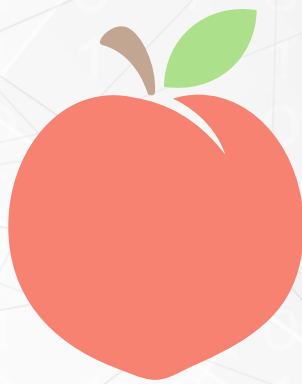




CS Peach

Powered by Cremencing.com

FREE CHEAT SHEET CREMENCING.COM

# BAPI vs RFC vs IDoc, A Decision Guide



CS Peach  
cremencing.com  
2026 Edition

"Three tools that look like rivals but are actually three layers of the same system. Here is how to choose."

## THE COMPARISON

Six criteria that actually separate them

| CRITERIA       | BAPI                                       | RFC                                       | IDOC                                   |
|----------------|--------------------------------------------|-------------------------------------------|----------------------------------------|
| What it is     | Released, stable business API on an object | Any function module callable from outside | Structured document for async exchange |
| Sync / async   | Synchronous                                | Both                                      | Asynchronous                           |
| Upgrade-stable | Guaranteed                                 | Not guaranteed                            | Very stable                            |
| Error handling | Structured RETURN (BAPIRET2)               | You build it — exceptions                 | Status records, full audit             |
| Volume / retry | One call — not for mass                    | Depends how you build it                  | Built for volume                       |
| Best for       | Create/read/update an object now           | Custom or remote logic, non-SAP           | EDI, partners, high volume             |

**The relationship most developers miss:** a BAPI is an RFC-enabled function module — just a released, stable, business-object one. And IDocs are usually moved *using* tRFC underneath. They are not three rivals competing for the same job. They are three layers of one integration stack.

## WHICH ONE SHOULD I USE?

Follow the questions

Do you need the result back immediately, in the same step?

YES

NO

Standard SAP business object?

High volume or partner / EDI?

BAPI  
else customsync RFC

IDoc  
else async tRFC

Calling non-SAP or custom remote logic?

↓

RFC  
plain RFC-enabledFM, sync or async

### RULE OF THUMB

- BAPI — do it now, standard object.
- RFC — call custom or remote logic.
- IDoc — send it later, in volume, with audit.

| BAPI<br><i>the commit trap</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | RFC<br><i>the destination trap</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | IDoc<br><i>the partner-profile trap</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>WHERE IT SAVED THE PROJECT</p> <p>A team hand-coded order creation with direct table inserts — broke every upgrade. Switching to <b>BAPI_SALESORDER_CREATEFROMDAT2</b> made it upgrade-proof overnight.</p> <p>THREE ERRORS + FIX</p> <p><b>Forgot the COMMIT</b></p> <p><b>Fix:</b> call BAPI_TRANSACTION_COMMIT after, or nothing saves.</p> <p><b>Ignored the RETURN table</b></p> <p><b>Fix:</b> check BAPIRET2fortypeE/A before assuming success.</p> <p><b>Used an unreleased FM</b></p> <p><b>Fix:</b> only use FMs marked released in the BAPI explorer.</p> | <p>WHERE IT SAVED THE PROJECT</p> <p>A pricing engine lived in a separate system. A <b>synchronous RFC</b> meant no data duplication and always-current logic — a cross-system JOIN was impossible.</p> <p>THREE ERRORS + FIX</p> <p><b>Wrong / missing destination</b></p> <p><b>Fix:</b> setup and test in SM59 before writing code.</p> <p><b>No comms error handling</b></p> <p><b>Fix:</b> catch SYSTEM_FAILURE and COMMUNICATION_FAILURE.</p> <p><b>Missing S_RFC authorization</b></p> <p><b>Fix:</b> calling user needs S_RFC — check with SU53.</p> | <p>WHERE IT SAVED THE PROJECT</p> <p>50,000 deliveries a day to a partner. A sync call would collapse. <b>IDocs queued, processed in background</b>, and failures reprocessed in WE19 without losing a record.</p> <p>THREE ERRORS + FIX</p> <p><b>Partner profile missing (WE20)</b></p> <p><b>Fix:</b> no profile=status29. Configure WE20 first.</p> <p><b>Stuck in status 30</b></p> <p><b>Fix:</b> status30 waits for RSEOUT00 — schedule or push it.</p> <p><b>Segment version mismatch</b></p> <p><b>Fix:</b> agree basic type and version pre-go-live.</p> |

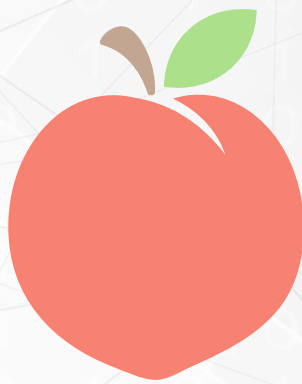


Laeeq Siddique

Free resource by Cremencing Solutions cremencing.com  
CS Peach-Al consultant for SAP ABAP teams



CREMENCING  
SOLUTIONS



CS Peach

Powered by Cremencing.com  
CH ABAP Peach

FREE CHEAT SHEET CREMENCING.COM

# Extending SAP Without Breaking Clean Core

EXTRA  
BONUS



CS Peach  
cremencing.com  
2026 Edition

"Choosing the wrong enhancement type locks you out of clean core. This page is how you choose right — User Exit vs BAdI, the Enhancement Framework, and where integration is heading in 2026."

## User Exit vs BAdI

the choice that dates your code

### What a User Exit is

Old enhancement points inside SAP programs — [SMOD/CMOD](#). Single implementation only. Being phased out.

### What a BAdI is

Business Add-In — **object-oriented, multiple implementations**, filter-enabled. The modern replacement. Managed in [SE18/SE19](#).

### Which to choose

**Always prefer a BAdI** if one exists. Use a User Exit only if no BAdI or enhancement point is offered.

"I check for a BAdI first, then an enhancement spot, and only fall back to a classic exit if nothing modern exists."

### Kernel vs classic BAdI

New (kernel) BAdIs are faster and support inheritance. Classic BAdIs are older but still common. Know both.

### Why it matters

**User Exits and modifications block clean core.** BAdIs and released enhancement spots are the upgrade-safe way to extend.

## Enhancement Framework

five rules to survive upgrades

### 1. Explicit over implicit

Use **explicit enhancement points** SAP provides. Implicit ones (start/end of a method or FM) are a fallback.

### 2. Never modify — enhance

A modification changes SAP code and needs SPAU at every upgrade. **An enhancement sits alongside** and survives.

### 3. Spots, not source edits

Enhancement spots and sections are the intended tools — [ENHANCEMENT...ENDENHANCEMENT](#). Cleaner than touching source.

### 4. Document every one

Enhancements are easy to forget and hard to find at upgrade time. **Keep a register** of every one and why.

### 5. Released for cloud?

In S/4HANA, only **released enhancement options** work in a clean-core context. Verify before you build.

## Mistakes that cost days

learned the hard way

### Sync for high volume

A sync call for 10,000 records ties up a work process and times out. **Volume = async / IDoc, always.**

### Hand-coding a BAPI's job

Direct table inserts for orders or materials. **Breaks every upgrade.** The BAPI exists — use it.

### No idempotency on retries

Reprocessing creates duplicates with no duplicate check. **Design for safe reprocessing.**

### Ignoring RETURN / status

Assuming success without checking BAPIRET2 or IDoc status. **Silent failures are the worst kind.** BAPIs need explicit commit; mixing with your own updates in one LUW half-saves data. Keep LUW boundaries clean.

### Forgetting the COMMIT

## Integration in 2026

where SAP is actually heading

### OData is the new front door

For UI and external APIs, **OData via RAP** increasingly replaces custom RFCs. Know how to expose a CDS view as OData.

### Integration Suite / CPI

Cloud middleware on BTP. Where partner and cloud integration now lives — **iFlows instead of point-to-point RFC**.

### Released APIs on API Hub

SAP publishes released APIs at [api.sap.com](#). **Check here before building anything custom.**

### Events & Event Mesh

Publish/subscribe instead of polling. The modern async pattern beyond IDocs.

### But IDoc is not dead

**IDocs still run EDI and high-volume B2B everywhere.** Nobody rips out working IDoc interfaces. Classic skills still pay.

## THE 30-SECOND ANSWER — SAY THIS IN INTERVIEWS

- **Need it now, standard object?** "The released BAPI, then BAPI\_TRANSACTION\_COMMIT."
- **Custom or remote logic?** "A sync RFC to a destination set up in SM59, with exception handling."
- **High volume or partner EDI?** "IDocs — async, reprocessable in WE19, with status tracking."
- **New UI or external API?** "In 2026, a CDS view exposed as OData through RAP, not a custom RFC."

## KEY TRANSACTIONS TO MEMORISE

- **BAPI** — [BAPI explorer](#) · [SE37](#) · [SW01](#)
- **RFC** — [SM59](#) [SE37](#) [ST22](#)
- **IDoc** — [WE20](#) · [WE19](#) · [WE02](#) · [BD87](#) · [WE60](#)
- **Enhancements** — [SE18/SE19](#) · [SMOD/CMOD](#) · [SPAU](#)

## THE ONE PRINCIPLE THAT TIES IT TOGETHER

- **Use the most standard, most released, most upgrade-safe option available — every time.**
- Released BAPI over hand-coded logic. BAdI over User Exit. Enhancement over modification. Released API over custom RFC.
- This single habit separates a developer whose code survives upgrades from one whose code breaks every time.
- **In 2026 it has a name — clean core — and it is the most valuable habit you can build.**



Laeeq Siddique

Free resource by Cremencing Solutions [cremencing.com](#)  
CS Peach-Alconsultant for SAP ABAP teams



CREMENCING  
SOLUTIONS