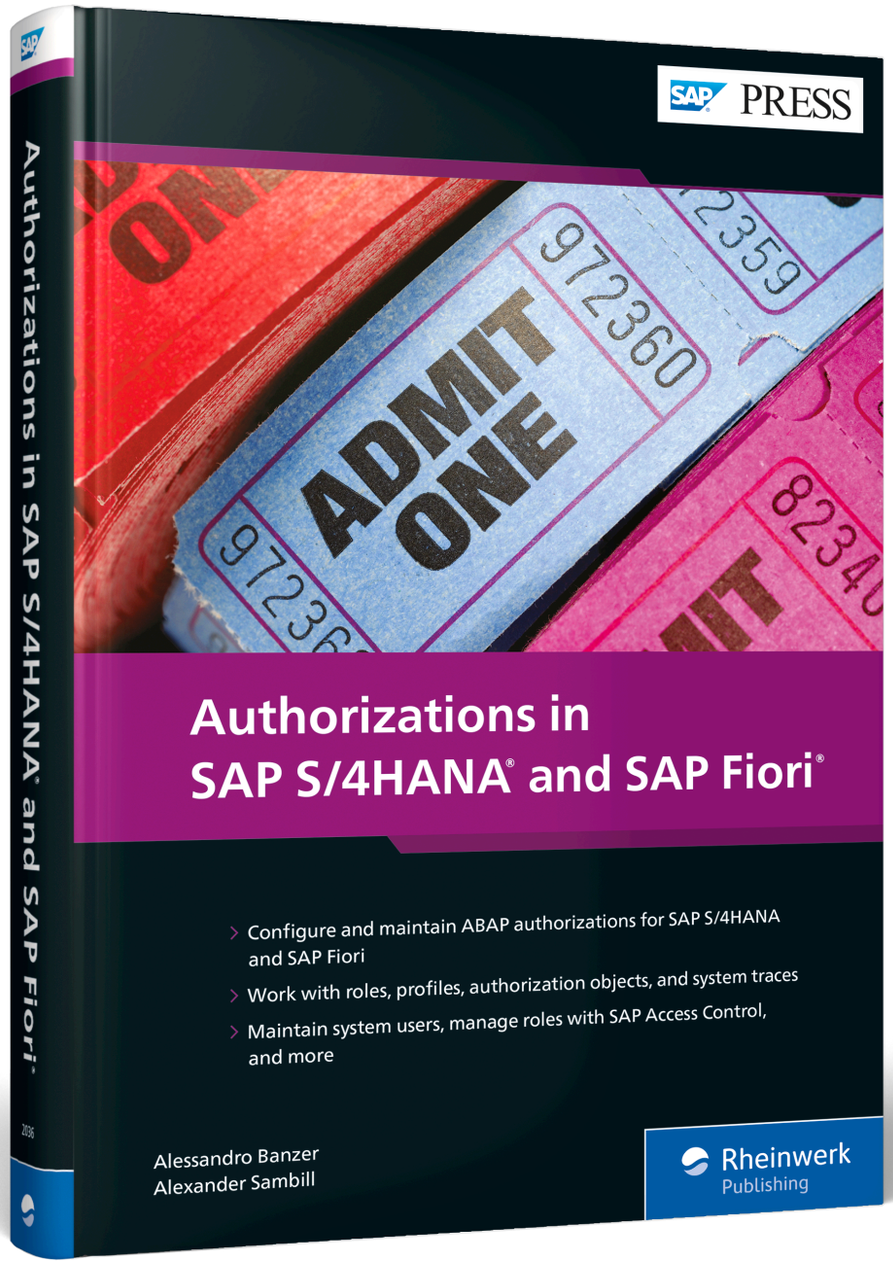


Browse the Book

This sample chapter provides an overview of different role types and how to use them. It starts by covering the different role design approaches, then dives into more detailed explanations of each role type and how they can be used in different scenarios. Next, it covers how to use segregation of duties (SoD) to separate critical functions of business processes among different roles. The chapter closes with best practices for developing a standard role naming convention.

Alessandro Banzer, Alexander Sambill

Authorizations in SAP S/4HANA and SAP Fiori

625 pages, 2022, $89.95
ISBN 978-1-4932-2036-6

www.sap-press.com/5203

Chapter 3
Designing Authorization Concepts

Designing authorization concepts is an ambitious task that requires thorough planning and deep understanding of the technical components of an authorization concept. You'll also need to understand fully how your organization runs its processes.

This chapter provides an overview of different role design concepts and approaches, including why to favor some over others and how to use them in practice. The overall goal of a role concept is to help establish maximum security, provide sufficient authorizations for end users to fulfill their job duties, simplify user maintenance, and help you sustainably maintain your roles.

Designing an authorization concept is a complex task with multiple drivers that must be taken into account. First and foremost, you must ensure that your end users can do their jobs and run the business, but at the same time, you'll need to adhere to access governance and compliance laws and regulations. The pitfalls of a poorly designed authorization concept include loss of productivity, high maintenance effort, exposed risk for the organization, failure in audits, inefficient access provisioning, and many more. Therefore, properly thinking through a concept that is sustainable, maintainable, transparent, and upgradable in the future is vital.

Ideally, your company discusses and addresses security considerations before an actual implementation. However, more often, companies have not proactively addressed security design beforehand and are thus challenged with costly role redesign projects shortly after go-live. All too frequently, companies change, expand, and merge with or acquire other companies, thus resulting in changes to the security design. A solid and sustainable design must be flexible enough to adjust to these changes.

3.1 Role Design Approaches

For various reasons, as shown in Figure 3.1, a company may consider (re)designing its authorization concept. The main drivers are system upgrades (e.g., migration to SAP S/4HANA); remediation of compliance issues, such as segregation of duties (SoD) or legal requirements; mergers and acquisitions that lead to integration projects; or simply because the current design is not maintainable and access governance cannot be ensured.

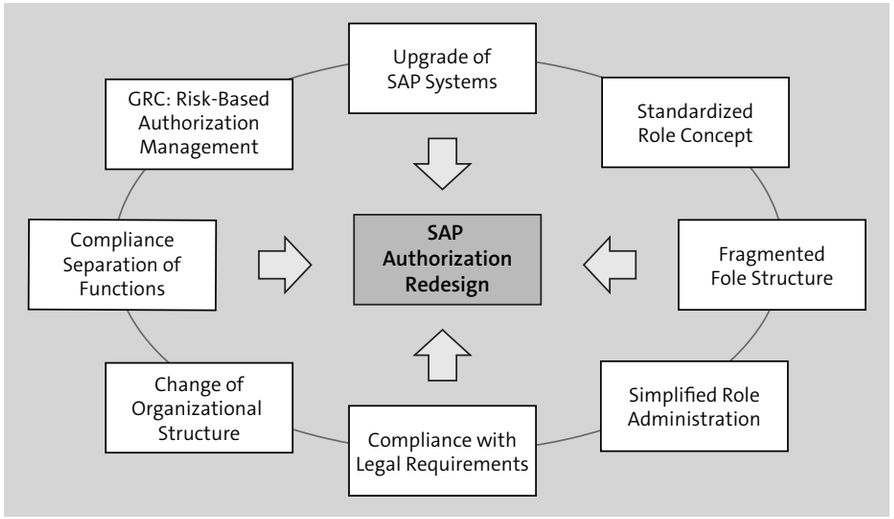


Figure 3.1 Reason for Authorization Redesign

Figure 3.2 shows some important trends regarding role design.

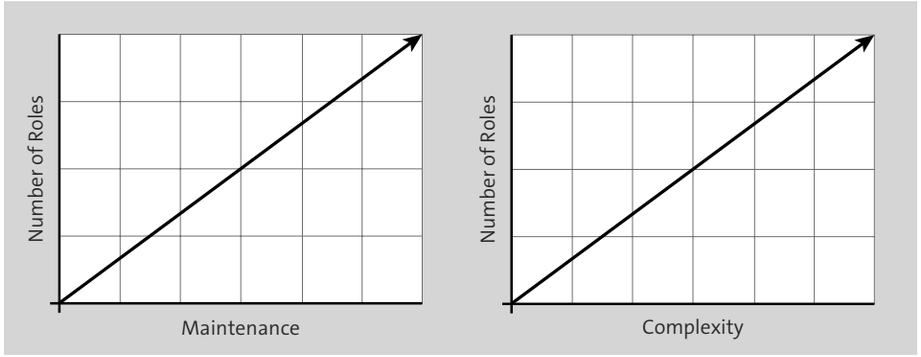


Figure 3.2 Role Design Complexity and Maintenance Effort

Regardless of the approach taken, your role concept should be as simple as possible to ensure that the investment is sustainable and long term. The number of roles should (if possible) be kept small and specific so that maintenance overhead is reduced. Complexity can be reduced when utilizing function-related (job-related) single roles. This approach increases flexibility and transparency for future adjustments of business processes, especially when considering a risk approach due to SoD or other compliance regulations.

Implementing an authorization concept can be performed in multiple ways. Two main approaches exist for building an authorization concept, but a hybrid of the two approaches can also be adopted. The first approach is the "top-down" approach, which

is a proactive way to design and conceptualize the necessary authorizations upfront, based on an analysis of business processes and job functions. With the "bottom-up" approach, the analysis starts with the available statistical usage data as well as with existing authorizations and their assignments.

Both approaches have their pros and cons. The "top-down" approach helps address security risks and requirements during the blueprinting phase but might prove to be difficult to implement. This approach requires a deep knowledge of the business processes, laws and regulations, and security implications like financial and audit requirements. The "bottom-up" approach is particularly difficult when remediating SoD conflicts, which requires inputs and an understanding of the business processes performed by business process managers. Without this knowledge, this approach tends to lead to a high number of roles built specifically to remediate SoD conflicts, which can lead to difficulty in provisioning as well as negatively impact the maintainability and sustainability of the authorization concept. This potential issue stems from the fact that, in most cases, SoD conflicts are avoided on the role level, which requires a separation of each job function.

Let's now briefly summarize the two approaches:

- **Top-down approach**
  Information will be acquired by interviews and an analysis of the business processes performed in your organization:
  - Analysis of business processes
  - Jobs within processes
  - Core activities
  - User groups
- **Bottom-up approach**
  Information will be acquired by an analysis of the current system and its user history data:
  - Existing authorizations
  - Existing role assignments and users
  - Use of transactions and other menu objects
  - Existing organizational restrictions in the system

To further highlight the difference between the bottom-up and top-down approaches, to summarize, the bottom-up approach uses existing data (transaction usage, current role and authorization assignments, etc.) to drive the role design. In contrast, the top-down approach utilizes business processes to shape the role design. Regardless of the approach, defining security requirements early in the project is important as being proactive helps ensure efficiency during implementation. In addition, we recommend leveraging tools like SAP Access Control or similar solutions, to analyze and monitor role design to address potential SoD violations early on.

Combining both approaches allows for the best results when redesigning authorization concepts, as shown in Figure 3.3. You can take usage data and existing authorizations from the "bottom-up" approach and use that information to come up with a draft design. That draft can then be discussed with business process managers to separate out functions and fine-tune the design based on how your company runs its processes.

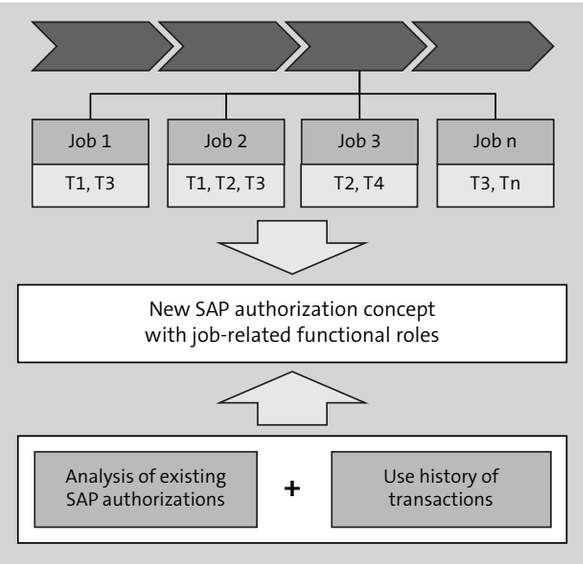


Figure 3.3 Top-Down and Bottom-Up Approaches

Before beginning the work of creating roles as part of your effort to enhance security in your SAP systems, an important decision must be made. You must decide between using *job-based roles* or *task-based roles*, and this choice will impact how your roles will be designed.

Let's briefly look at these two concepts next:

- **Job-based roles**
  Job-based roles are designed for including all of a person's job activities into one single role. For instance, a person with the position of a "sales order specialist" must perform certain activities such as creating, changing, and displaying sales orders. A role encompassing all those activities for that position is an example of a job-based role design. Utilizing this role design method uses fewer roles by consolidating job duties into one role instead of creating a role for each individual job task. Be mindful, however, that this approach can be prone to SoD conflicts due to the broader access subsumed under one role. While utilizing this method, make sure you put effort into mitigating these conflicts.

  Using the top-down approach "to create your job-based roles, you first must analyze business processes and their jobs-related tasks. This analysis helps outline of how

the job-based role should look. Then, you'll move on to identifying the core activities within those jobs. This step assists with discovering the relevant transactions for the job-based role. Finally, you'll map out your user groups to help you assign users with the right role assignments. Knowing how users are grouped together (i.e., a department) allows for efficient job role assignments through user groups.

- **Task-based roles**
  In a task-based role design, a role is created for each job task within a job function. Continuing our earlier "sales order specialist" example, this position requires several tasks such as creating, changing, and displaying sales orders. In a task-based role design approach, a separate role will be created for each of the job task: One role will be designed for creating sales orders; another role, for changing sales orders; and the final role, for displaying sales orders—for a total of three roles. As a result, more roles are required in a task-based role in comparison to a job-based design where the role count would be one. SoD conflicts still need to be considered, but they may be reduced (on a role level) since access is more granular in these roles.

  Using the bottom-up approach" to create your task-based roles, you would first analyze existing authorizations. Being knowledgeable on existing authorizations allows you to gauge existing access levels. Combining this information with the next step while reviewing existing user role assignments, you can tailor appropriate access to users based on the tasks performed in their business functions. Finally, analyzing the use of transactions helps you select which tasks should be included a task-based role. Understanding how transactions correlate to the tasks helps you properly build roles for different areas of your business.

3.2 Role Types

From a technical point of view, only two types of roles are available in SAP—single roles and composite roles. Single roles can further be divided into child roles that only differ from the parent role on the organizational level. However, many organizations have additional "types" of roles, for example, enabler/value roles, which technically are still single roles but have distinct intentions.

3.2.1 Single Roles

A single role contains all the necessary authorization objects and field values (organizational and non-organizational) required for the transactions that the role contains. In SAP, many authorization objects are represented by two types of authorization fields: the activity field and the organization value field. Technically, you cannot separate these two fields because the kernel evaluates them together during an authority check, and thus, both fields must be defined in the same role.

Hint

See Chapter 2, Section 2 for more information about the cumulative aspect of SAP authorizations.

However, a single role can have multiple individual authorization instances to represent different combinations of field values when these values cannot be merged into one authorization instance to be evaluated by the system check. Thus, a single role can also be a composite of authorization instances.

Typically, when someone talks about the single role concept, they are referring to a job-/position-based role design. In such cases, the role contains all the required authorizations for a user's job/position in a single role. A user might, however, have more than one job/position, such as a purchaser and a contract manager. Each role has authorizations to complete the transactions that are contained in each of the two single roles so that they are functional on their own. With this approach, no dependencies exist.

However, many "single role" designs won't contain all of a user's required authorizations. Some employees might have additional authorizations to execute special tasks. As a result, such employees may get an "additional access" single role assigned to them, for instance, with the ability to close accounting periods or approve purchase requisitions. Figure 3.4 shows an example concept of functional roles. Likewise, a common practice, as shown in Table 3.1, is to create a "basic authorization" role that contains transactions and authorizations that are common for all users, for example, the ability to print or to access office mailboxes in SAP.

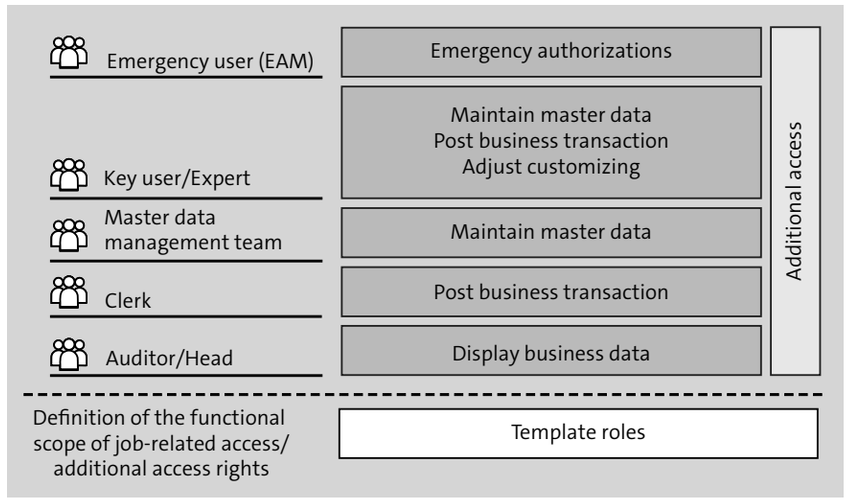


Figure 3.4 Concept of Functional Roles

Types of Role	Subtypes	Explanation
Single Roles	Template roles (Layer 00)	The role is used in the project (design phase) as a template (master role) for all singles role to be created.
	Function-related (job) roles (Layer 01-05)	Terms for the creation of functions or job profiles within the business processes such as accountants, buyers, sellers, Basis administrators, SAP developers, etc.
	Special roles (critical) (Layer 06)	Assigned for specific use or critical authorizations (such as SoD, SFX, FDA, DSG) and are targeted at specific users.
	Special roles for controlling (Layer 06)	This is only used if the requirement is to make restrictions in the area of controlling, such as cost centers, orders, profit centers, etc.
Composite Roles	Function-related roles	Terms for the summary of the function-specific roles and special roles (such as accountants, buyers, Basis staff, etc.

Table 3.1 Types of Roles and Subtypes

Likewise, you can also use single roles for role derivation. Derived roles consist of a master or parent role and additional child roles that differ from the master and from each other only in their organizational values. Please note that limitations exist with this approach. If you try to promote non-organizational fields to organizational fields, then, in this case, the values must all be the same within one role, regardless of which authorization object is using the field. In other words, you should not use different non-organizational fields in combination with derived roles because the values across all child roles will always be the same as the master role and will affect all objects.

Hint

Chapter 2, Section 2.3.5, contains extensive information about promoting non-organizational authorization fields.

3.2.2 Composite Roles

A composite role is a collection of single roles grouped together into a common composite role menu. As a result, you can indirectly assign multiple single roles to a user by assigning a composite role that contains single roles. However, you may be tempted to build numerous smaller single roles without considering the required user assignments.

Many businesses using SAP leverage composite roles to reduce the number of direct single role assignments to users. However, this approach often leads to less transparency, higher effort for maintaining more roles, and an increased risk of accidental inheritance of incorrect authorizations from a single role impacting multiple composites. The ability to change single roles rapidly leads to the side effect of having more single roles to maintain.

Technically, a composite role is a bundle of single roles to map a task-level single role (in the worst case, a transaction-level single role) to a broader role. The goal of a composite role is often to simplify the assignment of roles to users since composite roles often represent a job function. In addition to the number of single roles that may result, your organizational structure and its changes can also play a role in the choice of approach.

3.2.3 Enabler Roles

In recent years, we've seen role concepts in which the SAP standard has not been followed. These concepts are often called *enabler role concepts* or *value role concepts*. In these concepts, you would separate organizational authorization values from functional authorizations, which results in the need for two roles to execute a transaction successfully.

In these authorization concepts, a functional role contains all the authorization objects and values, but not the organizational level. A second, also required, enabler role contains the "missing" organizational values and enlarges a user's authorizations. Thus, to successfully execute a transaction, a user requires both the functional role and its corresponding enabler role.

The idea behind enabler roles lies in the desire to isolate organizational fields and simplify user assignments when it comes to purely organizational, non-functional distinctions.

However, any organizational type of field, which also has an "activity" field in the same authorization object, cannot be separated into a different role. This limitation is almost always the case in SAP authorization objects. Due to this absolute disadvantage, enabler roles sometimes cannot work as designed, which typically results in a proliferation of authorizations in enabler roles and the number of enabler roles, which often exceeds the number of users.

The enabler role concept sometimes looks tempting on paper, and some businesses might expect enabler roles to behave similarly to organizational management in SAP ERP Human Capital Management (SAP ERP HCM). However, you cannot apply it to authorization objects and role-based authorization concepts nor to the modern menu-based visibility access concepts, such as SAP Fiori applications and SAP Enterprise Portal/SAP Business Warehouse (SAP BW) reporting.

In addition to enabler roles for organizational values, subsets of your authorization concept can benefit from implementing enablers, for example, to restrict access to controlling scenarios or to implement a release strategy for purchase orders.

In purchasing, for example, a release strategy for purchase orders is defined, which means that purchase orders released through Transaction ME24(N) require specific authorizations. The release strategy might define the required approval level based on the purchase order amount. A user who wants to approve a purchase order requires the correct release code and release group authorizations through authorization object M_EINK_FRG. Since approval limits are sometimes not directly related to a job function, the release strategy is often defined in enabler roles that are individually assigned to users.

Enabler roles are special roles that can be valuable for maintaining a sustainable role concept while reducing the complexity that comes with including these special authorizations in job roles. For more information on how to determine when to use or not use an enabler role, see Section 3.4.

3.2.4 Authorization Templates and Standard Roles

SAP standard roles and authorization templates can serve as reference points for role creation so that you don't have to start from scratch when creating roles. SAP provides standard roles and authorization templates to find business authorizations and transactions that may be relevant to your business. Using these templates and roles as starting points can save time in terms of the research effort to identify authorizations and transactions for a particular role. Template roles usually have a wide coverage of authorizations, so they should be used as a reference to tailor roles to your business needs. For instance, the authorization template SAP_AIF_USER is for personnel responsible for error handling and monitoring interfaces. This template comes with a set of authorizations and transactions, such as editing fields, restarting, and canceling data messages. However, you must keep in mind that templates are valuable in sandbox and testing environments, but should be avoided in productive landscapes. Authorization templates are manually added authorizations that are far reaching. SAP standard roles are also far reaching, and the roles are routinely updated by SAP. As a result, during the next system upgrade, SAP-delivered roles are overwritten. Therefore, you should never use standard roles or standard authorization templates directly in your productive landscape as assignments to your end users. Standard templates and roles from SAP should only serve as reference points for building your own distinct custom roles.

To add authorizations from a template, follow these steps:

1. Go to Transaction PFCG.
2. Enter your role name and click on the kind of role you want to create (**Single** or **Composite**). In this example, we are creating a single role.

3. Go to the **Authorizations** tab.
4. Click on the icon next to **Expert Mode** for **Profile Generation**.
5. As shown in Figure 3.5, follow the menu path **Edit • Insert authorization(s) • From template...**.

Using templates as a starting point, you can then tailor transactions and authorizations as needed for your company.

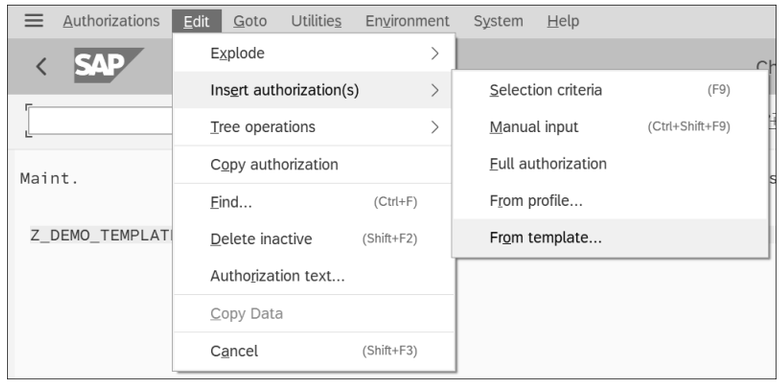


Figure 3.5 Inserting Authorizations from a Template

Please keep in mind that inserting authorizations from a template adds all the authorizations as manual objects, as shown in Figure 3.6. This approach goes against best practices but is helpful when trying to understand what authorizations are required. To properly authorize your roles, you should add the transactions through your role menu so that you can leverage authorization proposals from Transaction SU24.

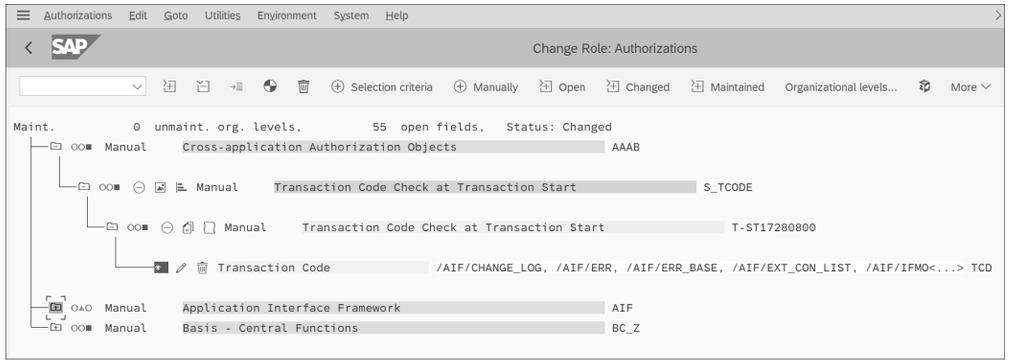


Figure 3.6 Manually Inserted Authorizations from a Template

In addition to using templates as a reference, you can also refer to SAP-delivered standard roles that are available in Transaction PFCG. SAP standard roles start with the prefix "SAP_".

3.2.5 Comparison of the Role Design Concepts

Table 3.2 provides a detailed overview of the pros and cons of the various role design concepts and is a product of years of experience and best practices. Earlier, Figure 3.4 showed a proposed model for building your own roles.

Requirements	Enabler Roles	Single Roles (with Optional Derivation)	Composite Roles
Effort of initial role definition	Well supported	Well supported	Partially supported
Possibility to optimize grouping of applications	Partially supported	Well supported	Partially supported
Possibility to individualize role assignments	Not supported	Well supported	Partially supported
Reduction of Functional/organizational redundancies	Partially supported	Well supported	Partially supported
Effort of functional role maintenance	Not supported	Well supported	Not supported
Effort of organizational role maintenance	Partially supported	Partially supported	Partially supported
Effort of user assignment maintenance	Not supported	Partially supported	Partially supported
Compliance in data protection at role level	Well supported	Well supported	Well supported
Compliance with SoD at the role level	Partially supported	Well supported	Well supported
Possibility of role mining	Partially supported	Partially supported	Not supported
Cascading of SAP role design	N/A	Well supported	Not supported
Reporting (auditing, transparency)	Not supported	Well supported	Partially supported

Table 3.2 Comparison of Different Role Designs

Requirements	Enabler Roles	Single Roles (with Optional Derivation)	Composite Roles
Upgrade capability (e.g., enhancement packages [EHPs], SAP S/4HANA)	Not supported	Well supported	Well supported
Maintenance after role creation	Not supported	Well supported	Not supported

Table 3.2 Comparison of Different Role Designs (Cont.)

As shown in Table 3.2, building single roles (with optional derivation) is considered a best practice for SAP role designs. This approach offers the greatest flexibility, transparency, and the broadest support in terms of requirements and upgrades. On the other side of the spectrum, you might use enabler roles, which are the least flexible and have several design and maintenance disadvantages.

3.3 Segregation of Duties

No matter the approach you choose, you should always consider SoD during your role design. SoD is an internal risk management control used to separate critical functions of a business process among different individuals. SoD is an effort to lessen risks and fraud by preventing one individual from having too much authorization within a business process. A good demonstration of SoD would be separating the responsibilities of creating a purchase order and the receiving of goods to two different individuals. Leaving both purchasing tasks to one person can lead to increased risk of fraud because no control is in place to prevent an individual from engaging in fraud.

Keeping SoD in consideration during role design helps improve security by reducing the area where risks can occur and be exploited. Depending on your company's architecture, modeling the design of your SoD process can be time consuming and cumbersome. Thus, be mindful when creating your SoD policies and risk levels. SoD risk levels help your business prioritize risk mitigation. An example of risk levels could be critical, high, medium, or low. A critical risk in most cases would be prioritized over a medium- or low-level risk. This classification scheme helps a business control how and when risks are dealt with. Knowing the organizational risks and SoD rules can help you calculate the number of roles that need to be created when segregating access on a role level, as shown in Table 3.3. Regardless of which role design approach you choose, SoD should be an integral component of your role design process.

Number of Risks	Number of SoD Functions	Number of Organizations (Departments)	Calculation of Number of Single Roles
20	15 to 30	10 company codes	15 × 10 = 150 30 × 10 = 300
50	40 to 75	10 company codes	40 × 10 = 400 75 × 10 = 750

Table 3.3 Role Calculation Based on SoD

A few key aspects to keep in mind when considering SoD in your role concept include the following:

- Keep risks low. Discuss the necessity of specific risks with your internal and external auditors. More risk leads to more separation, and thus, more roles are required.
- Define the criticality of the specific risks by using risk levels (critical, high, medium, low, etc.). This approach helps you prioritize risks.
- Encapsulate SoD functions in separate roles only when risks have been defined with the highest criticality.
- Encapsulate SoD functions in separate roles only when risks have no defined mitigation. If mitigations and compensating controls are available, mitigate the risk rather than split on the role level.
- Encapsulate SoD functions in separate roles only when a functional separation is possible and can be implemented in the future organizational and procedural framework.

SoD can have a huge influence on role design, and you must determine on which level the separation should take place. At the end of the day, SoD conflicts matter on the user level but must be separated on the role level in order to fully remediate. However, depending on the risk and whether mitigations are available, SoD conflicts should be managed on the role level.

3.4 Determining When to Use Enabler Roles

The enabler role concept is a non-standard approach to role design. Its origins are based on the assumption that you cannot create a role that can display all items but only change a subset. Technically, this impossibility is because an authorization object with two or more fields will check all of them and require that they are present in the same role authorization.

The following section describes a simple example of an enabler role for Transaction FK03 (Display Vendors). The single role contains the transaction (a transactional single role), and another single role contains the organizational values (the enabler role).

The transactional single role contains Transaction FK03 with all its authorization objects and values, as shown in Figure 3.7. The organizational level (**BUKRS** in this example) for object **F_LFA1_BUK** is left empty, or the authorization object is deactivated.

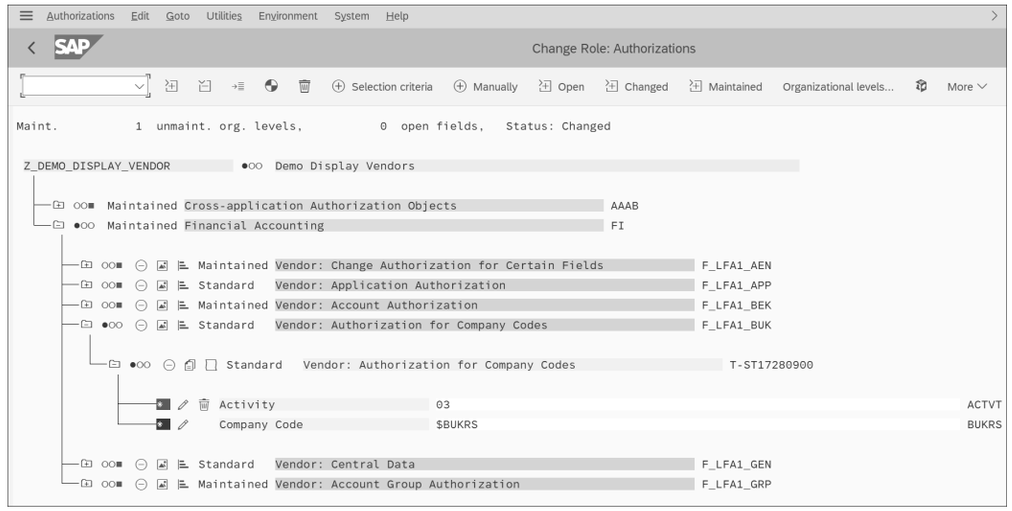


Figure 3.7 Example of the Transactional Role for Transaction FK03

A second role, the enabler role, contains the organizational values for authorization object F_LFA1_BUK only, as shown in Figure 3.8. No other authorizations—other than the objects with organizational values—are maintained in this enabler role.

Figure 3.8 Enabler Role Containing Object F_LFA1_BUK

When the two roles are assigned to a user, you can check the user buffer to see the totality of assigned authorizations. In the user buffer (Transaction SU56), shown in Figure 3.9, you'll see the two authorization profiles assigned to the user, with two instances of the authorization object **F_LFA1_BUK** Notice how the **ACTVT** value **03** comes from the transactional single role, but the company code comes from the enabler role.

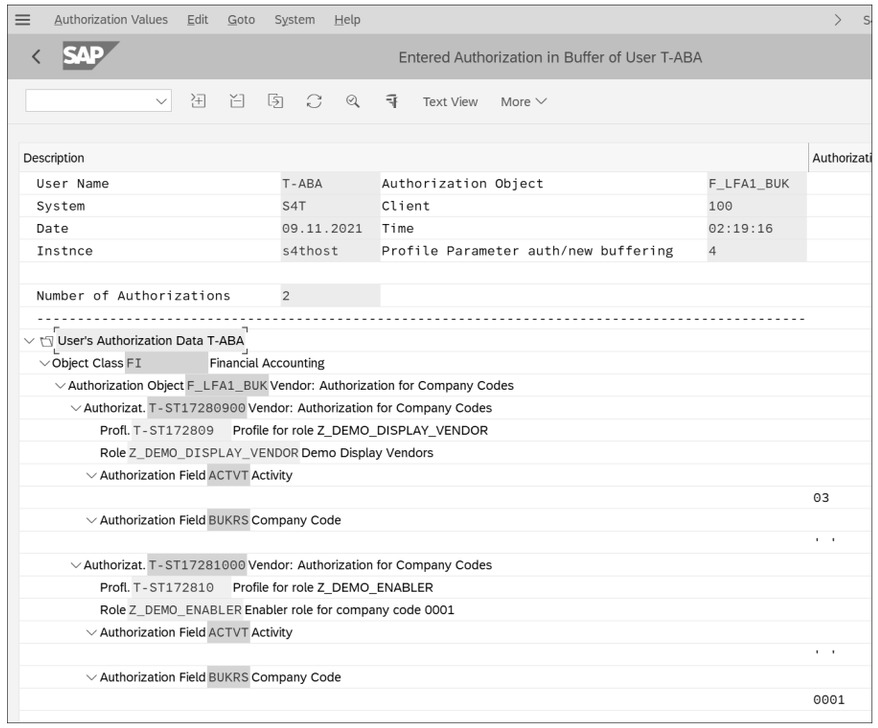


Figure 3.9 User Buffer of User with Both Roles Assigned

Since the two authorizations are in different authorization profiles, the authorization check does not succeed. This failure is because, when the kernel performs an authority check, it checks the values of one authorization instance. As shown in Figure 3.10, the user can still successfully launch Transaction FK03 but is missing the authorization for company code 0001.

At this point, the user can successfully execute the transaction but cannot view vendors in company code 0001, even though the enabler role contains the company code. Since the kernel checks the activity and the company code in the same authorization instance, the authorization check was unsuccessful. Therefore, you cannot have activity in one role and the organizational field in another and expect that the check will be successful as a whole.

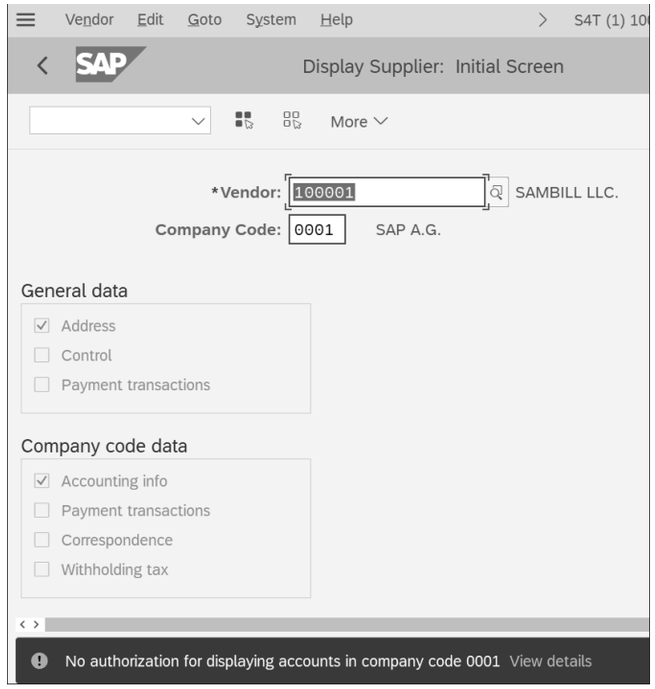


Figure 3.10 Authorization Error while Accessing Company Code 0001 in Transaction FK03

To counter this issue, you must authorize your enabler roles with the activity fields that control the company code. In this particular case, you must ensure that authorization object F_LFA1_BUK is restricted to display access (**ACTVT** = **03**) because, otherwise, you'll overauthorize these users. This potential issue is a main concern when using enabler roles since your enablers tend to carry far-reaching authorizations that may cause critical access and SoD conflicts.

Another downside to this approach is the fact that, in typical enabler concepts, the enabler roles contain almost every authorization object that has organizational fields. Since many critical business authorization objects within SAP contain organizational fields, often all these fields are added to the enabler role, since the goal of the enabler concept is to keep the number of enablers low.

In addition to this complexity, enabler role designs can also lead to several other issues, such as the following:

- You're breaking with the SAP standard. Transaction SU24 is used to propose required authorization objects into Transaction PFCG. Breaking this standard and manually inserting objects will impact upgrades, security patches, complexity, where-used list reporting, and more.

- You must use manually inserted objects with enabler roles instead of relying on solid Transaction SU24 proposals for the activity type of fields. As a result, authorization objects and values in the role profiler are not related to an application. You cannot use the where-used list to see why the authorization object was added in the first place. Over time, as role requirements change, typically obsolete authorization values become a mess that no longer matches their original intention to "enable" the transaction.
- Maintenance of roles after creation is typically more labor intensive than a derived role design due to the lack of automation. The number of enabler roles can explode such that there are more roles than there are users in the system.
- You cannot run SoD analysis on the role level with governance, risk, and compliance (GRC) tools like SAP Access Control or check on S_TCODE and the related authorizations of other objects. If transactions, activities, and organizational authorization objects are separated, you would need to simulate the correct pairing of roles to have the full set of authorizations to be assigned to a user. Please note that SoD conflicts matter the most when assigned to a user. At this point, enabler role concepts are tempted to use composite roles for role assignments, in this way adding the additional disadvantages of composite roles on top of enabler roles.
- Enabler role designs increase the complexity of role mining during the user provisioning process. For a requestor who has to request access, finding two roles is significantly harder than finding one. Especially when using tools like SAP Identity Management (SAP ID Management) or SAP Access Control, a requestor must both understand the concept as well as identify and find matching pairs of roles.
- Role testing becomes exponentially more complex since you'll need to test pairs of roles. This effort continues to be required as an organizational structure changes and roles must be tested again to ensure they reflect these changes.

Most of these issues arise because of missing Transaction SU24 content for what the application needs. Over time and by necessity, manually inserted authorization objects become obsolete, become incorrect, and overauthorize users. Since manually inserted objects are not attached to a transaction, no easy way exists to identify to which transaction they belong and why they were added to the enabler role in the first place. In the long run, maintaining and upgrading these roles is a nightmare.

If composite roles are additionally used for user assignments, then a cascading problem occurs, and changing an enabler role is nearly impossible—thus, administrators typically must create more new enabler roles to compensate.

3.5 Role Naming Convention

Before you start building your roles, we recommend spending some time developing a role naming convention that suits your organization. A common mistake is building roles without any convention, thus resulting in a messy design that is hard to maintain in the long run. Also, the lack of naming convention increases the complexity of finding and assigning roles to end users. Especially when companies use tools such as SAP ID Management or SAP Access Control for role provisioning through workflows where a requester has to enter the roles that shall be assigned to a user.

Table 3.4 is an example of a role naming convention that has been implemented successfully with many clients.

Character Number	Description	Examples
1	Fixed value "Z" for customer namespace	
2	Identifier for the system level	P – Production D – Development T – Test S – Sandbox Q – Quality I – Integration
3	Fixed character "_" for separation	
4	Identifier for role type	S – Single role C – Composite role M – Master role
5	Identifier for user group	C – "Common" role for basic functions for all users E – end user role for job function I – IT roles for administrators, developers, etc. T – Technical roles for remote function calls (RFC), batch jobs, Web Services, etc. R – Reference user roles
6	Fixed character "_" for separation	

Table 3.4 Example of a Role Naming Convention

Character Number	Description	Examples
7 – 8	Identifier for business area	BC – Basis BI – Business Intelligence CR – Customer Relations CO – Controlling EC – ERP Component FI – Finance HR – Human Resource MM – Material Management PI – Process Integration PS – Project Systems SD – Sales & Distribution SM – Solution Manager SR – Supplier Relations
9	Fixed character "_" for separation	
10	Identifier for leading organizational element	A – All N – None C – Company Code P – Purchasing organization P – Plant D – Division
11	Fixed character "_" for separation	
12 – 15	Organizational value in regard to the organizational element	A – GLOB (all) N – GLOB (none) C – $BUKRS (value from company code) S – $VKORG (value from sales organization) P – $WERKS (value from plant)
16	Fixed character "_" for separation	

Table 3.4 Example of a Role Naming Convention (Cont.)

Character Number	Description	Examples
17 – 19	Identifier for activity group in regard to the user group	COM – Common job authorizations REP – Reporting with display only authorization EAM – emergency access authorizations KEY – Key user authorizations MDM – master data authorization ADM – administration authorization HPU – High-privileged user (HPU) authorizations
20	Fixed character "_" for separation	
21+	Free text	

Table 3.4 Example of a Role Naming Convention (Cont.)

Based on the role naming convention in Table 3.4, a few practical examples might resemble the following role names:

- `ZP_SI_BC_N_GLOB_HPU_OPEN_CLIENT`
  A single role for a productive system (PRD) that does not include any organizational data and is considered an HPU role to open clients.
- `ZP_CE_FI_C_1000_REP_ACCOUNTANT`
  A composite role for a PRD that includes organizational values for company code 100 with display-only access for reporting purposes, typical for an accountant.

With this kind of role naming convention, not only can an administrator easily understand a number of roles, so can an end user. Also, since identifiers are defined and follow a specific pattern, role analysis can be simplified. For example, a role that contains "REP" in characters 17 to 19 is a reporting-only role that should not include any insert, change, update, or delete authorizations. For role quality and sustainability purposes, a strictly enforced role naming convention is therefore mandatory.

3.6 Summary

The decision to (re)design a company's authorization concept can be stem from a variety of reasons. Some main drivers include system upgrades (e.g., migration to SAP S/4HANA); remediating compliance issues (e.g., SoD, legal requirements); mergers and acquisitions that lead to integration projects; or because the current design does not meet the desired security standards of your company. When choosing between the types of approaches (top-down or bottom-up), be mindful of how each can affect your

authorization concept design. Additionally, some areas to take note of when designing your company's authorization concept include the following:

- Existing SoD policies
- Role naming conventions
- Job- or task-based roles
- Knowing when to use single, derived, composite, enabler, or template roles

In the next chapter, we'll discuss the Xiting Authorizations Management Suite (XAMS).

Contents

12 Migrating Authorizations to SAP S/4HANA 497

13 Migrating Authorizations to SAP S/4HANA with the Xiting Authorizations Management Suite 567

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

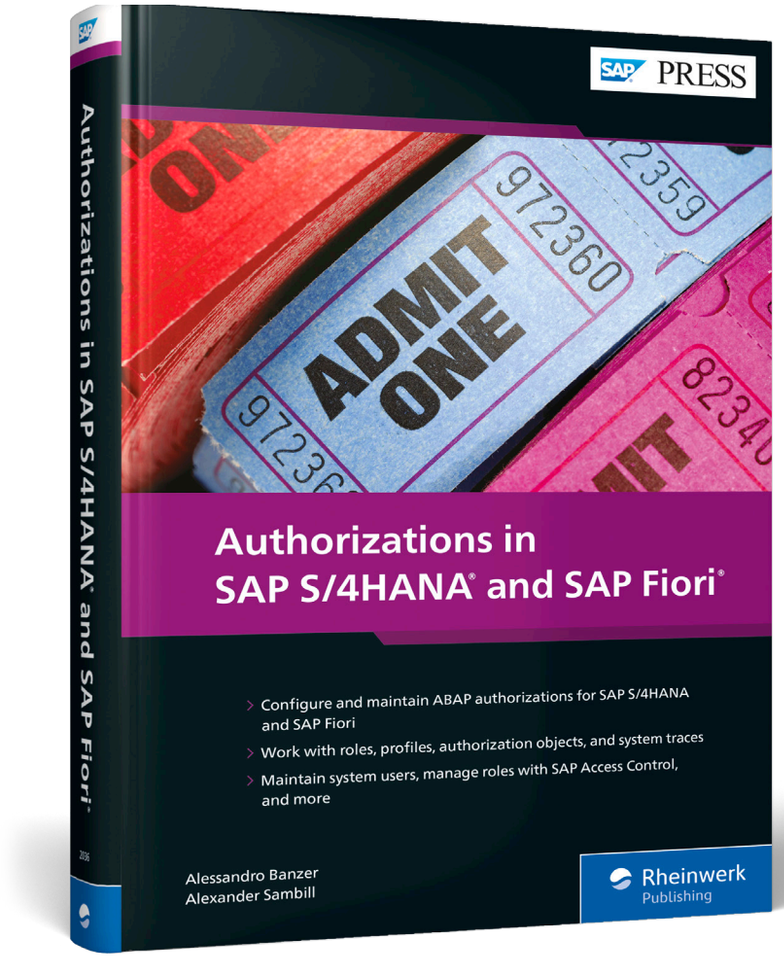
P

Q

R

S

T



Alessandro Banzer, Alexander Sambill

Authorizations in SAP S/4HANA and SAP Fiori

625 pages, 2022, $89.95
ISBN 978-1-4932-2036-6

www.sap-press.com/5203

Alessandro Banzer is the chief executive officer of Xiting. He has worked in information technology since 2004, specializing in SAP since 2009. Since then, Alessandro has been involved with global SAP projects in various roles. Alessandro is an active contributor and moderator in the governance, risk, and compliance space on SAP Community, as well as a speaker at SAPPHIRE, ASUG, SAPinsider, and other SAP-related events. He holds a degree in business information technology, as well as an Executive Master of Business Administration from Hult International Business School in London, UK.

Alexander Sambill is a senior SAP security consultant and certified SAP trainer at Xiting Germany. He is a security-minded professional with consulting and sales experience in many industries. During his years of work within SAP security, he specialized himself for SAP authorizations in SAP ERP and SAP S/4HANA with SAP Fiori. Alexander leads authorization migration and redesign projects for small and large enterprises, educates customers, and solves individual custom use cases. He is also a federally certified instructor (IHK) in commerce and industry. Alexander is a passionate writer and active blogger of technical and scientific articles, e-books, white papers, surveys, and more about SAP security and authorizations. He is also the content manager of publications for SAP authorizations at Xiting AG.