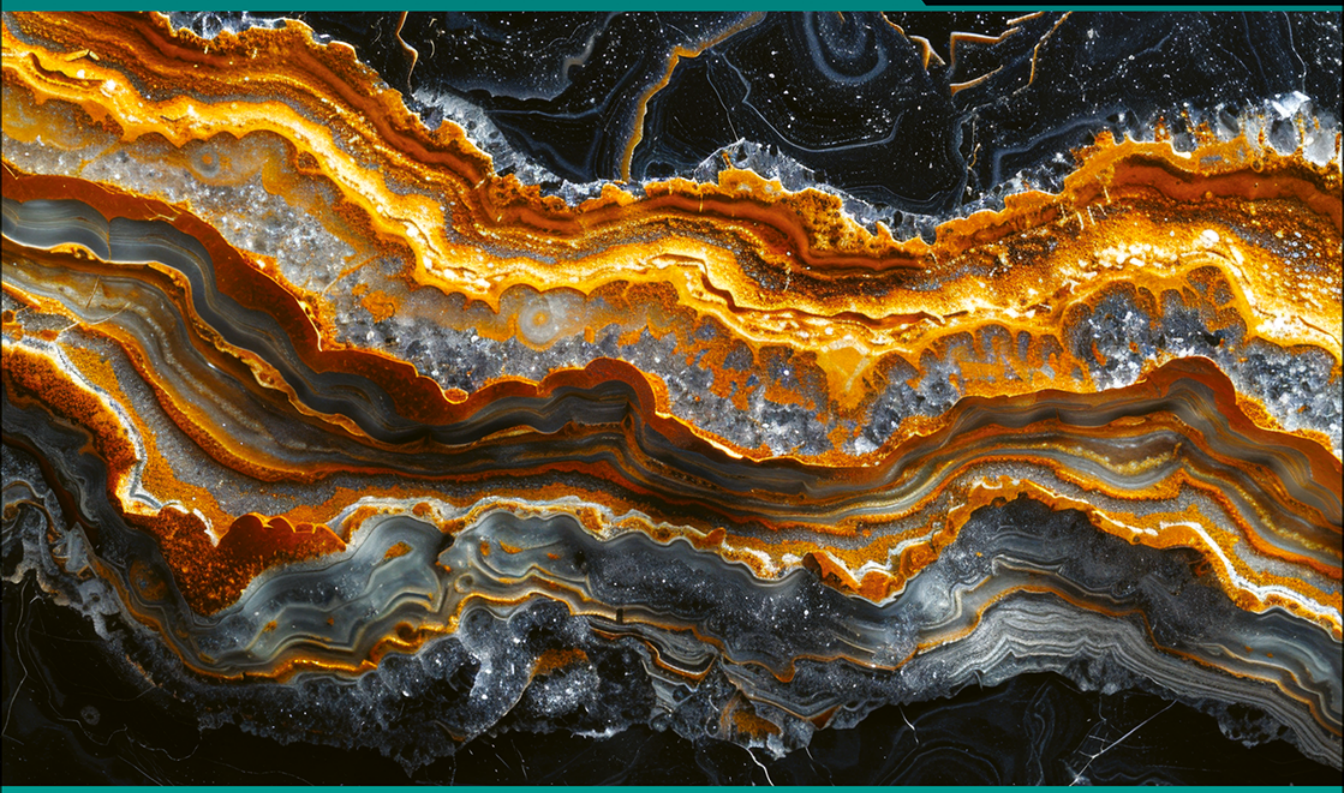
SAP PRESS



# Clean Core for SAP®

## Extensibility and Upgrade Stability for SAP S/4HANA

- Understand clean core guidelines for development, integration, data management, and operations
- Use new extensibility models, extension patterns, and programming methodologies for compliant custom development
- Address existing technical debt using SAP tools and services

Smitha Banda
Vani Krishnamoorthy

Rheinwerk Publishing

# Contents

Preface ..... 15

## PART I Foundations of Clean Core

### 1 What Is Clean Core? ..... 21

**1.1 Architectural Shift: From Custom Core to Clean Core** ..... 22
1.1.1 A Brief History of SAP Implementations ..... 22
1.1.2 New Business Imperatives ..... 25

**1.2 Principles and Execution** ..... 27
1.2.1 Definition of Clean Core ..... 27
1.2.2 The Five Guiding Principles of the Clean Core ..... 30
1.2.3 The Clean Core in Practice: The TnX Credit Check Reimagined ..... 32
1.2.4 The Inherited Landscape ..... 33

**1.3 Clean Core: Common Misconceptions** ..... 35
1.3.1 Flexibility and Scope ..... 35
1.3.2 Applicability and Lifecycle ..... 36
1.3.3 Value and Effort ..... 37

**1.4 Summary** ..... 39

### 2 Process Dimension ..... 41

**2.1 Defining a Clean Process** ..... 41

**2.2 Balancing Standards and Innovation via Business Process Management** 43
2.2.1 Understanding Business Process Management as a Foundation for Transformation ..... 43
2.2.2 The Role of Business Process Management in a Clean Core Strategy 44
2.2.3 Business Process Management in SAP Activate ..... 45

**2.3 Fit to Standard as the Anchor** ..... 50

**2.4 Tools for Fit to Standard and Process Discovery and Design** ..... 55
2.4.1 SAP LeanIX ..... 55
2.4.2 SAP Signavio ..... 56
2.4.3 SAP Cloud ALM ..... 59

**2.5 Summary** ..... 59

**3 Extending the Core with Innovation** ..... 61

**3.1 Why Clean Extensions Promote Innovation** ..... 62
3.1.1 Infrastructure Change and Custom Code Risk ..... 62
3.1.2 Clean and Decoupled Extensions ..... 63
3.1.3 Released Application Programming Interfaces ..... 65

**3.2 Clean Core Extension Levels** ..... 66
3.2.1 Clean Core Levels ..... 67
3.2.2 Classification Decision Flowchart ..... 73
3.2.3 Cloudification Repository ..... 76
3.2.4 Clean Extension Key Performance Indicators ..... 79
3.2.5 Clean Extension Governance ..... 81

**3.3 Extension Types** ..... 83

**3.4 SAP Application Extension Methodology** ..... 85

**3.5 Summary** ..... 87

**4 Connecting the Core Through Integrations** ..... 89

**4.1 Defining Clean Integrations** ..... 90

**4.2 Path to a Clean Core** ..... 91

**4.3 SAP Integration Solution Advisory Methodology** ..... 96

**4.4 Integration Domains, Integration Styles, and Use Case Patterns** ..... 99
4.4.1 Integration Domains ..... 100
4.4.2 Integration Styles ..... 101
4.4.3 Use Case Patterns ..... 104

**4.5 Tools and Platforms** ..... 105

**4.6 Integration Governance** ..... 110

**4.7 Summary** ..... 114

**5 Data at the Core** ..... 115

**5.1 Defining Clean Data** ..... 116

**5.2 Path to Clean Data** ..... 118
5.2.1 Data Strategy ..... 118
5.2.2 Data Governance ..... 120
5.2.3 Data Quality ..... 122
5.2.4 Data Volume Management ..... 124
5.2.5 Data Protection ..... 125

**5.3 Tools and Technologies** ..... 127

5.3.1 SAP Data and Analytics Advisory Methodology ..... 128
5.3.2 SAP Business Data Cloud ..... 130
5.3.3 SAP Master Data Governance ..... 131

**5.4 Summary** ..... 133

## 6 Operating the Core ..... 135

**6.1 The Foundation That Makes the Clean Core Real** ..... 136

**6.2 What Clean Operations Encompasses** ..... 137
6.2.1 Governance and Operational Practices ..... 139
6.2.2 Enable the Integrated Toolchain ..... 141
6.2.3 Adopt the Latest SAP Autonomous Suite Innovations ..... 143
6.2.4 Ensure Operational Efficiency ..... 145
6.2.5 Establish Secure Operations ..... 147
6.2.6 Implement Agile Delivery and Operational Excellence: DevOps ..... 147
6.2.7 Track and Reduce Technical Debt ..... 149

**6.3 Tools and Enablers** ..... 150
6.3.1 RISE with SAP Methodology ..... 150
6.3.2 SAP LeanIX ..... 151
6.3.3 SAP Cloud ALM ..... 152
6.3.4 SAP Business Technology Platform ..... 152
6.3.5 Tricentis ..... 152

**6.4 Summary** ..... 152

# PART II Building Clean Extensions

## 7 Extensibility Models ..... 157

**7.1 On-Stack Extensions** ..... 159
7.1.1 Key User Extensions ..... 160
7.1.2 Developer Extensibility ..... 166
7.1.3 Classic Extensibility ..... 169
7.1.4 The Wrapper Pattern ..... 171
7.1.5 When No Clean Path Exists ..... 176
7.1.6 The Mechanics of the Upgrade Shield ..... 177
7.1.7 Common Pitfalls ..... 178

**7.2 Side-by-Side Extensions** ..... 179
7.2.1 Placement and Design ..... 179
7.2.2 Prebuild Considerations ..... 182

7.2.3 Lifecycle Independence ... 185
7.2.4 Governance ... 185
7.2.5 Common Pitfalls ... 186

**7.3 Hybrid Extensions** ... 187

**7.4 Summary** ... 190

**8 Extension Patterns and Architecture** ... 193

**8.1 The Requirement Triage** ... 194

**8.2 Extending Reports** ... 195
8.2.1 Direct Table Access and Why It Fails ... 197
8.2.2 Custom Analytical Queries ... 198
8.2.3 Developer Extensibility for Reporting ... 204

**8.3 User Interface Extensions** ... 206
8.3.1 User Interface Adaptations and Personalization ... 208
8.3.2 Metadata-Driven Extensions ... 209
8.3.3 Entry Point Optimization: SAP Fiori Launchpad ... 212
8.3.4 UI5 Adaptation Projects ... 213

**8.4 Extending the Data Model** ... 214
8.4.1 Data Model Extension Triage ... 215
8.4.2 Custom Fields ... 216
8.4.3 Custom Business Objects ... 220
8.4.4 Custom Business Configurations ... 223

**8.5 Business Logic Extensions** ... 225
8.5.1 Custom Logic Using Key User Extensibility ... 226
8.5.2 Complex Logic Extensions Using Developer Extensibility ... 227

**8.6 Extending Outputs** ... 234

**8.7 Extending Process Flows** ... 238
8.7.1 Agent Determination ... 239
8.7.2 Classic Workflow ... 241

**8.8 Building Custom Applications** ... 242

**8.9 Summary** ... 243

**9 Programming Methodologies** ... 245

**9.1 SAP Application Extension Methodology** ... 246
9.1.1 Phase 1: Assess Extension Use Case ... 247
9.1.2 Phase 2: Assess Extension Technology ... 251
9.1.3 Phase 3: Define Extension Target Solution ... 256

9.1.4 Assessment for Less Complex Extensions ..... 259

**9.2 SAP Cloud Application Programming Model** ..... 260

9.2.1 Domain Modeling Using Core Data Services ..... 262
9.2.2 The Service Layer ..... 265
9.2.3 Authorization and Trust Management ..... 267
9.2.4 Events and Event Handlers ..... 269
9.2.5 Metadata-Driven User Interface ..... 272

**9.3 ABAP RESTful Application Programming Model** ..... 274

9.3.1 Architecture ..... 275
9.3.2 Data Model ..... 278
9.3.3 Business Objects ..... 281
9.3.4 Query ..... 286
9.3.5 Business Service ..... 286
9.3.6 ABAP RESTful Application Programming Model Extensibility ..... 289
9.3.7 Choosing a Programming Model ..... 296

**9.4 ABAP Cloud** ..... 297

9.4.1 The API Contract ..... 299
9.4.2 ABAP Cloud Language Scope ..... 302
9.4.3 Transitioning from Classic ABAP to ABAP Cloud ..... 303
9.4.4 SAP Joule for Developers and ABAP AI Capabilities ..... 313
9.4.5 Enforcing ABAP Cloud Compliance at Scale ..... 316

**9.5 Summary** ..... 316

**10 Development Tools** ..... 319

**10.1 The Landscape: SAP Build and Joule** ..... 320

**10.2 Joule Studio** ..... 324

**10.3 Intent-Driven Development** ..... 327

10.3.1 The Pipeline ..... 328
10.3.2 Paths from Intent to Production ..... 331
10.3.3 Clean Core Relevance ..... 333

**10.4 Powering Intent with Business Context** ..... 333

**10.5 Trusted Governance and the Managed Runtime** ..... 334

**10.6 Side-by-Side Extensibility as the Default** ..... 335

**10.7 Availability, Adoption, and the Commercial Model** ..... 336

**10.8 Summary** ..... 338

## 11 ABAP Test Cockpit ..... 339

**11.1 ABAP Test Cockpit as a Governance Control** ..... 341

**11.2 ABAP Test Cockpit in the SAP BTP ABAP Environment** ..... 344

**11.3 ABAP Test Cockpit in SAP S/4HANA** ..... 347

11.3.1 Getting the Checks Right ..... 347

11.3.2 How the Wrong Check Reveals Itself ..... 348

11.3.3 Configuring the Transport Gate ..... 349

**11.4 Custom ABAP Test Cockpit Checks** ..... 351

11.4.1 Building a Custom Variant ..... 351

11.4.2 Separating Governance from Quality ..... 353

**11.5 Summary** ..... 354

# PART III Clean Core Journey

## 12 Mindset ..... 359

**12.1 Guidelines** ..... 359

**12.2 Extending the Mindset to Implementation Partners** ..... 363

**12.3 Summary** ..... 364

## 13 Clean Core in a Greenfield vs. Brownfield Implementation ..... 365

**13.1 Greenfield** ..... 366

**13.2 Brownfield and Bluefield** ..... 367

**13.3 Technical Debt** ..... 371

13.3.1 Make Technical Debt Visible ..... 371

13.3.2 Categorize What to Retain, Refactor, Retire, and Rethink ..... 372

13.3.3 Clean as You Go ..... 374

13.3.4 Adopt a Strangler Pattern ..... 375

13.3.5 Decouple Customizations from the Core ..... 376

**13.4 From Principles to Practice** ..... 376

**13.5 Summary** ..... 379

**14 RISE with SAP** ..... 381

**14.1 RISE with SAP Methodology** ..... 381

**14.2 Solution Standardization Board** ..... 387

**14.3 Governance in Operation** ..... 391

14.3.1 Solution Standardization Board: Operating Model ..... 391

14.3.2 Key Design Decision Process ..... 392

14.3.3 Governance Artifacts ..... 394

14.3.4 The Clean Core Runbook ..... 396

14.3.5 Roles and Responsibilities ..... 397

**14.4 Summary** ..... 398

**15 Clean Core Governance and Maturity Framework** ..... 399

**15.1 Clean Core Governance and Maturity Framework** ..... 399

15.1.1 Process Dimension ..... 400

15.1.2 Extensibility Dimension ..... 402

15.1.3 Data Dimension ..... 404

15.1.4 Integration Dimension ..... 406

15.1.5 Operations Dimension ..... 408

15.1.6 Conducting the Assessment and Acting on Results ..... 410

15.1.7 The Clean Core Health Score ..... 410

**15.2 TnX Manufacturing: Practical Example** ..... 411

15.2.1 Stage One - Mid-Implementation: Establishing the Baseline ..... 411

15.2.2 Stage Two - Go-Live: The First Formal Assessment ..... 413

15.2.3 Stage Three - Twelve Months Post Go-Live: Measuring Progress ..... 415

15.2.4 Stage Zero: Pretransformation ..... 418

**15.3 Summary** ..... 419

The Authors ..... 421

Index ..... 423

Chapter 4

# Connecting the Core Through Integrations

*The integration dimension of clean core is a critical enabler for achieving a streamlined and sustainable SAP landscape, while connecting SAP seamlessly to the broader enterprise ecosystem, including non-SAP applications, data platforms, cloud services, and external partners. Clean integration ensures efficient system communication, alignment of business processes with standard SAP capabilities, and the ability to extend functionality without compromising maintainability or future upgrades. This chapter examines the principles, patterns, tools, and governance practices that enable clean integration across SAP and non-SAP systems, helping organizations minimize complexity, preserve core integrity, accelerate innovation, and reduce long-term operational costs.*

Enterprise IT landscapes now span core SAP systems, industry cloud solutions, third-party software as a service (SaaS) platforms, partner networks, and legacy applications. In this interconnected environment, integration is no longer a technical necessity alone—it's a foundational capability.

As organizations grow through regional expansion, acquisitions, and specialized business models, applications multiply, cloud and on-premise environments coexist, and integration points proliferate. Poorly designed or overly customized integrations often become fragile, tightly coupled, and difficult to maintain. Data silos emerge, information flow slows, and technical debt increases. This places significant strain on IT teams and limits the organization's ability to respond quickly to new business demands, adopt AI-driven capabilities, or scale efficiently across increasingly interconnected supply chains and partner ecosystems.

Addressing this challenge requires rethinking integration as a strategic discipline rather than a technical afterthought. A sustainable SAP landscape depends on a clean core supported by a future-ready integration approach; the two are inseparable. Without disciplined integration design and governance, even the most modern ERP core becomes constrained by complexity at its edges.

This chapter covers the integration side of the clean core discipline. It starts by defining what clean integration means and the path to getting there, then examines SAP's Integration Solution Advisory Methodology as the structured approach for making integration decisions. From there, it walks through integration domains, styles, and use case patterns, followed by the tools and platforms SAP provides to implement them. The chapter closes

with integration governance, the practices that determine whether your integration layer stays clean over time or drifts back to where it started.

Note that clean integration applies whether your SAP core runs on-premise, in a private cloud, or in a public cloud. Although the tooling differs across those targets, the discipline doesn't.

## 4.1 Defining Clean Integrations

Most SAP landscapes carry integration debt, such as old IDocs nobody fully understands, Remote Function Calls (RFCs) to systems that were retired five years ago, or custom proxies whose business owner left the company two reorganizations ago. None of this debt was created on purpose. It accumulated, one project at a time, because integration was treated as a delivery problem rather than an architectural one.

A clean core needs clean integrations around it. If the enterprise resource planning (ERP) system stays standard but the interfaces remain custom, the upgrade still hurts. Every release still triggers regression testing of dozens of custom flows. Every new business capability still negotiates with a brittle interface layer before it can ship.

Clean integration means connecting SAP and non-SAP systems using patterns that SAP supports as standard and that survive upgrades without rework. In practice, this points to a few specific choices:

- Use released application programming interfaces (APIs) that are published on SAP Business Accelerator Hub at *https://api.sap.com* rather than internal entries that SAP doesn't commit to keeping stable.
- Use business events through SAP Integration Suite or SAP Integration Suite, advanced event mesh rather than point-to-point RFCs, especially where loose coupling matters.
- Treat IDocs, RFCs, and the older proxy-based interfaces as legacy: They still work, but they earn their place by exception now, not by default.
- Push custom logic out of the core to side-by-side extensions on SAP Business Technology Platform (SAP BTP; see Chapter 7 for details) rather than into modifications or classical user exits.

Clean integration is achieved through two complementary phases that together ensure long-term sustainability: a corrective phase that addresses existing technical debt, and a preventive phase that ensures new integrations don't reintroduce complexity. These phases are commonly referred to as the *get clean* and *keep clean* approaches. Chapter 13 introduces get clean and keep clean phases for landscape-level transformation.

The get clean approach focuses on assessing the current integration landscape, challenging historical design decisions, and modernizing legacy interfaces where appropriate. The keep clean approach, in contrast, establishes governance, standards, and decision frame-

works that guide all future integration requirements. This ensures that once improvements are made, they are preserved over time.

Together, these approaches form a continuous improvement cycle that protects the SAP core while enabling flexibility, innovation, and ongoing business transformation.

## 4.2 Path to a Clean Core

The next step is understanding how organizations can achieve clean integration in practice. Implementing a clean integration strategy involves not only adopting standard tools and APIs but also following structured processes and strong governance. The following three guiding rules serve as a roadmap to ensure that integrations remain efficient, scalable, and aligned with business objectives. Together, they define the integration strategy and approach to guide organizations through a get clean execution phase typically applied in brownfield and bluefield implementations and enabling a continuous keep clean mindset applicable both to greenfield and brownfield implementations. More on greenfield and brownfield implementations can be found in Chapter 13. The three guidelines are described here:

- **Plan and establish an integration strategy.**
  A clean integration begins with a well-defined integration strategy. SAP Integration Solution Advisory Methodology provides the foundation for establishing this strategy. It provides structured guidance to map integration domains, choose integration styles (cloud, on-premise, or hybrid), define technology mappings, and set governance principles. Using SAP Integration Solution Advisory Methodology ensures that integration initiatives are scalable, maintainable, and aligned with business goals, while providing a repeatable approach across the enterprise. Refer to Section 4.3 for details on SAP Integration Solution Advisory Methodology.
- **Define and execute the get clean process.**
  The get clean process focuses on assessing the current integration landscape and identifying areas for improvement. This includes analyzing the as-is architecture versus the target to-be state, evaluating existing technical debt, and pinpointing opportunities to replace legacy interfaces with standardized, released SAP APIs or business events that align with clean core principles.

  A key aspect of this assessment is deliberately addressing legacy integration patterns that no longer reflect SAP's strategic direction. A commonly encountered example is the continued use of IDocs. When an organization identifies IDocs in its current as-is architecture, the first step isn't to immediately replace them. Instead, the scenario must be clearly understood. This begins by validating the underlying business requirement: What purpose does the IDoc serve? Which business process depends on it? Is this requirement still relevant, or has it persisted mainly due to historical design decisions?

For each interface, two questions come first: What business process depends on this, and is that process still active? For example, a sales-order replication to a regional distributor the company sold two years ago isn't an integration problem; it's a decommissioning problem. Inventory updates to a warehouse system that the new SAP S/4HANA logistics module already covers isn't a replacement; it's a removal. Skipping this step is how get clean programs end up modernizing interfaces that nobody needed in the first place.

Once the business need is confirmed, the technical and operational characteristics of each interface need a closer look. The recurring questions are:

- *Is the communication pattern still right?* An IDoc was the right answer in 2008 when the volumes were daily batches of a few thousand records and the receiving system could tolerate hours of latency. The same interface today may be feeding a real-time inventory display in a customer-facing app—same data, different latency requirement and different pattern. Is asynchronous still required, or would a synchronous call now meet the need?
- *What are the expected message volumes and throughput requirements?* An interface that ran once a day at 10,000 records is a different design problem from one running continuously at 10,000 records per minute.
- *Are there specific demands around reliability, sequencing, or error handling?* Some scenarios require guaranteed-once delivery, ordered processing, or transactional consistency across systems. These constraints shape which target technology fits.
- *Is there a released SAP API or business event on SAP Business Accelerator Hub (https://api.sap.com) that covers the functional scope?* SAP Business Accelerator Hub is the source of truth for what SAP supports as standard, and the same place to find prebuilt integration packages, business process blueprints, and reference architectures that may already cover the use case. If the API exists, the question is whether to switch to it. If it doesn't, the question becomes whether to wait for it—and what to do in the meantime.
- *What does replacement cost?* Adding SAP Application Interface Framework, SAP Integration Suite, or advanced event mesh to the landscape costs licenses, runtime, and skills. A clean target architecture isn't free, and the business case has to absorb the new operational footprint.
- *What is the value of replacing it?* The value gained is better decoupling, more useful error handling, an interface that survives the next upgrade without modification, and alignment with the cloud-ready target architecture. These are real benefits, but not infinite benefits—some interfaces aren't worth the disruption.

From a clean core perspective, this is an architectural decision rather than an ideological one. IDocs and other legacy patterns aren't wrong by category; they are wrong where a released, supported alternative now covers the same scope and the business value justifies the move. They are right where no alternative yet exists and the existing interface is doing its job.

A concrete example is a custom RFC function module that an external warehouse system calls to create material master records using `BAPI_MATERIAL_SAVEDATA`. In SAP S/4HANA Cloud Public Edition, `BAPI_MATERIAL_SAVEDATA` isn't released for external consumption. The clean core replacement is the Product (A2X) OData API, available on SAP Business Accelerator Hub. The migration involves switching the calling system to OAuth-secured HTTPS, mapping the legacy Business Application Programming Interface (BAPI) fields to the OData schema, and retesting the error scenarios. The interface keeps doing the same thing for the business but now sits inside something SAP commits to maintaining.

A second example, more common than the first, is a custom IDoc with Z-segments—typically an extension of `MATMAS05` or `ORDERS05`—that the team built 10 years ago to carry fields that didn't fit the standard segments. In an SAP S/4HANA migration, the segments themselves usually survive, but the partner profiles in Transaction WE20 and the distribution model in Transaction BD64 need rework, and the custom enhancements in the IDoc processing function modules need to be reapplied. The replacement option is to publish the same data as a business event with the additional fields carried in event extensions, which removes both the segment fragility and the partner-profile maintenance. Whether to make that move depends on volumes and the number of receivers; for a single receiver with low volumes, retaining the IDoc may be the right call.

Not every interface should be replaced. Where no released API or event yet covers the use case, retaining the existing interface is the pragmatic call. There are two things to do in that situation: Log the gap, and request the missing capability through the SAP Customer Influence portal. The portal is how SAP prioritizes which APIs and events to release next; customer requests visibly affect the roadmap. Retaining an IDoc-based integration as an interim is acceptable as long as the gap is tracked and the eventual move is on the plan.

This discussion assumes the harder question is whether to replace an interface. There's a separate trap waiting for teams that decide not to replace something but to move it: An interface that works in the source system, gets transported to the target, and then fails because part of it doesn't move with the transport.

The classic case is the ABAP proxy. A proxy class in SAP ERP is generated, not transported. It's built in Transaction SPROXY against a service interface definition in the Enterprise Service Repository (ESR) of whichever middleware system the source ERP points to. In the target SAP S/4HANA system, the proxy has to be regenerated against the new middleware endpoint—whether that is SAP Process Integration (SAP PI)/SAP Process Orchestration (SAP PO) or Cloud Integration—and the custom code inside the proxy method has to be transferred manually. A standard transport won't carry a working proxy across systems. No automated remediation tool fills the gap. Teams that plan proxy migration as a transport exercise tend to discover the problem during testing if they are lucky and after go-live if they aren't.

A related problem appears when a team decides to migrate some interfaces to Cloud Integration and leave the rest in SAP PI/SAP PO. The `SAP_PROXY_ESR` destination in

Transaction SM59 is a single, system-level configuration. It points to one ESR endpoint. Every ABAP proxy on that system is generated against whichever endpoint that destination points to. There's no way to have some proxies routed to SAP PI/SAP PO and others to Cloud Integration on the same source system. The decision is all-or-nothing: Either every ABAP proxy moves to Cloud Integration or every one stays in SAP PI/SAP PO. Teams that split the interface migration without accounting for this constraint end up either consolidating mid-program or building an operational workaround, and the workaround isn't what SAP recommends.

The broader lesson generalizes beyond proxies. A lift-and-shift integration strategy assumes every interface is transportable code. Some aren't. The get clean assessment should flag any interface technology whose remediation is generation-time rather than transport-time. ABAP proxies are the obvious case. Transaction SOAMANAGER reliable-messaging configurations, partner profiles in Transaction WE20, classical RFC destinations in Transaction SM59, and middleware artifacts such as SAP PI/SAP PO mappings or Cloud Integration iFlows fall into the same category. Each one requires manual effort in the target system. Budget that effort before the cutover plan is locked, not after.

- **Establish and maintain a continuous keep clean approach.**
  Clean core integration isn't a one-time initiative. It requires a continuous keep clean approach to ensure that new integration requirements don't reintroduce technical debt over time. This involves defining clear governance structures, decision criteria, and assessment processes that are consistently applied whenever a new integration requirement arises. This is applicable both in greenfield and brownfield/bluefield implementations.

  At the center of this approach is a standardized method for evaluating new or changed integration needs. Organizations must clearly define how integration requirements are assessed, which stakeholders are involved in decision-making, and which architectural criteria must be met before a technology choice is finalized.

  The mechanism is straightforward: a defined assessment process, a decision body that owns it (typically the Integration Center of Excellence [CoE]—see Section 4.6 later in this chapter), and a tool that records the decisions. SAP's Integration Assessment tool, available with SAP Integration Suite, supports this. It guides architects and integration designers through a configurable questionnaire and produces a coverage analysis showing which SAP technologies fit the requirement and how the integration scenario maps to the target architecture.

  The questionnaire ships with defaults but is configurable. Organizations should extend it to include non-SAP tools, organization-specific patterns, custom rules, and any preferred-vendor decisions already made. The assessment isn't limited to SAP-only landscapes; the value increases when the same approach applies across the full integration estate rather than only the SAP portion of it.

  The questions inside the questionnaire are those a good integration architect would ask, such as the following:

- Does this need to be synchronous or asynchronous?
- What is the volume?
- Is the receiver inside the enterprise (application to application [A2A]), a partner (business to business [B2B]), or a government body (business to government [B2G])?
- Is there a released API or event?
- Can predelivered SAP integration content be reused rather than built from scratch? Where available, it should be preferred.
- Is protocol conversion needed, for example, Simple Object Access Protocol (SOAP) to OData?
- Which technical connectors apply? OData, REST, web services, or application-specific adapters?
- What monitoring and error-handling capabilities are required?

The value of running these questions through a tool isn't that the tool is smarter than the architect; it's that the answers, the rationale, and the resulting decision are recorded, applied consistently across projects, and reviewable later.

Here's an example. The business asks for material master data to be replicated from SAP S/4HANA to a third-party product information management (PIM) system in near real time with light transformation along the way. Backend performance and capacity constraints rule out direct synchronous calls into SAP S/4HANA; the integration has to be decoupled. The assessment surfaces the relevant pattern (event-driven A2A) based on the loose-coupling and real-time propagation requirements. Further questions refine the design. A lightweight transformation is needed, so a pure point-to-point pattern isn't enough; a released OData service exists for the read side, which avoids any custom APIs; the receiver is internal, so a B2B configuration isn't required; and predelivered SAP integration content is checked first and applied where applicable.

The output is a recommendation: business event publication from SAP S/4HANA via advanced event mesh, consumption and transformation in Cloud Integration, and downstream call via the PIM system's REST API. SAP Application Interface Framework handles exception management in SAP S/4HANA for the source-side errors, and SAP Cloud ALM provides end-to-end runtime monitoring across the two clouds. The architect signs the assessment record. Six months later, when a second team asks for the same thing for the vendor master, the previous decision is the starting point rather than the conversation starting from zero—and the same pattern gets applied consistently.

The keep clean approach depends on monitoring as well as decisions. SAP Application Interface Framework gives business users the ability to see and act on interface errors inside SAP S/4HANA without raising an IT ticket. A logistics clerk seeing a failed shipment IDoc can correct the missing field and resubmit. SAP Cloud ALM extends monitoring across hybrid and cloud landscapes for the cross-system operational view. Both tools are covered in detail in Section 4.5.

> The point of running get clean and keep clean phases together is that the cleanup is sustained rather than reversed. Most failed integration modernization programs did so because one of the two was missing—either there was a cleanup with no governance to hold it, or there was governance applied to an estate that was never cleaned up in the first place.

Clean integration also establishes the foundation for advanced capabilities such as embedded analytics, workflow automation, and business AI. These innovations depend on consistent, reliable, and well-governed data flows across systems. API-based and event-driven integrations enable near real-time data propagation, which is essential for intelligent scenarios such as predictive insights, automated decision-making, and cross-system orchestration. Without clean integration, such capabilities remain isolated experiments rather than scalable enterprise solutions.

## 4.3 SAP Integration Solution Advisory Methodology

SAP Integration Solution Advisory Methodology is SAP's framework for treating integration as something that gets planned and governed rather than ordered project by project. Rather than impose a specific technology stack, it gives an organization a common vocabulary for integration scenarios, a way to classify them, and a sequence of decisions for moving from "we have a lot of interfaces" to "we have an integration strategy." The methodology is technology-neutral by design: SAP provides the reference content, the tooling, and the example patterns, but organizations can incorporate non-SAP and third-party tools where they fit. This keeps SAP Integration Solution Advisory Methodology applicable across diverse enterprise landscapes rather than locking it to a single vendor stack.

The framework covers all the integration dimensions that show up in modern landscapes—on-premise, cloud, hybrid, edge, partner, and machine—and the integration styles that span them: process, data, analytics, user, and machine integration. Together, these provide a unified way to classify, design, and govern integrations consistently.

In many organizations today, integration is still approached project by project. Integration requests get initiated informally, design decisions are made under delivery pressure, and documentation is fragmented or incomplete. The result is the pattern most landscapes already display: siloed implementations, inconsistent patterns, limited governance, and no shared language for evaluating the next request. As cloud adoption, hybrid landscapes, and digital business models increase complexity, this approach produces operational risk and architectural drift—and the organization struggles to scale integrations, modernize legacy interfaces, or introduce capabilities such as event-driven architectures.

SAP Integration Solution Advisory Methodology addresses these challenges by introducing a structured, standards-based approach with clear terminology and consistent decision-making. The methodology turns integration from reactive implementation into a governed capability aligned with enterprise architecture and business objectives. It's structured around four phases that an organization works through in order and then revisits as the landscape evolves, as shown in Figure 4.1:

1. **Phase 1: Assess your integration strategy.**
   The first phase scopes the integration estate at a strategic level, before any technology choice is made. The output is a shared understanding of what kinds of integration the organization does. Key activities include the following:
   - Scoping integration domains with scenarios such as on-premise to on-premise, on-premise to cloud, cloud to cloud, and machine to cloud, as described in detail in Section 4.4
   - Scoping integration styles, including process, data, analytics, user, and machine integration
   - Scoping integration use-case patterns such as A2A, B2B, and B2G
   - Establishing a catalog of integration use-case patterns to classify and standardize future integration requests

   The exit criterion for the phase is a catalog the organization has agreed on, with example scenarios for each pattern. The catalog is the artifact every later assessment compares against. If a new integration request fits a pattern already in the catalog, the decision should be consistent with how that pattern was last decided.

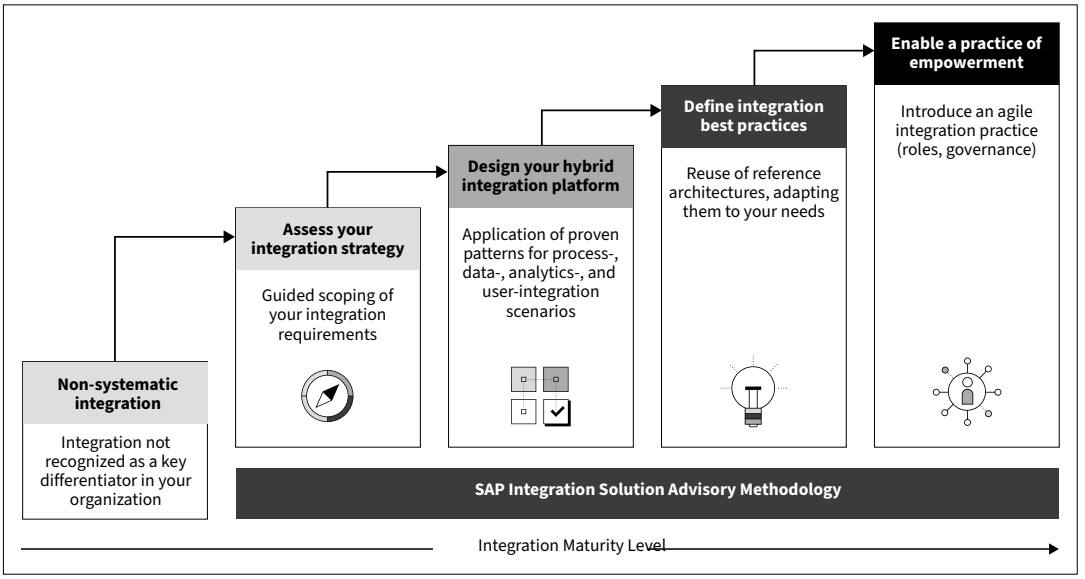



*Figure 4.1: SAP Integration Solution Advisory Methodology: Integration Maturity Levels*

2. **Phase 2: Design your hybrid integration platform.**
   The second phase designs the hybrid integration platform that supports the catalog. The core activity is technology mapping: matching each use-case pattern to the SAP or partner tool that fits it best, as shown in Figure 4.2. Here are examples of the mappings most organizations end up with: Cloud Integration for A2A process integration with mappings; advanced event mesh for event-driven A2A; API Management for managed external API

exposure; SAP Integration Suite's Trading Partner Management for B2B; and SAP Datasphere for analytics integration. Key activities in this phase include the following:

- Applying proven integration patterns for process, data, analytics, and user integration scenarios.
- Performing technology mapping to align integration styles and use cases with appropriate platforms and tools.
- Defining integration policies such as the architectural principles, security requirements, and lifecycle rules that apply across all integrations. This is where decisions such as "all external-facing APIs go through API Management" or "no new IDoc-based interfaces without architecture board approval" get written down. Policies are easier to defend in delivery pressure when they exist as a document rather than as one person's preference.
- Assessing existing interfaces against the target architecture and identifying candidates for modernization, redesign, or retirement.

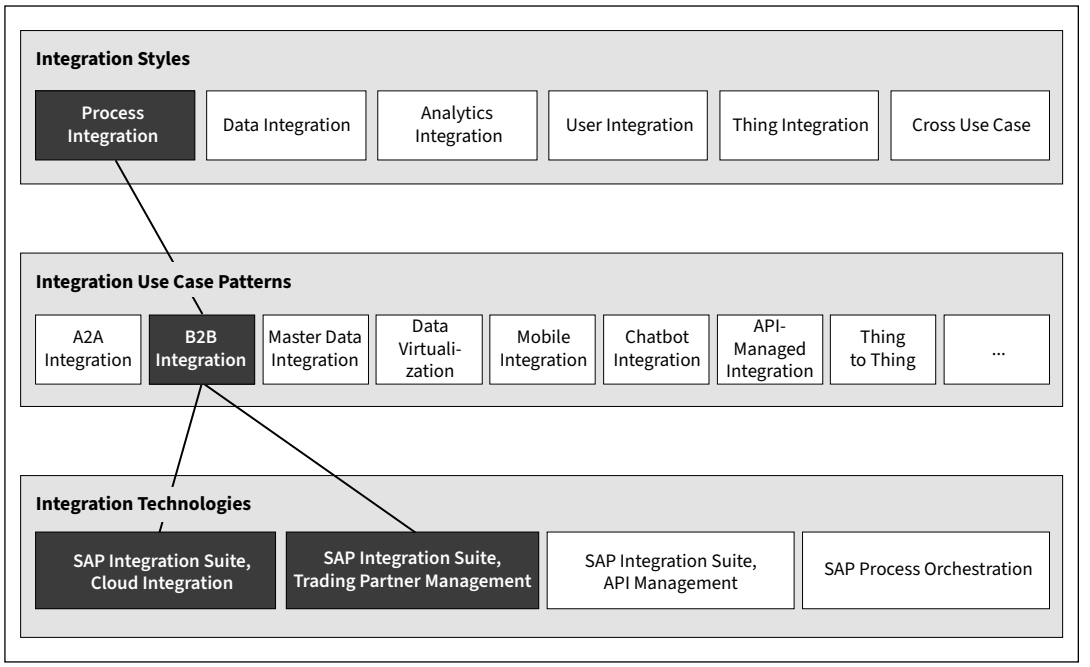



*Figure 4.2: Technology Mapping*

3. **Phase 3: Define integration best practices.**
   Once the platform is designed, the third phase produces the design guidelines that developers and architects use day to day. Key activities include the following:
   - Reusing reference architectures for the common patterns and adapting them to organizational needs.

- Defining clear integration dos and don'ts to guide design and implementation decisions.
- Creating a comprehensive integration architecture blueprint covering naming conventions for integration flows (iFlows), packages, APIs, and queues, plus error-handling standards.
- Developing tailored development guidelines for each integration technology and pattern. The guidelines for Cloud Integration are different from those for advanced event mesh, which are different again from those for API Management.

The point of this phase is to reduce variability between projects. Two integration developers working in different teams should arrive at the same design when they get the same requirement. The CoE is the body that owns these guidelines and keeps them current as SAP's tooling evolves.

4. **Phase 4: Enable a practice of empowerment.**
   The final phase turns integration into an organizational capability rather than a methodology document. Key activities include the following:
   - Introducing an agile integration practice with defined roles (integration architect, integration developer, solution architect, business expert—see Section 4.6)
   - Establishing the right organizational structure, typically an Integration CoE
   - Implementing integration governance processes for decision-making and oversight
   - Ensuring integration quality assurance through standards, reviews, and continuous improvement

SAP Integration Solution Advisory Methodology is iterative. Phase 4 informs Phase 1. The CoE is the body that reviews the catalog, updates the technology mapping when SAP releases new capabilities, and revises the guidelines as the team learns what works. By applying SAP Integration Solution Advisory Methodology, organizations move from reactive, ungoverned integration delivery to a structured model where integration decisions are consistent, use-case driven, and aligned with both business objectives and long-term architectural sustainability.

Full SAP Integration Solution Advisory Methodology documentation, templates, and reference content are available on the SAP Help portal at *https://help.sap.com/docs/sap-btp-guidance-framework/sap-integration-solution-advisory-methodology/sap-integration-solution-advisory-methodology-overview*. SAP also publishes integration catalogs and example scoping documents that can be adapted rather than written from scratch.

## 4.4 Integration Domains, Integration Styles, and Use Case Patterns

This section defines the building blocks of enterprise integration by outlining integration domains, categorizing integration styles, and classifying integration use-case patterns to guide design, technology selection, and governance.

### 4.4.1 Integration Domains

A *domain* describes the connectivity context, that is, which side of the network boundary each system is on. Scoping integration domains is a foundational activity in defining an enterprise integration strategy, independent of specific business processes or technologies. By identifying these domains up front, organizations can make informed architectural decisions, apply appropriate security and governance models, and select integration patterns that are fit for purpose. The security model, performance characteristics, and connection technology differ across domains. As shown in Figure 4.3, the common integration domains are as follows:

- **On-premise to on-premise**
  Covers integrations between two systems hosted within the same on-premise environment or data center. These scenarios typically involve tightly coupled enterprise systems such as ERP, warehouse management, or legacy applications. They prioritize low latency, transactional consistency, and direct connectivity, and historically have used RFC, IDoc, or proxy-based patterns.

  Example: An on-premise SAP ERP system exchanges financial postings with an on-premise warehouse management system in real time.

- **On-premise to cloud/cloud to on-premise**
  Addresses hybrid scenarios where one side of the interface sits inside the corporate network and the other in a public cloud. It's common in landscapes where core systems remain on-premise while innovation happens in the cloud. These interfaces require secure network traversal—typically the cloud connector for inbound calls into the corporate network, or reverse proxy and TLS for outbound—and the latency and availability profile differs from on-premise to on-premise.

  Example: An on-premise SAP S/4HANA system sends sales order data to a cloud-based customer relationship management (CRM) application.

- **Cloud to cloud**
  Connects applications hosted in different cloud environments or platforms. These integrations must account for multitenant architectures, scalability, standardized interfaces, and standardized authentication (OAuth 2.0, OpenID Connect). They tend to be the easiest interfaces to govern because both endpoints are designed for API-based interaction from the start.

  Example: A cloud-based human capital management (HCM) system such as SAP SuccessFactors integrates with a cloud analytics platform to provide workforce insights.

- **Machine to cloud**
  Covers integrations where physical devices, sensors, or machines communicate with cloud platforms. These scenarios are common in internet of things (IoT) and industry use cases and involve high-frequency, event-based data with specialized protocols. The volumes are higher than typical A2A integration.

Example: Temperature sensors monitoring equipment in a manufacturing plant publish telemetry via Message Queuing Telemetry Transport (MQTT) to advanced event mesh for use by predictive-maintenance models in SAP Asset Performance Management.

- **User to on-premise or user to cloud**
  Covers integrations driven by end-user actions—submitting an expense, approving a purchase request, or looking up a customer record across systems. These integrations are typically synchronous, low-volume per call, and front-ended by a user interface, emphasizing secure access, role-based authorization, and responsive interfaces. The SAP Fiori launchpad, SAP Build Work Zone, and SAP Build Process Automation all live in this domain.

  Example: A business user approves purchase requests through a cloud-based workflow application that communicates with backend systems via APIs, while a finance user accesses an on-premise ERP system through a corporate portal to review financial postings.

By scoping integration domains early, organizations gain clarity on connectivity requirements, security considerations, and architectural constraints before addressing integration styles or technologies. This structured approach reduces ambiguity, supports consistent governance, and ensures that integration decisions are aligned with both current and future-state enterprise architectures.

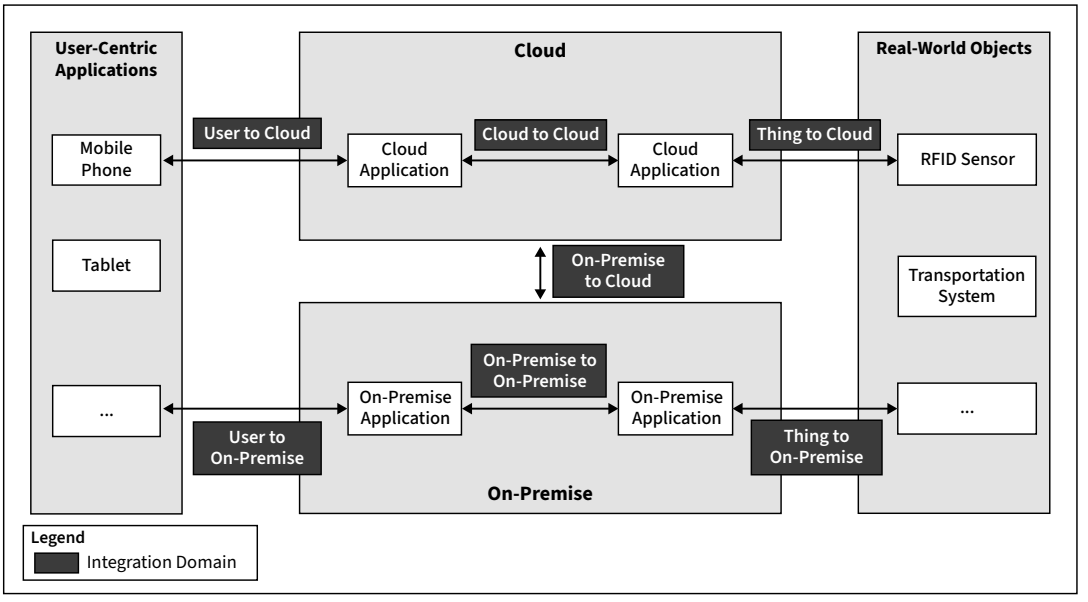



*Figure 4.3: Integration Domains*

### 4.4.2 Integration Styles

A *style* describes what flows across the interface. The same two systems can be connected through several styles for different purposes, and each style points to different SAP tools.

SAP Integration Solution Advisory Methodology uses integration styles to categorize integrations based on what is being integrated, providing a structured approach to selecting the right technology, architecture, and operational model for each type. As shown in Figure 4.4, the common integration styles are as follows:

- **Process integration**
  Focuses on transactional, real-time interactions between applications to support business processes, including small payloads, synchronous or near-synchronous communication, and strict transactional consistency. Cloud Integration in SAP Integration Suite is the workhorse here.

  Example: When a purchase order is created in a cloud-based procurement system, the corresponding update in an on-premise ERP system can be executed immediately using Cloud Integration for real-time API calls, or orchestrated via advanced event mesh for event-driven workflows. Similarly, employee onboarding can be automated across HR and payroll systems using workflow-based integrations.
- **Data integration**
  Handles batch-oriented or stream-based large-volume data movement, ensuring that systems remain synchronized over time. These integrations emphasize throughput and scalability rather than low-latency responses. SAP Datasphere is the primary tool for analytics-oriented data integration, and SAP Master Data Integration on SAP BTP handles the master-data side specifically.

  Example: Sales, inventory, and customer master data are replicated nightly from an on-premise ERP system to a cloud data warehouse, supporting consolidated reporting and analytics. Another example is harmonizing product information from multiple legacy systems into a centralized master data repository.
- **Analytics integration**
  Delivers trusted and harmonized data to analytical platforms for reporting, business intelligence, and predictive insights. It overlaps with data integration but is specifically about delivering data into reporting and AI contexts. SAP Datasphere and SAP Analytics Cloud are the typical targets; SAP Business Data Cloud is the broader product that brings them together.

  Example: IoT sensor data is streamed from manufacturing equipment into SAP Datasphere for predictive-maintenance models, with results visualized in SAP Analytics Cloud, combining structured and event-driven data sources.
- **User integration**
  Supports the user-driven scenarios from Section 4.4.1—typically API-based, with a UI layer and authentication. SAP Build Process Automation handles the workflow side, and API Management secures the backend exposure.

  Example: An employee submitting an expense report via a cloud portal triggers workflow approvals and updates in the ERP system, orchestrated using SAP Build Process Automation and API Management. Likewise, service agents accessing multiple systems through

a unified SAP Fiori interface rely on real-time API integrations to provide consistent and accurate information.

- **Machine or thing integration**
  Enables devices, sensors, and machines to communicate with enterprise systems. These high-frequency, event-driven scenarios often use specialized protocols. Advanced event mesh is built for this style.

  Example: Sensors monitoring chemical tank levels publish events to advanced event mesh, triggering automated alerts and inventory updates in downstream SAP applications. Similarly, production machines can stream operational data to SAP Manufacturing Execution systems using SAP Integration Suite for protocol mediation and workflow orchestration.

Each integration style has its own technical, operational, and governance profile:

- Process integration weights transactional integrity and synchronous messaging.
- Data integration weights throughput and large-volume handling.
- Analytics integration weights data quality and semantic preservation.
- User integration weights authentication and response time.
- Machine integration weights ingest rate and protocol flexibility.

Tagging an interface against its style is what tells the assessment process which tool and which monitoring approach apply.

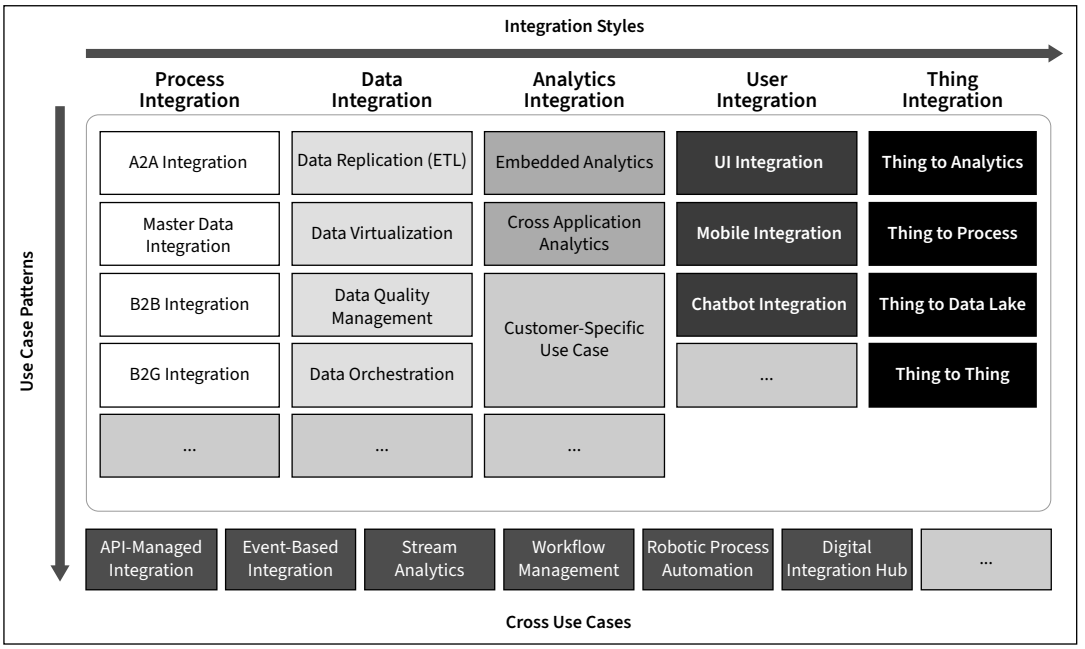



*Figure 4.4: Integration Styles and Use Case Patterns*

### 4.4.3 Use Case Patterns

A *use case pattern* describes the relationship between the sender and the receiver. Once integration domains and styles are established, the use case pattern determines what security, validation, and operational treatment the interface needs. The main patterns are as follows:

- **Application-to-application (A2A)**
  Covers real-time or near real-time interactions between applications fully controlled and trusted within the enterprise. Trust is implicit. These integrations are usually process-oriented and transactional, and they are often API-driven with small payloads and strict consistency requirements. Integration can be tightly controlled through internal API Management and shared identity providers.

  Example: Inventory updates are synchronized between an on-premise ERP system and a warehouse management system. Both applications belong to the enterprise, and integration can be tightly controlled using Cloud Integration or event-driven orchestration via advanced event mesh.

- **Master data integration**
  This is a specialized A2A pattern for keeping consistent, high-quality reference data across systems—customers, vendors, products, cost centers, and organizational hierarchies. It often uses a central master data management solution to synchronize key business objects. SAP Master Data Governance is the typical hub when one is needed; SAP also offers SAP Master Data Integration for synchronizing master data between SAP cloud applications without a central hub.

  Example: Customer data from various legacy systems is consolidated into a centralized SAP Master Data Governance solution and validated records are replicated to ERP, CRM, and cloud platforms. SAP Master Data Integration can combine batch and real-time techniques depending on data criticality and latency requirements.

- **Business-to-business (B2B)**
  Covers interactions with external partner systems—suppliers, customers, logistics providers, financial institutions—that aren't fully controlled or trusted by the organization. Security, message validation, acknowledgment, and exception handling are weighted more heavily than in A2A because direct access to partner systems is limited. SAP Integration Suite's Trading Partner Management capability supports the common B2B protocols (AS2, AS4, EDIINT, Peppol) and message format standards (EDIFACT, X12, IDoc-over-AS2).

  Example: Invoices or purchase orders are exchanged with a supplier's ERP system via AS2-secured EDIFACT, with acknowledgment tracking and a configured retry policy for failed deliveries.

- **Business-to-government (B2G)**
  Addresses interactions with government agencies and regulators, often involving specific compliance, security, or archiving requirements. The receiving system is outside

enterprise control, message formats are mandated, and compliance requirements typically include digital signatures, archiving, and audit trails. These integrations may be synchronous or asynchronous depending on the regulatory mandate.

Example: E-invoices are submitted to the Peppol network in Europe, ZATCA in Saudi Arabia, or Nota Fiscal Eletrônica in Brazil. The SAP Document and Reporting Compliance offering covers many of these mandates with prebuilt content, which avoids reinventing the compliance logic for each country.

The three classifications work together. A single interface might be tagged as on-premise to cloud domain, process integration style, and B2B use-case pattern. That combination determines which SAP tool, security model, monitoring approach, and governance treatment apply. Tagging each interface this way is the input the Integration Assessment tool uses to recommend a target architecture, and it's what makes the assessment process repeatable rather than starting from zero for every request.

## 4.5 Tools and Platforms

SAP's integration tooling is a stack rather than a single product. Different tools own different stages of the lifecycle such as business process design, interface documentation, technical assessment, runtime, and monitoring. Clean integration depends on the tools being used as a chain, with each handoff captured, so that the rationale for an integration decision is traceable from business requirement to running interface. The goal isn't to use every tool, but to use the right tool at the right stage and to leave an artifact behind at each handoff. This section walks through each tool in the order it appears in that chain, plus what each one does that the alternatives don't:

- **SAP Signavio**
  SAP Signavio is SAP's process modeling platform. For integration purposes, its role is to be the place where business process steps get linked to the integrations that support them. When a process is modeled in SAP Signavio, each step is tagged with the interfaces that fulfill it. When the business changes the process, the impact on integrations is visible immediately. The differentiator is that SAP Signavio links integrations to processes from the business side, while SAP LeanIX links them from the architecture side. Both are needed because business owners and architects ask different questions of the same landscape. Key capabilities include the following:
  - Business process modeling: Define, visualize, and analyze end-to-end business processes.
  - Traceability to integrations: Link each process step to supporting integrations, ensuring that technical implementations directly support business objectives.
  - Process change management: When processes evolve, SAP Signavio ensures that integration requirements are updated accordingly, preventing misalignment.

Example: A manufacturing company using SAP Signavio can model its order-to-cash process and link each step from order entry, inventory check, and invoicing to specific SAP and non-SAP integrations. Any change in the business process automatically triggers a review of the impacted interfaces.

- **SAP LeanIX**
  SAP LeanIX is the enterprise architecture management tool SAP acquired in 2023. Its role in the integration chain is to be the authoritative inventory of every interface in the landscape, with the metadata an architect needs: sender system, receiver system, business object, data mapping, owning team, criticality, and related business process. When SAP S/4HANA is upgraded, SAP LeanIX is the source for which interfaces are affected. When a system is retired, SAP LeanIX is the source for which downstream systems need to be informed. Without an SAP LeanIX–style inventory, integration governance is conjecture. Most landscapes have somewhere between 2 and 10 times as many interfaces as anyone can list from memory, and getting them into a repository is the first concrete step of any get clean program. Key capabilities include the following:
  - Interface documentation: records sender and receiver systems, data mappings, business objects, and process links.
  - Architecture visibility: maintains an inventory of technical components, their dependencies, and integration points.
  - Impact assessment: enables assessment of potential risks, gaps, or redundancies in integration landscapes.

  Example: A finance team can view all interfaces related to invoice processing. If a system is scheduled for an upgrade, SAP LeanIX helps identify dependencies and assess whether integrations need redesigning to maintain clean core standards.
- **Integration Assessment**
  Integration Assessment is an SAP Integration Suite service that supports the technical decision for each interface. It takes the tagging from SAP Integration Solution Advisory Methodology (domain, style, pattern), runs through a configurable questionnaire, and produces a recommendation: which SAP tool, integration pattern, and monitoring approach. The differentiator is that it records the decision. A questionnaire-and-recommendation flow is replicable to anyone with experience; what's hard is keeping the decisions consistent across many projects over many years. The tool's value is the historical record more than the live recommendation—when a second team brings a similar requirement two years later, the previous decision is the starting point rather than the conversation starting from zero. Key capabilities include the following:
  - Evaluate integration patterns and complexity through SAP-delivered questionnaires that are customizable by customers.
  - Assess alignment with clean core principles.
  - Identify optimization opportunities for existing integrations.

Example: A company considering an API-based integration versus a traditional IDoc-based interface can use the assessment tool to weigh factors such as scalability, maintainability, and cloud readiness—and the recorded outcome becomes the standard for similar future requests.

- **SAP Integration Suite**
  SAP Integration Suite is the cloud-based platform where most modern SAP integrations run. It bundles several capabilities under one license: Cloud Integration (the iFlow runtime, successor to SAP PI/SAP PO for new integrations), API Management (managed API exposure with security, throttling, transformation, and analytics policies), Open Connectors (prebuilt connectors to common non-SAP SaaS applications), Trading Partner Management (the B2B side covering AS2, AS4, Peppol), and Integration Advisor (mapping assisted by machine learning [ML] for B2B message formats).

  The differentiator versus SAP PI/SAP PO is twofold. First, Integration Suite is cloud-delivered: SAP runs the runtime, applies upgrades, and absorbs the operational burden that on-premise SAP PI/SAP PO required organizations to handle. Second, it ships with a prebuilt content library on SAP Business Accelerator Hub covering common SAP-to-SAP and SAP-to-non-SAP scenarios. A common pattern in get clean programs is to start by surveying which existing SAP PI/SAP PO interfaces are already covered by prebuilt Integration Suite content; the percentage is usually higher than the team expects, which reduces the migration build effort significantly. SAP has confirmed the end-of-mainstream-maintenance dates for SAP PI/SAP PO that, for most organizations, force a Cloud Integration migration decision in the medium term—the SAP PI/SAP PO sunset is the most common trigger for an integration modernization program. Key capabilities include the following:
  - Modern integration patterns: supports API-first, event-driven, and hybrid integration approaches
  - Reusable content: offers prebuilt connectors, integration packages, and templates to accelerate development
  - Security and scalability: ensures standardized, secure, and scalable connectivity across the landscape

  Example: A retail organization can use SAP Integration Suite to expose SAP S/4HANA business objects via standard APIs while integrating cloud applications for pricing, logistics, and customer engagement without modifying the core system.

- **SAP Integration Suite, advanced event mesh**
  Advanced event mesh is the cloud event broker SAP offers as part of its integration stack, built on Solace's platform. Its role is to distribute business events across systems in real time, decoupling publishers from subscribers. The differentiator versus Cloud Integration is that Cloud Integration is for orchestrated flows where the sender expects an outcome (synchronous-ish, request-response-shaped). Advanced event mesh is for fire-and-forget event distribution where the sender doesn't know or care who consumes the event. The two are complementary; most landscapes use both, with Cloud Integration handling the orchestrated cases and advanced event mesh handling the broadcast

cases. Each subscriber can be added or removed without touching the publisher—that is the architectural payoff of event-driven integration over point-to-point. Key capabilities include the following:

- Event-based communication: publishes and consumes business events to decouple systems and reduce point-to-point dependencies
- Real-time responsiveness: allows immediate reactions to business events such as order creation, shipment updates, or inventory changes
- Support for intelligent processes: powers automated notifications, adaptive workflows, and reactive business scenarios

Example: When a sales order is created in SAP S/4HANA, advanced event mesh can instantly trigger inventory checks, warehouse processing, and customer notifications across multiple systems without tight coupling between the publisher and the consumers.

- **SAP Datasphere**

  SAP Datasphere is the cloud data platform for combining SAP and non-SAP data with business semantics intact. For integration purposes, it owns the data-integration style: pulling data from SAP and non-SAP sources, modeling it without physically replicating it into the SAP core, and exposing it to analytics consumers. The differentiator is that SAP Datasphere preserves the business context of SAP data—measures, dimensions, hierarchies, time-dependent attributes—rather than flattening it to rows and columns as a generic extract, transform, load (ETL) tool would. SAP-aware consumers downstream get the semantic model; non-SAP consumers get the data through standard interfaces. Key capabilities include the following:

  - Data integration and modeling: combines data from SAP and non-SAP sources without replication into the core
  - Business semantics: preserves business context and relationships for analytics and reporting
  - Governed access: ensures data consistency, quality, and compliance through centralized governance

  Example: A finance team can use SAP Datasphere to integrate financial data from SAP S/4HANA and external planning systems, enabling real-time profitability analysis without creating custom data extracts in the core system.

- **SAP Business Data Cloud**

  Announced in February 2025, SAP Business Data Cloud is the umbrella offering that brings SAP Datasphere, SAP Analytics Cloud, the SAP Business Warehouse (SAP BW) data products, and a partnership with Databricks into a single managed environment for enterprise data. It's not a peer of SAP Datasphere—it's the broader product that includes it. For clean core purposes, the practical impact is that integration to analytics increasingly goes through SAP Business Data Cloud rather than directly to SAP Datasphere; the data products it ships are premodeled and intended for direct consumption, which removes a layer of modeling work from analytics-integration projects. For organizations

already running on SAP Datasphere, the move to SAP Business Data Cloud is the next architectural decision worth tracking. Key capabilities include the following:

- Master and transactional data management: ensures consistent data definitions across systems
- Data sharing across landscapes: supports data consumption by analytics, AI, and downstream applications
- Compliance and reliability: maintains data integrity and governance across hybrid and cloud environments

Example: A global enterprise can use SAP Business Data Cloud to ensure that customer and product master data remains consistent across SAP S/4HANA, CRM, analytics platforms, and AI-driven applications.

- **SAP Cloud ALM**

  SAP Cloud ALM tracks integration projects through implementation and into operations. For integration governance, its role is the cross-landscape visibility layer: which interfaces are in development, which are in production, which are failing, and which need attention. It complements SAP Application Interface Framework, which monitors errors inside SAP S/4HANA, by extending the view across hybrid and multicloud environments. A typical division of responsibility is that SAP Application Interface Framework handles the exception queues in SAP S/4HANA that business users own, and SAP Cloud ALM handles the cross-system runtime health that the operations team owns. Key capabilities include the following:

  - Project tracking: Monitor progress, milestones, and deliverables for integration projects.
  - Operational monitoring: Track system performance, error rates, and interface health.
  - Governance oversight: Ensure that integrations comply with established standards and guidelines.

  Example: During a global rollout, SAP Cloud ALM enables the project team to track which integrations are ready for production, which require remediation, and which are operating outside clean core compliance.

- **SAP Application Interface Framework**

  SAP Application Interface Framework is the monitoring and error-handling tool that runs inside SAP S/4HANA. Its differentiator is the audience—SAP Application Interface Framework is designed for business users to monitor and act on interface errors without IT involvement. A logistics clerk seeing a failed shipment IDoc in SAP Application Interface Framework can correct the field that caused the failure and resubmit. Without SAP Application Interface Framework, the same correction requires an IT ticket, a developer, and a few hours of turnaround. For clean integration, SAP Application Interface Framework earns its place specifically for the interfaces where business knowledge is needed to fix errors—most often master data and document-based interfaces. Not every interface needs to be in SAP Application Interface Framework; for technical-only interfaces, SAP

Cloud ALM monitoring is sufficient. Configuring SAP Application Interface Framework for everything is a common overengineering trap. Key capabilities include the following:

- Real-time error alerts with clear, actionable messages
- Business-user-friendly interface to correct or resubmit failed messages
- Logging and audit trails to maintain compliance and traceability

Example: A logistics team can quickly identify a failed shipment interface, correct missing data, and resubmit the transaction without IT intervention, reducing downtime and improving responsiveness.

Together, these tools provide an end-to-end view of the integration landscape, linking business processes to their supporting integrations and their operational monitoring. The following example shows how the chain operates from business requirement through implementation to monitoring:

1. A business process modeled in SAP Signavio identifies the need for real-time data synchronization between SAP S/4HANA (ERP) and a cloud-based CRM system as part of the order-to-cash process.
2. The required interfaces are documented in SAP LeanIX, capturing sender and receiver systems, data mappings, business objects, and links back to the originating business process, providing full architectural visibility and traceability.
3. Integration Assessment evaluates the optimal integration approach—API-based integration using SAP Integration Suite, or an event-driven pattern using advanced event mesh—based on scalability, clean core compliance, and future extensibility. The recorded decision becomes the starting point for the next similar request.
4. The integration is implemented using SAP Integration Suite, with business events published via advanced event mesh where near real-time responsiveness is required. Relevant transactional and analytical data is harmonized and consumed through SAP Datasphere or SAP Business Data Cloud, enabling reporting and downstream analytics without impacting the core system.
5. SAP Cloud ALM monitors development, deployment, and runtime performance of the integrations, while SAP Application Interface Framework enables business users to monitor and resolve integration errors without direct IT intervention.

Every step leaves an artifact behind, and the next team that needs to integrate a different business object has a template to follow rather than starting from zero. This is what the toolchain is supposed to look like in practice. Without it, integration decisions become anecdotal and the discipline doesn't survive past the first round of staff changes.

## 4.6 Integration Governance

The get clean and keep clean phases describe what to do. Governance describes who decides, how decisions get recorded, and how the organization learns from one project to

the next. Without governance, the assessment process happens in one project, then quietly stops happening in the project after it, and the patterns the architecture team thought it had stopped start reappearing.

The structure most organizations land on is a CoE—a team that serves as the central authority for defining integration strategies, principles, and standards. By centralizing knowledge and accountability, the CoE provides visibility across the end-to-end integration landscape, from business process requirements to enterprise architecture considerations and detailed integration assessments. The CoE doesn't build interfaces in delivery teams; it owns the decisions that delivery teams reference when they build interfaces. This is what prevents ad hoc or misaligned integrations from proliferating across the landscape.

Integration governance can begin at different points in an organization's transformation journey, and the trigger often shapes its initial focus. Some organizations establish governance while transforming legacy SAP PO landscapes to SAP Integration Suite. Others do it during cloud infrastructure and interface migration initiatives, or as part of a broader RISE with SAP journey—whether through system conversion, selective transformation, or greenfield implementation. The trigger matters less than the establishment itself; clean integration is hard to maintain without a body that owns the standards. In every case, the objective is the same: Modernize the integration landscape, capture business value from the modernization, and keep the integration estate able to evolve as new business models arrive.

A working CoE typically has four roles, though one person may hold more than one in smaller organizations:

- *Integration architects* are responsible for defining the overall technical strategy, the architecture of integrations, and the integration development guidelines. They are the people the assessment tool's questionnaire is calibrated to and the ones who decide what goes into the catalog.
- *Integration developers* focus on building and maintaining integrations while adhering to the guidelines. They are usually outside the CoE in delivery teams, but the CoE owns their standards.
- *Solution architects* ensure that integrations align with enterprise architecture and business objectives. They are the bridge to broader landscape decisions—application strategy, platform strategy, security architecture.
- *Business experts* provide the domain knowledge that anchors integrations to actual business processes. They are the people the architect calls when "Is this requirement still real?" needs an answer.

These roles collaborate to maintain visibility of both technical and business contexts, which is particularly important when organizations are transitioning from on-premise to cloud-based integration models.

A central pillar of integration governance is the creation of implementation design guidelines. These guidelines ensure that integrations are designed and implemented to consistent standards aligned with clean core principles, promoting simplicity, standardization,

and a sustainable enterprise architecture. This matters especially during transformation initiatives, where legacy integration patterns need to be reassessed and modernized rather than replicated in the cloud. Key activities include the following:

- Defining standardized design principles for interfaces, including naming conventions for iFlows, packages, APIs, and queues; mapping rules and transformation logic conventions; and error-handling mechanisms
- Establishing comprehensive documentation strategies to provide transparency across the organization
- Setting security standards—which authentication mechanism applies to which scenario
- Setting lifecycle conventions—versioning, deprecation, retirement of interfaces
- Maintaining a lean interface repository (typically SAP LeanIX) that clearly defines the following:
  - Sender and receiver systems
  - Mapping rules and transformation logic
  - Business objects and associated business processes
  - Interface attributes and metadata

The repository serves as a reference for future development, ensuring that all new integrations adhere to the CoE's standards and facilitate continuous governance and optimization. It also plays a key role during migration programs by helping identify redundant, obsolete, or noncompliant interfaces that should be redesigned or retired. Most landscapes that haven't maintained such a repository discover during their first thorough inventory that they hold at least one interface no one is actively using but no one has dared retire because the dependency is unclear. The repository is how those dependencies become clear enough to act on.

SAP's integration tooling evolves faster than most enterprise tools do. SAP Integration Suite has shipped material new capabilities every quarter for several years. The CoE has to invest in keeping its people current; a CoE that was current in 2023 and hasn't been re-enabled since is now actively giving wrong advice about, for instance, SAP Build Process Automation's role in workflow scenarios or which capabilities have moved between SAP Integration Suite and adjacent products in successive releases. An effective enablement journey involves the following:

- Structured training programs to keep CoE personas up-to-date on new technologies, integration patterns, and SAP product capabilities
- Awareness of emerging integration tools and best practices, ensuring the CoE remains at the forefront of innovation
- Knowledge-sharing sessions, hands-on workshops, and hackathons to reinforce practical application of new learning
- Quarterly product-update sessions led by the architects, summarizing what changed in the SAP integration stack and what it means for the standards

This enablement allows the CoE to guide the organization not only through technical migrations—such as moving from SAP PO to SAP Integration Suite—but also through strategic transformations that unlock new business capabilities.

Integration governance is a continuous journey, like clean core itself. Ongoing optimization ensures that integrations remain aligned with business objectives and current SAP capabilities over time. An interface implemented in 2022 might have a better replacement option today simply because SAP released the corresponding API or business event in the interim. The CoE's job is to surface those opportunities and prioritize them against new work. Key activities include the following:

- Periodic review of all integrations to identify those that don't fully adhere to clean core principles, with recommendations for reimplementation where possible
- Evaluation of whether interfaces previously implemented outside guidelines can now be aligned with clean core standards using newer SAP capabilities
- Updating interface requirements and mappings to reflect business process changes, technical improvements, and SAP product updates
- Using the CoE as the decision-making hub to determine when and how integration improvements should be implemented

Effective integration governance directly contributes to KPIs that measure the health, stability, and effectiveness of the integration landscape:

- **Upgrade stability**
  The number of APIs and interfaces adhering to clean core standards versus those that don't. This is the headline clean core metric for integration. Tracked quarterly from the SAP LeanIX repository, it should trend upward. A static or declining number means new projects are adding legacy patterns faster than get clean is removing them.
- **Operational risk**
  The extent to which interfaces are tracked and monitored using tools such as SAP Application Interface Framework and SAP Cloud ALM. Interfaces that fail silently are operational risk; closing the gap is straightforward governance work.
- **Innovation and adoption**
  Evaluation of the tools and technologies being used, including SAP Integration Suite, SAP Datasphere, SAP Business Data Cloud, and advanced capabilities such as advanced event mesh. This metric drives the SAP PI/SAP PO sunset migration explicitly and gives the CoE a quantitative basis for prioritizing it.
- **Agile decision-making and operational risk mitigation**
  Ensuring that each integration is linked to its respective business process. This linkage should be reflected in the following:
  - SAP Signavio for business process modeling
  - SAP LeanIX as the interface repository for enterprise and integration architecture
  - SAP Cloud ALM for project implementation tracking and operational oversight

This interconnected toolchain provides end-to-end visibility across business processes, enterprise architecture, and integration landscapes, supporting data-driven decision-making.

KPIs are only useful if they are reviewed. Most working CoEs hold a quarterly integration governance review to surface the indicators, discuss exceptions, and agree on corrective actions. The forum matters as much as the metrics; KPIs without a review meeting tend to drift.

**Sustaining Clean Integrations**

Like clean core, clean integrations are a journey rather than a destination. Sustained governance requires the following:

- Maintaining an interface repository with detailed definitions of senders, receivers, mappings, business objects, and processes
- Revisiting integrations periodically to implement clean solutions where earlier implementations were suboptimal
- Evaluating whether integration requirements can now adopt a standardized approach in line with the latest SAP capabilities

By combining CoE oversight, structured enablement, design guidelines, documentation, and a robust toolchain, organizations can ensure that integration governance is sustainable, transparent, and aligned with business and technical objectives. The governance loop closes when the CoE has a current strategy, a current set of guidelines, a current repository, a current set of KPIs, and a periodic review cadence. Without all five, the discipline degrades; with them, it sustains.

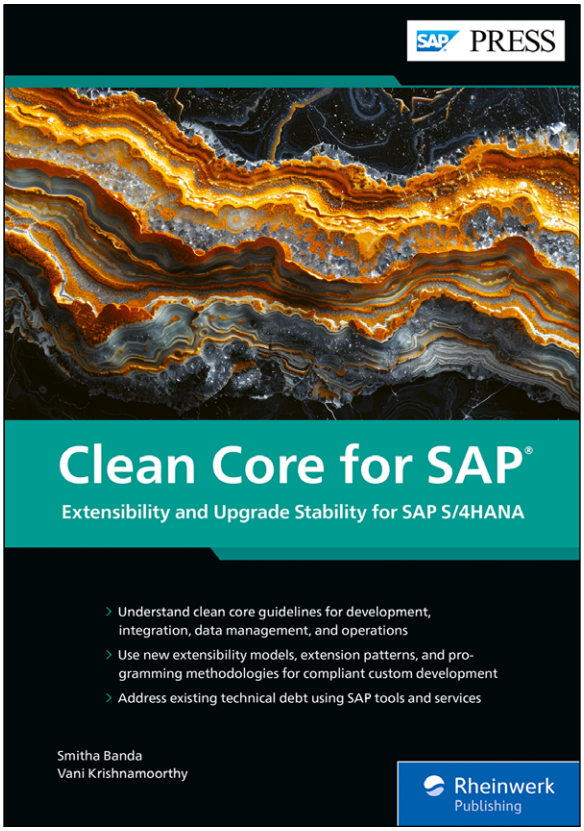
## 4.7 Summary

Clean integration is a critical pillar of the clean core strategy. An ERP that stays standard but is surrounded by a custom integration layer hasn't solved the problem—it has moved the problem.

The chapter set out how to avoid that outcome. The get clean approach addresses the integration debt that already exists, and keep clean prevents the next project from rebuilding it. A CoE holds the discipline in place across both, supported by the toolchain that makes each decision traceable from business requirement through to running interface. None of this is one project. It's an operating practice that has to run for as long as the SAP landscape does.

Most failed integration modernization programs do so because one of those four—the cleanup, the governance, the CoE, or the toolchain—was missing or treated as optional.

While clean integration ensures that data flows reliably and consistently across the enterprise, the value of that data depends entirely on its quality, accuracy, and governance, which is the focus of the next chapter.




Smitha Banda, Vani Krishnamoorthy

# Clean Core for SAP

## Extensibility and Upgrade Stability for SAP S/4HANA

- Understand clean core guidelines for development, integration, data management, and operations
- Use new extensibility models, patterns, and methodologies for compliant custom development
- Address existing technical debt using SAP tools and services

**www.sap-press.com/6302**



The Authors

Smitha Banda is a senior business development manager at SAP and the head of the SAP BTP practice for North America. Vani Krishnamoorthy is a chief architect at SAP, where she advises SAP S/4HANA customers on clean core strategy and implementation.

**ISBN 978-1-4932-2848-5 • 429 pages • 09/2026**

**E-book:** $74.99 • **Print book:** $79.95 • **Bundle:** $89.99

Rheinwerk Publishing