PUBLIC

Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions

Guidelines for extension project managers, key users, and ABAP developers

VERSION 2.2 / APRIL 2024

Activity log

Version 2.2, April 2024
Repaired broken links in chapter 4.1
Version 2.1, April 2024
Minor changes in text for clarification and better understanding.
Version 2.0, April 2023

CHAPTER	NOTE
Document	Activity log added.
4.2	Added chapter about analytics in ABAP Cloud.
4.6	Updated recommendations for BC apps for SAP S/4HANA Cloud Private Edition, and SAP S/4HANA on-premise.
5.1	Updated recommendations for the setup of the 3-tier model.
5.2	Update on recommendations for wrappers.

Table of Contents

1 INTRODUCTION

Extensibility is a key capability of every Enterprise Resource Planning (ERP) solution. It enables customers to create a competitive advantage by customizing their business processes and allows partners to enrich ERP with tailor-made solutions. The importance of extensibility has been confirmed for SAP's on-premise ERP and will remain valid for the more standardized cloud ERP.

SAP S/4HANA is SAP's flagship product providing the intelligent ERP in the cloud and on-premise. This document helps SAP S/4HANA customers and partners to choose, implement and use the extensibility options correctly, and takes into account the various customer environments (public cloud, private cloud or on-premise).

The goal is to move from classic custom ABAP extensions to an SAP S/4HANA extensibility model that allows customers to consume SAP innovations smoothly, leading to future-proof extensions that are ready for the next cloud transformation steps.

1.1 Motivation for a new extensibility model

During the last decades SAP's on-premise customers and partners have mainly used classic ABAP extensibility to extend their ERP solution. Classic extensibility allows ABAP developers to use and even to modify all SAP objects. This is very powerful and flexible. But the missing clear interface between SAP code and the extensions adds a lot of customer specific test and adaptation effort during SAP upgrades.

In the public cloud there are no customer-specific SAP upgrade projects. Instead automated software updates run for all customers in parallel. Therefore, classic extensibility, based on classical custom ABAP code, can no longer be used.

SAP S/4HANA Cloud Public Edition provides a new upgrade-stable cloud extensibility model that clearly separates SAP code and extensions via mandatory public SAP APIs and SAP extension points. It supports the following standard extensibility scenarios:

- **On-stack** extensions that depend on proximity to or tight coupling with SAP S/4HANA Cloud data, transactions or apps and therefore run on the SAP S/4HANA Cloud stack.
- **Side-by-side** extensions running on the separated SAP Business Technology Platform (SAP BTP) for all other loosely-coupled extension scenarios integrating with SAP S/4HANA.

The different personas working on extensibility use cases are supported via:

- **Low-code/no-code tools** for key users/business experts.
- A **development environment** (IDE) for professional developers.

SAP S/4HANA Cloud covers these dimensions with the following extensibility options:

SAP S/4HANA CLOUD EXTENSIBILITY OPTIONS		
PERSONAS	**Tightly coupled extension on SAP S/4HANA (on-stack extensions)**	**Loosely-coupled extension on the side-by-side Platform SAP BTP (side-by-side extensions)**
BUSINESS EXPERT, KEY USER, CITIZEN DEVELOPER	SAP S/4HANA Cloud key user extensibility	Low-code/no-code solutions (SAP Build apps, SAP Business application studio, SAP Build Process automation and others)

SAP S/4HANA CLOUD EXTENSIBILITY OPTIONS		
PERSONAS	**Tightly coupled extension on SAP S/4HANA (on-stack extensions)**	**Loosely-coupled extension on the side-by-side Platform SAP BTP (side-by-side extensions)**
DEVELOPER	Developer extensibility using the SAP S/4HANA Cloud ABAP Environment (a.k.a. Embedded Steampunk[1])	Java, Node.js SAP BTP environments SAP BTP ABAP Environment (a.k.a. Steampunk[1])

Table 1.1 - SAP S/4HANA Cloud extensibility options

All these extensibility options use only SAP APIs and SAP extension points that have been released and are stable.

This new SAP S/4HANA Cloud extensibility model, first introduced in SAP S/4HANA Cloud Public Edition, is now available in all SAP S/4HANA editions, to achieve smoother upgrades everywhere and to pave the way to the cloud.

Table 1.2 below lists the different SAP S/4HANA editions and why a customer should adopt the new cloud extensibility model in the respective environment. The right column in the table lists which topics are covered in this guide.

	WHY CLOUD EXTENSIBILITY?	WHAT WILL BE EXPLAINED
SAP S/4HANA Cloud Public Edition	**Mandatory** Classical ABAP extensibility is not supported	• The SAP S/4HANA cloud extensibility model • When to use which cloud extensibility option • The ABAP Cloud development model
SAP S/4HANA Cloud Private Edition and on-premise **Greenfield installation**	**Start with the superior cloud extensibility model** Smoother SAP software updates Future-safe extensions for the next cloud transformation steps	• Technical setup to use the new cloud extensibility model • When to use the cloud extensibility model and when classic extensibility still has to be used • Guidance on how to handle classic extensibility in parallel with the new cloud extensions
SAP S/4HANA Cloud Private Edition and on-premise **Installation with converted classic extensions**	**Get the benefits described above for new and existing extensions** Developers learn the new extensibility model and can transform classic extensions step-by-step to cloud-ready extensions	• Best practices on how to manage, eliminate or transform existing classic extensions

Table 1.2 - Overview about the positioning of the new cloud extensibility model in the different SAP S/4HANA editions

[1] Steampunk and Embedded Steampunk are the SAP internal project names to deliver the ABAP environment on SAP BTP and in SAP S/4HANA Cloud. These project names are often used in development-related blogs and articles.

The list below gives some recommendations on how to read this guide:

- Chapters 1-3 are recommended for all readers to get an overview of the ABAP-related aspects of the new cloud extensibility model and to understand when to use which extensibility option.
- Chapter 4 introduces the new ABAP Cloud development model.
- Chapters 5 and 6 provide initial guidance on how to apply the cloud extensibility model in SAP S/4HANA Cloud Private Edition or on-premise. Here we differentiate between greenfield SAP S/4HANA systems and converted SAP S/4HANA systems containing existing custom ABAP code.

This guide concentrates on the ABAP-based extensibility options (key user and developer extensibility on SAP S/4HANA and the SAP BTP ABAP Environment).

For low-code/no-code tools on SAP BTP please refer to Low-Code/No-Code Development Tools in SAP community.

More information on Node.js and Java development on SAP Business Technology Platform as well as the SAP Cloud Application Programing Model (CAP) is referenced on the More Information page at the end of this document.

2 THE SAP S/4HANA CLOUD EXTENSIBILITY MODEL

SAP S/4HANA Cloud Public Edition allows customers and partners to take full advantage of all the innovations delivered by SAP without the need to take responsibility for the cloud infrastructure and operations. SAP manages all operation and lifecycle management tasks such as continuous feature delivery or providing hotfixes and regular upgrades to new software versions.

Consequently, the new cloud extensibility model as explained in chapter 1 is mandatory to ensure that customer and partner extensions continue to run without any change or adaptation effort independent of changes made by SAP.

Therefore, all extensions must adhere to the following rules to ensure cloud-readiness:

- **Rule 1** - Extensions can only use released remote or local SAP APIs. SAP keeps these APIs stable.
- **Rule 2** - SAP objects can only be extended via predefined extension points. SAP keeps these extension points stable. Modifications of SAP objects as in on-premise systems are no longer supported.
- **Rule 3** - Extensions can only use cloud-enabled and released technologies.

Many benefits of the classic extensibility model known from the on-premise world have been preserved:

- Extensions are built using the exact same programming model that SAP uses to develop the standard applications.
- The ABAP code of the SAP applications can be analyzed and inspected by customers and partners. This allows seamless end-to-end debugging of the extensions and provides a quick learning path for extension projects.

2.1 Overview of the extensibility options for SAP S/4HANA Cloud

In this chapter we will introduce the three ABAP-based extensibility options for SAP S/4HANA Cloud editions that follow the rules of the cloud extensibility model (→ Figure 2.1).

Type 1 and 2 extensions run directly on the SAP S/4HANA Cloud stack. They are implemented based on the technology stack of the core solution (ABAP Platform for SAP S/4HANA):

- Key user extensibility (1) for low-code/no-code extensions created by key users, such as adapting the user interface or adding custom fields.
- On-stack developer extensibility (2) using the SAP S/4HANA Cloud ABAP Environment for developer extensions that are implemented in ABAP directly on the SAP S/4HANA Cloud technology stack.

Type 3 extensions run side-by-side to the core on SAP BTP:

- Side-by-side extensibility (3) for developer extensions using the development and runtime environments offered by SAP BTP. For ABAP this is the SAP BTP ABAP Environment.

All three extensibility options are based on the usage of public SAP interfaces providing access to public SAP APIs and SAP extension points.

Key user extensions and on-stack developer extensions have access to **local** public APIs released by the underlying ABAP Platform or the SAP S/4HANA Cloud applications. These APIs can be released for key user extensibility, for on-stack developer extensibility or for both options.

Side-by-side extensions can access SAP S/4HANA Cloud business objects through **remote** SAP APIs (e.g., OData services). In addition, these side-by-side extensions on SAP BTP ABAP Environment have access to the exposed local public interfaces of the underlying ABAP Platform stack.

Both SAP BTP ABAP Environment and SAP S/4HANA Cloud ABAP Environment are based on the same ABAP Platform stack and use the same local ABAP Platform interfaces.

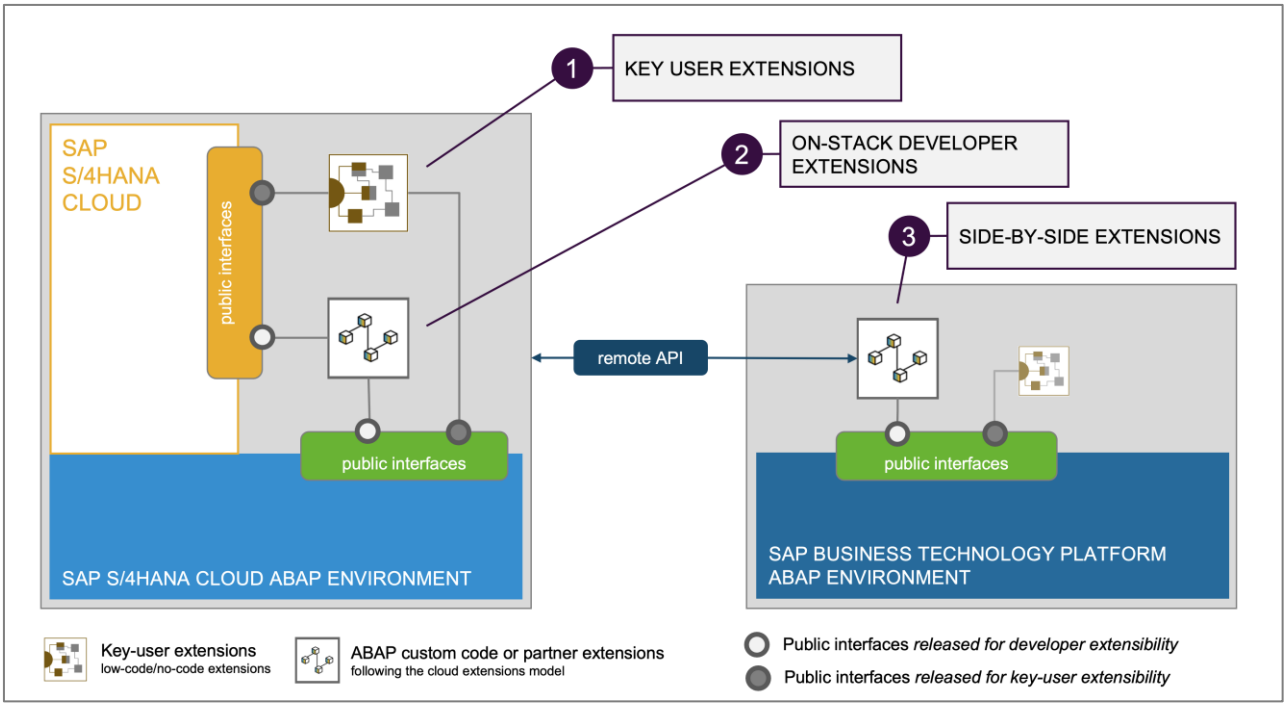

Figure 2.1 - Overview of extensibility options

The three extensibility options are not isolated from each other. In many scenarios they are combined, for example developing a side-by-side application in conjunction with a thin on-stack extensibility layer that offers more suitable remote APIs to access the SAP S/4HANA Cloud functionality.

2.2 Key user extensibility

SCENARIO	Low-code/no-code adaptations and extensions of SAP S/4HANA applications
USE CASES	Adapting UIs, adding custom fields, adding custom business objects and more.
PERSONA	Business expert, implementation consultant, citizen developer, key user.

Key user extensibility (formerly also known as in-app extensibility) empowers business experts to add extensions to SAP S/4HANA Cloud solutions without the need to dive deeply into the implementation details of the underlying SAP applications. Key users typically have a deep knowledge of the SAP S/4HANA processes and business configuration. The focus is on so-called *last mile extensions* extending SAP S/4HANA user interfaces, processes, or forms using low-code/no-code tools.

Key users typically have no or only limited coding skills and therefore do not need a fully integrated development environment, with capabilities such as versioning, dependency handling, refactoring, or debugging.

The main argument for using key user extensibility is that simple extensions can be realized quicker than with developer extensibility, because the communication overhead between the business expert (responsible for specification of the extension, and later for testing and approval) and the developer (responsible for development and developer test) is avoided.

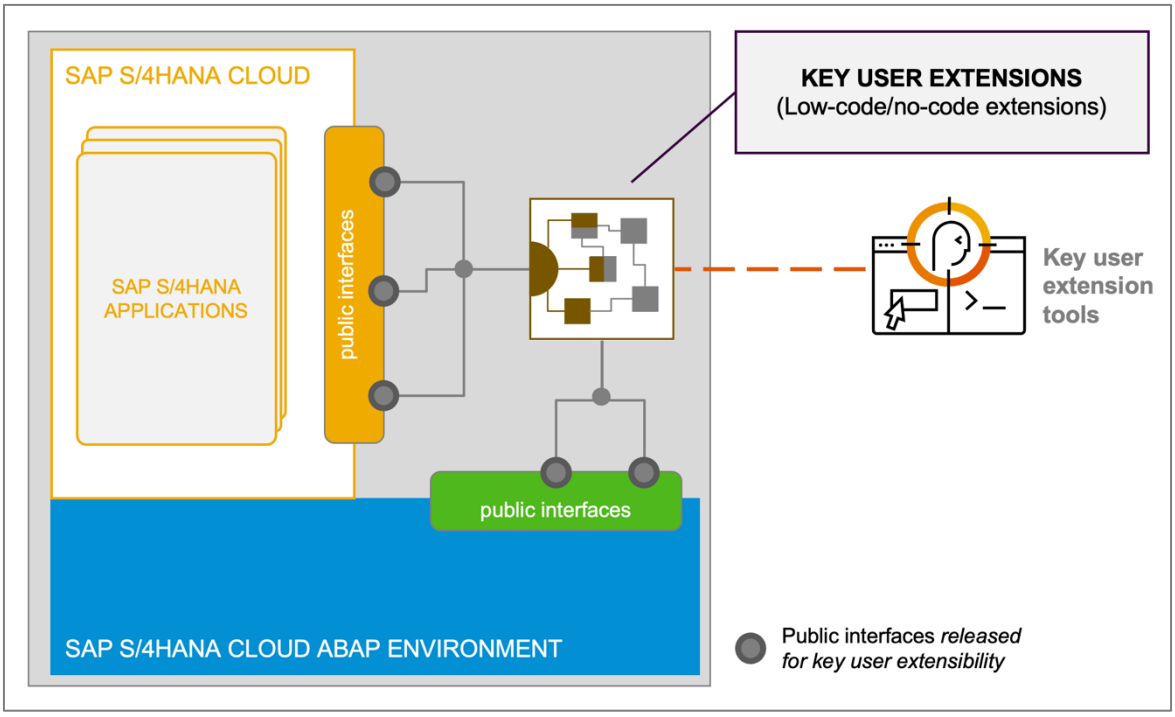

Figure 2.2 - Key user extensibility

With the provided key user tools the key user can achieve the following:

- Adapt the screen layout such as moving fields and field groups, hiding fields, changing labels etc.
- Add custom fields to business objects. The custom field is then available in the entire application stack (from the UI to the database tables or for developer extensibility).
- Add custom business objects to handle custom data with very little coding efforts.
- Add custom logic to validate the custom fields using cloud BAdIs.
- Add custom fields to a process group (e.g., from sales quotation and sales order to delivery and invoice) to provide a consistent end-to-end extensibility.
- Add custom Core Data Service (CDS) views and create new analytical applications (reports, KPIs, ...).
- Copy and adapt print and email form templates.

The adaptations made by a key user are registered in transport requests which can be propagated from a development environment to quality assurance and production.

The following screenshots illustrate typical key user tasks such as adding custom fields (→ Figure 2.3) or user interface adaptation for SAP Fiori apps (→ Figure 2.4).

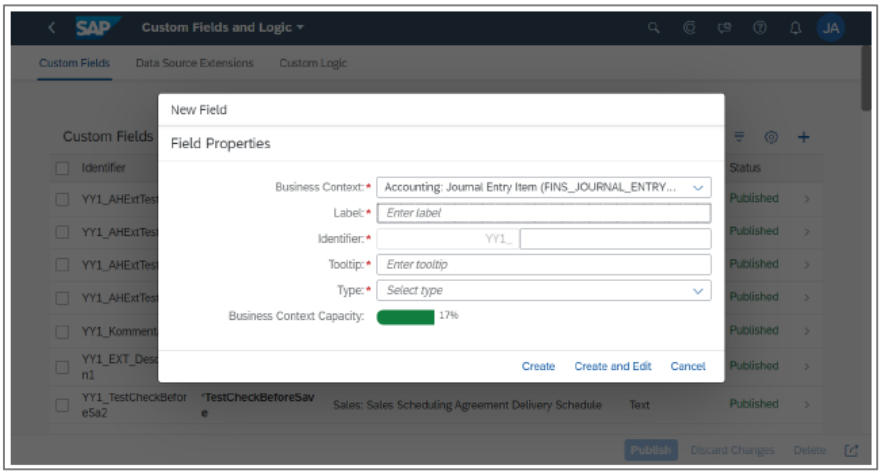

Figure 2.3 – Adding a custom field with key user extensibility

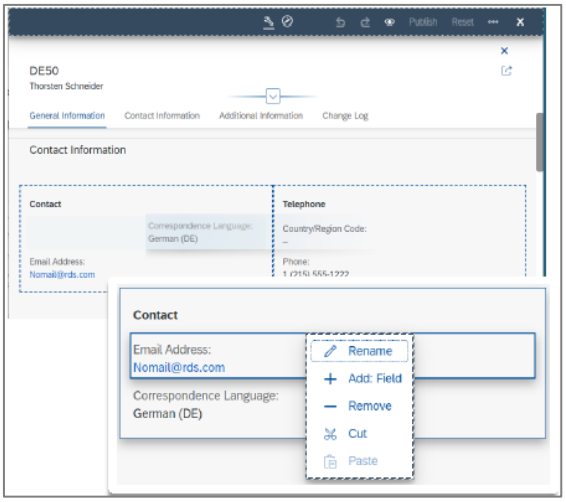

Figure 2.4 – Key user adaptation

The following white paper, blog posts and documentation provide more details on key user extensibility:

- Custom Extensions in SAP S/4HANA Implementations - A Practical Guide for Senior IT Leadership.
- The Key User Extensibility Tools of SAP S/4HANA.
- SAP S/4HANA Extensibility: A Learning Journey.
- SAP Help Portal - Extending an SAP Fiori Application.
- Extending SAP-delivered SAP Fiori elements apps using adaptation projects to create SAP S/4HANA app variants.

2.3 On-stack developer extensibility with SAP S/4HANA Cloud ABAP Environment

SCENARIO	Custom ABAP development projects that need proximity or coupling to SAP S/4HANA data, transactions, or apps
USE CASES	Custom applications with frequent or complex SQL access to SAP S/4HANA data. Custom extensions running in the same logical unit of work[2] (LUW) as SAP applications. Tailored custom remote APIs or services which serve side-by-side SAP BTP apps.
PERSONA	ABAP developer.

On-stack developer extensibility[3] is intended for development projects requiring proximity to or coupling with SAP S/4HANA data, transactions, or apps. The requirements of the extension project go beyond the scope of key user extensions and therefore require full access to development capabilities like debugging, refactoring support, version control etc.

In contrast to side-by-side extensions, on-stack extensions are developed and run on the same software stack as the underlying SAP S/4HANA Cloud system. This allows extensions to access SAP S/4HANA logic and data via SAP extension points, local SAP APIs or via SQL queries.

Typical on-stack scenarios are extensions with frequent or complex SQL access to SAP data (e.g., SQL queries that join customer and SAP data), which cannot be realized easily by remote data access or data replication. Another typical on-stack pattern are extensions that store custom data in the same logical unit of work[3] as the extended SAP S/4HANA app.

[2] A sequence of programming steps and data updates distributed across multiple work processes, whose database changes are updated within a single database commit.
[3] On-stack developer extensibility uses the SAP S/4HANA Cloud ABAP Environment.

On-stack developer extensibility allows ABAP developers to connect to an SAP S/4HANA Cloud system using the ABAP Development Tools (→ Figure 2.5). This feels almost like developing custom ABAP code on an SAP S/4HANA on-premise system. On-stack developer extensibility offers the standard ADT tool support like ABAP Unit, ABAP Test Cockpit, ABAP profiler, ABAP debugger and the well-known SAP lifecycle management (change and transport system).

However, with SAP S/4HANA Cloud ABAP Environment, extensions are developed using the new ABAP Cloud development model (→ Chapter 4). The ABAP Cloud development model ensures that no SAP object is modified and that only local public SAP APIs and public SAP extension points are used in the extensions.

SAP S/4HANA Cloud and the ABAP Platform offer a large set of local APIs and extension points which can be used in on-stack extensions. Developers can explore these in the *released objects tree* of ADT (→ Figure 2.6). Additionally, the SAP API business hub[4] provides a section for developer extensibility and SAP CDS views showing the core local SAP APIs.

Depending on the use case, one of these options can be used to expose the functionality built using on-stack developer extensibility on an SAP Fiori UI:

- Create a custom SAP Fiori elements or freestyle SAPUI5 app using SAP Fiori tools.
- Extend an SAP-delivered app, e.g., with additional fields using Developer Adaptation[5].
- Extend an SAP-delivered SAP Fiori elements app using ABAP CDS annotations or XML annotations.

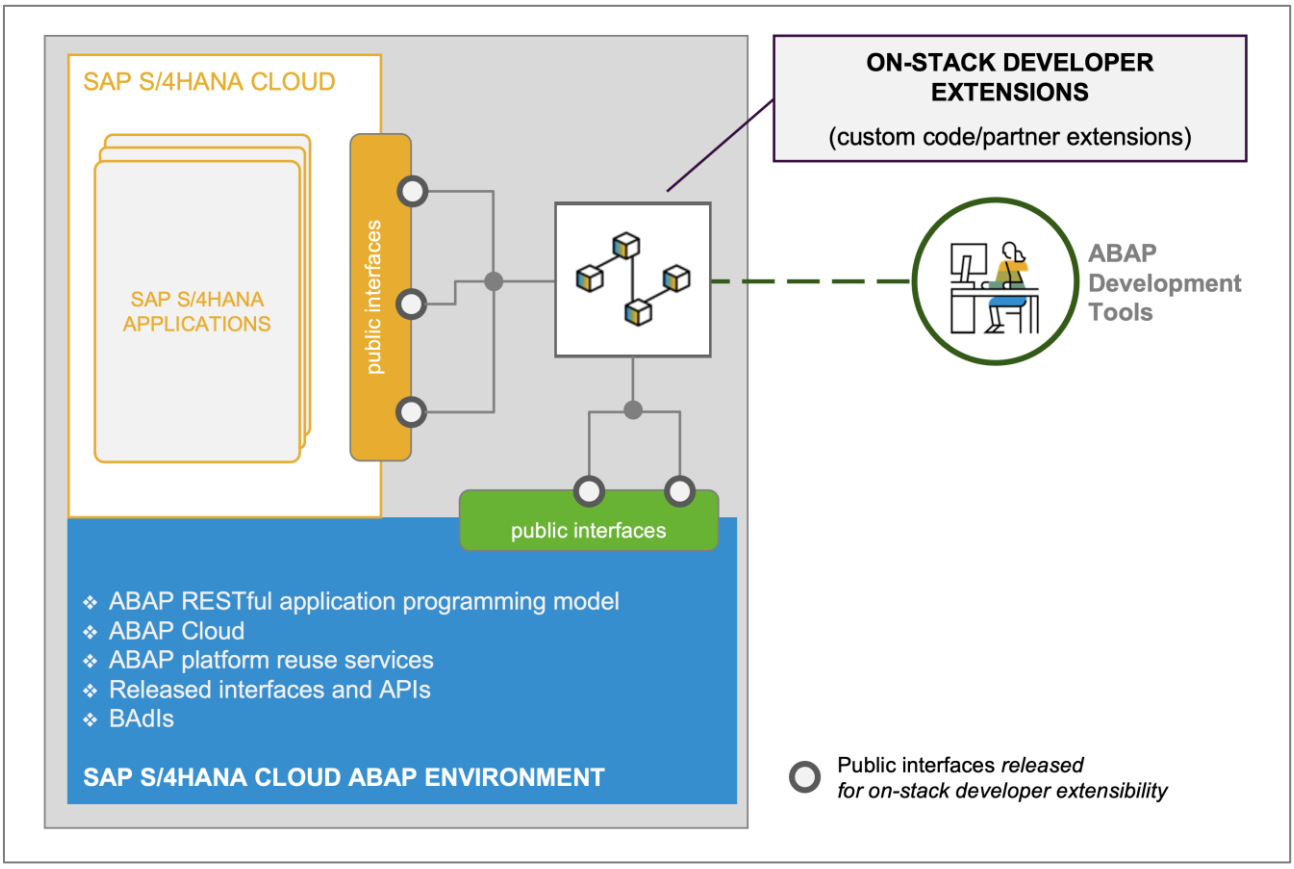

Figure 2.5 - SAP S/4HANA Cloud ABAP Environment

[4] Business Object Interface | SAP S/4HANA Cloud | SAP API Business Hub.
[5] Note that Developer Adaptation is not yet available for SAP S/4HANA Cloud.

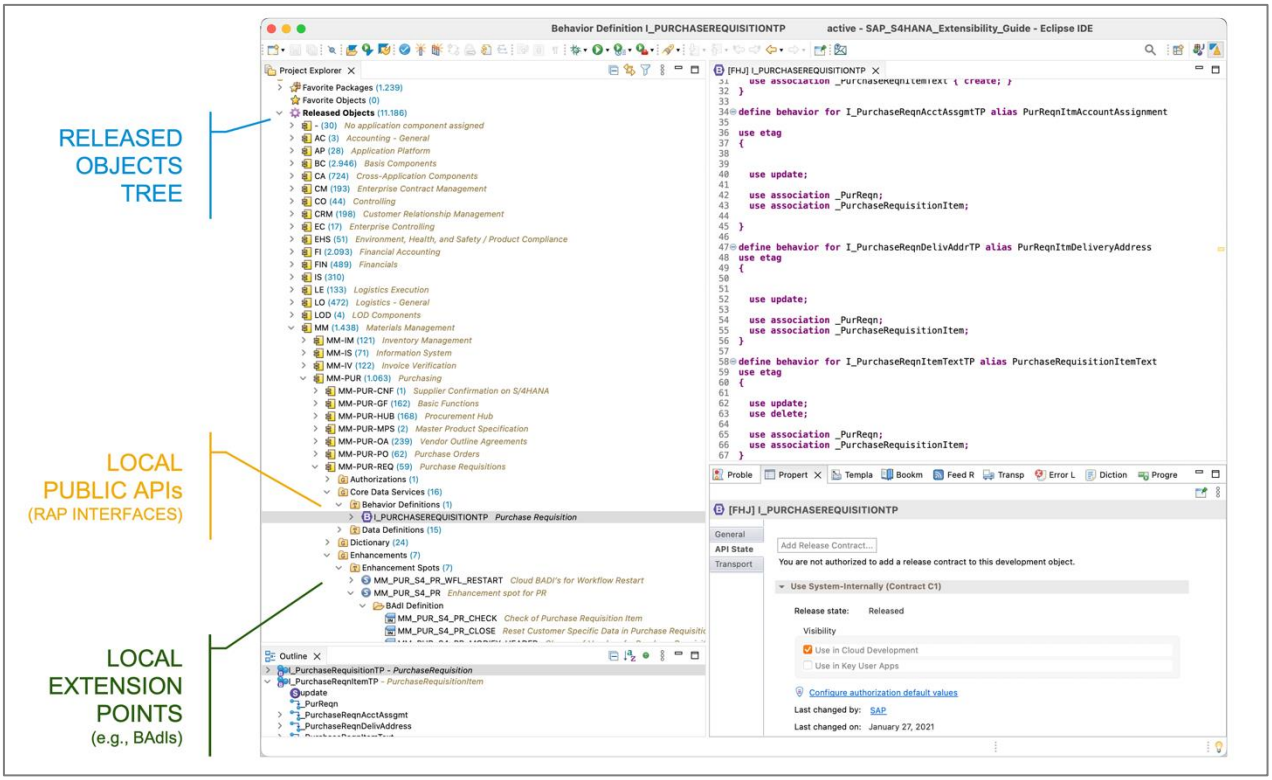

Figure 2.6 - Released object tree in ABAP Development Tools

2.4 Side-by-side extensibility using the SAP BTP ABAP Environment

SCENARIO	Loosely-coupled applications and partner SaaS solutions
USE CASES	Custom applications for a separate target group (no ERP users). Custom application workload that shall run separated from ERP. Custom applications needing proximity to intelligent SAP BTP services like machine learning, AI etc. Solutions integrating with several ERP systems and cloud services. SaaS applications provided by partners.
PERSONA	ABAP developer.

The SAP BTP ABAP Environment provides the ABAP Platform as a service on SAP BTP. ABAP-minded customers and partners can reuse their ABAP skillset to build new cloud solutions, or to transform already existing on-premise ABAP assets to the cloud.

The cloud applications and extensions run side-by-side to the extended SAP S/4HANA system. This model is the preferred option for scenarios which are loosely-coupled to SAP S/4HANA data, transactions, or apps.

A typical side-by-side use case is the hub scenario. A hub solution integrates with several ERP systems and cloud services, e.g., to consolidate SAP S/4HANA data and trigger derived actions via separate cloud services. By their very nature, hub scenarios are loosely-coupled and can run side-by-side with the SAP S/4HANA systems.

Another use case is that partners want to provide a SaaS solution and therefore need to operate their service independently of SAP S/4HANA Cloud.

SAP BTP ABAP Environment is specially optimized for large SAP or partner SaaS solutions which benefit from the multitenancy offerings and services for partners to run and operate the solution.

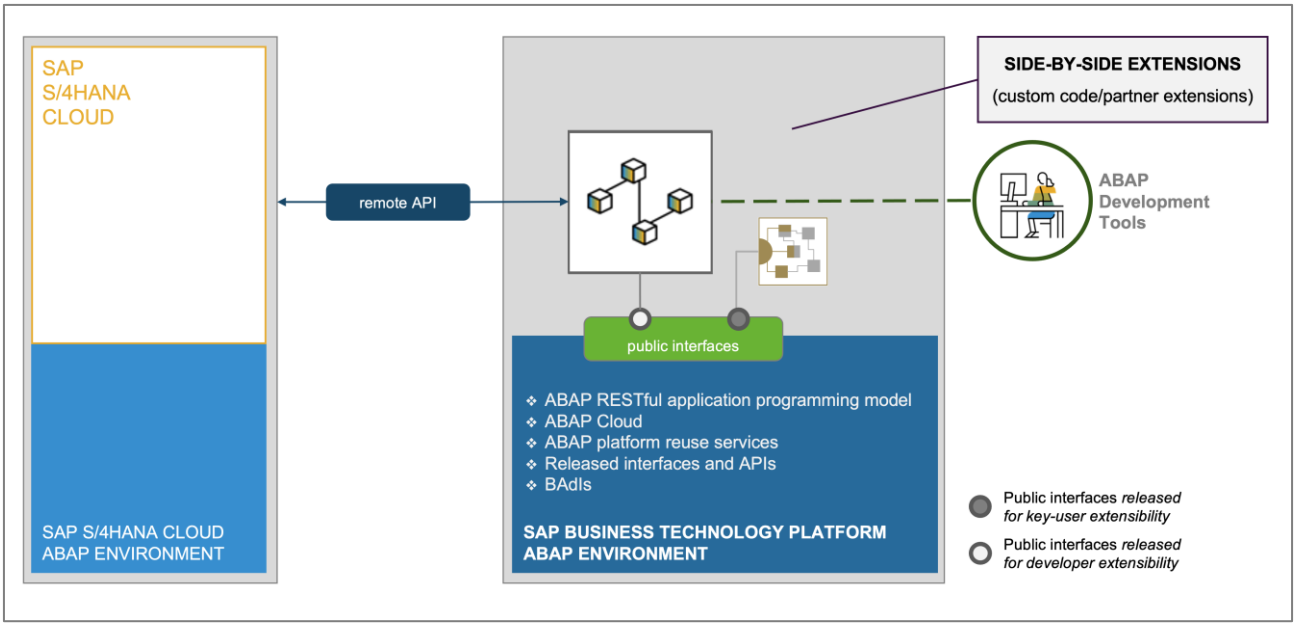

Figure 2.7 – Side-by-side extensions with SAP BTP, ABAP Environment

An ABAP developer accesses SAP BTP with the ABAP Development Tools and uses the ABAP Cloud development model (→ Chapter 4) to build SAP Fiori apps[6] or services based on the ABAP RESTful application programming model (RAP) (→ Figure 2.7). The ABAP Cloud development model ensures that only released local APIs of the underlying ABAP Platform can be used.

One main difference compared to the on-stack extensibility model is that accessing business objects of SAP S/4HANA Cloud is only possible using **remote APIs** which are published in the SAP Business Accelerator Hub (https://api.sap.com/).

2.5 Summary of the extensibility options for SAP S/4HANA Cloud

The following picture shows a recap of the three ABAP-related extensibility options which are available in SAP S/4HANA Cloud, and which were introduced in the previous sections.

	KEY USER EXTENSIBILITY Business expert, implementation consultant, citizen developer, key user	ON-STACK DEVELOPER EXTENSIBILITY ABAP developer	SIDE-BY-SIDE EXTENSIBILITY ABAP developer
SCENARIO	Low-code/no-code adaptations and extensions of SAP S/4HANA applications	Custom ABAP development projects that need proximity or coupling to SAP S/4HANA data, transactions, or apps	Loosely-coupled applications and partner SaaS solutions
USE CASES	Adapting UIs, adding custom fields, adding custom business objects etc.	Custom applications with frequent or complex SQL access to SAP S/4HANA data Custom extensions running in the same logical unit of work (LUW) as SAP applications Tailored custom remote APIs or services which serve side-by-side SAP BTP apps	Custom applications for a separate target group (no ERP users) Custom application workload that shall run separated from ERP Custom applications needing proximity to intelligent SAP BTP services like machine learning, AI etc. Solutions integrating with several ERP systems and cloud services SaaS applications provided by partners
BENEFIT	Fully managed and integrated in SAP S/4HANA Cloud No or only very basic development skills required	Development of extensions inside the SAP S/4HANA Cloud system No remote access or data replication Use and extend released SAP S/4HANA Cloud objects	Decoupled extensions independent of SAP S/4HANA Cloud operation and lifecycle management
	On-stack extension domain		Side-by-side extension domain

Figure 2.8 - Summary of ABAP-related extensibility options in SAP S/4HANA Cloud

[6] The corresponding SAP Fiori UI can be created using SAP Fiori elements or SAPUI5 freestyle.
→ Developing Apps with SAP Fiori Elements and → SAP UI5 documentation)

3 WHEN TO USE WHICH CLOUD EXTENSIBILITY OPTION

This chapter describes when to use which extensibility option. In many cases a combination of the extensibility options is needed to solve the extensibility task.

The major aspects to consider are:

- **Extension use case** - is the extension a new application, or an extension to an SAP app?
- **Extension architecture** - is the extension loosely or tightly coupled to SAP S/4HANA?
- **Extension scope & size** - who provides the extension for which purpose? For example, is the extension a small extension that is provided by key users in a line of business organization, a custom development project that is provided by a development organization, or a partner SaaS application that is provided to many customers (even independent from the core product)?

The following Figure 3.1 provides a decision tree to select the right extensibility option.

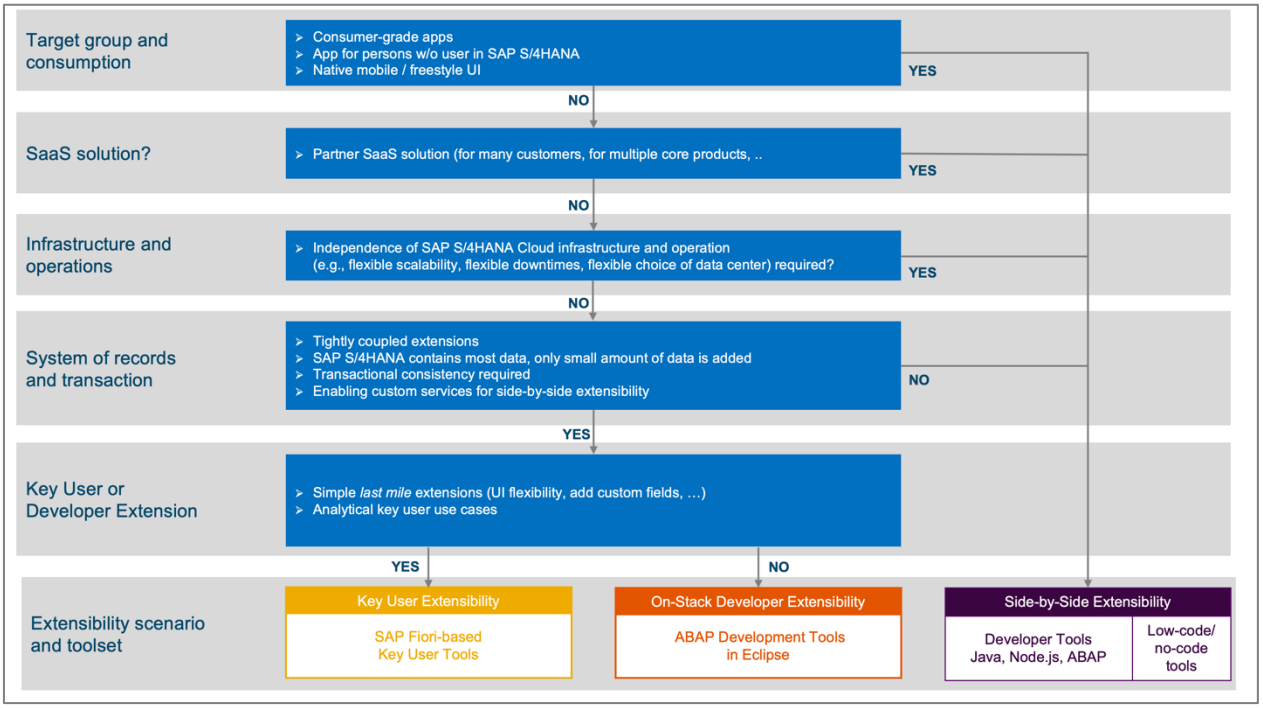

Figure 3.1 - Sequence diagram on how to find the right extensibility options.

The first three options describe typical side-by-side scenarios on SAP BTP:

- **External target group or consumption via mobile and consumer-grade UIs:** this includes native mobile apps, apps that need freestyle UI development for consumer-grade UIs, or apps for persons without a named user in SAP S/4HANA.
- **Partner SaaS solution:** a partner solution shall be provided as a cloud service operated by a partner. This use case can only be realized separated from the SAP S/4HANA Cloud operation model and is therefore always a side-by-side scenario. SAP BTP provides multitenancy concepts to minimize the cost of service and provides services to build and operate the partner SaaS solution on SAP BTP.
- **Independent infrastructure and operation:** side-by-side on SAP BTP, customers and partners are independent of the cloud infrastructure and operation model of SAP S/4HANA Cloud. The main benefits are:
  - Customers can flexibly access resources for the respective runtime environment. Resources can be scaled without impact on the core system. This means customers can physically outsource the workload and resource consumption of the extensions to safeguard the performance of their ERP system.
  - Customers can schedule the downtime of their extensions independently of SAP S/4HANA Cloud.

- Customers can choose the data center for the SAP BTP extension. This can be of interest if e.g., the co-location of the extension with certain SAP BTP services like AI or machine learning is more important than the proximity to SAP S/4HANA Cloud.
- Customers can freely choose which services, libraries or products shall be used or integrated in their SAP BTP extension.

The next important selection criteria are the extension requirements concerning proximity to and coupling with SAP S/4HANA Cloud data, transactions, and apps. Here we distinguish between loosely- and tightly-coupled extensions:

Loosely-coupled extensions and custom apps:

- Stand-alone applications or new process steps with occasional usage of SAP S/4HANA Cloud data.
- Data is replicated to SAP BTP or read via remote API calls from the core product.
- Custom data is not changed together with core data (no transactional consistency required).
- Data hubs or integration hubs that integrate, collect, or distribute data from many systems across company units are typical loosely-coupled scenarios.

For loosely-coupled extensions and custom apps, side-by-side extensibility on SAP BTP is the default choice.

Tightly coupled extensions need coupling and proximity to SAP S/4HANA Cloud data, transactions, or apps. Typical patterns for tightly coupled extensions are:

- Frequent read-access or changes to SAP data (many roundtrips).
- Reading of customer data and SAP data in complex SQL queries (e.g., joins) and with high data volume.
- Required transactional consistency when customer data and SAP data is changed jointly.
- Extends the UI of an SAP app, extends the SAP data model, implements the extension point of an SAP app.
- Uses local SAP APIs to build an own tailored remote service for a side-by-side SAP BTP solution.

Tightly-coupled scenarios shall be realized with key user or on-stack developer extensibility.

The final decision step for tightly coupled extension is to choose between on-stack developer and key user extensibility:

- Key user extensibility shall be chosen for extensions by single business experts as explained in chapter 2.2.
- Developer extensibility shall be chosen for extensions requiring professional ABAP development as explained in chapter 2.3.

Customers may additionally define a company-specific policy that considers available business expert and developer resources as well as the extension scenario to choose between these options.

3.1 Additional aspects to consider

Finally, the following aspects should be considered for extension projects:

- **Consider the knowledge and experience of your development teams.** With the ABAP-based on-stack and side-by-side extensibility options, you can leverage the ABAP skills of your development teams. Loosely-coupled extensions on SAP BTP can similarly be realized in other development environments, but that would require a different set of skills.
- **Balance between homogenous versus hybrid extensions:** customers and partners must find the right balance between homogenous (only on-stack or only side-by-side) versus hybrid (mixture of on-stack and side-by-side) extensions. Homogenous extensions should be preferred for simpler lifecycles, but in some cases, hybrid extensions are the best solution.
- **Layering of key user and developer extensibility:** on-stack extensions are often a mixture of key user extensions and developer extensibility. For these scenarios customers must consider that these extension types are layered (key user extensions on top of developer extensions). Objects built with

developer extensibility can be released for key user extensibility, but not vice versa. Consequently, objects that are to be re-used in both layers must be built with developer extensibility [7].

3.2 Examples

3.2.1 Key user extensibility

A customer adds custom fields for new package properties (color of the ordered package) and customer member status (level of membership) to the standard SAP S/4HANA Cloud sales process.

These new custom fields are then part of all standard SAP S/4HANA Cloud sales screens and all process steps. This includes prefilling, validating, and using the custom fields along the entire sales process (propagate along sales quotation, sales order, outbound delivery, and billing document).

Additionally, the customer creates custom analytical reports and a custom form template for the order confirmation printout. The new custom fields are part of the analytical reports and the printout as well.

This comprehensive example can be completely realized with key user extensibility tools, see:

Key User Extensibility in SAP S/4HANA Cloud Sales | SAP Blogs

3.2.2 On-stack developer extensibility

A customer simplifies the process for recurring employee purchases below a certain amount. He uses the ABAP RESTful application programming model (→ chapter 4.1) and SAP Fiori elements to implement an employee self-service app on SAP S/4HANA Cloud ABAP environment.

The app uses local public procurement APIs and extension points to validate, create and release the purchase orders automatically.

3.2.3 Side-by-side extensibility

A customer uses multiple SAP S/4HANA systems in several of their subsidiaries to manage his purchasing processes. To improve the purchasing process he develops a central purchasing approver determination application on SAP BTP.

Based on a set of business rules, the application identifies the allowed approvers for specific purchasing documents and provides the information as a central remote OData API to the distributed purchasing processes.

[7] Exception: custom fields created with key user tools are usable in both layers. For details on the interaction between developer and key user extensibility, see the documentation, section Key User Extensibility and Developer Extensibility.

4 THE ABAP CLOUD DEVELOPMENT MODEL

With the launch of SAP BTP ABAP Environment, SAP introduced a new **ABAP Cloud development model** which leads to modern, cloud-ready, and upgrade-stable ABAP applications and extensions. ABAP Cloud development means first and foremost:

- Only public SAP APIs and SAP extension points can be used.
- No modifications to SAP objects are allowed.
- Use ABAP Development Tools in Eclipse (ADT) as your ABAP development environment.
- Build RAP (ABAP RESTful application programming model) based SAP Fiori apps or services.
- Technologies like Dynpro or Web Dynpro are not supported.

The ABAP Cloud development rules are enforced via:

- ABAP compiler and ABAP runtime checks: the ABAP Cloud development model uses ABAP Cloud as the language version[8]. This cloud-optimized ABAP language defines the set of supported ABAP statements and launches syntax- or runtime errors if e.g., a non-public SAP API is used (→ Chapter 4.2).
- ABAP authorization checks: ABAP Cloud redefined the authorizations in the ABAP Cloud developer role (e.g., no authorization to change SAP objects)[9].

The ABAP Cloud development model is mandatory for ABAP apps developed on SAP BTP and for extensions built on SAP S/4HANA Cloud Public Edition and **can** be enabled in SAP S/4HANA Cloud Private Edition and SAP S/4HANA on-premise when desired.

This chapter will provide details about the main building blocks of this model:

- The ABAP RESTful application programming model (RAP).
- Analytics in ABAP Cloud.
- ABAP Cloud – the cloud-optimized ABAP language version.
- Reuse services.
- Identity and access management.
- Connectivity.
- Business APIs and extension points.
- Business configuration.

The primary purpose of RAP is to build and expose back-end services based on semantic data models. The services are then consumed by SAP Fiori apps or exposed as Web APIs. The RAP programming model is therefore at the core of every extension project (→ Chapter 4.1).

For the implementation layer ABAP Cloud is used (→ Chapter 4.2). In addition, SAP offers a large library of reuse services (→ Chapter 4.4).

SAP Fiori applications follow a role-based approach which can be properly modelled with the identity and access management capabilities of the underlying ABAP Platform (→ Chapter 4.4.3).

Since extensions typically interface, interact and integrate with SAP's business solutions a large set of connectivity options and protocols is needed (→ Chapter 4.4.4). Stable business APIs and extension points provide the semantic link to the standard application (→ Chapter 4.5).

Last, but not least, business configuration frameworks and tools provide the flexibility to deliver extensions for a wider range of business scenarios and use cases (→ Chapter 0).

In the following we will examine the various building blocks in more detail.

[8] ABAP Cloud is also referred to as ABAP for Cloud Development, ABAP classic is also referred to as Standard ABAP.

[9] The ABAP Cloud development model for SAP HANA Cloud private edition is available with release ≥ 2022 and for SAP HANA on-premise with ABAP Platform 2022.

4.1 The ABAP RESTful application programming model

In recent years the requirements for business applications and related technologies have significantly changed. End users expect enhanced user experience qualities. For example, continuous work - start working at home, continue during commuting and finalize the task at the company or the ability to accomplish tasks on different device types. In addition, end users expect personalized, web-like business apps. So, user experience is taking center stage in a constantly evolving business landscape.

On the other hand, modern business apps must be able to run in hybrid system landscapes supporting cloud, on-premise, and virtualization. In addition, these modern business apps must continue to meet the requirements of large enterprises including all the steps of the product lifecycle.

To satisfy all these business needs and to make the development of modern business applications efficient, a standardized ABAP programming model that guides ABAP developers and optimally supports cloud operation, SAP Fiori, and SAP HANA, has become a pressing need.

To meet this need, SAP offers the ABAP RESTful application programming model (RAP). RAP is the new standard ABAP programming model used at SAP to build SAP S/4HANA Fiori apps and services. RAP is the recommended programming model for customers and partners for all ABAP based SAP S/4HANA extensions[10].

4.1.1 The big picture

RAP is a set of concepts, tools, languages, powerful frameworks, and best practices provided on the ABAP Platform for efficient and rapid development of innovative and cloud-ready enterprise applications, as well as the extension of SAP standard applications in the cloud and on-premise. RAP is deeply integrated with the ABAP language; moreover, the ABAP development tools have been optimized to support the RAP development flow. The main RAP building blocks are:

- **Core Data Services** (CDS) used for SAP HANA optimized queries, to define semantically rich data models for all application domains, and to define the transactional behavior of the modeled entities.
- The modernized and extended **ABAP language** used to implement business logic.
- The **OData protocol** used for stateless communication.
- The concept of **business object (BO)** used for building transactional applications.
- The concept of **business service** used to define services.

RAP offers a standardized development flow in the modern, Eclipse-based ABAP Development Tools (ADT) and a rich feature set for building applications from different domains, either from scratch or by reusing existing custom code. Built-in capabilities such as ABAP unit tests (→ Unit testing with ABAP unit), the ABAP cross trace tool (→ Working with ABAP Cross trace tools), and knowledge transfer documents (→ Documenting ABAP development objects | SAP Blogs) are offered along the RAP development stack to cover core software quality concepts such as testability, supportability and documentability.

Different types of services can be built with RAP:

- OData-based services for building role-based and responsive SAP Fiori apps.
- OData-based services for exposure as Web APIs.
- Business events.

Figure 4.1 gives an overview of the RAP big picture. A detailed description of the architecture can be found in the RAP developer guide: RAP Developer Guide on SAP Help Portal

[10] The ABAP RESTful application programming model is available in SAP S/4HANA Cloud Public Edition, SAP Business Technology Platform and for SAP HANA Cloud private edition starting with release 2022 and for SAP HANA on-premise with ABAP Platform 2022.

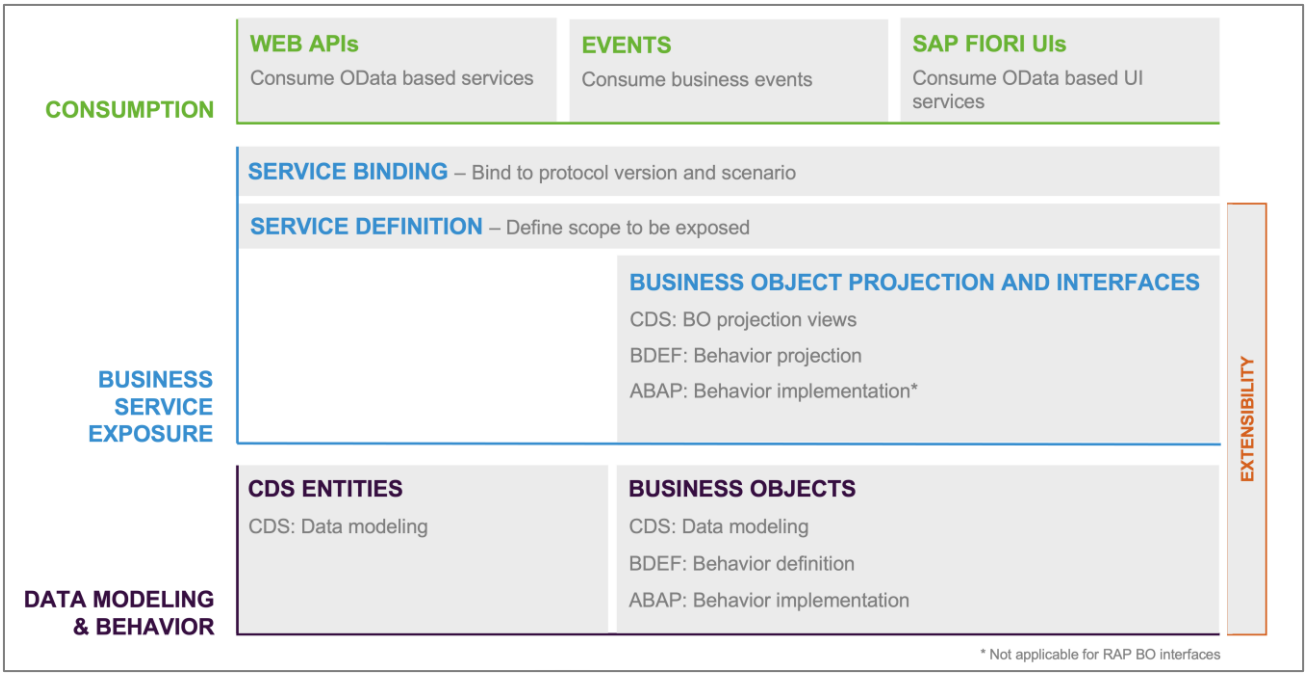

Figure 4.1 - RAP big picture

Cloud-ready and upgrade-stable apps and services can be built when using RAP together with ABAP Cloud, released SAP APIs and objects.

SAP uses RAP to provide local APIs on SAP S/4HANA Cloud for central business objects like Sales Order. These APIs can be used for on-stack developer extensibility to e.g., create or change a Sales Order.

Examples of these local RAP-based SAP APIs can be found in the developer extensibility section of SAP API business hub[11].

Find more information on RAP here: Modern ABAP Development with the ABAP RESTful Application Programming Model

A more detailed description on supported RAP implementation scenarios and when to use which option can be found in this blog: Modernization with the ABAP RESTful Application Programming Model (RAP) | SAP Blogs

4.1.2 Extensibility of RAP business objects (RAP BO)

Business objects developed with RAP can have built-in extensibility options[13]. They can be extended as part of the developer extensibility model. This is particularly useful when SAP S/4HANA business objects or RAP business objects from partner applications have to be extended. The extensibility options for RAP business objects are of course designed for lifecycle stability and a separation of concerns between the original RAP BO and its extensions.

By default, extensibility of a RAP BO is disabled. Therefore, RAP BO extensibility must be explicitly enabled on the original BO[12], to allow extensions at dedicated extension points. The extension options are developed to be controlled on a fine-granular level, to give full control over which parts of a BO can be extended with which kind of extension. Trying to define extensions for non-enabled extension points of the original BO leads to a syntax error and it cannot be activated[13].

[11] https://api.sap.com/products/SAPS4HANACloud/developerextensibility/bointerface
[12] The original RAP BO is the base BO on which the RAP BO extension bears
[13] Extensibility enablement of RAP BOs created by SAP can only be enabled or changed by SAP and not by customers or partners

It is crucial to understand that:

- Each RAP BO extension can only define behavior specifically for **its own** extension elements and operations and is independent of all other RAP BO extensions.
- RAP BO extensions cannot redefine or change any behavior for any elements of the original RAP BO.

From the consumer's perspective, the original RAP BO and all associated RAP BO extensions appear as one large RAP BO.

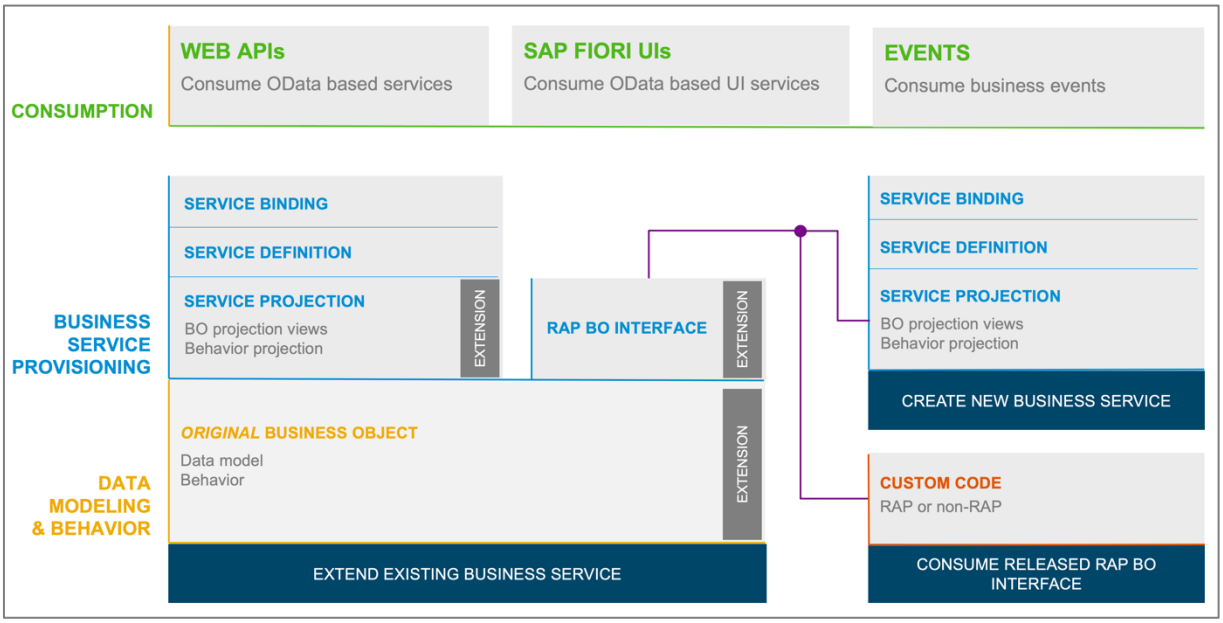

Figure 4.2 - RAP Extensibility options overview

The major extensibility scenarios for RAP BOs, as shown in Figure 4.2 are:

- Adding a new field including related business logic.
- Enriching the business logic of the BO.
- Adding a new action resulting in a new button in an SAP Fiori application.
- Adding a completely new entity to an SAP BO.

The extensibility options for RAP BOs are described in Table 4.1 below and links to further information are provided.

EXTENSIBILITY OPTION	EXTENSIBILITY TASK
DATA MODEL EXTENSION Build full-stack data model extensions by adding new fields and associations including corresponding behavior characteristics and authorization control.	**Extensibility enablement**: Add annotations and extension include structures to the original RAP BO to enable data model extensions. For more information about enabling full-stack data extensibility, see Extensibility-Enablement for CDS Data Model Extensions. For an implementation example, see Enabling Field Extensions. **Extension development**: Extends the original RAP BO with new fields or associations including field characteristics depending on the options defined by the extensibility-enabler. For more information about how to develop data model extensions, see CDS Data Model Extensions. For an implementation example, see Develop Field Extensions.

EXTENSIBILITY OPTION	EXTENSIBILITY TASK
BEHAVIOR EXTENSION Build additional behavior like new validations, determinations or actions including dynamic feature control and other field-related behavior.	**Extensibility enablement**: Enables data model extensibility and behavior extensibility on the original RAP BO. For more information about how to enable your BO for behavior extensions, see Extensibility Enablement for Behavior Extensions. For an implementation example, see Enabling Non-Standard Behavior and Field-Related Behavior and Enabling Standard Behavior Extensions.
	Extension development: Extends the original RAP BO with new validations, determinations or actions depending on the options defined by the extensibility-enabler. For more information about how to develop different behavior extensions, see Behavior Extensions. For an implementation example, see Develop Behavior Extensions.
NODE EXTENSION Build additional BO nodes with own behavior and data model with node extensibility.	**Extensibility development**: Enables node extensibility on the original RAP BO. For more information about node extensibility enabling, see Extensibility-Enablement for the Node Extensibility. For an implementation example, see Enabling Node Extensions.
	Extension provisioning: Extend the original BO with new nodes that have their own data model and behavior. For more information about how to develop node extension, see Node Extensions.

Table 4.1 - Extensibility options for RAP BOs

4.2 Analytics in ABAP Cloud

Analytics enables multi-dimensional reporting and data analysis in ABAP in two main scenarios:

Embedded analytics

Embedded analytics allows you to build sophisticated and complex analytical data models to evaluate and analyze business data in your ABAP system. In embedded analytics, the ABAP analytical engine is part of the software stack and operates on the same data persistence layer as the transactional applications. The analytical queries operate directly on the business data without data replication to an external data warehouse system. Instead, the real-time business data is queried to always evaluate the most recent changes and trends in your business data.

Side-by-side analytics scenarios

In a side-by-side analytics scenario, the business data is replicated from an analytical source system to other analytical clients (like SAP Analytics Cloud) or to a data warehouse (like SAP Data Warehouse Cloud). The business data is replicated from an ABAP system based on a predefined integration scenario to allow seamless data exchange. The analytical data model itself is implemented outside of the ABAP system and only the analytical data the source system is used and is part of the analytical consumption scenario.

Another option to realize side-by-side analytics scenarios is data federation. Data federation is used in scenarios where live access to the source system is required, e.g., to access the most recent data without a time lag, or to rely on data access control mechanisms inside the source system. Data calls are not executed locally in the target system but delegated from there to one or more source systems (where the data is located) and executed on source side.

The following chapters focus on the embedded analytics use case and highlight the advantages of this approach.

4.2.1 The big picture of embedded analytics in ABAP Cloud

Embedded analytics unites the advantages of CDS data modeling and the analytical capabilities of the analytical engine to create real-time evaluations based on business data within the same ABAP system: The CDS framework is at the core of the ABAP Cloud development model, enabling the development of cloud-ready and lifecycle-stable analytical data models for all SAP S/4HANA deployment options and for SAP BTP ABAP environment.

The main building blocks of an analytical application are[14]:

- **Dimensions**: master data-like data used as attributes in the data analysis.
- **Cubes**: an analytical interface view that is used in analytical queries. Cube views are at the center of the multi-dimensional data model.
- **Hierarchies**: define a data hierarchy and can be used for drill-down or roll-up operations to change the data granularity of the result set.
- **Analytical queries** (CDS Analytical Projection Views)[15]: the initial layout of the initial data set and granularity of the query, based on a cube.
- **Service Binding**: the Information Access Protocol (InA) is the protocol used to expose an analytical data model for consumption by analytical clients like SAP Analytics Cloud.

The following graphic provides a more detailed overview of the main building blocks used to build a data model for embedded analytics:

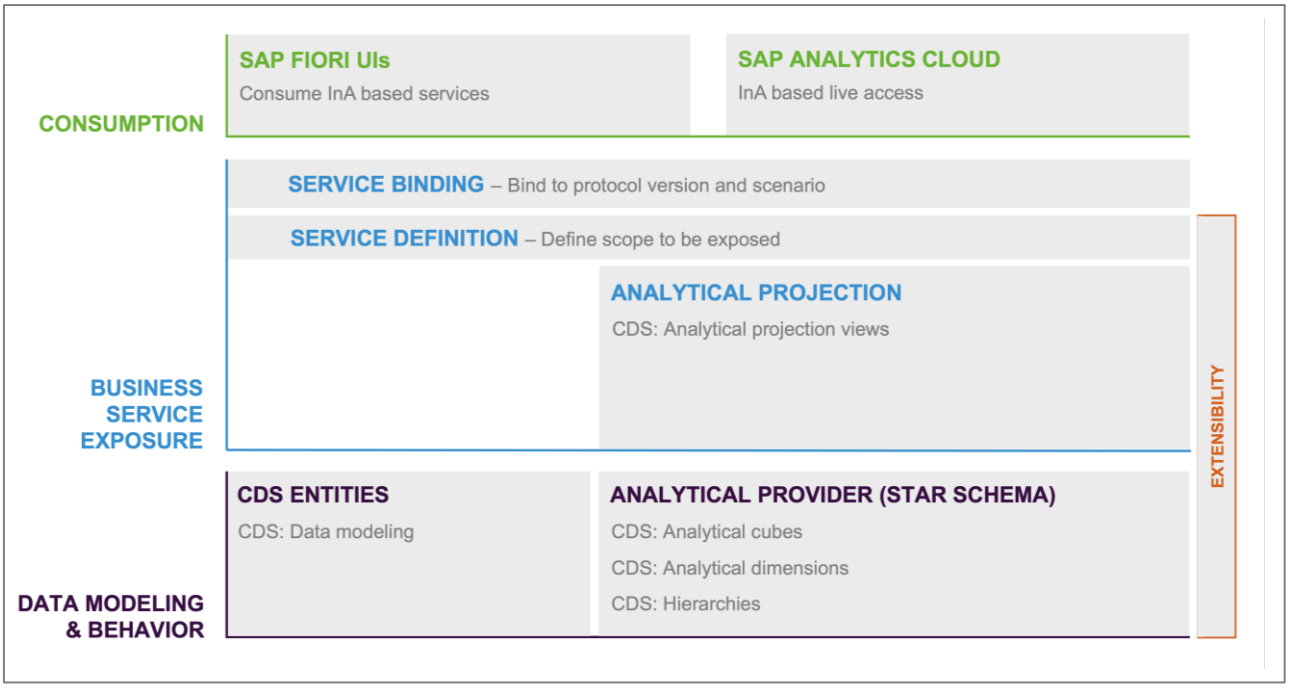

Figure 4.3 - Detailed overview of the main building blocks used to build a data model for embedded analytics.

An analytical data model is multidimensional and is typically modeled in a star or snowflake schema to enable easy data analysis at different levels of granularity[16]. CDS allows easy data modeling and data preparation for display in dashboards such as those in SAP Analytics Cloud.

[14] For more information, see https://help.sap.com/docs/BTP/65de2977205c403bbc107264b8eccf4b/483cc0637280445b98e98775dd0383b1.html

[15] For more information, see https://blogs.sap.com/2022/02/18/cds-analytical-projection-views-the-new-analytical-query-model/comment-page-1/

[16] For more information, see https://blogs.sap.com/2022/11/30/embedded-analytics-with-abap-cloud-a-brief-overview-part-1/

For developer extensibility the analytical data models and services are developed in the modern, Eclipse-based ABAP Development Tools (ADT). ADT offers an efficient development flow, and a rich feature set for creating analytical services.

Key user extensibility for analytical scenarios offers a rich SAP Fiori-based toolset for key users for creating and managing custom analytical queries, analytical reports, KPIs and analytical stories.

See also: Key User Extensibility Overview | sap.com and SAP S/4HANA embedded analytics – User Roles | Blogs).

4.2.2 Extensibility for analytical data models

Extensibility is a built-in quality for all ABAP data models based on CDS. Based on an opt-in approach, the entire data model stack can be enabled for different extensibility use cases depending on the specific data model requirements. Each layer in the stack offers possibilities for dedicated extension points, enabling a well-defined separation of concerns between the original data model and the extensions.

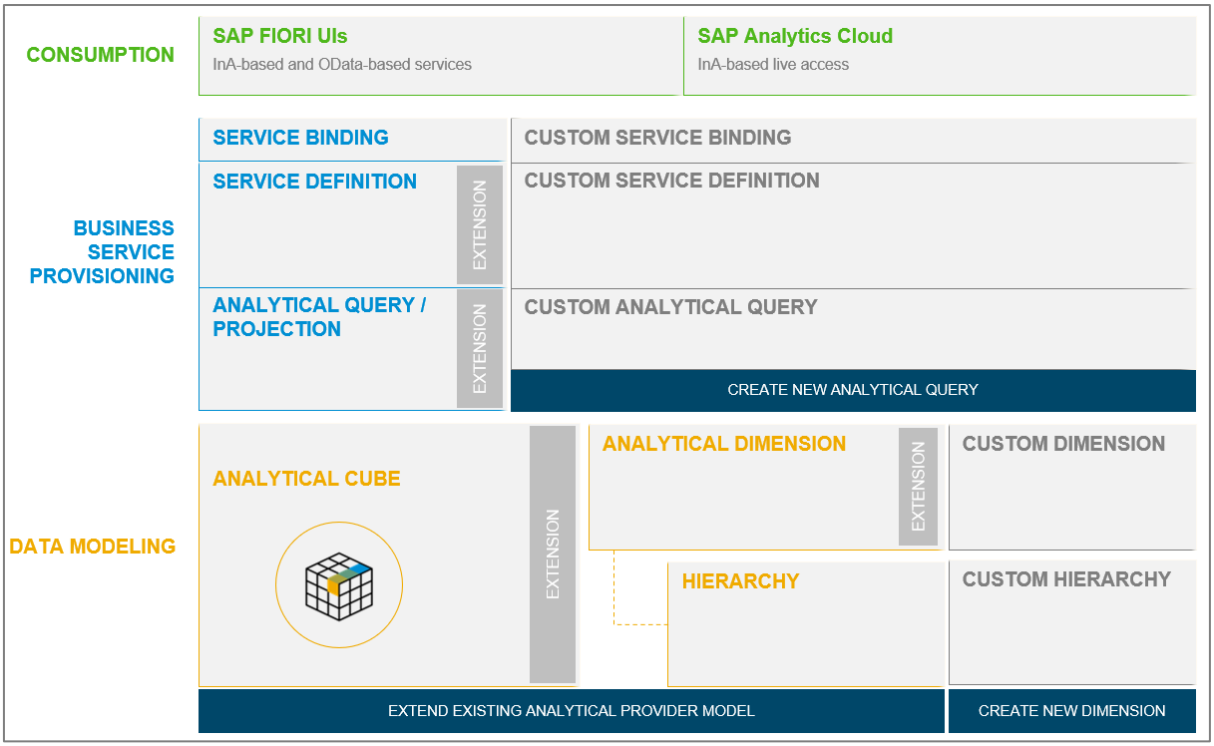

Figure 4.4 - Overview on extensibility options for analytical data models

In an analytical data model, each part of the data model can be extended individually using predefined extension points in the original analytical data model. You can use extension points to make your own analytical data models extensible or to extend pre-delivered SAP content:

EXTENSIBILITY USE CASE	EXTENSIBILITY TASK
DIMENSION EXTENSION Add new hierarchies to a dimension or add new fields or associations to the dimension to diversify the data model.	**Extensibility Enablement:** Add annotations and extension include structures to the original dimension to enable data model extensions like fields or associations.
	Extension Development:

EXTENSIBILITY USE CASE	EXTENSIBILITY TASK
	• Add new hierarchy: To extend a dimension with a new hierarchy, create a new hierarchy and add it as an association to the corresponding dimension. • Add new field extension: Extend the original dimension with additional fields.
CUBE EXTENSION Add new dimensions to a cube to extend the scope of the data analysis or add new measures to the cube to calculate additional values.	**Extensibility Enablement:** Add annotations and extension include structures to the original data model to enable data model extensions like fields or additional dimensions.
	Extension Development: • Add new dimension: To extend a cube with a new extension, create a new dimension or use a pre-delivered SAP dimension and add it as an association to the corresponding dimension. • Add new field(measure): Extend the original cube with additional measures for new calculations.
QUERY EXTENSION Add new numeric fields to the query to extend the scope of data analysis.	**Extensibility Enablement:** Add annotations and extension include structures to the original data model to enable data model extensions like additional numeric fields for calculations in queries.
	Extension Development: Add new fields to extend the scope of the data analysis.
SERVICE DEFINITION EXTENSION Create a new UI based on a released query or extend the service definition to add new queries to a service definition.	**Extensibility Enablement:** Enable the service definition for extensibility to add additional queries to a service.
	Extension Development: • Create a custom UI based on a released UI to adapt the UI to your business requirements. • Add a new query to a service definition to extend the original service definition with additional queries.

Table 4.2 - Extensibility options for analytical data models

4.3 Cloud-optimized ABAP language

ABAP is a programming language optimized for business applications, both at the small and at the large scale. It is designed to minimize the total costs of development for business applications. As such, the ABAP language has evolved over a long time from procedural and dynamic programming to ABAP objects and is the foundation of the ABAP RESTful application programming model. Thereby, both its capabilities and the number of ABAP keywords increased steadily, allowing for enormous flexibility but leading to complexity at the same time.

Now, for the ABAP Cloud development model, the scope of ABAP language commands has been refined to simplify and standardize ABAP development and enable cloud-ready programming. In the new language version named **ABAP for Cloud development**, only modern ABAP statements and concepts, with a focus on cloud-enabled, object-oriented design and modern programming models are supported. Using non-supported statements results in syntax errors. To ensure the integrity of the cloud extensibility model, direct access to the file system, profile parameters, or ABAP server operations is not permitted.

ABAP Cloud is mandatory in SAP S/4HANA Cloud Public Edition and in SAP BTP ABAP environment and **can** be enabled in SAP S/4HANA Cloud Private Edition and SAP S/4HANA on-premise when desired.

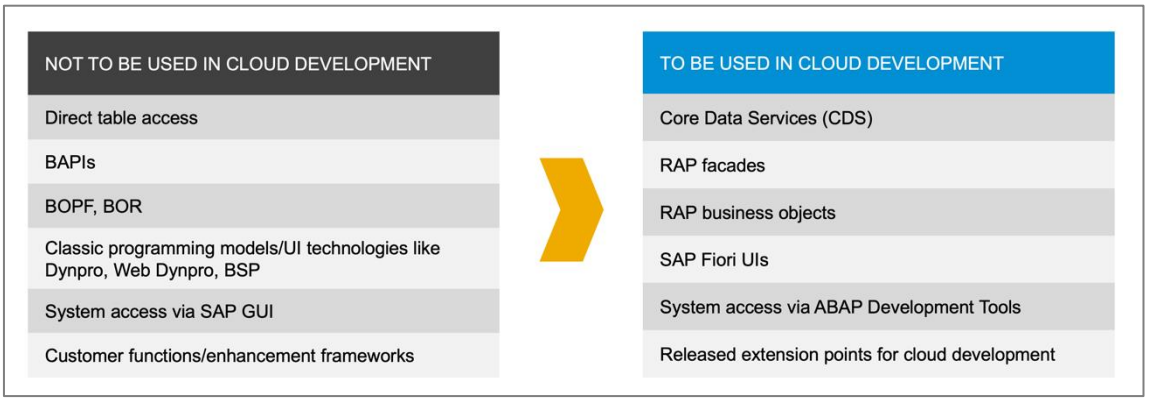

Figure 4.5 - Concepts in Cloud development

More details are provided in chapter 5. An overview of programming concepts is shown below and serves as a (non-comprehensive) list of criteria to decide which concepts can be used or not.

SAP data can only be accessed by using ABAP SQL or ABAP managed database procedures on released SAP CDS views, or by released SAP APIs. For more information on ABAP Cloud scope, see the ABAP Keyword Documentation.

Many technical ABAP Platform features and reuse services are available via public APIs and libraries (→ Chapter 4.4).

4.4 ABAP Platform reuse services

Besides RAP as programming model for cloud-ready development and ABAP Cloud as language version, the ABAP Cloud development model comprises further aspects like usage of modern reuse services, released APIs, and modern business configuration and connectivity frameworks.

4.4.1 Released local APIs

The released local APIs constitute the local public interface of the SAP code. This stable interface is the key to the lifecycle independence of SAP code and customer and partner extensions. Hence, only released interfaces, released APIs, frameworks and services can be used in extensions.

The scope of released local ABAP Platform APIs is continuously enhanced and already covers major building blocks:

- Data access APIs (CDS views).
- Technical ABAP Services (Parallel Processing, Compression/Decompression, Runtime Info, Dynamic Programming, XSLT, …).
- Technical Reuse Services (→ Chapter 4.4.2).

- APIs required to develop integration scenarios (OData, HTTP, RFC, SOAP, Business Events → Chapter 4.4.4).

Important SAP APIs and objects that are not suited for cloud development have been replaced by new and modern APIs that fit into the cloud extensibility model. For these objects, having a direct successor released for cloud development, the syntax error messages directly point towards the successor to be used. An example is given below in Figure 4.4. A direct select on the database table `T005` is forbidden, the successor CDS view `I_COUNTRY`, is directly indicated in the syntax error.

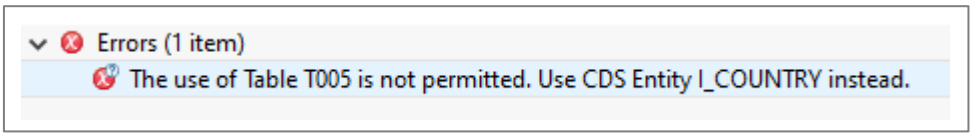

Figure 4.4 - Syntax errors for direct table access

All released APIs are shown in the *released objects tree* in ABAP Development Tools. If released APIs are missing, please make use of the Customer Influence Channels for the ABAP Platform (https://influence.sap.com/sap/ino/#campaign/2911).

This way you can shape and sharpen the public APIs available for your extensions in the cloud extensibility model.

4.4.2 Technical reuse services

Technical Reuse Services are essential to lower the TCD and TCO of business applications. Hence, dedicated released APIs and configuration possibilities for well-known ABAP Platform reuse services like Change Documents, Number Ranges, Email Processing, Factory Calendar, and others are provided. In general, to adhere to the principles for cloud development explained above:

- Reuse services are consumed by calling the corresponding released local APIs.
- Necessary development objects (like a number range object) are developed in ABAP Development Tools.
- Required configuration and monitoring is done via SAP Fiori Apps.

Just as with the scope of the ABAP language, cloud-optimized services used by SAP S/4HANA in the area of reuse services are enabled for consumption in the context of developer extensibility These services replace older services and frameworks. Popular examples of technical services that shall be used in developer extensibility are provided in the table below. Note that this table is only a snapshot providing hints on which services to focus, but is neither a complete list of all services, nor does it state that all services listed there are already available in the full functional scope. Instead, the services listed to be used will be enabled and integrated into the programming model step-by-step.

TECHNICAL REUSE SERVICES USED IN CLOUD DEVELOPMENT		
SERVICE	**STATUS**	**RETIRED PREDECESSOR**
Email Service	●	SAP Office
Email Templates	○	
Factory Calendar	●	
Notes for Application Objects	○	SAPscript Longtexts, STXL
Knowledge Transfer Documents	●	SAPscript, SE61, DOKTL
Forms Processing integrating SAP BTP Forms by Adobe	●	SAPscript, Smartforms
Adobe Forms	○	SAPscript, Smartforms

TECHNICAL REUSE SERVICES USED IN CLOUD DEVELOPMENT		
SERVICE	**STATUS**	**RETIRED PREDECESSOR**
Number Ranges	●	
Change Documents	●	
Archive Development Kit	○	
Information Lifecycle Management	○	
Printing Queue	●	Spool
Application Jobs	●	Classical Batch Job SM36
Application Logs	●	
XCO ABAP Repository	●	Legacy Workbench APIs, e.g., for DDIC
XCO Standard Library	●	
Translation	●	
Units of Measurement	●	
SAP BTP Document Management Service	●	KPro, Content Server, SAP Office
SAP BTP Workflow	●	
SAP BTP Rules	●	
Time zones	●	
Exchange Rates and Currency Conversion	●	
Attachment Service	○	Generic Object Services, Business Document Service, Archive Link, SAP Office
SAP S/4HANA Output Control	○	Post Processing Framework
Legend: ● available ○ planned		

Table 4.3 - Technical reuse services for cloud development

4.4.3 Identity and Access Management

In cloud products, a strict content separation between users and roles is needed to ensure stable cloud operations. Thus, customers do not have direct access to the classical transaction for role maintenance (PFCG). Instead, SAP delivers a new cloud Identity and Access Management (IAM), which consists of:

- IAM SAP Fiori apps for user and role maintenance on top of IAM catalogs and role templates (maintain business roles, …) and APIs for user upload e.g., from SAP Identity Provisioning Service or other SCIM providers.
- The IAM app, IAM business catalog, and IAM business role templates, for creating IAM content for their own applications and for SAP applications (with released authorization objects).
- IAM catalogs and role templates for cloud-enabled applications delivered by SAP application development.

- Key user extensibility: with a custom catalog extension, customers can extend an existing SAP IAM catalog (e.g., add authorization allowing the user to start a custom OData service) in an SAP Fiori app.

Business users can authenticate via the SAP cloud identity services. The Identity Authentication Service can be set up in proxy mode to use a corporate identity provider in hybrid landscapes.

More information can be found in this SAP community blog: Integrating Identity Authentication service in SAP Cloud Platform – Proxy & Conditional Authentication scenarios.

4.4.4 Connectivity

Connectivity capabilities of the ABAP Platform are key to enable integration between SAP cloud products, SAP BTP services, customer extensions and external services. In hybrid landscapes, the SAP Cloud Connector can be used to integrate with protected internal customer landscapes by providing detailed control over a secure tunnel connection.

The integration with external APIs is simplified through the generation of typed proxies in the system from uploaded metadata via the service consumption model (OData, SOAP, RFC) and event consumption model.

Several classic integration technologies can no longer be used (e.g. IDoc), only released frameworks are available for cloud development. A list of the current scope is given below.

INTEGRATION FRAMEWORKS RELEASED FOR CLOUD DEVELOPMENT
OData services
Business Events
HTTP services
RFC via cloud enabled WebSocket RFC
SOAP consumption (SOAP service provider planned)
SQL service for external ODBC clients
SAP information access (InA) for analytical clients

Table 4.4 - Integration frameworks for cloud development

To ensure content separation between credentials, customers do not have direct access to classical transactions for destinations (SM59) and logical ports (SOAMANAGER) in SAP S/4HANA Cloud.

Instead, SAP delivers a new Cloud Communication Management, which contains the following:

- The ABAP Platform provides SAP Fiori apps for both inbound and outbound communication (Maintain Communication Arrangements, Maintain Communication User …).
- Custom development: using the development objects Communication Scenario, Inbound & Outbound Services, customers can create integration content for their own applications and for SAP applications (with released services).
- SAP application development delivers Communication Scenarios for supported integrations.
- Key user extensibility: key users can create their own Communication Scenarios for OData services generated from released CDS views and own applications in an SAP Fiori app.

4.5 SAP S/4HANA business APIs, extension points and events

In addition to the ABAP Platform content, in SAP S/4HANA deployments released business APIs and extension points are available in the ABAP Cloud development model (→ Table 4.5).

REMOTE CONSUMPTION	LOCAL CONSUMPTION	LOCAL EXTENSION POINTS
• OData services • SOAP services • Events	• CDS Views • RAP BO interfaces • Classes • Events (planned)	• Business Add-Ins (BAdIs) • RAP BO extensions

Table 4.5 - Available business APIs and extension points in SAP S/4HANA

Information on the released SAP S/4HANA APIs can be found in the product documentation and on the SAP API Business Hub. If an API is missing, please request it in the Customer Influence Channel for SAP S/4HANA: (https://influence.sap.com/sap/ino/#/campaign/2759).

4.6 Business Configuration (BC)

The ABAP Platform provides two programming models for business configuration (BC) apps:

- SAP GUI-based BC apps (SM30 – Table View Maintenance, SE54 - Create Table Maintenance View).
- SAP Fiori/RAP-based BC apps.

Some aspects of business configuration apps are different from other apps, and the programming models reflect these differences. BC maintenance apps are used less often, and they are usually very simple compared to the apps for master or transactional data. On the other hand, applications often have a greater number of BC maintenance apps compared to the apps for master and transactional data. This calls for highly standardized apps. Therefore, the programming models support standardized:

- Maintenance UIs.
- Authorization management.
- Change log.
- Maintenance of language-dependent texts.
- Transport recording and BC content management.

SAP GUI/Web GUI is not available in SAP BTP ABAP Environment and in SAP S/4HANA Cloud Public Edition. Therefore customers and partners can only implement SAP Fiori/RAP-based BC apps. To support development, SAP provides a wizard to generate the required RAP objects and a generic SAP Fiori UI. With these tools, customers can create simple maintenance apps with low development effort. Details are described in the following blog post: Create a Business Configuration Maintenance App

Customers and partners using SAP S/4HANA on-premise or SAP S/4HANA Cloud Private Edition can decide per scenario whether they go for SAP GUI or SAP Fiori-based BC apps.

As of version SAP S/4HANA release 2022, customers can use the generator for Business Configuration objects. The Custom Business Configurations SAP Fiori app is not yet available, but the release is planned for an upcoming on-premise version. This means in 2022, customers must create SAP Fiori apps using Business Application Studio and a tile for the SAP Fiori Launchpad per BC app. This limitation will disappear once the Custom Business Configurations SAP Fiori app is released.

	SAP S/4HANA Cloud Private Edition and on-premise	SAP S/4HANA Cloud Public Edition	SAP BTP ABAP Environment
SAP GUI-based custom BC apps (SM30/SE54) (including IMG integration)	✓	×	×
SAP Fiori/RAP-based custom BC apps (including generator in ADT)	✓ (2022)	✓	✓
Custom Business Configurations SAP Fiori app (generic UI, no dedicated Fiori app required)	planned	✓	✓
Integration SAP Fiori/RAP-based custom BC apps into the SAP implementation guide (IMG)	planned	✓	IMG is not available

Table 4.6 - Availability matrix for Business Configuration apps

Customers and partners using SAP S/4HANA on-premise or SAP S/4HANA Cloud Private Edition should define a roadmap towards SAP Fiori/RAP for BC apps along the following considerations:

- SAP Fiori/RAP-based BC apps should be used:
  - For new greenfield projects and development teams that have no experience in SM54/SM30.
  - For BC apps with advanced UI requirements and rich custom logic.
- Maintenance views based on SM54/SM30 can be still used:
  - If the maintenance view is simple, has no advanced UI requirements and custom logic.
  - If customers have already a lot of maintenance views based on SE54/SM30, and the development team has experience in SE54/SM30.
- As of today, SAP Fiori/RAP-based BC apps do not support the following features. If these features are used, then SE54/SM30 must be used:
  - Solution Manager Integration for customizing synchronization.
  - Cross client comparison (SCU0).
  - Content delivery via BC sets for custom configuration tables.

Customers can transform their maintenance views to SAP Fiori/RAP using the generator for Business Configuration objects. Configuration tables must fulfil the prerequisites of the generator.[17]

Maintenance views based on SM54/SM30 are created in tier 3. To get access to the data in tier 1, customers must create a CDS view on top of the configuration table and release it for ABAP for Cloud Development.

[17] https://help.sap.com/docs/btp/sap-business-technology-platform/business-configuration-maintenance-object (prerequisite paragraph), and https://help.sap.com/docs/btp/sap-business-technology-platform/generating-business-configuration-maintenance-object-with-generate-abap-repository-objects-wizard (prerequisites paragraph)

5 EXTENDING A NEW SAP S/4HANA CLOUD PRIVATE EDITION OR SAP S/4HANA ON-PREMISE SYSTEM

Today, in SAP S/4HANA Cloud Private Edition and SAP S/4HANA on-premise deployments, classic ABAP custom code is generally used for extensions. This endangers smooth upgrades and makes further cloud transformation steps more difficult. Hence, as explained in chapter 1 and chapter 2, reusing the SAP S/4HANA Cloud extensibility model also in private cloud or on-premise deployments is beneficial and recommended.

In this chapter we provide guidance on how to achieve this goal. For simplicity, we will only refer to the SAP S/4HANA private cloud, keeping in mind that the statements also hold for on-premise.

In the private cloud we need compromises with respect to the pure cloud extensibility model since the broader functional scope of SAP S/4HANA Cloud Private Edition is not covered by public SAP APIs and typically existing classic ABAP custom code must be handled.

Therefore, the private cloud must allow extension use cases that are not supported in the public cloud (e.g., using non-released APIs, extending Dynpro-based transactions).

To manage the coexistence of these different extensibility models in private cloud deployments we propose to work with three extensibility tiers as shown in Figure 5.1:

- **Tier 1 – Cloud development:** default choice for all new extensions following the SAP S/4HANA Cloud extensibility model.
- **Tier 2 – Cloud API enablement:** mitigation of missing local public APIs or extension points. Here, custom wrappers for non-released SAP objects are built and released for cloud development so that they can be used in tier 1. Once SAP provides a public local API, the custom wrapper can be removed. In this sense, tier 2 serves as an extension to tier 1 which will follow the same ABAP Cloud development model besides the usage of the wrapped non-released SAP object.
- **Tier 3 – Legacy Development:** classic extensibility based on classic ABAP custom code that is not supported in the ABAP Cloud development model. The goal is to avoid developments in this tier and follow the ABAP Cloud development model as much as possible (governed via ATC) to minimize the risk of upgrade issues. Guidance on how to achieve this for transformed legacy code is given in the next chapter.

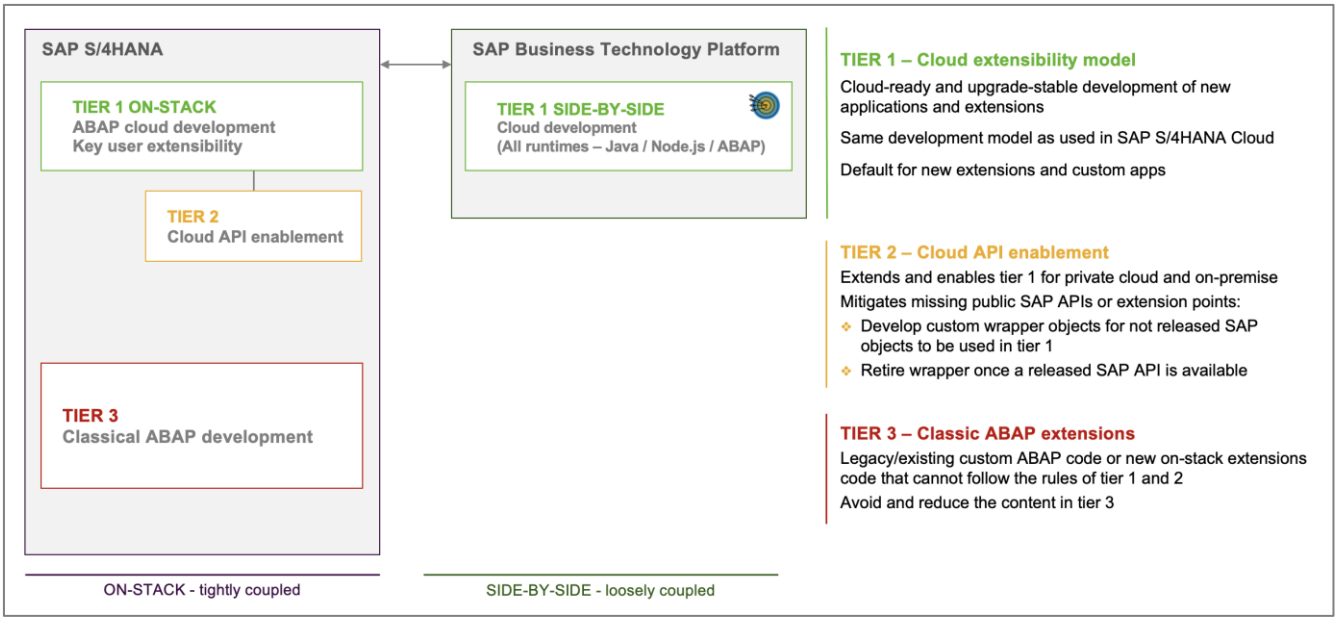

Figure 5.1 - Overview of the three-tier extensibility model

5.1 Setting up the three tier model

5.1.1 General Setup

The basic idea for the three tier model is

- In tier 1 the ABAP Cloud development model is enforced by syntax and runtime checks and cannot be circumvented.
- Tier 2 and tier 3 follow the classic ABAP development model but use the ABAP test cockpit (ATC) to enforce the ABAP Cloud rules as much as possible. Via ATC exemptions, controlled violations of ABAP Cloud rules are possible.

How this works in detail is explained in the following sections.

To realize the first point above, tier 1 shall be implemented as a separate software component with ABAP language version ABAP for Cloud Development. This is the relevant setting to force all objects created in this software component to follow the ABAP Cloud development model. Any violation (e.g. using a non-released SAP object) will result in syntax and runtime errors. For tier 2 and 3, no specialized software component shall be created, but packages in HOME or an existing software component with ABAP language version Standard ABAP shall be used[18]. In this way deviations from the ABAP Cloud development model that are required in these tiers can be tolerated. Below we will describe the ATC setup that allows one to control the development in these tiers.

In tier 2, wrappers around non-released SAP objects are developed and need to be released for Cloud development. This is required to allow tier 1 developments to use these wrappers. That means that the released wrappers in tier 2 form a clear custom interface to SAP development objects.

The general rules as to how development objects in the ABAP Cloud development model can access objects in the classic ABAP development model and vice versa is shown below:

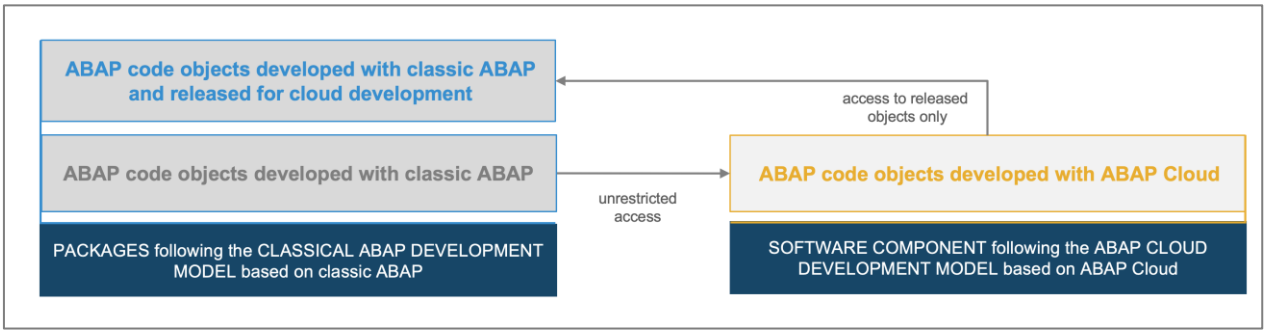

Figure 5.2 – Setup of the three tier model

ABAP development objects in software components following the ABAP Cloud development model can only use:

- Development objects inside the own software component.
- Released SAP development objects.
- Custom development objects released for cloud development of a software component following the classic ABAP development model.

Development objects in software components following the classic ABAP development model have unrestricted access to other objects:

- Development objects inside their own software component.
- Released and non-released SAP development objects.
- Custom development objects in software components based on classic ABAP and ABAP Cloud.

[18] When working with namespaces, it is also possible to use separated software components for tier 2 and 3.

With these access rules, we can now understand the interplay of the tiers as shown below:

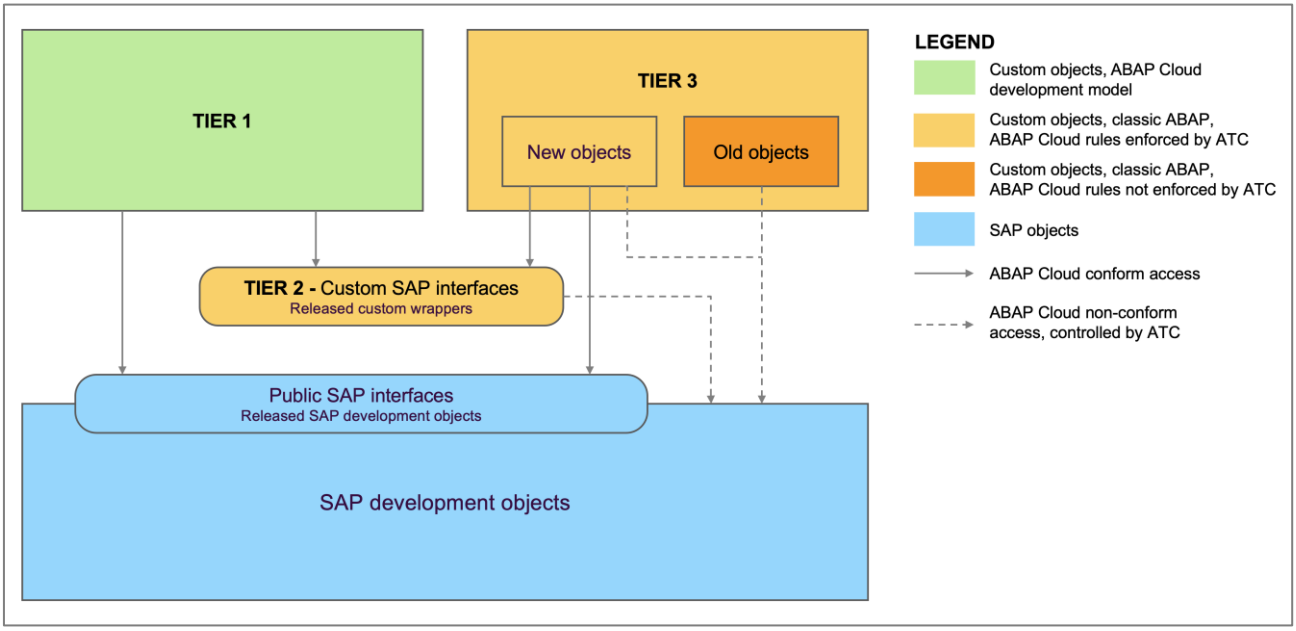

Figure 5.3 - Interplay between the three tiers

- Tier 1 comprises custom development objects strictly following ABAP Cloud rules. Besides objects in their own software component, they can only access released objects.
- These released development objects are provided by the public SAP interfaces and in addition the released wrappers in tier 2.
- Tier 2 is the custom SAP Interface developed in classic ABAP, that mitigates missing public SAP APIs. ABAP Cloud rules are enforced by ATC both for the development of wrapper objects as well as the access to non-released SAP development objects.
- Tier 3 contains the classic ABAP development, and we distinguish between new and old objects created here. For new objects in tier 3, ABAP Cloud rules shall be enforced by ATC. Existing objects in tier 3 may not follow the ABAP Cloud rules and can be exempted using a base line approach in ATC.

Another important mechanism to control the development of extensions is the use of proper authorizations. With the new authorization object `S_ABPLNGVS` administrators can control if a developer can create objects belonging to the ABAP Cloud development model only, or also objects based on classic ABAP. Restricting the developer to the modern programming model with ADT is also based on the developer's authorization and can be controlled by the authorization object `S_DEVELOP`[19]. In this way it is possible to create a dedicated developer role to enforce ABAP Cloud development in tier 1. The role template SAP_BC_ABAP_DEVELOPER_5 may be used as a starting point. For development in tier 2 and 3 an extended development role can be provided and assigned to selected developers only.

5.1.2 ATC Setup

In this section we explain how to set up the ABAP test cockpit (ATC) to enforce ABAP Cloud rules in the three tier model.

As explained above already, for tier 2 and tier 3 the ABAP Cloud development model shall be enforced by ATC checks. In this way it is also possible to push classic ABAP developments towards ABAP Cloud, but to allow the usage of non-released SAP APIs and ABAP statements from Standard ABAP scope where required. At the same time the ABAP test cockpit provides a monitoring and governance option for these deviations from the ABAP Cloud development model.

[19] In private cloud systems, there is no option to technically force developers to use ABAP Development Tools in Eclipsed only. But we strongly recommend using ADT.

This is why we recommend defining one global check variant in the ATC configuration that contains the required checks explained below. Furthermore, the ATC transport settings shall be configured so that priority 1 and 2 findings block the release of transport tasks and requests. This means that with the global check variant the ABAP Cloud rules can be ensured for all tiers during development and when releasing transport requests.

The global check variant shall contain checks included in the two standard ATC variants ABAP_CLOUD_DEVELOPMENT_DEFAULT and ABAP_CLOUD_READINESS. With the first variant basic ABAP Cloud development rules are covered. The second variant contains the checks of category *Cloud Readiness*:

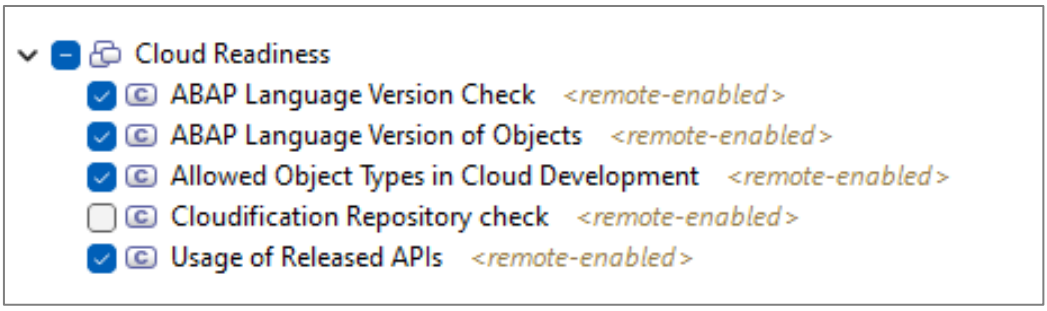

Figure 5.4 - ATC variants for cloud readiness check

The checks in this category are described in the table below.

ABAP Language Version Check	SLIN_VERS	Analyses the custom code against the ABAP Language Version *ABAP for Cloud Development*
ABAP Language Version of Objects	SYCM_CHECK_ABAP_LANGU_VERSION	Checks if the language version of the objects matches to the language version of the software component
Allowed Object Types in Cloud Development	SYCM_CLOUD_ALLOWED_OBJCT_TYPES	Checks if the object types are supported in ABAP Cloud
Usage of Released APIs	SYCM_CLOUD_RELEASED_OBJECTS	Analyses the objects for the usage of released APIs

Table 5.1 - Description of checks in category Cloud Readiness

To create the global check variant out of these checks we recommend creating a copy of the ATC variant ABAP_CLOUD_DEVELOPMENT_DEFAULT via ADT as a customer variant and manually adding the ATC checks of the category *Cloud Readiness* explained above.

ATC exemptions shall be used to tolerate and monitor deviations from ABAP Cloud Development in tier 2 and 3. As exemptions can be granted on several scope levels (package, object, finding) there is a lot of flexibility to decide how strict the usage of ABAP Cloud will be enforced in a custom development project. Especially for tier 3, the exemptions with object and package level scope might be helpful to allow classic ABAP development in a controlled way. For example, in the case of the extension of a Dynpro-based SAP transaction the usage of ABAP Dynpro statements is necessary, but the usage of released APIs can still be governed. In this case an exemption for the ABAP Language Version check is necessary for the object but not for the check for the usage of released APIs.

The ATC exemption browser can be used to monitor the exemptions and their scope and is therefore the recommended tool to control the adoption of ABAP Cloud development rules. In case it is required to deal

with legacy objects that cannot be adapted to obey ABAP Cloud rules, the use of an ATC baseline is recommended to suppress the findings in these objects.

More detailed information on how to use ATC to control usage of ABAP Cloud can be found in the blog post https://blogs.sap.com/2023/05/22/how-the-abap-test-cockpit-supports-you-to-adopt-abap-cloud/

5.1.3 Setup Summary

An overview of setup, tools, and rules in the three tiers is given in the table below.

	TOOLS / RULES FOR ON-STACK CUSTOM ABAP CODE	**STRUCTURING**
Tier 1 **Cloud Development** **This tier is dedicated for the cloud ready development of new applications and extensions**	The ABAP Cloud development model is used • Development objects are created in language version ABAP for Cloud Development • ABAP Cloud Rules are enforced by syntax and runtime checks. • ADT is the mandatory IDE. • Use developer role template SAP_BC_ABAP_DEVELOPER_5.	Software component with language version ABAP for Cloud Development.
Tier 2 **Cloud API Enablement** **This tier is dedicated to the development and release of wrappers around non released SAP APIs**	• Wrapper objects are created in language version Standard ABAP • These wrappers shall be released for Cloud Development to be usable in tier 1. • ABAP Cloud rules are enforced via the ATC. • ADT is the recommended IDE. • Use developer role template SAP_BC_ABAP_DEVELOPER_5 and add the authorization for Standard ABAP via S_ABPLNGVS.	New packages in HOME or own software component for classic ABAP if a namespace is used.
Tier 3 **Legacy Development** **This tier is dedicated to the usage of legacy technology**	• Development objects are created in language version Standard ABAP • For new objects, ABAP Cloud rules are enforced by ATC • Old objects can be exempted from ABAP Cloud ATC checks via base line	Packages in HOME or existing software component for classic ABAP

Table 5.2 – Setup summary

5.1.4 Transformation Towards ABAP Cloud

The ABAP language version for individual development objects in a software component for classic ABAP can be changed to ABAP for Cloud Development if required (→ Figure 5.5).

This option can be used to transform code step-by-step to prepare it for a move to a software component based on ABAP Cloud[20]. The ABAP Test Cockpit check variant `ABAP_CLOUD_READINESS` can be used to check beforehand whether the rules of the ABAP Cloud development model are respected during this transformation[21].

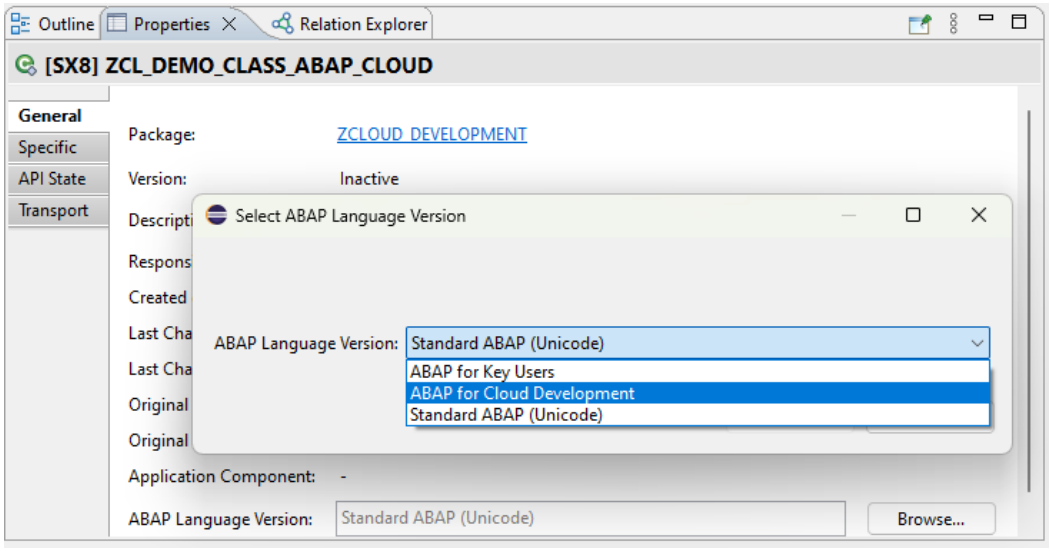

Figure 5.5 - Setting the ABAP Language Version for individual objects in ADT.

5.1.5 Custom Code Use Cases and Their Related Tier

To make the three-tier model more tangible, here are some recommendations for typical custom code use cases and their mapping to the three-tier model:

CATEGORY	USE CASE	RELATED TIER	RECOMMENDATION
EXTENSIONS	Custom field on DB table or CDS view via released extension include	1	
	Implementation of a released SAP BAdI	1	
	Custom field on DB table or CDS view via extension include that has not been released or via classic append	3	Monitor when extension include will be available and adapt accordingly
	Implementation of an SAP BAdI that has not been released	3	Monitor when BAdI will be released and adapt accordingly
	Extension of an SAP Fiori app (RAP based)	1	
	Extension of an SAP Fiori app (SEGW, BOPF, UI5)	3	
	Extension of an SAP application with legacy UI technology, e.g., SAP GUI transaction	3	Monitor if SAP Fiori application will be available

[20] Note that the individual change of the ABAP language version is possible for ABAP code objects (e.g., classes, interfaces) as of SAP S/4HANA release 2019, while the change of the language version for DDIC, CDS and other objects is possible as of SAP S/4HANA release 2022

[21] For more information, see also: How to check your custom ABAP code for SAP BTP ABAP Environment

CATEGORY	USE CASE	RELATED TIER	RECOMMENDATION
CUSTOM APPLICATIONS	Custom SAP Fiori app (RAP based) for the core SAP S4/HANA Cloud scope	1	
	Custom SAP Fiori app (SEGW, BOPF, UI5)	3	Prefer to use RAP. Exception could be know-how or reuse
	Custom application with legacy UI technology (e.g., ABAP report with ALV, WD application)	3	Should be prevented
	SE54 based BC UI	3	
WRAPPER API	Wrapper classes around SAP objects that have not been released (e.g., BAPI) for the core SAP S/4HANA Cloud scope	2	Monitor when the released RAP interface is available and replace the wrapper
	Wrapper CDS view for SAP table or CDS views that have not been released for the core SAP S4/HANA Cloud scope	2	Monitor when a released CDS view is available and replace the wrapper
MODIFICATIONS	Applying SAP note with manual corrections, e.g., DDIC object from SAP BASIS	3	Reset modification once the correction is part of the core
	Implementation of user/customer exits (e.g., SAPMV45A)	3	Monitor if SAP released BAdI can be used instead
	Modification (Business driven)	3	Should be allowed only in exceptional cases

Table 5.3 - Custom code use cases and their mapping

A major difference between SAP S/4HANA Cloud Public Edition on the one hand and SAP S/4HANA Cloud Private Edition on the other is the Identity and Access Management (IAM) and Communication Management (COM). In SAP S/4HANA Cloud Private Edition, there is no strict content separation between SAP and customers with regard to users, roles, and communications. The administrative SAP Fiori apps for IAM and COM are not available. Instead, the classic transactions must be used (PFCG, SM59, SOAMANAGER, …).

5.2 Tier 2: When and how to use classical ABAP code to mitigate missing APIs

As explained above, the target for any new extension is tier 1. If a suitable public API is missing, custom wrapper objects around the non-public SAP APIs will be created in tier 2. The wrapper objects are developed based on classic ABAP and need to be released for cloud development as shown in Figure 5.6.

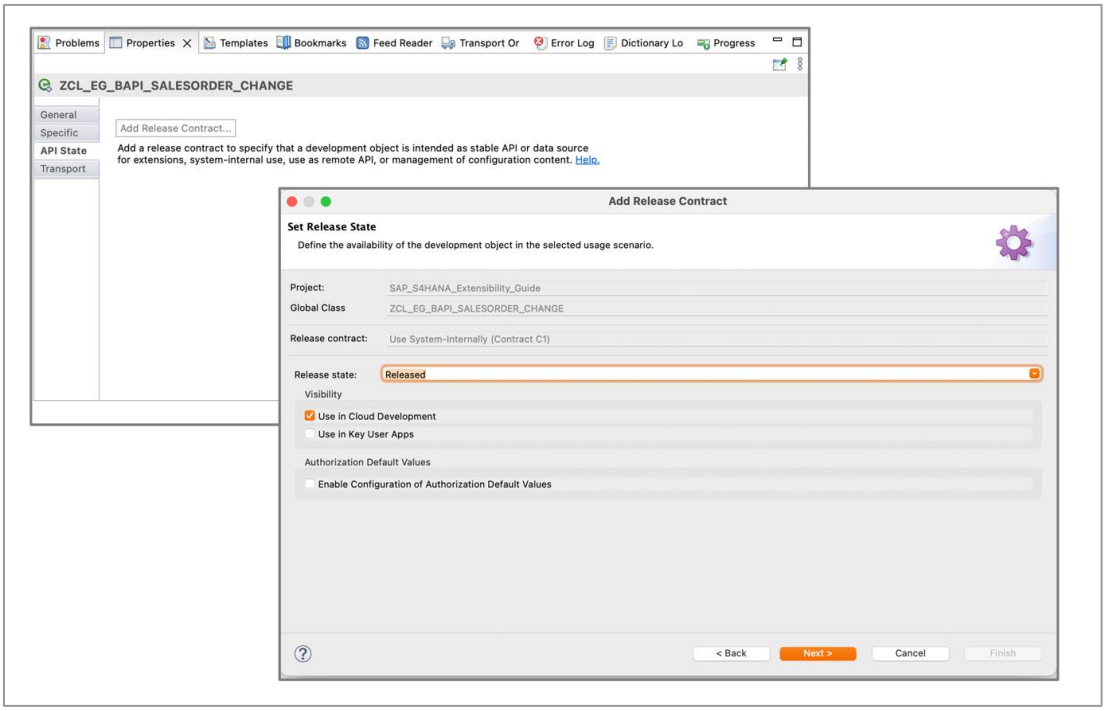

Figure 5.6 - ABAP class released for cloud development.

The rest of the code and other objects created in tier 2 will follow the ABAP Cloud development model. Once SAP releases the necessary APIs, the non-released objects used in the wrappers can be replaced by public local APIs and the code can be moved from tier 2 to tier 1.

In the following we provide more guidance as to how to develop such wrappers:

- **CDS views**
  If a suitable SAP CDS view exists that is not yet released, a custom CDS view on top of the SAP CDS view can be created and released for cloud development. CDS views that are referenced in the SAP CDS views (e.g., association targets or value help views) may need a custom wrapper as well.
  If a suitable SAP CDS view does not exist at all, a custom CDS view on top of the database table needs to be created.
- **ABAP classes/interfaces**
  If a suitable SAP function module or class exists that is not released, a custom wrapper class can be created and released for cloud development.
  When creating such a wrapper class, no copies of non-released data types (data elements, structures, table types) used in the signature of the object will be created. Instead, those data types will be wrapped as well, using TYPES declarations in the public section of the wrapper class. In this way, only the wrapper class itself needs to be released for cloud development.
  Source code 5.1 shows an example of a wrapper of `BAPI_SALESORDER_CHANGE`.
- **Authorizations**
  If a classical SAP API is used, it is likely that non-released authorization objects will be checked. These authorization objects cannot be used with ABAP Cloud. Nevertheless, PFCG roles can be built using both authorization objects that have been released as well as ones that have not been released. To make the non-released authorization objects required for wrappers transparent, it is recommended to provide SU22 data for the wrapper when releasing it for cloud development and to create SU22 variants for each designated usage of the wrapper. These can be used for role maintenance.

Further information and details regarding tier 2 will be provided in a dedicated guide.

```
CLASS zcl_xx_bapi_salesorder_change DEFINITION
  PUBLIC
  FINAL
  CREATE PRIVATE.

  PUBLIC SECTION.
    TYPES salesdocument TYPE bapivbeln-vbeln.
    TYPES order_header_in TYPE bapisdh1.
    TYPES order_header_inx TYPE bapisdh1x.
    TYPES t_return TYPE STANDARD TABLE OF return WITH EMPTY KEY.
    TYPES t_partners TYPE STANDARD TABLE OF partners WITH EMPTY KEY.
   ...

    CLASS-METHODS bapi_salesorder_change
      IMPORTING
        salesdocument         TYPE salesdocument
        order_header_in       TYPE order_header_in OPTIONAL
        order_header_inx      TYPE order_header_inx
    ...
      EXPORTING
        return                TYPE t_return
      CHANGING
        partners              TYPE t_partners OPTIONAL
       ...

  PROTECTED SECTION.
  PRIVATE SECTION.

ENDCLASS.



CLASS zcl_xx_bapi_salesorder_change IMPLEMENTATION.

  METHOD bapi_salesorder_change.

    CALL FUNCTION 'BAPI_SALESORDER_CHANGE'
      EXPORTING
        salesdocument         = salesdocument
        order_header_in       = order_header_in
        return                = return
        partners              = partners
     ...
  ENDMETHOD.

ENDCLASS.
```

Source code 5.1 - Simplified wrapper for BAPI_SALESORDER_CHANGE

5.3 Tier 3: Classic extensions

Extensions that do not follow the rules of ABAP Cloud development and cannot be developed in tier 1 and 2 need to be developed in tier 3. The clear recommendation is to control and minimize extensions developed in this tier because they may lead to errors during upgrades. An extension in tier 1 and 2 is always to be preferred over an extension in tier 3.

In the following sections we give guidance on how to reduce the negative impact of such extensions for selected extension scenarios.

5.3.1 Using classic structure extensions

Structure extensions are extensions of the data model of database tables, structures, CDS views, and services (OData, SOAP, BAPI, IDOC, …).

A general rule for related objects is: no modifications to the corresponding SAP objects.

DDIC

- Append fields to non-released structures and database tables using existing customer includes (CI includes) or extension includes (EEW includes).
- Use domain value appends, search help appends, search help exits.

CDS

- Use CDS extends.
- Use CDS metadata extension.

OData Services

- Use redefinition of OData services.
  More information can be found in: How to redefine RDS based OData services? | SAP Blogs.

5.3.2 Using classical business logic extension techniques

Classic business logic extensions should follow the below recommendations below:

- BAdIs: even if a BAdI is not released for usage in cloud development it is a good choice for customers to extend the application. Most likely the BAdI will be released with an upcoming release or will be replaced with a released BAdI as successor.
- User exits, like VOFM, SAPMV45A: user exits are coding parts pre-defined by SAP (form routines, includes) in an SAP namespace where the customer can implement custom code to extend the SAP business process. The predefined coding parts from SAP are stable and typically do not lead to issues after an upgrade. Technically the custom code here is treated as a modification. It is planned that BAdIs will replace these extensions over time.
- Customer exits (SMOD/CMOD): customer exits are the predecessor technology to BAdIs and will be replaced. Anyhow you can still use them in exceptional cases where no BAdIs exist yet in the application. Most likely they will be replaced completely with upcoming releases.
- Explicit enhancement spots: these are extension points that were mainly created by SAP to enable industry-specific adaptations to the core ERP applications. We recommend using them only in exceptional cases to include custom-defined BAdIs into the SAP core. You should not directly include extension code in the enhancement spot.
- Implicit enhancement spots: this extension technique is very similar to modifications and should therefore not be used to extend the core applications.

5.3.3 Modifications

Modifications of SAP core objects should be avoided completely. If they cannot be prevented, we strongly recommend using the modification assistant. This makes the adjustment of the modifications after an upgrade manageable in contrast to free-style modifications. Moreover, the ATC check *Search customer modifications* allows one to establish a governance for this purpose.

6 MANAGING AND TRANSFORMING EXISTING CODE IN SAP S/4HANA CLOUD PRIVATE EDITION AND SAP S/4HANA ON-PREMISE

6.1 How to handle existing classic ABAP custom code in parallel with the new cloud ready extensions

In chapter 5 we described how customers should apply the cloud extensibility model in a new implementation project. Now we want to look at SAP S/4HANA projects where a classical conversion is the basis for the transition. In this scenario the customer must handle existing classic ABAP custom code.

In the 3-tier model described in chapter 5 the converted classic ABAP custom code will be in tier 3.

The goal is to minimize and control the content of tier 3. Therefore, we recommend leaving behind as much old legacy code as possible when converting to SAP S/4HANA – cleaning up after the conversion involves much more work.

We do not want to repeat all the recommendations for custom code in a conversion project as this is already described in the whitepaper:

Custom Extensions in SAP S/4HANA Implementations - A Practical Guide for Senior IT Leadership

In a nutshell the procedure is:

- Analyze the classic ABAP custom code with the Custom Code Migration App to estimate the impact.
- Detect unused classic ABAP custom code based on usage analysis and remove it during the conversion.
- Adapt the classic ABAP custom code by using ABAP Test Cockpit and automate the adaptation with quick fixes.

We now want to focus on the recommendations for customers as to how to apply the cloud extensibility model after a system conversion. In principle, the three-tier model is also valid for these customers with the difference that the amount of legacy code in tier 3 is much higher. Rewriting the complete legacy code with the cloud extensibility model is expensive and would be like a new implementation project. Therefore, we recommend the following options for the existing legacy custom code:

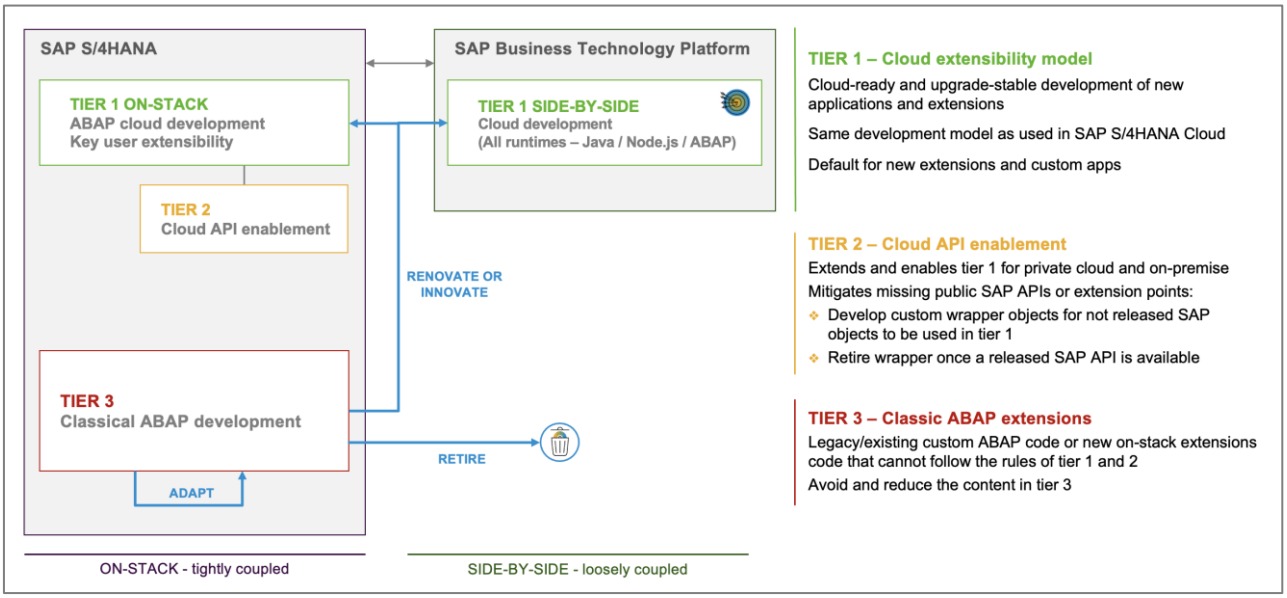

Figure 6.1 - Options for custom code migration

Retire unused code: the analysis of classic ABAP custom code in SAP S/4HANA projects shows that roughly 60% (sometimes even more) of the code is not used productively in a typical ERP system. If you did not use the option of removing classic ABAP custom code during system conversion, we recommend that

you use the *custom code usage analysis* in the productive SAP S/4HANA system to detect unused code and retire the code accordingly.

After an SAP S/4HANA transition large parts of the classic ABAP custom code are no longer relevant because the use case of the customer extension is now part of the core application. We recommend removing the customer extension and using the core logic instead.

The same applies for clones of SAP code. Remove the clones and use the cloud extensibility model to meet the business requirements directly in the core application.

Typical tasks for custom code retirement:

- Collect usage data in production with transaction SUSG/SCMON.
- Analyze usage data in SAP's Custom Code Migration app.
- Remove/backup unused classic ABAP custom code with the Custom Code Migration app.
- Use the clone finder.

Renovate & Innovate on SAP BTP: another portion of the classic ABAP custom code no longer fits the modern approach to Intelligent ERP. Instead, the SAP Business Technology Platform (SAP BTP) should be used with its broad set of business services and innovative technologies like machine learning.

Replace the classic ABAP custom code and move the extension use case to SAP BTP. Especially for loosely coupled extensions, this is the right way to keep the core clean[22].

Typical tasks are:

- Analyze dependencies between extensions and the core with the Custom Code Migration App (more information: Custom Code Migration app in SAP community).
- Transfer custom code from SAP S/4HANA to SAP BTP.
- Adapt custom code in SAP BTP via ATC and Quick fixes.

Renovate & Innovate on the SAP S/4HANA ABAP platform: after a system conversion there is still a certain amount of custom code which does not match the categories above. These are mainly extensions or even legacy modifications of SAP code, which are tightly coupled with the core application logic of SAP S/4HANA. For such code the recommendation is to analyze the business case and decide if a new implementation based on the ABAP Cloud development model is the way to go. In that case follow the guidance of chapter 5. For an efficient use of ABAP Test Cockpit to control the usage of ABAP Cloud a baseline might be helpful to suppress initial findings in legacy code.

Typical tasks are:

- Analyze custom code complexity with Custom Code Migration App to detect TCO drivers.
- Make your custom code compliant with the ABAP Cloud development model following the guideline from chapter 4.
- Replace usage of not-released SAP objects with released APIs, e.g., CDS views instead of DB tables.
- To extend or change an SAP object use only released extension points (released BAdIs, released RAP BOs, custom fields, or logic from key user extensibility).

Adapt

If you want the custom code to run on SAP S/4HANA just like before the conversion - without using the new cloud development model - you must at least adapt the code relating to the simplification database. We recommend using ABAP Test Cockpit and quick fixes in ADT to adapt your custom code. Using quick fixes will significantly reduce your workload for adapting custom code by means of semi-automatic adaptation, and you can expect up to a 60% automation rate for the most frequent findings of the typical SAP S/4HANA simplification use cases. Consequently, we recommend also using SQL Monitor to detect performance

[22] The *clean core* concept is a mindset and philosophy supported with governance and guidelines that lay the foundation for a flexible and future ready ERP. The concept includes different dimensions as extensibility, integration, data, processes, and operations. With the focus on extensibility, the clean core means to keep extensions strictly separate from the SAP application. Extensions access SAP business objects only through well defined, released and upgrade-stable interfaces.

optimization candidates which can be optimized with the new ABAP code pushdown capabilities, for example AMDP.

Typical tasks for **adaptation**:

- Adapt usage of SAP objects based on SAP S/4HANA readiness ATC checks, the Simplification Database, and the Quick fixes in ADT.
- Usage of SQL Monitor to detect performance optimization candidates.
- Adapt your SQL usage to make it SAP HANA ready, e.g., fix ORDER BY issues.
- Use CDS, AMDP and ABAP SQL to adapt performance-critical SQL queries.

The following overview gives some guidance for the transition of classic ABAP custom code towards the cloud extensibility model:

	USE CASE	RELATED TIER	GUIDANCE/RECOMMENDATION
EXTENSIONS	Existing custom field on DB table or CDS view	3	Migrate to released extension includes (if available)
	Existing implementation of an SAP BAdI	3	Migrate to released BAdI and adapt the implementation to *ABAP Cloud*
	Existing extension of an SAP Fiori app (SEGW, BOPF, UI5)	3	Migrate the SAP Fiori app extension (or parts of it) to the newly delivered RAP application if available
	Existing implementation of user exit (e.g., SAPMV45A)	3	Migrate to released BAdIs
	Existing modifications	3	Check if released BAdI is available to replace the modification
	Existing enhancement implementation (implicit, explicit)	3	Check if released BAdI is available to replace the modification
	Existing Custom SAP Fiori app (SEGW, BOPF, SAP UI5)	3	Refactor with RAP/SAP Fiori Elements
CUSTOM APPLICATIONS	Classic ABAP reports (with or w/o ALV)	3	Refactor with RAP/SAP Fiori Elements
	Dynpro and Web Dynpro Applications	3	Refactor with RAP/SAP Fiori Elements

Table 6.1 - Overview: transition of classic ABAP custom code towards the cloud extensibility model

6.2 How to handle existing custom code in a RAP compliant fashion

It is no surprise that a lot of existing custom ABAP code is based on legacy UI technologies like (Web) Dynpro or Business Server Pages (BSP). Some of the apps based on these technologies can be transformed with minimal effort, e.g., using the generator for repository objects (a.k.a. RAP generator) or by using the wizard for creating SAP Fiori Business Configuration apps (→ Chapter 0). Nevertheless, transformation of most of the apps to SAP Fiori can be very complex. Consequently, many of these applications need to be rewritten to make them future proof. The following blog post describes how to assess existing custom ABAP code to create a RAP modernization strategy.

Modernization with the ABAP RESTful Application Programming Model (RAP)

7 MORE INFORMATION

KEY USER EXTENSIBILITY

- Custom Extensions in SAP S/4HANA Implementations - A Practical Guide for Senior IT Leadership
- The Key User Extensibility Tools of SAP S/4HANA
- SAP S/4HANA Extensibility: A Learning Journey
- SAP Help Portal - Personalizing and Adapting Apps
- Low-Code/No-Code Development Tools
- SAP Help Portal - Extending an SAP Fiori Application
- Extending SAP-delivered SAP Fiori elements apps using adaptation projects to create SAP S/4HANA app variants

SAP FIORI AND SAP UI5

- SAP Fiori tools
- Developer Adaptation
- ABAP CDS annotations
- Developing Apps with SAP Fiori Elements
- SAP UI5 documentation

ABAP RESTFUL APPLICATION PROGRAMMING MODEL

- RAP Developer Guide on SAP Help Portal
- Modernization with the ABAP RESTful application programming model (RAP) | SAP Blogs
- Modern ABAP Development with the ABAP RESTful Application Programming Model
- Working with ABAP Cross trace tools
- Unit testing with ABAP unit
- Documenting ABAP development objects | SAP Blogs)

CUSTOM CODE TRANSFORMATION

- How to check your custom ABAP code for SAP BTP ABAP Environment | SAP Blogs
- Custom Extensions in SAP S/4HANA Implementations - A Practical Guide for Senior IT Leadership
- How to redefine RDS based OData services? | SAP Blogs
- Get started with the ABAP custom code migration process | SAP Blogs

ABAP CLOUD

- ABAP Keyword Documentation

CUSTOMER INFLUENCE CHANNELS

- ABAP Platform (https://influence.sap.com/sap/ino/#campaign/2911)
- SAP S/4HANA (https://influence.sap.com/sap/ino/#/campaign/2759)

BUSINESS CONFIGURATION

- Create a Business Configuration Maintenance App

NONE ABAP BASED DEVELOPMENT ON SAP BUINESS TECHNOLOGY PLATFORM

- The SAP Business Technology Platform
- Start developing on SAP Business Technology Platform
- SAP Business Technology platform developer center

LIST OF FIGURES

LIST OF TABLES

LIST OF SOURCE CODE EXAMPLES

GLOSSARY

A	
ABAP for cloud development	Refined, simplified, and standardized ABAP language (based on → Classic ABAP) for cloud-ready programming. Only modern ABAP statements and concepts, with a focus on cloud-enabled, object-oriented design and modern programming models are allowed
ABAP Development Tools (ADT)	Integrated development environment (IDE) for ABAP development implemented as a plugin for the Eclipse platform (www.eclipse.org)
ABAP Platform reuse services	A set of technical and business-related services to be easily integrated and used in ABAP development, addressing recurring development requirements
ADT	→·ABAP Development Tools
ABAP Cloud (development model)	A modern development model for cloud-ready, and upgrade-stable ABAP applications and extensions that follows the clear cloud rules to avoid the need to make adaptations after a version change of the SAP software
ATC checks	→ ABAP Test Cockpit
ABAP Test Cockpit (ATC)	Toolset directly integrated in → ABAP Development Tools for doing static and dynamic quality checking of ABAP code and associated objects
B	
Business user	Persona that personalizes SAP SaaS applications (e.g., adapt UI layouts, report views and dashboards, …), by using → key user (extensibility) tools
C	
Citizen developer	Persona that composes and extends applications and automations on SAP Business Technology Platform using the → Key user (extensibility) tools SAP Build apps, SAP Business application studio, SAP Build Process automation and others)
Classic ABAP	Highly optimized programming language for business applications, both small-scale and large-scale. Classic ABAP allows developers to use the full range of ABAP statements and concepts provided by SAP
Classic (custom) (ABAP) code	ABAP code and artefacts developed for the traditional ABAP programming models such as Dynpro, BSP and Web Dynpro
Classic custom ABAP extensions	→ Classic extensibility (model)

Classic extensibility (model)	An extensibility model that offers a great degree of freedom for classical ABAP code, but that requires serious adaptation efforts after an upgrade
Clean core	The *clean core* concept is a mindset and philosophy supported with governance and guidelines that lay the foundation for a flexible and future ready ERP. The concept includes different dimensions as extensibility, integration, data, processes, and operations. With the focus on extensibility, the *clean core* means to keep extensions strictly separate from the SAP application. Extensions access SAP business objects only through well defined, released and upgrade-stable interfaces.
Cloud extensibility model	An extensibility model for SAP S/4HANA that ensures that the customer and partner extensions continue to run without any change or adaptation effort independent of changes made by SAP during cloud updates
Cloud-optimized ABAP language	→ ABAP Cloud

E

Embedded steampunk	SAP internal project name for → SAP S/4HANA Cloud ABAP Environment

H

Hybrid extensions	An extension scenario where different extensibility options (→ key user extensibility, → on-stack (developer) extensibility, → side-by-side extensibility) are jointly used

I

In-app extensibility	→ key user extensibility

K

Key user	Persona that extends SaaS applications by using → key user (extensibility) tools to implement simple business requirements (custom fields, layout changes, validation rules, simple custom logic, …)
Key user (extensibility) tools	A set of web-based, easy-to-use low-code/no-code tools used for → key user extensibility
Key user extensibility	An extensibility approach that allows key users to make small adaptions to standard applications or create lightweight custom apps without advanced development skills. The key user implements the extensions and apps directly on the software stack using dedicated key user apps that provide users with detailed guidance and convenient standard solutions for common problems

L

Local public interfaces	Local public interfaces provide non-remote access to SAP APIs. These stable interfaces are the key to the lifecycle independence of SAP code and customer and partner extensions
Logical unit of work	A sequence of programming steps and data updates distributed across multiple work processes, whose database changes are updated within a single database commit
Loosely-coupled extensions	Extensions with a low proximity to data, transactional behavior, or apps and therefore typically run on a separate platform like the → SAP Business Technology Platform
Low-code/no-code tools	Toolset to develop apps with little or no code, mostly visual and drag-and-drop simplicity, based on a large library of pre-built content. → key user (extensibility) tools
LUW	→ Logical unit of work

O

On-stack (developer) extensibility	An extensibility approach that allows ABAP developers to create sophisticated extensions and apps using a development environment that supports professional development teams. In the ABAP platform, developer extensibility allows ABAP developers to implement the extensions and apps directly on the software stack using the → ABAP Development Tools
Original (RAP) BO	The original RAP BO is the base BO on which the RAP BO extension bears

P

Private cloud	A cloud infrastructure that allows customers to run their solutions with maximum flexibility. However, adaptations are typically necessary after a version change of the SAP solution
Public cloud	A standardized cloud infrastructure that allows customers to run their solutions with minimal cost. A version change of SAP software has no impact on the custom extensions
Public interface	An interface such as a Public API that is publicly documented and available with stable semantics after a version change of SAP software

R

Released APIs	An API that is publicly documented and released for use in extensions
Released interfaces	→ Released APIs
Released objects tree	A view in → ABAP Development Tools project explorer showing all SAP released → local public interfaces and local extension points

Remote API	An interface of a software component allowing communication between two or more stacks, providing a service to other software components

S

SAP BTP	→ SAP Business Technology Platform
SAP Business Technology Platform, ABAP Environment	SAP Platform-as-a-Service (PaaS) offering for ABAP development that enables developers to make use of their traditional on-premise ABAP knowledge to develop and run ABAP applications in the SAP Business Technology Platform, either as an extension to SAP software or as standalone applications (https://community.sap.com/topics/btp-abap-environment))
SAP Business Technology Platform (SAP BTP)	SAP Business Technology Platform brings together application development, data and analytics, integration, and AI capabilities into one unified environment optimized for SAP applications (https://community.sap.com/topics/business-technology-platform)
SAP extension points	Extension points (such as Business Add Ins) offered by SAP to customers and partners to extend the SAP logic with custom and partner logic through well-defined interfaces
SAP S/4HANA Cloud ABAP Environment	SAP S/4HANA Cloud ABAP Environment enables modern ABAP development for SAP customers and partners of → tightly coupled extensions *running directly on the SAP S/4HANA technology stack*
SAP S/4HANA	SAP's flagship enterprise resource planning product (ERP) is available in the cloud (→ SAP S/4HANA Cloud Public Edition or → SAP S/4HANA Cloud Private Edition) or on-premise (→ SAP S/4HANA on premise)
SAP S/4HANA on premise	SAP S/4HANA on premise offers enterprise management capabilities for the Intelligent Enterprise. SAP S/4HANA is deployed and managed in the customer's premises
SAP S/4HANA Cloud Private Edition	SAP S/4HANA Cloud Private Edition offers enterprise management capabilities for the Intelligent Enterprise. SAP S/4HANA Cloud Private Edition is deployed and managed by SAP in private cloud data centers offered by hyper scalers and SAP
SAP S/4HANA Cloud Public Edition	SAP S/4HANA Cloud Public edition offers enterprise management capabilities for the Intelligent Enterprise. SAP S/4HANA Cloud Public Edition is deployed and managed by SAP in public cloud data centers offered by hyper scalers and SAP
SAP API Business Hub	An SAP operated platform to explore, discover and consume APIs, pre-packaged integrations, business services and sample apps from SAP (https://api.sap.com/)
Side-by-side extensibility	An extensibility approach that allows developers to implementing sophisticated development projects, such as creating custom UIs, custom applications (in Java, Node.JS, ABAP), or analytical or workflow extensions. The development projects are implemented on → SAP

	Business Technology Platform (SAP BTP) and integrated via released remote APIs
Side-by-side extension	→ side-by-side extensibility
Steampunk	SAP internal project name for → SAP Business Technology Platform, ABAP Environment

T

Tightly coupled extensions	Extensions with a proximity to data, transactional behavior, or apps and therefore typically run on the same stack