Integration
<input type="checkbox"/>	PG00	C002	0	PG0001	1 PoC-Based Integration
<input type="checkbox"/>	PG00	C002	0	PG0024	1 PoC-Based Integration

Figure 5.71 Customizing Results Analysis: Key Relevancy

Other Fulfillment Events

RAR provides many additional event types we could look at to determine how they fit customer requirements. Referring back to [Figure 5.51](#), we'll highlight a few of the most important additional types in this section:

- **RO Contract Release Order - Call Off**

Used in cases when a customer signs a contract that specifies only the total quantity that will be delivered, but details are missing. In this case, an additional sales order will be created to specify the exact quantity delivered. Another option when the RO fulfillment type might be useful is when you're dealing with the consignment process. For this delivery to work, specific customizing needs to be integrated with sales and distribution (see [Figure 5.72](#)), which is beyond the scope of this book.

<input type="checkbox"/>	ZCI	KEN	C Call-Off Order with predecessor
<input type="checkbox"/>	ZCPT	ZCPI	C Call-Off Order with predecessor
<input type="checkbox"/>	ZCPT	ZCPT	C Call-Off Order with predecessor
<input type="checkbox"/>	ZCR	G2N	M Credit/Debit Memos with referenc...
<input type="checkbox"/>	ZCR	ZG2W	M Credit/Debit Memos with referenc...
<input type="checkbox"/>	ZDR	L2N	M Credit/Debit Memos with referenc...

Figure 5.72 Setup for Event Type RO

Once this is set, you need to maintain the proper fulfillment type (RO) in BRPlus POB type determination tables. Note that the RO fulfillment type works only for quantity-relevant materials—services can't be recognized with it.

- **AD Acceptance Date**

In some cases, the customer needs to separately confirm that the goods delivered fit their expectations. In this case, a separate update to the sales order will be provided with this date. Similar to previous cases, revenue and cost, if necessary, will be deferred until this date is provided.

- **MA Manual Fulfillment**

In some cases, there is no possibility to use any trigger for revenue recognition except manually determining the amount of revenue to be recognized. Fulfillment can be done in total or as delta fulfillment. In addition, manual fulfillment can be useful as a source for any custom fulfillment event the customer might create.

5.4 Modifying Contracts

Contract modification refers to the changes made to a contract during its lifetime. Contract modifications are covered in detail in the standard, where two main contract modification options are included: (1) contract modification that comes as a change in scope and/or price (or both), and (2) change of estimates when any subsequent change in the transactional price will be allocated to the contract in the same way as at contract inception. According to article 18 of IFRS 15, a contract modification is a change in the scope or price (or both) of a contract that is approved by the parties to the contract.

A contract modification exists when the parties to a contract approve a modification that either creates new or changes the existing enforceable rights and obligations of the parties to the contract. To enable contract modifications in the system, you need to enable it on the accounting principle level ([Section 5.1.2](#)). As shown in [Figure 5.73](#), you can select the **Cont. Mod.** checkbox.

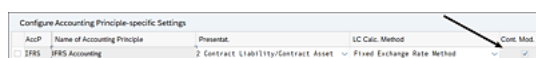


Figure 5.73 Enabling Contract Change

In the following sections, we'll discuss your options for modifying contracts, including a few important scenarios.

5.4.1 Prospective or Retrospective

There are three types of contract modifications:

- **Contract modification - prospective change**
Modification that is accounted for same way as an entity would terminate a contract and create a new one with the remaining distinct goods and services.
- **Contract modification - retrospective change**
Modification is accounted for as change to an already recognized revenue (increase or reduction) at the moment of contract modification in the form of a cumulative catchup.
- **Contract modification - mixed change**
Modification represents reallocation between remaining unit-distinct and non-unit-distinct parts.

One of the most important things to do while determining the change type is to determine whether the POB is unit distinct or not. This can be determined by BRFplus decision table DT_PROCESS_POB, and you can change this directly while managing a contract.

Contract change will be automatically triggered only if some of the data gets changed in the contract:

- Price of any element is changed.
- POBs are deleted or added.
- Quantity of event-based POBs is changed.
- Start/end date of time-based POBs is changed.

This means that only modifications that trigger contract reallocation are considered as contract changes. For example, changing a POB name won't trigger the modification process.

Each time a contract change should be executed, the system goes through the following steps:

1. Checks if the modification option is allowed for the accounting principle
2. Checks if the change type is specified in the UI (e.g., the data you've supplied supersedes that provided by a BAdI)
3. Checks what change type was supplied by the BAdI

We'll explain the process flow for prospective changes and retrospective changes in the following sections.

Prospective Changes

Prospective change is where the system is trying to apply change only to remaining periods when the contract is active—it doesn't change the past. Before the change is applied, the system calculates the remaining SSP and remaining transaction price using the following formula:

$$\text{Remaining SSP} = \text{SSP} \times (1 - \text{Fulfillment percentage})$$

OR

$$\text{Remaining SSP} = \text{SSP} \times (1 - \text{Fulfilled period}/\text{Duration})$$

$$\text{Remaining transactional price} = \text{Total contractual price} - \text{Recognized revenue of unit-distinct POB}$$

Let's apply this formula in an example, as shown in [Table 5.13](#).

POB	POB Name	Contractual Price	SSP (Total)	Start Date	End Date	Allocation Percentage	Allocation Amount
18047	Device	-	500.00			29%	352.94
18048	Service	1,200.00	1,200.00	1/24/2017	1/23/2018	71%	847.06
Total		1,700.00					1,200.00

Table 5.13 Contract with Two POBs

Here, there are two POBs in one contract. One POB is event based, and the other one is to be recognized over time. Based on SSP ratios, the system calculates the allocation ratio, which provides in return the amount of revenue given to each POB. Device revenue will be recognized as point in time while service revenue will be recognized over time.

Now, the customer is changing service on February 13 without a contract extension. Calculation of revenue to be recognized is shown in [Table 5.14](#).

POB	18047	18048	10849	Total
-----	-------	-------	-------	-------

POB Name	Device	Service	Service	
Original Contractual Price	-	1200.00	2266.67	3466.67
Original Allocation Amount	352.94	847.06		
Recognized Revenue	352.94	16.47		369.41
Balance Contractual Price	-	830.59		
Incremental Transactional Price		(1,133.33)		(1,133.33)
Revised Transactional Price		66.67	2,266.67	2,333.34
Revised Transactional Price Minus Posted Revenue (for Allocation)	(352.94)	50.20	2,266.67	1,963.92
SSP (Total)	500.00	66.67	2,266.67	2,833.34
Number of Days	N/A	20	340	360

Table 5.14 Contract after Modification

Let's look at each item separately:

- **Recognized revenue**

This has been recognized as of the last posting period and is determined by two POBs:

- 18047: Device (event-based), all revenue is recognized as event fulfilled.
- 18048: Service (time-based), following this formula:
(Service allocation revenue ÷ Days of contract) × Days revenue recognized in the month = Recognized revenue
 So, for this example, $(847.06 ÷ 360) × 7 = 16.47$.

The contract term is 360 days, and the contract start date is January 24, 2017. The business has chosen to calculate on a 360-day fiscal year as the deferral method.

- **Balance contractual price**

The balance contractual price is the result of the original allocation amount - the recognized revenue.

- **Incremental transactional price**

The incremental transactional price is the result of the original contractual price - the revised transactional price.

- **Revised transactional price**

The revised transactional price is the result of the $(\text{original contractual price} ÷ 360) × 20$, where

- 360 = days in contract
- 20 = 7 days of January recognized and 13 days of February (last day of modification)

- **Revised transactional price less posted revenue for allocation**

This is the result of the revised transactional price - the recognized revenue.

- **SSP**

For the service POB, you need both prices—original and updated—for time that the

service was used and for the device POB original amount only.

- **Number of days**

Number of days from the contract start date till the date of modification. For our example, this looks like the following:

- Old service = (7 days of January) + (13 days of February) = 20 days
- New service = 360 days - 20 days = 340 days

In [Table 5.15](#), you can see how the system calculates prorated amounts once the service gets modified.

	Service	Service (Used)	Service (New)	Total
SSP (Total)	500.00	66.67	2,266.67	2,833.34
Number of Days	N/A	20	340	360
Revised SSP/Day		3.33	6.67	
SSP for Period of Posted Revenue	500.00	23.33	-	
Balance Revised SSP	-	43.33	2,266.67	2,310.00
Total Revised SSP	500.00	66.67	2,266.67	
Revised Allocation Transactional Price (without Recognized Revenue)	-	36.84	1,927.08	1,963.92
Recognized Revenue	352.94	16.47		
Revised Allocated Transactional Price	352.94	53.31	1,927.08	2,333.34

Table 5.15 Contract after Modification: Final

Let's go through these items as well:

- Revised SSP/day = $66.67 \div 20 = 3.33$. Device isn't part of this calculation as it has been fully fulfilled.
- SSP for the period of posted revenue:
 - Device = Full SSP
 - Old service = Revised SSP/day $\times 7 = 3.33$
 - New service = Not posted yet, hence blank
- Balance revised SSP = SSP (total) - SSP for period of posted revenue
- Total revised SSP:
 - Device = Full as is recognized already
 - Old service POB = 66.67 (Revised SSP total)
 - New service POB = 2,266.67 (Revised SSP total)
- Revised allocated transactional price without recognized revenue:

- Old service POB = $(43.33 \div 2266.67) \times 1963.62 \times (\text{Total of revised transactional price less posted revenue column})$
- New service POB = $(2266.67 \div 2310) \times 1963.62$
- Recognized revenue:
 - 352.94 = Original allocated revenue for Jan for device
 - 16.47 = Prorated allocated revenue for Jan for old service
- Revised allocated transactional price = revised allocation transactional price (without recognized revenue) + recognized revenue

Figure 5.74 shows the numbers RAR provides in return after modification is processed.

Start Date	End Date	Performance Oblig.	Performance O.	Contractual Price	Standalone Sell.	Allocated Amount
		15047	Device	0.00	500.00	352.94
24.01.2017	13.02.2017	15048	Service	66.67	66.67	53.32
14.02.2017	23.01.2018	15049	Service	2,266.67	2,266.67	1,927.08

Figure 5.74 RAR Computation after Modification

Retrospective Changes

Opposite to prospective change, retrospective change occurs on partially fulfilled POBs of which fulfillments aren't unit distinct. When there are no POBs with unit-distinct fulfillment or all POBs with unit-distinct fulfillment are fully fulfilled, the system applies retrospective changes to the contract. When applying a retrospective change, the system will perform the following calculations:

Remaining standalone selling price = SSP

Remaining price = Total contractual price

Catchup = (Allocated price - Allocated remaining price) × Fulfillment percentage

Here you see that for retrospective changes, the system calculates cumulative catchup, which represents correction to revenue that is already reported. You can see how calculation works in the example shown in Table 5.16.

	Contractual Price	SSP	Allocation	Fulfillment Percentage	Recognized Revenue
POB 1	2,000.00	2,000.00	2,000.00	100%	2,000.00
POB 2	200.00	200.00	200.00	20%	40.00
POB 3	500.00	500.00	500.00	30%	150.00
Total	2,700.00	2,700.00	2,700.00		2,190.00

Table 5.16 Retrospective Change Example

Here you have three POBs in a contract where the first POB is completely fulfilled and the other two only partially. Then, the contractual price and SSP of POB 3 are raised to EUR 600 and EUR 600. The system applies a retrospective change by calculating the remaining SSPs and uses them to reallocate prices of the remaining contractual price. The remaining SSPs are calculated in the following way:

$$\text{Remaining SSPs} = \text{SSP}$$

The remaining price is the contractual price of all partially fulfilled POBs of which fulfillment isn't unit distinct. The system redistributes the remaining price to POBs in proportion to their remaining SSPs. The catchup is calculated in the following way:

$$\text{Catchup} = (\text{Allocated price} - \text{Allocated remaining price}) \times \text{Fulfillment percentage}$$

The result of how catchup is calculated is showed in [Table 5.17](#).

	POB 1	POB 2	POB 3	Total
Calculated Price	2,000.00	200.00	600.00	2,700.00
SSP	2,000.00	200.00	600.00	2,700.00
Allocation	2,000.00	200.00	600.00	2,700.00
Fulfillment Percentage	100%	20%	30%	
Recognized Revenue	2,000.00	40.00	150.00	2,190.00
Remaining Price				800.00
Remaining SSP	0	200.00	600.00	800.00
Allocated Remaining Price		200.00	600.00	800.00
Catchup			30.00	300.00

Table 5.17 Retrospective Change with Catchup

Standard versus Modification

RAR has a list of changes that are automatically used to trigger either prospective or retrospective change. Depending on the version, this list might differ but, for example, change of the SSP or start date of time-based POBs will always trigger retrospective change, while adding a quantity or extending a duration of POBs will trigger prospective modification.

You may be tempted to modify this behavior simply because of the accounting treatment of retrospective change: all effects will be cumulated in the currently open period with catchup. SAP provides BAdI `FARR_CHANGE_MODE_DETERMINATION` for you to use if you feel that the standard needs to be enhanced. However, be very careful before using this BAdI because results might be unpredictable. For example, adding a POB with a big SSP in a contract with a very small amount of remaining SSP will trigger reallocation of that big portion of revenue to the remaining POBs, whereas retrospective change in this case would give more reasonable results.

5.4.2 Contract Combination

Contract combination is a process where you merge several contracts to create a single contract. Before looking at options from SAP, let's discuss how contract combination needs to be performed. The contract combination process is tightly tied to the process of identifying a deal. Different sales documents need to be combined, and arrangement needs to be looked at as a whole if one or more of the following are met:

- Different contracts are negotiated with a single commercial objective.
- Payment in one contract depends on performance defined in another one.
- Goods or services promised in contracts are single POBs.

To add to the preceding criteria, there is also the time component: the decision regarding whether contracts will be combined under the same IFRS 15 contract is made at contract inception, and contracts need to be created in nearly same time, which can vary from days to months depending on the company.

There are several points that need to be addressed:

- Contracts that are part of the same deal and addressing the same customer
- Contracts entered in almost the same time
- Contracts that have a single POB spread across them

In RAR, you can get many documents that represent sales or agreements about sales with customers. You can have framework agreements, master contracts, sales orders, and so on. Each of them can represent a deal at a different point in time. Let's look at one simple example in [Figure 5.75](#).

You have one framework agreement with a customer where you agreed on the total value or quantity of goods that need to be delivered in a certain time. However, in each sales order, you're applying different discounts: the first one applies only a 10% discount because the customer purchased product A, but in the second order, a 20% discount is applied because the customer also bought product B with product A.

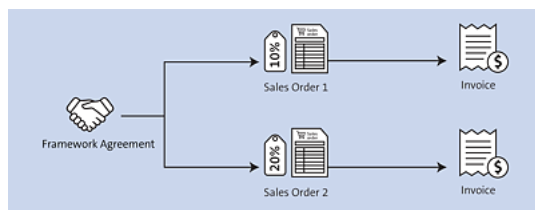


Figure 5.75 Contract Combination Rules

So, does this case suffice to be treated as a contract combination? The answer is no. Even if you have a framework agreement that binds together sales orders, it's clear that negotiation happens on the level of each sales order separately. All of these contracts, even if they have the same framework agreement behind them, need to be accounted for as separate contracts.

[Figure 5.76](#) depicts a very similar situation. We're signing an agreement on the level of the group of hospitals, and, based on that agreement, we're agreeing on certain terms and conditions. In this example, because an agreement has been signed for all hospitals that

are members of groups XYZ, we agreed that unified discounts will be applied across all orders. In this case, the answer to whether should these contracts be combined is yes because it's clear that terms and conditions are applied on the framework agreement level (or some other document), and all documents that will be created with reference to this agreement need to belong to the same IFRS 15 contract.

The next question is about the customer. The previous example might be used to explain complexity that can happen around even something that is considered relatively simple. So, who is our customer?

A customer is usually retrieved from a specific partner as defined in the sales document. There are a few standard parties (e.g., sold to, ship to, payer), but it also isn't uncommon to have custom partner functions defined too.

We're signing an agreement (or deal) with the hospital group, which is represented as business partner XYZ. Each sales order (or sales contract) sent to each hospital can have a different customer given, and you can even have a third, such as a partner to whom you're shipping. To complicate it further, all these invoices can be paid by different parties: they can be paid centrally by the hospital group or by the hospital itself.

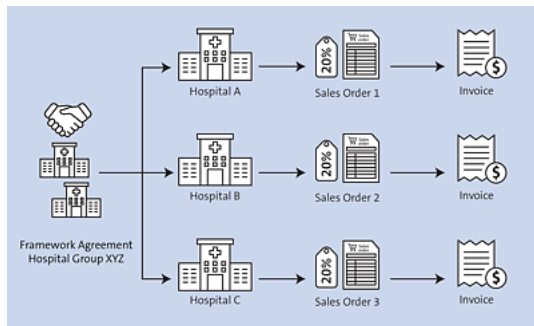


Figure 5.76 Contract Combination Group Level

RAR is rather straightforward here because a customer in the IFRS 15 contract system will pick up the customer with business function payer. One more limitation is that one contract can have only one customer, so there are no partner functions in RAR.

The third topic to mention is the timing of contract creation. The standard states that contracts created at nearly the same time need to be combined. Nearly the same time can be understood by different companies in different ways, ranging from the same day to several months.

As an example, the company enters into two contracts, A and B, with the same customer within 15 days of each other. Company policy considers 30 days as "near the same time," so these two contracts will form a *contract combination* and will need to be accounted for (allocations made) as a single contract. Now 90 days later, another contract, C, is sold to the same customer, which adds more products to contract A. This isn't within the 30-day policy so it doesn't need to be combined; however, if contract C isn't sold at SSP, then it forms a *contract modification*. This modification must be accounted for along with the prior two combined contracts, and allocations must be changed accordingly. This would not have been the case prior to IFRS 15, so there is a clear impact on revenue recognized.

All of these things need to be assessed and accounted for before implementing RAR. Clear rules should exist about when contracts will be combined and when contract modification will be applied.

SAP provides you with options to do manual and automated contract combinations. When it comes to automated combination, all contracts that have the same customer and reference ID populated will be automatically combined. The **Cust. Reference** field usually contains the customer purchase order number (see [Figure 5.77](#)).

If we're talking about integration with sales and distribution, the reference ID will be retrieved from the customer reference field in the sales document (table field BSTKD). However, usage of the **Cust. Reference** field is sometimes not enough, so you need to perform contract combination manually. To do this, you can use the Run Revenue Search app.

The screenshot shows a SAP contract details screen. At the top, there are several input fields: 'Rental_reagent Contr:' with value '41019476', 'Net Value:' with value '24,000.00' and currency 'EUR', 'Sold-To Party:' with value '2100031025', 'Ship-To Party:' with value '220001168', 'Cust. Reference:' with value 'Text0YZZ11', and 'Cust. Ref. Date:'. Below these fields are navigation tabs: 'Sales', 'Item Overview', 'Item detail', 'Ordering party', 'Procurement', 'Shipping', and 'Reason for rejection'. At the bottom, there is a 'Description:' field and 'Contract Start:' (01.01.2023) and 'Contract End:' (31.12.2023) fields.

Figure 5.77 Combination Rules

There are two options available for a contract combination: **Perform Contract Combination** and **Quick Combine**. Let's start with the **Perform Contract Combination** option. You'll first need to select the contracts you'd like to combine on the following screen, as shown in [Figure 5.78](#). Enter the contracts in the **Revenue Accounting Contract** field and press **Enter** to view them in the **Result List**.

The screenshot shows the 'Revenue Accounting Contract' selection screen. It includes fields for 'Company Code', 'Operational Document', and 'Created by'. Below these is a 'Results List' section with a table of contracts. The table has columns for 'Revenue Accounting Contract', 'Company', 'Company Name', 'Accounting Principle', 'Customer', 'Customer Name', 'No. of Prof. Obj.', and 'Transaction Price'. Two contracts are listed in the table.

Revenue Accounting Contract	Company	Company Name	Accounting Principle	Customer	Customer Name	No. of Prof. Obj.	Transaction Price
300000000	IFRS	210001168	IFRS	210001168		21	11,779.20
300000000	IFRS	210001168	IFRS	210001168		2	6.00

Figure 5.78 Selection of Contracts

After clicking **Perform Contract Combination**, the **Set Parameters** popup will appear, as shown in [Figure 5.79](#). **Change Type** determines whether the contract combination will be treated as a retrospective (**Contract Modification**) or prospective (**Change of Estimates**) change. If prospective, you need to enter the **Effective Date** when it's becoming active.

The screenshot shows the 'Set Parameters' popup window. It has a title bar with 'Set Parameters' and a close button. There are two main sections: 'Change Type:' and 'Effective Date:'. The 'Change Type:' dropdown menu is open, showing options: 'Change of Estimates' (selected), 'Contract Modification', and 'Change from Inception Date without Re-allocation'. The 'Effective Date:' field also has a dropdown menu with 'Change of Estimates' selected.

Figure 5.79 Change Type Selection

After making your selections, press **Enter**, and you'll get list of POBs to be combined, as shown in [Figure 5.80](#). Click the **Combine Contracts** button to create one contract.

Contract / Operation...	Contr Desc / Perf Oblig...	Leading Perf...	Composition	Standalone Selling Price	Contractual Price from...
Total					11,779.20
30030644	(Rev) No text available				11,779.20
41011717					1,405.98
38813	(REAGENT (RR)		Distinct	79.46	0.00
38814	(REAGENT (RR)		Distinct	11,753.92	0.00
38815	(REAGENT (RR)		Distinct	113.20	184.50
38816	(REAGENT (RR)		Distinct	158.74	186.00
38817	(REAGENT (RR)		Distinct	298.24	989.10
38818	(REAGENT (RR)		Distinct	2,856.00	0.00
38819	(REAGENT (RR)		Distinct	4,220.00	0.00
38820	(REAGENT (RR)		Distinct	3,019.45	0.00
38821	(REAGENT (RR)		Distinct	75.72	0.00
38822	(REAGENT (RR)		Distinct	181.41	0.00
38823	(REAGENT (RR)		Distinct	27.65	21.26
38824	(REAGENT (RR)		Distinct	27.65	21.26

Figure 5.80 Selection of POBs for Combination

For the **Quick Combine** method, you only have one option: what you're creating as a new contract, and the system will take all POBs from both into one. The difference between quick combine and regular combination is that in quick combine, all POBs from one contract will be merged into another, so you can't choose partial combination.

[Figure 5.81](#) shows the screen for selecting your target contract for a quick combination, which can either be the top contract or a new contract that you create.

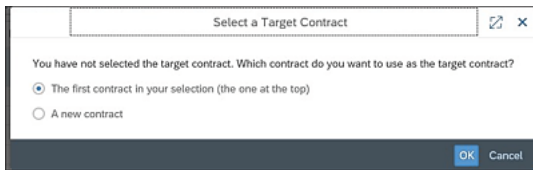


Figure 5.81 Quick Combine

However, if you want to automatize contract combinations, you need to do some development. There is a standard solution for this which SAP has provided in standard class `CL_FARR_RAI2_BADI_IMPL`, called *automatic contract combination*.

Run Transaction SE18, and search for the BAdI. The standard class has method **COMBINE_CONTRACT**, as shown in [Figure 5.82](#) with its parameters.

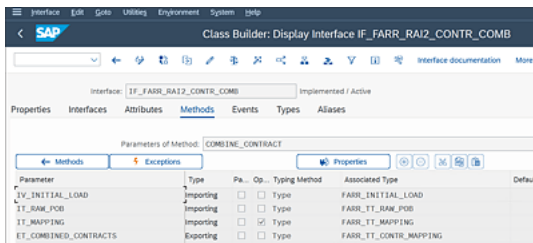


Figure 5.82 Standard Class for Contract Combination

The parameters have importing table **IT_RAW_POB**, which is of type **FARR_IT_RAW_POB**, which is the table type for structure type **FARR_S_RAW_POB**.

The reference ID and reference type fields from the importing parameter **IT_RAW_POB** are the crucial fields for contract combination. The RAIs that have the same reference ID and reference type are combined into a single revenue accounting contract and processed together. There is logic for combing the contracts based on this factor in the `COMBINE_CONTRACT` method. The method is called for both the initial load during migration and for the productive run. You can view the source code by double-clicking on the method, as shown in [Figure 5.83](#).

```

Method: IF_FARR_RAI2_CONTR_COMB-COMBINE_CONTRACT active
1 METHOD if_farr_rai2_contr_comb-combine_contract.
2 DATA its_raw_pob TYPE ty_ts_raw_pob.
3
4 its_raw_pob = it_raw_pob.
5
6 IF NOT iv_initial_load IS INITIAL.
7   * Process INITIAL Load
8   combine_initial_load( EXPORTING its_raw_pob = its_raw_pob
9                       IMPORTING et_combined_contracts = et_combined_contracts ).
10
11 ELSE.
12   * Process PRODUCTIVE run
13   combine( EXPORTING its_raw_pob = its_raw_pob
14          IMPORTING et_combined_contracts = et_combined_contracts ).
15
16 ENDIF.
17 ENDMETHOD.

```

Figure 5.83 Combine Contract Method

Figure 5.84 shows the logic behind this method. You see that there are two methods called inside method IF_FARR_RAI2_CONTR_COMB-COMBINE_CONTRACT: COMBINE_INITIAL_LOAD and COMBINE. We'll discuss the details of that next.

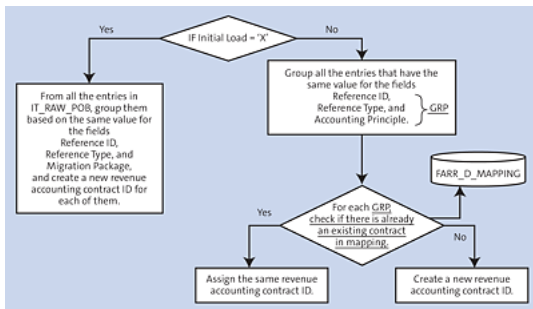


Figure 5.84 Logic for Contract Combination

Details of the methods called are explained as follows:

- Method COMBINE_INITIAL_LOAD: Initial load during migration**
 Method COMBINE_INITIAL_LOAD combines all the entries in the IT_RAW_POB that have the same value in fields REFERENCEID, REFERENCE TYPE, and MIGRATION PACKAGE. They are grouped together in the single revenue accounting contract. As a new combination of these three fields is found, then a new revenue accounting contract number is generated, and they are combined. In this method, there is no selection from table FARR_D_MAPPING because this is the migration run.
- Method COMBINE: Productive run combination**
 In the productive run for method COMBINE, all the items of importing parameter IT_RAW_POB are grouped together and have the same value for fields REFERENCEID, REFERENCE TYPE, and ACCOUNTING PRINCIPLE. This set is then checked to see if there are already corresponding entries in table FARR_D_MAPPING for the field combination. If there are entries in the mapping table, then those entries are assigned the same revenue accounting contract number. If there are no entries found in table FARR_D_MAPPING, then for this new combination of fields REFERENCEID, REFERENCE TYPE, and ACCOUNTING PRINCIPLE, a new revenue accounting ContractId is created and assigned.

These methods use the reference ID, reference type, and accounting principle for grouping the RAIs for contract combination. There will be situations where you'll need to group RAIs based on different fields. That is when you have to perform contract combinations based on other standard fields or even customer fields other than the reference ID, reference type, and accounting principle. To handle this requirement, you have the option of using enhancement spot FARR_ARL in which you have BADI definition FARR_BADI_CONTRACT_COMBINATION.

Enhancement spot FARR_ARL has four BAdI definitions of which three have already been discussed in [Chapter 4, Section 4.6](#), so we'll discuss the one we missed here. We'll look into the details of that BAdI in [Table 5.18](#).

BAdI Definition	Enhancement Spot	Interface	Description	Point Which It's Called
FARR_BADI_CONTRACT_COMBINATION	FARR_ARL	IF_FARR_RAI2_CONTR_COMB	Determine contract ID per RAI	Executed during processing of RAIs

Table 5.18 Details of the BAdI Definition

Enhancement spot FARR_ARL and BAdI FARR_BADI_CONTRACT_COMBINATION can be implemented using the general implementation steps. This BAdI is executed during processing of RAIs. It's only called for order item RAIs and only as long as no POB exists in the results analysis engine that represents this order item.

Go to Transaction SE18, enter the enhancement spot as "FARR_ARL", and then click on **Display** to see this BAdI definition. When you double-click on BAdI interface **IF_FARR_RAI2_CONTR_COMB** (not shown), you can see that there is only one method: **COMBINE_CONTRACT** with signature and import/export parameters, as shown in [Figure 5.85](#).

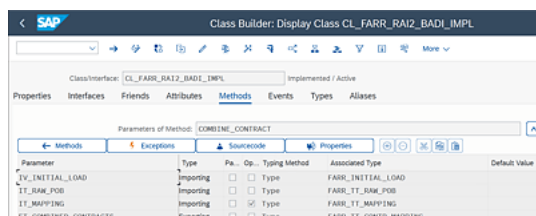


Figure 5.85 Method COMBINE_CONTRACT

This method is executed before RAIs of the order item type are processed.

You can see that it matches with parameters of the standard method for contract combination as explained previously. Importing table IT_RAW_POB has structure type FARR_S_RAW_POB.

Custom fields are also available for you to use in your combining logic. You can implement any required logic to combine the RAIs per your requirement to have the same revenue accounting contract ID. The IT_MAPPING parameter has all table FARR_D_MAPPING entries relevant for this set of entries, so you can validate the new combination with the mapping table entries and assign the same revenue accounting contract ID if available; otherwise, you can create a new one and use it further.

5.4.3 Terminating Contracts

To define a contract, it needs to be enforceable. In many cases that means the party who decides to leave the contract before its expiry will suffer some kind of consequences. When one of the parties decides to cancel the contract before its time, this is a special case of contract modification called *contract termination*.

If the customer decides to do this, there must be some consequences, which depend on the type of contract (did it include only goods or a combination of goods and services?) and the type of process that will follow after the contract is terminated (e.g., will there be returns if goods are involved, is there some kind of termination fee to be paid, etc.).

In the standard itself, contract termination isn't covered separately because of the different nature or different terms and conditions that each contract might have. It's mentioned in part with the enforceability of payments and in general in parts of contract modifications. In addition, the standard is more or less silent when it comes to financial presentations of P&L effects that might occur after the contract is terminated.

Overall, in case of early termination, the entity needs any kind of P&L effect that exists for the future (either contract asset or contract liability) to transfer as a current P&L effect, which is then called an *impairment effect*. The same rules apply for both revenue and cost.

To understand what impairment is and how it's calculated, let's look at one simple example: a customer enters into a contract with a Telco company for 24 months for a service and a device. The calculation of revenue recognized looks like [Table 5.19](#).

POB	Transactional Price	SSP	Allocation %	Allocated Amount	Point in Time	Monthly Revenue	Contract Asset
Device	1.00	2,000.00	50%	1,000.50	1,000.50		958.81
Service	2,000.00	2,000.00	50%	1,000.50		41.69	
Total	2,001.00	4,000.00	100%	2,001.00			

Table 5.19 Contract Termination Calculation

In this simplified case, you see that because both device and service have SSPs that are equal, the amount of revenue allocated to them will be same. Because we're recognizing revenue that is allocated to the POB immediately (as a point in time), the whole amount will be represented as a contract asset. However, revenue that is allocated to service will be recognized as over time, equally for the whole duration of the contract. Accounting wise, this means that this revenue will offset contract assets until the end of the contract when assets will be zero. In essence, contract assets in this case represent future service revenue that will be recognized.

What happens when a customer can't honor their contract? In this case, you need to balance contract assets to zero because you're not expecting any more future revenue. The correct thing to do here is to repost whatever the outstanding balance is on the contract asset side to P&L as an impairment cost.

What about Termination Fees?

From the previous example, it's clear that if a customer decides to walk away from the contract, there will be a P&L impact for a service provider. Usually, in the contract, there

is a term regarding the fee the customer needs to pay if he decides to cancel the contract early. Now the question is should you include those fees as part of the contract or as a separate POB. Though this would be technically possible (still, technical integration for fulfilling that POB might be challenging), the business feasibility of doing it would be questioned. Remember that the standard doesn't give any clear guidance when it comes to recording termination fees, so the best solution is to not complicate the design further by including it in the contract.

Early termination in RAR starts by configuring proper BRFplus decision tables. Open application for account assignment (**YFARR_ACC_DETERMINE**), as shown in [Figure 5.86](#).

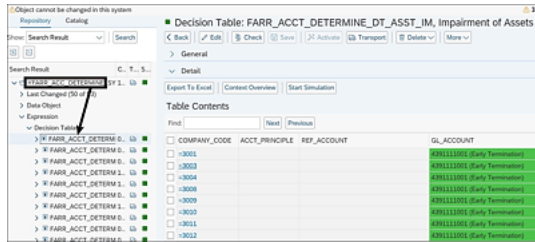


Figure 5.86 Early Termination Account Determination

Scroll to decision table **FARR_ACCT_DETERMINE_DT_ASST_IM**. Here you need to make a mapping between the company code (**COMPANY_CODE**) and general ledger account (**GL_ACCOUNT**) that will be used as a termination impairment account.

Once you're done with account determination, you need to look at the process. If the client is using integration with sales and distribution, the event for termination will come from the sales document. Here, you need to use the cancellation procedure in the **Termination** tab of the sales document, together with the cancellation date to represent that the whole document is terminated.

When such a document is processed, you'll receive RAIs created with populated data that includes the early termination (**Early Term**) flag, as shown in [Figure 5.87](#), and the termination date.

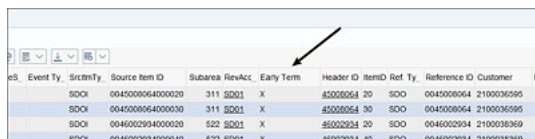


Figure 5.87 Transaction FARR_RA1_MON with Terminated Items

Once you process these RAIs, the terminated contract in RAR will be updated. However, there will be no additional postings happening until you run program B—calculate contract liability (see [Chapter 6](#)). At that moment you run that program, table **FARR_D_POSTING** will get updated with the necessary information, and the finance document is ready to be created. [Figure 5.88](#) shows the content of table **FARR_D_POSTING** after the ABC programs are run. **CA** and **CL** were balanced to zero, and there is a new category, **AI**, which contains the impairment value.

Following is list of things to have in mind while executing early termination:

- All RAIs will have the same value for the **Early Term** flag, which means either all or none are marked as early terminated.
- All RAIs with the **Early Term** flag will have the same effective date. This effective date is used to determine the end date of the revenue contract.
- The effective date can't fall within a closed period.
- The effective date is taken as the early termination date.
- When the early termination occurs on a revenue contract, revenue is recognized up to the effective date of the early termination. If there has been revenue recognized after (including) the early termination date, the revenue recognition is reversed.

CoCd	AcqP/Reconca Key	POB	OnTy	Catg	Dic	Posting GUID	Year	Per	ITC	Amount	Cur
				AI						- 2.786,17	EUR
				CA						+ 0,00	EUR
				CL						+ 895,61	EUR
				IC						+ 2.887,50	EUR
				RA						+ 0,00	EUR
				RV						+ 766,39	EUR
										+ 0,00	EUR

Figure 5.88 Termination Postings

Afterward, the revenue contract information is updated as follows:

- The contractual price of the POB is adjusted to the recognized revenue on the pricing conditions of the POB (except the allocation difference condition).
- The allocated amount is the cumulative recognized revenue of the POB (all price conditions, including allocation difference).
- For time-based POBs, the end date is updated to the effective date minus 1.

An entry is recorded in table FARR_D_POB_CTYPE for all POBs with the prospective change type.

5.4.4 Contract Freeze

Freezing a contract is one of the options that often exists between seller and customer, especially in subscription-based businesses. For example, users are often given an option to suspend or freeze a contract for a certain period of time during the contract duration. This process has different names depending on the line of business: contract freeze or contract suspension. This process can be, but isn't necessarily, followed by contract extension for the same time that the contract was suspended (see [Figure 5.89](#)).

Telco Example

Often in some countries, especially where expats represent a large number of total users of services, there is an option that one or more times during the contract duration, services are suspended, meaning that the subscription is put on hold. This process is then resumed by extending the same number of months for the existing contract. However, if a customer is unable to pay, then for a certain period, their services will be suspended, but the contract won't be extended for that period. So, the extension of the contract is an option, not a regular case, when it comes to contract suspensions.

Once the user signs the contract establishing that the service they subscribe to will be delivered over time where they pay for and consume the service simultaneously, this again means revenue will be recognized as time based. Now, after six months of using a service, the customer is opting for a contract suspension for two months because they have no need for it in the mentioned period. The company decides to extend their contract for the same time, and during the period of suspension, the customer is unable to use their service, meaning that revenue can't be recognized for that time.

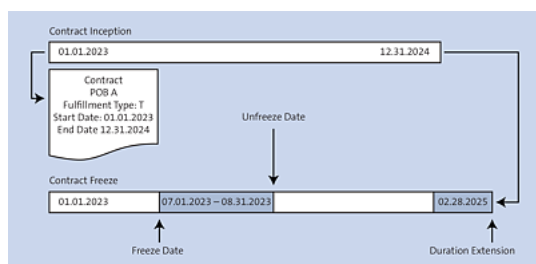


Figure 5.89 Contract Freeze Process with Extension

If the SAP freeze/unfreeze functionality is used, then no contract changes will be created. The Simple Object Access Protocol (SOAP) service called Finance Freeze Period – Apply only adjusts the fulfillments of the time-based POBs and doesn't change the contractual price of the POB-related or allocation-related fields. After freeze periods have been applied, the number of fulfillment days is recalculated considering the freeze periods, and the revenue is updated accordingly.

Before we go into how the freeze/unfreeze functionality is implemented, there are certain rules that need to be followed:

- **Deferral method**

It's strongly recommended to use deferral method 1 due to differences in calculation that can occur, as shown in [Table 5.20](#).

Here you have a period that got frozen on 02.20. Because every month is calculated as the real number of days, you have days calculated as frozen, which exactly matches the number of days the contract wasn't used (in this case, 9 because on 03.01, service was resumed).

Start Date	01.01.2023
End Date	12.31.2023
Freeze Date	02.20.2023
Unfreeze Date	03.01.2023
Number of Days Frozen	02.28 - 02.20 = 9 (20th included)

Table 5.20 Freeze Periods with Deferral Method 1

Now, in [Table 5.21](#), using deferral method 2, you treat all months as 30 days (equally). Therefore, what you get when you implement the same freeze as in the previous case is two extra days because those days were when service wasn't used. This is due to the fact that every month is treated as 30 days, so you get 11 as a result.

Start Date	01.01.2023
End Date	12.31.2023
Freeze Date	02.20.2023
Unfreeze Date	03.01.2023
Number of Days Frozen	02.30 - 02.20 = 11 (every month = 30)

Table 5.21 Freeze with Deferral Method 2

- **Start date**

The POB start date must not be empty (important to remember when you define the time-based POB type).

- **Overlapping periods**

Freeze periods for the same POB must not overlap. For example, it's not allowed to have freeze and unfreeze in the same period.

- **Prospective changes**

If a new freeze period is created, it must be after any date of prospective change.

- **Processing**

The API for processing freeze dates must be run after all RAIs are processed because contract modification can be triggered by them.

To implement the freeze/unfreeze functionality, you need to implement SOAP service RevenueAccountingContractSetFreezePeriodsIn. To do so, the first step is to run Transaction SOAMANAGER, which takes you to the screen shown in [Figure 5.90](#).

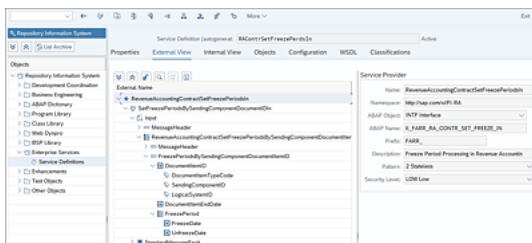


Figure 5.90 Transaction SOAMANAGER for the Freeze Service

If you switch to the **Internal View** tab, as shown in [Figure 5.91](#), you'll find all the details that need to be sent from the source system so the service can be used.

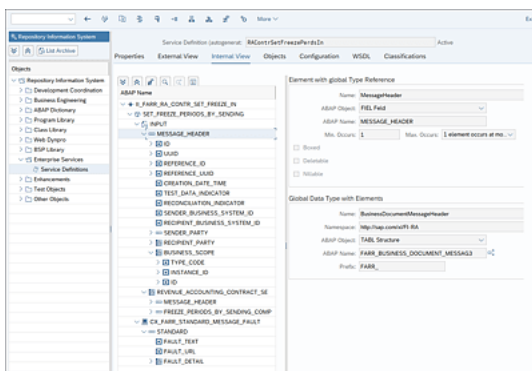


Figure 5.91 Services Available

Most important is that the source system also manages identification of POBs and is able to send data in the needed format by web service. Once the freeze data is processed, it will be saved in table FARR_D_POB_FRZ.

Freeze in Classic Contract Management

The freeze functionality was introduced and came with OCM, so it's not available in classic environments. The question is, what do you do when you need to handle freeze periods in CCM? You have two options. The first is that it will work with contract modification options (e.g., when the service is on suspend mode, it will have an end date, and later the new POB will be created with the new start date that will be equal to the unsuspend date). Revenue recognition won't be executed for that time, but it will happen by triggering prospective contract modification.

Another option that is much more complicated is the creation of a customer deferral method to handle freeze periods. Here, you need to be extremely careful because creation of custom deferral methods is one of the most sensitive enhancements that can be performed in RAR.

5.5 Handling Price Allocations

Once you introduce the SSP into the engine, there will be a change of revenue given to each POB, which is referred to as allocation. In this section, we'll describe how the allocation engine works because it's one of the most important areas of RAR as a tool for satisfying IFRS 15/ASC 606 requirements.

The allocation engine is the core of RAR's method for allocating the transaction price to the POBs. The engine provides proper amounts calculated to the POBs and plans how these POBs are fulfilled. The result of the allocation engine's work is shown in [Table 5.22](#).

	Device	Service	Total
Transactional Price	100	2,400	2,500
SSP	1,000	2,400	3,400
Allocated Percentage	29.41	70.59	100
Allocated Revenue	735.29	70.59	100
Allocation Effect	635.29	-635.29	0.00
Point in Time	735.29		
Monthly Revenue		73.53	
Start Date		05.01.2023	
End Date		04.01.2025	

	Device	Service	Total
Revenue First Month	735.29	63.73	799.02
Billing Amount	100	100	200
IFRS 15 Correction	-635.29	36.27	-599.02

Table 5.22 Allocation of Transactional Price to POBs

The system first compares SSPs and gives a portion of the transactional price that should be given to each POB. Besides this, the ratio is calculated of what portion of total revenue is given to each POB. This amount is compared to the transactional price to calculate the allocation effect, which represents how much or how little revenue POBs will get compared to what you'll bill the customer. The allocation effect must be zero at the end because you can recognize more than the transactional price.

The last step in allocation is calculation of spreading, that is, how revenue will be recognized over time. In this case, the system separates event-based POBs from time-based POBs. With event-based POBs, revenue will be recognized at the point in time, and time-based POBs will have a revenue schedule.

POBs can be excluded from allocation in two ways:

- SSP of a POB is equal to 0. In this case, the formula for allocating revenue gives 0 as a result ($SSP \div Total\ SSP \times Total\ transactional\ price$).
- POBs can be excluded from allocation either manually or by setup. See [Section 5.2.1](#) for details.

In [Figure 5.92](#), you can see the results after allocation. The system will show you **Contractual Price** (the amount that is allocated), **Allocated Amount** to specific POBs (amount according to SSP), and **Allocation Effect** (the difference between these two).

Performance Oblig.	Performance Obligat...	Leading/Linked	Leading P...	Contractual Price	Allocated Amount	Allocation Effect
<input type="checkbox"/> 72347	REAGENT (RR)			364.732,55	0,00	364.732,55
<input type="checkbox"/> 72348	REAGENT (RR)			145.888,43	0,00	145.888,43
<input type="checkbox"/> 72349	REAGENT (RR)			244.145,65	0,00	244.145,65
<input type="checkbox"/> 72350	REAGENT (RR)			0,00	2,44	2,44
<input type="checkbox"/> 72351	REAGENT (RR)			0,00	2.564,46	2.564,46
<input type="checkbox"/> 72352	REAGENT (RR)			0,00	223,65	223,65
<input type="checkbox"/> 72353	REAGENT (RR)			186,80	3,03	183,77
<input type="checkbox"/> 72354	REAGENT (RR)			499,80	3,40	496,40
<input type="checkbox"/> 72355	REAGENT (RR)			0,00	1.045,65	1.045,65
<input type="checkbox"/> 72356	REAGENT (RR)			0,00	696,94	696,94
<input type="checkbox"/> 72357	REAGENT (RR)			0,00	261,33	261,33
<input type="checkbox"/> 72358	REAGENT (RR)			569,65	1,30	568,35

Figure 5.92 Allocation Results

5.6 Summary

In this chapter, we covered topics spanning the complete lifecycle of a contract: starting from configuration, real-life examples, and examples with developments that are commonly used. This idea was to present the complexity of setting up RAR in an environment where IFRS 15 needs to be reported. The key message here is that all the topics need to be looked at first from a business point of view to determine how the contract should be defined, whether contract modification is necessary, and what the POB types and fulfillment events are. Later we put it together with what kind of setup is needed to be performed for these requirements to be met and, if necessary, examples of developments.

All of this gives a comprehensive picture of how basic things such as setting up contracts and POBs are key points in your IFRS 15 project. It can't be done in a silo where the business team is giving a requirement, and IT will simply execute a configuration. It requires business understanding from IT, but also understanding of how the engine works from the business side. This is the only way you can ensure that results are according to expectations and the setup isn't too complex.

A key part is the use of enhancements in the standard setup. SAP provides a lot of options to extend how the engine works, but special attention is needed before you decide to use them. We gave an example wherever

possible, along with warnings about when these enhancement options should be used with extra care.

In the next chapter, we'll continue by covering RAR posting and reporting options.

6 Revenue Posting and Reporting

So far, we've covered revenue recognition topics from the beginning: forming contracts, determining performance obligations (POBs), and calculating allocated revenue. Now that these activities are completed, we're ready to take the final steps: creating the documents that will represent the postings of our calculated revenue.

In this chapter, we'll focus on the revenue accounting and reporting (RAR) functionality in SAP S/4HANA from the end user perspective. We'll start by walking through key reporting concepts for revenue recognition. Then, we'll provide step-by-step guidance for posting using ABC programs, reconciliation with the Universal Journal, and integration with profitability analysis. We'll round out this chapter with information on revenue recognition reporting with SAP S/4HANA, including the latest tools such as SAP Fiori apps and core data services (CDS) views.

6.1 Basics of Reporting and Calculations

Before explaining revenue recognition posting logic in detail, it's important to start with the basic setup. Posting logic is determined in two places:

- BRFplus for account determination
- Additional specifications needed for making postings in some specific situations

But first, you need to understand how posting logic works and which settings are made at what step to make posting successful. Several sets of accounts are used to come up with International Financial Reporting Standards (IFRS) 15 revenue. We can start by explaining the basics of how posting works, which will be covered by a detailed walk-through in the following sections.

6.1.1 Posting Logic

RAR is primarily focused on calculating results: once data is calculated, it's stored in table `FARR_D_POSTING`, which is the staging area for data before it's sent to the general ledger in the form of financial accounting documents. Data enters this table as a result of three activities:

- Reversal of invoice coming from sales and distribution or another source application

- Running program for revenue transfer (program A; see [Section 6.2.1](#))
- Running program for liability calculation (program B, used both for contract assets/contract liabilities [CA/CL] and unbilled receivable/deferred revenue [UR/DR] methods; see [Section 6.2.2](#))

Let's look at the example shown in [Table 6.1](#) to see how postings are made in RAR. The contract has two performance obligations (POBs), with an allocation effect where one POB will be recognized as point in time and one as over time.

POB	Transactional Price	SSP	Ratio	Allocated Amount	Adjustment	Duration
Device	100.00	500.00	29.41	382.35	282.35	N/A
Service	1,200.00	1,200.00	70.59	917.65	-282.35	12 months
Total	1,300.00	1,700.00	100.00	1,300.00	0.00	

Table 6.1 Basic IFRS 15 Calculation

In this case, we have two POBs that are part of one contract, so we have a total contractual price being allocated among them. The contractual price of 1,300 is split between them according to the standalone selling price (SSP), which is 500 and 1,200, respectively. The company is giving a device discount but no discount for service. In addition, revenue for device is recognized as point in time, while service is recognized as over time for a duration of 12 months (in further explanations, we'll assume 360 days duration of a year for simplicity).

Because we're invoicing the device POB at the moment of contract inception, we'll have an invoice of 100 posted immediately. Together with that invoice, the first month of the service POB will be invoiced, which is $1,200 \div 12 = 100$, making the total invoice to the customer equal 200.

Now, the adjustment column tells us how revenue should be adjusted per each POB: device revenue needs to be increased by 282.35, which happens immediately because revenue is being recognized as point in time. The service POB revenue will be decreased by the same amount, but that is going to happen over a period of 12 months, meaning that revenue for each month needs to be decreased by $282.35 \div 12 = 23.53$.

As mentioned, when the contract is created, no postings will occur, and the first posting comes with invoicing to the customer, which will be reversed in RAR. For this reversal, RAR will use the receivable adjustment account (for balance sheet reversal) and revenue allocation account for profit and loss (P&L) reversal. After processing, postings (which are stored in table FARR_D_POSTING) will be as shown in [Figure 6.1](#).

Accounts Receivable	Billing revenue - Device	Billing revenue - Service	
300.00			Posting of invoice in SD
	300.00		Reversal of invoice in RAR
		300.00	
Receivable adjustment	Invoice correction - Device	Invoice correction - Service	
	282.35		
		17.65	

Figure 6.1 Postings after Invoice Reversal

If we look at results of postings on financial statement level, we have balance sheet part reversed and P&L is also balanced to zero.

Now, the next step is to perform calculation of revenue amount that should be recognized. RAR will make do this by posting recognized revenue amounts and revenue adjustment. These postings can be done on separate or same accounts. To cover the end-to-end scenario, we'll illustrate the case of separate accounts. The first step is posting revenue that is coming from the billing, and then adjusting it with the amount to come to IFRS 15 revenue. All postings will be balanced with the receivable adjustment account. In addition, it's worth of mentioning that these postings will be made by running the Transfer Revenue program (program A; see [Section 6.2.1](#)).

Revenue transfer takes into consideration any fulfillment event that might occur in the meantime and enters data in table FARR_D_POSTING as revenue recognized. As shown in [Figure 6.2](#), the balancing amount will be entered against the receivable adjustment account.

Accounts Receivable	Billing revenue - Device	Billing revenue - Service	
300.00			Posting of invoice in SD
	300.00		Reversal of invoice in RAR
		300.00	Revenue transfer
Receivable adjustment	Invoice correction - Device	Invoice correction - Service	
282.35	282.35		
		17.65	
	Revenue adjustment - Device	Revenue adjustment - Service	
		282.35	

Figure 6.2 Revenue Transfer Calculation

Looking at the balance sheet and P&L, we see that total revenue recognized fits the IFRS 15 calculation: device revenue was increased by 282.35, and service revenue was decreased by 1/12 of same amount. Now, we can see and evaluate the effect of IFRS 15: the total amount of revenue isn't changing; it will again correspond to the total price, which is agreed upon with the customer. However, there was a significant shift of the revenue between device and service, and now it's a visible effect on the hidden discount given to customers, so the customer subscribes to the service.

But the process isn't over. The balance of these postings is done against the receivable adjustment account, which, in this case, serves as a placeholder, a temporary account until it's verified whether it can be represented as a contract asset or a contract liability.

In this case, we'll be able to see the difference between calculation based on CA/CL and UR/DR. If we use the UR/DR method, it's enough that the invoice is posted, and the balance on the receivable adjustment account will be moved to an unbilled receivable. If we're using the CA/CL method, we need to wait until that invoice becomes due, and only then it will be represented as CA or CL.

Technically, we need to run program B (liability calculation, as it's used for UR/DR as well; see [Section 6.2.2](#)) so that this balance is moved to the proper category. After running this job, we see the effect of a complete cycle in RAR.

As shown in [Figure 6.3](#), the receivable adjustment account has a balance that is different from billing, which was reversed, and revenue that was recognized by running the revenue transfer program. This balance is reposted as a contract asset or contract liability.



Figure 6.3 Contract Liability Calculation

In this example, the amount is due, and it was represented as a contract asset. From next month, the only correction that will be posted is a service adjustment that will result in a decrease in contract assets, which will be balanced to zero if there are no changes to the contract in the next 11 months.

The company needs to understand that the preceding activity is the complete effect of implementing IFRS 15. In comparison to IAS 18, there will be a reallocation of revenue between point in time and over time revenue; however, the total amount of revenue won't be changed.

6.1.2 Customization for Posting of Revenue

As mentioned, customization required for postings is done in two areas: first, as general customizations in Transaction FARR_IMG, and, second, as an application in BRFplus later to come up with proper accounting entries.

We can start with setting up general customizations once we're done with general RAR settings (accounting principle, POB types, etc.; see [Chapter 4](#)). As for other dependencies, there is a dependency on general settings in financial accounting that need to be done, such as creation of a document type, general ledger accounts, and so on. We'll walk through the settings for postings and account assignment in the following sections.

General Posting Settings

You'll see all the settings needed for posting under **Revenue Accounting • Revenue Accounting Postings**, as shown in [Figure 6.4](#).

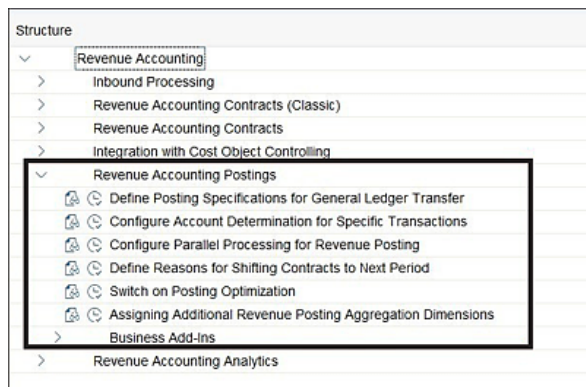


Figure 6.4 Setup of Posting Specification

The first step needed is the setup of document types and posting keys, which can be different per company code defined in RAR. Choose the **Define Posting Specifications for General Ledger Transfer** node, and you'll arrive at the screen shown in [Figure 6.5](#). Here, click the **Change** button, and, for each company code, enter a posting key (two separate columns, first for credit [**C...**] and after for debit [**D...**]), document **Type**, and **G/L Account** with the account assignment object to be used.

Define Posting Specification for General Ledger Transfer							
CoOd	C.	D.	Type	G/L Account	Segment	Profit Center	BusA Functional Area
<input type="checkbox"/>	3001	40	RR	9999911000		9999900001	
<input type="checkbox"/>	3001	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3004	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3008	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3009	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3010	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3011	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3012	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3013	50	RR	9999911000		9999900001	
<input type="checkbox"/>	3014	50	RR	9999911000		9999900001	


Figure 6.5 Details of the Posting Specification

Besides document types and posting keys, you also need to define a clearing account that will be used if the number of items is more than 1,000. The number of items that can be included in a general ledger document is limited. If the number of items to be posted during the transfer of totals records to the general ledger exceeds this number, it's necessary to split the documents. Typically, the general

ledger documents created by a document split don't have a balance of zero. You reach a balance of zero for these documents by posting to a transfer account that you specify here.

In addition to the account number, you can enter other account assignment data for the transfer account, such as **Segment**, **Business Area**, and **Profit Center** in this Customizing activity. SAP suggests you manage the transfer account as a balance sheet account.

The next step is the definition of reasons why contracts might be moved to the next period. Choose the **Define Reasons for Shifting Contracts to Next Period** activity, and you'll arrive at the screen shown in [Figure 6.6](#).



Reasons for Shifting Contracts to Next Period	
Reason	Description
1	Posting Error

Figure 6.6 Shifting Reasons

In this Customizing activity, you can define reasons for shifting contracts to other revenue accounting periods. When the accountant runs revenue postings at the end of an accounting period, some issues may prevent the postings from being successfully transferred to the corresponding ledgers. If the issues can't be solved, the accountant can choose to shift unfinished postings into the next accounting period. When performing such shifting, the accountant must specify a reason that explains why the shift in postings is necessary.

The Customizing activity maintains a list of predefined reasons, each identified with a **Reason** code and labeled with a **Description**. When the accountant specifies the shifting reason, you can choose one from this predefined list. To define the shift reason, it's enough for you to define the number and description. Once done, click **Save**, and the reason is ready to be used.

Now, navigate to the **Switch on Posting Optimization Customizing table** configuration activity, as shown in [Figure 6.7](#).

CoCd	AccP	Post. Opt	Agg. by D/C ind.

Figure 6.7 Switch on Posting Optimization Customizing Table

Posting optimization is one way the number of postings can be decreased per finance document. To reduce the number of general ledger documents, RAR aggregates posting items in the revenue accounting subledger as much as possible. Posting items with the same account assignments are aggregated together. As a result, the higher the aggregation level is, the fewer the posting items are.

However, if there is any attribute that is specific for each contract in the posting table, the revenue accounting subledger can barely be aggregated. For example, the profitability segment contains sales order numbers that are different for each contract. In such cases, we recommend you enable posting optimization. When this function is switched on, general ledger documents are posted and created by technical parameter KEYPP (0-999). Although the number of general ledger documents may increase with this function, your memory consumption can be reduced.

This function has a negative impact on the aggregation of posting items. Enable posting optimization, and the number of general ledger documents will increase, so we highly suggest you don't enable this function unless your aggregation level is very low. If you enable posting optimization when your aggregation level is high or even medium, you can expect worse performance and increased general ledger document numbers. For example, if posting optimization is enabled and each KEYPP contains posting items, RAR splits the posting volume into 1,000 packages that can be posted in parallel. As a result, at least 1,000 general ledger documents will be posted, which could be much more than that without posting optimization.

To perform the posting check, select **Simulation** as the posting mode, and start a revenue posting job. The system will check all selected contracts step by step and then perform revenue postings. The contracts posted successfully are entered into the general ledger, while the contracts with errors are skipped. In the case of posting to the general ledger, depending on the accounting bases, all postings are made to the relevant ledger group.

If you choose to aggregate the general ledger postings by debit/credit (**Agg by D/C ind**), you'll have the benefits of detailed posting records in the revenue accounting subledger table. Post revenue adjustments and contract asset amortization (or contract liability) in the general ledger.

In this case, you can aggregate the postings from the subledger table to the general ledger when you use the Revenue Posting program ([Section 6.2.3](#)). This is a Customizing option that allows you to decide whether you want to aggregate the general ledger posting lines by debit/credit indicator. This Customizing is based on the company code and accounting principle, and it should be kept the same.

If you trigger posting optimization, when some specific fields have the same value, the general ledger posting lines will be aggregated. Even if they have a different debit/credit indicator, the posting lines are still netted.

With this functionality on, you can still continue to use the revenue accounting subledger (table FIRA_AGGREGATE_BY_DC_INDICATOR) to provide detailed posted data. If you choose to enable posting optimization, we also suggest you enable parallel buffering of accounting documents by taking the following steps:

- Maintain object RF_BELEG in Transaction SNRO.
- Select the **Parallel Buffering** checkbox. The buffer size is recommended to be 1.

The last setting in general settings is to define if there are any additional dimensions that should be included in postings. To do so, go to the **Assigning Additional Revenue Posting Aggregation Dimensions** activity. You can select additional dimensions on the screen shown in [Figure 6.8](#) and click **Save**.

You can generate additional data records by adding fields as aggregation dimensions. Each new field that you add increases the data volume. This has a negative effect on performance and takes up more memory. For this reason, you shouldn't include any more account assignments than you actually require. This is especially true for those fields that can take a large number of values, such as internal order, customer order, or work breakdown structure (WBS) element.

To include customer fields in the general ledger documents that RAR creates, you must extend the RAR (INCL_EEW_FARR_REP) and general ledger (CI_COBL) structures with the same set of fields (the field names must be the same). You can define customer fields by following menu path **Financial Accounting • Financial Accounting Global Settings (New) • Ledgers • Fields • Customer Fields**, arriving at the screen shown in [Figure 6.9](#).

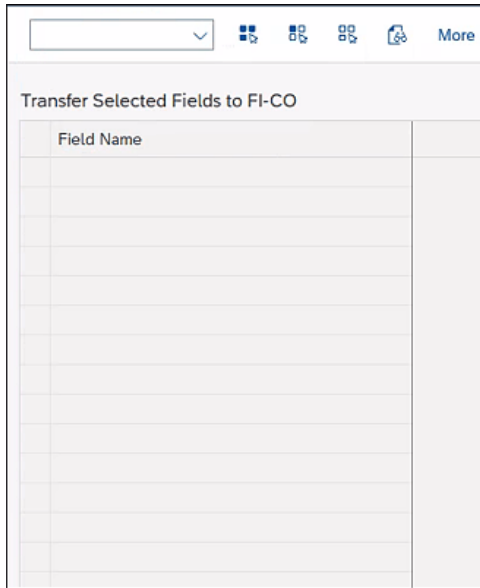


Figure 6.8 Assign Additional Dimensions

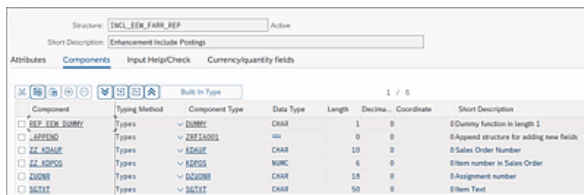


Figure 6.9 Include Custom Dimensions

All fields included in both CI_COBL and INCL_EEW_FARR_REP will be automatically transferred into general ledger documents. You can change the standard general ledger document fields (fields that aren't in the CI_COBL structure) via the following steps:

1. Enhance structure INCL_EEW_FARR_POSTING.
2. Implement business add-in (BAdI) FARR_POSTING_ENHANCEMENT in enhancement spot FARR_ES_POSTING.

Warning

If you add XBLNR to INCL_EEW_FARR_POSTING, don't assign any value to XBLNR. Assigning a value to XBLR will result in an error when posting to financial accounting.

BAdI FARR_POSTING_ENHANCEMENT provides method PROCESS_CUST_FIELDS, which can be used to set noncustomer fields in general ledger documents. Standard field changes might lead to incorrect general ledger documents, so always pay attention that all changes are made at your own risk.

If you use the PROCESS_CUST_FIELDS method, there are two parameters:

- **IS_RR_LINE_ITEM**

All information for a revenue accounting item (RAI; including customer fields defined in INCL_EEW_FARR_REP).

- **CS_ACC_IT**

All fields of the corresponding general ledger document item.

In structure INCL_EEW_FARR_REP, you've defined fields that are used in revenue accounting. If you add a new entry that doesn't exist in structure INCL_EEW_FARR_REP, the system reports an error message.

The overlap fields between structure INCL_EEW_FARR_REP and structure CI_COBL are default aggregation dimensions. If you still want to enhance INCL_EEW_FARR_POSTING with the fields from CI_COBL, then you can overwrite the data from posting table in BAdI FARR_POSTING_ENHANCEMENT. In normal cases, you should set the value to fields that don't exist in CI_COBL but exist in ACCIT.

Account Assignment Settings

With that, the general settings for postings are completed, and to finish integration with the general ledger settings, you need to perform account assignment settings. Similar to POB determination, you'll do that through BRFplus settings.

SAP delivers standard application FARR_ACC_DETERMINE, which needs to be copied to the customer namespace before it can be used. As shown in [Figure 6.10](#), select the application, and choose **More • Copy**.

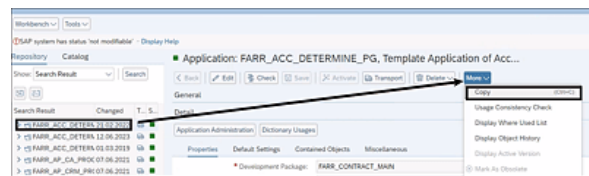


Figure 6.10 Copying FARR_ACC_DETERMINE

This will make a copy of the existing application, which then serves as a template. [Figure 6.11](#) shows the copied **YFARR_ACC_DETERMINE** application in the customer namespace.

Unlike POB determination applications, in account determination, there are no differences depending on the integration method (sales and distribution, SAP Billing and Revenue Innovation Management, or third party), and there is a unique application that will serve to all posting determinations.

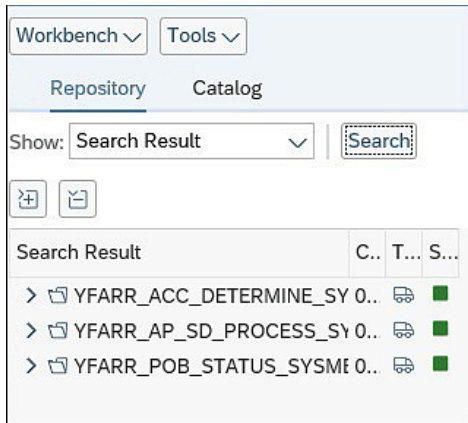


Figure 6.11 Posting Application Copied to the Customer Namespace

Once you open the application, you'll find all the decision tables available there, as shown in [Figure 6.12](#). For more information on these tables, see [Chapter 4, Section 4.5.1](#).

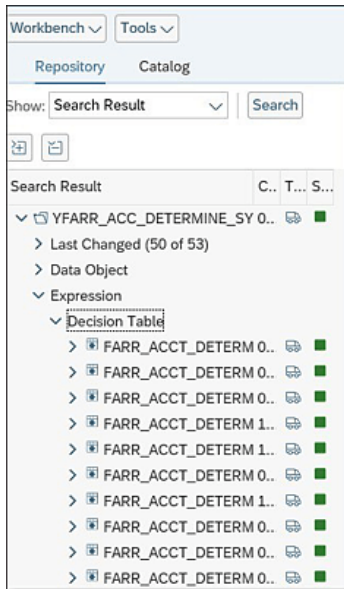


Figure 6.12 Available Decision Tables

Now, the next question is which accounts need to be assigned here. SAP provides many different options for this. We'll explain further only a few additional options that need to be considered in almost all cases.

When it comes to accounts that will be used for revenue reversal and posting adjustments, you might choose to have separate accounts per POB types. In the example shown in [Figure 6.13](#), you can see decision tables used for revenue adjustment postings. Reference account (**REF_ACCOUNT**) is an optional field and can be skipped by entering just POB types (**POB_TYPE**) and company code (**COMPANY_CODE**).

Decision Table: FARR_ACCT_DETERMINE_DT_CORR, Revenue Adjustment for Allo...

General

Detail

Export To Excel Context Overview Start Simulation

Table Contents

COMPANY_CODE	POB_NAME	POB_TYPE	REF_ACCOUNT	GL_ACCOUNT
<input type="checkbox"/> =3001	=INSTRUMENT(Capital Sales)	=Y0001 (INSTRUMENT(Capital Sales))		111111200 (Revenue Alloc Instr)
<input type="checkbox"/> =3001	=INSTRUMENT (LEASE)	=Y0001 (INSTRUMENT (LEASE))		111111200 (Revenue Alloc Instr)
<input type="checkbox"/> =3001	=REAGENT (RR)	=Y0001 (REAGENT (Revenue Rental))		111111201 (Revenue Alloc Reagent)
<input type="checkbox"/> =3001	=SERVICE (BILLING)	=Y0001 (SERVICE (BILLING))		111111202 (Revenue Alloc Service)
<input type="checkbox"/> =3001	=SERVICE(CONTR.FR)	=Y0001 (SERVICE (Contr. Free))		111111202 (Revenue Alloc Service)
<input type="checkbox"/> =3001	=TEST PROFILE (Revenue Rental)	=Y0001 (TEST PROFILE (Revenue Rental))		111111203 (Revenue TestProfile)
<input type="checkbox"/> =3001	=Recalculation contract (POC)	=Z0001 (Recalculation contract (POC))		111111203 (Revenue TestProfile)

Figure 6.13 Revenue Adjustment Postings

However, this is a customer decision. Table FARR_D_POSTING in the RAR posting helps reduce the historical data aggregation. Upon aggregation, detailed postings are replaced with aggregates on an inner-period level or even on a cross-period level. Detailed posting entries from table FARR_D_POSTING are moved from the in-memory storage in this case to lower-cost storage offered with SAP HANA native storage extension (NSE). Aggregations can also be reversed. In addition, you might choose to make this account the same or different as the billing account, and both ways have pros and cons. If you do a reversal on the same account, it's easy to identify whether all billing for the month was performed and whether some was missing (if the balance is different from 0). However, it might impact reporting where you need to look at the account balance only, and you need to include an additional dimension if you need to see if the reversal is from RAR or billing itself.

A similar approach can be taken for adjustment accounts: if you need to see the adjustment effect separately, it might be a good idea to split on a different account; otherwise, the same can be used.

In the example shown in [Figure 6.14](#), decision table FARR_ACCT_DETERMINE_DT_RADJ is used to configure the receivable adjustment account. Under the **Expression** section of the BRFPplus application, find the **Decision Tables** section, and select **FARR_ACCT_DETERMINE_DT_RADJ**. A more standard approach is for the client to have only one receivable adjustment account (**1121211100**, in this example), and to do so, all columns need to remain empty. In that case, all company codes and all accounting principles will share one receivable adjustment account. Another option is that if you enter a reference account (from the customer master record), multiple receivable adjustment accounts can be used.

Decision Table: FARR_ACCT_DETERMINE_DT_RADJ, Recievable Adjustment

General

Detail

Export To Excel Context Overview Start Simulation

Table Contents

ACCT_PRINCIPLE	COMPANY_CODE	REF_ACCOUNT	GL_ACCOUNT
<input type="checkbox"/>	=3001		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>	=3003		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>	=3004		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>	=300		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>	=3009		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>	=3010		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>	=3011		1121211100 (AR-Trade -RAR-Adj)

Figure 6.14 Receivable Adjustment

A similar rule is used for receivable adjustments. The balance, if any, on this account needs to be reported separately. Therefore, it's a good idea to keep it as a different account from CA/CL.

One thing to remember is that the complexity of account determination depends a lot on the accounting principles used (see [Chapter 5, Section 5.1.2](#)). If a company isn't focusing on revenue adjustments based on SSP, but rather makes only revenue deferrals, account assignment might be very simple. However, scenarios with parallel accounting principles, both cost and revenue deferrals, and adjustment calculation can lead to very complicated account determination.

All these options should be taken into consideration before opting for splitting many accounts versus grouping transactions in the same accounts.

It's worth mentioning that besides performing account assignment by using the BRFplus application, you can run an SAP Fiori app for simplified determination. In the SAP Fiori launchpad, you can navigate to the **Revenue Accounting Configuration** section, and choose the **Account Determination - For Classic Contract Management** tile for the app of the same name.

Note

Receivable adjustment integration isn't yet available for optimized contract management (OCM).

The simplified user interface (UI) for account determination allows you to make changes within the boundaries of the existing BRFplus structure for account determination. But for changes in the structure of account determination itself (e.g., adding a reference field), you have to go to BRFplus. The simplified UI contains settings for most of the decision tables in BRFplus, for example, **Recognized Revenue, Receivables Adjustment, Contract Asset, Contract Liability**). It's possible to use other attributes for finding the target accounts (e.g., for **Performance Obligation Type**).

6.1.3 Table FARR_D_POSTING and Revenue Categories

Table FARR_D_POSTING is the main source where all information needed for posting RAR documents is stored. This table will be populated each time an event that is relevant for creation of postings happens. So once the invoice is reversed in RAR, or revenue to be recognized is calculated, entries will be made in this table.

Open table FARR_D_POSTING. Let's first look at the keys in table FARR_D_POSTING, as shown in [Figure 6.15](#).

Search in Table: FARR_D_POSTING Postings

Number of hits: 355

Runtime: 0 Maximum no. of hits: 500

Insert Column:

CoCode	AccP	Reconcl. Key	POB	CnTy	Category	D/C	Posting GUID
AU02	IFRS	20220120000101	105010	ZPR0	RA	S	6045BD8846D21EEDB68A0E6AC15E2633
AU02	IFRS	20220120000101	105010	ZPR0	RV	H	6045BD8846D21EEDB68A0E6AC15E2633
AU02	IFRS	20220120000101	105024	ZPR0	IC	S	6045BD8846D21EEDB0F79CEF64E8ABD4
AU02	IFRS	20220120000101	105024	ZPR0	RA	H	6045BD8846D21EEDB0F79CEF64E8ABD4
AU02	IFRS	20220120000101	105025	ZPR0	IC	S	6045BD8846D21EEDB0F8717D6EE31440
AU02	IFRS	20220120000101	105025	ZPR0	RA	H	6045BD8846D21EEDB0F8717D6EE31440
AU02	IFRS	20220120000101	105025	ZPR0	RA	S	6045BD8846D21EEDB0F8C5AC769458EB
AU02	IFRS	20220120000101	105025	ZPR0	RV	H	6045BD8846D21EEDB0F8C5AC769458EB

Figure 6.15 Table FARR_D_POSTING Keys

The first two entries are company code (**CoCode**) and accounting principle (**AccP**). Company code doesn't require additional explanation, but regarding accounting principle, it's important to repeat that if you opt for parallel reporting by using two accounting principles, for each of them a separate set of entries is made in this table. See [Chapter 5, Section 5.1.2](#), for more information on accounting principles.

Reconciliation key (**Reconcl. Key**) is a way for RAR to distinguish separate events that might occur: each time you perform some activity that might be fulfillment, contract change, or posting an invoice, a new reconciliation key is created. This key has a predefined format: YYYYMMMXXXXXXX. You can notice for the last three digits that they begin with 101 and will continue incrementally for each change that is performed over the contract. The first 100 digits are reserved for contracts that are being shifted from the previous period.

The reconciliation key is determined in table FARR_D_RECON_KEY, and keys are used in most important tables in RAR as part of the unique key, as shown in [Figure 6.16](#).

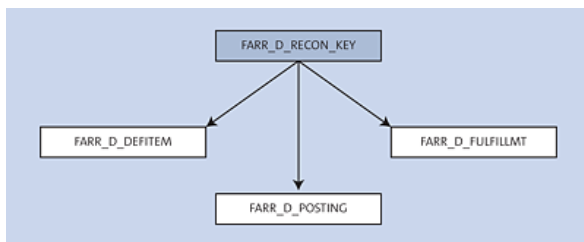


Figure 6.16 Table FARR_D_RECON_KEY Relationship

Looking in this table, you can see who created the entry, when it was done, and its status. Statuses have the following meaning:

- **O = Open**
Events were recorded but no posting exists yet.
- **P = Transferred**
A posting in revenue accounting exists after the revenue transfer run.

- **C = Closed**
A posting in the general ledger exists after the revenue posting run.
- **F = Failed**
Posting to the general ledger failed during the revenue posting run.
- **A = Aborted**
General ledger posting has been reversed.
- **R = Replaced**
Posting was shifted to the next period. The system has created a reconciliation key in the new period that replaces the original one.
- **M = Migration**
Posting was migrated from a legacy system. That means the company code was in a migration phase, and the event date was before the legacy data transfer date.

To close the period, the status of all entries in table `FARR_D_RECON_KEY` must be either **Closed**, or they need to be shifted to the next period.

Impact on Performance and Reconciliation Keys

Now it's becoming clearer how performance might be impacted by having overly large contracts. We already elaborated in this chapter that whenever some event happens in the contract, a new reconciliation key is created, and one of the tables impacted is table `FARR_D_DEFITEM`. Now if you have a contract with 100 time-based POBs and a duration of five years, for each POB, you'll have at least 60×2 (for each condition type) = 120 entries per each reconciliation key and per accounting principle. So, if you summarize for one such contract, you would have 12,000 entries, and each time some change happens in the contract, the number of entries will double. That is one of the reasons for the SAP recommendation to keep the number of POBs lower than a certain amount; more importantly, it's the signal to the client to pay attention while designing contracts with the potential for a large number of POBs.

After the reconciliation key, you'll find the number of POBs (**POB**) to which posting entries are related and condition types and posting categories that need to be looked at together. Posting categories are created by either a specific trigger or a transaction being executed. [Table 6.2](#) shows a list of posting categories and how they are created.

Posting Category	Posting Category Name	How It's Created
------------------	-----------------------	------------------

Posting Category	Posting Category Name	How It's Created
IC	Invoice correction	While reversing an invoice from the source system
RA	Receivable adjustment	As an offset account during the reversal of an invoice and while transferring to CA/CL
RV	Recognized revenue	After running the transfer revenue program
CA	Contract asset	After running the calculate liability program
CL	Contract liability	After running the calculate liability program
ED	Exchange rate difference	After running the transfer revenue program
UR	Unbilled receivable	After running the calculate liability program (if UR/DR is used)
DR	Deferred revenue	After running the calculate liability program (if UR/DR is used)
AI	Early termination	After running the transfer revenue program, if the contract is terminated
CC	Cost correction	After running the transfer revenue program
CJ	Deferred cost	After running the transfer revenue program
CO	Recognized cost	After processing the trigger for cost recognition

Table 6.2 Posting Categories

Some of these categories will be used only if a specific setup is made. For example, **CA** and **CL** will appear only if a specific calculation method is selected in defining the accounting principle, and the same goes for **UR** and **DR**. A similar case occurs for the **CC/CJ/CO** combination: correction of cost of goods sold (COGS) and recognition will appear only if you selected cost recognition as needed.

Some other posting categories will appear only when certain triggers occur. For example, **AI** will happen only if all RAIs from the source system have a flag for termination marked. Similarly, **ED** appears only when contracts are in a foreign currency (irrespective of the method selected, actual or fixed).

Next, we can look at the condition type (**CnTy**) together with the posting category (**Category**). As already mentioned, there is dependency involved regarding which condition type can appear with which posting category. Let's walk through the posting categories to see how they relate to the condition types:

- **IC**

The **IC** posting category will have only the condition type that comes as the main pricing condition, and it will have debit/credit (**D/C**) indicator **S** only. If you think about it, this makes perfect sense: **IC** represents the reversal of revenue coming from the invoice, so it must have only the main pricing condition and will always be a debit (to negate revenue coming from billing).

- **RA**

The **RA** posting category is posted in two situations: as offset category during invoice reversal and while calculating CA/CL. So, in the first case, it will be posted with the main condition type and debit side, but in the second case, it will be posted with the condition type showing the allocation effect and can have both the debit and credit indicator depending on the final result of the calculation: contract asset or contract liability.

- **RV**

The **RV** posting category gets created once recognized revenue is calculated. So, posting will have both condition types: posting with the main condition type and posting for allocation difference. These can occur with both debit and credit indicators because the allocated difference can be increased or decreased to the transactional price (which gets posted with the main condition type).

- **CL and CA**

CL and **CA** (or **UR** and **DR**) won't have a condition type, which makes sense because the condition type represents either the transactional price or allocation effect. Contract asset is the final result, so it can't be split into parts coming from the transactional price and allocation effect, meaning the condition type will be empty. In addition, the POB number will be there if you selected posting to the POB level; if the posting level is contract, the POB number will be empty.

- **ED**

The **ED** posting category has a predefined condition type that will always be entered with the line containing the exchange rate difference.

The last key in table `FARR_D_POSTING` is **Posting GUID**. That data element is known already from CRM systems where it was used as the unique key among the tables. The globally unique identifier (GUID) number can be either a 32-bit or 16-bit element. It's generated based on hardware information from the host computer, the system time, and a randomly generated number. This makes it unique even between different computers/systems, so we can't have two of the same numbers in the world. GUID is a very useful feature that is widely used, especially in system-to-system communication.

Scrolling further through the table, you'll see additional entries, as shown in [Figure 6.17](#).

D	Year	Period	TC Amount	Crcy	LC Amount	LCurr	Second LC	LCur2	Third LC	LCur3	Contract	GL Account	Stat	POB Type
8	2022	12	904.11-	AUD	904.11-	AUD	1,329.67-	USD			200530	2181000210		TIME_CREAT
A	2022	12	452.05-	AUD	452.05-	AUD	664.83-	USD			200532	2181000210		TIME_CREAT
E	2022	12	904.11-	AUD	904.11-	AUD	1,329.67-	USD			200542	2181000210		TIME_CREAT
9	2022	12	904.11-	AUD	904.11-	AUD	1,329.67-	USD			200544	2181000210		TIME_CREAT
C	2022	7	1,550.39-	EUR	1,074.94-	GBP	1,612.41-	USD			200077	1210000010		EVNT_INV
C	2022	7	1,240.32-	EUR	859.96-	GBP	1,289.94-	USD			200077	1210000010		EVNT_INV
6	2022	7	640.00-	GBP	640.00-	GBP	960.00-	USD			200079	1210000010		POC_MAN
4	2022	7	35,000.00-	GBP	35,000.00-	GBP	52,500.00-	USD			200084	1210000010		EVNT_MAN
9	2022	7	640.00-	GBP	640.00-	GBP	960.00-	USD			200090	1210000010		EVNT_MAN

Figure 6.17 Table FARR_D_POSTING Entries

The remaining fields are amount fields (one transactional amount and two for local currency 1 and 2 amounts). There are indicators on which contract postings belong to, which general ledger account will be posted, and what POB type is used for postings. If you check further left, you'll see account assignment objects and who/when created entries in this table.

Now we know all the steps and content in table FARR_D_POSTING, we can represent activities and how entries are made in [Figure 6.18](#).

To illustrate, let's cover one very simple case in RAR end to end. We'll create one sales order, see how it gets processed in RAR, invoice it, and follow the entries in table FARR_D_POSTING.

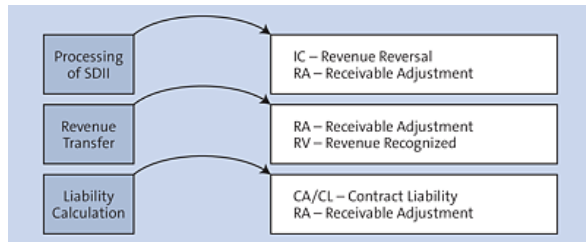


Figure 6.18 Relationship between Table FARR_D_POSTING and RAR Processes

In this example, you'll be working with integration with sales and distribution. Start by creating one sales order that will have a billing plan representing how billing should be done for the contract duration. We won't look at allocation effect, and POBs are being excluded from allocation.

Once you create and save the contract (refer to [Chapter 5, Section 5.1](#)), you see that the new item is visible in Transaction FARR_RAI_MON, as shown in [Figure 6.19](#).

Figure 6.19 Item in Table FARR_RAI_MON

Process this item, and a contract is created. Because this is a time-based POB, you'll get the whole revenue schedule created for this item. We won't go into detail

here about analyzing contract values (see [Chapter 5](#) for that information), but rather we'll focus on preparing postings.

By processing the order item, you see the first entry in table FARR_D_RECON_KEY, as shown in [Figure 6.20](#).

This fits with previous explanations: the first change was made to the contract, and entries are being created. If you check entries for this contract, they start with 2023001, which means, in this system, the first open period is 1 in fiscal year 2023. In addition, reconciliation keys are created for 12 periods, which corresponds to the contract duration of 12 months. Because you didn't do any activity in this contract, the status is **Open**, and all other items are empty.

CoCode	Acct	Contract	Year	Period	Reconcil Key	Prs Date	Substna	Status	Type	Reason	TM	Convert	Labclass	Run D	Date	Created	Time Stamp	Change	Time Stamp
GB04	IFRS	202300100000101	2023	1	202300100000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	2	202300200000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	3	202300300000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	4	202300400000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	5	202300500000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	6	202300600000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	7	202300700000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	8	202300800000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	9	202300900000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	10	202301000000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	11	202301100000101			O								20.230.507.080.544	20.230.507.080.544		
GB04	IFRS	202300100000101	2023	12	202301200000101			O								20.230.507.080.544	20.230.507.080.544		

Figure 6.20 Table FARR_D_RECON_KEY Entries

If you check table FARR_D_POSTING, it will contain no entries, which again corresponds with what was said before: that table isn't updated by just creating contracts and POBs. However, if you check table FARR_D_DEFITEM, you'll see that entries are being made there, as shown in [Figure 6.21](#).

Reconcil Key	POB	Chry	Deter	Cal	Contract	Event	Type	FuflType	CoCode	AcctP	Chry	Unit	Sic	Art	No	Target A/c	Stat	Post Amt	Postg Qty
202300100000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				1.907.21	0.081967		
202300200000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.79	0.0847		
202300300000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				1.907.21	0.081967		
202300400000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.79	0.084699		
202300500000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.79	0.0847		
202300600000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				1.907.21	0.081967		
202300700000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.79	0.084699		
202300800000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				1.907.21	0.081968		
202300900000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.79	0.084699		
202301000000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.78	0.0847		
202301100000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				1.901.64	0.079235		
202301200000101	106006	ZFR0	MA		2023006	T	GB04	IFRS	GBP	PC	5111000899	1132000990				2.032.79	0.084699		

Figure 6.21 Table FARR_D_DEFITEM Entries

The first part of the key is the reconciliation key number, which comes from table FARR_D_RECON_KEY. In addition, because this is a time-based POB, entries are created at the end of the contract, and the posting amount and posting quantity correspond to revenue, which should be calculated every month as revenue (because, in this case, using deferral method 1, the amounts aren't the same). In addition, because there is no allocation effect in this case, there is only one entry per item. However, if this contract had an allocation effect and a parallel reporting requirement, there could have been 36 entries, which would be doubled each time some change occurred in the contract.

The next step is to create an invoice for this sales order. Here, there are two milestones for billing: (1) 30% at the end of period 1, and (2) 70% at the end of period 2. Once you create the invoice in sales and distribution, you'll get the **SDII** item in Transaction FARR_RAI_MON, as shown in [Figure 6.22](#).

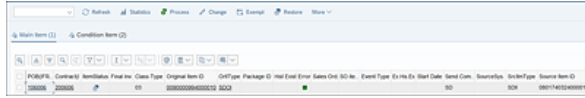


Figure 6.22 Invoice in Table FARR_RAI_MON

While searching for this RAI, the best option is to search the item by sales order number by selecting the order-relevant items option in the **Kind of Selection** section. This will ensure that all subsequent items created on the basis of the sales order will be displayed. Now, once you process this RAI, you can check table FARR_D_POSTING, as shown in [Figure 6.23](#).

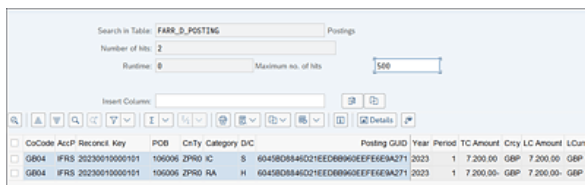


Figure 6.23 Table FARR_D_POSTING after Invoice Processing

You can see the debit posting with posting category **IC** (invoice correction) and credit posting with posting category **RA** (receivable adjustment). If you look at the total postings up to this moment, including the invoice issued by sales and distribution, you get the results shown in [Figure 6.24](#).

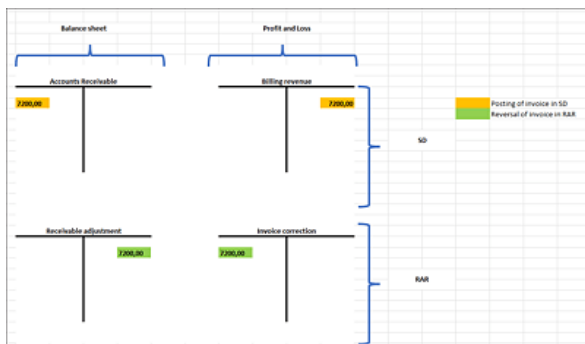


Figure 6.24 Posting Explanation after Invoice Processing

What happened is that the invoice reversal negated postings made in sales and distribution. Regarding the revenue side, depending on settings, postings can be made on the same account as billing or on a different correction account. The receivable adjustment account is a separate account that must be reported in the receivable section of the balance sheet, and it's used in a similar way to negate balance sheet postings coming from sales and distribution invoice processing.

However, it's worth repeating that postings are still not made in financial accounting; they are just stored in table FARR_D_POSTING, which serves as preparation for posting.

The next step is running the Transfer Revenue program (program A; see [Section 6.2.1](#)). The task of this program is to calculate revenue that should be recognized and post it on the receivable adjustment account as an offsetting entry. Once the program is done, you can check the content of table FARR_D_POSTING to see that new entries appeared, as shown in [Figure 6.25](#).

CoCode	Acc	Resoncl	Key	POB	CrTy	Cate	DIC	Posting CnID	Year	Per	TIC Amount	Crty	LC Amount	LCurr
GB	FR	20230010000101	106006	ZPRO	IC	S	6045B08846021EED8B960EEFE8A271	20	1		7,200.00	GBP	7,200.00	GBP
					IC						7,200.00	GBP		
GB04	FR	20230010000101	106006	ZPRO	RA	H	6045B08846021EED8B960EEFE8A271	20	1		7,200.00	GBP	7,200.00	GBP
GB04	FR	20230010000101	106006	ZPRO	S		6045B08846021EED8B9603C56070288D	20	1		1,967.21	GBP	1,967.21	GBP
					RA						5,232.79	GBP		
GB04	FR	20230010000101	106006	ZPRO	RV	H	6045B08846021EED8B9603C56070288D	20	1		1,967.21	GBP	1,967.21	GBP
					RV						1,967.21	GBP		
											0.00	GBP		

Figure 6.25 Result of the Revenue Transfer

Now, if you check the amount, recognized revenue is posted with posting category **RV** and amounts to **1,967.21**. If you take the whole contract value, which is 24,000, and divide it by number of days (365), that equals 65.75 per day. Period 1 being April has 30 days, so the system will multiply daily revenue by the number of days to come to the monthly revenue of 1,967.21. In this case, posting was done with condition type **ZPRO**, which represents the main condition type. Because items are excluded from allocation, there are no entries with the allocation correction condition type and no postings need to be done on the adjustment account. [Figure 6.26](#) shows these postings in T-accounts.

After these postings, you get the balance on posting category RA, which represents the difference between the billed amount (7,200) and recognized revenue (1,967.21). Because more was billed than could be recognized, this amount should be represented as a liability on the balance sheet. On the P&L side of the financial statement, the total balance is 1,967.21, which is the exact revenue that can be recognized according to IFRS 15.

The next step is done when you process the program for calculating CA/CL. Once this program is run for period 1, the balance is transferred from the receivable adjustment account to the respective account on the balance sheet side.

Once the program for liability is calculated, you can check how the entries look in table FARR_D_POSTING, as shown in [Figure 6.27](#).

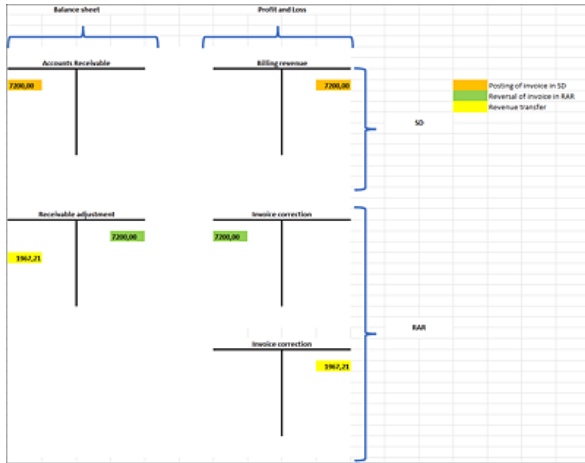


Figure 6.26 Postings after Revenue Transfer

CoCode	Acc.	Reconcil Key	PCB	ChTy	Cate	QIC	Posting	GLD	Year	P.	TC Amount	Crcy	L
GB	IFR	20230010000101	106006	CL	H	6045808846021EEDB972A9G6DAE8G2	20		1		5,232.79	GBP	
				CL							8,332.79	GBP	
GB04	IFR	20230010000101	106006	ZPR0	IC	S	6045808846021EEDB960E0E0E8A271	20	1		7,200.00	GBP	
				IC							7,200.00	GBP	
GB04	IFR	20230010000101	106006	RA	S	6045808846021EEDB972A9G6DAE8G2	20		1		5,232.79	GBP	
GB04	IFR	20230010000101	106006	ZPR0	H	6045808846021EEDB960E0E0E8A271	20		1		7,200.00	GBP	
GB04	IFR	20230010000101	106006	ZPR0	S	6045808846021EEDB960E0E0E8A271	20		1		1,967.21	GBP	
				RA							0.00	GBP	
GB04	IFR	20230010000101	106006	ZPR0	RV	H	6045808846021EEDB960E0E0E8A271	20	1		1,967.21	GBP	
				RV							1,967.21	GBP	
											0.00	GBP	

Figure 6.27 Table FARR_D_POSTING after Running of Liability Calculation Program

You can see that the balance to the **RA** posting category is moved to zero, and the remaining is transferred to the contract liability. To get a complete picture, look at the T-accounts, as shown in [Figure 6.28](#).

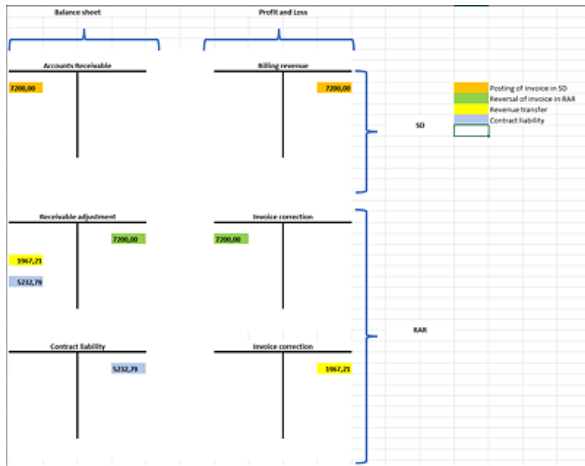


Figure 6.28 Postings after Contract Liability

Now let's provide an example of how postings in table FARR_D_POSTING are made. Again, to get postings made in financial accounting (either tables ACDOCA or BKP/BSEG), you still need to run program C ([Section 6.2.3](#)). But to get the complete picture of this process, take a look at table FARR_D_RECON_KEY, as shown in [Figure 6.29](#).

The **Status** of reconciliation key moved to **P (Transferred)**, which means the revenue for the period is being properly calculated. But before a period can be closed, liability needs to be calculated, as you can see in the **LiabAsset** field: if it's marked with an **X**, liability is calculated and that period can be closed. However, remember that in order for the period to be closed, all reconciliation keys for all contracts must be in this status. If this isn't the case, the system won't allow closure of the period, and you'll be asked to either resolve the problem or move contracts to the next period.

As we mentioned, the last step is to post actual revenue ([Section 6.2.3](#)). This is a technical step that represents postings from table FARR_D_POSTING as financial accounting documents. After the job is run, the table entry is updated with the **Run ID** of the exact posting job.

The screenshot shows the SAP table FARR_D_RECON_KEY. The table has columns for Contract, Year, Period, Reconcil. Key, P/B, Date, Substanz, Status, Type, Reason, TM, Cover, Labook, Run ID, Date, Created, Time Stamp, and Change. The status 'P' (Transferred) is highlighted for several entries. A dropdown menu is open for the 'Status' column, showing options: Open, Closed, In Process, Cancelled, Failed, Time Base, Liability/Asset, Migration, Simulation by RWI, Replaced by Shift Period Handling, and Transferred. An arrow points from the 'P' status in the table to the 'Transferred' option in the dropdown.

Contract	Year	Period	Reconcil. Key	P/B	Date	Substanz	Status	Type	Reason	TM	Cover	Labook	Run ID	Date	Created	Time Stamp	Change	
0804	IFRS	200606	2023	2	20230600000101		282	O										
0804	IFRS	200606	2023	3	20230600000101		282	O										
0804	IFRS	200606	2023	4	20230640000101		282	P										
0804	IFRS	200606	2023	5	20230600000101		282	O										
0804	IFRS	200606	2023	6	20230600000101		282	O										
0804	IFRS	200606	2023	7	20230670000101		282	O										
0804	IFRS	200606	2023	8	20230680000101		282	O										
0804	IFRS	200606	2023	9	20230690000101		282	O										
0804	IFRS	200606	2023	10	20230100000101		282	O										
0804	IFRS	200606	2023	11	20230100000101		282	O										
0804	IFRS	200606	2023	12	20230120000101		282	O										

Figure 6.29 Table FARR_D_RECON_KEY after Liability Calculation

6.2 Posting with ABC

To create postings from the RAR module, three programs are designed for that purpose:

- **Program A: Transfer Revenue (Transaction FARR_REV_TRANSFER (Transaction FARR_REV_TRANSFER)**
This program is used to calculate recognized revenue by running the Transfer Revenue app or by scheduling a job with Transaction FARR_REV_TRANSFER.
- **Program B: Calculate Contract Liabilities and Contract Assets (Transaction FARR_CONTRACT LIABILITY)**
This program calculates contract assets or contract liabilities by running the Calculate Contract Assets and Contract Liabilities app or using Transaction FARR_CONTRACT LIABILITY.
- **Program C: Revenue Posting Run (Transaction FARR_REVENUE_POSTINGS (Transaction FARR_REVENUE_POSTINGS)**
This program makes appropriate postings in financial accounting by running the Revenue Posting Run app or using Transaction FARR_REVENUE_POSTINGS.

These programs need to be run for all contracts at least once a month before the period can be successfully closed. Because they are always run in this sequence, they are commonly called the *ABC programs*.

Note that data won't be visible in financial accounting nor in reporting until the last program from the list (Transaction FARR_REVENUE_POSTINGS) is run. So, the idea to use RAR as the main tool for revenue reporting depends on properly planning how the ABC programs are executed. RAR isn't a real-time engine (meaning that as soon as some change happens in the sales order, you can see the effect in reporting) and always needs to be scheduled before figures will be visible in reports. So, the question is, how often is good enough?

There is no strict rule about what isn't often enough and what is too often, rather just a statement that to close the period, all contracts need to be processed successfully by ABC programs. So, does that mean once is good enough? There are a couple of reasons why this might not be the best idea unless you're facing a small number of contracts or specific business scenarios.

If you process ABC programs once a month, the first consequence is that reporting (in both finance and profitability analysis) will be available only once a month. In some cases, this isn't a big limitation, but in some others, expectations will be to see results more often than that. The second consideration is that if you process programs once a month, which is normally during the month-end closing process, you're already running in a tight time window.

As shown in [Figure 6.30](#), RAR comes after you've performed all activities in sales and distribution, meaning that all orders are created, deliveries are made, and invoicing to the customer is performed. For adjustments in terms of

credit/debit memos, these are done already, which makes all things ready for closing from the financials side.

You need to run processing of created RAIs as preparation for running ABC programs; after that, jobs can be scheduled, and final results are sent to respective modules. Once this is done, profitability analysis can proceed with its own activities of allocating or settling that revenue to the final receivers. For information on integration with profitability analysis, see [Section 6.3](#).

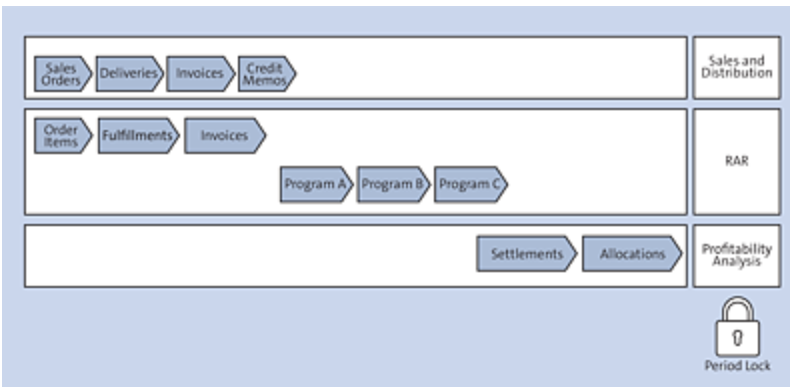


Figure 6.30 Closing Process with RAR

This is how month-end closing looks in more or less every company in the world, so what's the issue? All these activities are always being done in a very limited time. Businesses usually run processes in sales until the very last minute in the month, meaning in RAR, you need to wait for these to be completed. Once it's done, there is a very short time available for the finance team to complete activities, and each of these tasks has only a fraction of time dedicated to completing it. It's not unusual that both postings and processing of RAIs end in some kind of an error, which, depending on its severity, needs to be resolved or assessed before the month can be closed.

We can conclude from this that ABC programs need to be run more than once a month. Some clients have tried to match that by making RAR as close as possible to a real-time reporting engine by scheduling ABC programs to run as often as possible, even every 15 minutes. That proved not to be the best approach because, depending on the number of contracts and types of allocations, the duration of each program was taking more time than expected, and programs were often going into deadlock.

So, in terms of a recommendation, we can say that ABC programs should be run more than once a month, but detailed performance and resource evaluation is needed if the business requirement is to run the programs more times during business hours.

Business Example: Telecommunications

The telco line of business can be considered an exception. Data is processed by source systems during a month (for CRM or charging systems), and receivables are calculated at month end only (depending on the company, there might be more than one billing cycle during the month). So, data for RAR calculations is available only at exact month end. In this case, the client needs to ensure first of all resource availability so time isn't affected by the performance of systems and that all systems used are optimized to reach key performance indicators (KPIs), which are put in front of the technical team as business requirements.

Now, let's dive into all three ABC programs in the following sections.

6.2.1 Transfer Revenue

The first step in running ABC programs is to transfer revenue. Revenue transfer is used to calculate time-based revenue and prepare data for subsequent posting runs. Fulfillments that are calculated will be staged in table FARR_D_POSTING so they can be used in subsequent programs and included in the posting run.

Open the Transfer Revenue app or Transaction FARR_REV_TRANSFER. On the screen shown in [Figure 6.31](#), first of all select which **Company Code** and **Accounting Principle** will be used. If you're running multiple accounting principles, you need to indicate all of them separately for running the program. Next, select **Fiscal Year** and **Posting period** for which calculation is to be done. If the fiscal year isn't equal to the calendar year, you need to enter the fiscal year and period for which the calculation will be run. You can also enter a **Revenue Accounting Contract** for which calculation will be executed. Although this is useful if you want to look at some specific changes in the contract, this should be left blank usually so that all contracts can be considered.

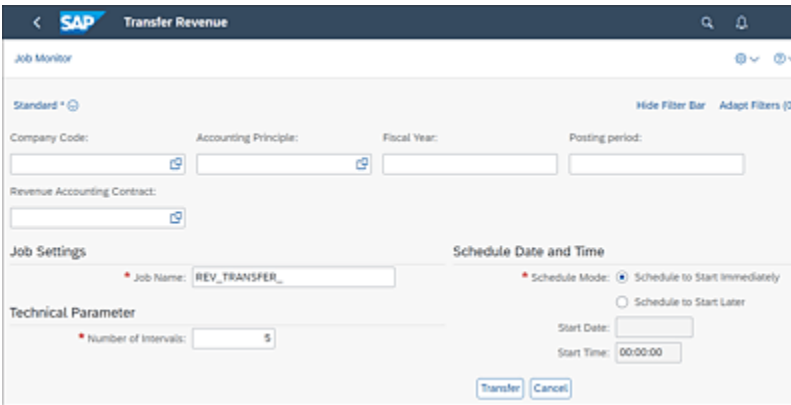


Figure 6.31 Revenue Transfer Options

Now, because running this program is scheduled in the background for performance reasons, you need to enter a name for the job in **Job Name**, when you want the job to start under **Schedule Data and Time**, and how many intervals to use in **Number of Intervals**. Once you're ready, click the **Transfer** button so the job can start running.

Clicking the **Job Monitor** button schedules the job, and you can see its progress as it's being executed.

The status will show whether the job is just scheduled, is currently running, or is completed. If the status is too long in the scheduled status, but you asked for immediate start, you need to check in Transaction SM37 to see the reason for the delay. By refreshing the table, you'll be able to verify if the process is finished or some other change in the job run has happened (see [Figure 6.32](#)).

Status	Job Name	Job Type	Created By	Planned Date/Time	Start Date/Time	End Date/Time
●	REV_TRANSFER_20230507	Transfer Revenue	Sreten Mitosavljevic	07.05.2023 08:47:02	07.05.2023 08:52:02	07.05.2023 08:52:10

Figure 6.32 Revenue Transfer Completed

Once the job process is over, you'll be able to see the status as green, as well as who scheduled this job and when, and when it was completed. In case of errors, you'll see a red indicator under the **Status** column with a message and information about the cause of the errors.

As soon as the revenue transfer job is completed, you can move to the next process, which is running the liability calculation.

6.2.2 Calculate Contract Liabilities and Contract Assets

This program is run as a predecessor of the job for making postings from RAR. This program will calculate the difference between the billed amount and the recognized revenue and then post it as either a contract asset or liability. Again, entries will just be stored in table FARR_D_POSTING and not posted yet; after running it, you'll be able to find entries with posting categories CA, CL, UR, and DR in the table for analysis.

Open the Calculate Contract Liabilities and Contract Assets app or run Transaction FARR_CONTRACT_LIABILITY to open the screen shown in [Figure 6.33](#).

Here, you choose **Company Code**, **Accounting Principle**, and **Posting period** just like in the previous program. One addition to this program is the **Value Date** field. As we already explained, the invoice due date and posting date have an impact on the period determination and calculation of CA/CL and UR/DR. The invoice posting date has an impact on which revenue accounting period will be determined, for

example, December 29, 2022, would usually determine 2022 - 012. The posting date is also used to calculate UR/DR, while due date is used to calculate CA/CL. If the posting date is less than or equal to the value date, those entries will be fetched as invoice amounts to calculate UR/DR during execution of report Calculate Contract Liability/Contract Asset.

The screenshot shows the SAP Job Monitor interface for the report 'Calculate Contract Liabilities and Contract Assets'. The interface includes several input fields and options:

- Company Code:** [Empty field]
- Accounting Principle:** [Empty field]
- Fiscal Year:** [Empty field]
- Posting period:** [Empty field]
- Revenue Accounting Contract:** [Empty field]
- Job Settings:**
 - Job Name:** [Empty field]
 - Value Date:** [Empty field]
- Schedule Date and Time:**
 - Schedule Mode:** Schedule to Start Immediately, Schedule to Start Later
 - Start Date:** [Empty field]
 - Start Time:** 00:00:00
- Technical Parameter:**
 - Number of Intervals:** 5

Buttons for 'Transfer' and 'Cancel' are located at the bottom right of the form.

Figure 6.33 Liability Calculation Job

Value Date Problems

Customers often face issues while working with the value date, especially during the testing phase. The most common problems are that the value date isn't modifiable by the user in the currently open period, any kind of manual entry in this field is ignored, and the system presets it with the current date. These issues are a problem during testing if the customer wants to test calculations in subsequent months; in those cases, the process for liability calculation should be executed multiple times.

For the first problem, the solution is described in SAP Note 3263233 (there are manual activities that need to be performed); for the second problem, the solution is given in SAP Note 3271649 (which needs to be implemented).

Once the program is scheduled, you can run it in a very similar way as revenue transfer (see [Figure 6.34](#)).

Status	Job Name	Job Type	Created By	Planned Date/Time	Start Date/Time
	LIABILITY_20230507	Calculate Contract Liabilities and Assets	Sreten Miskovjevic	07.05.2023 09:23:57	00.00.0000 00:00:00
●	REV_TRANSFER_20230507	Transfer Revenue	Sreten Miskovjevic	07.05.2023 09:03:20	07.05.2023 09:04:01
●	REV_TRANSFER_20230507	Transfer Revenue	Sreten Miskovjevic	07.05.2023 08:47:02	07.05.2023 08:52:02

Figure 6.34 Liability Calculation Being Scheduled

Again, the system will provide information about who/when started job and its current status (see [Figure 6.35](#)).

Field Name	Opera...	From
Company Code	Equals	GB04
Accounting Principle	Equals	IFRS
Fiscal Year	Equals	2023
Posting period	Equals	001
Contract	Equals	200606
Value Date	Equals	30.04.2023
Synchronous Call	Equals	X

Figure 6.35 Job Details Display

You can see what selections were made and based on which program is running, which is very useful if you need to validate that the variant defined is proper and that the

results are as expected. It should also be the starting point in analyzing errors.

One question often asked is can we somehow verify that the job was executed properly. The standard way is to go in Transaction SLG1 and evaluate contracts that were taken by the program. However, you can get some basic information already from the program by selecting option **Application Log** (see [Figure 6.36](#)).

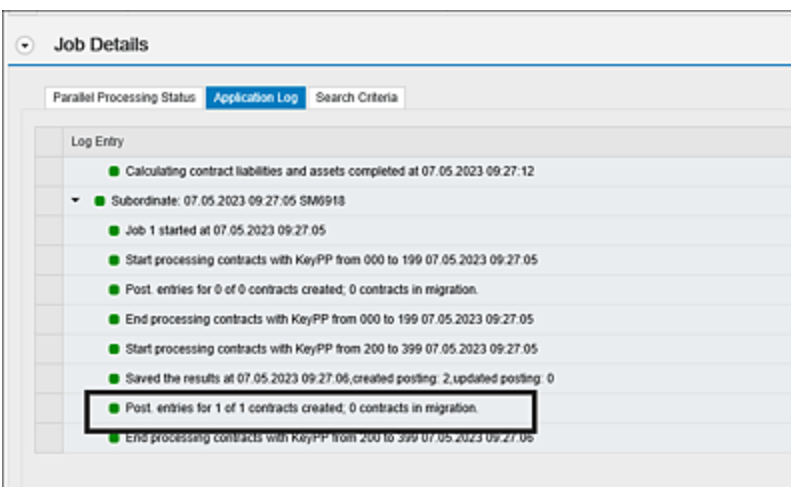


Figure 6.36 Application Log in Liability Calculation

In this case, you see that one contract was selected as eligible for calculation, and updates were made in table FARR_D_POSTING for another contract. Ideally, the number of selected contracts and updated contracts should be the same, but this can be slightly different in some cases: for example, if there was no invoicing, the contract liability program wouldn't calculate anything to be updated to table FARR_D_POSTING. The same will happen if a CA/CL method is assigned to the accounting principle and an invoice was created, but isn't due. In those cases, you'll see information

that the contract was considered, but updates weren't made.

6.2.3 Revenue Posting Run

The last step, which is actually creating postings of the accounting document, is Transaction FARR_REV_POSTING. By running this program, the system is picking up all values from table FARR_D_POSTING, which has been calculated by programs A and B, and posting them in the general ledger.

Open the Revenue Posting Run app or run Transaction FARR_REVENUE_POSTINGS to open the screen shown in [Figure 6.37](#).

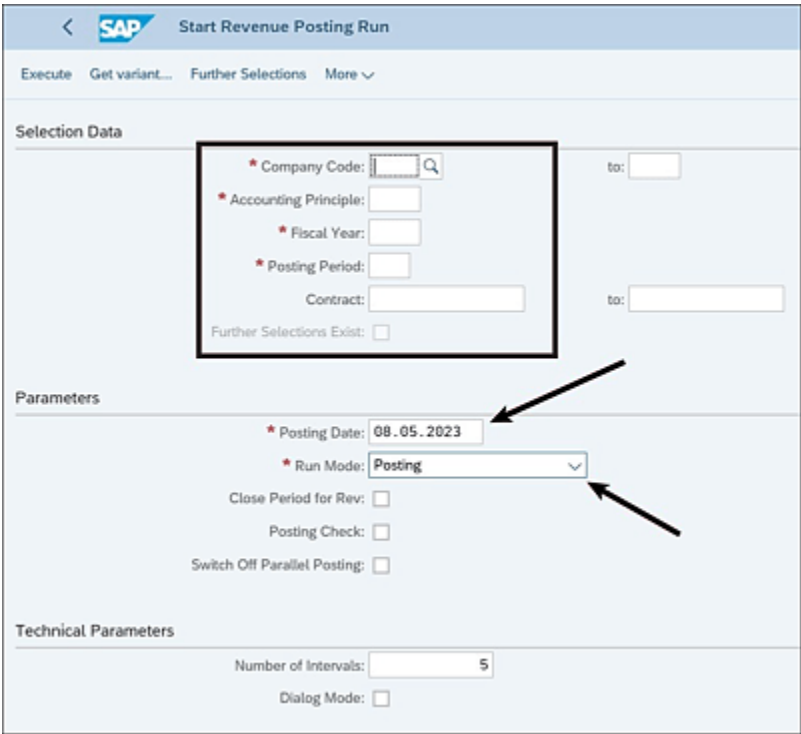


Figure 6.37 Program C Initial Screen

In this case, you see that the initial parameters for the program are very similar to the previous programs: you

need to specify **Company Code**, **Accounting Principle**, and **Fiscal Year/Posting Period** combination. Again, you can execute postings specific for one contract (used mainly for debugging or error resolution) or for all contracts, which is appropriate for the month-end closing process.

Unique to this program is that you can run postings first in simulation mode before you run the actual update run. To do so, switch the **Run Mode** dropdown to **Test Only**, as shown in [Figure 6.38](#). This is specific only to this program in contrast to programs B and C, which can be run only in update mode. This is because program C is taking over already calculated values, while programs A and B are performing actual calculations.

When you're ready to perform the posting, click the **Execute** button.

Note that error handling is being performed already at this stage. Once the program is executed, whether in test or productive mode, checks will be performed, and the process will finish only if errors aren't being detected. Similar to the previous cases, you can execute the program in dialog mode, in which messages will be displayed, or in the background, in which potential messages will be evaluated by analyzing the log via Transaction SLG1, as shown in [Figure 6.39](#).

The screenshot shows the SAP posting configuration interface. It is divided into three main sections:

- Selection Data:**
 - Company Code: 3003
 - Accounting Principle: IFRS
 - Fiscal Year: 2023
 - Posting Period: 1
 - Contract: (empty)
 - Further Selections Exist:
- Parameters:**
 - Posting Date: 08.05.2023
 - Run Mode: Test Only
 - Close Period for Rev:
 - Posting Check:
 - Switch Off Parallel Posting:
- Technical Parameters:**
 - Number of Intervals: 5
 - Dialing Mode:

Figure 6.38 Running Posting in Test Mode

The screenshot shows the SAP message log with the following entries:

Type	Message Text
Message Text	Revenue posting started at 08.05.2023 13:39:59
Period 2023001 is closed for rev. acct. for co. code 3003 acct. principle IFRS	
Revenue posting completed at 08.05.2023 13:39:59	

Figure 6.39 Error Message Due to a Closed Period

Now, when you finish the posting process, a financial document is created. As explained in [Section 6.1.2](#), the number of entries in a document highly depends on the optimization options selected. Either way, postings in RAR will be grouped and not done on the contract level. Let's look at the posting document in detail. Start Transaction FB03 or the Display Financial Document app, and enter **Document Number**, **Company Code**, and **Fiscal Year**, and the document will appear, as shown in [Figure 6.40](#).

The first question is, how do we identify if the document is from RAR? The main identifier is the document type, which is reserved for RAR postings, but this isn't a mandatory requirement—only a recommendation. For RAR postings, the reference transaction (**Ref. Transactn**) will be set as **FARA**,

and **TCode** will be marked as **FARR_REV_POST**. In addition, the accounting principle (**ActgPrinciple**) field will be populated with the principle for which we ran program (in this case, **IFRS**), which leads us to conclude that for parallel reporting requirements, two documents will be created.

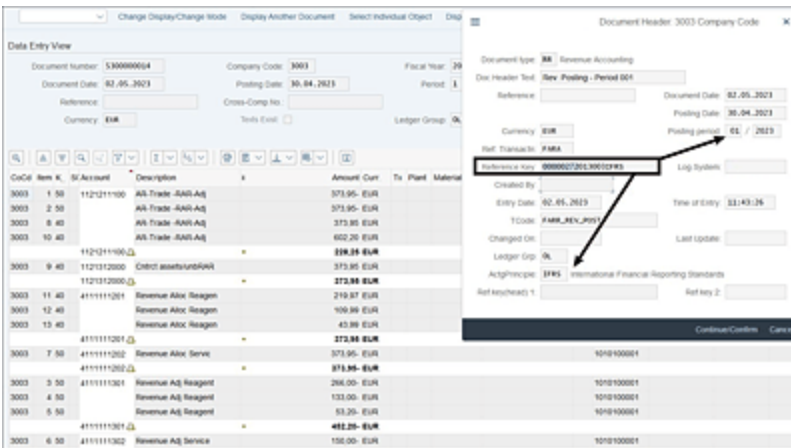


Figure 6.40 Posting Document Coming from RAR

But the most important field in this document is **Reference Key**. In this case, it's populated with some values that look like random numbers (**00000272013003IFRS**), but there is a rule for how it's populated. The meaning of this value is as follows:

- Characters 1-8 represent the posting run.
- Characters 9-10 represent the period for which the run was performed.
- Characters 11-14 are reserved for the company code.
- Characters 15-18 are dedicated to the accounting principle.

The posting run is a unique identifier of a specific posting job run, and it's given to all contracts that belong to a particular reconciliation key. Once the posting job is

completed, the same values can be found in table FARR_D_RECON_KEY, as shown in [Figure 6.41](#).

The next question is about reconciliation between RAR and financial accounting. To perform those tasks, you can use the Reconciliation – Subledger and G/L SAP Fiori app. By running this app, you can enter a specific posting run, and the system will display the posting structure from the accounting document, as shown in [Figure 6.42](#).

COGS	Acct/Contract	Year Per	Reconc Key	Pro Split Date	Subsena	Status	Type	Reason	TM	Convert	LibAss	Ra	Date	Created by	Time Stamp
3000	IFRS 30013327	20	1 20230010000191		341	C					X		275 04 05 2023	DEUTEGETHOFN	20.230.504.112.358
3000	IFRS 30013323	20	1 20230010000191		449	C					X		273 03 05 2023	DEUTEGETHOFN	20.230.502.151.845
3000	IFRS 30013321	20	1 20230010000191		124	C					X		273 02 05 2023	EYGO890	20.230.502.260.715
3000	IFRS 30013319	20	1 20230010000192		187	C					X		270 28 04 2023	EYGO890	20.230.428.142.415
3000	IFRS 30013319	20	1 20230010000193	28.04.2023	187	C					X		28 04 2023	EYGO890	20.230.428.142.415
3000	IFRS 30013319	20	1 20230010000191		187	C					X		269 28 04 2023	EYGO890	20.230.428.142.046
3000	IFRS 30013315	20	1 20230010000192		848	C					X		268 28 04 2023	EYGO890	20.230.424.132.144

Figure 6.41 Posting Key in FARR_D_RECON_KEY

GL Document	Revenue Assoc.	Performance Obl.	Performance Obl.	Posting Categ.	Condition Type	GL Account	Account Descr.	Debit/Credit	Amount	Currency	Amount in LC	Local C.
580000014	30013322	20205	REAGENT (RR)	Reconcilable Adjus...	ZNL1	1221211200	AR Trade (RR-Adj)	Credit	370,99	EUR	370,99	EUR
580000014	30013322	20205	REAGENT (RR)	Reconcilable Adjus...	ZP02			Debit	130,00	EUR	130,00	EUR
580000014	30013322	20206	REAGENT (RR)	Reconcilable Adjus...	ZNL1			Credit	43,00	EUR	43,00	EUR
580000014	30013322	20207	SERVICE (BILLING)	Reconcilable Adjus...	ZP02			Debit	53,00	EUR	53,00	EUR
580000014	30013322	20207	SERVICE (BILLING)	Reconcilable Adjus...	ZNL1			Credit	373,99	EUR	373,99	EUR
580000014	30013322	20207	SERVICE (BILLING)	Reconcilable Adjus...	ZNL1			Debit	373,99	EUR	373,99	EUR
580000014	30013322	20207	SERVICE (BILLING)	Reconcilable Adjus...	ZP02			Debit	150,00	EUR	150,00	EUR
580000014						1121201200	AR Trade Adst...		298,29	EUR	298,29	EUR
580000014						1121211200	AR Trade Adst...		298,29	EUR	298,29	EUR
580000014	30013323	20207	SERVICE (BILLING)	Contract Asset		1221212000	Credit asset/other	Debit	370,99	EUR	370,99	EUR
580000014						1121202000	Credit asset/other		370,99	EUR	370,99	EUR
580000014						1121212000	Credit asset/other		370,99	EUR	370,99	EUR
580000014	30013322	20206	REAGENT (RR)	Recognized Rev...	ZNL1	4011111205	Revenue Alloc Res...	Debit	219,97	EUR	219,97	EUR
580000014	30013322	20206	REAGENT (RR)	Recognized Rev...	ZNL1			Debit	109,99	EUR	109,99	EUR
580000014	30013322	20206	REAGENT (RR)	Recognized Rev...	ZNL1			Debit	62,99	EUR	62,99	EUR
580000014						4111112011	Revenue Alloc R...		370,99	EUR	370,99	EUR
580000014						4011111205	Revenue Alloc R...		370,99	EUR	370,99	EUR
580000014	30013322	20207	SERVICE (BILLING)	Recognized Rev...	ZNL1	4011111205	Revenue Alloc S...	Credit	373,99	EUR	373,99	EUR
580000014						4111112012	Revenue Alloc S...		373,99	EUR	373,99	EUR
580000014						4011111202	Revenue Alloc S...		373,99	EUR	373,99	EUR

Figure 6.42 Reconciliation App Results

Note

You can find a PDF version of [Figure 6.42](#) available for download at sap-press.com/5700 under the **Product supplements** section.

There are several more applications available that can provide information about postings in financial accounting and calculations made in table FARR_D_POSTING. See [Section 6.4.1](#) for more information.

6.3 Integrating with Profitability Analysis

Profitability analysis enables the evaluation of market segments (products, customers, orders, or any combination of these, or strategic business units, such as sales organizations or business areas) concerning the company's profit or contribution margin. The system aims to provide all relevant segments and departments with information that supports internal accounting and decision-making.

There are two forms of profitability analysis supported, and both types can be used simultaneously:

- **Costing-based profitability analysis**

This groups cost and revenues according to value fields and costing-based valuation approaches, whereas both can be defined, always assuring access to a complete, short-term profitability report.

- **Margin analysis (formerly called account-based profitability analysis)**

This is organized in accounts and uses an account-based valuation approach. The difference in this form is its use of cost and revenue elements, providing a profitability report permanently reconciled with financial accounting.

RAR integrates with profitability analysis using the sales and distribution entry to profitability analysis, as it behaves like sales and distribution billing documents. Therefore, the actual value flow is defined by assigning condition types to

value fields, similar to sales and distribution billing documents.

In the IMG, go to menu path **Controlling • Profitability Analysis • Flow of Actual Values • Transfer of Billing Documents • Assign Value Fields • Maintain Assignment of SD Conditions to CO-PA Value Field.**

Profitability analysis integration with RAR is handled mainly as controlling setup. If profitability analysis is active, the interface component profitability analysis characteristics are by default activated in the configuration of RAI classes for order items (RAI class type 01; see top screen in [Figure 6.43](#)). The activation of this interface component can't be deselected. The interface component contains all the fields of structure COPACRIT, as shown in the bottom screen in [Figure 6.43](#). Fields from structure COPACRIT are transferred to the profitability segment number by transforming raw items into processable items.

[Figure 6.44](#) shows different treatment of postings in costing-based profitability analysis and margin analysis. The first difference is that at the moment of goods issue, COGS are posted in margin analysis, whereas in costing-based profitability analysis, you need to wait for the invoice when COGS are getting posted statistically. If you add RAR cost corrections into the equation, reversal of COGS will be posted in margin analysis, but not in costing-based profitability analysis. At the end, the result of both approaches is the same, but with different postings in different steps.



Figure 6.43 Profitability Analysis Basic Setup

The following steps describe a typical business process for cost recognition:

1. The order includes the COGS in a condition. This is the main condition where subsequent accounts are derived from, and there can only be one main cost condition. Cost conditions other than the main cost condition are ignored with the exception of capitalized cost. The order creates a revenue accounting contract. There are no financial accounting postings in this step. For external components, the account assignment is done in BRFplus.

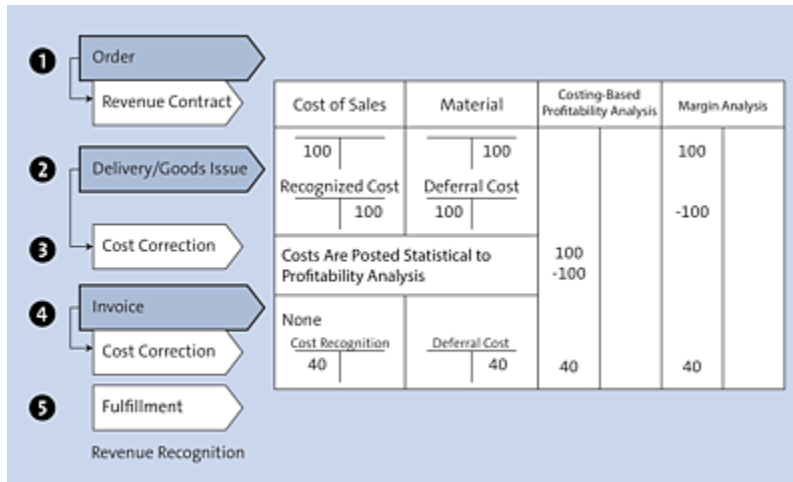


Figure 6.44 RAR: Profitability Analysis Interface

2. The goods issue or delivery triggers a financial accounting posting with cost of sales (CoS) in debit and material in credit. Margin analysis is updated if active.
3. The goods issue forwards the COGS to RAR. A correction posting is created with the posting run, which reverses the financial accounting and profitability analysis posting of step 2.
4. The invoice updates the costing-based profitability analysis with COGS. As the invoice is forwarded to RAR, a reverse posting is created with the posting run.
5. The fulfillment posts revenues and costs according to the cost matching principle. The costs are posted to financial accounting, costing-based profitability analysis, and margin analysis. In the example, 40% of the revenues have been fulfilled despite the delivery. Therefore, 40% of the CoS are posted to financial accounting and profitability analysis.

Cost is posted to profitability analysis when its corresponding revenue is posted. You need to set up the

condition type for goods issue (e.g., VPRS). Ensure that the cost element of the general ledger account derived from the condition type has cost element category 12 (sales deductions).

In costing-based profitability analysis, if you're using VPRS as the cost condition type, **Transfer +/-** must *not* be set. The cost condition needs to be defined this way because otherwise, the profitability analysis component would raise an error message if it gets positive and negative values for the same profitability segment during the posting run.

As RAR delivers positive and negative values to profitability analysis, the **Transfer +/-** indicator must be set for assigning the condition type to the value field, including the following settings:

- For the CoS (condition category G, e.g., VPRS), the **Transfer +/-** flag must not be set.
- In the right of returns (RORs), assign value fields for the ROR cost, and revenue adjustments are needed.

6.4 Reporting

Reporting is critical for analytics and monitoring the business process and status. As you already know, there is a classic and optimized approach to RAR; we'll explore the reporting supported in both in this section.

In RAR, reporting is required for disclosures, there are also reports that help in executing the operations and providing an understanding of business activities. Reporting is also very important for reconciliation because you have to reconcile data between the sender system and with finance. Reconciliation reports help gather the details of the financial accounting documents created and understand the difference between RAR and the general ledger.

6.4.1 SAP Fiori Applications

The reports in RAR have basically been divided into three sections, as shown in [Table 6.3](#). You can access these reports through the SAP Fiori launchpad with business role `SAP_BR_REV_ACCOUNTANT` assigned to your ID. You can use the App Finder to search for required revenue recognition apps and add them to your homepage. As an example, we've created a folder called **RAR** and added all the reporting apps under this folder, as shown in [Figure 6.45](#).

Disclosures

**Operational
Reports**

Reconciliation

Disclosures	Operational Reports	Reconciliation
<ul style="list-style-type: none"> Disaggregation of Revenue: By Multiple Dimensions Disaggregation of Revenue: By Customer Disaggregation of Revenue: By Customer Group Disaggregation of Revenue: By Performance Obligation Type Contract Balances 	<ul style="list-style-type: none"> Posted Amount: By Performance Obligation Type Posted Amount: By Contract FI Documents: By Contract Pending Review Worklist 	<ul style="list-style-type: none"> Reconciliation between Revenue Accounting and General Ledger Initial Load Report Transition Comparative Report Revenue Accounting Data Validation

Table 6.3 Reporting in RAR

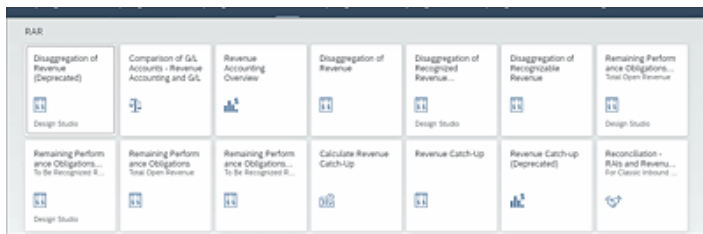


Figure 6.45 RAR Reports in SAP Fiori

To create a folder for RAR, follow these steps:

1. Launch the SAP Fiori launchpad.
2. Provide your login credentials, and you'll get the screen shown in [Figure 6.46](#).

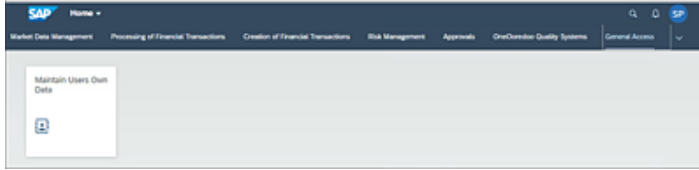


Figure 6.46 Initial Screen

3. Click on the user profile at the top-right corner (in this case, a blue circle with **SP** in it).
4. In the menu that drops down, click on **App Finder**, as shown in [Figure 6.47](#).

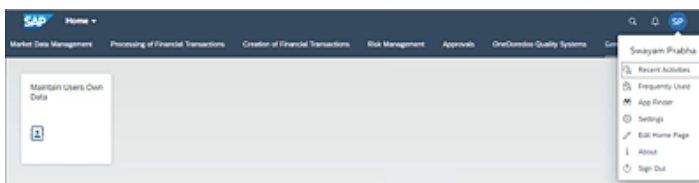


Figure 6.47 App Finder

5. The **App Finder** screen shown in [Figure 6.48](#) provides a list of apps on the left side, app tiles in the center, and a search box in the top right. In the search box, enter “REVENUE” (or anything that you want to search), and you get the list of apps related to the search word.

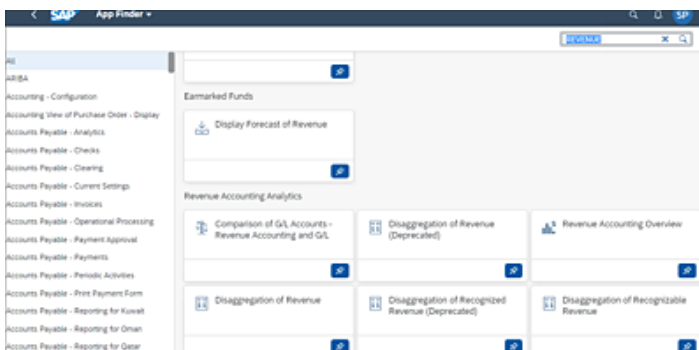


Figure 6.48 Apps Screen with the Search for “REVENUE”

6. Select an app, and click on the icon at the bottom right of the app.
7. In the text back that appears, create a new group, and add the app to it, as shown in [Figure 6.49](#).

In our case, we created a group named “RAR” and added all the required apps.

Classic Reporting

Previously, all the reports were available through SAP Business Client. Now with SAP S/4HANA, all the reporting is shifted to SAP Fiori apps.

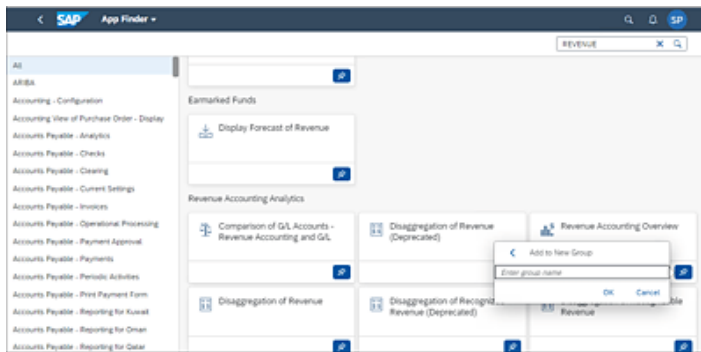


Figure 6.49 Add App to the New Group

SAP recommends using core data services (CDS) views to create analytical reports. Earlier, data sources were used, which aren't supported by SAP anymore. If you use data sources, you'll have to do manual activations. We'll explain the key reports for disclosures, operations, and reconciliation in the following sections.

Disclosures

Analytics and reporting revenue recognition are important for monitoring, reviewing, and forecasting revenues and costs. Analytic reports help in making strategic decisions and planning controls. Basically, disclosure reports are created because part of IFRS 15 has various disclosure paragraph requirements. Disclosures include a few different reporting activities, as shown in [Figure 6.50](#).

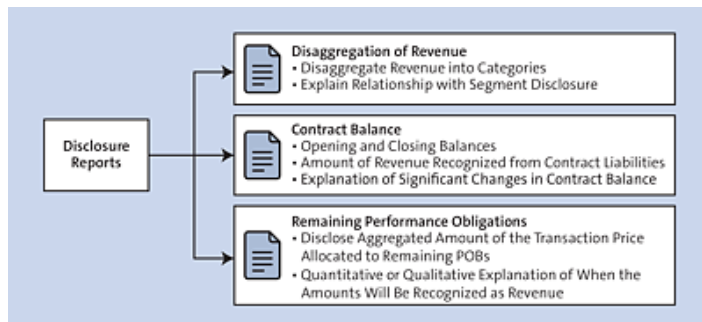


Figure 6.50 Disclosure Reports

Let's walk through each:

- **Disaggregation of revenue**

We'll explain a basic case with paragraph 114 in IFRS 15, which states the following:

"An entity shall disaggregate revenue recognized from contracts with customers into categories that depict how the nature, amount, timing and uncertainty of revenue and cashflows are affected by economic factors. An entity shall apply the guidance in paragraph B87-B89 when selecting the categories to use to disaggregate revenue."

This is just one paragraph of many, and we need reporting to help disclose this data. So RAR comes to the rescue by giving us multiple disclosure reports. Following are some sample reports:

- Disaggregation of Revenue (by multiple dimensions): This report helps you display revenue data for a particular accounting period aggregated by multiple dimensions such as customer, customer group, POB type, and so on.
- Disaggregation of Posted Revenue: This report helps you analyze posted revenue.
- Disaggregation of Recognized Revenue: This report helps you analyze the recognized revenue.

The Disaggregation of Revenue report is accessed through the SAP Fiori launchpad, as shown in [Figure 6.51](#).

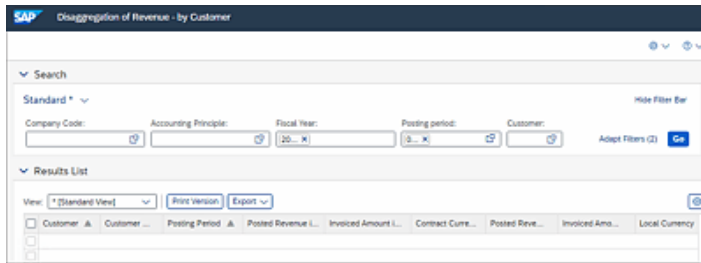


Figure 6.51 Disaggregation of Revenue - By Customer Report

- **Contract balance**

The Contract Balance report is designed to show the key figures (opening balance, posted amount, and closing balance) according to the company code, accounting principle, revenue contracts, and multiple other dimensions per period. Basically, the focus lies mainly on CA/CL or UR/DR to show the closing balance. In the accounting principle configuration, you choose either UR/DR or CA/CL, so this is where it comes from.

- **Remaining POBs**

POBs can be of two types: event based and time based. For event-based POBs, it's clear that the trigger of the event is when the revenue will be recognized. For time-based POBs, at a certain point in the reporting cycle, some POBs are unsatisfied. According to IFRS 15 paragraph 120, there are requirements to disclose the aggregated amount of the transaction price allocated to the unsatisfied POBs. It's also expected to disclose when the company or entity is expected to recognize the remaining revenue. For this, you may need to mention different timelines. The Disaggregation of Revenue report can also be used to meet this need.

Operational Reports

The operational reports are used for analytics to report the posted amount on different dimensions and in different currencies. They capture the transactional data and produce it in the format that is needed and designed.

Operational reports will be used by people in the organization who are responsible for day-to-day tasks related to tracking contract

creation and revenue recognition in the contracts. Usually, these reports are based on the level of a single contract and contain basic information coming from source applications and calculations done by RAR.

Following are a few examples of operational reports:

- **Monitor Revenue Contract**

This report will display details related to the revenue accounting contracts. You can check the activities on the contract using this report.

- **Revenue Contract Search (obsolete in SAP Fiori, but still convenient)**

This will display a list of contracts in the system, with hotspots on the contract number. Once you click the contract, you can see the details, such as the revenue schedule, fulfillment information, invoice information, and so on, as shown in [Figure 6.52](#).

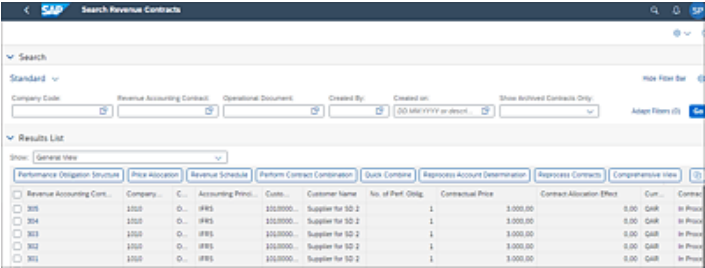


Figure 6.52 Search Revenue Contracts

- **Manage Revenue Contracts**

This will display the details of the contracts such as the **Contract Status**, **Business Partner**, **Contractual Price**, and other contract-level details, as shown in [Figure 6.53](#).

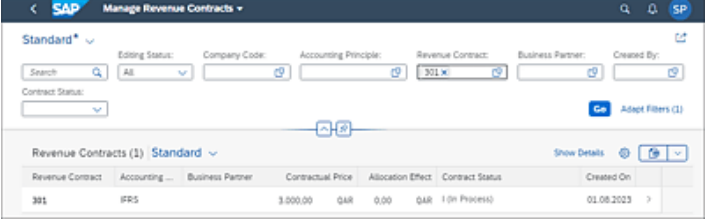
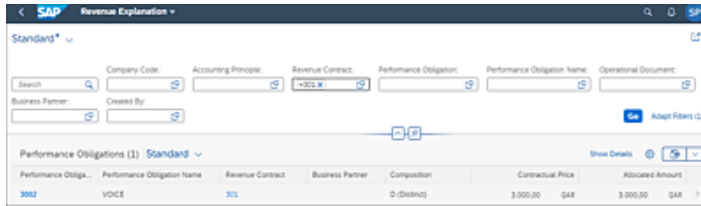


Figure 6.53 Manage Revenue Contracts

- **Revenue Explanation**

In this app, you can find the POBs of the contracts and other details, as shown in [Figure 6.54](#).



The screenshot shows the SAP Revenue Explanation app interface. At the top, there are search filters for Company Code, Accounting Principle, Revenue Contract (with value '1000000000'), Performance Obligation, Performance Obligation Name, and Operational Document. Below these are fields for Business Partner and Created By. A table titled 'Performance Obligations (1) Standard' is displayed with the following data:

Performance Obligation	Performance Obligation Name	Revenue Contract	Business Partner	Composition	Contractual Price	Allocated Amount
1000	VOICE	100	D (Default)		3,000.00 QAR	3,000.00 QAR

Figure 6.54 Revenue Explanation

Reconciliation

Posting financial documents from RAR comes as a result of calculations that are done in table FARR_D_POSTING. Therefore, it isn't an unusual requirement for reconciliation between these two modules.

When it comes to reconciliation between revenue accounting and the general ledger, two separate activities are required:

- Reconciliation between revenue postings and general ledger
- Reconciliation of accounts between revenue accounting and general ledger

In SAP S/4HANA, new reports enable this reconciliation to happen in a more transparent and clear way:

- Comparison of G/L Accounts - Revenue Accounting and G/L
- Reconciliation - Subledger and G/L

The first thing we need to look at is running analysis of financial accounting documents that came from RAR. To do this, use the Comparison of G/L Accounts - Revenue Accounting and G/L app. Documents in RAR that are posted come as a grouping on the controlling object level. When you run the application, you need to enter search criteria that are either a period/year combination or based on **Run ID** or **G/L Document** number. Once you run it, the screen shown in [Figure 6.55](#) appears.

Here is the visible split of the documents from the RAR perspective: by choosing **Run ID**, you're controlling which contracts will appear in the report. There's also a list of the visible split for POB types, condition types, and posting categories, making this report a mix between tables FARR_D_POSTING and ACDOCA.

However, this report can't tell you everything from RAR actually posted in finance. We need to be very clear on the expectations coming from this report: as mentioned, the period won't be able to be closed before posting is actually performed. Therefore, differences between tables ACDOCA and FARR_D_POSTING can happen only in some border cases.

G/L Document	Revenue Acc.	Performance Obl.	Posting Cat.	Condition Type	G/L Account	Account Descr.	Debit/Credit	Amount	Currency
5300000000	300313225	202596	REAGENT (RR)	Receivable Adjus...	1121211200	AR Trade -RAR-Aq	Credit	175,15	EUR
	300313225	202596	REAGENT (RR)	Receivable Adjus...	ZP02		Debit	282,37	EUR
	300313225	202597	REAGENT (RR)	Receivable Adjus...	ZALL		Credit	38,84	EUR
	300313225	202597	REAGENT (RR)	Receivable Adjus...	ZP02		Debit	141,28	EUR
	300313225	202598	REAGENT (RR)	Receivable Adjus...	ZALL		Credit	8,68	EUR
	300313225	202598	REAGENT (RR)	Receivable Adjus...	ZP02		Debit	13,42	EUR
	300313225	202599	INSTRUMENT (SEA...	Receivable Adjus...			Credit	176,26	EUR
	300313225	202599	INSTRUMENT (SEA...	Receivable Adjus...	R100		Credit	646,60	EUR
	300313225	202599	INSTRUMENT (SEA...	Receivable Adjus...	ZALL		Debit	176,26	EUR
	300313225	202599	INSTRUMENT (SEA...	Receivable Adjus...	ZPST		Debit	640,80	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...			Credit	12,20	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...	R100		Credit	41,69	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...	ZALL		Debit	12,20	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...	ZPST		Debit	41,69	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...			Credit	3,96	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...	R100		Credit	11,10	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...	ZALL		Debit	3,96	EUR
	300313225	202600	INSTRUMENT (SEA...	Receivable Adjus...	ZPST		Debit	11,10	EUR

Figure 6.55 Financial Postings from RAR

Once you run the Reconciliation – Subledger and G/L report, it displays the results shown in [Figure 6.56](#).

The report works in traffic light mode: if there are differences between postings in RAR and table ACDOCA, they will appear with a red light. If everything is OK, the traffic light will be green. Again, you can make selections based on the general ledger account, and reconciliation is possible because table FARR_D_POSTING also contains general ledger account numbers for postings derived from BRFplus settings.

Status	GL Account	Account Description	Revenue Accounting in	General Ledger in Doc.	Difference in Document C.	Document Currency	Revenue Accounting in	General Ledger in LC	Difference in LC	Local Currency
	111111100	20.442,85	20.442,85	0,00	EUR	20.442,85	20.442,85	0,00	EUR	
	111112000	938,24	938,24	0,00	EUR	938,24	938,24	0,00	EUR	
	210101000	10.051,42	10.051,42	0,00	EUR	10.051,42	10.051,42	0,00	EUR	
	411111200	608,64	608,64	0,00	EUR	608,64	608,64	0,00	EUR	
	411111202	1.369,61	1.369,61	0,00	EUR	1.369,61	1.369,61	0,00	EUR	
	411111202	763,62	763,62	0,00	EUR	763,62	763,62	0,00	EUR	
	411111202	20.960,22	20.960,22	0,00	EUR	20.960,22	20.960,22	0,00	EUR	
	411111202	20.290,00	20.290,00	0,00	EUR	20.290,00	20.290,00	0,00	EUR	
	411111400	10.079,77	10.079,77	0,00	EUR	10.079,77	10.079,77	0,00	EUR	
	439111000	1.111,76	1.111,76	0,00	EUR	1.111,76	1.111,76	0,00	EUR	

Figure 6.56 Reconciliation Report between General Ledger and RAR

Note

You can find PDF versions of [Figure 6.55](#) and [Figure 6.56](#) available for download at sap-press.com/5700 under the **Product supplements** section.

A prerequisite for this report to be used is that the general ledger accounts used for RAR postings aren't being used for any kind of different manual postings. If that is done, this report will show differences.

There is another app for reconciliation that is frequently used: Revenue Schedule. The Revenue Schedule app can be accessed from the SAP Fiori home screen or through the Search Revenue Contract app. Once you select a contract from the Search Revenue Contract app, you have the option to click **Revenue Schedule** at the top of the screen. After navigating to the Revenue Schedule app, filling in the selection parameters, and clicking **GO**, you'll arrive at the screen shown in [Figure 6.57](#), which provides the list of contracts for your selection criteria.

Revenue Contract	Accounting ...	Business Partner	Contractual Price	Allocation Effect	Created On	Created By
93	IFRS		1.200.000	KWD	0,000	KWD 13.09.2023 SPRABHA
92	IFRS		1.200.000	KWD	0,000	KWD 13.09.2023 SPRABHA
91	IFRS		1.200.000	KWD	0,000	KWD 13.09.2023 SPRABHA
90	IFRS		100.000	KWD	0,000	KWD 13.09.2023 SPRABHA
89	IFRS		300.000	KWD	20,000	KWD 12.09.2023 SPRABHA
88	IFRS		300.000	KWD	20,000	KWD 12.09.2023 SPRABHA
87	IFRS		600.000	KWD	40,000	KWD 12.09.2023 SPRABHA

Figure 6.57 Revenue Schedule Initial Screen with List of Contracts

The first column, **Revenue Contract**, is hotspot enabled. Once you click the contract that you want to view the revenue schedule for, you'll see the screen shown in [Figure 6.58](#).

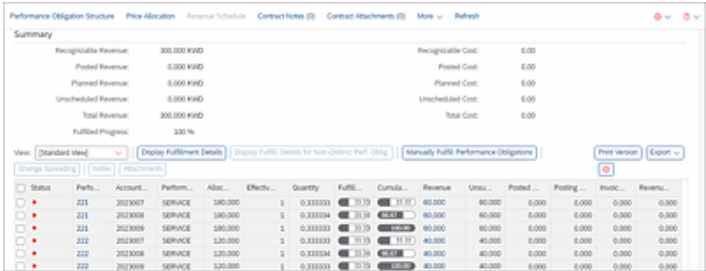


Figure 6.58 Revenue Schedule: Display for Selected Contract

The columns shown in [Figure 6.58](#) can be customized or changed per your requirements by clicking the **Settings** icon in the top-right corner. You'll open the screen shown in [Figure 6.59](#), where you can do the following:

- Select columns from **Hidden Columns** to display
- Remove unwanted columns from the displayed columns
- Move columns to your desired location (either to the top or bottom) using the arrows
- Save the layout and provide a name

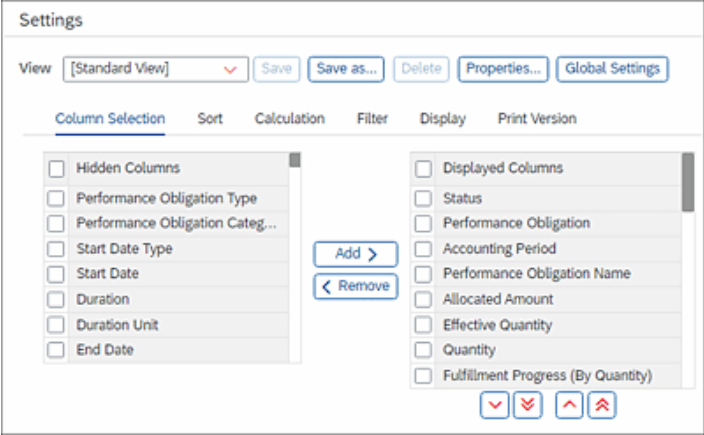


Figure 6.59 Choose the Columns to Be Displayed or Hidden for Revenue Schedule

Now, let's refer back to [Figure 6.58](#) and walk through the columns:

- **Status**

Shows a traffic light (red, in our example). The colors have the following meanings:

- No light: The revenue is just in the billing plan
- Orange: Revenue is to be recognized in the future
- Gray: Revenue is recognized, but not yet posted
- Green: Revenue is recognized and fully posted
- Red: Revenue is recognized, but not posted

- **Performance Obligation No.**

The second column displays the numbers of the POBs; in this case, there are two POBs (**221** and **222**).

- **Accounting Period**

This is the accounting period for each of the time-based POBs; there will be an item for each accounting period from the start date to the end date of the POB. For event-based POBs, there will be a single line, and it will have the accounting period value as **Unscheduled**, since they will not have a duration and will instead be recognized as a point in time.

- **Performance Obligation Name**

Here you can find the name of the POB.

- **Allocated Amount**

Here you can find the allocated amount per POB.

- **Effective Quantity**

This is the quantity value for the contract.

- **Quantity**

This shows the value of the quantity per POB per period.

- **Fulfillment Progress**

This percentage indicates how far the POB has progressed in an accounting period toward its completion.

- **Cumulated Percentage of Completion**

This is an accumulated value of the fulfillment progress for the

particular POB for the specific accounting period.

- **Revenue**

This column has the value of the allocation revenue for the POB for that accounting period.

The Revenue Schedule app gets its data from table FARR_D_DEFITM, which we discussed in [Chapter 5](#).

Reporting in Classic versus Optimized Contract Management

Classic contract management (CCM) has very limited reporting options, whereas optimized contract management (OCM) offers extended and enhanced reporting options ([Section 6.4.2](#)). [Table 6.4](#) clearly outlines the reports that are available in CCM and OCM.

SAP Fiori ID	SAP Fiori App Name	Type	Supports CCM	Obsolete
F4068	Disaggregation of Revenue (Design Studio)	Analytical, SAP Business Warehouse (SAP BW) query used	Yes	Yes
F4269	Disaggregation of Recognized Revenue (Design Studio)	Analytical, SAP BW query used		Yes
F4620	Contract Balance (Design Studio)	Analytical, SAP BW query used	Yes	Yes

SAP Fiori ID	SAP Fiori App Name	Type	Supports CCM	Obsolete
F4702	Remaining Performance Obligation - with Time Bands (Design Studio)	Analytical, SAP BW query used	Yes	Yes
F4703	Remaining Performance Obligation (Design Studio)	Analytical, SAP BW query used	Yes	Yes
F4956	Contract Balance Movements (Design Studio)	Analytical, SAP BW query used		Yes
W0154	Contract Balance Movements	Analytical, SAP BW query used		
W0155	Contract Balance	Analytical, SAP BW query used	Yes	
W0156	Disaggregation of Recognized Revenue	Analytical, SAP BW query used		
W0157	Disaggregation of Revenue	Analytical, SAP BW query used	Yes	
W0158	Remaining Performance Obligation - with Time Bands	Analytical, SAP BW query used	Yes	

SAP Fiori ID	SAP Fiori App Name	Type	Supports CCM	Obsolete
W0159	Remaining Performance Obligation	Analytical, SAP BW query used	Yes	
W0169	Revenue Catch-Up	Analytical, SAP BW query used		
W0176	Contract Balance Reclassification	Analytical, SAP BW query used	Yes	

Table 6.4 Reporting Apps in SAP Fiori

Note that the Contract Balance Reclassification app works exclusively with OCM and optimized inbound processing (OIP) only.

Some other standard reports that we want to mention here are shown in [Table 6.5](#).

Transaction	Program Name	Description
Transaction FARR_BIZ_RECON	FARR_BIZ_RECON	Reconcile the operational data with revenue accounting
Transaction FARR_MIG	FARR_MIGRATION_COCKPIT	Migration cockpit
Transaction FARR_INI_LOAD_REPORT	FARR_INITIAL_LOAD_REPORT	Initial load report for background processing

6.4.2 New Styles of Reporting and CDS Views

In the previous section, we discussed standard reports and the transition of reporting. In this section, we'll discuss the reporting styles that we adopted in the custom reports we developed during our projects. Reporting is crucial to run a business, and it's important to optimize the reports' execution time to provide users the data they need in real time without making them wait for long periods. With OCM and SAP S/4HANA, designing reports has been shifted using CDS views. Reports using CDS views are one part of the discussion, along with some other technical approaches to reporting.

Here, we'll be discussing the technical approach that we used to optimize our reports. During this time, we faced many challenges when trying to develop custom reports, of which the most prominent ones were memory management issues and performance management issues.

The technical approach being explained here is more relevant for technical consultants. It's undeniable that there may be many more optimized techniques, but these are the ones we used, and it could be useful for you to try them as well.

Staging Tables for Reporting

Consider a situation where report execution is taking a long time, and the user has to schedule it for background execution and then come back after the job has been completed to check the output. The output in the spool isn't very user-friendly and isn't so convenient for formatting and creating the desired format for display.

So, we planned to make one place in the program where the program itself schedules the job and fills the table, which is in the output format. We created a database table to store the output data, which is called a staging table, as mentioned previously. The report

program will have a two-step execution in which the first step is to first fill the output table, and the second step will display the output from the table. A flowchart to demonstrate the flow and the execution is shown in [Figure 6.60](#).

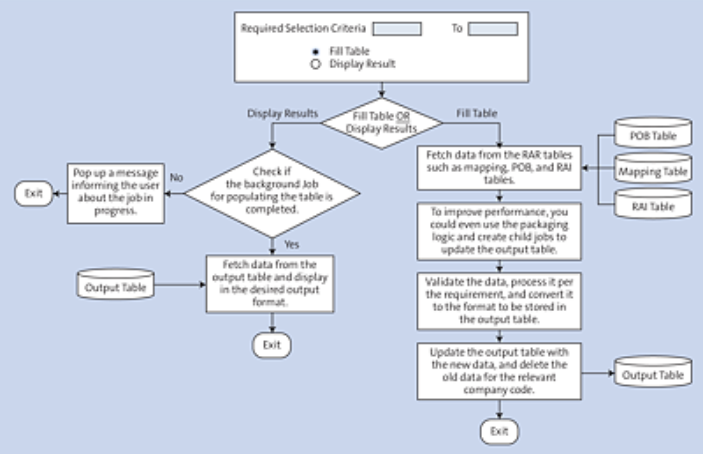


Figure 6.60 Flowchart for Reporting with Staging Tables

To demonstrate the usage of staging tables, let’s take a closer look at an example flow, which is provided in a generic format as the desired functionality and requirement could be added in the processing per the project requirement. The high-level process follows:

1. Say we have a requirement to design a custom reconciliation report to reconcile the data between SAP Billing and Revenue Innovation Management and RAR. The selection screen of the program will be designed using Transaction SE38, as shown in [Figure 6.61](#).

Figure 6.61 Sample Selection Screen of Reconciliation Report

- The mandatory things on the selection screen will be to include two push buttons. One will be for filling the table, and the other will be for the display of the output, as shown in [Figure 6.62](#).



Figure 6.62 Push Buttons to Display the Data and Fill the Data to the Table

- The **Fill Orders** button is designed to fill the table with the output data. We need to create a database table where the data from the program execution will be filled. Using Transaction SE11, we'll create a database table that will match the output format requirement. For our requirement, we created the table, and the data was filled, as shown in [Figure 6.63](#).
- When the user clicks on the **Fill Orders** button, the program will trigger a job in the background. The job will pull the necessary data from different RAR tables or CDS views that will be required per the logic and then save it in the database table. The previous entries will be overwritten. The scheduled job can be viewed using Transaction SM37, as shown in [Figure 6.64](#).

ID	NAME	TASK	PROGRAM	START DATE	END DATE	DURATION	DELAY	EXECUTING SERVER	TARGET HOST	STATUS			
879	23	CAR	SD3300000117022303CPCL	1000	23134073	OCF	28 11 2022	1009048	10	MAIL01R402124	4.1	P	Missing license for Credit/Rate
879	23	CAR	SD3300000117022303CPCL	1000	23134073	OCF	28 11 2022	1009048	10	MAIL01R402124	2.1	P	Missing license for Credit/Rate
879	23	CAR	SD3300000117022303CPCL	1000	23134073	OCF	28 11 2022	1009048	10	MAIL01R402124	3.1	P	Missing license for Credit/Rate
438	23	CAR	SD3300000117022303CPCL	1000	23113145	VF1	29 11 2022	1010142	10	MAIL01R10434	3.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23175796	CHC	30 11 2022	1061994	10	MAIL01R175796	3.1	P	Missing license for Credit/Rate
874	23	CAR	SD3300000117022303CPCL	1000	23113145	VF1	29 11 2022	1010142	10	MAIL01R10434	5.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23175796	CHC	30 11 2022	1061994	10	MAIL01R175796	2.1	P	Missing license for Credit/Rate
872	23	CAR	SD3300000117022303CPCL	1000	23174758	CY3	31 12 2022	1070722	10	MAIL01R174758	2.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23174758	CY3	31 12 2022	1070722	10	MAIL01R174758	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23133486	CHC	31 12 2022	1071512	10	MAIL01R33486	4.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23133486	CHC	31 12 2022	1071512	10	MAIL01R33486	13.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	3.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	3.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	3.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	3.1	P	Missing license for Credit/Rate
337	23	CAR	SD3300000117022303CPCL	1000	23191750	PAC	13 01 2023	1047878	10	MAIL01R191750	2.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
882	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	3.1	P	Missing license for Credit/Rate
432	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	4.1	P	Missing license for Credit/Rate
876	23	CAR	SD3300000117022303CPCL	1000	23188517	ESP	30 11 2022	1068601	10	MAIL01R8517	2.1	P	Missing license for Credit/Rate

Figure 6.63 Data Stored in the Table by Jobs

JobName	Spool	Job Creation	Status	Start date	Start T1	End time	Duration(s)	Delay	Executing server	Target Host
CREDIT		10.07.2023 07:42:10	Finished	10.07.2023 07:41:13	07:42:20	47	0	0	MAIL01R402124_04	
Summary						114	0	0		

Figure 6.64 Job Scheduled to Save Entries in the Table

- 5. When the user needs multiple different formats of output, you can design the database table to include all the columns that will be needed for all the output formats. The required output format can be designed in the program to display it with the necessary processing.
- 6. We need to check whether the job for filling the table is already scheduled, and then we need to pop up a message informing the user that the job is in progress already for filling the table, as shown in Figure 6.65. If there is no job running or scheduled, then we need to fetch the required data from the RAR tables, perform the required processing per the design requirements, and then populate the database table, which is designed for output. The message informs the user that the job is already scheduled.

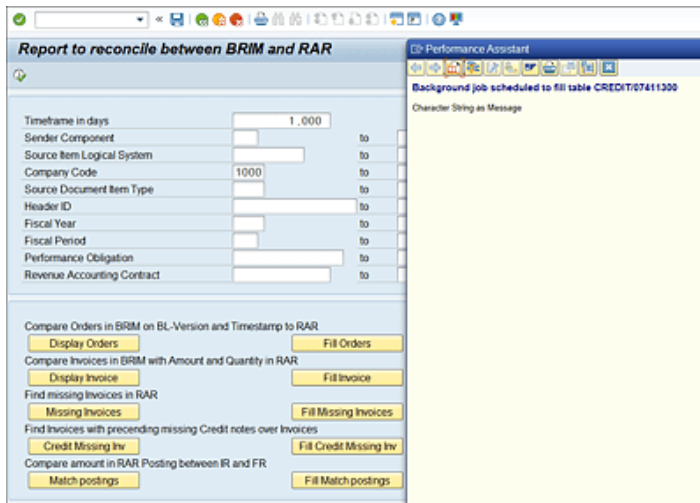


Figure 6.65 Message to Inform User That a Job Is Scheduled to Fill the Table

7. We can further enhance the design for performance by creating child jobs and adopt data packaging, which is discussed in [Chapter 7, Section 7.1](#), during our coverage of the custom parallel processing framework.
8. The other option is to display the data (the **Display Orders** button). There can be multiple formats, and they all should be programmed as options on the selection screen. Then, the data will have to be fetched from the output table, processed per the output format required, and then displayed in the grid format. A sample from our program is displayed in [Figure 6.66](#).

Se	Srtm/Type	Source Item ID	CoCd	ZZ BL UBL1	ZZ Booking	ZZ Charge Posting Date	ZZ In	Partner	Item ID	ZZ BL Vers	ZZV
ZG	CAB	8230400099587000COCNN	1000	MAEU224963092	224963092	09.02.2023		13349900	50	2 P	
ZG	CAB	8230400046626857000COCNN	1000	MAEU23210316	L23210316	09.02.2023		10503533	50	2 P	
ZG	CAB	823018004656785006COCNN	1000	MAEU222978122	1K7772446	20.01.2023		14268801	50	2 P	
ZG	CAB	8230330048747598004COCNN	1000	MAEU224295845	224295845	02.02.2023		10309596	50	2 P	
ZG	CAB	82304500506339450120COCNN	1000	MAEU8KA011496	8KA011496	14.02.2023		12530720	50	2 P	
ZG	CAB	823024004733103007COCNN	1000	MAEU224317174	1K7303334	20.01.2023		12532781	50	2 P	
ZG	CAB	8230310048457082008COCNN	1000	MAEU224764385	224764385	02.02.2023		14426573	50	2 P	
ZG	CAB	82304700511445040100COCNN	1000	MAEU231100099	L231100099	16.02.2023		10503533	50	2 P	
ZG	CAB	8230400048947259006COCNN	1000	MAEU224187453	224187453	09.02.2023		10164586	50	2 P	
ZG	CAB	82304000499504140120COCNN	1000	MAEU224527950	224527950	09.02.2023		10164586	50	2 P	
ZG	CAB	8230240047397306206COCNN	1000	MAEU224511593	224334730	20.01.2023		20048009	50	2 P	
ZG	CAB	8230470051143239004COCNN	1000	MAEU231949129	L231949129	16.02.2023		10503533	50	2 P	
ZG	CAB	8230400046626917002COCNN	1000	MAEU23210582	L23210582	09.02.2023		10503533	50	2 P	
ZG	CAB	82304500507449230120COCNN	1000	MAEU914846875	914846875	14.02.2023		13624537	50	2 P	
ZG	CAB	8230330048771932040COCNN	1000	MAEU224711601	224637527	02.02.2023		10311914	50	2 P	
ZG	CAB	8230110046264446006COCNN	1000	MAEU222612038	222612038	20.01.2023		10115142	50	2 P	
ZG	CAB	8230400049947413008COCNN	1000	MAEU224405487	224405487	09.02.2023		10164586	50	2 P	
ZG	CAB	82303300488549490120COCNN	1000	MAEU914821912	914821912	02.02.2023		14235937	50	2 P	
ZG	CAB	8230230047260488006COCNN	1000	MAEU224302116	224302116	20.01.2023		10782100	50	2 P	
ZG	CAB	8230230047188119018COCNN	1000	MAEU224327029	224327029	20.01.2023		13813939	50	2 P	
ZG	CAB	82304500507250720100COCNN	1000	MAEU009407217	009407217	14.02.2023		11128419	50	2 P	
ZG	CAB	8230400049950490140COCNN	1000	MAEU224555181	224555181	09.02.2023		10196309	50	2 P	
ZG	CAB	823047005111810700100COCNN	1000	MAEU224471940	224471940	16.02.2023		14423958	50	2 P	
ZG	CAB	82303300488549780100COCNN	1000	MAEU914833006	914833006	02.02.2023		14235937	50	2 P	
ZG	CAB	8230470051124780100COCNN	1000	MAEU225273155	224709103	16.02.2023		20260804	50	2 P	
ZG	CAB	82304500506386270140COCNN	1000	MAEU224741432	224741432	14.02.2023		10318009	50	2 P	

Figure 6.66 Sample Report Output

CDS Views for Reporting

CDS views are considered an advanced feature introduced by SAP in SAP NetWeaver AS for ABAP 7.40 SP05. CDS views are designed using SQL data definition language (DDL) syntax. CDS views offer optimal code pushdown as data model definitions are executed directly in the database layer. You can perform complex operations and calculations directly in the database. The CDS view is a changed approach to the belief that we needed to give a lesser load to the database and perform complex operations on data in the application layer. Now, with the CDS views and the SAP HANA database, the rule is to perform most of the calculations, aggregations, and other complex operations in the database itself to optimize performance. The data is definitely in the tables, as the name itself suggests that it's a view. CDS views are like coordinators that understand the tables and their relationships and help you access data and perform operations at a faster pace. Using CDS views is now common, and SAP even suggests that you use CDS views for building reports for OCM.

SAP has provided lot of CDS views in RAR that you can use for reporting. [Table 6.6](#) provides a list of standard CDS view provided by SAP. These CDS views directly reflect the sample reports we discussed in [Section 6.4.1](#).

Technical Name	SQL View Name	Description
I_RADEFRLITEMFORDSPCRYCUBE	IRADEFRLITEMCUBE	Revenue Schedule Deferral Item - Cube
C_RAYRTODTERECGDREVNQUERY	CRAYTDRECOGDREVQ	Disaggregation of Recognizable Revenue
I_RAPOSTINGITEMCUBE	IRAPOSTITEMCUBE	Posting Item - Cube

Technical Name	SQL View Name	Description
C_RAPOSTINGITEMQUERY	CRAPOSTITEMQUERY	Disaggregation of Revenue
I_REVNACCTGFULFILLMENT	IRAFULFILLMENT	Revenue Accounting Fulfillment
I_REVNACCTGDEFERRALITEM	IRADEFERRALITEM	Revenue Accounting Deferral Item
I_REVNACCTGSCHEDULEITEM	IRASCHEDITEM	Composite Revenue Accounting Schedule Item
I_REVENUEACCOUNTINGPOSTING	IRAPOSTING	Revenue Accounting Posting
I_REVENUEACCOUNTINGINVOICE	IRAINVOICE	Revenue Accounting Invoice
I_RAPERFOBLGNCHANGETYPEITEM	IRACHANGETYPEITM	Performance Obligation Change Type Item
C_RATOTALOPENREVENUEQUERY	CRATTOPNRVNQRY	Remaining Performance Obligations
I_RATOTALOPENREVENUECUBE	IRATTOPNRVNCB	Remaining Performance Obligations - Cube
C_RAOPENREVENUEPERPERIODQUERY	CRAOPNRVNPRDQRY	Remaining Performance Obligations with Time Bands - Query

Technical Name	SQL View Name	Description
I_RAOPENREVENUEPERPERIODCUBE	IRAOPNRVNPRDCB	Remaining Performance Obligations with Time Bands - Cube
C_RACONTRACTBALANCEQUERY	CRACONTRACTBALAQ	Contract Balance
I_RACONTRACTBALANCECUBE	IRACONTRBALCUBE	116A Contract Balance: Cube View
C_RACONTRBALANCEMOVEMENTQUERY	CRACONTRBALMVQ	Contract Balance Movements
I_RACONTRBALANCEMOVEMENTCUBE	ICONTRBALCUBE	Contract Balance Movements Cube View
C_RAANLYTSOVRVLSPPRICETESTQUERY	CRASACSSPTQ	Overall Result of SSP Test - Query
C_RAANALYTICSSSPRICETESTQUERY	CRASACSSPTSTQ	Test of Standalone Selling Price - Query

Table 6.6 CDS Views Available for Reporting in RAR

You should also create CDS views on the basic RAR tables FARR_D_DEFITEM and FARR_D_POB, which are huge tables with large volumes of data. These CDS views can then be used in the program. Along with these tables, you can use any other table or combination of tables such as table FARR_D_MAPPING, FARR_D_POB, and FARR_D_CONTRACTS that we want for our reports.

You can create CDS views using the Eclipse integrated development environment (IDE) with the ABAP Development Tools (ADT) or using SAP HANA Studio. However, the steps to create a CDS view are out of the scope of this book. Once the CDS view is created, you can use

it in the program. The syntax for calling a CDS view to fetch data is as follows:

```
Select * from ZCDS_VIEW into table @data(lt_itab).
```

Using AMDP in Reporting

Using ABAP-Managed Database Procedures (AMDPs), you can write code inside AMDP by using SQLScript, which is a database language. Similar to using SQL scripts, this language is easy to understand and code. After coding the logic inside the AMDP method, you can consume it in an ABAP report.

With AMDP, we can take advantage of new SAP HANA features (e.g., the code push-down technique), so all logic is still coded on the application layer, and then this logic will be executed on the database layer. AMDP is only supported in an ADT bundle or SAP HANA Studio. However, an AMDP method needs to be defined with standard interface `IF_AMDP_MARKER_HDB` to let the compiler know that it's an AMDP method; therefore, it's like an instruction that the code execution must be pushed to the database. AMDP needs to be implemented in SQLScript, which is the native language for the SAP HANA database.

As developers, it's a common situation to see that our program is taking much longer than we expected. For example, say that in the general analysis of our code, there are no time-consuming `SELECT` statements, but when we run Transaction ST05, we find in the analysis that the ABAP part is taking a long time. With the introduction of AMDP, we won't be facing this scenario, as we can write our complex logic or logic involving a lot of data traffic or data processing in the AMDP method. Once we write the code in the AMDP method, the code is pushed to the database and executed in the database, reducing a lot of data transfer and processing time. This helps us in designing performance-critical objects.

Further Resources

Note that complete technical instructions for using ABAP-related tools are beyond the scope of this book. For more information, see <https://developers.sap.com/tutorials/abap-environment-amdp-profiling.html>.

To create AMDPs, follow these basic steps:

1. Create a class using ADT.
2. Inside the class, you must have the declaration shown in [Figure 6.67](#) to know that this is an AMDP method.

```
PUBLIC SECTION.  
  
INTERFACES if_amdp_marker_hdb .
```

Figure 6.67 AMDP Method Declaration

3. Declare all the types and methods that you need, as shown in [Figure 6.68](#) and [Figure 6.69](#), respectively. You can declare all the required types (you could create the types using Transaction SE11 and use them here as well). Make sure you use the MANDT field, which carries the client number. This is mandatory as AMDP doesn't handle automatic detection of clients.

In our example for RAR, we developed a custom reconciliation report in one of our projects and had to declare a TYPE with fields, as shown in [Figure 6.68](#).

```
TYPES:  
BEGIN OF ltys_ReconciliationPosting,  
Mandt                TYPE mandt,  
CompanyCode          TYPE bukrs,  
AccountingPrinciple TYPE accounting_principle,  
RevnAcctgReconciliationKey TYPE farr_recon_key,  
PerformanceObligation TYPE farr_pob_id,  
ConditionType        TYPE kscha,  
RevnAcctgPostingCategory TYPE farr_post_category,  
DebitCreditCode      TYPE shkzg,  
RevnAcctgPostingUUID TYPE farr_posting_guid,  
ReclassificationIndicator TYPE          reclassind,  
2RevnAcctgReconciliationKey TYPE farr_recon_key,  
FiscalYear           TYPE gjahr,  
FiscalPeriod         TYPE poper,  
RevnAcctgPostAmtInSlsDecCrcy TYPE farr_amount_tc,  
SalesDocumentCurrency TYPE waers,  
RevnAcctgPostAmtInCoCodeCrcy TYPE farr_amount_lc,  
CompanyCodeCurrency   TYPE rwaer,  
RevnAcctgPostAmtInAddCur1 TYPE farr_amount_lc2,  
AdditionalCurrency1   TYPE hwaer2,  
RevnAcctgPostAmtInAddCur2 TYPE farr_amount_lc3,  
AdditionalCurrency2   TYPE hwaer3,  
RevenueAccountingContract TYPE farr_contract_id,  
PLAccount             TYPE saknr,  
ConditionIsForStatistics TYPE kstat,  
RARPerformanceObligationType TYPE farr_pob_type,  
DebitCreditCodeSales  TYPE farr_shkzg_va,
```

Figure 6.68 Types Declaration in the AMDP Class

4. For the same custom reconciliation report, we had to write methods; a sample of the method syntax is shown in [Figure 6.69](#). In this method, we're fetching the details of the contract and processing it for reconciliation.

```
METHODS
get_RevenueAccountingContract
AMDP OPTIONS
READ-ONLY
CDS_SESSION_CLIENT iv_clnt
IMPORTING
VALUE (iv_clnt)                TYPE mandt
VALUE (iv_companyCode)         TYPE bukrs
VALUE (iv_accountingPrinciple) TYPE accounting_principle
VALUE (iv_postingDate)         TYPE budat
VALUE (iv_filter)              TYPE ltyv_filter DEFAULT ''
EXPORTING
value (lt_view)                type ltyt_RevenueAccountingContract
value (lt_tabl)                type ltyt_RevenueAccountingContract
value (lt_all)                 type ltyt_RevenueAccountingContract
VALUE (et_RevenueAccountingContract) TYPE ltyt_RevenueAccountingContract
VALUE (et_ReconciliationPosting) TYPE ltyt_ReconciliationPosting
VALUE (et_DistinctContract)    TYPE ltyt_DistinctContract
RAISING
cx_amdp_execution_failed.
```

Figure 6.69 Method Declaration Sample

5. Now you implement the AMDP method in the required format shown in [Figure 6.70](#).

In the code snippet, you can see the particular format that is adopted. The meaning of the syntax is as follows:

- **BY DATABASE PROCEDURE**
Implements a database procedure. This is to inform your AMDP method to automatically create a procedure in the SAP HANA system.
- **FOR HDB**
Indicates the SAP HANA database.
- **LANGUAGE SQLSCRIPT**
Indicates the database-specific language in which AMDP is implemented.
- **OPTION READ-ONLY**
Indicates that you can only perform a read in the database procedure.
- **USING <name of table/view>**
Provides the name of the table if you use tables in this procedure.


```

METHOD get_revenueAccountingContract
BY REFERENCE PROCEDURE FOR NON LANGUAGE SQLSCRIPT
OPTIONS
  READ-ONLY
USING
  lt_Rev_Contract
  lt_Rev_AccountingContract
  lt_Rev_ReconciliationPosting
  lt_Rev_PerformanceObligation.
IF (lv_filter = 'TRIAL')
  then
    = select dat.* from lt_Rev_PerformanceObligation as dat
      where dat.mandt = lv_clnt
      and dat.RevenueAccountingContract is (select RevenueAccountingContract from lt_Rev_Contract where mandt = (lv_clnt and lclid = '2' ) )
  lv_Rev_row = select dat.* from lt_Rev_AccountingContract as dat
      where dat.mandt = lv_clnt
      and dat.RevenueAccountingContract is (select RevenueAccountingContract from lt_Rev_Contract where mandt = (lv_clnt and lclid = '2' ) )
  lv_Rev_row = select dat.* from lt_Rev_ReconciliationPosting as dat
      where dat.mandt = lv_clnt
      and dat.RevenueAccountingContract is (select RevenueAccountingContract from lt_Rev_Contract where mandt = (lv_clnt and lclid = '2' ) )
  else
  -- Get all information from PerformanceObligation as per selection on contracts
  lt_Rev_row = select * from lt_Rev_PerformanceObligation where mandt = (lv_clnt
  lt_Rev_row = APPLY_FILTER ( lt_Rev_row , lv_filter )
  -- Get the FOR us Invoice to document as given in lv_filter
  lt_Rev_row = APPLY_FILTER ( lt_Rev_row , lv_filter )
  lv_Rev_row = APPLY_FILTER ( lt_Rev_ReconciliationPosting , lv_filter )
  lv_Rev_row = APPLY_FILTER ( lt_Rev_ReconciliationPosting , lv_filter )
  END IF.

```

Figure 6.70 Implementing the AMDP Method

6. Next you consume the AMDP. In any of our reports, as already mentioned for the custom reconciliation report, we needed to create an AMDP, and we've mentioned the syntax for types and methods in the preceding points. Once the AMDP is ready, you can consume it. The syntax for calling an AMDP is very similar to calling any other methods of a class, as shown in [Figure 6.71](#).

```

TRY.
CALL METHOD go_reconciliationpostings->get_revenueaccountingcontract
EXPORTING
  iv_clnt                = sy-mandt
  iv_companycode         = ls_contract_first-bukrs
  iv_accountingprinciple = gm_prin
  iv_postingdate          = lv_date_type
  iv_filter               = lv_filter
IMPORTING
  et_revenueaccountingcontract = lt_RevAccContracts
  et_reconciliationposting      = lt_recon_posting
* et_distinctcontract          = gt_contract
.
CATCH cx_root INTO DATA(lo_x_root).
DATA(lv_message) = lo_x_root->get_text( ).
ENDTRY.

```

Figure 6.71 Consuming AMDP in the ABAP Program

7. You can see filter fields called `iv_filter = lv_filter`. The filter is for refining your selection by eliminating or including entries. This depends on how the filter is coded. In our case, the filter was used to provide the range of contracts that we wanted the AMDP to consider and get the data for only those relevant contracts. The filter needs to be coded in the syntax shown in [Figure 6.72](#).

```
lv_filter = |RevenueAccountingContract between '( ls_contract_first-contract_id )' and '( ls_contract_last-contract_id )'|.
```

Figure 6.72 Filter in AMDP

6.5 Summary

Throughout this book so far, you've seen that postings according to IFRS 15 bring changes to finance departments both for amounts that are going to be posted and also the number of entries done on different accounts. Before the project of IFRS 15 compliancy is even started, an organization needs to evaluate and estimate the expected impact. Because there is a shift being made between revenue that is recognized at a point in time and over time, there might be an effect on different departments. All of these parts of an organization need to be prepared for the change ahead.

In addition, there is a list of additional activities that will be done either during the month or at the month end. There is an impact on change management to prepare teams for these changes in the sense of defining additional roles and providing additional training.

The last step is the posting itself, which we discussed in this chapter, and RAR is flexible in this sense. One activity can be done in multiple ways, and it will be up to the organization to choose. For example, if you need to see revenue split on different levels, this might be done by postings on different accounts, or an organization might opt for creating additional reporting either through RAR or via profitability analysis.

Future maintenance needs must not be forgotten. The number of accounts will also determine how support for

future changes will be provided. However, users should not go with too lean of an approach. For example, a good idea is to have a dedicated account for manual corrections at month end. Once the balance of a certain account is looked at, it's cleaner to know that it comes from automatized postings only. In addition, by making manual postings on the same account that is used by RAR, usage of reconciliation reports is heavily limited.

In the next chapter, we'll move on to administration and troubleshooting processes for RAR.

7 Administration and Troubleshooting

This chapter focuses on administering and troubleshooting revenue accounting and reporting (RAR) implementations, particularly around handling large volumes of data and the resulting performance issues. We'll also discuss some custom tools.

During many of our projects, we came to understand the working style of the users and the system in depth. This led us to create some utilities as well as some required deliverables in a better form or pattern that is adapted to the users. There are times when one solution suits one customer, and the same isn't valid for others; each case is different, and so is each client and its users. The one thing that was common and consistent throughout was the huge volume of data and the challenge of managing that data. There are cases when the data volumes aren't that high, but still it's better to have tools and objects designed to handle large volumes of data as we all know that businesses do grow, and so does the data volume. Additionally, the world is advancing at a fast pace, and everybody expects instant results. In these days of quick solutions, we can't afford to have a long waiting period, and no one is ready to invest their precious time in waiting, so it becomes very important for us to design our deliverables while keeping this in mind.

Technical approaches that are used for achieving high performance will be discussed in this chapter, starting with a walk-through of addressing performance in your RAR system using the custom parallel processing framework (PPF). We'll also focus on some specific solutions that have been provided for reclassification, tools for troubleshooting and navigation, utilities, and data cleanup, which you can adapt per your project requirement.

Note on Technical Content

Most of this chapter is technically focused and is intended to help technical consultants get comfortable with RAR. If you're a beginner, this is the place for you to look out for information and get details about a lot of technical topics around RAR. However, we're assuming reader familiarity with coding fundamentals such as function modules, performs, includes, and more.

7.1 Parallel Processing Framework and Performance Issues

The PPF and performance tuning are grouped together here as both address the same issue. Let's start with the PPF, specifically focusing on developing a custom PPF. Then, we'll discuss modularization and packaging techniques in the next sections.

7.1.1 What Is Parallel Processing?

There are many PPF designs that are made and are being used in many projects. There is also a standard PPF that is available in SAP. The very word says that the processing is happening in parallel. So, to make it run in parallel, we need to provide divided data to these parallel processes.

As an example, let's say there is a requirement which says you have to write a complex program that fetches entries from huge database tables and creates accounting documents for each of them and then clears the database table for a specific run ID. Then, the developer develops a custom program for it. The program takes about 2 hours to create accounting documents for 90,000 entries. The functional consultant then informs the developer that there is a sequence of programs running in the production system, and this job has to be completed in 20 minutes because another job has to start immediately after this program completes execution. As a result of this requirement, we must improve the performance of this program and reduce its execution time to 20 minutes or less so that they align with the designed sequence of the jobs. By using various performance-tuning techniques, the developer managed to achieve 1 hour execution time. At this point, where performance tuning can't be enhanced further, the best thing that we can try is to design the program using a PPF. The parallel processing technique enables us to split the entries into parts and execute each part asynchronously. In our example, we made each part contain 20,000 entries and calls to the program, which is now designed using a PPF for each of these parts separately, and we noticed that the execution time was reduced drastically. [Figure 7.1](#) explains it further.

Standard PPF is used by most of the RAR standard programs. The standard PPF program actually has two parts. The first part is the standard PPF, which acts as a controller and takes over the task of making data ranges and calling the necessary controls for executing the program. The second part is an application program that has the logic to execute the business requirements. The basic principle of the PPF is to divide up processing into individual processing steps or events in which business-specific or application-specific logic is run. The application is designed to prepare the core logic in function modules (called

callback modules). More information about the standard PPF can be found here: <http://s-prs.co/v570000>.

We'll now shift our focus to discuss the custom PPF designed for an example project, as shown in [Figure 7.1](#).

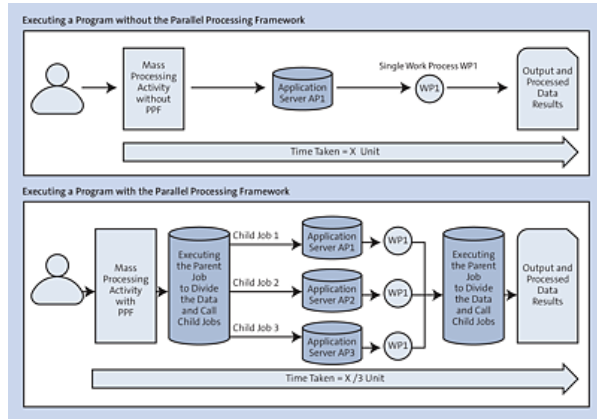


Figure 7.1 Parallel Processing Framework

In the top portion, a mass processing job is being executed without using a PPF. It's scheduled on a single application server and uses a single work process. It requires X unit of time to complete the processing and to show the results.

The lower portion shows the same job executed using the custom PPF. The job uses multiple application servers and multiple work processes. In our case, the actual work is split up into three work processes on three different application servers. The time taken for processing and to show the results is now significantly reduced to X/3 units—one-third of the actual time taken. This is the core that we need to understand. The execution process is split into parts and executed in different work processes on different or the same servers based on availability of work processes, and, as a result, the time taken for execution is significantly reduced. This is the core of parallel processing: divide and execute. So, we need to understand that we'll deploy additional system resources and work toward reducing the time taken. The important resources we use are the application servers and work processes. When it comes to normal programs, we'll use a single server and single work process; here, it changes based on the volume of data and the choice of the user to select the number of jobs to create by splitting the data. The greater the number of jobs, the less time taken for processing.

In the following sections, we'll explore these resources and how to view them in the system.

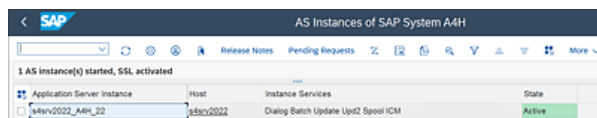
7.1.2 Application Server Instances

An SAP instance defines a group of resources such as memory, work processes, and so on, usually in support of a single application server within a client/server environment. Application servers share the same memory areas and are controlled by the same dispatcher process. We've generally seen when we provide the details in the SAP GUI that we give the system ID and then the instance number. Generally, there will be one application server and many application server instances.

In SAP, we have an application layer where the application programs are executed. It serves as a communicator between the presentation and database layers. The application server is where the dispatcher distributes the workload to the different work processes. The application server is a physical server used to handle and process the user request.

It's possible to install more than one instance on a single server provided they are differentiated by the instance number. So, we have a predefined set of application server instances that are available for any given system. The instance number is a two-digit number that varies between 00 and 97 (98 and 99 are reserved for routing purposes).

You can look at the instances available in a system by going to Transaction SM51, as shown in [Figure 7.2](#).



Application Server Instance	Host	Instance Services	State
s4sv2022_A4H_02	s4sv2022	Dialog Batch Update Upd2 Spool ICM	Active

Figure 7.2 Instances Available in a System

This transaction also shows you whether that instance is currently **Active** under the **State** column. In the **Active** state, the application server receives and processes requests and creates and sends requests to other application server instances. The **Instance Services** column shows the type of services offered by that instance. The following services (work processes) are available on the application server instance:

- **Dialog**
Dialog work processes are configured on the application server instance, which means online processing is possible.
- **Enqueue**
Enqueue work processes are configured on the application server instance and administer the lock table in the shared memory; other locking functionalities are handled here as well.
- **Update**
There are two types of update work processes: V1 and V2. They are configured

on the application server instance, which means it can execute (time-critical) V1 update tasks. V1 update jobs have higher priority than V2 jobs.

- **Upd2**
V2 update work processes are configured on the application server instance, which means it can execute (non-time-critical) V2 update tasks.
- **Batch**
Batch work processes are configured on the application server instance, which means background jobs can be run.
- **Spool**
Spool work processes are configured on the application server instance, which means print formatting can be run.
- **ICM**
The Internet Communication Manager (ICM) is configured on the application server instance, that is, connections from or to the internet can be made.
- **VMC**
The Virtual Machine Container (VMC) is active on the application server instance, which means both ABAP virtual machine and Java virtual machine (JVM) are available in the work process.
- **J2EE**
Java 2 Platform, Enterprise Edition (J2EE) is available on the application server instance.

7.1.3 Job Server Group

You can group job servers on different computers into a logical SAP Data Services component called a server group. A batch server group automatically measures resource availability on each job server in the group and distributes scheduled batch jobs to the job server with the lightest load at runtime. This means a server group looks for job servers with a lesser load and then allocates jobs to it.

We're talking about job server groups here because we'll be creating a job server group to be used in our custom PPF. A job server group is created using Transaction SM61.

In [Figure 7.3](#), you can see that in our example there are four different application servers—AP1, AP2, AP3, and AP4—and each has its own set of work processes. To create a job server group, we choose three out of the four servers to be included in the group. This is like a system that will work to share the load among themselves and allocate the work processes effectively when a group of jobs are assigned to it. In this case, we grouped AP2, AP3, and AP4 and created the job server group. So, when we schedule a job and use this job server group, then the

work processes across the application servers are used and allocated to complete the task.

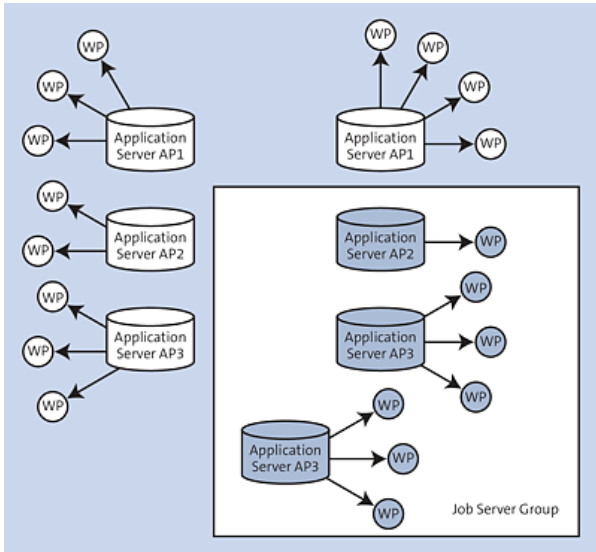


Figure 7.3 Job Server Group Illustration

Let’s now look at the steps to create a job server group:

1. Go to Transaction SM61 to arrive at the screen shown in [Figure 7.4](#).

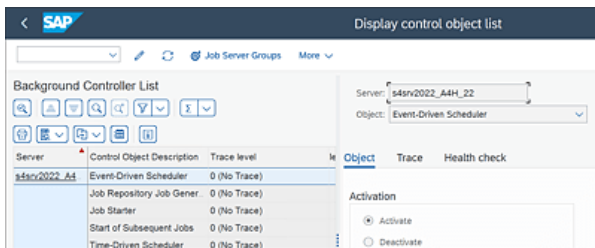


Figure 7.4 Job Server Group: Initial Screen

2. Click on the **Job Server Groups** button at the top of the screen. You’ll go to the screen shown in [Figure 7.5](#), where you click on the **Create Group** button (left-most button at the top of the screen).

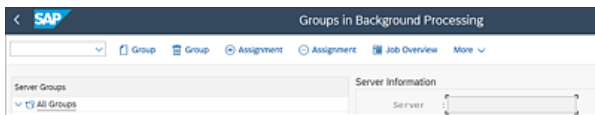


Figure 7.5 Job Server Group: Create Group

3. In the popup that appears, fill in the **Group Name** as shown in [Figure 7.6](#). Give the group name as “Job_server1”, and then click on **Continue**.

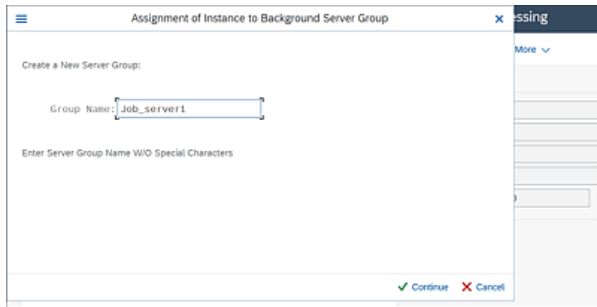


Figure 7.6 Job Server Group: Group Name

The group is now created, and you can see the group under the **Server Groups** on the left-hand side of [Figure 7.7](#). You then have to click on the **Assignment** button, which opens the window where you can add the application server instances (**InstanceName**) by choosing from the **F4** help and then clicking **Continue**. In this case, as it's a demo system, there is just one server instance, but in most practical projects, there will be multiple server instances. While selecting the servers, you have to make sure not to select all of them and leave a few for other critical system processing. You create a group by choosing multiple servers, clicking on **Assignment** as many times as the number of servers that you want to add, and selecting the required servers.

Finally, you can see the **Server Groups** and the list of application server instances added, as shown in [Figure 7.8](#).

You can either add this server group to new jobs or edit the existing jobs to include the job server group in them. The batch job group that is created will now be available as an option in Transaction SM36, as shown in [Figure 7.9](#), when you click the **Target** **F4** option. So, it's now ready for use to use in the custom PPF program.

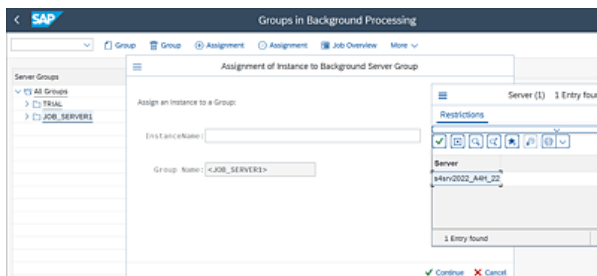


Figure 7.7 Job Server Group: Assign an Instance

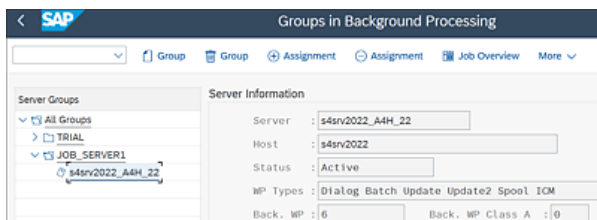


Figure 7.8 Job Server Group: Added to List

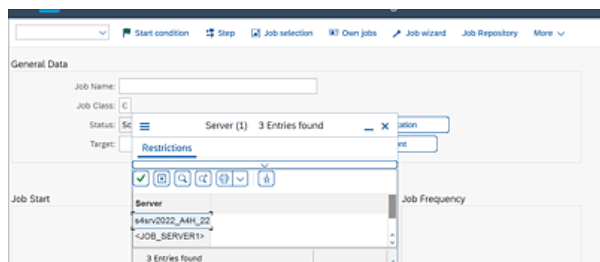


Figure 7.9 Job Server Group: Entry Found

7.1.4 Custom Parallel Processing Framework

Having explained the application server instances, job server group, and creation of job server groups and work processes, we'll now start looking at the custom PPF. The custom PPF is a framework that we designed to achieve faster processing of huge data volumes. This design has three mandatory parts. The first one is the parent job processing function module (which will have the core processing logic) and then the child jobs. The parent program is like a driver program that will provide the selection screen, identify the top node of processing, divide the date and data ranges, and create child jobs.

Once the requirement or the functional design is with the developer, you need to check the selection criteria and start with the parent program design. After creating the initial input screen of the program (called the *selection screen*), you need to analyze and identify the top node, which is the first-level data that will be extracted and divided into part of ranges and then sent to the child jobs. In RAR, the top node will generally be the HeaderID or ContractID—or anything else per the requirement.

So, let's start our deep dive into designing the PPF supported by snippets of code. We recommend following the software development lifecycle and designing a technical specification first, which is dependent on your specific requirements and beyond the scope of this book, and then start with the coding. Here, we'll directly talk about coding blocks related to the custom PPF.

With the help of the selection screen details provided in the functional design specification, we can decide the top node of processing. Let's say we have a requirement of reprocessing the revenue accounting items (RAIs) based on some validation. The selection screen will have the items shown in [Table 7.1](#).

Selection Element	Technical Name	Selection Type	Mandatory	Default Values	Description
-------------------	----------------	----------------	-----------	----------------	-------------

Selection Element	Technical Name	Selection Type	Mandatory	Default Values	Description
P_BUKRS	bukrs	Parameter	X	1000	Company code
P_PRIN	accounting_principle	Parameter	X	IFRS	Accounting principle
S_HeaderID	FARR_HEADER_ID	Select-option			Header ID
P_DATUM	DATUM	Parameter	X		Date

Table 7.1 Selection Screen Details for PPF Design

From [Table 7.1](#), we understand that the S_HeaderID is the key, and we need to split the processing into different jobs for ranges of HeaderID, as shown in [Figure 7.10](#).

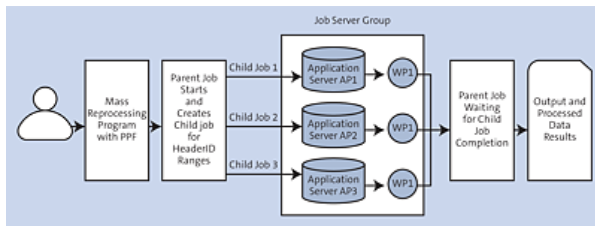


Figure 7.10 Parent and Child Jobs

Basically, we'll design and develop three parts: function module, main program as parent program, and main program as a child job. Let's take a closer look at each.

Function Module

You need to develop a function module, which will be the core of the design that will carry the processing logic. The function module will have almost the same importing parameters as the selection screen criteria, and the function module will be RFC enabled. In this example, it will be the following: company code, accounting principles, header ID range, and date. Additionally, you need to pass the package size (packaging of data will be discussed in [Section 7.1.6](#)), which could be any number per your requirement (generally, the default size is 1000).

This function module will then be called from each child job that the parent program will call later. Let's call this function module ZRAR_REPROCESS_CORE (you can name the function module per your project naming convention). Bear in mind that this is the core which has the entire processing logic, and each of the child

jobs will call this function module with the data range allocated to it from the main program.

Package size on the selection screen will have a default value of 1000, which you may change per your requirement. The package size will be used for grouping the top node, which, in this case, is the HeaderID, into a group that is the size of the package. The data and group are consolidated into the size of the package and passed for processing together. [Figure 7.11](#) shows the function module pseudocode for our example. The pseudocode is a notation resembling a simplified programming language used in program design, just for the purposes of our example.

Let's break down the key elements of this code:

- Each child job will be called with a range of HeaderID (or the top node field), or even multiple single values of HeaderIDs, and each child job will call the function module.
- If the HeaderID is passed as a range, then the selection of HeaderID (or the top node field) from the database will have to be repeated in the function module with the same logic as in the main program but with the limited range provided in the importing parameter of the function module.
- If the HeaderID is passed as multiple single values, then there's no need to select the HeaderID from the database again, as multiple single values indicate they are not a range and have to be considered as is.
- Internal table `gt_headerid_global` will have the entire HeaderID, which needs to be split into the package size for processing.
- `GW_MAX` and `GW_MIN` are the two variables created to keep track of the package size of data to be passed to internal table `gt_headerid_pack` from internal table `gt_headerid_global`.
- `Perform Processing` is called for the package size number of HeaderIDs, and here the logic specific to the requirement will be written.

```

FUNCTION GRAN_PSEUDOCODE_CODE.
41 Perform Initialize:"(1)Initialize global variable. Assign the importing variable to global variable and internal table.
42 IF IS_HEADERID IS NOT INITIAL.
43 READ TABLE IT_HEADERID INTO DATA(is_headerid INDEX 1. "Check if headerid is passed as a range or stack of multiple
44 " single values.
45
46 IF IS_HEADERID-option = 'RQ'."(the sending program or the main program has sent multiple single values of HeaderID
47 "so move it as is to the global internal table for header_id
48 gt_headerid_global[] = IT_HEADERID[]
49 ENDIF.
50 IF IS_HEADERID-option <> 'RQ'."(the sending program has sent range of HeaderID
51 Perform ON_headerid using IT_HEADERID changing GT_HEADERID_GLOBAL.
52 "First some selection logic will be applied for packing the header_id
53 " as in the main program but it will be for a smaller range value
54 " which is sent from the parent program and as available in the ranges )
55 Endif.
56 Endif.
57 Sort GT_HEADERID_GLOBAL[].
58 DESCRIBE TABLE GT_HEADERID_GLOBAL[] LINES ON TOTAL.
59 gw_max = 1. gw_max = p_pack. "Declare GW_MIN type integer and GW_MAX type integer. gw_min = 1 and gw_max = p_pack.
60 " where p_pack is the package size )
61 While gw_min <= gw_max.
62 APPEND LINES OF gt_headerid_global FROM gw_min TO gw_max TO gt_headerid_pack[].
63 "The internal table gt_headerid_pack[] will now have the number
64 " of records equal to package size.)
65 Perform Processing. " In this perform processing we have the core processing logic
66 " and this will be called for each package ).
67 gw_min = gw_max + 1. gw_max = gw_max + p_pack.
68 Endwhile.
69 Endfunction.

```

Figure 7.11 Pseudocode for the Function Module of Custom PPF

Example: Package Size

For example, if the total number of HeaderID is 5,000, and the package size is 1,000, then the following occurs:

1. $GW_MIN = 0$, $GW_MAX = 1000$, and $GW_TOTAL =$ total number of records in `GT_HEADERID_GLOBAL`, while $GW_MIN \leq GW_TOTAL$.
2. Records starting from GW_MIN to GW_MAX are appended to `GT_HEADER_ID_PACK` from `GT_HEADERID_GLOBAL`.
3. `Perform Processing` is called for `GT_HEADER_ID_PACK`.
4. Then, $GW_MIN = GW_MAX + 1$, and $GW_MAX = GW_MAX + PACKAGE$. This gets the next set of records per the package size.
5. `ENDWHILE`.

Main Program as Parent Program

The main program will have two roles: the first as the parent program and the second as the child job. The parent program's design should start with the selection screen where you design the selection screen per the selection criteria and then add the additional parameters on the selection screen that are specific to the custom PPF. In the selection screen, you'll add additional selection parameters. Along with the selection screen elements in [Table 7.1](#), [Table 7.2](#) shows the additional screen elements to be included for the custom PPF.

Selection Element	Type	Description
Schedule Job	Checkbox	Checkbox to indicate scheduling of jobs in the background when checked
Package Size	Parameter	Size of the package default value = 1000
No of Intervals	Parameter	No of child jobs to be scheduled
P_main	Parameter (No display)	Parameter that helps to distinguish between child jobs and main parent processing Default value = X for initial parent processing
HeaderID(S0_HD_ID)	Select-option (range) (No display)	Acts as the range for header ID for child jobs, as the parent program will call the same program when calling the child job but with different parameters

Table 7.2 Selection Screen for Main Program with Additional Parameters for the Custom PPF

After the selection screen design is completed, it will look like [Figure 7.12](#).

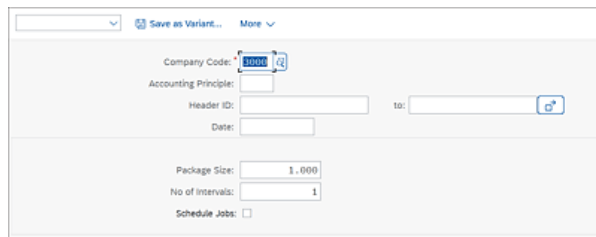


Figure 7.12 Selection Screen for the Example

Let's explain the processing flow of the main program as parent. The select-option `SO_HD_ID` isn't visible on the selection screen as it's made invisible through screen processing (this will be explained later). You can see that parameter `P_MAIN` also isn't visible on the selection screen shown in [Figure 7.12](#). This is like a secret ingredient. `P_MAIN` is made invisible on the selection screen because it's more for us to handle this during processing. It basically has the job of letting the processor know that the current execution of the main program is as the parent program when `P_MAIN = 'X'` and as the child job when `P_MAIN = Space`. So, it's more important to check the value of `P_MAIN` in the first place. The program flow starts with the following check: `If P_MAIN = 'X'`. A positive result means this is the processing for parent.

In the parent processing, it needs to first fetch the data by identifying the top node, which is the `HeaderID` here. You select all the `HeaderIDs` per the selection criteria and load them into an internal table. Then, you check if the **Schedule Jobs** checkbox has been selected. If it has been checked, you need to schedule child jobs. In the subroutine for scheduling the child jobs, you need to calculate the number of child jobs required to schedule. This is done by the **No of Intervals** parameter on the selection screen. This will tell you the number of child jobs the user is expecting to schedule. Each interval will be a child job, and the `HeaderIDs` need to be divided accordingly.

It's very important to clear input parameter `P_MAIN` when you schedule the job by submitting to the same program for the selected range of `HeaderIDs`. If `P_MAIN` isn't cleared, then you can see that the job will be considered as the main program and not as the child program. It's very clear from the following pseudocode that the main program processing starts with the check for `P_MAIN = 'X'`. The pseudocode for the main program as a parent is shown in [Figure 7.13](#).

```

10 IF P_MAIN = 'X', *Indicates that this is parent processing
11 Select the HeaderID as per the selection criteria into table gt_header_id_global[]
12 Endif.
13
14 IF p_main = space or p_interval = 1 or schedule_job = space. *This condition check indicates that the processing
15 * has come via child job of the user wants to run the
16 * program as a single session as P_INTERVAL = 1 and
17 * no child jobs to be created as schedule_job = space
18 RETURN processing_package. *where we have the call to the function module 'ZRAR_REPROCESS_CORE' for processing.
19 Else.
20 PERFORM group_child_jobs. *To schedule the child jobs, we need to calculate how many child jobs need to be
21 * scheduled. This is done by the No Of Intervals parameter on the screen. This will
22 * tell you the number of child jobs the user is expecting to schedule.
23 DESCRIBE TABLE gt_header_id_global LINES lv_total. *get the total number of HeaderIDs selected
24 gv_count = call lv_total / p_interval. *gv_count gives the number of jobs that will be scheduled
25 gv_min = 1 . gv_max = gv_count. *initialize two variables gv_min and gv_max and initialize them as shown.
26
27 WHILE gv_min <= lv_total
28 Read table GT_HEADER_ID_GLOBAL into DATA(lv_min) index gv_min.
29 Read table GT_HEADER_ID_GLOBAL into DATA(lv_max) index gv_max.
30 * Populate the select-option SO_HDID[] which is a hidden select_option on screen with the range of
31 * HeaderID between GV_MIN and GV_MAX and append it, so SO_HDID[] will have the HEADERID or
32 * range of HeaderID for the child job then populate all other selection criteria and meta
33 * data we clear the P_OPEN and prepare the internal table IT_TAB which is the internal table
34 * to be used for submitting to the same program w ith all the selection screen parameters
35 * and select_options and all other selection screen details.
36 * We get the range of HeaderID for the current child job. Schedule the child job.
37
38 Call Function 'JOB_OPEN'.
39 SUBMIT (sy repid) WITH SELECTION-SCREEN it_vari VIA JOB lv_testname
40 * * * * *
41 * * * * *
42 * * * * *
43 Call Function 'JOB_CLOSE'. *We will have to provide the Target Server group which we created
44 * * * * *
45 * * * * *
46 * * * * *
47 * * * * *
48 * * * * *
49 * * * * *
50 Call Function 'JOB_CLOSE'. *We will have to provide the Target Server group which we created
51 * * * * *
52 * * * * *
53 * * * * *
54 * * * * *
55 * * * * *
56 Call Function 'JOB_CLOSE'. *We will have to provide the Target Server group which we created
57 * * * * *
58 * * * * *
59 * * * * *
60 * * * * *
61 * * * * *
62 * * * * *
63 * * * * *
64 * * * * *
65 * * * * *
66 * * * * *
67 * * * * *
68 * * * * *
69 * * * * *
70 * * * * *
71 * * * * *
72 * * * * *
73 * * * * *
74 * * * * *
75 * * * * *
76 * * * * *
77 * * * * *
78 * * * * *
79 * * * * *
80 * * * * *
81 * * * * *
82 * * * * *
83 * * * * *
84 * * * * *
85 * * * * *
86 * * * * *
87 * * * * *
88 * * * * *
89 * * * * *
90 * * * * *
91 * * * * *
92 * * * * *
93 * * * * *
94 * * * * *
95 * * * * *
96 * * * * *
97 * * * * *
98 * * * * *
99 * * * * *
100 * * * * *
101 * * * * *
102 * * * * *
103 * * * * *
104 * * * * *
105 * * * * *
106 * * * * *
107 * * * * *
108 * * * * *
109 * * * * *
110 * * * * *
111 * * * * *
112 * * * * *
113 * * * * *
114 * * * * *
115 * * * * *
116 * * * * *
117 * * * * *
118 * * * * *
119 * * * * *
120 * * * * *
121 * * * * *
122 * * * * *
123 * * * * *
124 * * * * *
125 * * * * *
126 * * * * *
127 * * * * *
128 * * * * *
129 * * * * *
130 * * * * *
131 * * * * *
132 * * * * *
133 * * * * *
134 * * * * *
135 * * * * *
136 * * * * *
137 * * * * *
138 * * * * *
139 * * * * *
140 * * * * *
141 * * * * *
142 * * * * *
143 * * * * *
144 * * * * *
145 * * * * *
146 * * * * *
147 * * * * *
148 * * * * *
149 * * * * *
150 * * * * *
151 * * * * *
152 * * * * *
153 * * * * *
154 * * * * *
155 * * * * *
156 * * * * *
157 * * * * *
158 * * * * *
159 * * * * *
160 * * * * *
161 * * * * *
162 * * * * *
163 * * * * *
164 * * * * *
165 * * * * *
166 * * * * *
167 * * * * *
168 * * * * *
169 * * * * *
170 * * * * *
171 * * * * *
172 * * * * *
173 * * * * *
174 * * * * *
175 * * * * *
176 * * * * *
177 * * * * *
178 * * * * *
179 * * * * *
180 * * * * *
181 * * * * *
182 * * * * *
183 * * * * *
184 * * * * *
185 * * * * *
186 * * * * *
187 * * * * *
188 * * * * *
189 * * * * *
190 * * * * *
191 * * * * *
192 * * * * *
193 * * * * *
194 * * * * *
195 * * * * *
196 * * * * *
197 * * * * *
198 * * * * *
199 * * * * *
200 * * * * *
201 * * * * *
202 * * * * *
203 * * * * *
204 * * * * *
205 * * * * *
206 * * * * *
207 * * * * *
208 * * * * *
209 * * * * *
210 * * * * *
211 * * * * *
212 * * * * *
213 * * * * *
214 * * * * *
215 * * * * *
216 * * * * *
217 * * * * *
218 * * * * *
219 * * * * *
220 * * * * *
221 * * * * *
222 * * * * *
223 * * * * *
224 * * * * *
225 * * * * *
226 * * * * *
227 * * * * *
228 * * * * *
229 * * * * *
230 * * * * *
231 * * * * *
232 * * * * *
233 * * * * *
234 * * * * *
235 * * * * *
236 * * * * *
237 * * * * *
238 * * * * *
239 * * * * *
240 * * * * *
241 * * * * *
242 * * * * *
243 * * * * *
244 * * * * *
245 * * * * *
246 * * * * *
247 * * * * *
248 * * * * *
249 * * * * *
250 * * * * *
251 * * * * *
252 * * * * *
253 * * * * *
254 * * * * *
255 * * * * *
256 * * * * *
257 * * * * *
258 * * * * *
259 * * * * *
260 * * * * *
261 * * * * *
262 * * * * *
263 * * * * *
264 * * * * *
265 * * * * *
266 * * * * *
267 * * * * *
268 * * * * *
269 * * * * *
270 * * * * *
271 * * * * *
272 * * * * *
273 * * * * *
274 * * * * *
275 * * * * *
276 * * * * *
277 * * * * *
278 * * * * *
279 * * * * *
280 * * * * *
281 * * * * *
282 * * * * *
283 * * * * *
284 * * * * *
285 * * * * *
286 * * * * *
287 * * * * *
288 * * * * *
289 * * * * *
290 * * * * *
291 * * * * *
292 * * * * *
293 * * * * *
294 * * * * *
295 * * * * *
296 * * * * *
297 * * * * *
298 * * * * *
299 * * * * *
300 * * * * *
301 * * * * *
302 * * * * *
303 * * * * *
304 * * * * *
305 * * * * *
306 * * * * *
307 * * * * *
308 * * * * *
309 * * * * *
310 * * * * *
311 * * * * *
312 * * * * *
313 * * * * *
314 * * * * *
315 * * * * *
316 * * * * *
317 * * * * *
318 * * * * *
319 * * * * *
320 * * * * *
321 * * * * *
322 * * * * *
323 * * * * *
324 * * * * *
325 * * * * *
326 * * * * *
327 * * * * *
328 * * * * *
329 * * * * *
330 * * * * *
331 * * * * *
332 * * * * *
333 * * * * *
334 * * * * *
335 * * * * *
336 * * * * *
337 * * * * *
338 * * * * *
339 * * * * *
340 * * * * *
341 * * * * *
342 * * * * *
343 * * * * *
344 * * * * *
345 * * * * *
346 * * * * *
347 * * * * *
348 * * * * *
349 * * * * *
350 * * * * *
351 * * * * *
352 * * * * *
353 * * * * *
354 * * * * *
355 * * * * *
356 * * * * *
357 * * * * *
358 * * * * *
359 * * * * *
360 * * * * *
361 * * * * *
362 * * * * *
363 * * * * *
364 * * * * *
365 * * * * *
366 * * * * *
367 * * * * *
368 * * * * *
369 * * * * *
370 * * * * *
371 * * * * *
372 * * * * *
373 * * * * *
374 * * * * *
375 * * * * *
376 * * * * *
377 * * * * *
378 * * * * *
379 * * * * *
380 * * * * *
381 * * * * *
382 * * * * *
383 * * * * *
384 * * * * *
385 * * * * *
386 * * * * *
387 * * * * *
388 * * * * *
389 * * * * *
390 * * * * *
391 * * * * *
392 * * * * *
393 * * * * *
394 * * * * *
395 * * * * *
396 * * * * *
397 * * * * *
398 * * * * *
399 * * * * *
400 * * * * *
401 * * * * *
402 * * * * *
403 * * * * *
404 * * * * *
405 * * * * *
406 * * * * *
407 * * * * *
408 * * * * *
409 * * * * *
410 * * * * *
411 * * * * *
412 * * * * *
413 * * * * *
414 * * * * *
415 * * * * *
416 * * * * *
417 * * * * *
418 * * * * *
419 * * * * *
420 * * * * *
421 * * * * *
422 * * * * *
423 * * * * *
424 * * * * *
425 * * * * *
426 * * * * *
427 * * * * *
428 * * * * *
429 * * * * *
430 * * * * *
431 * * * * *
432 * * * * *
433 * * * * *
434 * * * * *
435 * * * * *
436 * * * * *
437 * * * * *
438 * * * * *
439 * * * * *
440 * * * * *
441 * * * * *
442 * * * * *
443 * * * * *
444 * * * * *
445 * * * * *
446 * * * * *
447 * * * * *
448 * * * * *
449 * * * * *
450 * * * * *
451 * * * * *
452 * * * * *
453 * * * * *
454 * * * * *
455 * * * * *
456 * * * * *
457 * * * * *
458 * * * * *
459 * * * * *
460 * * * * *
461 * * * * *
462 * * * * *
463 * * * * *
464 * * * * *
465 * * * * *
466 * * * * *
467 * * * * *
468 * * * * *
469 * * * * *
470 * * * * *
471 * * * * *
472 * * * * *
473 * * * * *
474 * * * * *
475 * * * * *
476 * * * * *
477 * * * * *
478 * * * * *
479 * * * * *
480 * * * * *
481 * * * * *
482 * * * * *
483 * * * * *
484 * * * * *
485 * * * * *
486 * * * * *
487 * * * * *
488 * * * * *
489 * * * * *
490 * * * * *
491 * * * * *
492 * * * * *
493 * * * * *
494 * * * * *
495 * * * * *
496 * * * * *
497 * * * * *
498 * * * * *
499 * * * * *
500 * * * * *
501 * * * * *
502 * * * * *
503 * * * * *
504 * * * * *
505 * * * * *
506 * * * * *
507 * * * * *
508 * * * * *
509 * * * * *
510 * * * * *
511 * * * * *
512 * * * * *
513 * * * * *
514 * * * * *
515 * * * * *
516 * * * * *
517 * * * * *
518 * * * * *
519 * * * * *
520 * * * * *
521 * * * * *
522 * * * * *
523 * * * * *
524 * * * * *
525 * * * * *
526 * * * * *
527 * * * * *
528 * * * * *
529 * * * * *
530 * * * * *
531 * * * * *
532 * * * * *
533 * * * * *
534 * * * * *
535 * * * * *
536 * * * * *
537 * * * * *
538 * * * * *
539 * * * * *
540 * * * * *
541 * * * * *
542 * * * * *
543 * * * * *
544 * * * * *
545 * * * * *
546 * * * * *
547 * * * * *
548 * * * * *
549 * * * * *
550 * * * * *
551 * * * * *
552 * * * * *
553 * * * * *
554 * * * * *
555 * * * * *
556 * * * * *
557 * * * * *
558 * * * * *
559 * * * * *
560 * * * * *
561 * * * * *
562 * * * * *
563 * * * * *
564 * * * * *
565 * * * * *
566 * * * * *
567 * * * * *
568 * * * * *
569 * * * * *
570 * * * * *
571 * * * * *
572 * * * * *
573 * * * * *
574 * * * * *
575 * * * * *
576 * * * * *
577 * * * * *
578 * * * * *
579 * * * * *
580 * * * * *
581 * * * * *
582 * * * * *
583 * * * * *
584 * * * * *
585 * * * * *
586 * * * * *
587 * * * * *
588 * * * * *
589 * * * * *
590 * * * * *
591 * * * * *
592 * * * * *
593 * * * * *
594 * * * * *
595 * * * * *
596 * * * * *
597 * * * * *
598 * * * * *
599 * * * * *
600 * * * * *
601 * * * * *
602 * * * * *
603 * * * * *
604 * * * * *
605 * * * * *
606 * * * * *
607 * * * * *
608 * * * * *
609 * * * * *
610 * * * * *
611 * * * * *
612 * * * * *
613 * * * * *
614 * * * * *
615 * * * * *
616 * * * * *
617 * * * * *
618 * * * * *
619 * * * * *
620 * * * * *
621 * * * * *
622 * * * * *
623 * * * * *
624 * * * * *
625 * * * * *
626 * * * * *
627 * * * * *
628 * * * * *
629 * * * * *
630 * * * * *
631 * * * * *
632 * * * * *
633 * * * * *
634 * * * * *
635 * * * * *
636 * * * * *
637 * * * * *
638 * * * * *
639 * * * * *
640 * * * * *
641 * * * * *
642 * * * * *
643 * * * * *
644 * * * * *
645 * * * * *
646 * * * * *
647 * * * * *
648 * * * * *
649 * * * * *
650 * * * * *
651 * * * * *
652 * * * * *
653 * * * * *
654 * * * * *
655 * * * * *
656 * * * * *
657 * * * * *
658 * * * * *
659 * * * * *
660 * * * * *
661 * * * * *
662 * * * * *
663 * * * * *
664 * * * * *
665 * * * * *
666 * * * * *
667 * * * * *
668 * * * * *
669 * * * * *
670 * * * * *
671 * * * * *
672 * * * * *
673 * * * * *
674 * * * * *
675 * * * * *
676 * * * * *
677 * * * * *
678 * * * * *
679 * * * * *
680 * * * * *
681 * * * * *
682 * * * * *
683 * * * * *
684 * * * * *
685 * * * * *
686 * * * * *
687 * * * * *
688 * * * * *
689 * * * * *
690 * * * * *
691 * * * * *
692 * * * * *
693 * * * * *
694 * * * * *
695 * * * * *
696 * * * * *
697 * * * * *
698 * * * * *
699 * * * * *
700 * * * * *
701 * * * * *
702 * * * * *
703 * * * * *
704 * * * * *
705 * * * * *
706 * * * * *
707 * * * * *
708 * * * * *
709 * * * * *
710 * * * * *
711 * * * * *
712 * * * * *
713 * * * * *
714 * * * * *
715 * * * * *
716 * * * * *
717 * * * * *
718 * * * * *
719 * * * * *
720 * * * * *
721 * * * * *
722 * * * * *
723 * * * * *
724 * * * * *
725 * * * * *
726 * * * * *
727 * * * * *
728 * * * * *
729 * * * * *
730 * * * * *
731 * * * * *
732 * * * * *
733 * * * * *
734 * * * * *
735 * * * * *
736 * * * * *
737 * * * * *
738 * * * * *
739 * * * * *
740 * * * * *
741 * * * * *
742 * * * * *
743 * * * * *
744 * * * * *
745 * * * * *
746 * * * * *
747 * * * * *
748 * * * * *
749 * * * * *
750 * * * * *
751 * * * * *
752 * * * * *
753 * * * * *
754 * * * * *
755 * * * * *
756 * * * * *
757 * * * * *
758 * * * * *
759 * * * * *
760 * * * * *
761 * * * * *
762 * * * * *
763 * * * * *
764 * * * * *
765 * * * * *
766 * * * * *
767 * * * * *
768 * * * * *
769 * * * * *
770 * * * * *
771 * * * * *
772 * * * * *
773 * * * * *
774 * * * * *
775 * * * * *
776 * * * * *
777 * * * * *
778 * * * * *
779 * * * * *
780 * * * * *
781 * * * * *
782 * * * * *
783 * * * * *
784 * * * * *
785 * * * * *
786 * * * * *
787 * * * * *
788 * * * * *
789 * * * * *
790 * * * * *
791 * * * * *
792 * * * * *
793 * * * * *
794 * * * * *
795 * * * * *
796 * * * * *
797 * * * * *
798 * * * * *
799 * * * * *
800 * * * * *
801 * * * * *
802 * * * * *
803 * * * * *
804 * * * * *
805 * * * * *
806 * * * * *
807 * * * * *
808 * * * * *
809 * * * * *
810 * * * * *
811 * * * * *
812 * * * * *
813 * * * * *
814 * * * * *
815 * * * * *
816 * * * * *
817 * * * * *
818 * * * * *
819 * * * * *
820 * * * * *
821 * * * * *
822 * * * * *
823 * * * * *
824 * * * * *
825 * * * * *
826 * * * * *
827 * * * * *
828 * * * * *
829 * * * * *
830 * * * * *
831 * * * * *
832 * * * * *
833 * * * * *
834 * * * * *
835 * * * * *
836 * * * * *
837 * * * * *
838 * * * * *
839 * * * * *
840 * * * * *
841 * * * * *
842 * * * * *
843 * * * * *
844 * * * * *
845 * * * * *
846 * * * * *
847 * * * * *
848 * * * * *
849 * * * * *
850 * * * * *
851 * * * * *
852 * * * * *
853 * * * * *
854 * * * * *
855 * * * * *
856 * * * * *
857 * * * * *
858 * * * * *
859 * * * * *
860 * * * * *
861 * * * * *
862 * * * * *
863 * * * * *
864 * * * * *
865 * * * * *
866 * * * * *
867 * * * * *
868 * * * * *
869 * * * * *
870 * * * * *
871 * * * * *
872 * * * * *
873 * * * * *
874 * * * * *
875 * * * * *
876 * * * * *
877 * * * * *
878 * * * * *
879 * * * * *
880 * * * * *
881 * * * * *
882 * * * * *
883 * * * * *
884 * * * * *
885 * * * * *
886 * * * * *
887 * * * * *
888 * * * * *
889 * * * * *
890 * * * * *
891 * * * * *
892 * * * * *
893 * * * * *
894 * * * * *
895 * * * * *
896 * * * * *
897 * * * * *
898 * * * * *
899 * * * * *
900 * * * * *
901 * * * * *
902 * * * * *
903 * * * * *
904 * * * * *
905 * * * * *
906 * * * * *
907 * * * * *
908 * * * * *
909 * * * * *
910 * * * * *
911 * * * * *
912 * * * * *
913 * * * * *
914 * * * * *
915 * * * * *
916 * * * * *
917 * * * * *
918 * * * * *
919 * * * * *
920 * * * * *
921 * * * * *
922 * * * * *
923 * * * * *
924 * * * * *
925 * * * * *
926 * * * * *
927 * * * * *
928 * * * * *
929 * * * * *
930 * * * * *
931 * * * * *
932 * * * * *
933 * * * * *
934 * * * * *
935 * * * * *
936 * * * * *
937 * * * * *
938 * * * * *
939 * * * * *
940 * * * * *
941 * * * * *
942 * * * * *
943 * * * * *
944 * * * * *
945 * * * * *
946 * * * * *
947 * * * * *
948 * * * * *
949 * * * * *
950 * * * * *
951 * * * * *
952 * * * * *
953 * * * * *
954 * * * * *
955 * * * * *
956 * * * * *
957 * * * * *
958 * * * * *
959 * * * * *
960 * * * * *
961 * * * * *
962 * * * * *
963 * * * * *
964 * * * * *
965 * * * * *
966 * * * * *
967 * * * * *
968 * * * * *
969 * * * * *
970 * * * * *
971 * * * * *
972 * * * * *
973 * * * * *
974 * * * * *
975 * * * * *
976 * * * * *
977 * * * * *
978 * * * * *
979 * * * * *
980 * * * * *
981 * * * * *
982 * * * * *
983 * * * * *
984 * * * * *
985 * * * * *
986 * * * * *
987 * * * * *
988 * * * * *
989 * * * * *
990 * * * * *
991 * * * * *
992 * * * * *
993 * * * * *
994 * * * * *
995 * * * * *
996 * * * * *
997 * * * * *
998 * * * * *
999 * * * * *
1000 * * * * *

```

Figure 7.13 Pseudocode for the Main Program of the Custom PPF

Note

You can find PDF versions of [Figure 7.11](#) and [Figure 7.13](#) available for download at www.sap-press.com/5700 under the **Product supplements** section.

Main Program as Child Job

The same program as a child job will be called from the SUBMIT statement in the parent program. Each call to SUBMIT creates a new child job, and you already know how the calculation for the number of child jobs is done.

Next, you need to be concerned with the difference in execution flow of the same program as a child job when called through the SUBMIT statement from the parent program. The important changes are in the selection screen elements, as follows:

- **Parameter P_MAIN = SPACE**
Note P_MAIN needs to be cleared for the main program to be executed as the child job; otherwise, it will be considered as a parent program when P_MAIN = 'X'.
- **Select-option so_HDID[]**
The select-option so_HDID[] is an invisible data carrier, as it's hidden on the selection screen. so_HDID[] will be populated with the HeaderID range that is allocated to the child job from the main parent program.

The pseudocode for the program as child job is shown in [Listing 7.1](#).

Check if P_MAIN = SPACE, if yes It indicates it is a child job
 Check if SO_HDID[] is not initial .
 Move the HeaderID from SO_HDID to GT_HEADER_ID_GLOBAL[] internal table
 Call the function module 'ZRAR_REPROCESS_CORE'.
 From there the flow continues as explained above in the processing of the function module 'ZRAR_REPROCESS_CORE'.

Listing 7.1 Pseudocode for Child Job of the Custom PPF

Program Execution

Now we're all set with the code part and understand the concept of custom parallel processing. Certainly, many developers design their own custom PPFs as a result of individual preference and creativity, so it's good to keep in mind that there will be a lot of other ways to go about this process. For our example, we've showcased what worked best in our experience. The important thing to note is that this solution isn't limited to RAR, and it could be extended to any other modules. It doesn't depend on KEYPP or any other module-specific parameters but is more generic and extendable to any other modules.

Let's look at the program execution and the results that you can expect. We schedule the main parent program with the required input details, as shown in [Table 7.3](#).

Selection Element	Selection Type	Default Values
P_BUKRS	Parameter	1000
P_PRIN	Parameter	IFRS
S_HeaderID		Initial input range
P_DATUM	Parameter	Date of selection
P_JOBS	Parameter	X
P_PACKAGE_SIZE	Parameter	1000
P_INTERVALS	Parameter	20
P_MAIN	Parameter	X (this is set by default and isn't available on the selection screen)
SO_HDID	Select-option	Blank

Table 7.3 Input to the Selection Screen of Custom PPF

The program will be scheduled to run in the background, and you can see the details of the job in Transaction SM36. There will be one parent program and 20 scheduled child jobs as we set the interval to 20, which determines the number of child jobs. You can see that the parent job waits until all the child jobs are completed, and then it's completed. With this, we complete the custom PPF design, implementation details, and testing details.

7.1.5 Modularization

If you're someone who is into cooking, then you've experienced that it's easier to cook when things are arranged, organized, and modularized. *Modularization* refers to things of the same category placed or grouped together. It's also a practice to label containers or store ingredients in transparent containers. That's the case with code as well—we need to design modular and organized code that also has self-explanatory variables and names of functions and methods. The code should be readable and communicate what is being done in each module. Writing a big program must be planned by breaking it into smaller units or processing blocks. Then each unit put together becomes the complete program that we're trying to build. It will add more clarity if the modular code blocks or units are named per what is being done inside them or what the purpose of each unit is. Further, we could also have meaningful names for the variables, constants, and flags, so that we can know what is being stored in them or what they are being used for (like the transparent or labeled containers).

Another important thing about modularization is there will be a drastic reduction in redundant code. We could create a block of code that needs to be written at multiple places and then just call the same block at all those places by changing the necessary parameters; in addition, we can also call that block where we need that functionality instead of having to write it multiple times.

All of this shows that modularization leads to clarity and reusability. Following are the benefits of using modularizations:

- The code becomes more readable.
- It's easier to maintain and change a modular code.
- You can avoid redundant code.
- Code is reusable.
- Data can be encapsulated.
- Code becomes structured.

There are multiple modularization techniques. In ABAP, the modular pieces are called processing blocks. There are two kinds of processing blocks: those that are called from outside a program by the ABAP runtime system (event blocks and dialog modules), and those that can be called by ABAP statements in ABAP programs (subroutines, function modules, methods).

With all this in mind, you now know you can modularize your code either by the procedural approach or by defining a class and its method. For our example, we'll follow the approach shown in [Figure 7.14](#). We recommend this approach to keep the declarations and processing separate and to make locating the code easy.

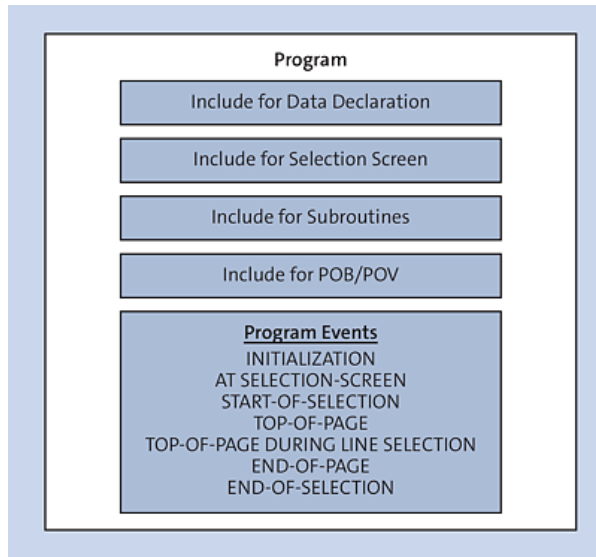


Figure 7.14 Modularization in ABAP Programs

As shown in [Figure 7.15](#), our program will be designed with includes. Each include is used for a specific purpose, as follows:

- **Include for data declaration**

This include will have all the declarations for the internal table types, structure types, and all the type declarations. The data declarations, such for as the internal tables, work areas, global data and global internal tables, and constants, will also be defined and declared.

- **Include for selection screen**

This include will have the selection screen details such as the parameters, select-options, and other screen elements, along with their default values if needed, and other screen attributes will be defined here.

- **Include for subroutines**

All the forms will be implemented here.

- **Include for PBO/PAI/POV**

If the object is a module pool object, then the process before output (PBO), process after input (PAI), and process on value request (POV) modules will be implemented here.

- **Include for classes**

This include is for the local class if the program has one.

```

-----*
TOP Include
-----*
INCLUDE /ora_reprocess_top IF FOUND. " include for data declaration
-----*
Include For Selection screen
-----*
INCLUDE /ora_reprocess_scr IF FOUND. " Include for selection screen
-----*
Include For Classes
-----*
INCLUDE /ora_reprocess_cls IF FOUND. " Include for classes
-----*
Include For Subroutine
-----*
INCLUDE /ora_reprocess_f01 IF FOUND. " Include for subroutine
-----*
INITIALIZATION
-----*

```

Figure 7.15 Modularization

Performs are modularizing blocks of code. The name of the perform should actually convey what is being done in the block in short, and what it requires for processing will also be passed to the perform as parameters such as `using`, `changing`, and `returning` parameters. To make the code very legible and understandable, you should use self-explanatory names for the performs and pass the parameters with more meaningful names regarding what is being used in the `using` parameters and what is being changed in the `changing` parameters. Following is an example of how the perform should be structured:

"Perform select_pob using lt_contracts changing gt_pobs."

This snippet will explain that the perform is selecting the POBs using the contracts from internal table `lt_contract` and changing internal table `gt_pob`. Not just the performs, even the variable names, should be self-explanatory. Let's take an example where we're performing some validations and setting the `FINAL_INVOICE` field in table `FARR_D_POB` as X. The check for final is where we try to check if the POB is final or not final based on the invoices. In the code, if we're using a flag, then it would be more meaningful to call that flag `GW_POB_FINAL` instead of just `GW_FLAG` because the variable will inform the person who is trying to debug or understand the code that this flag is for POB and to check if it's final.

7.1.6 Packaging

Packaging is the bundling of data records to be processed together or commit together to enhance the performance. We designed our custom PPF in [Section 7.1.1](#) to improve performance. There is mention of package size in that section, and we've also defined the same on the selection screen as **Package Size** (refer to [Figure 7.12](#)). Don't confuse this with the `PACKAGE SIZE` in ABAP `SELECT` statements.

Let's revisit our custom PPF to demonstrate how packaging has been implemented in our example. In [Figure 7.16](#), you can see a program that needs to process 2,000 records.

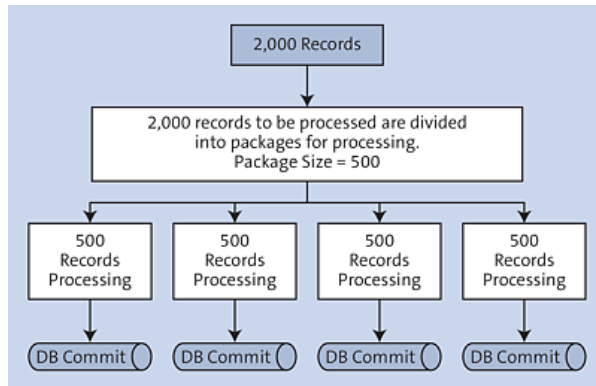


Figure 7.16 Package and Commit

That program has a selection parameter for package size, and the user has provided the value as 500. Therefore, the program starts by selecting the 2,000 entries and then moves the package size number of records, that is, 500 records, into another temporary internal table and does all the processing except for database commits. The complete processing and updating of the records (i.e., internal table processing) is done for all the package size number of records, and then finally the commit is done at the end of every package processing and not for every record. This helps improve performance, as the commit takes time, and instead of calling the commit for each record, we do it at the end of the package at once for all the records in the package.

You can bring this packaging logic to any program and any internal table processing, which will help speed up any slow-running program.

Packaging has many advantages. First, it helps optimize the performance of any given program. Second, if there is a memory issue, even that can be handled by packaging. For example, if we're selecting data from a huge table for a huge number of records, there may be memory allocation issues. If you incorporate packaging logic, and instead of selecting data for the entire set of records, you select for the package number of records, then the memory allocation issue can be handled with no runtime dumps. This is called setting the package externally.

7.2 Reclassification

As you already know, RAR is about revenue recognition. Table `FARR_D_POSTING` is crucial here, and it acts like a subledger before posting to the general ledger. Table `FARR_D_POSTING` is populated after programs A and B are executed, and then we use the entries from table `FARR_D_POSTING` to run program C, which is the posting program. For more information on the ABC programs, see [Chapter 6, Section 6.2](#).

In the context of RAR, *reclassification* is done to ensure posting is made to the correct general ledger. After the posting program (program C) has been executed, the posting has already been done to a particular general ledger, but there could be situations where you need to make corrections to that and post them to different accounts. To correct this, you need to create a custom posting program that will post entries from custom table `ZFARR_D_POSTING`.

Why do we need custom table `ZFARR_D_POSTING`, and how do we perform the posting from this custom table? To answer this question, we'll provide some situations where creating custom table `ZFARR_D_POSTING` is a necessity. Let's rename the custom table as table `ZFARR_D_POSTING`. Following are cases of requirements in the project team that might require creating custom table `ZFARR_D_POSTING` (these are just a few cases; there could be many more based on your business requirements):

- The payer isn't known during contract inception; there could be a dummy payer created as a placeholder for the payer. So, the revenue and balance sheet may be posted to the wrong accounts at contract inception.
- The payer changes several times during the lifecycle of the contract. As the balance sheet accounts are determined at contract inception based on the payer, RAR doesn't support changing the balance sheet accounts after the contract/POB is created.
- Multiple payers are involved, due to which creating a separate contract and terminating POBs every time a payer changes for each charge would result in an unmanageable number of current and obsolete contracts.
- In exceptional situations, a bundle charge may be split between several payers. Bundle POBs cannot be split among several payers due to the residual approach used for the revenue allocation within the bundle.

In standard SAP, the following shortcomings required developing custom table `ZFARR_D_POSTING`:

- RAR doesn't support intercompany contracts (contracts where the revenue allocation is performed among POBs that are fulfilled by various company codes).
- Partial bills of lading created for the same POB could have various payers.

In the following sections, we'll walk through the reclassification process. We'll create the structure of custom table `ZFARR_D_POSTING` and then explain how to populate and extend it.

7.2.1 Creation and Structure of `ZFARR_D` Tables

To begin our discussion of creating our custom ZFARR_D tables, it's important to provide a refresher of reconciliation keys. Then, we'll discuss custom table ZFARR_D_POSTING, which is called table ZFARR_D_POSTING, and then the custom reconciliation key table called table ZFARR_D_RECONKEY.

All the actions that have accountability effects on a contract in any period of the given year are tracked under a given reconciliation key. There is a reconciliation key created for a contract for every period based on the activity, and there can be multiple reconciliation keys for a contract for a single period. The reconciliation key is also called the *ReconKey*.

The basic structure of the reconciliation key is that the first four digits represent the fiscal year, the next three digits represent the posting period, plus a seven-digit number, as you can see in [Table 7.4](#).

Fiscal Year: First Four Digits	Posting Period: Three Digits	Seven-Digit Number
2024	005	0000001

Table 7.4 Reconciliation Key

We explained how to access reconciliation keys through table FARR_D_RECON_KEY and the different statuses in [Chapter 6, Section 6.1.3](#). For the purposes of our discussion in this chapter, the following are key points to keep in mind:

- The reconciliation key is governed by the revenue account period being open or closed. A reconciliation key for a contract will be created by different events for a given period only if the revenue accounting period is open for that period. If the period is closed, then a new reconciliation key will be created for the next period.
- During the month-end closing, the status of the reconciliation key for the contracts should be in status C for all the reconciliation keys in that period for all contracts.
- Reconciliation key is one of the key fields in table FARR_D_POSTING and will be used during program C, which is the posting program.
- Reconciliation key is also part of the key fields in table FARR_D_DEFITEM (the revenue forecast table).

The reconciliation key is created by a few methods of class CL_FARR_RECON_KEY_DB_ACCESS, as shown in [Figure 7.17](#). There are multiple methods in the class, and some are based on how you need to create the reconciliation key: single or multiple.

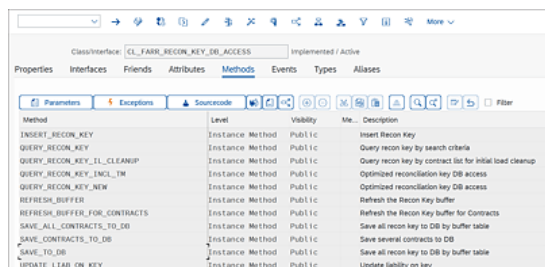


Figure 7.17 Class for Reconciliation Key-Related Methods

Now let's shift our focus to custom table ZFARR_D_POSTING that we designed for our requirement. Note that we aren't highlighting the functional requirement that led to the creation of custom table ZFARR_D_POSTING; instead, we'll elaborate on the technical approach and how it can be achieved for any functional requirements.

Let's walk through the main technical steps:

1. Create table ZFARR_D_POSTING

Table FARR_D_POSTING will be copied as is to table ZFARR_D_POSTING. You can add any additional fields that you want to add. This table will have all the fields in table FARR_D_POSTING, including the ones added during customization, such as the customer fields and the field added to store the custom RECONKEY or ZRECONKEY that will be created. There will be an indicator to check if you need to reverse or reclass entries.

To create table ZFARR_D_POSTING, go to Transaction SE11 as shown in [Figure 7.18](#). Provide the **Database table** name as "FARR_D_POSTING", and then click on **Copy** to open the **Copy Table** popup. Enter the **Table** name as "ZFARR_D_POSTING", and click the green checkmark.

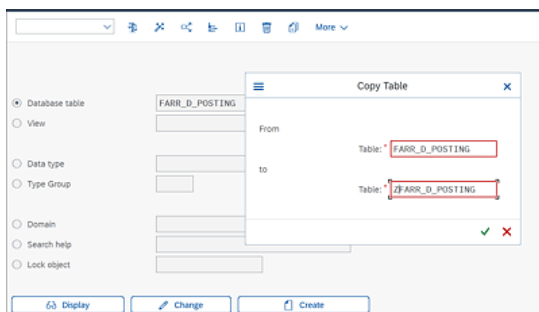


Figure 7.18 Creating Table ZFARR_D_POSTING

Then, as shown in [Figure 7.19](#), add the additional fields such as ZRECON_KEY, which is the custom reconciliation key to be stored in custom table ZFARR_D_RECONKEY, and RECLASSIND, which is the indicator for reclass to reverse or reclass with the values R or C, respectively.

11	RECLASSIND	11	X	RECLASSIND	CHAR01	CHAR	1
12	ZRECON_KEY	12	X	FARR_RECON_KEY	FARR_RECON_KEY	CHAR	14

Figure 7.19 Additional Custom Fields in Table ZFARR_D_POSTING

Then, complete all the steps for the table creation and activate it. For more information on adding fields and creating the table, see [Chapter 2, Section 2.3](#).

2. Create table ZFARR_D_RECONKEY

As in the case of standard posting, the flow reconciliation key is created and updated in table FARR_D_RECON_KEY. You have to take the same approach and create a new reconciliation key, which again is a custom reconciliation key and isn't connected to the original reconciliation key. To save that, you have to create a custom table for the custom reconciliation key, which, in this case, will be called ZFARR_D_RECONKEY, and that will have the same structure as FARR_D_RECON_KEY. This table needs to be updated every time you update or create entries in table ZFARR_D_POSTING.

To create table ZFARR_D_RECONKEY, go to Transaction SE11 once again. Provide the table name as "FARR_RECON_KEY", and click on **Copy** to open the **Copy Table** popup, as

shown in [Figure 7.20](#). Enter the **Table** name as “ZFARR_D_RECONKEY”.

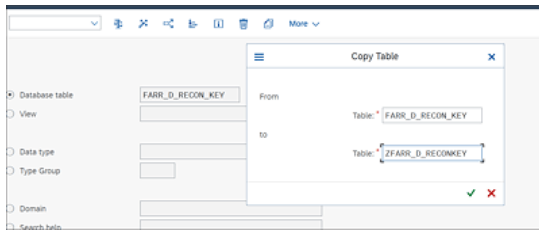


Figure 7.20 Creating Table ZFARR_D_RECONKEY

Then, complete all the steps for table creation and activate it.

3. Create the general ledger mapping table

There will be a custom table where the account mapping has to be maintained. Let’s call that table ZFARR_GL_MAPPING. This table is needed to maintain the mapping so that you can map to which account the postings are going to be made. This general ledger mapping table will have the source account from table FARR_D_POSTING, and this account will be mapped in the general ledger table to get the target account to which the correction postings will be done. As you’re familiar with the steps required for table creation using Transaction SE11, let’s focus on the table structure in [Figure 7.21](#).

Row	Field name	Position	Key	Data element	Domain	Datatype	Length	Lowercase	Domain text
1	CLASIF	1	X	NUM2	NUM2	NUM2	4		Classif
2	COMPANY_CODE	2	X	BUSID3	BUSID3	COLA	4		Company Code
3	ACCT_POSTING	3	X	ACCTPOSTM_BALANCE	ACCTPOSTM_BALANCE	COLA	4		Accounting Principle
4	POST_CAT	4	X	SPARM_POST_CATEGORY	SPARM_POST_CATEGORY	COLA	2		Category for Posting Document
5	BALANCE	5	X	NUM8	NUM8	COLA	11		GL Account Number
6	GLACCOUNT	6	X	NUM8	NUM8	COLA	11		GL Account Number

Figure 7.21 General Ledger Mapping Table Structure

4. Design the program with the logic per the requirement to create entries in new table ZFARR_D_POSTING

The new program will use the custom PPF or have a design per the requirement. The program should take care of the points in [Section 7.2.2](#) to make sure it’s in sync with the standard approach because you’ll be copying and extending the standard posting program to post the entries from custom table ZFARR_D_POSTING. This will be elaborated on in the next section.

You might wonder about identifying the entries to be reclassified. In our project, we implemented the BAdI FARR_BADI_RAI2, and the ENRICH method was implemented with the logic to identify the order RAIs that had to be reclassified based on certain custom field values. The order RAIs were saved in a custom table. This custom table must be used in the program to identify the POBs that were created by these order RAIs and then the contracts created by them.

5. Make a copy of the standard posting program and then make changes

The standard program fetches entries from subledger table FARR_D_POSTING for creating posting or for the posting program. It also fetches the reconciliation keys from database table FARR_D_RECON_KEY for updating reconciliation keys. As we’re cloning the standard program and making changes for it to fetch entries from custom table ZFARR_D_POSTING and custom table ZFARR_D_RECONKEY for reconciliation keys, we should ensure to make relevant changes to call these custom tables in the place of the standard table. These changes should be focused on picking entries from new custom reconciliation key table ZFARR_D_RECONKEY and custom posting table ZFARR_D_POSTING.

7.2.2 Populating the Custom Posting Table

In this section, we'll focus on populating tables ZFARR_D_RECONKEY and ZFARR_D_POSTING in the program designed to update them (i.e., step 4 in the previous section). The program will have selection criteria per the requirement to pull all the necessary contracts. Further database selections will depend on the selected set of contracts. Let's walk through the technical steps:

1. The program will have to select the relevant contracts from database table FARR_D_CONTRACTS based on the selection contract range provided on the selection screen.
2. Filter the selected contract per the requirement and populate it into internal table GT_CONTRACTS_GLOBAL. In our case, these contracts will be identified as relevant for reclassification by matching the POBs of the contracts with the order RAIs, and the POBs stored in the custom table as marked for reclassification. Every project can have specific requirements for filtering the contracts.
3. For all the contracts in internal table GT_CONTRACTS_GLOBAL, pull the table FARR_D_POSTING entries and store them in internal table GT_FARR_D_POSTING.
4. Pull table ZFARR_D_POSTING entries matching the selection set of contracts.
5. Get the general ledger account details from table ZFARR_GL_MAPPING, which will have the source account that will be matched from table FARR_D_POSTING's account and will have the target account to which you'll have to do the correction posting.
6. Start your loop on the contracts. For each contract, get the maximum number of the custom reconciliation key stored in table ZFARR_D_RECONKEY, and to that reconciliation key, add 1 so that you get the next reconciliation key to be used and updated in table ZFARR_D_RECONKEY for this execution.
7. To get the maximum of ZRECONKEY, first make a copy of class CL_FARR_RECON_KEY_DB_ACC to ZCL_FARR_RECON_KEY_DB_ACC. In that class, use method GET_MAX_RECON_KEY_FOR_MULTIPLE, and change from standard table FARR_D_RECONKEY to custom table ZFARR_D_RECONKEY. (The same class has a lot of other methods for updating, creating, deleting, and saving the RECONKEY. You need to analyze and change the table to the custom table for RECONKEY in the copied class's methods: FARR_D_RECON_KEY to ZFARR_D_RECONKEY.)
8. Have an inner loop on the POBs of the contracts. For each POB, get table FARR_D_POSTING entries and process each of the entries in table FARR_D_POSTING for the POB and contract in the current loop. Then, validate per the project-specific requirements and the kind of correction posting requirements.
In our project, we had the requirement where we had to create a reversal and repost to a different account for a particular table FARR_D_POSTING entry. This was captured as entries and updated in the internal table to populate table ZFARR_D_POSTING.
9. The loop continues to validate the similar entries in custom table ZFARR_D_POSTING. Accordingly, create the posting entries for reversing the existing entry and creating a new posting entry.
10. The entries will then be inserted or updated in custom table ZFARR_D_POSTING.
11. The new custom reconciliation key will be updated in table ZFARR_D_RECONKEY.

These points highlight the general steps that will be required, which can also be visualized in the flow diagram of the program shown in [Figure 7.22](#).

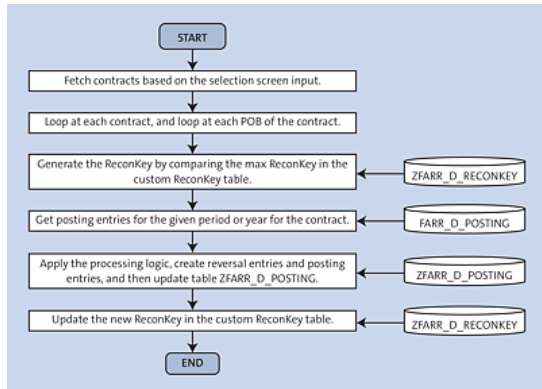


Figure 7.22 Flow Diagram for the Reclassification Program

The validation and insertion/updating of entries in table ZFARR_D_POSTING will be based on the project requirement. Generally, this table will be an additional subledger.

[Figure 7.23](#) shows how the custom posting table ZFARR_D_POSTING fits into the process of the posting flow.

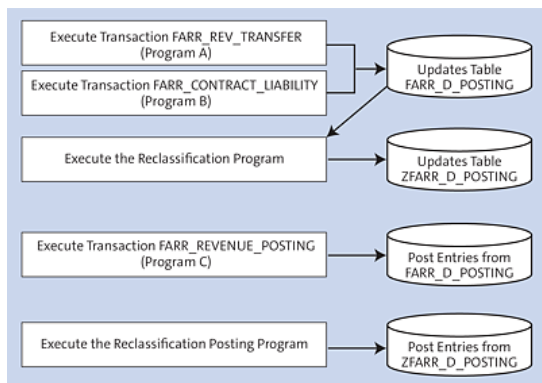


Figure 7.23 Posting Flow with the Custom Posting Table

7.2.3 Extending the Standard Posting Program

The existing posting program FARR_REV_POST will be called to create postings using the entries from table FARR_D_POSTING and to update the RECONKEY in standard table FARR_D_RECONKEY.

Now to extend the posting program to post entries from custom table ZFARR_D_POSTING table and the ZFARR_D_RECONKEY tables will require creating a copy of the FARR_REVENUE_POSTING program to ZFARR_REVENUE_POSTING. You need to change the table references to custom tables, which will require changes in multiple places. As it's very difficult to explain the whole of it here, we'll highlight the objects that had to change and the classes that are created by copying the original. You can analyze the code and change the table to ZFARR_D_POSTING and ZFARR_D_RECONKEY wherever the standard tables FARR_D_POSTING and FARR_D_RECONKEY are found.

[Table 7.5](#) provides the list of objects that we made custom copies of from the standard and the methods changed in the copied classes to reference custom tables ZFARR_D_POSTING and ZFARR_D_RECONKEY to be called in the custom ZFARR_REVENUE_POSTING program.

Standard Class/ Program	To Be Copied to Custom	Methods in Custom to Be Changed	Description
CL_FARR_AC_DOCUMENT_HAND	ZCL_FARR_AC_DOCUMENT_HAND	DO_HANDLE_POST	Any changes to the document type for posting can be done here.
CL_FARR_DB_UPDATE	ZCL_FARR_DB_UPDATE	UPDATE_SET_WHERE	Bypass validation or only allow table names starting with /1RA*.
CL_FARR_GRANULAR_POSTING	ZCL_FARR_GRANULAR_POSTING	POSTING_CHECK_IN_KEYPP	Rename the cursor object to the custom table cursor object.
CL_FARR_POSTING_FLOW	ZCL_FARR_POSTING_FLOW	CONSTRUCTOR	Rename the RECON KEY management object to match the custom object.
CL_FARR_POSTING_FLOW	ZCL_FARR_POSTING_FLOW	HANDLE_POSTING_ENTRIES	Rename the POSTING DOCUMENT handler object to the custom object.
CL_FARR_RECONKEY_CURSOR_ACCESS	ZCL_FARR_RECONKEY_CURSOR_ACCESS	READ_CONTRACT_ID_BUKRS_PACKAGE	Read the reconciliation key from table ZFARR_D_RECON_KEY.
CL_FARR_RECON_KEY_DB_ACCESS	ZCL_FARR_RECON_KEY_DB_ACCESS	SAVE_ALL_CONTRACTS_TO_DB	Change the table to update to ZFARR_D_RECON_KEY.
CL_FARR_RECON_KEY_MGMT	ZCL_FARR_RECON_KEY_MGMT	QUERY_RECON_KEY_NEW	Update to retrieve a value from table ZFARR_D_RECON_KEY.
CL_FARR_REV_COLLECT_DB_ACCESS	ZCL_FARR_REV_COLLECT_DB_ACCESS	SELECT_TABLE	Change SQL to select from tables ZFARR_D_POSTING and ZFARR_D_RECON_KEY.

Standard Class/ Program	To Be Copied to Custom	Methods in Custom to Be Changed	Description
CL_FARR_REV_COLLECT_DB _ACCESS	ZCL_FARR_REV_COLLECT_DB _ACCESS	COLLECT_POSTING_DATA _GRANULAR	Change SQL to select from table ZFARR_D_POSTING for posting entries.
CL_FARR_UPDATE_SET_WHERE	ZCL_FARR_UPDATE_SET _WHERE	SET_HOST_VARIABLE	Change SQL to select from table ZFARR_D_RECON_KEY.
CL_FARR_AC_DOCUMENT	ZCL_FARR_AC_DOCUMENT	CONVERT_TO_ACCIT_ACCCR	Copy from standard CL_FARR_AC_DOCUMENT.

Table 7.5 List of Objects to Be Copied and Changed for Custom Posting Program

7.3 Data Cleanup

Data is the core of any process and it's the feeder to the processes and automations. Data needs to be correct and in the right format for any process to work correctly. Reporting, which is very crucial for any business, won't be accurate if the data has issues. In RAR or any other module, data is the starting point and on which we build our designs and our processes—most importantly, the RAIs are created from the data. There are times when we invest a lot of time trying to investigate or understand what went wrong in the program, jobs, or process, and finally figure out that the issue was with the data. So, the first thing that we should always do is validate and clean up the data.

In classical inbound processing, we know that there are three stages where the data moves from one status to another. First, the data enters as raw, then transfers to processable, and finally is processed when the contracts and the POB are all created in the system. If there is some issue in the system, and then, after a lot of analysis, we find that the issue was in the data, it's too late to catch the data issue, resulting in a lot of back tracing and reworking to make the correction. Therefore, we should scan and clean up the data and then move it into the Adapter Reuse Layer (ARL). It's recommended to look at any errors in data format, values, or data type before we even get the data in the **Raw** status.

First, however, let's look at how SAP has designed the data checks in RAR. We'll also explore the business add-in (BAI)

that is provided and also the programs that you can use. Finally, we'll discuss the custom ways of introducing more checks.

7.3.1 Data Checks

SAP has also incorporated a lot of checks in standard RAR to check for data consistencies and data correctness. In standard RAR, there are two types of checks:

- **Inflight checks**

Inflight checks are designed to prevent data inconsistencies from occurring at all. The inflight checks use the prefix C.

- **Data validations**

Data validation checks detect inconsistencies once they are written permanently to the database. The data validation checks use the prefix E.

Both in-flight checks and data validation checks may be updated by SAP, as and when required. This means that more error categories can be added by SAP depending on whether further common patterns are discovered. SAP recommends that you monitor these on a regular basis.

There is a certain number of category checks, but because this number of checks will definitely change as SAP adds more, we haven't mentioned the numbers here.

We introduced these checks in [Chapter 4, Section 4.4](#), from a functional perspective; in the following sections, we'll take a look at them from a technical perspective.

Inflight Checks

Inflight checks are the checks performed on the data before committing the data to the database and help prevent faulty data from being saved in the database. If you've enabled classical inbound processing, you know the journey that the data takes from the raw tables to the processed tables, as well as how tedious and time-consuming it would be to correct the data issues at the end when the data is in **Processed** status. Inflight checks are just the right thing to track the errors early and get them corrected or resent by the sending system. Letting the faulty data be saved and committed to the database will only complicate things; it's much better to let inflight checks help identify faulty data so you can keep it out of the system in the first place.

Data resulting from the creation or update process of the RAIs or the contracts is first stored in a data buffer that can be written permanently to the database at a later stage (if no errors occurred). Inflight checks validate the constraints between the data buffers. The inflight checks are completed before the database commit, and the database commit is only executed if no inconsistencies are found. Transactions are stopped for contracts with errors. Other contracts without errors can continue.

From Transaction SE91, you can view the message class that stores all the standard messages for the inflight check, as shown in [Figure 7.24](#).

You use the message class for custom messages via message number **100** as it has the options to pass the message parameters, as shown in [Figure 7.25](#).

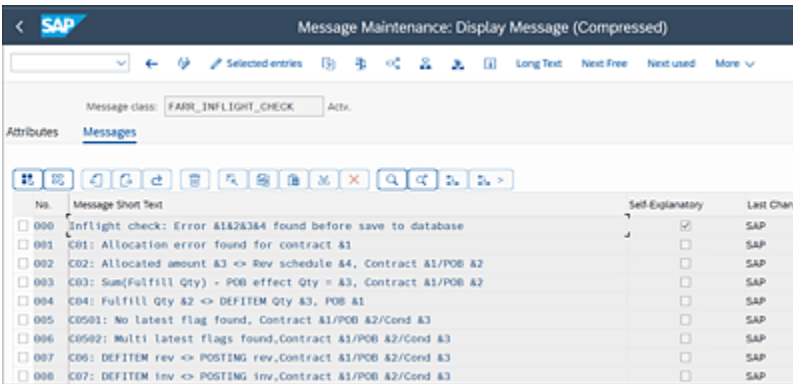


Figure 7.24 Message Class for Inflight Checks

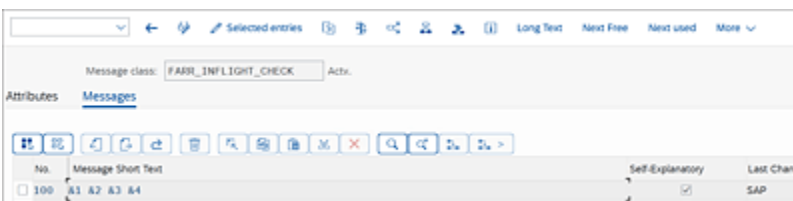


Figure 7.25 Message Class for Inflight Checks (Cont.)

If you want to make additions to the inflight checks or customize the existing check, SAP recommends implementing BAdI FARR_EXTENDED_CHECK. Then, you create a new class as a subclass of CL_FARR_DATA_EXTENDED_CHECK, as shown in [Figure 7.26](#), and implement the redefinition of the following method to suit your needs and add your logic.

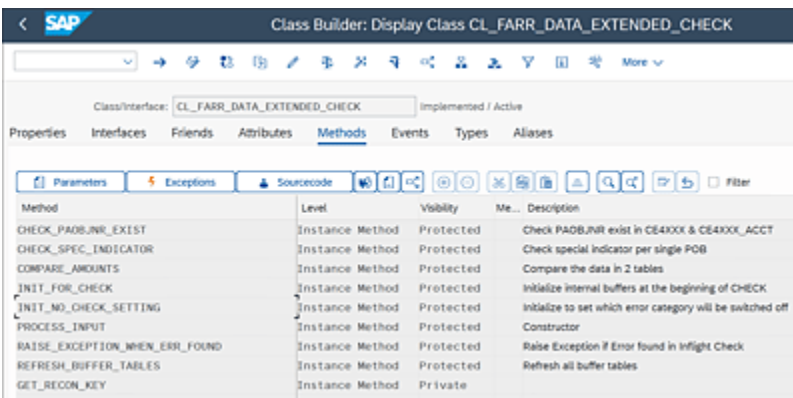


Figure 7.26 Standard Class CL_FARR_DATA_EXTENDED_CHECK for Inflight Checks

The BAdI for inflight checks gives you the flexibility to do the following:

- Switch inflight checks to on or off status.
- Create an exceptions list that will have the list of contracts you want to exclude from the preventative checks.
- Customize the existing checks or add new checks that are specific to your requirement.

Go to Transaction SE18, and enter “FARR_EXTENDED_CHECK” in the **Enhancement Spot** field to arrive at the screen shown in [Figure 7.27](#).

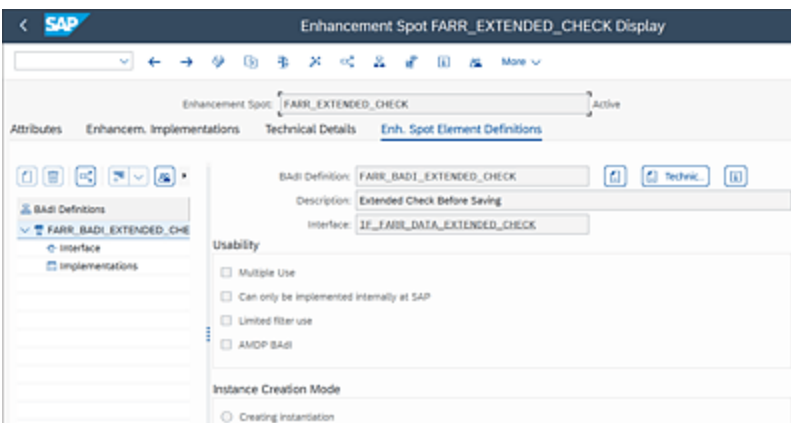


Figure 7.27 BAdI: FARR_EXTENDED_CHECKS

[Figure 7.27](#) shows the details of **FARR_EXTENDED_CHECK**. To see the methods, double-click on the implementation on the left side, and you’ll be taken to the implementing class **CL_FARR_DATA_EXTENDED_CHECK**, as shown in [Figure 7.28](#).

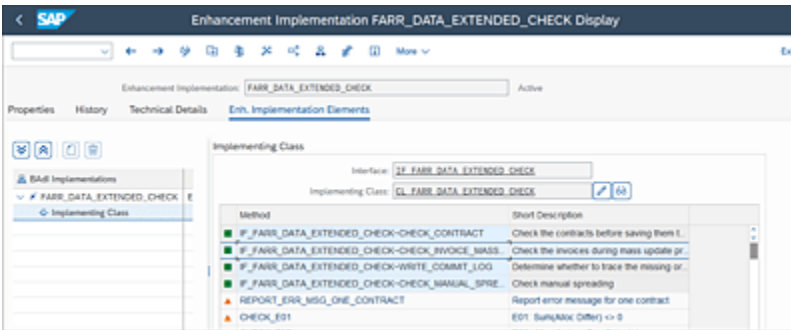


Figure 7.28 Methods of Class CL_FARR_DATA_EXTENDED_CHECK

For example, you could use method INIT_NO_CHECK_SETTING for switching OFF or ON any particular error category checks. There is existing code for method INIT_NO_CHECK_SETTING as you can see from class CL_FARR_DATA_EXTENDED_CHECK. Select and double-click on the method to arrive at its code, as shown in [Figure 7.29](#). You just need to set ABAP_TRUE for the checks that you want to skip. This has to be done in the implementing class's method.

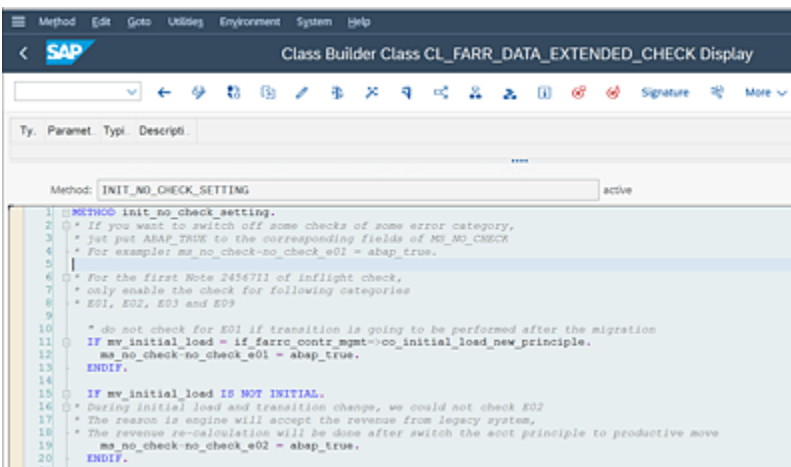


Figure 7.29 Switch On/Off Inflight Checks

```

Class Builder Class CL_FARR_DATA_EXTENDED_CHECK Disp
-----
Ty. Parameter      Typing      Description
-----
IO_MSG_HANDLER    TYPE REF TO CL_FARR_MESSAGE_HANDLER  FARR message handler
IS_CONTRACT_DATA_BUFFER  TYPE FARR_S_CONTRACT_DATA_BUFFER OPTIONAL  One contract buffer data
....

Method: IF_FARR_DATA_EXTENDED_CHECK~CHECK_CONTRACT active
-----
34   its_masn_chng_data = its_masn_chng_data
35   its_change_type_buffer = its_change_type_buffer
36   its_change_type_old = its_change_type_old
37   it_cond_type_data_buffer = it_cond_type_data_buffer
38   iv_fiscal_year = iv_fiscal_year
39   iv_period = iv_period.
40
41   * Initialize buffers at the beginning of check
42   init_for_check( ).
43
44   * Call all of the checks
45   IF ms_no_check-no_check_e01 IS INITIAL.
46     check_e01( ). * Allocation error, Sum(CORR) <> 0
47   ENDIF.
48
49   IF ms_no_check-no_check_e02 IS INITIAL.
50     check_e02( ).
51   ENDIF.
52
53   IF ms_no_check-no_check_e03 IS INITIAL.

```

Figure 7.30 Method IF_FARR_DATA_EXTENDED_CHECK~CHECK_CONTRACT

As already mentioned, to implement this BAdI, you have to first create your own implementation class, inherit class CL_FARR_DATA_EXTENDED_CHECK, and then override its required methods. Then, in the implementation, you can make your own custom logic. For example, let's say you're implementing method CHECK_CONTRACT: you have to first call the implementation of the superclass and then add the requirement-specific logic to it. Existing code for method IF_FARR_DATA_EXTENDED_CHECK~CHECK_CONTRACT is shown in [Figure 7.30](#).

Each of the C errors are very specific and are thoughtfully well designed. They help you understand the error, and they point to the exact issue with the data. The data is then stopped from being processed further. The errors are all categorized; in general, the categories check the following in a specific way:

- In a given contract, the total sum of allocation for all the POBs plus the total allocated price equals the total transaction price.

- The allocated amount of POB equals the total amount of all the periods for this POB in the deferral table.
- For an event-based POB, the effective quantity equals the total quantity of all fulfillments.
- The effective quantity of a time-based POB equals the scheduled quantity from the deferral and fulfillment tables.
- Check that the latest deferral item flag is set correctly.
- The total number of posted revenue equals the scheduled number in the deferral table.
- The total number of posted invoice correction equals the invoiced amount in the deferral item.
- Check for the special indicator.
- The allocation effect of the POB equals the difference between the allocated amount and the transaction price.
- The posted amount in the posting data buffer equals the transferred invoice amount in the invoice data buffer.
- Check that there should be no POBs without deferral items.
- The signs of the transaction currency and local currency should be different.
- The reconciliation keys are in sync with the modifications in the deferral items.
- Check for consistent situations during RAI processing for contracts with fixed exchange rate methods.

For more information on the checks from SAP, see <http://s-prs.co/v570001>.

The error categories are listed in [Table 7.6](#). They will be better interpreted by functional consultants, but it's always good to have an idea and understanding of these even for a developer to know where exactly the issue is.

Error Category	Description
C01	There is an incorrect balance of allocation effects.
C02	The allocated amount isn't equal to the revenue schedule of a POB.
C03	The POB quantity isn't equal to the fulfilled quantity for a fully fulfilled POB.
C04	The fulfilled, unsuspending quantity isn't equal to the fulfilled deferral item quantity for a time-based POB.
C05	The latest entry flag for a deferral item is incorrect or missing.
C0501	Some deferral items don't have the latest entry flag.
C0502	More than one or no deferral item has the latest entry flag for each condition type.
C06	The scheduled revenue amount isn't equal to the posted revenue amount.
C07	The scheduled invoice amount isn't equal to the posted invoice amount.

Error Category	Description
C08	The special indicator flag is incorrect or missing.
C09	The transaction price, allocated amount, and allocation effect of a POB are inconsistent.
C13	The transferred invoice amount isn't equal to the posted invoice amount in the transaction currency.
C15	A POB doesn't have a deferral item.
C17	When invoice correction and posted revenue amounts are generated, they don't display the same sign in all currencies.
C18	The profitability segment of the posting entry isn't the same as that of the POB.
C19	Created, updated, or deleted deferral item has an invalid reconciliation key.
C21	The posted amount in a local currency isn't consistent with the amount calculated using the fixed exchange rate.
C22	POB attributes are inconsistent.
C23	The POB should not have a spreading conflict.
C24	The remaining SSP isn't equal to zero for a POB with the end date in a previous period during a contract modification.

Error Category	Description
C25	The source of price flag is incorrect or missing.
C26	An error occurred during contract modification.
C27	The total scheduled revenue isn't equal to the posted revenue on the latest deferral item.

Table 7.6 Inflight Checks: Error Categories

Data Validations

Data validation checks are post database commit checks, which are implemented in RAR to validate data that is already written to the database tables. Data validation checks implement verifications of equivalent error categories.

Updates

Both inflight checks and data validation checks may be updated by SAP, as and when required. This means that more error categories can be added by SAP, depending on whether further common patterns are discovered. SAP recommends that you monitor these on a regular basis.

Under this, there are two transactions you can use:

- Data validation checks for revenue accounting contracts
- Data validation monitor

Let's start with data validation checks for revenue accounting contracts. Data validation checks are done using Transaction FARR_CONTR_CHECK or program RFARR_PP_CONTRACT_CHECK_START.

When you execute the program, it updates table FARR_D_CONS, which is the consistency check and has the primary key, as shown in [Figure 7.31](#). This can save data for multiple company codes.

Field	Key	Initials	Data element	Data Type	Length	Decimal	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>		MANDT	CLNT	3	0		Client
<input type="checkbox"/> COMPANY_CODE	<input checked="" type="checkbox"/>		BUNGS	CHAR	4	0		Company Code
<input type="checkbox"/> CONTRACT_ID	<input checked="" type="checkbox"/>		FARR_CONTRACT_ID	CHAR	14	0		Revenue Accounting Contract ID
<input type="checkbox"/> POBL_ID	<input checked="" type="checkbox"/>		FARR_POBL_ID	CHAR	16	0		Performance Obligation ID
<input type="checkbox"/> INCLUDE	<input type="checkbox"/>		FARR_S_CONS	STRU	0	0		RA - consistency check / contract data
<input type="checkbox"/> ACCT_PRINCIPLE	<input type="checkbox"/>		ACCOUNTING_PRINCIP	CHAR	4	0		Accounting Principle

Figure 7.31 Consistency Check Log Table

The table saves the error category for data validation, as you can see in [Figure 7.32](#).

The selection screen of the program is shown in [Figure 7.33](#). The program is used for mass updates and updates the entries in table FARR_D_CONS. The program is designed to use the PPF, which is the standard framework.

Field	Key	Initials	Data element	Data Type	Length	Decimal	Coordinate	Short Description
<input type="checkbox"/> ERR02	<input type="checkbox"/>		FARR_ERR02	CHAR	1	0		RA - Data Validation/ Error Category 02
<input type="checkbox"/> ERR03	<input type="checkbox"/>		FARR_ERR03	CHAR	1	0		RA - Data Validation/ Error Category 03
<input type="checkbox"/> ERR04	<input type="checkbox"/>		FARR_ERR04	CHAR	1	0		RA - Data Validation/ Error Category 04
<input type="checkbox"/> ERR05	<input type="checkbox"/>		FARR_ERR05	CHAR	1	0		RA - Data Validation/ Error Category 05
<input type="checkbox"/> ERR06	<input type="checkbox"/>		FARR_ERR06	CHAR	1	0		RA - Data Validation/ Error Category 06
<input type="checkbox"/> ERR07	<input type="checkbox"/>		FARR_ERR07	CHAR	1	0		RA - Data Validation/ Error Category 07
<input type="checkbox"/> ERR08	<input type="checkbox"/>		FARR_ERR08	CHAR	1	0		RA - Data Validation/ Error Category 08
<input type="checkbox"/> ERR09	<input type="checkbox"/>		FARR_ERR09	CHAR	1	0		RA - Data Validation/ Error Category 09
<input type="checkbox"/> ERR10	<input type="checkbox"/>		FARR_ERR10	CHAR	1	0		RA - Data Validation/ Error Category 10

Figure 7.32 Error Categories for Data Validation

The screenshot shows the SAP Data Validation - Revenue Contracts selection screen. It is divided into three main sections:

- Selection Data:** Contains fields for 'Accounting Principle' (with a range selection icon), 'Company Code', and 'Excluding Completed Contracts' (checkbox). There are also 'to:' fields for both 'Accounting Principle' and 'Company Code', each with a selection icon.
- Technical Parameters:** Contains 'Block Size For Mass Selection' (set to 1,000), 'Dialog Mode' (checkbox), and 'Synchronous Call' (checkbox).
- Settings for Application Log:** Contains 'External ID' (text field) and 'Problem Class' (dropdown menu set to 'Important').

Figure 7.33 Data Validation for Revenue Accounting Contracts

Next, let's look at the data validation monitor. You can access it using Transaction FARR_CONTR_MON or program RFARR_CONS_MONITOR. The selection screen of the program is shown in [Figure 7.34](#).

As shown in [Figure 7.34](#), there are fields for **Accounting Principle** and **Company Code**, as well as the option to give desired contract ranges. The processing can occur in three ways:

- **Read data online**
Reads the live data from the contract tables and other relevant tables and gets the data.
- **Read data from error table**
Reads table FARR_D_CONS and displays the result.
- **Save results to error table**
Saves the results to the error table, which is table FARR_D_CONS.

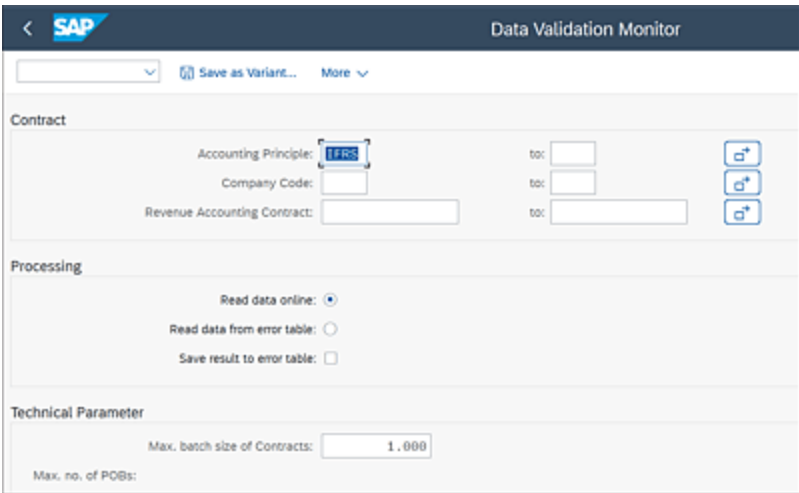


Figure 7.34 Data Validation Monitor Selection Screen

The results from the output screen list the error categories with complete information about the contract, POBs, and more. When you press **F1** for any of them and click on **More Information**, you get detailed information about the error (see [Figure 7.35](#)).

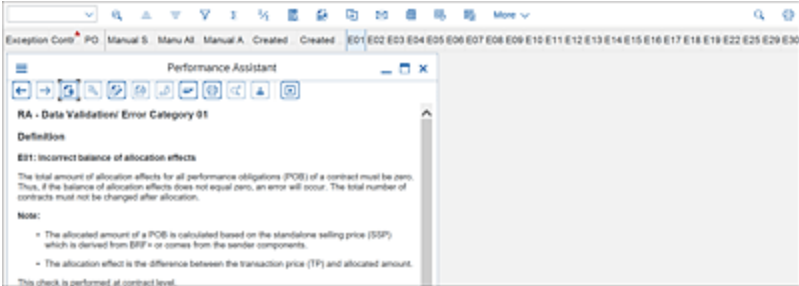


Figure 7.35 Error Category Details from the Consistency Check Monitor

[Table 7.7](#) shows the list of error category details for data validations.

Error Category	Description
E01	There is an incorrect balance of allocation effects.

Error Category	Description
E02	The allocated amount isn't equal to the revenue schedule of a POB.
E03	The POB effective quantity isn't equal to the fulfilled quantity for a fully fulfilled POB.
E04	The fulfilled, unsuspended quantity isn't equal to the fulfilled deferral item quantity for a time-based POB.
E05	The latest entry flag for a deferral item is incorrect or missing.
E06	The scheduled revenue amount isn't equal to the posted revenue amount.
E07	The scheduled invoice amount isn't equal to the posted invoice amount.
E08	The special indicator flag is incorrect or missing.
E09	The transaction price, allocated amount, and allocation effect of a POB are inconsistent.
E10	There is an incorrect balance on the receivable adjustment account in transaction currency.
E11	The deferred revenue/contract liability amount calculated based on the deferral items isn't equal to the corresponding posted amounts.

Error Category	Description
E12	The unbilled receivable/contract asset amount calculated based on the deferral items isn't equal to the corresponding posted amounts.
E13	The transferred invoice amount isn't equal to the posted invoice amount in the transaction currency.
E14	There is an incorrect balance on the receivable adjustment account in local currencies.
E15	A POB doesn't have a deferral item.
E16	The transferred invoice amount isn't equal to the posted invoice amount in the local currencies.
E17	An incorrect cumulated quantity is posted in the deferral items.
E18	There are inconsistent and invalid accounting objects.
E19	Reconciliation keys are missing in the reconciliation keys table.
E22	Inconsistent data appears in conflict management between the contract/POB and manual change data.
E25	The source of price flag is incorrect or missing.

Table 7.7 Data Validation Error Categories

7.3.2 Customizing

In the following sections, we'll discuss a few different customizing options for data cleanup activities.

Custom Program for Cleaning Up Raw RAIs

After looking at the standard data consistency checks and data validations, there are times and situations when you need to perform certain validations and checks, for example, on the raw items, and you only want to transfer the validated and nonfaulty data to **Processable** status. You can design a program that can pull data from the raw table, validate it, and then exempt any items that have errors.

The CleanUp program that we'll be explaining is relevant for any technical or functional consultant to get an idea of how to work on raw data before transferring the RAIs to the **Processable** status. We'll provide a flowchart and also some guidelines for achieving the same. You can call the CleanUp program before transferring the RAIs, as shown in [Figure 7.36](#). The CleanUp program can be designed in any way as needed per the project requirement. The CleanUp program will be required for achieving certain purposes such as removing duplicate items, identifying duplicate invoices based on some functional key, considering only the latest invoices and eliminating the older invalid invoices, and validating and checking custom fields before they can move to the next status.

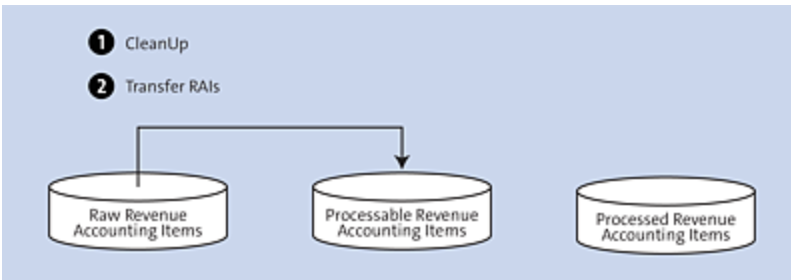


Figure 7.36 CleanUp Program

[Figure 7.37](#) clearly outlines the main steps of the CleanUp program, which will scan through the raw RAIs and perform the necessary validations and checks per the requirement and then exempt the raw RAIs from being moved to the **Processable** status. The raw RAIs that are exempted will be stored in tables `/1RA/0XX011MI` and `/1RA/0XX011CO`.

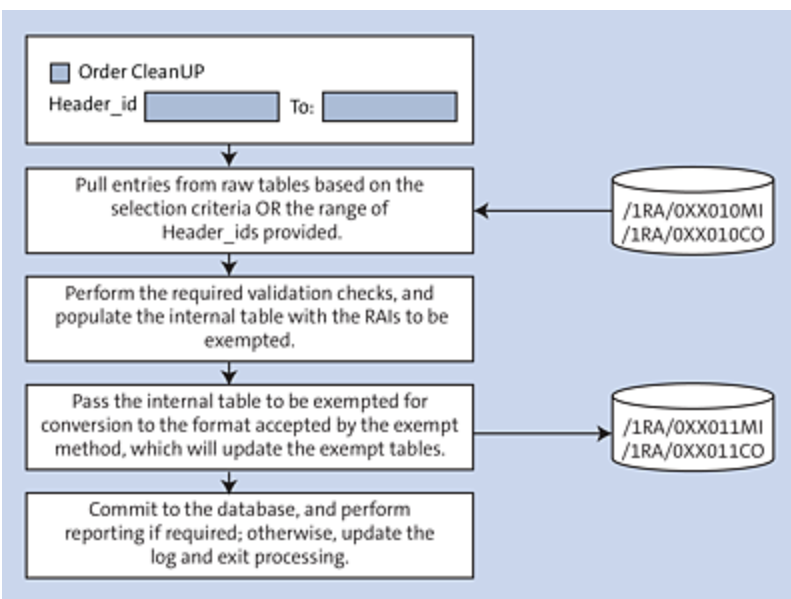


Figure 7.37 CleanUp Program Steps for Raw RAIs

We'll discuss the major steps in the CleanUp program at a very high level so that you can design your CleanUp program using this as their base. A custom CleanUp program should generally have all the following steps:

1. **Design the selection screen of the program**

You can design the selection screen per your requirement, and, in most cases, you can have the options to clean up order RAIs, invoice RAIs, or even fulfillment RAIs. You can have company code as one of the input fields and then provide the option to give the HeaderIDs as mandatory or optional ranges.

2. **Validate the selection screen input**

You can validate the company code and then the HeaderIDs if they are provided.

3. **Fetch data from the database**

You can filter and pull the RAIs from the raw tables for both main items and condition items from tables /1RA/0XX010MI and /1RA/0XX010C0 based on the selection criteria provided on the selection screen and then store them in the internal table.

4. **Packaging and child batch jobs**

If the volume of data is high, you can even design the data to be processed using the customized PPF as we've already discussed in [Section 7.1.1](#).

5. **Process the data**

You then need to apply the necessary checks and validations that you want to check on standard or custom fields and then create another internal table, which will store the RAIs that you want to exempt from processing further in the ARL.

6. **Exempt the raw RAIs**

When you exempt RAIs, you need to use the correct function module for exemption. The raw RAIs are

exempted using the function module shown in [Figure 7.38](#).

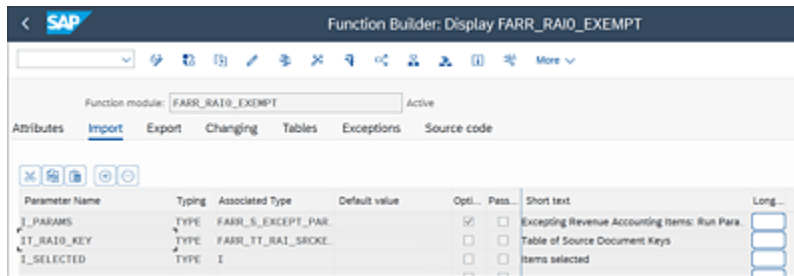


Figure 7.38 Function Module to Exempt Raw RAIs

7. Find the exempted entries

The raw RAIs that are exempted will be found in tables /1RA/0XX011MI for main items and table /1RA/0XX011C0 for condition items.

7.3.3 Business Add-In FARR_BADI_RAIO2

The raw RAIs are transferred and moved from **Raw** status to **Processable** status using Transaction FARR_RAIO_MON or Transaction FARR_RAIO_TRANS. If you plan to incorporate data cleanup or validate RAIs at this point, you can do so by using enhancement spot FARR_ARL and BAdI FARR_BADI_RAIO2, which you can implement via the usual BAdI implementation process.

In this BAdI, there are two methods: ENRICH and CHECK_BEFORE_SAVE. You can use method ENRICH to apply validations and check data consistency, compare and apply any other processing requirements to the raw RAIs before they are changed to processable RAIs, and ensure that the contracts are created or updated with the right data. Along with validation, you can also add the logic to assign values

to standard or custom fields to include default values for custom or standard fields of the RAIs while they are moved to **Processable** status. If you're checking the RAIs or validating the RAIs and find errors, the errored raw RAIs won't be processed further.

Go to Transaction SE18 to view the details of the BAdI; [Figure 7.39](#) shows the ENRICH method.

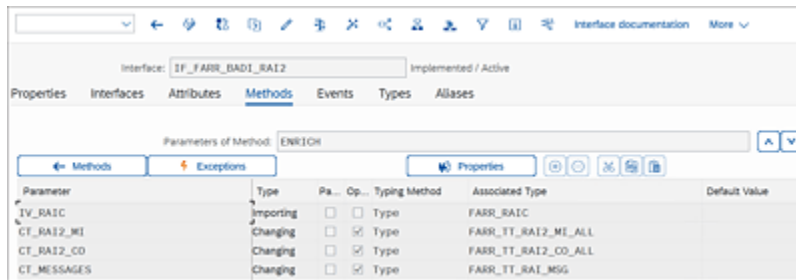


Figure 7.39 ENRICH Method Parameters

In case of errors, the changing parameter **CT_MESSAGES** has to be filled for the RAIs that have errors during validation. The message structure contains attributes for the error message, as well as the key fields for the RAI. It's crucial for the correct processing of the erroneous RAIs that these key fields are filled in the message structure.

[Figure 7.40](#) shows a representation of BAdI FARR_BADI_RA12 being called and its components.

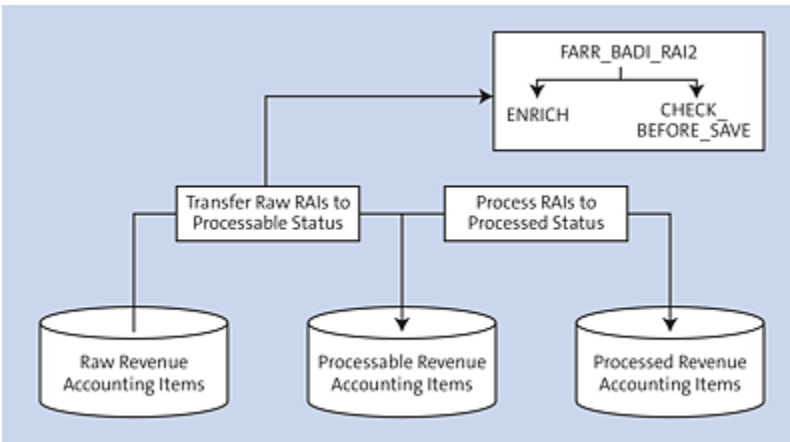


Figure 7.40 BAdI FARR_BADI_RA12

7.3.4 Message Capturing by Simulating

During one of our projects, we developed an object where we captured the messages of the method that is used for the transfer of mass RAIs in simulation mode. As a result, we could capture all errors and corrected the data before it was even transferred to **Processable** status, as we execute the method in simulation mode. This is extremely helpful when you have a lot of errors that are both standard errors and custom errors or checks and validations. This can be designed as a utility or simulating tool for error management. The basic concept of error management is first to detect the error, then understand the error, and finally help the user correct the error.

The tool is focused on organizing and grouping errors while in simulation mode. If we're in execution mode of any method call, we'll be committing the data to the database, so here, we call the method in the tool in simulation mode. Specifically, in RAR, when the data load is huge, we try to create the RAIs and move them from **Raw** status to

Processable using Transaction FARR_RAI_TRANS for the transfer. If there are errors, all the errors are updated in the log and can be seen in Transaction SLG1. However, the way the errors appear in Transaction SLG1 isn't really helpful for us to group a particular type of messages and have a count of it. So, this tool was designed, and it was very handy for detecting errors of similar type, getting the count, and identifying the errored data in groups.

We implemented the tool during the transfer of RAIs in **Raw** status to **Processable** status and also during the process of RAIs from the **Processable** to **Processed** status. Let's discuss the case when using the tool in the transfer.

First, we need to go to Transaction SE11 and create database table ZTRANSFER_MSG to store the messages, as shown in [Table 7.8](#). Having a database table for storing the messages helps in storing and formatting the display so we can group similar kinds of errors.

Field	Data Element	Data Type	Length	Description
MANDT	MANDT	CLNT	3	Client
HEADER_ID	FARR_HEADER_ID	CHAR	20	Header ID of source document for RAI
TIMESTAMP_UTC	FARR_RAI_TMSTMP	DEC	15	Timestamp UTC for RAI

Field	Data Element	Data Type	Length	Description
MSGCOUNT	ZZMSGCOUNTER	INT4	10	Counter of messages for a header ID
DOC_TYPE	ZZORA_DOC_TYPE	CHAR	10	Order or invoice type
MSGID	SYMSGID	CHAR	20	Message class
MSGTY	SYMSGTY	CHAR	1	Message type
MSGNO	SYMSGNO	NUMC	3	Message number
MSGV1	SYMSGV	CHAR	50	Message variable
MSGV2	SYMSGV	CHAR	50	Message variable
MSGV3	SYMSGV	CHAR	50	Message variable
MSGV4	SYMSGV	CHAR	50	Message variable

Field	Data Element	Data Type	Length	Description
MSGTX	BALMSG	CHAR	255	Application log: formatted message text
CREATED_ON	UDATUM	DATS	8	Last changed on
CREATED_AT	UZEIT	TIMS	6	Time

Table 7.8 Table for Error Management

After the table is created, we need to create a program using Transaction SE38 called program ZRAR_ERROR_MGMNT. The selection screen technical details are shown in [Table 7.9](#).

Name	Type	Data Type	Description
P_BUKRS	Parameter	BUKRS	Company code.
S0_HEAD	Select_option	FARR_HEADER_ID	Header ID.

Name	Type	Data Type	Description
P_DEL	Parameter	CHECKBOX	Deletion indicator for table content deletion. Based on this indicator, the content of database table ZTRANSFER_MSG will be deleted if the checkbox is checked; otherwise, it won't be deleted.
P_DATE	Parameter	SY_DATUM	Created on. This will be used for deleting the entries from table ZTRANSFER_MSG for the date mentioned. CREATED_ON >= P_DATE
P_TIME	Parameter	SY_UZEIT	Created at. This will be used for deleting the entries from table ZTRANSFER_MSG for the time and date combination mentioned on the selection screen. CREATED_AT >= P_TIME
P_MSGID	Parameter	SYMSGID	Message class.

Name	Type	Data Type	Description
P_MSGNO	Parameter	SYMSGNO	Message number.

Table 7.9 Error Management Tool's Selection Screen Technical Details

Using these details, we can create the selection screen of the program, which will look something like [Figure 7.41](#).

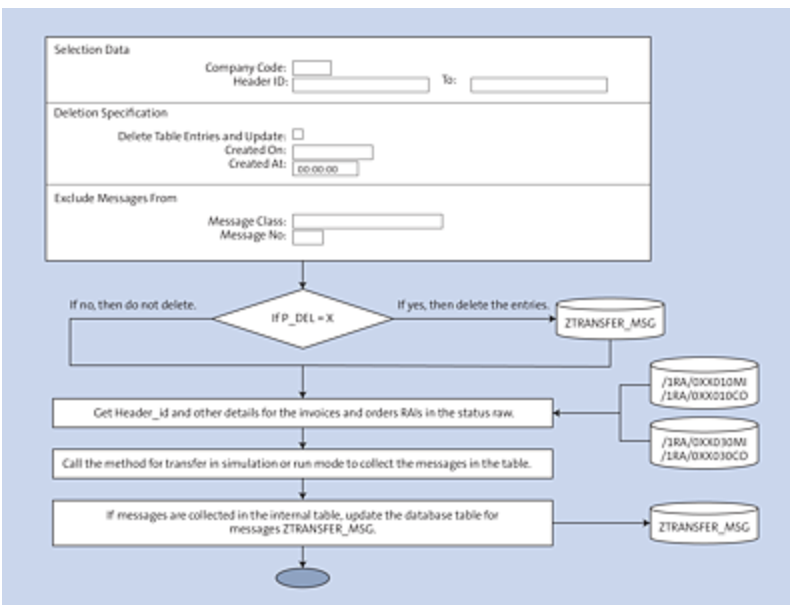


Figure 7.41 Error Management Tool

We need to have logic for validating the selection screen input and have the message handler and all other basic code needed for all the programs. Here we'll focus mainly on the core logic, which is specific to the error management tool.

The processing flow of the program is as follows:

1. P_DEL

If P_DEL is set, then perform Delete_DB. In this perform, we delete entries from database table ZTRANSFER_MSG based on the selection screen parameters Created on and Created

at, as well as delete the entries from table ZTRANSFER_MSG using the following code lines:

```
(Delete FROM ZTRANSFER_MSG WHERE CREATED_ON >= p_date
and CREATED_AT >= p_time.
Endif.
```

This deletes data from the previous execution of the program based on the date and time provided on the selection screen.

2. perform get_header changing gt_header_id.

Select distinct header IDs from the raw tables for orders and invoices based on the selection screen header ID values if it's provided on the selection screen. Store in internal table gt_header_id. The select statements are shown in [Listing 7.2](#).

```
select distinct header_id
  from /1RA/0CA010MI
  into table gt_header_id
  where header_id in so_head.
  if sy-subrc eq 0.
    sort cgt_header_id ascending.
  endif.
```

```
select distinct header_id
  from /1RA/0CA030MI
  appending table gt_header_id
  where header_id in so_head.
  if sy-subrc eq 0.
    sort cgt_header_id ascending.
  endif.
```

Listing 7.2 Select Statements for Fetching the Header IDs from Order and Invoice Raw RAIs

3. If GT_HEADER_ID has values

perform get_timestamp changing gw_tims

In this perform, we get the current timestamp in UTC format.

4. Perform call_rai_transfer

In perform call_rai_transfer, we have two calls: one for order and the other for invoice. Because we have both the orders and invoices, we need to separate them as follows:

- Perform transfer_order: Get all the SRCDOC_ID and other details of the RAIs from the raw table for order /1RA/0XX010MI and store the information in internal table lt_rai0_pack.
- Perform transfer_Invoice: Get all the SRCDOC_ID and other details of the RAIs from the raw table for invoice /1RA/0XX030MI and store the information in internal table lt_rai0_pack.

Once we have the data in internal table lt_rai0_pack, we have to call the following perform for the actual transfer of RAIs:

```
perform call_header_transfer using gw_doc_type lt_rai0_pack.
```

In this perform, the GW_DOC_TYPE will have the value ORDER for order RAIs and INVOICE for invoice RAIs. Method IF_FARR_RAI~TRANSFER_TO_RAI2 is used for transfer of RAIs from **Raw** to **Processable** status, and we call this method as shown next. We call the method for both orders and invoices using the details from internal table lt_rai0_pack.

The actual code lines of the perform call_header_transfer are shown in [Listing 7.3](#).

```
perform call_header_transfer using gw_doc_type lt_rai0_pack. For ORDER and Invoice
if lo_farr_rai is not bound.
    try.
        create object lo_farr_rai
            exporting
```

```

        IO_MSG_HANDLER = go_msg_handler
        iv_sub_obj      = if_farrc_msg_handler_cons=>co_subobj_rai_transfer
        iv_max_probcl  = if_farrc_msg_handler_cons=>co_probclass_low.
    catch cx_farr_message.
        "#EC NO_HANDLER
endtry.

```

```

try.
    lo_farr_rai->IF_FARR_RAI~TRANSFER_TO_RAI2(
        exporting
            iv_test      = gc_x
            iv_no_commit = gc_x
            iv_no_results = gc_x
            iv_last      = gc_x
            it_rai0_pack = lt_rai0_pack ).
*
*
        importing
            et_messages  = lt_messages ).
    catch cx_farr_message.
        message id sy-msgid type sy-msgty number sy-msgno
            with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
endtry.

```

```

CALL METHOD go_msg_handler->get_all_msgs
IMPORTING
    ett_farr_msg = lt_messages.

```

```

go_msg_handler->save_and_close_app_log( ).

```

```

if lt_messages[] is not initial.
read table lt_rai0_pack into data(ls_rai) index 1.
clear : lw_MSGCOUNT, ls_found_error.

```

```

loop at lt_messages into data(ls_messages) where MSGTY <> 'I'.
    ls_found_error = gc_x.
    ls_err_tab-HEADER_ID      = ls_rai-HEADER_ID.
    lw_MSGCOUNT = lw_MSGCOUNT + 1.
    ls_err_tab-MSGCOUNT      = lw_MSGCOUNT.
    ls_err_tab-DOC_TYPE       = LGW_DOC_TYPE.
    ls_err_tab-TIMESTAMP_UTC  = gw_tims.
    ls_err_tab-MSGID          = ls_messages-MSGID.
    ls_err_tab-MSGTY          = ls_messages-MSGTY.
    ls_err_tab-MSGNO          = ls_messages-MSGNO.
    ls_err_tab-MSGV1          = ls_messages-MSGV1.
    ls_err_tab-MSGV2          = ls_messages-MSGV2.
    ls_err_tab-MSGV3          = ls_messages-MSGV3.
    ls_err_tab-MSGV4          = ls_messages-MSGV4.

```

```

clear ls_msg_write.
CALL FUNCTION 'WRITE_MESSAGE'
EXPORTING
    msgid = ls_messages-MSGID
    msgno = ls_messages-MSGNO

```

```

    msgty = ls_messages-MSGTY
    msgv1 = ls_messages-MSGV1
    msgv2 = ls_messages-MSGV2
    MSGV3 = ls_messages-MSGV3
    MSGV4 = ls_messages-MSGV4
*   MSGV5 = ' '
IMPORTING
    messg = ls_msg_write.

ls_err_tab-MSGTX          = ls_msg_write-MSGTX.
ls_err_tab-CREATED_ON    = sy-datum.
ls_err_tab-CREATED_AT    = sy-uzeit.
append ls_err_tab to gt_err_tab.
clear ls_err_tab .
endloop.

if ls_found_error is initial.
    ls_err_tab-HEADER_ID  = ls_rai-HEADER_ID.
    lw_MSGCOUNT = lw_MSGCOUNT + 1.
    ls_err_tab-MSGCOUNT  = lw_MSGCOUNT.
    ls_err_tab-TIMESTAMP_UTC = gw_tims.
    ls_err_tab-MSGID      = 'No Errors Found'.
    ls_err_tab-DOC_TYPE   = LGW_DOC_TYPE.
    ls_err_tab-CREATED_ON = sy-datum.
    ls_err_tab-CREATED_AT = sy-uzeit.
    append ls_err_tab to gt_err_tab.
    clear ls_err_tab .
endif.
endif.
endif.

```

Listing 7.3 Code for Perform CALL_HEADER_TRANSFER

If P_MSGID or P_MSGNO is populated on the selection screen, then delete the entries from gt_err_tab[] which match the input in either P_MSGID or p_MSGNO. We provided this in case you want to ignore some specific messages. We delete them from the table and store the ones that are of interest.

Update the entries of gt_err_tab[] to table ZTRANSFER_MSG, as shown in [Listing 7.4](#).

```

if gt_err_tab[] is not initial.
    modify ZTRANSFER_MSG from table gt_err_tab.
    commit work.
endif.

```

Listing 7.4 Code to Update Database Table ZTRANSFER_MSG with the List of Errors

Once the code is completed, we have to activate the code. Once the code is activated, we can execute the program by pressing **F8**, and then we get the display of the ABAP List Viewer (ALV) grid.

The error will be stored in database table ZTRANSFER_MSG. The error messages are stored with all the information that we need to analyze the error. This table is more organized when compared to the way the errors are stored in Transaction SLGI, where we can see pages of running messages. From this table, we can analyze the errors by grouping and segregating them. We can choose a particular message for a specific header ID and then correct it. Then, we search for the same message for all other header IDs and correct them. This eases the work of error detection and correction. All this can be done without having to commit to the database in simulation mode. Then, after we've corrected the issues, we can go Transaction FARR_RAI_TRANS to transfer the RAIs.

A simple flowchart of the program is shown in [Figure 7.42](#).

The screenshot shows the 'Error management Report' interface. At the top, there's a navigation bar with 'Customer Connection' and 'Error management Report'. Below this, there are buttons for 'Save as Variant...' and 'More'. The interface is divided into three main sections: 'Selection Data', 'Deletion specification', and 'Exclude Messages from'. 'Selection Data' includes a 'Company Code' field with 'E000' entered, and 'Header Id' and 'to' fields. 'Deletion specification' has a checkbox for 'Delete Table entries & update', and 'Created On' and 'Created At' (00:00:00) fields. 'Exclude Messages from' has 'Message Class' and 'Message No' fields.

Figure 7.42 Error Management Tool Flowchart

This design has the following benefits:

- The program will have simulation mode and commit mode, so you can execute the transfer of RAIs in simulation and store the messages without having to commit to the database. This way, you can correct the errors and then run again in commit mode.
- The messages are stored in the table, which makes it easy to organize, group, and correct them.
- The selection screen can be designed to have additional filtering options such as the header ID, which will help us select the header ID that you want to execute and then analyze the errors.
- This also helps in testing as you can use one header ID in simulation mode, and the errors will mostly be repetitive, making it redundant to run all the data for the same type of errors.
- You also have options to segregate messages of a particular class by giving the message class as one of the

selection parameters. Further, you could also add the message numbers to be filtered. This way, you could even look at only custom errors or maybe standard errors only per your requirement.

- The program will have a summarized ALV grid display consolidating the messages by header ID, message class, and message number.

7.4 Helpful Tips and Tricks

To conclude this chapter, we'd like to walk through a few tips and tricks that have proven useful in our projects and that are generally good to know if you're working with RAR administration. We'll start with our custom navigator tool, which helps guide you through the RAR system and objects. We'll then cover additional tricks to keep in mind when working with custom RAIs.

7.4.1 Navigator

Navigator is a simple custom tool that we created during most of our projects, as shown in [Figure 7.43](#). It's a very handy tool or utility that will help you navigate around the RAR objects and transactions. RAR is a step-by-step process of RAI creation in **Raw** status, RAR transfer, and RAR processing; this tool helps you sail through each step in the process.

As a beginner, it's tough to remember all the steps and the transaction codes in RAR; in general that's the case with any module, so we made this navigator, which is just like a process guide or a map with the steps to be followed for the complete execution of RAR steps.

In the navigator tool, all the standard transactions, custom programs, and other utilities related to RAR; the RAR-specific tables; and custom tables can be connected in a single place. It's a very simple design and is very easy to code and to enhance.

To call the navigator, we created Transaction ZRAR_NAVIGATE.

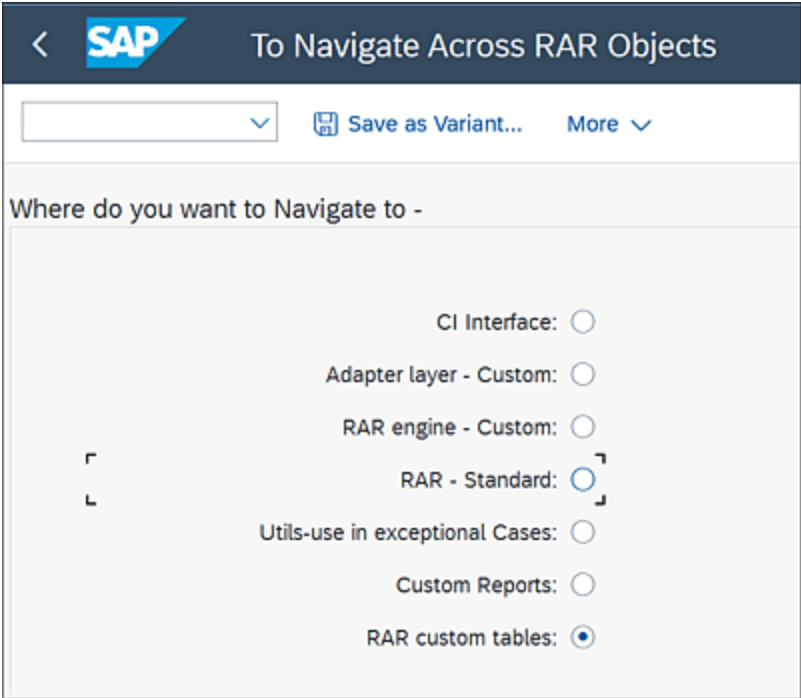


Figure 7.43 Navigator

When you click on any of the radio buttons, you can see the list of transaction codes available. In [Figure 7.44](#), the **RAR - Standard** radio button has been clicked, so you can see all the standard RAR transaction codes that are frequently used. They are arranged in the order of access so you won't miss any steps.

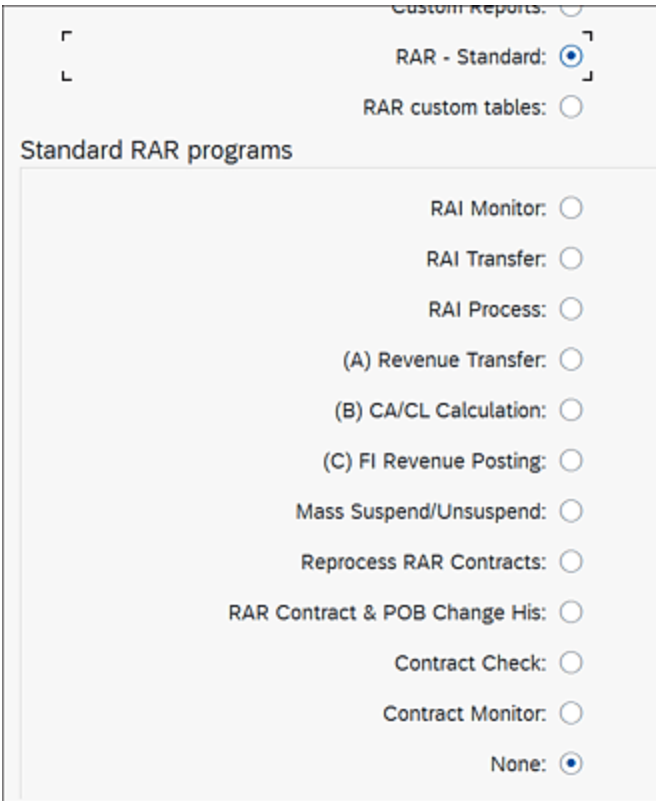


Figure 7.44 Standard RAR Transaction Codes

The ABC programs are also listed here along with other required programs. You can design and group even custom programs, as shown in [Figure 7.44](#), such as the **Adapter layer - Custom** and **RAR custom tables** radio buttons.

Follow these steps to set up your own navigator tool:

1. Go to Transaction SE38, and create a custom program called ZNAVIGATOR. The program will have the logic for calling the transaction codes in the **At Selection Screen Output** block.
2. Create the selection screen in which the first block will be for the main options, as shown in [Listing 7.5](#).

SELECTION-SCREEN BEGIN OF BLOCK f1 WITH FRAME TITLE TEXT-001.

*Standard RAR programs

```

PARAMETER: p_ci type char01 RADIOBUTTON GROUP grp1 USER-COMMAND std, "CI
Interface
    p_adp type char01 RADIOBUTTON GROUP grp1"           Adapter Layer Custom
    p_eng type char01 RADIOBUTTON GROUP grp1,"          RAR Engine Custom
    p_std type char01 RADIOBUTTON GROUP grp1,"          RAR Standard
    p_uti type char01 RADIOBUTTON GROUP grp1,"UtlsUse in Exceptional Cases
    p_cus type char01 RADIOBUTTON GROUP grp1,"          Custom Reports
    p_tab type char01 RADIOBUTTON GROUP grp1 DEFAULT 'X' ." Custom Tables

```

Listing 7.5 Selection Screen for the Main Radio Buttons

3. Add the code under each main radio button. [Listing 7.6](#) shows the sample for **RAR - Standard**.

```

SELECTION-SCREEN BEGIN OF BLOCK s1 WITH FRAME TITLE text-s01.
parameters : p_std1 type c radiobutton group st1 USER-COMMAND r1 MODIF ID 001,
    p_std2 type c radiobutton group st1 MODIF ID 001,
    p_std3 type c radiobutton group st1 MODIF ID 001,
    p_std4 type c radiobutton group st1 MODIF ID 001,
    p_std5 type c radiobutton group st1 MODIF ID 001,
    p_std6 type c radiobutton group st1 MODIF ID 001,
    p_std7 type c radiobutton group st1 MODIF ID 001,
    p_std7A type c radiobutton group st1 MODIF ID 001,
    p_std7B type c radiobutton group st1 MODIF ID 001,
    p_std8 type c radiobutton group st1 MODIF ID 001,
    p_std9 type c radiobutton group st1 MODIF ID 001,
    P_STD10 type c radiobutton group st1 MODIF ID 001 DEFAULT 'X' .
SELECTION-SCREEN END OF BLOCK s1.

```

Listing 7.6 Code for RAR - Standard Block Expansion on Selection Screen

4. Hide all the other radio buttons and display the options only for the one that is selected by looping on the screen, as shown in [Listing 7.7](#).

```

AT SELECTION-SCREEN OUTPUT.
LOOP AT SCREEN
    IF p_std IS NOT INITIAL.
        IF screen-group1 = '001'.
            screen-input = 1.
            screen-active = 1.
            MODIFY SCREEN.
        ENDIF.
    ELSE.
        IF screen-group1 = '001'.
            screen-input = 0.
            screen-active = 0.
        ENDIF.
    ENDIF.

```

```

        MODIFY SCREEN.
    ENDIF.
ENDIF.
endloop.

```

Listing 7.7 Code Block to Display the Details under the Selected Radio Button and Hide the Others

5. You need to have a call transaction to each of the radio buttons under the subblock. So, whenever a radio button is selected, it will basically call a transaction code. A sample code piece for the block **RAR - Standard** is shown in [Listing 7.8](#).

```

IF p_std IS NOT INITIAL.
  CASE 'X'.
    WHEN p_std1.
      CALL TRANSACTION 'FARR_RAI_MON'.
    WHEN p_std2.
      CALL TRANSACTION 'FARR_RAI_TRANS'.
    WHEN p_std3.
      CALL TRANSACTION 'FARR_RAI_PROC'.
    WHEN p_std4.
      CALL TRANSACTION 'FARR_REV_TRANSFER'.
    WHEN p_std5.
      CALL TRANSACTION 'FARR_LIABILITY_CALC'.
    WHEN p_std6.
      CALL TRANSACTION 'FARR_REV_POST'.
    WHEN p_std7.
      CALL TRANSACTION '/HSUD/ORA_MASS_UN SUS'.
    WHEN p_std7A.
      CALL TRANSACTION 'FARR_REPR_CNTR'.
    WHEN p_std7B.
      CALL TRANSACTION 'FARR_CONTR_HIS'.
    WHEN p_std8.
      CALL TRANSACTION 'FARR_CONTR_CHECK'.
    WHEN p_std9.
      CALL TRANSACTION 'FARR_CONTR_MON'.
    WHEN OTHERS.
  ENDCASE.
ENDIF.

```

Listing 7.8 Code Block for Calling the Transaction for the RAR - Standard Radio Button Option

6. The **None** radio button allows the user to choose the others; while the default is always on **None**, this doesn't flow anywhere.
7. You can also design the radio button to take you to a link using the code shown in [Listing 7.9](#).

```
when p_eng4r.  
  call method cl_gui_frontend_services=>execute  
    exporting  
      document = 'https://fn.prod.erp.maersk.com/sap/bc/ui2/flp?sap-  
client=400&sap-language=EN&sap-accessibility=X#AnalyticQuery-browse&/sap-iapp-  
state=ASYWMLEQLL1Q70BL3S7Z70WBAU59SH6RKZTF3KPD'  
    exceptions  
      others    = 1.
```

Listing 7.9 Code Block for Navigation to Links

You can enhance it to use for anything you like that is feasible from a technical perspective.

7.4.2 Additional Information

In this section, you can find more information about the function modules, methods, and other technical objects that you can use. These objects are already in the system in RAR; we're just listing them here so you know what these objects are used for and how to access them.

Mass Creation of RAIs

Some requirements call for you to need to fetch data from staging tables and create RAIs using this data. In such a situation, you'll need a function module or methods to pass large amounts of data and create RAIs en masse. To create

RAIs en mass in your custom program, you can use sample code calling the method.

Go to Transaction SE24, enter the class name as “CL_FARR_RAI” in the **Object Type** field, and then click on **Display** (see [Figure 7.45](#)).

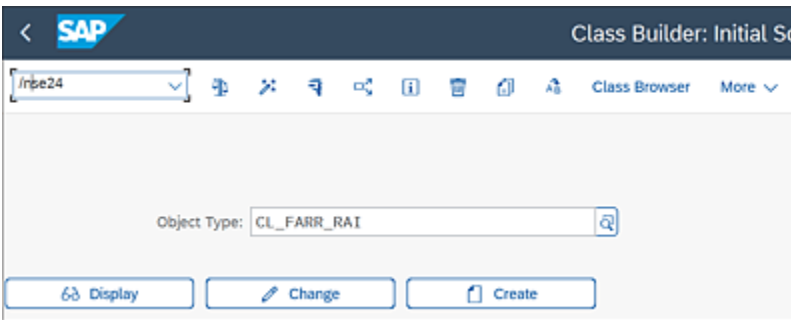


Figure 7.45 Class CL_FARR_RAI

You'll then see a list of methods, as shown in [Figure 7.46](#).

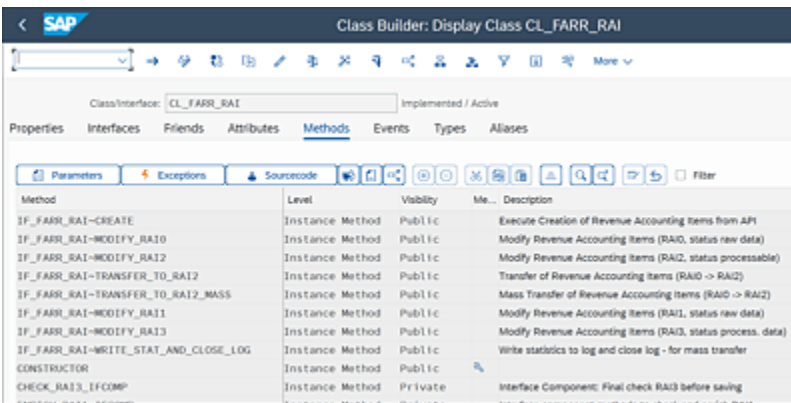


Figure 7.46 Methods of Class CL_FARR_RAI,

From the list of methods, you use **IF_FARR_RAI~CREATE** for creating RAIs. [Figure 7.47](#) shows a sample piece of code to demonstrate how to call the method for creating RAIs from a custom program. In `IV_RAIC = CA01`, CA01 is the order class and will create order RAIs; if you pass CA03, it will create RAIs for invoices.

```

DATA: lo_create      TYPE REF TO cl_farr_rai.
IF lo_create IS INITIAL.
  CREATE OBJECT lo_create.
ENDIF.

TRY.
  CALL METHOD lo_create->if_farr_rai~create
  EXPORTING
    iv_raic      = 'CA01'
    it_api_mi    = ugt_miapi
    it_api_co    = ugt_coapi
  IMPORTING
    et_messages = lt_messages.
  CATCH cx_farr_message.
**    MESSAGE
ENDTRY. "

```

Figure 7.47 Creating RAIs

Transfer RAIs to the Processable Status

Generally, you transfer raw RAIs to processable using Transaction FARR_RAI_MON. Alternatively, you can call Transaction FARR_RAI_TRANS to transfer raw RAIs to the **Processable** status in your custom code; you can use the code sample shown in [Figure 7.48](#).

```

data: lo_farr_rai type ref to CL_FARR_RAI.

try.
  create object lo_farr_rai
  exporting
    IO_MSG_HANDLER = go_msg_handler
    iv_sub_obj     = if_farrc_msg_handler_cons->co_subobj_rai_transfer
    iv_max_probcl = if_farrc_msg_handler_cons->co_probclass_low.
  catch cx_farr_message.
  endtry.

try.
  lo_farr_rai->IF_FARR_RAI-TRANSFER_TO_RAI2(
    exporting
      iv_test      = gc_x
      iv_no_commit = gc_x
      iv_no_results = gc_x
      iv_last      = gc_x
      it_rai0_pack = lt_rai0_pack ).
  importing
    et_messages = lt_messages ).
  catch cx_farr_message.
  message id sy-msgid type sy-msgty number sy-msgno
  with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
endtry.

```

Figure 7.48 Transfer Raw RAIs to the Processable Status

Exempt Raw RAIs

During the project, you may have requirements that call for validating the RAIs that are in different statuses. If they don't satisfy your conditions, then you may have to exempt them. Exempting RAIs is basically stopping the RAI from being processed further. There are two types of RAI exemptions:

- **Exempt raw RAIs**

Exempted raw RAIs are stored in table /1RA/0XX011MI.

- **Exempt processable RAIs**

Exempted processable RAIs are stored in table /1RA/0XX013MI.

To exempt raw RAIs, go to Transaction FARR_RAI_MON, select the required RAIs, and click on the **Exempt** button, as shown in [Figure 7.49](#).

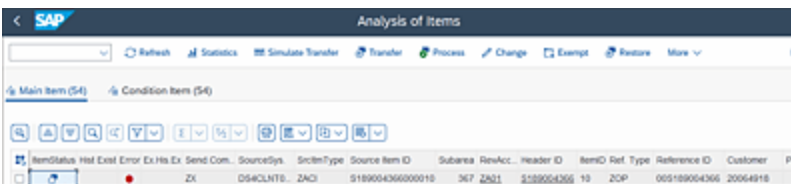


Figure 7.49 Transaction FARR_RAI_MON and the Exempt Button

There will be requirements where you'll have to exempt RAIs from your programs; for that, you can use the function module. Go to Transaction SE37, and display function module FARR_RAI0_EXEMPT, which is used for exempting raw RAIs (see [Figure 7.50](#)).


```

1 FUNCTION farr_rai0_exempt .
2 -----
3 ***Local Interface:
4 ** IMPORTING
5 ** REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL
6 ** REFERENCE(IT_RAI0_KEY) TYPE FARR_TT_RAI_SROCKEY
7 ** REFERENCE(I_SELECTED) TYPE I
8 ** RAISING
9 ** CK_FARR_MESSAGE
10 -----
11 *

```

Figure 7.50 Exempt Raw Entries from Being Moved to Processable

Exempt Processable RAIs

To exempt processable RAIs from being moved to the **Processed** status, you can use Transaction FARR_RAI_MON; however, if you want to exempt them from your custom program, you use function module FARR_RAI2_EXEMPT. To view the function module, go to Transaction SE37, and enter “FARR_RAI2_EXEMPT” in the **Function module** field, as shown in [Figure 7.51](#).

```

1 FUNCTION farr_rai2_exempt .
2 -----
3 ***Local Interface:
4 ** IMPORTING
5 ** REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL
6 ** REFERENCE(IT_RAI2_KEY) TYPE FARR_TT_RAI_SROCKEY
7 ** REFERENCE(I_SELECTED) TYPE I
8 ** RAISING
9 ** CK_FARR_MESSAGE
10 -----
11 *
12 DATA: lt_rai_enq          TYPE farr_tt_rai_enqueue,
13        lt_keypp_bukra_enq TYPE farr_tt_keypp_bukra,
14        lt_rai2_key_del    TYPE farr_tt_rai_srokey,

```

Figure 7.51 Exempt Processable Entries from Being Processed

Restore RAIs

The RAIs that are exempted can be restored via Transaction FARR_RAI_MON. Again, when you restore RAIs, there are two

types:

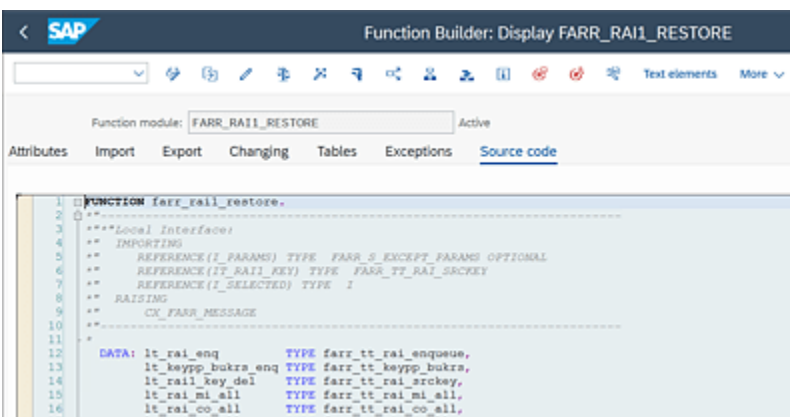
- **Restore raw exempted RAIs**

Restoring raw RAIs will move RAIs from table /1RA/0XX011MI to raw table /1RA/0XX010MI.

- **Restore processable exempted RAIs**

Restoring processable RAIs will move RAIs from table /1RA/0XX013MI to processable table /1RA/0XX012MI.

To restore raw RAIs from a program, you need to use function module FARR_RAI1_RESTORE. To display the function module, go to Transaction SE37, and enter “FARR_RAI1_RESTORE” in the **Function module** field, as shown in [Figure 7.52](#).



```
1 FUNCTION farr_rai1_restore.
2 ***Local Interface***
3 ** IMPORTING
4 ** REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL
5 ** REFERENCE(IT_RAI1_KEY) TYPE FARR_TT_RAI_SCKEY
6 ** REFERENCE(I_SELECTED) TYPE I
7 ** RAISING
8 ** CX_FARR_MESSAGE
9 **
10
11
12 DATA: lt_rai_enq          TYPE farr_tt_rai_enqesue,
13        lt_keypp_bukrs_enq TYPE farr_tt_keypp_bukrs,
14        lt_rai1_key_del    TYPE farr_tt_rai_srckey,
15        lt_rai_mi_all      TYPE farr_tt_rai_mi_all,
16        lt_rai_co_all      TYPE farr_tt_rai_co_all,
```

Figure 7.52 Function Module to Restore Exempted Raw RAIs

Similarly, to restore exempted processable RAIs, another function module has to be called from the custom program. To display the function module, go to Transaction SE37, enter “FARR_RAI3_RESTORE” in the **Function module** field, and click **Display** (see [Figure 7.53](#)).

```

1 FUNCTION farr_rai3_restore.
2
3 ***** Local Interface *****
4 ** IMPORTING
5 ** REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL
6 ** REFERENCE(IT_RAI3_KEY) TYPE FARR_TT_RAI_SCKEY
7 ** REFERENCE(I_SELECTED) TYPE I
8 ** RAISING
9 ** CX_FARR_MESSAGE
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 7.53 Restore Exempted Processable RAIs

Display Useful Methods

There are lot of methods that can be very useful for any custom designs and requirements. Go to Transaction SE24, enter “CL_FARR_RAI” in the **Class/Interface** field, and then click on **Display** to see the list of various methods, as shown in [Figure 7.54](#). A lot of methods in this class can be used in your code to extract data, create raw RAIs, modify RAIs, and so on. It’s always good to keep a note of such useful classes and methods.

Method	Level	Visibility	Me... Description
IF_FARR_RAI-CREATE	Instance Method	Public	Execute Creation of Revenue Accounting Items from API
IF_FARR_RAI-MODIFY_RA10	Instance Method	Public	Modify Revenue Accounting Items (RA10, status raw data)
IF_FARR_RAI-MODIFY_RA12	Instance Method	Public	Modify Revenue Accounting Items (RA12, status processable)
IF_FARR_RAI-TRANSFER_TO_RA12	Instance Method	Public	Transfer of Revenue Accounting Items (RA10 -> RA12)
IF_FARR_RAI-TRANSFER_TO_RA12_MASS	Instance Method	Public	Mass Transfer of Revenue Accounting Items (RA10 -> RA12)
IF_FARR_RAI-MODIFY_RA11	Instance Method	Public	Modify Revenue Accounting Items (RA11, status raw data)
IF_FARR_RAI-MODIFY_RA13	Instance Method	Public	Modify Revenue Accounting Items (RA13, status process. data)
IF_FARR_RAI-WRITE_STAT_AND_CLOSE_LOG	Instance Method	Public	Write statistics to log and close log - for mass transfer
CONSTRUCTOR	Instance Method	Public	
CHECK_RA13_IFCOMP	Instance Method	Private	Interface Component: Final check RA13 before saving
FARR_RAI1_IFCOMP	Instance Method	Private	Interface component methods to check and modify RA11

Figure 7.54 Class CL_FARR_RAI

7.5 Summary

In this chapter, we introduced a lot of technical objects that were developed in our RAR projects. Technical details of some of the objects were given with flowcharts, pseudocode, or code lines to provide a high-level exposure to the development of these objects. The various tools we discussed were very useful in our experience, and we recommend developing the navigator tool for all projects.

Being aware of the function modules, classes, and methods in RAR is always an added advantage to the developer, so we listed the most commonly used ones here in this chapter. Data cleanup, data validation, and error management were all major topics of this chapter.

The next chapter will focus on the migration from classic to optimized contract management (OCM).

8 Migrating to Optimized Contract Management

To adopt and begin using the latest innovations from revenue accounting, you can migrate from classic contract management (CCM) to optimized contract management (OCM). This migration, as any other, comes with many options and caveats; therefore, it deserves a dedicated chapter to describe it in detail.

The first question that comes up is, why migration? For all experienced users in SAP, it's common knowledge that migrations are usually painful processes followed by technical challenges, data reconciliations, sleepless nights due to the need for downtime, regression testing, and so on. Whenever you're considering migration from one version of software to another, you should consider two main issues: whether there is a business case for migration, and whether you're familiar with all the technical aspects of migration.

In this section, we'll cover both. But still the question remains: Why is there a need for migration? As mentioned before, revenue accounting and reporting (RAR) went through a certain evolution process since it was developed until today. At first, it was thought of as a tool for fulfilling International Financial Reporting Standards (IFRS) 15 requirements, and it ended up as a primary tool for revenue

recognition in the latest SAP S/4HANA versions. While going through this evolution, certain features were incorporated in the existing architecture, but at a certain point, it came to its limits. To incorporate features that were required by the market, a deep redesign of the basics was required.

Let's take, for example, one of the main limitations of time-based performance obligations (POBs), which was known by most users: day-based contract modification. CCM relies on fractions calculated by nominator/denominator logic in table FARR_D_DEFITEM. To implement day-based contract modification, the system needed to take into consideration effective dates of fulfillments, invoices, and orders. That shift in logic required much deeper changes both in table structures and program code in which the RAR engine was written.

This is the main reason that migrating from CCM to OCM is considered a technical migration only. The balance of unbilled receivable/deferred revenue (UR/DR) or contract asset/contract liability (CA/CL) before and after migration won't change. In addition, none of the events, such as fulfillments or invoices, will be reprocessed during migration.

In this chapter, we'll provide guidance from preparation to performing migration to post-migration cleanup activities. But first, let's go further in establishing our business case.

8.1 Business Case for Migration

As mentioned, each migration represents a set of activities that need to be executed in a certain order, so it's often looked at as a separate project. The complexity of migration in the RAR space isn't on the same level as other migrations that you might experience (e.g., migration to the new general ledger or to SAP S/4HANA), but it deserves special attention. The first step in migration is to determine what kind of benefits are coming with the new version, and whether you need migration at all.

[Figure 8.1](#) shows the RAR versions for SAP S/4HANA releases.

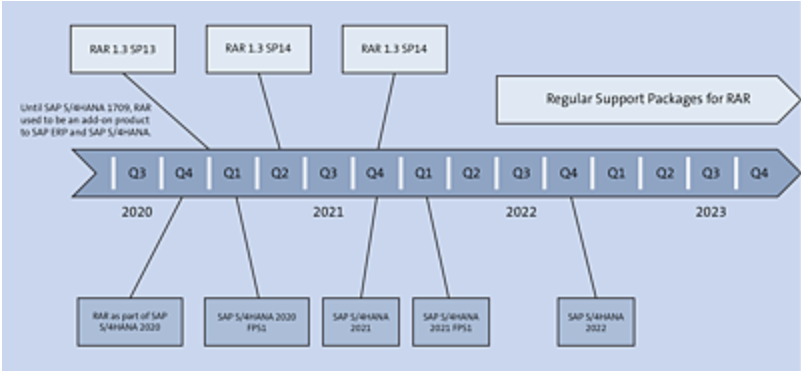


Figure 8.1 Current Versions of RAR on SAP ERP and SAP S/4HANA

As of SAP S/4HANA version 1809, RAR was moved to be an integral part of the SAP S/4HANA stack instead of being an add-on like it was delivered for SAP ERP. It's worth repeating that OCM is a product that will be further improved and extended with new features in RAR. However, it already comes with some features that might be useful for clients such as a new engine for processing contract modifications, processing asset impairments, and performing early termination of POBs.

All these features might be a rationale for you considering the move to OCM because it would allow that custom logic to be retired and replaced with standard. Let's take, for example, the calculation of impairment after contract migration. A solution for this process was developed by customers using different approaches, either as custom programs running at month end or using functionalities as contract deletions in the background. Either of these approaches had limitations because either the activity required manual work (which consequently had the possibility of errors plus audit requirements) or it was very limited as to which business processes would apply. With OCM, you get contract termination as a standard functionality.

Contract Termination: Limitations

The reason SAP hasn't implemented a termination process since the beginning is that there are very different ways the process is implemented with different customers. In some cases, goods return is done after termination, but in some cases, it's not; likewise, in some cases, there is a termination fee that also needs to be considered; and so on. It took time until SAP developed a process generic enough so that it can be used by most customers: the system simply takes the balance of CA/CL, and, according to POBs, it will repost that balance to impairment accounts defined in Business Rules Framework plus (BRFplus). However, there are still some more complicated applications that require enhancing the standard process (e.g., the split of impairment costs to POBs according to some custom rules).

In addition, if migration to OCM is bundled together with migration to optimized inbound processing (OIP) instead of using classic inbound processing, it could bring further benefits such as automatic processing of revenue accounting items (RAIs).

RAI Processing

Users often experienced errors such as “predecessor item not found” when there were high volumes of orders and fulfillment events coming into the system. This happens if the number of RAIs is large, and they try to concurrently access Adapter Reuse Layer (ARL) tables. With optimized versions of inbound processing and contract management, a new parallel processing framework (PPF) was developed that offers many improvements in that area (see [Chapter 7, Section 7.1](#)). For users, the most visible one is that once the sales document is saved, RAI is automatically processed into contract management, bypassing the need for manual triggering of Transaction FARR_RAI_MON and avoiding the errors just mentioned.

It’s also important to mention that you can’t reverse migration back to the classic version once migration is performed.

8.2 Preparation Activities for Migration

Before migration can be started, it's a good idea to prepare both in the sense of activities that need to be executed and how the system should look before migration can actually start. By this, we refer to the setup you'll need to do as well as becoming familiar with which functionalities in OCM replace those in CCM and which functionalities don't exist anymore.

We'll walk through some key preparation activities in the following sections.

8.2.1 Contract Management Activation

The first step in Customizing is setting up a separate contract category for OCM because new contracts will need to be created according to OCM. As already shown in [Chapter 4](#), there is a separate node in Transaction FARR_IMG (follow menu path **Revenue Accounting Contracts • Select Contract Management for Contract Categories**) where the contract category can have OCM activated (see [Figure 8.2](#)).

Select Contract Management for Contract Categories				
CoCd	Company Name	Contr. Cat	Contract Category Description	Contract With CM Instead of CM Classic
<input type="checkbox"/> US19		0001	Revenue Contract	<input type="checkbox"/>
<input type="checkbox"/> US19		0002	Revenue contracts (Classic)	<input type="checkbox"/>

Figure 8.2 Contract Management with OCM

For each combination of company code and contract category, you can configure how the system creates new contracts. Setting the **Contract With CM Instead CM Classic** indicator means that the system will create them using contract management.

If this indicator isn't set, or if a pair of company codes and contract category isn't on the list, these contracts will be created by CCM.

8.2.2 BRFplus Verification

The next step is to verify the BRFplus settings. Before the system uses the setup performed in Transaction FARR_IMG, it will be validated against BRFplus. That is the reason why you must check Customizing and create required updates.

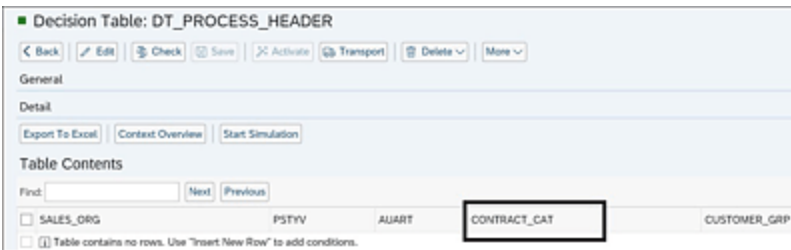


Figure 8.3 Customizing in BRFplus

As we've discussed, contracts are migrated from CCM to OCM, and inbound processing will create contracts using OCM. However, inbound processing is still using derivation as defined in BRFplus decision table DT_PROCESS_HEADER in applications for definition of POBs, as shown in [Figure 8.3](#). In this decision table, you must verify that the contract category (**CONTRACT_CAT**) is the same category that is being customized for OCM.

You first define the contract category in Customizing to be used with OCM, and then the same category is used in decision table DT_PROCESS_HEADER. This will be of particular importance if you'll be using both CCM and OCM in your system.

8.2.3 Business Add-In Modifications

Now, let's look at technical limitations, which mainly refers to business add-ins (BAdIs) that are replaced with new ones or even that are retired, as shown in [Table 8.1](#). This step is extra important because it affects user developments, and you need to know that there will be changes in the existing development and maybe even updates of the business process.

BAdIs for CCM	BAdIs for OCM
FARR_BADI_ALLOCATION_ENGINE, FARR_BADI_ALLOCATION_METHOD	FARR_BADI_PRICE_ALLOCATION
FARR_BADI_DEFERRAL_METHOD	FARR_BADI_DEFERRAL_METHOD_V2
FARR_CHANGE_MODE_DETERMINATION	FARR_BADI_CHANGE_TYPE_DET
FARR_BADI_TM_REMAINING_PERC	Obsolete

Table 8.1 BAdIs Changed in OCM

The first difference will be noticed by customers making enhancements to the standard allocation engine using the delivered BAdIs. In CCM, allocation BAdIs are executed as a two-step process: first, price allocation is prepared, and later, allocation is performed. Allocation starts by looking at

the change method: If a contract is created or the change type of a contract is a change of estimates, attributes of all POBs and pricing condition types are available for price allocation. If the change type of a contract is a contract modification, only attributes of open POBs, remaining pricing condition types, and remaining allocation effects are available for price allocation.

After that, the BAdI checks the total allocatable amount of the standalone selling price (SSP) by verifying whether there are POBs excluded from the allocation or where SSP would be 0 (their transactional price is still included in the allocation). Next, the BAdI checks what kind of POBs are included in the allocation: linked, compound, or distinct. They are then grouped so that POBs that aren't members of any group would receive an allocatable amount first, where the allocable amount is the total of the allocable pricing condition types and allocation effects. Once data is prepared, it would call BAdI `FARR_BADI_ALLOCATION_METHOD` to perform the actual allocation.

In OCM, this logic was completely redesigned with BAdI `FARR_BADI_PRICE_ALLOCATION` and is performed in several steps:

1. Determine the allocated price of residual POBs via the following formula:

$$\text{Allocated price of residual POB} = \text{Max}\{(\text{Total transaction price} - \sum \text{SSP of each POB}, 0)\}$$

2. Determine the allocated price for each nonresidual POB via the following formula:

$$\text{Allocated price of each nonresidual POB} = (\text{Total contract price} - \text{Allocated price of residual POB}) \times \text{SSP of this POB} \div \sum \text{SSP of each POB}$$

3. Revenue accounting performs price allocation based on the SSP if no POB is defined as a residual POB.
4. Revenue accounting checks if each POB is SSP-compliant based on the contractual price, SSP, and SSP range amount:
 - If the POB doesn't have an SSP range amount defined, the SSP range amount is taken as 0.
 - A POB is SSP-compliant if the contractual price of the POB is within the SSP +/- SSP range amount.
 - If all POBs are SSP-compliant, the allocated price of each POB is equal to the contractual price of the POB.

If the total SSP is 0, the allocated price of each POB is equal to the contractual price of the POB.

BAdI `FARR_BADI_DEFERRAL_METHOD` was used by those who needed some custom logic when it comes to splitting revenue in different periods for time-based POBs. It would require that the user creates a new deferral key, and that custom logic would be written in method `GENERATE_FULFILL_ENTRY`. The main difference between the old BAdI and the new BAdI, `FARR_BADI_DEFERRAL_METHOD_V2`, is how they work with table `FARR_D_DEFITEM`.

One more example where new BAdI can become very useful is working with suspension periods. Contract suspensions are commonly used in some industries. For example, in telecommunications (telco; especially in regions with a high number of expats), customers have an option to simply suspend a contract for a maximum defined time without extension of the contract, and revenue for this period must not be recognized.

By using the mentioned BAdI, that can be relatively easily achieved. Let's consider an example:

- Original contract start: 01.01.2023
- Original contract end: 12.31.2024
- Suspension start: 06.01.2023
- Suspension end: 06.30.2023

This contract now has two validity dates compared to the one in the original case:

- First: 01.01.2023 -05.31.2023
- Second: 07.01.2023 -12.31.2024

Now, we'll achieve this by sending two different calls to method `GENERATE_FULFILLMENTS`: in both calls, we'll work only with `DEFERRALPERIODSTARTDATE` and `DEFERRALPERIODENDDATE` parameters because there was no change in the POB end date:

- First call:
 - `DEFERRALPERIODSTARTDATE = 01.01.2023`
 - `DEFERRALPERIODENDDATE = 05.31.2023`
- Second call:
 - `DEFERRALPERIODSTARTDATE = 07.01.2023`
 - `DEFERRALPERIODENDDATE = 12.31.2024`

In only a few lines of code using BAdI `FARR_BADI_DEFFERAL_METHOD_V2`, you can implement the setting for contract suspension. However, when implementing this

BAdI, be aware that detailed testing needs to happen before using it.

One of the most commonly used BAdIs—BAdI FARR_BADI_CHANGE_TYPE_DET—was for determining whether the contract was going through a change of estimates (retrospective change) or contract modification (prospective change). Because RAR was determined based on internal rules regarding whether the system executed the first or second type, users often implemented this BAdI to gain more control over the process.

The main difference between CCM and OCM is what default implementations would trigger, as shown in [Table 8.2](#).

	Change of Estimates	Change of Modifications
CCM	<ul style="list-style-type: none"> • Update the POB exclude from the allocation attribute from true to false or from false to true without any allocation difference. • Change the estimated quantity of the POB. • Update the residual allocation of the POB. • Change the SSP, SSP currency, or SSP tolerance amount without any change to the quantity, duration, or price. 	<ul style="list-style-type: none"> • Change or switch the pricing condition types without total contractual price changing. • Add a new POB with an SSP greater than 0, an SSP tolerance that isn't equal to 0, or with a contractual price

	<ul style="list-style-type: none"> • Change the SSP tolerance percentage. 	Change of Modifications
	<ul style="list-style-type: none"> • Change the contractual price of all allocation-relevant POBs to 0. 	<p>that isn't equal to 0.</p> <ul style="list-style-type: none"> • Delete a POB. • Change the SSP, SSP currency, or SSP tolerance amount with a quantity, price, or duration change. • Change the contractual price of a POB (unit-distinct or non-unit-distinct) that's from operational documents, whereby there's at least one open POB in the contract. • Create an invoice that changes the contractual price where there is at least one open

	Change of Estimates	Change of Modifications
		POB in the contract.
OCM	<ul style="list-style-type: none"> • A POB is deleted. • The start date of a time-based POB is changed. • Deferral method of a time-based POB is changed. • Transaction price of a POB is changed to 0. • Transaction price, original transaction price, start date, end date, deferral method, and quantity of a POB aren't changed, but any of the following fields of the POB are changed: SSP, SSP range percentage, SSP range amount, residual flag, and prevent allocation flag. 	All other scenarios can be used.

Table 8.2 Treatment of Contract Modifications

Because, in the past, users would implement this BAdI to enforce contract modification over change of estimates in some scenarios, in OCM, this might become obsolete. With OCM, the way RAR is determined was changed to be treated

as prospective or retrospective. For some business scenarios where the user was enforcing the change type, it might be covered by the standard.

You should carefully evaluate and analyze changes that come with the mentioned BAdIs. The reason is that the impact can be different: some BAdIs require reimplementation (e.g., the allocation BAdI where instead of two BAdIs, there's only one), but some other implementations might become completely obsolete, as in the case of enforcing certain types of contract change.

8.2.4 Process Changes

Besides BAdI implementations, with OCM, there is also an impact on business processes: some of them change how they are processed, but some of them aren't supported anymore (see [Table 8.3](#)). More detailed information can be found in OSS notes that describe specifics of migration from CCM to OCM, so here we'll focus only on limitations that can be resolved before migration starts. However, it's very important that before migration starts, you get familiarized with features that might not be available once migration to OCM is finished.

Process	Resolution
Contracts with errors	The error must be resolved before the contract can be migrated.

Process	Resolution
Contracts with In Review status	The review must be resolved, and the contract must be in the In Process status.
Completed contracts	Completed contracts aren't migrated before reopening.
Contracts with fixed exchange rate method	This will be automatically changed to the actual foreign exchange (FX) method
POBs with finalization date after the system's current date	The date of finalization will be changed, or the POB will be migrated at a later time.
Time-based POB with a start date that's determined by a fulfillment event	Migration is only supported if the quantity is 1.
POBs with early termination	Early termination is canceled.

Table 8.3 Process Change Coming with OCM Migration

The first thing you need to do is make sure there are no contracts that are either in error or conflict resolution processes because such contracts won't be migrated. Reasons for errors need to be found and resolved before the process can continue. A similar activity needs to be performed for POBs with termination and POBs where the finalization date is at a later point in time than the date when the actual migration is being performed.

In this phase, it becomes handy to use the data validation check reports that SAP provides. It becomes very easy to locate contracts with errors and if possible, resolve them, or else exclude those contracts from the migration activities.

During the migration of contracts with a fixed exchange rate, the system changes the local currency calculation method to A (actual exchange rate method). The exchange rate difference is calculated during the next execution of the Transfer Revenue (Transaction FARR_REV_TRANS) report.

If time-based POBs have a start date of value 3 (start date determined by fulfillment event), migration of such items will be performed only if the quantity is 1 and the start date type is going to be changed to 2 (available after creation of POB).

Beside this, it's important to mention that certain processes aren't supported:

- Contracts with POBs that have manually calculated allocation
- POBs with simplified invoice handling
- Contract acquisition costs POBs
- POBs that are part of compound or bill of materials (BOM) structures

All of these represent detailed information about all the processes and enhancement points being changed or made obsolete when it comes to migration to OCM. OCM is undoubtedly a more advanced concept and brings a lot of improvements to RAR, both process wise and technically. However, before you decide that migration is the right

approach, be sure to carefully evaluate whether OCM is compatible with your existing processes and what kind of effort migration may represent.

8.3 Performing the Migration

Once all preparation activities are finished, you're ready to perform the actual migration activities. There are two prerequisites that need to be performed before actual migration can start. First is that no reconciliation keys can be in the **Open** status at the moment migration starts. Closing reconciliation keys will be done automatically by executing the revenue transfer program (program A; see [Chapter 6](#)). All reconciliation keys in future periods must have status **O (Open)**. If any reconciliation keys exist in a future period, migration must be performed at a later point in time.

The second prerequisite is related to processes that can't run during migration. Migration itself doesn't require downtime; however, depending on the selection parameters and the data volume, a migration run can potentially take a significant amount of time to complete. To avoid errors during the migration of contracts, processing of RAIs for the duration of the migration shouldn't be performed.

Once you're done, everything is ready to start to migrate contracts from CCM to OCM. We'll discuss the core activities and some troubleshooting guidance in the following sections.

8.3.1 Migration Activities

SAP delivers Transaction FARR_CCM_OCM_MIG_CON (program name FARR_CCM_OCM_MIGRATION) to complete

migration activities, as shown in [Figure 8.4](#).

First, select an **Accounting Principle**. Many customers have parallel accounting in their RAR systems, so if both contracts need to be selected, you'll need to perform migration twice. Usually, two accounting principles are created but only one revenue split according to allocation is used. You need to assess what the benefits would be if both contracts were selected in migration, or if migration will be related to one accounting principle.

The next entry that is mandatory is the **Company Code** field for which you can select multiple companies for the migration in one run.

If there are contracts that should be skipped, you can make a list of contracts and enter a range in the next parameter, **Revenue Accounting Contract**. This is done in the standard way by using multiple entry features. If a range is entered, it's crosschecked against the entries in the previous **Accounting Principle** and **Company Code** fields.

The screenshot shows a software interface titled "Selection Parameters" for migrating from CCM to OCM. It is organized into four distinct sections:

- Selection Parameters:** This section contains three main input fields, each with a search icon to its right. The first is labeled "*Accounting Principle:" and is currently empty. The second is labeled "*Company Code:" and is also empty, with a "to:" label and another search icon to its right. The third is labeled "Revenue Accounting Contract:" and is empty, with a "to:" label and another search icon to its right.
- Additional Migration Parameters:** This section contains three checkboxes, all of which are currently unchecked:
 - Mig. Fixed Rate RA Contract:
 - Mig. POB with Start Date Ty: 3:
 - Mig. POB with Manual Spread:
- Run Parameters:** This section contains two numeric input fields and two checkboxes:
 - *Max Work Proc Usage in Percent:
 - *Block Size for Mass Selection:
 - Dialog Mode:
 - Simulation Mode:
- Settings for Application Log:** This section contains a dropdown menu and a checkbox:
 - Problem class:
 - Show Application Log:

Figure 8.4 Program for Migrating from CCM to OCM

The checkboxes in the **Additional Migration Parameters** section are used if you're using special features in existing contracts. The **Mig. Fixed Rate RA Contract** flag is used to enable the migration of revenue accounting contracts with a fixed local currency calculation. During the migration of these contracts, the system changes the local currency calculation method to A (actual exchange rate method). The exchange rate difference is calculated during the next execution of the Transfer Revenue (Transaction FARR_REV_TRANS) report. The checkboxes are left unselected by default.

Set the **Mig. POB with Start Date Ty. 3** flag to enable the migration of time-based POBs with start date type **3** (available after POB creation). The migration of these POBs is only supported if the quantity is **1**. If this POB is migrated, the start date type is changed to **2** (available after POB creation) by the system. The default value is **No**.

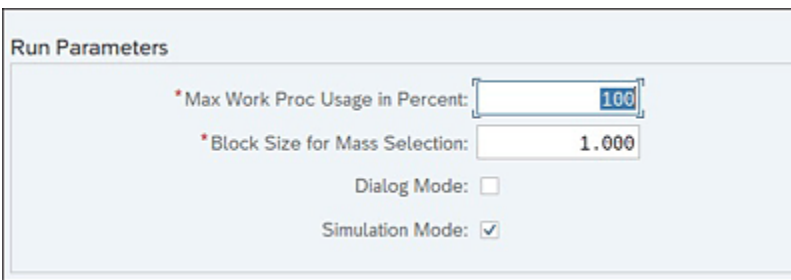
Set the **Mig. POB with Manual Spread** flag to enable the migration of time-based POBs with a manual spreading of the revenue schedule. The migration of these POBs supports the functionality of freeze periods in OCM. To apply freeze periods, insert the required information (POB ID, freeze date, unfreeze date) into table FARR_D_POB_FRZ before you start the migration report, as shown in [Figure 8.5](#).

Fid name	O...	Fr/Value	To value	More	Output	Technical name
Client					<input checked="" type="checkbox"/>	CLIENT
POB					<input checked="" type="checkbox"/>	POB_ID
Freeze Date					<input checked="" type="checkbox"/>	FREEZE_DATE
Unfreeze Date					<input checked="" type="checkbox"/>	UNFREEZE_DATE
Created by					<input checked="" type="checkbox"/>	CHANGED_BY
Time Stamp					<input checked="" type="checkbox"/>	CHANGED_ON

Figure 8.5 Freeze Periods Entry

During the migration, the system removes the manual revenue schedule spreading and applies the freeze period. The default value is **No**.

The next step is to set up **Run Parameters**, as shown in [Figure 8.6](#). These parameters have similar functions as while running RAR programs: based on resources, you'll define how migration programs are going to run.



The screenshot shows a dialog box titled "Run Parameters". It contains the following fields and options:

- Max Work Proc Usage in Percent:** A text input field containing the value "100".
- Block Size for Mass Selection:** A text input field containing the value "1.000".
- Dialog Mode:** An unchecked checkbox.
- Simulation Mode:** A checked checkbox.

Figure 8.6 Run Parameters Setup

In the first field, **Max Work Proc Usage in Percent**, enter what percentage of available processing power will be dedicated to the migration program. In the next field, **Block Size for Mass Selection**, you define the block size, and these two settings in combination determine how fast migration will be executed. There are two aspects to consider:

- What resources are available in the application server on which RAR is running? The number of processes depends on the number of CPUs and cores per CPU.
- Block size determines the number of items that will be assigned to each process.

In [Chapter 7, Section 7.1](#), you can find more information about the PPF setup. The last two options available in the

Run Parameters section are methods of running: **Simulation Mode** is a test run, and selecting **Dialog Mode** determines that the program will be run in the foreground (rather than the background). Although migration doesn't require downtime, bearing in mind that very often the SAP S/4HANA or SAP ERP instance is on the same system as RAR, it's always a good idea to run migration in the background.

Once the setup is done, the program can be scheduled to run using standard SAP transactions such as Transaction SM37 and Transaction SM36, or by an external job management tool if the same is being used. Migration will be executed on the first day of an open RAR period. Steps will be performed automatically by the system in the following order for each revenue contract:

1. The RAR contract is locked.
2. Data validation checks are executed to make sure the contract is consistent.
3. Data is converted to the OCM format. Most of the data remains unchanged, aside from the following:
 - Local currency calculation method is changed to A.
 - Contract asset and contract liability accounts are redetermined.
 - Attribute contract balance presentation is changed according to Customizing settings.
 - If the POB is time-based, the attribute effective quantity is filled with the number of days between the

start date and the end date according to the deferral method.

- The effective quantity unit is changed to days.
- If the POB is based on percentage of completion (POC), the attribute effective quantity is set to 100, and the effective quantity unit is changed to %.
- The **Fulfillments Based on Values** flag is selected if the event type is CI, and the **Indicates Whether Quantity on the Invoice is Relevant or Not** flag is selected.

Entries in fulfillment table FARR_D_FULFILLMT and deferral item table FARR_D_DEFITEM that belong to reconciliation keys with a status other than **O (Open)** are marked as legacy data by selecting the **Legacy Entry** flag. These lines will no longer be loaded by OCM when an event is processed. Instead, new lines are created to represent the total of the recognized revenue, fulfilled quantity, and billed amount for each condition type. Future fulfillments and deferral items of time-based POBs are then recreated based on the new day-based deferral methods.

All fields that hold a numerator or denominator (identified by the suffix **_NM** and **_DN**) are cleared. These columns aren't used by OCM. Make sure that you adapt any reporting solutions based on deferral item table FARR_D_DEFITEM if they depend on the quantity and quantity unit when they change. This is very important if you have reporting based on those columns. Contracts that are now processed by OCM have the **RAR Version Code** flag set to **X**.

8.3.2 Migration Errors

During the migration run, there are a few errors that can occur, and most of them are related to limitations mentioned at the beginning about what kind of contracts can and can't be migrated. However, some errors can happen, most likely due to different stages of the system on which migration is being performed.

It's always a good idea to run migration with smaller number of contracts and in foreground mode. That way, you can see, evaluate, and correct errors before real, productive migration is run. Let's walk through a few examples of errors you might encounter.

If you see the kinds of messages shown in [Figure 8.7](#), it's a good idea to run a validation report and verify what kind of error is behind the message. Remember, SAP won't let any contract be migrated that already has validation errors.

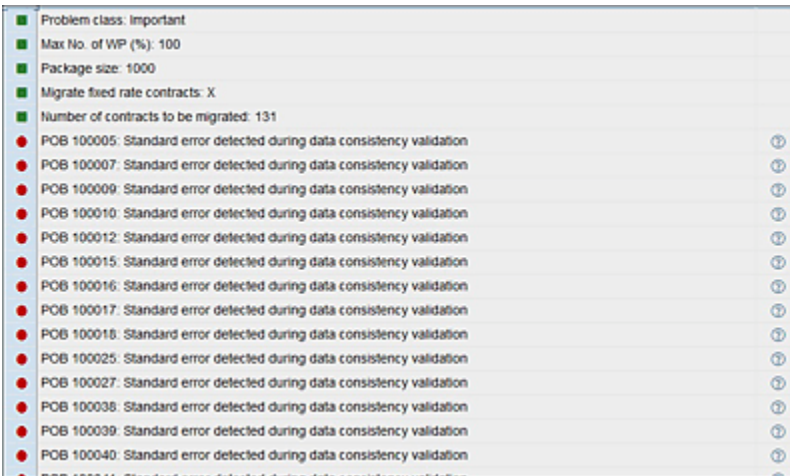


Figure 8.7 Validation Errors

[Figure 8.8](#) shows the kind of error you'll get if you try to migrate contracts that are integrated with results analysis.

At the time of writing, integration with results analysis can be done only by using CCM; OCM isn't supported.

■	Simulation mode
■	Problem class: Important
■	Max No. of WP (%): 100
■	Package size: 1000
■	Migrate fixed rate contracts: X
■	Number of contracts to be migrated: 1
●	POB 116297: POB with CO object number is not supported
■	Number of migrated contracts: 0

Figure 8.8 CO Object Error

In [Figure 8.9](#), the system is warning that migrating contracts with **Fixed Local Calculation Method is not supported**. In this case, look again at the options in the program and you'll see that in the **Additional Migration Parameters** section, the **Mig. Fixed Rate RA Contract** option needs to be selected to avoid this error. After migration is completed, all contracts will be translated according to the actual exchange rate.

Date/Time/User	Number	External ID	Object
17.03.2023 19:15:13 SM6918	144	SM69182023031719...	Reven
● Problem class Important			
144			

Ty...	Message Text
●	Contract 100012: Fixed Local Currency Calculation Method is not supported
●	Contract 100014: Fixed Local Currency Calculation Method is not supported
■	Contract 100015: Completed contract is not supported
●	Contract 100016: Fixed Local Currency Calculation Method is not supported
●	Contract 100017: Fixed Local Currency Calculation Method is not supported
●	Contract 100018: Fixed Local Currency Calculation Method is not supported
●	Contract 100019: Fixed Local Currency Calculation Method is not supported
●	Contract 100021: Fixed Local Currency Calculation Method is not supported
●	Contract 100023: Fixed Local Currency Calculation Method is not supported
●	Contract 100024: Fixed Local Currency Calculation Method is not supported

Figure 8.9 Fixed Currency Translation Error

In this case, as shown in [Figure 8.10](#), errors occur because there are open reconciliation keys in the migration period or there are events in the future. It's one of the preconditions for running migration that all reconciliation keys in the migration period are closed. However, if there are objects with open reconciliation keys, you can either exclude those contracts or opt for migrating them in the future when those problems won't be visible and will let migration complete.

Date/Time/User	Number	External ID
17.03.2023 19:18:59 SM6918	161	SM69182023031719.
Problem class Important	161	

Ty...	Message Text
●	POB 100116: Invoice exists with open recon key in migration period
●	POB 112002: Invoice exists with open recon key in migration period
●	POB 112040: Invoice exists with open recon key in migration period
●	POB 112040: Invoice exists with open recon key in migration period
●	POB 108117: Invoice exists with open recon key in migration period
●	POB 108127: Invoice exists with open recon key in migration period
●	Event-based POB 116282: Fulfillment exists with posting day in future period
●	Event-based POB 116291: Fulfillment exists with posting day in future period
●	Event-based POB 116289: Fulfillment exists with posting day in future period
●	Contract 100014: Open recon key exists in closed period 008/2022
●	Contract 100016: Open recon key exists in closed period 008/2022
●	Contract 100026: Open recon key found in period 011/2022
●	Contract 100027: Open recon key found in period 011/2022
●	Contract 100029: Open recon key found in period 011/2022
●	Contract 100035: Open recon key found in period 011/2022
●	Contract 100037: Open recon key found in period 011/2022

Figure 8.10 Errors in Migration of Open Reconciliation Keys

Errors in account determination can also appear, as shown in [Figure 8.11](#). These errors must be solved before migration of those contracts can be finished. The best way is to analyze which contracts have such errors, fix the problems, and reprocess the contracts before continuing the migration process.

● Account determination for contract liability failed for company code AU02	①
● Account determination for contract 100012 failed, see error log	
● Account determination for contract 100012 failed, see error log	
● Account determination for contract liability failed for company code AU02	②
● Account determination for contract 100017 failed, see error log	
● Account determination for contract 100017 failed, see error log	
● Account determination for contract liability failed for company code AU02	③
● Account determination for contract 100018 failed, see error log	
● Account determination for contract 100018 failed, see error log	
● Account determination for contract liability failed for company code AU02	④
● Account determination for contract 100019 failed, see error log	
● Account determination for contract 100019 failed, see error log	
● Account determination for contract liability failed for company code AU02	⑤
● Account determination for contract 100024 failed, see error log	

Figure 8.11 Account Determination Errors

8.4 Post-Migration Cleanup

Once migration is over, you'll see the results: contracts in table `FARR_D_CONTRACT` will have the proper values, and changes in the database tables will be made. Before business as usual can be continued, it's good practice to perform cleanup activities, which means running validation reports after completing migration. Validations are related mainly to performing data consistency checks by executing activities as described in SAP Note 2567106.

In some cases, the migration itself will throw an error that post-migration validation checks are in error, but it's good practice to run Transaction `FARR_CONTR_CHECK` once more to be sure no contracts are in error after migration is over.

In addition, it's a good idea to run a few simple scenarios that can be reversed (called health check testing or smoke testing) and verify that the system works according to expectations after migration. To perform a health check, you complete one cycle (or a few cycles) of the most often used or simple scenarios that can either be ignored or reversed. For example, you could create a simple sales order, try passing it to RAR, and see if all the enhancements are working like they were before migration and according to expectations. This gives a quick indication that the migration process was successful, and you can continue with regular business as usual.

8.5 Summary

Migration from CCM to OCM is a rather simple and straightforward type of migration because it's mainly technical and doesn't involve changes in data that is already in the contracts. However, to have a successful migration and benefit from improvements that come with OCM, it's essential that migration is properly planned and executed.

Key considerations for planning include the following:

- Do you have developments that would be impacted by technical changes (either by BAdIs that are made obsolete or changes in table structures)? This is by far the most important topic to be analyzed because it can be a driver for migration not to be performed at all.
- Can your business processes support OCM? If you're running POC scenarios or if your system has compound structures, it's not possible to include such contracts in migration. It's possible to have contracts running according to CCM and OCM in parallel in the system, but costs of maintenance for such a system are higher, so you should compare them with the potential benefits of having OCM.
- Will downtime be necessary during migration? Downtime isn't needed, but you can't process RAIs during the migration execution.
- Do you have events in the future that affect the best moment for migration execution?

All these and more are important factors that can lead to a successful migration to OCM if properly planned.

That wraps up our journey with the RAR functionality. However, before we reach the end of this book, we'll take a first look into the new event-based revenue recognition (EBRR) solution in the next chapter.

9 Event-Based Revenue Recognition

When looking at the portfolio of products that SAP offers for revenue recognition, it's easy to get confused: you have results analysis, revenue accounting and reporting (RAR), and the newest tool, event-based revenue recognition (EBRR). In this chapter, we'll offer high-level information about EBRR when it's the preferred solution for revenue recognition.

One of the shortcomings of RAR in SAP S/4HANA is dealing with certain business scenarios that don't arise directly from International Financial Reporting Standards (IFRS) 15 or Accounting Standards Codification (ASC) 606 requirements, but rather from an organization's business model. SAP has released a new functionality, *event-based revenue recognition (EBRR)*, which is better tailored to customer projects and sales orders. In this chapter, we'll introduce EBRR with a look at the background of the solution, its use in the sell-from-stock and customer project scenarios, and the role of revenue recognition keys.

9.1 Solution Background

As mentioned in [Chapter 1](#), an entity might face different issues when it comes to revenue recognition. Depending on the industry, this can be regarding fulfilling matching principle requests (reporting costs and revenue in the same period), allocating revenue to proper performance obligations (POBs), dealing with huge volumes of data, or constant contract modifications. In

In addition, changes in business models are occurring where more industries that were traditionally oriented to sales of goods are now trying to sell their products as services to ensure revenue flow over time.

We discussed in the previous eight chapters how the RAR solution from SAP can help entities in fulfilling these requirements. However, based on the recent SAP portfolio, RAR isn't the only solution that can be used for this purpose.

With EBRR, costs and revenues are posted as they occur and are immediately matched and posted. Looking at the SAP portfolio in [Table 9.1](#) (current as of time of writing), you see that SAP offers three main tools for revenue recognition. The oldest, classic tool is *results analysis*, which has been on the market for a long time and was built to measure the progress of project execution and make appropriate financial postings. In that sense, it evolved over time, and, even now, it's the go-to tool when it comes to projects and internal orders financial management of progress. Besides the variety of features available, results analysis is also very extendable: there are multiple enhancement points the customer can use to tailor it to the company's specific needs. Finally, being on the market for a while means that a significant pool of people is now proficient and possesses the knowledge to work with results analysis.

Features	Results Analysis	RAR	EBRR
----------	------------------	-----	------

Features	Results Analysis	RAR	EBRR
Five-step model	<p>No. Results analysis was initially built to track the financial progress of a project's delivery. There are no contracts or POBs.</p>	<p>Yes. This was built fully around IFRS 15 and ASC 606 requirements. The model is very extendable in terms of providing support for different business scenarios.</p>	<p>Yes. The latest versions support all five steps, including allocation of transactional price. However, it might be a challenge to support IFRS 15 requirements for specific business scenarios.</p>

Features	Results Analysis	RAR	EBRR
Parallel reporting	No. The purpose wasn't to support different reporting requirements. Some extensions are possible when it comes to projects scenarios.	Yes. One of the main reasons the tool was developed was to fill missing feature in old revenue recognition tools related to parallel reporting.	Yes. The latest versions support parallel reporting.
Projects/professional services scenario	Yes. Results analysis was made specifically for this purpose and can be extended in different ways to match most customer scenarios.	Limited. RAR needs integration with results analysis and supports cost-based, revenue-based, and classic contract management (CCM) methods for revenue recognition.	Yes. However, EBRR isn't extendable like classic results analysis because it's based on scenarios coming from the cloud.

Features	Results Analysis	RAR	EBRR
Over time revenue recognition	Results analysis can be customized to fulfill this requirement, but only when linked with a work breakdown structure (WBS) or internal order.	Yes. Native support is provided for the time-based method with different deferral methods.	Yes. Native support is provided for the over time revenue recognition.
Over time cost recognition	Results analysis can be customized to fulfill this requirement, but only when linked with a WBS or internal order.	No. RAR requires integration with results analysis.	Yes. Native support is provided.

Features	Results Analysis	RAR	EBRR
Point-in-time revenue/cost recognition	Results analysis can be customized to fulfill this requirement, but only when linked with WBS or internal order.	Yes. Different methods are available to support all kind of business requirements.	Limited. Some features are missing, such as link drop ship or call off orders fulfillment.
Documents bundling	No.	Yes. Native support is provided for contract combination.	Limited. The process needs to be designed around existing features.
Integration with external components	This can be built in results analysis.	Native support is provided and is extendable.	Limited. The process needs to be designed around existing features.

Table 9.1 Tools Available with Features

However, the idea of results analysis was always to be a tool used while integrated with Project System or internal orders. Of course, if a customer decides to use WBS as the account assignment object for all of their sales, it could be applied outside the classic

project systems environment. In reality, however, this would be too limiting an approach, so it isn't used in practice. Even if this would be possible, when it comes to IFRS 15 and ASC 606, the results analysis key wouldn't be able to satisfy the five-step model with creation of POBs or allocation of transaction price. In that sense, the tool can't be used for IFRS 15 reporting.

The next tool, RAR, is now close to a decade on the market, but it still can't be considered a veteran solution like results analysis. RAR was built as a tool to enable IFRS 15 reporting, so that's definitely a strong point. In addition, irrespective of the industry undergoing an IFRS 15 implementation, we can claim with confidence that RAR can be used to provide valid IFRS 15 results. There are also many implementations where RAR was positioned as one segment in even non-SAP environments where the main role was to do IFRS 15 reporting. To add to the point, RAR wasn't developed as a standalone tool, but as an upgrade to Transaction VF44, which essentially means the experience accumulated in an old revenue recognition tool was used in RAR and upgraded. Maybe the biggest advantage of RAR is that extensibility makes it very applicable in industries that are becoming more and more service oriented. If a company is thinking about switching business models where instead of selling machines as standalone units, they will merge them with some services (e.g., maintenance, operations) and bill them based on some measurements over a period of time, RAR supports these scenarios natively (keeping in mind that some modifications in business processes are performed to reflect business change).

But there are business scenarios that aren't so easily integrated with RAR. When it comes to the professional services industry, RAR isn't calculating the percentage of completion (POC) by itself: it's being done by results analysis, and RAR is just taking over values. Even in this case, it's limited to three results analysis key methods (cost-based method, revenue-based method, and CCM). Most customers won't find this limiting because POC is mainly calculated

on these methods, but there are business scenarios such as time and material that require adjustments and tweaks to be enabled in RAR. Even once they are enabled, there are limitations that businesses would need to respect when working with RAR. In addition, the extensibility that RAR offers is a burden in some cases: for some businesses, it might turn out to be complex and too lengthy of an implementation without providing tangible benefits.

Last, we come to EBRR, the newest of all tools for revenue recognition. As mentioned previously, natively, EBRR was developed on SAP S/4HANA Cloud, public edition and later ported to other versions of SAP S/4HANA. That makes it the most modern solution, but at same time, it comes with a number of predefined scenarios that reveal its main benefit: if the user can fit into those scenarios, then implementation is rapid without too much difficulty. In addition, some scenarios that are pain points to RAR are natively supported by EBRR, hence limiting the number of necessary extensions and enhancements. Being modern means EBRR also makes full use of the latest features from SAP S/4HANA, such as direct integration with the Universal Journal. This saves time used for reconciliation and makes the overall closing process much faster and transparent. With the implementation of EBRR, customers can achieve further simplifications: results analysis keys become obsolete. This means if the customer is able to fit into the standard scenarios offered by EBRR, they might achieve further reduction in total cost of ownership (TCO) by simplifying other modules that are used.

The biggest advantage is at same time the biggest limitation of EBRR. The solution isn't (or better said, isn't yet) extendable like RAR or results analysis. So, if some extensions are needed by nature of the business, it might turn out that without a specific business process design, requirements simply can't be fulfilled. In addition, some areas are yet to be developed, such as support of different contract combinations, modifications of contracts, or

integration with external systems. EBRR is simply not ready to take over tasks that are normally done by RAR. Again, EBRR is a very new tool, which means the number of people working with it isn't as high. All these factors need to be taken into consideration before a company opts for this solution for revenue recognition reporting.

EBRR is focused on solving revenue reporting issues that are built around different scenarios, as we'll discuss in the next sections.

9.2 Sell-from-Stock Scenario

The *sell-from-stock* scenario covers all steps from creation of sales order to delivery of goods to billing based on delivery. These steps are followed by EBRR and margin reporting based on postings made. [Figure 9.1](#) shows the sell-from-stock scenario process flow alongside its integration with EBRR.

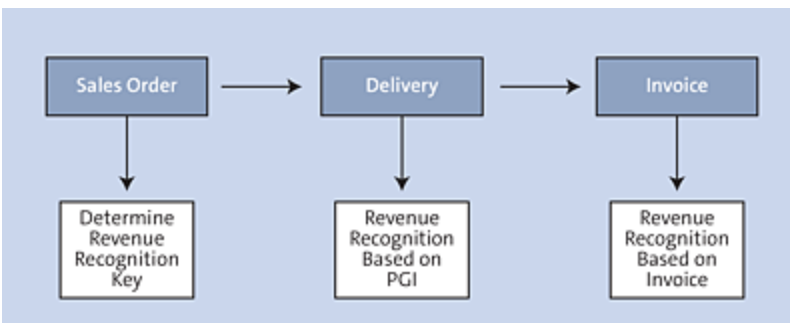


Figure 9.1 Process of Sales Integrated with EBRR

The standard process of the sell-from-stock scenario, as shown in [Figure 9.2](#), starts with the creation of a sales order based on which delivery and post goods issue (PGI) are done. Like RAR, an EBRR key is determined already at the moment of sales order creation.

Based on the method, either revenue will be recognized at PGI or when the invoice is sent to the customer. In all cases, costs and revenues are posted at the same time, ensuring that the matching principle rule is fulfilled.

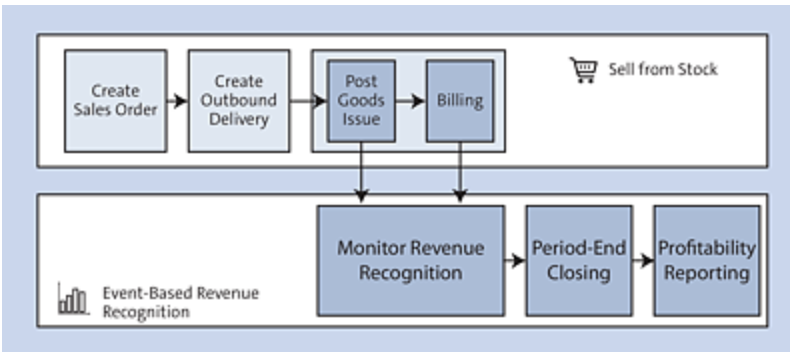


Figure 9.2 EBRR Based on Sell from Stock

In this scenario, when goods are posted from stock, revenue is calculated based on the quantity delivered, and both revenue and costs are accrued and posted. Reporting is available here and in margin analysis, and there is no need to take additional steps for settlement.

Revenue recognition is controlled by the revenue recognition key assigned in Customizing (see [Section 9.4](#) for more information). These keys allow different scenarios to be executed:

- Recognize revenue based on invoices
- Recognize costs and revenues at moment of final delivery, until then, defer all (partial deliveries will be deferred)
- Override EBRR logic: costs are recognized at PGI and revenue invoicing
- Recognize costs based on planned costs from the material master, and recognize revenue based on conditions in sales document
- Recognize revenue based on invoicing, and reduce the accrued revenue directly

It's very important to mention that EBRR supports different treatment of revenue and cost recognition based on the ledger: it's possible to implement different ways for different local and global reporting requirements.

9.3 Sales-Oriented Scenario for Customer Projects

Projects are structured forms of certain activities and tasks to be performed to achieve certain outcomes. In SAP, projects are maintained in a specific module called Project System. There are many different kinds of projects, but two types are the most dominant: projects as cost collectors with or without capitalization and revenue-related projects. The focus of EBRR is on customer projects.

Certain elements are common for all projects and need to be fulfilled so that EBRR can be used as the tool for revenue recognition:

- **WBS elements**

The project has to be in a structured form, and each task must be represented as a WBS element. This element is used as an account assignment object and can have a results analysis key assigned to it. This results analysis key determines how POC will be calculated and how revenue will be recognized at the end. Keep in mind that WBS elements will either inherit the results analysis key defined in WBS elements or the revenue recognition key assigned in EBRR. [Figure 9.3](#) shows an example of results analysis keys created in one organization. Usually, they are used for pure POC calculation, but, in some cases, they can be also used instead of settlements. That's the reason many organizations try to standardize results analysis keys as part of their SAP S/4HANA transformation.

RA Key	RA	Cap.ca	EBRavRec	ReckeyArea	Result Analysis Description
PG0001	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA POC
PG0002	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Quantity Based SKP
PG0003	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA MTS Rev bud - No WFO at FMBL
PG0004	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Production
PG0005	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Projects - OCM
PG0007	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Revenue based, All Items (K) - No WFO at FMBL
PG0008	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Revenue based - WFO at FMBL
PG0009	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Quantity based delivery
PG0010	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Revenue based, Sub positions (B) - No WFO at FMBL
PG0011	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		PG Warranty RA
PG0020	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Resource-related billing
PG0021	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Straight line POC
PG0022	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Straight line POC billing plan
PG0023	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA ZHWP
PG0024	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA POC

Figure 9.3 Example of Different Results Analysis Keys Available

- **Billing indicator**

As mentioned, WBS elements can be used either as cost collectors or as revenue objects. As revenue objects, this revenue needs to be allowed to be posted to the WBS element. This is enabled by applying the **Billing Element** checkbox under **Operative indicators** in the WBS definition, as shown in [Figure 9.4](#).

Responsibilities	Operative indicators
Pers.Resp.No.: 71663714	Planning Element: <input checked="" type="checkbox"/>
Applicant no.: <input type="text"/>	Acct asst elem.: <input checked="" type="checkbox"/>
Resp. cost ctr: <input type="text"/>	Billing Element: <input type="checkbox"/>
Req. cost center: <input type="text"/>	Grouping WBS element: <input type="checkbox"/>
Req. co. code: <input type="text"/>	

Figure 9.4 Definition of Billing in WBS Element

- **Order assignment**

Revenue usually comes through the sales process, which starts with a sales order (there are other documents to be used at the start, but for simplicity, let's limit it to sales orders). The WBS element needs to be assigned to the sales order for revenue to be posted to it. [Figure 9.5](#) shows an example **WBS element** added to the sales order under the **Account Assignment** tab.

The screenshot shows the SAP Account Assignment interface. At the top, there are navigation buttons (back, forward, search, etc.). Below that, the 'Sales Document Item' is set to '10' and the 'Material' is 'YXE01ES225'. The 'Item category' is 'ZPMT' and the 'Consulting Services' are 'Ellipse PSO_PSO_TE'. A menu bar includes 'Sales A', 'Sales B', 'Contract data', 'Shipping', 'Billing Document', 'Conditions', 'Account Assignment' (which is selected), and 'Schedule lines'. The 'Account assignment' section contains the following fields: 'Business Area' (empty), 'Profit Center' (A002285140), 'WBS element' (AUS-00000188.02.01.06), 'Order' (empty), and 'Profit Segment' (0).

Figure 9.5 Assignment of a WBS Element to Sales Order

While creating WBS elements, you need to understand which statuses are important for EBRR. WBS elements can go through different statuses, but two are most important:

- **Completed (TECO status)**

This means that all revenue and costs are recognized, and no further calculation of POC and revenue recognition will occur. However, invoicing and posting to the project are still possible.

- **Closed**

The prerequisite for this status is that there are no more balance sheet items in the project posted. Once the project is set to this status, no more postings can happen with WBS as the account assignment object.

Once a project with a WBS structure is properly created, you next execute steps that are valid for all sales-related projects. The first step is the planning process. In some cases, planning is a necessary step (when revenue to be recognized is calculated as a comparison between planned and actual values), but, in some cases, it might not be necessary (time and material projects).

In this example, you're creating sales orders and assigning a WBS element as the account assignment object. Note that

the sales order–project relationship must be 1:1 as must the sales order item–WBS element.

EBRR provides specific SAP Fiori apps to monitor revenue recognition. In this case, you'll access the Event-Based Revenue Recognition – Sales Order app to analyze revenue recognition postings. The main purpose for the app is to provide an overview of all postings related to a specific event; however, users can also trigger additional postings.

The sales order item category (which can be found in every line item in a sales document) is called the *contract type* in EBRR, and it determines how a project will be billed, which has a direct connection to the revenue recognition method. Three basic methods are used for billing, as defined already in SAP S/4HANA Cloud, public edition:

- **Time and material billing**

This is the standard approach when a customer is billed based on consumption of that WBS element according to an agreed price. The customer is charged for the costs that might occur up front. In this case, revenue will be recognized based on time spent, and sales price and billed amount are deferred.

- **Fixed-price billing**

In this scenario, the customer is billed based on a billing plan defined in the sales document. Revenue can be recognized in multiple ways, which usually is the cost-based method where we're comparing actual versus planned costs, and then the calculated POC is applied to revenue. The difference between billed amount and recognized revenue is accrued.

- **Periodic service billing**

This is widely used in cases where the customer is billed in periods that are again defined in the billing plan (quarterly/half yearly/yearly). Each billing plan item has a period assigned (the main difference compared to fixed-price billing), and revenue is recognized based on the period and amount billed.

[Figure 9.6](#) shows the high-level process of recognizing revenue through projects. The first step is planning, in which you'll plan both costs and revenues. Now, depending on the method for calculating POC, you'll use either costs or revenue as the basis for calculation.

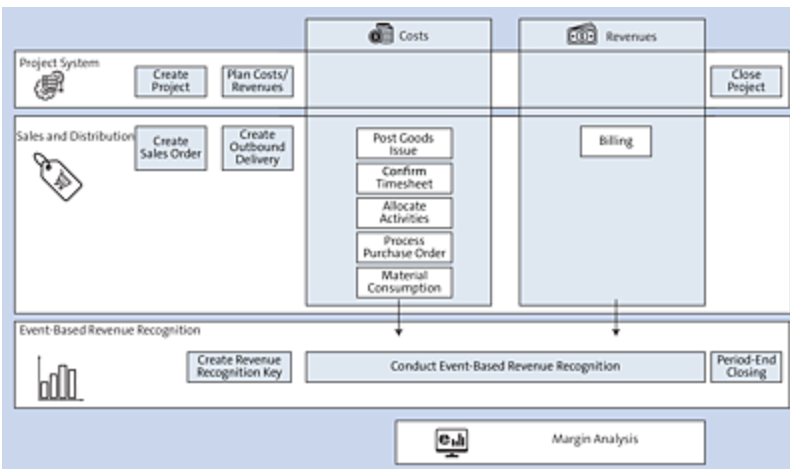


Figure 9.6 Project-Related Process for Revenue Recognition

The next step is posting costs, which can come to the project as either direct costs by, for example, material issued to the project, or as different types of activity allocation performed on a WBS element (this is often the case in professional services projects where time is entered and multiplied by unit cost).

At the milestone or period completion, billing is performed. Billing can be done as regular billing or some sort of resource-related billing. Once both steps are completed, the WBS element will contain all elements so revenue recognition can be calculated.

The process is finalized in EBRR with a separate app: Run Revenue Recognition - Sales Orders. An overview of postings will be displayed, and you can run simulation before executing real postings.

9.4 Revenue Recognition Keys

Defining and assigning revenue recognition keys is a main part of configuring how revenue will be recognized in EBRR. The revenue recognition key defines the valuation of orders during period-end closing and allows you to determine how revenue is calculated during month-end closing. This step comes after defining the sources that represent lists of cost elements to be considered in calculating revenue recognition. For project-related revenue recognition, revenue recognition keys can be freely definable, but need to be linked to one of the supported revenue recognition methods.

Each recognition key needs to have an assignment rule applied. These rules define how revenue and costs will be recognized (e.g., cost-based POC, revenue-based POC). The next step is assignment of sources and assigning to each source how it will be used. For example, whether to recognize cost of goods sold (COGS) as incurred or amortize it over time is defined in the assignment of sources to the recognition key. At the end, the revenue recognition method is assigned to the ledger and company code, and then it's ready to be used.

9.5 Summary

In this chapter, we provided an introductory look at EBRR. As the new kid on the block, EBRR brings its own set of benefits and limitations when it comes to deciding whether it will be used for revenue recognition processes.

Having that said, opting for EBRR or RAR depends on the specific situation. It might not be so much related to the industry, for example, but to the extent to which customers can standardize their own processes and make them fit into the standard EBRR brings. Remember, EBRR came from the cloud, which makes it relatively fast for implementation. However, that cloud-oriented mindset also means that the customer expects to fit as much as possible to the standard, which is brought about by scenarios. If that isn't possible, extensions of EBRR might become challenging, and fitting it into existing business processes may even be impossible.

If business scenarios require customization or the client's process of selling goods/services to customers is very diverse and complex, RAR might be the go-to solution. The list of extensions that can be applied is significant, while at same time, the tool remains able to be the main revenue reporting source. Of course, the cost is obvious: the more enhancements to the standard, the more complex implementation becomes.

For these reasons, customers need to be aware of the features and limitations that accompany all of these solutions. For more details on EBRR, take a look at

Introducing Event-Based Revenue Recognition (EBRR) with SAP S/4HANA (SAP PRESS, 2023; www.sap-press.com/5679).

Important Business Add-Ins

[Table A.1](#) provides a list of business add-ins (BADIs) that are relevant for revenue accounting and reporting (RAR). While not all of these BADIs may be necessary in your project, we've provided this list in case any developer may need it.

Enhancement Spot	BAdI	Description
FARR_IC_SD	FARRIC_BADI_ORDER	Used to process sales order information and transfer it to revenue accounting via its interface structures. Also, used to include nonstandard fields and modify values of customer-specific fields.
	FARRIC_BADI_DELIVERY	Used to process delivery and goods issue information and transfer it to RAR's Adapter Reuse Layer (ARL). Also includes and modifies customer fields.
	FARRIC_BADI_INVOICE	Used to process invoice information and transfer it to RAR's ARL.

Enhancement Spot	BAdI	Description
FARR_ARL	FARR_BADI_CONTRACT_COMBINATION	Used to override the standard contract combination logic and design your own logic for contract combination.
	FARR_BADI_RAI0	Used to modify and validate revenue accounting items (RAIs) with the raw status.
	FARR_BADI_RAI2	Used to modify and validate the content of RAIs in the raw status before they are transferred to the processable status.
	FARR_BADI_RAI4	Used to modify and validate the content of RAIs in the processable status before they are moved to the processed status.
FARR_COAC_DERIVE_TM_ATTR	FARR_BADI_COAC_DERIVE_TM_ATTR	Used to apply your own processing logic when you need to derive the duration of the performance obligation (POB) for capitalized costs. The implementation of this BAdI is optional.

Enhancement Spot	BAdI	Description
FARR_CHANGE_MODE_DETERMINATION	FARR_CHANGE_MODE_DETERMINATION	Used to determine the change type of contract changes for POBs. The change type includes change of estimates, contract modification, and an attribute change from the inception date without price reallocation.
FARR_DEFERRAL_METHOD	FARR_BADI_DEFERRAL_METHOD	Used to handle fulfillment events that are passed from the operational system. When a fulfillment event occurs, this BAdI takes the event date and returns the fulfillment items to be created. Implementation of this BAdI will work when the fulfillment type of the corresponding deferral item is "time".
FARR_ES_POSTING	FARR_POSTING_ENHANCEMENT	Used to enable changing the standard fields during posting to financial accounting.

Enhancement Spot	BAdI	Description
FARR_DISTRIBUTE_INVOICE	FARR_BADI_DISTRIBUTE_INVOICE	Used to support posting contract asset/contract liability (CA/CL) (unbilled receivable and deferred revenue).
FARR_DIST_NET_CLCA_AMT_TO_POB	FARR_DIST_NET_CLCA_AMT_TO_POB	If you're using a fixed rate, then this BAdI distributes contract CA/CL or unbilled/deferred revenue to the POB. It also checks that the distributed values are the same as the total value.
FARR_CHECK_ES	FARR_BADI_CHECK	Used to simulate the operational document in the integration component to compare records. Useful in reconciliation with RAR.
	FARR_BADI_CHECK_RAI_PP_MON	Used for application log profile enhancement in the reconciliation report FARR_CHECK_CONS.
	FARR_BADI_CHECK_SELECTION	Used to select the operational documents in the integration component. Useful in reconciliation with RAR.

Enhancement Spot	BAdI	Description
	FARR_BADI_CHECK_SYMESSAGE	Used for overwriting of system messages in the reconciliation report FARR_CHECK_CONS.
FARR_TM_REMAINING_PERCENTAGE	FARR_BADI_TM_REMAINING_PERC	Used to calculate the remaining percentage and the change trend of the remaining duration of time-based POBs. The remaining percentage and the remaining duration change trend are used when you perform a contract modification on a contract with a time-based POB.
FARR_COMPOUNG_GROUP_POB_ATTR	FARR_BADI_COMPND_GRP_POB_ATTR	Used for the determination of compound group POB attributes.
FARR_DTMN_POB_4_DEF_METHD_BADI	FARR_BADI_DTMN_POB_DEF_METHOD	Used for the determination of POBs for the deferral method to adjust fulfillment.
FARR_LINKED_POB_ACCT_ASGT	FARR_BADI_LINKED_POB_ACCT_ASGT	Used to change the account assignment of the linked POB.
FARR_LOG_POB_DATA	FARR_BADI_LOG_POB_DATA	Used to log POB data.
FARR_POB_CUST_VALIDATION	FARR_BADI_POB_CUST_VALIDATION	Used for POB customizing validation.

Enhancement Spot	BAdI	Description
FARR_CALC_LIAB_ASSET	FARR_BADI_CALC_LIAB_ASSET	Used to calculate liabilities and assets.
FARR_ENH_SPOT_BIZ_RECON	FARR_BADI_BIZ_RECON	Used to integrate third-party sender components for the reconciliation of the operational data with RAR.
FARR_PRODUCTIVE_CLEANUP	FARR_BADI_PRODUCTIVE_CLEANUP	Used in the cleanup and reverse the productive data program of RAR.
ES_FARR_FOUNDATION	FARR_ACPR_BUKR_CHECKS	Used to define custom specific checks for validating the status switch for company code and accounting principle validation.
FARR_WORKLIST_SET_REVIEW	FARR_BADI_SET_REVIEW_WORKLIST	Used to implement custom logic when the user clicks Mark as Reviewed on a regular monitoring worklist.
FARR_DS_CLEAR_DELDEFITM	FARR_BADI_CLEAR_DELDEFITM	Used to check if table FARR_D_DELDEFITEM should be cleared.

Table A.1 List of BAdIs

The Authors



Sreten Milosavljević has worked as a senior SAP finance consultant for more than 20 years. Thanks to the perfect storm of International Financial Reporting Standards (IFRS) 9, IFRS 15, and IFRS 16, since 2016, his primary focus has been on working with revenue accounting and reporting and contract lifecycle management. During this time, he gained experience with most of the industries that are impacted by the implementation of either IFRS 15 or Accounting Standards Codification (ASC) 606, including telecommunications, healthcare, manufacturing, and high-tech. Sreten is currently involved in solving complex topics for a variety of customers, especially regarding integration points between different systems.



Swayam Prabha Shankara has been a technical consultant for more than 16 years. Over this period, she has worked with different technologies to apply the latest approaches. She has been working with revenue accounting and reporting in the SAP space since 2017, beginning with a project for a major telecommunications customer that had

complex issues related to data volume and performance. She has continued working on different revenue accounting and reporting projects, updating her skills for SAP HANA, core data services (CDS) views, and system design and development.

Index

↓ **A** ↓ **B** ↓ **C** ↓ **D** ↓ **E** ↓ **F** ↓ **G** ↓ **H** ↓ **I** ↓ **J** ↓ **K** ↓ **L** ↓ **M** ↓ **N**
↓ **O** ↓ **P** ↓ **R** ↓ **S** ↓ **T** ↓ **U** ↓ **V** ↓ **W**

A ↑

ABAP Dictionary [→ Section 2.3]

ABAP-Managed Database Procedures (AMDPs)
[→ Section 6.4]

ABC programs [→ Section 1.4] [→ Section 6.2]

Acceptance date [→ Section 5.3]

Account assignment [→ Section 5.1] [→ Section 6.1]

Accounting principle [→ Section 5.1] [→ Section 6.1]
[→ Section 8.3]

company codes [→ Section 5.1]

number ranges [→ Section 5.1]

parallel [→ Section 5.1]

Accounting Standards Codification (ASC) 606
[→ Section 1.1]

Actual exchange rate method [→ Section 5.1]

Adapter Reuse Layer (ARL) [→ Section 1.4] [→ Section 2.3] [→ Section 3.1] [→ Section 4.1]

Administration [→ Chapter 7]

Allocation [→ Section 1.2] [→ Section 5.2] [→ Section 5.5] [→ Section 8.2]

effect [→ Section 5.1]

engine [→ Section 5.5]

Allow method [→ Section 4.6]

App Finder [→ Section 6.4]

Append structure [→ Section 2.3]

sets [→ Section 2.3]

Application [→ Section 4.5] [→ Section 4.5] [→ Section 5.2]

assign [→ Section 4.5]

Application log [→ Section 6.2]

Application server instance [→ Section 7.1]

B ↑

Batch server group [→ Section 7.1]

Bill of materials (BOM) [→ Section 5.2]

Billable amount [→ Section 5.1]

Billing [→ Section 2.2]

indicator [→ Section 9.3]

Business add-in (BAdI) [→ Appendix A]

FARR_BADI_CONTRACT_COMBINATION [→ Section 5.4]

FARR_BADI_DEFERRAL_METHOD_V2 [→ Section 8.2]

FARR_BADI_PRICE_ALLOCATION [→ Section 8.2]

FARR_BADI_RA10 [→ Section 4.6] [→ Section 5.1]

FARR_BADI_RA12 [→ Section 4.6] [→ Section 5.1]
[→ Section 7.3]

FARR_BADI_RA14 [→ Section 4.6]

FARR_EXTENDED_CHECK [→ Section 7.3]

FARR_POSTING_ENHANCEMENT [→ Section 6.1]

FARRIC_BADI_ORDER [→ Section 3.1]

IF_FARR_BADI_DETNI_IP_VERSION [→ Section 3.2]

OCM [→ Section 8.2]

signature and implementation [→ Section 4.6]

Business Rules Framework plus (BRFplus) [→ Section 1.4] [→ Section 4.5]

account assignment [→ Section 6.1]

architecture [→ Section 4.5]

expert mode [→ Section 4.5]

OCM [→ Section 8.2]

POB types [→ Section 5.2]

RAR integration [→ Section 4.5]

setup [→ Section 4.5]

transport [→ Section 5.2]

Business support services (BSS) [→ Section 1.3]

C ↑

Calculation method [→ Section 5.1]

Call-off order [→ Section 5.3]

Capital sales contract [→ Section 1.3]

Child job [→ Section 7.1] [→ Section 7.1]

Classic contract management (CCM) [→ Section 1.4]
[→ Section 2.4] [→ Section 5.1]

contract modifications [→ Section 8.2]

migrate to OCM [→ Section 8.1]

reports [→ Section 6.4]

Classic inbound processing [→ Section 3.2]

CleanUp program [→ Section 7.3]

Client [→ Section 2.1]

Cloud [→ Section 2.1]

Company code check [→ Section 4.6]

Compound structure [→ Section 5.2]

Condition item [→ Section 2.3] [→ Section 4.1]
[→ Section 5.3]

Condition type [→ Section 4.1] [→ Section 5.1]
[→ Section 5.3] [→ Section 6.1]

FX differences [→ Section 5.1]

Conservatism [→ Section 5.1]

Consideration payable to customer [→ Section 1.2]

Consistency check [→ Section 3.2] [→ Section 4.4]

Consistency monitor [→ Section 4.4]

Contract [→ Section 1.2] [→ Section 2.3]

balance report [→ Section 6.4]

contract ID [→ Section 2.3]

currency [→ Section 5.1]

freeze [→ Section 5.4]

identify [→ Section 1.2]

shift [→ Section 6.1]

suspensions [→ Section 8.2]

terminate [→ Section 1.4] [→ Section 5.4]

[→ Section 8.1]

type [→ Section 9.3]

Contract assets/contract liabilities (CA/CL) [→ Section 5.1] [→ Section 5.1] [→ Section 6.1] [→ Section 6.1]

calculation [→ Section 5.1] [→ Section 5.1]

[→ Section 6.2]

CL netting [→ Section 5.1]

contract level [→ Section 5.1]

POB level [→ Section 5.1] [→ Section 5.1]

Contract Balance Reclassification app [→ Section 6.4]

Contract combination [→ Section 1.3] [→ Section 5.4]

automated [→ Section 5.4]

perform [→ Section 5.4]

quick combine [→ Section 5.4]

Contract management [→ Section 5.1]

activate OCM [→ Section 8.2]

select type [→ Section 5.1]

Contract modification [→ Section 5.1] [→ Section 5.4]
[→ Section 5.4]

prospective versus retrospective [→ Section 5.4]

Contract Search app [→ Section 5.3] [→ Section 5.3]

Contract-by-contract basis [→ Section 1.2]

Control transfer [→ Section 1.2]

Convergent invoicing [→ Section 2.2]

Core data services (CDS) view [→ Section 6.4]

Cost object controlling [→ Section 5.3]

Cost of goods sold (COGS) [→ Section 6.3]

Cost recognition [→ Section 5.1] [→ Section 5.2]

Costing-based profitability analysis [→ Section 6.3]

Cost-plus method [→ Section 1.2]

Credit posting [→ Section 6.1]

Customer [→ Section 1.2]

Customer acceptance clause [→ Section 5.3]

Customer field [→ Section 2.3] [→ Section 3.3]
[→ Section 4.6] [→ Section 4.6] [→ Section 6.1]
select [→ Section 2.3]

Customer invoice fulfillment [→ Section 5.3]

Customer projects [→ Section 9.3]

Customer relationship management (CRM) [→ Section
1.3] [→ Section 2.2]

D ↑

Data cleanup [→ Section 7.3]
customizing [→ Section 7.3]

Data hub [→ Section 2.2]

Data model [→ Section 2.3]

Data validation [→ Section 7.3] [→ Section 7.3]
check [→ Section 4.4] [→ Section 4.4]
monitor [→ Section 7.3]

Day-based contract modification [→ Section 1.4]

Debit posting [→ Section 6.1]

Debit/credit indicator [→ Section 6.1]

Decision table [→ Section 4.5] [→ Section 4.5]
[→ Section 5.2]
change [→ Section 4.5]
revenue posting [→ Section 6.1]
settings [→ Section 5.2]

simplified GUI [→ Section 4.5]
Deferral category [→ Section 5.3]
Deferral method [→ Section 5.3]
 custom [→ Section 5.3]
 freeze periods [→ Section 5.4]
Deferral table [→ Section 5.3]
Direct posting [→ Section 2.4]
Disaggregation of revenue [→ Section 6.4]
Disclosure [→ Section 6.4] [→ Section 6.4]
Discount [→ Section 1.2]
Distinction [→ Section 1.2]
Due date [→ Section 5.1]
Dynamic processing flow [→ Section 4.6]

E ↑

Easy Enhancement Workbench (EEW) [→ Section 2.3]
Enforceability [→ Section 1.2]
Enhancement spot [→ Appendix A]
Enrich method [→ Section 4.6] [→ Section 4.6]
[→ Section 5.1] [→ Section 7.3]
Error category [→ Section 7.3] [→ Section 7.3]
Error resolution [→ Section 4.4]

categories [→ Section 7.3] [→ Section 7.3]

error table [→ Section 7.3]

with Transaction FARR_CONTR_CHECK [→ Section 4.4]

without Transaction FARR_RAI_MON [→ Section 4.4]

Event type [→ Section 5.3] [→ Section 5.3] [→ Section 5.3]

Event-based POB [→ Section 5.3] [→ Section 5.3]

Event-based revenue recognition (EBRR) [→ Section 1.4] [→ Section 9.1] [→ Section 9.1]

comparison [→ Section 9.1]

revenue recognition keys [→ Section 9.4]

Event-Based Revenue Recognition – Sales Order app [→ Section 9.3]

Exchange rate method [→ Section 5.1]

Exempted item [→ Section 4.2] [→ Section 7.3] [→ Section 7.4]

Exemption type [→ Section 4.2]

Expression [→ Section 4.5] [→ Section 4.5] [→ Section 5.2] [→ Section 6.1]

Extension [→ Section 2.3]

define [→ Section 3.3]

FARR_RAI_MON [→ Section 4.3]

populate [→ Section 3.3]

Extension include [→ Section 2.3] [→ Section 3.3]

Extract, transform, load (ETL) [→ Section 1.3]
[→ Section 2.2]

F ↑

Financial accounting [→ Section 6.2]

reconciliation [→ Section 6.4]

Financial document [→ Section 6.2]

analysis [→ Section 6.4]

Fixed exchange rate method [→ Section 5.1]

Fixed-price billing [→ Section 9.3]

Flow control table [→ Section 4.6]

Foreign currency [→ Section 5.1]

Fulfillment event [→ Section 3.1]

Fulfillment item [→ Section 2.3] [→ Section 4.1]

Fulfillment type [→ Section 5.3]

Function [→ Section 4.5]

Function module [→ Section 7.1] [→ Section 7.1]

G ↑

General ledger account [→ Section 6.4]

General ledger mapping [→ Section 7.2]

Globally unique identifier (GUID) [→ Section 6.1]

Goods delivery [→ Section 5.3]

Goods issue fulfillment [→ Section 5.3]

H ↑

Header ID [→ Section 2.3] [→ Section 4.1] [→ Section 7.1]

Hierarchy [→ Section 2.3]

I ↑

Impairment effect [→ Section 5.4]

Implementation class [→ Section 5.1]

Inbound processing [→ Section 3.1] [→ Section 3.2]
usage parameters [→ Section 3.2]

Incoterm [→ Section 1.2] [→ Section 1.3]

Index [→ Section 2.3]

Inflight check [→ Section 4.4] [→ Section 7.3]

Input costs method [→ Section 1.3]

Input method [→ Section 1.2]

Instrument [→ Section 1.3]

Integration [→ Section 2.2]

billing [→ Section 2.2]

CRM [→ Section 2.2]

non-SAP systems [→ Section 2.2]

profitability analysis [→ Section 6.3]

results analysis [→ Section 5.3]

sales and distribution [→ Section 2.2]

Interface component [→ Section 2.3] [→ Section 3.2]

activate [→ Section 3.2]

list [→ Section 3.2]

Interface generation [→ Section 3.2]

International Financial Reporting Standards (IFRS)

[→ Section 1.1]

Five-step model [→ Section 1.2]

IFRS 15 [→ Section 1.1] [→ Section 1.2] [→ Section 1.4] [→ Section 2.1] [→ Section 6.1]

industry impact [→ Section 1.3]

Invoice [→ Section 6.1]

Invoice item [→ Section 2.3] [→ Section 4.1]

Invoice reversal [→ Section 6.1] [→ Section 6.1]

Item category [→ Section 3.1]

J ↑

Job monitor [→ Section 6.2]

Job server group [→ Section 7.1]

create [→ Section 7.1]

K ↑

KEYPP field [→ Section 4.1]

L ↑

Lease contract [→ Section 1.3]

Leasing [→ Section 1.3]

Life sciences [→ Section 1.3]

Logical system [→ Section 2.3] [→ Section 4.1]

Loyalty points [→ Section 1.3]

M ↑

Main item [→ Section 2.3] [→ Section 4.1]

Main program

child job [→ Section 7.1]

parent program [→ Section 7.1]

Manage Revenue Contracts app [→ Section 6.4]

Manual fulfillment [→ Section 5.3]

Manufacturing [→ Section 1.3]

Margin analysis [→ Section 1.4] [→ Section 6.3]

Market observable price [→ Section 1.2]

Mass process [→ Section 2.3]

Mass transfer [→ Section 2.3]

Matching principle [→ Section 9.2]

Message capturing [→ Section 7.3]

Message class [→ Section 7.3]

Microsoft Excel [→ Section 5.2]

Migration [→ Section 8.1]

activities [→ Section 8.3]

business case [→ Section 8.1]

cleanup [→ Section 8.4]

errors [→ Section 8.3]

perform [→ Section 8.3]

preparation [→ Section 8.2]

Modularization [→ Section 7.1]

Monitor Revenue Contract app [→ Section 6.4]

Month-end closing [→ Section 6.2]

N ↑

Navigator [→ Section 7.4]

set up [→ Section 7.4]

Noncash consideration [→ Section 1.2]

Number range [→ Section 5.1]

O ↑

Operational report [→ Section 6.4] [→ Section 6.4]

Operations support systems (OSS) [→ Section 1.3]

Optimized contract management (OCM) [→ Section 1.4] [→ Section 2.4] [→ Section 5.1]

BAdIs [→ Section 8.2]

contract modifications [→ Section 8.2]

migration [→ Section 8.1]

process changes [→ Section 8.2]

reports [→ Section 6.4]

Optimized inbound processing (OIP) [→ Section 1.4] [→ Section 2.3] [→ Section 2.4] [→ Section 3.2]

activate [→ Section 3.2]

tables [→ Section 3.2]

Order assignment [→ Section 9.3]

Order item [→ Section 2.3] [→ Section 4.1] [→ Section 4.1]

process [→ Section 4.1]

Original item [→ Section 4.1]

Output cost method [→ Section 1.3]

Output method [→ Section 1.2]

Over time revenue recognition [→ Section 1.2] [→ Section 1.3] [→ Section 5.3]

calculation [→ Section 5.3]

P ↑

Package size [→ Section 7.1]

Packaging [→ Section 7.1]

Parallel accounting principle [→ Section 5.1]

Parallel buffering [→ Section 6.1]

Parallel processing [→ Section 7.1]

Parallel processing framework (PPF) [→ Section 2.4]

[→ Section 4.1] [→ Section 7.1]

custom [→ Section 7.1]

function module [→ Section 7.1]

packaging [→ Section 7.1]

program execution [→ Section 7.1]

selection screen [→ Section 7.1]

standard [→ Section 7.1]

Parent program [→ Section 7.1]

processing [→ Section 7.1]

Percentage of completion (POC) [→ Section 1.2]

[→ Section 1.3] [→ Section 5.3]

event [→ Section 5.3]

Perform [→ Section 7.1]

Performance [→ Section 6.1] [→ Section 7.1] [→ Section 7.1]

Performance obligation (POB) [→ Section 1.2]

[→ Section 1.3] [→ Section 5.2]

allocation data [→ Section 5.2]

BRFplus [→ Section 5.2]

compound [→ Section 5.2]
create time-based POB [→ Section 5.3]
define types [→ Section 5.2]
fulfillment data [→ Section 5.2]
fulfillment type [→ Section 5.3]
manage [→ Section 5.3]
POB ID [→ Section 2.3]
posting categories [→ Section 6.1]
remaining [→ Section 6.4]
table FARR_D_DEFITEM [→ Section 5.3]
time based [→ Section 1.4]

Periodic service billing [→ Section 9.3]

Perpetual contract [→ Section 1.3]

Point-in-time revenue recognition [→ Section 1.2]
[→ Section 5.3]

Portfolio approach [→ Section 1.2]

Posting [→ Section 6.1] [→ Section 7.2]

ABC programs [→ Section 6.2]

account assignment [→ Section 6.1]

categories [→ Section 6.1] [→ Section 6.1]

custom [→ Section 7.2]

extend program [→ Section 7.2]

GUID [→ Section 6.1]

keys [→ Section 6.1] [→ Section 6.1] [→ Section 6.2]

optimization [→ Section 6.1]
run [→ Section 6.2]
settings [→ Section 6.1]

Posting period [→ Section 5.1]
status [→ Section 5.1]

Postpaid contract [→ Section 1.3]

Postponed item [→ Section 3.2] [→ Section 4.2]

Predecessor item [→ Section 3.2] [→ Section 4.1]
process [→ Section 4.1]

Prepaid contract [→ Section 1.3]

Presentation method [→ Section 5.1]
calculations [→ Section 5.1]

Price allocation [→ Section 5.5]

Proactive measure [→ Section 4.4]

Probability [→ Section 1.2]

Processable item [→ Section 2.3] [→ Section 4.1]
custom [→ Section 4.6]
exempt [→ Section 7.4]

Processed item [→ Section 2.3]
custom [→ Section 4.6]

Processing block [→ Section 7.1]

Processing method [→ Section 4.1]

Profitability analysis [→ Section 6.3]
Program enhancement [→ Section 3.2]
Project System [→ Section 9.3]
Proof of delivery [→ Section 3.1]
 fulfillment [→ Section 5.3]
Prospective change [→ Section 5.4]

R ↑

Raw item [→ Section 2.3] [→ Section 4.1]
 cleanup [→ Section 7.3]
 custom [→ Section 4.6]
 exempt [→ Section 7.4]
Reactive measure [→ Section 4.4]
Reagent [→ Section 1.3] [→ Section 5.1]
Receivable adjustment [→ Section 6.1]
Reclassification [→ Section 7.2]
Reconciliation [→ Section 3.1] [→ Section 6.2]
[→ Section 6.4]
 reports [→ Section 6.4] [→ Section 6.4]
Reconciliation key [→ Section 5.3] [→ Section 6.1]
[→ Section 6.2] [→ Section 7.2]
 custom [→ Section 7.2]
 number [→ Section 6.1]

Reference key [→ Section 6.2]

Relationship [→ Section 2.3]

Remote function call (RFC) [→ Section 1.4]

Reporting [→ Section 6.1] [→ Section 6.4]

AMDP [→ Section 6.4]

CCM versus OCM [→ Section 6.4]

CDS views [→ Section 6.4]

disclosures [→ Section 6.4]

operational reports [→ Section 6.4]

staging tables [→ Section 6.4]

Residual approach [→ Section 1.2] [→ Section 5.2]

Restoration reason [→ Section 4.2]

Results analysis [→ Section 5.3] [→ Section 9.1]

calculations [→ Section 5.3]

comparison [→ Section 9.1]

keys [→ Section 5.3]

methods [→ Section 5.3]

Retrospective change [→ Section 5.4]

Return order item [→ Section 3.2]

Revenue [→ Section 1.1]

categories [→ Section 6.1]

Revenue accounting and reporting (RAR) [→ Section 1.1] [→ Section 1.4] [→ Section 2.1] [→ Section 2.4] [→ Section 9.1]

add-on [→ Section 2.1]
architecture [→ Section 1.4]
closing [→ Section 6.2]
comparison [→ Section 9.1]
data model [→ Section 2.3]
data sources [→ Section 2.3]
integration [→ Section 2.2] [→ Section 3.1]
landscape [→ Section 2.1]
mark as relevant [→ Section 3.1]
posting [→ Section 6.1]
reconciliation [→ Section 6.2]
sidecar approach [→ Section 2.1]
tables [→ Section 2.3]
versions [→ Section 1.4] [→ Section 8.1]

Revenue accounting item (RAI) [→ Section 1.4]
[→ Section 2.3] [→ Section 4.1]

automatic processing [→ Section 4.1]
batch processing [→ Section 1.4]
change [→ Section 4.2]
custom [→ Section 4.6]
customize content [→ Section 4.6]
error resolution [→ Section 4.4]
error status [→ Section 4.1]
exempt [→ Section 4.2] [→ Section 7.4]
extend monitor [→ Section 4.3]

manage [→ Section 4.2]
manual processing [→ Section 4.1]
mass creation [→ Section 7.4]
mass processing [→ Section 4.1]
modifiable fields [→ Section 2.3] [→ Section 4.2]
monitor [→ Section 2.3]
postponed [→ Section 3.2]
predecessors [→ Section 4.1]
process [→ Section 2.3] [→ Section 3.2] [→ Section 4.1] [→ Section 4.1]
restore [→ Section 7.4]
status [→ Section 2.3] [→ Section 2.3] [→ Section 2.3] [→ Section 3.2] [→ Section 4.1]
tables [→ Section 2.3]
timestamps [→ Section 4.1]
transfer [→ Section 2.3] [→ Section 2.3] [→ Section 7.4]

Revenue accounting item (RAI) class [→ Section 2.3]
[→ Section 2.3]

activate [→ Section 2.3]
activate interfaces [→ Section 3.2]
configuration [→ Section 3.2]
custom fields [→ Section 2.3]
extend [→ Section 3.3]
generate interfaces [→ Section 2.3]
maintain [→ Section 2.3]

OIP [→ Section 3.2]

type [→ Section 2.3] [→ Section 3.2]

Revenue amount [→ Section 5.3] [→ Section 6.1]

Revenue Contract Search app [→ Section 6.4]

Revenue Explanation app [→ Section 6.4]

Revenue forecast [→ Section 5.3]

Revenue posting [→ Section 6.1] [→ Section 7.2]

ABC programs [→ Section 6.2]

account assignment [→ Section 6.1]

categories [→ Section 6.1]

custom [→ Section 7.2]

customize [→ Section 6.1]

extend program [→ Section 7.2]

keys [→ Section 6.1]

run [→ Section 6.2] [→ Section 6.2]

settings [→ Section 6.1]

Revenue recognition [→ Section 1.2] [→ Section 1.4]

calculation [→ Section 6.1]

choosing your tool [→ Section 2.4]

data model [→ Section 2.3]

design your landscape [→ Section 2.1]

reporting [→ Section 6.1] [→ Section 6.4]

Revenue schedule [→ Section 5.3]

Revenue Schedule app [→ Section 6.4]

Revenue transfer [→ Section 6.1]

Revenue Transfer program [→ Section 5.1]

Rule set [→ Section 4.5]

Run ID [→ Section 5.1]

S ↑

Sales and distribution [→ Section 2.2] [→ Section 3.1]

Sales document type [→ Section 3.1]

SAP Billing and Revenue Innovation Management
[→ Section 2.2] [→ Section 3.2] [→ Section 6.4]

SAP Business Client [→ Section 6.4]

SAP Convergent Charging [→ Section 2.2]

SAP Convergent Mediation by DigitalRoute [→ Section
2.2]

SAP Customer Financial Management [→ Section 2.2]

SAP Customer Relationship Management (SAP CRM)
[→ Section 2.2]

SAP ERP [→ Section 1.4] [→ Section 2.4]

SAP Fiori app [→ Section 6.4]

CCM versus OCM [→ Section 6.4]

SAP Fiori launchpad [→ Section 6.4]

SAP HANA [→ Section 6.4]

SAP Revenue Accounting and Reporting [→ Section 1.4]

SAP S/4HANA [→ Section 1.4] [→ Section 2.4]
releases [→ Section 8.1]

SAP S/4HANA Cloud, public edition [→ Section 9.1]

SAP S/4HANA Service [→ Section 2.2]

SAP Subscription Billing [→ Section 2.2]

Selection screen [→ Section 7.1] [→ Section 7.3]
main program [→ Section 7.1]

Sell-from-stock scenario [→ Section 9.2]

Sender component [→ Section 2.3]
assign source item type [→ Section 2.3]
enable integration [→ Section 3.1]

Sequence [→ Section 2.3]

Server group [→ Section 7.1]

Significant financing component [→ Section 1.2]

Simulation mode [→ Section 7.3]

Singleton class [→ Section 4.6]
attributes [→ Section 4.6]
create [→ Section 4.6]
methods [→ Section 4.6]

Source item type [→ Section 2.3]

Staging table [→ Section 6.4]

Standalone selling price (SSP) [→ Section 1.2]
[→ Section 1.3] [→ Section 5.2] [→ Section 5.3]

BRFplus [→ Section 4.5]

change [→ Section 4.1]

determine [→ Section 1.2]

tolerance [→ Section 5.2]

Standard class [→ Section 4.6]

Start date type [→ Section 5.3]

Static class [→ Section 4.6]

Structure [→ Section 2.3]

Subscription order management [→ Section 2.2]

Suspension period [→ Section 8.2]

Switch method [→ Section 4.6]

System landscape [→ Section 2.1]

RAR [→ Section 2.1]

regional split [→ Section 2.1]

T ↑

Table [→ Section 2.3]

/IRA [→ Section 1.4] [→ Section 2.3]

add index [→ Section 2.3]

BRFplus [→ Section 4.5]

conversion [→ Section 2.3]

custom [→ Section 2.3]

display structure [→ Section 2.3]

DT_PROCESS_BOM [→ Section 5.2]

DT_PROCESS_COMPOUND [→ Section 5.2]
DT_PROCESS_POB_ADD [→ Section 5.2]
DT_PROCESS_SSP [→ Section 5.2]
FARR_ACCT_DETERMINE_DT_ASST_IM [→ Section 5.4]
FARR_D_CONS [→ Section 4.4] [→ Section 7.3]
FARR_D_CONTRACT [→ Section 8.4]
FARR_D_DEFITEM [→ Section 5.3] [→ Section 5.3]
[→ Section 6.1]
FARR_D_FULFILLMNT [→ Section 5.3]
FARR_D_MAPPING [→ Section 5.1]
FARR_D_POB [→ Section 1.4] [→ Section 5.2]
FARR_D_POSTING [→ Section 5.1] [→ Section 6.1]
[→ Section 6.1] [→ Section 6.1] [→ Section 6.1]
[→ Section 6.1] [→ Section 6.1] [→ Section 7.2]
FARR_D_RECON_KEY [→ Section 6.1] [→ Section 6.1]
[→ Section 6.2] [→ Section 7.2]
FARR_RAI_MON [→ Section 5.3]
OIP [→ Section 3.2]
populate custom [→ Section 7.2]
staging [→ Section 6.4]
standard [→ Section 2.3]
temporary [→ Section 2.3]
ZFARR_D [→ Section 7.2]
ZTRANSFER_MSG [→ Section 7.3]

Table maintenance generator [→ Section 4.6]

Telecommunications [→ Section 1.3] [→ Section 6.2]

Termination fee [→ Section 5.4]

Time and material billing [→ Section 9.3]

Time value of money [→ Section 1.3]

Time-based POB [→ Section 5.3] [→ Section 5.3]
[→ Section 6.1]

settings [→ Section 5.3]

Transaction

BRFPLUS [→ Section 4.5] [→ Section 5.2]

FARR_CCM_OCM_MIG_CON [→ Section 8.3]

FARR_CONTR_CHECK [→ Section 4.4] [→ Section
7.3] [→ Section 8.4]

FARR_CONTR_MON [→ Section 4.4] [→ Section 7.3]

FARR_CONTRACT_LIABILITY [→ Section 6.2]
[→ Section 6.2]

FARR_D_POSTING [→ Section 6.1]

FARR_IMG [→ Section 2.3] [→ Section 2.3]
[→ Section 2.3] [→ Section 3.2] [→ Section 4.5]
[→ Section 5.1] [→ Section 5.1] [→ Section 6.1]

FARR_LIABILITY_CALC [→ Section 2.4]

FARR_PERIOD_IN_CLOSING [→ Section 5.1]

FARR_RAI_CONF [→ Section 3.2] [→ Section 3.3]

FARR_RAI_MON [→ Section 1.4] [→ Section 2.3]
[→ Section 2.3] [→ Section 3.2] [→ Section 4.1]
[→ Section 4.1] [→ Section 4.3] [→ Section 5.3]
[→ Section 6.1]

FARR_RAI_PROC [→ Section 2.3] [→ Section 4.1]
FARR_RAI_TRANS [→ Section 2.3] [→ Section 4.1]
[→ Section 7.3] [→ Section 7.4]
FARR_REPR_PPRAI [→ Section 3.2]
FARR_REV_TRANSFER [→ Section 2.4] [→ Section
6.2] [→ Section 6.2]
FARR_REVENUE_POSTINGS [→ Section 6.2]
[→ Section 6.2]
FB03 [→ Section 6.2]
OB22 [→ Section 5.1]
OKG3 [→ Section 5.3]
SE11 [→ Section 2.3] [→ Section 4.2] [→ Section
4.6] [→ Section 6.4] [→ Section 7.2] [→ Section 7.3]
SE14 [→ Section 2.3]
SE16 [→ Section 5.1]
SE16N [→ Section 4.4]
SE17 [→ Section 2.3]
SE18 [→ Section 3.1] [→ Section 3.2] [→ Section
4.6] [→ Section 5.1] [→ Section 5.4] [→ Section 5.4]
[→ Section 7.3]
SE24 [→ Section 4.6] [→ Section 4.6] [→ Section
7.4] [→ Section 7.4]
SE37 [→ Section 7.4]
SE38 [→ Section 6.4] [→ Section 7.3] [→ Section
7.4]
SE91 [→ Section 7.3]
SHDB [→ Section 2.4]

SHDB PFW [→ Section 2.4]

SLG1 [→ Section 4.1] [→ Section 4.4] [→ Section 6.2] [→ Section 6.2] [→ Section 7.3]

SM30 [→ Section 4.6] [→ Section 5.3]

SM36 [→ Section 7.1]

SM37 [→ Section 6.2] [→ Section 6.4]

SM61 [→ Section 7.1]

SNRO [→ Section 6.1]

SOAMANAGER [→ Section 5.4]

SPRO [→ Section 3.1]

VF44 [→ Section 2.2] [→ Section 2.4]

VLPOD [→ Section 5.3]

ZRAR_NAVIGATE [→ Section 7.4]

Transactional price [→ Section 1.2] [→ Section 1.3]
[→ Section 5.3]

allocation [→ Section 1.2]

Transfer Revenue program [→ Section 5.1] [→ Section 6.1]
[→ Section 6.1] [→ Section 6.2] [→ Section 6.2]

Transport [→ Section 2.3]

request [→ Section 2.3]

Troubleshooting [→ Section 7.1]

Type [→ Section 2.3]

Unbilled receivable/deferred revenue (UR/DR)
[→ Section 5.1]

calculation [→ Section 5.1]

Unconditionality [→ Section 5.1]

Upload rule [→ Section 3.2]

V ↑

Variable consideration [→ Section 1.2] [→ Section 1.2]
[→ Section 1.3]

W ↑

Weighted average rate [→ Section 5.1]

Work breakdown structure (WBS) element [→ Section
9.3]

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: *Megan Fuerst*. We welcome every suggestion for improvement but, of course, also any praise! You can also share your reading experience via Twitter, Facebook, or email.

Supplements

If there are supplements available (sample code, exercise materials, lists, and so on), they will be provided in your online library and on the web catalog page for this book. You can directly navigate to this page using the following link: <https://www.sap-press.com/5700>. Should we learn about typos that alter the meaning or content errors, we will provide a list with corrections there, too.

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

This publication is protected by copyright in its entirety. All usage and exploitation rights are reserved by the author and Rheinwerk Publishing; in particular the right of reproduction and the right of distribution, be it in printed or electronic form.

© **2024 by Rheinwerk Publishing Inc., Boston (MA)**

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, Expenselt, SAP ActiveAttention, SAP Adaptive Server Enterprise, SAP

Advantage Database Server, SAP ArchiveLink, SAP Ariba, SAP Business ByDesign, SAP Business Explorer (SAP BEx), SAP BusinessObjects, SAP BusinessObjects Explorer, SAP BusinessObjects Web Intelligence, SAP Business One, SAP Business Workflow, SAP BW/4HANA, SAP C/4HANA, SAP Concur, SAP Crystal Reports, SAP EarlyWatch, SAP Fieldglass, SAP Fiori, SAP Global Trade Services (SAP GTS), SAP GoingLive, SAP HANA, SAP Jam, SAP Leonardo, SAP Lumira, SAP MaxDB, SAP NetWeaver, SAP PartnerEdge, SAPPHIRE NOW, SAP PowerBuilder, SAP PowerDesigner, SAP R/2, SAP R/3, SAP Replication Server, SAP Roambi, SAP S/4HANA, SAP S/4HANA Cloud, SAP SQL Anywhere, SAP Strategic Enterprise Management (SAP SEM), SAP SuccessFactors, SAP Vora, Triplt, and Qualtrics are registered or unregistered trademarks of SAP SE, Walldorf, Germany.

Limitation of Liability

Regardless of the care that has been taken in creating texts, figures, and programs, neither the publisher nor the author, editor, or translator assume any legal responsibility or any liability for possible errors and their consequences.

The Document Archive

The Document Archive contains all figures, tables, and footnotes, if any, for your convenience.

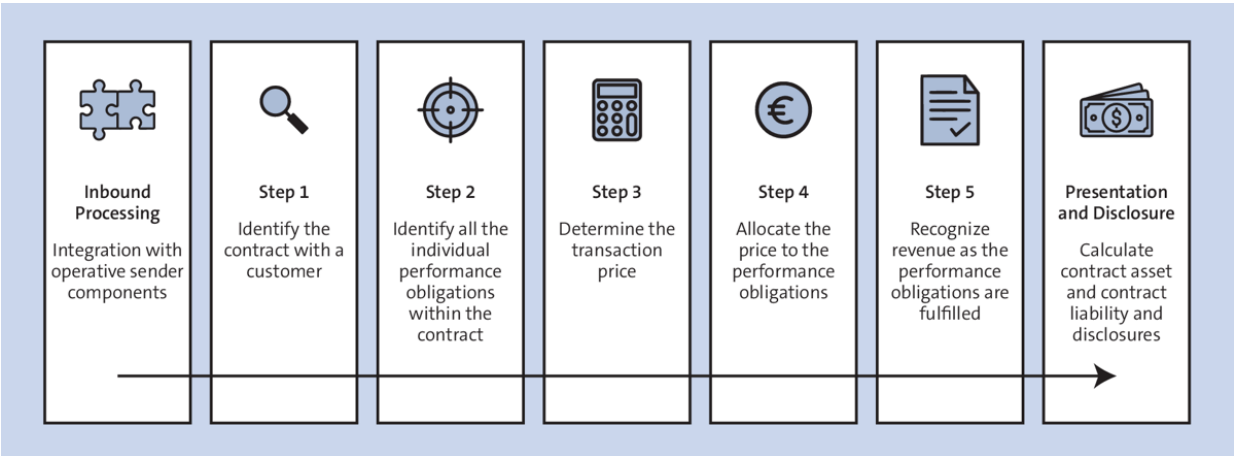


Figure 1.1 Five-Step Model of Revenue Recognition

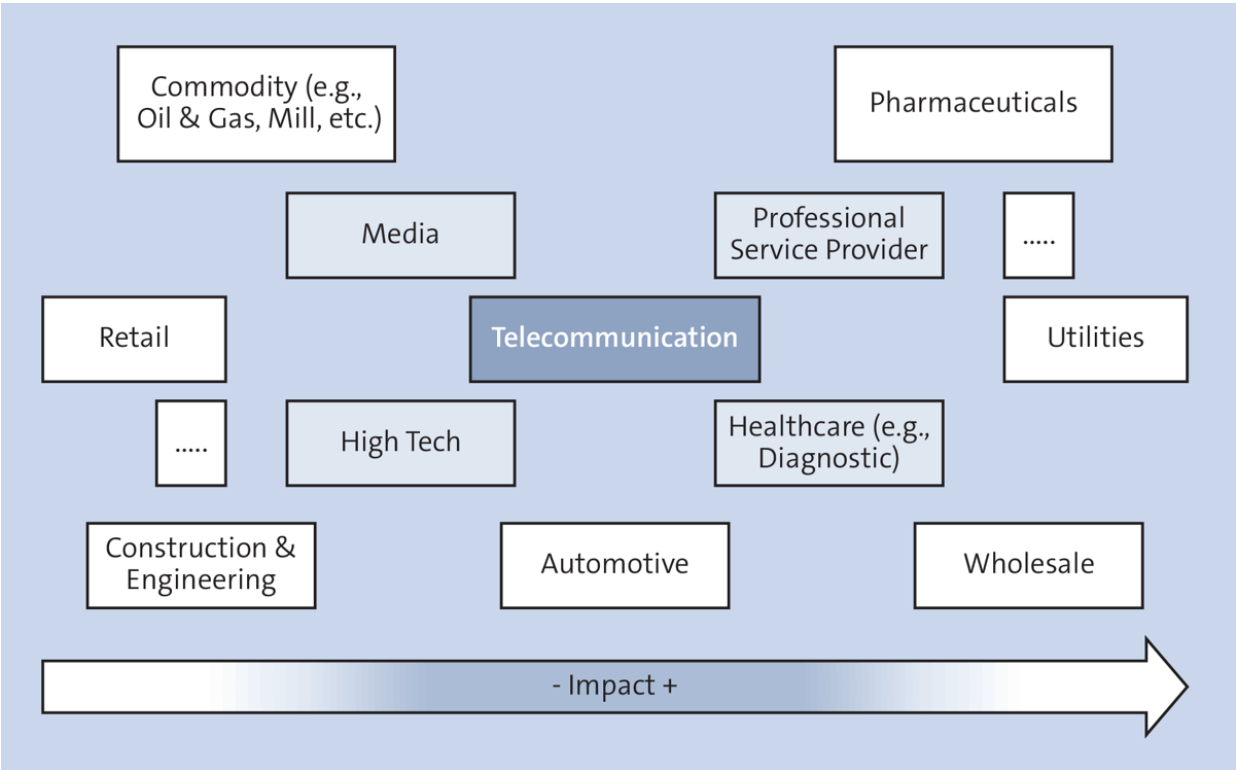


Figure 1.2 Industry Impact of the IFRS 15 Implementation

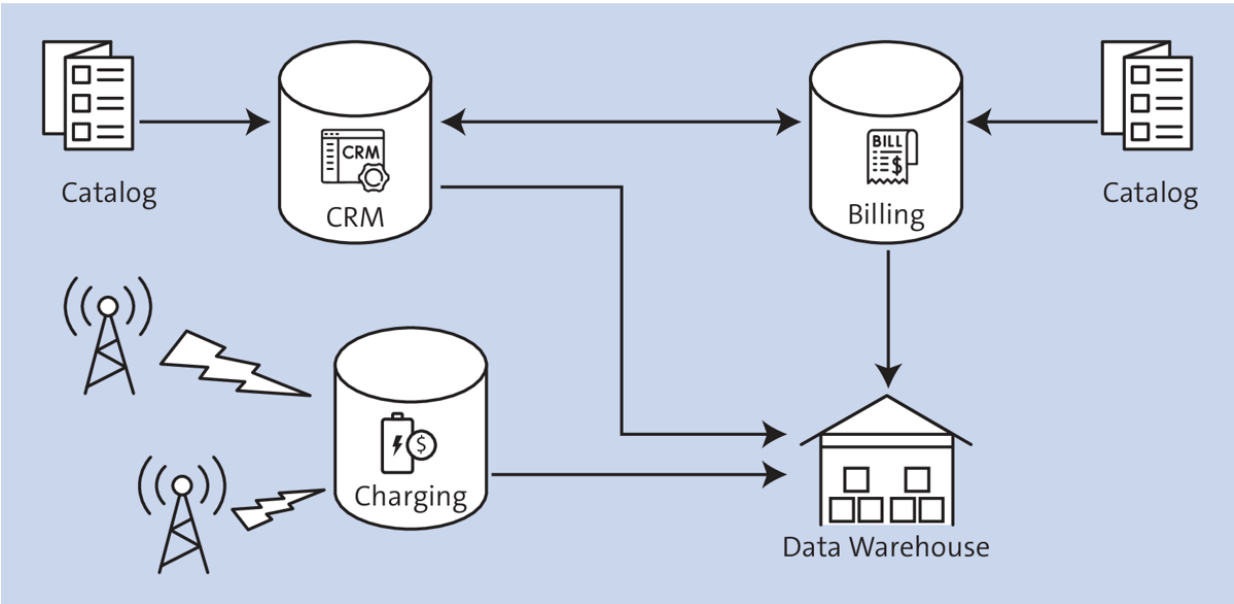


Figure 1.3 Landscape of Business Support Systems in the Telco Industry

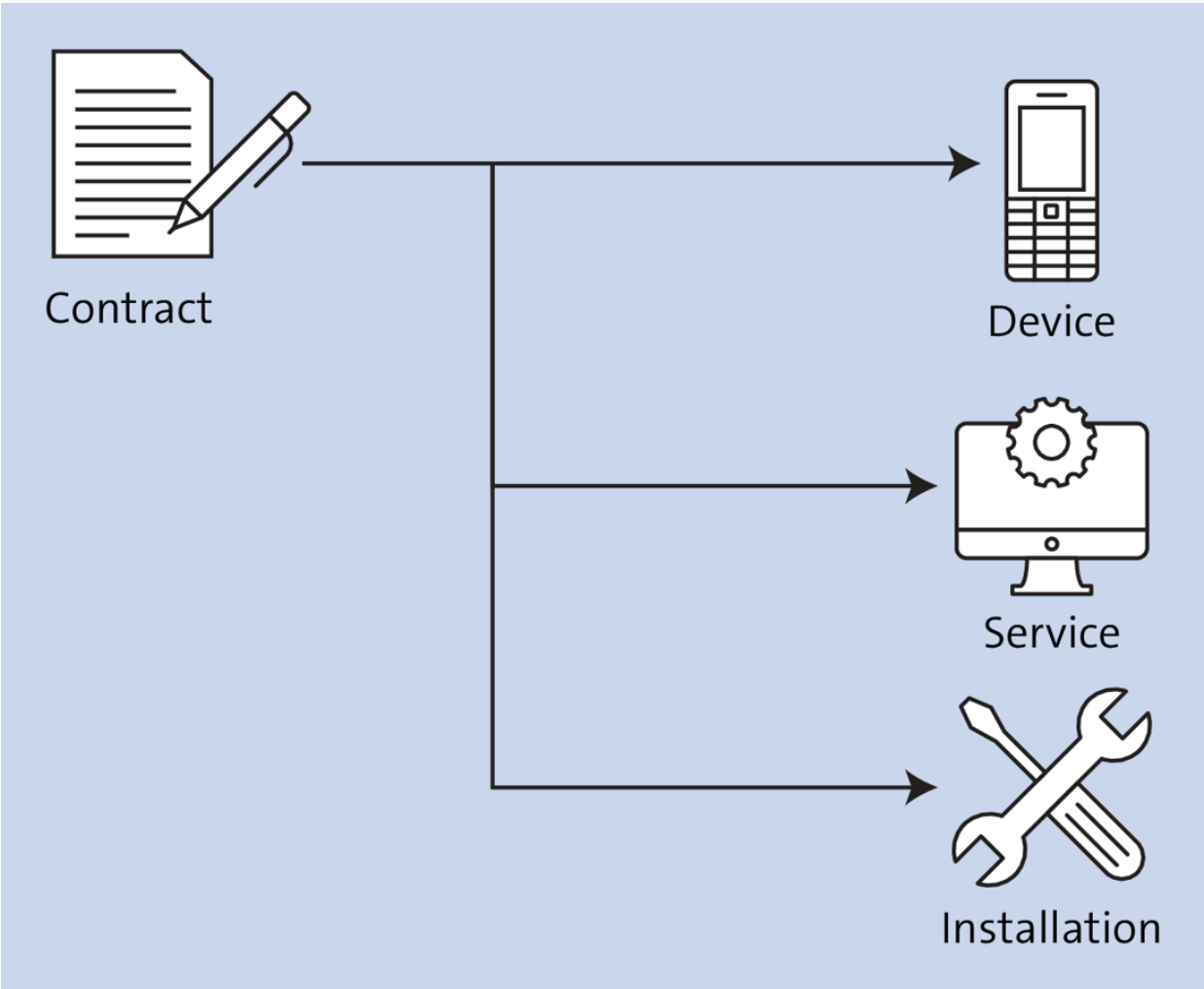


Figure 1.4 Example of a Simple Telco Contract

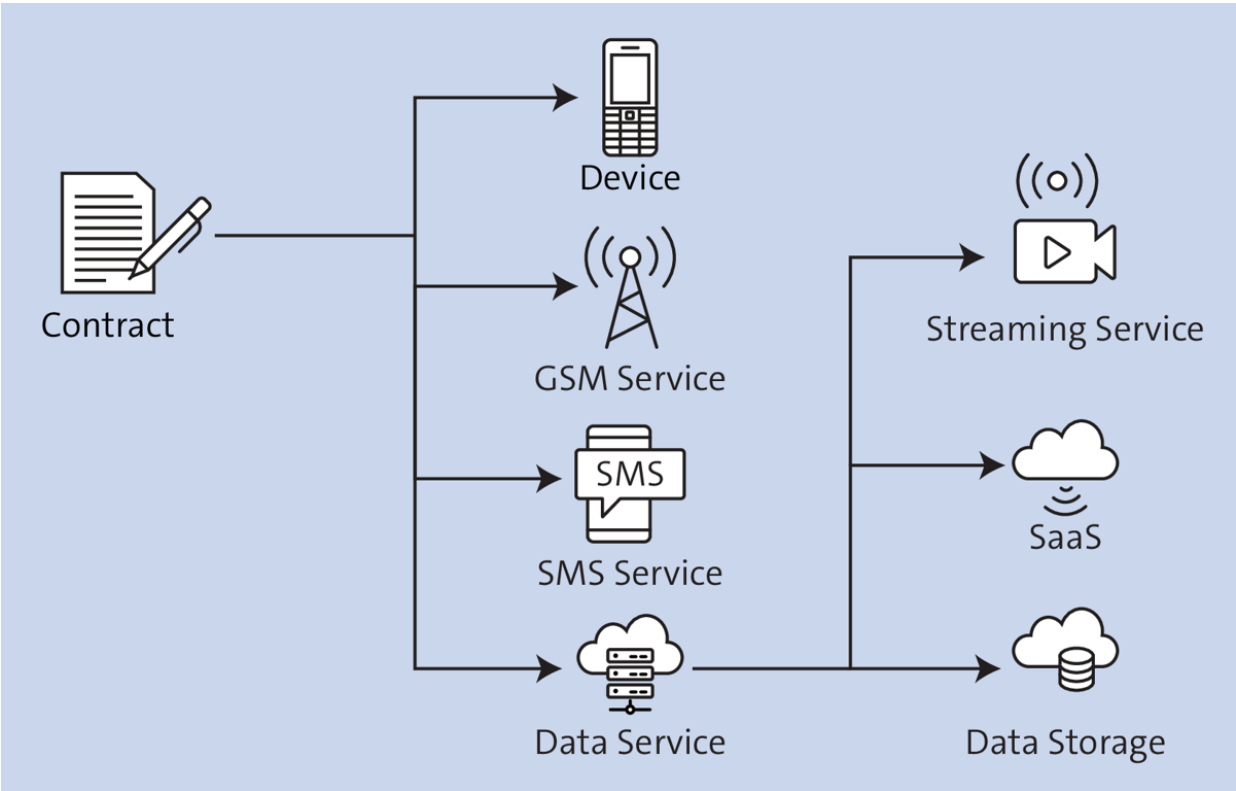


Figure 1.5 Example of an Expanded Telco Contract

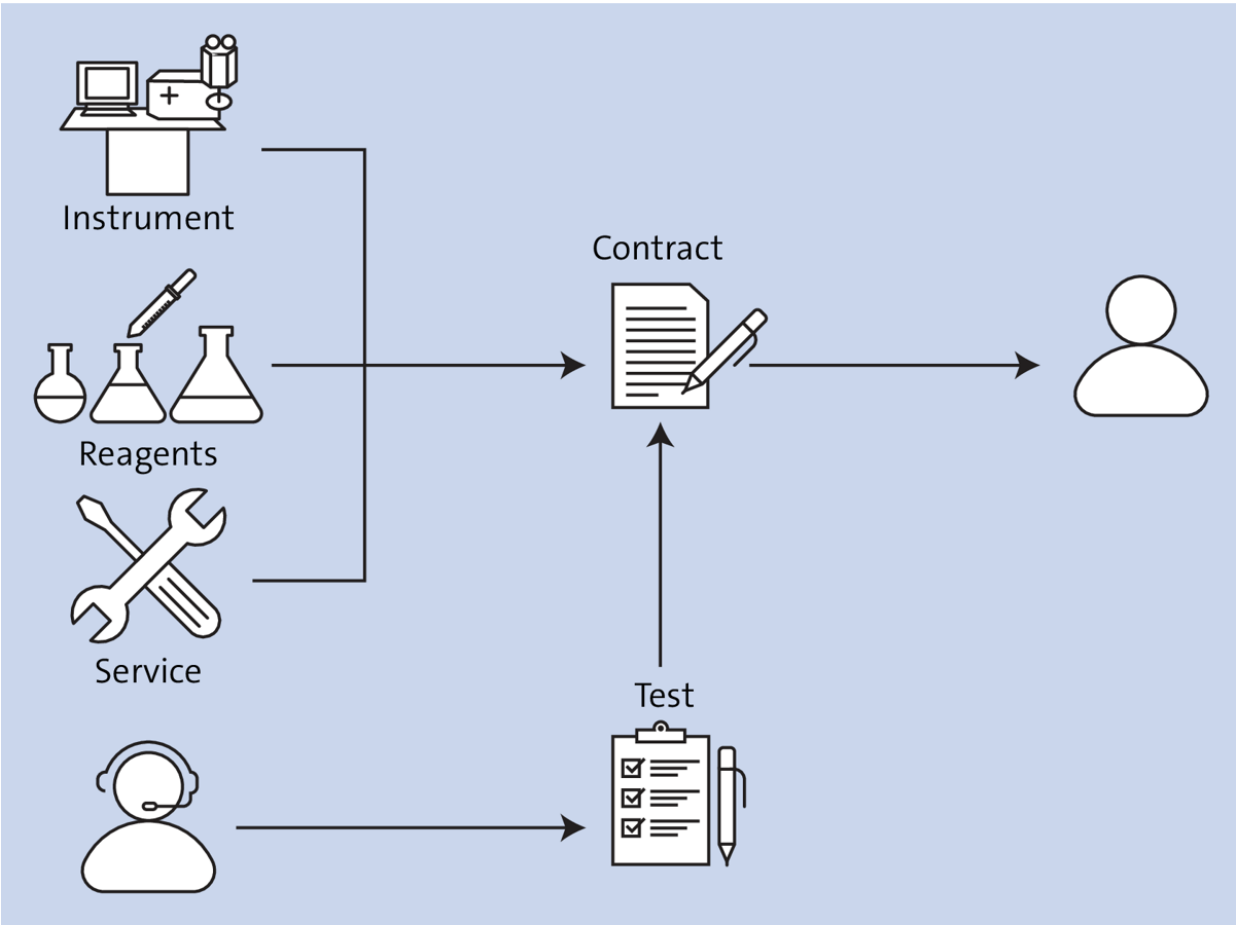


Figure 1.6 Process in the Diagnostics Industry

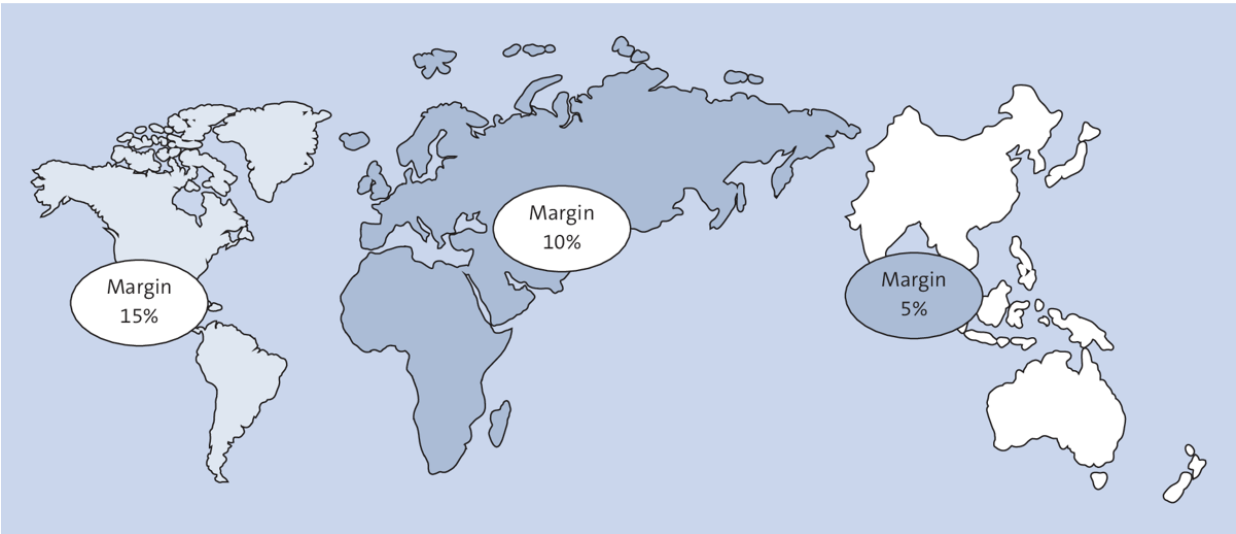


Figure 1.7 Different Margins in the Regions

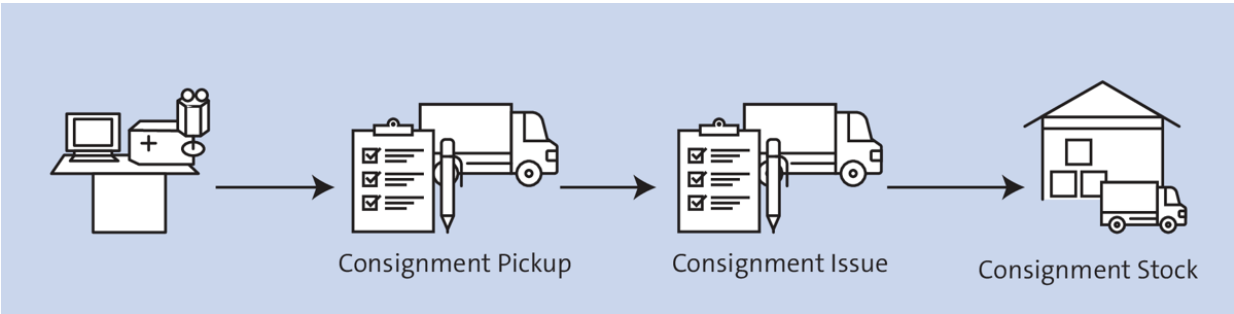


Figure 1.8 Process of Lease Sales of an Instrument

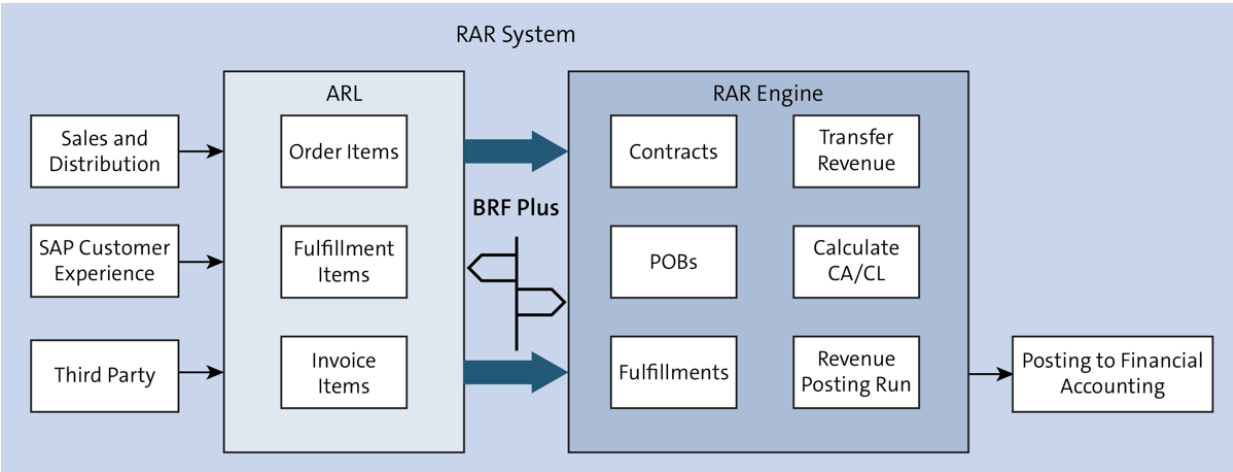


Figure 1.9 RAR Architecture

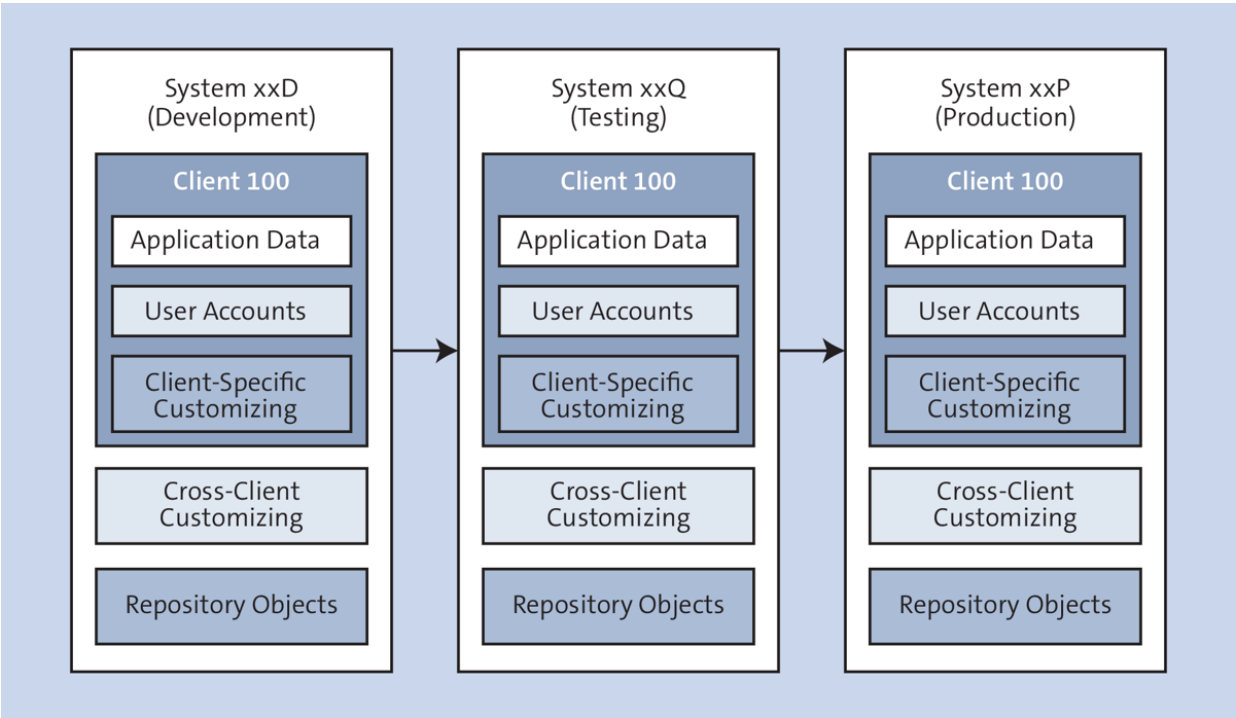


Figure 2.1 Standard SAP Landscape

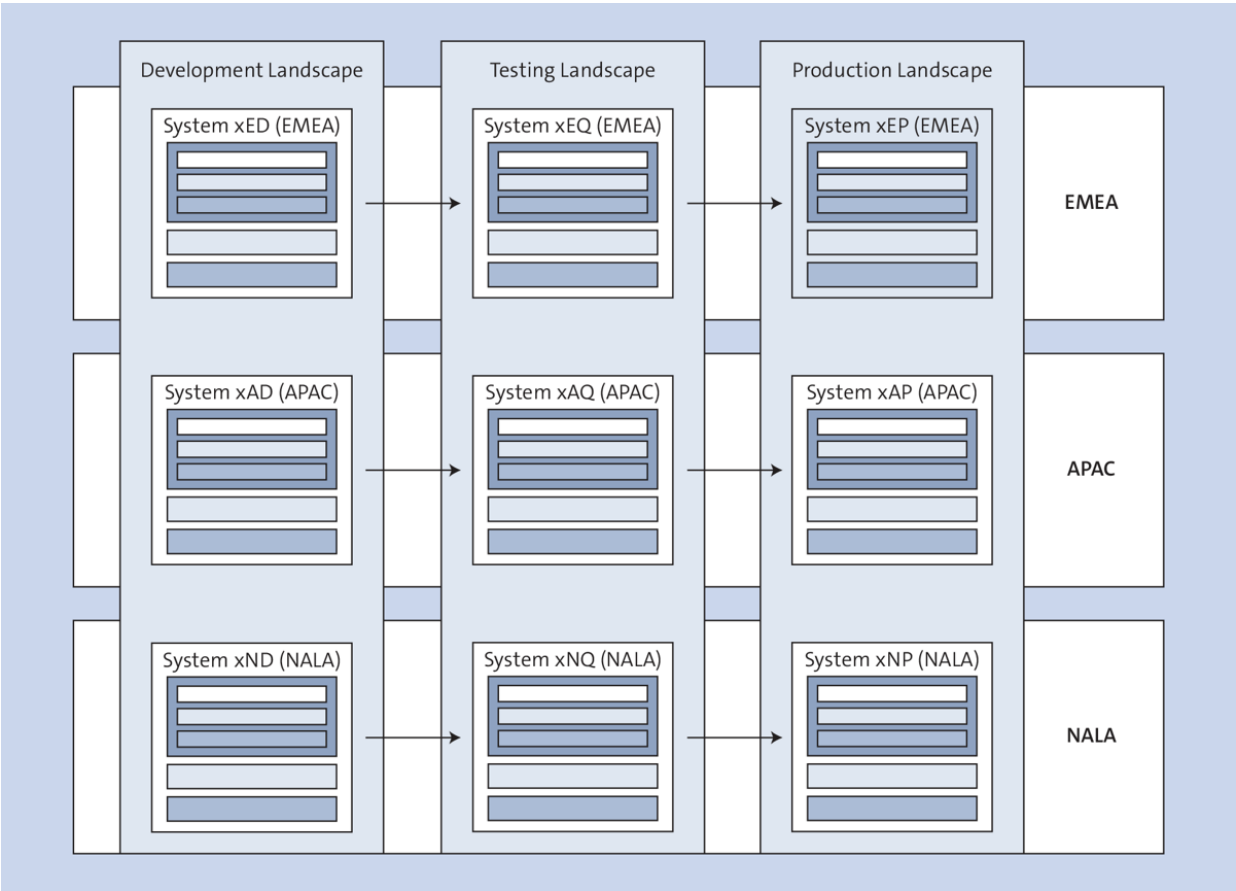


Figure 2.2 Landscape with Regional Split

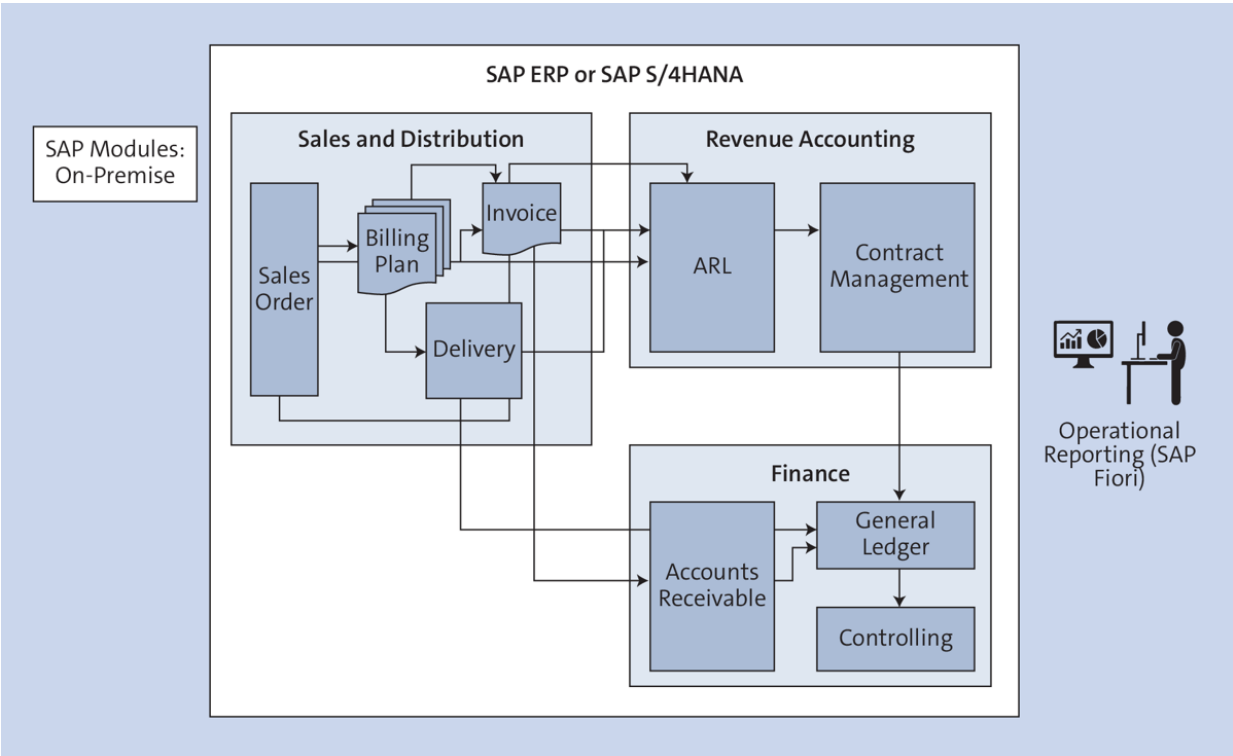


Figure 2.3 Landscape with the Revenue Accounting Add-On

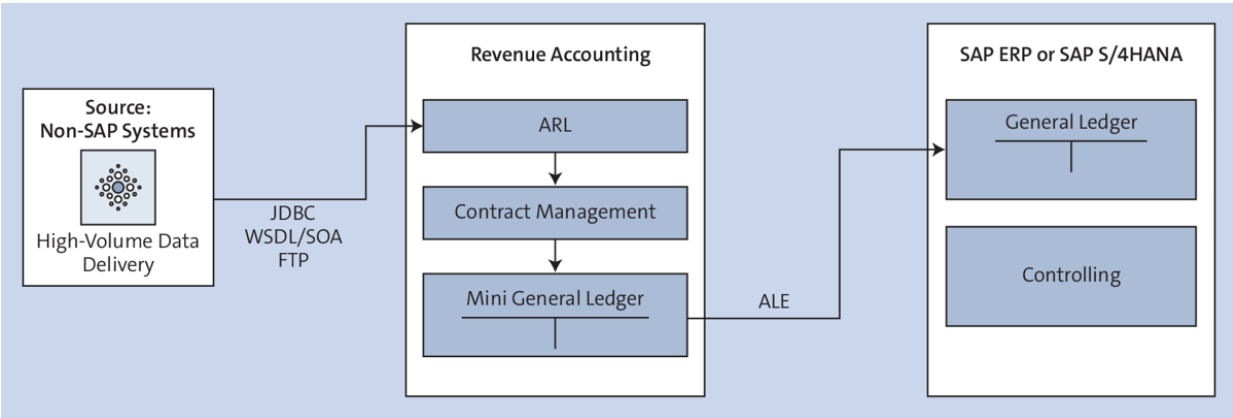


Figure 2.4 Sidecar Approach

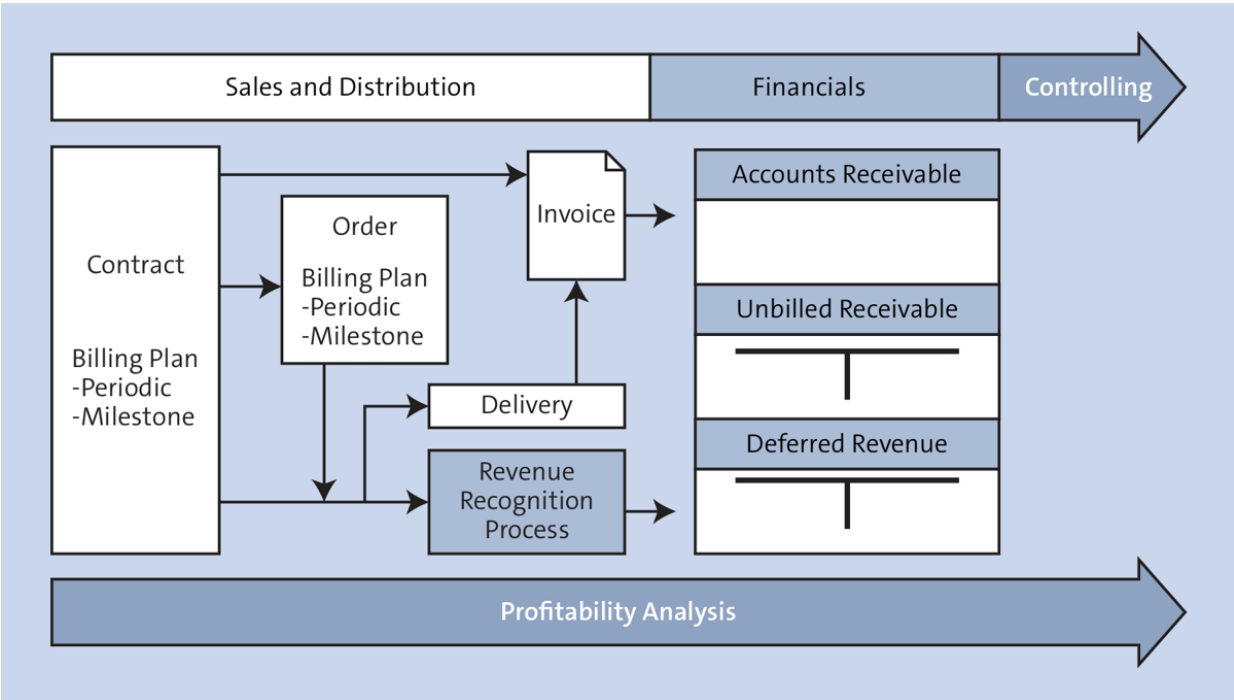


Figure 2.5 Transaction VF44 Processes

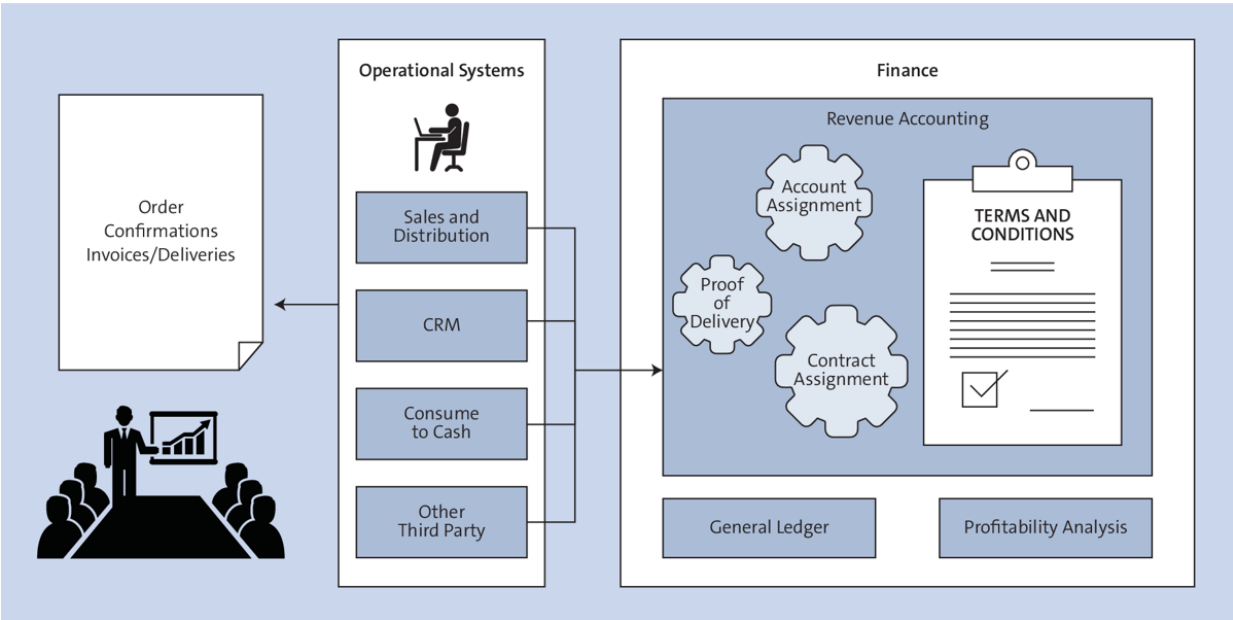


Figure 2.6 RAR Integrated with Sales and Distribution

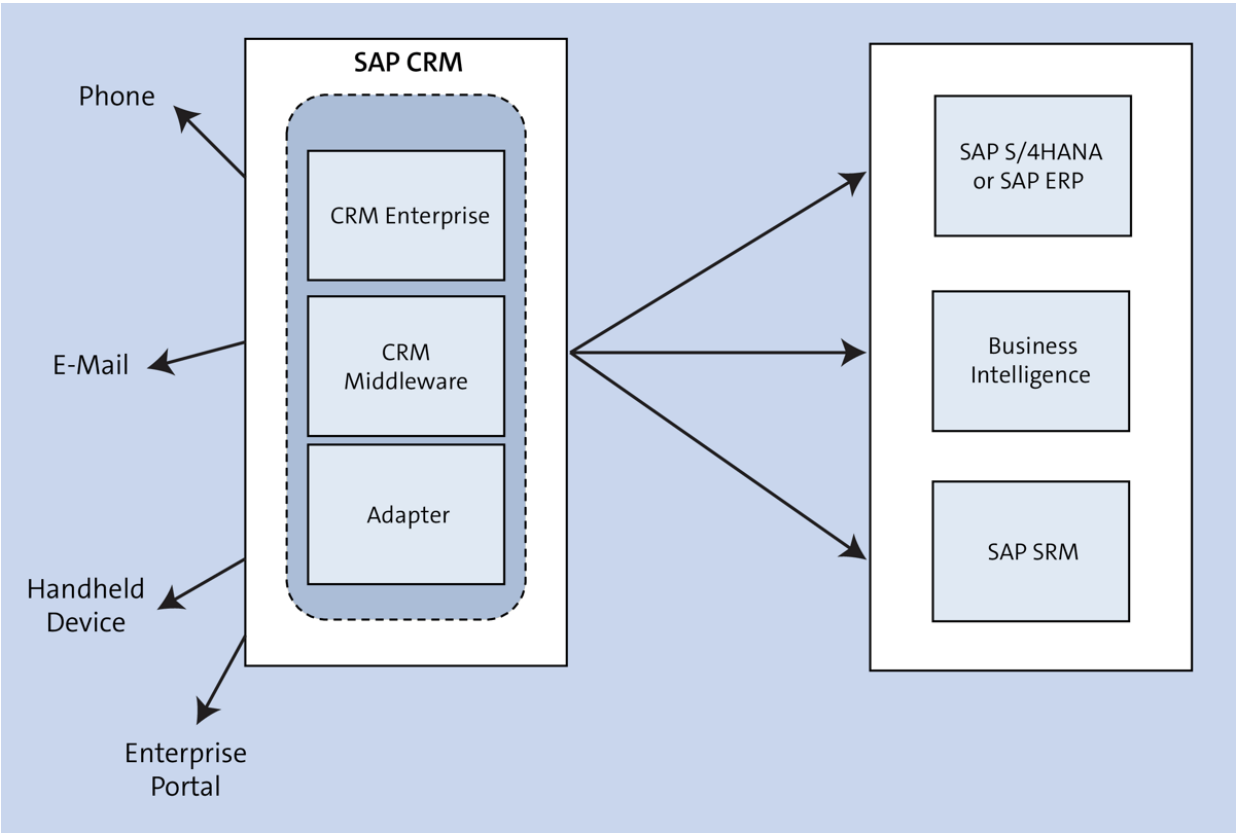


Figure 2.7 SAP CRM Architecture

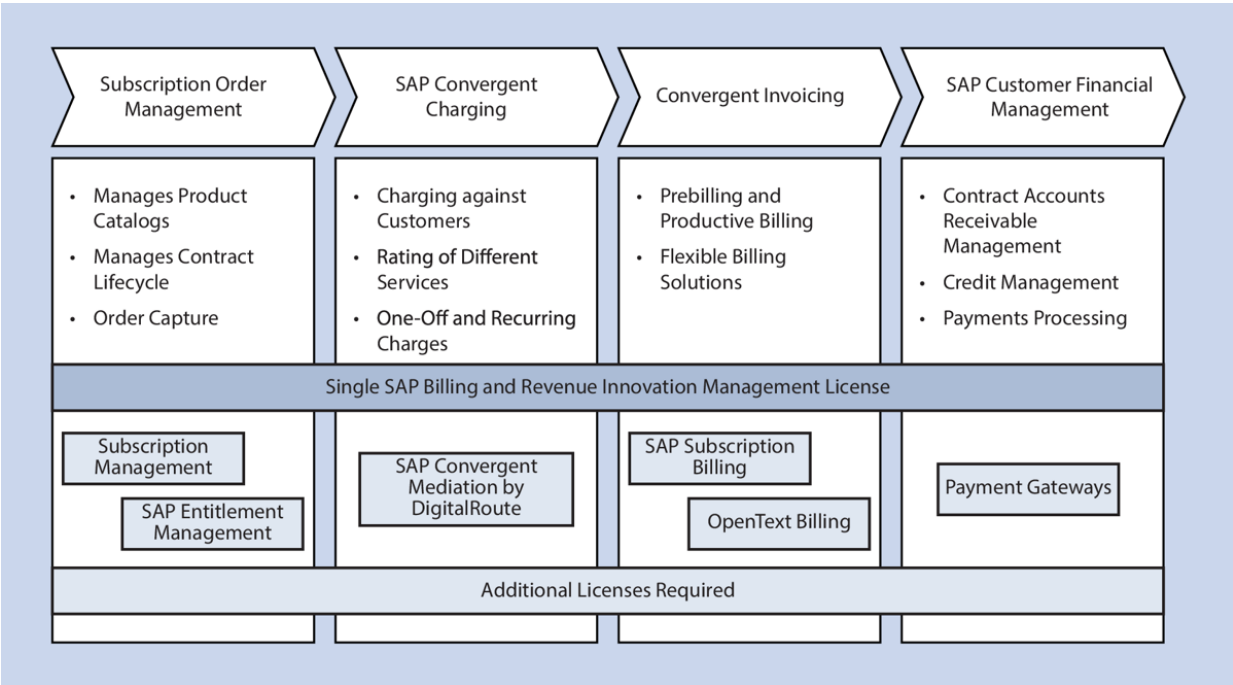


Figure 2.8 SAP Billing and Revenue Innovation Management Architecture

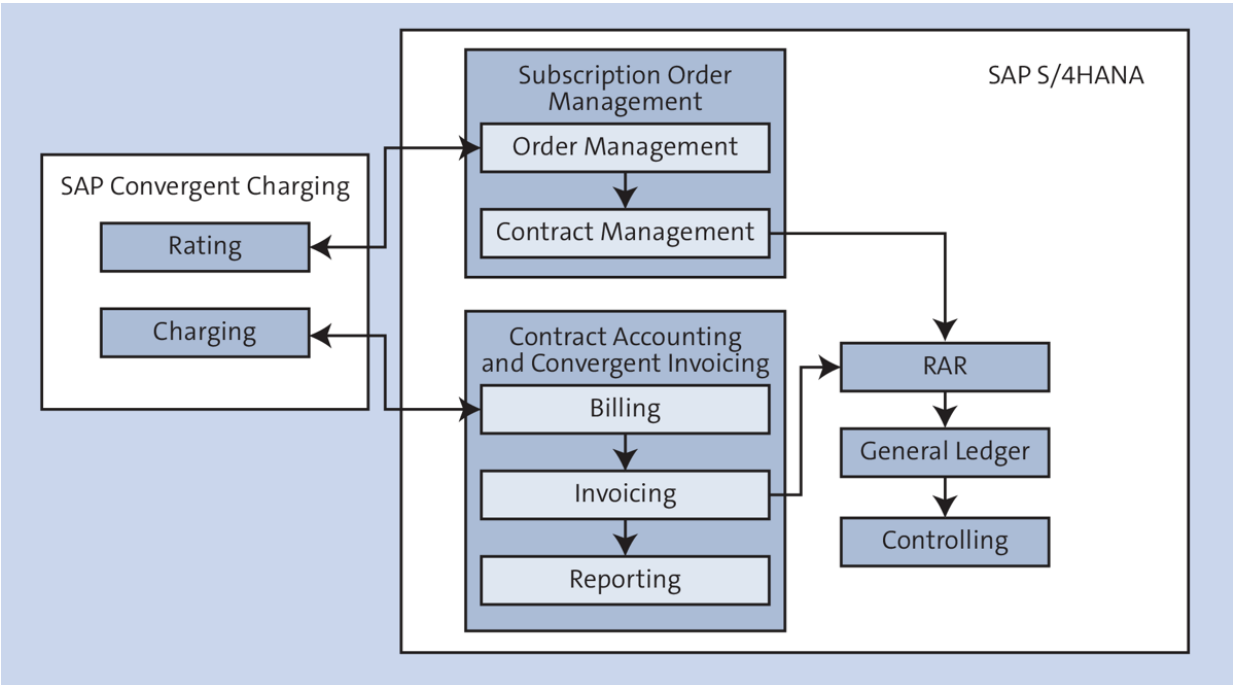


Figure 2.9 SAP Billing and Revenue Innovation Management: Integration Points

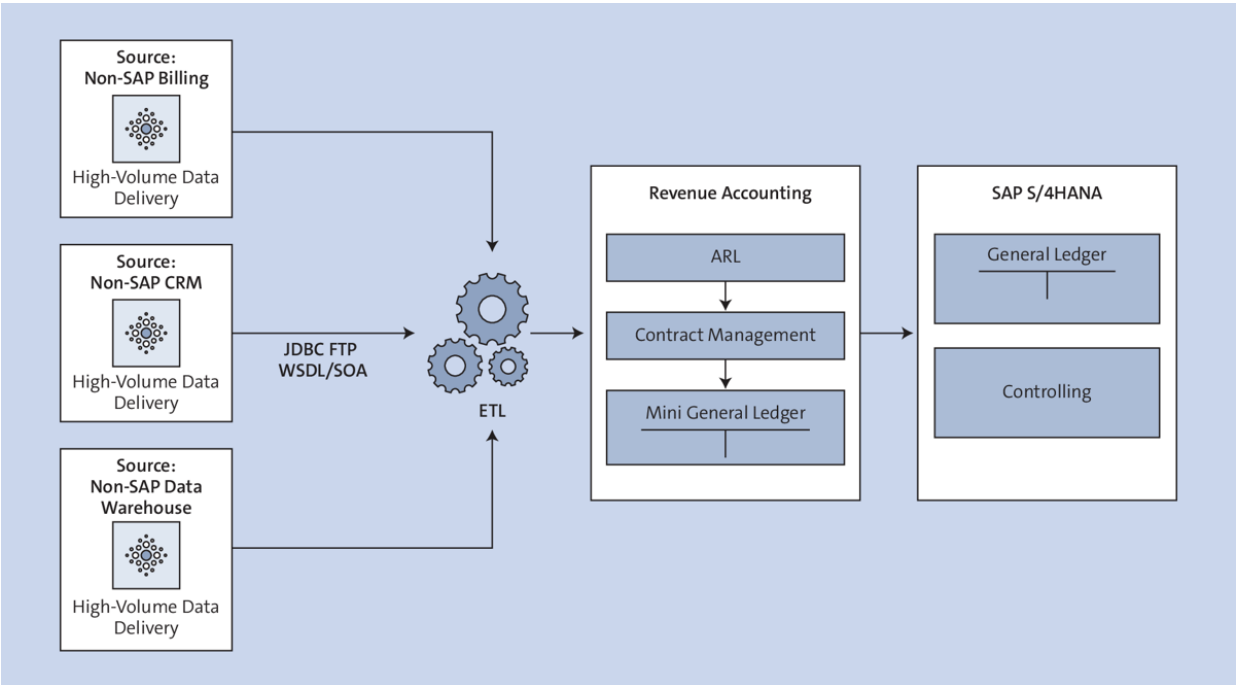


Figure 2.10 RAR Landscape with ETL

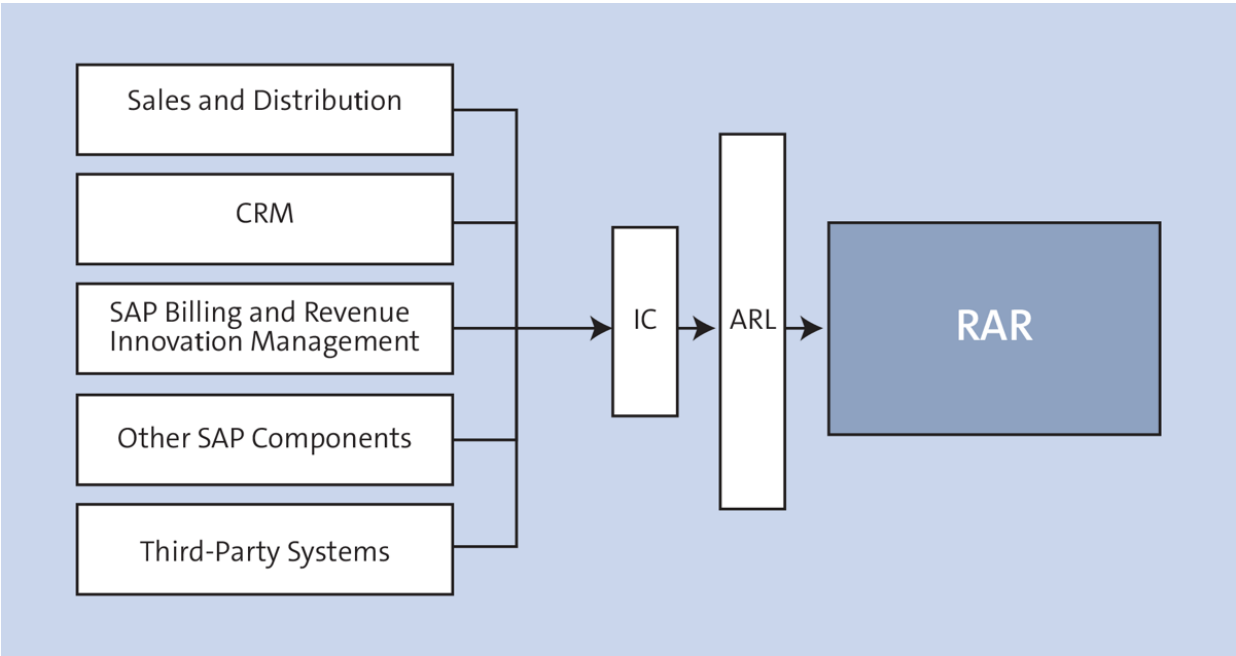


Figure 2.11 Data from Sources to RAR

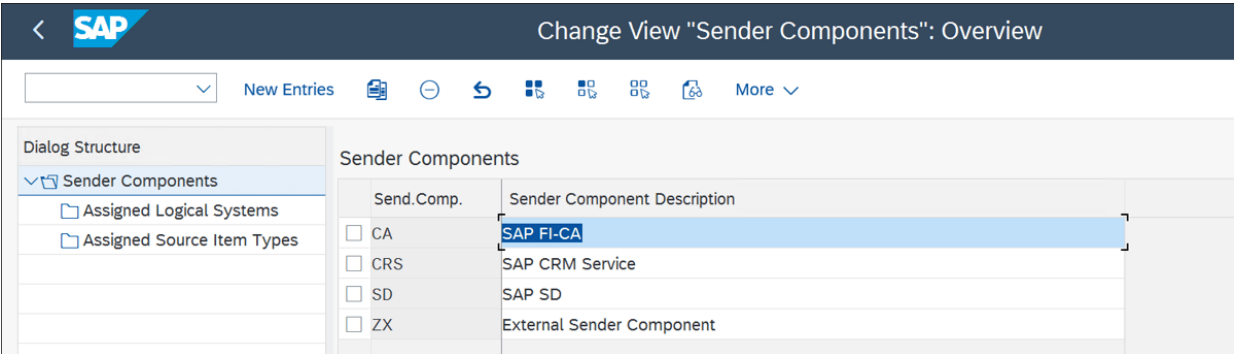


Figure 2.12 Sender Components Screen

The screenshot shows the SAP interface for 'Change View "Logical Systems": Overview'. The header includes the SAP logo and a search bar. Below the header is a toolbar with 'New Entries' and several icons. The main content area is titled 'Logical Systems' and contains a table with the following data:

Log.System	Name of Logical System
<input type="checkbox"/> A4HCLNT100	Logical System A4H Client 100

Figure 2.13 Define Logical Systems

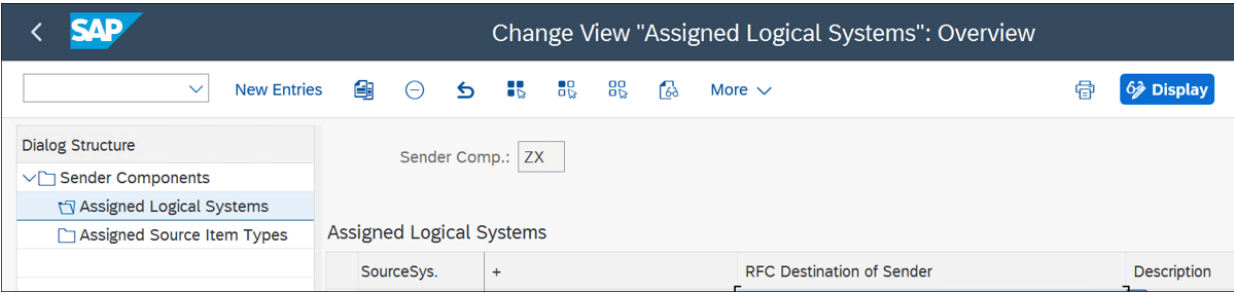


Figure 2.14 Assign Logical Systems

Change View "Source Item Types": Overview

Source Item Types

SrcitmType	Class Type	Item Type Description
<input type="checkbox"/> SDFI	Fulfillment Item	✓ Fulfillment Item
<input type="checkbox"/> SDIG	Order Item	✓ Invoice generated POB
<input type="checkbox"/> SDII	Invoice Item	✓ Invoice Item
<input type="checkbox"/> SDOI	Order Item	✓ Order Item
<input type="checkbox"/> SDPI	Invoice Item	✓ Billing Plan Item
<input type="checkbox"/> ZACI	Order Item	✓ External Order Item Type
<input type="checkbox"/> ZAII	Invoice Item	✓ External Invoice Item Type

Figure 2.15 Source Item Types for Order, Fulfillment, and Invoice

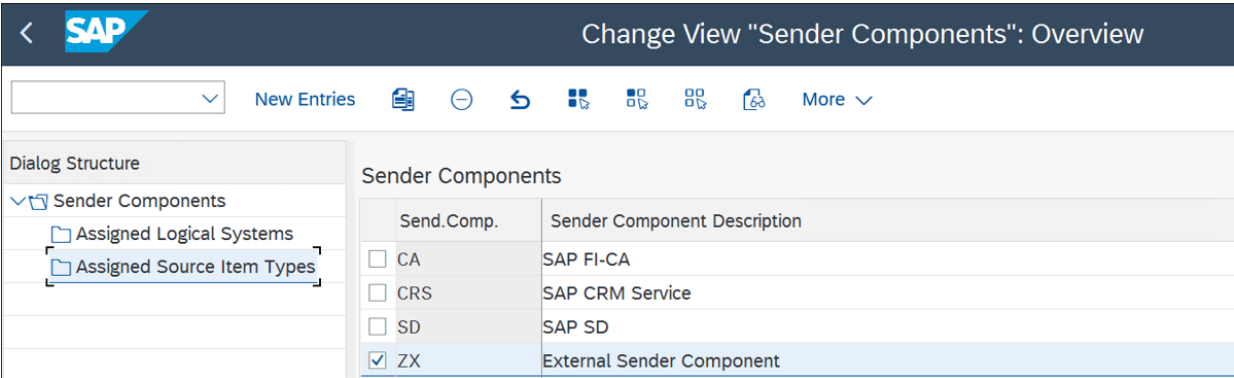


Figure 2.16 Assign Source Item Type to Sender Component

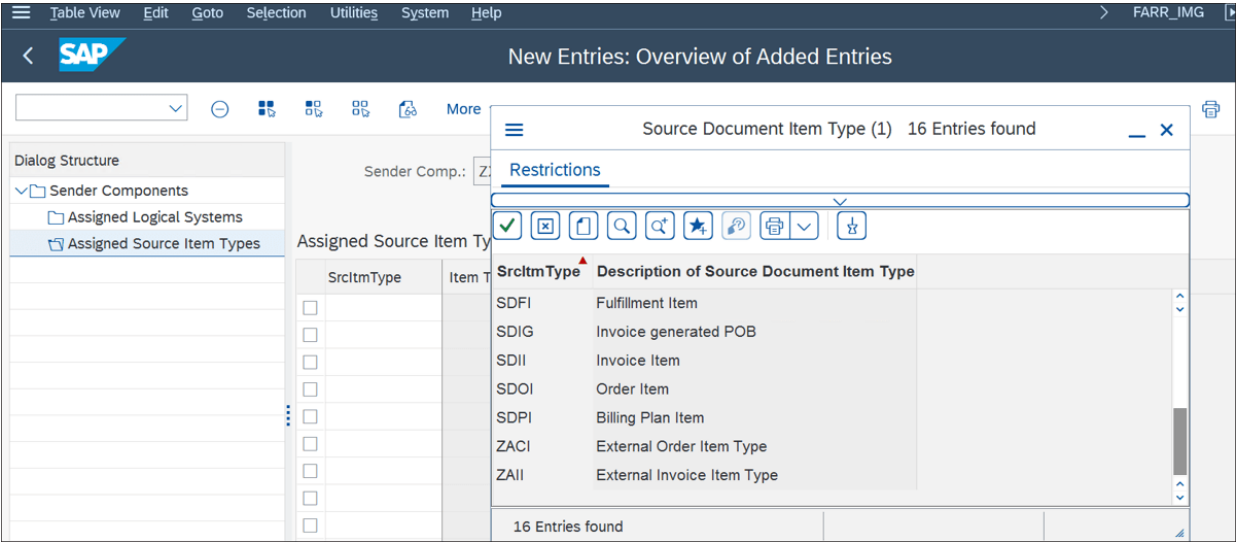


Figure 2.17 Assign Source Item Type to Sender Component (Cont.)

The screenshot displays the SAP S/4HANA 'Change View' for 'Interface Components for RAI Classes'. The interface includes a top navigation bar with menu items like 'Table View', 'Edit', 'Goto', 'Selection', 'Utilities', 'System', and 'Help'. Below the navigation bar, there are search and action buttons such as 'New Entries', 'Display', and 'More'. The main content area is divided into two sections: 'Dialog Structure' on the left and a table on the right.

The 'Dialog Structure' section shows a tree view with the following items:

- Interface Components for RAI Classes
 - Assigned Structures
 - Prerequisite Components
 - Program Enhancements
 - Key Fields
 - Assign Components to Class Type

The table 'Interface Components for RAI Classes' contains the following data:

Interface Component	Description of Interface Component	Data Element for Documentation
<input type="checkbox"/> BASIC_CO	Basic Fields for Conditions	FARR_IC_BASIC_CO
<input type="checkbox"/> BASIC_CO01	Basic Fields for Order Items (Conditions)	FARR_IC_BASIC_CO01
<input type="checkbox"/> BASIC_MI	Basic Fields for Main Items	FARR_IC_BASIC_MI
<input type="checkbox"/> BASIC_MI01	Basic Fields for Order Items (Main Items)	FARR_IC_BASIC_MI01
<input type="checkbox"/> BASIC_MI02	Basic Fields for Fulfillment Items (Main Items)	FARR_IC_BASIC_MI02
<input type="checkbox"/> BASIC_MI03	Basic Fields for Invoice Items (Main Items)	FARR_IC_BASIC_MI03
<input type="checkbox"/> CA_BASIC_MI	CA Basic Fields for Main Items	FARR_IC_CA_BASIC_MI
<input type="checkbox"/> CA_MI01	CA Fields for Order Items	FARR_IC_CA_MI01
<input type="checkbox"/> COPA_MI01	CO-PA Fields for Order Items (COPACRIT)	FARR_IC_COPA_MI01
<input type="checkbox"/> CRM_MI01	CRM Fields for Order Items	FARR_IC_CRM_MI01
<input type="checkbox"/> SD_MI01	SD Fields for Order Items	FARR_IC_SD_MI01

Figure 2.18 Interface Components

Field Name	Data element	Data Type	Length	Short Description
HEADER_ID	FARR_HEADER_ID	C Data Type	20	Header ID of Source Document for Revenue Accounting Item
ITEM_ID	FARR_ITEM_ID	CHAR	15	Item ID of Source Document for Revenue Accounting Item
KEYPP	FARR_KEYPP	NUMC	3	Subarea for Parallelization
RAIC_TYPE	FARR_RAIC_TYPE	CHAR	2	Revenue Accounting Item Class Type
RAIC	FARR_RAIC	CHAR	4	Revenue Accounting Item Class
STATUS	FARR_RAI_STATUS	CHAR	1	Status of Revenue Accounting Item
EXCHIST	FARR_RAI_EXCHIST	CHAR	1	History Record Exists for Exemption
CHHIST	FARR_CHHIST	CHAR	1	History Exists
BUKRS	BUKRS	CHAR	4	Company Code
QUANTITY	FARR_QUANTITY	QUAN	18	Quantity
QUANTITY_UNIT	FARR_QUANTITY_UNIT	UNIT	3	Unit of Measure
INITIAL_LOAD	FARR_INITIAL_LOAD	CHAR	1	Initial Load
LOG_HANDLE	BALLOGHNDL	CHAR	22	Application Log: Log Handle
MIG_PACKAGE	FARR_MIG_PACKAGE	CHAR	4	Migration Package ID
CREA_USER	FARR_CREA_USER	CHAR	12	User who created the revenue accounting item
CREA_TMSTP UTC	FARR_CREA_TMSTP UTC	DEC	15	Time the revenue accounting item was created

Figure 2.19 Fields Available in Interface Component BASIC_MI

SAP Maintain Revenue Accounting Item Classes

Selected entries Interface Customer Fields Indexes More

Classes of Revenue Accounting Items

Active	RecAccte...	Class Type	Name	Configuration Status
<input type="checkbox"/>	YA01	Order Item	Order Item Class	In Processing

Figure 2.20 Revenue Accounting Item Class Screen

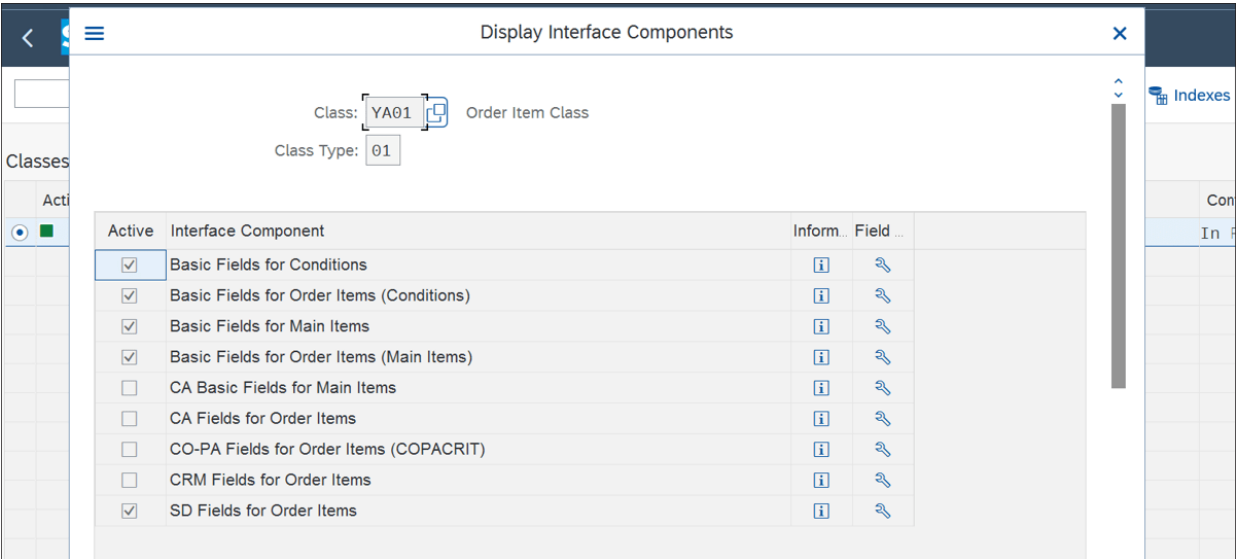


Figure 2.21 Screen to Select Interface Components

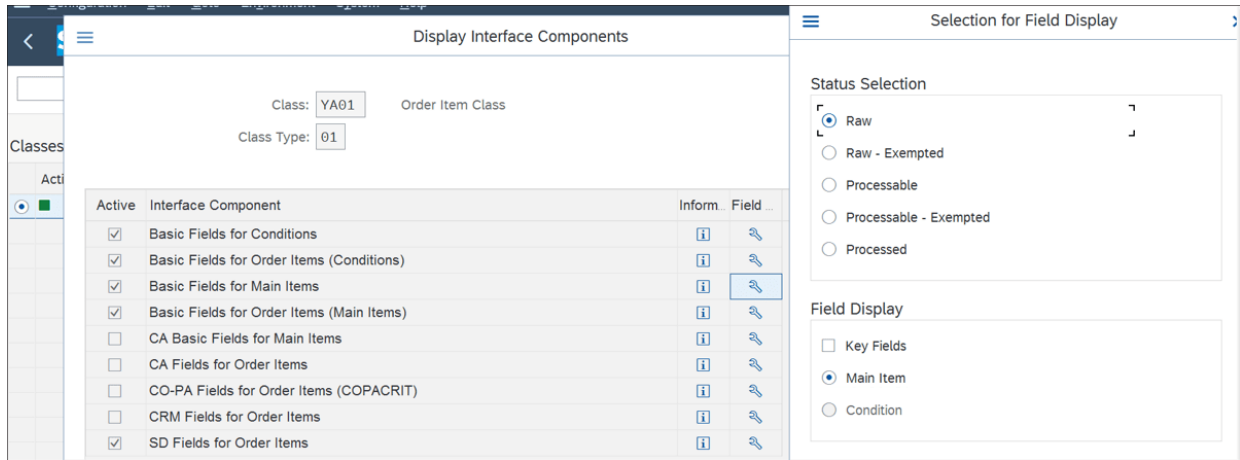


Figure 2.22 Navigating to the Field Details of the Interface Component

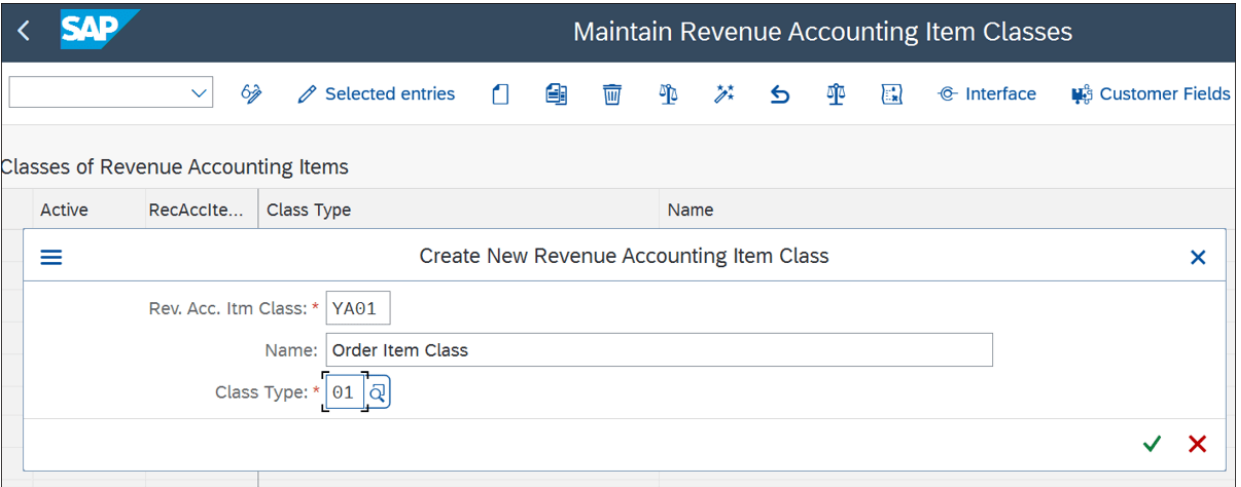


Figure 2.23 Custom Order Item Class Creation

Class: YA01 Order Item Class

Class Type: 01

Active	Interface Component	Inform...	Field ...
<input checked="" type="checkbox"/>	Basic Fields for Conditions		
<input checked="" type="checkbox"/>	Basic Fields for Order Items (Conditions)		
<input checked="" type="checkbox"/>	Basic Fields for Main Items		
<input checked="" type="checkbox"/>	Basic Fields for Order Items (Main Items)		
<input type="checkbox"/>	CA Basic Fields for Main Items		
<input type="checkbox"/>	CA Fields for Order Items		
<input type="checkbox"/>	CO-PA Fields for Order Items (COPACRIT)		
<input type="checkbox"/>	CRM Fields for Order Items		
<input type="checkbox"/>	SD Fields for Order Items		

✓ ↻ Only Modifiable Components

Figure 2.24 Select the Interface Component for YA01

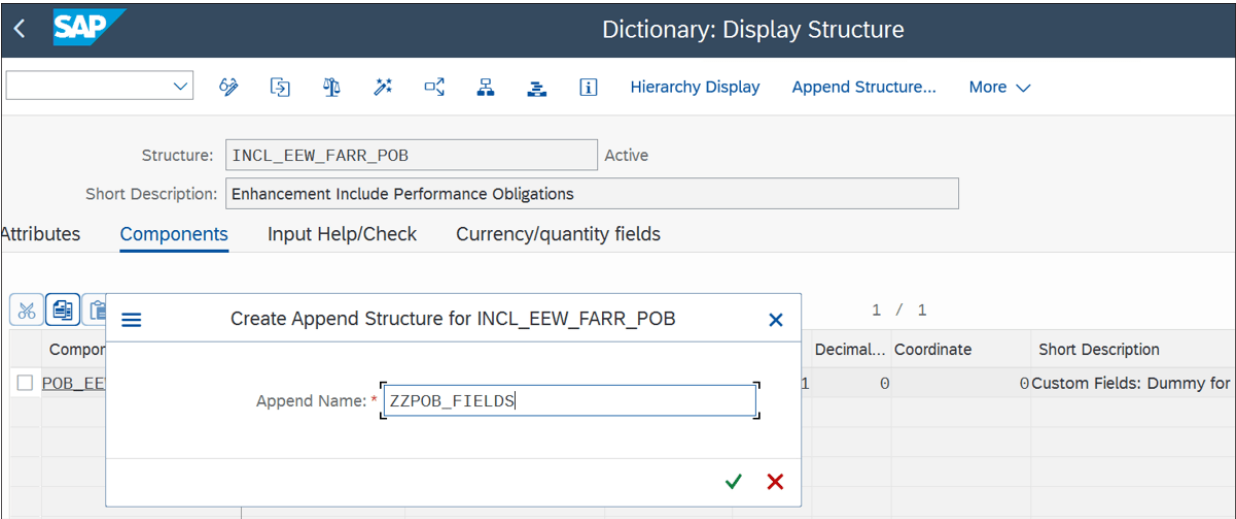


Figure 2.25 Creating an Append Structure for INCL_EEW_FARR_POB

SAP Dictionary: Change Append Structure

Append Structure: ZZPOB_FIELDS New

Short Description: * Additional Fields for POB

Attributes Components Input Help/Check Currency/quantity fields

Built-In Type Show Appending Obj 1 / 2

Component	Typing Method	Component Type	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> ZZPOB_CUR	Types	▼ ZPOB_WAERS	CUKY	5	0		0 POB Currency
<input type="checkbox"/> ZZPOB_STATUS	Types	▼ ZPOB_STATUS	CHAR	9	0		0 POB Status

Figure 2.26 Custom Fields Added to the Append Structure

SAP Dictionary: Display Structure

Structure: INCL_EEW_FARR_POB Active

Short Description: Enhancement Include Performance Obligations

Attributes Components Input Help/Check Currency/quantity fields

1 / 4

Component	Typing Method	Component Type	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> POB_EEW_DUMMY	Types	∨ GFD_DUMMY	CHAR		1	0	∅ Custom Fields: Dummy for Use in Extension In
<input type="checkbox"/> .APPEND	Types	∨ ZZPOB_FIELDS	∅		0	0	∅ Additional Fields for POB
<input type="checkbox"/> ZZPOB_CUR	Types	∨ ZPOB_WAERS	CUKY		5	0	∅ POB Currency
<input type="checkbox"/> ZZPOB_STATUS	Types	∨ ZPOB_STATUS	CHAR		9	0	∅ POB Status

Figure 2.27 Structure INCL_EEW_FARR_POB with Additional Custom Fields

Transparent Table: Active

Short Description:

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> POB_EEW_DUMMY	<input type="checkbox"/>	<input type="checkbox"/>	CFD_DUMMY	CHAR		1	0	0 Custom Fields: Dummy for Use in Extension
<input type="checkbox"/> .APPEND	<input type="checkbox"/>	<input type="checkbox"/>	ZZPOB_FIELDS	STRU		0	0	0 Additional Fields for POB
<input type="checkbox"/> ZZPOB_CUR	<input type="checkbox"/>	<input type="checkbox"/>	ZPOB_WAERS	CUKY		5	0	0 POB Currency
<input type="checkbox"/> ZZPOB_STATUS	<input type="checkbox"/>	<input type="checkbox"/>	ZPOB_STATUS	CHAR		9	0	0 POB Status
<input type="checkbox"/> .INCLUDE	<input type="checkbox"/>	<input type="checkbox"/>	INCL_EEW_FARR_REP	STRU		0	0	0 Enhancement Include Postings
<input type="checkbox"/> REP_EEW_DUMMY	<input type="checkbox"/>	<input type="checkbox"/>	CFD_DUMMY	CHAR		1	0	0 Custom Fields: Dummy for Use in Extension
<input type="checkbox"/> .INCLUDE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_S_POB_REF	STRU		0	0	0 Structure of POB references
<input type="checkbox"/> HI_LEVEL_POB_ID	<input type="checkbox"/>	<input type="checkbox"/>	FARR_HI_LEVEL_POB_	CHAR		16	0	0 Higher-Level Performance Obligation ID

Figure 2.28 Table FARR_D_POB with the New Additional Custom Fields

Structure: INCL_EEW_FARR_REP Active

Short Description: Enhancement Include Postings

Attributes Components Input Help/Check Currency/quantity fields

Built-In Type 1 / 3

Component	Typing Method	Component Type	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> REP_EEW_DUMMY	Types	CFD_DUMMY	CHAR	1	0		Custom Fields: Dummy for Use in Extension Includes
<input type="checkbox"/> .APPEND	Types	ZZREP_FIELDS	---	0	0		Additional Reporting Fields
<input type="checkbox"/> ZZPOB_DESC	Types	ZZDESC	CHAR	35	0		POB Description

Figure 2.29 Structure INCL_EEW_FARR_REP with Additional Fields

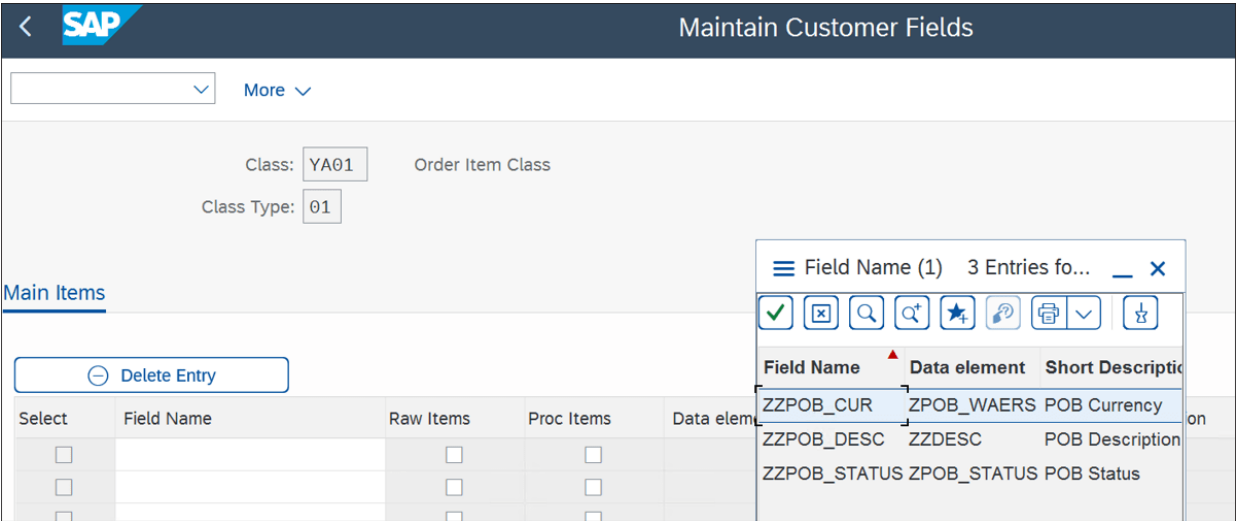


Figure 2.30 Customer Fields for Addition

SAP Maintain Customer Fields

Class: YA01 Order Item Class
 Class Type: 01

Main Items

Delete Entry

Select	Field Name	Raw Items	Proc Items	Data element	Short Description
<input type="checkbox"/>	ZZPOB_CUR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPOB_WAERS	POB Currency
<input type="checkbox"/>	ZZPOB_DESC	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZZDESC	POB Description
<input type="checkbox"/>	ZZPOB_STATUS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPOB_STATUS	POB Status

Figure 2.31 Customer Fields Selected for Class YA01

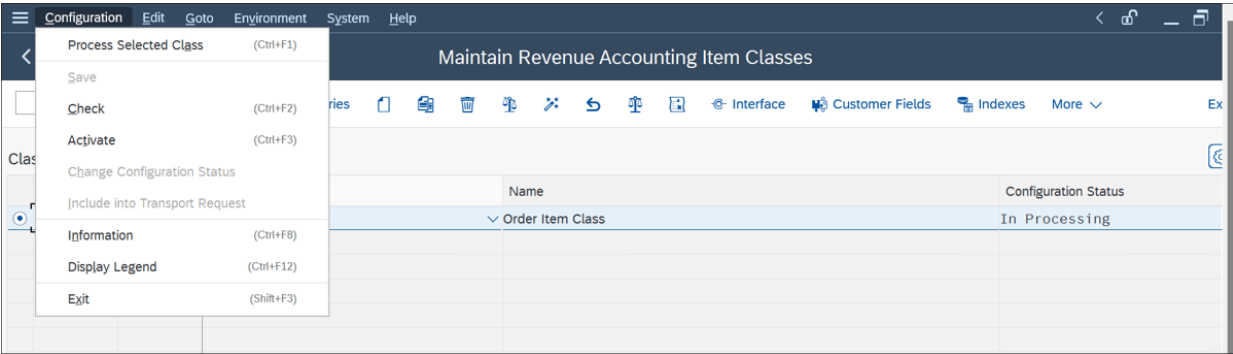



Figure 2.32 Class YA01 Activation


Generate Interfaces for Revenue Accounting Item Classes

Classes of Revenue Accounting

	Generation Sta...	Rec...	Class T
<input checked="" type="radio"/>	◆	YA01	Order
<input type="radio"/>			
<input type="radio"/>			
<input type="radio"/>			
<input type="radio"/>			
<input type="radio"/>			
<input type="radio"/>			

Generation of Results

Statistical Information

Status	Message Text	Counter
<input checked="" type="checkbox"/>	Generated Tables/Structures	12
<input checked="" type="checkbox"/>	Generated Table Indexes	6
<input checked="" type="checkbox"/>	Generated Table Types	12
<input checked="" type="checkbox"/>	Generated Function Groups	1
<input checked="" type="checkbox"/>	Generated Function Modules	11

Figure 2.33 Generation of Class YA01

Transparent Table: /1RA/0YA010MI Active

Short Description: Items for Class YA01 - Raw Data

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

1 /

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		0 Client
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO	CHAR	3	0		0 Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS	CHAR	10	0		0 Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCIY	CHAR	4	0		0 Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		0 Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_TMSTMP	DEC	15	0		0 Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> KEYPP	<input type="checkbox"/>	<input type="checkbox"/>	FARR_KEYPP	NUMC	3	0		0 Subarea for Parallelization
<input type="checkbox"/> RAIC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAIC_TYPE	CHAR	2	0		0 Revenue Accounting Item Class Type
<input type="checkbox"/> RAIC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAIC	CHAR	4	0		0 Revenue Accounting Item Class

Figure 2.34 Raw Items Main Records:
/1RA/0YA010MI

Field	Key	Initia...	Data element	Data Type	Length	Decima...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT		3	0	0 Client
<input type="checkbox"/> KEYPP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_KEYPP	NUMC		3	0	0 Subarea for Parallelization
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO	CHAR		3	0	0 Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS	CHAR		10	0	0 Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCTY	CHAR		4	0	0 Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR		35	0	0 Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_TMSTMP	DEC		15	0	0 Timestamp UTC for Revenue Accounting
<input type="checkbox"/> CONDITION_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KSCHA	CHAR		4	0	0 Condition Type

Figure 2.35 Raw Items Condition Records:
/1RA/0YA010CO

SAP Transfer Revenue Accounting Items

Further Selections More

Selection Data

Rev. Acc. Itm Class:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Sender Component:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Source Item Logical System:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Header ID:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Company Code:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Further Selections Exist:	<input type="checkbox"/>			

Technical Parameters

Number of Intervals:

Block Size For Mass Selection:

Simulation Mode:

Dialog Mode:

Synchronous Call:

Figure 2.36 Transfer Revenue Accounting Items

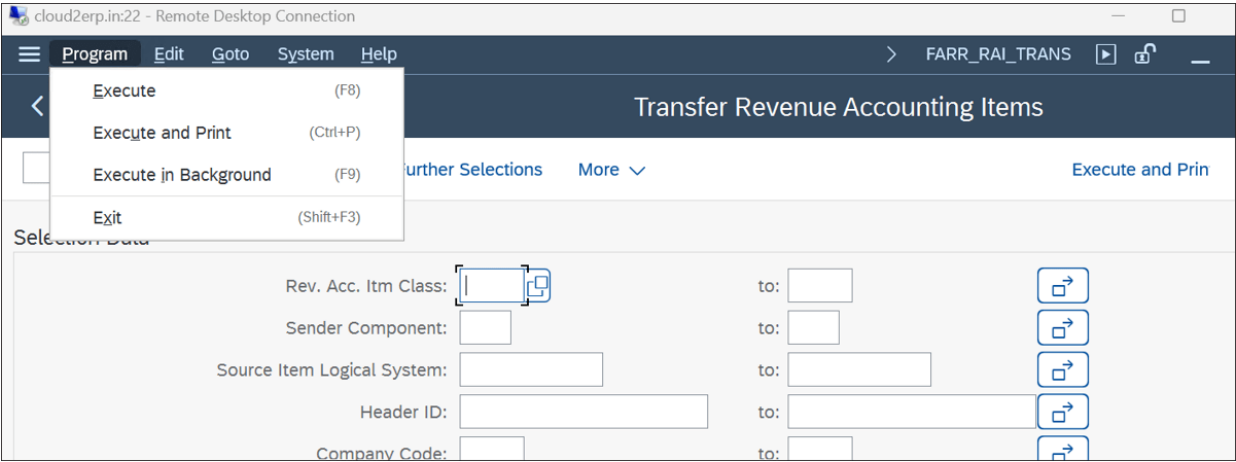









Figure 2.37 Transaction FARR_RAI_TRANS Executing in the Background

 Manage Revenue Accounting Items

   Further Selections  Personalize  More 

Kind of Selection

Kind of Selection:

Item Status

Item Status:

Source Documents







Component:	<input type="text"/>	to:	<input type="text"/>	
Logical System:	<input type="text"/>	to:	<input type="text"/>	
Type:	<input type="text"/>	to:	<input type="text"/>	
Header ID (Orders):	<input type="text"/>	to:	<input type="text"/>	
Item ID (Orders):	<input type="text"/>	to:	<input type="text"/>	
Date:	<input type="text"/>	to:	<input type="text"/>	

Figure 2.38 Transaction FARR_RAI_MON for Transferring of RAIs from Raw to Processable

The screenshot shows the SAP 'Analysis of Items' interface. At the top, there is a menu bar with 'Analysis', 'Edit', 'Goto', 'Environment', 'System', and 'Help'. Below the menu is a toolbar with buttons for 'Refresh', 'Statistics', 'Simulate Transfer', 'Transfer', 'Process', 'Change', 'Exempt', and 'Restore'. The main area displays two filters: 'Main Item (53)' and 'Condition Item (52)'. Below the filters is a toolbar with various icons for search, filter, and display options. The main data area is a table with the following columns: ItemStatus, Hist, Exist, Error, Ex.His, Ex, Send, Com, SourceSys, SrcItem, Source Item ID, Subarea, RevAcc, Header ID, ItemID, Ref. Type, Reference ID, and Customer. The table contains 10 rows of data, each with a checkbox in the first column and a status indicator (red or green dot) in the 'Error' column.

ItemStatus	Hist	Exist	Error	Ex.His	Ex	Send	Com	SourceSys	SrcItem	Source Item ID	Subarea	RevAcc	Header ID	ItemID	Ref. Type	Reference ID	Customer	Pa
<input type="checkbox"/>			●			ZX		DS4CLNT0...	ZACI	S189004366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	
<input type="checkbox"/>			■			ZX		DS4CLNT0...	ZACI	S189004366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	
<input type="checkbox"/>			■			ZX		DS4CLNT0...	ZACI	S189004366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	
<input type="checkbox"/>			●			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
<input type="checkbox"/>			●			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
<input type="checkbox"/>			■			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
<input type="checkbox"/>			■			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
<input type="checkbox"/>			●			ZX		DS4CLNT0...	ZACI	S189004366000030	367	ZA01	S189004366	30	ZOP	00S189004366	20064918	
<input type="checkbox"/>			●			ZX		DS4CLNT0...	ZACI	S189004366000030	367	ZA01	S189004366	30	ZOP	00S189004366	20064918	

Figure 2.39 Choosing the RAIs to Transfer in Transaction FARR_RAI_MON

SAP Dictionary: Display Table

Transparent Table: **/1RA/0YA012MI** Active
 Short Description: Items for Class YA01 - Processable

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		0 Client
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO	CHAR	3	0		0 Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS	CHAR	10	0		0 Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCIY	CHAR	4	0		0 Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		0 Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_TMSTMP	DEC	15	0		0 Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> KEYPP	<input type="checkbox"/>	<input type="checkbox"/>	FARR_KEYPP	NUMC	3	0		0 Subarea for Parallelization
<input type="checkbox"/> RAIC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAIC_TYPE	CHAR	2	0		0 Revenue Accounting Item Class Type

Figure 2.40 Processable Items Main Records: Table /1RA/0YA012MI

Transparent Table: /1RA/0YA012CO Active
Short Description: Items for Class YA01 - Processable

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		0 Client
<input type="checkbox"/> KEYPP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_KEYPP	NUMC	3	0		0 Subarea for Parallelization
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO	CHAR	3	0		0 Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS	CHAR	10	0		0 Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCTY	CHAR	4	0		0 Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		0 Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_TMSTMP	DEC	15	0		0 Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> CONDITION_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KSCHA	CHAR	4	0		0 Condition Type
<input type="checkbox"/> PL_ACCOUNT	<input type="checkbox"/>	<input type="checkbox"/>	FARR_PL_ACCOUNT	CHAR	10	0		0 Profit and Loss Account

Figure 2.41 Processable Items Condition Records:
Table /1RA/0YA012CO

SAP Process Revenue Accounting Items

Execute and Print

Further Selections More

Selection Data

Rev. Acc. Itm Class:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Sender Component:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Source Item Logical System:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Header ID:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Reference Type:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Reference ID:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Company Code:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Further Selections Exist:	<input type="checkbox"/>			

Technical Parameters

Number of Intervals:	<input type="text" value="2"/>
Block Size For Mass Selection:	<input type="text" value="1.000"/>
Dialog Mode:	<input type="checkbox"/>
Synchronous Call:	<input type="checkbox"/>

Figure 2.42 Process RAIs

Transparent Table: /1RA/0YA014MI Active

Short Description: Items for Class YA01 - Processed

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

1 / 89

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		0 Client
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO	CHAR	3	0		0 Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS	CHAR	10	0		0 Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCTY	CHAR	4	0		0 Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		0 Source Item ID
<input type="checkbox"/> TIMESTAMP_UTC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAI_TMSTMP	DEC	15	0		0 Timestamp UTC for Revenue Accounting Item
<input type="checkbox"/> KEYPP	<input type="checkbox"/>	<input type="checkbox"/>	FARR_KEYPP	NUMC	3	0		0 Subarea for Parallelization
<input type="checkbox"/> RAIC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAIC_TYPE	CHAR	2	0		0 Revenue Accounting Item Class Type
<input type="checkbox"/> RAIC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAIC	CHAR	4	0		0 Revenue Accounting Item Class

Figure 2.43 Processed Items Main Records: Table /1RA/0YA014MI

SAP Dictionary: Display Table

Transparent Table: /1RA/0YA014CO Active
 Short Description: Items for Class YA01 - Processed

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDI	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDI	CLNT	3	0		0 Client
<input type="checkbox"/> KEYPP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_KEYPP	NUMC	3	0		0 Subarea for Parallelization
<input type="checkbox"/> SRCDOC_COMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO	CHAR	3	0		0 Sender Component of Source Item
<input type="checkbox"/> SRCDOC_LOGSYS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS	CHAR	10	0		0 Logical System of the Source Item
<input type="checkbox"/> SRCDOC_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCTY	CHAR	4	0		0 Source Document Item Type
<input type="checkbox"/> SRCDOC_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_RAI_SRCID	CHAR	35	0		0 Source Item ID
<input type="checkbox"/> CONDITION_TYPE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	KSCHA	CHAR	4	0		0 Condition Type
<input type="checkbox"/> TIMESTAMP_UTC	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAI_TMSTMP	DEC	15	0		0 Timestamp UTC for Revenue Accounting Item

Figure 2.44 Processed Items Condition Records:
Table /1RA/0YA014CO

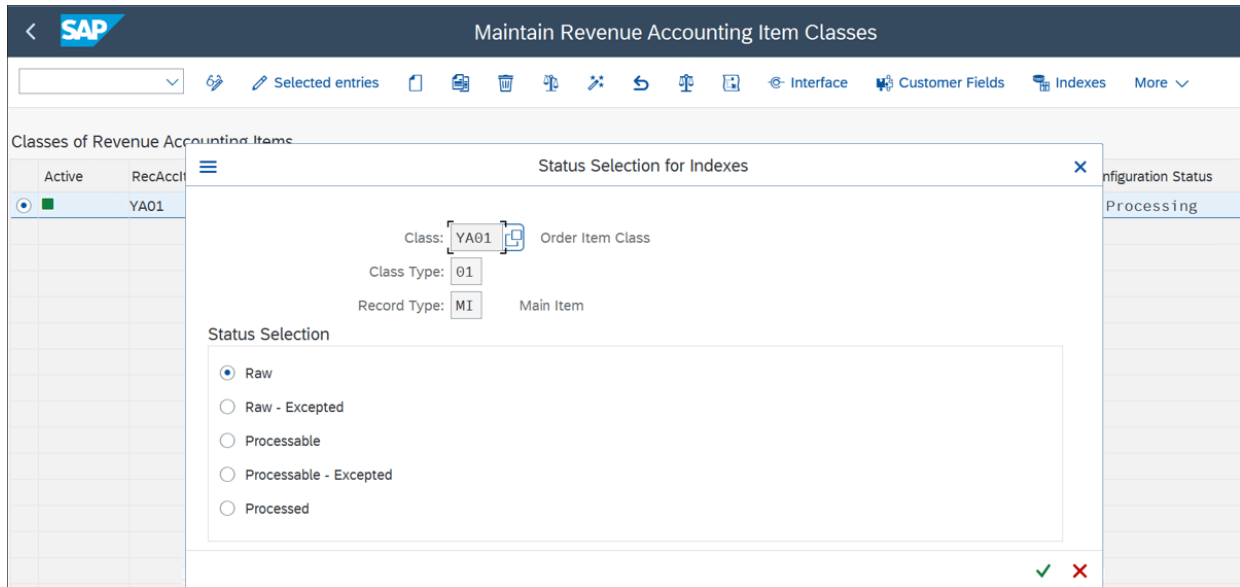


Figure 2.45 Creating Indexes on RAI Tables

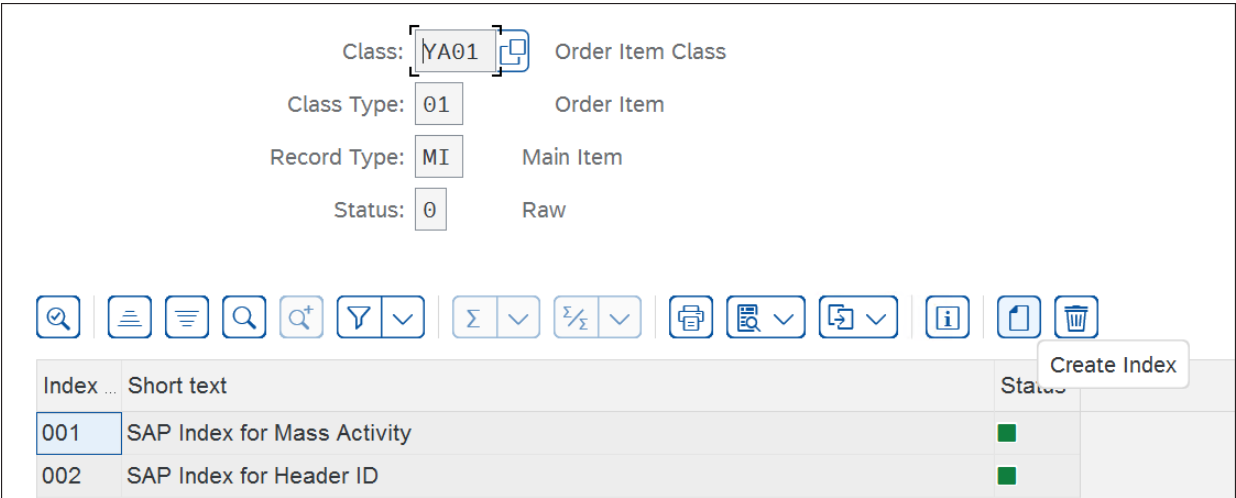


Figure 2.46 Creating the Index

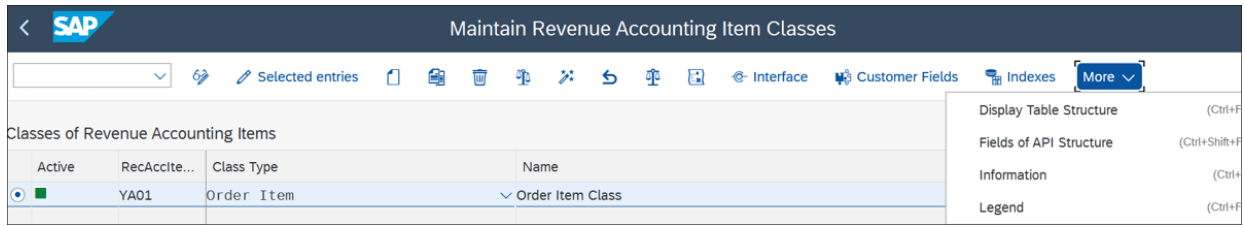


Figure 2.47 Display Table Structure

SAP Display Table Structure

Other Status More

Class: **YA01** Order Item Class

Class Type: **01** Order Item

Status: **2** Processable

Main Items Conditions

Field Name	Tabl...	K. C.	Data element	Domain	A. Data Type	No. of ...
MANDT	1	<input checked="" type="checkbox"/>	MANDT		C CLNT	3
SRCDOC_COMP	2	<input checked="" type="checkbox"/>	FARR_RAI_SRCCO		C CHAR	3
SRCDOC_LOGSYS	3	<input checked="" type="checkbox"/>	FARR_RAI_SRCLS		C CHAR	10
SRCDOC_TYPE	4	<input checked="" type="checkbox"/>	FARR_RAI_SRCTY		C CHAR	4
SRCDOC_ID	5	<input checked="" type="checkbox"/>	FARR_RAI_SRCID		C CHAR	35
TIMESTAMP.UTC	6	<input checked="" type="checkbox"/>	FARR_RAI_TMSTMP		P DEC	15
KEYPP	7	<input type="checkbox"/>	FARR_KEYPP		N NUMC	3
RAIC_TYPE	8	<input type="checkbox"/>	FARR_RAIC_TYPE		C CHAR	2
RAIC	9	<input type="checkbox"/>	FARR_RAIC		C CHAR	4

Figure 2.48 Display Table Structure

▼

🔗 Other Status
More ▼

Class:

Order Item Class

Class Type:

Order Item

Status:

Raw

Main Items
Conditions

Field Name	Tabl...	K. C.	Data element
VBELV	80	<input type="checkbox"/> <input type="checkbox"/>	VBELV
VBTYP	81	<input type="checkbox"/> <input type="checkbox"/>	FARR_VBTYPL
VTWEG	82	<input type="checkbox"/> <input type="checkbox"/>	VTWEG
WERKS	83	<input type="checkbox"/> <input type="checkbox"/>	WERKS_EXT

Figure 2.49 Custom Fields Missing in the Raw Table for Main Records

SAP Display Table Structure

Other Status More

Class: YA01 Order Item Class

Class Type: 01

Status: 2 Processable

Main Items Conditions

Field Name	Tabl...	K. C.	Data element	Domain
VBELV	81	<input type="checkbox"/>	VBELV	
VBTYP	82	<input type="checkbox"/>	FARR_VBTYPL	
VTWEG	83	<input type="checkbox"/>	VTWEG	
WERKS	84	<input type="checkbox"/>	WERKS_EXT	
ZZPOB_CUR	85	<input checked="" type="checkbox"/>	ZPOB_WAERS	WAERS
ZZPOB_DESC	86	<input checked="" type="checkbox"/>	ZZDESC	CHAR35
ZZPOB_STATUS	87	<input checked="" type="checkbox"/>	ZPOB_STATUS	CHAR9

Figure 2.50 Custom Fields Available in the Processable Status Table for Main Records

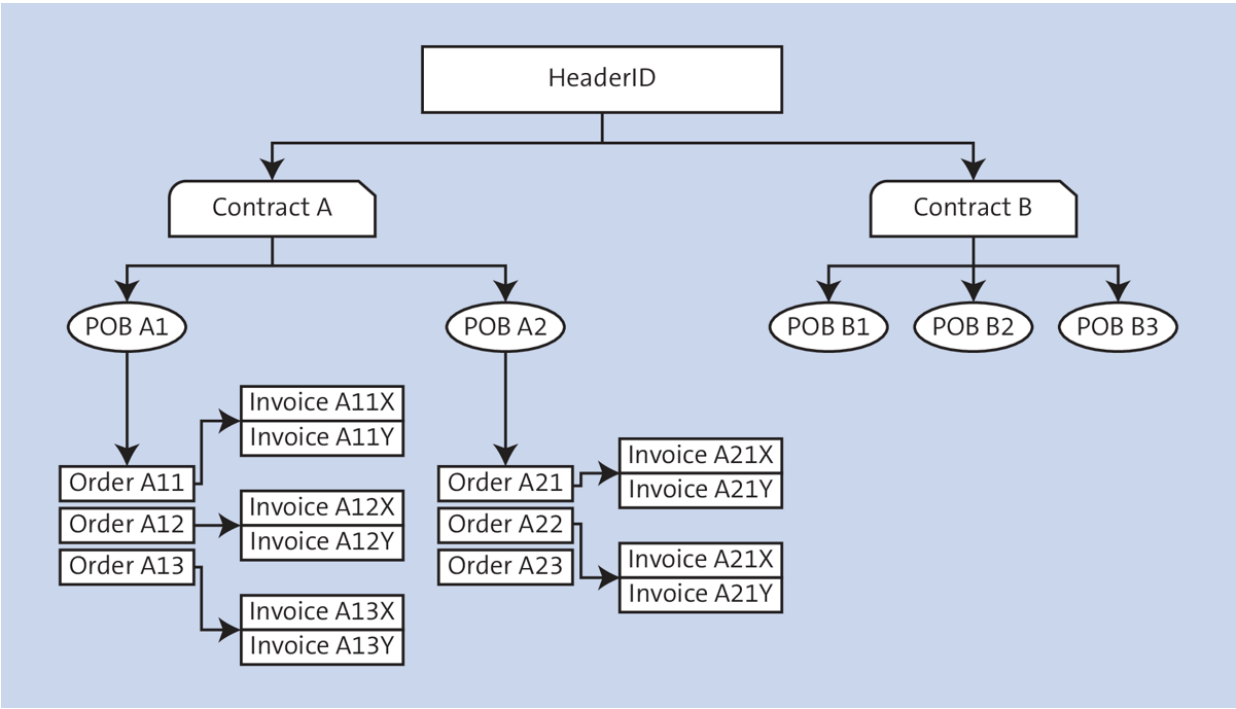


Figure 2.51 Hierarchies and Relationships

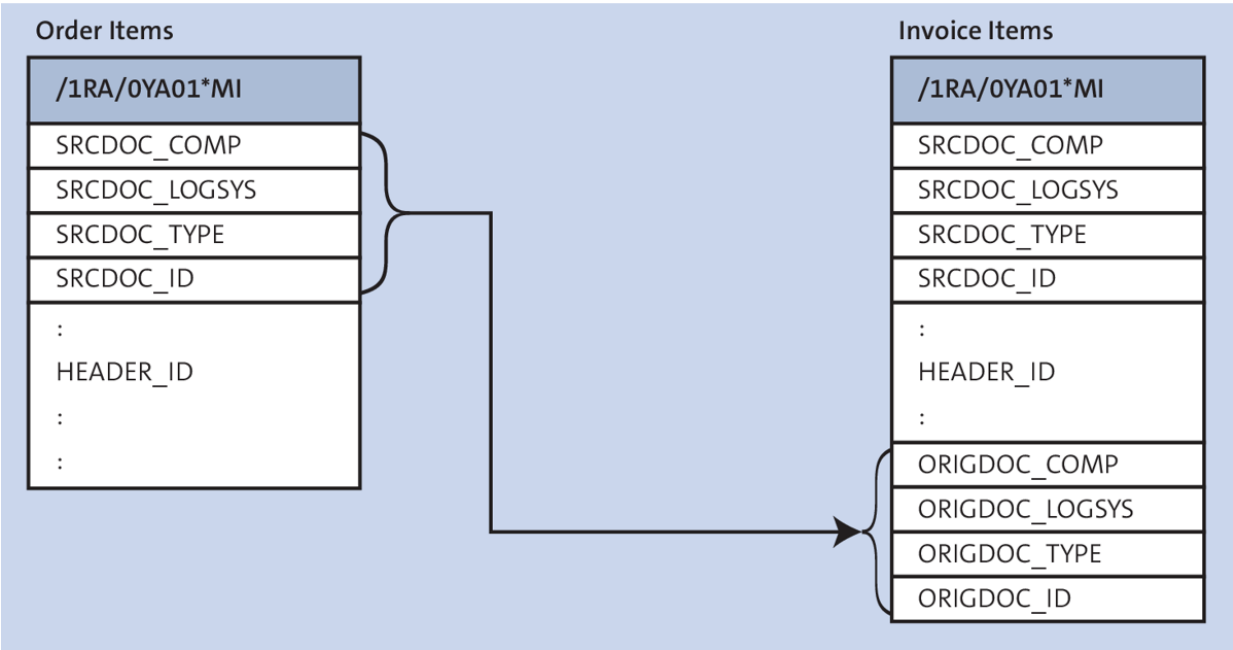


Figure 2.52 Orders to Invoice Relationship

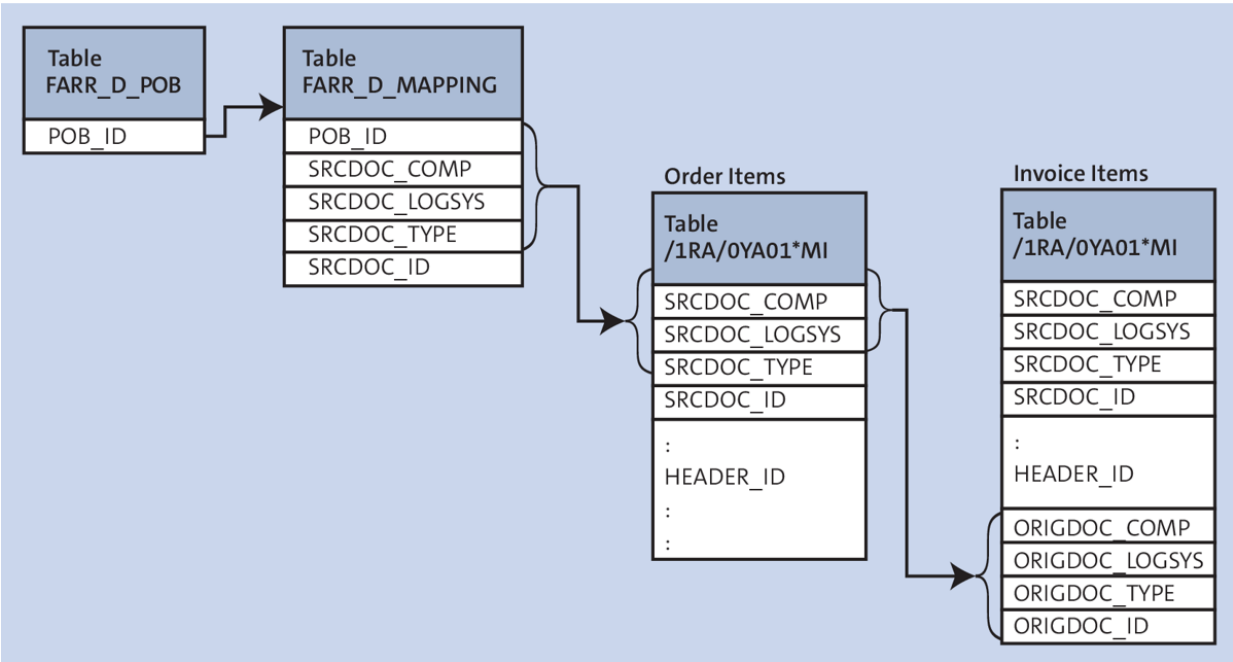


Figure 2.53 POB_ID to Processed Invoice RAIs

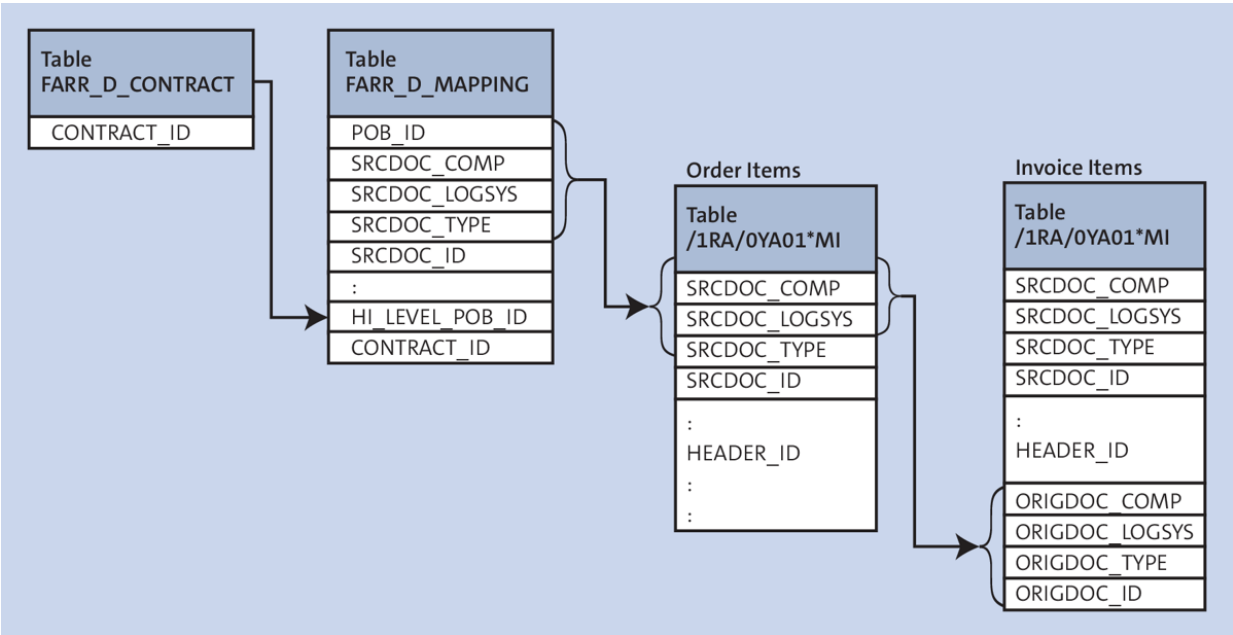


Figure 2.54 Contracts to Processed Invoice RAIs

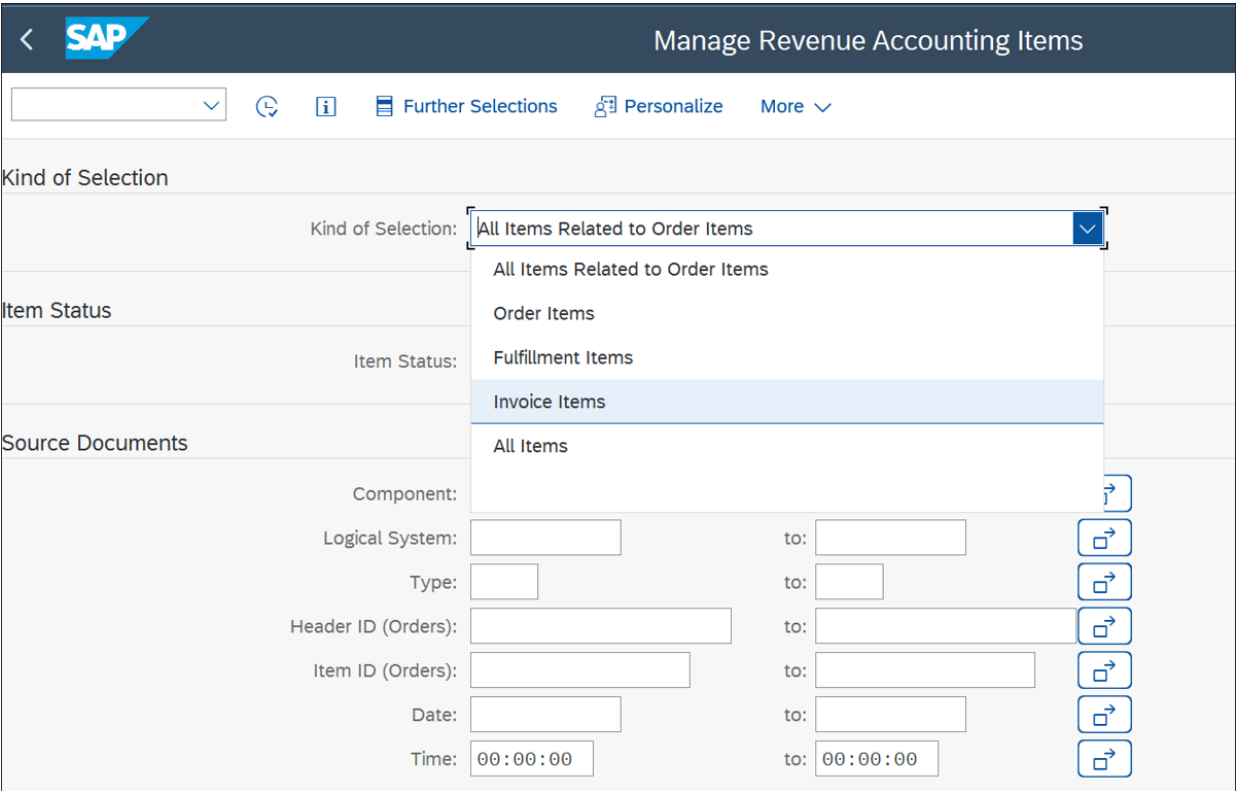


Figure 2.55 Transaction FARR_RAI_MON Screen for Selection

The screenshot displays the SAP 'Analysis of Items' interface. At the top, there is a menu bar with options: Analysis, Edit, Goto, Environment, System, Help. Below the menu is the SAP logo and the title 'Analysis of Items'. A toolbar contains buttons for Refresh, Statistics, Simulate Transfer, Transfer, Process, Change, Exempt, Restore, and More. Below the toolbar, there are two filters: 'Main Item (53)' and 'Condition Item (52)'. A secondary toolbar includes icons for search, list, filter, sort, and other functions. The main area is a table with the following columns: ItemStatus, Hist, Exist, Error, Ex.His, Ex, Send, Com, SourceSys, SrcItem, Source Item ID, Subarea, RevAcc, Header ID, ItemID, Ref. Type, Reference ID, Customer, and Pa. The table contains 10 rows of data, each representing an item with various attributes and status indicators (red and green dots).

ItemStatus	Hist	Exist	Error	Ex.His	Ex	Send	Com	SourceSys	SrcItem	Source Item ID	Subarea	RevAcc	Header ID	ItemID	Ref. Type	Reference ID	Customer	Pa
			●			ZX		DS4CLNT0...	ZACI	S189004366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	
			■			ZX		DS4CLNT0...	ZACI	S189004366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	
			●			ZX		DS4CLNT0...	ZACI	S189004366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	
			●			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
			●			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
			■			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
			■			ZX		DS4CLNT0...	ZACI	S189004366000020	367	ZA01	S189004366	20	ZOP	00S189004366	20064918	
			●			ZX		DS4CLNT0...	ZACI	S189004366000030	367	ZA01	S189004366	30	ZOP	00S189004366	20064918	
			●			ZX		DS4CLNT0...	ZACI	S189004366000030	367	ZA01	S189004366	30	ZOP	00S189004366	20064918	

Figure 2.56 List of Items Available for Transfer, Process, Exempt, and Restore in Transaction FARR_RAI_MON

The screenshot shows the SAP Fiori interface for the transaction FARR_RAI_MON. The main content area displays 'Statistics of the Selected Items' with a table of statistical information. The table has three columns: 'Status', 'Message Text', and 'Counter'. The first four rows represent individual item statuses, and the last two rows are summary rows for 'Total Main Items' and 'Total Condition Items'. The first row is highlighted with a blue selection box.

Status	Message Text	Counter
	RAI class ZA01 Main Items Status 'RAW'	38
	RAI class ZA01 Condition Items Status 'RAW'	34
	RAI class ZA01 Main Items Status 'PROCESSABLE'	15
	RAI class ZA01 Condition Items Status 'PROCESSABLE'	18
Σ	Total Main Items	53
Σ	Total Condition Items	52

Figure 2.57 Statistics as in Transaction FARR_RAI_MON

RevAccCl	Rec. Type	Status	Field Name	FieldAttr
<input type="checkbox"/> ZA01	All Record Types	▼ All Statuses	PRCTR	

Figure 2.58 Define Modifiable Fields for Revenue Accounting Items

SAP Change Revenue Accounting Items

More

Hand Down Changes

Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Type	Reference ID	Customer	Partner	CoCd	Profit Center	ProcTime	Final Time	Finall...	Targ...
J4366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918		1010	CH04212701	0	0		
J4366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918		1010	CH04212701	0	0		
J4366000010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918		1010	CH04212701	0	0		

Figure 2.59 Modifiable Fields in RAI in Transaction FARR_RAI_MON

I1 I2 I3	ED Log Text
<ul style="list-style-type: none"> ❗ ❗ ❗ ❗ ❗ ❗ ❗ ❗ 	<p>Request: convert Table /■■■■/ORA_0021 (■■■■179/18.03.22/16:19)</p> <p>Process: mnr8ap0401_20_27082</p> <p>Test activation of Table /■■■■/ORA_0021</p> <p>Test activation of Table /■■■■/ORA_0021 successful</p> <p>⚠️ A restart log already exist for table /■■■■/ORA_0021</p> <p>Conversion of table /■■■■/ORA_0021 was restarted</p> <p>Table /■■■■/ORA_0021 is converted using shadow field updates</p> <p>The conversion is continued at step 2</p> <p>Type of conversion: T -> T</p>
<ul style="list-style-type: none"> ❗ ❗ 	<p>Beginning of Step /■■■■/ORA_0021-STEP2 (16:19:55)</p> <p>Renaming of table /■■■■/ORA_0021 to QCM /■■■■/ORA_00 on the database</p> <p>❗ TABL QCM /■■■■/ORA_00 already exist</p> <p>❗ Renaming of table /■■■■/ORA_0021 to QCM /■■■■/ORA_00 failed</p>

Figure 2.60 Issue Log

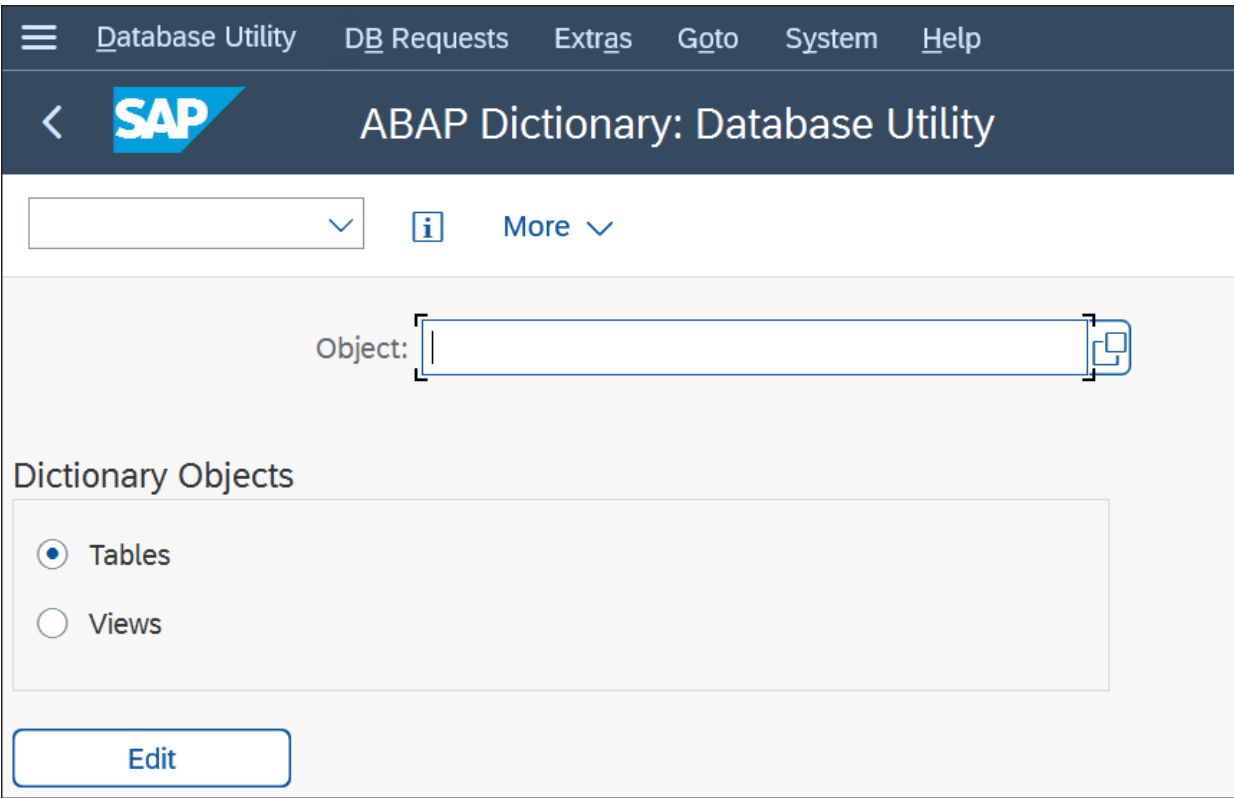


Figure 2.61 Transaction SE14 for Deleting QCM Tables

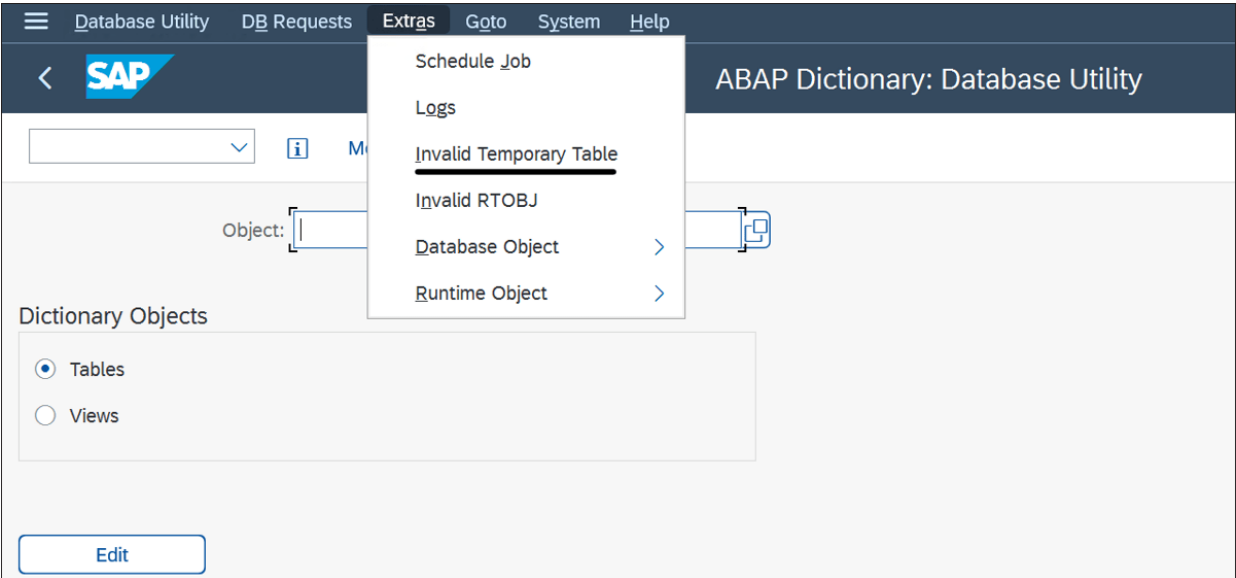


Figure 2.62 Select Invalid Temporary Tables

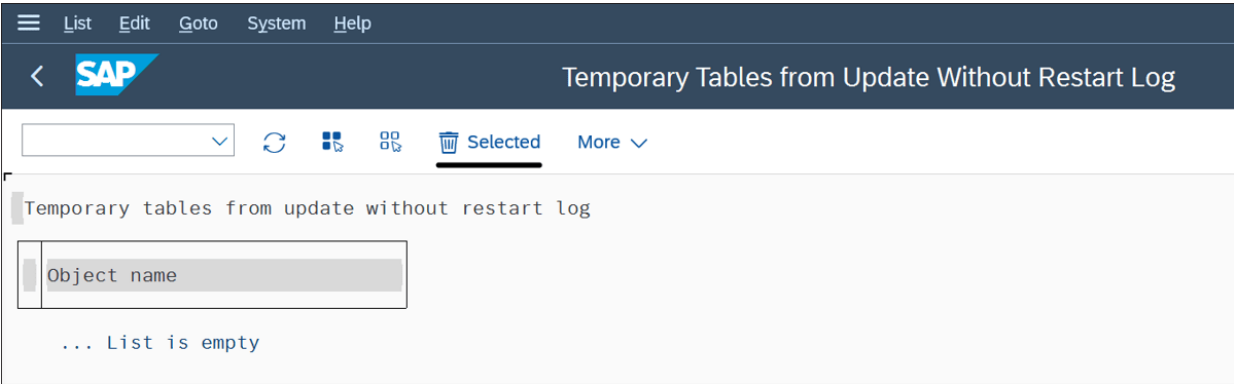


Figure 2.63 Temporary Tables Display and Deletion

Activate Integration Component for Revenue Account

Revenue Account

Destination: [NONE]

Figure 3.1 Enabling Integration between Sales and Distribution and RAR

Activate Functions to Integrate with Revenue Accounting												
SOrg.	Date	MBP x PI	Date	SvcFulfill	Date	IB Fulfill	Date	PI Fulfill	Date	POD Fulf	Date	AD Fulfill



Figure 3.2 Integration with Additional Functions

Activate Item Categories for Revenue Accounting					
	SOrg.	SaTy	ItCa	Type	Package ID
<input type="checkbox"/>	AU40	ZCRM	ZG2N	Credit/Debit Memos with reference to	
<input type="checkbox"/>	AU40	ZDRM	ZL2N	Credit/Debit Memos with reference t...	
<input type="checkbox"/>	AU40	ZLV	L2N	Credit/Debit Memos with reference t...	
<input type="checkbox"/>	AU40	ZMRO	ZLMP	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZLMT	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZLPW	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZMDS	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZMMF	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZMMS	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZMSH	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZPMF	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZPMT	Relevant for Revenue Accounting	
<input type="checkbox"/>	AU40	ZMRO	ZRRL	Relevant for Revenue Accounting	
<input type="checkbox"/>	ID40	ZCRM	ZG2N	Credit/Debit Memos with reference t...	
<input type="checkbox"/>	ID40	ZDRM	ZL2N	Credit/Debit Memos with reference t...	

Figure 3.3 Integration between Sales Documents and RAR

Implementing Class

Interface:

Implementing Class:  

Method	Short Description
IF_FARRIC_ORDER~ORDER_DATA_TO_ARL	Modify Sales Order Fields
IF_FARRIC_ORDER~ORDER_DATA_TO_ARL_CUSTOM	Modify Customer-Specific Sales Order Fie...
IF_FARRIC_ORDER~CLEAR_RELTYPE_FLAG	Clear the FARR_RELTYPE flag for item
IF_FARRIC_ORDER~DETERMINE_MIG_PACKAGE	Determine Migration Package ID
IF_FARRIC_ORDER~EXCLUDE_CONDITIONS	select conditions not to be trnsfered to Re...
IF_FARRIC_ORDER~INCLUDE_SDPI_CONDITIONS	select SDPI condtions to be transferred to...
IF_FARRIC_ORDER~INCLUDE_CONDITIONS	select conditions to be transferred to Reve...
IF_FARRIC_ORDER~ENABLE_CURRENCY_CHANGE	Enable the change of order currency

Figure 3.4 Clearing of Relevancy Flag

Dialog Structure		Interface Components for RAI Classes		
<ul style="list-style-type: none"> ▼ Interface Components for RAI Class └─ Assigned Structures └─ Prerequisite Components └─ Program Enhancements └─ Key Fields └─ Assign Components to Class Type 	<input type="checkbox"/>	Interface Component	Description of Interface Component	Data Element for Documentation
	<input type="checkbox"/>	BASIC_CO	Basic Fields for Conditions	FARR_IC_BASIC_CO
	<input type="checkbox"/>	BASIC_CO01	Basic Fields for Order Items (Conditions)	FARR_IC_BASIC_CO01
	<input type="checkbox"/>	BASIC_MI	Basic Fields for Main Items	FARR_IC_BASIC_MI
	<input type="checkbox"/>	BASIC_MI01	Basic Fields for Order Items (Main Items)	FARR_IC_BASIC_MI01
	<input type="checkbox"/>	BASIC_MI02	Basic Fields for Fulfillment Items (Main Items)	FARR_IC_BASIC_MI02
	<input type="checkbox"/>	BASIC_MI03	Basic Fields for Invoice Items (Main Items)	FARR_IC_BASIC_MI03
	<input type="checkbox"/>	CA_BASIC_MI	CA Basic Fields for Main Items	FARR_IC_CA_BASIC_MI
	<input type="checkbox"/>	CA_MI01	CA Fields for Order Items	FARR_IC_CA_MI01
	<input type="checkbox"/>	COPA_MI01	CO-PA Fields for Order Items (COPACRIT)	FARR_IC_COPA_MI01
	<input type="checkbox"/>	CRM_MI01	CRM Fields for Order Items	FARR_IC_CRM_MI01
	<input type="checkbox"/>	SD_MI01	SD Fields for Order Items	FARR_IC_SD_MI01

Figure 3.5 Interface Components

Component: <input type="text" value="SD_MI01"/>			
Description: <input type="text" value="SD Fields for Order Items"/>			
Assigned Structures			
Rec. Type	Status	Structure	Condit. Active
<input type="checkbox"/> Main Item	▼ All Statuses	▼ FARR_S_ICMI01_SD	<input type="checkbox"/>
<input type="checkbox"/> Main Item	▼ Raw	▼ FARR_S_ICMI01_SD0	<input type="checkbox"/>
<input type="checkbox"/> Main Item	▼ Raw - Exempted	▼ FARR_S_ICMI01_SD0	<input type="checkbox"/>

Figure 3.6 Assigned Structures

Component:

Program Enhancements

Ev	No	Method for Interface Component
<input type="checkbox"/> Final Check Before Saving Processable Items	0	RAI2_CHECK_SD_MI01

Figure 3.7 Enhancement of Data Method

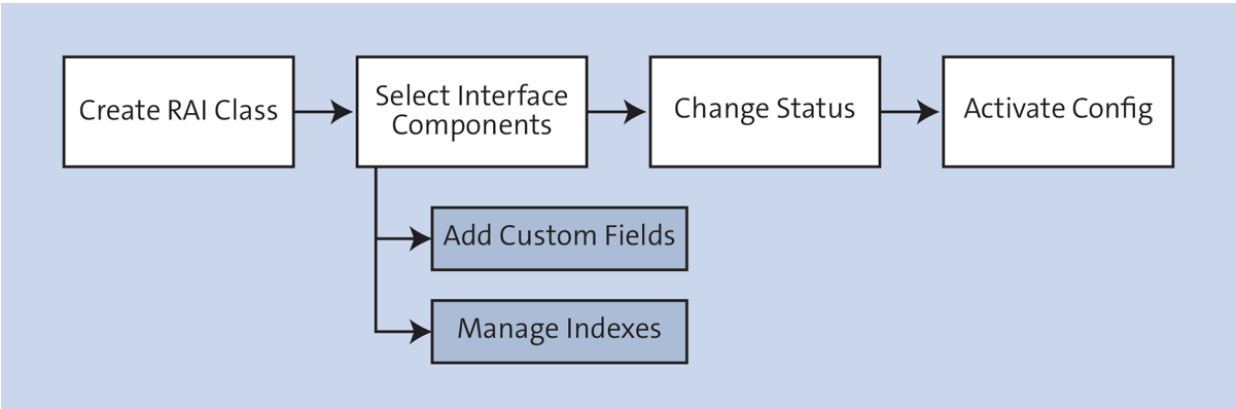


Figure 3.8 Steps for Creation of RAI Classes

Classes of Revenue Accounting Items						
	Active	RecAcclte...	Class Type	Name	Configuration Status	Generatn Trgt System
<input type="radio"/>	<input checked="" type="checkbox"/>	SD01	Order Item	PG RAI Class - Order Item.	Transportable	<input checked="" type="checkbox"/>
<input type="radio"/>	<input checked="" type="checkbox"/>	SD02	Fulfillment Item	PG RAI Class - Fulfilment Item	Transportable	<input checked="" type="checkbox"/>
<input type="radio"/>	<input checked="" type="checkbox"/>	SD03	Invoice Item	PG RAI Class - Invoice Item	Transportable	<input checked="" type="checkbox"/>

Figure 3.9 Setup of RAI Classes

Display Interface Components

Class: PG RAI Class - Order Item.

Class Type:

Active	Interface Component	Inform...	Field ...
<input checked="" type="checkbox"/>	Basic Fields for Conditions	i	key
<input checked="" type="checkbox"/>	Basic Fields for Order Items (Conditions)	i	key
<input checked="" type="checkbox"/>	Basic Fields for Main Items	i	key
<input checked="" type="checkbox"/>	Basic Fields for Order Items (Main Items)	i	key
<input type="checkbox"/>	CA Basic Fields for Main Items	i	key
<input type="checkbox"/>	CA Fields for Order Items	i	key
<input checked="" type="checkbox"/>	CO-PA Fields for Order Items (COPACRIT)	i	key
<input type="checkbox"/>	CRM Fields for Order Items	i	key
<input checked="" type="checkbox"/>	SD Fields for Order Items	i	key

Figure 3.10 Interface Component Selection



Figure 3.11 Perform Consistency Check

Generation St...	Rec...	Class Type	Name	Release Status	Generation Date	Generated ...	Generated By
<input type="radio"/>	<input checked="" type="checkbox"/>	SD01	Order Item	PG RAI Class - Order Item.		08.04.2023 22:49:54	DDIC
<input type="radio"/>	<input checked="" type="checkbox"/>	SD02	Fulfillment Item	PG RAI Class - Fulfilment Item		21.02.2022 09:44:36	DDIC
<input type="radio"/>	<input checked="" type="checkbox"/>	SD03	Invoice Item	PG RAI Class - Invoice Item		21.02.2022 09:59:05	DDIC
<input type="radio"/>							

Figure 3.12 Generation of Interfaces Option

The screenshot shows the SAP interface for 'Change View "Upload Rules": Overview'. At the top, there is a navigation bar with the SAP logo and a back arrow. Below the navigation bar is a toolbar with a search field, 'New Entries' button, and several icons for actions like print, zoom, and refresh. The main content area is titled 'Upload Rules' and contains a table with the following data:

	RevAccCl	Upload Rule
<input type="checkbox"/>	ZA01	Create Items as Processable Revenue Accounting Items
<input type="checkbox"/>	ZA03	Create Items as Processable Revenue Accounting Items

Figure 3.13 Upload Rules

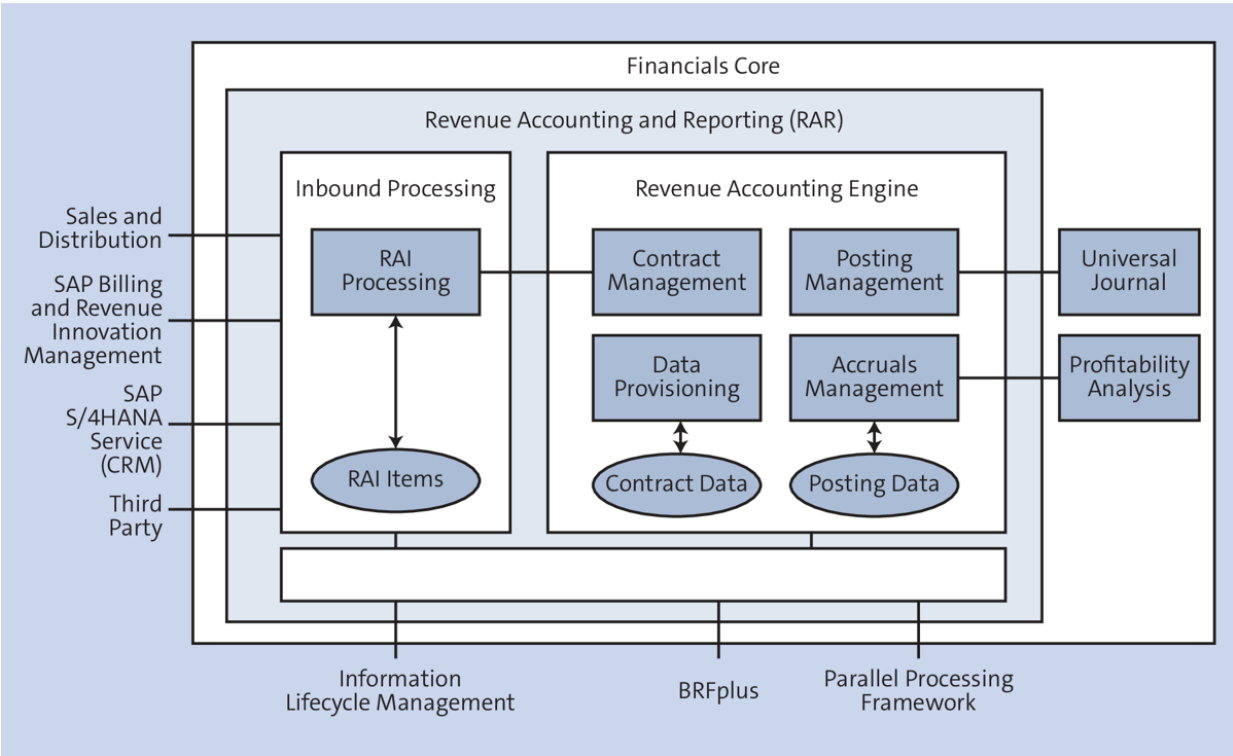


Figure 3.14 Architecture of Optimized Inbound Processing

Interface: IF_FARR_BADI_DET_N_IP_VERSION Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters Exceptions

Method	Level	M...	Description
DETERMINE_VERSIONS	Instance Method		Determination of Inbound processing Version

Figure 3.15 BAdI IF_FARR_BADI_DET_N_IP_VERSION

SAP Class Builder: Display Interface IF_FARR_BADI_DET_N_IP_VERSION

Interface: IF_FARR_BADI_DET_N_IP_VERSION Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: DETERMINE_VERSIONS

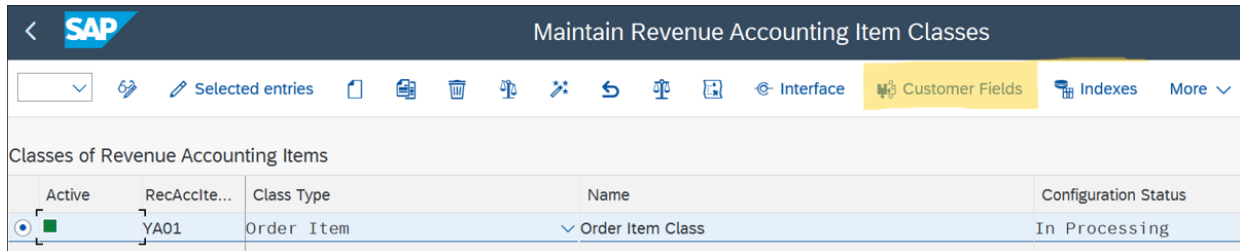
← Methods Exceptions Properties

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
ITS_ORDER_ITEM	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	TY_TS_ORDER_ITEM_BADI
CTS_SRCDOC_VERSION	Changing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TS_SRCDOC_IP_VERSION

Figure 3.16 Method DETERMINE_VERSIONS

```
3
4 INTERFACES if_badi_interface .
5
6 TYPES: BEGIN OF ty_s_order_item_badi,
7         bukrs      TYPE bukrs,
8         sales_org  TYPE farr_sales_org,
9         header_id  TYPE farr_header_id,
10        kunnr      TYPE kunnr,
11        partner    TYPE bu_partner,
12        kdgrp      TYPE kdgrp.
13        INCLUDE    TYPE farr_s_srcdoc          AS grp_srcdoc.
14        INCLUDE    TYPE farr_contract_item_incl_eew_ps AS enh_contr_item.
15    TYPES: END OF ty_s_order_item_badi.
16
17    TYPES: ty_ts_order_item_badi TYPE SORTED TABLE OF ty_s_order_item_badi
18           WITH UNIQUE KEY grp_srcdoc.
19
```

Figure 3.17 Parameters of Method DETERMINE_VERSIONS



Active	RecAccite...	Class Type	Name	Configuration Status
<input checked="" type="checkbox"/>	YA01	Order Item	Order Item Class	In Processing

Figure 3.18 Adding Customer Fields to the RAI Class

SAP Maintain Customer Fields

More ▾

Class: Order Item Class
 Class Type: Order Item

Main Items

Select	Field Name	Raw Items	Proc Items	Data element	Short Description
<input type="checkbox"/>	ZZPOB_CUR	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPOB_WAERS	POB Currency
<input type="checkbox"/>	ZZPOB_DESC	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZZDESC	POB Description
<input type="checkbox"/>	ZZPOB_STATUS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPOB_STATUS	POB Status

Figure 3.19 Choosing the Custom Fields

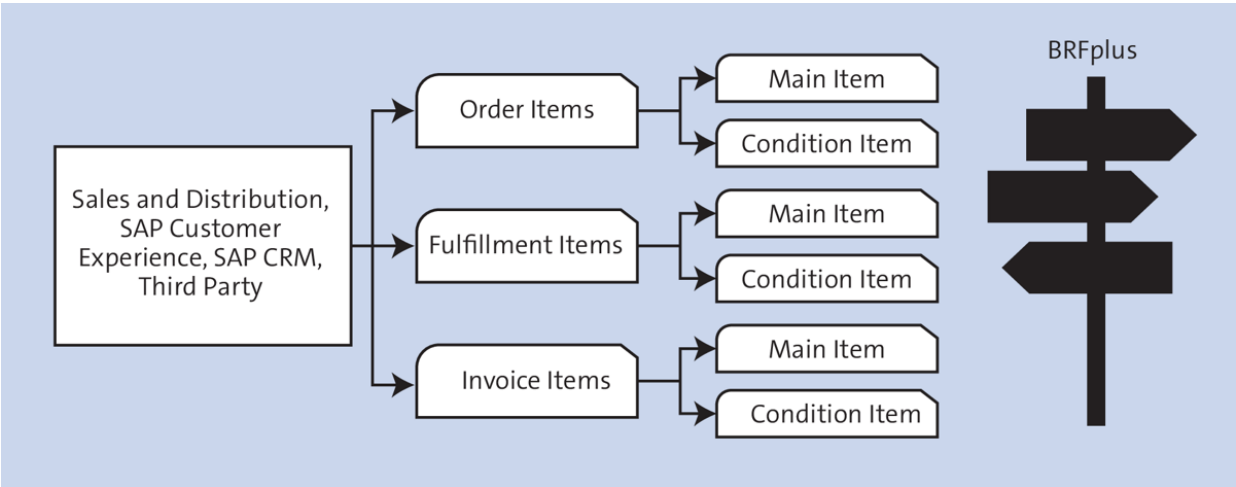


Figure 4.1 ARL Design

<input type="text"/> Refresh Statistics Simulate Transfer Transfer Process Initial Load Process Change Exempt Restore More																	
Main Item (100) Condition Item (209)																	
<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>																	
	POB(IFR...	Contract()	ItemStatus	Hist Exist	Error	Ex.His.Ex	Send.Com...	SourceSys.	SrcItemT...	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Type	Reference ID	Customer
<input type="checkbox"/>							SD		SDOI	0099000671000010	815	SD01	99000671	10	SDO	0099000671	100088450
<input type="checkbox"/>							SD		SDOI	0099000672000010	580	SD01	99000672	10	SDO	0099000672	100088450
<input type="checkbox"/>							SD		SDOI	04MxY4RIZksXmm	968	SD01	877936	10			877936

Figure 4.2 Main and Condition Items in Transaction FARR_RAI_MON



Further Selections

Personalize

More

Kind of Selection

Kind of Selection:

Item Status

Item Status:

Source Documents

Component:

to:



Logical System:

to:



Type:

to:



Header ID (Orders):

to:



Item ID (Orders):

to:



Date:

to:



Time:

to:



Master Data

Company Code:

to:



Business Partner Number:

to:



Customer:

to:



Revenue Accounting Item Classes

Class:

to:



Further Attributes

Reference Type:

to:



Reference ID:

to:



Error Status:

to:



Revenue Accounting Item ID:

to:



Further Selections Exist:

Figure 4.3 Transaction FARR_RAI_MON: Initial Screen

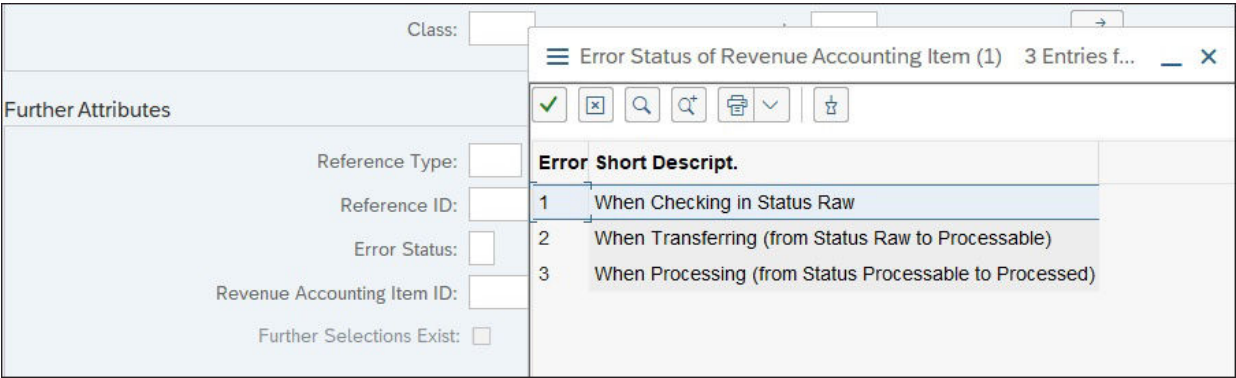


Figure 4.4 Error Status Display

Technical Criteria

No restriction of No. of Hits:

Maximum No. of Hits:

Layout Result Screen:

Figure 4.5 Technical Criteria for Display

ItemStatus	Hist Exist	Error	Ex His Ex	Send Com.	SourceSys	SrcItmType	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Type	Reference ID	Customer	Partner	CoCode	Err...	Creation Date	Create Tim
		■		SD		SDOI	0800760206000010	869	SD01	800760206	10	SDO	0800760206	1930000015	GB04			02.02.2023	12.45.11
		■		SD		SDOI	0800760207000010	728	SD01	800760207	10	SDO	0800760207	1930000015	GB04			02.02.2023	13.15.17
		■		SD		SDOI	0800760207000020	728	SD01	800760207	20	SDO	0800760207	1930000015	GB04			02.02.2023	13.15.17
		■		SD		SDOI	0800760207000030	728	SD01	800760207	30	SDO	0800760207	1930000015	GB04			02.02.2023	13.15.17
		■		SD		SDOI	0800760207000040	728	SD01	800760207	40	SDO	0800760207	1930000015	GB04			02.02.2023	13.15.17
		■		SD		SDOI	0800760207000050	728	SD01	800760207	50	SDO	0800760207	1930000015	GB04			02.02.2023	13.15.17
		■		SD		SDOI	0800760208000010	428	SD01	800760208	10	SDO	0800760208	1930000013	GB04			02.02.2023	13.45.12
		■		SD		SDOI	0800760209000010	693	SD01	800760209	10	SDO	0800760209	1930000013	GB04			02.02.2023	13.45.12
		●		SD		SDOI	X9900554000010	490	SD01	X9900554	10	SDO	X9900554	1930000013	GB04		3	23.12.2022	15.34.59
		●		SD		SDOI	X9900555000010	409	SD01	X9900555	10	SDO	X9900555	1930000013	GB04		3	23.12.2022	01.06.55
		■		SD		SDII	0801740001000010	439	SD03	801740001	10			1930000013	GB04			05.12.2022	10.09.53

Figure 4.6 Transaction FARR_RAI_MON: Results

Date/Time/User	Number	External ID	Object text	Subsubject Text
02.02.2023 01:33:59 JS4753	7	JS475320230202013...	Revenue Accounting	Processing of Rev
● Problem class Important	4			
▲ Problem class Medium	2			
■ Problem class Additional Information	1			

Ty...	Message Text	LTxt
■	Start of processing of order item SDOI/X9900554000010 for accounting principle IFRS	
●	Inflight check: Error C01/Contract 100870 found before save to database	
●	C01: Allocation error found for contract 100870	?
▲	Performance obligation type of performance obligation \$000000000000004 is missing	
▲	Name of leading performance obligation \$000000000000004 is missing	
●	Required field 'Fulfillment Type' of performance obligation \$000000000000004 is empty.	
●	Required field 'Event Type' of performance obligation \$000000000000004 is empty.	

Figure 4.7 Transaction FARR_RAI_MON: Error



Figure 4.8 Transaction FARR_RAI_MON: Options

Statistics of the Selected Items		
Statistical Information		
Status	Message Text	Counter
	RAI class SD01 Main Items Status 'PROCESSABLE'	10
	RAI class SD01 Condition Items Status 'PROCESSABLE'	20
	RAI class SD03 Main Items Status 'PROCESSABLE'	90
	RAI class SD03 Condition Items Status 'PROCESSABLE'	92
Σ	Total Main Items	100
Σ	Total Condition Items	112

Figure 4.9 Transaction FARR_RAI_MON: Statistics Display

Date/Time/User	Number	External ID	Object text	Subobject
02.02.2023 19:51:32 SM6918	5	SM69182023020219...	Revenue Accounting	Processing
Problem class Important	3			
Problem class Additional Information	2			

Ty...	Message Text
	***** PROCESS *****
	Start of processing of order item SDOI/800760206000010 for accounting principle IFRS
	New performance obligation 108126 created for contract 100873
	1 items processed (record type Main Item, class SD01)
	2 items processed (record type Condition Item, class SD01)

Figure 4.10 Transaction FARR_RAI_MON: Processing Results

Selection Data

Rev. Acc. Itm Class:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Sender Component:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Source Item Logical System:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Header ID:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Company Code:	<input type="text"/>	to:	<input type="text"/>	<input type="button" value="→"/>
Further Selections Exist:	<input type="checkbox"/>			

Technical Parameters

Number of Intervals:	<input type="text" value="2"/>
Block Size For Mass Selection:	<input type="text" value="1.000"/>
Simulation Mode:	<input checked="" type="checkbox"/>
Dialog Mode:	<input type="checkbox"/>
Synchronous Call:	<input type="checkbox"/>

Settings for Application Log

External ID:	<input type="text"/>
Problem class:	<input type="text" value="2 Important"/>

Figure 4.11 Transaction FARR_RAI_TRANS

Selection Data	
Rev. Acc. Itm Class: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Sender Component: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Source Item Logical System: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Header ID: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Reference Type: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Reference ID: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Company Code: <input type="text"/>	to: <input type="text"/> <input type="button" value="→"/>
Further Selections Exist: <input type="checkbox"/>	
Technical Parameters	
Number of Intervals: <input type="text" value="2"/>	
Block Size For Mass Selection: <input type="text" value="1.000"/>	
Dialog Mode: <input type="checkbox"/>	
Synchronous Call: <input type="checkbox"/>	
Settings for Application Log	
External ID: <input type="text"/>	
Problem Class: <input type="text" value="2 Important"/>	

Figure 4.12 Transaction FARR_RAI_PROC

Navigation icons: [Back] [Home] [Print] [Refresh] [Search] [Help] [Display Original] [Documentation] [Supplementary Do]

Data element: Active

Short Description:

Attributes | **Data Type** | Further Characteristics | Field Label

Elementary Type

- Domain
 - Digit Numerical Fields Without Check
 - Data Type: Numerical Text
 - Length:
- Built-in type
 - Data Type:
 - Length:
- Reference Type
- Referenced Type
 -

Figure 4.13 KEYPP Field Technical Description

Active

Short Description:

[Attributes](#)
[Tables](#)
[Lock parameter](#)

W	Lock parameter	Table	Field
<input checked="" type="checkbox"/>	MANDT	FARR_S_KEYPP_BUKRS_ENQ	MANDT
<input checked="" type="checkbox"/>	KEYPP	FARR_S_KEYPP_BUKRS_ENQ	KEYPP
<input checked="" type="checkbox"/>	BUKRS	FARR_S_KEYPP_BUKRS_ENQ	BUKRS
<input checked="" type="checkbox"/>	OBJPP	FARR_S_KEYPP_BUKRS_ENQ	OBJPP

Figure 4.14 KEYPP Lock Object

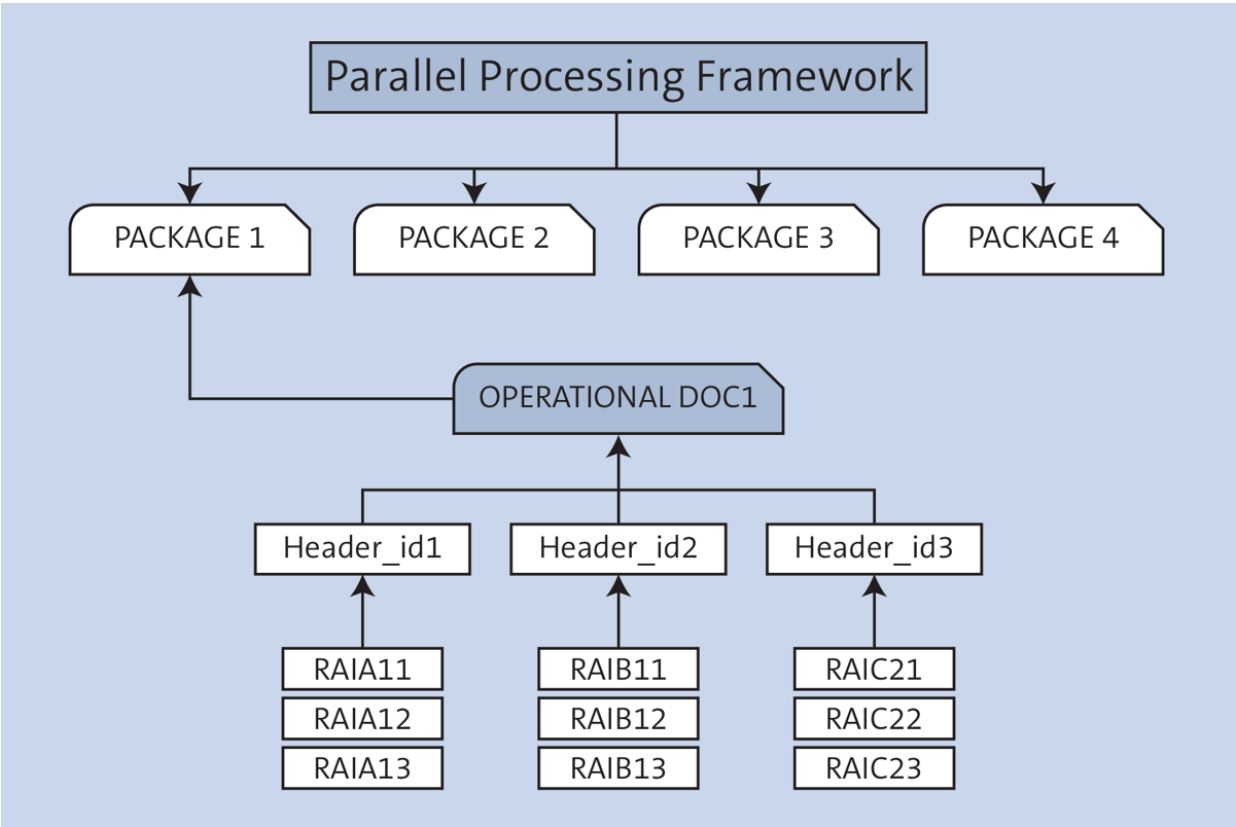


Figure 4.15 Parallel Processing Model

POB(IFR_ Contract)	ItemStatus	Hist	Exist	Error	Ex	His	Ex	Send Com.	SourceSys	SrcItem	Type	Source Item ID	Subarea	RevAcc.	Header ID	ItemID	Ref	Type	Reference ID	Customer	Partner	CoCode	Err.	Creation Date	Create Tm	Quantity	Re
									SD	SDOI		0090000802000010	421	SD01	90000802	10	SDO		0090000802	1000884500	GB04			12.02.2023	09.45.44	100	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	
									SD	SDPI		009000080200001000000018520000	421	SD03	90000802	10			0090000802	1000884500	GB04			12.02.2023	09.45.44	1	

Figure 4.16 Processing Order Items in Transaction FARR_RAI_MON

Date/Time/User	Number	External ID	Object text	Subobject
12.02.2023 09:09:09 SM6918	19	SM69182023021209...	Revenue Accounting	Processing
■ Problem class Important	5			
■ Problem class Additional Information	14			

Ty...	Message Text			
■	***** PROCESS *****			
■	Start of processing of order item SDOI/90000802000010 for accounting principle IFRS			
■	New performance obligation 103010 created for contract 200310			

Figure 4.17 Contract Created in Transaction FARR_RAI_MON

POB(FR_	Contract)	ItemStatus	Hist	Exist	Error	Ex	Hs	Ex	Send	Com.	SourceSys.	SrcItem	Type	Source	Item	ID	Subarea	RevAcc.	Header	ID	Item	Ref.	Type	Reference	ID	Customer	Partner	CoCode	Err.	Creation	Date	Create	Tim
103010	200310								SD		SDOI	0090000802000010					421	SD01	90000802	10	SDO	0090000802	1000884500		GB04			12.02.2023	10:18:43				
103010	200310								SD		SDOI	0090000802000010					421	SD01	90000802	10	SDO	0090000802	1000884500		GB04			12.02.2023	10:20:19				
103010	200310								SD		SDPI	009000080200001000000018520000					421	SD03	90000802	10				1000884500		GB04			12.02.2023	10:18:43			

Figure 4.18 Changes in FARR_RAI_MON

Main Item (26)		Condition Item (28)										
ItemStatus	Error	Send.Com...	SrcItemType	Source Item ID	Cond.type	P/L Account	Timestamp	TC Amount	Currency	Statu...	Condit...	Main Cond.
<input type="checkbox"/>		SD	SDOI	0090000802000010	ZPR0	5111000003	20.230.212.091.843	90.000,00	GBP		P	X
<input type="checkbox"/>		SD	SDOI	0090000802000010	ZSSP		20.230.212.091.843	90.000,00	GBP	X	P	
<input type="checkbox"/>		SD	SDOI	0090000802000010	ZPR0	5111000003	20.230.212.092.019	90.000,00	GBP		P	X
<input type="checkbox"/>		SD	SDOI	0090000802000010	ZSSP		20.230.212.092.019	96.000,00	GBP	X	P	

Figure 4.19 Transaction FARR_RAI_MON:
Conditions

		Condition Record	Analysis		Update					
Pricing Elements										
I...	CnTy	Name	Amount	Crcy	per	UoM	Condition Value	Curr.	Status	N
<input type="checkbox"/>	ZPRO	PG: List Price.	1,00	GBP		1 PC	1,00	GBP		
<input type="checkbox"/>	ZKUM	Cumulation cond-Stat	1,00	GBP			1,00	GBP		
		Gross Value	1,00	GBP		1 PC	1,00	GBP		
		Net Value of Item	1,00	GBP		1 PC	1,00	GBP		
<input type="checkbox"/>	ZSSP		1,00	GBP		1 PC	1,00	GBP		
		Net Value 2	1,00	GBP		1 PC	1,00	GBP		
<input type="checkbox"/>	ZTXD	Tax Service – Doc Lv	0,000	%			0,00	GBP		
<input type="checkbox"/>	ZTXE	TaxSvc call – doc lv	0,000	%			0,00	GBP		
<input type="checkbox"/>	ZWST	Output Tax	0,000	%			0,00	GBP		
		Total	1,00	GBP		1 PC	1,00	GBP		

Figure 4.20 Pricing Procedure

<input type="checkbox"/>	ItemStatus	Hist Exist	Error	Send.Com...	SourceSys.	SrcltmType	Source Item ID	Cond.type	P/L Account
<input checked="" type="checkbox"/>				SD		SDOI	0090000847000010	ZPR0	5111000003
<input checked="" type="checkbox"/>				SD		SDOI	0090000847000010	ZSSP	

Figure 4.21 Condition Items in Transaction
FARR_RAI_MON

Quantity: PC Net: GBP
Tax:

Pricing Elements

...	Only	Name	Amount	Crcy	per	UoM	Condition Value	Curr.	Status	NumCCo
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZPRG PG:Global List Price	15,00	GBP		1 PC	15,00	GBP		1
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZKUM Cumulation cond-Stat	12,00	GBP			7,20	GBP		0
		Gross Value	15,00	GBP		1 PC	15,00	GBP		1
		Net Value of Item	15,00	GBP		1 PC	15,00	GBP		1
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZSSP	12,00	GBP		1 PC	7,20	GBP		1
		Net Value 2	7,20	GBP		1 PC	7,20	GBP		1
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZTXD Tax Service – Doc Lv	0,000	%			0,00	GBP		0
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZTXE TaxSvc call – doc lv	0,000	%			0,00	GBP		0
<input type="checkbox"/>	<input checked="" type="checkbox"/>	ZWST Output Tax	0,000	%			0,00	GBP		0
		Total	15,00	GBP		1 PC	15,00	GBP		1
		Standard - USA /With	15,00	GBP		1 PC	15,00	GBP		1

Figure 4.22 Changed Condition Type during Invoicing

Execute Get variant... Program Documentation Further Selections Personalize More

Kind of Selection

Kind of Selection: All Items Related to Order Items

Item Status

Item Status: 03 Raw and Processable Items

Source Documents

Component:		to:		
Logical System:		to:		
Type:		to:		
Header ID (Orders):	41007138	to:		
Item ID (Orders):		to:		
Date:		to:		
Time:	00:00:00	to:	00:00:00	

Master Data

Figure 4.23 Transaction FARR_RAI_MON with Order Number

<input type="checkbox"/>	ItemStatus	Hist Exist	Error	Ex.His.Ex	Send.Com...	SourceS...	SrcItmTy...	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Ty...
<input type="checkbox"/>			■		SD		SDFI	0070000118001190 20221124145422	012	SD02	70000118	1190	
<input type="checkbox"/>			■		SD		SDFI	0070000118001200 20221124145422	012	SD02	70000118	1200	
<input type="checkbox"/>			■		SD		SDFI	0070000118001210 20221124145422	012	SD02	70000118	1210	
<input type="checkbox"/>			■		SD		SDFI	0070000118001220 20221124145422	012	SD02	70000118	1220	
<input type="checkbox"/>			■		SD		SDII	9110000084000002	962	SD03	9110000084	2	
<input type="checkbox"/>			■		SD		SDII	9110000084000003	962	SD03	9110000084	3	
<input type="checkbox"/>			■		SD		SDII	9110000084000004	962	SD03	9110000084	4	
<input type="checkbox"/>			■		SD		SDII	6110000011001190	012	SD03	6110000011	1190	
<input type="checkbox"/>			■		SD		SDII	6110000011001200	012	SD03	6110000011	1200	
<input type="checkbox"/>			■		SD		SDII	6110000011001210	012	SD03	6110000011	1210	
<input type="checkbox"/>			■		SD		SDII	6110000011001220	012	SD03	6110000011	1220	
<input type="checkbox"/>			■		SD		SDII	9110000086001190	012	SD03	9110000086	1190	
<input type="checkbox"/>			■		SD		SDII	9110000086001200	012	SD03	9110000086	1200	
<input type="checkbox"/>			■		SD		SDII	9110000086001210	012	SD03	9110000086	1210	
<input type="checkbox"/>			■		SD		SDII	9110000086001220	012	SD03	9110000086	1220	

Figure 4.24 Results for Items with Predecessors

■	Start of processing of invoice item SDII/9110000286000001 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDII/9110000286000001)
■	Start of processing of invoice item SDII/9110000285000001 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDII/9110000285000001)
■	Start of processing of invoice item SDII/6110000022000020 for accounting principle IFRS
■	Start of processing of invoice item SDII/9110000600000004 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDII/9110000600000004)
■	Start of processing of invoice item SDII/9110000600000003 for accounting principle IFRS
●	Performance obligation to be invoiced could not be determined (SDII/9110000600000003)

Figure 4.25 Error for Invoice Processing and POB Determination

■	***** PROCESS *****
■	Invoice RAI SDII/9110000240001020, there is unprocessed previous RAI to same origdoc SDOI/41007251001020.
■	Invoice RAI SDII/9110000282001010, there is unprocessed previous RAI to same origdoc SDOI/41007273001010.
■	Start of processing of invoice item SDII/9110000539000004 for accounting principle IFRS

Figure 4.26 Invoice Error Due to an Unprocessed Order RAI

Maintain Changeable Fields of Rev. Accounting Item Classes					
RevAccCl	Rec. Type	Status	Field Name	FieldAttr	
<input type="checkbox"/> SD01	All Record Types	▼ All Statuses	▼ QUANTITY		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼
	All Record Types	▼ All Statuses	▼		▼

Figure 4.27 Changeable Fields for RAIs

	POB(IFR...	Contract()	ItemStatus	Hist Exist	Error	Ex.His.Ex	Send.Com...	SourceSys	SrcItem Type	Source Item ID	Subarea	RevAcc...	Head
<input checked="" type="checkbox"/>			♂		●		SD		SDOI	0089003818000010	581	SD01	8900
<input type="checkbox"/>			♂		●		SD		SDOI	0089004352000010	304	SD01	8900

Figure 4.28 Change Option in Transaction
FARR_RAI_MON

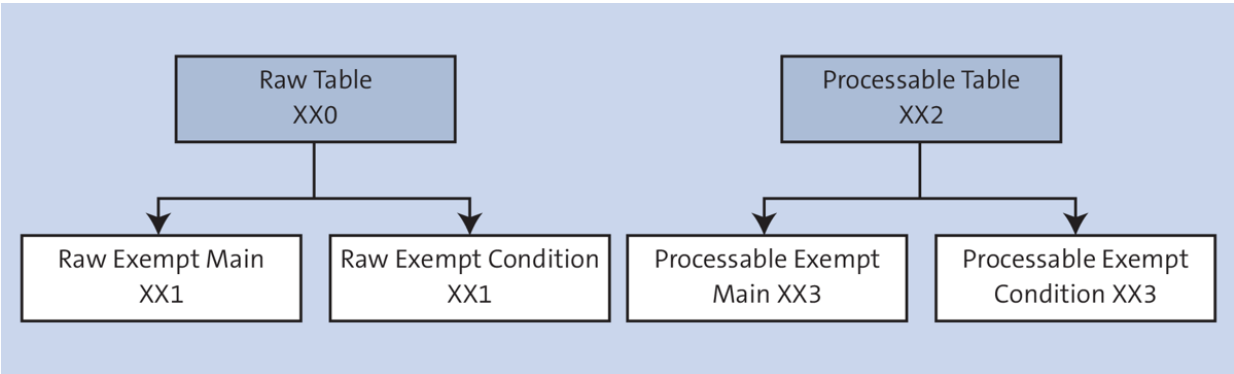


Figure 4.29 Exempted Items Table Structure

Exemption History			EXCHIST
Exemption Date (UTC)	15.02.2023	20230215	EXCDATE
Time of Exemption (UTC)	12:36:41	123641	EXCTIME
Exemption Reason	EX	EX	EXCREASON
User for Exemption	SM6918	SM6918	EXCUSNAM

Figure 4.30 Exemption Items Fields

< **SAP** Display View "Exemption Reasons for

Q ☰ ☰ ☰ ☰ More ∨

Exemption Reasons for Revenue Accounting Items

ExempRsn	Exemption Reason	StatusGrp	ExempType
<input type="checkbox"/> EX	Exempt	All	∨ It is possible to restore the item

Figure 4.31 Defining the Exemption Reason

		Refresh	Statistics	Process	Change	Exempt	Restore	More																														
Main Item (80)		Condition Item (119)																																				
POB(IFR... Contract)	ItemStatus	Hist	Exist	Error	Ex.His	Ex.	Send	Com.	SourceSys	SrcItem	Type	Source	Item	ID	Subarea	RevAcc...	Header	ID	Item	Ref.	Type	Reference	ID	Customer	Partner	CoCo...	Err...	Creation	Date	Create	Tim	Quantity	Reason					
							SD		SDOI			0089004364000010	341	SD01	89004364	10	SDO	0089004364	20001003	CH04	09.02.2023	14:34:59	1															
							SD		SDOI			0089004364000010	341	SD01	89004364	10	SDO	0089004364	20001003	CH04	09.02.2023	14:40:18	1															
							SD		SDOI			0089004364000010	341	SD01	89004364	10	SDO	0089004364	20001003	CH04	09.02.2023	14:41:21	1															
							SD		SDOI			0089004364000010	341	SD01	89004364	10	SDO	0089004364	20001003	CH04	09.02.2023	14:45:04	1															
							SD		SDOI			0089004366000010	160	SD01	89004366	10	SDO	0089004366	20064918	CH04	09.02.2023	15:09:31	1															

Figure 4.32 Exemption in Transaction
FARR_RAI_MON

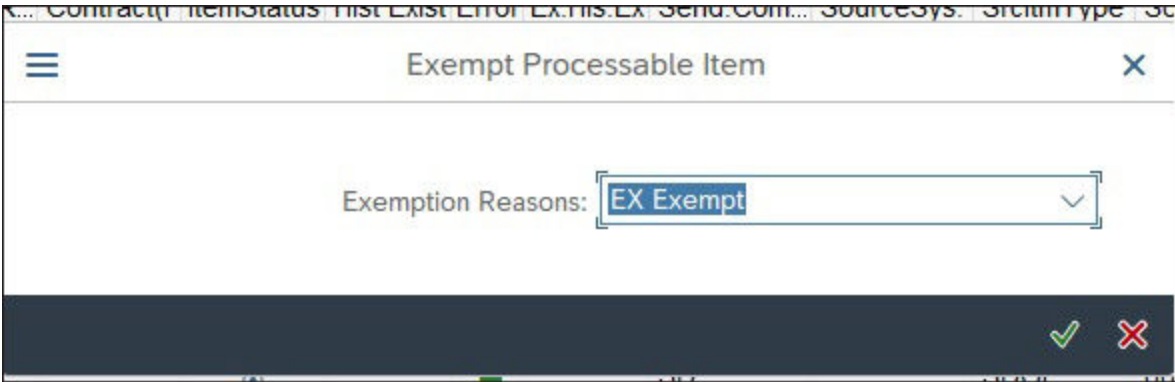


Figure 4.33 Selecting the Exempt Reason

Kind of Selection

Kind of Selection: 1 Order Items

Item Status

Item Status: 02 Processable Items

Source Documents

Component: 03 Raw and Processable Items

Logical System: 04 Processed Items

Type: 05 All Items

Header ID (Orders): 06 Manual Selection

Item ID (Orders): 07 Exempted Items

The image shows a software interface with three main sections: 'Kind of Selection', 'Item Status', and 'Source Documents'. The 'Kind of Selection' section has a dropdown menu set to '1 Order Items'. The 'Item Status' section has a dropdown menu set to '02 Processable Items'. The 'Source Documents' section contains several fields with dropdown menus: 'Component' is set to '03 Raw and Processable Items', 'Logical System' is set to '04 Processed Items', 'Type' is set to '05 All Items', 'Header ID (Orders)' is set to '06 Manual Selection', and 'Item ID (Orders)' is set to '07 Exempted Items'. The 'Item ID (Orders)' dropdown is currently open, showing a list of options: '01 Raw Items', '02 Processable Items', '03 Raw and Processable Items', '04 Processed Items', '05 All Items', '06 Manual Selection', and '07 Exempted Items'. The '07 Exempted Items' option is highlighted in blue.

Figure 4.34 Exempted Items in Transaction FARR_RAI_MON

<input type="text"/> Refresh Statistics Change Exempt Restore More																	
Main Item (8) Condition Item (12)																	
<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>																	
<input type="checkbox"/>	POB(IFRS)	Contract(I	ItemStatus	Hist Exist	Error	Ex.His.Ex	Send Comp.	SourceSys.	SrcitmType	Source Item ID	Subarea	RevAccCl	Header ID	ItemID	ExempDate	Reason	ExempUse
<input type="checkbox"/>			<input type="checkbox"/>		■		SD		SDOI	0089004364000010	341	SD01	89004364	10	18.02.2023	EX	SM6918
<input type="checkbox"/>			<input type="checkbox"/>		■		SD		SDOI	0089004364000010	341	SD01	89004364	10	18.02.2023	EX	SM6918
<input type="checkbox"/>			<input type="checkbox"/>		■		SD		SDOI	0089004364000010	341	SD01	89004364	10	18.02.2023	EX	SM6918
<input type="checkbox"/>			<input type="checkbox"/>		■		SD		SDOI	0089004364000010	341	SD01	89004364	10	18.02.2023	EX	SM6918
<input type="checkbox"/>	104007	200407	<input type="checkbox"/>		■		SD		SDOI	0090000820000010	009	SD01	90000820	10	15.02.2023	EX	SM6918
<input type="checkbox"/>	104007	200407	<input type="checkbox"/>		■		SD		SDOI	0090000820000010	009	SD01	90000820	10	15.02.2023	EX	SM6918
<input type="checkbox"/>	104007	200407	<input type="checkbox"/>		■		SD		SDOI	0090000820000010	009	SD01	90000820	10	15.02.2023	EX	SM6918
<input type="checkbox"/>	104012	200412	<input type="checkbox"/>		■		SD		SDOI	0090000825000010	563	SD01	90000825	10	15.02.2023	EX	SM6918

Figure 4.35 Exempted Items List in Transaction FARR_RAI_MON



Figure 4.36 Further Selections in Transaction FARR_RAI_MON

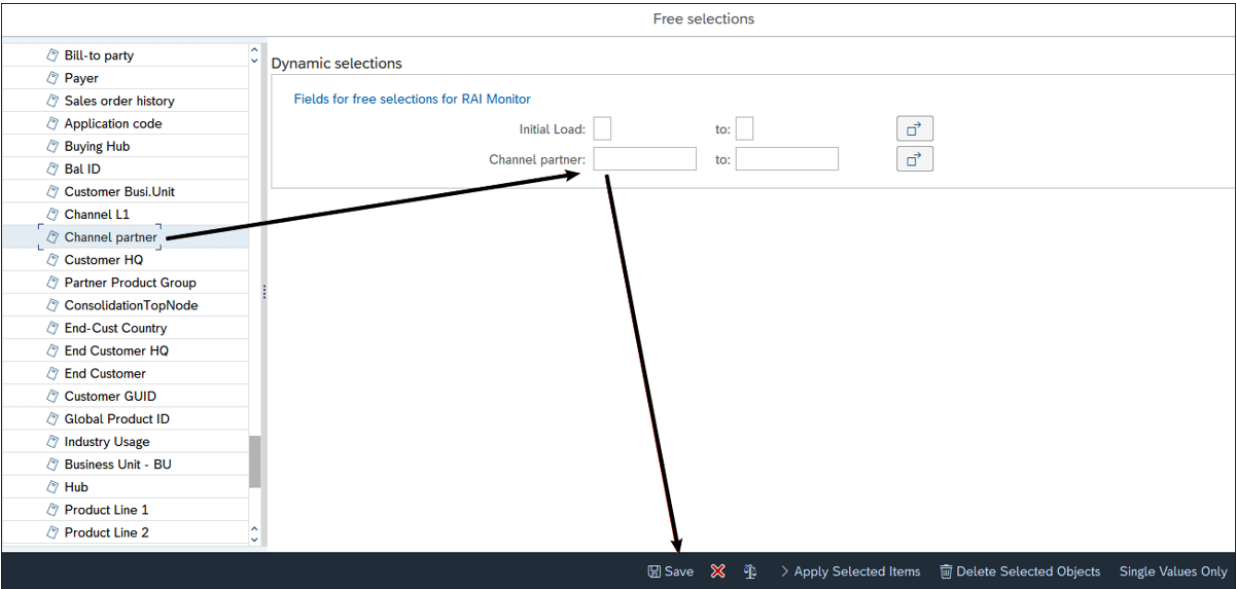


Figure 4.37 Data Selection for Further Options

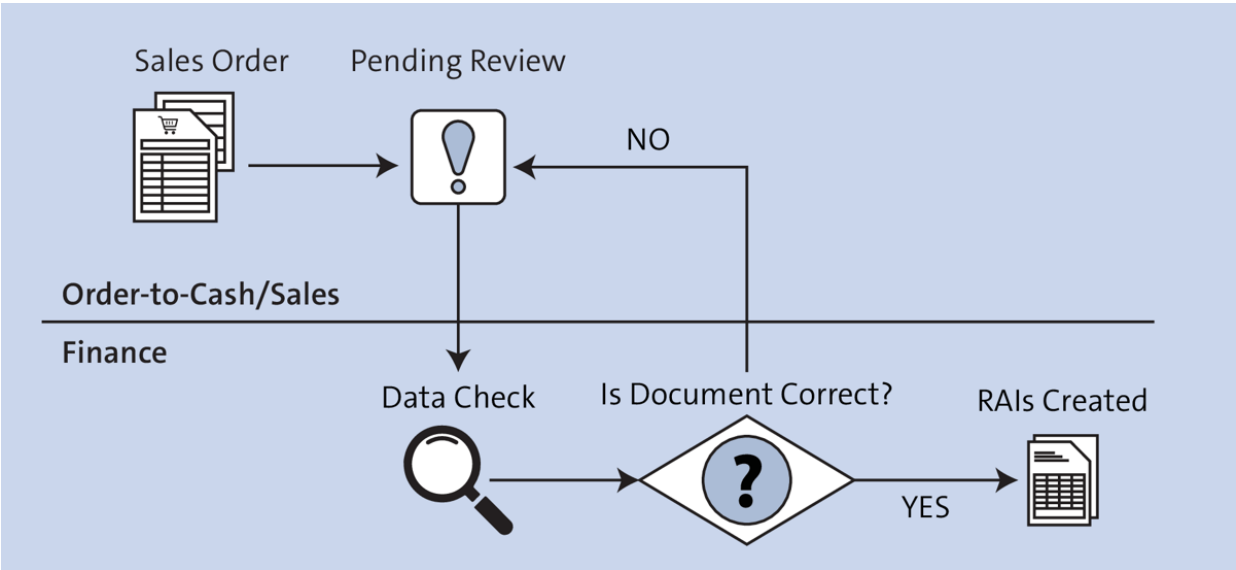


Figure 4.38 Statuses While Processing RAIs

Main Item (49)		Condition Item (68)																				
ItemStatus	Hist	Exist	Error	Ex	His	Ex	Send Com.	SourceSys	SrcItmType	Source Item ID	Subarea	RevAcc.	Header ID	ItemID	Ref. Type	Reference ID	Customer	Partner	CoCo.	Err.	Creation Date	Create Tim
			●				SD	SDOI		0800760259000010	526	SD01	800760259	10	SDO	0800760259	1200000344	NL03	3	16.02.2023	22:21:30	
			●				SD	SDOI		0800760276000010	436	SD01	800760276	10	SDO	0800760276	1000055298	ID02	3	22.02.2023	10:14:57	
			●				SD	SDOI		0800760276000020	436	SD01	800760276	20	SDO	0800760276	1000055298	ID02	3	22.02.2023	07:27:54	
			■				SD	SDOI		0800760293000010	617	SD01	800760293	10	SDO	0800760293	1930000001	AU02			25.02.2023	10:11:36
			●				SD	SDOI		0800760296000010	930	SD01	800760296	10	SDO	0800760296	1200000383	NL03	3	22.02.2023	09:31:11	
			●				SD	SDOI		0800760296000020	930	SD01	800760296	20	SDO	0800760296	1200000383	NL03	3	22.02.2023	09:31:11	
			●				SD	SDOI		0800760297000010	271	SD01	800760297	10	SDO	0800760297	1200000383	NL03	3	22.02.2023	09:49:06	
			●				SD	SDOI		0800760297000020	271	SD01	800760297	20	SDO	0800760297	1200000383	NL03	3	22.02.2023	09:49:06	

Figure 4.39 Errors in Transaction FARR_RAI_MON

Ty...	Message Text	LTxt
■	Start of processing of order item SDOI/800760259000010 for accounting principle IFRS	
●	Inflight check: Error C01/Contract 102008 found before save to database	
●	C01: Allocation error found for contract 102008	?
●	Start date of time-based performance obligation \$0000000000000001 is empty	?
●	Either a duration or an end date must be specified for perf. obligat. \$0000000000000001	

Figure 4.40 Details of Error in Transaction FARR_RAI_MON

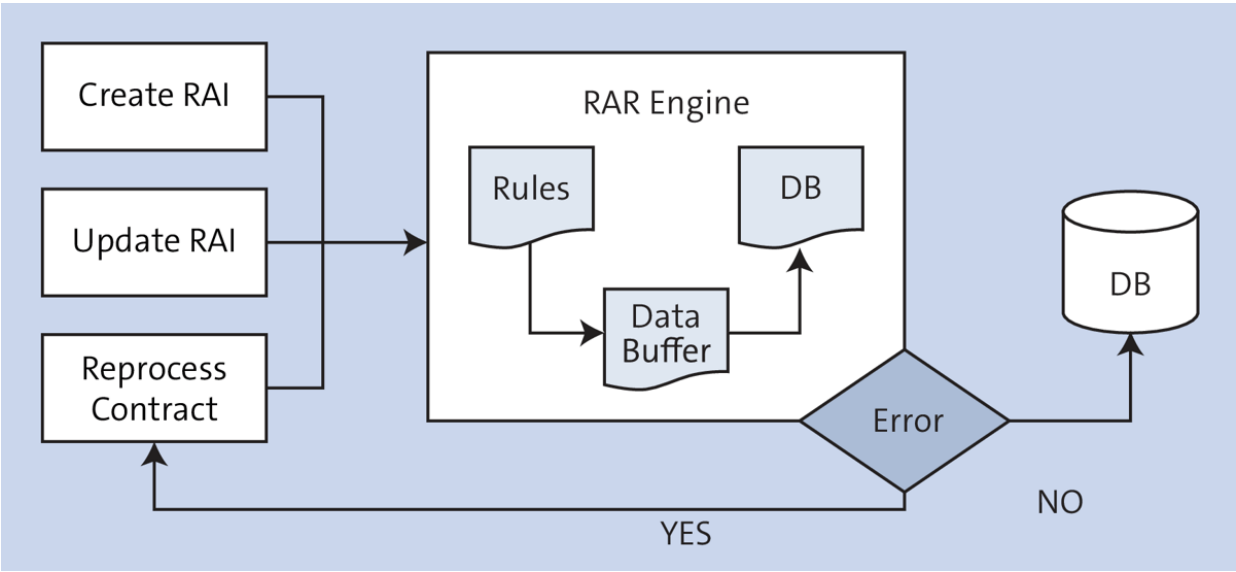





Figure 4.41 Inflight Error Logic

Selection Data

Accounting Principle:  to: 

Company Code: to: 

Excluding Completed Contracts:

Technical Parameters

Block Size For Mass Selection:

Dialog Mode:

Synchronous Call:

Settings for Application Log

External ID:


Problem class: 

Figure 4.42 Transaction FARR_CONTR_CHECK

Contract			
Accounting Principle:	<input type="text"/>	to: <input type="text"/>	<input type="button" value="→"/>
Company Code:	<input type="text"/>	to: <input type="text"/>	<input type="button" value="→"/>
Revenue Accounting Contract:	<input type="text"/>	to: <input type="text"/>	<input type="button" value="→"/>
Processing			
Read data online:	<input checked="" type="radio"/>		
Read data from error table:	<input type="radio"/>		
Save result to error table:	<input type="checkbox"/>		
Technical Parameter			
Max. batch size of Contracts:	<input type="text" value="1.000"/>		
Max. no. of POBs:			

Figure 4.43 Transaction FARR_CONTR_MON

Exception	Contra...	POB	CoCode	Acc. Princ.	Post Liab	Post Level	Post Liab Leadg/Lnk	LC Calcul	Leading	Higher-Lvl	Excl Alloc	BOM Root	Comp	FulfilType	Event Type	Val R
o	200064	101176	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
o	200064	101177	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200064															
o	200066	101179	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200066															
o	200069	101184	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200069															
o	200074	101193	CH04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level					X		D	E	MA	X
o	200074	101194	CH04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level					X		D	E	MA	X
	200074															
o	200075	101195	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200075															
o	200076	101196	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200076															
o	200077	101197	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
o	200077	101198	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200077															
o	200078	101199	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	O	MA	
	200078															
o	200079	101200	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	O	MA	X
	200079															
o	200080	101201	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
o	200080	101202	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	CI	X
	200080															
o	200081	101203	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	MA	X
	200081															
o	200082	101204	GB04	IFRS	Contract Liability/Contract Asset	Post at Performance Obligation Level							D	E	MA	X
	200082															

Figure 4.44 Error Monitor Results

Exception Contra.	POB	SumBookLia	SumPlaAsse	SumBooAsse	ManSpread	Spread Cf	Manual Amt	Manual Con	Manual Spr	Manu Alloc	Manual Any	Created on	Created At	E01	E02	E03	E04	E05	E06	E07	E08	E09	E10	E11	E12	E13	E14	E15	E16	E17	E18	E19	E22	E25					
☐	200064	101176	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
☐	200064	101177	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
☐	200066	101179	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
☐	200069	101184	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
☐	200074	101193	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
☐	200074	101194	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
☐	200075	101195	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
☐	200076	101196	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
●	200077	101197	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
●	200077	101198	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
☐	200078	101199	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
☐	200078	101199	0,00	0,00	0,00							25.02.2023	17.55.21	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 4.45 Errors Found While Processing Consistency Check

Table: /1RA/0SD012MI Items for Class SD01 - Processable

Text table: No texts

Layout: Maintain entries

Maximum no. of hits: 500

Get Field:

Selection Criteria

Fld name	O...	Fr.Value	To value	More	Output	Technic
HiLItemTyp	⬇			⬇	☑	HILDOC
HiLItemID	⬇			⬇	☑	HILDOC
Error	⬇					
Delivery-Relevant	⬇					
BillPlanTyp	⬇					
Sales Org.	⬇					
Delete	⬇					
Initial Load	⬇					
Handle	⬇					
ReceivAccount	⬇					
Inception Date	⬇					

☰ Error Status of Revenue Accounting Item (1) 3 Entries f

☑ ☒ 🔍 🔍 📄 ▼

Error Short Descript.

1	When Checking in Status Raw
2	When Transferring (from Sta Short Descript. processable)
3	When Processing (from Status Processable to Processed)

Figure 4.46 Errors in Processable Table

Object:

Subobject: (generic input)

External ID:

Time Restriction

From (Date/Time):

To (Date/Time):

Log Triggered By

User:

Transaction code:

Program:

Log Class

Only very important logs

Only important logs

Also less important logs

All logs

Subject (1) 31 Entries found

Restrictions

Subject	Subject Text
ACCRUAL	Posting Run in Revenue Accounting
BIZ_RECON	
CLEANUP	Cleanup of Revenue Accounting Items and Dependent Objects
CONTR_CHECK	Calculation of Contract Liabilities and Assets
CONTR_LIAB	Calculation of Contract Liabilities and Assets
CONTR_MGMT	Processing of Revenue Accounting Contracts
CONTR_ROUNDING	
DATA_MIGRATION	Data Migration: Upgrade of Revenue Accounting
MC	
PERIOD_IN_CLOSE	
POB_CANCELLATION	End Performance Obligation
PROD_CLEANUP	Cleanup of Revenue Accounting Items and Dependent Objects

Figure 4.47 Transaction SLG1

>	16.02.2023 07:49:28 SM6918	5	SM69182023021607...	Revenue Accounting	Creation of Revenue ...	
∨	16.02.2023 08:11:31 KS1720	6	KS17202023021608...	Revenue Accounting	Creation of Revenue ...	
	● Problem class Important	6				
∨	16.02.2023 08:11:32 KS1720	17	KS17202023021608...	Revenue Accounting	Creation of Revenue ...	
	● Problem class Important	17				
>	16.02.2023 08:34:05 SM6918	5	SM69182023021608...	Revenue Accounting	Processing of Revenu...	FARR_RAI_M...
>	16.02.2023 08:34:05 SM6918	1	SM69182023021608...	Revenue Accounting	Processing of Revenu...	FARR_RAI_M...

Ty...	Message Text					LTxt
	***** CREATION *****					
	**** Item creation started for class SD03					
	● For item with ID SDPI/900008350000100000001893000001, the original item could not be found					?
	● For item with ID SDPI/900008350000100000001893000002, the original item could not be found					?
	● For item with ID SDPI/900008350000100000001893000003, the original item could not be found					?
	● For item with ID SDPI/900008350000100000001893000004, the original item could not be found					?
	● For item with ID SDPI/900008350000100000001893000005, the original item could not be found					?
	● For item with ID SDPI/900008350000100000001893000006, the original item could not be found					?
	● For item with ID SDPI/900008350000100000001893000007, the original item could not be found					?

Figure 4.48 List of Errors in Transaction SLG1

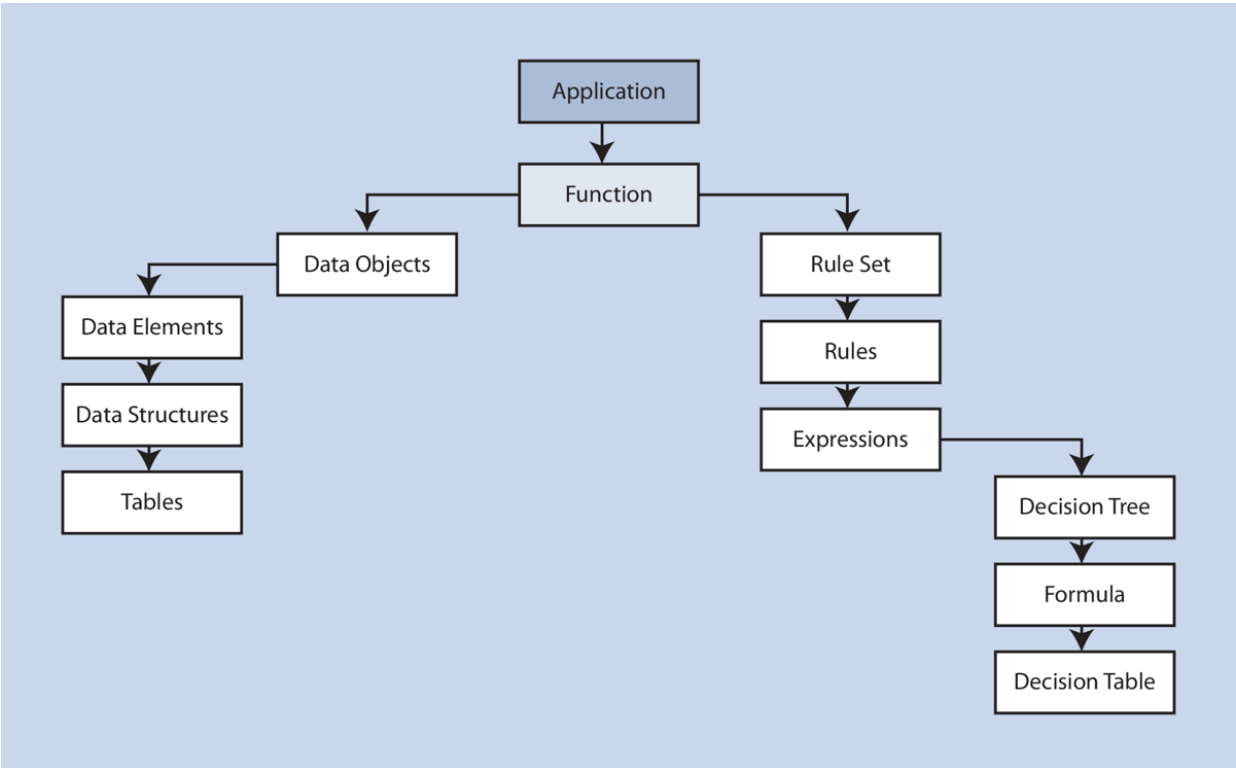


Figure 4.49 BRFplus Architecture

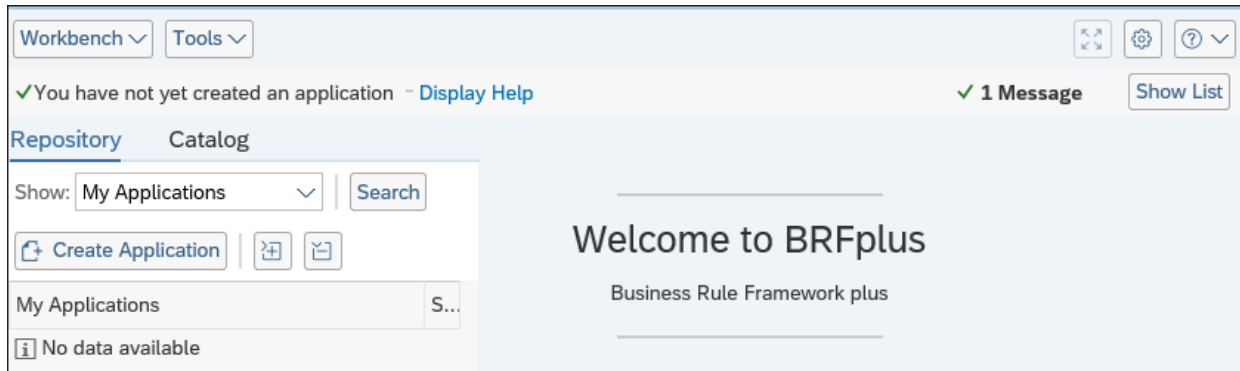


Figure 4.50 Transaction BRFPLUS

Search

Search Criteria

Application ... is equal to FARR*

Object Type is equal to Application

Name is equal to *

Also include objects from default BRFplus application:

Maximum Number of Results: 200

Search Clear Reset

Ok Cancel

Figure 4.51 BRFplus Application Selection

BRF+ Rule Configuration	
BRF+ Related Process	BRF Application
<input type="checkbox"/> AD Account Determination	▼ FARR_ACC_DETERMINE_PG1
<input type="checkbox"/> PS Performance Obligation Status	▼ FARR_POB_STATUS_PG1

Figure 4.52 Posting Application Assignment

Dialog Structure		ID for Decision Table UI	
ID for Decision Table UI		Decision Table ID Used for Customizing	BRF Application
Related Decision Table ID			Decision Table Name
<input type="checkbox"/>	PROCESS_BOM		FARR_AP_SD_PROCESS_PG1
<input type="checkbox"/>	PROCESS_COMPOUND		FARR_AP_SD_PROCESS_PG1
<input type="checkbox"/>	PROCESS_DEFERRAL		FARR_AP_SD_PROCESS_PG1
<input type="checkbox"/>	PROCESS_HEADER		FARR_AP_SD_PROCESS_PG1
<input type="checkbox"/>	PROCESS_POB		FARR_AP_SD_PROCESS_PG1
<input type="checkbox"/>	PROCESS_POB_ADD		FARR_AP_SD_PROCESS_PG1
<input type="checkbox"/>	PROCESS_SSP		FARR_AP_SD_PROCESS_PG1

Figure 4.53 Assignment of BRFplus Applications to the Simplified GUI

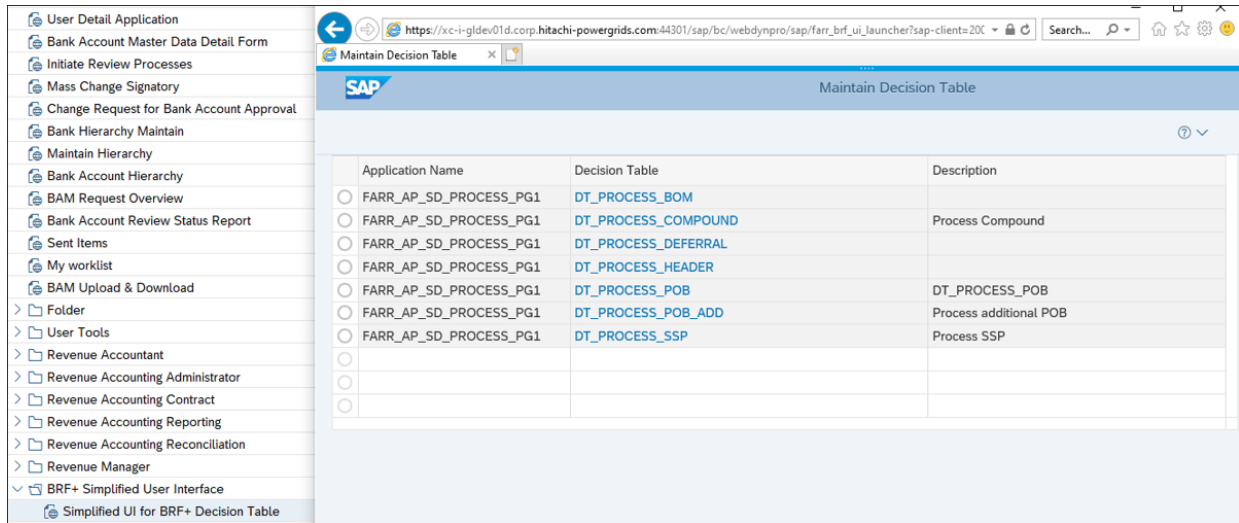


Figure 4.54 Simplified GUI

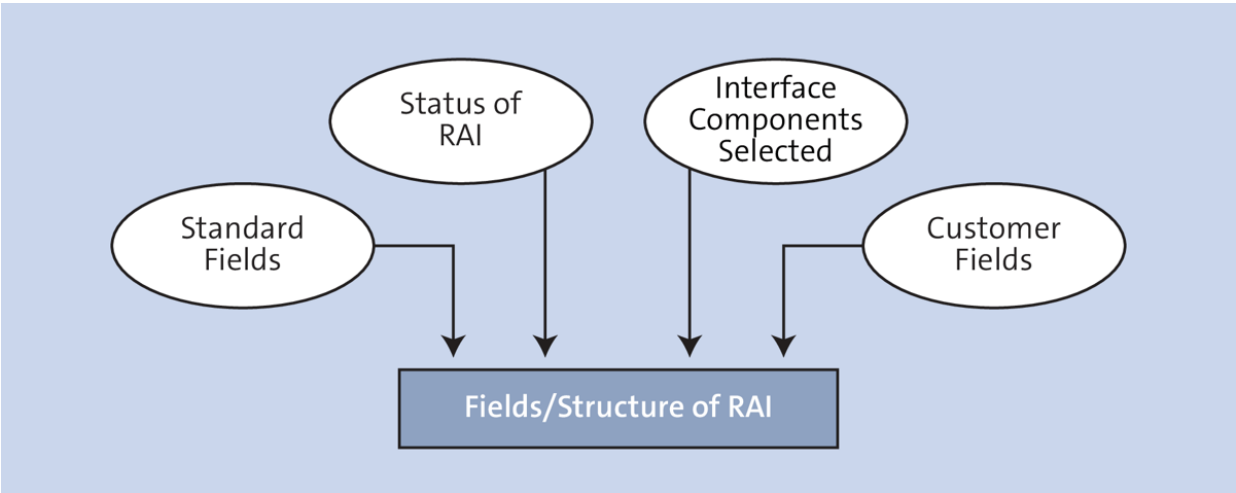


Figure 4.55 RAI Structure

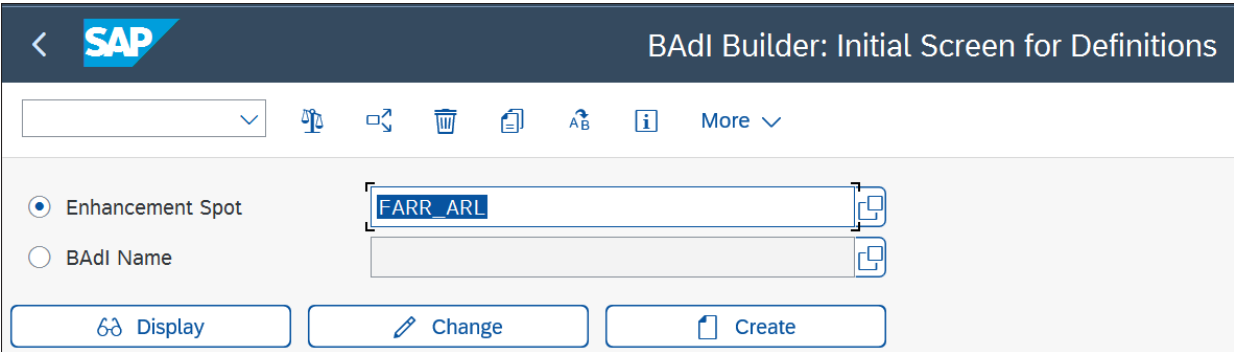


Figure 4.56 Enhancement Spot FARR_ARL

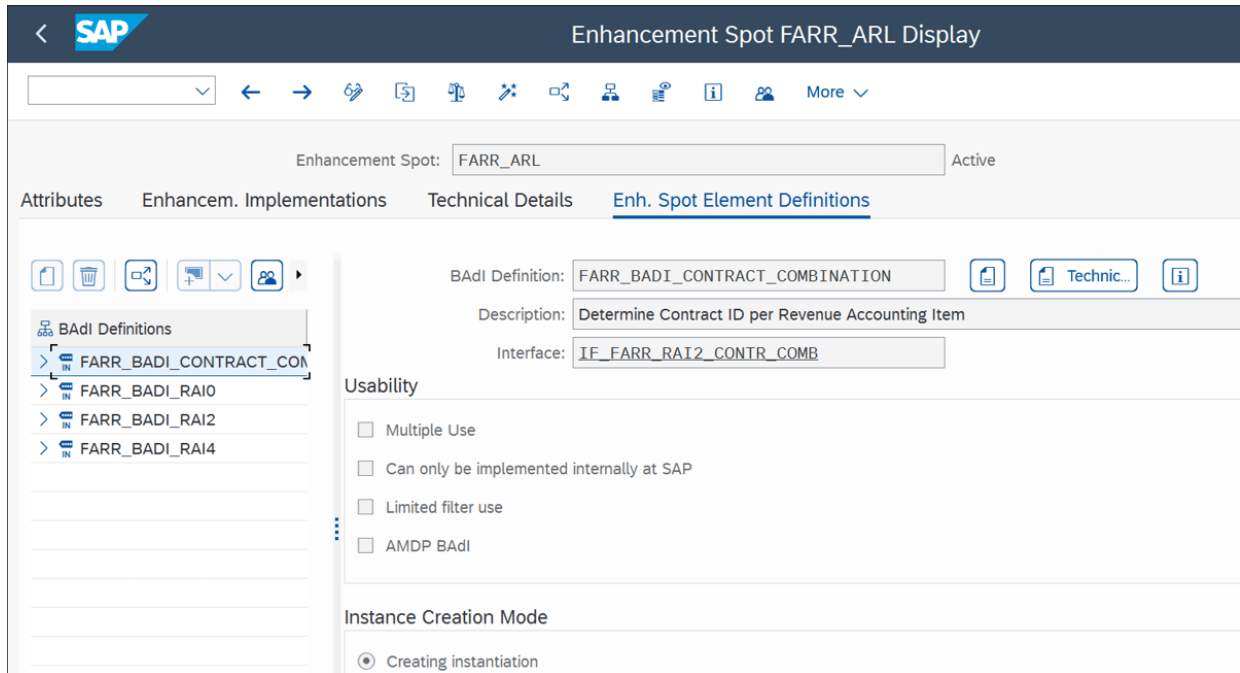


Figure 4.57 BAdI Definition of FARR_ARL

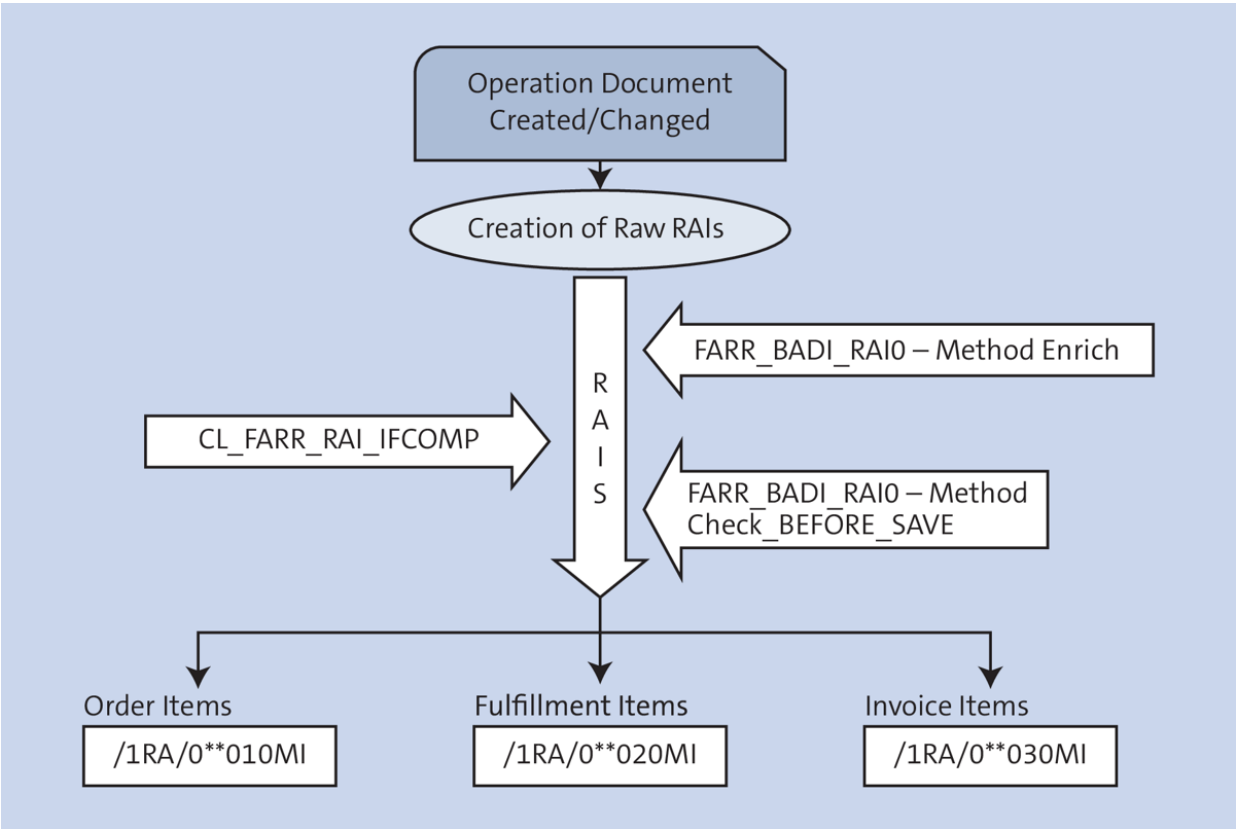


Figure 4.58 FARR_BADI_RAIO Triggered during RAI Creation

< **SAP** Class Builder: Display Interface IF_FARR_BADI_RAIO

Interface: IF_FARR_BADI_RAIO Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters Exceptions

Method	Level	Me...	Description
ENRICH	Instance	Method	Enrich Raw Revenue Accounting Item
CHECK_BEFORE_SAVE	Instance	Method	Check Raw Revenue Accounting Item Before Saving

Figure 4.59 Methods in Interface IF_FARR_BADI_RAIO

Interface: IF_FARR_BADI_RAI0 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: ENRICH

← Methods Exceptions Properties + - ✂ 📄

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC
CT_RAI0_MI	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI0_MI_ALL
CT_RAI0_CO	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI0_CO_ALL
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI_MSG

Figure 4.60 FARR_BADI_RAI0: ENRICH Method

Interface: IF_FARR_BADI_RAI0 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: CHECK_BEFORE_SAVE

← Methods Exceptions Properties + -

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC
IT_RAI0_MI	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI0_MI_ALL
IT_RAI0_CO	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI0_CO_ALL
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI_MSG

Figure 4.61 BAdI FARR_BADI_RAI0:
CHECK_BEFORE_SAVE Method

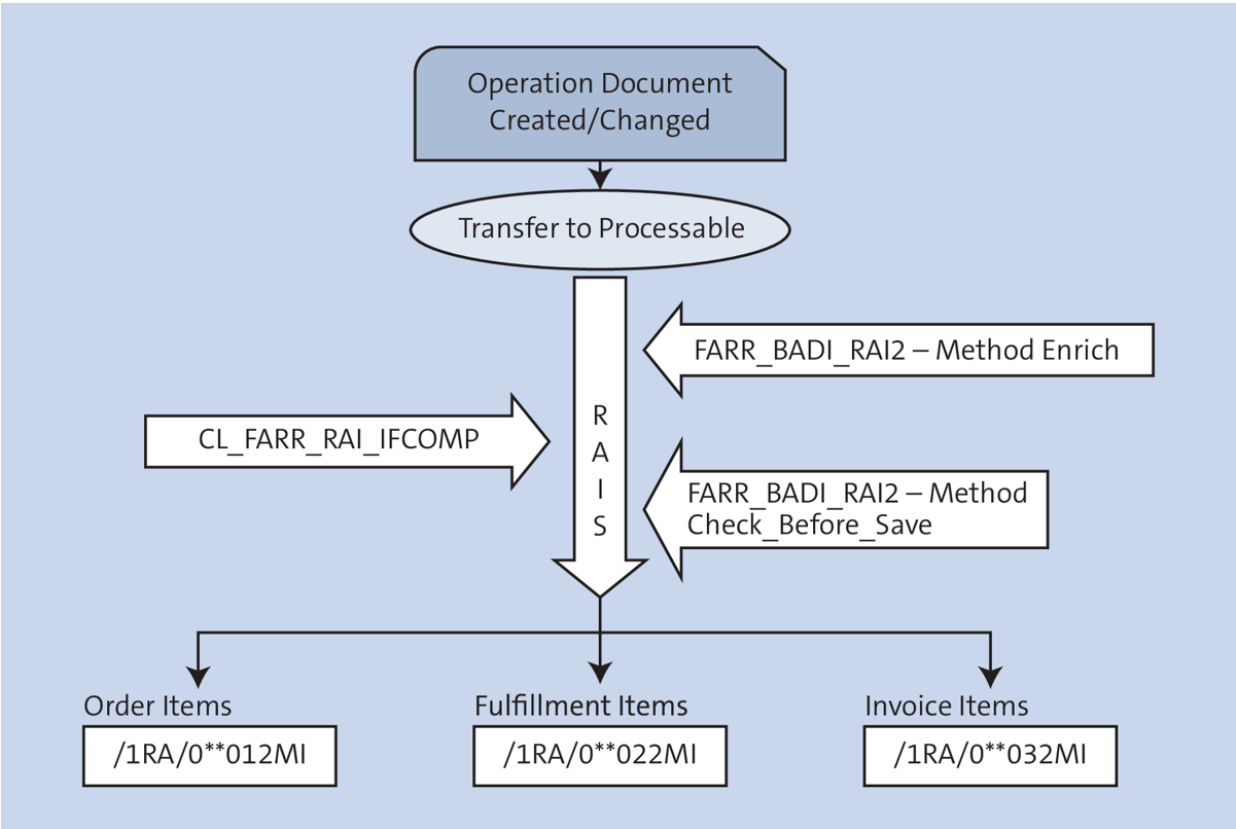


Figure 4.62 BAdI FARR_BADI_RAI2: Triggered during RAI Transfer

Interface: IF_FARR_BADI_RAI2 Implemented / Active

Properties Interfaces Attributes Methods Events Types Aliases

Parameters of Method: ENRICH

← Methods ⚡ Exceptions ⚙ Properties + -

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC
CT_RAI2_MI	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL
CT_RAI2_CO	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR TT RAI MSG

Figure 4.63 BAdi FARR_BADI_RAI2: ENRICH Method

Interface: IF_FARR_BADI_RAI2 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: CHECK_BEFORE_SAVE

← Methods ⚡ Exceptions Properties + -

Parameter	Type	Pa...	Op...	Typing Method	Associated Type
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC
IT_RAI2_MI	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL
IT_RAI2_CO	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI_MSG

Figure 4.64 BAdI FARR_BADI_RAI2:
CHECK_BEFORE_SAVE Method

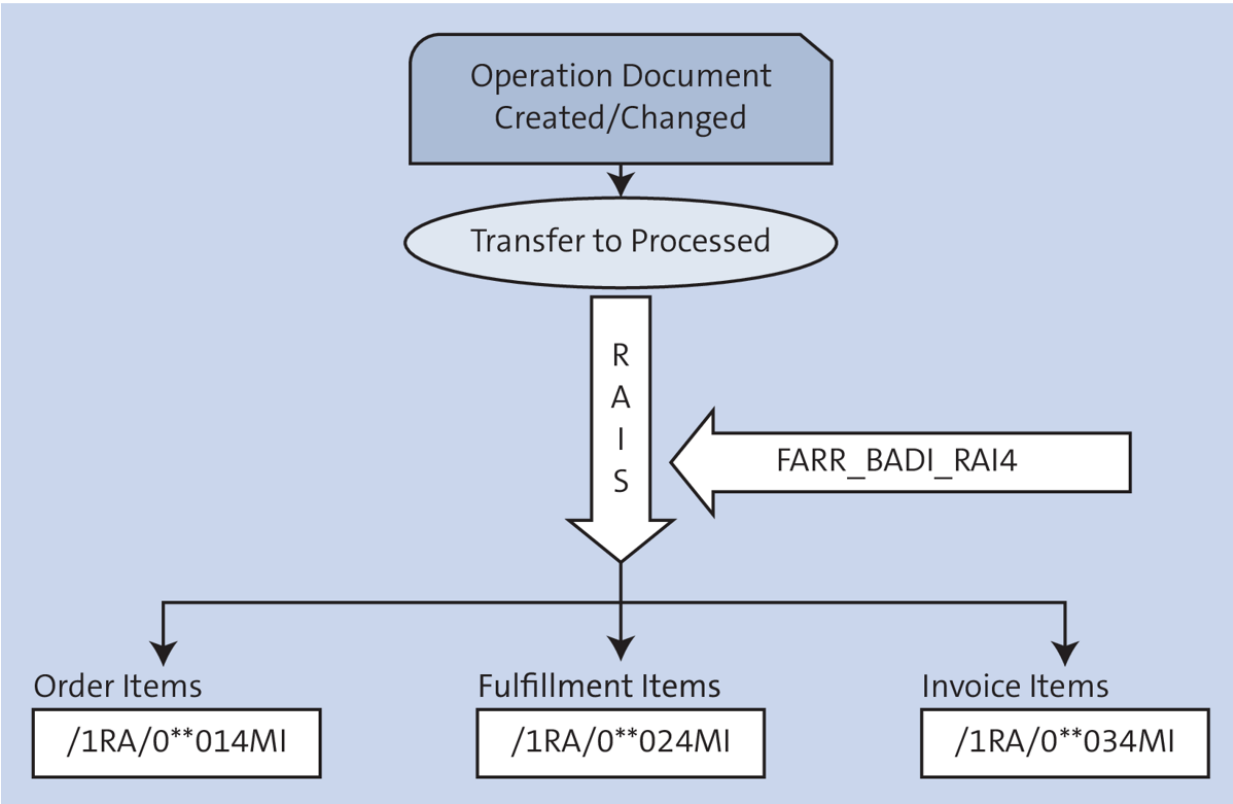


Figure 4.65 BAdI FARR_BADI_RA14: Triggered during the RAI Process

SAP Class Builder: Display Interface IF_FARR_BADI_RAI4

Interface: IF_FARR_BADI_RAI4 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters Exceptions

Method	Level	Me...	Description
EXCLUDE_COMPANY_CODES	Instance Method		Exclude RAIs of company codes from processing
EXCLUDE_RAIS_AT_PROC_START	Instance Method		Exclude RAIs from being processed
MODIFY_PREDOC_DATA	Instance Method		Modify aggregated data of a predoc chain
HANDLE_ASSUMED_INVOICE	Instance Method		Decide if the assumed invoice should be processed
HANDLE_FINAL_DATE_CHANGE	Instance Method		Avoid automatic creation of cancellation of assumed invoice

Figure 4.66 Interface IF_FARR_BADI_RAI4 Methods

SAP Class Builder: Display Interface IF_FAR

Interface: IF_FARR_BADI_RAI4 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: EXCLUDE_COMPANY_CODES

← Methods Exceptions Properties

Parameter	Type	Pa...	Op...	Ty...	Associated Type
IV_PROCESSING_MODE	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_PROCESSING_MODE
ET_EXCLUDED_COMPANY_CODE	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_COMPANY_CODE_MSG

Figure 4.67 EXCLUDE_COMPANY_CODES Method

Interface: IF_FARR_BADI_RAI4 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: EXCLUDE_RAIS_AT_PROC_START

← Methods ⚡ Exceptions Properties (+)

Parameter	Type	Pa...	Op...	Typing...	Associated Type
IV_PROCESSING_MODE	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_PROCESSING_MODE
IT_RAI_MI	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL
IT_RAI_CO	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL
ET_EXCLUDED_RAI	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI_MSG
EV_EXCLD_MODE	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	CHAR1

Figure 4.68 EXCLUDE_RAIS_AT_PROC_START Method

Interface Edit Goto Utilities Environment System Help

< SAP Class Builder: Display Interface IF_FARR_BADI_RAI4

Interface: IF_FARR_BADI_RAI4 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: MODIFY_PREDOC_DATA

← Methods Exceptions Properties

Parameter	Type	Pa...	Op...	Typing Method	Associated Type	Default Value
IT_RAI_MI_PREDOC_CHAIN	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI_MI_ALL	
IT_RAI_CO_PREDOC_CHAIN	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAI_CO_ALL	
IS_RAW_POB_AGGREGATED	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_S_RAW_POB	
IT_RAW_POB_CO_AGGREGATED	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAW_POB_CO	
ES_RAW_POB_AGGREGATED	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_S_RAW_POB	

Figure 4.69 MODIFY_PREDOC_DATA Method

Interface: IF_FARR_BADI_RAI4 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: HANDLE_ASSUMED_INVOICE

← Methods ⚡ Exceptions Properties

Parameter	Type	Pa...	Op...	Typing...	Associated Type
IV_RAW_INV	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_S_RAW_POB
RV_PROCESSABLE	Returning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Type	XFELD

Figure 4.70 HANDLE_ASSUMED_INVOICE Method

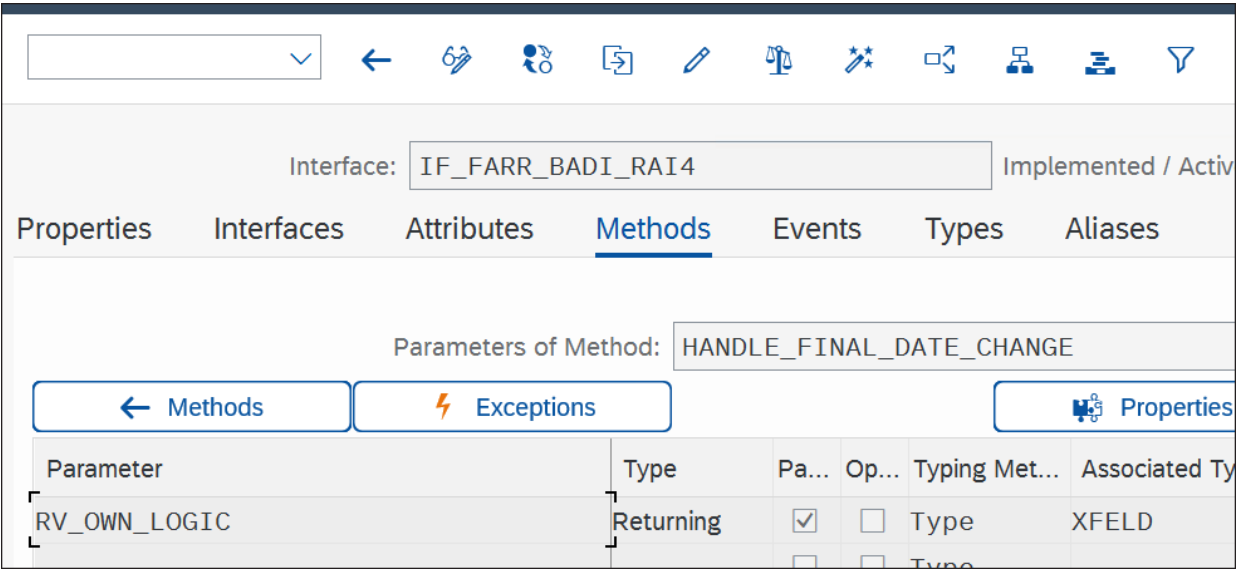


Figure 4.71 HANDLE_FINAL_DATE_CHANGE Method

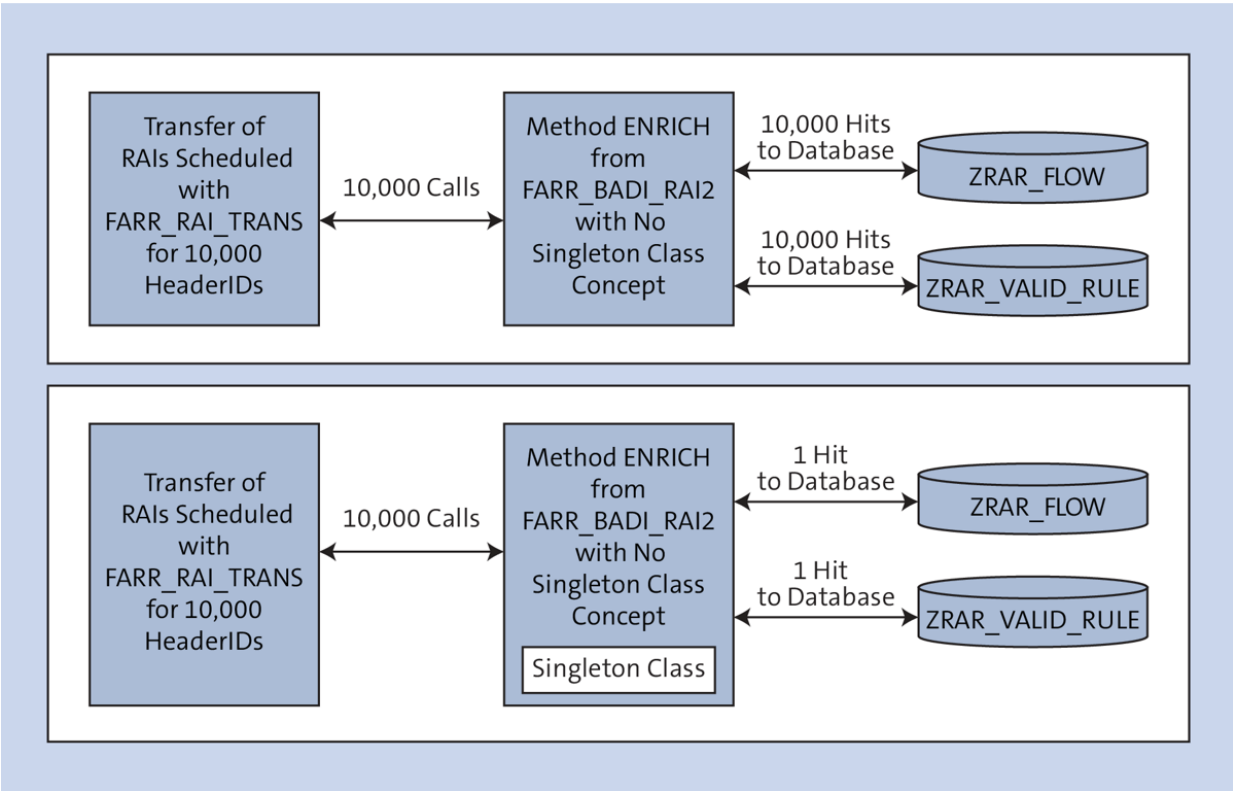


Figure 4.72 Singleton Class in ENRICH: FARR_BADI_RAI2

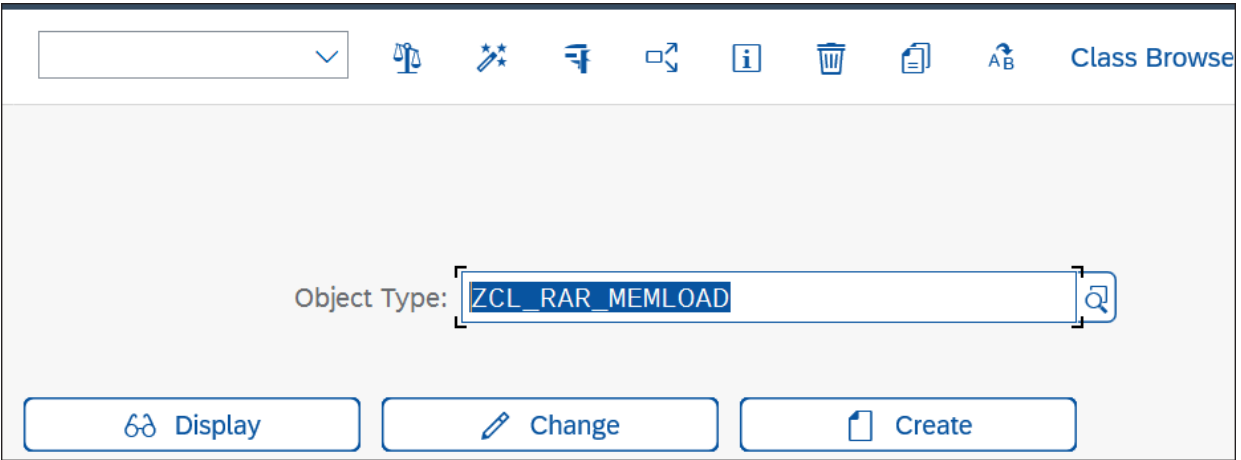


Figure 4.73 Creating a Singleton Class

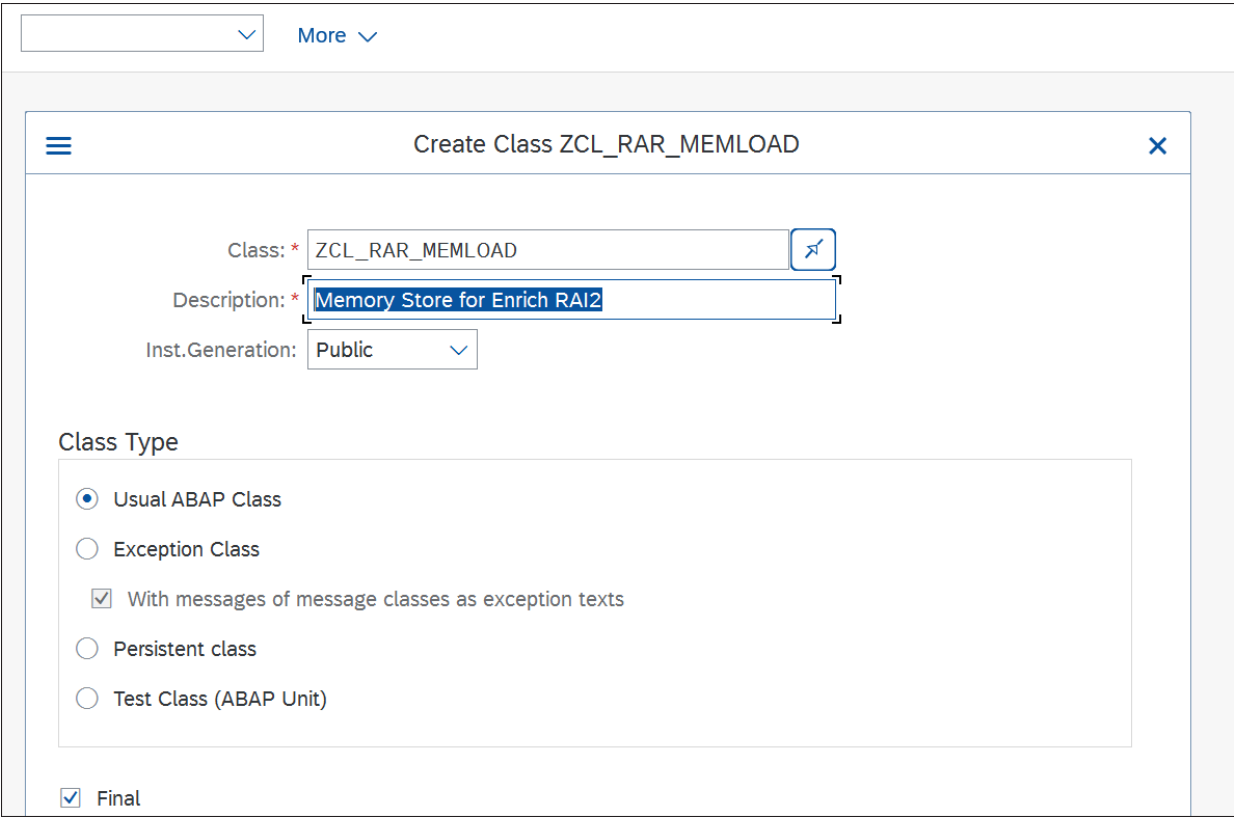


Figure 4.74 Singleton Class

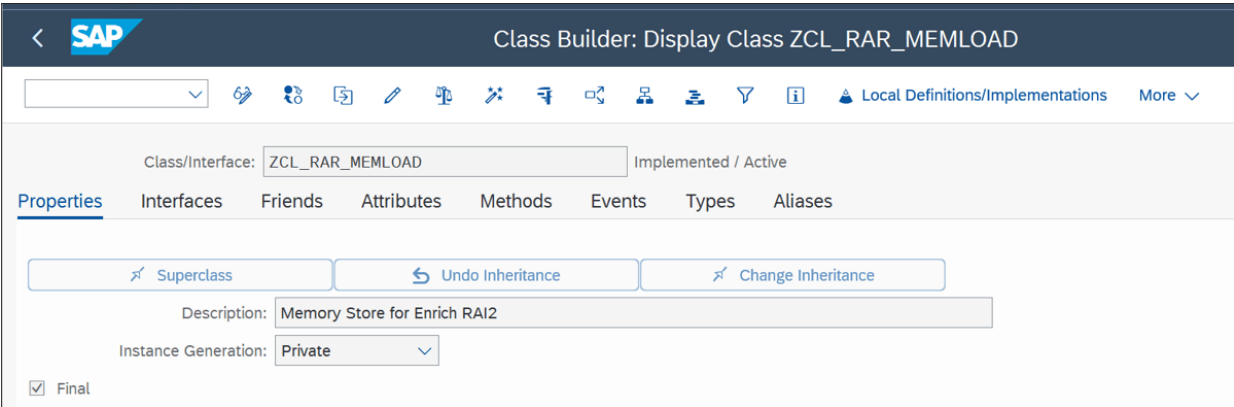


Figure 4.75 Singleton Class Creation

SAP Class Builder: Change Class ZCL_RAR_MEMLOAD

Class/Interface: ZCL_RAR_MEMLOAD Implemented / Active

Properties Interfaces Friends **Attributes** Methods Events Types Aliases

Properties Filter

Attribute	Level	Visibility	Re...	Typing	Associated Type
GO_INSTANCE	Static Attribute	Private	<input type="checkbox"/>	Type Ref To	ZCL_RAR_MEMLOAD
GT_ZRARC_FLOW_MGMT	Static Attribute	Private	<input type="checkbox"/>	Type	ZRAR_FLOW_TT

Figure 4.76 Singleton Class Attributes

SAP Class Builder Class ZCL_RAR_ME

Ty.	Paramet...	Typi...	Descripti...

Method: CONSTRUCTOR

```
1  method CONSTRUCTOR.  
2    DATA : lt_ZRARC_FLOW_MGMT TYPE ZRARC_FLOW_MGMT_tt.  
3  
4  
5    CLEAR : lt_ZRARC_FLOW_MGMT[], GT_ZRARC_FLOW_MGMT[].  
6    SELECT * FROM ZRAR_FLOW  
7    into TABLE lt_ZRARC_FLOW_MGMT  
8    where calling_cls_prg = 'IF_FARR_BADI_RAI2~ENRICH'.  
9    IF SY-SUBRC EQ 0.  
10       GT_ZRARC_FLOW_MGMT[] = LT_ZRARC_FLOW_MGMT[].  
11    ENDIF.  
12  
13  endmethod.
```

Figure 4.77 Example of a Constructor of Singleton Class


Ty.	Parameter	Typing	Description
	value(RO_INSTANCE)	TYPE REF TO ZCL_RAR_MEMLOAD	Memory Store for Enrich RAI2
....			
Method: GET_INSTANCE			active
1	method GET_INSTANCE.		
2			
3	if go_instance is NOT bound.		
4	CREATE OBJECT go_instance.		
5	endif.		
6	ro_instance = go_instance.		
7	endmethod.		

Figure 4.78 Singleton Class Instance Method

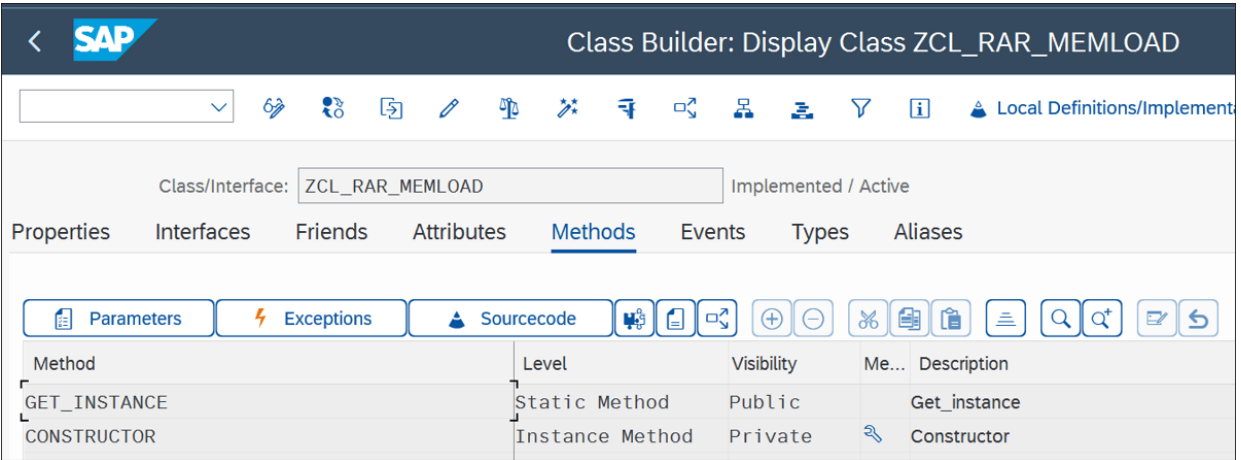


Figure 4.79 Two Methods of the Sample Singleton Class

```
IF go_object IS NOT BOUND.  
  CALL METHOD zclar_arl_mem_load=>get_instance  
  RECEIVING  
    ro_instance = go_object.  
  
ENDIF.  
CLEAR : gt_datval[], gt_precedance[].  
IF go_object IS BOUND.  
  gt_ZRARC_FLOW_MGMT[] = go_object->gt_ZRARC_FLOW_MGMT.
```

Figure 4.80 Singleton Class

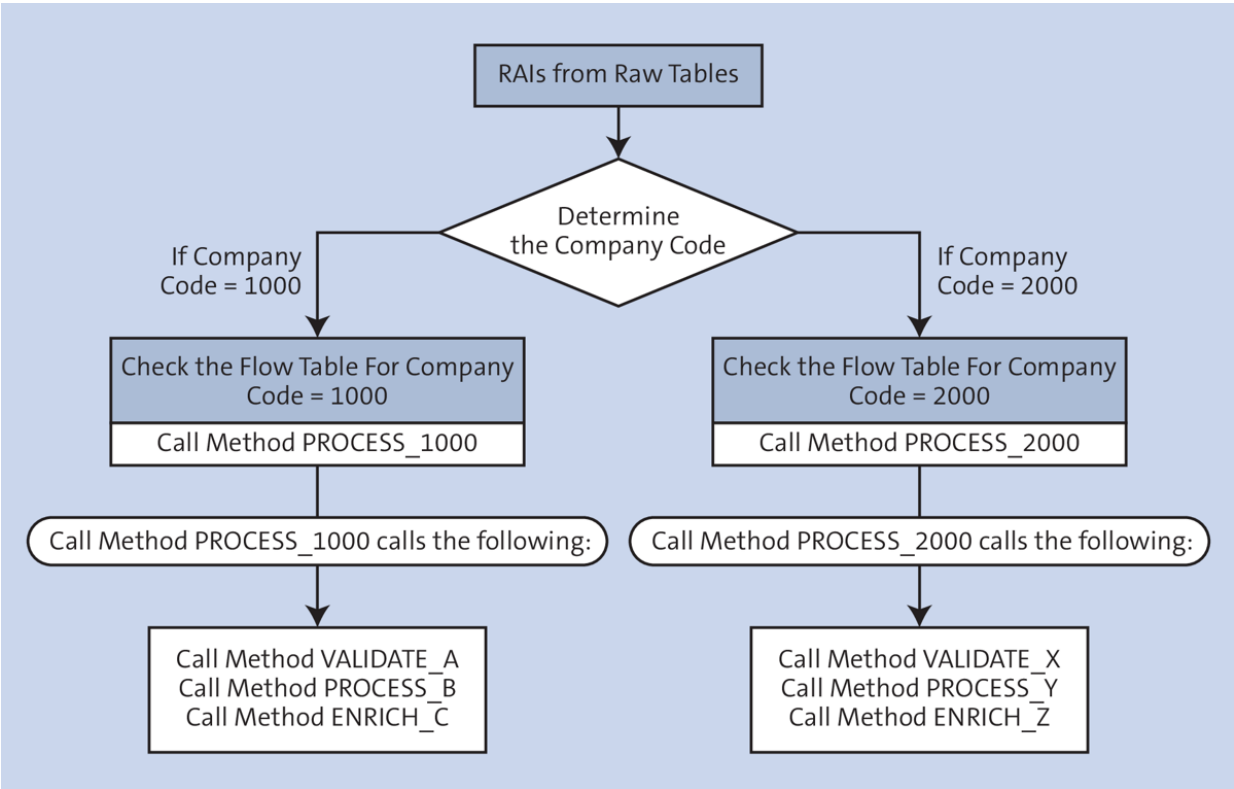


Figure 4.81 Dynamic Flow Determination at Runtime Based on Company Code

SAP Dictionary: Display Table

Transparent Table: ZRARC_FLOW_MGMT Active
 Short Description: Dynamic Processing Flow Management table

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT		3	0	0 Client
<input type="checkbox"/> BUKRS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	BUKRS	CHAR		4	0	0 Company Code
<input type="checkbox"/> SRCDOC_COMP	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAI_SRCCO	CHAR		3	0	0 Sender Component of Source Item
<input type="checkbox"/> SCRDOC_TYPE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_RAI_SRCTY	CHAR		4	0	0 Source Document Item Type
<input type="checkbox"/> CONTRACT_CAT	<input type="checkbox"/>	<input type="checkbox"/>	FARR_CONTRACT_CATE	CHAR		4	0	0 Contract Category
<input type="checkbox"/> FLOW	<input type="checkbox"/>	<input type="checkbox"/>	ZFLOW	CHAR		25	0	0 Flow
<input type="checkbox"/> CALLING_CLS_PRG	<input type="checkbox"/>	<input type="checkbox"/>	ZCALLING_CLASS	CHAR		61	0	0 Calling Class or Program
<input type="checkbox"/> CLSNAME	<input type="checkbox"/>	<input type="checkbox"/>	SEOCLSNAME	CHAR		30	0	0 Object Type Name
<input type="checkbox"/> CPDNAME	<input type="checkbox"/>	<input type="checkbox"/>	SEOCPDNAME	CHAR		61	0	0 Full Component Name

Figure 4.82 Flow Table

< **SAP** New Entries: Overview of Added Entries

Dynamic Processing Flow Management table

	CoCd	Send.Comp.	Src...	Contr....	Flow	calling cls prg	Class/Interface	Interface Component
<input type="checkbox"/>	1000	ZX	*	*	SWITCH	IF_FARR_BADI_RAI2~ENRICH	ZCLRAR_BADIRAI2	PROCESS_1000
<input type="checkbox"/>	2000	ZA	*	*	SWITCH	IF_FARR_BADI_RAI2~ENRICH	ZCLRAR_BADIRAI2	PROCESS_2000
<input type="checkbox"/>	1000	ZX	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	VALIDATE_A
<input type="checkbox"/>	1000	ZX	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	PROCESS_B
<input type="checkbox"/>	1000	ZX	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	ENRICH_C
<input type="checkbox"/>	2000	ZA	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	VALIDATE_X
<input type="checkbox"/>	2000	ZA	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	PROCESS_Y
<input type="checkbox"/>	2000	ZA	*	*	ALLOW	ZCLRAR_BADIRAI2	ZCLRAR_BADIRAI2	ENRICH_Z

Figure 4.83 Flow Table Entries


```

1) METHOD if_farr_badi_rai2~enrich.
2)   DATA: lv_meth          TYPE string,
3)         lv_class        TYPE string,
4)         lt_ptab         TYPE abap_parmbind_tab,
5)         lt_etab         TYPE abap_excpcbind_tab,
6)         lt_raimsg       TYPE farr_tt_rai_msg,
7)         lt_zrarc_flow_mgmt TYPE zrarc_flow_mgmt_tt.
8)   DATA: exc_ref        TYPE REF TO cx_sy_dyn_call_error.
9)
10)  READ TABLE ct_rai2_mi INTO DATA(ls_mi_determine) INDEX 1.
11)  IF sy-subrc EQ 0.
12)    SELECT * FROM zrarc_flow_mgmt
13)    INTO TABLE lt_zrarc_flow_mgmt
14)    WHERE bukrs          = ls_mi_determine-bukrs
15)      and FLOW           = 'SWITCH'
16)      AND calling_cls_prg = 'IF_FARR_BADI_RAI2~ENRICH'.
17)  IF sy-subrc EQ 0.
18)    SORT lt_zrarc_flow_mgmt ASCENDING.
19)  ENDIF.
20)
21)  lt_ptab = VALUE #( ( name = 'IV_RAIC'
22)                    kind = cl_abap_objectdescr=>exporting
23)                    value = REF #( iv_raic ) )
24)                ( name = 'CT_RAI2_MI'
25)                    kind = cl_abap_objectdescr=>changing
26)                    value = REF #( ct_rai2_mi ) )
27)                ( name = 'CT_RAI2_CO'
28)                    kind = cl_abap_objectdescr=>changing
29)                    value = REF #( ct_rai2_co ) )
30)                ( name = 'CT_MESSAGES'
31)                    kind = cl_abap_objectdescr=>changing
32)                    value = REF #( lt_raimsg ) ) ).
33)  lt_etab = VALUE #( ( name = 'OTHERS' value = 4 ) ).

```

Figure 4.84 Dynamic Call of Methods

Dynamic Processing Flow Management table							
CoCd	Send.Comp.	Src...	Contr...	Flow	calling cls prg	Class/Interface	Interface Component
1000	ZX	*	*	SWITCH	IF_FARR_BADI_RAI2~ENRICH	ZCLRAR_BADIRAI2	PROCESS_1000

Figure 4.85 Switch Method Entry

```

34> READ TABLE lt_zrarc_flow_mgmt INTO DATA(ls_zrarc_flow_mgmt) WITH KEY bukrs = ls_mi_determine-bukrs
35>                                srcdoc_comp = ls_mi_determine-srcdoc_comp
36>                                FLOW = 'SWITCH'.
37> IF sy-subrc EQ 0.
38>   TRY.
39>     CALL METHOD (ls_zrarc_flow_mgmt-clsname)=>(ls_zrarc_flow_mgmt-cpdname)
40>       PARAMETER-TABLE
41>         lt_ptab
42>     EXCEPTION-TABLE
43>       lt_etab.
44>     IF lt_raimsg[] IS NOT INITIAL.
45>       APPEND LINES OF lt_raimsg TO ct_messages.
46>     ENDIF.
47>     CATCH cx_sy_dyn_call_error INTO exc_ref.
48>       MESSAGE exc_ref->get_text( ) TYPE 'I'.
49>     ENDTRY.
50>   ENDIF.
51> ENDIF.
52> ENDMETHOD.

```

Figure 4.86 Dynamic Call of Methods

Structure	
<input type="checkbox"/>	∨ [Revenue Accounting]
<input type="checkbox"/>	> Inbound Processing
<input type="checkbox"/>	> Revenue Accounting Contracts (Classic)
<input type="checkbox"/>	> Revenue Accounting Contracts
<input type="checkbox"/>	> Integration with Cost Object Controlling
<input type="checkbox"/>	> Revenue Accounting Postings
<input type="checkbox"/>	> Revenue Accounting Analytics

Figure 5.1 Configuration of Contract Management

Select Contract Management for Contract Categories				
CoCd	Company Name	Contr. Cat	Contract Category Description	Contract With CM Instead of CM Classic
<input type="checkbox"/> US19		0001	Revenue Contract	<input type="checkbox"/>
<input type="checkbox"/> US19		0002	Revenue contracts (Classic)	<input type="checkbox"/>

Figure 5.2 Selection of Contract Management Type

Table: No texts

Text table:

Layout:

Maximum no. of hits: Maintain entries

Get Field:

Selection Criteria

Fld name	O...	Fr.Value	To value	More	Output	Technical name
Balance Presentation	<input type="button" value="↕"/>			<input type="button" value="→"/>	<input checked="" type="checkbox"/>	LIAB_METHOD
Contr. Term. Date	<input type="button" value="↕"/>			<input type="button" value="→"/>	<input checked="" type="checkbox"/>	ASSET_IMPAIRMENT_DATE
RAR version code	<input type="button" value="↕"/>			<input type="button" value="→"/>	<input checked="" type="checkbox"/>	RAR_VERSION_CODE
Exchange rate	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	EXCHANGE_RATE
Exchange Rate 2	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	EXCHANGE_RATE2
Exchange Rate 3	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	EXCHANGE_RATE3
Created by	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	CREATED_BY
Created on	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	CREATED_ON
Changed by	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	LAST_CHANGED_BY
Changed on	<input type="button" value="↕"/>				<input checked="" type="checkbox"/>	LAST_CHANGED_ON

☰ RAR version code (2) 3 Entries found ✕

RAR version code	Short Descript.
	Version Undefined/Unknown
X	Version 2.0
A	Version 1.3

Figure 5.3 Table FARR_D_CONTRACT with Values for Optimized Contract Management

AccP	Name of Accounting Principle	Presentat.
IFRS	International Financial Reporting Standards	2 Contract Liability/Contract Asset
LG	Local GAAP	1 Unbilled Receivable/Deferred Revenue

Figure 5.4 Setup of Parallel Accounting Principles

Search in Table: **FARR_D_MAPPING** Mapping Table to map source document to POBs of RA-Contracts

Number of hits: **6**

Runtime: **0** Maximum no. of hits: **500**

Insert Column:

	POB	Send.Comp.	SourceSys.	SrcItem Type	Source Item ID	POB Contract	AccP	Ref. Type	Reference ID	RevAccCI	Header ID	SoftDelete	Arch.Date
<input type="checkbox"/>	71628	SD		SDOI	0041011458000020	300308024	IFRS	SDO	0041011458	SD01	41011458		
<input type="checkbox"/>	71629	SD		SDOI	0041011458000030	300308024	IFRS	SDO	0041011458	SD01	41011458		
<input type="checkbox"/>	71630	SD		SDOI	0041011458000040	300308024	IFRS	SDO	0041011458	SD01	41011458		
<input type="checkbox"/>	75347	SD		SDOI	0041011458000020	300308099	LG	SDO	0041011458	SD01	41011458		
<input type="checkbox"/>	75348	SD		SDOI	0041011458000030	300308099	LG	SDO	0041011458	SD01	41011458		
<input type="checkbox"/>	75349	SD		SDOI	0041011458000040	300308099	LG	SDO	0041011458	SD01	41011458		

Figure 5.5 Multiple Contracts Created for One Sales Order

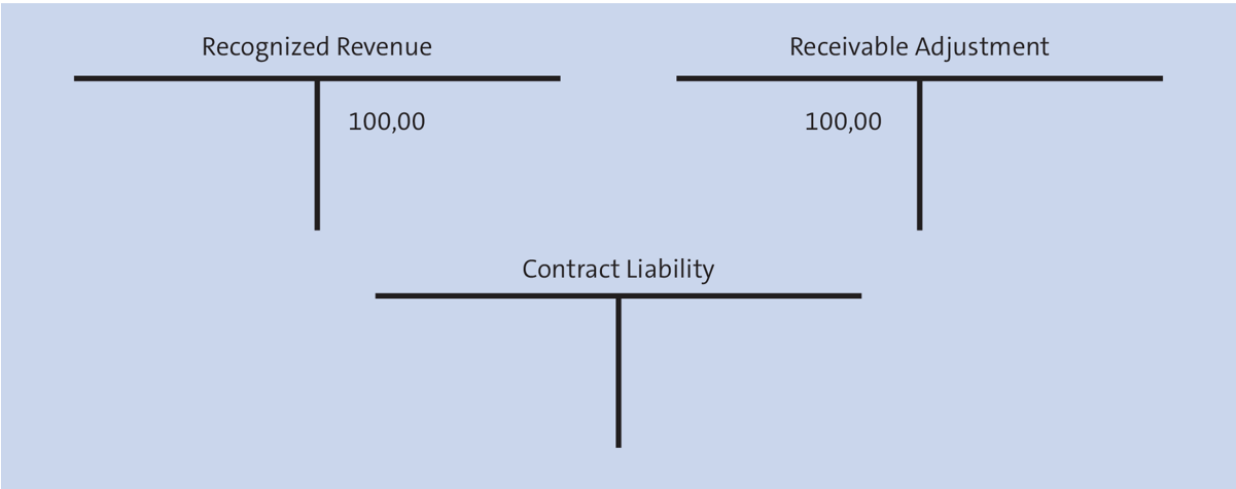


Figure 5.6 Recognized Revenue with the CA/CL Presentation Method

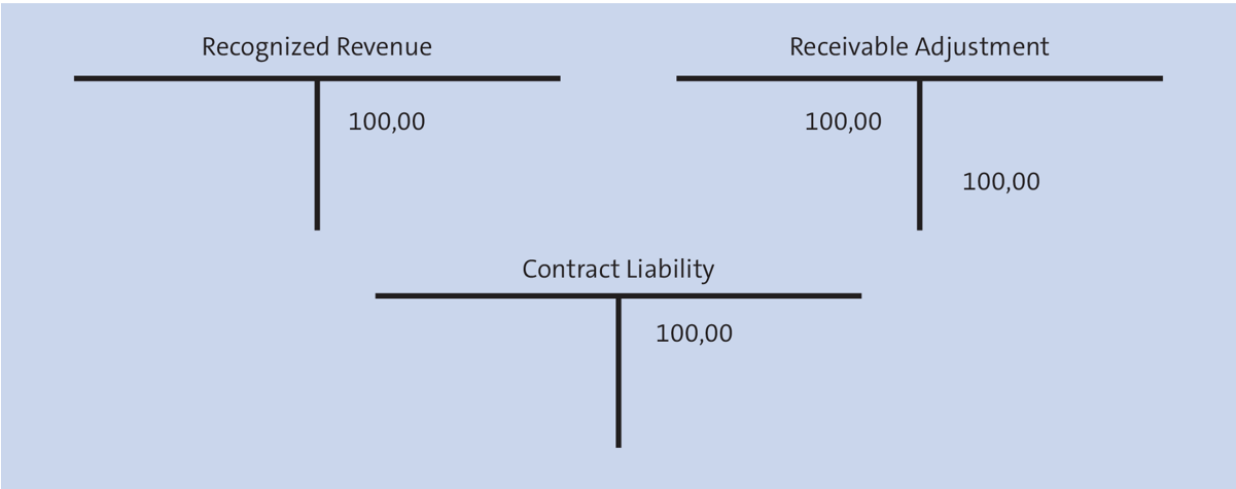


Figure 5.7 Posting of Contract Liability in the CA/CL Presentation Method

▼ Check Where-Used List Delete Copy... Rename... Application help More ▼

Enhancement Spot

BAdI Name

FARR_BADI_RAI2

Display Change Create

Figure 5.8 BAdI to Be Implemented



Implementing Class	
Interface:	<u>IF_FARR_BADI_RAI2</u>
Implementing Class:	<u>ZCL</u>  
Method	Short Description
■ IF_FARR_BADI_RAI2~ENRICH	Enrich Processable Revenue Accounting I...
■ IF_FARR_BADI_RAI2~CHECK_BEFORE_SAVE	Check Processable Revenue Accounting I...
■ FILL_REFERENCE	

Figure 5.9 Implementing Class with Methods to Be Used

Main Item (3)		Condition Item (3)											
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>													
<input type="checkbox"/>	SourceSys	Due Date	Posting Date	SrcItem Type	Source Item ID	Subarea	RevAccCl	Header ID	ItemID	Ref. Type	Ref ID	InitLoad	
<input type="checkbox"/>		01.10.2022	01.10.2022	SDII	0800260292000010	518	<u>SD03</u>	800260292	10				
<input type="checkbox"/>		01.10.2022	01.10.2022	SDII	0800260300000010	176	<u>SD03</u>	800260300	10				
<input type="checkbox"/>		01.10.2022	01.10.2022	SDII	0800260301000010	099	<u>SD03</u>	800260301	10				

Figure 5.10 Due Date Updated with Posting Date

CoCd	Acc...	Reconcil. Key	POB	CnTy	Cate...	D/C	Posting GUID	Year	Peri...	TC Amount	Crcy
AU02	IFR...	20220120000101	119078		CL	H	000D3A2EDBC91EDDB3F54F2873BA3AA0	20...	12	31.816,08-	AUD
					CL	<input type="checkbox"/>				31.816,08-	AUD
AU02	IFR...	20220120000101	119078	ZPRO	IC	S	000D3A2EDBC91EDDB3F53FA1D03BB9F8	20...	12	80.424,21	AUD
					IC	<input type="checkbox"/>				80.424,21	AUD
AU02	IFR...	20220120000101	119078		RA	S	000D3A2EDBC91EDDB3F54F2873BA3AA0	20...	12	31.816,08	AUD
AU02	IFR...	20220120000101	119078	ZPRO		H	000D3A2EDBC91EDDB3F53FA1D03BB9F8	20...	12	80.424,21-	AUD
AU02	IFR...	20220120000101	119078	ZPRO		S	000D3A2EDBC91EDDB3F54C2F7885FA95	20...	12	48.608,13	AUD
					RA	<input type="checkbox"/>				0,00	AUD
AU02	IFR...	20220120000101	119078	ZPRO	RV	H	000D3A2EDBC91EDDB3F54C2F7885FA95	20...	12	48.608,13-	AUD
					RV	<input type="checkbox"/>				48.608,13-	AUD
						<input type="checkbox"/>				0,00	AUD

Figure 5.11 Contract Liability Calculation

CoCd	Acc...	Reconcil. Key	POB	CnTy	Cate...	D/C	Posting GUID	Year	Peri...	TC Amount
AU02	IFR...	20220120000101	119079	ZPR0	IC	S	000D3A2EDBC91EDDB3F5710DBF54FB99	20...	12	20.106,05
					IC	<input type="checkbox"/>				■ 20.106,05
AU02	IFR...	20220120000101	119079	ZPR0	RA	H	000D3A2EDBC91EDDB3F5710DBF54FB99	20...	12	20.106,05-
AU02	IFR...	20220120000101	119079	ZPR0		S	000D3A2EDBC91EDDB3F57C1293ECBC3F	20...	12	48.608,13
					RA	<input type="checkbox"/>				■ 28.502,08
AU02	IFR...	20220120000101	119079	ZPR0	RV	H	000D3A2EDBC91EDDB3F57C1293ECBC3F	20...	12	48.608,13-
					RV	<input type="checkbox"/>				■ 48.608,13-
						<input type="checkbox"/>				■ ■ 0,00

Figure 5.12 Contract Asset Calculation

Reserved Condition Types

Allocation Difference:	<input type="text" value="CORR"/>
Right-of-Return Revenue Adjustment:	<input type="text" value="RROR"/>
Right-of-Return Cost Adjustment:	<input type="text" value="CROR"/>
Exchange Rate Difference:	<input type="text" value="EXRD"/>
Early Termination:	<input type="text"/>

Figure 5.13 Reserved Condition Types

CoCd	Acc...	Reconcl. Key	POB	CnTy	Cate...	D/C	Posting GUID	Year	Peri...	TC Amount	Crcy	LC Amount	LCurr	Second LC	LCur2
A...	IFR...	20220100000102	100041	EXRD	ED	H	000D3A4425D01EDDA8D28FD2312808C0	20...	10	0,00	AUD	0,00	AUD	1.225,00-	USD
										0,00	AUD				
AU02	IFR...	20220080000101	100041	ZPR0	IC	S	000D3A2A85EB1EDD9FE8B6F081D0F4D4	20...	8	100.000,00	AUD	100.000,00	AUD	68.292,00	USD
AU02	IFR...	20220090000102	100041	ZPR0		S	000D3A2A85EB1EDD9FE8B6F081D1F4D4	20...	9	100.000,00	AUD	100.000,00	AUD	68.292,00	USD
AU02	IFR...	20220100000102	100041	ZPR0		S	000D3A4425D01EDDA8D28FD2312768C0	20...	10	100.000,00	AUD	100.000,00	AUD	69.517,00	USD
										300.000,00	AUD				
AU02	IFR...	20220080000101	100041	ZPR0	RA	H	000D3A2A85EB1EDD9FE8B6F081D0F4D4	20...	8	100.000,00-	AUD	100.000,00-	AUD	68.292,00-	USD
AU02	IFR...	20220090000101	100041	ZPR0		S	000D3A2EDBC91EED9F8278C84275B2AD	20...	9	100.000,00	AUD	100.000,00	AUD	68.292,00	USD
AU02	IFR...	20220090000102	100041	ZPR0		H	000D3A2A85EB1EDD9FE8B6F081D1F4D4	20...	9	100.000,00-	AUD	100.000,00-	AUD	68.292,00-	USD
AU02	IFR...	20220100000101	100041	ZPR0		S	000D3A2EDBC91EDDA3FCE1DC9929622D	20...	10	100.000,00	AUD	100.000,00	AUD	68.292,00	USD
AU02	IFR...	20220100000102	100041	EXRD		S	000D3A4425D01EDDA8D28FD2312808C0	20...	10	0,00	AUD	0,00	AUD	1.225,00	USD
AU02	IFR...	20220100000102	100041	ZPR0		H	000D3A4425D01EDDA8D28FD2312768C0	20...	10	100.000,00-	AUD	100.000,00-	AUD	69.517,00-	USD
AU02	IFR...	20220110000101	100041	ZPR0		S	000D3ABD0E3E1EDDA8D2C87E52112542	20...	11	100.000,00	AUD	100.000,00	AUD	68.292,00	USD
										0,00	AUD				
AU02	IFR...	20220090000101	100041	ZPR0	RV	H	000D3A2EDBC91EED9F8278C84275B2AD	20...	9	100.000,00-	AUD	100.000,00-	AUD	68.292,00-	USD
AU02	IFR...	20220100000101	100041	ZPR0		H	000D3A2EDBC91EDDA3FCE1DC9929622D	20...	10	100.000,00-	AUD	100.000,00-	AUD	68.292,00-	USD
AU02	IFR...	20220110000101	100041	ZPR0		H	000D3ABD0E3E1EDDA8D2C87E52112542	20...	11	100.000,00-	AUD	100.000,00-	AUD	68.292,00-	USD
										300.000,00-	AUD				
										0,00	AUD				

Figure 5.14 Contract with FX Differences

Configure Accounting Principle-specific Settings			
AccP	Cost Recog	Level for Posting Contract Lia. and Ass.	Default Contract Lia. and Ass. Calculation Method
<input type="checkbox"/> IFRS	<input type="checkbox"/>	Post at Performance Obligation Level	Calculate per POB and net by Contract

Figure 5.15 CA/CL on the Performance Obligations Level

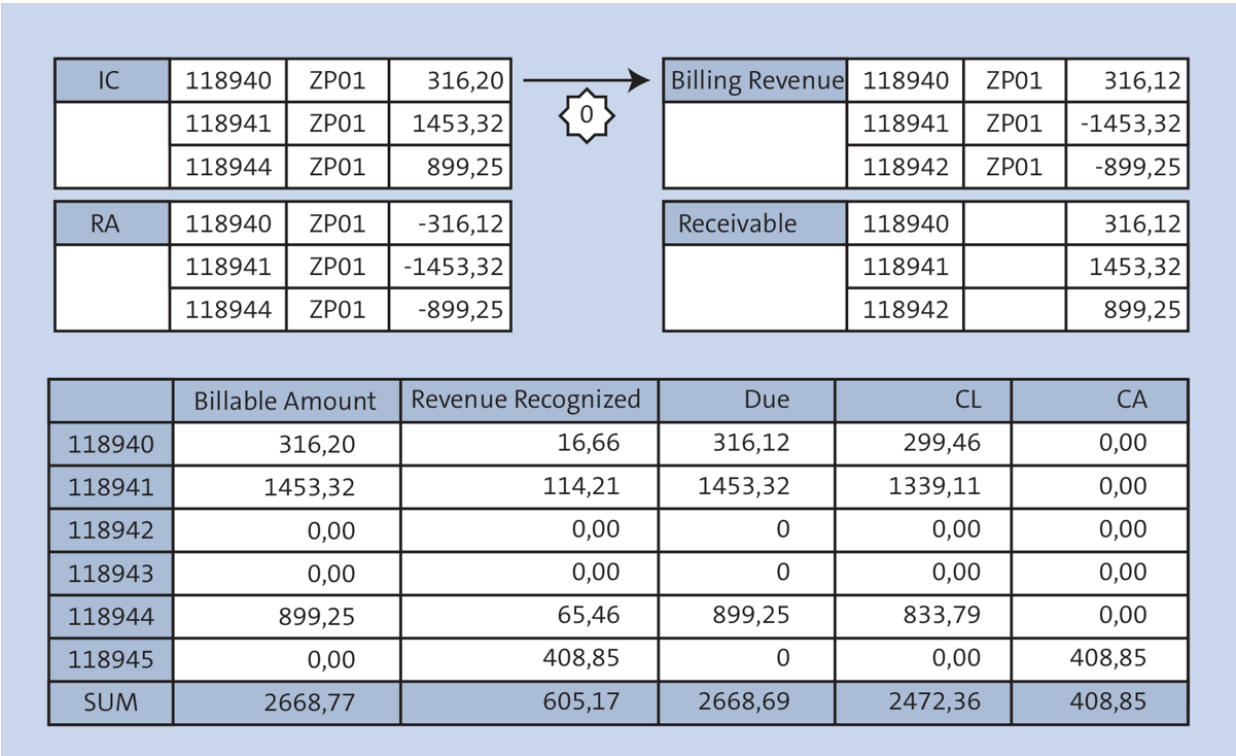


Figure 5.16 Posting after Processing of Invoice RAIs

Supported Company Codes per Accounting Principle									
	AccP	CoCd	Transf.Dat	Status	Adopt.Date	Src.Acc.Pr	Ext.Acc.Pr	End.D.Usq.	App.Ear.OP
<input type="checkbox"/>	IFRS	AU02	30.06.2022	P0 Productive	∨ 01.07.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	CH04	31.10.2020	P0 Productive	∨ 01.11.2020	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	FR02	31.10.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	GB04	31.10.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	ID02	30.06.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	JP02	30.06.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	NL03	31.10.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	US19	31.10.2020	P0 Productive	∨ 01.11.2020	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	ZA02	30.06.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>
<input type="checkbox"/>	IFRS	ZA03	30.06.2022	P0 Productive	∨ 01.11.2022	IFRS	<input type="checkbox"/>		<input type="checkbox"/>

Figure 5.17 Assignment of Company Codes











Number Range Object: FARR_CTRID		Revenue Recognition Contract
 Intervals	 Intervals	 NR Status
Number Range Object: FARR_POB		Revenue Recognition Performance Obligation
 Intervals	 Intervals	 NR Status
Number Range Object: FARR_RN_ID		Revenue Accounting Run ID
Company Code: GB04		
 Intervals	 Intervals	 NR Status

Figure 5.18 Setup of Numbering Objects

No	From No.	To Number	NR Status	Ext
01	00000000100000	00000000199999	102399	<input type="checkbox"/>
02	00000000200000	00000000299999	0	<input type="checkbox"/>
03	00000000700000	00000000799999	0	<input type="checkbox"/>
04	00000000600000	00000000699999	0	<input type="checkbox"/>
05	00000000300000	00000000399999	0	<input type="checkbox"/>

Figure 5.19 Number Ranges for Contracts and POBs

Customizing of Contract Category			
	Contr. Cat	Description	No. Range
<input type="checkbox"/>	0001	Revenue Contract	01
<input type="checkbox"/>	0002	Revenue contracts (Classic)	02

Figure 5.20 Contract Categories

Revenue Accounting Period Close					
	CoCd	AccP	Fr. FYear	Fr. Period	Status
<input type="checkbox"/>	AU02	IFRS	2022	11	Open
<input type="checkbox"/>	C002	IFRS	2020	11	Open
<input type="checkbox"/>	FR02	IFRS	2022	11	Open
<input type="checkbox"/>	GB04	IFRS	2022	11	Open
<input type="checkbox"/>	ID02	IFRS	2022	11	Open
<input type="checkbox"/>	JP02	IFRS	2022	11	Open
<input type="checkbox"/>	NL03	IFRS	2022	11	Open
<input type="checkbox"/>	US19	IFRS	2020	11	Open
<input type="checkbox"/>	ZA02	IFRS	2022	11	Open
<input type="checkbox"/>	ZA03	IFRS	2022	11	Open

Figure 5.21 Periods Management in RAR

Configuration of Performance Obligation Types	
POB Type	Description
<input type="checkbox"/> EVNT_MAN	Event Based POB - Manual Fulfillment
<input type="checkbox"/> EVNT_MAX	Event Based POB - Manual Fulfillment
<input type="checkbox"/> EVNT_TNM	Time and material fulfillment
<input type="checkbox"/> TIME_CREAT	Time Based POB - Start Date Available on Creation
<input type="checkbox"/> ZPOC	PoC method based POB

Figure 5.22 POB Types Definition Initial Screen

POB Type:

Description:

General Data

Perf.Obligat.Nam:

No Cost Recognition

Fulfillment Data

Fulfillment Type:

Event Type:

Allocation Data

Excl. from Alloc.

Residual

StdAlone Price:

SSP Tolerance:

SSP Tol. Perc.: %

Currency:

Figure 5.23 POB Types Entering Details

Fulfillment Data

Fulfillment Type: ▾

Event Type: ▾

Start Date Type: ▾

Duration:

Deferral Method:

Figure 5.24 Time-Based POB Type

Assign BRFplus to Revenue Accounting Item Classes		
	RevAccCl	BRF Application
<input type="checkbox"/>	SD01	FARR_AP_SD_PROCESS_PG1

Figure 5.25 BRFplus Assignment of Application

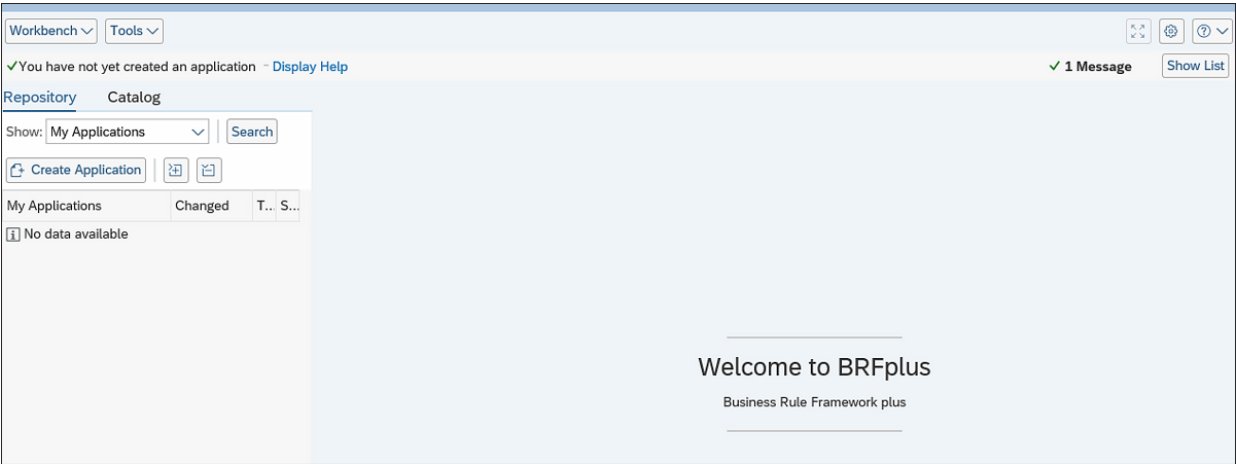


Figure 5.26 BRFplus Initial Screen

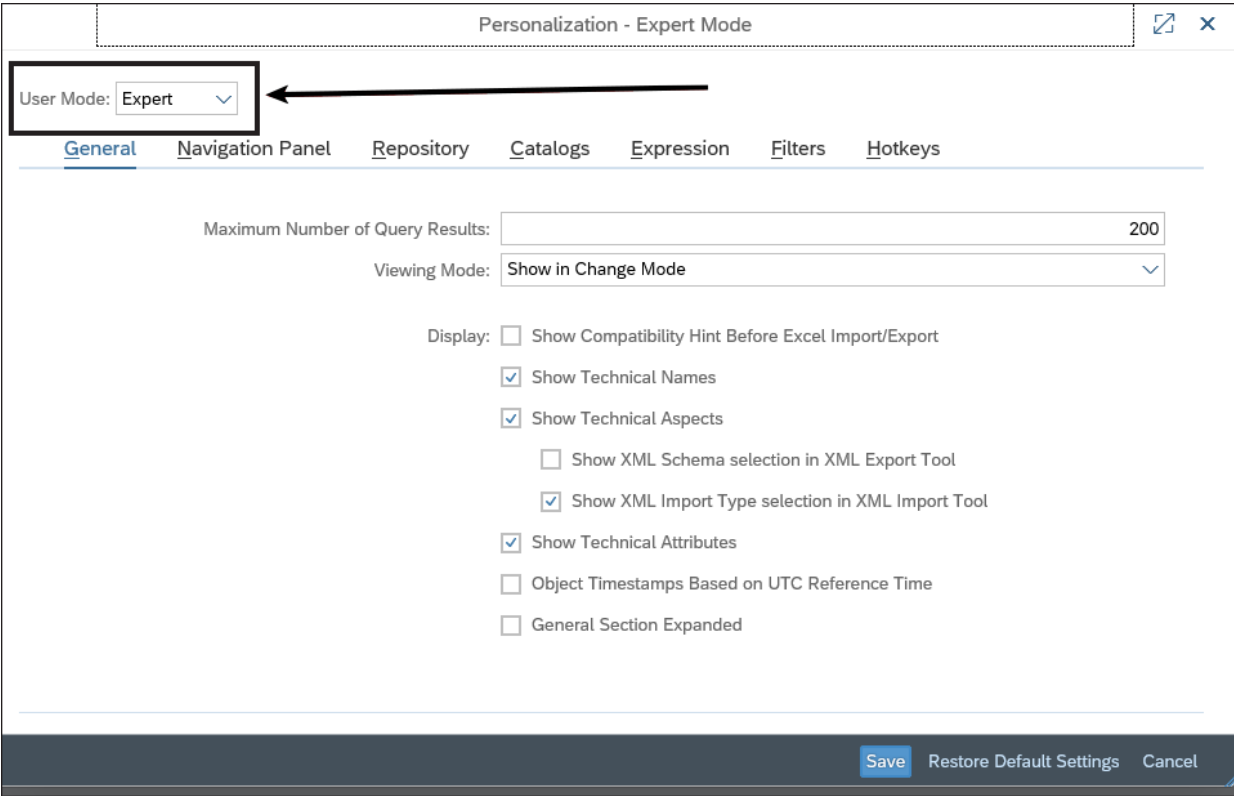


Figure 5.27 Expert Mode Selected

Search

Search Criteria

Application ...	is equal to	FARR*	+	-
Object Type	is equal to	Application	+	-
Name	is equal to	*	+	-

Also include objects from default BRFplus application:

Maximum Number of Results: 200

Search Clear Reset

Ok Cancel

Figure 5.28 BRFplus Selection

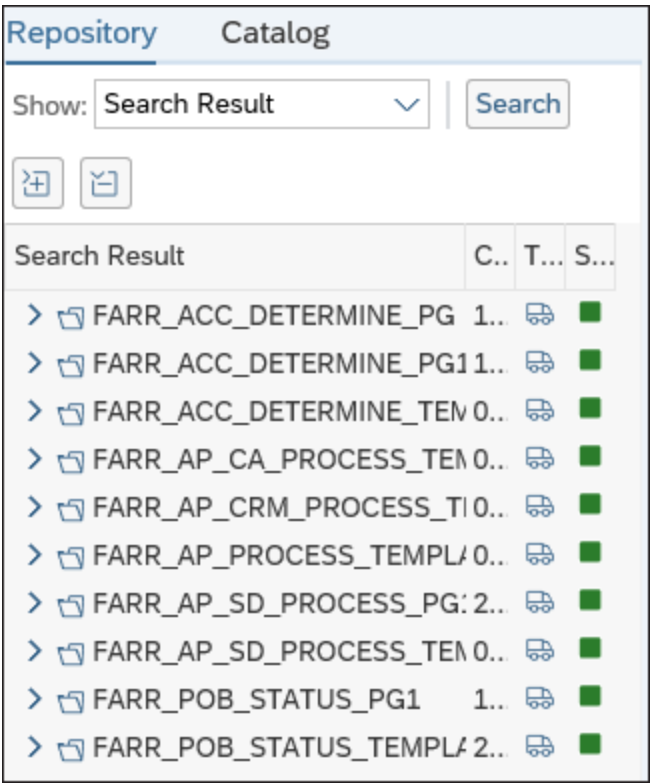


Figure 5.29 BRFplus Applications

Repository **Catalog**

Show: Search Result Search

Search Result C.. T... S...

- > FARR_ACC_DETERMINE_PG 1..
- > FARR_ACC_DETERMINE_PG11..
- > FARR_ACC_DETERMINE_TEN 0..
- > FARR_AP_CA_PROCESS_TEN 0..
- > FARR_AP_CRM_PROCESS_TI 0..
- > FARR_AP_PROCESS_TEMPL/ 0..
- ▼ FARR_AP_SD_PROCESS_PG: 2..
 - > Last Changed (50 of 200)
 - > Data Object
 - ▼ Expression
 - > DB Lookup
 - ▼ Decision Table
 - > DT_PROCESS_BOM 1..
 - > DT_PROCESS_COMPI 1..
 - > DT_PROCESS_DEFER 1..
 - > DT_PROCESS_HEADE 2..
 - > **DT_PROCESS_POB 1..**
 - > DT_PROCESS_POB_A 1..
 - > DT_PROCESS_SSP 1..
- > Table Operation
- > Function

Decision Table: DT_PROCESS_POB

< Back Edit Check Save Activate Transport Delete ▾ More

General

Detail

Export To Excel Context Overview Start Simulation

Table Contents

Find: Next Previous

checkbox	KUNNR	MATNR
<input type="checkbox"/>		=VXA01ES7565
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		

Figure 5.30 BRFplus Decision Table

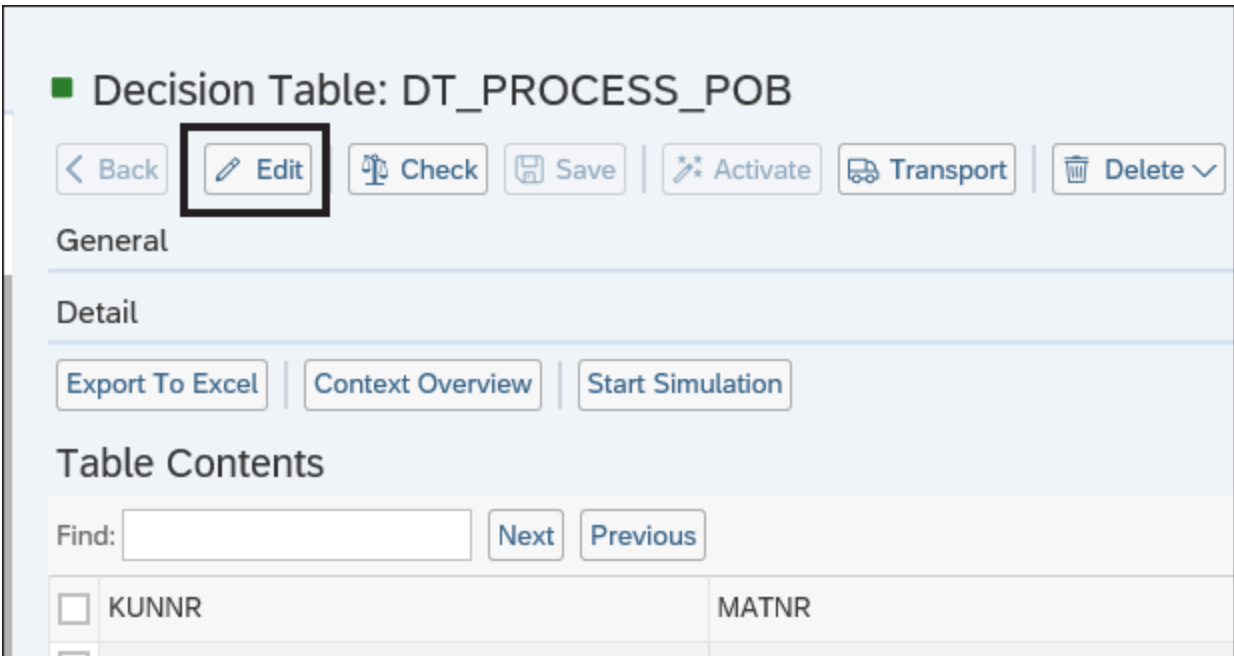


Figure 5.31 Editing a Decision Table

The screenshot shows a Microsoft Excel spreadsheet in Protected View. The ribbon at the top includes File, Home, Insert, Page Layout, Formulas, Data, Review, View, Automate, and Help. A yellow warning bar at the top reads: "PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View." with an "Enable Editing" button. The active cell is A1, containing the text "Customer". The spreadsheet has six columns: A (Customer), B (Material), C (Sales Org.), D (Distr. Channel), E (Compound Group), and F (Accounting Principle). Row 1 contains the headers for these columns. Row 2 contains the value "=VXA01ES7565" in the Material column. Rows 3 through 13 are empty.

	A	B	C	D	E	F
1	Customer	Material	Sales Org.	Distr. Channel	Compound Group	Accounting Principle
2		=VXA01ES7565				
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						

Figure 5.32 Export to Microsoft Excel from BRFplus

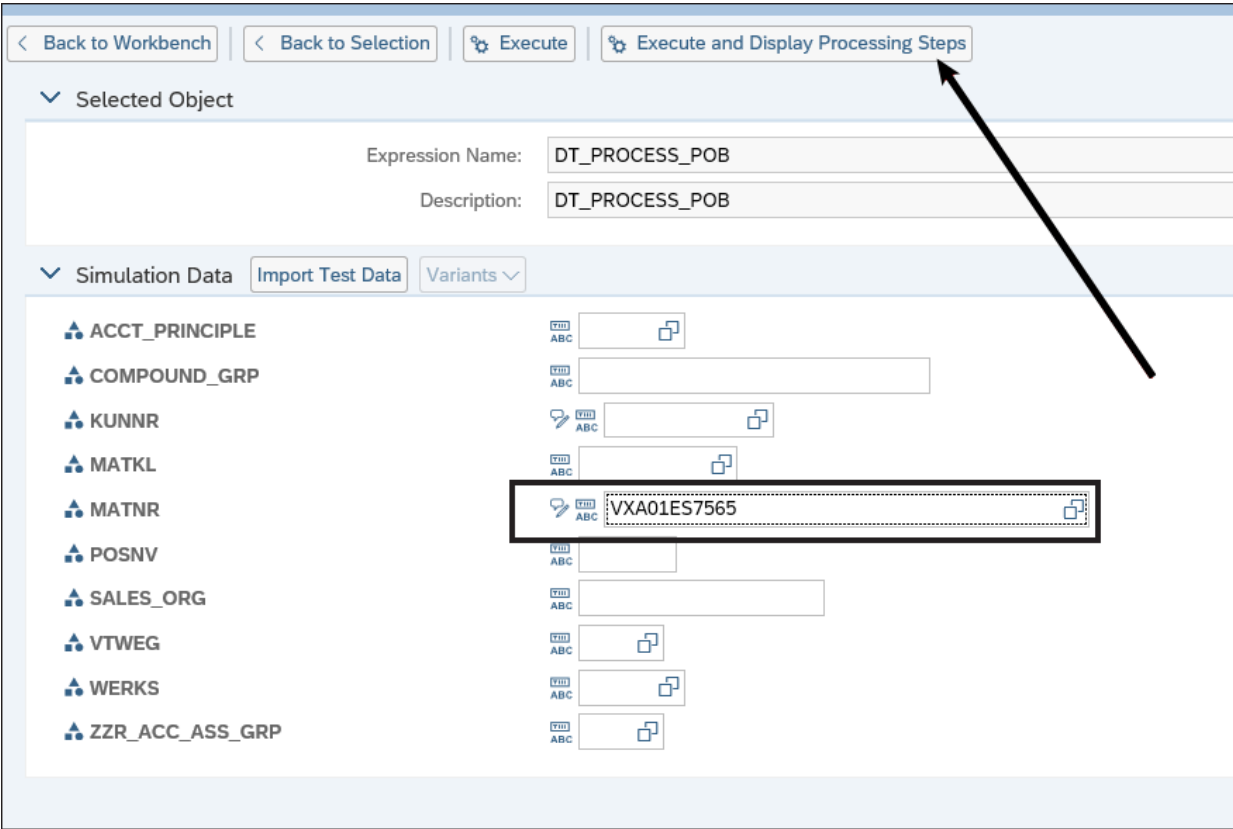


Figure 5.33 Execution of Simulation

POB_NAME:	<input type="text" value="ANNUAL LICENSE"/>
POB_TYPE:	<input type="text" value="TIME_CREAT"/> Time Based POB - Start Date Available on Creation
RESIDUAL_POB:	<input type="text"/>
FULLFILL_TYPE:	<input type="text"/>
EVENT_TYPE:	<input type="text"/>
DEFERRAL_METHOD:	<input type="text"/>
DURATION:	<input type="text" value="0"/>
START_DATE:	<input type="text" value="01.01.0001"/>
END_DATE:	<input type="text" value="01.01.0001"/>
PREVENT_ALLOC:	<input type="text"/>
START_DATE_TYPE:	<input type="text"/>
STATUS:	<input type="text"/>
REVIEW_REASON:	<input type="text"/>
REVIEW_DATE:	<input type="text" value="01.01.0001"/>
PERCENTAGE:	<input type="text" value="0"/>
INVOICE_EFFECT_TYPE:	<input type="text"/> Changing the Price When a Certain Criterion Is Met
DISTINCT_FULFILL:	<input type="text"/> System Default(Compound/Non-Distinct/PoC: Not Unit-distinct)

Figure 5.34 Results of Simulation

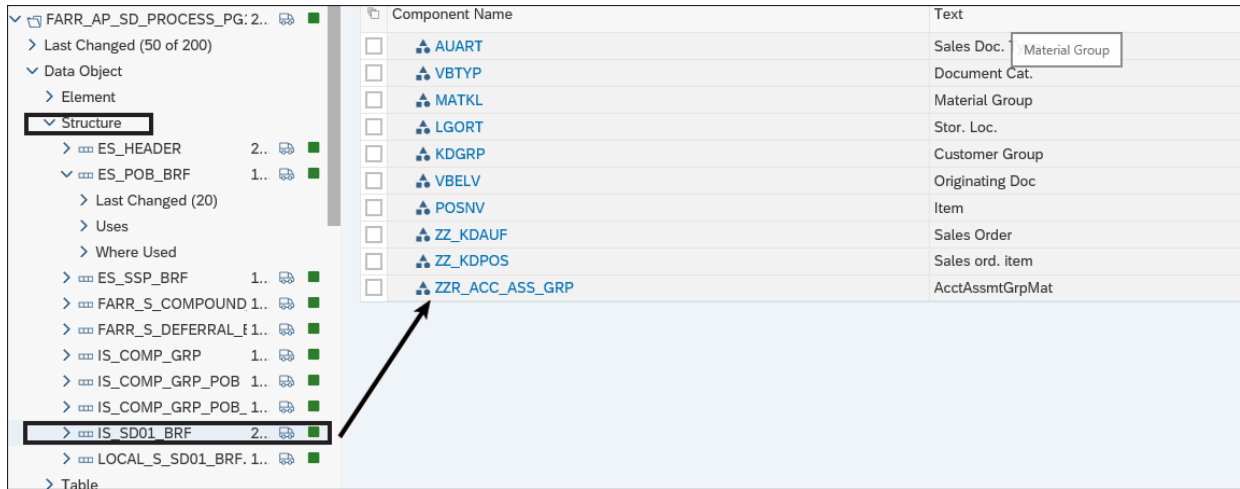


Figure 5.35 Adding a Field to the Structure

Table Settings	
	COMPOUND_GRP

Figure 5.36 Table Settings for BRFplus

Result Data Object: ES_POB_BRF

Table Check Settings

Overlap Check Settings:

Completeness Check Settings:

List of Columns

Condition Columns

	Column Name	Text	Mandatory Input	Column Accessibility
<input type="radio"/>	<input type="checkbox"/> KUNNR	KUNNR	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> MATNR	MATNR	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> SALES_ORG	SALES_ORG	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> VTWEG	VTWEG	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> COMPOUND_GRP	COMPOUND_GRP	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> ACCT_PRINCIPLE	ACCT_PRINCIPLE	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> WERKS	WERKS	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> MATKL	MATKL	<input type="checkbox"/>	Full Access (Changes Allowe
<input type="radio"/>	<input type="checkbox"/> POSNV	POSNV	<input type="checkbox"/>	Full Access (Changes Allowe
<input checked="" type="radio"/>	<input type="checkbox"/> ZZR_ACC_ASS_GRP	ZZR_ACC_ASS_GRP	<input type="checkbox"/>	Full Access (Changes Allowe

Result Columns

Figure 5.37 New Field as Selection Criterion

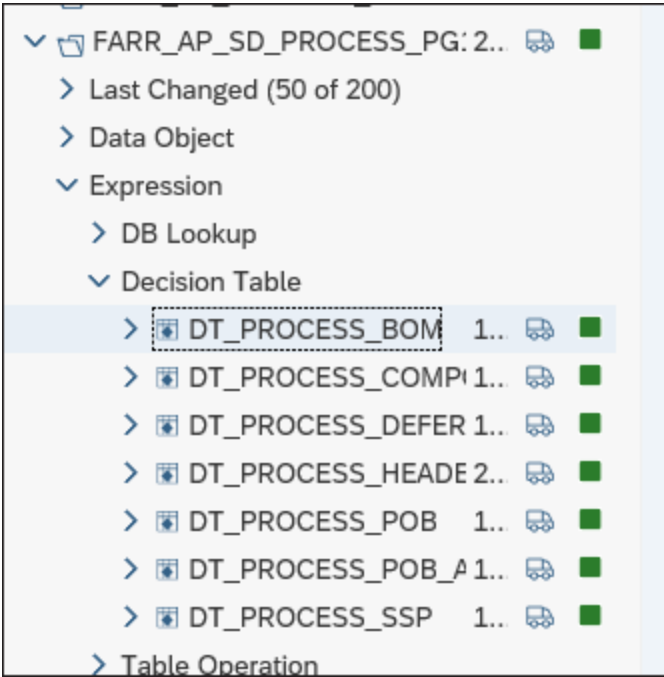


Figure 5.38 Decision Tables Used in Contract and POB Creation

Fulfillment Data

Fulfillment Type: T Time-Based ▾

Event Type: ▾

Start Date Type: 1 Available on Crea... ▾

Duration:

Deferral Method: 1

Figure 5.39 Time-Based POB Settings

Def. Meth. ▲	Description
1	Linear Distribution, Day-Specific, 365/366 Basis
2	Linear Distribution, Day-Specific, 360 Basis
3	Linear Distribution, Day-Specific, 360 Basis, with Rounding Adjustment
4	Linear Distribution, Day-Specific, 365/366 Basis, with Rounding in the Last Period
F	Recognition in First Period
L	Recognition in Last Period
S	Linear Distribution, Period-Specific

Figure 5.40 Deferral Methods Available

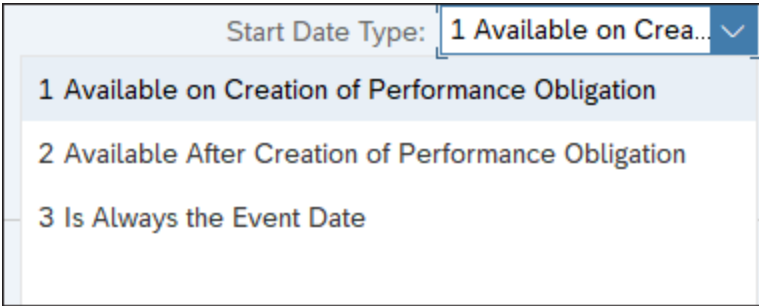


Figure 5.41 Start Date Options When Creating Time-Based POB

Net value: 168.000,00 AUD

Billing plan

BillingPlanType: Z6 PG: Periodic Billing Soft Arch 50 Monthly on First of Month InAdvance:

Start date: 01.12.2022 13 Contract Start (Contract header) Dates from: 01.12.2022

End date: 30.11.2023 14 Contract Ende (Contract header) Dates until: 30.11.2023 09

Horizon: 30.04.2024 52 Horizon 1 Year Corr: Cal-Id:

Dates

Figure 5.42 Billing Plan Dates

Items (3) Standard <input type="button" value="v"/>		<input type="text" value="Search"/>							
Performance Obligation	Contractual Price	Standalone Selling Price	Allocated Amount	Allocation Effect	Freeze Status	Invoiced Amount	Quantity Fulfillment Progress	Effective Quantity	Fulfillment Type
203642 REAGENT (RR)	0,00 EUR	250.000,00 EUR	360.000,00 EUR	360.000,00 EUR		0,00 EUR	<input type="text" value="0%"/>	10,000 PC	E (Event-Based)
203643 SERVICE (BILLING)	120.000,00 EUR	0,00 EUR	0,00 EUR	-120.000,00 EUR		10.000,00 EUR	<input type="text" value="8,33%"/>	360,000 DAY	T (Time-Based)
203644 SERVICE (BILLING)	240.000,00 EUR	0,00 EUR	0,00 EUR	-240.000,00 EUR		20.000,00 EUR	<input type="text" value="8,33%"/>	360,000 DAY	T (Time-Based)

Figure 5.43 Revenue in a Contract Based on Time

Revenue Schedule													
Summary													
Recognized Revenue:		0,00 EUR											
Posted Revenue:		0,00 EUR											
Planned Revenue:		0,00 EUR											
Unscheduled Revenue:		360.000,00 EUR											
Total Revenue:		360.000,00 EUR											
Fulfilled Progress:		0 %											
View: [Standard View] [Display Fulfillment Details] [Display Fulfill. Details for Non-Distinct Perf. Oblig.] [Manually Fulfill Performance Obligations] [Change Spreading] [Notes] [Attachments] [Print View]													
Status	Performanc...	Accounting Pe...	Performance...	Allocated A...	Effective Quan...	Quantity	Fulfillment...	Cumulative Fu...	Revenue	Unsuspend...	Posted Revenue	Posting Price	Invoiced A...
<input type="checkbox"/>	▲ 203642	Unscheduled	REAGENT (RR)	360.000,00	10	10	100,00	100,00	360.000,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	◆ 203643	2023001	SERVICE (BILLI...	0,00	360	30	8,33	8,33	0,00	0,00	0,00	0,00	10.000,00
<input type="checkbox"/>	▲ 203643	2023002	SERVICE (BILLI...	0,00	360	30	8,34	16,67	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023003	SERVICE (BILLI...	0,00	360	30	8,33	25,00	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023004	SERVICE (BILLI...	0,00	360	30	8,33	33,33	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023005	SERVICE (BILLI...	0,00	360	30	8,34	41,67	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023006	SERVICE (BILLI...	0,00	360	30	8,33	50,00	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023007	SERVICE (BILLI...	0,00	360	30	8,33	58,33	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023008	SERVICE (BILLI...	0,00	360	30	8,34	66,67	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023009	SERVICE (BILLI...	0,00	360	30	8,33	75,00	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023010	SERVICE (BILLI...	0,00	360	30	8,33	83,33	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023011	SERVICE (BILLI...	0,00	360	30	8,34	91,67	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	▲ 203643	2023012	SERVICE (BILLI...	0,00	360	30	8,33	100,00	0,00	0,00	0,00	0,00	0,00
<input type="checkbox"/>	◆ 203644	2023001	SERVICE (BILLI...	0,00	360	30	8,33	8,33	0,00	0,00	0,00	0,00	20.000,00
<input type="checkbox"/>	▲ 203644	2023002	SERVICE (BILLI...	0,00	360	30	8,34	16,67	0,00	0,00	0,00	0,00	0,00

Figure 5.44 Revenue Schedule

Reconcl. Key	POB	CnTy	Defer. Cal	Contract	Event Type	FulfilType	CoCo	AccP	Crcy	Unit	Src A/c No	Target A/c	Stat	Condition	Special Proj	Sys	Orig. Amt.	Delta Amt.	Delta Qty	Effect.Val	Effect Qty	Catch Amt.	Post.Am
<input type="checkbox"/> 20230010000101	203644	ZALL	MA	300313228	T		3003	IFRS	EUR	DAY	4111111202	1121211100	P	D			0,00	240.000,00-	360	240.000,00-	360	0,00	20.000,00
<input type="checkbox"/> 20230010000101	203644	ZP02	MA	300313228	T		3003	IFRS	EUR	DAY	4111111302	1121211100	P	P			240.000,00	240.000,00	360	240.000,00	360	0,00	20.000,00
<input type="checkbox"/> 20230020000101	203644	ZALL	MA	300313228	T		3003	IFRS	EUR	DAY	4111111202	1121211100	P	D			0,00	0,00	0	240.000,00-	360	0,00	20.000,00
<input type="checkbox"/> 20230020000101	203644	ZP02	MA	300313228	T		3003	IFRS	EUR	DAY	4111111302	1121211100	P	P			240.000,00	0,00	0	240.000,00	360	0,00	20.000,00
<input type="checkbox"/> 20230030000101	203644	ZALL	MA	300313228	T		3003	IFRS	EUR	DAY	4111111202	1121211100	P	D			0,00	0,00	0	240.000,00-	360	0,00	20.000,00
<input type="checkbox"/> 20230030000101	203644	ZP02	MA	300313228	T		3003	IFRS	EUR	DAY	4111111302	1121211100	P	P			240.000,00	0,00	0	240.000,00	360	0,00	20.000,00
<input type="checkbox"/> 20230040000101	203644	ZALL	MA	300313228	T		3003	IFRS	EUR	DAY	4111111202	1121211100	P	D			0,00	0,00	0	240.000,00-	360	0,00	20.000,00

Figure 5.45 Table FARR_D_DEFITEM

SAP Dictionary: Display Domain

Domain: Active

Short Description:

Properties Definition Value Range

Single Vals

	I	Fixed	Short Description
<input type="checkbox"/>		MA	Main
<input type="checkbox"/>		RR	Rights of Return
<input type="checkbox"/>		RE	Rights of Return Expire

Figure 5.46 Deferral Categories

Deferral Category	Short Description
MA	Main
RR	Rights of Return
RE	Rights of Return Expire
TV	Time Value of Money

Figure 5.47 Category Descriptions

<input type="checkbox"/>	<u>FULLTYPE</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>FARR_FULFILL_TYPE</u>	CHAR	1	0	0 Fulfillment Type
<input type="checkbox"/>	<u>COMPANY_CODE</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>BUKRS</u>	CHAR	4	0	0 Company Code
<input type="checkbox"/>	<u>ACCT_PRINCIPLE</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>ACCOUNTING_PRINCIP</u>	CHAR	4	0	0 Accounting Principle
<input type="checkbox"/>	<u>AMOUNT_CURK</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>WAERS</u>	CUKY	5	0	0 Currency Key
<input type="checkbox"/>	<u>QUANTITY_UNIT</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>FARR_QUANTITY_UNIT</u>	UNIT	3	0	0 Unit of Measure
<input type="checkbox"/>	<u>SOURCE_ACCT</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>FARR_SOURCE_ACCT</u>	CHAR	10	0	0 Source Account Number
<input type="checkbox"/>	<u>TARGET_ACCT</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>FARR_TARGET_ACCT</u>	CHAR	10	0	0 Target Account Number
<input type="checkbox"/>	<u>STATISTIC</u>	<input type="checkbox"/>	<input type="checkbox"/>	<u>KSTAT</u>	CHAR	1	0	0 Condition is used for statistics

Figure 5.48 Fulfillment Types

<input type="checkbox"/> DOC_AMT_DELTA	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_DOC_A_CURR	23	2	0 Delta Amount of a Cond. for a POB in a Posting Period
<input type="checkbox"/> DOC_QTY_DELTA	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_DOC_Q_QUAN	18	6	0 Delta Quantity for a Perf. Obligation in a Posting Period
<input type="checkbox"/> DOC_AMT_CUMULATE	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_DOC_A_CURR	23	2	0 Effective Value of a Condition Type for a Perf. Obligation
<input type="checkbox"/> DOC_QTY_CUMULATE	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_DOC_Q_QUAN	18	6	0 Effective Quantity of a Performance Obligation
<input type="checkbox"/> REV_AMT_CATCHUP	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_REV_A_CURR	23	2	0 Amount of retrospective Revenue Catchup
<input type="checkbox"/> REV_AMT_DELTA	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_REV_A_CURR	23	2	0 Amount for Posting
<input type="checkbox"/> REV_QTY_DELTA	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_REV_Q_QUAN	18	6	0 Quantity for Posting
<input type="checkbox"/> REV_QTY_DIFF_DELTA	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_REV_Q_QUAN	18	6	0 Revenue Quantity Difference
<input type="checkbox"/> REV_QTY_ND_DELTA	<input type="checkbox"/>	<input type="checkbox"/> FARR_DEFITEM_REV_Q_QUAN	18	6	0 Revenue Quantity (Non-distinct)

Figure 5.49 Revenue Amount

```
CL_FARR_FULFILLMENT_MGMT===== / CL_FARR_FULFILLMENT_MGMT===== / 118
METHOD / GENERATE_FULFILL_PER_POB (CL_FARR_FULFILLMENT_MGMT)
Desktop 1 Desktop 2 Desktop 3 Standard Structures Tables Objects Detail Data Explorer
108     ENDIF.
109
110     IF mv_initial_load IS NOT INITIAL.
111         ls_fulfill-recon_key      = mv_initial_load_recon_key.
112     ENDIF.
113
114     "Fulfill Type: Time-based or ROR case
115     IF lv_fulfill_tm_flag = abap_true
116     OR ls_fulfill-deferral_cat = if_farrc_contr_mgmt=>co_deferral_cat
117     TRY .
118         CALL METHOD generate_tm_fulfill
119         EXPORTING
120             is_fulfill      = ls_fulfill
121             iv_group_guid   = iv_group_guid
122             iv_leading_flag = lv_leading_pob
123         IMPORTING
124             et_fulfill      = lt_fulfill_duration.
125         CATCH cx_farr_message INTO lx_farr_message.
126             ev_error = abap_true.
127             CLEAR et_fulfill.
128             EXIT.
129     ENDTRY.
130     LOOP AT lt_fulfill duration ASSIGNING <ls_fulfill>
```

Figure 5.50 Calculation of REV_AMT_DELTA

Fulfillment Data

Fulfillment Type:

Event Type:

- AD Acceptance Date
- CC Contract Acquisition Cost
- CI Customer Invoice
- CS Consumption
- GI Goods Issue
- IB Intercompany Billing Process
- MA Manual Fulfillment
- PD Proof of Delivery
- PI Purchase Invoice - Drop Shipping
- RO Contract Release Order - Call Off

Figure 5.51 Event Types for POB Fulfillments

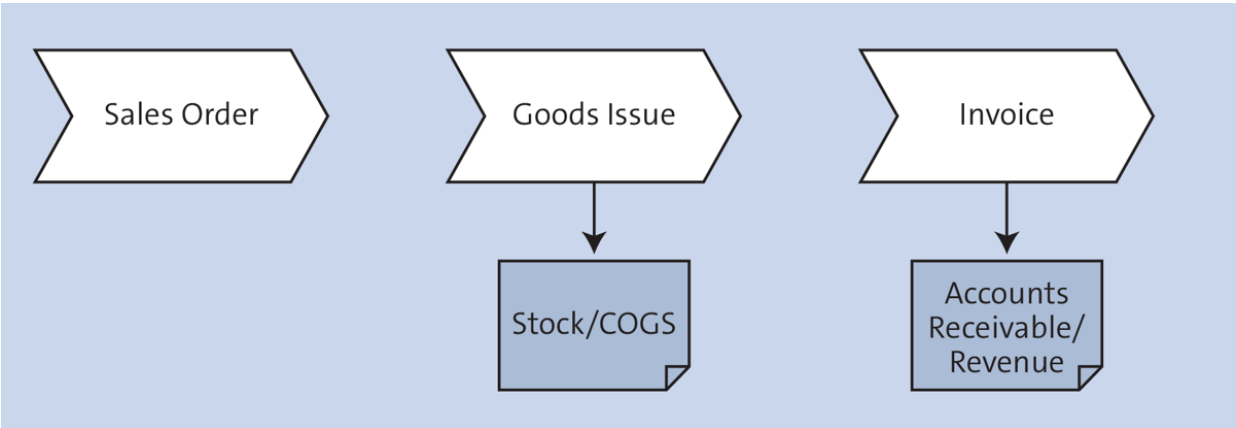


Figure 5.52 Sales and Distribution Process

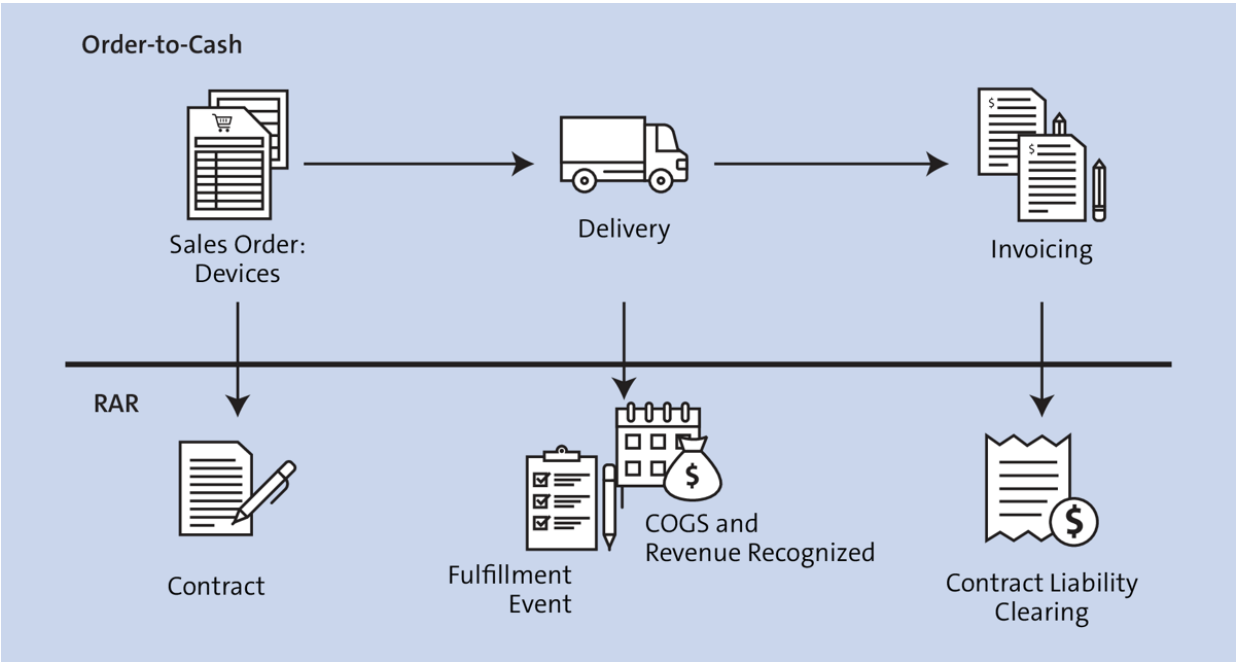





Figure 5.53 Goods Issue as Fulfillment Event

Rental_reagent Contr: Net Value: EUR

Sold-To Party: 

Ship-To Party:

Cust. Reference: Cust. Ref. Date:  

Sales **Item Overview** Item detail Ordering party Procurement Shipping Reason for rejection

Description:

Contract Start: Contract End:

Billing Block: Pricing Date:










Order Reason:

Sales Area: / /

Master Contract:

Shp.Cond.:

Business Area:

All Items

Item	Material	Target Quantity	UoM	Item Description
<input type="checkbox"/>	10 CD994563	1.200	PC	

Figure 5.54 Sales Order Created

Main Item (1)		Condition Item (2)													
ItemStatus	Hist Exist	Error	Ex.His Ex	Send.Com...	SourceS...	Event Ty...	SrcitmTy...	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Ty...	Reference ID	Customer
<input type="checkbox"/>				SD			SDOI	0041023265000010	341	SD01	41023265	10	SDO	0041023265	2100035025

Figure 5.55 SDOI RAI Created

<input type="checkbox"/>	ItemStatus	Hist	Exist	Error	Send.Comp.	SourceSys.	SrcItem	Type	Source Item ID	Cond.type	P/L Account	TC Amount	Currency
<input type="checkbox"/>				■	SD		SDOI		0041023265000010	ZP01	4111111111	599.760,00	EUR
<input type="checkbox"/>				■	SD		SDOI		0041023265000010	ZSSP		354.288,00	EUR

Figure 5.56 Condition Types Created

■	***** PROCESS *****
■	Start of processing of order item SDOI/41023265000010 for accounting principle IFRS
■	New performance obligation 203662 created for contract 300313233
■	Start of processing of order item SDOI/41023265000010 for accounting principle LG
■	New performance obligation 203663 created for contract 300313234
■	1 items processed (record type Main Item, class SD01)
■	2 items processed (record type Condition Item, class SD01)

Figure 5.57 Processing of RAIs

Search

Standard * ⌵

Company Code: Revenue Accounting Contract: Operational Document: Created by: Created on: Show Archived

41023265

Results List

Show: General View

Performance Obligation Structure Price Allocation Revenue Schedule Perform Contract Combination Quick Combine Reprocess Account Determination Reprocess Contracts Comprehensive View

Revenue Accounting Contr...	Company...	Company Name	Accounting Principle	Customer	Customer Name	No. of Perf. Oblig.	Transaction Price
<input type="checkbox"/> 300313234	3003		LG	21000350...		1	599.760,00
<input type="checkbox"/> 300313233	3003		IFRS	21000350...		1	599.760,00

Figure 5.58 Contracts Created

Performance Obligation Structure Price Allocation Revenue Schedule Contract Attachments (0) More Refresh

Contract Information

Accounting Principle: IFRS Number of Performance Obligations: 1
 Company Code: 3003 Number of Operational Documents: 1
 Allocation Effect Exists: Contract Status: In Process
 Receivable Account: 1121211003 A/R external Local Currency Calc. Method: Actual Exchange Rate Method

Performance Obligations

Add Manual Performance Obligations Manually Fulfill Performance Obligations Edit Change Type Notes Attachments

<input type="checkbox"/>	Start Date	Start Date Type	Duration	Duration Unit	End Date	Deferral Method	Fulfilled Progress	Fulfillment Type	Event Type
<input type="checkbox"/>			0,0				0.00	Event-Based	Goods Issue

Figure 5.59 Details of the Contract

Main Item (1) Condition Item (0)															
ItemStatus	Hist Exist	Error	Ex.His.Ex	Send.Com...	SourceS...	Event Ty...	SrcitmTy...	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Ty...	Ref ID	Custo
<input checked="" type="checkbox"/>				SD		GI	SDFI	49000211982023000001	341	SD02	4900021198	1			

Figure 5.60 Fulfillment Process

Contract Information

Accounting Principle:
 Number of Performance Obligations:

Company Code:
 Number of Operational Documents:

Allocation Effect Exists:
 Contract Status:

Receivable Account: A/R external
 Local Currency Calc. Method:

Performance Obligations

<input type="checkbox"/>	Start Date	Start Date Type	Duration	Duration Unit	End Date	Deferral Method	Fulfilled Progress	Fulfillment Type	Event Type	Currency
<input type="checkbox"/>			0,0				<input type="range" value="20.83"/>	Event-Based	Goods Issue	EUR

Figure 5.61 Contract with Fulfillment

Main Item (1)		Condition Item (2)															
ItemStatus	Hist Exist	Error	Ex.His Ex	Send Com...	SourceS...	Event Ty...	SrcitmTy...	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Ty...	Ref ID	Customer	Pa	
				SD			SDII	9110000762000010	341	SD03	9110000762	10			2100026929		

Figure 5.62 Invoice RAI

003	A/R external	Local Currency Calc. Method: Actual Exchange Rate Method				
Fulfill Performance Obligations ▾ Edit Change Type Notes Attachments						
Effective Remaining Q...	Effective Remaining A...	Unit Distinct	Negative Amount	Performance Obligati...	Fulfilled Quantity	Invoiced Quantity
1.200	599.760,00	Unit-Distinct	<input type="checkbox"/>	YRR001	250	250

Figure 5.63 Contract Search after Invoicing

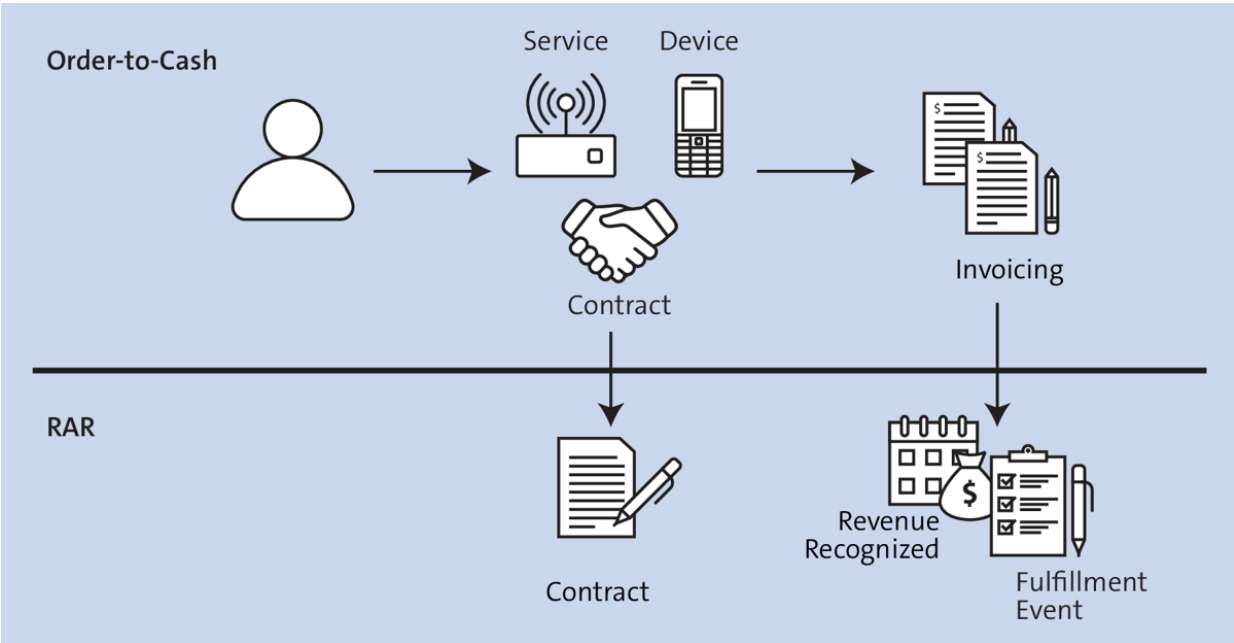


Figure 5.64 Customer Invoice as a Fulfillment Event

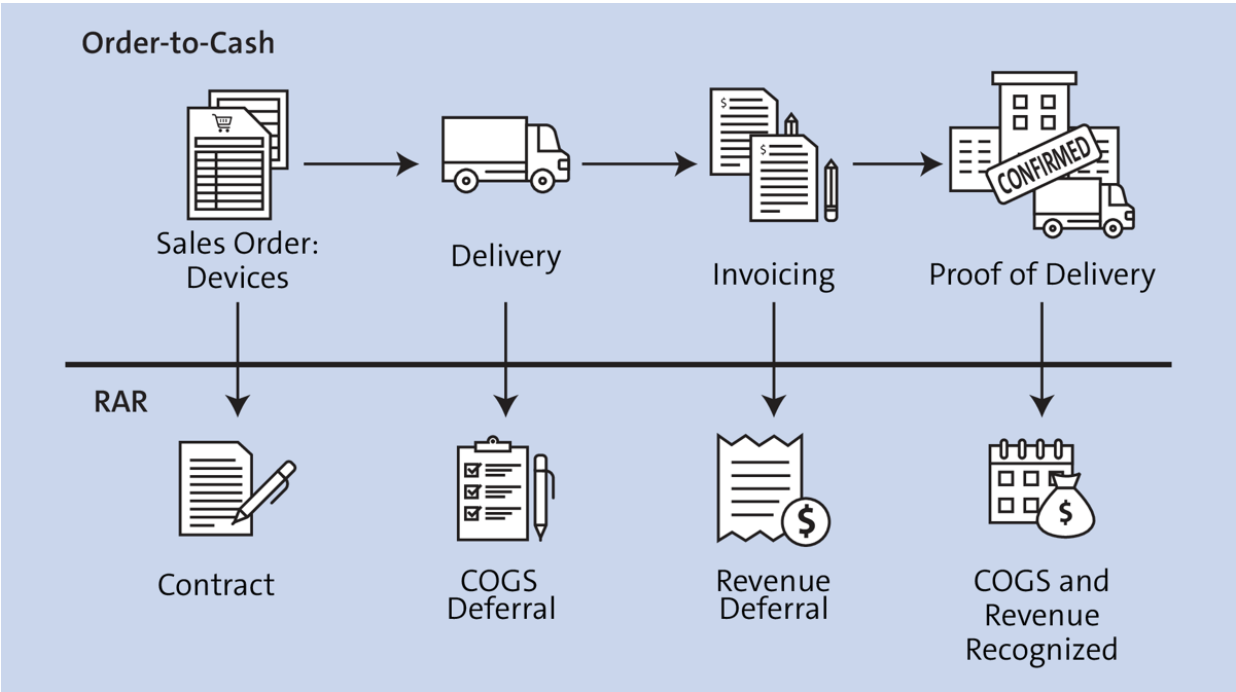


Figure 5.65 Proof of Delivery Process

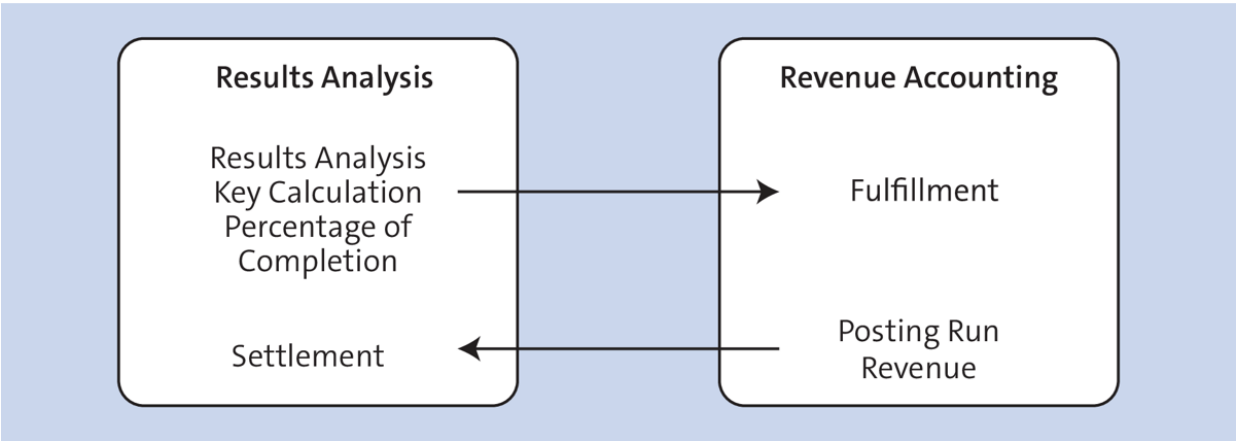


Figure 5.66 Percentage of Completion Method

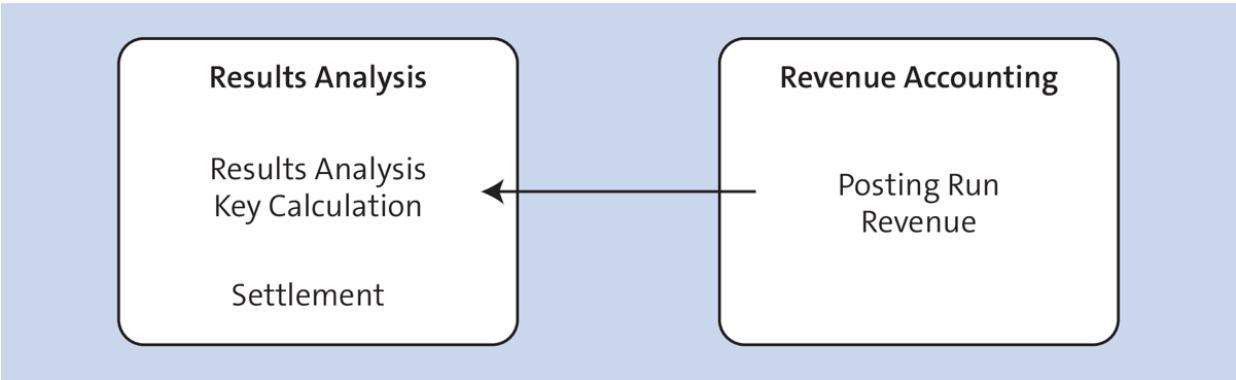


Figure 5.67 Revenue Method

SAP Change View "Simplified Maintenance of Valua

More ▾

CO Area: RA Version: RA Key:

Results Analysis Method:

Results Analysis Method (CO-PC) (1) 17 Entries found

Restrictions

Meth	Designation of Results Analysis Method
01	Revenue-Based Method - With Profit Realization
02	Revenue-Based Method - Without Profit Realization if Actual Revenue < Plan Costs
03	Cost-Based POC Method
04	Quantity-Based Method
05	Quantity-Based POC Method
06	POC Method on Basis of Revenue Planned by Period
07	POC Method on Basis of Project Progress Value Determination
08	Derive Cost of Sales from "Old" Resource-Related Billing of CO Line Items
09	Completed Contract Method
10	Inventory Determination, Without Planned Costs, Without Milestone Billing
11	Inventory Determination, Without Planned Costs, With Milestone Billing
12	Inventory Determination, Reserve for Follow-Up Costs, Without Milestone Billing
13	Inventory Determination "WIP at Actual Costs" for Objects Not Carrying Revenue
14	Derive Cost of Sales from Resource-Related Billing of Dynamic Items
15	Derive Revenue from Resource-Related Billing and Simulation of Dynamic Items

Status Control
 Results Analysis with Status
 Cancel Inventory w/Status
 Cancel Inventory/Reserves with Status

Profit Basis
 Plan Value of Obj. and Dependent Objects
 Sales Order Cost Estimate
 Std Price of Material for Sales Order

Valuation Level
 Valuation at Totals Level

Minimum Values

Figure 5.68 Maintenance of the Results Analysis Key

POB Type: ZPOC

Description: PoC method based POB

General Data

Perf.Obligat.Nam: POC BASED POB

No Cost Recognition

Fulfillment Data

Fulfillment Type: 0 Percentage of Co... ▾

Event Type: ▾

Figure 5.69 Definition of the POB Type

RA Customizing for RR: POC Currency, ACC Principle						
	COAr	CoCd	RAVn	AccP	CC	
<input type="checkbox"/>	JP01	02	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	AU02	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	FR02	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	GB04	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	ID02	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	NL02	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	NL03	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	US19	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	ZA02	0	IFRS	10 Company code currency	▼
<input type="checkbox"/>	PG00	ZA03	0	IFRS	10 Company code currency	▼

Figure 5.70 Customizing Results Analysis: Version Relevancy

RA Customizing: RR-Relevant RA Keys, RA Methods					
	COAr	CoCd	RAVn	RA Key	IntMeth
<input type="checkbox"/>	JP01	02	0	PG0001	1 PoC-Based Integration ∨
<input type="checkbox"/>	JP01	JP02	0	PG0024	1 PoC-Based Integration ∨
<input type="checkbox"/>	JP01	ZA02	0	PG0001	1 PoC-Based Integration ∨
<input type="checkbox"/>	JP01	ZA02	0	PG0024	1 PoC-Based Integration ∨
<input type="checkbox"/>	PG00	AU02	0	PG0001	1 PoC-Based Integration ∨
<input type="checkbox"/>	PG00	AU02	0	PG0023	1 PoC-Based Integration ∨
<input type="checkbox"/>	PG00	AU02	0	PG0024	1 PoC-Based Integration ∨
<input type="checkbox"/>	PG00	C002	0	PG0001	1 PoC-Based Integration ∨
<input type="checkbox"/>	PG00	C002	0	PG0024	1 PoC-Based Integration ∨

Figure 5.71 Customizing Results Analysis: Key Relevancy

<input type="checkbox"/>	ZCI	KEN	C Call-off Order with predecessor	▼
<input type="checkbox"/>	ZCPT	ZCPI	C Call-off Order with predecessor	▼
<input type="checkbox"/>	ZCPT	ZCPT	C Call-off Order with predecessor	▼
<input type="checkbox"/>	ZCR	G2N	M Credit/Debit Memos with referenc...	▼
<input type="checkbox"/>	ZCR	ZG2W	M Credit/Debit Memos with referenc...	▼
<input type="checkbox"/>	ZDR	L2N	M Credit/Debit Memos with referenc...	▼

Figure 5.72 Setup for Event Type RO

Configure Accounting Principle-specific Settings				
AccP	Name of Accounting Principle	Presentat.	LC Calc. Method	Cont. Mod.
<input type="checkbox"/> IFRS	IFRS Accounting	2 Contract Liability/Contract Asset	Fixed Exchange Rate Method	<input checked="" type="checkbox"/>



Figure 5.73 Enabling Contract Change

Performance Obligations

List View Hierarchical View

View: New View Add Manual Performance Obligations Manually Fulfill Performance Obligations Edit Change Type Notes

Start Date	End Date	Performance Obligati...	Performance O...	Contractual Price	Standalone Selli...	Allocated Amount
		18047	Device	0,00	500,00	352,94
24.01.2017	13.02.2017	18048	Service	66,67	66,67	53,32
14.02.2017	23.01.2018	18049	Service	2.266,67	2.266,67	1.927,08

Figure 5.74 RAR Computation after Modification

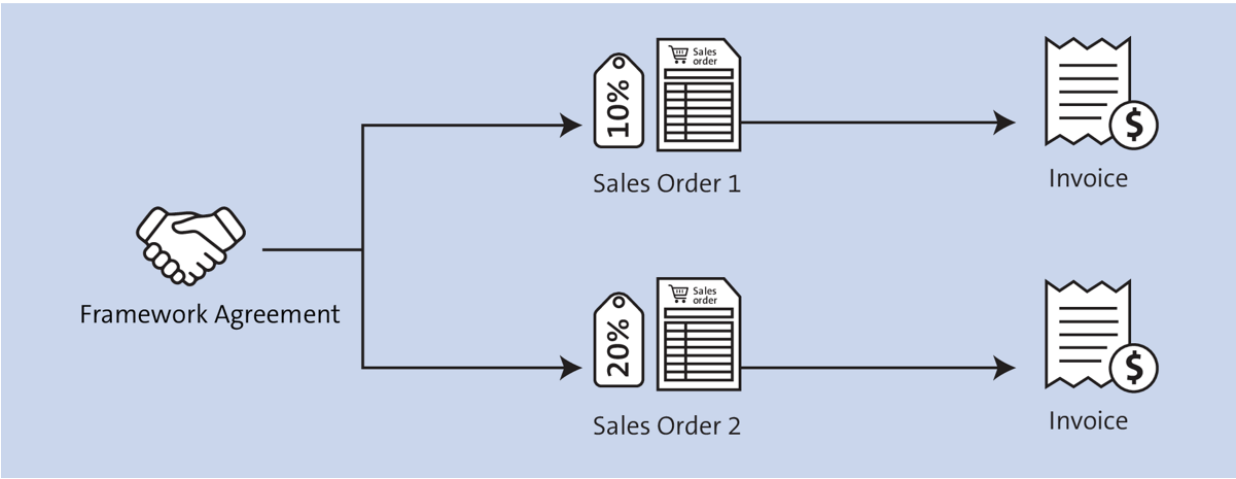


Figure 5.75 Contract Combination Rules

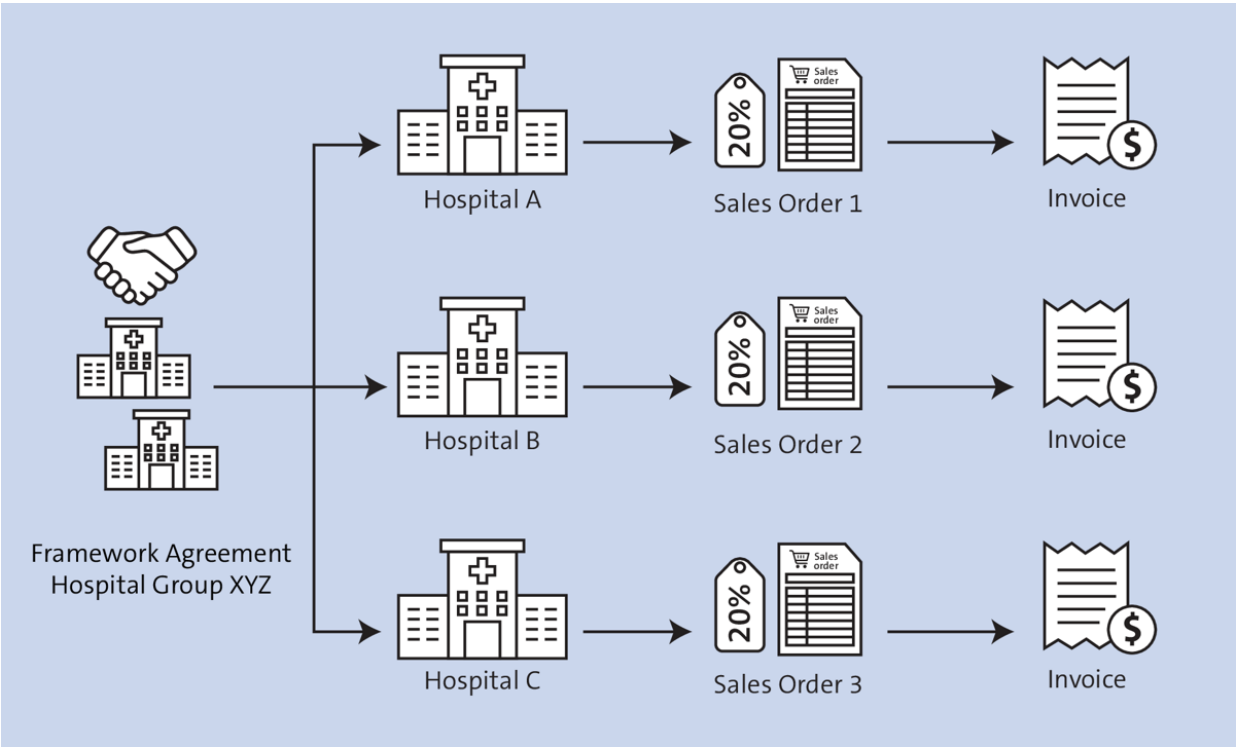


Figure 5.76 Contract Combination Group Level




Rental_reagent Contr:	41019476	Net Value:	24.000,00	EUR
Sold-To Party:	2100035025			
Ship-To Party:	2200013168			
Cust. Reference:	TestXYZZ11	Cust. Ref. Date:		 
Sales Item Overview Item detail Ordering party Procurement Shipping Reason for rejection				
Description:				
Contract Start:	01.01.2023		Contract End:	31.12.2023

Figure 5.77 Combination Rules

Standard * ☺

Company Code: Revenue Accounting Contract: 1 more Operational Document: Created by: Created on: Show Archived Co

Results List

Show: **General View** ▾

<input type="checkbox"/>	Revenue Accounting Contr...	Company...	Company Name	Accounting Principle	Customer	Customer Name	No. of Perf. Oblig.	Transaction Price	Co
<input type="checkbox"/>	300306644			IFRS	21000272...		21	11.779,20	
<input type="checkbox"/>	300300102			IFRS	21000272...		2	0,00	

Figure 5.78 Selection of Contracts

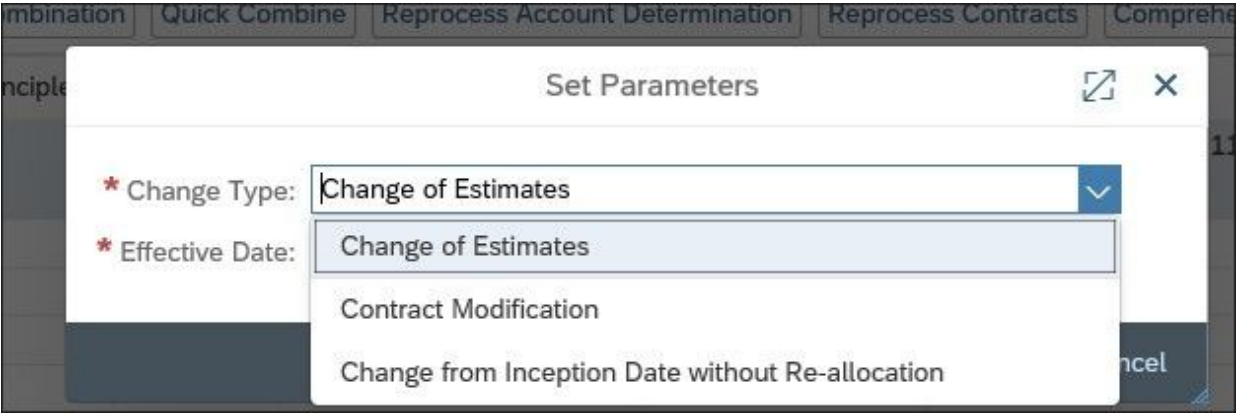


Figure 5.79 Change Type Selection

Work Area: Recent Contracts with

Cut Paste Combine Contracts Remove Contract(s) from Work Area

<input type="checkbox"/>	Contract / Operation...	Contr.Desc./Perf.Oblig....	Leading Perf...	Composition	Standalone Selling Price	Contractual Price from...	Co
<input type="checkbox"/>	Total					11.779,20	
<input type="checkbox"/>	300306644 (Rev	No text available				11.779,20	
<input type="checkbox"/>	41011717					1.405,98	
<input type="checkbox"/>	38813	(REAGENT (RR)		Distinct	79,46	0,00	
<input type="checkbox"/>	38814	(REAGENT (RR)		Distinct	11.753,92	0,00	
<input type="checkbox"/>	38815	(REAGENT (RR)		Distinct	113,20	164,50	
<input type="checkbox"/>	38816	(REAGENT (RR)		Distinct	158,74	186,80	
<input type="checkbox"/>	38817	(REAGENT (RR)		Distinct	298,24	989,10	
<input type="checkbox"/>	38818	(REAGENT (RR)		Distinct	2.858,00	0,00	
<input type="checkbox"/>	38819	(REAGENT (RR)		Distinct	4.220,00	0,00	
<input type="checkbox"/>	38820	(REAGENT (RR)		Distinct	3.019,45	0,00	
<input type="checkbox"/>	38821	(REAGENT (RR)		Distinct	75,72	0,00	
<input type="checkbox"/>	38822	(REAGENT (RR)		Distinct	181,41	0,00	
<input type="checkbox"/>	38823	(REAGENT (RR)		Distinct	27,65	21,86	
<input type="checkbox"/>	38824	(REAGENT (RR)		Distinct	27,65	21,86	
<input type="checkbox"/>	38825	(REAGENT (RR)		Distinct	27,65	21,86	

Figure 5.80 Selection of POBs for Combination

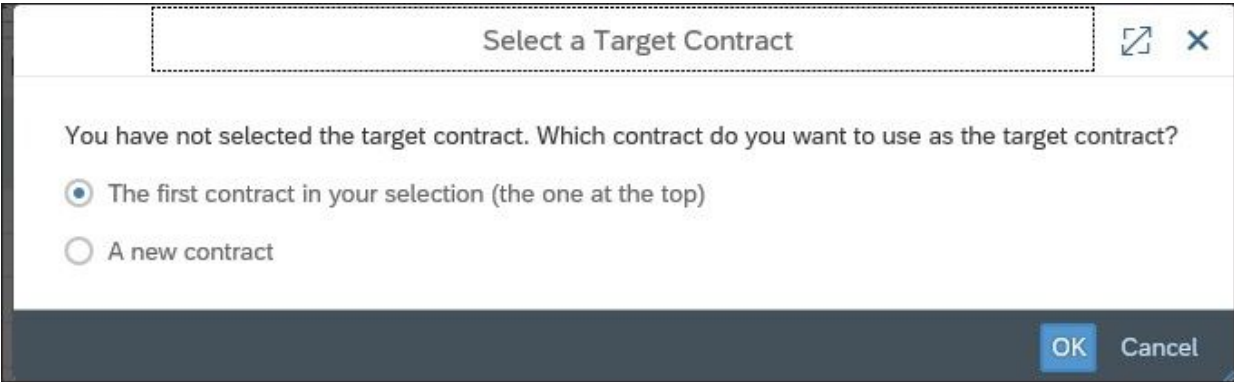


Figure 5.81 Quick Combine

Interface: IF_FARR_RAI2_CONTR_COMB Implemented / Active

Parameters of Method: COMBINE_CONTRACT

Parameter	Type	Pa...	Op...	Typing Method	Associated Type	Default
IV_INITIAL_LOAD	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_INITIAL_LOAD	
IT_RAW_POB	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAW_POB	
IT_MAPPING	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_MAPPING	
ET_COMBINED_CONTRACTS	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_CONTR_MAPPING	

Figure 5.82 Standard Class for Contract Combination

```
Method: IF_FARR_RAI2_CONTR_COMB~COMBINE_CONTRACT active
1  METHOD if_farr_rai2_contr_comb~combine_contract.
2  DATA lts_raw_pob TYPE ty_ts_raw_pob.
3
4  lts_raw_pob = it_raw_pob.
5
6  IF NOT iv_initial_load IS INITIAL.
7      " Process INITIAL Load
8      combine_initial_load( EXPORTING its_raw_pob = lts_raw_pob
9                          IMPORTING et_combined_contracts = et_combined_contracts ).
10
11 ELSE.
12     " Process PRODUCTIVE run
13     combine( EXPORTING its_raw_pob = lts_raw_pob
14            it_mapping = it_mapping
15            IMPORTING et_combined_contracts = et_combined_contracts ).
16
17 ENDIF.
ENDMETHOD.
```

Figure 5.83 Combine Contract Method

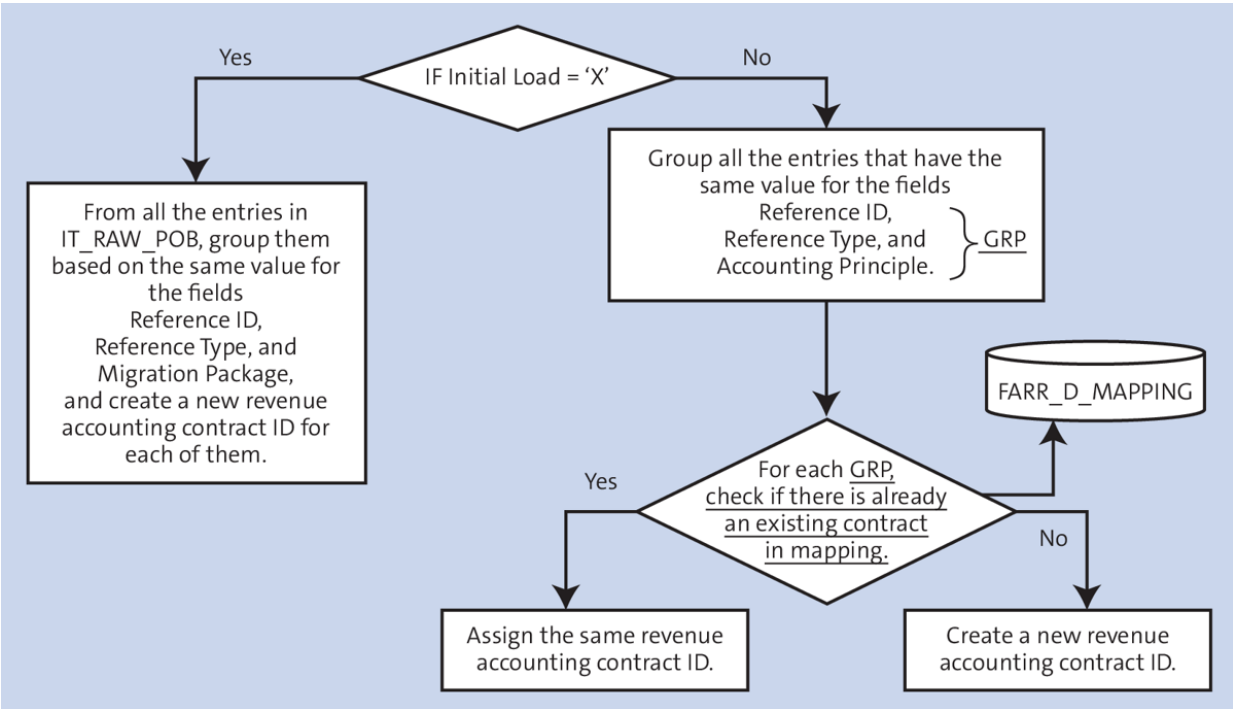


Figure 5.84 Logic for Contract Combination

SAP Class Builder: Display Class CL_FARR_RAI2_BADI_IMPL

Class/Interface: CL_FARR_RAI2_BADI_IMPL Implemented / Active

Properties Interfaces Friends Attributes **Methods** Events Types Aliases

Parameters of Method: COMBINE_CONTRACT

← Methods Exceptions Sourcecode Properties

Parameter	Type	Pa...	Op...	Typing Method	Associated Type	Default Value
IV_INITIAL_LOAD	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_INITIAL_LOAD	
IT_RAW_POB	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_RAW_POB	
IT_MAPPING	Importing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_MAPPING	
ET_COMBINED_CONTRACTS	Exporting	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_TT_CONTR_MAPPING	

Figure 5.85 Method COMBINE_CONTRACT

Object cannot be changed in this system

Repository Catalog

Show: Search Result Search

Search Result C.. T... S...

YFARR_ACC_DETERMINE SY 1..

Last Changed (50 of 13)

Data Object

Expression

Decision Table

FARR_ACCT_DETERM 0..

FARR_ACCT_DETERM 0..

FARR_ACCT_DETERM 0..

FARR_ACCT_DETERM 1..

FARR_ACCT_DETERM 1..

FARR_ACCT_DETERM 1..

FARR_ACCT_DETERM 0..

FARR_ACCT_DETERM 1..

FARR_ACCT_DETERM 0..

FARR_ACCT_DETERM 0..

FARR_ACCT_DETERM 0..

Decision Table: FARR_ACCT_DETERMINE_DT_ASST_IM, Impairment of Assets

Back Edit Check Save Activate Transport Delete More

General

Detail

Export To Excel Context Overview Start Simulation

Table Contents

Find: Next Previous

COMPANY_CODE	ACCT_PRINCIPLE	REF_ACCOUNT	GL_ACCOUNT
<input type="checkbox"/> =3001			4391111001 (Early Termination)
<input type="checkbox"/> =3003			4391111001 (Early Termination)
<input type="checkbox"/> =3004			4391111001 (Early Termination)
<input type="checkbox"/> =3008			4391111001 (Early Termination)
<input type="checkbox"/> =3009			4391111001 (Early Termination)
<input type="checkbox"/> =3010			4391111001 (Early Termination)
<input type="checkbox"/> =3011			4391111001 (Early Termination)
<input type="checkbox"/> =3012			4391111001 (Early Termination)

Figure 5.86 Early Termination Account Determination

eS...	Event Ty...	SrcItmTy...	Source Item ID	Subarea	RevAcc...	Early Term	Header ID	ItemID	Ref. Ty...	Reference ID	Customer	P
		SDOI	0045008064000020	311	<u>SD01</u>	X	<u>45008064</u>	20	SDO	0045008064	2100036595	
		SDOI	0045008064000030	311	<u>SD01</u>	X	<u>45008064</u>	30	SDO	0045008064	2100036595	
		SDOI	0046002934000020	522	<u>SD01</u>	X	<u>46002934</u>	20	SDO	0046002934	2100038369	
		SDOI	0046002934000040	522	<u>SD01</u>	X	<u>46002934</u>	40	SDO	0046002934	2100038369	

Figure 5.87 Transaction FARR_RAI_MON with Terminated Items

Search in Table: Postings

Number of hits:

Runtime: Maximum no. of hits:

Insert Column:

<input type="checkbox"/>	CoCd	AccP	Reconcl. Key	POB	CnTy	Categ...	D/C	Posting GUID	Year	Per...	ΣTC Amou...	Crcy L
<input type="checkbox"/>						AI					▪ 2.786,17-	EUR
<input type="checkbox"/>						CA					▪ 0,05-	EUR
<input type="checkbox"/>						CL					▪ 895,61	EUR
<input type="checkbox"/>						IC					▪ 2.657,00	EUR
<input type="checkbox"/>						RA					▪ 0,00	EUR
<input type="checkbox"/>						RV					▪ 766,39-	EUR
<input type="checkbox"/>											▪ 0,00	EUR

Figure 5.88 Termination Postings

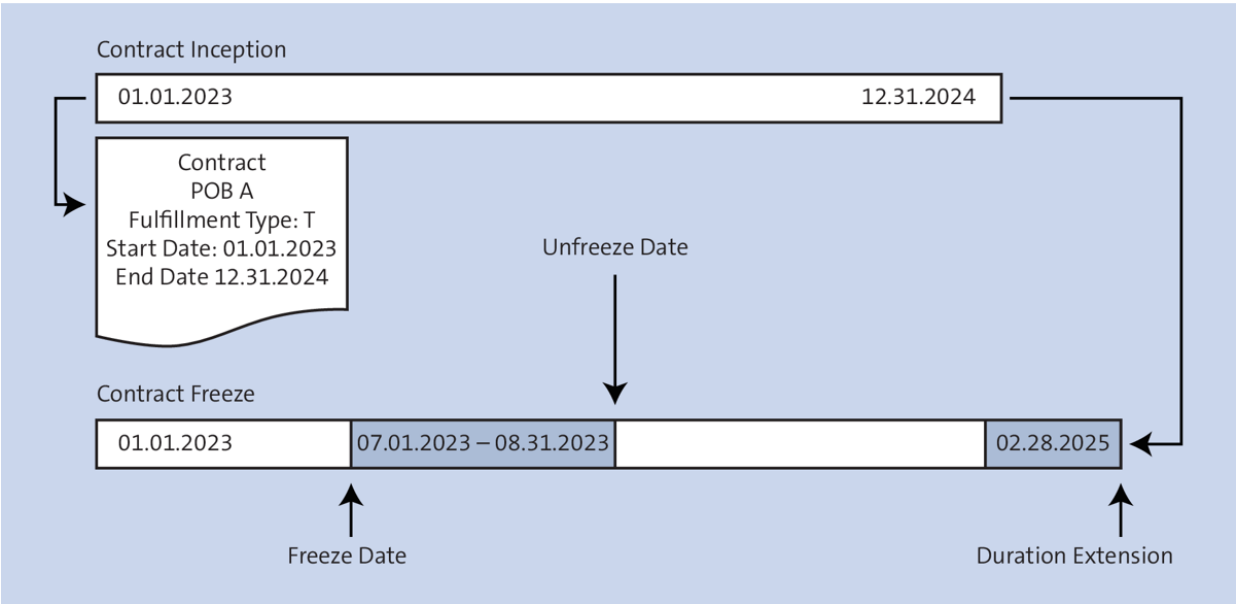


Figure 5.89 Contract Freeze Process with Extension

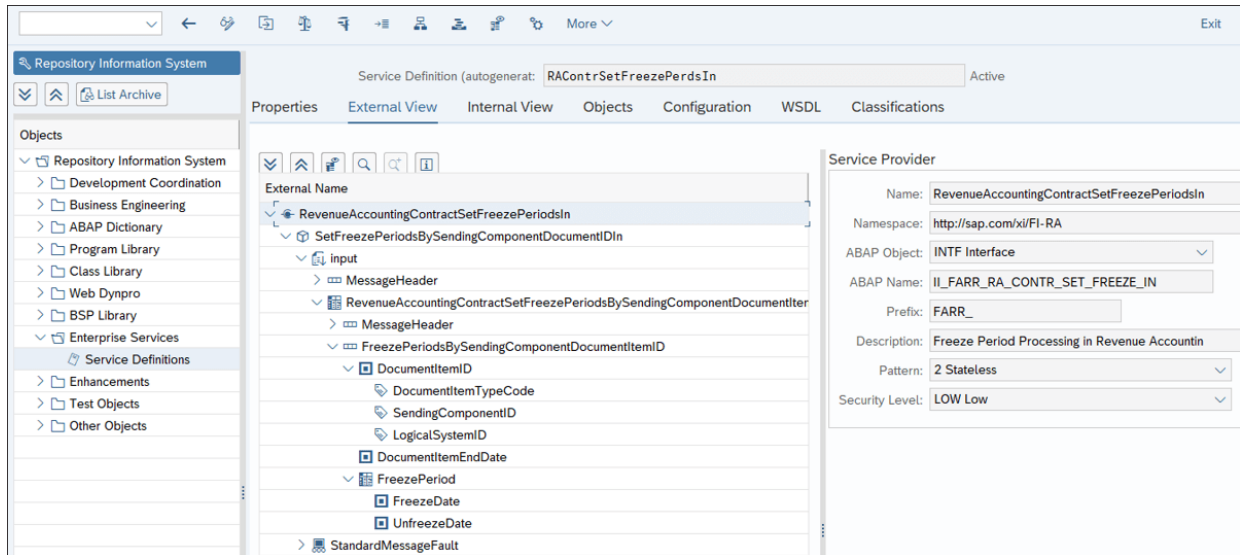


Figure 5.90 Transaction SOAMANAGER for the Freeze Service

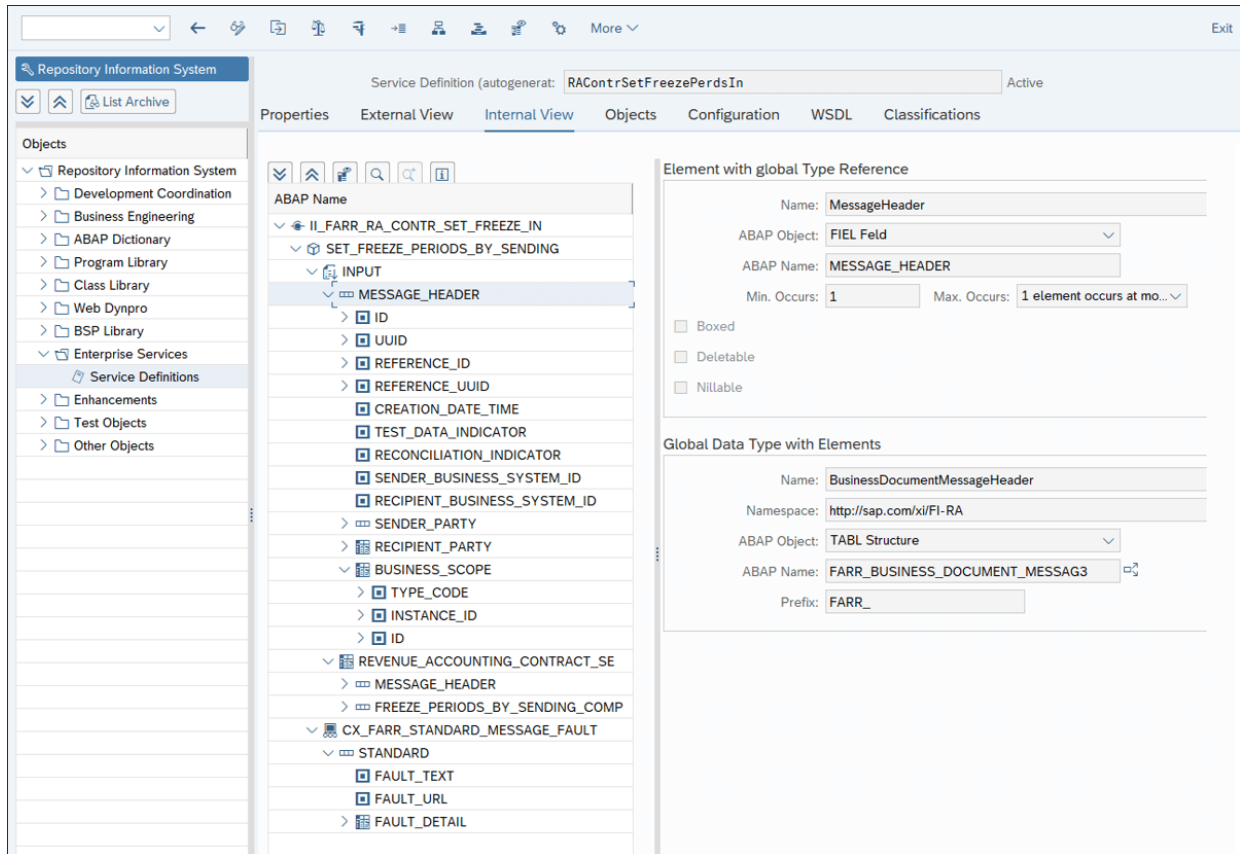


Figure 5.91 Services Available

Performance Obligations

Add Manual Performance Obligations Edit Change Type Notes Attachments

<input type="checkbox"/>	Performance Oblig...	Performance Obligation...	Leading/Linked	Leading P...	Contractual Price	Allocated Amount	Allocation Effect
<input type="checkbox"/>	72347	REAGENT (RR)			364.732,55	0,00	364.732,55-
<input type="checkbox"/>	72348	REAGENT (RR)			145.888,40	0,00	145.888,40-
<input type="checkbox"/>	72349	REAGENT (RR)			244.145,65	0,00	244.145,65-
<input type="checkbox"/>	72350	REAGENT (RR)			0,00	2,44	2,44
<input type="checkbox"/>	72351	REAGENT (RR)			0,00	2.564,46	2.564,46
<input type="checkbox"/>	72352	REAGENT (RR)			0,00	223,65	223,65
<input type="checkbox"/>	72353	REAGENT (RR)			186,80	3,03	183,77-
<input type="checkbox"/>	72354	REAGENT (RR)			499,80	3,40	496,40-
<input type="checkbox"/>	72355	REAGENT (RR)			0,00	1.045,65	1.045,65
<input type="checkbox"/>	72356	REAGENT (RR)			0,00	696,94	696,94
<input type="checkbox"/>	72357	REAGENT (RR)			0,00	261,33	261,33
<input type="checkbox"/>	72358	REAGENT (RR)			569,65	1,10	568,55-

Figure 5.92 Allocation Results

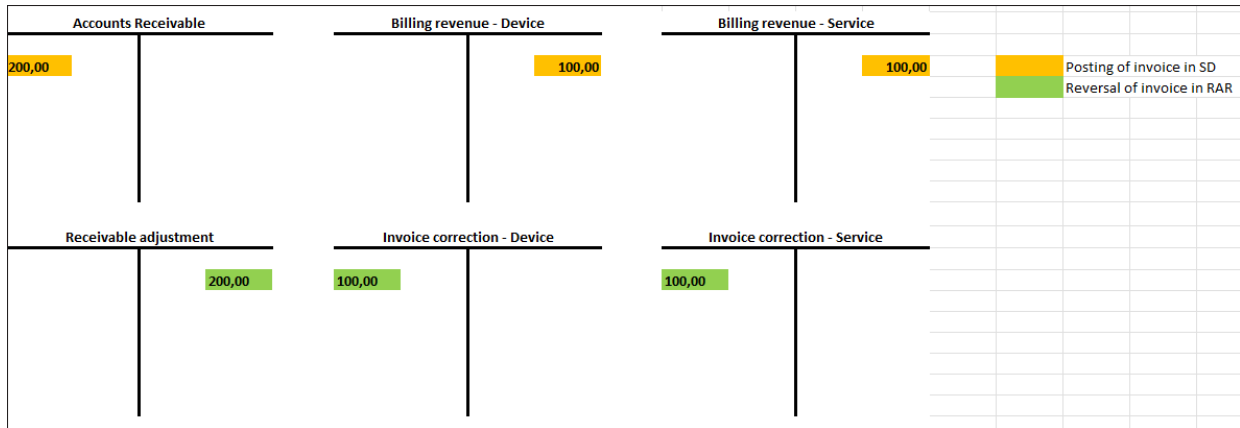


Figure 6.1 Postings after Invoice Reversal

Accounts Receivable	Billing revenue - Device	Billing revenue - Service	
200,00	100,00	100,00	Posting of invoice in SD
			Reversal of invoice in RAR
			Revenue transfer
Receivable adjustment	Invoice correction - Device	Invoice correction - Service	
458.82	100,00	100,00	
200,00	100,00	100,00	
	Revenue adjustment - Device	Revenue adjustment - Service	
	282.35	23.53	

Figure 6.2 Revenue Transfer Calculation

Accounts Receivable		Billing revenue - Device		Billing revenue - Service				
200,00			100,00		100,00			Posting of invoice in SD
								Reversal of invoice in RAR
								Revenue transfer
								Contract liability calculation
Receivable adjustment		Invoice correction - Device		Invoice correction - Service				
	200,00	100,00		100,00				
458.82			100,00		100,00			
	258.82							
Contract Asset		Revenue adjustment - Device		Revenue adjustment - Service				
258.82			282.35	23.53				

Figure 6.3 Contract Liability Calculation

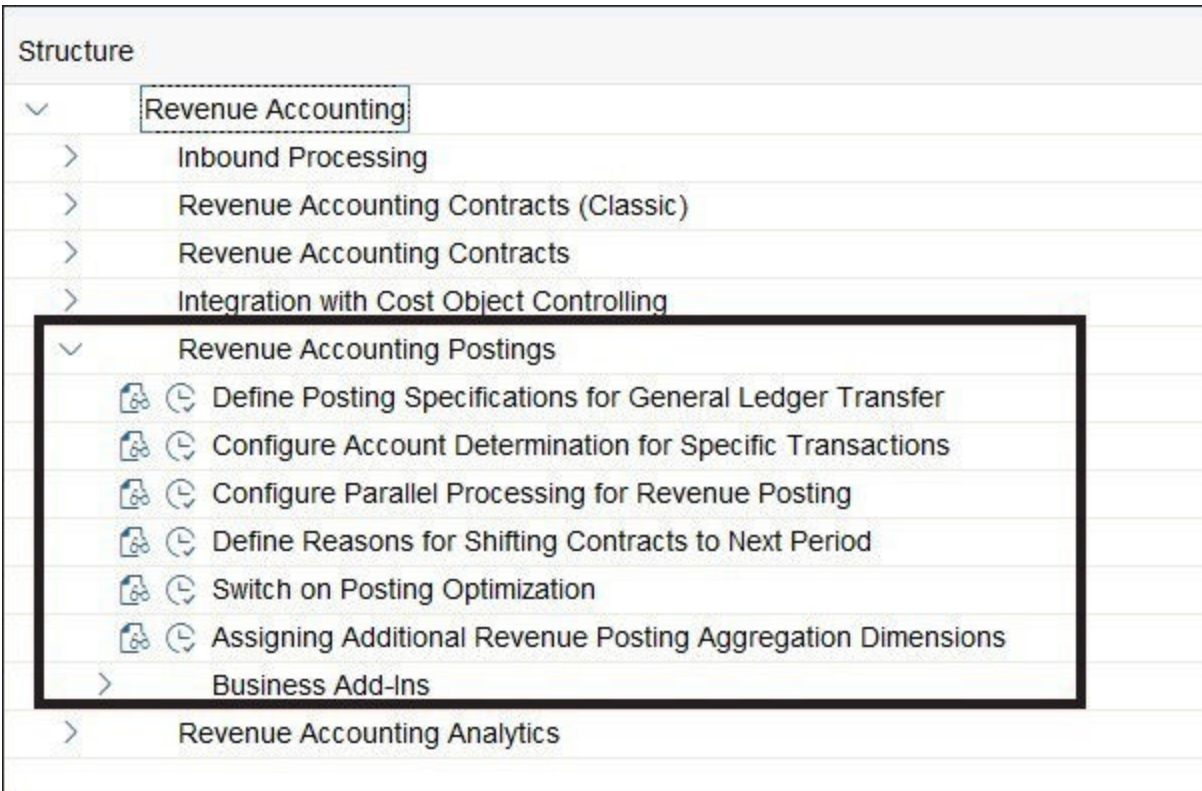


Figure 6.4 Setup of Posting Specification

Define Posting Specification for General Ledger Transfer									
	CoCd	C...	D...	Type	G/LAccount	Segment	Profit Center	BusA	Functional Area
<input type="checkbox"/>	3001	<input type="checkbox"/>	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3003	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3004	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3008	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3009	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3010	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3011	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3012	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3013	50	40	RR	9999911000		9999900001		
<input type="checkbox"/>	3014	50	40	RR	9999911000		9999900001		

Figure 6.5 Details of the Posting Specification

Reasons for Shifting Contracts to Next Period		
	Reason	Description
<input type="checkbox"/>	1	Posting Error
<input type="checkbox"/>		
<input type="checkbox"/>		

Figure 6.6 Shifting Reasons

The image shows a screenshot of a software interface. At the top, there is a search bar and several icons: a four-pane window icon, a grid icon, a document icon, and a 'More' dropdown menu. Below this is a header row for a table titled "Switch on Posting Optimization Customizing table". The table has four columns: "CoCd", "AccP", "Post. Opt", and "Agg. by D/C ind.". The table body contains several empty rows, suggesting it is a template or a table with no data currently loaded.

CoCd	AccP	Post. Opt	Agg. by D/C ind.

Figure 6.7 Switch on Posting Optimization Customizing Table

The image shows a software interface window. At the top, there is a header bar containing a dropdown menu on the left and four icons (a grid, a grid with arrows, a grid with a plus sign, and a document with a plus sign) followed by the text 'More'. Below the header bar, the title 'Transfer Selected Fields to FI-CO' is displayed. Underneath the title is a table with a single column labeled 'Field Name'. The table has 15 rows, with the first row containing the header text and the remaining 14 rows being empty.

Field Name

Figure 6.8 Assign Additional Dimensions

Structure: INCL_EEW_FARR_REP Active
Short Description: Enhancement Include Postings

Attributes Components Input Help/Check Currency/quantity fields

1 / 6

Component	Typing Method	Component Type	Data Type	Length	Decima...	Coordinate	Short Description
<input type="checkbox"/> REP_EEW_DUMMY	Types	▼ DUMMY	CHAR	1	0		0 Dummy function in length 1
<input type="checkbox"/> _APPEND	Types	▼ ZRFIA001		0	0		0 Append structure for adding new fields
<input type="checkbox"/> ZZ_KDAUF	Types	▼ KDAUF	CHAR	10	0		0 Sales Order Number
<input type="checkbox"/> ZZ_KDPOS	Types	▼ KDPOS	NUMC	6	0		0 Item number in Sales Order
<input type="checkbox"/> ZUONR	Types	▼ DZUONR	CHAR	18	0		0 Assignment number
<input type="checkbox"/> SGTXT	Types	▼ SGTXT	CHAR	50	0		0 Item Text

Figure 6.9 Include Custom Dimensions

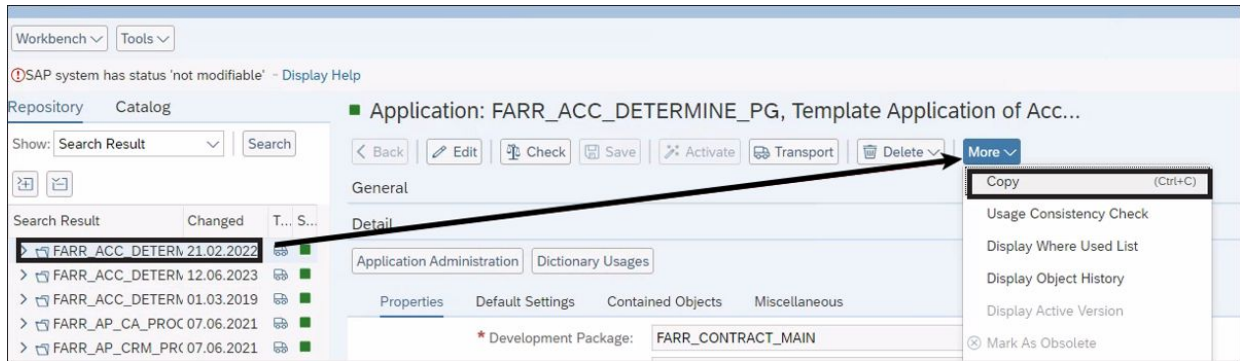


Figure 6.10 Copying FARR_ACC_DETERMINE

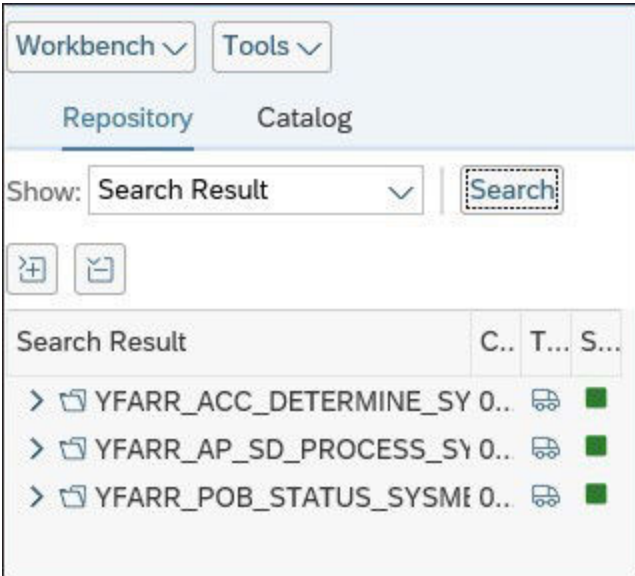


Figure 6.11 Posting Application Copied to the Customer Namespace

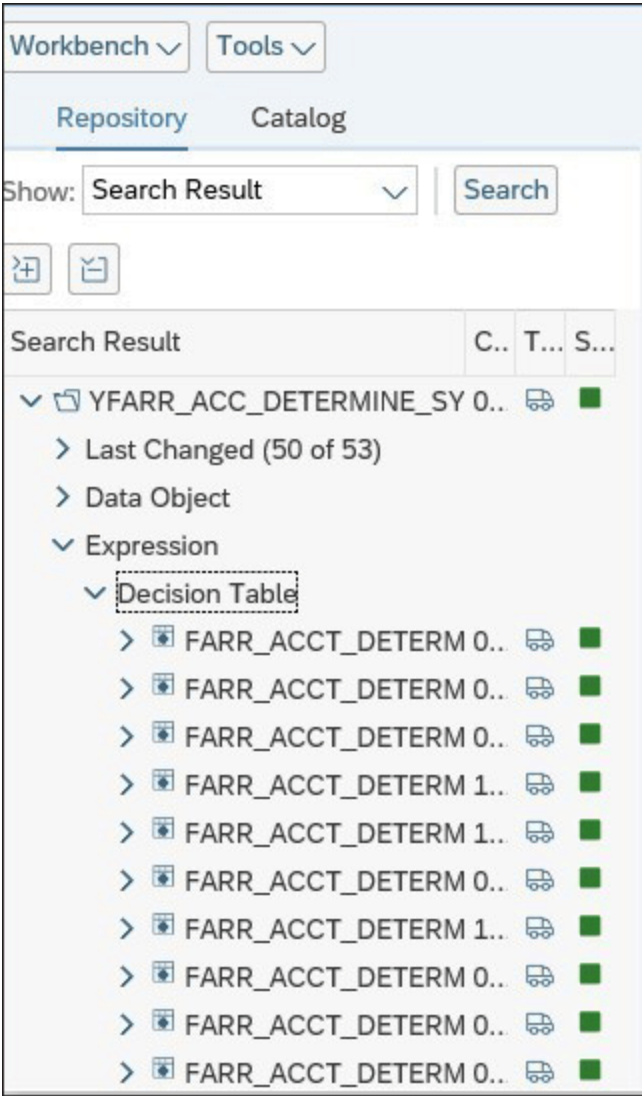


Figure 6.12 Available Decision Tables

■ Decision Table: FARR_ACCT_DETERMINE_DT_CORR, Revenue Adjustment for Allo...

> General

Table Contents

Find:

<input type="checkbox"/>	COMPANY_CODE	POB_NAME	POB_TYPE	REF_ACCOUNT	GL_ACCOUNT
<input type="checkbox"/>	=3001	=INSTRUMENT(Capital Sales)	=YI0001 (INSTRUMENT(Capital Sales))		4111111200 (Revenue Alloc Instr)
<input type="checkbox"/>	=3001	=INSTRUMENT (LEASE)	=YRI001 (INSTRUMENT (LEASE))		4111111200 (Revenue Alloc Instr)
<input type="checkbox"/>	=3001	=REAGENT (RR)	=YRR001 (REAGENT (Revenue Rental))		4111111201 (Revenue Alloc Reagen)
<input type="checkbox"/>	=3001	=SERVICE (BILLING)	=YS0001 (SERVICE (BILLING))		4111111202 (Revenue Alloc Servic)
<input type="checkbox"/>	=3001	=SERVICE(CONTR.FR)	=YRS001 (SERVICE (Contr. Free))		4111111202 (Revenue Alloc Servic)
<input type="checkbox"/>	=3001	=TEST PROFILE (Revenue Rental)	=YRT001 (TEST PROFILE (Revenue Rental))		4111111203 (RevAlloc TestProfile)
<input type="checkbox"/>	=3001	=Recalculation contract (FOC)	=ZP0001 (Recalculation contract (FOC))		4111111203 (RevAlloc TestProfile)

Figure 6.13 Revenue Adjustment Postings

■ Decision Table: FARR_ACCT_DETERMINE_DT_RADJ, Recievable Adjustment

|
 |
 |
 |
 |
 |
 |

> General

▼ Detail

|
 |

Table Contents

Find:

<input type="checkbox"/>	ACCT_PRINCIPLE	COMPANY_CODE	REF_ACCOUNT	GL_ACCOUNT
<input type="checkbox"/>		=3001		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>		=3003		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>		=3004		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>		=300		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>		=3009		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>		=3010		1121211100 (AR-Trade -RAR-Adj)
<input type="checkbox"/>		=3011		1121211100 (AR-Trade -RAR-Adj)

Figure 6.14 Receivable Adjustment

Search in Table: Postings

Number of hits:

Runtime: Maximum no. of hits:

Insert Column:

<input type="checkbox"/>	CoCode	AccP	Reconcil. Key	POB	CnTy	Category	D/C	Posting GUID
<input type="checkbox"/>	AU02	IFRS	20220120000101	105010	ZPR0	RA	S	6045BD8846D21EEDB68A0E6AC15E2633
<input type="checkbox"/>	AU02	IFRS	20220120000101	105010	ZPR0	RV	H	6045BD8846D21EEDB68A0E6AC15E2633
<input type="checkbox"/>	AU02	IFRS	20220120000101	105024	ZPR0	IC	S	6045BD8846D21EEDB0F79CEF64E8ABD4
<input type="checkbox"/>	AU02	IFRS	20220120000101	105024	ZPR0	RA	H	6045BD8846D21EEDB0F79CEF64E8ABD4
<input type="checkbox"/>	AU02	IFRS	20220120000101	105025	ZPR0	IC	S	6045BD8846D21EEDB0F8717D6EE31440
<input type="checkbox"/>	AU02	IFRS	20220120000101	105025	ZPR0	RA	H	6045BD8846D21EEDB0F8717D6EE31440
<input type="checkbox"/>	AU02	IFRS	20220120000101	105025	ZPR0	RA	S	6045BD8846D21EEDB0F8C5AC769458EB
<input type="checkbox"/>	AU02	IFRS	20220120000101	105025	ZPR0	RV	H	6045BD8846D21EEDB0F8C5AC769458EB

Figure 6.15 Table FARR_D_POSTING Keys

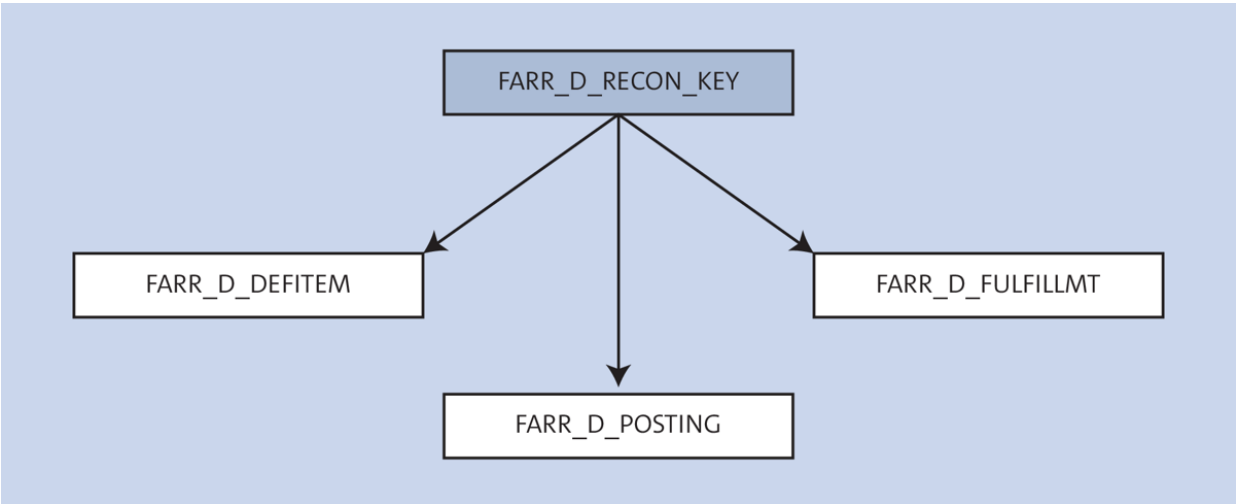


Figure 6.16 Table FARR_D_RECON_KEY Relationship

D	Year	Period	TC Amount	Crcy	LC Amount	LCurr	Second LC	LCur2	Third LC	LCur3	Contract	G/L Account	Stat	POB Type
8	2022	12	904,11-	AUD	904,11-	AUD	1.329,67-	USD			200530	2181000210		TIME_CREAT
A	2022	12	452,05-	AUD	452,05-	AUD	664,83-	USD			200532	2181000210		TIME_CREAT
E	2022	12	904,11-	AUD	904,11-	AUD	1.329,67-	USD			200542	2181000210		TIME_CREAT
9	2022	12	904,11-	AUD	904,11-	AUD	1.329,67-	USD			200544	2181000210		TIME_CREAT
C	2022	7	1.550,39-	EUR	1.074,94-	GBP	1.612,41-	USD			200077	1210000010		EVNT_INV
C	2022	7	1.240,32-	EUR	859,96-	GBP	1.289,94-	USD			200077	1210000010		EVNT_INV
6	2022	7	640,00-	GBP	640,00-	GBP	960,00-	USD			200079	1210000010		POC_MAN
4	2022	7	35.000,00-	GBP	35.000,00-	GBP	52.500,00-	USD			200084	1210000010		EVNT_MAN
9	2022	7	640,00-	GBP	640,00-	GBP	960,00-	USD			200090	1210000010		EVNT_MAN

Figure 6.17 Table FARR_D_POSTING Entries

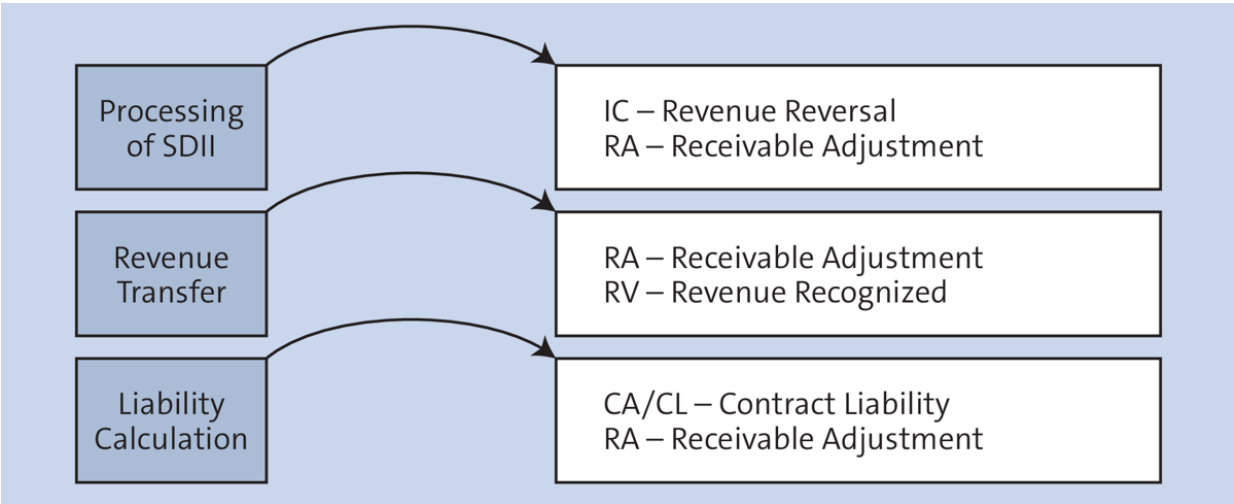


Figure 6.18 Relationship between Table FARR_D_POSTING and RAR Processes

Main Item (1)		Condition Item (3)													
POB(IFR...	Contract(I	ItemStatus	Final Inv.	Class Type	OrigItemID	OrtType	Package ID	Hist Exist	Error	Sales Ord.	SO ite...	Event Type	Ex.His.Ex	Start Date	Send.Com...
				01					■	90000994	10			01.04.2023	SD

Figure 6.19 Item in Table FARR_RAI_MON

Search in Table: Reconciliation Keys

Number of hits:

Runtime: Maximum no. of hits:

Insert Column:

<input type="checkbox"/>	CoCode	AccP	Contract	Year	Period	Reconcil. Key	Pro Date	Subarea	Status	Type	Reason	TM Convert	LiabAsset	Run ID	Date	Created	Time Stamp	Change	Time Stamp
<input type="checkbox"/>	GB...	IFRS	200606	2023	1	20230010000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	2	20230020000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	3	20230030000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	4	20230040000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	5	20230050000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	6	20230060000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	7	20230070000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	8	20230080000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	9	20230090000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	10	20230100000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	11	20230110000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544
<input type="checkbox"/>	GB04	IFRS	200606	2023	12	20230120000101		282	O							SM6918	20.230.507.080.544	SM6918	20.230.507.080.544

Figure 6.20 Table FARR_D_RECON_KEY Entries

Search in Table: Deferral Items

Number of hits:

Runtime: Maximum no. of hits:

Insert Column:

<input type="checkbox"/>	Reconcil. Key	POB	CnTy	Defer. Cat	Contract	Event Type	FulfilType	CoCode	AccP	Crcy	Unit	Src A/c No	Target A/c	Stat	Post.Amt	Postg.Qty.
<input type="checkbox"/>	20230010000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		1.967,21	0,081967
<input type="checkbox"/>	20230020000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,79	0,0847
<input type="checkbox"/>	20230030000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		1.967,21	0,081967
<input type="checkbox"/>	20230040000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,79	0,084699
<input type="checkbox"/>	20230050000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,79	0,0847
<input type="checkbox"/>	20230060000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		1.967,21	0,081967
<input type="checkbox"/>	20230070000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,79	0,084699
<input type="checkbox"/>	20230080000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		1.967,21	0,081968
<input type="checkbox"/>	20230090000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,79	0,084699
<input type="checkbox"/>	20230100000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,78	0,0847
<input type="checkbox"/>	20230110000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		1.901,64	0,079235
<input type="checkbox"/>	20230120000101	106006	ZPR0	MA	200606		T	GB04	IFRS	GBP	PC	5111009899	1132000990		2.032,79	0,084699

Figure 6.21 Table FARR_D_DEFITEM Entries

POB(FR_	Contract(I	ItemStatus	Final Inv.	Class Type	Original Item ID	OrtType	Package ID	Hist Exist	Error	Sales Ord.	SO It...	Event Type	Ex.His Ex	Start Date	Send.Com...	SourceSys.	SrcItemType	Source Item ID
<input type="checkbox"/>	106006	200606		03	0090000994000010	SDQI									SD		SDII	0801740324000010

Figure 6.22 Invoice in Table FARR_RAI_MON

Search in Table: Postings

Number of hits:

Runtime: Maximum no. of hits

Insert Column:

<input type="checkbox"/>	CoCode	AccP	Reconcil. Key	POB	CnTy	Category	D/C	Posting GUID	Year	Period	TC Amount	Crcy	LC Amount	LCurr
<input type="checkbox"/>	GB04	IFRS	20230010000101	106006	ZPR0	IC	S	6045BD8846D21EEDBB960EEFE6E9A271	2023	1	7.200,00	GBP	7.200,00	GBP
<input type="checkbox"/>	GB04	IFRS	20230010000101	106006	ZPR0	RA	H	6045BD8846D21EEDBB960EEFE6E9A271	2023	1	7.200,00-	GBP	7.200,00-	GBP

Figure 6.23 Table FARR_D_POSTING after Invoice Processing

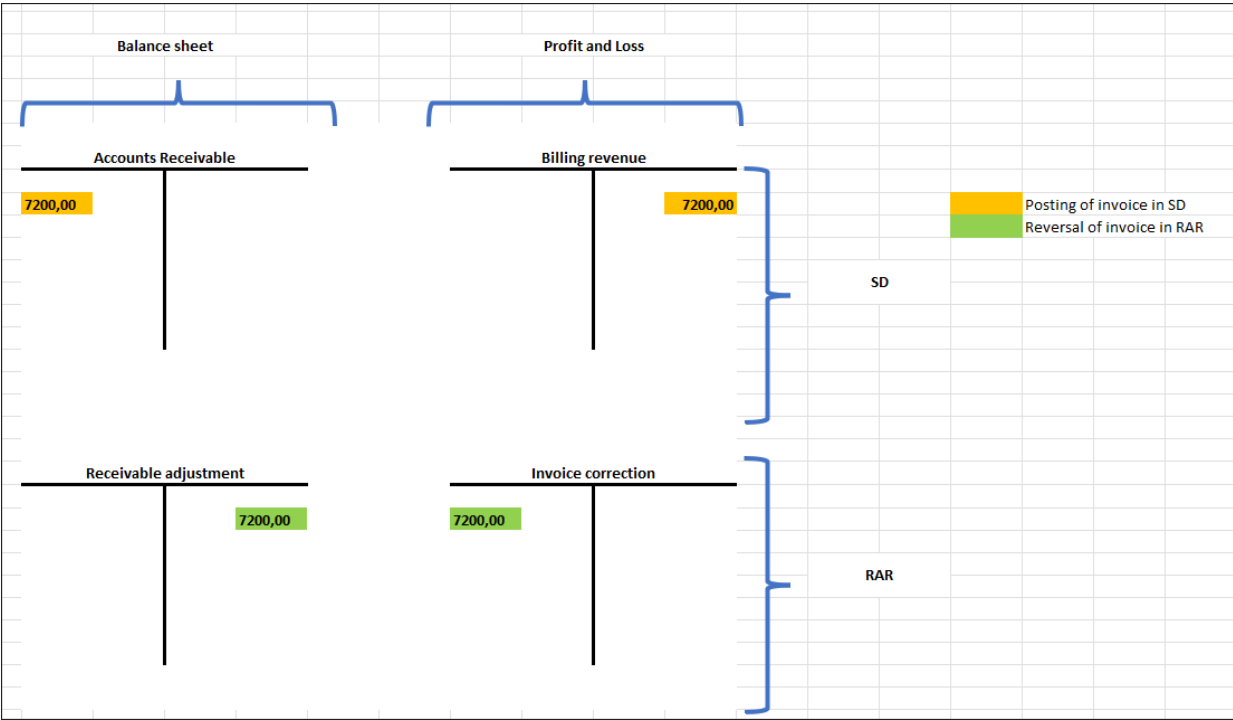


Figure 6.24 Posting Explanation after Invoice Processing

<input type="checkbox"/>	CoCode	Acc...	Reconcil. Key	POB	CnTy	Cate...	D/C	Posting GUID	Year	Peri...	TC Amount	Crcy	LC Amount	LCurr
<input type="checkbox"/>	GB...	IFR...	20230010000101	106006	ZPR0	IC	S	6045BD8846D21EEDBB960EEFE6E9A271	20...	1	7.200,00	GBP	7.200,00	GBP
						IC	<input type="checkbox"/>				7.200,00	GBP		
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPR0	RA	H	6045BD8846D21EEDBB960EEFE6E9A271	20...	1	7.200,00-	GBP	7.200,00-	GBP
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPR0		S	6045BD8846D21EEDBB96D3C56070288D	20...	1	1.967,21	GBP	1.967,21	GBP
						RA	<input type="checkbox"/>				5.232,79-	GBP		
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPR0	RV	H	6045BD8846D21EEDBB96D3C56070288D	20...	1	1.967,21-	GBP	1.967,21-	GBP
						RV	<input type="checkbox"/>				1.967,21-	GBP		
<input type="checkbox"/>											0,00	GBP		

Figure 6.25 Result of the Revenue Transfer

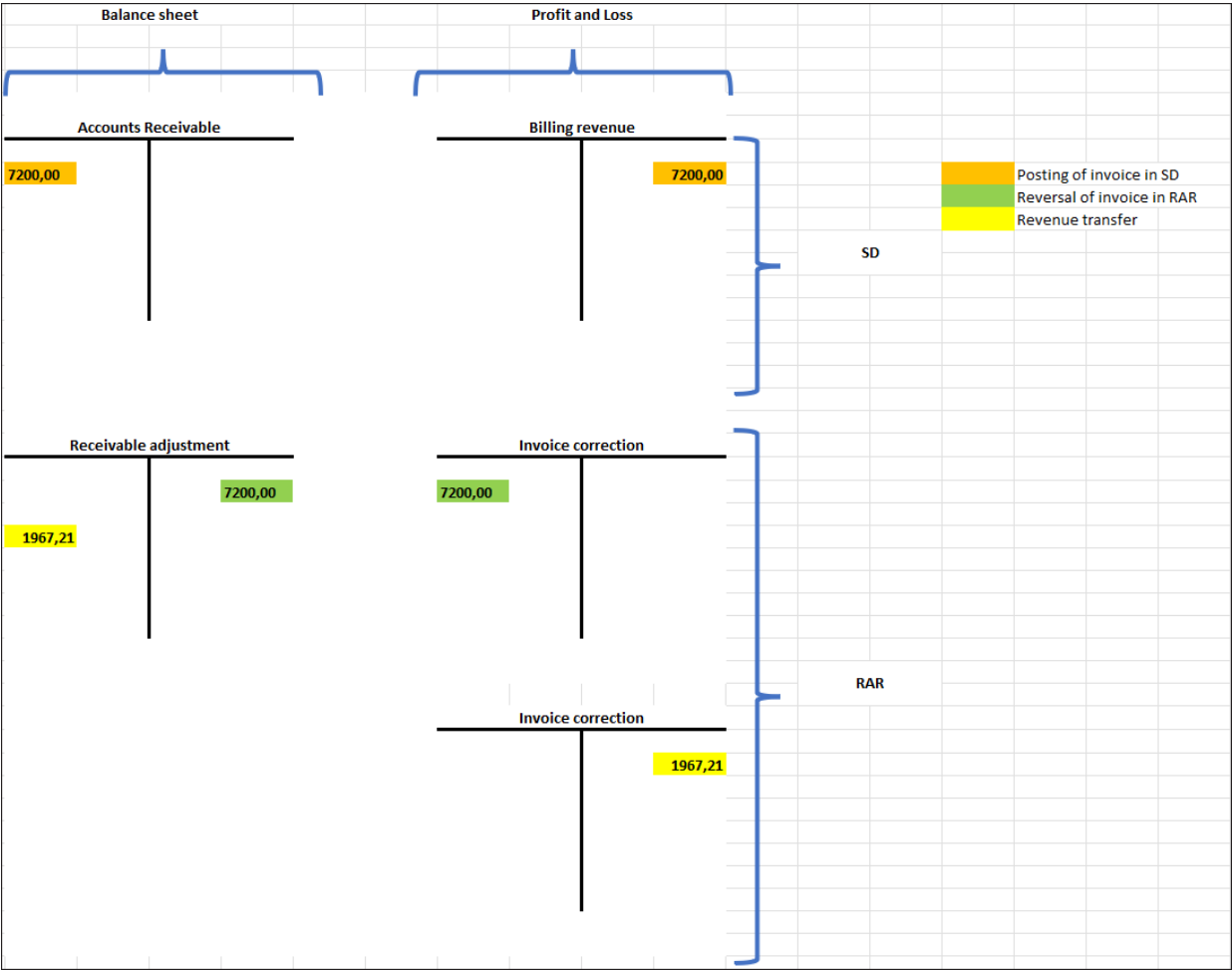


Figure 6.26 Postings after Revenue Transfer

<input type="checkbox"/>	CoCode	Acc...	Reconcil. Key	POB	CnTy	Cate...	D/C	Posting GUID	Year	P...	TC Amount	Crcy	L
<input type="checkbox"/>	GB...	IFR...	20230010000101	106006		CL	H	6045BD8846D21EEDBB972A9D6EDAEB62	20...	1	5.232,79-	GBP	
						CL	<input type="checkbox"/>				5.232,79-	GBP	
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPRO	IC	S	6045BD8846D21EEDBB960EEFE6E9A271	20...	1	7.200,00	GBP	
						IC	<input type="checkbox"/>				7.200,00	GBP	
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006		RA	S	6045BD8846D21EEDBB972A9D6EDAEB62	20...	1	5.232,79	GBP	
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPRO		H	6045BD8846D21EEDBB960EEFE6E9A271	20...	1	7.200,00-	GBP	
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPRO		S	6045BD8846D21EEDBB96D3C56070288D	20...	1	1.967,21	GBP	
						RA	<input type="checkbox"/>				0,00	GBP	
<input type="checkbox"/>	GB04	IFR...	20230010000101	106006	ZPRO	RV	H	6045BD8846D21EEDBB96D3C56070288D	20...	1	1.967,21-	GBP	
						RV	<input type="checkbox"/>				1.967,21-	GBP	
<input type="checkbox"/>											0,00	GBP	

Figure 6.27 Table FARR_D_POSTING after Running of Liability Calculation Program

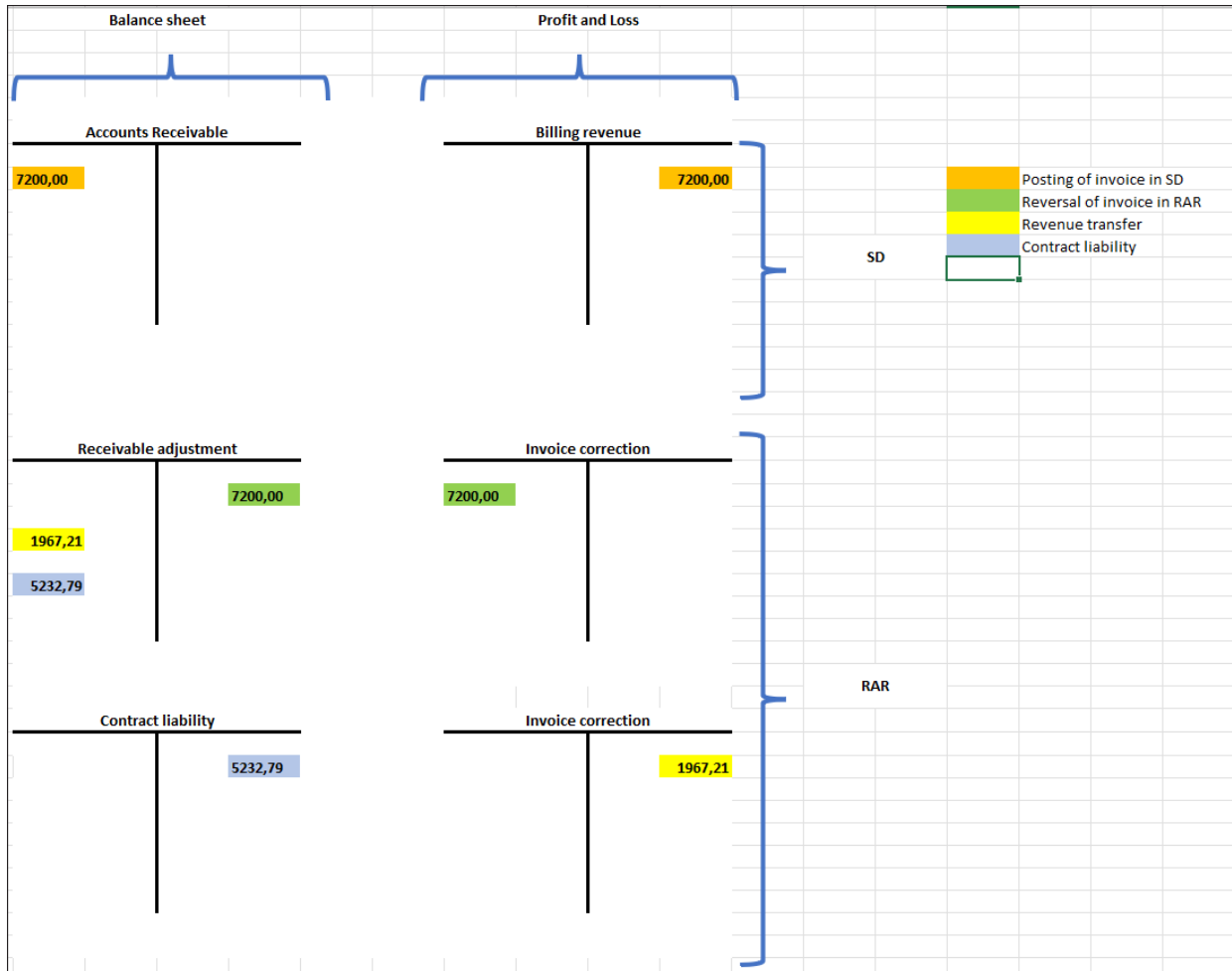


Figure 6.28 Postings after Contract Liability

Search in Table: Reconciliation Keys

Number of hits:

Runtime: Maximum no. of hits:

Insert Column:

CoCode	AccP	Contract	Year	Period	Reconcl. Key	Pro Date	Subarea	Status	Type	Reason	TM.Convert	LiabAsset	Run ID	Date	Created	Time Stamp	Change
GB04	IFRS	200606	2023	1	20230010000101		282	P				X			SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	2	20230020000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	3	20230030000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	4	20230040000101		282	P							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	5	20230050000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	6	20230060000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	7	20230070000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	8	20230080000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	9	20230090000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	10	20230100000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	11	20230110000101		282	O							SM6918	20.230.507.080.544	SM6918
GB04	IFRS	200606	2023	12	20230120000101		282	O							SM6918	20.230.507.080.544	SM6918

Status of Revenue Reconciliation Key (1) 11 Entries found

Reconciliation Key Status	Short Descript.
O	Open
C	Closed
I	In Process
A	Canceled
F	Failed
T	Time Base
L	Liability/Asset
M	Migration
S	Simulation by RWIN
R	Replaced by Shift Period Handling
P	Transferred

Figure 6.29 Table FARR_D_RECON_KEY after Liability Calculation

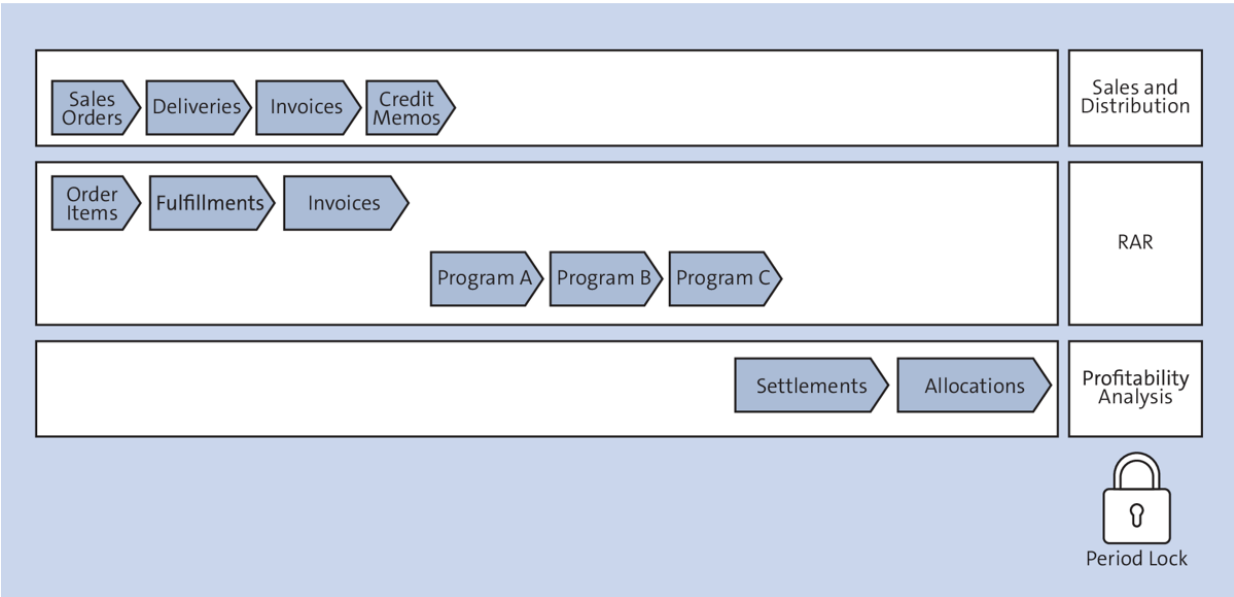


Figure 6.30 Closing Process with RAR

SAP Transfer Revenue

Job Monitor

Standard * (help icon) Hide Filter Bar Adapt Filters (0)

Company Code: Accounting Principle: Fiscal Year: Posting period:

Revenue Accounting Contract:

Job Settings **Schedule Date and Time**

* Job Name: * Schedule Mode: Schedule to Start Immediately
 Schedule to Start Later

Technical Parameter Start Date:

* Number of Intervals: Start Time:

Figure 6.31 Revenue Transfer Options

Revenue Posting Job Monitor

FI Documents and Revenue Accounting Contracts

✔ Job list refreshed

Revenue Posting Jobs

Display: All

Status	Job Name	Job Type	Created By	Planned Date/Time	Start Date/Time	End Date/Time
●	REV_TRANSFER_20230507	Transfer Revenue	Sreten Milosavljevic	07.05.2023 08:47:02	07.05.2023 08:52:02	07.05.2023 08:52:10

Figure 6.32 Revenue Transfer Completed

< **SAP** Calculate Contract Liabilities and Contract Assets 🔍 🔔

Job Monitor ⚙️ ?

Standard * ☹️ Hide Filter Bar Adapt Filters (0)

* Company Code: * Accounting Principle: Fiscal Year: Posting period:

Revenue Accounting Contract:

Job Settings **Schedule Date and Time**

* Job Name: * Schedule Mode: Schedule to Start Immediately
 Schedule to Start Later

* Value Date:

Start Date:

Start Time:

Technical Parameter

* Number of Intervals:

Figure 6.33 Liability Calculation Job

https://xc-i-gldev01d.corp.hitachi-powergrids.com:44301/ui2/nwbc/CIMS4H_NPR_FUNCT-CONSULTANT/?sap-nwbc-node=navigate_absolute&sap-nwbc-new_window=X

Revenue Posting Job Monitor

FI Documents and Revenue Accounting Contracts

Revenue Posting Jobs

Display: All [Delete Job] [Refresh] [Show]

Status	Job Name	Job Type	Created By	Planned Date/Time	Start Date/Time
	LIABILITY_20230507	Calculate Contract Liabilities and Assets	Sreten Milosavljevic	07.05.2023 09:23:57	00.00.0000 00:00:00
	REV_TRANSFER_20230507	Transfer Revenue	Sreten Milosavljevic	07.05.2023 09:03:20	07.05.2023 09:04:01
	REV_TRANSFER_20230507	Transfer Revenue	Sreten Milosavljevic	07.05.2023 08:47:02	07.05.2023 08:52:02

Job Details


Figure 6.34 Liability Calculation Being Scheduled

Job Details		
<input type="button" value="Parallel Processing Status"/> <input type="button" value="Application Log"/> <input type="button" value="Search Criteria"/>		
Field Name	Opera...	From
Company Code	Equals	GB04
Accounting Principle	Equals	IFRS
Fiscal Year	Equals	2023
Posting period	Equals	001
Contract	Equals	200606
Value Date	Equals	30.04.2023
Synchronous Call	Equals	X

Figure 6.35 Job Details Display

Job Details	
<div style="display: flex; justify-content: space-between;"> Parallel Processing Status Application Log Search Criteria </div>	
Log Entry	
	<ul style="list-style-type: none"> ● Calculating contract liabilities and assets completed at 07.05.2023 09:27:12 ▼ ● Subordinate: 07.05.2023 09:27:05 SM6918 <ul style="list-style-type: none"> ● Job 1 started at 07.05.2023 09:27:05 ● Start processing contracts with KeyPP from 000 to 199 07.05.2023 09:27:05 ● Post. entries for 0 of 0 contracts created; 0 contracts in migration. ● End processing contracts with KeyPP from 000 to 199 07.05.2023 09:27:05 ● Start processing contracts with KeyPP from 200 to 399 07.05.2023 09:27:05 ● Saved the results at 07.05.2023 09:27:06,created posting: 2,updated posting: 0 <li style="border: 2px solid black; padding: 2px;">● Post. entries for 1 of 1 contracts created; 0 contracts in migration. ● End processing contracts with KeyPP from 200 to 399 07.05.2023 09:27:06

Figure 6.36 Application Log in Liability Calculation

 **Start Revenue Posting Run**

Execute Get variant... Further Selections More ▾

Selection Data

* Company Code: to:

* Accounting Principle:


* Fiscal Year:


* Posting Period:

Contract: to:

Further Selections Exist:

Parameters

* Posting Date: 

* Run Mode: ▾ 

Close Period for Rev:

Posting Check:

Switch Off Parallel Posting:

Technical Parameters

Number of Intervals:

Dialog Mode:

Figure 6.37 Program C Initial Screen

Selection Data

* Company Code: to:

* Accounting Principle:

* Fiscal Year:

* Posting Period:

Contract: to:

Further Selections Exist:

Parameters

* Posting Date:

* Run Mode:

Close Period for Rev:

Posting Check:

Switch Off Parallel Posting:

Technical Parameters

Number of Intervals:

Dialog Mode:

Figure 6.38 Running Posting in Test Mode

Type	Message Text
■	Revenue posting started at 08.05.2023 13:39:59
●	Period 2023001 is closed for rev. acct. for co. code 3003 acct. principle IFRS
■	Revenue posting completed at 08.05.2023 13:39:59

Figure 6.39 Error Message Due to a Closed Period

Change Display/Change Mode Display Another Document Select Individual Object Disp

Data Entry View

Document Number: Company Code: Fiscal Year: Document Date: Posting Date: Period: Reference: Cross-Comp.No.: Currency: Texts Exist: Ledger Group:

CoCd	Item K.	SC Account	Description	±	Amount	Curr.	Tx	Plant	Material
3003	1 50	1121211100	AR-Trade -RAR-Adj		373,95-	EUR			
3003	2 50		AR-Trade -RAR-Adj		373,95-	EUR			
3003	8 40		AR-Trade -RAR-Adj		373,95	EUR			
3003	10 40		AR-Trade -RAR-Adj		602,20	EUR			
		1121211100		*	228,25	EUR			
3003	9 40	1121312000	Cntrct assets/umbRAR		373,95	EUR			
		1121312000		*	373,95	EUR			
3003	11 40	4111111201	Revenue Alloc Reagen		219,97	EUR			
3003	12 40		Revenue Alloc Reagen		109,99	EUR			
3003	13 40		Revenue Alloc Reagen		43,99	EUR			
		4111111201		*	373,95	EUR			
3003	7 50	4111111202	Revenue Alloc Servic		373,95-	EUR			1010100001
		4111111202		*	373,95-	EUR			
3003	3 50	4111111301	Revenue Adj Reagent		266,00-	EUR			1010100001
3003	4 50		Revenue Adj Reagent		133,00-	EUR			1010100001
3003	5 50		Revenue Adj Reagent		53,20-	EUR			1010100001
		4111111301		*	452,20-	EUR			
3003	6 50	4111111302	Revenue Adj Service		150,00-	EUR			1010100001

Document Header: 3003 Company Code

Document type: Revenue Accounting
 Doc. Header Text:
 Reference: Document Date:
 Posting Date:
 Currency: Posting period: /
 Ref. Transactn:
 Reference Key: Log. System:
 Created By:
 Entry Date: Time of Entry:
 TCode:
 Changed On: Last Update:
 Ledger Grp:
 ActgPrinciple: International Financial Reporting Standards
 Ref key(head) 1: Ref key 2:

Continue/Confirm Cancel

Figure 6.40 Posting Document Coming from RAR

Search in Table: Reconciliation Keys

Number of hits:

Runtime: Maximum no. of hits:

Insert Column:

CoCd	AccP Contract	Year Peri...	Reconcl. Key	Pro Split Date	Subarea	Status	Type	Reason	TM.Convert	LiabAss...	Ru...	Date	Created by	Time Stamp
<input type="checkbox"/>	3003	IFRS 300313327	20... 1 20230010000101		341	C				X	275	04.05.2023	DEUTEGETHOFN	20.230.504.112.358
<input type="checkbox"/>	3003	IFRS 300313323	20... 1 20230010000101		449	C				X	273	03.05.2023	DEUTEGETHOFN	20.230.502.151.843
<input type="checkbox"/>	3003	IFRS 300313321	20... 1 20230010000101		124	C				X	272	02.05.2023	EYG0890	20.230.502.093.713
<input type="checkbox"/>	3003	IFRS 300313319	20... 1 20230010000102		187	C				X	270	28.04.2023	EYG0890	20.230.428.142.415
<input type="checkbox"/>	3003	IFRS 300313319	20... 1 20230010000103	28.04.2023	187	C				X		28.04.2023	EYG0890	20.230.428.142.415
<input type="checkbox"/>	3003	IFRS 300313319	20... 1 20230010000101		187	C				X	269	28.04.2023	EYG0890	20.230.428.142.046
<input type="checkbox"/>	3003	IFRS 300313313	20... 1 20230010000102		848	C				X	268	28.04.2023	EYG0890	20.230.424.132.144

Figure 6.41 Posting Key in FARR_D_RECON_KEY

View: [Standard View] [Print Version] [Export]													
G/L Document	Revenue Acco...	Performance Obli...	Performance Obli...	Posting Categ...	Condition Type	G/L Account	Account Descri...	Debit/Credit	Amount	Currency	Amount in LC	Local Cu	
5300000014	300313321	205205	REAGENT (RR)	Receivable Adjus...	ZALL	1121211100	AR-Trade -RAR-Adj	Credit	109,99	EUR	109,99	EUR	
	300313321	205205	REAGENT (RR)	Receivable Adjus...	ZP01			Debit	133,00	EUR	133,00	EUR	
	300313321	205206	REAGENT (RR)	Receivable Adjus...	ZALL			Credit	43,99	EUR	43,99	EUR	
	300313321	205206	REAGENT (RR)	Receivable Adjus...	ZP01			Debit	53,20	EUR	53,20	EUR	
	300313321	205207	SERVICE (BILLING)	Receivable Adjus...	ZALL			Credit	373,95	EUR	373,95	EUR	
	300313321	205207	SERVICE (BILLING)	Receivable Adjus...	ZALL			Debit	373,95	EUR	373,95	EUR	
	300313321	205207	SERVICE (BILLING)	Receivable Adjus...	ZP01			Debit	150,00	EUR	150,00	EUR	
5300000014						1121211100	^ AR-Trade -RAR-...		• 228,25	EUR	• 228,25	EUR	
5300000014						^ 1121211100			• • 228,25	EUR	• • 228,25	EUR	
5300000014	300313321	205207	SERVICE (BILLING)	Contract Asset		1121312000	Cntrct assets/unBRAR	Debit	373,95	EUR	373,95	EUR	
5300000014						1121312000	^ Cntrct assets/un...		• 373,95	EUR	• 373,95	EUR	
5300000014						^ 1121312000			• • 373,95	EUR	• • 373,95	EUR	
5300000014	300313321	205204	REAGENT (RR)	Recognized Rev...	ZALL	4111111201	Revenue Alloc Rea...	Debit	219,97	EUR	219,97	EUR	
	300313321	205205	REAGENT (RR)	Recognized Rev...	ZALL			Debit	109,99	EUR	109,99	EUR	
	300313321	205206	REAGENT (RR)	Recognized Rev...	ZALL			Debit	43,99	EUR	43,99	EUR	
5300000014						4111111201	^ Revenue Alloc R...		• 373,95	EUR	• 373,95	EUR	
5300000014						^ 4111111201			• • 373,95	EUR	• • 373,95	EUR	
5300000014	300313321	205207	SERVICE (BILLING)	Recognized Rev...	ZALL	4111111202	Revenue Alloc Servic	Credit	373,95	EUR	373,95	EUR	
5300000014						4111111202	^ Revenue Alloc S...		• 373,95	EUR	• 373,95	EUR	
5300000014						^ 4111111202			• • 373,95	EUR	• • 373,95	EUR	

Figure 6.42 Reconciliation App Results

Class: **SD01** 01 Order Item
 Class Type: **01**

Active	Interface Component	Inform...	Field ...
<input checked="" type="checkbox"/>	Basic Fields for Conditions	I	
<input checked="" type="checkbox"/>	Basic Fields for Order Items (Conditions)	I	
<input checked="" type="checkbox"/>	Basic Fields for Main Items	I	
<input checked="" type="checkbox"/>	Basic Fields for Order Items (Main Items)	I	
<input type="checkbox"/>	CA Basic Fields for Main Items	I	
<input type="checkbox"/>	CA Fields for Order Items	I	
<input checked="" type="checkbox"/>	CO-PA Fields for Order Items (COPACRIT)	I	
<input type="checkbox"/>	CRM Fields for Order Items	I	
<input checked="" type="checkbox"/>	SD Fields for Order Items	I	

Interface Component: **COPA_MI01** CO-PA Fields for Order Items (COPACRIT)
 Status: **2** Processable
 Fld Disp.: **MI** Main Item

Field Name	Data element	Data Type	Lngh	Short Description
ABSMG_ME	MEINS	UNIT	3	Base Unit of Measure
BONUS	BONUS	CHAR	2	Volume rebate group
BRSCH	BRSCH	CHAR	4	Industry key
BZIRK	BZIRK	CHAR	6	Sales District
EFORM	RKEG_EFORM	CHAR	5	Form of manufacture
GEBIE	RKEG_GEBIE	CHAR	4	Area
KDGRP	KDGRP	CHAR	2	Customer Group
LAND1	LAND1_GP	CHAR	3	Country/Region Key
MAABC	MAABC	CHAR	1	ABC Indicator
MATKL	MATKL	CHAR	9	Material Group
VKBUR	VKBUR	CHAR	4	Sales Office
VKGRP	VKGRP	CHAR	3	Sales Group
ARTNR	ARTNR	CHAR	40	Product number
BUKRS	BUKRS	CHAR	4	Company Code
FKART	FKART	CHAR	4	Billing Type
GSBER	GSBER	CHAR	4	Business Area
KAUFN	KDAUF	CHAR	10	Sales Order Number
KDPOS	POSNR_VA	NUMC	6	Sales Document Item
KMBRND	RKESKMBRND	NUMC	2	Brand
KMCATG	RKESKMCATG	NUMC	2	Business field
KMHI01	HIEZU01	CHAR	10	Customer hierarchy Level 1
KMHI02	HIEZU02	CHAR	10	Customer hierarchy Level 2

Figure 6.43 Profitability Analysis Basic Setup

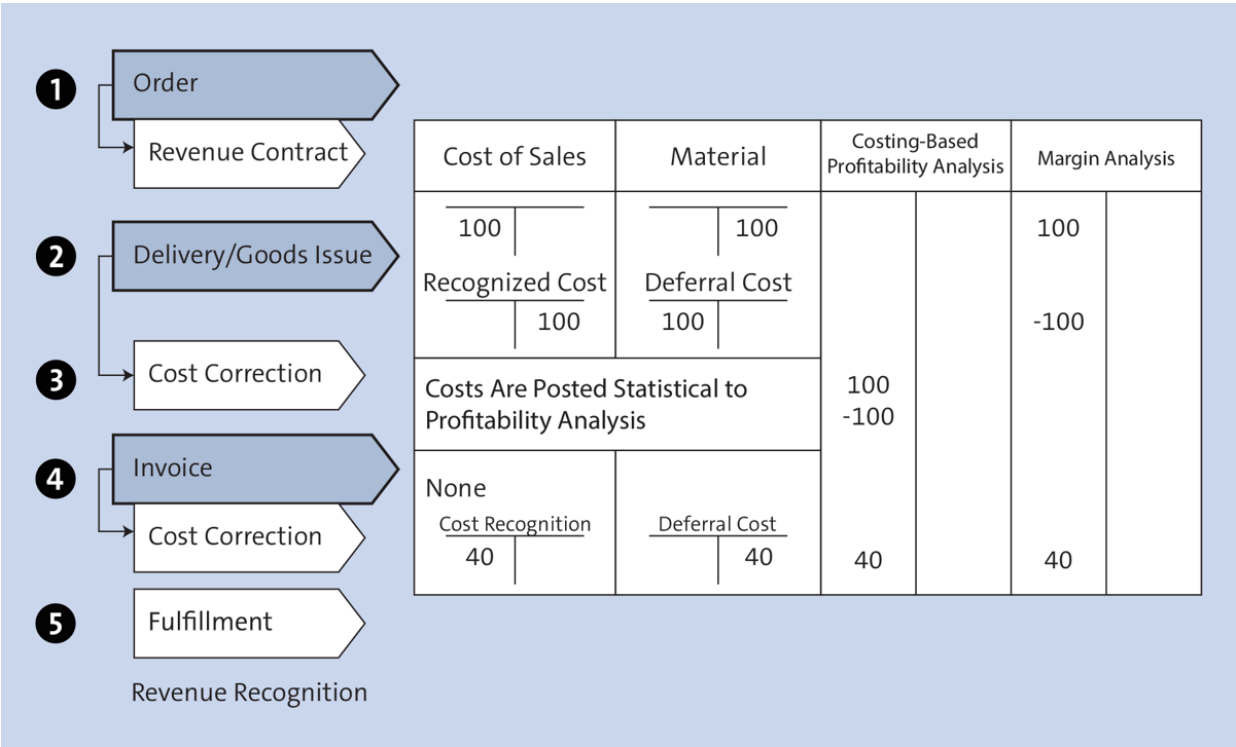


Figure 6.44 RAR: Profitability Analysis Interface

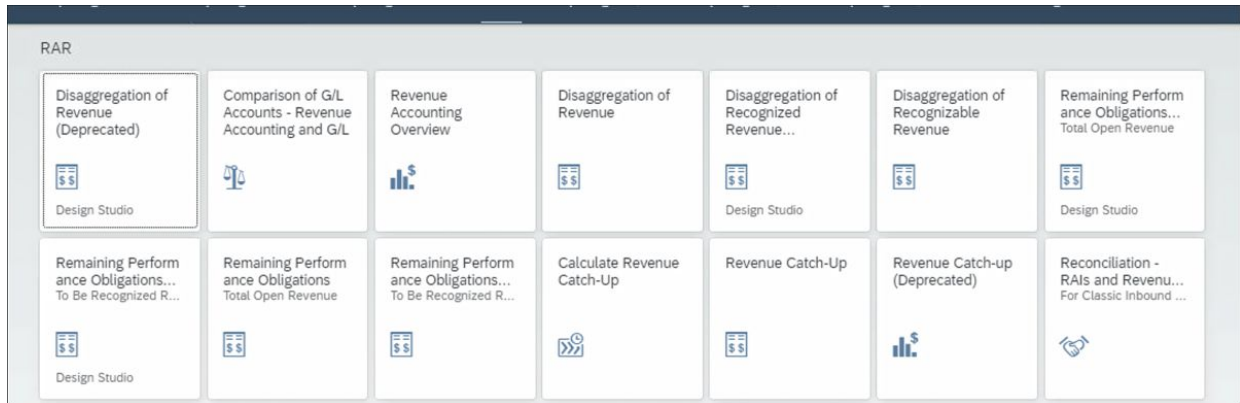


Figure 6.45 RAR Reports in SAP Fiori

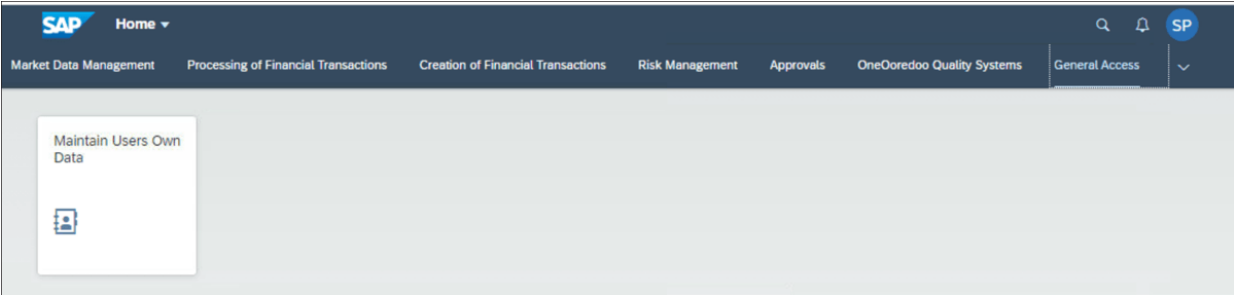


Figure 6.46 Initial Screen

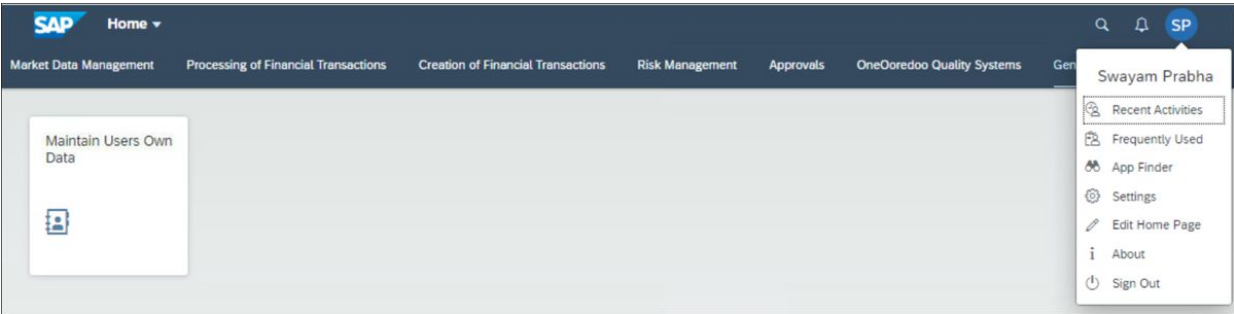


Figure 6.47 App Finder

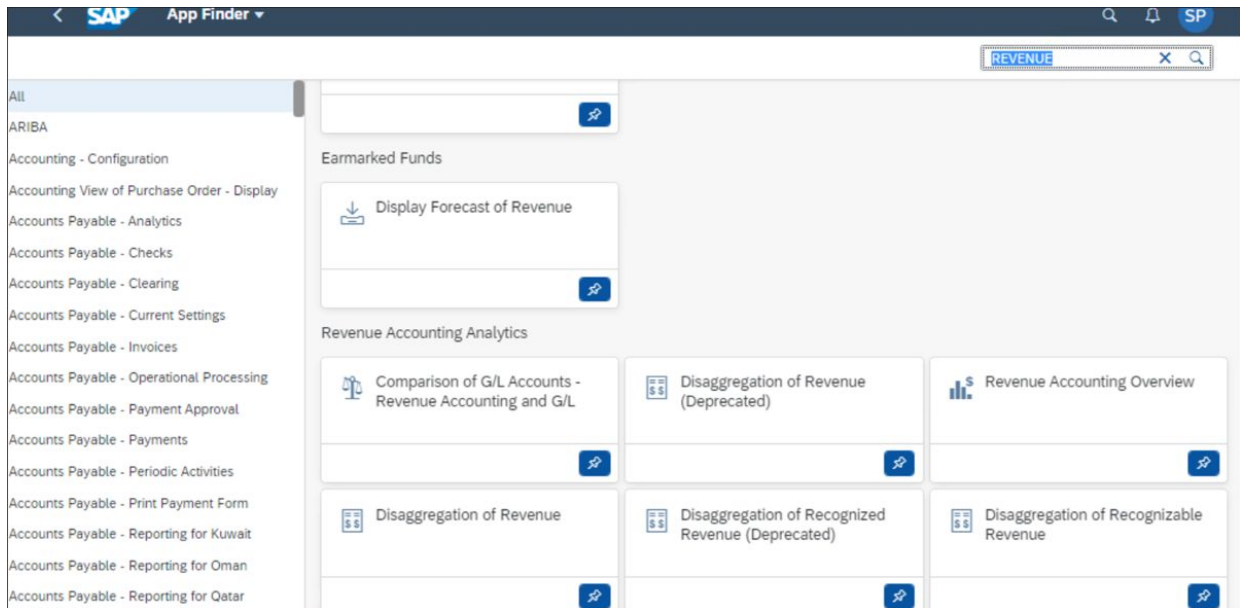


Figure 6.48 Apps Screen with the Search for “REVENUE”

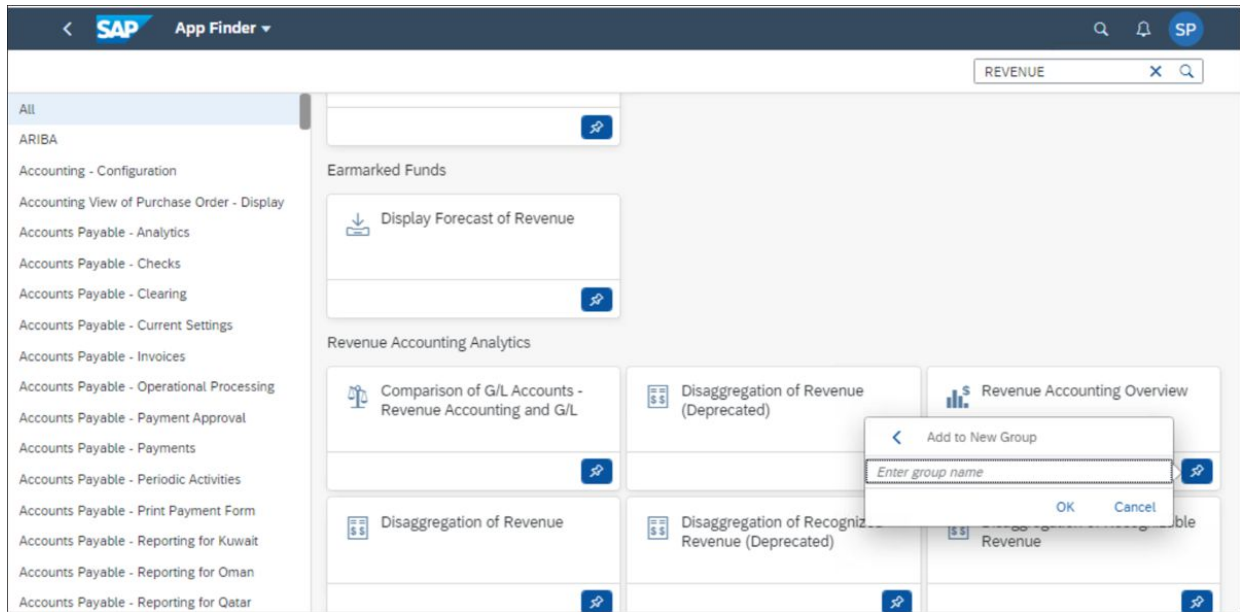


Figure 6.49 Add App to the New Group

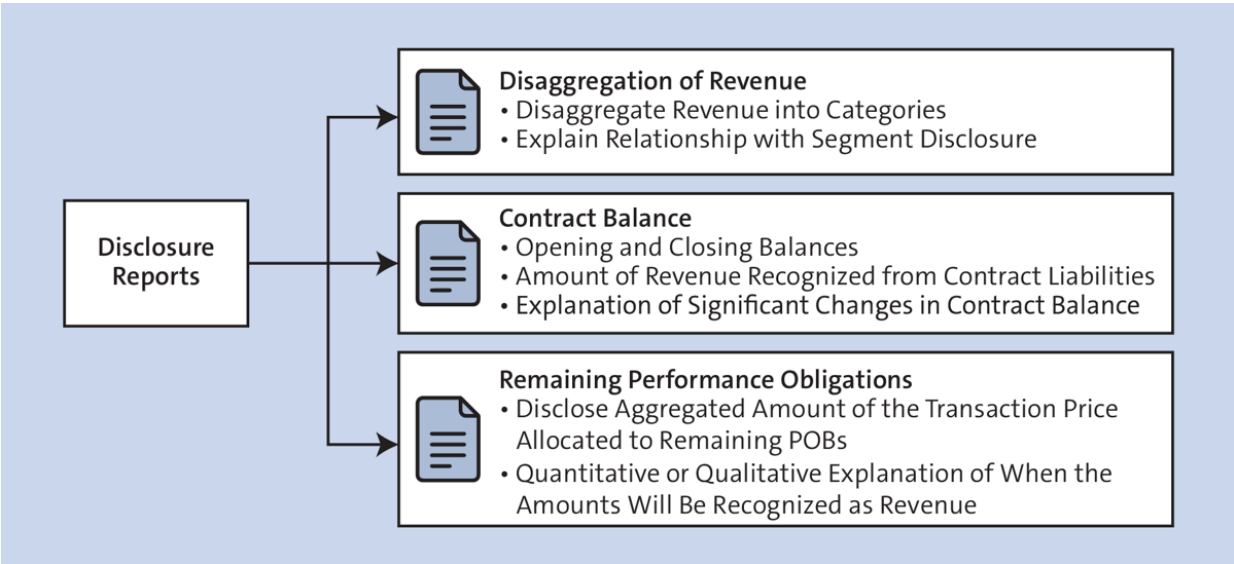


Figure 6.50 Disclosure Reports

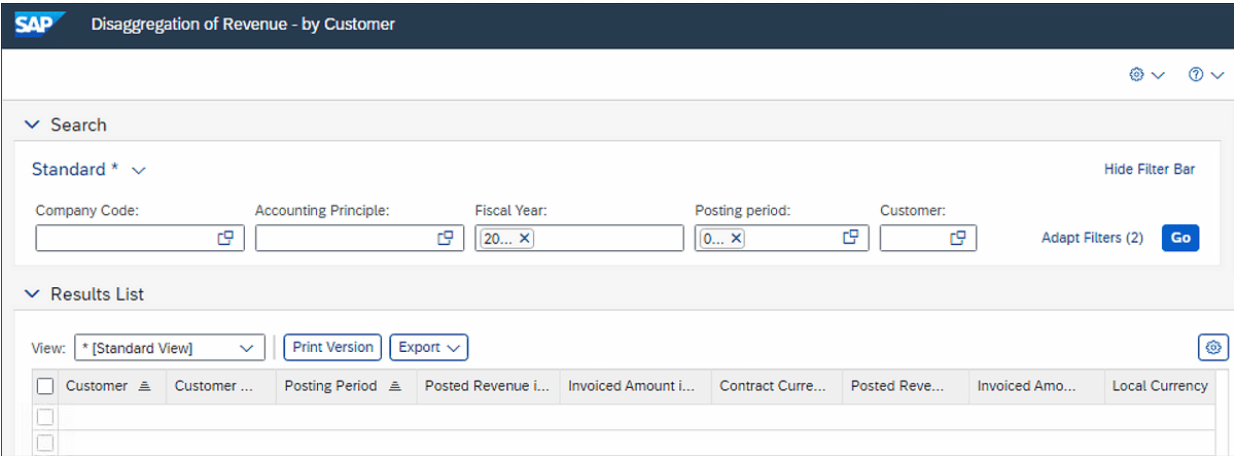


Figure 6.51 Disaggregation of Revenue – By Customer Report

SAP Search Revenue Contracts

Search

Standard

Company Code: Revenue Accounting Contract: Operational Document: Created By: Created on: DD.MM.YYYY or descri... Show Archived Contracts Only:

Adapt Filters (0) Go

Results List

Show: General View

Performance Obligation Structure Price Allocation Revenue Schedule Perform Contract Combination Quick Combine Reprocess Account Determination Reprocess Contracts Comprehensive View

<input type="checkbox"/>	Revenue Accounting Cont...	Company...	C...	Accounting Princi...	Custo...	Customer Name	No. of Perf. Oblig.	Contractual Price	Contract Allocation Effect	Curr...	Contract
<input type="checkbox"/>	305	1010	O...	IFRS	1010000...	Supplier for SD 2	1	3.000,00	0,00	QAR	In Proce
<input type="checkbox"/>	304	1010	O...	IFRS	1010000...	Supplier for SD 2	1	3.000,00	0,00	QAR	In Proce
<input type="checkbox"/>	303	1010	O...	IFRS	1010000...	Supplier for SD 2	1	3.000,00	0,00	QAR	In Proce
<input type="checkbox"/>	302	1010	O...	IFRS	1010000...	Supplier for SD 2	1	3.000,00	0,00	QAR	In Proce
<input type="checkbox"/>	301	1010	O...	IFRS	1010000...	Supplier for SD 2	1	3.000,00	0,00	QAR	In Proce

Figure 6.52 Search Revenue Contracts

SAP Manage Revenue Contracts

Standard* ▼

Editing Status: All ▼ Company Code: Accounting Principle: Revenue Contract: 301 ✕ Business Partner: Created By:

Contract Status:

[Go](#) Adapt Filters (1)

Revenue Contracts (1) Standard ▼ [Show Details](#)

Revenue Contract	Accounting ...	Business Partner	Contractual Price	Allocation Effect	Contract Status	Created On
301	IFRS		3.000,00	QAR 0,00	QAR I (In Process)	01.08.2023 >

Figure 6.53 Manage Revenue Contracts

SAP Revenue Explanation

Standard* ↕

Search Company Code: Accounting Principle: Revenue Contract: Performance Obligation: Performance Obligation Name: Operational Document:

Business Partner: Created By: Go Adapt Filters (1)

Performance Obligations (1) Standard ↕ Show Details ⚙️ 📄 ⌵

Performance Obliga...	Performance Obligation Name	Revenue Contract	Business Partner	Composition	Contractual Price		Allocated Amount	
3002	VOICE	301		D (Distinct)	3.000,00	QAR	3.000,00	QAR

Figure 6.54 Revenue Explanation

Search											
Standard *											
Company Code:		Accounting Principle:		Fiscal Year:		Posting period:		G/L Document:		Posting Date:	
3003		IFRS		2023		001				DD.MM.YYYY or description	
Results List											
View: [Standard View] Print Version Export 											
<input type="radio"/>	G/L Document	Revenue Acco...	Performance Obli...	Performance Obli...	Posting Categ...	Condition Type	G/L Account	Account Descri...	Debit/Credit	Amount	Currency
<input checked="" type="radio"/>	5300000000	300313225	203596	REAGENT (RR)	Receivable Adjus...	ZALL	1121211100	AR-Trade -RAR-Adj	Credit	175,15	EUR
<input type="radio"/>		300313225	203596	REAGENT (RR)	Receivable Adjus...	ZP02			Debit	282,57	EUR
<input type="radio"/>		300313225	203597	REAGENT (RR)	Receivable Adjus...	ZALL			Credit	38,84	EUR
<input type="radio"/>		300313225	203597	REAGENT (RR)	Receivable Adjus...	ZP02			Debit	141,28	EUR
<input type="radio"/>		300313225	203598	REAGENT (RR)	Receivable Adjus...	ZALL			Credit	8,68	EUR
<input type="radio"/>		300313225	203598	REAGENT (RR)	Receivable Adjus...	ZP02			Debit	13,42	EUR
<input type="radio"/>		300313225	203599	INSTRUMENT (LEA...	Receivable Adjus...				Credit	176,26	EUR
<input type="radio"/>		300313225	203599	INSTRUMENT (LEA...	Receivable Adjus...	R100			Credit	640,60	EUR
<input type="radio"/>		300313225	203599	INSTRUMENT (LEA...	Receivable Adjus...	ZALL			Debit	176,26	EUR
<input type="radio"/>		300313225	203599	INSTRUMENT (LEA...	Receivable Adjus...	ZPST			Debit	640,60	EUR
<input type="radio"/>		300313225	203601	INSTRUMENT (LEA...	Receivable Adjus...				Credit	12,20	EUR
<input type="radio"/>		300313225	203601	INSTRUMENT (LEA...	Receivable Adjus...	R100			Credit	41,69	EUR
<input type="radio"/>		300313225	203601	INSTRUMENT (LEA...	Receivable Adjus...	ZALL			Debit	12,20	EUR
<input type="radio"/>		300313225	203601	INSTRUMENT (LEA...	Receivable Adjus...	ZPST			Debit	41,69	EUR
<input type="radio"/>		300313225	203602	INSTRUMENT (LEA...	Receivable Adjus...				Credit	3,86	EUR
<input type="radio"/>		300313225	203602	INSTRUMENT (LEA...	Receivable Adjus...	R100			Credit	11,10	EUR
<input type="radio"/>		300313225	203602	INSTRUMENT (LEA...	Receivable Adjus...	ZALL			Debit	3,86	EUR
<input type="radio"/>		300313225	203602	INSTRUMENT (LEA...	Receivable Adjus...	ZPST			Debit	11,10	EUR

Figure 6.55 Financial Postings from RAR

Standard *

Hide Filter Bar Adapt Filter

Company Code: Accounting Principle: Fiscal Year: Posting period: G/L Account: Has Difference: Run in Background:

G/L Docs Only from RAR:

▼ Results List

View: Print Version

Status	G/L Account	Account Description	Revenue Accounting in...	General Ledger in Doc...	Difference in Document C...	Document Currency	Revenue Accounting in...	General Ledger in LC	Difference in LC	Local Currency
<input type="radio"/>	1121211100		20.442,85	20.442,85	0,00	EUR	20.442,85	20.442,85	0,00	EUR
<input type="radio"/>	1121312000		936,24-	936,24-	0,00	EUR	936,24-	936,24-	0,00	EUR
<input type="radio"/>	2151911000		10.031,62	10.031,62	0,00	EUR	10.031,62	10.031,62	0,00	EUR
<input type="radio"/>	4111111200		605,64-	605,64-	0,00	EUR	605,64-	605,64-	0,00	EUR
<input type="radio"/>	4111111201		1.369,61	1.369,61	0,00	EUR	1.369,61	1.369,61	0,00	EUR
<input type="radio"/>	4111111202		763,61-	763,61-	0,00	EUR	763,61-	763,61-	0,00	EUR
<input type="radio"/>	4111111301		20.580,12-	20.580,12-	0,00	EUR	20.580,12-	20.580,12-	0,00	EUR
<input type="radio"/>	4111111302		20.150,00-	20.150,00-	0,00	EUR	20.150,00-	20.150,00-	0,00	EUR
<input type="radio"/>	4111111600		10.079,77	10.079,77	0,00	EUR	10.079,77	10.079,77	0,00	EUR
<input type="radio"/>	4391111001		1.111,76	1.111,76	0,00	EUR	1.111,76	1.111,76	0,00	EUR

Figure 6.56 Reconciliation Report between General Ledger and RAR

Standard* ⌵ 🔗

Company Code: ✕ 🔗
 Accounting Principle: ✕ 🔗
 Revenue Contract: 🔗
 Business Partner: 🔗
 Created By: 🔗
 Adapt Filters (2)

Revenue Contracts (91) | Standard ⌵ Show Details ⚙️ 📄 ⌵

Revenue Contract	Accounting ...	Business Partner	Contractual Price	Allocation Effect	Created On	Created By	
93	IFRS		1,200,000 KWD	0,000 KWD	13.09.2023	SPRABHA	>
92	IFRS		1,200,000 KWD	0,000 KWD	13.09.2023	SPRABHA	>
91	IFRS		1,200,000 KWD	0,000 KWD	13.09.2023	SPRABHA	>
90	IFRS		100,000 KWD	0,000 KWD	13.09.2023	SPRABHA	>
89	IFRS		300,000 KWD	20,000 KWD	12.09.2023	SPRABHA	>
88	IFRS		300,000 KWD	20,000 KWD	12.09.2023	SPRABHA	>
87	IFRS		600,000 KWD	40,000 KWD	12.09.2023	SPRABHA	>

Figure 6.57 Revenue Schedule Initial Screen with List of Contracts

Performance Obligation Structure Price Allocation Revenue Schedule Contract Notes (0) Contract Attachments (0) More Refresh

Summary

Recognizable Revenue:	300,000 KWD	Recognizable Cost:	0,00
Posted Revenue:	0,000 KWD	Posted Cost:	0,00
Planned Revenue:	0,000 KWD	Planned Cost:	0,00
Unscheduled Revenue:	0,000 KWD	Unscheduled Cost:	0,00
Total Revenue:	300,000 KWD	Total Cost:	0,00
Fulfilled Progress:	100 %		

View: [Standard View] Display Fulfillment Details Display Fulfill. Details for Non-Distinct Perf. Oblig. Manually Fulfill Performance Obligations Print Version Export

Change Spreading Notes Attachments

<input type="checkbox"/>	Status	Perfo...	Account...	Perform...	Alloc...	Effectiv...	Quantity	Fulfill...	Cumula...	Revenue	Unsu...	Posted ...	Posting ...	Invoic...	Revenu...
<input type="checkbox"/>	♦	221	2023007	SERVICE	180,000	1	0,333333	33.33	33.33	60,000	60,000	0,000	0,000	0,000	0,000
<input type="checkbox"/>	♦	221	2023008	SERVICE	180,000	1	0,333334	33.34	66.67	60,000	60,000	0,000	0,000	0,000	0,000
<input type="checkbox"/>	♦	221	2023009	SERVICE	180,000	1	0,333333	33.33	100.00	60,000	60,000	0,000	0,000	0,000	0,000
<input type="checkbox"/>	♦	222	2023007	SERVICE	120,000	1	0,333333	33.33	33.33	40,000	40,000	0,000	0,000	0,000	0,000
<input type="checkbox"/>	♦	222	2023008	SERVICE	120,000	1	0,333334	33.34	66.67	40,000	40,000	0,000	0,000	0,000	0,000
<input type="checkbox"/>	♦	222	2023009	SERVICE	120,000	1	0,333333	33.33	100.00	40,000	40,000	0,000	0,000	0,000	0,000

Figure 6.58 Revenue Schedule: Display for Selected Contract

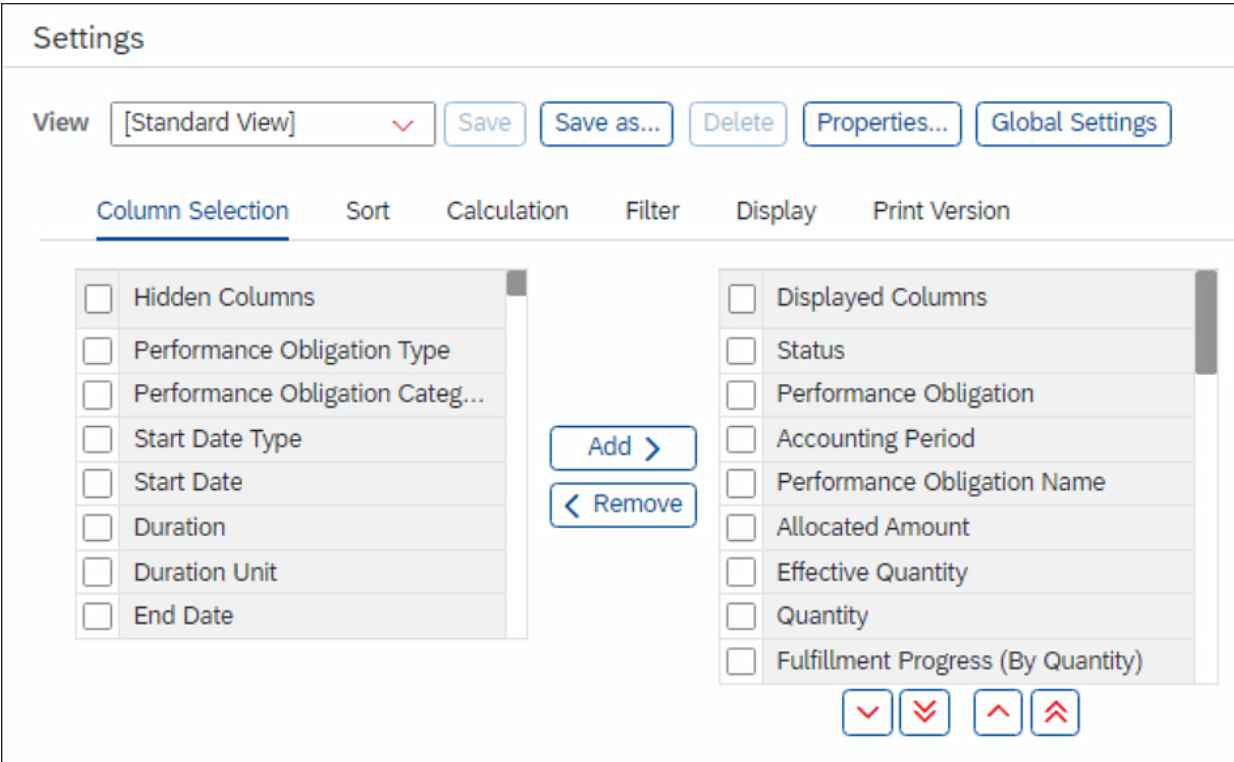


Figure 6.59 Choose the Columns to Be Displayed or Hidden for Revenue Schedule

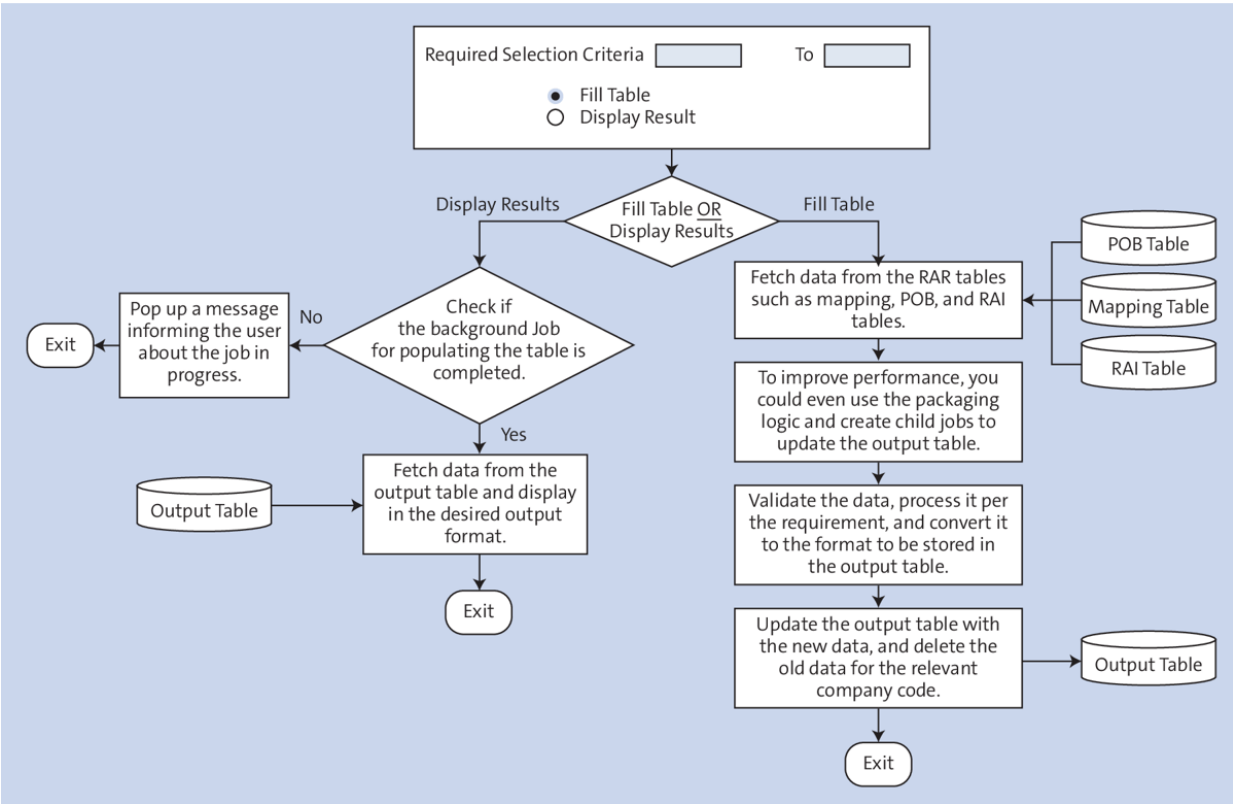





Figure 6.60 Flowchart for Reporting with Staging Tables


Report to reconcile between BRIM and RAR


Timeframe in days


Sender Component to 


Source Item Logical System to 


Company Code to 


Source Document Item Type to 

Header ID to 

Fiscal Year to 

Fiscal Period to 

Performance Obligation to 

Revenue Accounting Contract to 

Compare Orders in BRIM on BL-Version and Timestamp to RAR

Compare Invoices in BRIM with Amount and Quantity in RAR

Find missing Invoices in RAR

Find Invoices with preceding missing Credit notes over Invoices

Compare amount in RAR Posting between IR and FR

Figure 6.61 Sample Selection Screen of Reconciliation Report



Figure 6.62 Push Buttons to Display the Data and Fill the Data to the Table

ZRAT_CREDITN: Display of Entries Found

Search in Table: ZRAT_CREDITN Staging for ZP_RA_ReconciliationCreditN
 Number of hits: 1,000
 Runtime: 0 Maximum no. of hits: 1,000

Insert Column

ID	REVNAC	RAS	RASNDGCOMPDOCUMENTITEM	COMPA	BOOKINGNUMBER	REVE	POSTINGDATE	CAINVOICING	REVENUEPAYER	REVNACCTG	UNIQUEBILLOFLAD	BILLOFL	SOU	PAY	INVOICECHECK
873	ZG	CAII	I22328003887723016OCNN	1000	914683130	BAS	24.11.2022		10860590	10	MAEU914683130	4	1	P	Missing Invoice for Credit Note
871	ZG	CAII	I223320039117382010OPOL	1000	223340973	ODF	28.11.2022		10320508	10	MAEU223340973	2	1	P	Missing Invoice for Credit Note
879	ZG	CAII	I223320039150001007OPOL	1000	223076698	AME	28.11.2022		13294880	10	MAEU223076698	3	1	P	Missing Invoice for Credit Note
438	ZG	CAII	I223330039252203013OCNN	1000	223116458	VP1	29.11.2022		10115142	10	MAEU223116458	3	1	P	Missing Invoice for Credit Note
882	ZG	CAII	I223330039330859012OPOL	1000	1KT755786	OHC	30.11.2022		13491990	10	MAEU1KT755786	3	1	P	Missing Invoice for Credit Note
874	ZG	CAII	I223340039416871007OPOL	1000	609319978	DTI	30.11.2022		14330167	10	MAEU609319978	5	1	P	Missing Invoice for Credit Note
880	ZG	CAII	I223350039519105006OPOL	1000	223134758	ODF	01.12.2022		13142846	10	MAEU223134758	2	1	P	Missing Invoice for Credit Note
872	ZG	CAII	I223350039533119013OCNN	1000	223154756	CP2	01.12.2022		13679322	10	MAEU223154756	2	1	P	Missing Invoice for Credit Note
866	ZG	CAII	I223360039672766003OCNN	1000	223151055	EFF	02.12.2022		13794635	10	MAEU223151055	4	1	P	Missing Invoice for Credit Note
878	ZG	CAII	I223370039835454008OPOL	1000	223325406	OHC	03.12.2022		20175153	10	MAEU223325406	4	1	P	Missing Invoice for Credit Note
867	ZG	CAII	I223390039950581010OCNN	1000	223398948	BAS	05.12.2022		10115518	10	MAEU223398948	13	1	P	Missing Invoice for Credit Note
416	ZG	CAII	I223400040070827006OPOL	1000	914695317	EXP	06.12.2022		13499931	10	MAEU914695317	2	1	P	Missing Invoice for Credit Note
421	ZG	CAII	I223400040113821003OPOL	1000	223398719	EXP	06.12.2022		13620479	10	MAEU223398719	3	1	P	Missing Invoice for Credit Note
422	ZG	CAII	I223410040268281004OPOL	1000	223391423	OHC	07.12.2022		13604024	10	MAEU223391423	2	1	P	Missing Invoice for Credit Note
424	ZG	CAII	I223420040394207003OPOL	1000	223354605	OHC	08.12.2022		17563	10	MAEULSA354605	2	1	P	Missing Invoice for Credit Note
425	ZG	CAII	I223430040486115005OPOL	1000	609324572	OHC	09.12.2022		12605852	10	MAEU609324572	3	1	P	Missing Invoice for Credit Note
442	ZG	CAII	I223430040506541010OCNN	1000	914699843	LSS	08.12.2022		13372024	10	MAEU914699843	2	1	P	Missing Invoice for Credit Note
327	ZG	CAII	I223430040512933002OPOL	1000	223181750	PAE	13.01.2023		14419218	10	MAEU223181750	2	1	P	Missing Invoice for Credit Note
883	ZG	CAII	I223430040567018016OCNN	1000	223253427	BAF	09.12.2022		13674613	10	MAEU223253427	4	1	P	Missing Invoice for Credit Note
884	ZG	CAII	I223430040570015016OCNN	1000	914694779	LSS	09.12.2022		13987104	10	MAEU914694779	4	1	P	Missing Invoice for Credit Note
413	ZG	CAII	I223430040570193012OCNN	1000	914689529	BAF	09.12.2022		13987104	10	MAEU914689529	3	1	P	Missing Invoice for Credit Note
423	ZG	CAII	I223430040578011594IPOL	1000	222977558	DTI	09.12.2022	7500239404	11101482	10	MAEU222977558	2	1	P	Missing Invoice for Credit Note
430	ZG	CAII	I223460040802937837MI	1000	IK0061846	EET	10.12.2022	7500251660	10798777	10	MAEU0061846	3	1	P	Missing Invoice for Credit Note
431	ZG	CAII	I223460040802937890IPOL	1000	IK0062187	OHC	10.12.2022	7500251776	10798777	10	MAEU0062187	3	1	P	Missing Invoice for Credit Note
859	ZG	CAII	I223460040817710001OCNN	1000	223408832	EFF	12.12.2022		13879661	10	MAEU223408832	4	1	P	Missing Invoice for Credit Note

Figure 6.63 Data Stored in the Table by Jobs

Job Overview

Refresh Release Spool Job log Step Job details Application servers

Job overview from: 10.07.2023 at: : :
to: 10.07.2023 at: : :
Selected job names: CREDIT
Selected user names:

Scheduled Released Ready Active Finished Canceled
 Event-Driven Event ID: : :
 ABAP program Program name : :

JobName	Spool	Job CreatedB	Status	Start date	Start T1	End time	Duration(s)	Delay	Executing server	Target Host
<input type="checkbox"/> CREDIT			Active	10.07.2023	07:42:30		47	0	MNR4AP0401_NR4_04	
<input checked="" type="checkbox"/> CREDIT			Finished	10.07.2023	07:41:13	07:42:20		0	MNR4AP0402_NR4_04	
*Summary							114	0		

Figure 6.64 Job Scheduled to Save Entries in the Table

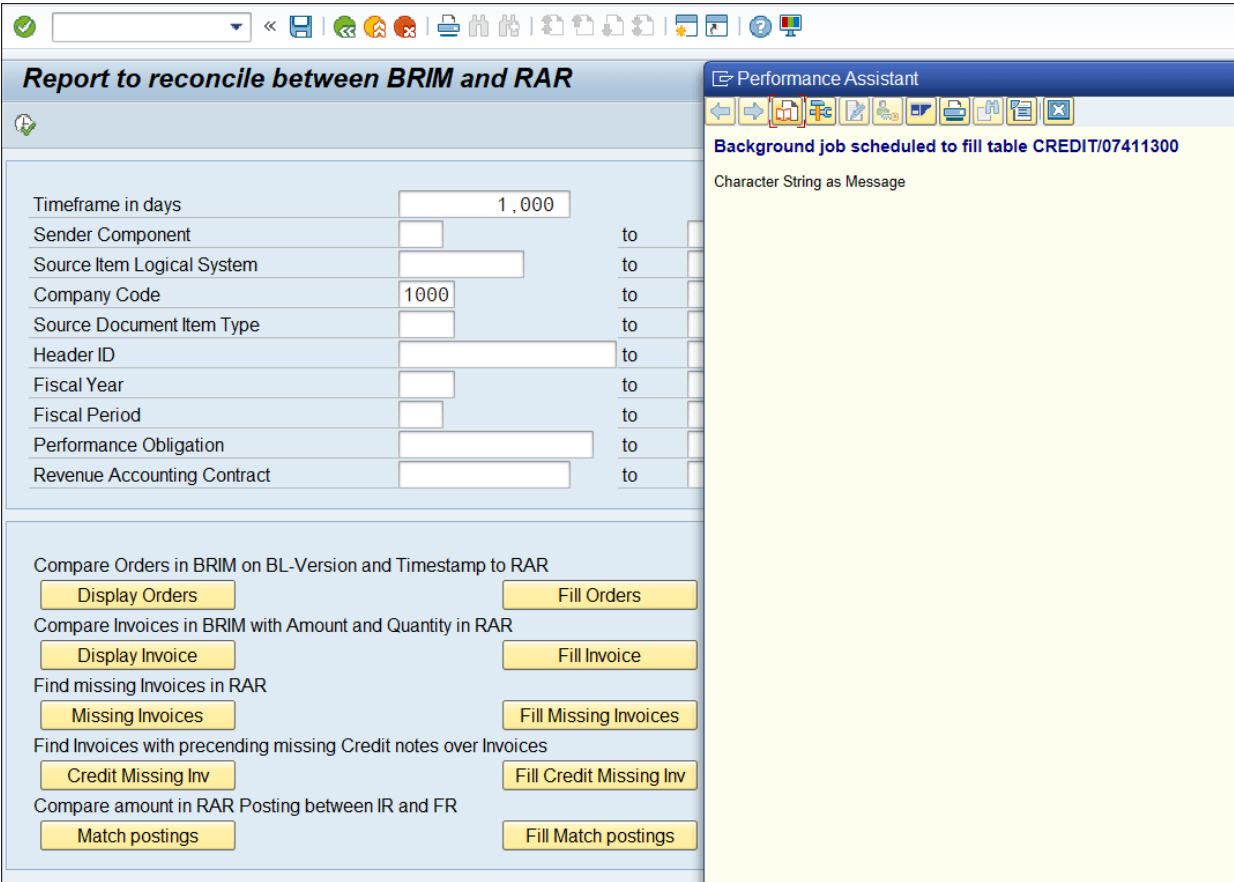


Figure 6.65 Message to Inform User That a Job Is Scheduled to Fill the Table

Reconciliation Reports

Se.	SrcLtmType	Source Item ID	CoCd	ZZ BL UBLI	ZZ Booking	ZZ Charge	Posting Date	ZZ In.	Partner	Item ID	ZZ BL Vers	ZZP/
ZG	CAI	II230400049920878004OCNN	1000	MAEU224963692	224963692	BAS	09.02.2023		13249060	10	2 P	
ZG	CAI	II230400049826867006OCNN	1000	MAEUL23210316	L23210316	BAS	09.02.2023		10503533	10	2 P	
ZG	CAI	II230180046556785006OCNN	1000	MAEU223978122	1KT772446	BAS	20.01.2023		14266801	10	2 P	
ZG	CAI	II230330048747598004OCNN	1000	MAEU224295845	224295845	BAS	02.02.2023		10309596	10	2 P	
ZG	CAI	II230450050633945012OCNN	1000	MAEUBKA011496	BKA011496	BAS	14.02.2023		12530720	10	2 P	
ZG	CAI	II230240047333103007OCNN	1000	MAEU224431774	1KT800334	BAS	20.01.2023		12532781	10	2 P	
ZG	CAI	II230310048457682008OCNN	1000	MAEU224764385	224764385	BAS	02.02.2023		14426573	10	2 P	
ZG	CAI	II230470051144504010OCNN	1000	MAEUL231H0099	L231H0099	BAS	16.02.2023		10503533	10	2 P	
ZG	CAI	II230400049947259006OCNN	1000	MAEU224187453	224187453	BAS	09.02.2023		10164586	10	2 P	
ZG	CAI	II230400049950414012OCNN	1000	MAEU224527950	224527950	BAS	09.02.2023		10164586	10	2 P	
ZG	CAI	II230240047397304026OCNN	1000	MAEU224511393	224334730	BAS	20.01.2023		20046009	10	2 P	
ZG	CAI	II230470051143239004OCNN	1000	MAEUL231H0129	L231H0129	BAS	16.02.2023		10503533	10	2 P	
ZG	CAI	II230400049826917002OCNN	1000	MAEUL23210582	L23210582	BAS	09.02.2023		10503533	10	2 P	
ZG	CAI	II230450050744923012OCNN	1000	MAEU914846875	914846875	BAS	14.02.2023		13624537	10	2 P	
ZG	CAI	II230330048771932040OCNN	1000	MAEU224711601	224637527	BAS	02.02.2023		10311014	10	2 P	
ZG	CAI	II230170046238446006OCNN	1000	MAEU223612838	223612838	BAS	20.01.2023		10115142	10	2 P	
ZG	CAI	II230400049947413008OCNN	1000	MAEU224405487	224405487	BAS	09.02.2023		10164586	10	2 P	
ZG	CAI	II230330048854946012OCNN	1000	MAEU914821912	914821912	BAS	02.02.2023		14235937	10	2 P	
ZG	CAI	II230230047260488006OCNN	1000	MAEU224302116	224302116	BAS	20.01.2023		10782100	10	2 P	
ZG	CAI	II230230047168119018OCNN	1000	MAEU224327029	224327029	BAS	20.01.2023		13813939	10	2 P	
ZG	CAI	II230450050725072016OCNN	1000	MAEU609407217	609407217	BAS	14.02.2023		11128419	10	2 P	
ZG	CAI	II230400049900649014OCNN	1000	MAEU224555181	224555181	BAS	09.02.2023		10196339	10	2 P	
ZG	CAI	II230470051110700010OCNN	1000	MAEU224471940	224471940	BAS	16.02.2023		14428658	10	2 P	
ZG	CAI	II230330048854978010OCNN	1000	MAEU914833606	914833606	BAS	02.02.2023		14235937	10	2 P	
ZG	CAI	II230470051124783016OCNN	1000	MAEU225273155	224709103	BAS	16.02.2023		20260694	10	2 P	
ZG	CAI	II230450050638627014OCNN	1000	MAEU224741432	224741432	BAS	14.02.2023		10318069	10	2 P	

Figure 6.66 Sample Report Output

```
PUBLIC SECTION.
```

```
INTERFACES if_aindp_marker_hdb .
```

Figure 6.67 AMDP Method Declaration

```

TYPES:
  BEGIN OF ltys_ReconciliationPosting,
    Mandt                TYPE mandt,
    CompanyCode          TYPE bukrs,
    AccountingPrinciple TYPE accounting_principle,
    RevnAcctgReconciliationKey TYPE farr_recon_key,
    PerformanceObligation TYPE farr_pob_id,
    ConditionType        TYPE kscha,
    RevnAcctgPostingCategory TYPE farr_post_category,
    DebitCreditCode      TYPE shkzg,
    RevnAcctgPostingUUID TYPE farr_posting_guid,
    ReclassificationIndicator TYPE          reclassind,
    ZRevnAcctgReconciliationKey TYPE farr_recon_key,
    FiscalYear           TYPE gjahr,
    FiscalPeriod         TYPE poper,
    RevnAcctgPostAmtInSlsDocCrcy TYPE farr_amount_tc,
    SalesDocumentCurrency TYPE waers,
    RevnAcctgPostAmtInCoCodeCrcy TYPE farr_amount_lc,
    CompanyCodeCurrency   TYPE hwaer,
    RevnAcctgPostAmtInAddCur1 TYPE farr_amount_lc2,
    AdditionalCurrency1   TYPE hwae2,
    RevnAcctgPostAmtInAddCur2 TYPE farr_amount_lc3,
    AdditionalCurrency2   TYPE hwae3,
    RevenueAccountingContract TYPE farr_contract_id,
    PLAccount             TYPE saknr,
    ConditionIsForStatistics TYPE kstat,
    RAPERformanceObligationType TYPE farr_pob_type,
    DebitCreditCodeSales   TYPE farr_shkzg_va,
  END OF ltys_ReconciliationPosting.

```

Figure 6.68 Types Declaration in the AMDP Class

```

METHODS
  get_RevenueAccountingContract
  AMDP OPTIONS
  READ-ONLY
  CDS SESSION CLIENT iv_clnt
  IMPORTING
    VALUE(iv_clnt)                TYPE mandt
    VALUE(iv_companyCode)         TYPE buks
    VALUE(iv_AccountingPrinciple) TYPE accounting_principle
    VALUE(iv_PostingDate)         TYPE budat
    VALUE(iv_filter)              TYPE ltyv_filter DEFAULT ''
  EXPORTING
    value(lt_view)                type ltyt_RevenueAccountingContract
    value(lt_tabl)                type ltyt_RevenueAccountingContract
    value(lt_all)                 type ltyt_RevenueAccountingContract
    VALUE(et_RevenueAccountingContract) TYPE ltyt_RevenueAccountingContract
    VALUE(et_reconciliationPosting) TYPE ltyt_ReconciliationPosting
    VALUE(et_DistinctContract)    TYPE ltyt_DistinctContract
  RAISING
    cx_amdp_execution_failed.

```

Figure 6.69 Method Declaration Sample

```

.....
Method attributes.
.....
Instantiation: Public
.....

METHOD get_RevenueAccountingContract
  BY DATABASE PROCEDURE FOR HDB LANGUAGE SQLSCRIPT
  OPTIONS
  READ-ONLY
  USING
    ZI_Max_Contract
    ZI_RA_LoadedInvoices
    ZI_RA_ReconciliationPosting
    ZI_RA_PerformanceObligation.

  if :iv_filter = 'TABLE'
  then
    lt_pob      = select dat.* from ZI_RA_PerformanceObligation as dat
                  where dat.mandt = :iv_clnt
                  and dat.RevenueAccountingContract in (select RevenueAccountingContract from ZI_Max_Contract where mandt = :iv_clnt and IdisZ = 'Z' ) ;
    lt_view_raw = select dat.* from ZI_RA_LoadedInvoices as dat
                  where dat.mandt = :iv_clnt
                  and dat.RevenueAccountingContract in (select RevenueAccountingContract from ZI_Max_Contract where mandt = :iv_clnt and IdisZ = 'Z' ) ;
    lt_tabl_raw = select dat.* from ZI_RA_ReconciliationPosting as dat
                  where dat.mandt = :iv_clnt
                  and dat.RevenueAccountingContract in (select RevenueAccountingContract from ZI_Max_Contract where mandt = :iv_clnt and IdisZ = 'Z' ) ;
  else
    -- Get all information from PerformanceObligation as per selection on Contracts
    lt_pob_client = select * from ZI_RA_PerformanceObligation where mandt = :iv_clnt;
    lt_pob       = APPLY_FILTER ( :lt_pob_client , :iv_filter );
    -- Get the POB to Invoice to documents as give it a filter
    lt_view_raw = APPLY_FILTER ( ZI_RA_LoadedInvoices,      :iv_filter );
    lt_tabl_raw = APPLY_FILTER ( ZI_RA_ReconciliationPosting, :iv_filter );
  end if;

```

Figure 6.70 Implementing the AMDP Method

```

TRY.
CALL METHOD go_reconciliationpostings->get_revenueaccountingcontract
EXPORTING
  iv_clnt                = sy-mandt
  iv_companycode         = ls_contract_first-bukrs
  iv_accountingprinciple = gw_prin
  iv_postingdate         = lw_date_type
  iv_filter              = lv_filter
IMPORTING
  et_revenueaccountingcontract = lt_RevAcccontracts
  et_reconciliationposting     = lt_recon_posting
*   et_distinctcontract        = gt_contract
.
CATCH cx_root INTO DATA(lo_x_root).
  DATA(lv_message) = lo_x_root->get_text( ).
ENDTRY.

```

Figure 6.71 Consuming AMDP in the ABAP Program

```
lv_filter = |RevenueAccountingContract between '{ ls_contract_first-contract_id }' and '{ ls_contract_last-contract_id }' |.
```

Figure 6.72 Filter in AMDP

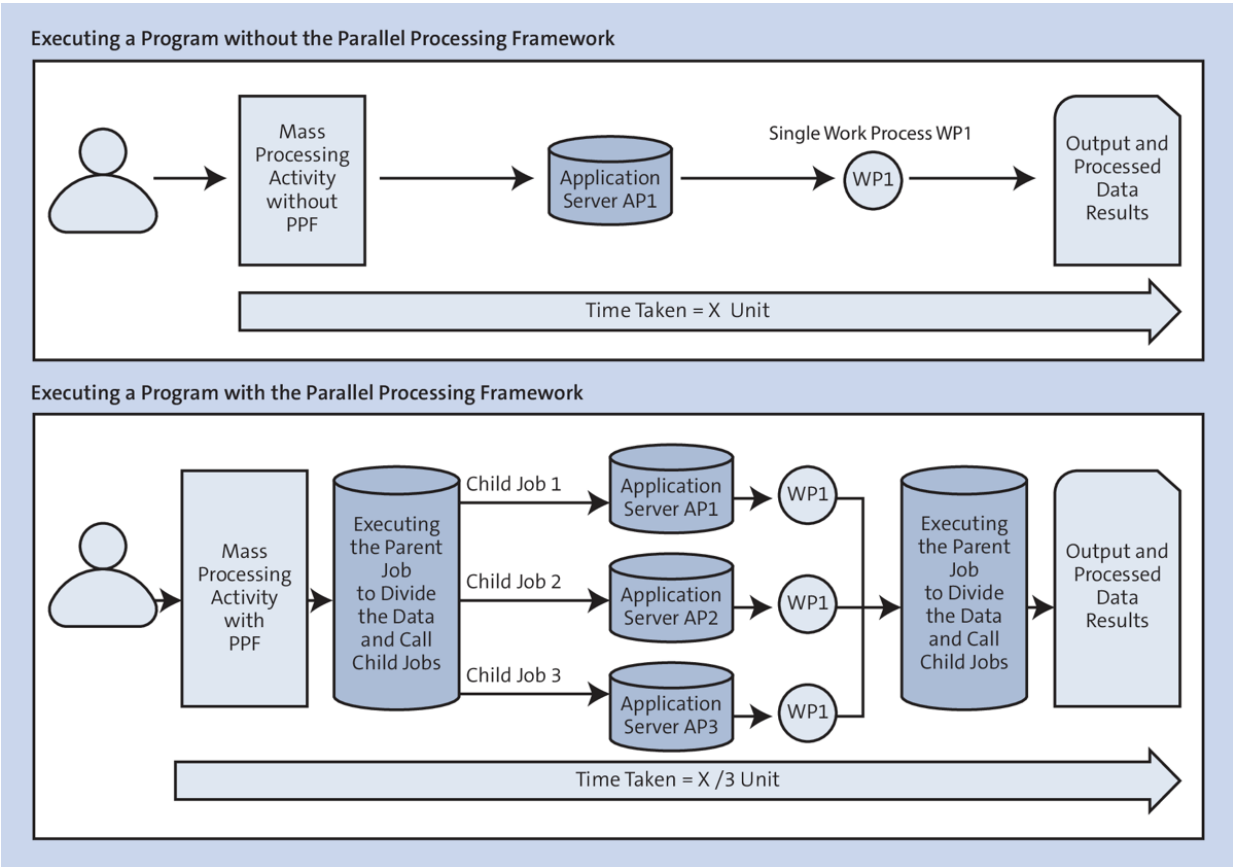


Figure 7.1 Parallel Processing Framework

The screenshot shows the SAP interface for 'AS Instances of SAP System A4H'. At the top, there is a navigation bar with the SAP logo and a title. Below the navigation bar, there is a status bar indicating '1 AS instance(s) started, SSL activated'. The main content is a table with the following columns: Application Server Instance, Host, Instance Services, and State. One instance is listed: 's4srv2022_A4H_22' on host 's4srv2022', with services 'Dialog Batch Update Upd2 Spool ICM', and its state is 'Active'.

Application Server Instance	Host	Instance Services	State
<input type="checkbox"/> s4srv2022_A4H_22	s4srv2022	Dialog Batch Update Upd2 Spool ICM	Active

Figure 7.2 Instances Available in a System

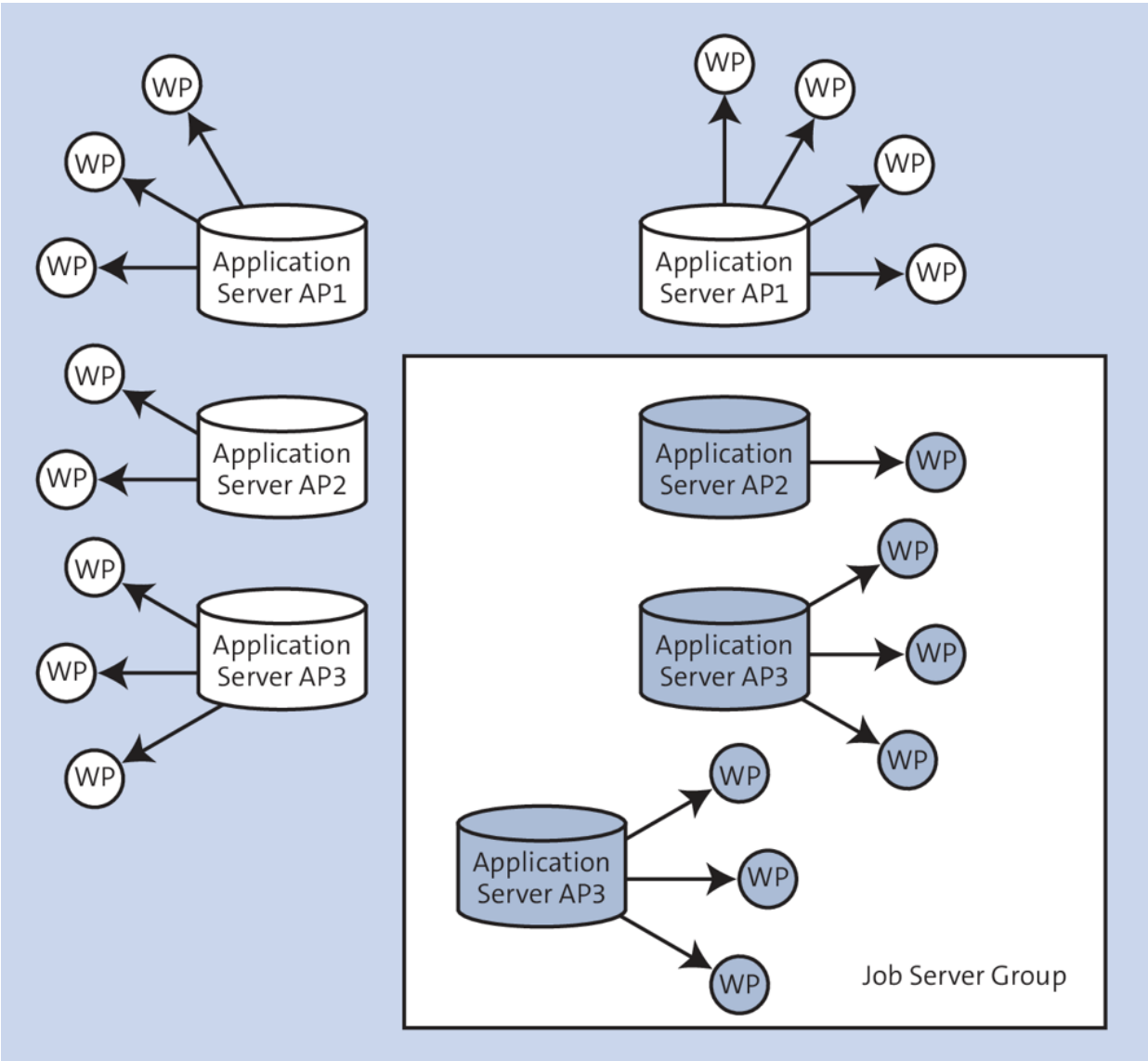


Figure 7.3 Job Server Group Illustration

Display control object list

Job Server Groups

Background Controller List

Server	Control Object Description	Trace level
s4srv2022_A4...	Event-Driven Scheduler	0 (No Trace)
	Job Repository Job Gener...	0 (No Trace)
	Job Starter	0 (No Trace)
	Start of Subsequent Jobs	0 (No Trace)
	Time-Driven Scheduler	0 (No Trace)

Server: s4srv2022_A4H_22

Object: Event-Driven Scheduler

Object Trace Health check

Activation

- Activate
- Deactivate

Figure 7.4 Job Server Group: Initial Screen

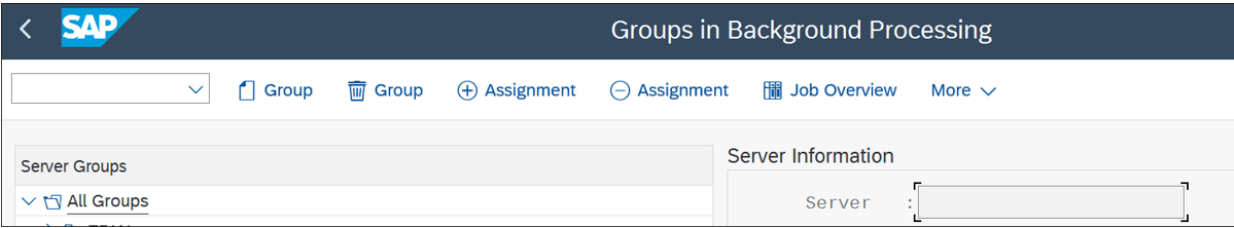


Figure 7.5 Job Server Group: Create Group

The image shows a web application window titled "Assignment of Instance to Background Server Group". The window has a dark blue header with a hamburger menu icon on the left and a close button (X) on the right. Below the header, the text "Create a New Server Group:" is displayed. A text input field labeled "Group Name:" contains the text "Job_server1". Below the input field, the text "Enter Server Group Name W/O Special Characters" is displayed. At the bottom right of the form, there are two buttons: "Continue" with a green checkmark icon and "Cancel" with a red X icon. On the right side of the window, a dark blue sidebar is partially visible with the text "Processing" and a "More" dropdown menu.

Figure 7.6 Job Server Group: Group Name

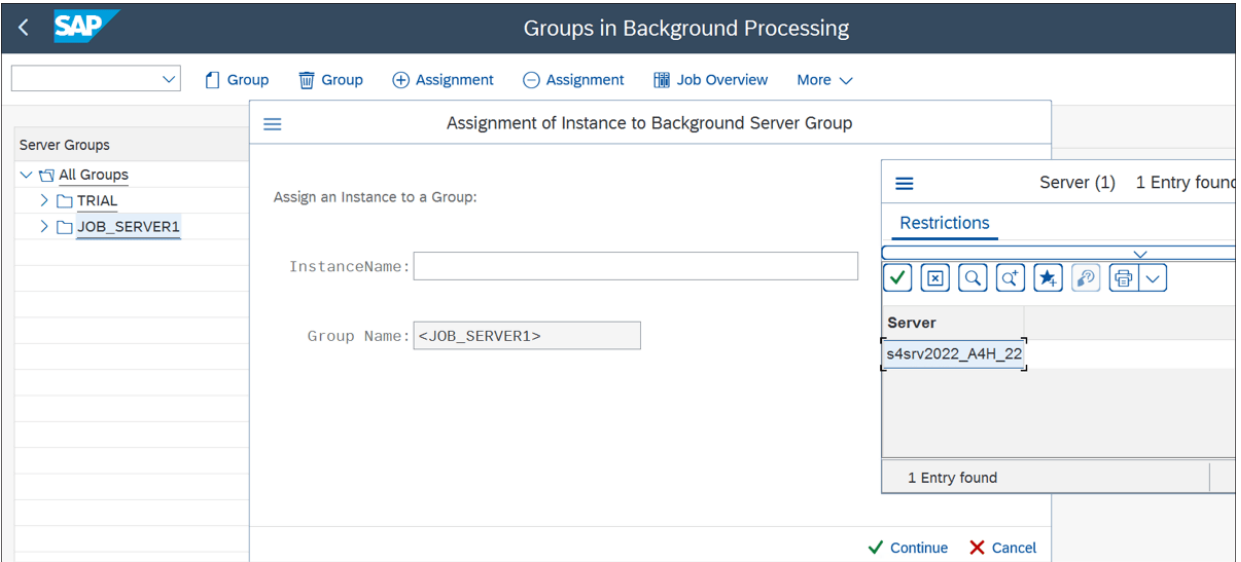


Figure 7.7 Job Server Group: Assign an Instance

The screenshot displays the SAP 'Groups in Background Processing' interface. The top navigation bar includes the SAP logo and the title 'Groups in Background Processing'. Below the navigation bar, there are several action buttons: a search dropdown, 'Group' (add), 'Group' (delete), '+ Assignment', '- Assignment', 'Job Overview', and 'More'. The main content area is divided into two sections: 'Server Groups' on the left and 'Server Information' on the right. In the 'Server Groups' section, a tree view shows 'All Groups' expanded to 'JOB_SERVER1', which contains the server 's4srv2022_A4H_22'. The 'Server Information' section displays details for the selected server: Server: s4srv2022_A4H_22, Host: s4srv2022, Status: Active, WP Types: Dialog Batch Update Update2 Spool ICM, Back. WP: 6, and Back. WP Class A: 0.

Server Information	
Server :	s4srv2022_A4H_22
Host :	s4srv2022
Status :	Active
WP Types :	Dialog Batch Update Update2 Spool ICM
Back. WP :	6
Back. WP Class A :	0

Figure 7.8 Job Server Group: Added to List

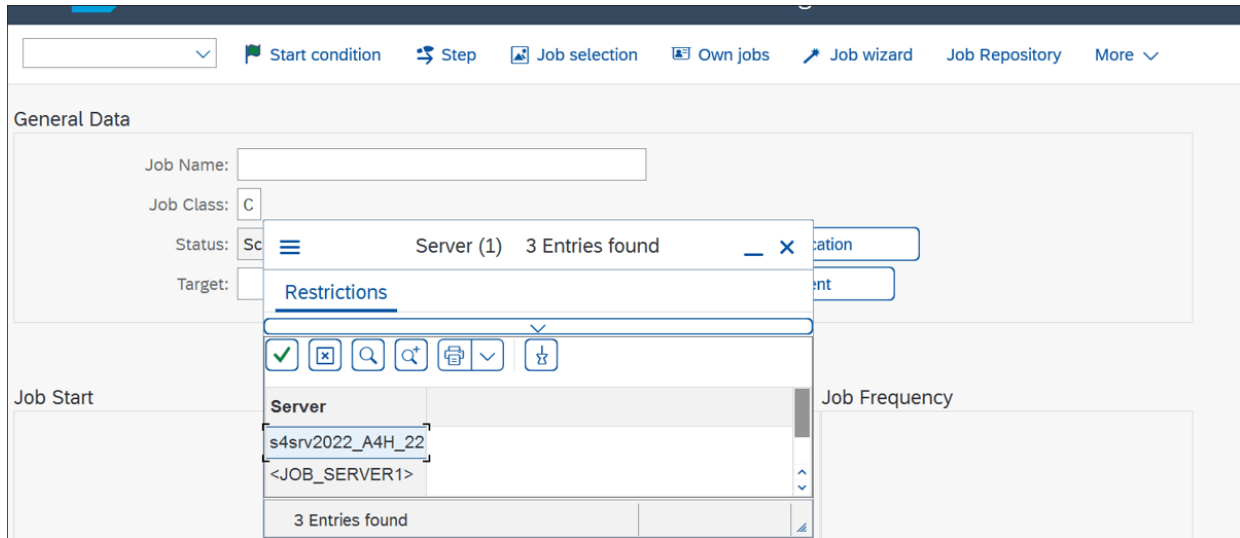


Figure 7.9 Job Server Group: Entry Found

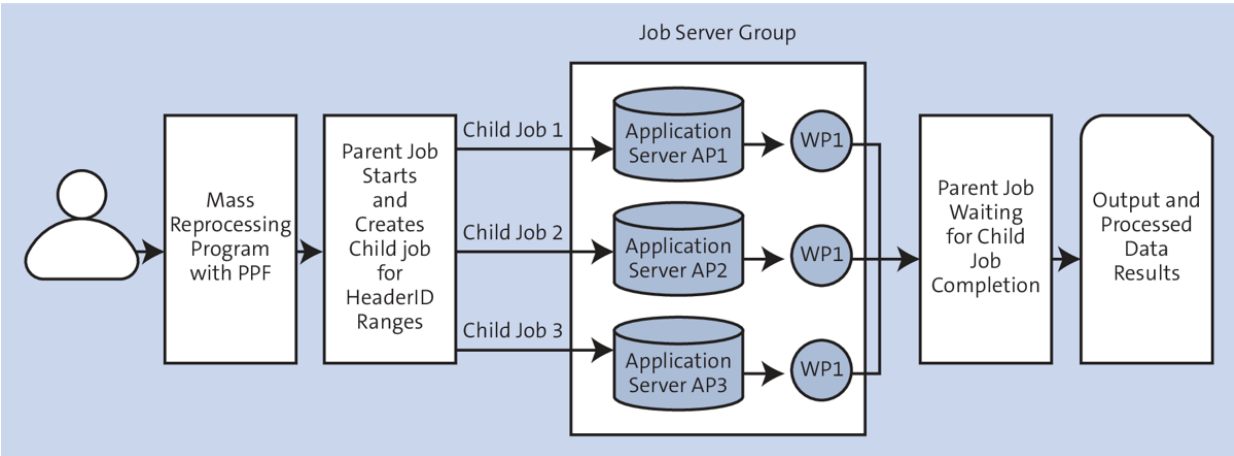


Figure 7.10 Parent and Child Jobs

```

7) Function ZRAR_REPROCESS_CORE.
8) Perform initialize."Initialize global variables. Assign the importing variable to global variables and internal tables.
9) if It_HeaderID[] is NOT INITIAL .
10)   READ TABLE It_HeaderID INTO DATA(ls_headerId) INDEX 1. "Check if headerId is passed as a range or stack of multiple
11)   " single values.
12)
13) If LS_HEADERID-option = 'EQ' ."the sending program or the main program has sent multiple single values of HeaderID
14)   "so move it as is it to the global internal table for header_id)
15)   gt_HeaderID_global[] = It_HeaderID[].
16) endif.
17) If LS_HEADERID-option <> 'EQ'."(the sending program has sent ranges of HeaderID)
18)   Perform Get_headerid using It_HeaderID changing GT_HeaderID_global.
19)   " ( the same selection logic will be applied for fetching the header_id
20)   " as in the main program but it will be for a smaller range value
21)   " which is sent from the parent program and as available in the ranges )
22)
23) Endif.
24) Endif.
25) Sort GT_HeaderID_Global[].
26) DESCRIBE TABLE GT_HeaderID_Global[] LINES GW_TOTAL.
27) gw_min = 1. gw_max = p_pack. "Declare GW_MIN type integer and GW_MAX type integer. gw_min = 1 and gw_max = p_pack.
28)   " ( Where P_PACK is the package size )
29) While GW_MIN <= GW_TOTAL.
30)   APPEND LINES OF gt_headerid_global FROM gw_min TO gw_max TO gt_headerid_pack().
31)   " (The internal table gt_headerid_pack() will now have the number
32)   " of records equal to package size.)
33)   Perform Processing." In this perform processing we have the core processing logic
34)   " and this will be called for each package ).
35)   gw_min = gw_max + 1. gw_max = gw_max + p_pack.
36) Endwhile.
37) Endfunction.

```

Figure 7.11 Pseudocode for the Function Module of Custom PPF

Save as Variant... More

Company Code: * 3000

Accounting Principle:

Header ID: to:

Date:

Package Size: 1.000

No of Intervals: 1

Schedule Jobs:

Figure 7.12 Selection Screen for the Example

```

8)  ⊞ If P_MAIN = 'X'. "indicates that this is parent processing
9)  ⊞ Select the HeaderIDs as per the selection criteria into table gt_header_id_global[].
10) ⊞ Endif.
11)
12) ⊞ If p_main = space or p_interval = 1 or schedule_Jobs = Space. "this condition check indicates that the processing
13) ⊞                                     "has come via child job or the user wants to run the
14) ⊞                                     "program as a single execution as P_INTERVAL = 1 and
15) ⊞                                     "no child jobs to be created as Schedule Jobs = Space
16) ⊞ PERFORM processing_package. "Where we have the call to the function module 'ZRAR_REPROCESS_CORE' for processing.
17) ⊞ Else.
18) ⊞ PERFORM group_child_jobs.  "To schedule the child jobs, we need to calculate how many child jobs need to be
19) ⊞                                     "scheduled. This is done by the No Of Intervals parameter on the screen. This will
20) ⊞                                     "tell you the number of child jobs the user is expecting to schedule.
21) ⊞ DESCRIBE TABLE gt_header_id_global LINES lv_total. "get the total number of HeaderIDs selected
22) ⊞ gw_count = ceil( lv_total / p_intvl ). "GW_COUNT Gives the number of jobs that will be scheduled
23) ⊞ gw_min = 1 . gw_max = gw_count.          "Declare two variables gw_min and GW_MAX and initialize them as shown.
24) ⊞
25) ⊞ WHILE gw_min LE lv_total .
26) ⊞   Read table GT_HEADER_ID_GLOBAL into DATA(ls_min) index gw_min.
27) ⊞   Read table GT_HEADER_ID_GLOBAL into DATA(ls_max) index gw_max.
28) ⊞   " Populate the select-option SO_HDID[] which is a hidden select_option on screen with the range of
29) ⊞   " HeaderID between GW_MIN and GW_MAX and append it, so SO_HDID[] will have the HEADERID or
30) ⊞   " range of headerIDs for the child job then populate all other selection criteria and make
31) ⊞   " sure we clear the P_MAIN and prepare the internal table IT_VARI which is the internal table
32) ⊞   " to be used for submitting to the same program with all the selection screen parameters
33) ⊞   " and select_options and all other selection screen details.
34) ⊞   " We get the range of HeaderIDs for the current child job. Schedule the child job.
35) ⊞
36) ⊞ Call Function 'JOB_OPEN'.
37) ⊞ SUBMIT (sy-repid) WITH SELECTION-TABLE it_vari VIA JOB lv_taskname
38) ⊞   NUMBER lv_jobcount AND RETURN. We submit to the same program.
39) ⊞ Call function 'JOB_CLOSE'. "we will have to provide the Target Server group which we created
40) ⊞   " TARGETG GROUP = 'Job_Server1'. This is where we use the Job server group that
41) ⊞   "we created above to gro up the application server instances. This helps to
42) ⊞   "distribute the jobs across different application server instances.
43) ⊞ call function 'SHOW_JOBSTATE'. " this is to check so that the parent job checks the completion of all the
44) ⊞   "child jobs and then exits only when all the child jobs processing is completed.
45) ⊞   "Then we again continue with the next set of headerId ranges to create next child job
46) ⊞   "by incrementing the GW_MIN and GW_MAX.
47) ⊞   GW_MIN = GW_MAX . GW_MAX = gw_max + gw_count .
48) ⊞
49) ⊞ ENDWHILE.

```

Figure 7.13 Pseudocode for the Main Program of the Custom PPF

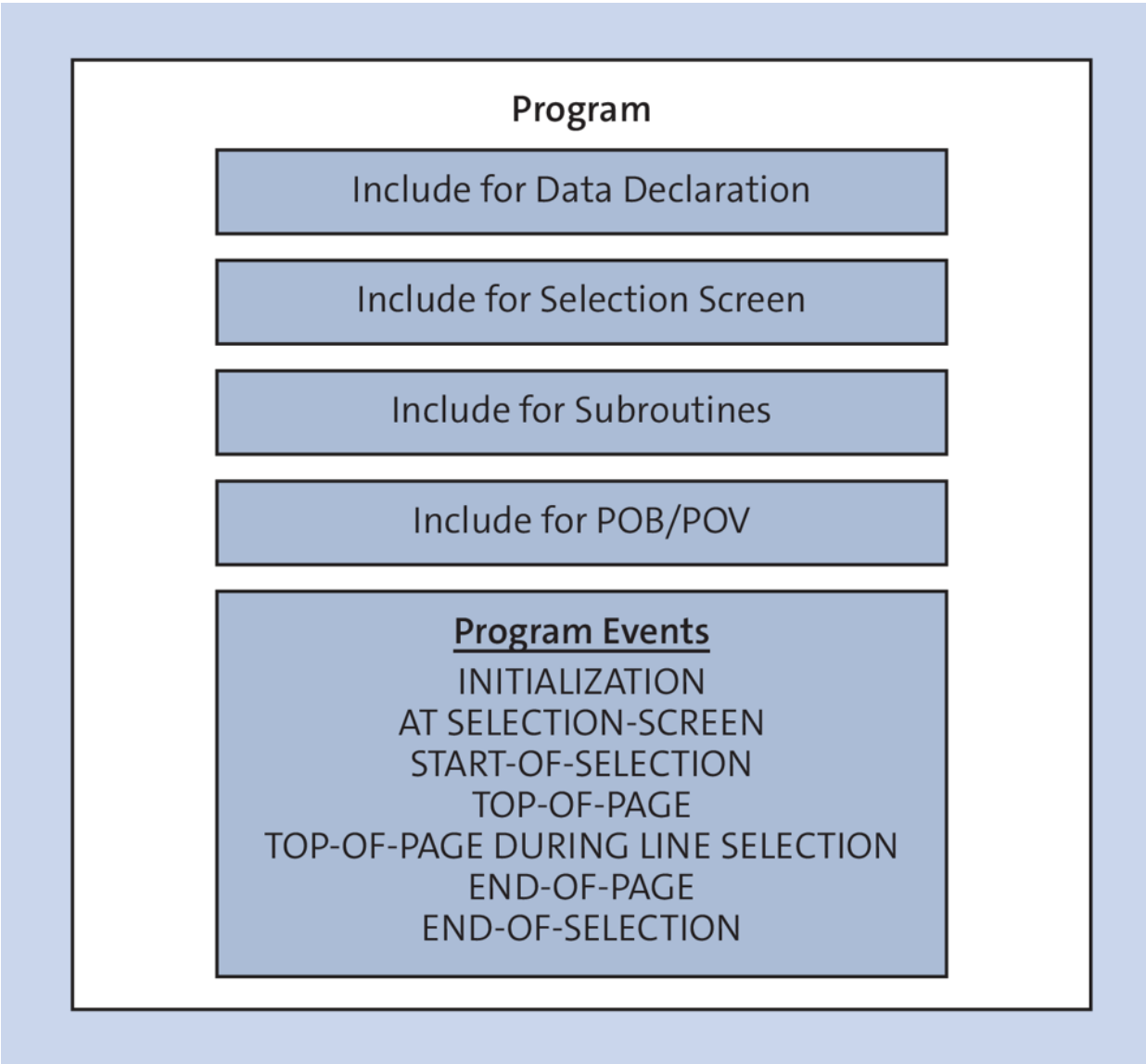


Figure 7.14 Modularization in ABAP Programs

```
-----*
                TOP Include                                *
-----*
INCLUDE      /ora_reprocess_top IF FOUND. " include for data declaration
-----*
                Include For Selection screen              *
-----*
INCLUDE      /ora_reprocess_scr IF FOUND. " Include for selection screen
-----*
                Include For Classes                      *
-----*
INCLUDE      /ora_reprocess_cls IF FOUND. " Include for classes
-----*
                Include For Subroutine                   *
-----*
INCLUDE      /ora_reprocess_f01 IF FOUND. " Include for subroutine
-----*
                I N I T I A L I Z A T I O N              *
-----*
```

Figure 7.15 Modularization

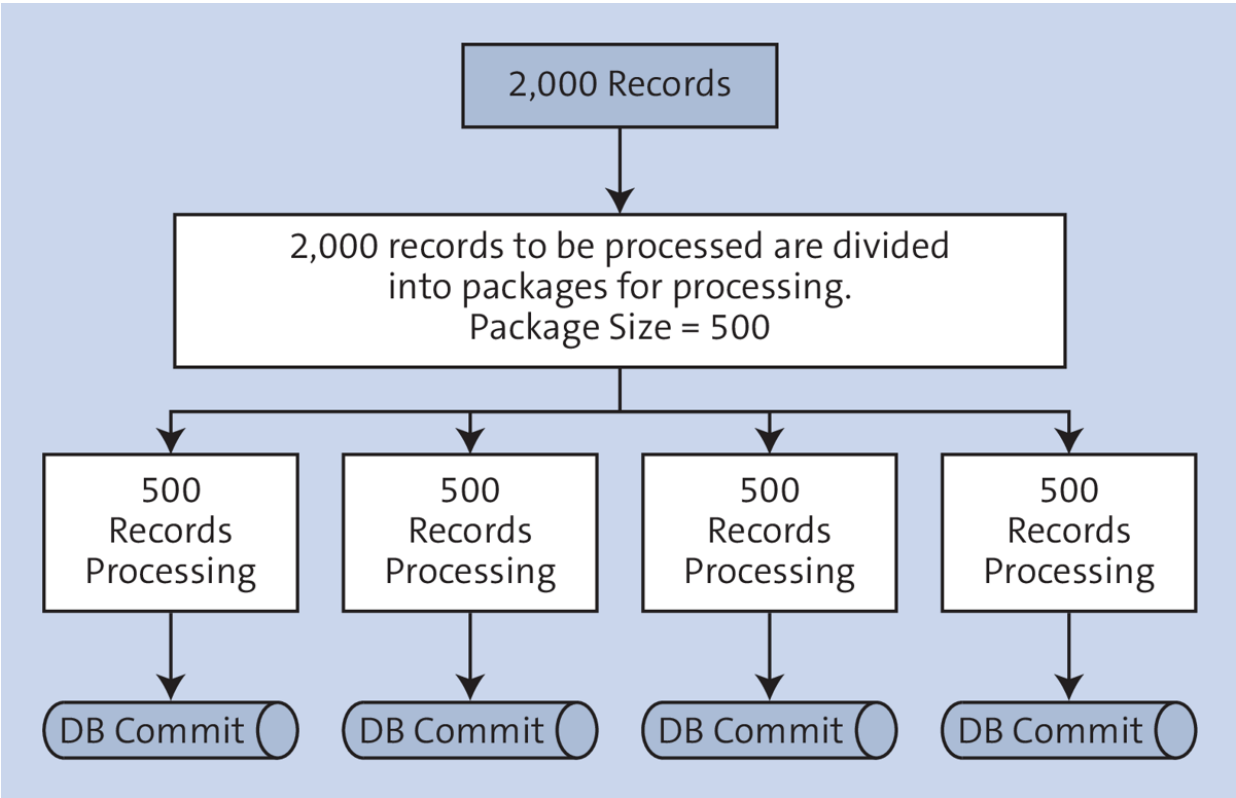


Figure 7.16 Package and Commit

Class/Interface: Implemented / Active

Properties Interfaces Friends Attributes **Methods** Events Types Aliases

Parameters Exceptions Sourcecode [Icons]

Method	Level	Visibility	Me...	Description
INSERT_RECON_KEY	Instance Method	Public		Insert Recon Key
QUERY_RECON_KEY	Instance Method	Public		Query recon key by search criteria
QUERY_RECON_KEY_IL_CLEANUP	Instance Method	Public		Query recon key by contract list for initial load cleanup
QUERY_RECON_KEY_INCL_TM	Instance Method	Public		Optimized reconciliation key DB access
QUERY_RECON_KEY_NEW	Instance Method	Public		Optimized reconciliation key DB access
REFRESH_BUFFER	Instance Method	Public		Refresh the Recon Key buffer
REFRESH_BUFFER_FOR_CONTRACTS	Instance Method	Public		Refresh the Recon Key buffer for Contracts
SAVE_ALL_CONTRACTS_TO_DB	Instance Method	Public		Save all recon key to DB by buffer table
SAVE_CONTRACTS_TO_DB	Instance Method	Public		Save several contracts to DB
SAVE_TO_DB	Instance Method	Public		Save all recon key to DB by buffer table
UPDATE_LIAB_ON_KEY	Instance Method	Public		Update liability on key

Figure 7.17 Class for Reconciliation Key-Related Methods

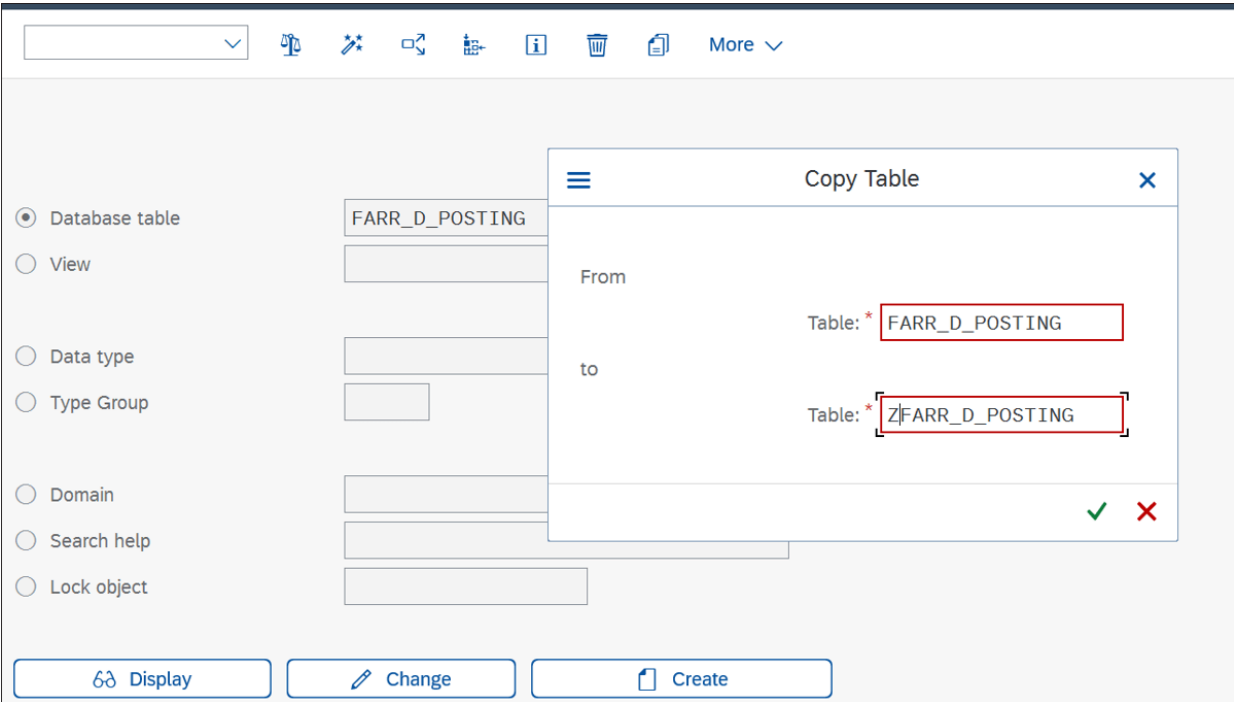


Figure 7.18 Creating Table ZFARR_D_POSTING

11	RECLASSIND	11	X	RECLASSIND	CHAR01	CHAR	1
12	ZRECON_KEY	12	X	FARR_RECON_KEY	FARR_RECON_KEY	CHAR	14

Figure 7.19 Additional Custom Fields in Table ZFARR_D_POSTING

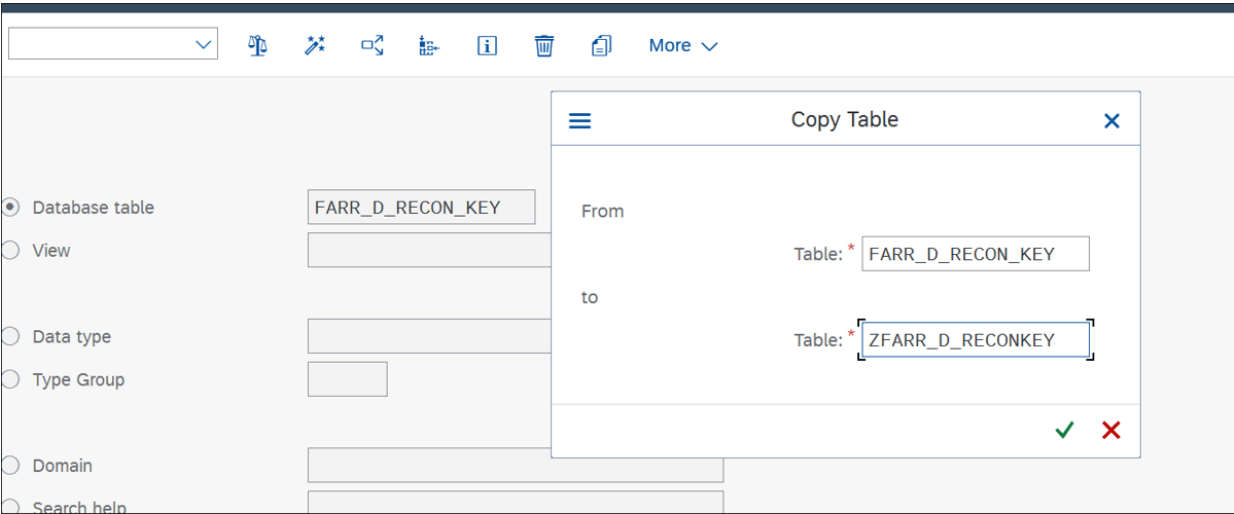


Figure 7.20 Creating Table ZFARR_D_RECONKEY

Row	Field name	Position	Key	Data element	Domain	Datatype	Length	Lowercase	Domain text
1	CLIENT	1	X	MANDT	MANDT	CLNT	3		Client
2	COMPANY_CODE	2	X	BUKRS	BUKRS	CHAR	4		Company Code
3	ACCT_PRINCIPLE	3	X	ACCOUNTING_PRINCIPLE	ACCOUNTING_PRINCIPLE	CHAR	4		Accounting Principle
4	POST_CAT	4	X	ZFARR_POST_CATEGORY	ZFARR_POST_CATEGORY	CHAR	2		Category for Posting Document
5	GLSOURCE	5	X	SAKNR	SAKNR	CHAR	10		G/L Account Number
6	GLTARGET	6		SAKNR	SAKNR	CHAR	10		G/L Account Number

Figure 7.21 General Ledger Mapping Table Structure

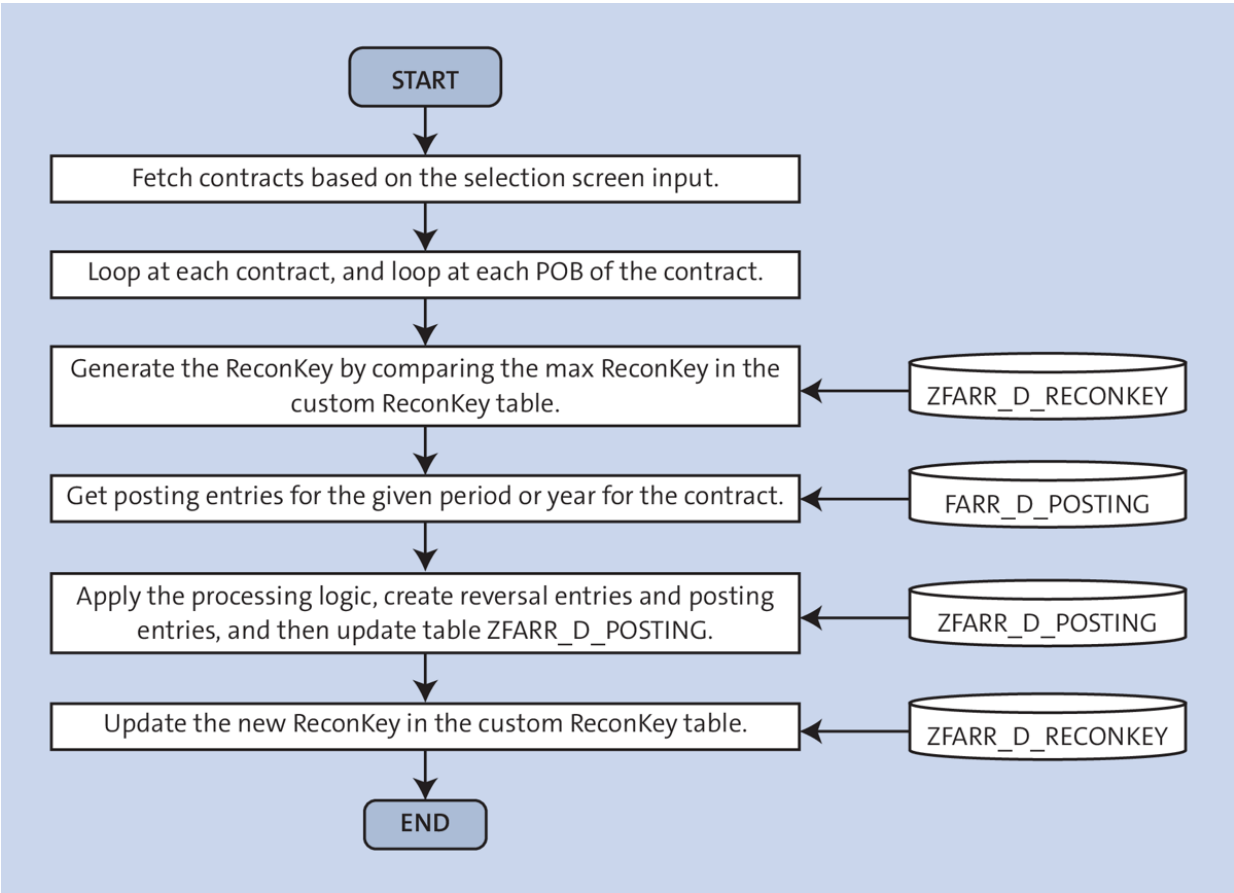


Figure 7.22 Flow Diagram for the Reclassification Program

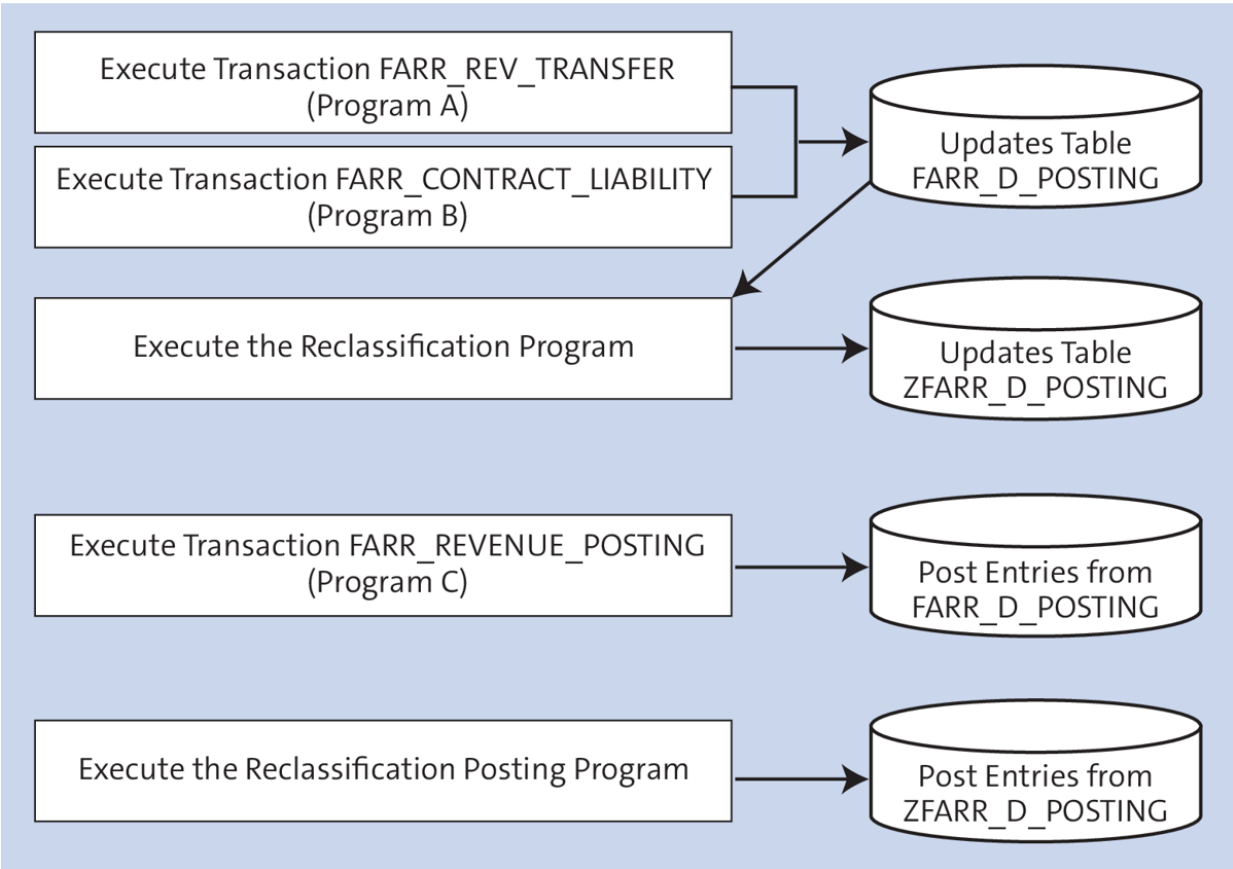


Figure 7.23 Posting Flow with the Custom Posting Table

Message Maintenance: Display Message (Compressed)

Message class: FARR_INFLIGHT_CHECK Activ.

Attributes Messages

No.	Message Short Text	Self-Explanatory	Last Change
<input type="checkbox"/> 000	Inflight check: Error &1&2&3&4 found before save to database	<input checked="" type="checkbox"/>	SAP
<input type="checkbox"/> 001	C01: Allocation error found for contract &1	<input type="checkbox"/>	SAP
<input type="checkbox"/> 002	C02: Allocated amount &3 <> Rev schedule &4, Contract &1/POB &2	<input type="checkbox"/>	SAP
<input type="checkbox"/> 003	C03: Sum(Fulfill Qty) - POB effect Qty = &3, Contract &1/POB &2	<input type="checkbox"/>	SAP
<input type="checkbox"/> 004	C04: Fulfill Qty &2 <> DEFITEM Qty &3, POB &1	<input type="checkbox"/>	SAP
<input type="checkbox"/> 005	C0501: No latest flag found, Contract &1/POB &2/Cond &3	<input type="checkbox"/>	SAP
<input type="checkbox"/> 006	C0502: Multi latest flags found,Contract &1/POB &2/Cond &3	<input type="checkbox"/>	SAP
<input type="checkbox"/> 007	C06: DEFITEM rev <> POSTING rev,Contract &1/POB &2/Cond &3	<input type="checkbox"/>	SAP
<input type="checkbox"/> 008	C07: DEFITEM inv <> POSTING inv,Contract &1/POB &2/Cond &3	<input type="checkbox"/>	SAP

Figure 7.24 Message Class for Inflight Checks

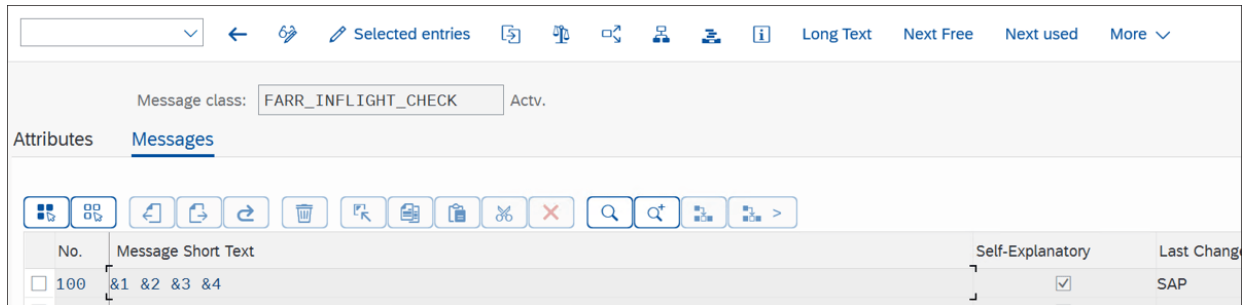


Figure 7.25 Message Class for Inflight Checks (Cont.)

SAP Class Builder: Display Class CL_FARR_DATA_EXTENDED_CHECK

Class/Interface: CL_FARR_DATA_EXTENDED_CHECK Implemented / Active

Properties Interfaces Friends Attributes **Methods** Events Types Aliases

Parameters Exceptions Sourcecode [Icons] Filter

Method	Level	Visibility	Me...	Description
CHECK_PAOBJNR_EXIST	Instance Method	Protected		Check PAOBJNR exist in CE4XXX & CE4XXX_ACCT
CHECK_SPEC_INDICATOR	Instance Method	Protected		Check special indicator per single POB
COMPARE_AMOUNTS	Instance Method	Protected		Compare the data in 2 tables
INIT_FOR_CHECK	Instance Method	Protected		Initialize internal buffers at the beginning of CHECK
INIT_NO_CHECK_SETTING	Instance Method	Protected		Initialize to set which error category will be switched off
PROCESS_INPUT	Instance Method	Protected		Constructor
RAISE_EXCEPTION_WHEN_ERR_FOUND	Instance Method	Protected		Raise Exception if Error found in Inflight Check
REFRESH_BUFFER_TABLES	Instance Method	Protected		Refresh all buffer tables
GET_RECON_KEY	Instance Method	Private		

Figure 7.26 Standard Class CL_FARR_DATA_EXTENDED_CHECK for Inflight Checks

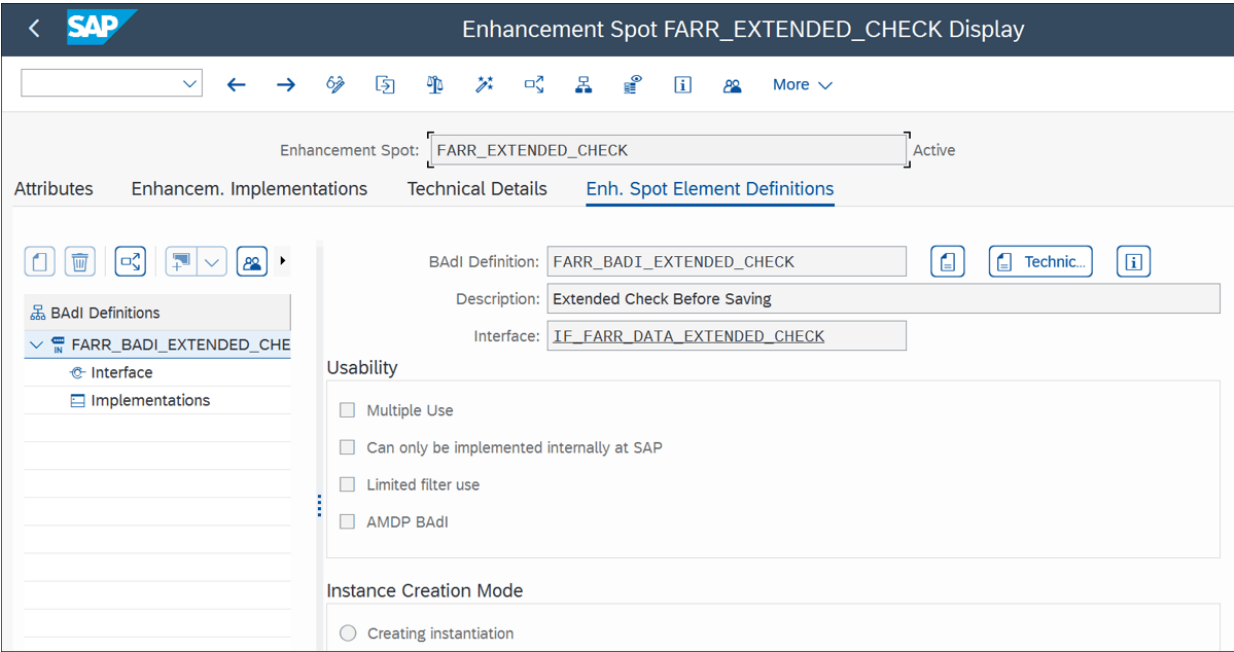


Figure 7.27 BAdI: FARR_EXTENDED_CHECKS

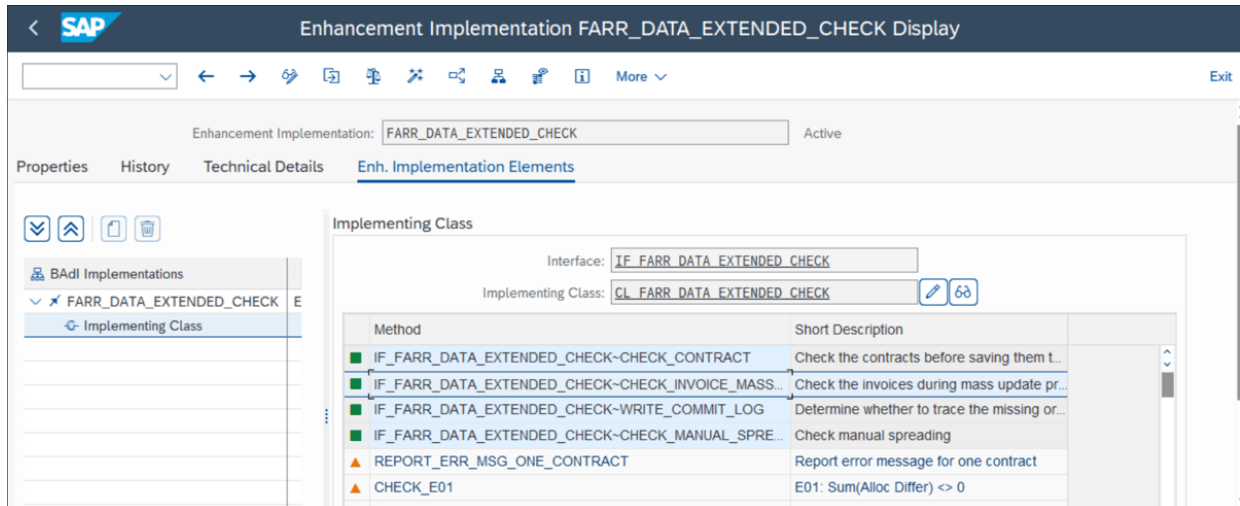


Figure 7.28 Methods of Class CL_FARR_DATA_EXTENDED_CHECK

Method: INIT_NO_CHECK_SETTING active

```
1 METHOD init_no_check_setting.
2 * If you want to switch off some checks of some error category,
3 * jut put ABAP_TRUE to the corresponding fields of MS_NO_CHECK
4 * For example: ms_no_check-no_check_e01 = abap_true.
5 |
6 * For the first Note 2456711 of inflight check,
7 * only enable the check for following categories
8 * E01, E02, E03 and E09
9
10 " do not check for E01 if transition is going to be performed after the migration
11 IF mv_initial_load = if_farrc_contr_mgmt=>co_initial_load_new_principle.
12     ms_no_check-no_check_e01 = abap_true.
13 ENDIF.
14
15 IF mv_initial_load IS NOT INITIAL.
16 * During initial load and transition change, we could not check E02
17 * The reason is engine will accept the revenue from legacy system,
18 * The revenue re-calculation will be done after switch the acct principle to productive move
19     ms_no_check-no_check_e02 = abap_true.
20 ENDIF.
21
```

Figure 7.29 Switch On/Off Inflight Checks

SAP Class Builder Class CL_FARR_DATA_EXTENDED_CHECK Display

Ty.	Parameter	Typing	Description
	IO_MSG_HANDLER	TYPE REF TO CL_FARR_MESSAGE_HANDLER	FARR message handler
	IS_CONTRACT_DATA_BUFFER	TYPE FARR_S_CONTRACT_DATA_BUFFER OPTIONAL	One contract buffer data

Method: IF_FARR_DATA_EXTENDED_CHECK~CHECK_CONTRACT active

```

34     its_manl_chng_data      = its_manl_chng_data
35     its_change_type_buffer = its_change_type_buffer
36     its_change_type_old   = its_change_type_old
37     it_cond_type_data_buffer = it_cond_type_data_buffer
38     iv_fiscal_year        = iv_fiscal_year
39     iv_period              = iv_period.
40
41     * Initialize buffers at the beginning of check
42     init_for_check( ).
43
44     * Call all of the checks
45     IF ms_no_check-no_check_e01 IS INITIAL.
46         check_e01( ). " Allocation error, Sum(CORR) <> 0
47     ENDIF.
48
49     IF ms_no_check-no_check_e02 IS INITIAL.
50         check_e02( ).
51     ENDIF.
52
53     IF ms_no_check-no_check_e03 IS INITIAL.

```

Figure 7.30 Method IF_FARR_DATA_EXTENDED_CHECK~CHECK_CONTRACT

Transparent Table: **FARR_D_CONS** Active
 Short Description: RA - consistency check / contract data

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		Client
<input type="checkbox"/> COMPANY_CODE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	BUKRS	CHAR	4	0		Company Code
<input type="checkbox"/> CONTRACT_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_CONTRACT_ID	CHAR	14	0		Revenue Accounting Contract ID
<input type="checkbox"/> POB_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	FARR_POB_ID	CHAR	16	0		Performance Obligation ID
<input type="checkbox"/> _INCLUDE	<input type="checkbox"/>	<input type="checkbox"/>	FARR_S_CONS	STRU	0	0		RA - consistency check / contract data
<input type="checkbox"/> ACCT_PRINCIPLE	<input type="checkbox"/>	<input type="checkbox"/>	ACCOUNTING_PRINCIP	CHAR	4	0		Accounting Principle

Figure 7.31 Consistency Check Log Table

Transparent Table: FARR_D_CONS Active

Short Description: RA - consistency check / contract data

Attributes Delivery and Maintenance **Fields** Input Help/Check Currency/Quantity Fields Indexes

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
<input type="checkbox"/> ERR02	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR02	CHAR		1	0	0 RA - Data Validation/ Error Category 02
<input type="checkbox"/> ERR03	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR03	CHAR		1	0	0 RA - Data Validation/ Error Category 03
<input type="checkbox"/> ERR04	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR04	CHAR		1	0	0 RA - Data Validation/ Error Category 04
<input type="checkbox"/> ERR05	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR05	CHAR		1	0	0 RA - Data Validation/ Error Category 05
<input type="checkbox"/> ERR06	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR06	CHAR		1	0	0 RA - Data Validation/ Error Category 06
<input type="checkbox"/> ERR07	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR07	CHAR		1	0	0 RA - Data Validation/ Error Category 07
<input type="checkbox"/> ERR08	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR08	CHAR		1	0	0 RA - Data Validation/ Error Category 08
<input type="checkbox"/> ERR09	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR09	CHAR		1	0	0 RA - Data Validation/ Error Category 09
<input type="checkbox"/> ERR10	<input type="checkbox"/>	<input type="checkbox"/>	FARR_ERR10	CHAR		1	0	0 RA - Data Validation/ Error Category 10

Figure 7.32 Error Categories for Data Validation

< **SAP** Data Validation - Revenue Contracts

⌵ ⌂ ⓘ More ⌵

Selection Data

Accounting Principle: ⓘ to: ⓘ

Company Code: to: ⓘ

Excluding Completed Contracts:

Technical Parameters

Block Size For Mass Selection:

Dialog Mode:


Synchronous Call:

Settings for Application Log

External ID:




Problem Class: ⌵

Figure 7.33 Data Validation for Revenue Accounting Contracts

 Data Validation Monitor

Save as Variant... More

Contract

Accounting Principle:	<input type="text" value="IFRS"/>	to:	<input type="text"/>	
Company Code:	<input type="text"/>	to:	<input type="text"/>	
Revenue Accounting Contract:	<input type="text"/>	to:	<input type="text"/>	

Processing

Read data online:

Read data from error table:

Save result to error table:

Technical Parameter

Max. batch size of Contracts:

Max. no. of POBs:

Figure 7.34 Data Validation Monitor Selection Screen

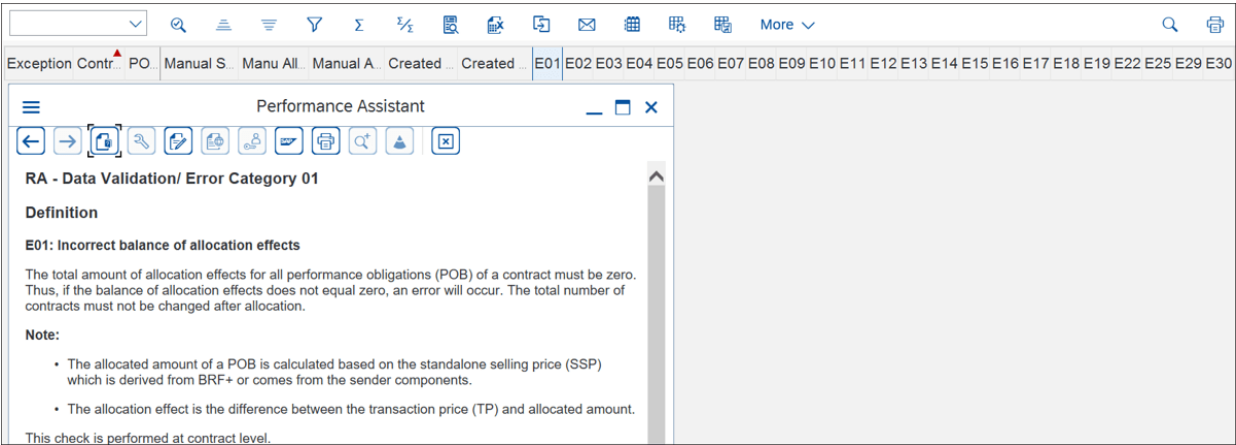


Figure 7.35 Error Category Details from the Consistency Check Monitor

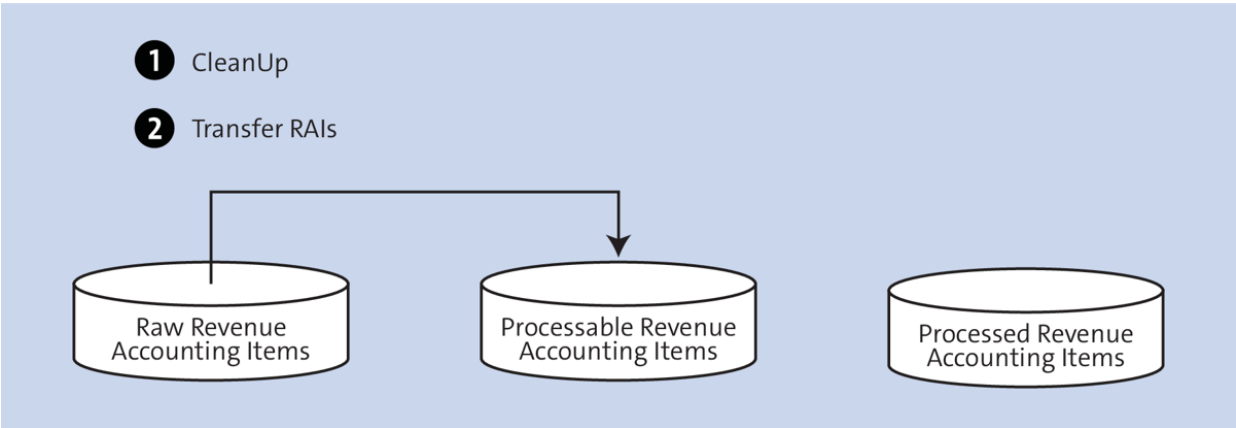


Figure 7.36 CleanUp Program

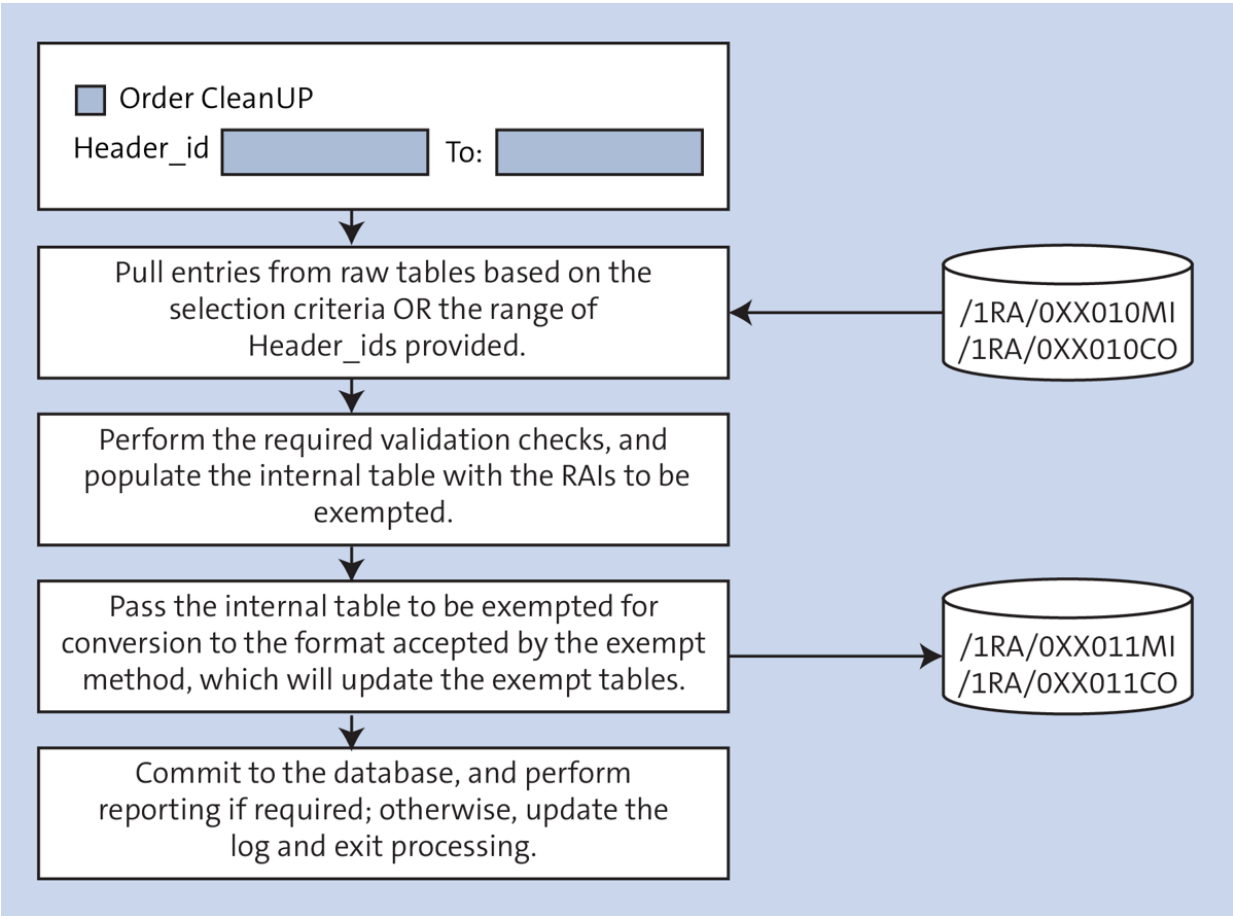


Figure 7.37 CleanUp Program Steps for Raw RAIs

SAP Function Builder: Display FARR_RAI0_EXEMPT

Function module: FARR_RAI0_EXEMPT Active

Attributes **Import** Export Changing Tables Exceptions Source code

Parameter Name	Typing	Associated Type	Default value	Opti...	Pass...	Short text	Long...
I_PARAMS	TYPE	FARR_S_EXCEPT_PAR.		<input checked="" type="checkbox"/>	<input type="checkbox"/>	Excepting Revenue Accounting Items: Run Para.	<input type="text"/>
IT_RAI0_KEY	TYPE	FARR_TT_RAI_SRCCKE.		<input type="checkbox"/>	<input type="checkbox"/>	Table of Source Document Keys	<input type="text"/>
I_SELECTED	TYPE	I		<input type="checkbox"/>	<input type="checkbox"/>	Items selected	<input type="text"/>

Figure 7.38 Function Module to Exempt Raw RAIs

Interface: IF_FARR_BADI_RAI2 Implemented / Active

Properties Interfaces Attributes **Methods** Events Types Aliases

Parameters of Method: ENRICH

← Methods Exceptions Properties

Parameter	Type	Pa...	Op...	Typing Method	Associated Type	Default Value
IV_RAIC	Importing	<input type="checkbox"/>	<input type="checkbox"/>	Type	FARR_RAIC	
CT_RAI2_MI	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_MI_ALL	
CT_RAI2_CO	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI2_CO_ALL	
CT_MESSAGES	Changing	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Type	FARR_TT_RAI_MSG	

Figure 7.39 ENRICH Method Parameters

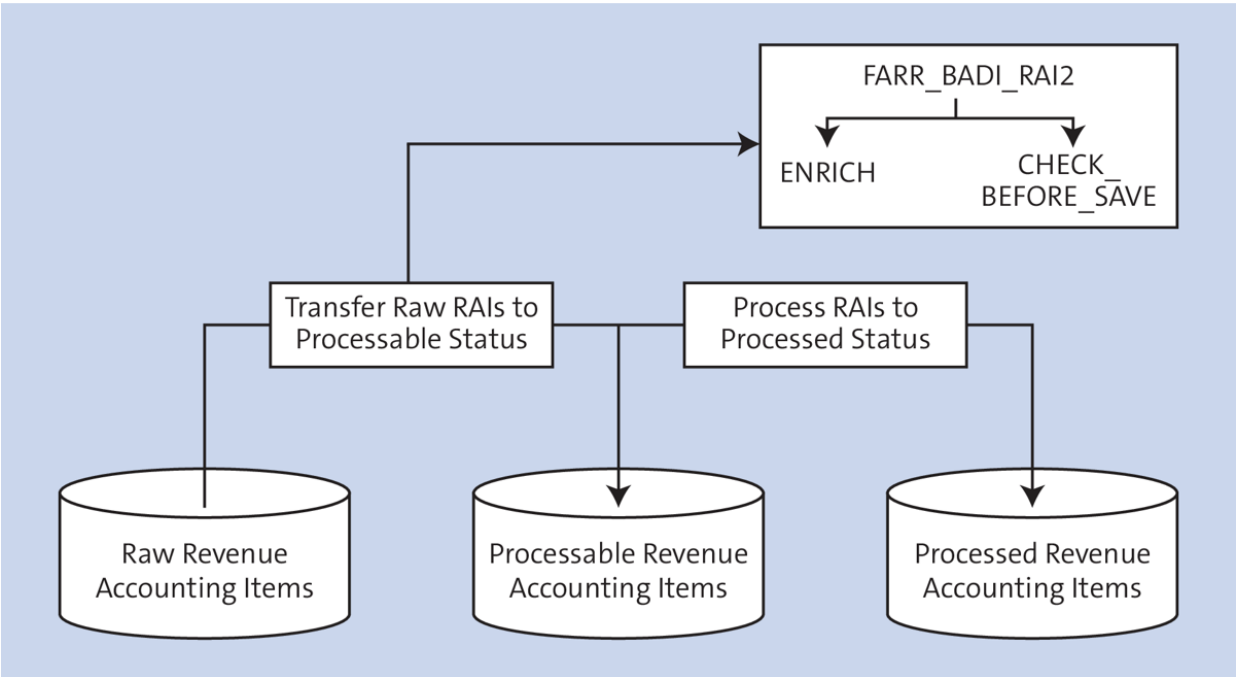


Figure 7.40 BAdI FARR_BADI_RA12

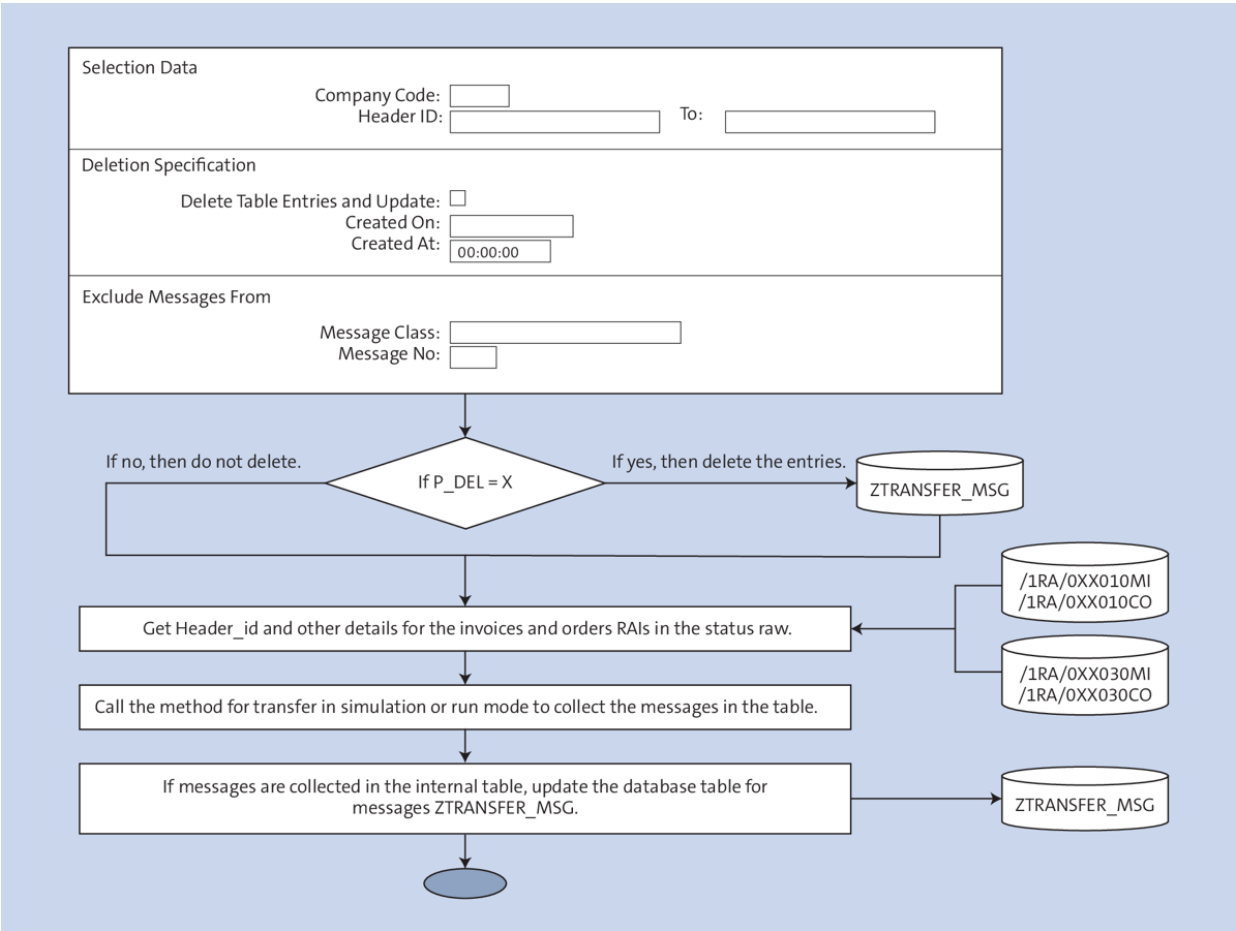


Figure 7.41 Error Management Tool

Customer Connection Error management Report

Save as Variant... More

Selection Data

* Company Code: 2000

Header Id: to:

Deletion specification

Delete Table entries & update:

Created On: Created At: 00:00:00

Exclude Messages from

Message Class: Message No:

Figure 7.42 Error Management Tool Flowchart

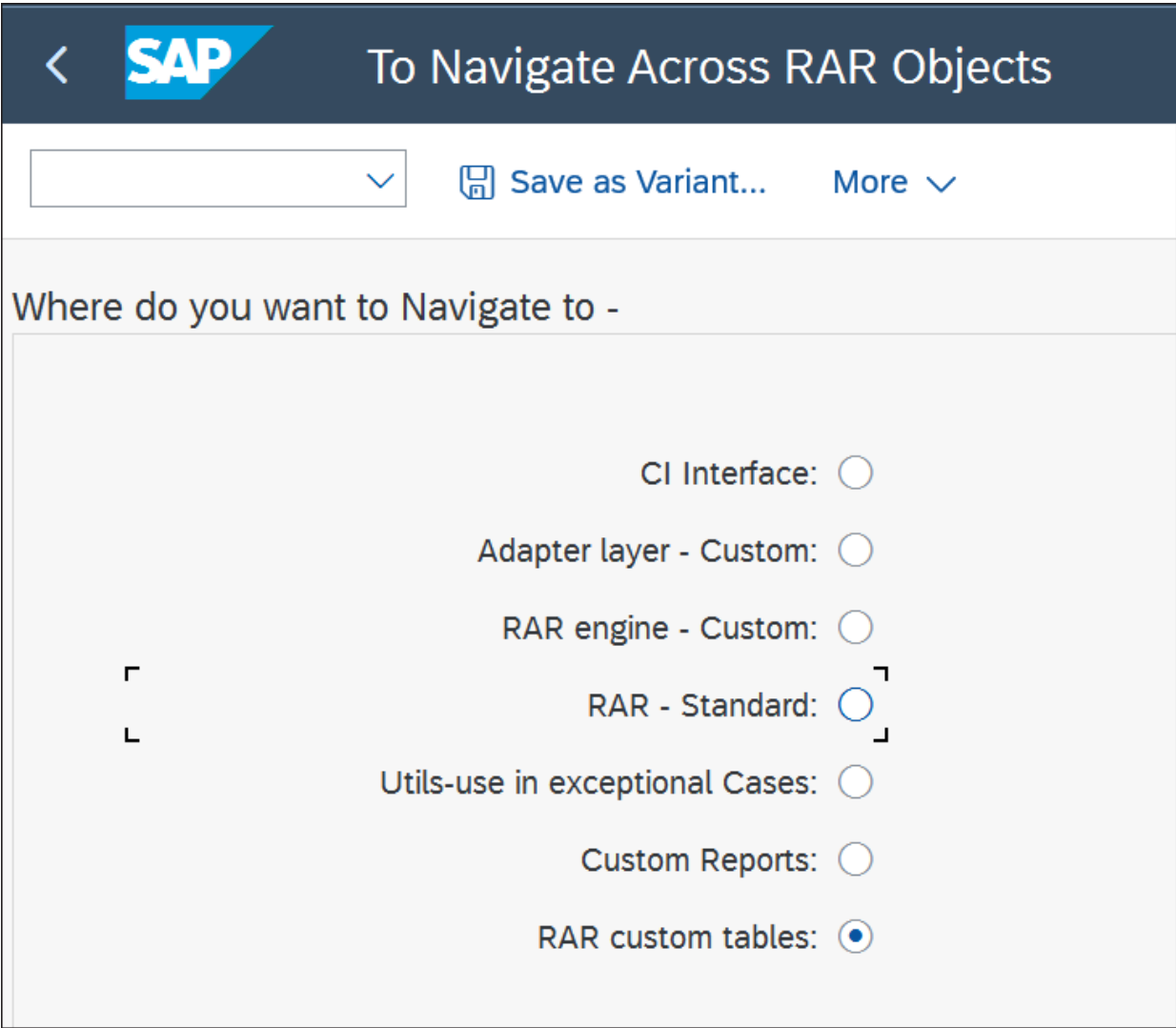


Figure 7.43 Navigator

Custom Reports:



RAR - Standard:



RAR custom tables:

Standard RAR programs

RAI Monitor:

RAI Transfer:

RAI Process:

(A) Revenue Transfer:

(B) CA/CL Calculation:

(C) FI Revenue Posting:

Mass Suspend/Unsuspend:

Reprocess RAR Contracts:

RAR Contract & POB Change His:

Contract Check:

Contract Monitor:

None:

Figure 7.44 Standard RAR Transaction Codes

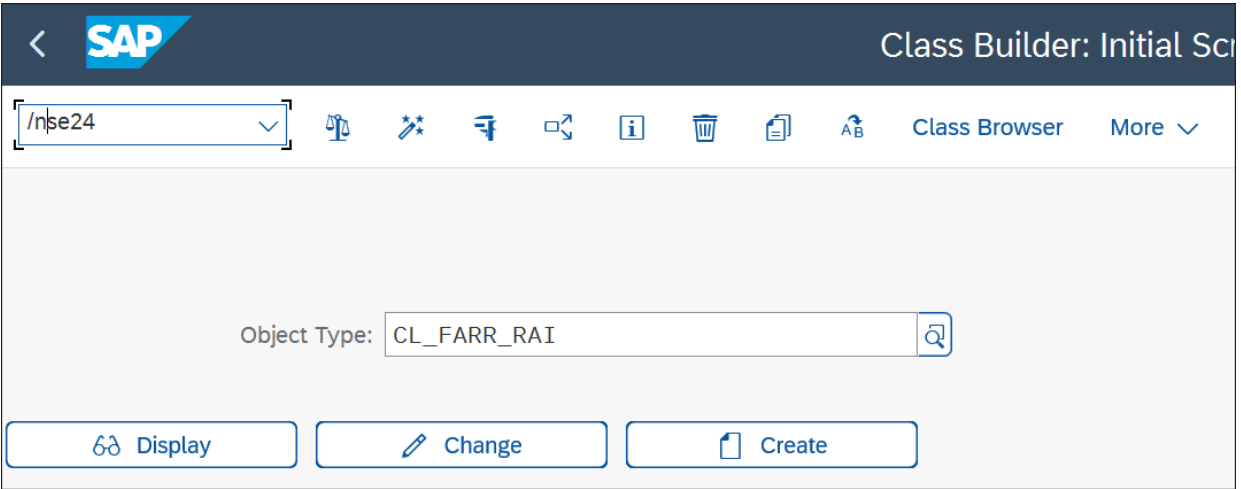


Figure 7.45 Class CL_FARR_RAI

SAP Class Builder: Display Class CL_FARR_RAI

Class/Interface: CL_FARR_RAI Implemented / Active

Properties Interfaces Friends Attributes **Methods** Events Types Aliases

Parameters Exceptions Sourcecode [Icons] Filter

Method	Level	Visibility	Me...	Description
IF_FARR_RAI~CREATE	Instance Method	Public		Execute Creation of Revenue Accounting Items from API
IF_FARR_RAI~MODIFY_RAI0	Instance Method	Public		Modify Revenue Accounting Items (RAI0, status raw data)
IF_FARR_RAI~MODIFY_RAI2	Instance Method	Public		Modify Revenue Accounting Items (RAI2, status processable)
IF_FARR_RAI~TRANSFER_TO_RAI2	Instance Method	Public		Transfer of Revenue Accounting Items (RAI0 -> RAI2)
IF_FARR_RAI~TRANSFER_TO_RAI2_MASS	Instance Method	Public		Mass Transfer of Revenue Accounting Items (RAI0 -> RAI2)
IF_FARR_RAI~MODIFY_RAI1	Instance Method	Public		Modify Revenue Accounting Items (RAI1, status raw data)
IF_FARR_RAI~MODIFY_RAI3	Instance Method	Public		Modify Revenue Accounting Items (RAI3, status process. data)
IF_FARR_RAI~WRITE_STAT_AND_CLOSE_LOG	Instance Method	Public		Write statistics to log and close log - for mass transfer
CONSTRUCTOR	Instance Method	Public		
CHECK_RAI3_IFCOMP	Instance Method	Private		Interface Component: Final check RAI3 before saving
FINISH_RAI1_IFCOMP	Instance Method	Private		Interface component method to check and finish RAI1

Figure 7.46 Methods of Class CL_FARR_RAI,

```

DATA: lo_create      TYPE REF TO cl_farr_rai.
      IF lo_create IS INITIAL.
          CREATE OBJECT lo_create.
      ENDIF.

TRY.
    CALL METHOD lo_create->if_farr_rai~create
        EXPORTING
            iv_raic      = 'CA01'
            it_api_mi    = ugt_miapi
            it_api_co    = ugt_coapi
        IMPORTING
            et_messages = lt_messages.
    CATCH cx_farr_message.
**      MESSAGE
ENDTRY. "

```

Figure 7.47 Creating RAIs


```

data: lo_farr_rai type ref to CL_FARR_RAI.

try.
    create object lo_farr_rai
        exporting
            IO_MSG_HANDLER = go_msg_handler
            iv_sub_obj      = if_farrc_msg_handler_cons=>co_subobj_rai_transfer
            iv_max_probcl  = if_farrc_msg_handler_cons=>co_probclass_low.
        catch cx_farr_message.
            "#EC NO_HANDLER
        endtry.

try.
    lo_farr_rai->IF_FARR_RAI~TRANSFER_TO_RAI2(
        exporting
            iv_test          = gc_x
            iv_no_commit     = gc_x
            iv_no_results    = gc_x
            iv_last          = gc_x
            it_rai0_pack     = lt_rai0_pack ).
        importing
            et_messages      = lt_messages ).
    catch cx_farr_message.
        message id sy-msgid type sy-msgty number sy-msgno
            with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    endtry.

```

Figure 7.48 Transfer Raw RAIs to the Processable Status

The screenshot shows the SAP 'Analysis of Items' interface. At the top, there is a navigation bar with the SAP logo and the title 'Analysis of Items'. Below this is a toolbar with various action buttons: Refresh, Statistics, Simulate Transfer, Transfer, Process, Change, Exempt, and Restore. The 'Exempt' button is highlighted with a blue border. Below the toolbar, there are two tabs: 'Main Item (54)' and 'Condition Item (54)'. A secondary toolbar contains icons for search, list, filter, and other functions. At the bottom, a table displays item data with columns for ItemStatus, Hist Exist, Error, Ex.His.Ex, Send.Com..., SourceSys, SrcitmType, Source Item ID, Subarea, RevAcc..., Header ID, ItemID, Ref. Type, Reference ID, and Customer. The first row of data shows a red dot in the 'Error' column and the value 'ZX' in the 'Send.Com...' column.

ItemStatus	Hist Exist	Error	Ex.His.Ex	Send.Com...	SourceSys	SrcitmType	Source Item ID	Subarea	RevAcc...	Header ID	ItemID	Ref. Type	Reference ID	Customer	Part
		●		ZX	DS4CLNT0...	ZACI	S18900436600010	367	ZA01	S189004366	10	ZOP	00S189004366	20064918	

Figure 7.49 Transaction FARR_RAI_MON and the Exempt Button

```
1 FUNCTION farr_rai0_exempt .
2 **-----
3 ***Local Interface:
4 ** IMPORTING
5 ** REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL
6 ** REFERENCE(IT_RAI0_KEY) TYPE FARR_TT_RAI_SRCKEY
7 ** REFERENCE(I_SELECTED) TYPE I
8 ** RAISING
9 ** CX_FARR_MESSAGE
10 **-----
11 **
```

Figure 7.50 Exempt Raw Entries from Being Moved to Processable

The screenshot shows the SAP Function Builder interface for the function module FARR_RAI2_EXEMPT. The 'Source code' tab is active, displaying the following code:

```
1 FUNCTION farr_rai2_exempt .
2 *"-----
3 *"*Local Interface:
4 *"  IMPORTING
5 *"    REFERENCE(I_PARAMS) TYPE  FARR_S_EXCEPT_PARAMS OPTIONAL
6 *"    REFERENCE(IT_RAI2_KEY) TYPE  FARR_TT_RAI_SRCKEY
7 *"    REFERENCE(I_SELECTED) TYPE  I
8 *"  RAISING
9 *"    CX_FARR_MESSAGE
10 *"-----
11
12 DATA: lt_rai_enq          TYPE farr_tt_rai_enqueue,
13        lt_keypp_bukrs_enq TYPE farr_tt_keypp_bukrs,
14        lt_rai2_key_del    TYPE farr_tt_rai_srckey,
```

Figure 7.51 Exempt Processable Entries from Being Processed

```
1 FUNCTION farr_rail_restore.  
2 *"  
3 *"*Local Interface:  
4 *" IMPORTING  
5 *" REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL  
6 *" REFERENCE(IT_RAI1_KEY) TYPE FARR_TT_RAI_SRCKEY  
7 *" REFERENCE(I_SELECTED) TYPE I  
8 *" RAISING  
9 *" CX_FARR_MESSAGE  
10 *"  
11 *  
12 DATA: lt_rai_enq TYPE farr_tt_rai_enqueue,  
13        lt_keypp_bukrs_enq TYPE farr_tt_keypp_bukrs,  
14        lt_rail_key_del TYPE farr_tt_rai_srckey,  
15        lt_rai_mi_all TYPE farr_tt_rai_mi_all,  
16        lt_rai_co_all TYPE farr_tt_rai_co_all,  
17        lt_rai_mi_all TYPE farr_s_rai_mi_all
```

Figure 7.52 Function Module to Restore Exempted Raw RAIs

```
1 FUNCTION farr_rai3_restore.  
2 ***  
3 ***"Local Interface:  
4 *** IMPORTING  
5 *** REFERENCE(I_PARAMS) TYPE FARR_S_EXCEPT_PARAMS OPTIONAL  
6 *** REFERENCE(IT_RAI3_KEY) TYPE FARR_TT_RAI_SRCKEY  
7 *** REFERENCE(I_SELECTED) TYPE I  
8 *** RAISING  
9 *** CX_FARR_MESSAGE  
10 ***  
11 *  
12 DATA: lt_rai_eng TYPE farr_tt_rai_enqueue,  
13 lt_keypp_bukrs_eng TYPE farr_tt_keypp_bukrs,  
14 lt_rai3_key_del TYPE farr_tt_rai_srckey,  
15 lt_rai_mi_all TYPE farr_tt_rai_mi_all,  
16 lt_rai_co_all TYPE farr_tt_rai_co_all,  
17 ls_rai_mi_all TYPE farr_s_rai_mi_all,
```

Figure 7.53 Restore Exempted Processable RAIs

SAP Class Builder: Display Class CL_FARR_RAI

Class/Interface: CL_FARR_RAI Implemented / Active

Properties Interfaces Friends Attributes **Methods** Events Types Aliases

Parameters Exceptions Sourcecode [Icons] Filter

Method	Level	Visibility	Me...	Description
IF_FARR_RAI~CREATE	Instance Method	Public		Execute Creation of Revenue Accounting Items from API
IF_FARR_RAI~MODIFY_RAI0	Instance Method	Public		Modify Revenue Accounting Items (RAI0, status raw data)
IF_FARR_RAI~MODIFY_RAI2	Instance Method	Public		Modify Revenue Accounting Items (RAI2, status processable)
IF_FARR_RAI~TRANSFER_TO_RAI2	Instance Method	Public		Transfer of Revenue Accounting Items (RAIO -> RAI2)
IF_FARR_RAI~TRANSFER_TO_RAI2_MASS	Instance Method	Public		Mass Transfer of Revenue Accounting Items (RAIO -> RAI2)
IF_FARR_RAI~MODIFY_RAI1	Instance Method	Public		Modify Revenue Accounting Items (RAI1, status raw data)
IF_FARR_RAI~MODIFY_RAI3	Instance Method	Public		Modify Revenue Accounting Items (RAI3, status process. data)
IF_FARR_RAI~WRITE_STAT_AND_CLOSE_LOG	Instance Method	Public		Write statistics to log and close log - for mass transfer
CONSTRUCTOR	Instance Method	Public		
CHECK_RAI3_IFCOMP	Instance Method	Private		Interface Component: Final check RAI3 before saving
FNTRCH_RAI1_IFCOMP	Instance Method	Private		Interface component method to check and enrich RAI1

Figure 7.54 Class CL_FARR_RAI

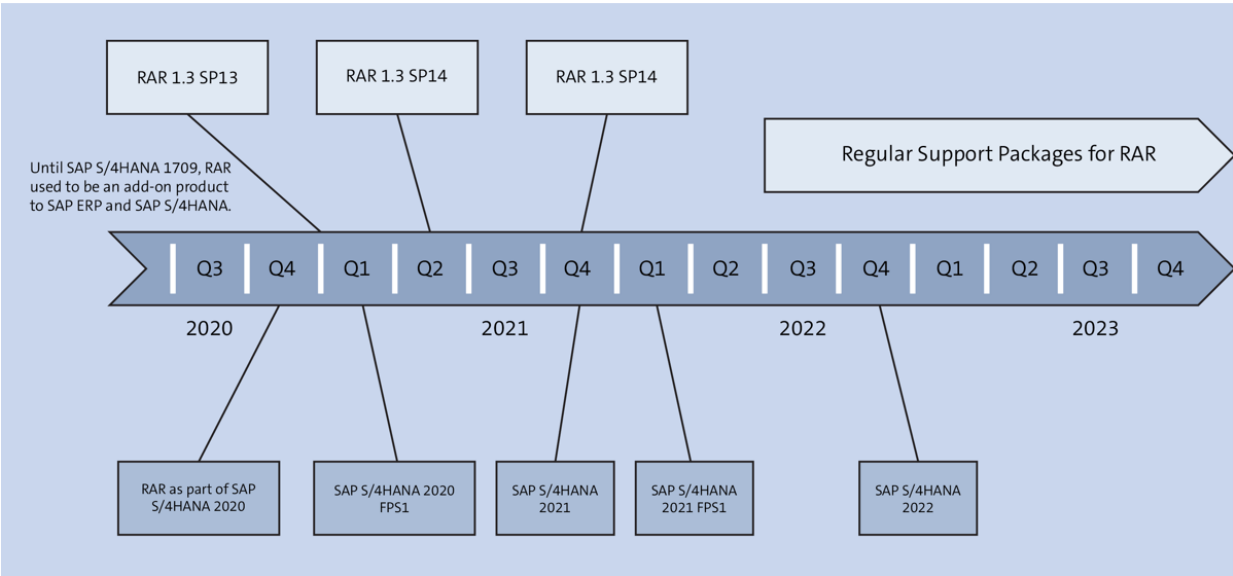


Figure 8.1 Current Versions of RAR on SAP ERP and SAP S/4HANA

Select Contract Management for Contract Categories					
	CoCd	Company Name	Contr. Cat	Contract Category Description	Contract With CM Instead of CM Classic
<input type="checkbox"/>	US19		0001	Revenue Contract	<input type="checkbox"/>
<input type="checkbox"/>	US19		0002	Revenue contracts (Classic)	<input type="checkbox"/>

Figure 8.2 Contract Management with OCM

■ Decision Table: DT_PROCESS_HEADER

[Back](#) | [Edit](#) | [Check](#) | [Save](#) | [Activate](#) | [Transport](#) | [Delete](#) | [More](#)

General

Detail

[Export To Excel](#) | [Context Overview](#) | [Start Simulation](#)

Table Contents


Find: [Next](#) [Previous](#)


<input type="checkbox"/>	SALES_ORG	PSTYV	AUART	CONTRACT_CAT	CUSTOMER_GRP
--------------------------	-----------	-------	-------	---------------------	--------------


[i](#) Table contains no rows. Use "Insert New Row" to add conditions.

Figure 8.3 Customizing in BRFplus

Selection Parameters

*Accounting Principle: 

*Company Code: to: 

Revenue Accounting Contract: to: 

Additional Migration Parameters

Mig. Fixed Rate RA Contract:

Mig. POB with Start Date Ty. 3:

Mig. POB with Manual Spread:

Run Parameters


*Max Work Proc Usage in Percent:

*Block Size for Mass Selection:

Dialog Mode:

Simulation Mode:

Settings for Application Log

Problem class: 

Show Application Log:

Figure 8.4 Program for Migrating from CCM to OCM

Table:

Text table:

Layout:

Maximum no. of hits: Maintain entries

Get Field:

Selection Criteria

Fld name	O...	Fr.Value	To value	More	Output	Technical name
Client						CLIENT
POB	<input type="button" value="Filter"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="More"/>	<input checked="" type="checkbox"/>	POB_ID
Freeze Date	<input type="button" value="Filter"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="More"/>	<input checked="" type="checkbox"/>	FREEZE_DATE
Unfreeze Date	<input type="button" value="Filter"/>			<input type="button" value="More"/>	<input checked="" type="checkbox"/>	UNFREEZE_DATE
Created by	<input type="button" value="Filter"/>			<input type="button" value="More"/>	<input checked="" type="checkbox"/>	CHANGED_BY
Time Stamp	<input type="button" value="Filter"/>			<input type="button" value="More"/>	<input checked="" type="checkbox"/>	CHANGED_ON

Figure 8.5 Freeze Periods Entry

Run Parameters

* Max Work Proc Usage in Percent:

* Block Size for Mass Selection:

Dialog Mode:

Simulation Mode:

Figure 8.6 Run Parameters Setup

■	Problem class: Important	
■	Max No. of WP (%): 100	
■	Package size: 1000	
■	Migrate fixed rate contracts: X	
■	Number of contracts to be migrated: 131	
●	POB 100005: Standard error detected during data consistency validation	?
●	POB 100007: Standard error detected during data consistency validation	?
●	POB 100009: Standard error detected during data consistency validation	?
●	POB 100010: Standard error detected during data consistency validation	?
●	POB 100012: Standard error detected during data consistency validation	?
●	POB 100015: Standard error detected during data consistency validation	?
●	POB 100016: Standard error detected during data consistency validation	?
●	POB 100017: Standard error detected during data consistency validation	?
●	POB 100018: Standard error detected during data consistency validation	?
●	POB 100025: Standard error detected during data consistency validation	?
●	POB 100027: Standard error detected during data consistency validation	?
●	POB 100038: Standard error detected during data consistency validation	?
●	POB 100039: Standard error detected during data consistency validation	?
●	POB 100040: Standard error detected during data consistency validation	?
●	POB 100041: Standard error detected during data consistency validation	?

Figure 8.7 Validation Errors

■	Simulation mode
■	Problem class: Important
■	Max No. of WP (%): 100
■	Package size: 1000
■	Migrate fixed rate contracts: X
■	Number of contracts to be migrated: 1
●	POB 116297: POB with CO object number is not supported
■	Number of migrated contracts: 0

Figure 8.8 CO Object Error

Date/Time/User	Number	External ID	Object
17.03.2023 19:15:13 SM6918	144	SM69182023031719...	Reven
● Problem class Important	144		

Ty...	Message Text
●	Contract 100012: Fixed Local Currency Calculation Method is not supported
●	Contract 100014: Fixed Local Currency Calculation Method is not supported
■	Contract 100015: Completed contract is not supported
●	Contract 100016: Fixed Local Currency Calculation Method is not supported
●	Contract 100017: Fixed Local Currency Calculation Method is not supported
●	Contract 100018: Fixed Local Currency Calculation Method is not supported
●	Contract 100019: Fixed Local Currency Calculation Method is not supported
●	Contract 100021: Fixed Local Currency Calculation Method is not supported
●	Contract 100023: Fixed Local Currency Calculation Method is not supported
●	Contract 100024: Fixed Local Currency Calculation Method is not supported

Figure 8.9 Fixed Currency Translation Error

Date/Time/User	Number	External ID
▼ ● 17.03.2023 19:18:59 SM6918	161	SM69182023031719.
● Problem class Important	161	

Ty...	Message Text
●	POB 100116: Invoice exists with open recon key in migration period
●	POB 112002: Invoice exists with open recon key in migration period
●	POB 112040: Invoice exists with open recon key in migration period
●	POB 112040: Invoice exists with open recon key in migration period
●	POB 108117: Invoice exists with open recon key in migration period
●	POB 108127: Invoice exists with open recon key in migration period
●	Event-based POB 116282: Fulfillment exists with posting day in future period
●	Event-based POB 116291: Fulfillment exists with posting day in future period
●	Event-based POB 116289: Fulfillment exists with posting day in future period
●	Contract 100014: Open recon key exists in closed period 008/2022
●	Contract 100016: Open recon key exists in closed period 008/2022
●	Contract 100026: Open recon key found in period 011/2022
●	Contract 100027: Open recon key found in period 011/2022
●	Contract 100029: Open recon key found in period 011/2022
●	Contract 100035: Open recon key found in period 011/2022
●	Contract 100037: Open recon key found in period 011/2022

Figure 8.10 Errors in Migration of Open Reconciliation Keys

●	Account determination for contract liability failed for company code AU02	?
●	Account determination for contract 100012 failed, see error log	
●	Account determination for contract 100012 failed, see error log	
●	Account determination for contract liability failed for company code AU02	?
●	Account determination for contract 100017 failed, see error log	
●	Account determination for contract 100017 failed, see error log	
●	Account determination for contract liability failed for company code AU02	?
●	Account determination for contract 100018 failed, see error log	
●	Account determination for contract 100018 failed, see error log	
●	Account determination for contract liability failed for company code AU02	?
●	Account determination for contract 100019 failed, see error log	
●	Account determination for contract 100019 failed, see error log	
●	Account determination for contract liability failed for company code AU02	?
●	Account determination for contract 100024 failed, see error log	

Figure 8.11 Account Determination Errors

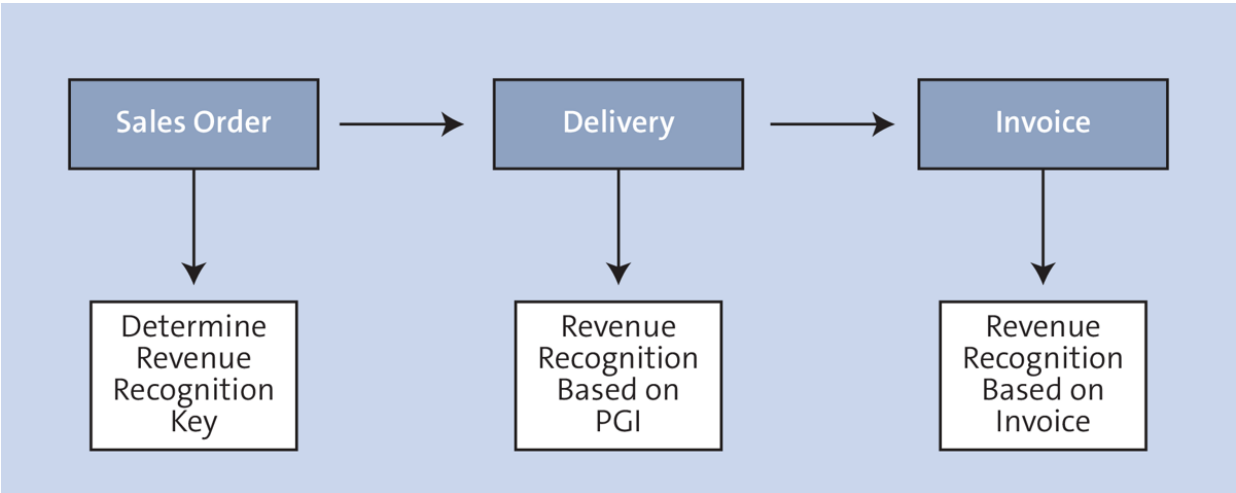


Figure 9.1 Process of Sales Integrated with EBRR

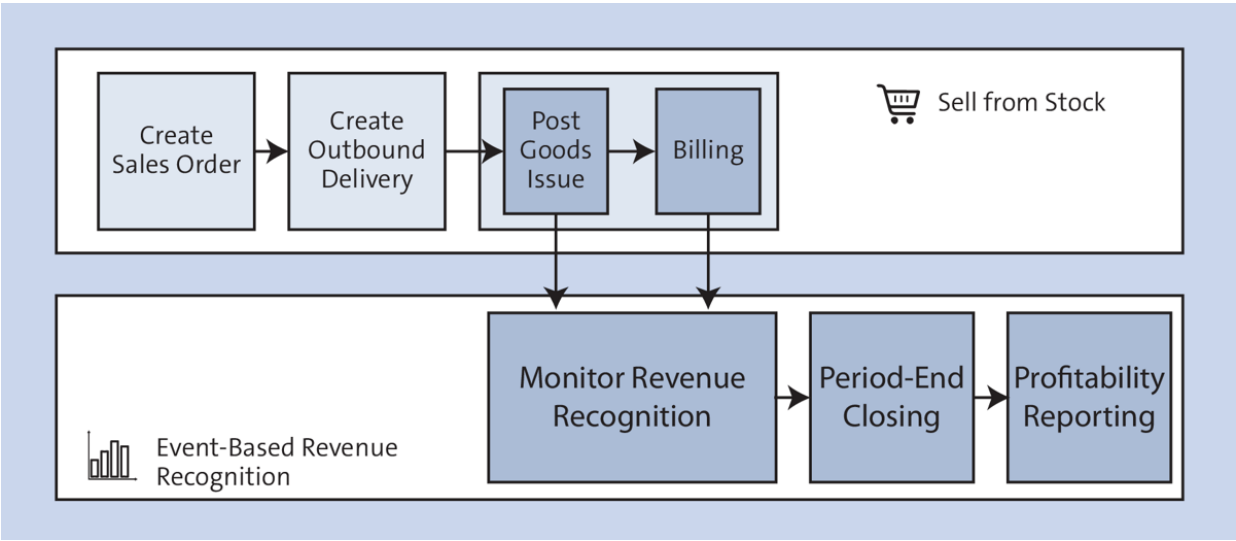


Figure 9.2 EBRR Based on Sell from Stock

Project def.: AUS-00000189

Detail:

Overview(s):

Results Analysis Key (1) 15 Entries f

Restrictions

Basic Data **Control** Administration Long

Project Profile: Software Project

Accounting

Budget Profile:

Planning Profile: Z00005

Interest Profile:

Investment Profile:

Results analysis key:

Simulation profile: 0000001

PartnerDetermProced.:

Default Values for New WBS Elements

RA Key	RA	Cap.ca...	EBRevRec	RecKeyArea	Result Analysis Description	Ad
PG0001	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA POC	
PG0002	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Quantity Based SKF	
PG0003	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA MTS Rev.bsd - No WPD at FNBL	
PG0004	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Production	
PG0005	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Projects - CCM	
PG0007	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Revenue based, All items (K) - No WPD at FNBL	
PG0008	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Revenue based - WDP at FNBL	
PG0009	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Quantity based delivery	
PG0010	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Revenue based, Sub positions (B) - No WPD at FNBL	
PG0011	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		PG Warranty RA	
PG0020	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA Resource-releated billing	
PG0021	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Straight line POC	
PG0022	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		Straight line POC billing plan	
PG0023	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA ZNWIP	
PG0024	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		RA POC	

Figure 9.3 Example of Different Results Analysis Keys Available

Responsibilities	Operative indicators
Pers.Resp.No.: <input type="text" value="71663714"/>	Planning Element: <input checked="" type="checkbox"/>
Applicant no.: <input type="text"/>	Acct asst elem.: <input checked="" type="checkbox"/>
Resp. cost cntr: <input type="text"/> <input type="text"/>	Billing Element: <input type="checkbox"/>
Req.cost center: <input type="text"/> <input type="text"/>	
Req. co.code: <input type="text"/>	Grouping WBS element <input type="text"/>

Figure 9.4 Definition of Billing in WBS Element

Navigation: < << >> >

Sales Document Item: 10 Item category: ZPMT Consulting Services
Material: VXE01ES225 Ellipse PSO_PSO_TE

Sales A Sales B Contract data Shipping Billing Document Conditions Account Assignment Schedule lines

Account assignment

Business Area:

Profit Center: AU02285140 Order:

WBS element: AUS-00000188.02.01.06 Profit. Segment:

Figure 9.5 Assignment of a WBS Element to Sales Order

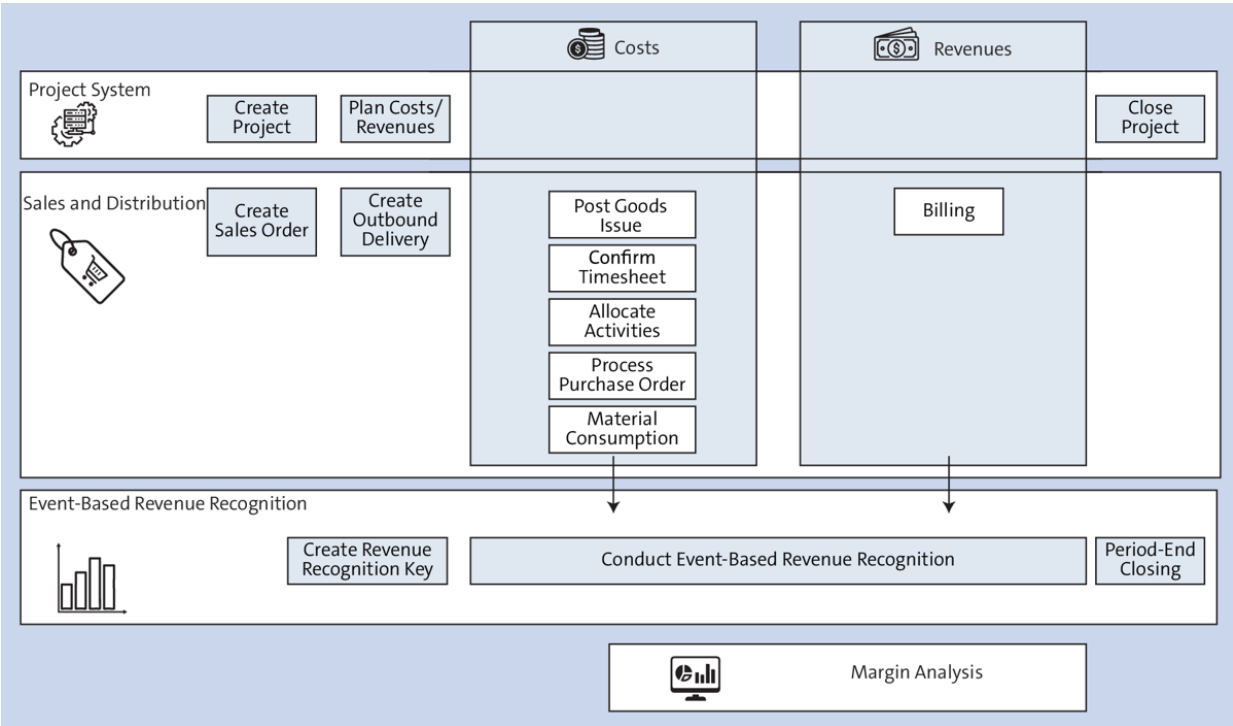


Figure 9.6 Project-Related Process for Revenue Recognition



