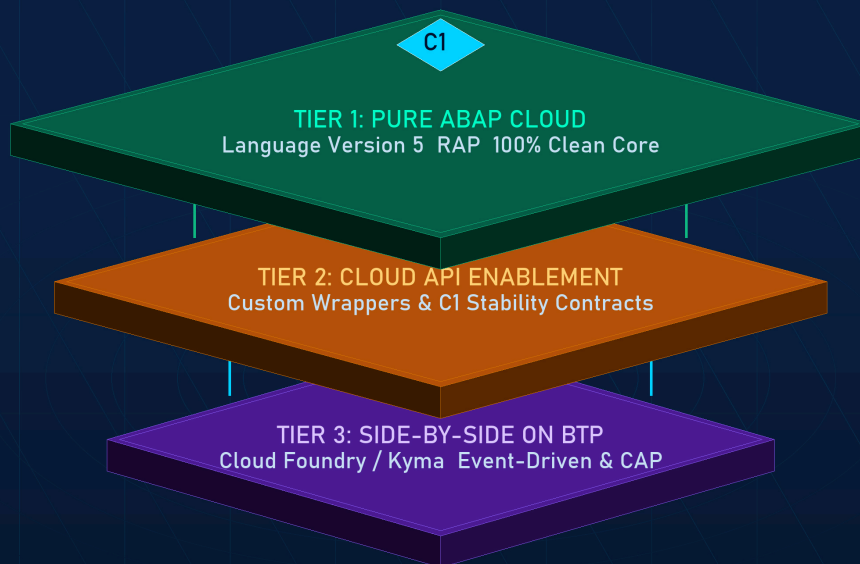


SAP S/4HANA & SAP BTP ENTERPRISE PLAYBOOK

THE DEFINITIVE GUIDE

ABAP CLOUD & CLEAN CORE

Architectural & Practical Guide to SAP BTP, Embedded
Steampunk, RAP, and the 3-Tier Extensibility Model



• RESTFUL ABAP (RAP)

• CDS VIEW ENTITIES V2

• EVENT-DRIVEN ARCHITECTURE

• SIDE-BY-SIDE PAAS (BTP)

• EMBEDDED STEAMPUNK

• ZERO DOWNTIME UPGRADE

AUTHOR

ANDREA ANTONIO PERRELLI

Senior SAP & Clean Core Enterprise Solutions Architect



THE DEFINITIVE GUIDE TO ABAP CLOUD & CLEAN CORE EXTENSIBILITY

From the SE80 Heritage to Modern Enterprise Software Engineering on SAP S/4HANA & SAP BTP

Author: Andrea Antonio Perrelli (10x SAP Certified Solution Architect)

Language: English (Official International Edition)

Edition: Master Enterprise Edition 2026



Complete Table of Contents

- [Introduction & Historical Evolution](#)
- Author Profile
- The R/3 and ECC 6.0 Era: Freedom That Became a Technological Prison
- The SAP HANA Revolution and the Arrival of S/4HANA
- The Inception of ABAP Cloud and the Clean Core Imperative
- [Module 1: Big Picture, BTP Architecture & Clean Core Extensibility](#)
- 1.1 Positioning ABAP Cloud in the SAP Ecosystem
- 1.2 From the 3-Tier Model to the Clean Core Maturity Model (Levels A–D)
- 1.3 Code Governance, Language Versions and Released APIs
- [Module 2: Integrated Development Environment \(IDE Eclipse & ADT\)](#)
- 2.1 ADT Architecture and Configuration
- 2.2 Modern Lifecycle Management, Execution, and Versioning
- [Module 3: Data Declarations and New Expressive Syntax](#)
- 3.1 Data Types and References
- 3.2 Constructor Operators and Expressive Syntax
- [Module 4: Operations, Control Logic and Messaging](#)
- 4.1 Modern Conditional Constructs
- 4.2 Cloud-Compliant Message Handling
- [Module 5: Advanced Internal Table Manipulation](#)
- 5.1 Table Categories and Direct Access
- 5.2 Modern Iterations and In-Place Modifications
- [Module 6: Dynamic Programming and RTTI \(Run Time Type Identification\)](#)
- 6.1 Run Time Type Services (RTTS)
- 6.2 Dereferencing and Dynamic Data Manipulation

- [Module 7: Exception Handling and Code Resilience](#)
- 7.1 Class-Based Exceptions
- 7.2 Creating Custom Exception Classes
- [Module 8: Data Modeling with Core Data Services \(CDS\)](#)
- 8.1 DDL and Modern CDS Architecture
- 8.2 Relationships, Entity Types, and Security
- [Module 9: Push-Down Data Processing: Modern ABAP SQL and AMDP](#)
- 9.1 Advanced Features of Modern ABAP SQL
- 9.2 AMDP (ABAP Managed Database Procedures)
- [Module 10: The Transactional Framework: RAP \(RESTful Application Programming Model\)](#)
- 10.1 Architecture and Behavior Definition (BDEF)
- 10.2 Business Behavior Components
- 10.3 Entity Manipulation Language (EML)
- [Module 11: OData Exposure and Consumption \(E2E Flow\)](#)
- 11.1 Service Definition and Service Binding
- 11.2 Publishing to SAP Fiori Launchpad (IAM - Identity & Access Management)
- 11.3 Outbound Integrations (Consuming External Services)
- 11.4 Event-Driven Architecture and Asynchronous Decoupling on SAP BTP
- [Module 12: Object-Oriented Programming and Cloud Design Patterns](#)
- 12.1 Moving Beyond Procedural Paradigms
- 12.2 Architectural Design Patterns in ABAP Cloud
- [Module 13: Cloud Extensibility and BAdIs](#)
- 13.1 From SE18/SE19 to Released BAdIs in ADT
- 13.2 Invoking and Implementing BAdIs
- [Module 14: Automated Testing, Quality and Background Processing](#)
- 14.1 Unit Testing with ABAP Unit
- 14.2 Test Double Framework
- 14.3 Background Processing in ABAP Cloud

THE DEFINITIVE GUIDE TO ABAP CLOUD & CLEAN CORE EXTENSIBILITY

From the R/3-ECC Monolith to Modern Transactional Architecture on SAP S/4HANA & SAP BTP

Author Profile

Andrea Antonio Perrelli is an Enterprise SAP Solution Architect and Cloud Engineer, renowned for his extensive expertise in transactional architecture and extensibility across the modern SAP ecosystem. He holds over **10 Official SAP Certifications**, including premier credentials: * **SAP Certified Solution Architect - SAP Business AI Platform** * **SAP Certified Technology Architect - SAP BTP** * **SAP Certified Development Associate - ABAP Cloud** * **SAP Certified Development Associate - SAP S/4HANA Cloud, Developer Extensibility** * **SAP Certified Associate - SAP Integration Suite & Cloud Architectures**

Leading mission-critical enterprise transformations for multinational industrial corporations, he designs resilient high-throughput transactional architectures, decoupled extensibility platforms adhering to **Clean Core** principles, and advanced Generative AI pipelines seamlessly interfaced with core ERP backbones.

This book represents the methodological, architectural, and practical distillation of years of hands-on enterprise execution: an unapologetically technical guide free from academic oversimplifications and sales slogans, engineered to transform traditional SAP developers and architects into masters of **ABAP Cloud**.

NOTE

****Editorial Notice on Trademarks and Intellectual Property:**** SAP, SAP S/4HANA, SAP BTP, SAP Fiori, and all other SAP product and service names mentioned herein are trademarks or registered trademarks of SAP SE in Germany and globally. This publication is an independent educational and architectural volume authored for engineering training purposes and is neither sponsored, affiliated with, nor endorsed by SAP SE.

From the Monolith to the Cloud Era: The Evolution of SAP Engineering

1. The R/3 and ECC 6.0 Era: Freedom That Became a Technological Prison

For over three decades, the **ABAP (Advanced Business Application Programming)** language has served as the bedrock of global enterprise computing. Originally conceived as a procedural reporting language in the 1980s (ABAP/4 in SAP R/2 and R/3), it empowered enterprises worldwide to tailor complex, differentiated business workflows.

However, the classic development paradigm in SAP R/3 and ECC 6.0 introduced a feature that proved as catastrophic to architectural longevity as it was powerful: **unlimited, ungoverned access to every internal layer of the system**. Inside traditional transactions such as [SE38](#) (ABAP Editor), [SE24](#) (Class Builder), [SE37](#) (Function Builder), and the unified [SE80](#) (Object Navigator), developers possessed virtually unbounded authority:

1. Direct Read and Write Operations on the Underlying Relational Database:

Developers could execute arbitrary native Open SQL statements ([SELECT](#) , [INSERT](#) , [UPDATE](#) , [DELETE](#)) directly against core standard dictionary tables (such as [VBAK](#) , [VBAP](#) , [MARA](#) , [BKPF](#)).

2. Accessing Tables Without Transactional Consistency Checks:

Custom interfaces and reports frequently modified accounting or logistics document statuses with an unconstrained [UPDATE BKPF SET BSTAT = '...'](#) , bypassing standard software validation routines, balance sheet reconciliations, and logical locking protocols ([ENQUEUE](#) / [DEQUEUE](#)).

3. Core Modifications to Standard Code:

By procuring "Access Keys" from SAP, development teams could alter standard program sources

(`SAPL...` , system includes), overwriting original routines written by SAP product engineers in Walldorf.

4. Fragmentation of Extensibility Mechanisms:

Over decades, a chaotic patchwork of extension technologies accumulated:

5. **Procedural User Exits:** Empty subroutine declarations (`FORM ... ENDFORM`) left inside standard routines (such as the notorious `MV45AFZZ` include for SD sales orders), where dozens of developers spliced thousands of lines of custom logic without namespace boundaries.
6. **Customer Exits (SMOD/CMOD):** Code blocks encapsulated in function modules prefixed with `EXIT_...`.
7. **Classic BAdIs (SE18/SE19):** Early Object-Oriented interfaces relying on macros and adapter wrappers.
8. **Enhancement Framework (Implicit & Explicit Points):** The capability to inject custom code at the beginning or end of virtually any standard function, method, or code line, rendering the landscape an opaque house of cards impossible to test deterministically.
9. **The Dynpro Legacy and the SAP GUI Client:**
Applications were architected exclusively for the monolithic **SAP GUI** desktop client, tightly bound to the procedural lifecycle of PBO (*Process Before Output*) and PAI (*Process After Input*) processors, rigid Table Controls, and screen-dependent synchronous messaging.

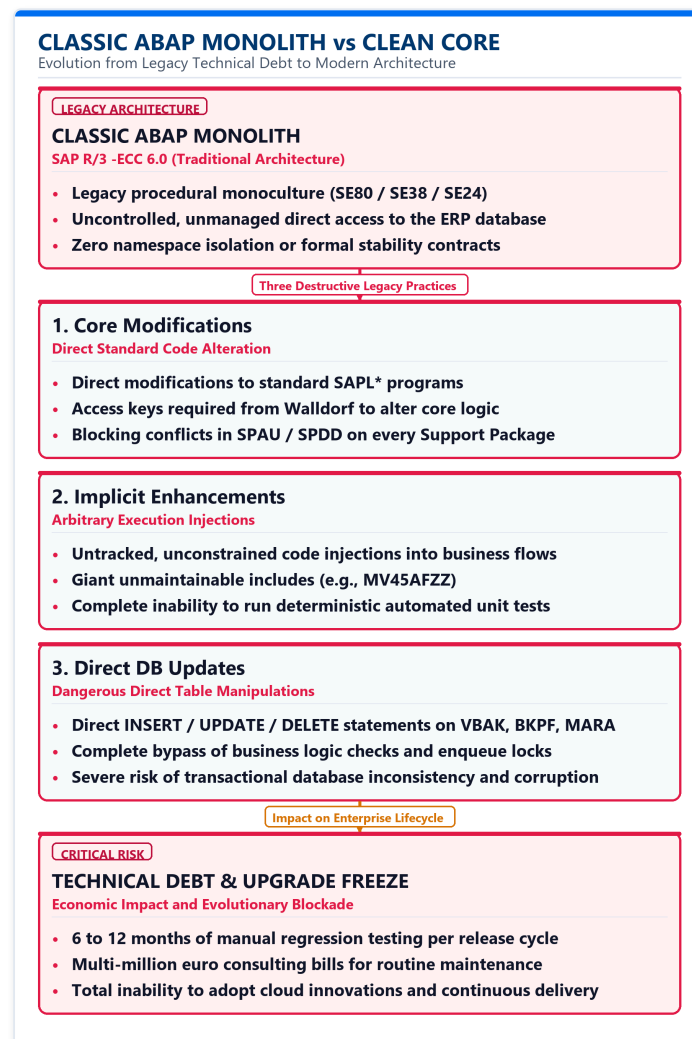


Figure: Classic ABAP Monolith vs Clean Core

The Catastrophic Consequence: The Upgrade Freeze

This architectural deregulation culminated in an industry-wide crisis: the **Upgrade Freeze**.

Every time SAP released a *Support Package* or *Enhancement Package (EhP)*, standard updates collided directly with custom modifications (requiring laborious conflict resolution via [SPAU](#) / [SPDD](#)). System upgrades routinely required 6 to 18 months of high-risk project cycles, millions of euros in consulting fees, and thousands of hours of manual regression testing.

2. The SAP HANA Revolution and the Arrival of S/4HANA

In 2011, SAP unveiled the in-memory **SAP HANA** database, followed in 2015 by the launch of **SAP S/4HANA**. The fundamental architectural paradigm underwent a seismic shift: * The database evolved from a passive storage layer into a massively parallel computational engine (columnar computing, on-the-fly aggregations, predictive vector operations). * **Core Data Services (CDS)** were introduced to push heavy data manipulation from the ABAP application layer directly down into the database engine (**Code Push-Down**). * The user experience was transformed: replacing the SAP GUI desktop client with **SAP Fiori**, architected around open web standards (HTML5, OData, SAPUI5, and RESTful protocols).

Despite these technological leaps, the vast majority of S/4HANA migrations opted for "Brownfield" conversions, carrying forward decades of technical debt and unreleased table accesses. With the rapid ascent of **Software-as-a-Service (SaaS)** delivery models—where cloud software is upgraded continuously and automatically by SAP—this monolithic approach became unsustainable.

3. The Inception of ABAP Cloud and the Clean Core Imperative

To allow enterprises to consume continuous cloud updates without breaking critical business operations, SAP completely reinvented its language foundation and software architecture, formalizing the **Clean Core** paradigm.

⚠️ IMPORTANT

****Definition of Clean Core:**** A Clean Core ERP environment is one where standard software remains pristine, unmodified, and strictly decoupled from custom extensions. All customizations, business logic modifications, and integrations occur exclusively via ****stable, contractually committed, and upgrade-proof interfaces (Released APIs)****.

To operationalize this vision across enterprise landscapes, SAP released **SAP BTP, ABAP Environment** (codenamed **Steampunk**) in 2018, bringing a modern, secure ABAP runtime into the public cloud PaaS. In 2022, this exact runtime was integrated directly into the core S/4HANA stack (available from S/4HANA Cloud and S/4HANA 2022 On-Premise / Private Cloud onward) under the designation **Embedded Steampunk**.

The unified model spanning both environments is officially named **ABAP Cloud**:

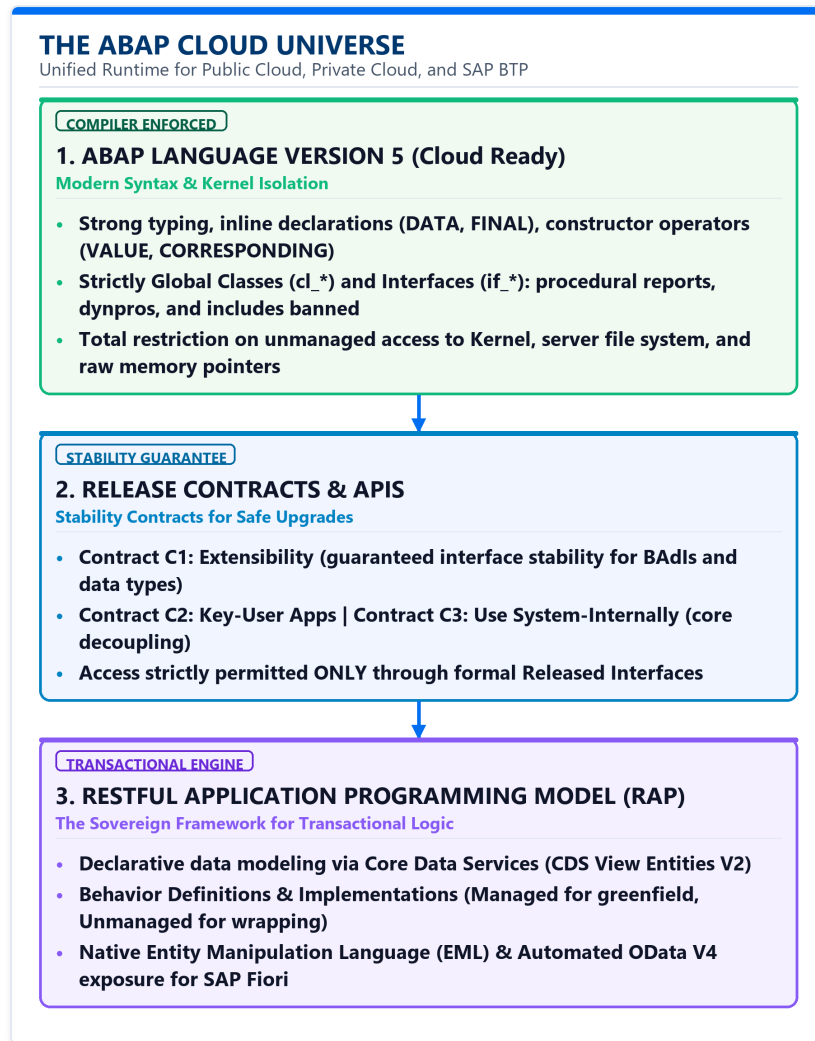


Figure: The ABAP Cloud Universe

Enterprise Clean Core & SAP BTP Reference Architecture

To visualize how the PaaS public cloud domain interfaces with the core ERP transactional backbone, the reference topology below follows official **SAP BTP Solution Diagram** conventions:

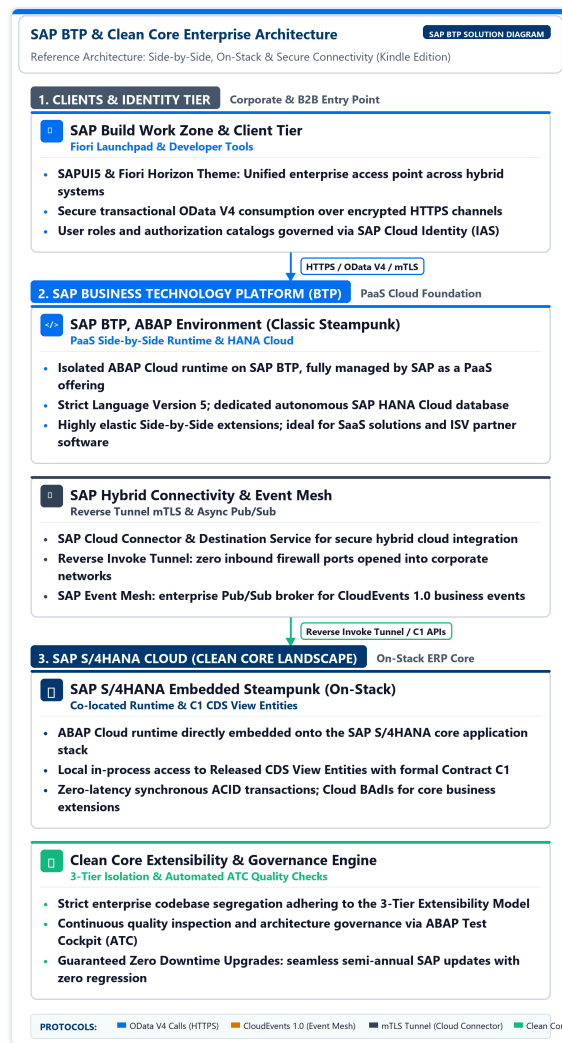


Figure: SAP BTP & Clean Core Enterprise Architecture - Solution Diagram

Core Principles of ABAP Cloud

1. A Decisive Break with Procedural Legacy:

Classic executable reports ([PROGRAM](#)), procedural function modules ([FUNCTION MODULE](#)), includes, and dynpro screen transactions ([CALL SCREEN](#)) are **strictly forbidden and will not compile**. The sole first-class citizens are modern Object-Oriented artifacts: **Global Classes** and **Global Interfaces**.

2. Access Exclusively Governed by Release Contracts:

Querying arbitrary standard tables is prohibited. Reading or mutating business entities requires consuming a **Released CDS View Entity** or a **Released RAP Business Object** backed by a formal stability contract ([C1](#)).

3. Exclusive Development in Eclipse (ADT):

Transaction [SE80](#) belongs to history. The only supported development environment for ABAP Cloud is **ABAP Development Tools (ADT)** hosted in Eclipse, featuring cloud-compiler syntax checking, refactoring tooling, native Git integration, and distributed asynchronous debugging.

4. Standardized Transactional Engineering (RAP):

Ad-hoc transactional frameworks and uncoordinated database updates are obsolete. The mandatory standard framework for business data manipulation is the **RESTful Application Programming Model**

(RAP), ensuring transactional ACID integrity, automated *Draft Handling*, logical locking, managed persistence, and native OData V4 / SAP Fiori support.

5. **Zero Downtime Upgrades:**

All code written in strict compliance with ABAP Cloud is mathematically guaranteed **never to break across underlying OS upgrades, HANA database updates, or semi-annual SAP application releases.**

Upgrades become seamless technical events, empowering the enterprise to innovate continuously.

Syllabus Structure and Learning Path

This volume methodically covers the **14 Specialization Modules** defined in the official curriculum. Every subject is delivered through: * **Deep architectural fundamentals and design rationale.** * **"Before & After" comparisons:** analyzing legacy ECC workarounds versus modern ABAP Cloud implementations. * **Production-grade ABAP code**, fully compilable under Language Version 5 rules. * **ABAP Test Cockpit (ATC) governance rules** and Clean Core quality metrics.

Whether you are architecting an S/4HANA Cloud migration, engineering side-by-side solutions on SAP BTP Steampunk, or elevating your engineering organization to modern cloud standards, this playbook provides your definitive architectural roadmap.

MODULE 1: Big Picture, BTP Architecture and Clean Core Extensibility

1.1 Positioning ABAP Cloud in the SAP Ecosystem

In contemporary enterprise landscapes, **ABAP Cloud** is not merely an updated dialect of ABAP syntax; it represents **the unified development model across the entire modern SAP product portfolio**, spanning all deployment topologies: from multi-tenant Public Cloud to hyperscaler Private Cloud and On-Premise deployments.

To master ABAP Cloud, architects must examine its two execution environments:

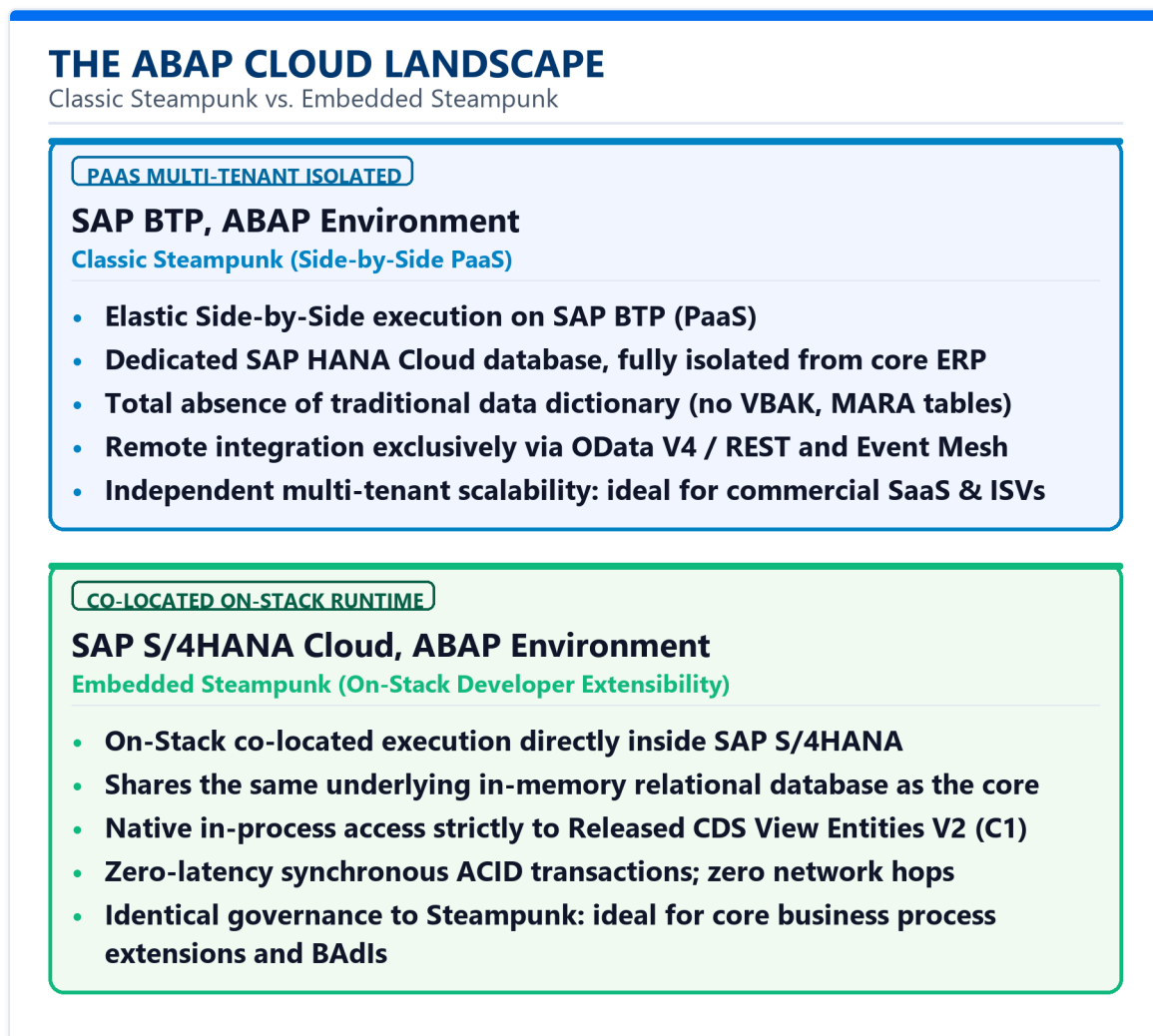


Figure: The ABAP Cloud Universe: Classic Steampunk vs Embedded Steampunk

1. SAP BTP, ABAP Environment (Classic Steampunk)

Launched in 2018 as a Platform-as-a-Service (PaaS) offering on **SAP Business Technology Platform**, *Classic Steampunk* delivers a fully managed ABAP runtime operated by SAP. * **Isolated Architecture:** Instances reside within autonomous BTP subaccounts powered by dedicated SAP HANA Cloud database instances, physically and logically decoupled from corporate core ERP systems. * **Absence of Native ERP Dictionary:** Classic Steampunk contains no standard ERP tables (`VBAK` , `MARA` , `KNA1`). All business data from S/4HANA or third-party platforms is consumed via remote endpoints (OData, REST) or asynchronous event streams (SAP Event Mesh). * **Primary Use Cases:** * Multi-tenant commercial SaaS products built by Independent Software Vendors (ISVs) and SAP partners. * Elastic enterprise extensions requiring complete lifecycle decoupling from the ERP backend. * B2B customer/vendor portals accessible to external users who lack nominal S/4HANA named licenses.

2. SAP S/4HANA Cloud, ABAP Environment (Embedded Steampunk)

Beginning with S/4HANA Cloud 2208 and S/4HANA 2022 (for Private Cloud and On-Premise), SAP embedded the exact same *Steampunk* runtime **directly into the S/4HANA core application stack**. * **Co-located Transactional Execution:** ABAP Cloud code executes on the identical application server and accesses the same in-memory SAP HANA database hosting enterprise operational data. * **Zero-Latency In-Process Execution:** Engineers execute high-performance queries and manipulate entities without incurring network serialization, JSON marshalling, or HTTP overhead. * **Guaranteed Compiler Enforcement:** Despite physical co-location, the ABAP Cloud compiler strictly blocks unauthorized access to private standard code. Interaction occurs exclusively through SAP-provided **Released APIs**. * **Primary Use Cases:** * High-volume transactional extensions (e.g., complex dynamic pricing, real-time customs processing). * Business consistency validations requiring synchronous ACID transactional integrity. * Core standard workflow enhancements executed through Cloud BAdI implementations.

3. ABAP Cloud as the Unified Runtime

ABAP Cloud permanently unifies SAP development across all platforms: * **SAP S/4HANA Cloud Public Edition** (100% SAP-managed multi-tenant cloud), * **SAP S/4HANA Cloud Private Edition** (hyperscaler-hosted single-tenant cloud), * **SAP S/4HANA On-Premise** (version 2022 and beyond), * **SAP BTP, ABAP Environment**.

The programming model, syntax rules, Eclipse ADT tooling, and RAP persistence frameworks are 100% identical. An artifact developed in ABAP Cloud for S/4HANA Private Cloud can be deployed or repurposed on SAP BTP or Public Cloud with negligible refactoring, future-proofing enterprise software investments for decades.

Physical and Logical Topology of ABAP Cloud Runtimes

The following diagram, structured according to official **SAP BTP Solution Diagram** specifications, contrasts the deployment boundaries, physical topology, and integration channels of *Classic Steampunk* and *Embedded Steampunk*:

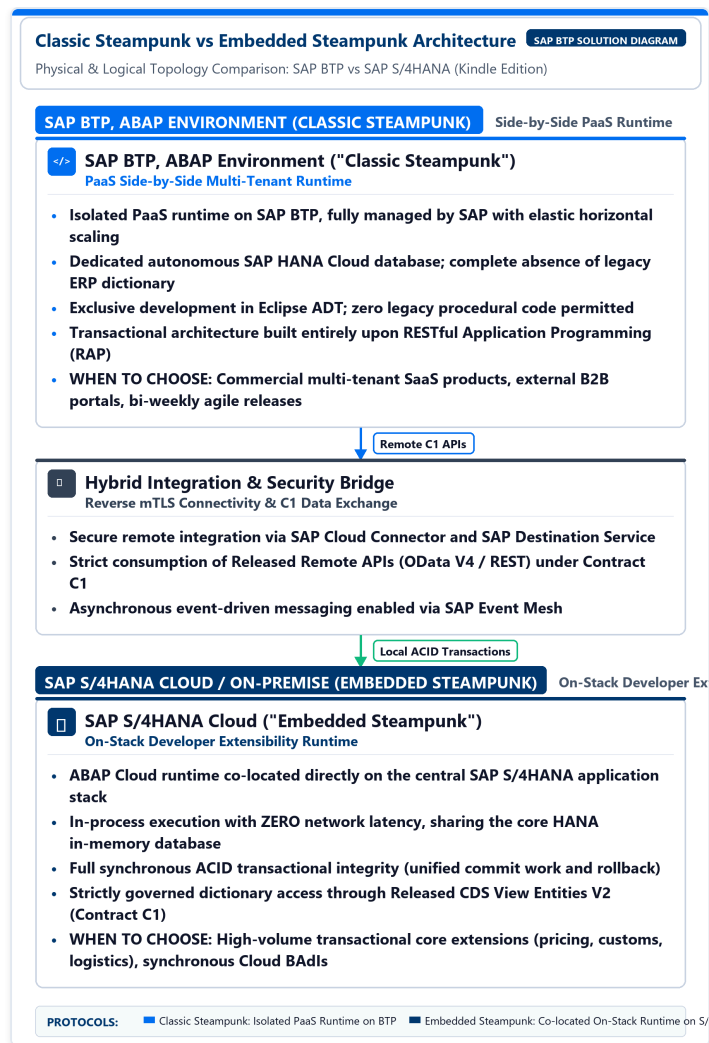


Figure: Classic Steampunk vs Embedded Steampunk Architecture - Solution Diagram

1.2 From the 3-Tier Model to the Clean Core Maturity Model (Levels A–D)

To govern the transition of legacy systems toward a Clean Core, SAP formalized two complementary methodologies: the **3-Tier Extensibility Model** and the **4-Level Clean Core Maturity Model (Levels A–D)**.

The 3-Tier Extensibility Model

In landscapes where modern and legacy code coexist (notably S/4HANA Private Cloud and On-Premise), rewriting millions of lines of historic code overnight is impossible. The 3-Tier model establishes clear boundaries:

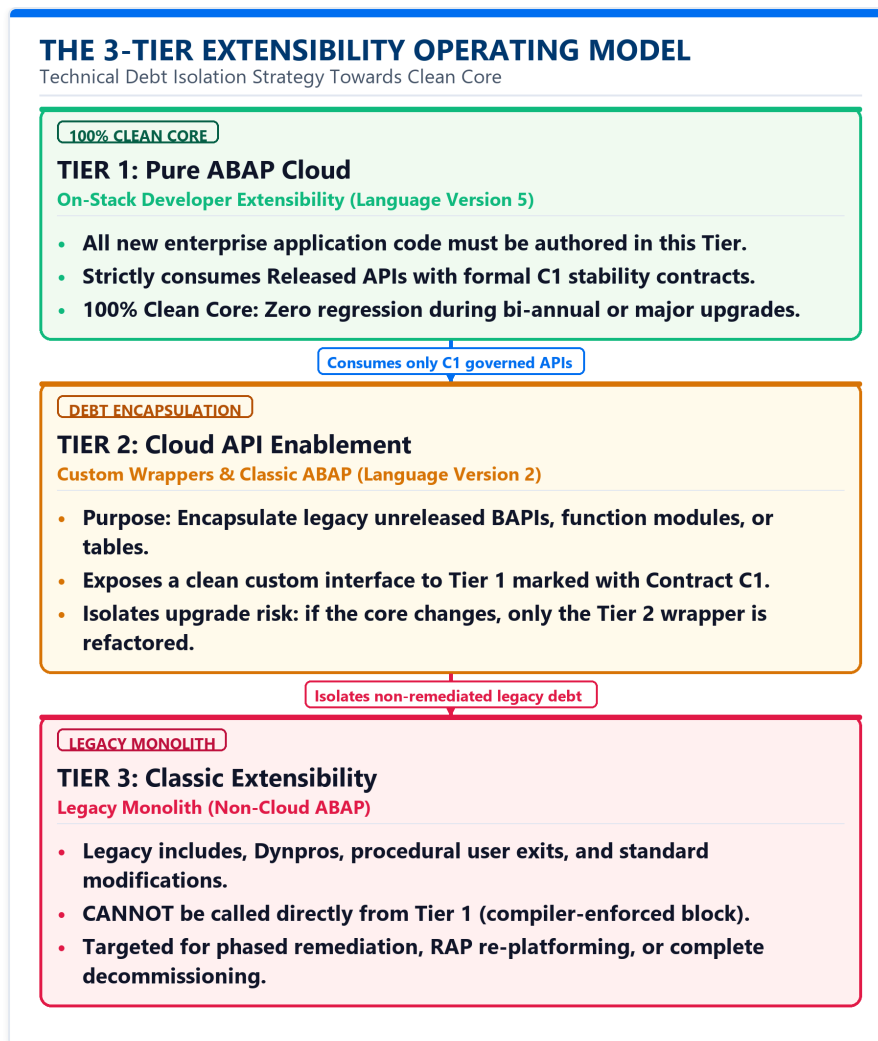


Figure: The 3-Tier Extensibility Operating Model

1. Tier 1 (Pure ABAP Cloud):

All new code must be developed here. Software packages are configured with **ABAP Language Version 5**. Consuming unreleased objects is strictly prohibited. This layer is entirely upgrade-safe.

2. Tier 2 (Custom Wrappers / Controlled Mitigation):

If Tier 1 business logic requires standard SAP functionality or data for which Walldorf has not yet published an official Released API, developers construct an isolated wrapper class in *Classic ABAP* (Language Version 2) in Tier 2. This wrapper encapsulates the legacy call (e.g., an unreleased BAPI or table) and **exposes a clean custom interface to Tier 1 explicitly released under Release Contract C1**. If a future SAP upgrade affects the legacy object, only the Tier 2 wrapper requires maintenance, leaving the Tier 1 application pristine.

3. Tier 3 (Classic Extensibility):

Encompasses historical custom code developed over previous decades. Configured in Classic ABAP, it cannot be called from Tier 1 and must be methodically refactored, modernized, or decommissioned.

Governance Architecture of Release Contracts (Tiers 1, 2, and 3)

The reference architecture diagram below details software component boundaries, execution layers, and integration with SAP BTP:



Figure: Clean Core 3-Tier Extensibility Architecture - Solution Diagram

The Clean Core Maturity Model (Levels A–D)

While the 3-Tier model organizes code placement, the **Clean Core Maturity Model (Levels A–D)** establishes quantitative and qualitative metrics used by Enterprise Architects and the ABAP Test Cockpit to measure organizational Clean Core compliance:

CLEAN CORE MATURITY MODEL (LEVELS A -D)				
Compliance Criteria, Upgrade Risk Analysis, and Governance				
Level	Name	Architectural Criteria	Upgrade Risk	Prescribed Action
Level A	Clean	ABAP Cloud (on-stack/BTP). Strictly C1 Released APIs. RAP framework.	Zero (Zero regression)	Ideal target state. No remediation required.
Level B	Compliant Classic	Classic ABAP with modern OO. Stable consolidated APIs. Zero core mods.	Low (Governed)	Monitor SAP changelog; refactor to Level A when C1 APIs release.
Level C	Conditionally Clean	Consumes unreleased internal SAP objects without modifying standard.	Medium-High	Map technical debt, add pre-upgrade checks, wrap in Tier 2.
Level D	Unclean	Critical violations: core mods, direct INSERT/UPDATE on VBAK, kernel calls.	Critical (Severe break)	Mandatory remediation. Block upgrades and re-platform to RAP/Tier 2.

Figure: Clean Core Maturity Model (Levels A - D)

1.3 Code Governance, Language Versions and Released APIs

The robustness of ABAP Cloud stems from the fact that architectural rules are not discretionary guidelines; they are **strictly enforced by the ABAP Kernel compiler**.

1. ABAP Language Version 5 Rules (*ABAP for Cloud Development*)

When a software package or repository object is assigned to **Language Version 5** (*ABAP for Cloud Development*), the compiler enforces strict boundaries:

RESTRICTIONS OF ABAP LANGUAGE VERSION 5

Enforced Rules of the Cloud-Ready Kernel

COMPILER BLOCKED

STRICTLY FORBIDDEN CONSTRUCTS

Language Version 5 -Non-Compilable by Kernel

- ❑ Legacy display/terminal I/O statements (WRITE, ULINE, SKIP)
- ❑ Creating or calling Dynpro screens (CALL SCREEN, PBO/PAI)
- ❑ Procedural program artifacts (REPORT, PROGRAM, FUNCTION MODULE)
- ❑ Direct filesystem access on the server (OPEN DATASET)
- ❑ Direct table modifications on standard ERP entities (UPDATE/INSERT on VBAK)
- ❑ Consuming unreleased SAP objects (tables, internal classes, classic BADIs)
- ❑ Operating system command executions (CALL 'SYSTEM')

NATIVE CLOUD

PERMITTED CLOUD STANDARDS

Language Version 5 -The Only Valid Production Constructs

- ❑ Exclusively Global Classes (cl_*) and Global Interfaces (if_*)
- ❑ Exclusive Data Modeling with CDS View Entities (V2)
- ❑ RAP Transactional Framework & Entity Manipulation Language (EML)
- ❑ Exclusive consumption of Released APIs with C1/C2/C3 contracts
- ❑ External communications strictly via HTTP Client and Comm. Arrangements
- ❑ Modern expressive syntax (VALUE, CORRESPONDING, FINAL)
- ❑ Automated testing via ABAP Unit and native Test Double Framework

Figure: ABAP Language Version 5 Restrictions

If a developer attempts to compile the following code in an ABAP Cloud method:

● ● ● Anti-Pattern: Direct DB Access and Procedural Calls
ABAP CLOUD (FORBIDDEN)

```

1  " CODE EXAMPLE NON-COMPILABLE IN ABAP CLOUD:
2  SELECT * FROM vbak INTO TABLE @DATA(lt_orders). " ERROR: VBAK has no release contract!
3  UPDATE vbak SET netwr = 1000 WHERE vbeln = '0000000010'. " FATAL ERROR: Direct table updates forbidden!
4  CALL FUNCTION 'BAPI_SALESORDER_CREATEFROMDAT2'. " ERROR: BAPI not released for ABAP Cloud!

```

Listing: Anti-Pattern: Direct DB Access and Procedural Calls

The compiler blocks activation immediately with critical syntax errors:

Object "VBAK" is not released for use in ABAP for Cloud Development (Language Version 5).

Direct modification of SAP tables is strictly prohibited.

2. Release Contracts

An SAP repository object becomes consumable in ABAP Cloud only after being formally **released** by SAP product engineering under a stability contract:

Contract **C1** : Contract for Extensibility

The foundational contract for enterprise developers and architects: * **Stability Commitment:** SAP guarantees that interface signatures, parameter types, method semantics, and transactional behaviors remain **permanently backward-compatible**. * **Zero Post-Upgrade Regressions:** If SAP evolves a **C1** interface, it adds optional parameters without modifying existing logic or removing mandatory fields. * **Scope:** Cloud-Ready BAdIs, Released CDS View Entities, and Released System Utilities (e.g., `c1_abap_regex`, JSON libraries).

Contract **C2** : Use in Key User Apps

Assigned to entities consumable not only by developers in ADT, but also by functional consultants and key users inside SAP Fiori **Key User Extensibility** applications (e.g., *Custom Fields*, *Custom Logic*).

Contract **C3** : Use System-Internally

Employed for architectural decoupling across internal SAP standard software components, ensuring modularity within standard solutions.

RELEASE CONTRACT COMPLIANCE MATRIX

Official SAP Stability Contracts for Released APIs

Contract	Target Audience	Stability Guarantee	Primary Scope
C1	Developer (ADT)	Formal signature & runtime behavior stability. Zero break post-upgrade.	Cloud-Ready BAdIs, CDS Interfaces, Utility Classes
C2	Key User (Fiori)	Safe consumption within no-code / low-code Fiori applications.	Fiori Custom Fields, Custom Logic, Custom Queries
C3	SAP Internal	Guaranteed cross-component decoupling across standard modules.	Internal data exchange & transactions between S/4 modules

Figure: Release Contract Compliance Matrix

3. Automated Governance via ABAP Test Cockpit (ATC)

To prevent non-compliant code from reaching Quality Assurance or Production environments, governance is automated through the **ABAP Test Cockpit (ATC)**.

In Clean Core enterprises, ATC executes as a **mandatory release gate on Transport Requests (CTS / gCTS)**.

Clean Core Check Variants:

- **ABAP_CLOUD_READINESS** : Verifies that all objects in the transport belong to Language Version 5 packages and have zero dependencies on unreleased artifacts.
- **CL_CI_TEST_EXTENSIBILITY** : Validates that custom extensions adhere strictly to **C1** and **C2** release contracts.
- **Security & Authorization Checks**: Ensures comprehensive Data Control Language (DCL) enforcement and eliminates unvalidated dynamic code execution.

If a transport request contains even a single **Priority 1** Cloud Readiness error, **the system physically rejects the transport release**, guaranteeing that technical debt cannot infiltrate production landscapes.

In the next module, we explore the primary workbench where modern development takes place: **Eclipse IDE with ABAP Development Tools (ADT)**.

 **END OF FREE SAMPLE**

You've Completed the Free Architectural Sample

Unlock All 14 Modules, 40 Retina Blueprints & 61 Production Code Listings

What awaits you in the Complete Enterprise Edition:

- **Modules 02–05:** Eclipse ADT, Modern ABAP Cloud Syntax & Internal Table Optimizations
- **Modules 06–07:** RTTI/RTTS Dynamic Services & Class-Based Resilience Patterns
- **Modules 08–09:** Advanced CDS View Entities & Push-Down SQL / AMDP Procedures
- **Modules 10–11:** Complete RAP Transactional Engine (EML, Managed/Unmanaged, Draft) & IAM
- **Modules 12–14:** Enterprise Cloud Design Patterns, Released BAdIs & ABAP Unit Tests
- **40 Standalone 300 DPI Retina Diagrams** ready for board architecture slides
- **Full GitLab Repository Access** with all 36 patterns and offline editable assets

 **Get the Complete Enterprise Suite (€9.80 • 80% OFF)**

 **Buy on Amazon: Paperback (\$19.99) & Kindle (\$12.99)**

 [Visit the Live Interactive Architecture Hub & Developer Tools](#)

Authored by **Andrea Antonio Perrelli**

SAP Certified Enterprise Solution Architect • 10x SAP Certified • Business AI & SAP BTP Specialist