

PUBLIC

Clean core extensibility

White paper

Successor to previous white paper on
extensibility: Clean core extensibility
for SAP S/4HANA Cloud

Table of contents

Executive summary	3	4	How to achieve a clean core for extensibility?	25
1	Introduction	5	4.1 RISE with SAP Methodology	25
		4.1.1	Governance and maturity—An approach to assessing clean core governance	26
2	Why a clean core is important	7	4.1.2 Clean core measurement framework	27
2.1	Overcoming obstacles and unlocking innovation	7	4.1.3 Measurability: Extensibility KPIs	28
3	What a clean core means for extensibility	8	4.2 Reaching clean core with “Stay Clean” and “Get Clean”	31
3.1	Extensibility as part of the 5 guiding principles of clean core	8	4.3 Stay Clean: How to create clean extensions	32
3.2	Extensibility options: On-stack and side-by-side	9	4.3.1 Functional requirements	32
3.2.1	On-stack: SAP Build (through ABAP Cloud) or classic extensibility	10	4.3.2 Extension architecture	36
3.2.2	Side-by-side with SAP Build	12	4.3.3 Extension implementation	38
3.2.3	The best of both worlds: on-stack and side-by-side extensions	15	4.3.4 Extension deployment	40
3.3	SAP’s clean core extensibility model and the clean core level concept	17	4.4 Get Clean: How to measure the current state of clean core?	42
3.3.1	The clean core level concept	17	4.4.1 RISE with SAP Methodology dashboard	42
3.3.2	Evolution of 3-tier extensibility to level concept	18	4.4.2 Assessing clean core KPIs individually	43
3.3.3	Overview of clean core levels	19	4.5 How to achieve a clean core	44
3.3.4	Why, what and how—Understanding the new clean core levels	20	4.5.1 Key steps and recommendations for your clean core journey	45
3.4	Partner add-ons and clean core	24	5	F.A.Q.
3.4.1	SAP Solution Extensions	24		
3.4.2	Clean core APIs from partners	24	6	Appendix

Executive summary

Why clean core matters

Over time, many ERP landscapes become weighed down by custom code, undocumented changes, and hard-to-maintain integrations. This limits flexibility, slows down upgrades, and increases the total cost of ownership.

The clean core approach addresses these challenges by promoting the use of standard, best practiced-aligned processes and enabling strategic extensions for differentiation. Process standardization supports our customers in the transition to SAP Business Suite, including the public cloud. It enables keeping the systems current through faster and efficient upgrades for the private cloud, enable faster innovation cycles, and make it easier to adopt and leverage AI. It focuses on building resilient business processes supported by seamless integration, efficient operations, and strong data quality.

By embracing a clean core approach, customers can:

- Accelerate innovation cycles
- Minimize technical debt
- Simplify system maintenance and upgrades

Rather than limiting agility, clean core ensures that extensions are done the right way—with long-term stability, scalability, and innovation readiness in mind.

What is new: The evolution of the 3-tier model—Introducing the clean core level concept

SAP is evolving its extensibility guidelines with the **clean core level concept**, a new model designed to simplify qualification criteria and provide a more pragmatic approach to managing custom code in SAP Cloud ERP Private¹ as a part of the journey to SAP Business Suite. By evolving

the previous 3-tier model, this new maturity model categorizes extensions into four distinct levels (A, B, C, and D) based on their architectural integrity, upgrade safety, and alignment with clean core principles.

This approach allows for a more granular assessment of an extension's quality and upgrade stability, moving beyond a simple binary “clean” vs. “unclean” classification. The core recommendation remains a “SAP BTP first” strategy, aiming for the highest level, but the new model also acknowledges the realities of complex system landscapes and supports customers to tailor their clean core journeys. The fundamental principle remains unchanged: decouple extensions from standard SAP code to ensure upgrade stability.



How it works: The four clean core levels explained

Each extension is evaluated using a standardized methodology based on how it is built, how well it is decoupled from the core, and how easily it can be upgraded or maintained:

- **Level A: Extend with SAP Build**—Fully compliant extensions using only publicly released and stable interfaces backed by formal stability contracts.
 - Side-by-side: Built on SAP BTP using pro-code and low-code tools for app development and process automation.
 - On-stack: Built within SAP Cloud ERP Private using the ABAP Cloud development model with publicly released APIs.
- **Level B: Leverage classic APIs**—use SAP’s classic APIs and technologies with well-defined, documented, and generally upgrade-stable interfaces.
- **Level C: Access internal objects**—Partially compliant extensions that rely on SAP internal objects, offering flexibility for legacy scenarios. To reduce upgrade risk, SAP provides the changelog for SAP objects to help identify incompatible changes early and plan upgrades proactively.
- **Level D: Not recommended extensions**—Extensions not considered “clean” that use explicitly non-recommended objects or techniques (e.g., objects marked as “noAPI” in the [Cloudification Repository](#), modifications to SAP objects, direct write access to SAP tables, or implicit enhancements). These represent the highest risk and create significant technical debt.

Conclusion: Empowering a strategic modernization journey

The clean core level concept delivers tangible benefits by:

- **Providing differentiated assessment:** Offering a more nuanced, risk-based evaluation of classic ABAP code.
- **Increasing transparency:** Clearly defining low-risk “classic APIs” versus high-risk “not recommended” objects.
- **Enabling a pragmatic path forward:** Helping customers manage their existing codebases more effectively and prioritizing remediation, focusing first on the highest-risk extensions (Level D).

The clean core level concept does not alter the ultimate goal: building decoupled, upgrade-stable extensions, with Level A as the ideal. Instead, it provides a more practical and transparent framework to achieve it.

It empowers organizations to make informed decisions, understand the specific risks associated with their custom code, and strategically plan their modernization efforts. By prioritizing the elimination of Level D extensions and leveraging greater transparency around Levels B and C, customers can reduce technical debt, simplify future upgrades, and preserve the long-term integrity and agility of their SAP S/4HANA core system.

Further details of the clean core level concept and guidance on achieving clean core for extensibility are provided in the following chapters.

1 Introduction

SAP software is designed to deliver broad and deep functionality across lines of business and industries.

At the core of this breadth is the synergy of applications, data, and AI. Think of a flywheel that builds continuous momentum: applications generate the data, data provides trusted context, and AI turns that context into action. Each action feeds the next, driving resilience, efficiency, and growth.

The SAP Business Suite brings this synergy to life. It provides an integrated cloud portfolio that spans ERP and line-of-business applications, unified by a common data foundation and embedded, industry-specific AI, all underpinned by a business technology platform for integration, automation, and extensibility. Together, these capabilities connect and optimize every function across the enterprise, ensuring AI is grounded in reliable data and executed through proven processes, rather than applied as an isolated wrapper.

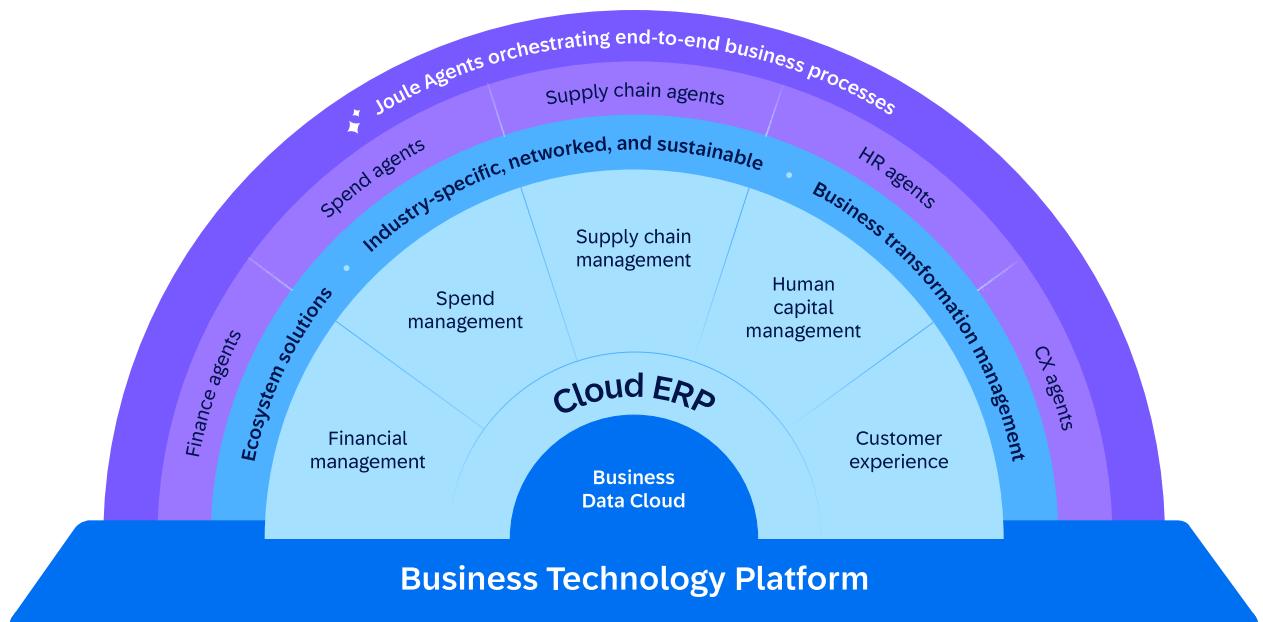


Figure 1: SAP Business Suite

The clean core approach

However, no two businesses are exactly alike, and true differentiation often requires more than just standard processes. SAP recognizes this and enables customers to extend its solutions in ways that foster innovation while preserving the stability of the core system.

This balance between standardization and differentiation lies at the heart of SAP's clean core strategy. Clean core encourages organizations to adopt standardized, upgrade-stable functionality for non-differentiating processes, while providing flexible extensibility options for capabilities that deliver unique business value. It is about

maintaining a stable and reliable digital foundation—reducing complexity, technical debt, and upgrade effort—so that innovation can happen where it delivers the most value.

To fully realize this value, it's important to implement SAP solutions in a way that supports long-term agility and sustainability. When combined with a clean core approach—avoiding excessive extensions and data silos—organizations can reduce maintenance overhead, ensure consistent data quality, and accelerate time to value.

Whether you are an existing on-premise customer or a new customer, SAP provides comprehensive journeys to guide your transition to the SAP Business Suite in alignment with clean core principles. We offer the RISE with SAP journey for existing SAP ERP customers and the GROW with SAP journey for net new SAP customers.

We recognize that many of you have existing on-premise ERP systems with customizations that have led to sizeable technical debt over time, and modernizing those systems to a cloud operating model with SAP Business Suite may pose a significant challenge. To support your transformation, the RISE with SAP journey is anchored in a proven methodology that provides a structured approach and includes a focused clean core success plan (a roadmap with clear tasks and activities), quality gate checkpoints, and a new dashboard that gives visibility into the clean core status of customer systems. RISE with SAP Methodology helps ensure that as customers modernize their on-premise ERP assets, they do so in alignment with clean core principles. Supported by SAP experts and partners, customers can adapt their clean core strategy to their unique business needs.

Scope of this whitepaper

This whitepaper provides a detailed view on the clean core extensibility model to help you modernize your existing ERP systems starting with adopting S/4HANA Cloud Private Edition as part of the RISE with SAP journey. While SAP's clean core strategy is focused on five principles, i.e., Processes, Extensibility, Data, Integrations and

Operations, this paper focuses exclusively on the **extensibility principle**. It provides a detailed view of the extensibility model for S/4HANA Cloud Private Edition, including extension options, architectural considerations, governance recommendations, and implementation guidance to help organizations build business-differentiating innovations without compromising core integrity. By following these guidelines, customers can develop upgrade-stable, maintainable extensions that align with clean core standards and support continuous business transformation in the cloud.

Chapter overview

Chapter 2—Why a clean core is important

Explores the business drivers behind clean core, the limitations of legacy systems, and how clean core enables innovation, agility, and easier upgrades.

Chapter 3—What a clean core means for extensibility

Introduces the clean core principles, with a focus on extensibility options including on-stack and side-by-side models, the clean core level concept, and alignment of partner solutions.

Chapter 4—How to achieve a clean core for extensibility

Presents a practical, step-by-step framework for governance, measurement, and implementation of clean extensions. Covers tools, methodologies (such as the SAP Application Extension Methodology), and KPIs to monitor and enforce clean core adherence.

Chapter 5—FAQ

Addresses common questions and concerns around clean core extensibility strategy, technologies, and implementation.

Chapter 6—Appendix

Provides supporting materials, links, and templates to help implement a clean core strategy effectively.

2 Why a clean core is important

2.1 Overcoming obstacles and unlocking innovation

The accelerating pace of technological innovation presents immense opportunities—but only for organizations with a core that's ready to adapt. Many businesses are held back by legacy on-premise systems that limit their ability to respond to change quickly and effectively. Common challenges include:

- Lack of standardization
- Excessive customization
- Poor data quality

These obstacles increase complexity and reduce agility, making it difficult to capitalize on emerging technologies and trends, such as business AI. To overcome these constraints, organizations

need to transform and modernize their systems through a clean core approach. This means replacing historically grown complexity with standardized, modular, and flexible systems that are easier to maintain and evolve.

A clean core enables:

- Faster innovation cycles
- Reduced time to value
- Improved agility across the business
- Enhanced data quality for AI-driven innovation

By adopting clean core principles, businesses can unlock continuous, incremental innovation and deliver greater business value over time—turning their ERP system from a bottleneck into a catalyst for growth.

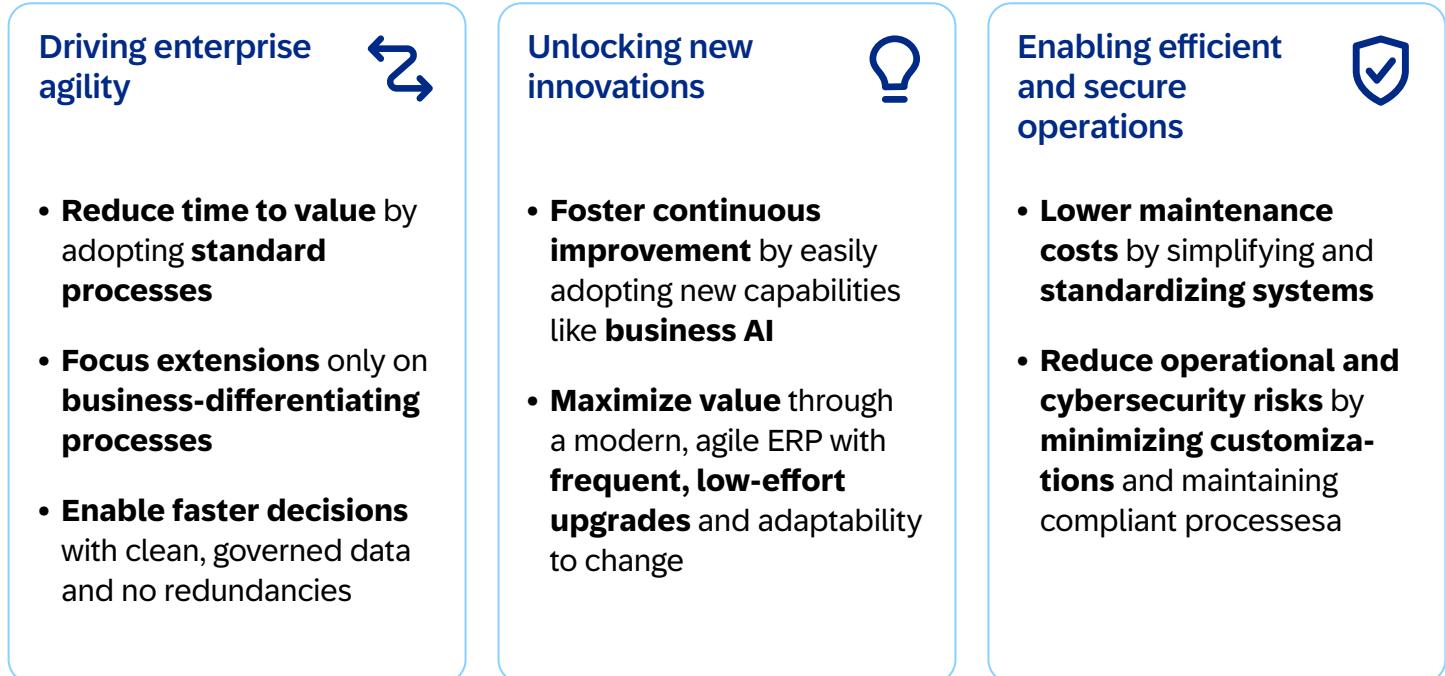


Figure 2: Value of clean core—Shifting IT spend from managing technical debt to driving innovation

3 What a clean core means for extensibility

3.1 Extensibility as part of the 5 guiding principles of clean core

Clean core refers to a **set of guiding principles** that support continuous business transformation by fostering **agile, innovative, and efficient ERP systems**. The principles focus on building resilient business processes and extensions, supported by seamless integration, efficient operations, and high data quality.

Each principle has a specific goal and area of focus.

1. Business processes

Goal: *Keep competitiveness while reducing complexity.*

Focus: Streamline and standardize processes to remain agile without over-customizing the core.

2. Extensibility

Goal: *Decouple extensions from the standard.*

Focus: Enable on-stack and side-by-side extensibility using [SAP Build](#), enabling key users and developers to customize without modifying the SAP core, ensuring easier upgrades and better maintainability.

3. Data

Goal: *Control data according to the latest standards.*

Focus: Ensure data is governed and compliant with current standards for better quality, security, and integration.

4. Integration

Goal: *Keep the landscape reliable and flexible.*

Focus: Use released APIs and modern integration concepts (e.g., event-based integration) to connect systems without dependencies that compromise flexibility.

5. Operations

Goal: *Keep the operations effective and efficient.*

Focus: Streamline system management, automate wherever possible and/or needed, and ensure high availability and performance with minimal manual effort.

While this whitepaper focuses primarily on extensibility, it is closely linked to the other guiding principles. To keep the core clean, enhancements should only be made where processes are clearly classified as differentiating for an organization and delivering measurable value. A clean business process aligns closely with the SAP standard and maintains competitiveness while reducing complexity. Similarly, maintaining clean coding practices depends on embedding quality controls and governance in operational processes, like incorporating thorough code checks within the development lifecycle. Similarly, maintaining clean coding practices depends on embedding quality controls and governance in **operational processes**, like incorporating thorough code checks within the development lifecycle.

By highlighting these interdependencies—such as how efficient business processes enable better extensibility, or how extensibility tools align with operational frameworks—it becomes clear that each principle reinforces the others. Together, they form the foundation for a flexible and sustainable SAP environment.

3.2 Extensibility options: On-stack and side-by-side

For S/4HANA Cloud Private Edition, extensibility use cases are typically supported through two architectural approaches:

- On-stack extensibility for extensions that are tightly coupled with the ERP core and run on the same stack
- Side-by-side extensibility for extensions that are loosely coupled and run on a separate extensibility platform.

In SAP context, on-stack extensibility is used when an extension should be tightly connected to an SAP S/4HANA Cloud solution. Side-by-side extensibility is used when for loosely coupling an extension with the ERP system and run it on SAP Business Technology Platform (SAP BTP). Often, a combination of on-stack and side-by-side

extensibility, both using SAP Build, is needed to fulfill a business requirement.

Extensibility options: when to use what?

The following guidelines and recommendations help decide which parts of an extension should run on SAP S/4HANA as an on-stack extension, and which should run on SAP BTP as a side-by-side extension.

Note: This guidance includes insights to help navigate the decision-making process effectively, but it is not exhaustive. It is important to analyze each extension use case and make an informed decision about which technologies work best for each case. For more information and support, see the [SAP BTP Guidance Framework](#) and use the [SAP Application Extension Methodology](#).

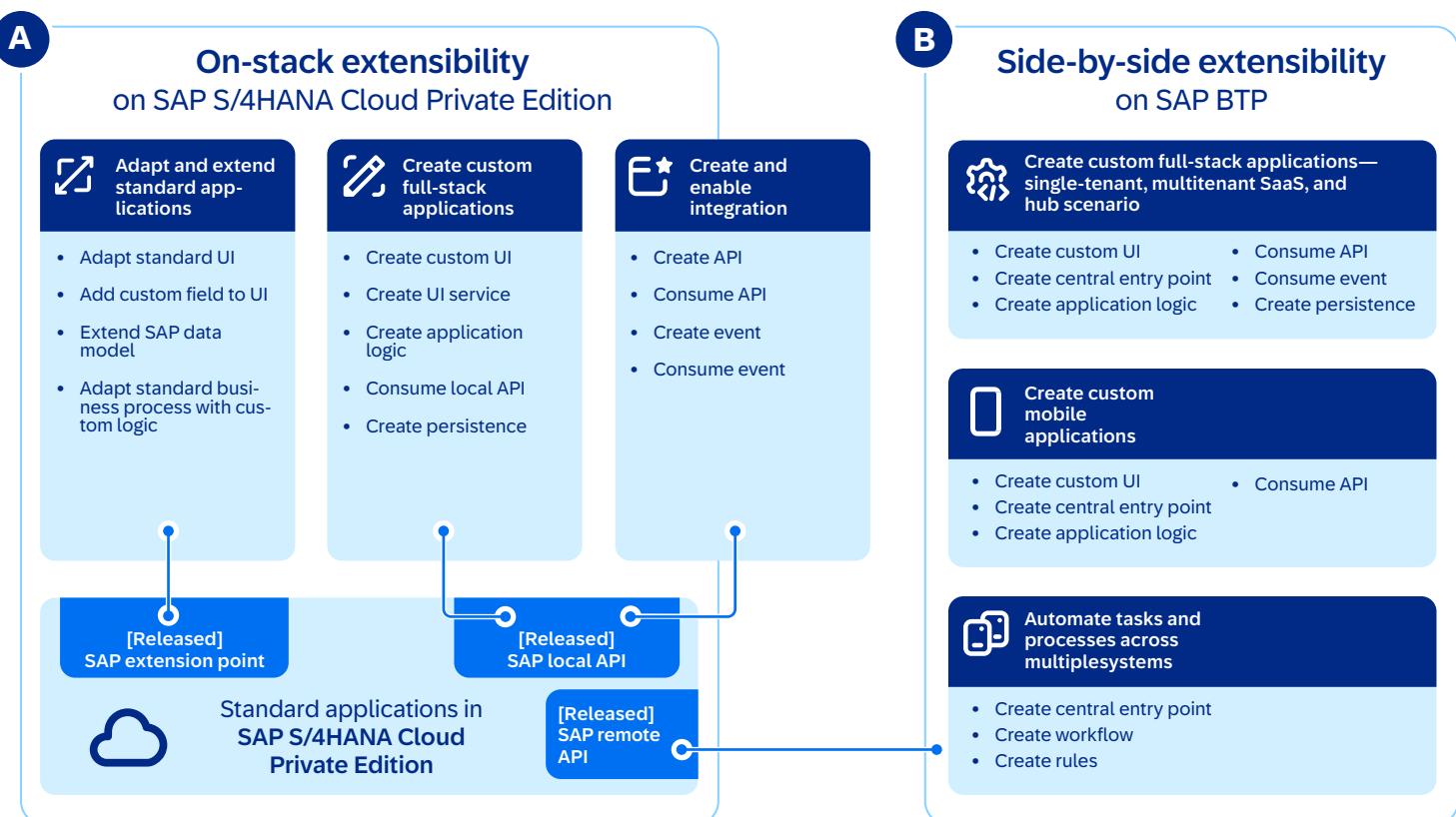


Figure 3: Extension use case pattern for SAP S/4HANA Cloud Private Edition

Figure 3 shows common extension use case patterns for SAP S/4HANA Cloud Private Edition—with on-stack extensions on the left and side-by-side extensions on the right. For more information, refer to the [extension architecture guide on extension use case patterns](#).

When planning extensions, it is often helpful to first explore the possibilities offered on SAP Business Technology Platform. Placing an early focus on the opportunities available on SAP BTP helps identify innovation potential, ensures future upgrade readiness, and benefits from the broad set of platform services. At the same time, on-stack extensions remain an option where business or technical requirements clearly call for them.

3.2.1 On-stack: SAP Build (through ABAP Cloud) or classic extensibility

On-stack extensions are typically required when an extension must operate in close proximity to the SAP S/4HANA Cloud system, interacting directly with core data, transactions, or applications.

Common on-stack use case patterns include:

- **Adapt and extend standard applications:**

Typically used to perform the following extension tasks:

- Adapt a standard user interface (UI)
- Add a custom field to a standard UI
- Extend the SAP data model
- Adapt a standard business process with custom logic within a logical unit of work by implementing the extension point of an SAP application.

- **Create custom full-stack applications:** To develop new applications that require proximity to SAP business objects and services according to the previously mentioned characteristics. Full-stack applications consist of a back-end and front-end implementation and, therefore, span the full stack.

- **Create and enable integration: Uses released local APIs from SAP or events to build a custom remote API or event, for side-by-side extensions on SAP BTP. For more information on released APIs or events, see this page on [developer extensibility](#).**

- **Typical characteristics of an on-stack extension:**

- Transactional consistency is required when updating both SAP standard and custom tables or fields together.
- Custom integrations, such as extensions of standard objects (APIs or services) are needed to enable side-by-side extensions.
- Reading high data volumes, which requires complex SQL queries and joins.
- Frequent access and changes to data from SAP standard tables and fields, leading to many roundtrips.
- Extension of existing SAP S/4HANA Cloud Private Edition core applications (e.g. UI adaptations, custom field / logic / business objects).



On-stack extensions on S/4HANA Cloud Private Edition must be implemented with the ABAP Cloud development model². In SAP S/4HANA Cloud Private Edition and in on-premise environments, ABAP Cloud is also recommended to meet the highest clean core standards (see “Level A” in Chapter 3.3.1), though classic extensibility options are still available.

The role and benefits of ABAP Cloud in the extension strategy

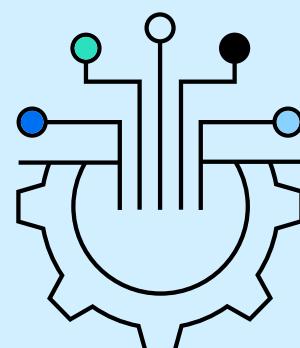
ABAP Cloud³, part of SAP Build, is the standard development model for creating cloud-ready business applications, services, and extensions.

- **Unified development model:** ABAP Cloud provides a single, consistent model for building extensions both on-stack and side by side. It is available on SAP S/4HANA Cloud Private Edition, on-premise systems, and on SAP BTP. Used to develop customer and partner extensions as well as SaaS applications on SAP BTP, it is the primary model SAP uses to develop the core itself. Developers can leverage their existing know-how of ABAP development tools for cloud-ready and clean extensibility.
- **Cloud readiness and clean core compliance by default:** The new language version introduced with ABAP Cloud allows the enforcement of cloud and clean core rules through compiler and syntax checks, ensuring they cannot be bypassed.
- **Comprehensive model-driven architecture:** ABAP Cloud supports online transactional processing (OLTP) supported by the ABAP RESTful application programming model, online analytical processing (OLAP), and the development of new integrations and APIs. Transactional consistency is maintained through a strictly controlled logical unit of work.
- **Rich set of prebuilt technical and business services:** ABAP Cloud includes reusable services such as logging, change documents, number ranges, jobs, factory calendar, currency conversion, units of measure, and many more. These services do not require additional license and can reduce both total cost of development and ownership for complex business applications. They drive a high level of standardization and supportability of custom extensions. Default configurations (calendars, countries, languages, currencies, units etc.) are delivered automatically and are ready to use out-of-the-box.

- **Built-in cloud qualities and tooling:** ABAP Cloud drives cloud-readiness with features like scalability and elasticity. It also enforces code quality that promotes testability, supportability, and documentability of custom extensions. The extensibility model supports end-to-end scenarios, including business configuration, localization and internationalization.

- **Tailored tools for different users:** ABAP Cloud is firmly integrated into proven ABAP concepts, such as lifecycle management or identity and access management. It provides dedicated tools for specific user requirements, such as ABAP development tools for developers and no code/low code tools for key users.

- **AI-powered development:** ABAP Cloud utilizes SAP's copilot Joule to improve developer productivity within the ABAP development tools. An AI software development kit for ABAP allows developers to consume AI services from SAP's generative AI hub on SAP BTP. This way, they can leverage prebuilt AI functionality in custom ABAP extensions. Additionally, ABAP custom code migration is enhanced by generative AI, accelerating the transformation towards a clean core.



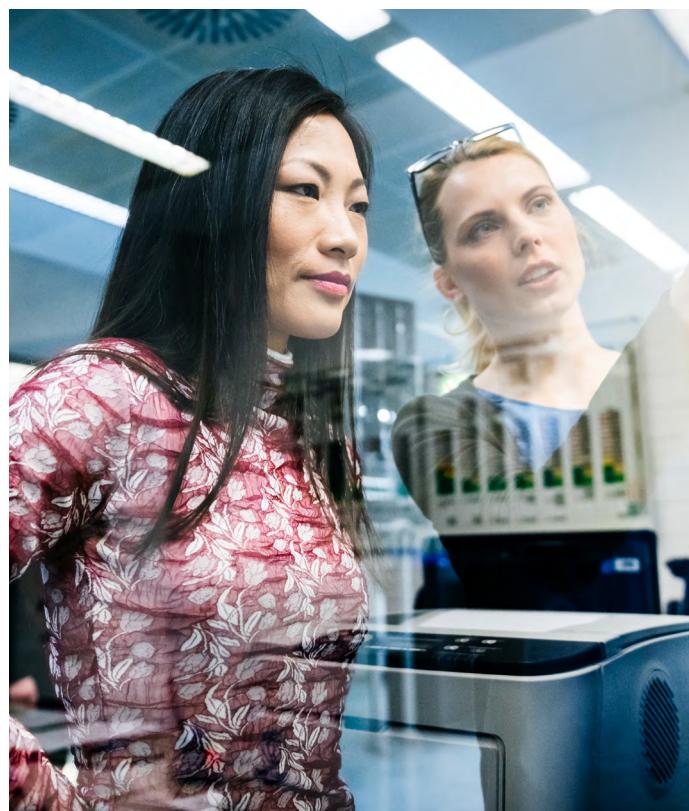
3.2.2 Side-by-side with SAP Build

[SAP BTP](#) is an integrated offering comprising application development and automation, integration, data and analytics, and AI services. The platform offers users the ability to turn data into business value, compose and automate business processes, and—most importantly—build extensions for SAP S/4HANA Cloud Private Edition quickly. It provides integration capabilities to help ensure business processes are connected across SAP and third-party solutions. SAP Build, part of SAP BTP, enables the development of side-by-side extensions that are loosely coupled with the core of SAP S/4HANA. Custom applications and extensions that run on SAP BTP can interact with SAP S/4HANA using released remote APIs and events without modifying the core⁴.

Common use case patterns for side-by-side extensions include:

- **Create custom, full-stack applications:**

- Full-stack, single-tenant applications—Customers and implementation partners develop on SAP BTP.



- Full-stack, multitenant SaaS applications—Independent software vendors (ISVs) in the SAP ecosystem develop on SAP BTP and distribute to their customers. For an implementation example, see this mission on the [SAP Discovery Center site](#).
- Hub scenario—Provides a central hub on SAP BTP to collect and distribute data from various systems, to integrate with several ERP systems and cloud services. For examples, see this page on [github.com](#).

- **Create custom mobile apps:** Provides mobile development capabilities. For an implementation example, see this mission on [SAP Discovery Center](#).

- **Automate tasks and processes across multiple systems:** Provides low-code/no-code capabilities to automate processes. For an implementation example, see this mission on [SAP Discovery Center](#).

- **Typical characteristics and examples for side-by-side extensions:**

- Stand-alone applications or new business process steps that can be decoupled. Data from custom tables and fields is managed independently from SAP standard tables and fields, so transactional consistency is not required.
- Demand to use a broad spectrum of development preferences, skill sets, and languages such as Java, JavaScript, or ABAP, along with open-source libraries.
- Applications for external users who do not have an SAP S/4HANA Cloud Private Edition user account or direct access to the respective ERP system.
- Independent of infrastructure, scalability operations, and lifecycle—enabling frequent updates and continuous delivery.

SAP BTP provides the platform services and technologies needed to implement these extension patterns. For further details, see the next chapter, the [extension architecture guide](#), or the [SAP BTP Developer's Guide](#).

The role and benefits of SAP BTP in the extension strategy

SAP BTP supports two programming models. The [SAP Cloud Application Programming Model \(CAP\)](#) is designed for JavaScript (Node.js), TypeScript, and Java development. CAP services are developed and run on SAP BTP, Cloud Foundry runtime and SAP BTP, Kyma runtime. It offers libraries to implement and consume services, as well as generic provider implementations that handle common requests automatically. CAP also allows the integration of open-source libraries.⁵

The ABAP RESTful application programming model is a central part of [ABAP Cloud](#) and is used in the SAP BTP ABAP environment. It supports model-driven development and includes tools for creating services, business objects, and integrations. Key components like SAP HANA Cloud, job scheduling, logging, printing, and XLSX processing are part of the SAP standard and readily available.

Both models are based on core data services (CDS), a modelling language for defining domain models and services. They are designed for integration with UI frameworks such as SAPUI5 and SAP Fiori, enabling the rapid development of enterprise-grade applications.

Encouraging innovation: By enabling side-by-side extensibility, SAP BTP encourages innovation without disrupting the core SAP S/4HANA suite. Developers can experiment with new features and applications on SAP BTP. When creating extensions, they can choose from supported programming languages such as ABAP Cloud, JavaScript, Java, or even Python (e.g., for AI use cases). Open-source libraries can be used within these extensions.

Reducing the total cost of ownership: Using SAP BTP for side-by-side extensions can help reduce the total cost of ownership. Since these extensions are clean core compliant and loosely coupled from SAP S/4HANA Private Edition, they do not interfere with upgrades, thereby minimizing the effort required for testing and maintenance.

Supporting the transition to a clean core: By providing a platform for side-by-side extensibility, SAP BTP supports the transition toward a clean core for the previously mentioned scenarios and use cases. While extending side-by-side, the SAP S/4HANA suite remains unchanged, delivering clean core benefits outlined earlier.

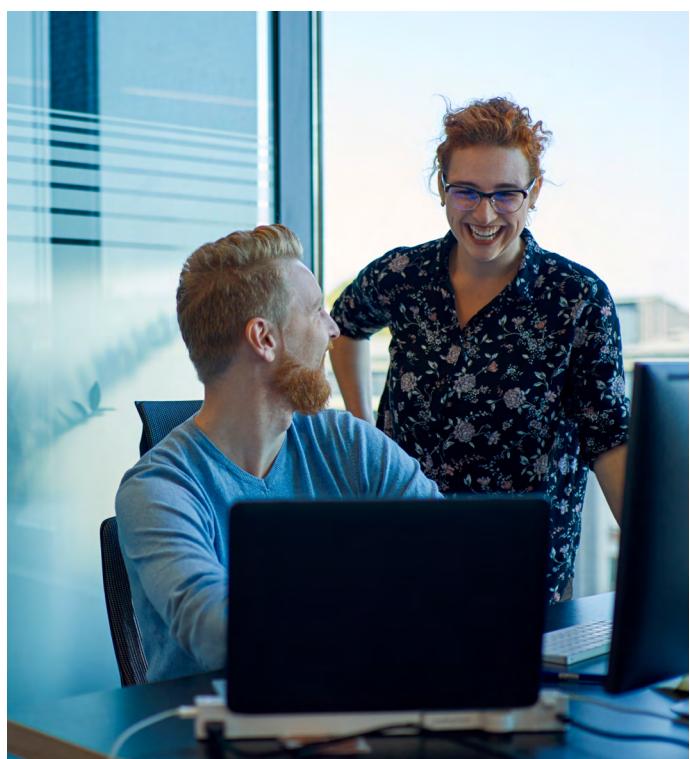
Extend SAP S/4HANA Cloud with SAP Build

The go-to place for creating extensions according to SAP best practices is [SAP Build](#). As part of SAP BTP, SAP Build combines AI-powered pro-code (ABAP, Java, JavaScript) and low-code applications. With tight integration into SSAP S/4HANA Cloud, it empowers both developers and key users to create extensions for SAP software with greater efficiency. It allows customers to accelerate ERP modernization, foster innovation, and automate processes—all within a single comprehensive suite of tools.

Streamline extension of SAP S/4HANA

Private Edition

Application development and process automation with SAP Build leverages the previously described programming models CAP and ABAP Cloud. These models are aligned with clean core principles, enabling scalable, secure, and stable extensions.



Accelerate extension of SAP S/4HANA Private Edition with AI capabilities enabled by Joule

SAP Build unlocks new levels of ERP efficiency through the latest AI capabilities enabled by Joule. SAP's AI Copilot offers conversational guidance in the business context. Uniquely trained on SAP data and processes, it helps developers write code and design workflows for SAP S/4HANA Cloud Private Edition across ABAP, Java, JavaScript, and low-code tools. As an integral part of SAP Build, Joule increases developer productivity and accelerates the development of extensions.

Joule generates high-quality code and code explanations aligned with SAP's programming models, reducing development time for both new and experienced developers and accelerating the transition to cloud ERP with a clean core.

AI capabilities enabled by Joule can also be used to extend ERP through low-code application development. Joule assists with process automation, expediting the creation of workflows for SAP S/4HANA and guiding workflow approvers with automated recommendations.

Fusion team development

SAP Build fosters collaboration between diverse, multiskilled teams (developers and key users alike) providing what they need to create enterprise-grade extensions. It enables asset sharing and seamless technology bridges with built-in governance and lifecycle management (see Figure 4).

The [SAP Build lobby](#) is the central entry point to start an extension project with SAP Build, be it for side-by-side extension development on SAP BTP or for on-stack extension development. Depending on the use case, the respective development environment can be launched, such as ABAP development tools in a pro-code, on-stack scenario, and either CAP or ABAP development tools for a side-by-side extension. The lobby provides many convenient features, like monitoring capabilities and access to prebuilt content to boost productivity.

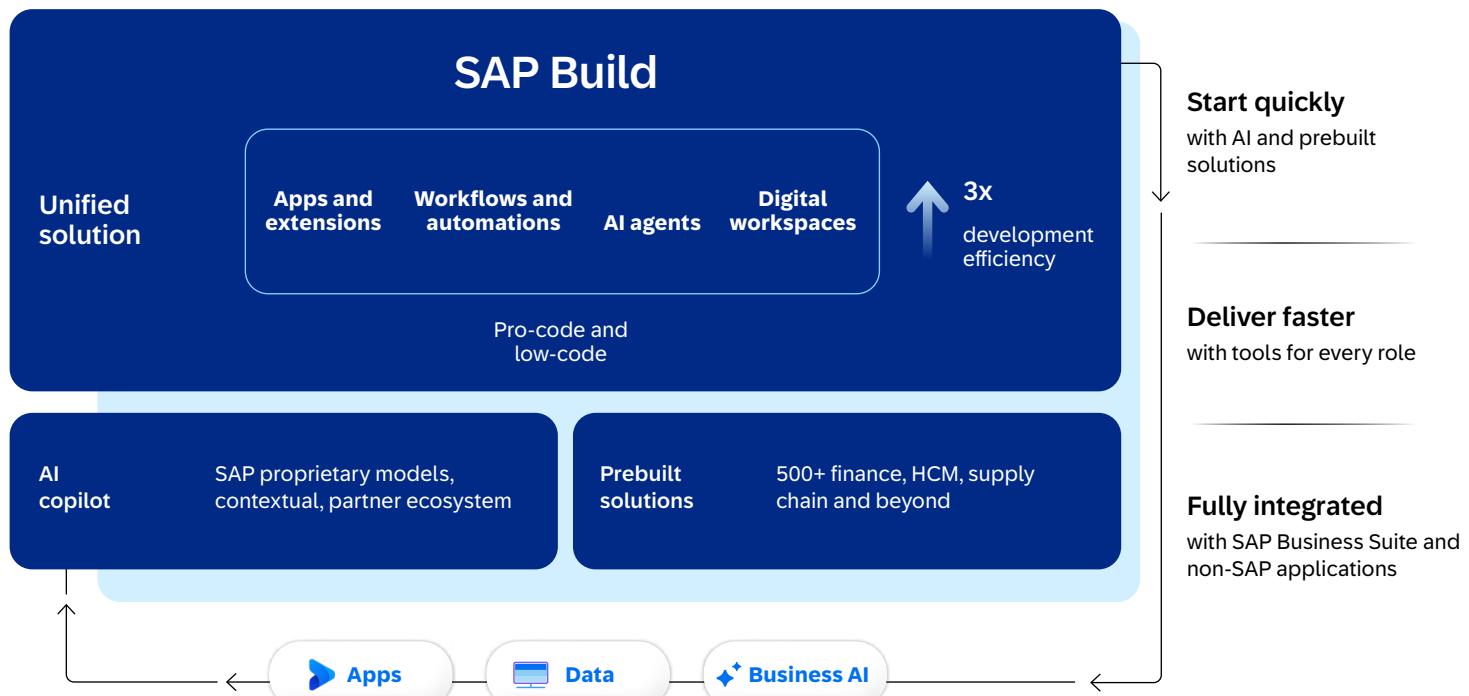


Figure 4: SAP Build—what it comprises, how it can be used and which solutions can benefit from it

3.2.3 The best of both worlds: on-stack and side-by-side extensions

Both on-stack and side-by-side extension options complement each other by providing a comprehensive ecosystem for building, extending, and running enterprise applications. SAP BTP serves as an underlying platform for cloud services and providing integration capabilities for extensions developed with ABAP Cloud, CAP, and low-code tools, each supported by their respective integrated development environments.

In some scenarios, elements from both worlds might be needed. The following are a few examples.

Hybrid deployment capabilities:

Extensibility for SAP S/4HANA Cloud Private Edition often follows a hybrid deployment model, where some components remain on-premise while others run in the cloud. SAP BTP supports this hybrid approach, allowing organizations to deploy extensions in ways that best fit their needs. Both CAP and ABAP Cloud facilitate cloud-based development and extension while integrating with SAP S/4HANA Cloud Private Edition, helping to ensure a cohesive and unified landscape. As described previously, ABAP Cloud also supports tightly coupled extension scenarios through developer extensibility.

Hybrid development practices:

SAP BTP promotes modern development practices such as microservices architecture and DevOps methodologies. Both CAP and ABAP Cloud align with these practices by offering cloud-native development tools for Java, JavaScript, and ABAP developers. On the ABAP side, this synergy allows developers to adopt modern development paradigms while leveraging their existing ABAP skills, thereby accelerating extension projects for SAP ERP systems. Fusion team development (covered in the next chapter) might require developers with different skills to jointly develop an enterprise application. A back-end service for this application might benefit from a tight coupling to the core of SAP S/4HANA, and is therefore built on-stack (via the ABAP Cloud development model). Another developer might

create a mobile UI based on that service using SAP Fiori tools in SAP Build, possibly supported by generative AI capabilities to increase productivity. The resulting full-stack application is deployed partly on-stack and side-by-side.

Integration and connectivity:

SAP BTP provides robust integration capabilities for connecting SAP S/4HANA Cloud Private Edition with other SAP and third-party systems. ABAP Cloud integrates with SAP BTP's integration services, enabling developers to orchestrate complex integration scenarios involving extensions and other cloud services based on ABAP. This integration helps ensure smooth data exchange and interoperability across the landscape.

To summarize, both on-stack and side-by-side technologies (and those available in both environments) complement each other effectively in the transformation journey. Together, they empower organizations to extend and customize SAP S/4HANA efficiently, adopting modern ways of working and accommodating preferences, thereby meeting evolving business requirements and driving digital innovation.



The role and benefits of SAP BTP in the context of SAP S/4HANA Cloud Private Edition

SAP BTP plays an essential role in the strategy for SAP S/4HANA Cloud Private Edition. These solutions and SAP BTP are inseparable, as SAP is developing major parts of its innovations as modular apps on SAP BTP, such as SAP Green Ledger, SAP Digital Manufacturing, SAP Advanced Financial Closing, sustainability management, and next-generation industry solutions. SAP is following this strategy to strengthen its clean core approach:

- The core retains essential business processes needed across all industries (core processes).
- Differentiating processes (edge, industry, innovative, next-generation, and more) are built as modular apps on or moved to SAP BTP, establishing a modular SAP S/4HANA Cloud Private Edition.



Customers and partners should follow the same strategy and establish two-layer IT:

- **Layer 1:** A stable clean core hosts essential core processes that keep the business running and focuses on process automation to reduce costs.
- **Layer 2:** An innovation layer delivers fast-paced innovations with flexibility and modularity without disrupting the core. This layer enables differentiated processes that provide competitive advantages and drive growth. (Examples: sales-order automation, just-in-time production, customer loyalty management)

SAP BTP is an innovation layer next to a stable clean core and offers significant advantages:

- It is deeply integrated and interwoven with the core ERP and shares the underlying metadata and business model.
- Core ERP uses services from SAP BTP, such as the SAP HANA service for SAP BTP, SAP Integration Suite, SAP AI Core infrastructure, and ABAP Cloud
- The Extension Wizard launches directly in SAP S/4HANA Private Edition, allowing users to navigate to tools for extension creation on SAP BTP (see Chapter 4)—“Extend SAP S/4HANA with SAP Build”) or to the service center, which provides APIs, business APIs, and remote function calls.
- [SAP Business Accelerator Hub](#) provides a vast amount of built-in, up-to-date content, including integrations, connectors, APIs, localization, and more.
- Joule offers support specialized in SAP software development for both ABAP Cloud and CAP.

3.3 SAP's clean core extensibility model and the clean core level concept

3.3.1 The clean core level concept

SAP introduces a clean core level concept (comparable to energy labels⁵) to make clean core easier and to improve transparency regarding your extension's clean core rating.⁶ Extensions can be labelled based on four levels (A to D), with Level A being the best and Level D the worst. This can better reflect the different qualities of extensions

compared to a binary decision about whether a custom code object can be considered "clean."

SAP recommends that customers extend the SAP S/4HANA Cloud Private Edition with the always highest applicable level. Depending on your specific requirements, you might need to lower your clean core level to "B" to achieve the necessary flexibility while accepting the downgrade in the clean core level.

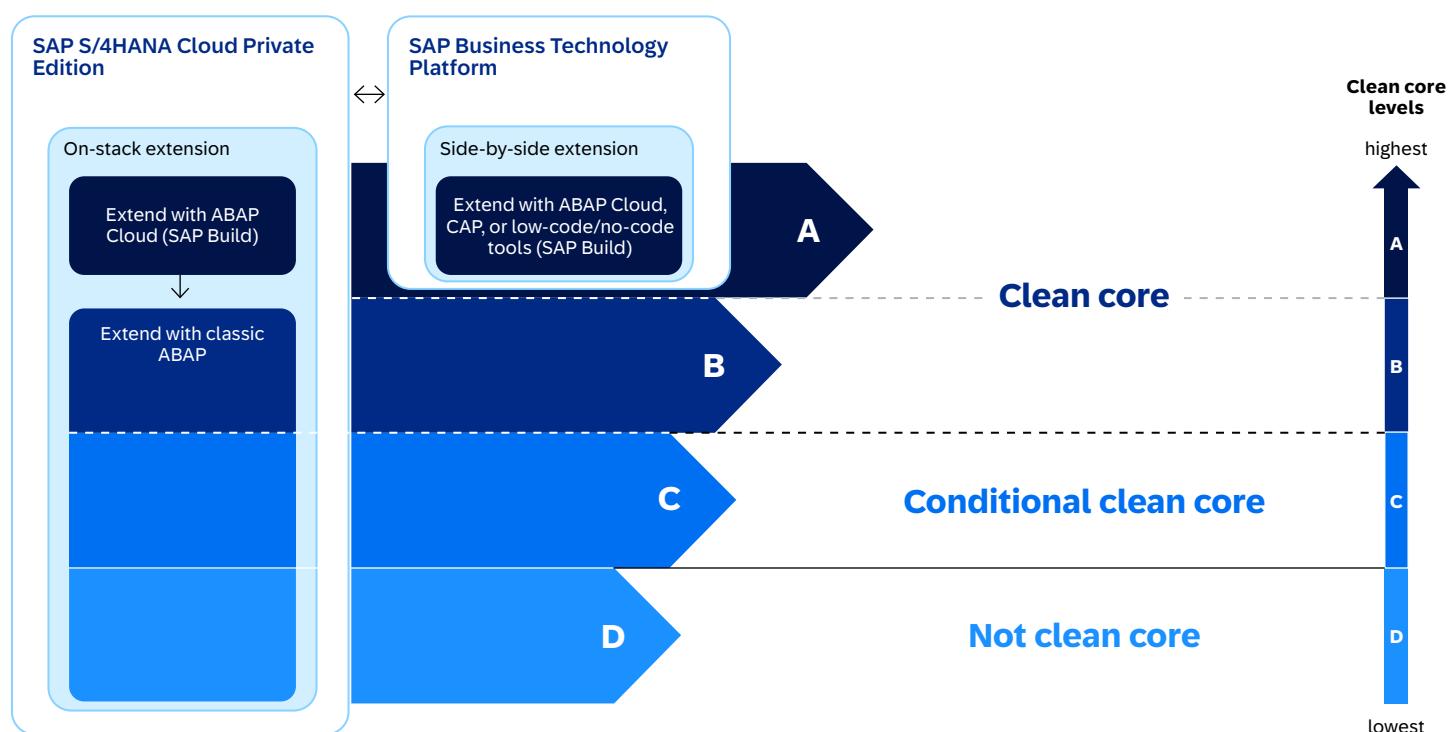


Figure 5: Clean core levels

SAP Build includes AI-enhanced pro-code and low-code tools for on-stack and side-by-side application development and process automation.

To be clear: It is (still) clearly recommended to create extensions in Level A and based on a "BTP first" strategy.

By prioritizing higher clean core levels (A over B over C), you can reduce technical debt, simplify system upgrades, improve scalability, and align with SAP best practices. This approach ensures a standardized, future-proof core system that supports seamless integration with modern technologies and minimizes maintenance challenges. At the same time, this levelled approach offers better insights, also for large-scale legacy extensions, compared to earlier concepts. This is

achieved through more transparency within the classic ABAP developments. To improve your upgrade stability, first focus on the worst extensions (Level D).

Each extension can be seen using the lowest-ranked technology inside it. For example, if two technologies are used in one extension with Level A but one with Level C, the whole extension can be seen as Level C.

If you are interested in more details on the level concept, please find below a brief overview of the levels and in addition refer to chapter 3.3.4.

3.3.2 Evolution of 3-tier extensibility to level concept

The shift from SAP's 3-tier extensibility model⁷ to the clean core level concept represents a significant advancement in the evaluation of extensibility within ERP systems from a clean core perspective. The level-based approach enhances transparency, particularly in the use of classic ABAP code, by offering a more nuanced distinction between upgrade-stable, recommended extensions and those that require timely remediation. Therefore, clean core becomes more achievable. This refined assessment framework addresses the practical need to balance the continued use of proven extensions with the long-term goals of system integrity and upgrade stability.

3.3.2.1 Why SAP evolved the model?

The clean core concept continues to resonate strongly in the market. In the context of extensibility, the principle of decoupling extensions from the SAP standard to ensure upgrade stability is widely recognized as essential. However, the prevailing clean core guidelines based on the 3-tier extensibility model have traditionally been rather stringent. These guidelines direct customers to build upgrade-safe extensions primarily via publicly released APIs, using SAP BTP for side-by-side scenarios and ABAP Cloud for on-stack scenarios, while restricting technologies and frameworks to those also available in the public cloud (and available in Tier 1).

Despite the merits of this approach, many customer landscapes contain significant portfolios of legacy custom code, and even new SAP extensions often continue to utilize classic ABAP technologies. Applying a uniform and restrictive evaluation to all such use cases fails to reflect the spectrum of upgrade risks involved. In reality, some classic usage patterns—such as implementations using ABAP List Viewer—pose minimal risk to upgrade stability and should not be classified as inherently “unclean”.

In response, SAP has evolved its approach to provide a more differentiated, transparent assessment of classic extensibility. The updated clean

core level concept acknowledges the value of both upgrade-stable legacy code and newly built extensions based on classic frameworks, while maintaining the overarching goal of decoupling extensions, ideally through SAP-released APIs.

3.3.2.2 What has changed?

With the clean core level concept, SAP no longer categorizes all classic ABAP objects as high-risk or incompatible with upgrade-stable systems. The new classification introduces the following improvements:⁸

- **[Classic APIs]:** Customers may now use selected classic APIs that, while not built on the latest frameworks, are recognized as upgrade-stable. This pragmatic stance acknowledges proven, stable technologies without forcing unnecessary code rewrites.
- **[Not recommended objects]:** SAP now clearly flags objects known to introduce upgrade challenges, helping customers avoid risky extensions early in their projects.
- **[Internal SAP objects]:** A changelog-based approach enables organizations to assess the upgrade risk of internal SAP objects in advance, supporting informed decision-making before major upgrade projects.

3.3.2.3 What has not changed?

The clean core level concept continues to uphold extensions built via SAP BTP and ABAP Cloud (both SAP Build) as the gold standard—Level A—representing the cleanest, most future-ready approach. To support the idea of decoupling, SAP recommends a BTP-first strategy for extensions. While the model now accommodates certain classic APIs, the core guidance remains unchanged: always strive for the highest level of extensibility available.

In summary, the shift from the 3-tier extensibility model to the more flexible, risk-focused level concept empowers organizations to embrace innovation without jeopardizing the core system's integrity. It recognizes both legacy realities and modern best practices, helping customers modernize at their own pace while safeguarding upgrade paths and operational excellence.

3.3.3 Overview of clean core levels

This chapter provides an overview of the clean core levels. For more detailed information on each level, refer to Chapter 3.3.4.

Level A—Extend with SAP Build

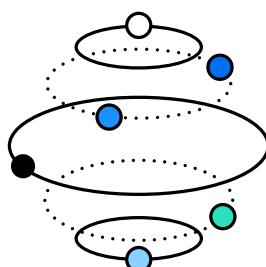
Extensions in the highest Level A are only allowed to interact with SAP standard functionality through publicly released interfaces and extension points. These interfaces come with a stability contract⁹ that ensures the stability of the interfaces. Level A extensions typically follow the most current extension models and can be implemented across two different extension domains:

- Side-by-side: Using the full capabilities of the SAP Build portfolio on SAP BTP—including AI-enhanced pro-code and low-code tools for application development and process automation.
- On-stack: Within the core ERP system by following the ABAP Cloud development model as part of the SAP Build portfolio based on released local APIs (CDS Views, business object interfaces, extension points).

Level B—Leverage classic APIs

Level B extensions use “classic APIs” from SAP and leverage classic technologies and frameworks.

These classic APIs and frameworks provide upgrade-stable, well-defined and documented touchpoints for extensions (e.g., BAPIs) and can be found in the [Cloudification Repository](#). The upgrade stability is not based on a stability contract, but on nominations of the respective APIs by SAP product experts.



Level C—Conditionally clean development in classic ABAP using “internal objects”

Extensions at Level C go beyond Levels A and B by also accessing internal objects from SAP.

SAP technically enables customers access to these SAP-internal objects, but they are not declared as “released,” “recommended,” or “cloud-ready,” nor are they officially released or supported. There is no guaranteed documentation of the internal SAP objects, and SAP does not guarantee long-term stability or endorse their use.

While SAP does not recommend the use of internal objects, it is highly common, especially in legacy code. To reflect this reality, SAP introduced a “Changelog for SAP Objects” to offer a mitigation path for stability risks.¹⁰

This changelog for SAP objects allows customers to check whether internal objects used in their custom developments have changed incompatibly in newer releases, enabling proactive upgrade planning.

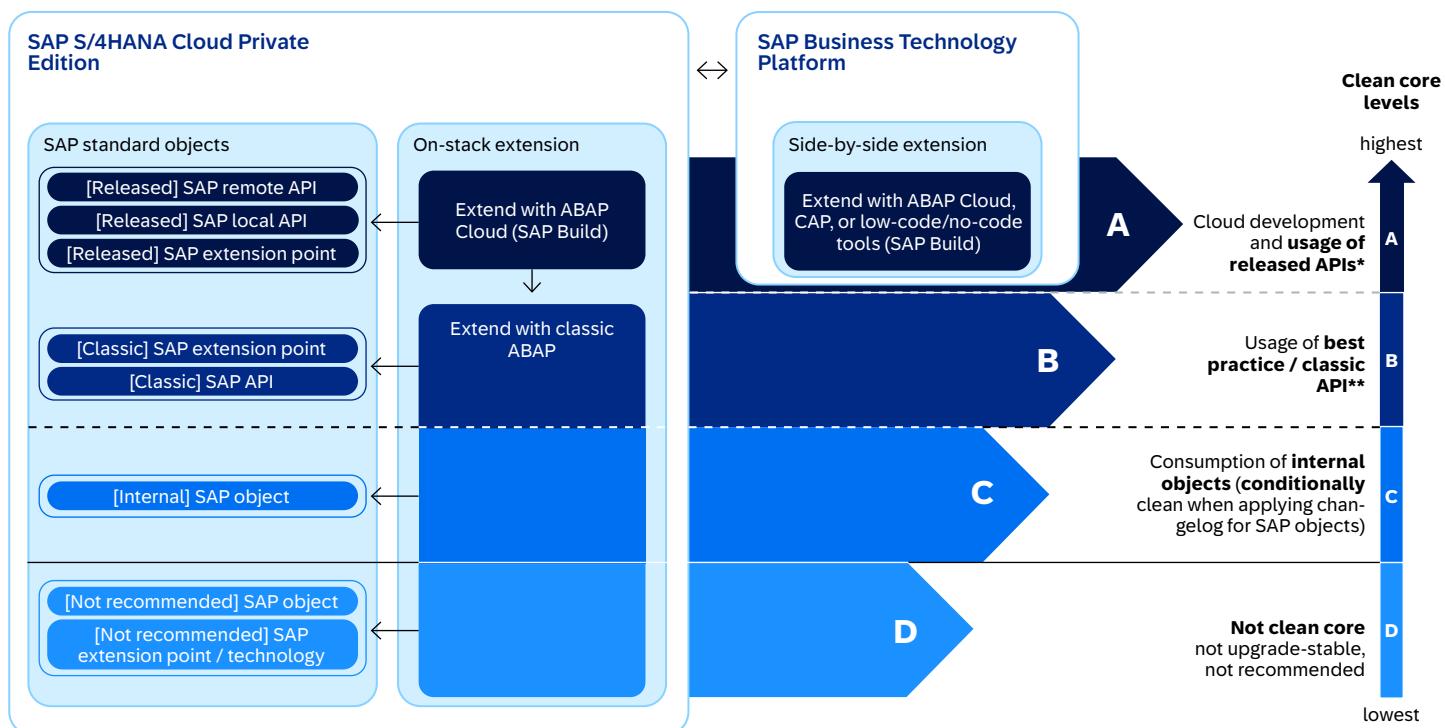
Although the changelog for SAP objects helps mitigate upgrade risks, SAP considers internal object usage to be only conditionally clean. In case internal objects have a released successor available, they are mentioned in the [Cloudification Repository](#) (“internalAPI”).

Level D—Use of “not recommended” extension options

Extensions at Level D use objects from SAP that are classified as “not recommended”.

This includes SAP objects explicitly classified as unfit for customer use (“noAPI” in the [Cloudification Repository](#)). Furthermore, it includes modifications, selected usage patterns for individual object types (e.g., write access to SAP tables), and selected extension techniques and development patterns (e.g., implicit enhancements).

3.3.4 Why, what and how—Understanding the new clean core levels



SAP Build includes AI-enhanced pro-code and low-code tools for on-stack and side-by-side application development and process automation.

* available on SAP Business Accelerator Hub

** available in Cloudification Repository Viewer

Figure 6: Clean core levels mapped to SAP standard objects

The following chapters will describe in detail what extension technologies are considered clean core by SAP, and what can be expected from extensions assigned to a given level.

As illustrated in Figure 6, each level is associated with a qualifier indicating the types of SAP objects that may be used. The table below indicates how the ABAP test cockpit supports the clean core level concept:

Clean core level	SAP objects qualifier	ABAP test cockpit behavior ¹¹
Level A	[Released] SAP remote APIs, SAP local APIs and SAP extension points	<ul style="list-style-type: none"> • No finding
Level B	[Classic] SAP extension points and SAP APIs	<ul style="list-style-type: none"> • Priority 3 finding (information message)
Level C	[Internal] SAP objects	<ul style="list-style-type: none"> • Priority 2 finding (warning message)
Level D	[not recommended] SAP objects and extension point / technology	<ul style="list-style-type: none"> • Priority 1 finding (error message)

3.3.4.1 Level A—Creating clean extensions based on released APIs

3.3.4.1.1 What to expect from Level A extensions?

Level A represents the highest standard in SAP extensibility, offering customers future-ready solutions with maximum upgrade stability. By exclusively utilizing released APIs and extension points, as well as modern technologies, frameworks, and development patterns, customers ensure their extensions are fully aligned with SAP's clean core vision. Released objects at this level are officially supported by SAP, well-documented in the SAP Business Accelerator Hub, and governed by a transparent lifecycle. However, it is important to note that Level A coverage does not extend to the complete scope of SAP S/4HANA Cloud Private Edition, and full coverage is not planned for the future.

3.3.4.1.2 Which customer extensions are Level A?

Customer extensions qualify for Level A when they are built solely on released SAP APIs and extension points. This applies to both side-by-side implementations on SAP BTP (SAP Build for both CAP and ABAP Cloud) and on-stack development directly within the ERP core. For side-by-side developments on SAP BTP, only extensions that exclusively use released APIs are eligible for Level A.¹² On-stack extensions must be built using key user or developer extensibility tools using the relevant ABAP language versions.

3.3.4.1.3 What does Level A include on the SAP side?

Level A comprises extensions that rely on SAP objects and extension points that are officially released and governed under clearly defined stability contracts.¹³ Released interfaces and extension points can be discovered in the [SAP Business Accelerator Hub](#), via the [Cloudification Repository](#), and directly in the [system](#).

3.3.4.1.4 Special view: Level A extensions and expectations on SAP S/4HANA Cloud Private Edition

Level A's strict reliance on released content and modern extension technologies aligns natively with the development approach for SAP S/4HANA Public Edition. However, due to differences in scope and codebase, on-stack extensions created for SAP S/4HANA Cloud Private Edition are generally not portable to the SAP S/4HANA Public Edition without adaptations.

In contrast, side-by-side extensions developed on SAP BTP are more likely to be compatible without bigger adaptation efforts as part of a private-to-public cloud transformation. The reason: they inherently decouple business logic from the core system. Nevertheless, complete parity is not guaranteed: API availability and application scope may differ between SAP S/4HANA Public Edition and SAP S/4HANA Cloud Private Edition versions (e.g., industry-specific solutions may have no equivalent in the public cloud).

3.3.4.2 Level B: Use of classic APIs

3.3.4.2.1 What to expect from Level B extensions?

Level B extensions address requirements that cannot be fulfilled using only the set of released APIs. At this level, customers rely on APIs and extension points nominated as "classic APIs". This means they are well-established, broadly recommended, and thoroughly documented by SAP. Although these objects are not governed by formal stability contracts, they have a longstanding history of reliable use. Level B offers a significantly broader scope for process enhancements and customizations compared to Level A, while still following SAP's best practices for upgrade stability. Extensions at this level benefit from SAP's internal quality assurance processes and represent proven, low-risk options for extending core capabilities.

3.3.4.2.2 Which customer extensions are Level B?

Customer extensions qualify for Level B when they use only SAP objects and development patterns officially classified as **classic APIs** or **released APIs**. This includes a variety of extension scenarios like:

- Wrappers around classic APIs to facilitate their use in Level A applications—for example, leveraging **BAPI_PO_CREATE1** for integration with transfer order processes.
- Classic ABAP extensions, such as ABAP List Viewer implementations within Dynpro (SAP GUI) or Web Dynpro environments, utilizing classic APIs like **CL_GUI_ALV_GRID**.
- Custom objects developed using the ABAP language version “Standard”, if they do not reference restricted or unsupported SAP objects.

Level B extensions retain a significant degree of upgrade stability by adhering to proven development practices and by utilizing extension mechanisms specifically intended for customer use.

3.3.4.2.3 What does Level B include on the SAP side?

On the SAP side, Level B encompasses objects and extension points that have been explicitly nominated by SAP experts as classic APIs, despite not being covered by formal stability contracts. These include a wide range of legacy APIs, user exits, BAdIs, and established frameworks—such as SAP GUI and the ABAP List Viewer grid—which have been deliberately exposed for customer use in both SAP S/4HANA on-premise and SAP ECC. These technologies are mature, widely adopted, and recognized for not introducing general upgrade issues when used as recommended. Where released APIs are not available, classic APIs and extension points offer the next-best extensibility option.

The key resources to identify classic APIs and the extension technologies and frameworks are the [Cloudification Repository](#) and [SAP Note 3578329](#) for the classification of frameworks.

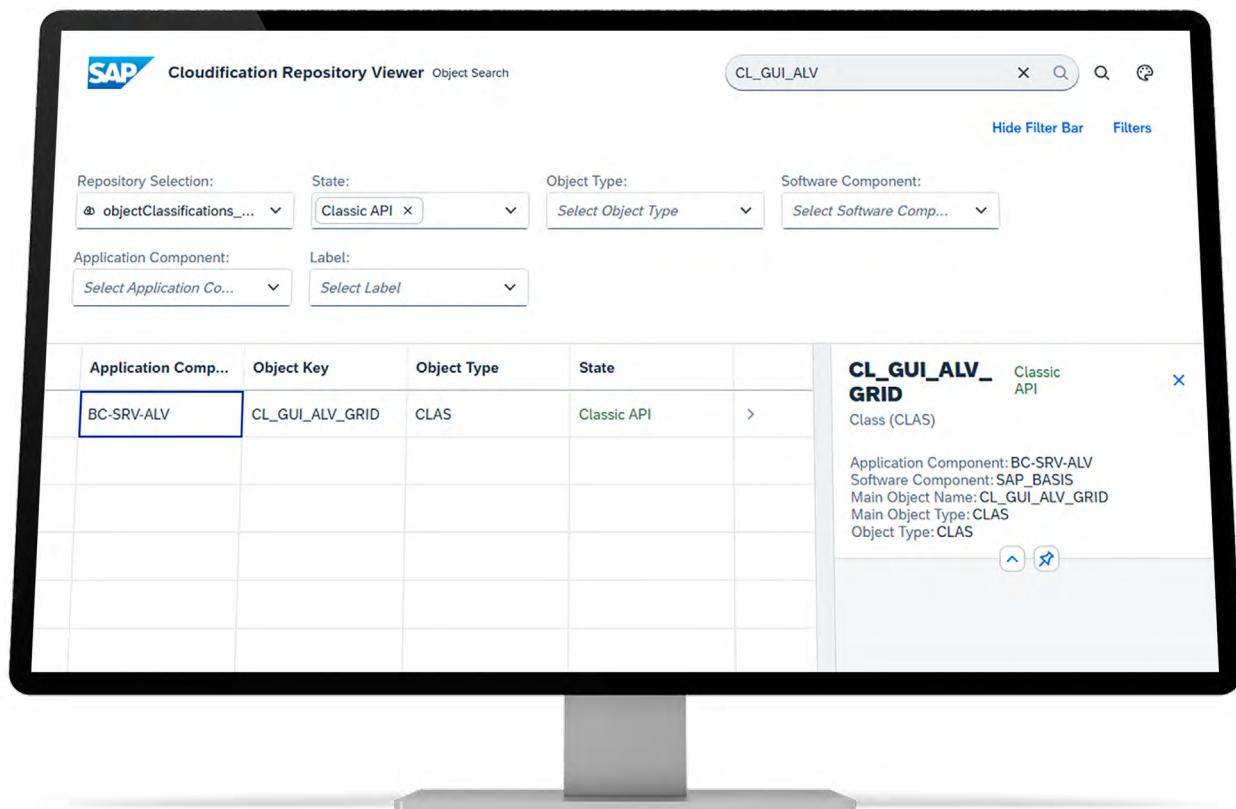


Figure 7: Searching the Cloudification Repository

3.3.4.3 Level C: Use of internal objects

3.3.4.3.1 What to expect from Level C extensions?

Level C extensions refer to scenarios in which customers use SAP internal objects that are not classified or intended for customer use. These internal objects are neither officially released, recommended, nor cloud-ready, and do not fall under SAP's formal support scope. Documentation for these objects may be incomplete or absent, and SAP provides no long-term stability or usage guarantees. While these extensions offer access to technical capabilities beyond Level A and B, they pose notable risks concerning future upgrade stability. However, with the new changelog for SAP objects, customers will be able to proactively analyze and plan for potential upgrade impacts when Level C objects are subject to incompatible changes.

3.3.4.3.2 Which customer extensions are Level C?

Customer extensions fall into Level C when they use objects or development patterns that are neither formally released (Level A) nor nominated as mature (Level B). Typical examples include the direct use of internal function modules, classes, or read access to SAP tables that have not been endorsed for external consumption. These extensions often address specific customer requirements not covered by higher-level APIs but inherently carry the risk of future incompatibility and lack the guarantees associated with recommended extension points.

3.3.4.3.3 What does Level C include on the SAP side?

Level C encompasses the broad set of SAP objects not designated as released (see Level A), nominated as classic APIs (see Level B), or explicitly classified as not recommended (see Level D).

By default, all SAP objects are considered internal and subject to change without notice or compatibility safeguards. Certain usage patterns, such as read-only access to SAP tables, may still fall under Level C, provided they do not

contravene SAP's extension policies. Importantly, SAP may reclassify internal objects as classic APIs or as not recommended objects, which would result in a level reassignment (to Level B or D, respectively). Such changes are typically driven by customer feedback, evolving product strategies, or observed upgrade challenges, underscoring the fluid nature of Level C.

3.3.4.3.4 Special view: Changelog for SAP objects approach¹⁴

To enhance transparency and support customers in managing Level C risks, SAP introduces a changelog for SAP objects.¹⁵ This mechanism will proactively identify SAP internal objects targeted for incompatible changes in upcoming releases of SAP S/4HANA Cloud Private Edition. This way, customers can plan refactoring efforts well before upgrade events to mitigate their risks associated with the usage of SAP internal objects.

Key features of the changelog for SAP objects include:

- Automated ABAP test cockpit checks to analyze custom code and detect incompatible changes and deletions of referenced SAP objects
- Early access to information about future incompatible changes affecting internal objects
- Increased planning reliability and transparency, facilitating timely resource allocation and reducing upgrade project delays

Objects listed in the changelog for SAP objects are not automatically demoted to “not recommended”. Their continued usage may still be allowed, depending on the scope and criticality of the changes. Nevertheless, changelog findings should encourage developers to refactor extensions toward released or classic APIs, where feasible. By adopting the changelog for SAP objects as part of their clean core journey, customers can significantly minimize upgrade risks, achieve proactive change management, and ensure smoother transitions across SAP software versions.

3.3.4.4 Level D: Not clean core

3.3.4.4.1 What to expect from Level D extensions?

Level D represents the most critical risk category within the clean core level concept and is associated with the highest degree of technical debt. Extensions in this category expose customers to multiple risks and significant maintenance efforts, both during upgrades and in day-to-day operations. They may compromise system stability, data integrity, and business agility, potentially hindering innovation due to ongoing compatibility challenges. As such, Level D extensions should be prioritized for removal or rework without delay. Immediate remediation is strongly recommended to realign with SAP's support guidelines and ensure long-term operational reliability.

3.3.4.4.2 Which customer extensions are Level D?

All customer extensions that rely on SAP objects or development patterns explicitly marked as **not recommended** by SAP fall into Level D. Typical examples include modifications, extensions, including unsupported write operations on SAP tables, implicit enhancements, or any attempts to utilize objects designated as out-of-scope for customer consumption.

3.3.4.4.3 What does Level D include on the SAP side?

On the SAP side, Level D encompasses all objects and patterns that have been officially declared as **not recommended** for external use. This includes objects labeled with the “noAPI” designation, which are easily identifiable in the [Cloudification Repository](#) by filtering for the “state” set to “noAPI”. In addition, Level D covers modifications, certain object usage patterns (such as direct write access to SAP core tables), and discouraged development techniques like implicit enhancements. The comprehensive list of discouraged objects and patterns provides customers with the necessary transparency to identify and systematically eliminate these high-risk elements from their SAP landscapes.

3.4 Partner add-ons and clean core

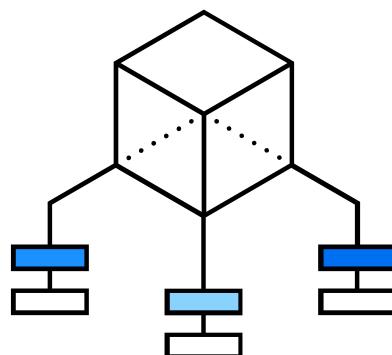
Partner add-ons are integral to SAP's clean core strategy, providing specialized expertise that enhances system architecture while minimizing customizations. These solutions are developed by leveraging SAP-endorsed technologies, ensuring that they align with the clean core principles and contribute to maintaining a streamlined system environment. By incorporating partner solutions, SAP can extend functionality without disrupting core processes. The [SAP Integration and Certification Center \(SAP ICC\)](#) offers an open certification program for partner extensions that comply with SAP's clean core extensibility criteria, as outlined in the blog “[Certification of Partner Solutions following Clean Core](#)”. Through this program, partners can demonstrate clean core compatibility to SAP customers and obtain the official designation: SAP-certified for clean core with SAP S/4HANA Cloud.

3.4.1 SAP Solution Extensions

Existing SAP Solution Extensions with add-ons on SAP S/4HANA Cloud Private Edition have been validated by SAP in collaboration with the respective partners. SAP has conducted a thorough review and confirmed that these solutions meet the required clean core compliance criteria.

3.4.2 Clean core APIs from partners

Partners can offer clean core APIs to be used by customers. Level A APIs will be offered as part of the corresponding extension. Classic APIs (Level B) can be offered in responsibility of the partner as part of the [Cloudification Repository](#).



4 How to achieve a clean core for extensibility?

4.1 RISE with SAP Methodology

RISE with SAP is a comprehensive journey to help on-premises installed base customers to modernize and move to running ERP systems in the cloud. The journey is anchored in the RISE with SAP Methodology—proven approach from SAP that bundles together several elements to support customers in their digital transformation journey. The RISE with SAP Methodology is designed to provide a holistic approach to cloud transformation, replacing legacy systems with SAP Business Suite, reducing complexity and facilitating a

smoother transition for businesses of all sizes. It helps customers modernize their business processes based on clean core principles to enable continuous innovation.

The methodology provides structured guidance and recommended steps throughout a series of defined phases, based on the established SAP Activate methodology. The RISE with SAP Methodology is built on a clean core strategy, with all services and tools working together throughout the customer lifecycle.



Existing landscape

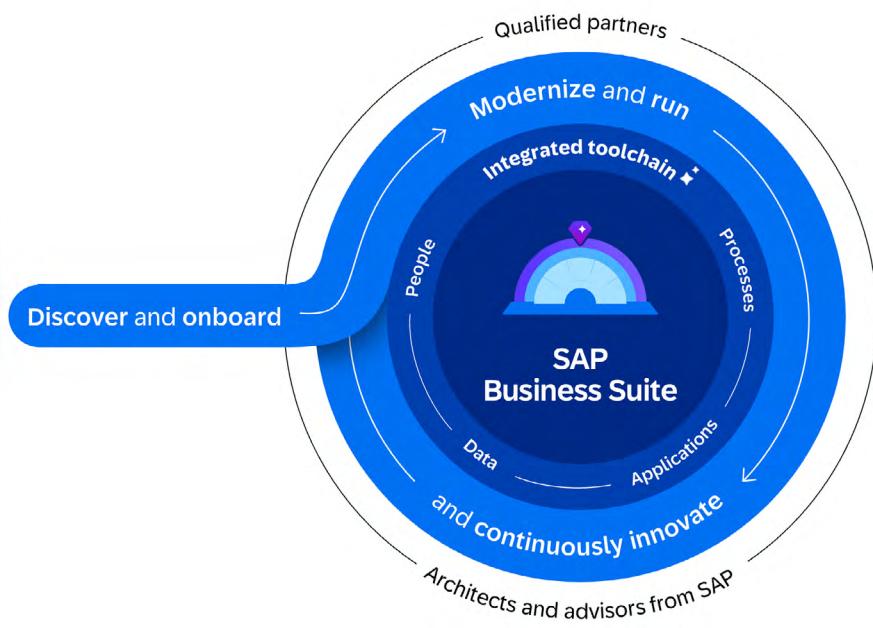


Figure 8: RISE with SAP Methodology

The RISE with SAP Methodology follows a standardized framework, helping companies adopt SAP Business Suite applications faster with a tailored transformation roadmap that meets their business needs. It leverages an integrated toolchain for streamlined collaboration and efficient

project execution, thus ensuring a cohesive, efficient workflow throughout. Finally, the RISE with SAP Methodology is guided by experts from SAP and qualified partners through an end-to-end engagement model.

With its various elements, the RISE with SAP Methodology enables customers to modernize their business processes based on clean core principles, facilitating continuous innovation and leveraging the clean core measurement framework (Figure 8). This is designed to help customers systematically progress toward a clean core by integrating governance and maturity assessments with actionable insights and continuous performance tracking through KPIs, all seamlessly integrated and brought to life using the RISE with SAP Methodology. At its heart, this framework combines structured evaluation of governance capabilities with quantitative measurement of clean core adherence, supporting informed decision-making and sustained improvement.

4.1.1 Governance and maturity— An approach to assessing clean core governance

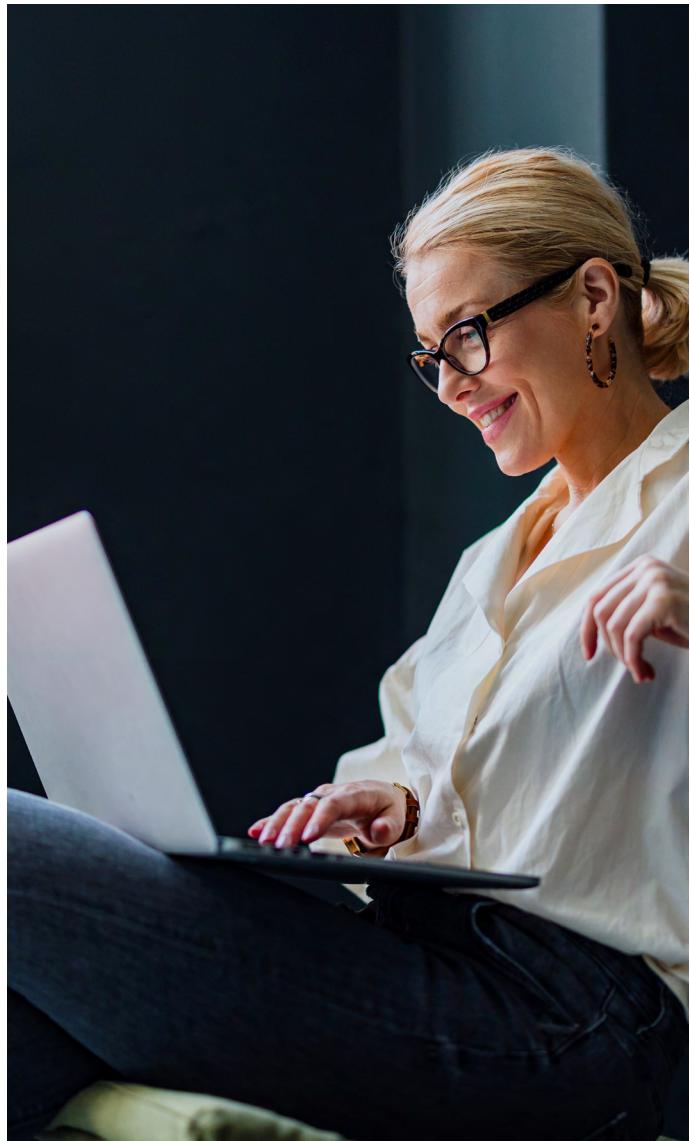
The governance and maturity approach was developed to critically assess and strengthen the governance processes within organizations, aligning them with clean core best practices. As part of the RISE with SAP Methodology, it works in alignment with the established quality gates to provide a deeper, practice-focused view of clean core governance.

While this approach covers all five clean core principles, this whitepaper focuses on **Extensibility**. For that, the assessment framework is built around 12 key practices: 7 practices evaluate governance maturity, 5 address the underlying system setup required to support clean core goals effectively.

Each practice represents a distinct governance domain crucial to maintaining a clean and sustainable extensibility landscape. Maturity is measured through detailed evaluation criteria—specific questions aligned to each practice—scored on a maturity scale from 0 (not started) to 5 (expert level). This scoring enables organizations to benchmark their current state against their target maturity levels.

By identifying the gap between current and desired scores per practice, companies can pinpoint areas requiring improvement. The resulting insights help define targeted actions, which can then be prioritized and scheduled in a structured roadmap. This ensures that governance enhancements are both practical and achievable within a defined timeline, driving continuous progress towards a robust clean core extensibility framework.

Templates for conducting the governance and maturity assessment are available [here](#).



4.1.2 Clean core measurement framework

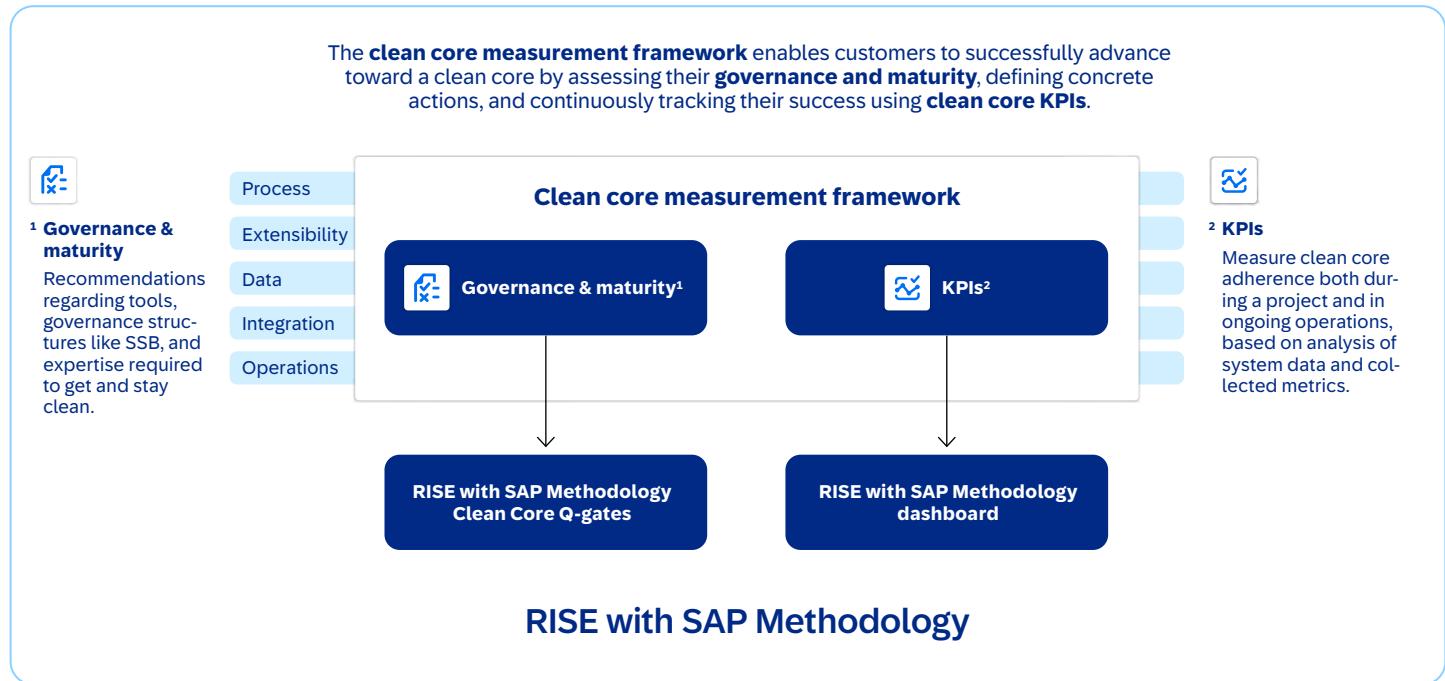


Figure 9: The clean core measurement framework at a glance

The clean core measurement framework (figure 9) is designed to help customers systematically progress toward a clean core by integrating governance and maturity assessments with actionable insights and continuous performance tracking through KPIs. At its heart, this framework combines structured evaluation of governance capabilities with quantitative measurement of clean core adherence, supporting informed decision-making and sustained improvement.

Central to the framework is the **governance and maturity** approach. It provides tailored recommendations on governance structures, tools, and expertise needed to establish and maintain a clean core. This approach assesses an organization's current maturity level across defined practices for each clean core principle, including extensibility, and allows organizations to set clear target scores aligned with strategic ambitions. The governance assessment is flexible and can be conducted at any phase of an SAP transformation journey—before, during, or after—and integrates with the established **RISE with SAP**

Methodology clean core quality gates for seamless process alignment. For a detailed explanation, refer to the chapter Governance and maturity—An approach to assessing clean core governance.

Complementing governance assessment, the framework incorporates **clean core KPIs** to continuously track adherence throughout both projects and ongoing operations. These KPIs are derived from system data and enable objective measurement of clean core compliance in real-time. Automated KPI reporting is embedded into the **RISE with SAP Methodology dashboard**, facilitating transparent monitoring and early identification of improvement areas. For an in-depth look at KPI management, see the chapter on Clean core KPIs.

Together, the governance and maturity approach and KPIs form a closed loop within the clean core measurement framework: governance and maturity outline what actions are needed, while KPIs measure the effectiveness of those actions. This integrated approach empowers organizations to

define concrete improvement plans, prioritize efforts, and progressively evolve SAP systems toward a sustainable, agile, and low-debt clean core.

To illustrate, think of maintaining physical fitness: governance and maturity reflect adopting healthy habits—nutrition, exercise, and mental wellness—while KPIs represent measurable outcomes such as hydration levels, workout frequency, and meditation duration. Similarly, clean core governance establishes best practices, and KPIs quantify adherence. Together, they enable continuous progress.

4.1.3 Measurability: Extensibility KPIs

The KPIs measure clean core adherence during a project and ongoing operations. They are derived from data collected and analyzed from a customer's system. SAP's default way to measure and visualize clean core-related KPIs is the RISE with SAP Methodology dashboard (see chapter 4.4.1), which is constantly evolving. In addition, we will also explain how the KPIs can be derived and what tools can be leveraged to gain insights for different stakeholder levels.

Below table provides a high-level overview of the extensibility-related KPIs.

KPI	Description	Actionable insights	Key benefits
Clean core share	Classification of custom code objects based on their “worst” reference level according to the clean core level concept	Highlights where improvement is needed, e.g., by decreasing the share of level D objects	<ul style="list-style-type: none"> • Visualize custom code distribution • Identify areas needing immediate attention • Track improvement over time • Measure progress • Support strategic decision-making • Ensure long-term maintainability
Technical debt score	Score of development objects to assess the associated technical debt. Allows for aggregations based on various criteria such as development packages	Identifies objects with the greatest technical debt	<ul style="list-style-type: none"> • Clear and objective measure of the factors influencing a system's maintainability • Determine distance of each object from the ideal clean core level A
Unused code share	Indicates the portion of custom objects that are not utilized	Highlights areas for potential cleanup and optimization	<ul style="list-style-type: none"> • Maintain a lean and efficient system architecture
Business modifications	Number of business modifications in a system	Identify the extent of business modifications, which constitute risk and effort for upgrades	<ul style="list-style-type: none"> • Identify need for cleaner extension practices

In the following, the purpose, measurement method and relevant data sources are explained for each KPI.

4.1.3.1 Clean core share

4.1.3.1.1 Why measure clean core share?

This KPI offers a comprehensive overview of how custom code objects are distributed across the different clean core levels. It helps identify areas that may require immediate attention and where improvements can be expected, for example, a reduction in the proportion of Level D objects.

4.1.3.1.2 What are the insights of clean core share?

Clean core share classifies each custom code object based on its worst reference level according to the clean core level concept. This indicates how closely new developments adhere to the cleanest approach (Level A) or how much they deviate from the approved levels.

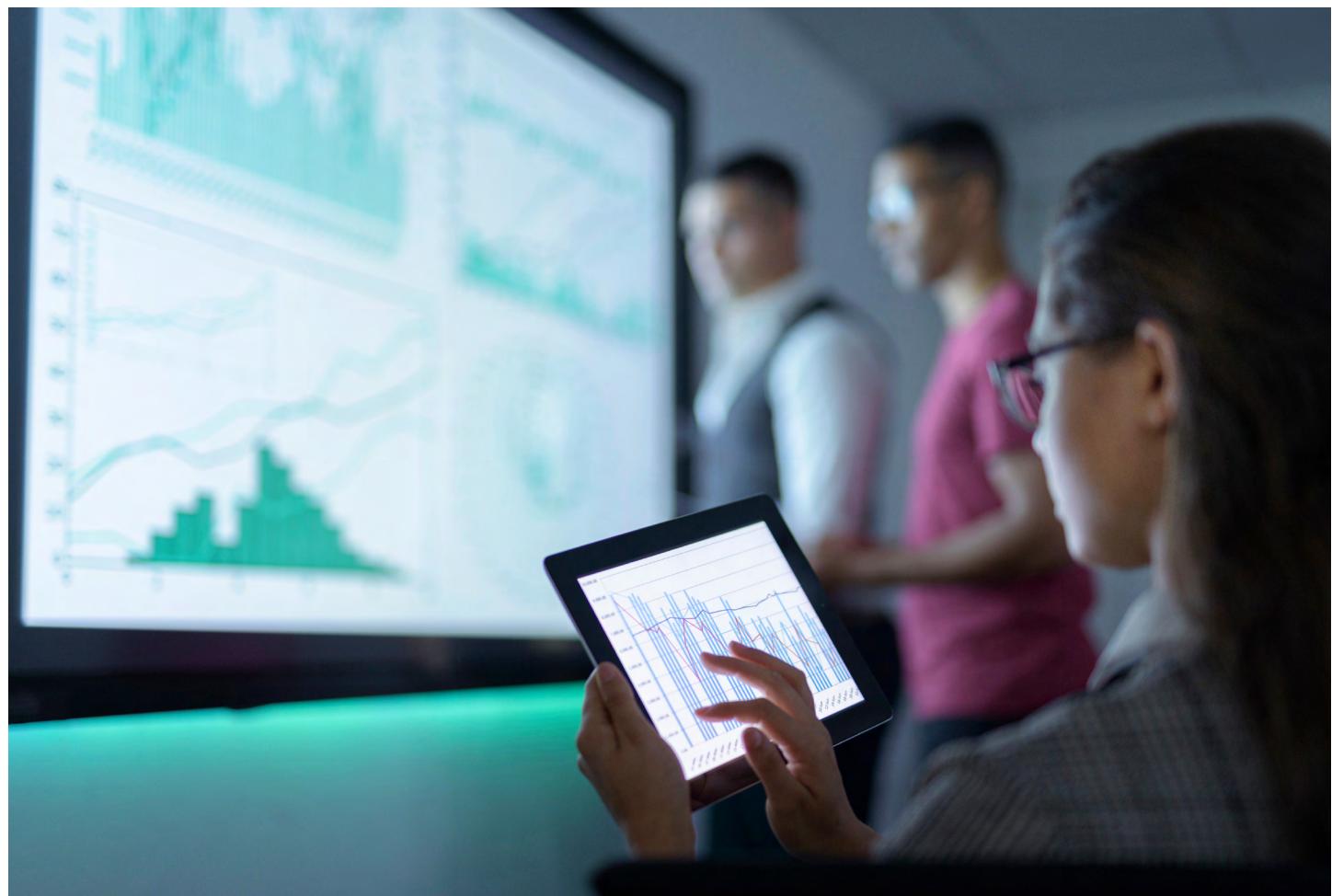
4.1.3.1.3 How can you measure clean core share?

Clean core share is calculated by classifying each object according to the level of its worst/lowest reference. For instance, if referencing an internal object, it is categorized as Level C, while ABAP Cloud objects are categorized as Level A. The share is calculated as the distribution of objects across the levels, e.g., Level A: 10% (100 objects), Level B: 50% (500 objects), Level C: 35% (350 objects), Level D: 5% (50 objects).

4.1.3.2 Technical debt score

4.1.3.2.1 Why measure technical debt score?

While clean core share offers a general overview, the technical debt score is more actionable. It quantifies the degree of technical debt in each custom code object, providing a clear and objective measure of the factors influencing a system's maintainability.



4.1.3.2.2 What are the insights of technical debt score?

The technical debt score is calculated per object and can be aggregated by various criteria such as development packages. It highlights where the largest technical debt resides within your extensions by measuring the deviation of each object from the ideal clean core level A. When combined with the size of the extension object, the technical debt score helps determine the relative magnitude of the debt.

4.1.3.2.3 How can you measure technical debt score?

Technical debt score is a weighted score based on the severity of references to SAP objects. Weighting factors are assigned to findings in ABAP test cockpit¹⁶ as follows: 10 points per error, 5 points per warning, and 1 point per information message. The score is the sum of these weighted values. Example: 5 errors, 20 warnings, and 10 information messages result in a score of $5 * 10 + 20 * 5 + 10 * 1 = 160$ points.

4.1.3.3 Unused code share

4.1.3.3.1 Why measure unused code share?

Every custom code object represents potential technical debt. Identifying and removing unused code objects is essential for maintaining a clean and efficient system.

4.1.3.3.2 What are the insights of unused code share?

Custom objects are customer-created extensions outside the SAP standard. A key clean core goal is to minimize the number of unused custom objects. Unused code share measures the portion of custom objects that are not utilized, highlighting areas for potential cleanup and optimization.

4.1.3.3.3 How can you measure the unused code share?

Measuring unused code share requires activating usage data measurements. ABAP Call Monitor (SCMON) is recommended for collecting this data, along with transaction SUSG for aggregating usage data over time.

Based on the total number of custom objects in a system, the share of unused objects can be determined. For example, if 300 out of 1,000 custom objects are not used, the unused code share is 30%).

4.1.3.4 Business modifications (available in RISE with SAP Methodology dashboard)

4.1.3.4.1 Why measure the number of business modifications?

Modifications involve changes to SAP standard code outside of extension points. As such, they generally pose upgrade risks and can lead to additional maintenance efforts.

4.1.3.4.2 What are the insights of the number of business modifications?

Business modifications are a subset of classic extensibility and are logged in the SMODILOG table. They address functionality gaps in the SAP standard but are not compliant with clean core principles and are not upgrade-stable. Measuring this KPI helps identify the extent of such modifications, highlighting the need for cleaner extension practices.

4.1.3.4.3 How can you measure the number of business modifications?

The modification analysis¹⁷ explains how to extract data from the SAP table SMODILOG using patterns to identify technical modifications that were not created intentionally as business modifications. The results can be reviewed in the custom code analytics dashboard or the RISE with SAP Methodology dashboard (see Chapter 4.4.1 for an example).

4.2 Reaching clean core with “Stay Clean” and “Get Clean”

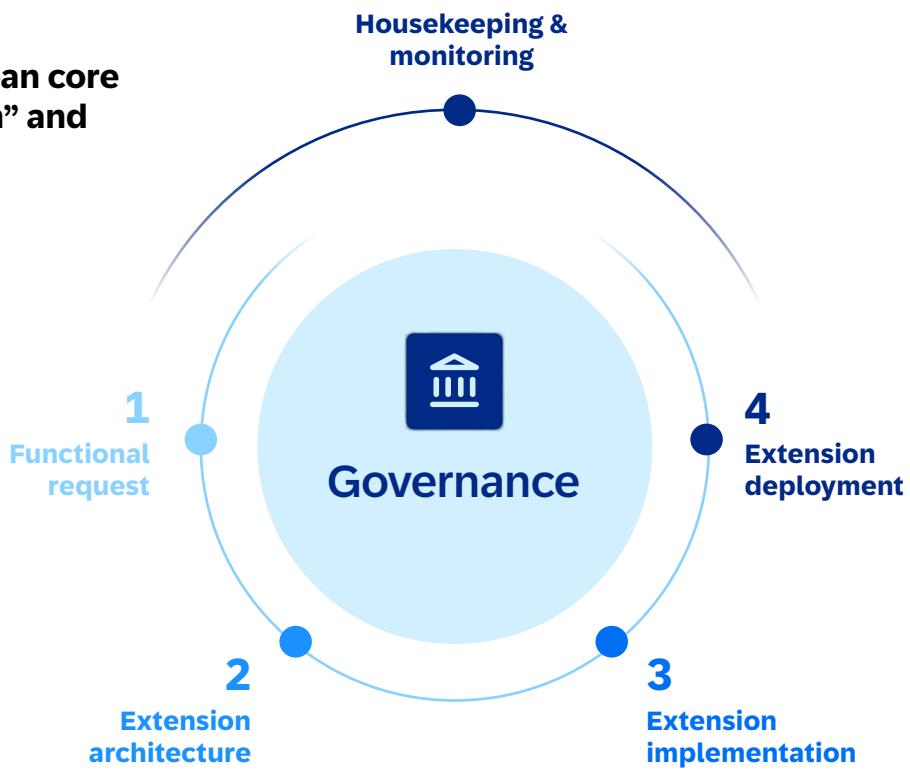


Figure 10: Elements of the Stay Clean and Get Clean patterns

Achieving a clean core in SAP extensibility requires a structured, continuous governance process that steers every extension towards minimizing technical debt while maximizing business value. Two complementary patterns guide how to implement this approach: **Stay Clean** and **Get Clean**.

Stay Clean focuses on preventing unnecessary complexity from accumulating with each new extension. It starts by rigorously assessing the **functional requirement**. Any proposed extension must have a clear business case, cannot be solved by standard SAP functionalities, and should align with a process differentiation strategy (reflecting clean core process principle). Next, the **extension architecture** step ensures that the best-fit architectural approach is selected through a governed, structured framework such as the SAP Application Extension Methodology, with proper documentation for transparency and repeatability. During **extension implementation**, developers receive focused support and controls—like authorization checks and automated quality tools such as ABAP Ttest cockpit—to ensure they adhere to architectural

standards, alongside thorough documentation and governance for any deviations. Finally, **extension deployment** ties into operational clean core principles by enforcing that only compliant and quality-checked code moves into production. Automated code validations and a tightly controlled exemption process guarantee alignment with overall governance.

Get Clean complements this preventative approach by addressing existing technical debt. It involves measuring relevant KPIs to pinpoint improvement areas, adopting the “boy scout principle” that encourages developers to leave code cleaner than they found it during any modification (factored into project estimates), and setting realistic yet ambitious goals for technical debt reduction in legacy code bases. Together, these patterns establish a holistic, iterative governance framework that sustains a clean core foundation for extensibility, balancing innovation speed with long-term system maintainability.

The following chapters will explore the steps involved to achieve a clean core based on this approach.

4.3 Stay Clean: How to create clean extensions

4.3.1 Functional requirements

Functional requirements are business needs that are not met by the standard SAP solution. They are typically identified during fit-to-standard workshops and managed within SAP Cloud ALM¹⁸, which links them directly to business processes. While functional requirements are common, non-functional aspects like data privacy, security, and scalability are also critical drivers for extension design.

4.3.1.1 Requirement gathering

One of the key objectives of fit-to-standard workshops is to ensure that requirements are collected in the proper business context. SAP Cloud ALM (which is an integral part of the RISE with SAP Methodology) streamlines the requirement collection process by offering capabilities that allow users to create requirements in the simplest possible format, within the correct business context. Identifying requirements and defining extensions is closely tied to the clean core principle of “Processes”. Requirements can be created and linked not only at the process level but also at almost every comprised diagram element, whether the process is an SAP standard process or a customer process. For additional guidance on building resilient business processes based on industry standards and evaluating the need for differentiating extensions, refer to the [Clean Core business processes](#) for SAP S/4HANA Cloud white paper. This resource also covers the fit-to-standard approach in more detail.

A requirement can be broken down into smaller user stories, but the initial focus for business

users is simply to state the need; expert consultants later determine the optimal solution, whether through configuration or extension.

When a functional gap requires an extension, it must be developed in a clean core compliant manner. This means the extension must be decoupled from core ERP code to ensure it does not break an upgrade (e.g. due to data corruption or system failure) and that an upgrade does not break the extension. The clean core extensibility model is designed to enable this separation, allowing customers to receive continuous innovation from SAP without disruption.

Crucially, an extension must provide tangible business value. An extension, even if technically sound, is considered “unclean” if it does not address a differentiating business need or if the same capability could be achieved through SAP standard functionality or a certified partner solution. Therefore, every potential extension must be subject to a well-defined governance process.

4.3.1.2 Governance: establish and adhere to clean core guidelines

To keep the core clean, the organization needs to set up a business process management practice. The primary clean core objective is to stay as close to the SAP standard as possible, ensuring any deviation provides clear, value-adding differentiation. This requires setting up a prioritized and segmented business process hierarchy and classifying business processes into standard, differentiating (enhanced and custom-developed), and innovating layers to optimize resource allocation and focus development on high-impact areas. The business process hierarchy consists of both functional, modular processes as well as end-to-end processes.

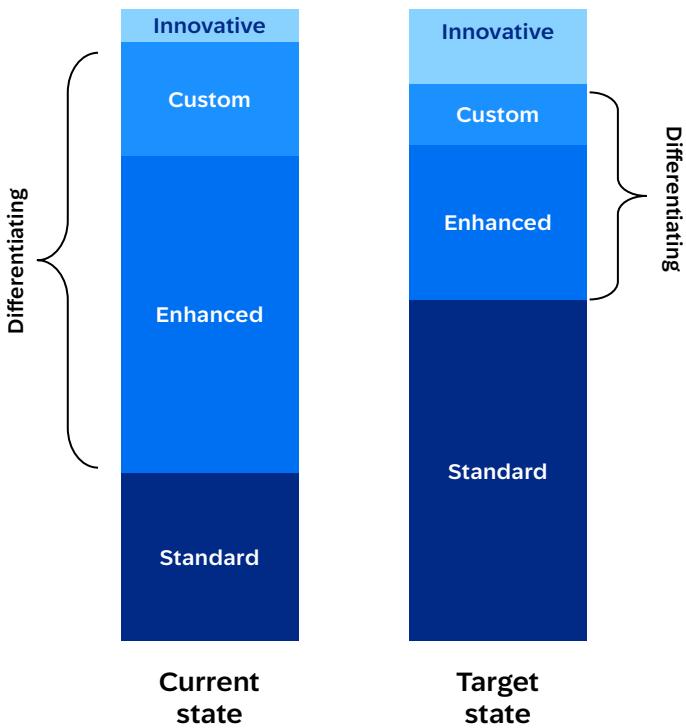
**Figure 11:** Share of process categories

Figure 11 illustrates an example comparison between the current and the target distribution of process categories, emphasizing an increase in standard and innovative processes.

Before segmenting processes, the organization should determine its required business capabilities, apply pace-layering, and define the as-is and to-be architecture that supports those capabilities. This step ensures that all SAP solutions are part of the application portfolio, avoiding process deviations where standard solutions are available. Furthermore, having identified both business capabilities and associated applications, it guides customers to the SAP process reference content, ensuring clean-core adherence while executing a fit-to-standard approach. By always linking back functional processes to business capabilities and the pace-layering, strong guidance is given on whether deviations or extensibility should be considered at all.

SAP supports the setup of an enterprise architecture and business process management with its integrated toolchain, which consists of SAP LeanIX and SAP Signavio. Another element of the toolchain is SAP Cloud ALM. Processes defined in SAP Signavio can be transferred to SAP Cloud ALM, where any delta requirements can be recorded and associated with the business processes.

In parallel, organizations should establish a governance framework to ensure that when extensions are required, they are approved only where clearly justified. The rationale and expected business value of each extension must be documented, and decisions aligned with clean-core principles to minimize complexity and long-term maintenance overhead.

Figure 12 outlines a structured decision-making process for evaluating and implementing new business requirements or gaps within an SAP system, supporting a make-or-buy decision.

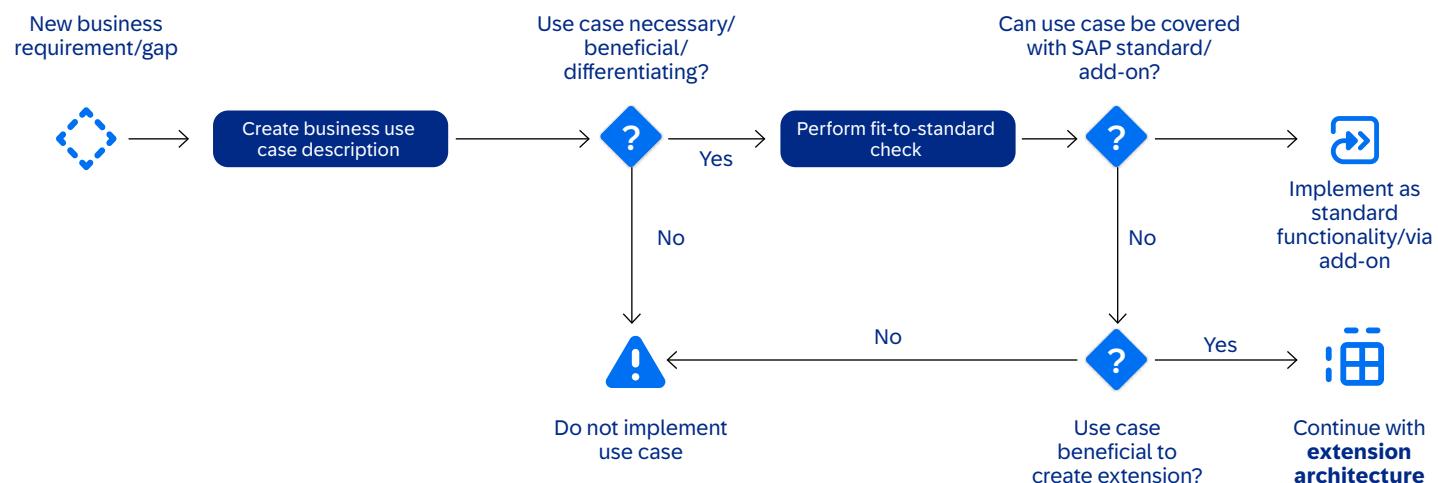


Figure 12: Handling of business requirements

If a use case cannot be covered by standard solutions and is beneficial enough to justify creating a custom extension, the process continues with the **extension architecture** approach for implementation.

Depending on the design outlined during the definition and documentation of the functional requirement, the SAP Application Extension Methodology (see [next chapter](#)) will be applied to define the extension architecture.

Centrally governing to-be processes, business requirements, and design approvals can be accomplished by establishing a central authority or team (such as a [Solution Standardization Board](#)) to oversee the design and implementation of business processes in SAP ERP, validate the necessity of an extension and ensure SAP standard options are considered first. This enables organizations to:

- Align all processes with best practices while meeting business requirements
- Enforce “clean” development according to individual needs
- Require clear documentation and justification for the selected extension domain
- Ensure that all deviations are approved in a standardized and consistent manner

4.3.1.3 Using SAP Cloud ALM to support clean core governance

SAP Cloud ALM is pivotal for establishing a clean core. By leveraging SAP Cloud ALM, organizations can adopt SAP's best practice content, manage fit-to-standard workshops, and streamline their requirements management processes. This helps limit customizations, reducing complexity and maintaining the integrity of the core system. Integrated analytics and traceability features provide a clear view of clean core compliance, enabling businesses to track and manage their projects effectively.

To classify SAP Cloud ALM entities according to the established clean core governance, tags (e.g. “clean” and “not clean”) can be created for processes, requirements, user stories, and features.

These tags can be applied on the Solution Process Traceability analytical page in SAP Cloud ALM. They allow evaluation of whether planned or implemented solution processes and the assigned requirements, user stories¹⁹ and features are classified as “clean” or “not clean”. From this view, users can also navigate to related traceability and overview pages (e.g. Requirement Traceability) to check further details.

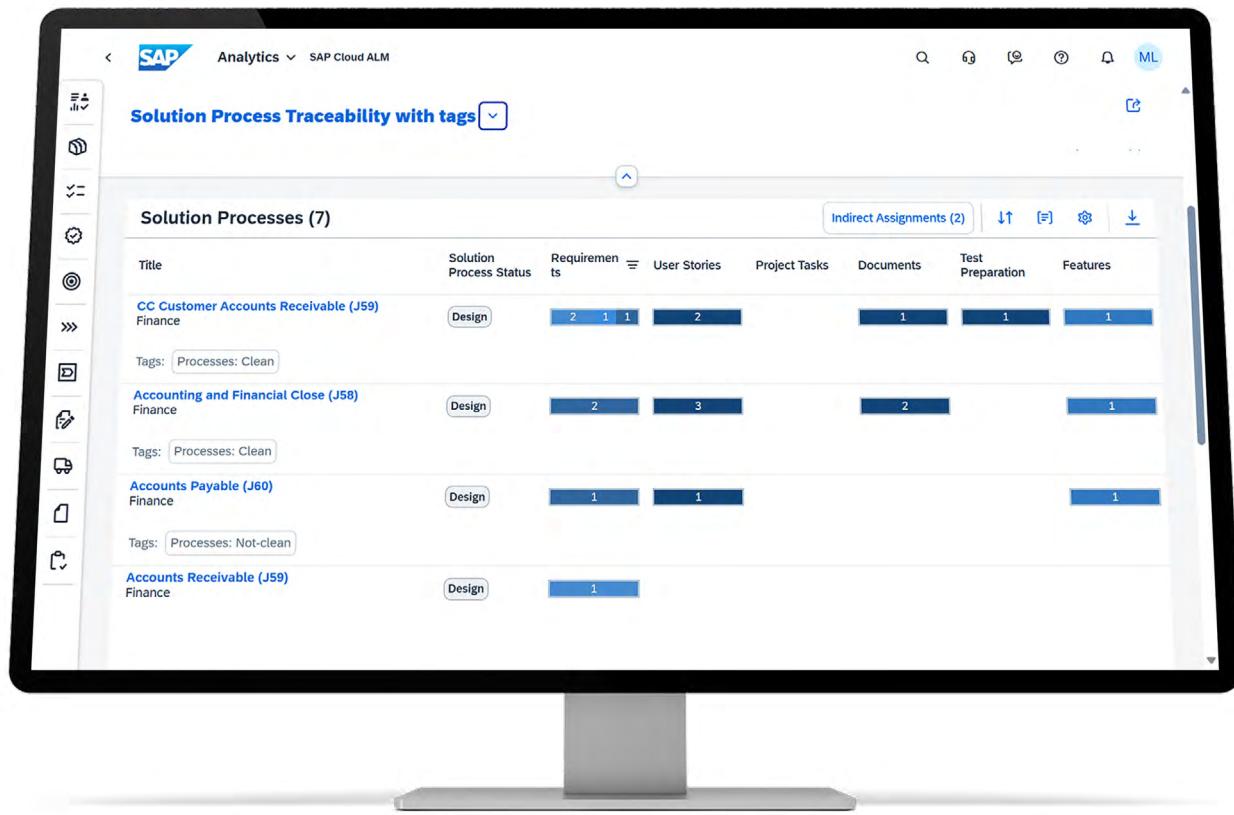


Figure 13: Solution Process Traceability in SAP Cloud ALM

When an inconsistency is identified—such as a standard requirement linked to a “not-clean” user story—the responsible team must be consulted to either adjust the scope or re-evaluate the tagging.

Following this procedure improves risk and effort estimation during the planning phase and helps ensure that project teams adhere to clean core principles.

4.3.2 Extension architecture

The SAP Application Extension Methodology offers a structured, technology-agnostic approach for customers and partners to define an organization-specific extension strategy.

Following this methodology ensures all project stakeholders use the same terminology and

quickly arrive at a common understanding of the business use case and future solution. It provides an overview of available technical extension building blocks, enabling informed decisions about the future extension architecture. It also supports the creation of an extension framework, to guide architects and developers within the organization.

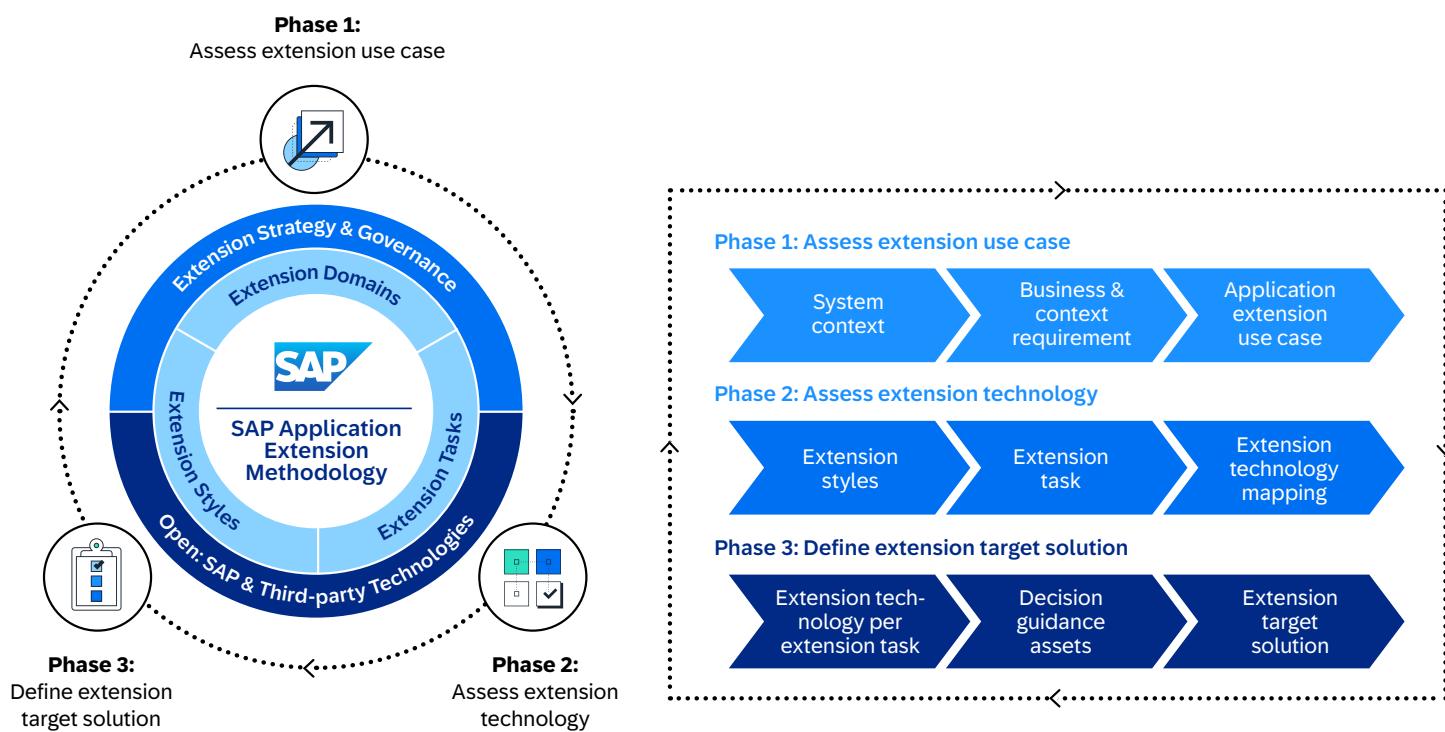


Figure 14: SAP Application Extension Methodology

Phase 1 focuses on defining the extension use case based on the business context and requirements within a defined area. It involves understanding the system context and documenting the extension application use case accordingly.

Phase 2 introduces key concepts such as extension styles, extension tasks, and extension domains. It provides an overview about various extension technologies called technical extension building blocks. In combination with the extension tasks, they help translate business needs into technical requirements.

Phase 3 is based on the overall requirements and the technology mapping between extension tasks and building blocks in the previous phase and supports informed decision-making on which technical extension building blocks to use for the target solution. Various decision guidance assets—such as the [Extension Architecture Guide](#), SAP Discovery Center, and [SAP Architecture Center](#)—offer further support in shaping and refining the target solution.

Example

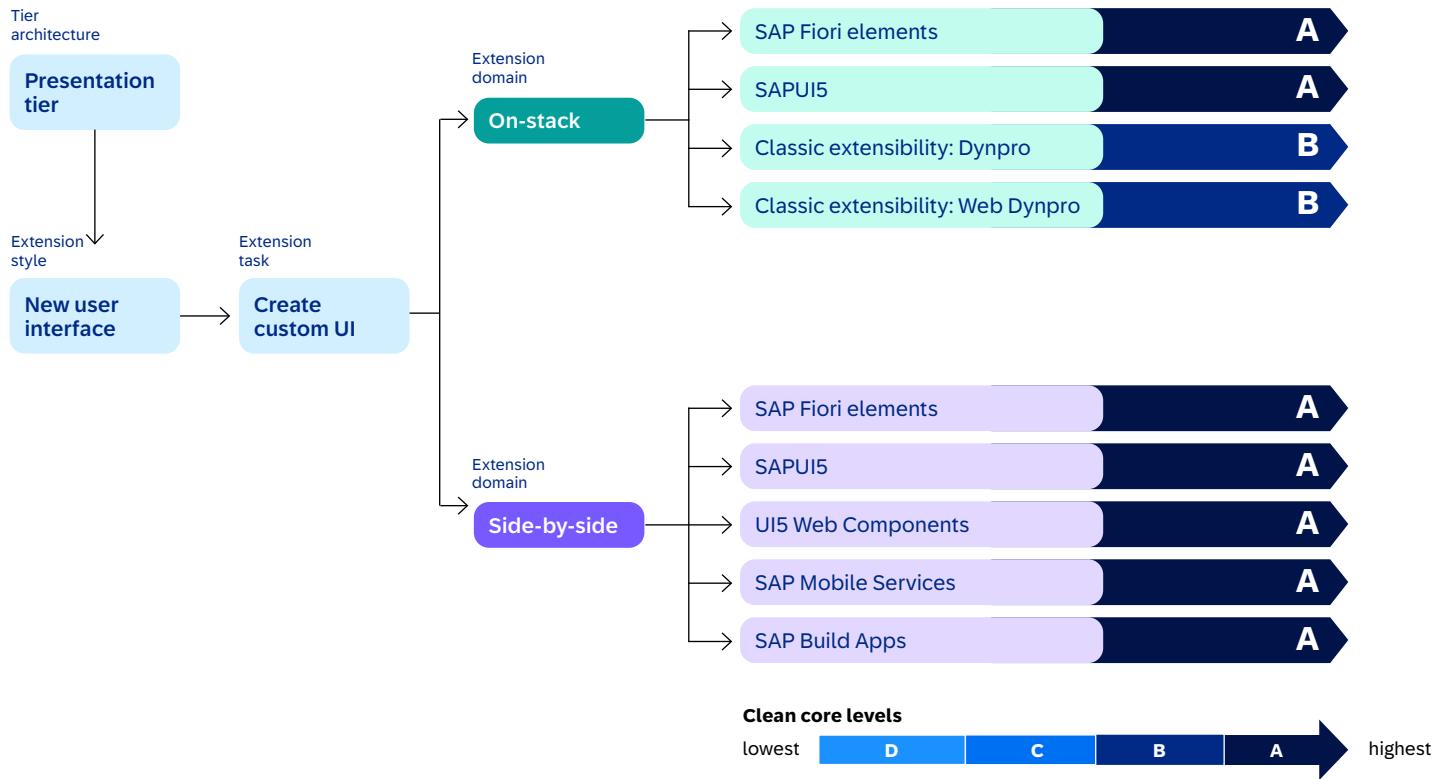


Figure 15: Decision guidance per extension task

Source of truth for clean core levels: SAP Note [3578329](#)

To perform a thorough analysis, organizations should use the Extension Task Guidance Template and the Extension Technology per Extension Task Mapping Template to document custom guidance before selecting technical extension building blocks.

This process supports the development of clear design guidance to ensure the best clean core level is considered. The SAP Application Extension Methodology templates also assist in setting sequencing and evaluation criteria for choosing the most suitable technical building blocks.

Developers may prefer familiar tools over optimal solutions, especially when they are unfamiliar with ABAP Cloud or SAP BTP capabilities. This structured approach helps guide architecture decisions.

After completing the methodology, several deliverables are produced: an application extension use case description (from phase 1), a decision on technical extension building blocks (from phase 3), and an extension target solution (from phase 3). These deliverables can be used as input for the SAP Cloud ALM user story.

For more information about the SAP Application Extension Methodology and access to templates, visit the [SAP Help Portal](#).

4.3.3 Extension implementation

Establish a clean core mindset

Good development practices involve writing code that not only addresses immediate needs but also considers the bigger picture: clean architecture, system stability, long-term maintainability, and alignment with the organization's future strategy. A practical example: creating a report based on an ABAP List Viewer grid may not cause problems during system upgrades, but it becomes technical debt when the organization plans to migrate to the SAP S/4HANA Cloud solution.

Code isn't just technical—it's strategic.

However, adopting a clean core mindset goes beyond writing code. It also involves taking the time to consider reuse, stepping back to focus on upskilling, and re-evaluating development approaches instead of rushing into implementation. A clean core mindset shapes how extensions are planned and executed.

Development guidelines

To translate the clean core mindset into daily practice, it's important to establish clear development guidelines. These guidelines help ensure consistent practices, whether it's about using the right APIs, avoiding certain enhancements, or following naming conventions and coding standards. When the rules are clear, developers can focus on building high-quality, future-proof solutions without second-guessing what's allowed or expected.

Well-defined and documented standards also make it easier to onboard new developers and keep teams aligned across different projects. These guidelines should be easy to access, regularly updated, and reinforced through tools like automated code checks and peer reviews. With such a structure in place, teams will have a solid foundation. While these guidelines might always be individual, there are good starting points to consider and refer to when it comes to development guidelines:

- Guide on "[Extend SAP S/4HANA in the cloud and on premise with ABAP-based extensions](#)"
- [Clean ABAP Guide & Cheat Sheet](#)
- [ABAP Cheat Sheets](#)

Skill development

After establishing the development rules, it is important to enable the workforce based on their roles and skills: Developers need to learn how to build modern business applications for SAP S/4HANA and evolve from classic ABAP developers to ABAP Cloud developers.

SAP Learning offers several free learning journeys, such as:

- Acquiring Core ABAP Skills²⁰
- Managing Clean Core²¹
- Practicing Clean Core²²

These paths help prepare for certification as an SAP Certified Associate—Back-End Developer—ABAP Cloud.

Additionally, lead developers and solution architects should broaden their knowledge to maintain a comprehensive view of extension development. This can be achieved through the learning journey [Becoming an SAP BTP Solution Architect](#), which also offers a certification.

This enablement process requires time and management commitment. Establishing one or more centers of excellence for relevant topics can support the rollout and help drive clean core adoption across the organization.

Developing clean extensions

At the beginning of an implementation, it is essential to identify all touchpoints with the SAP standard, such as user interfaces (applications, forms, reports), integrations (integration flows, events), business logic (business add-ins), and APIs (CDS views, business objects interfaces, BAPIs), as well as persistence (tables). This approach helps clarify all potential dependencies on the SAP standard and the core and shows how tightly coupled an extension will be.

With this information in place, development should be planned using a top-down approach, moving from clean core Level A to clean core Level D, transitioning from tightly coupled to loosely or entirely decoupled extensions.

For tightly coupled extensions, the starting point

is the [SAP S/4HANA Key User Extensibility](#) options (Level A) and the search for suitable released extension points. A released extension point may be, for example, a field extension or business logic extension within a business context. The method of identifying these released extension points can vary depending on the specific SAP S/4HANA Private Edition version.

On older releases of SAP S/4HANA Cloud Private Edition, extensibility options may need to be checked directly within the documentation of the target application. There are accessible in the SAP Fiori Apps Reference Library by searching for the application for the corresponding system version, navigating to the Implementation Information tab and locating the Extensibility section. The App Extensibility documentation linked there provides details about extensibility options, such as the business contexts used for this application. For other objects, transaction SCFD_REGISTRY can be used to find every extension point registered in a system.

When a requirement is too complex to be implemented with key user extensibility, an extension with developer extensibility (Level A) is necessary. In side-by-side or hybrid extension domains, the process typically begins with exploring remote public APIs on the SAP Business Accelerator Hub, such as OData services, REST services, or events that can be exposed from the core system. The [SAP Business Accelerator Hub](#) provides comprehensive resources for using these APIs, including business documentation and details on extension options. For on-stack extensions, [ABAP Development Tools for Eclipse can be used to identify released public APIs](#), like CDS views, business object interfaces, or business add-ins—by filtering objects based on their API status.

Note: It is essential to review the referenced business documentation for each API, as certain APIs may not be suitable—for example, if a business feature is not implemented. If applicable, check whether a predecessor classic API exists.

Where classic ABAP objects are already known, users can identify the successor object by checking the API state properties or consulting the Cloudification Repository.

The Cloudification Repository also offers a glimpse into the future of released APIs. By selecting the desired system version, it is possible to check whether an API will become available in an upcoming release. If no released local public API exists, the next step is to explore classic APIs in the Cloudification Repository (Level B).

To identify the suitable extension technology or framework, refer to [SAP Note 3578329](#), which categorizes legacy technologies into clean core levels and offers clear guidance on recommended usage.

When stepping outside the boundaries of Level A, developers should implement additional safeguards during application development. Since internal objects can be unstable, it is essential to monitor the changelog for SAP objects to receive early warnings about incompatible changes and to exercise extra caution when using such objects. To verify expected behaviour, a unit test should be created and run after each system upgrade, especially when using internal or not recommended objects.

SAP strongly advises to maximize development within Level A (ABAP Cloud) to reduce the reliance on Level B to D objects. One effective strategy is to encapsulate classic APIs, internal objects, or not recommended objects within a wrapper, which can then be exposed for ABAP Cloud using an API contract. Details are available in the ABAP Cloud API Enablement Guidelines.²³

4.3.4 Extension deployment

Once extensions are designed and implemented following the clean core approach, the next crucial step is enforcing those principles through structured governance and technical controls. Even the most disciplined development strategies can be undermined by a single unreviewed transport or an improperly managed exemption. To ensure long-term sustainability, code quality,

and system stability, it is essential to embed automated enforcement and accountability mechanisms directly both into the development and deployment processes.

Figure 16 shows the different clean core governance options, explained in the following paragraphs.

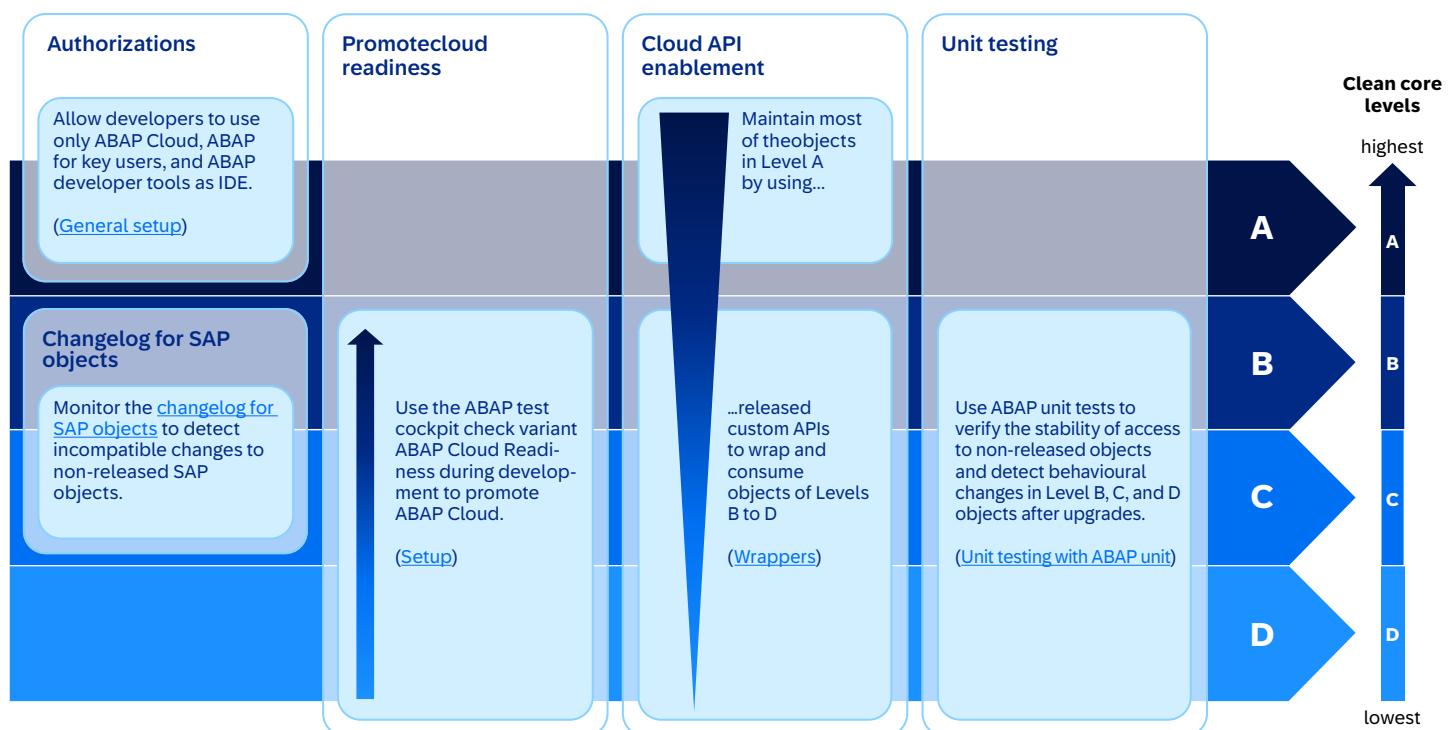


Figure 16: SAP Application Extension Methodology

For full documentation, read the document [“Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions”](#).

In non-ABAP environments, static code analysis tools (e.g., CDS Lint for CAP) are used to ensure that code follows the defined rules of coding. The core of operational clean core governance within SAP S/4HANA Cloud Private Edition systems relies on automated checks using the ABAP test cockpit. These checks should be tightly integrated into the transport release process. SAP offers pre-delivered ABAP test cockpit variants, which serve as a dependable baseline for enforcing clean core rules. Organizations are encouraged to customize these variants to align with their internal policies. Incorporating these variants into the transport release process ensures that

checks run automatically whenever developers release transports. When high-priority issues (usually priority 1 or 2) are identified, the transport should be blocked until the violations are addressed or explicitly exempted through a defined governance process. To set up governance for the classic ABAP development model, please refer to the ABAP Extensibility Guide.²⁴

However, no enforcement strategy is complete without a structured exemption process. While clean core policies should be strictly applied, legitimate exceptions may arise. These must be consciously acknowledged and governed to

prevent gradual degradation of development standards. A formal exemption procedure should be established, with a designated Quality Manager (ideally a senior developer or architect) responsible for reviewing and either approving or rejecting requests.²⁵ Every exemption request must include a clear justification, a reference to the specific finding being waived, and a list of the development objects involved. All decisions must be recorded within ABAP test cockpit to ensure full traceability and accountability. Temporal limits should be defined for each exemption, and broad or generic exemptions at the object or package level should be avoided. Periodic reviews of existing exemptions, typically covering 10–20% of cases, help verify ongoing relevance and support the continuous refinement of development guidelines.

To enforce clean core development structurally, one of the most effective measures is restricting the ABAP language version to ABAP Cloud. This restriction can be implemented by assigning the S_ABPLNGVS authorization object and/or by creating a dedicated software component²⁶ that only supports ABAP Cloud. Establishing such a component introduces a clear architectural boundary and allows for a separation of developer roles. Standard ABAP Cloud developers can be distinguished from trusted developers with elevated privileges, making it easier to manage who is permitted to work with legacy or unrestricted objects. This separation enhances transparency, simplifies maintenance, and reduces the risk of introducing non-compliant code.

In addition to managing language version restrictions, organizations should activate ABAP test cockpit checks that monitor API usage and enhancement technologies. These checks, provided in [SAP Note 3565942](#), introduce clean core levels by classifying APIs and technologies based on their stability and upgrade safety. For setup details, refer to the ABAP Extensibility Guide²⁷ in chapter “Setting up the governance for the classic ABAP development model.”

When working with Level B, C, or D objects, developers inherently use the standard ABAP language

version, which is a superset that includes both ABAP Cloud and ABAP for key users. Although using the ABAP Cloud Readiness check variant is optional, it is strongly recommended. Activating this variant helps reduce unnecessary dependencies on standard ABAP statements and promote the development of cloud-ready and upgradeable code in Level A.

To support the adoption of Level A objects and further increase their share, organizations should create encapsulated wrappers for non-released SAP objects. These wrappers can then be released for use in ABAP Cloud, effectively bridging the gap between non-compliant legacy code and modern, compliant development practices.

Automated ABAP unit tests play a vital role in maintaining code stability, particularly when dealing with classic APIs (Level B), objects of uncertain stability (Level C), or developments that lack formal APIs (Level D). While structural changes (e.g., renamed fields) can often be detected at compile time, behavioral changes, like altered logic or calculations, typically go unnoticed. ABAP unit tests act as a safeguard against such regressions, ensuring that expected code behavior is maintained even as the underlying implementation evolves.

SAP continuously publishes updates to classic APIs and internal objects in the changelog for SAP objects. By reviewing this changelog regularly, development teams can proactively identify potential breaking changes, further reinforcing system stability.

Finally, fostering a culture of transparency and continuous improvement is essential. Insights gained from ABAP test cockpit scans and exemption reviews should be shared with the wider development team. This openness builds a shared sense of responsibility for maintaining high-quality, compliant code and encourages ongoing learning and refinement. By integrating governance deeply into both tools and culture, organizations can ensure that their SAP environments remain stable, scalable, and ready for future innovation.

4.4 Get Clean: How to measure the current state of clean core?

4.4.1 RISE with SAP Methodology dashboard²⁸

As customers embark on their transformation journey, they require tools that provide visibility into their current state, the progress of key projects, and insights to guide their next steps and strategy. This is where the RISE with SAP Methodology dashboard in SAP Cloud ALM becomes essential, helping organizations keep their ERP systems agile, innovative, and optimized for continuous improvement to unlock the full value of SAP solutions.

The RISE with SAP Methodology dashboard plays a pivotal role for businesses adopting a clean core approach. By focusing on resilient processes and efficient operations, organizations can achieve seamless integration. The adoption of standard processes enhances overall consistency, while the ability to create unique extensions promotes differentiation. This approach ensures that business-critical systems remain agile, cost-effective, and ready for innovation.

The RISE with SAP Methodology dashboard supports customers on their transformation journey. SAP S/4HANA Cloud Private Edition customers can access the RISE with SAP tab on the SAP Cloud ALM launchpad, from which they can open the System View dashboard.²⁹ This dashboard provides a clear overview of the system's clean core compliance status.

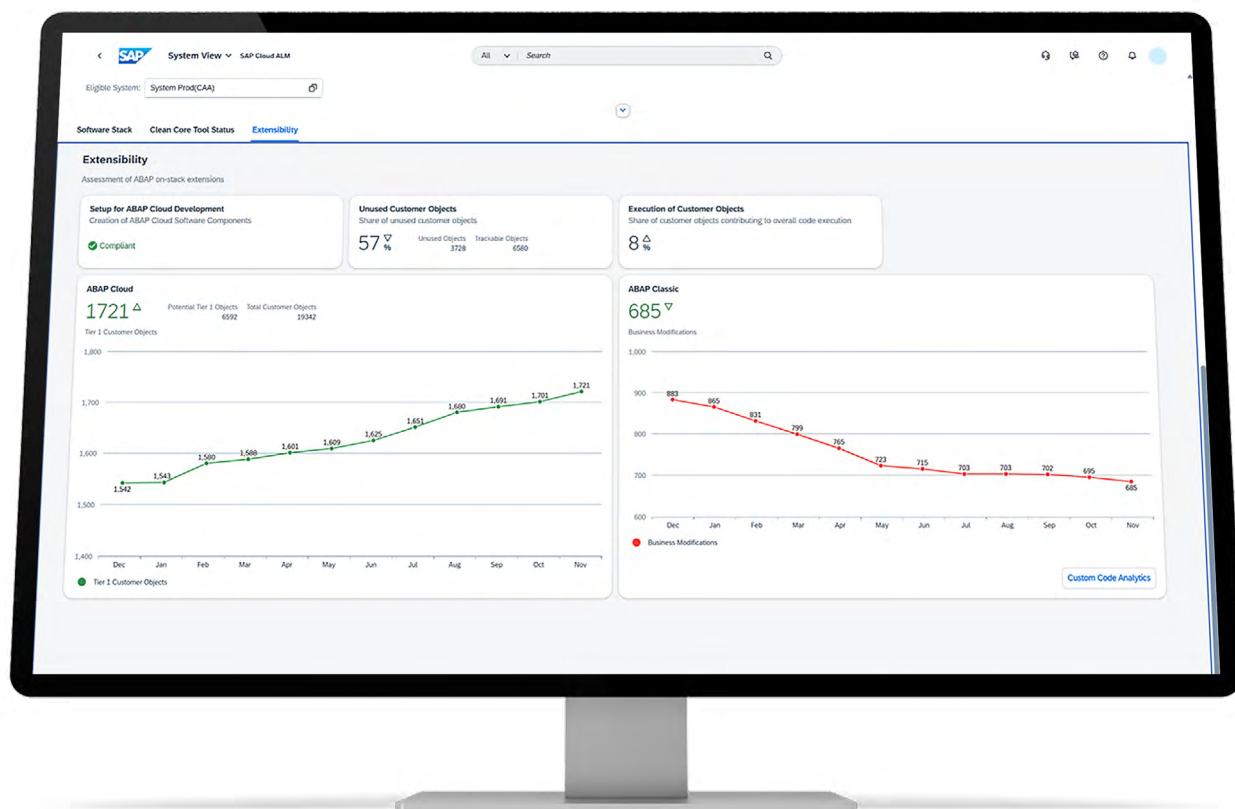


Figure 17: RISE with SAP Methodology dashboard—System View

In the System View, customers can verify whether the current SAP S/4HANA Cloud Private Edition product version installed on their system is up to date and ensure that the required tools for collecting clean core-related data are active and functioning as expected. Finally, they can review current extensions to evaluate and prepare for any potential issues in their next upgrade.

Below is a high-level summary of the System View's capabilities. The System View and its individual sections is described in detail in [SAP Help Portal](#).

In general, the System View in RISE with SAP Methodology dashboard consists of the following tabs:

- **Software Stack**

Gives you an overview of your system's software stack and provides the key results of the latest SAP Readiness Check for SAP S/4HANA Cloud Private Edition Upgrades that was run for your system.

- **Clean Core Tool Status**

This section shows the status of the tools you need in order to collect clean-core-related data. Each tool mentioned here provides data that is necessary for measuring the clean core compliance of your system.

- **Extensibility**

This section gives you an overview of different extensibility-related KPIs (see also chapter "Measurability: Extensibility KPIs"). The KPIs currently³⁰ provided in the System View of RISE with SAP Methodology dashboard are:

- Setup for ABAP Cloud Development
Shows whether you followed the recommendation to create ABAP Cloud software components.
- Unused Customer Objects (see also chapter Measurability: KPIs)

- Execution of Customer Objects
Displays how often executable customer objects were run in your system compared to the total execution of all executable objects in your system (including SAP, partner, and customer objects)
- ABAP Cloud: Level A Customer Objects
Represents a limited Clean Core Share view (see also chapter Measurability: KPIs), as only Level A objects are considered³¹
- ABAP Classic: Business Modifications
Belong to the group of not recommended extensions, as business modifications are neither upgrade-stable nor cloud-ready

Note:

Customers can already measure the levels defined in the new extensibility concept using ABAP test cockpit.

However, these levels are not yet visualized in the RISE with SAP Methodology dashboard. Full dashboard integration is planned for a future release.

4.4.2 Assessing clean core KPIs individually

While SAP offers a robust framework for tracking KPIs through the RISE with SAP Methodology dashboard, certain scenarios may call for more granular insights than those available in the standard dashboard—either currently or in future releases. One common requirement is the application of individual filter and analysis criteria, most often mapping packages and namespaces to business units, allowing clean core KPIs to be analyzed not only at the system level but also across organizational boundaries.

Whereas Chapter 4.4.1 discussed the standard approach provided through the RISE with SAP Methodology dashboard, it is also possible to extract the underlying KPI data manually, especially for the KPIs described in Chapter 4.1.2. In particular, metrics such as the clean core share and the technical debt score can be calculated by leveraging SAP tools that classify custom code objects based on their referenced elements and

ABAP test cockpit check results.³²

For further customization and to align the technical debt score with specific landscapes and governance processes, SAP promotes the open-source custom ABAP test cockpit check Project

Kernseife³³. It supports the integration of a custom classification model for SAP objects to derive a technical debt score tailored to unique organizational requirements. This approach yields more actionable insights and enhances the ability to prioritize interventions at reducing technical debt.

4.5 How to achieve a clean core

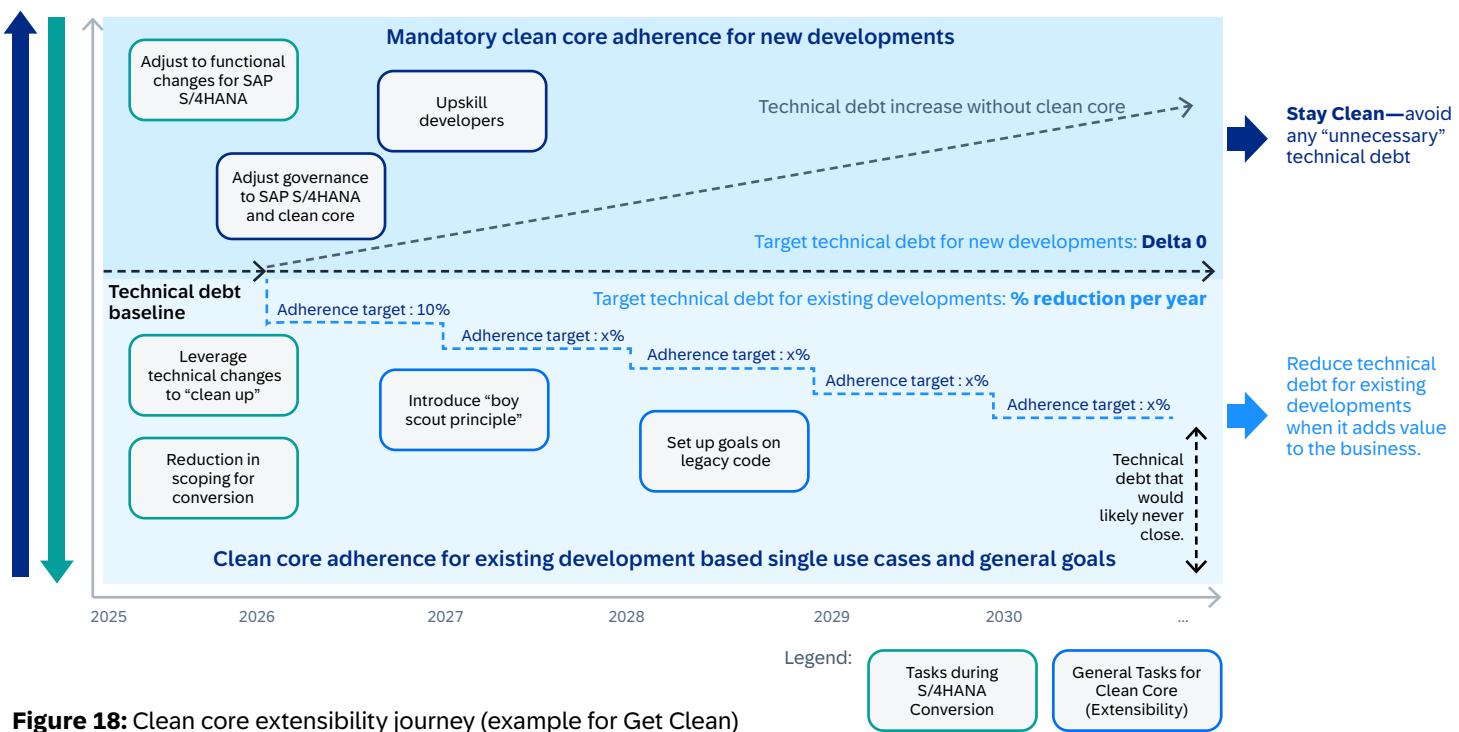


Figure 18: Clean core extensibility journey (example for Get Clean)

Every clean core journey is unique.

This white paper has outlined general strategies and principles, but your own approach will be shaped by your starting point and the strategic goals you set for the years ahead. Whether you are already operating productively with strict clean core rules in place, preparing for a system conversion with legacy extensions, or launching a greenfield implementation, your path will look different.

No matter where you begin, clean core is an approach that can—and needs to—be tailored to specific needs. Some organizations may already be close to their ideal state, while others need to manage technical debt accumulated through years or decades of custom development. Regardless of your point of departure, two key goals are universally valid:

A) Stay Clean

For all new developments, your target should be the strict adherence to clean core principles. This means avoiding any unnecessary technical debt by ensuring every extension follows clear guidelines. As highlighted in the Chapter “Stay Clean: How to create clean extensions”, this requires robust governance, up-to-date guidelines, and enabling your development teams through ongoing training and upskilling. Even if your organization is not yet on SAP S/4HANA Cloud Private Edition and is still running SAP ECC, you can start by applying as much of the clean core model as possible. Extensions built on SAP BTP, for example, can already be implemented. By building these habits now, you lay the groundwork for a future transition to “clean” ABAP Cloud extensions on current SAP S/4HANA Cloud Private Edition releases.

B) Get Clean

Next, focus on your existing landscape—or the landscape you are building. Begin by measuring the technical debt using tools such as the RISE with SAP Methodology dashboard or ABAP test cockpit. Use this baseline to drive your long-term reduction strategy: identify unused or obsolete code, prioritize lighthouse extensions (rebuild these with clean technologies when you need to rework them anyway), and set realistic but ambitious goals—such as an initial reduction target of 10%. Remember, technical debt left unchecked will continue to grow, but continuous incremental improvements, like adopting the “boy scout principle” (always leave code cleaner than you found it), will add up over time.

Clean core is an ongoing journey, not a one-time effort. Progress is measured not only by technology but by the maturity of your processes, governance, and—most importantly—your people. Adapt your approach to your business priorities and always balance targeted improvements with the value they add to the business.

4.5.1 Key steps and recommendations for your Clean core journey

While every clean core journey will be different based on goals, current situation and system history, the following summary guides you like a compass on this journey:

- **Establish clear governance:** Define clean extension guidelines and ensure they cover all relevant SAP S/4HANA Cloud Private Edition developments.
- **Empower your governance:** Involve both business and technical teams—all backed up by a clear commitment from top management.
- **Assess your starting point:** Get a clear picture of your current technical debt and clean core adoption. Use tools to measure and create transparency.
- **Upskill and empower your developers:** Regularly train your teams around clean core principles and new capabilities (such as ABAP Cloud in SAP Build).
- **Adjust for conversion scenarios:** If you are migrating, align with SAP S/4HANA Cloud Private Edition functional changes, and reduce scope or complexity wherever feasible.
- **Set incremental, achievable goals:** Aim for ambitious yet realistic technical debt reduction targets (e.g., 10% per year).
- **Prioritize business value:** Focus remediation efforts on code and customizations with tangible business impact.
- **Continuously monitor and improve:** Incorporate clean core measurement and improvement into regular development cycles and reviews.
- **Adopt the “boy scout principle”:** Encourage a culture of cleaning up code incrementally with every change.
- **Prepare for the future:** Leverage current opportunities (like SAP BTP extensions) and lay the groundwork for further clean core adoption as your landscape evolves.

By consistently acting on these principles, your organization can limit technical debt, boosts agility, and builds a resilient SAP landscape ready for continuous innovation.

4.5.2 How the RISE with SAP Methodology can help you in this journey

As detailed in 4.1, RISE with SAP Methodology provides you with the framework that helps in setting up the right clean core KPIs, measurements to understand where you stand in your journey towards clean core, and a success plan in Cloud ALM with best practice-based tasks and activities to guide you in the journey. In addition, this framework also embeds periodic checkpoints in the form of quality gates and reports with recommendations as well as actions to help you stay on the path of transformation with a clean core mindset. These actions can be added directly into the clean core success plan in SAP Cloud ALM, which is an SAP Activate roadmap that is specifically targeted at clean core and how to run a clean core project. Clean core quality gates further support adherence to the success plan and reinforce the project's objectives.

Enterprise Architects from SAP help you assess clean core leveraging these quality gates as well as the Governance & Maturity approach (both part of RISE with SAP Methodology) as detailed in 4.1.1, including assessment of the customer's current and desired target maturity, across the different clean core dimensions.

The Enterprise Architect also supports you with a periodic review, leveraging a combination of the above approaches to assess your maturity across the clean core dimensions. The Enterprise Architects also manage the action plan with you, helping in prioritizing identified actions to improve the current maturity.



5 F.A.Q.

More information can be find in the related document: [Level_Concept_FAQ.docx](#)



A photograph showing two people in a server room. A man with glasses and a dark sweater is leaning over a desk, looking at a laptop screen. A woman in a light blue blazer is standing beside him, also looking at the laptop. They appear to be discussing something. The background shows rows of server racks.

Table of contents

Clean core extensibility FAQ	3
Classification of SAP Objects	3
Q&A: What are the criteria for an object to be classified by SAP as "not recommended" / "noAPI"?	3
Q&A: If there are different reasons, why an object can be classified as "not recommended" / "noAPI", where can I see this?	3
Q&A: Does a "not recommended" / "noAPI" classified object always have a successor and where can I see it?	3
Q&A: For released APIs there are contracts C1 and C2 to differentiate between local and remote usage. Does the classification apply to local as well as remote usage?	4
Q&A: If an SAP object gets deleted, does it always (also) get classified as "noAPI"?	4
Q&A: Is it guaranteed that objects classified as classic API never change incompatibly?	5
Q&A: What if an object classified as classic API changes incompatibly? Does it then get reclassified as "not recommended" / "noAPI"?	5
Q&A: I want an existing API to be classified as classic API or be released. I want to request a successor API or have a question on an "noAPI" object... Where can I address and request this?	5
Development model	5
Q&A: Which changes of the release state or classification of SAP objects are possible, under which conditions?	4
Q&A: If an SAP object gets deleted, does it always (also) get classified as "noAPI"?	4
Q&A: In which cases should I still wrap objects (former tier 2)? Is this also required when developing in ABAP classic?	5
Q&A: Are ABAP classic customer objects that comply with all restrictions of ABAP Cloud (e.g., only referencing released object) also considered Level A?	6
Q&A: Is a customer object which references no SAP at all also considered Level A?	6
Classification of frameworks and technologies	6
Q&A: How does the level concept translate to the ABAP language versions (ABAP Cloud for developers, ABAP Cloud for key users)?	5
Q&A: In which cases should I still wrap objects (former tier 2)? Is this also required when developing in ABAP classic?	5
Q&A: Are ABAP classic customer objects that comply with all restrictions of ABAP Cloud (e.g., only referencing released object) also considered Level A?	6
Q&A: Does ATC also give error / warning / info messages, when using Level D, C, B classified frameworks and technologies?	6
Q&A: How does the level concept relate to technologies and frameworks? For SAP objects, SAP releases or classifies them (e.g., "not recommended" / "noAPI", "classic API") and this then results in custom code referencing these objects being level A, B, C, D.	6
Q&A: What does a conditional classification of a framework or technology mean?	6

6 Appendix

Below, you can find relevant links that can support you on your journey towards a clean core.

Links	Descriptions
ABAP Cloud	Dive deeper into ABAP Cloud, the ABAP development model to build cloud-ready business apps, services, and extensions.
Clean core strategy website	Read up on how an ERP clean core strategy helps your organization ensure that business-critical systems remain agile, cost-effective, and ready to adopt innovation.
Guide: Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions	Overview the extensibility options for ABAP-based extensions on SAP S/4HANA Cloud editions and guidelines for project managers, key users, and ABAP developers.
Extension Architecture Guide	Learn about the extension architecture guide and how you can use it to determine which SAP technology should be used in a given scenario.
Extension use case patterns guidelines	Determine which parts of an extension run on SAP S/4HANA on-stack and which parts run side-by-side on the SAP Business Technology Platform.
From Classic ABAP to ABAP Cloud and clean core	Get an overview of the implementation of and certification options for ABAP Cloud add-ons for SAP partners.
How to Get Started with Joule Studio	Explore Joule Studio, the latest capability in SAP Build, that enables developers to build Joule skills and AI agents tailored to unique business needs. These capabilities extend Joule copilot, orchestrate cross-functional workflows, and enhance enterprise efficiency.
RISE with SAP Methodology website	Get details on the RISE with SAP Methodology that helps on-premises customers modernize their business processes based on clean core principles to enable continuous innovation.
Overview: SAP Application Extension Methodology	Learn about this structured, technology-agnostic approach for customers and partners to define an organization-specific extension strategy.
SAP Architecture Center	Find solution reference architectures that help businesses adopt SAP solutions to turn data into valuable business insights.
SAP BTP ABAP environment community page	Join the SAP community for SAP BTP ABAP environment.
SAP BTP Developer's Guide	Learn about the SAP BTP Developer's Guide and how you can use it to improve the process of implementing a business application on SAP BTP.

Links	Descriptions
SAP BTP Developer's Guide	Learn about the SAP BTP Developer's Guide and how you can use it to improve the process of implementing a business application on SAP BTP.
SAP BTP Guidance Framework	Discover the central access point for architects, developers and administrators to build and run enterprise-grade solutions on SAP BTP.
SAP Build website	Explore the unified application development and process automation solution, enabling personalization and efficiency at scale.
SAP Business Accelerator Hub	Find everything needed to accelerate integrations, extensions, and innovations (e.g., APIs, events, CDS views, pre-built automation content, etc.)
SAP Cloud Application Programming Model website	Learn how to build cloud-native applications with maximized productivity at minimized costs, based on proven best practices served out of the box.
SAP Discovery Center	Explore missions (step-by-step guidance on use case implementation), SAP BTP services, and SAP Business AI offerings.

Sources

¹ “SAP Cloud ERP Private” refers specifically to the product SAP S/4HANA Cloud, private edition (PCE).

² For more information, see [“Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions,” SAP brochure](#), April 2024.

³ For more information on ABAP Cloud, see the [documentation](#) on the SAP Help Portal site.

⁴ For more information about available SAP standard APIs and events, see [SAP Business Accelerator Hub](#).

⁵ See also: [Understanding the Energy Label](#) by European Commission for reference.

⁶ Refer to chapter 3.3.2 for details on the transformation from the 3-tier extensibility model to the clean core-level concept.

⁷ SAP’s 3-tier extensibility model was introduced with the release of ABAP Cloud and described the approach to distinguish between classic extensibility (Tier 3) and modern, stable and

“clean” extensibility (Tier 1). The connection through Tier 2 (“API enablement) described the approach to wrap classic APIs and release them for usage in ABAP Cloud.

⁸ To increase the focus, the qualifiers in this chapter are marked with brackets [].

⁹ “contract” is not a legally binding commercial agreement, but a technical term

¹⁰ The changelog will also include classic APIs in the rare case of incompatible changes.

¹¹ For more details on the relevant ABAP test cockpit check, see the [ABAP extensibility guide](#) and [SAP Note 3565942](#).

¹² Notably, Level A corresponds to Tier 1 in the legacy 3-tier extensibility model.

¹³ For more information on Release Contract, see the [documentation](#) on the SAP Help Portal site.

¹⁴ Changelog for SAP Objects is currently planned to be released aligned with the release of SAP S/4HANA 2025

- ¹⁵ The changelog is focused on internal objects but will also include classic APIs in case of incompatible changes.
- ¹⁶ The new clean core ABAP test cockpit check provides base data for the technical debt score and the clean core share. For more details, refer to [SAP Note 3565942](#) and the document “[Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions](#)”.
- ¹⁷ The calculation logic of business modifications is explained in more detail in the document [Modification Analysis](#)
- ¹⁸ Although SAP recommends to use SAP Cloud ALM (as part of SAP’s integrated toolchain) for requirements management, this is not mandatory. Other tools can also be used.
- ¹⁹ Customers applying SAP’s Application Extension Methodology (see chapter [Extension Architecture](#)) should use the deliverables created during the different phases (e.g. extension use case description) as input for the user story documentation in SAP Cloud ALM.
- ²⁰ Learning journey “Acquiring Core ABAP Skills”—<https://learning.sap.com/learning-journeys/acquire-core-abap-skills>
- ²¹ Learning journey “Managing Clean Core for SAP S/4HANA Cloud”—<https://learning.sap.com/learning-journeys/managing-clean-core-for-sap-s-4hana-cloud>
- ²² Learning journey “Practicing Clean Core Extensibility For SAP S/4HANA Cloud”—<https://learning.sap.com/learning-journeys/practicing-clean-core-extensibility-for-sap-s-4hana-cloud>

²³ [ABAP Cloud API Enablement Guidelines for SAP S/4HANA Cloud Private Edition and SAP S/4HANA](#)

²⁴ Setup governance for the classic ABAP development model in [Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions](#)

²⁵ How to create a Quality Manager Role on SAP Help Portal [Maintaining the List of ATC Approvers](#)

²⁶ How to create a software component on SAP Help Portal [Create Software Component](#)

²⁷ Setting up the governance for the classic ABAP development model in [Extend SAP S/4HANA in the cloud and on premise with ABAP based extensions](#)

²⁸ Check the “[What’s New in SAP Cloud ALM](#)” space on a regular basis to learn about new features and enhancements in RISE with SAP Methodology dashboard

²⁹ More detailed information about the System View is available on [SAP Help Portal](#).

³⁰ As mentioned before, the RISE with SAP Methodology dashboard is continuously being enhanced and supplemented with further KPIs.

³¹ Please be aware that Key User objects are not yet reflected in this evaluation.

³² Chapter references ABAP test cockpit checks published in SAP Note 3565942.

³³ More information on Project Kernseife can be found here: <https://github.com/SAP/project-kernseife>