Business Objects in ABAP Cloud

A Practical Introduction to RAP

SAP Inside Track NRW • 20 min

By Victor Fibechola Okpanachi

Why RAP? The Foundation of ABAP Cloud

ABAP Cloud is the future. RAP is how you build everything in it.

CDS Views

Define your data model — what fields exist, what the key is, associations to other entities.

Behavior Definition

Define what you can DO — create, update, delete, validations, determinations, actions.

OData Automatic

RAP generates your OData service. No Gateway coding. No manual registration.

Fiori Ready

Consume directly in Fiori Elements. The annotations on your CDS drive the UI automatically.

Sample application

Sales Quotation Approval App.

Where do you start?

- A sales rep creates a quotation — customer, product, quantity, price
- The system calculates the total and sets status to Open — automatically
- The manager clicks Approve or Reject in Fiori
- That's it. But underneath: a full RAP Business Object.

In 20 Minutes You Will Be Able To:

01

Explain what a Business Object is

Data Definition + Behavior — the two parts every BO has

02

Read a Behavior Definition

Open any BDEF in Eclipse and understand every line

03

Know why Projection exists

The difference between BO Interface and BO Projection

04

Use EML in ABAP code

MODIFY ENTITIES, READ ENTITIES, COMMIT ENTITIES

05

Analyze a BO you didn't write

Navigate it in Eclipse like you built it yourself

PART 1

What is a Business Object?

"It's not a table. It's not a class. It's the combination of both — with rules."

A Business Object Has 2 Main Parts

① Data Definition (CDS View)

ABAP

```
define root view entity
  ZI_SalesQuotTP
  as select from zsales_quot
{
  key QuotID,
      CustomerID,
      ProductID,
      Quantity,
      UnitPrice,
      TotalAmount,
      Status
}
```

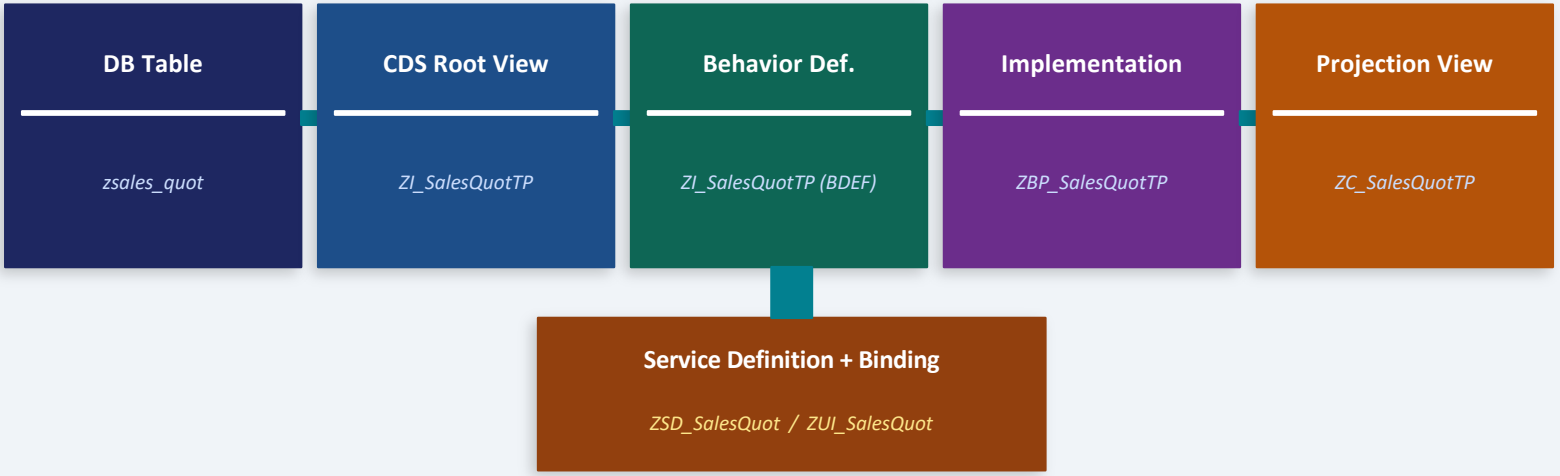
② Behavior Definition (BDEF)

ABAP

```
managed implementation
  unique;

define behavior for
  ZI_SalesQuotTP {

  create; update; delete;

  validation validateCustomer
  determination calculateTotal
  action approveQuotation
}
```

"The CDS View defines WHAT exists. The BDEF defines what you can DO."

The Business Object at a Glance



EML Test Class: ZCL_QUOT_EML_TEST ← talk to your BO from ABAP code

Behavior Definition

ZI_SalesQuotTP.bdef (open this in Eclipse)

ABAP

```
managed implementation
  unique;

define behavior for
  ZI_SalesQuotTP
  persistent table zsales_quot {

  field ( readonly ) QuotID;

  create; update; delete;

  validation validateCustomer
    on save.create.update;

  determination calculateTotal
    on modify : fields Quantity,
                       UnitPrice;

  determination setInitialStatus
    on modify : create;

  action approveQuotation;
  action rejectQuotation;
}
```

managed implementation unique;

SAP handles Create / Update / Delete automatically. No boilerplate.

field (readonly) QuotID;

Users cannot change the key field. Enforced at framework level.

validation validateCustomer

Fires on Save. If CustomerID is empty → error message, save blocked.

determination calculateTotal

Fires when Quantity or UnitPrice changes. Auto-recalculates TotalAmount.

action approveQuotation

A button in Fiori. Calls your ABAP method. Sets Status = 'A'.

L I V E D E M O

Build the Sales Quotation BO

01 CDS Root View

ZI_SalesQuotTP — the shape of the BO

02 Behavior Definition

managed, fields, validations, determinations, actions

03 Behavior Implementation

ZBP_SalesQuotTP — the actual ABAP logic

04 Projection View

ZC_SalesQuotTP — what Fiori sees

05 Service + Launch Fiori

Create a quotation, watch the total calculate, approve it

06 EML Test Class

MODIFY ENTITIES / COMMIT ENTITIES — live in code

A Sales Quotation BO

All 8 objects — in Eclipse, right now

Database Table

zsales_quot

Stores quotation data: customer, product, qty, price, status

CDS Root View

ZI_SalesQuotTP

Data Definition — fields, key, associations

Behavior Definition

ZI_SalesQuotTP

managed, create/update/delete, validation, determination, action

Behavior Implementation

ZBP_SalesQuotTP

ABAP class — validateCustomer, calculateTotal, approve, reject

Projection View

ZC_SalesQuotTP

Fiori-facing view with @UI annotations

Service Definition

ZSD_SalesQuot

Exposes the projection as an OData service

Service Binding

ZUI_SalesQuot_O4

UI service binding — launch Fiori Preview from here

EML Test Class

ZCL_QUOT_EML_TEST

Test your BO with MODIFY ENTITIES / READ ENTITIES

Validations — Guard the Data

Validations run on Save. If they fail, the save is blocked and the user sees your error message.

ABAP

```
METHOD validateCustomer.
  READ ENTITIES OF ZI_SalesQuotTP
    ENTITY SalesQuot
    FIELDS ( CustomerID )
    WITH CORRESPONDING #( keys )
    RESULT DATA(lt_quot).

  LOOP AT lt_quot INTO DATA(ls_quot).
    IF ls_quot-CustomerID IS INITIAL.
      APPEND VALUE #(
        %tky = ls_quot-%tky
        %msg = new_message_with_text(
          severity = if_abap_behv_message=>
                     severity-error
          text = 'CustomerID is required' )
      ) TO reported-salesquot.
    ENDIF.
  ENDLOOP.
ENDMETHOD.
```

What RAP does with this:

- User clicks Save in Fiori
- RAP calls validateCustomer
- CustomerID empty → error added
- Save is blocked, message shown

✓ Two validations in our BO:

validateCustomer — CustomerID must not be empty
validateDates — BeginDate must be before EndDate

Determinations — Automatic Calculations

Determinations fire automatically when data changes. The user never triggers them — RAP does.

ABAP

```
METHOD calculateTotal.
  READ ENTITIES OF ZI_SalesQuotTP
    ENTITY SalesQuot
    FIELDS ( Quantity UnitPrice )
    WITH CORRESPONDING #( keys )
    RESULT DATA(lt_quot).

  MODIFY ENTITIES OF ZI_SalesQuotTP
    ENTITY SalesQuot
    UPDATE FIELDS ( TotalAmount )
    WITH VALUE #(
      ( %tky        = <quot>-%tky
TotalAmount  = <quot>-Quantity
                    * <quot>-UnitPrice
      ) ).
ENDMETHOD.
```

Trigger: on modify fields

- User changes Quantity → fires
- User changes UnitPrice → fires
- TotalAmount updates automatically
- No button needed — fully automatic

Two determinations:

setInitialStatus — sets Status = 'O' on create
calculateTotal — fires when Quantity or UnitPrice changes ← LIVE DEMO

Actions — Business Operations as Buttons

Actions are special operations — not Create/Update/Delete. They appear as buttons in Fiori automatically.

ABAP

```
METHOD approveQuotation.
  READ ENTITIES OF ZI_SalesQuotTP
    ENTITY SalesQuot
    FIELDS ( Status )
    WITH CORRESPONDING #( keys )
    RESULT DATA(lt_quot).

  LOOP AT lt_quot INTO DATA(ls).
*
    IF ls-Status = 'A'.
      CONTINUE. "already approved"
    ENDIF.

    MODIFY ENTITIES OF ZI_SalesQuotTP
      ENTITY SalesQuot
      UPDATE FIELDS ( Status )
      WITH VALUE #( (
        %tky   = ls-%tky
        Status = 'A' ) ).
  ENDLOOP.
ENDMETHOD.
```

approveQuotation

Sets Status = 'A' (Approved)
Stamps who approved and when

rejectQuotation

Sets Status = 'R' (Rejected)
Requires a rejection reason

Not create/update/delete — custom business logic wired to a Fiori button

Managed vs. Unmanaged — Which Do You Pick?

Managed ✓ (our choice)

- SAP handles Create / Update / Delete
- Less code — you focus on business logic
- Built-in draft handling
- Best for new tables you control
- → What we use in this demo

ABAP

```
managed implementation unique;

" SAP writes the CUD for you.
" You only code validations,
" determinations, actions.
```

Unmanaged (advanced)

- You write Create / Update / Delete yourself
- More code — total control
- Required for legacy / external tables
- Used for complex multi-entity scenarios
- → Use when managed isn't enough

ABAP

```
unmanaged implementation
  in class ZBP_LegacyTP unique;

" You implement every operation.
" Full control, full responsibility.
```

PART 2

Projection & Interface: The Safety Layer

"The BO Interface is the engine. The Projection is the steering wheel."

BO Interface vs. BO Projection — Side by Side

BO Interface (ZI_SalesQuotTP)

ABAP

```
define root view entity
  ZI_SalesQuotTP
  as select from zsales_quot
{
  key QuotID,
      CustomerID,
      ProductID,
      Quantity,
      UnitPrice,
      TotalAmount,
      Status
}

" No @UI annotations here.
" Clean. No Fiori knowledge.
```

BO Projection (ZC_SalesQuotTP)

ABAP

```
@UI.headerInfo: {
  typeName: 'Quotation' }

define root view entity
  ZC_SalesQuotTP
  as projection on
     ZI_SalesQuotTP
{
  @UI.lineItem: [{ position: 10 }]
  key QuotID,
  @UI.lineItem: [{ position: 20 }]
      CustomerID,
  @UI.lineItem: [{ position: 50 }]
      TotalAmount,
  @UI.lineItem: [{ position: 60,
    criticality: #Status }]
      Status
}
```

PART 3 — Analyzing a BO You Didn't Write

Eclipse skill: read any BO in under 60 seconds

The 3-step method — works on any RAP BO in any system:

1

Open the Business Object Navigator

Right-click the root CDS view → Open With → Business Object Navigator
See the full tree: entities, behaviors, validations, actions — at a glance.

```ABAP
Right-click ZI_SalesQuotTP → Open
With →
Business Object Navigator
```

2

Read the Behavior Definition first

Open the .bdef file. In 30 seconds you know: Is it managed or unmanaged? What operations are allowed? What validations and determinations exist?

```ABAP
managed implementation unique;
create; update; delete;
validation validateCustomer
determination calculateTotal
action approveQuotation
```

3

Jump to the Implementation

Ctrl+Click any validation or action name in the BDEF → jumps directly to the ABAP method. No searching, no guessing.

```ABAP
Ctrl+Click 'validateCustomer'
→ Opens ZBP_SalesQuotTP
→ Directly to the method body
```

PART 4 — EML: Talk to Your BO from ABAP Code

Entity Manipulation Language — not SQL, not function modules

ABAP

```
METHOD create_test_quotation.
  MODIFY ENTITIES OF ZI_SalesQuotTP
    ENTITY SalesQuot
    CREATE FIELDS ( CustomerID ProductID
                    Quantity   UnitPrice )
    WITH VALUE #( (
      CustomerID = 'CUST-001'
      ProductID  = 'PROD-42'
      Quantity   = 5
      UnitPrice  = 199 ) )
  REPORTED DATA(lt_reported)
  FAILED   DATA(lt_failed)
  MAPPED   DATA(lt_mapped).


  COMMIT ENTITIES.

  " Check what was created:
  DATA(lv_new_id) =
    lt_mapped-salesquot[ 1 ]-%pid.
ENDMETHOD.
```

MODIFY ENTITIES

Like INSERT but goes through your BO — triggers validations, determinations automatically.

REPORTED / FAILED / MAPPED

FAILED = something went wrong. MAPPED = new keys. REPORTED = user messages.

COMMIT ENTITIES

NOT the same as COMMIT WORK. Always use COMMIT ENTITIES in RAP context. Never mix.

Why it matters

Write unit tests without Fiori. Trigger the BO from batch jobs, APIs, other classes.

What We Built Today

✓ Business Object

CDS Root View + Behavior Definition + Implementation

✓ Managed BO

SAP handles CUD — you write validations, determinations, actions

✓ Validation

validateCustomer — blocks Save if CustomerID is empty

✓ Determination

calculateTotal — auto-fires when Quantity or UnitPrice changes

✓ Action

approveQuotation — button in Fiori, pure ABAP logic

✓ Projection vs Interface

BO Interface = engine. Projection = steering wheel. Keep them separate.

✓ BO Navigator in Eclipse

Right-click root CDS → Open With → Business Object Navigator

✓ EML

MODIFY ENTITIES + COMMIT ENTITIES — talk to your BO from ABAP

Your RAP Journey Starts Now

Try it: SAP BTP Trial → ABAP Environment → Eclipse ADT

Learn: learning.sap.com → search 'RAP' → free learning journey

Ask: community.sap.com/topics/abap — tag: rap

Slides + code available after the talk

Let's Connect

LinkedIn
Victor Fibechola Okpanachi

YouTube
sapvic