ABAP CLEAN CORE

---

1. INTRODUCTION TO CLEAN CORE

1.1 What Is Clean Core?

Clean Core means keeping your **SAP S/4HANA system as standard as possible**, avoiding modifications in SAP's core, and shifting custom logic to **extensions and side-by-side developments** using **modern ABAP and BTP** tools.

It ensures:

- Easier upgrades (less testing, fewer conflicts)
- Better stability and performance
- Seamless integration with cloud innovations
- Lower TCO (Total Cost of Ownership)

1.2 Why Is It Important?

Area	Without Clean Core	With Clean Core
Upgrade Effort	High due to core mods	Minimal
Innovation Adoption	Slower	Faster
Maintenance	Complex	Simplified
Governance	Poor	Standardized
Scalability	Limited	Flexible (BTP Ready)

1.3 SAP's Strategy

SAP enforces clean core via:

- **ABAP Cloud** (Steampunk)
- **Key User Extensibility**
- **Developer Extensibility**
- **SAP BTP & RAP (RESTful ABAP Programming Model)**
- **Upgrade-safe APIs (Released Objects Only)**

---

2. FOUNDATIONS OF ABAP CLEAN CORE

2.1 Key Principles

1. **No Modifications in SAP Objects**
2. **Use Released APIs Only**
3. **Separate Custom Code from Standard**
4. **Follow RAP & BTP extensibility**
5. **Test, Validate, and Document Extensions**
6. **Adopt Clean ABAP Guidelines**

2.2 SAP's Three Extensibility Types

Type	Who Uses It	Toolset	Scope
In-App (Key User)	Functional users	Fiori App "Custom Fields & Logic"	UI & Data Model
On-Stack (Developer Extensibility)	ABAP Developers	ABAP Cloud, ADT, RAP	Upgrade-safe, on S/4 stack
Side-by-Side (BTP)	Integration teams	SAP BTP, CAP, APIs	Decoupled Cloud extensions

---

2.3 Clean ABAP Guidelines (Core Rules)

- Use **modern ABAP syntax** (`DATA(...)`, `LOOP AT ... INTO DATA(...)`)
- Avoid **obsolete constructs** (e.g., `PERFORM`, `TABLES`, `CHAIN`)
- Prefer **interfaces, classes, and factory patterns**
- Apply **dependency injection** for testability
- Ensure **unit test coverage**
- Follow **naming standards and code readability**

Reference: SAP Clean ABAP Style Guide

---

3. TRANSITIONING FROM CLASSIC ABAP TO CLEAN CORE ABAP

3.1 ECC vs S/4HANA Mindset Shift

Area	ECC ABAP Mindset	S/4HANA Clean Core Mindset
Enhancements	User exits, implicit enhancements	BAdIs, Custom Fields & Logic
Z-tables everywhere	Reuse released CDS views	Extension of standard entities
Direct DB access	CDS/AMDP-based access	Released CDS, RAP models

Area	ECC ABAP Mindset	S/4HANA Clean Core Mindset
Hard-coded UI	SAP Fiori	OData, RAP, UI annotations
Code in core	ABAP Cloud class	Released APIs only

3.2 Tools for Clean Core Migration

Tool	Purpose
ATC (ABAP Test Cockpit)	Scan for non-clean core violations
ABAP Cloud Readiness Check	Detect unreleased object usage
SCI (Code Inspector)	Code quality & syntax compliance
SPAU/SPDD	Handle modifications
SAP Cloud ALM	Manage clean-core operations

---

4. BUILDING CLEAN EXTENSIONS

4.1 Scenario: Adding Business Logic to Standard

Old Way: Implicit enhancement inside SAP include
Clean Way:

- Use BAdI (Business Add-In)
- Or use RAP Behavior Implementation Extension
- Or Key User "Custom Logic" app

Example (Custom Validation via BAdI):

```
CLASS zcl_sd_salesorder_validate DEFINITION
  PUBLIC FINAL CREATE PUBLIC.
  PUBLIC SECTION.
    INTERFACES if_ex_salesorder_check.
ENDCLASS.

CLASS zcl_sd_salesorder_validate IMPLEMENTATION.
  METHOD if_ex_salesorder_check~check_salesorder.
    IF is_salesorder-net_value > 1000000.
      RAISE EXCEPTION TYPE cx_salesorder_limit.
    ENDIF.
  ENDMETHOD.
ENDCLASS.
```

4.2 Scenario: Adding a Custom Field

1. Go to *Custom Fields & Logic App*
2. Add field (e.g., `ZDISCOUNT_REASON`)
3. Enable for:
    - UI

    - CDS
    - API
4. Extend logic in “Custom Logic” tab (Released APIs only)

4.3 Scenario: Side-by-Side Extension

- Build new app on **SAP BTP (CAP or RAP)**
- Consume S/4HANA data via OData APIs
- Deploy custom logic without touching S/4 core

Example:
Extend Order Approval process on BTP using CAP → Consume `/sap/opu/odata/sap/API_SALES_ORDER_SRV`.

---

5. REAL-WORLD DYNAMIC CLEAN CORE SCENARIOS

5.1 Case 1: Dynamic Business Rules

Use **BRF+ or BTP Rule Service** instead of hardcoding.

5.2 Case 2: Dynamic Reporting

Use **CDS Views + Annotations** instead of classical ALV reports.

5.3 Case 3: Dynamic Workflow

Use **Flexible Workflow** (Fiori App) — no custom workflow code.

5.4 Case 4: Custom Pricing Logic

Use **Pricing BAdIs** in SD, don’t alter `RV61AFZA`.

5.5 Case 5: Custom Data Migration

Use **BTP ETL or CDS Views with released entities** instead of direct `INSERT` into SAP tables.

---

6. CLEAN CORE GOVERNANCE FRAMEWORK

6.1 Roles and Responsibilities

Role	Clean Core Contribution
ABAP SME	Ensure code is clean, documented, tested
Solution Architect	Approve clean extension design
Functional Consultant	Use key user extensibility
BTP Developer	Build side-by-side apps
QA/Tester	Verify upgrade safety and functional parity

6.2 Clean Core Governance Checklist

- No SAP object modifications
- All custom code in Z namespace
- Uses released APIs/CDS views only
- Unit tests implemented
- Follows Clean ABAP syntax
- Performance validated via ST05/SQLM
- Documented in solution repository

---

7. FROM DEVELOPER TO CLEAN CORE SME

7.1 Skill Roadmap

Level	Focus Areas	Tools/Frameworks
Beginner	Clean ABAP syntax, ATC, CDS views	ADT, Eclipse
Intermediate	RAP, BAdIs, Key User extensibility	ABAP Cloud, RAP tools
Advanced	BTP extensions, governance, testing	CAP, GitHub, ALM
SME	Strategy, architecture, design review	SAP Cloud ALM, Steampunk

---

7.2 Continuous Learning Resources

Type	Resource	URL
SAP Learning	ABAP Cloud & Clean Core	learning.sap.com
GitHub	Clean ABAP Guidelines	github.com
Blog	SAP Clean Core Blog Series	community.sap.com
YouTube	*SAP Developers Channel (RAP, BTP, ABAP Cloud)*	youtube.com/sap

---

8. KEY TAKEAWAYS

- Clean Core is a **discipline, not a tool**.
- Always **separate custom code from SAP code**.

- Use **released APIs, RAP, and BTP**.
- Keep **upgrade safety** at the heart of development.
- Be a **Clean Core Ambassador** — educate peers.

---

9. SME INSIGHTS AND REAL PROJECT GUIDANCE

1. **Always document extensibility points used.**
2. **Propose side-by-side before on-stack extensions.**
3. **Use ATC mandatory checks in CI/CD pipeline.**
4. **Maintain an Extension Registry (Z-table or ALM).**
5. **Conduct periodic Clean Core audits.**

---

10. CLEAN CORE – FUTURE OUTLOOK

- By 2030, SAP aims all S/4HANA on **ABAP Cloud model**.
- Future extensions will be **hybrid RAP + BTP**.
- Governance will move to **SAP Cloud ALM with AI-based code quality monitoring**.