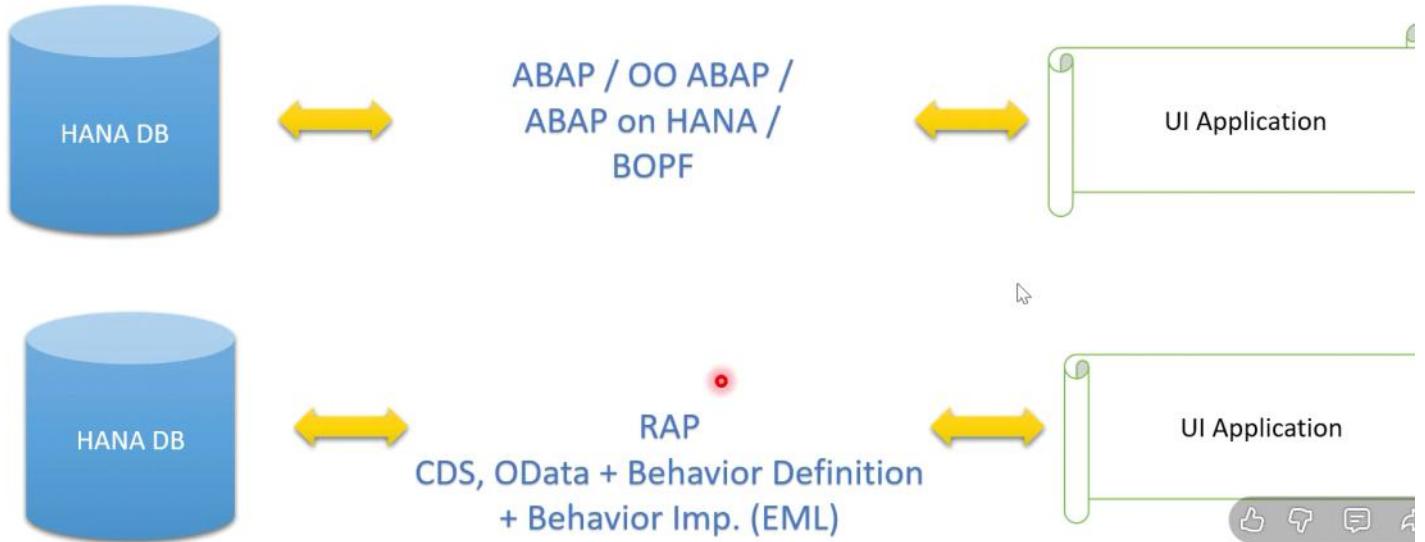


## ✓ About RAP



## ✓ About RAP

### CDS Views (Interface & Consumption)



## ✓ About RAP

### CDS Views (Interface & Consumption)

- Interface – core business logic & root
- Consumption – end view which consumable by OData/UI5 Apps.

### Metadata Extension (MDE)

- Extends the CDS view with CDS annotations.
- That is not specified in the DDL source code.
- @Metadata.allowExtensions: true in the CDS entity.
- App generates with Fiori elements.

## ✓ About RAP

### Behavior Definition

- Default CUD operation with mapping fields.
- 1 to 1 mapping.

### Behavior Implementation

- EML – Entity Manipulation Language

### Service Definition

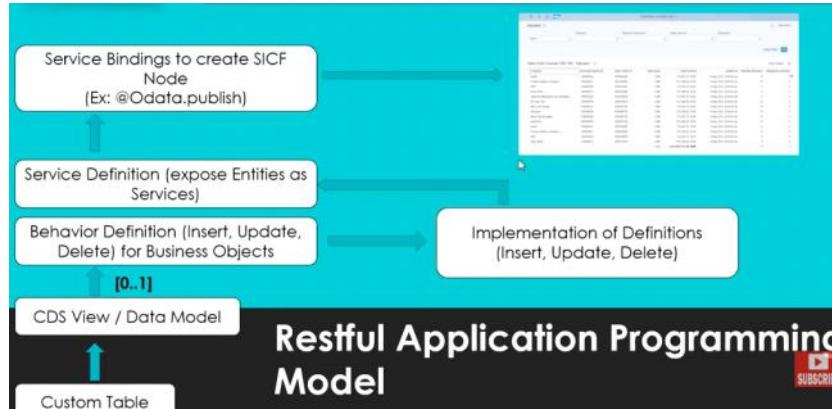
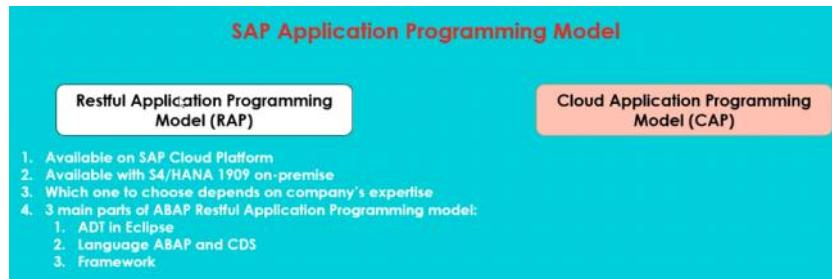
- Expose CDS View

### Service Binding

- Service creation

## Intro 2

24 November 2025 12:30



CODEINMINS

SAP Application Programming Model

1. Create Database Table
2. Create Interface View
3. Create Consumption View
4. Metadata Extension File
5. Define Business Objects – Define ROOT view
6. Create Behavior Definition for Interface
7. Create Behavior Definition for Consumption view
8. Create Service Definition – Expose Consumption View name
9. Service Bindings – Right click on Service Definition and then create Service Bindings

Table Creation

```

[TRL] YSTD_TABLE × [TRL] YRAP_STUDENT × [TRL] YRAP_C_STUDENT × [TRL] YRAP_C_STUDENT
1 @EndUserText.label : 'Stundet Table'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table ystd_table {
7
8   key client      : abap.clnt not null;
9   key id          : sysuuid_x16 not null;
10  firstname       : abap.char(20);
11  lastname        : abap.char(20);
12  age             : abap.numc(4);
13  course          : abap.char(20);
14  courseduration : abap.numc(4);
15  status           : abap_boolean;
16  gender           : abap.char(1);
17  dob              : abap.datas;
18
19 }

```

```

[TRL] YSTD_TABLE × [TRL] YRAP_STUDENT × [TRL] YRAP_C_STUDENT × [TRL] YRAP_C_STUDENT
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Interface View Entity For Student Table'
4 @Metadata.ignorePropagatedAnnotations: true
5 define root view entity Yrap_Student
6   as select from ystd_table
7 {
8   key id          as Id,
9     firstname    as Firstname,
10    lastname     as Lastname,
11    age          as Age,
12    course       as Course,
13    courseduration as Courseduration,
14    status        as Status,
15    gender        as Gender,
16    dob          as Dob
17 }
18

```

Below IF add Metadata Allow Annotation then Can add Metadata

```

[TRL] YSTD_TABLE × [TRL] YRAP_STUDENT × [TRL] YRAP_C_STUDENT × [TRL] YRAP_C_STUDENT
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Consumption View For Studnent'
4 @Metadata.ignorePropagatedAnnotations: true
5
6 @Metadata.allowExtensions: true
7
8 define root view entity YRAP_c_student
9   as projection on Yrap_Student as Student
10 {
11   @EndUserText.label: 'Student Id'
12   key Id,
13   @EndUserText.label: 'First Name'
14   Firstname,
15   @EndUserText.label: 'Last name'
16   Lastname,
17   @EndUserText.label: 'Age'
18   Age,
19   @EndUserText.label: 'Course'
20   Course,
21   @EndUserText.label: 'Course Duration'
22   Courseduration,
23   @EndUserText.label: 'Status'
24   Status,
25   @EndUserText.label: 'Gender'
26   Gender,
27   @EndUserText.label: 'Date Of Birth'
28   Dob
29 }
30

```

```

1 [TRL] YSTD_TABLE      D [TRL] YRAP_STUDENT      D [TRL] YRAP_C_STUDENT      E [TRL] YRAP_C_STUDENT ×
2 @Metadata.layer: #PARTNER
2 annotate entity YRAP_c_student with
3 {
4   @UI: { lineItem: [{ position: 10, label: 'Student Id' }], 
5         identification: [{ position: 10,label: 'Student Id' }] }
6   Id;
7   @UI: { lineItem: [{ position: 20, label: 'First Name' }], 
8         identification: [{ position: 20,label: 'First Name' }] }
9   Firstname;
10  @UI: { lineItem: [{ position: 30, label: 'Last Name' }], 
11        identification: [{ position: 30,label: 'Last Name' }] }
12  Lastname;
13  @UI: { lineItem: [{ position: 40, label: 'Age' }], 
14        identification: [{ position: 40,label: 'Age' }] }
15  Age;
16  @UI: { lineItem: [{ position: 50, label: 'Course' }], 
17        identification: [{ position: 50,label: 'Course' }] }
18  Course;
19  @UI: { lineItem: [{ position: 60, label: 'Course Duration' }], 
20        identification: [{ position: 60,label: 'Course Duration' }] }
21  Courseduration;
22  @UI: { lineItem: [{ position: 70, label: 'Status' }], 
23        identification: [{ position: 70,label: 'Status' }] }
24  Status;
25  @UI: { lineItem: [{ position: 80, label: 'Gender' }], 
26        identification: [{ position: 80,label: 'Gender' }] }
27  Gender;
28  @UI: { lineItem: [{ position: 90, label: 'DOB' }], 
29        identification: [{ position: 90,label: 'DOB' }] }
30  Dob;
31
32 }

```

CODEINMINS

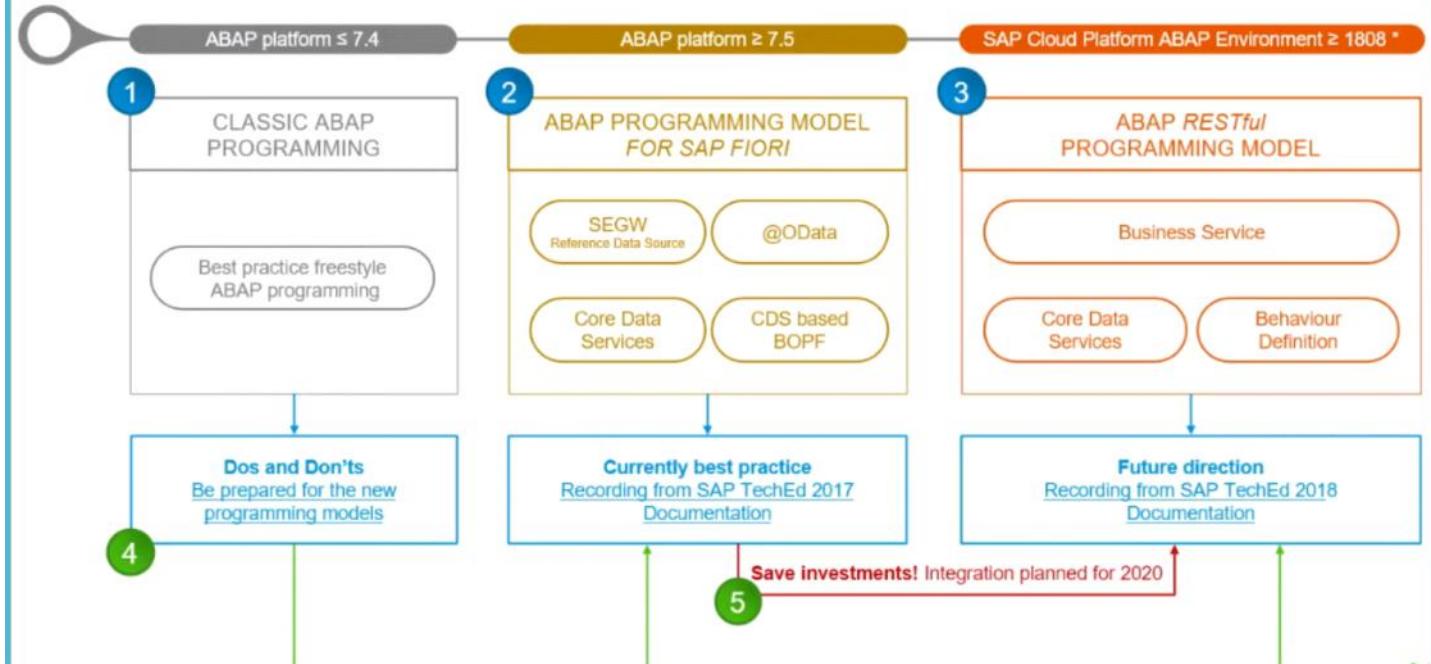
SAP Application Programming Model

- 1. Create Database Table**
- 2. Create Interface View**
- 3. Create Consumption View**
- 4. Metadata Extension File**
- 5. Define Business Objects – Define ROOT view**
- 6. Create Behavior Definition for Interface**
- 7. Create Behavior Definition for Consumption view**
- 8. Create Service Definition – Expose Consumption View name**
- 9. Service Bindings – Right click on Service Definition and then create Service Bindings**

#### What is Define Business Objects :

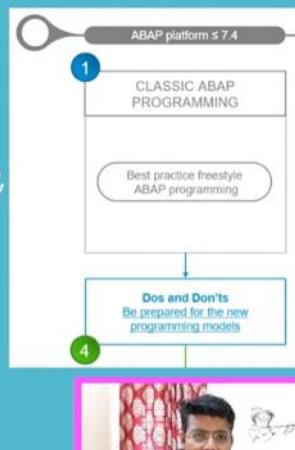
- 1. In ABAP Restful Application Programming Model a Business Object is collection of tree Nodes (Entity)... where each Node (Entity) represents one CDS View.**
- 2. There is always one Entity which will represent as ROOT.**
- 3. Business Object will have Behavior Definition. There can be only one Behavior Definition per Entity. This Behavior Definition contains different operations on Business Object ex. Create, Update, Delete etc.**
- 4. Business Object then can have Behavior Implementations**

## Evolution of the ABAP programming model



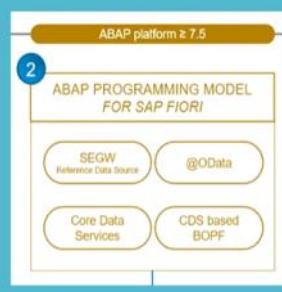
## Classic ABAP Application Programming

- Most SAP customers are using
- UI's Available :- Classic Dynpro, Web Dynpro ABAP, Floorplan Manager
- With the release 7.4 ABAP optimized for HANA
  - CDS Introduced
  - SAP Gateway became an integral part of the AS ABAP
  - To enhance user experience UI5 + OData
  - ADT Eclipse introduced
- We are not part of front end development since UI5 introduced



## ABAP Programming Model for SAP Fiori

- We can build HANA optimized web-based applications of all types, i.e., search, analytical and transactional apps
- To support transactional-based apps (CRUD operation) we need to use BOPF
- With ABAP 7.51 the draft capability
- CDS UI annotations can develop Fiori apps without any UI5 coding or less (Fiori elements)



## Drawbacks

- Difficult to integrate CDS + ODATA+ BOPF
- Brownfield implementation not supported for CRUD operation
- We need to integrate the legacy code
- We can't rebuild everything using BOPF
- We don't want look at technical aspect too much . SAP want us to focus on Business logic. Ex :- V2 -> V4
- SAP also moving all APP from BOPF ->RAP slowly if needed



## ABAP RESTful Application Programming Model

SAP Cloud Platform, ABAP Environment  
SAP S/4HANA ≥ 1909

ABAP RESTful APPLICATION PROGRAMMING MODEL

Business Service

Core Data Services      Behavior Definition & Implementation

- RAP stands for ABAP RESTful Application Programming Model.
- It's a programming model that defines the architecture for developing OData services in SAP BTP ABAP Environment or Application Server ABAP.
- RAP uses the SAP HANA database to persist application data.



## ABAP RESTful Application Programming Model

- Integrate the different concepts into the ABAP and CDS languages
- It is generally available in the **SAP BTP ABAP** Environment
- It is available from **SAP S/4HANA on-premises 1909** edition
- RAP is the only recommended programming model in SAP S/4HANA on-premise as of edition 2021 and SAP BTP ABAP environment.



## ABAP RESTful Application Programming Model

- RAP is based on proven and modern technologies and concepts. These include:
- Core Data Services (CDS)
  - The **modernized and extended ABAP Language** used to implement the business logic
  - The **OData protocol** used for stateless communication
  - The concept of **Business Objects** used for building transactional applications
  - The concept of Draft used for building stateful applications with stateless technologies
  - The concept of Business Services used to create OData-, InA-, and SQL-based services



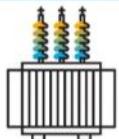
# The key players in the ABAP RESTful Application Programming Model are:



ABAP Developments  
Tools in Eclipse



Languages:  
ABAP and CDS



Powerful  
frameworks



## Big Picture-Architecture Overview



## DATA MODELING & BEHAVIOR

- It deals with data and the corresponding business logic.
- The RAP uses **CDS** to define and organize the data model.
  - CDS provides a framework for defining and consuming semantic data models
  - **CDS** entities are the **fundamental building blocks** for your application
- Data models support access to database
  - Transactional access ( Uses Business Object)
  - Query access ( Uses Queries)



# Business Object

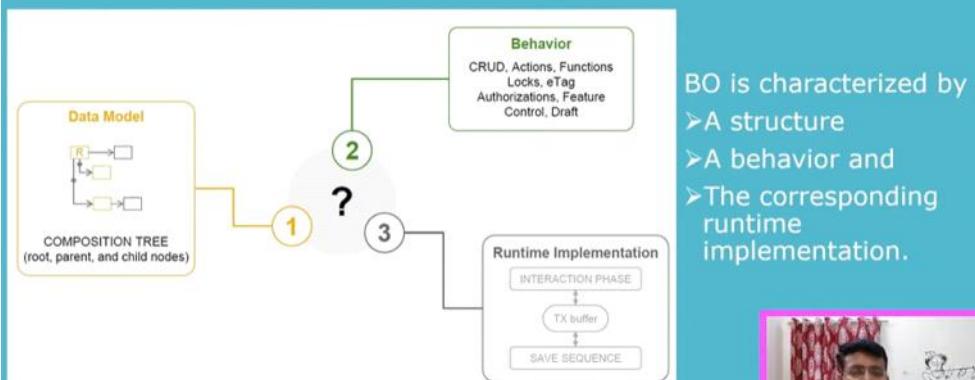
- A business object (BO) is a common term to represent a real-world artifact in enterprise application development such as the Travel, Sales order
- It have several nodes
  - The Travel => Booking => Booking Supplements
  - The Sales Order => Sales Item => Schedule Lines
- It have Common Transactional operation
  - Create
  - Update
  - Delete
  - Actions



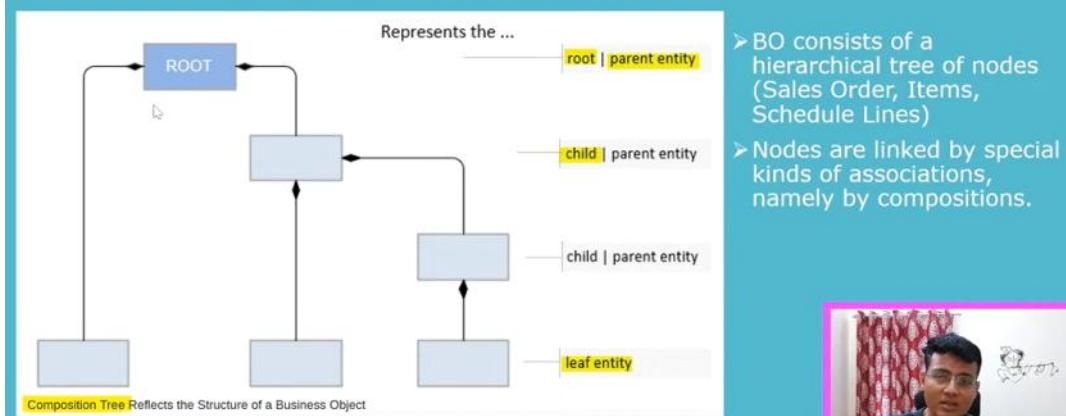
## A Business Object is characterized by

- a structure,
- a behavior and
- the corresponding runtime implementation.

# Business Object



## BO is characterized by A structure



# Association vs Composition

➤ Association and composition are two types of relationships between entities in CDS.

- Association is a general relationship between two entities. It can be one-to-one, one-to-many, or many-to-many.

```
define view entity ZI_RAP_Travel_ram  
as select from zrap_atrav_ram as Travel  
association [0..*] to ZI_RAP_Booking_ram as _Booking  
on $projection.TravelUUID = _Booking.TravelUUID
```

```
define view entity ZI_RAP_Booking_ram  
as select from zrap_abook_ram as Booking  
association[1..1] to ZI_RAP_Travel_ram as _Travel  
on $projection.TravelUUID = _Travel.TravelUUID
```



## Association vs Composition

➤ Association and composition are two types of relationships between entities in CDS.

- Composition is a stronger type of relationship between two entities. It represents a whole-part relationship, where the child entity cannot exist without the parent entity.

```
define root view entity ZI_RAP_Travel_ram  
as select from zrap_atrav_ram as Travel  
  
composition [0..*] of ZI_RAP_Booking_ram as _Booking  
  
define view entity ZI_RAP_Booking_ram  
as select from zrap_abook_ram as Booking  
association to parent ZI_RAP_Travel_ram as _Travel  
on $projection.TravelUUID = _Travel.TravelUUID
```



## Key differences

Feature	Association	Composition
Strength of relationship	Weak	Strong
Can the child entity exist without the parent entity?	Yes	No
Can the child entity be deleted independently of the parent entity?	Yes	No
Cardinality	Can be one-to-one, one-to-many, or many-to-many	Typically one-to-many

## Root Entity

- Root node defines the top node within a hierarchy in a business object's structure
- The root entity serves as a representation of the business object

```
define root view entity ZI_RAP_Travel_ram  
as select from zrap_atrav_ram as Travel  
  
composition [0..*] of ZI_RAP_Booking_ram as _Booking
```



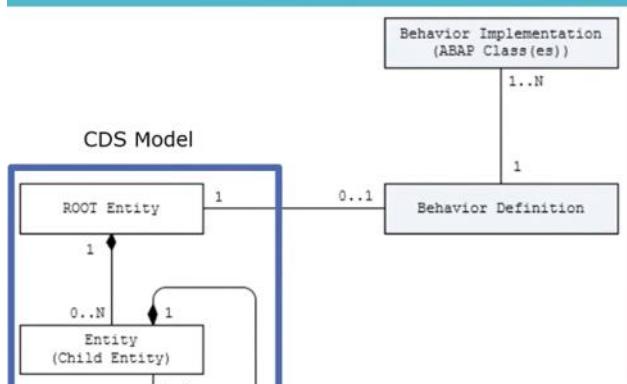
## Child and Leaf Entity

- A to-parent association in ABAP CDS is a specialized association which can be defined to model the child parent relationship between two CDS entities.

```
define view entity ZI_RAP_Booking_ram  
as select from zrap_abook_ram as Booking  
association to parent ZI_RAP_Travel_ram as _Travel  
on $projection.TravelUUID = _Travel.TravelUUID
```

- Leaf entity - if it represents a node in a business object's structure without any child nodes.

## Behavior of a Business Object



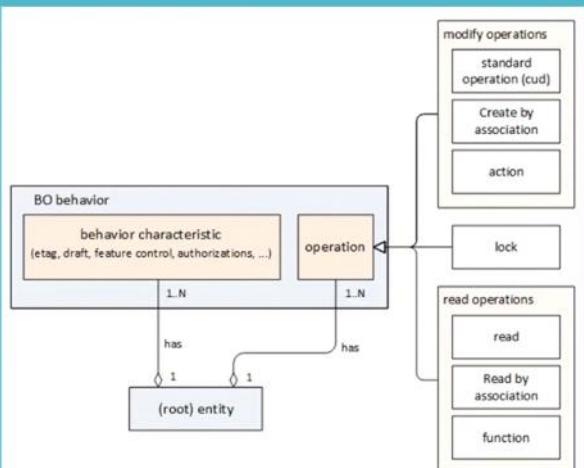
➤ **Behaviour definition** is an ABAP Repository object that describes the behavior of a business object.

➤ A behavior definition is defined using the Behavior Definition Language (BDL).

➤ The implementation of a behavior definition can be done in a single ABAP class (behavior pool) or can be split between an arbitrary set of ABAP classes.



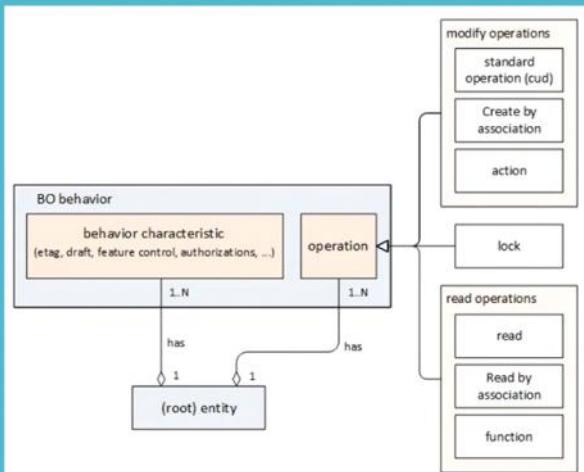
# Business Object's Behavior



- A behavior specifies the
  - Operations
  - Field properties
- It includes for each entity of the business object's composition tree.
  - A **behavior characteristic**
  - A **set of operations**



# Business Object's Behavior

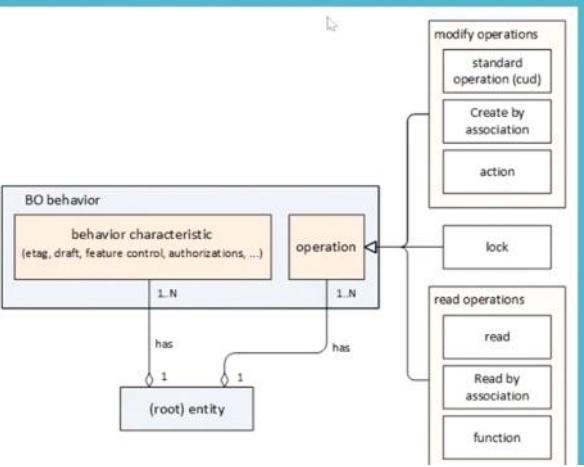


## Operations

- Modify operations :-
  - CUD
  - Action
- Read operations
  - Read
  - Read By Operation
  - Function



# Business Object's Behavior

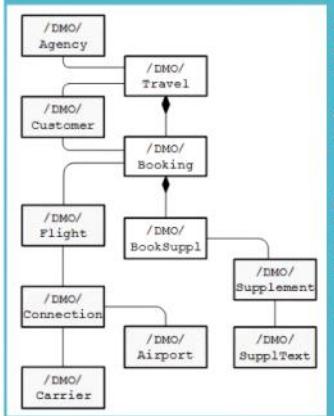


## Behavior Characteristic

- ETag
- Draft handling
- Feature control
- Numbering
- Authorization Control



# ABAP Flight Reference Scenario



1. /DMO/AGENCY :-General data about the travel agency ( Name, address )
2. /DMO/CUSTOMER :- Stores general data about customers ( Name, address )
3. /DMO/FLIGHT :- Stores general data about flights (Carrier Id connection id , flight date, price, max seat etc)
4. /DMO/CARRIER :- stores data about flight carriers ( id name )
5. /DMO/CONNECTION :- stores general data about flight connections (carrier\_id connection\_id airport\_from\_id airport\_to\_id departure\_time arrival\_time distance distance\_unit )
6. /DMO/AIRPORT :- stores data about airports (Name, city, country)
7. /DMO/SUPPLEMENT :- stores general data about the supplement
8. /DMO/TRAVEL :- stores general travel data.
9. /DMO/BOOKING :- stores data about a booked flight for a certain travel instance
10. /DMO/BOOK\_SUPPL :- stores data of booking supplements that can be booked for flights( supplement\_id price)



## Service Consumption

- The OData Service can be consumed by a Fiori application

## Business Service Provisioning

- Creating an OData Service.

## Data Modeling

- Defining the Data Model with CDS.

D [TRL] YRAP\_SAMPLE X

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'View Entity Flight Connection'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YRAP_Sample
6   as select from /dmo/connection
7 {
8   key carrier_id      as CarrierId,
9   key connection_id   as ConnectionId,
10  airport_from_id    as AirportFromId,
11  airport_to_id      as AirportToId,
12  departure_time     as DepartureTime,
13  arrival_time       as ArrivalTime,
14  distance           as Distance,
15  distance_unit      as DistanceUnit
16 }
17

```

## Relating Semantically Dependent Elements

- Amounts of money and Amounts of measures
- If we create annotations that define a link to the unit for the amounts. The amounts and their units are always handled as being dependent on each other in the OData service.
- This means that they are given the OData annotation sap:unit and sap:semantics in the OData metadata document.
- On UIs in particular, amounts are displayed with the correct decimals with regard to their unit



```
D [TRL] YRAP_SAMPLE ×  ↗ [TRL] YRAP_SERVICE1  ↗ [TRL] YSERVICE_BIND1
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'View Entity Flight Connection'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YRAP_Sample
6   as select from /dmo/connection
7 {
8   key carrier_id      as CarrierId,
9   key connection_id   as ConnectionId,
10  airport_from_id    as AirportFromId,
11  airport_to_id      as AirportToId,
12  departure_time     as DepartureTime,
13  arrival_time       as ArrivalTime,
14  @Semantics.quantity.unitOfMeasure: 'DistanceUnit'
15  distance           as Distance,
16  distance_unit      as DistanceUnit
17 }
18
```

## Designing the User Interface for a Fiori Elements App

- **UI annotations** can be used in CDS to configure the **look** of the **user interface** of a Fiori App.
- CDS offers the option of defining a universal setup for the **presentation** and **order of items**
- This is independent of the UI technology or application device(Reusability)
- You can use metadata extensions to separate the metadata specified by @UI or other UI related annotations from the actual data definition in the CDS view.



## A list Report

- The List Report in SAP Fiori Elements is a **template** that can be used to efficiently implement standard applications based on large lists.
- The list report offers powerful features for finding and acting on relevant data sets.
- It is often used as an entry point for navigating to the item details, which are usually shown on an object page.



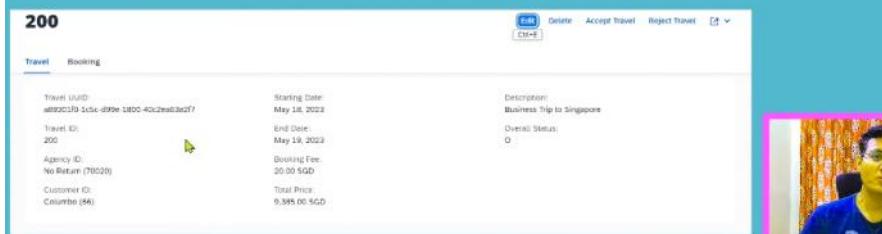
## The list report has several key functions

- Filter bar :- It will be on header
- Variant management :- Like we save variant in SE38
- Visibility :- Filter bar hide when we scroll down, We Can pin also
- Table toolbar :- It have title +No of items +Export functionality + table personalisation(sort, group, column selection) + Create +Delete +Copy

```
D [TRL] YRAP_SAMPLE X  O [TRL] YRAP_SERVICE1  S [TRL] YSERVICE_BIND1
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'View Entity Flight Connection'
4 @Metadata.ignorePropagatedAnnotations: true
5
6 @UI.headerInfo: {
7     typeName: 'Connection',                      // For Single Record
8     typeNamePlural: 'Connnection_s'             // For Multiple Records
9 }
10
11 define view entity YRAP_Sample
12   as select from /dmo/connection
13 {
14     @UI.lineItem: [{ position: 10, label: 'Airline'}]
15     key carrier_id      as CarrierId,
16
17     @UI.lineItem: [{ position: 20 }]
18     key connection_id   as ConnectionId,
19
20@   @UI.selectionField: [{ position: 10 }]
21     @UI.lineItem: [{ position: 30 }]
22     airport_from_id as AirportFromId,
23
24@   @UI.selectionField: [{ position: 20 }]
25     @UI.lineItem: [{ position: 40 }]
26     airport_to_id   as AirportToId,
27
28     @UI.lineItem: [{ position: 50, label: 'Departure Time' }]
29     departure_time  as DepartureTime,
30
31     @UI.lineItem: [{ position: 60, label: 'Arrival Time' }]
32     arrival_time    as ArrivalTime,
33 }
```

## Object Page

- Whereas the list report gives a general overview of the list items.
- The object page shows more detailed information about a single list item. You navigate to the object page by clicking the item in the list report.



## Facet in UI

- In SAP Fiori, a UI facet is a **grouping of related fields or controls** on a user interface.
- Facets are used to **organize information** into logical sections, making it easier for users to find and understand the data they are looking for.
- There are two main types of UI facets:
  - **Collection facets:** These facets display a list of records, each of which can contain one or more reference facets.
  - **Reference facets:** These facets display information about a single record, such as a line item or a chart.

## Facet in UI

- Facets can be used to **display** a wide variety of information, such as **Basic data fields, Contact information, Images, Charts, Maps**, Links to other applications
- Facets can be nested within each other
- In addition to the above, UI facets can also be used to:
  - Control the visibility of fields or controls on the user interface.
  - Define the order in which fields or controls are displayed.
  - Group fields or controls together for editing or filtering.

## IDENTIFICATION\_REFERENCE facet type

- It is used to display identification information about a record.
- This type of facet is typically used to display information such as a name, address, or contact information.
- The IDENTIFICATION\_REFERENCE type facet is used in conjunction with the @UI.identification annotation.
- The @UI.identification annotation is used to mark fields that contain identification information.
- When a field is marked with the @UI.identification annotation, the field's value is displayed in the IDENTIFICATION\_REFERENCE type facet.

**Simple Assign Facet Will not Be See But Have To Use Ui.Identification (tells Which we want See in sepeated pop up)**

```

10
11 define view entity YRAP_Sample
12   as select from /dmo/connection
13 {
14
15@   @UI.facet: [{ purpose: #STANDARD,
16     type: #IDENTIFICATION_REFERENCE,
17     position: 10,
18     label: 'Connections_Facet' }]
19
20   @UI.lineItem: [{ position: 10, label: 'Airline'}]
21   @UI.identification: [{ position: 10,label: 'Airline' }]
22 key carrier_id      as CarrierId,
23
24@   @UI.identification: [{ position: 20 }]
25   @UI.lineItem: [{ position: 20 }]
26 key connection_id   as ConnectionId,
27
28@   @UI.identification: [{ position: 30 }]
29   @UI.selectionField: [{ position: 10 }]
30   @UI.lineItem: [{ position: 30 }]
31   airport_from_id as AirportFromId,
32
33@   @UI.identification: [{ position: 40 }]
34   @UI.selectionField: [{ position: 20 }]
35   @UI.lineItem: [{ position: 40 }]
36   airport_to_id   as AirportToId,
37
38@   @UI.identification: [{ position: 50 }]
39   @UI.lineItem: [{ position: 50,label: 'Departure Time' }]
40   departure_time  as DepartureTime,
41
42@   @UI.identification: [{ position: 60 }]
43   @UI.lineItem: [{ position: 60, label: 'Arrival Time' }]
44   arrival_time    as ArrivalTime,

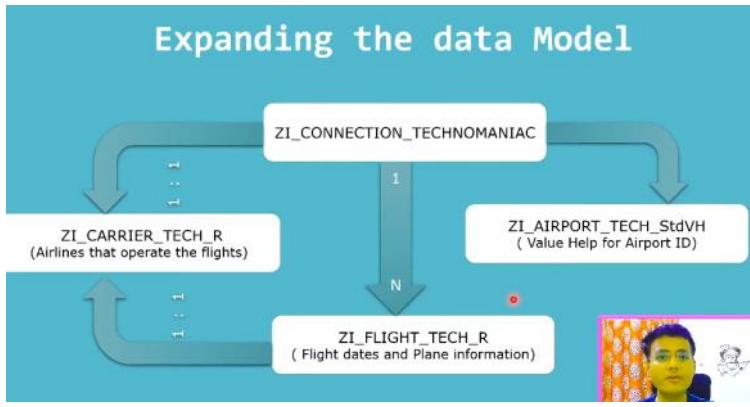
```

## Association & Search

25 November 2025 10:45

The screenshot shows a SAP Fiori application titled "SAP TECHNOMANIAC". The search bar at the top has "Departure Airport" and "Destination Airport" fields. Below the search bar is a button labeled "Go" and a link "Adapt Filters". The main area displays a table titled "Connections (20)". The table has columns: Airline, Flight Num..., Departure A..., Destination ..., Departure ..., %, and Arrival Time. The data shows several flights from AA and AZ. For example, AA flight 267B departs HAV at 6:15:00 AM and arrives MIA at 10:30:00 AM. AZ flight 788 departs VCE at 1:25:00 PM and arrives NRT at 10:13:00 AM.

- We need to add flight date, Plane types etc. in new facet
- Value help for selection fields
- Enhance search Capability of app
- Display text for the Airline ID
- Label Change



### Add Flight Using Association

```

11 define view entity YRAP_Sample
12   as select-from /dpo/connection as connection
13     association [...] to VI_Flight_R as _Flight
14       on $projection.CarrierId = _Flight.CarrierId
15       and $projection.ConnectionId = _Flight.ConnectionId
16   {
17     @UI.facet: [
18       purpose: #STANDARD,
19       type: #IDENTIFICATION_REFERENCE,
20       position: 10,
21       label: 'Connections_facet'
22     ],
23     (
24       purpose: #STANDARD,
25       type: #IDENTITY_REFERENCE,
26       position: 20,
27       label: 'Flights',
28       targetelement: '_Flight'
29     )
30   }
31   @UI.lineItem: [
32     { position: 10, label: 'Airline' },
33     @UI.identification: [
34       { position: 10, label: 'Airline' }
35     ]
36   key carrier_id as CarrierId,
37   key connection_id as ConnectionId,
38   @UI.identification: [
39     { position: 10 },
40     { position: 20 },
41     { position: 30 }
42   ],
43   airport_from_id as AirportFromId,
44   @UI.identification: [
45     { position: 40 },
46     @UI.selectionField: [
47       { position: 40 }
48     ],
49     @UI.lineItem: [
50       { position: 40 }
51     ],
52     @UI.lineItem: [
53       { position: 40, label: 'Departure Time' }
54     ],
55     departure_time as DepartureTime,
56   ],
57   @UI.lineItem: [
58     { position: 50 },
59     @UI.lineItem: [
60       { position: 50, label: 'Arrival Time' }
61     ],
62     arrival_time as ArrivalTime,
63   ],
64   @UI.lineItem: [
65     { position: 60 },
66     @Semantic.quantity.unitOfMeasure: 'DistanceUnit',
67     distance as Distance,
68     distance_unit as DistanceUnit,
69   ],
70   @Flight[] Expose Association
71 }
72

```

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Flight Information'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YI_flight_R
6 as select from /dmo/flight
7 {
8     @UI.lineItem: [{ position: 10 }]
9     key carrier_id as CarrierId,
10    @UI.lineItem: [{ position: 20 }]
11    key connection_id as ConnectionId,
12    @UI.lineItem: [{ position: 30 }]
13    key flight_date as FlightDate,
14    @UI.lineItem: [{ position: 40 }],
15    @Semantics.amount.currencyCode: 'CurrencyCode'
16    price as Price,
17    @UI.lineItem: [{ position: 50 }]
18    currency_code as CurrencyCode,
19    @UI.lineItem: [{ position: 60 }]
20    plane_type_id as PlaneTypeId,
21    @UI.lineItem: [{ position: 70 }]
22    seats_max as SeatsMax,
23    @UI.lineItem: [{ position: 80 }]
24    seats_occupied as SeatsOccupied
25 }
26

```

```

1 @EndUserText.label: 'Service Definition For Flight Connection'
2 define service Yrap_Service1 {
3     expose YRAP_Sample;
4     expose YI_flight_R;
5 }

```

```

define view entity YRAP_Sample
as select from /dmo/connection as connection
association [1..*] to YI_flight_R as _flight on $projection.CarrierId = _flight.CarrierId
and $projection.ConnectionId = _flight.ConnectionId
association [1] to YI_carrier_R as _Airline on $projection.CarrierId = _Airline.CarrierId

{
    @UI.facet: [{ purpose: #STANDARD,
    type: #IDENTIFICATION_REFERENCE,
    position: 10,
    label: 'Connections_Facet'},
    { purpose: #STANDARD,
    type: #LINEITEM_REFERENCE,
    position: 20,
    label: 'Flights',
    targetElement: '_flight'}
]

@UI.lineItem: [{ position: 10, label: 'Airline'}]
@UI.identification: [{ position: 10, label: 'AirLine' }]
#@ObjectModel.text.association: '_Airline'
key carrier_id as CarrierId,
@UI.identification: [{ position: 20 }]
@UI.lineItem: [{ position: 20 }]
key connection_id as ConnectionId,
@UI.identification: [{ position: 30 }]
@UI.selectionField: [{ position: 10 }]
@UI.lineItem: [{ position: 30 }]
airport_from_id as AirportFromId,

```

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Flight Information'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YI_flight_R
6 as select from /dmo/flight
7 association [1] to YI_carrier_R as _Airline on $projection.CarrierId = _Airline.CarrierId
8
9 {
10    @UI.lineItem: [{ position: 10 }]
11    #@ObjectModel.text.association: '_Airline'
12    key carrier_id as CarrierId,
13    @UI.lineItem: [{ position: 20 }]
14    key connection_id as ConnectionId,
15    @UI.lineItem: [{ position: 30 }]
16    key flight_date as FlightDate,
17    @UI.lineItem: [{ position: 40 }]
18    @Semantics.amount.currencyCode: 'CurrencyCode'
19    price as Price,
20    @UI.lineItem: [{ position: 50 }]
21    currency_code as CurrencyCode,
22    @UI.lineItem: [{ position: 60 }]
23    plane_type_id as PlaneTypeId,
24    @UI.lineItem: [{ position: 70 }]
25    seats_max as SeatsMax,
26    @UI.lineItem: [{ position: 80 }]
27    seats_occupied as SeatsOccupied,
28    _Airline
29 }
30

```

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Carrier Details'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YI_carrier_R
6   as select from /dmo/carrier
7 {
8   key carrier_id as CarrierId,
9   @Semantics.text: true
10  name as Name,
11  currency_code as CurrencyCode
12 }
13

```

#### Result

[Connections\\_Facet](#) [Flights](#)

#### Connections\_Facet

AirLine:	Departure Air
<a href="#">American Airlines Inc. (AA)</a>	JFK
Flight Number:	Destination A
15	SFO

#### Flights (2) Standard ▾

Airline ID	Flight Num...
<a href="#">American Airlines Inc. (AA)</a>	15
<a href="#">American Airlines Inc. (AA)</a>	15

## Adding Search Capabilities

- Include a search input field to execute a text and fuzzy search on multiple elements.
- The searches for **days** and **plane types** require the search to be operated on the associated view /DMO/I\_Flight\_R.
- In addition, we also want to be able to search for the **full airline name**. That is why the text provider view /DMO/I\_Carrier must also be search enabled.
- We want to search
  - Connections operated by one or more certain airlines
  - Connections on one or more certain days
  - Connections with one or more plane types
  - Or a combination of any of these



SAP Technomaniac ▾

Departure Airport:  Destination Airport:

[TRL] YRAP\_SAMPLE X [TRL] YRAP\_SERVICE [TRL] YSERVICE\_BIND1 [TRL] YI\_FLIGHT\_R [TRL] YI\_CARRIER\_R

```

011 @Search.searchable: true
012
013 define view entity YRAP_Sample
014   as select from /dmo/connection as connection
015   association [1..*] to YI_flight_R as _flight on $projection.CarrierId = _flight.CarrierId
016   and $projection.ConnectionId = _flight.ConnectionId
017   association [1] to YI_carrier_R as _Airline on $projection.CarrierId = _Airline.CarrierId
018
019 {
020
021   @UI.facet: [{ purpose: #STANDARD,
022     type: #IDENTIFICATION_REFERENCE,
023     position: 10,
024     label: 'Connections_Facet'
025   },
026   { purpose: #STANDARD,
027     type: #LINEITEM_REFERENCE,
028     position: 20,
029     label: 'Flights',
030     targetElement: '_flight'
031   }
032 ]
033
034   @UI.lineItem: [{ position: 10, label: 'Airline' }]
035   @UI.identification: [{ position: 10, label: 'Airline' }]
036   @ObjectModel.text.association: '_Airline'
037   @Search.defaultSearchElement: true
038   key carrier_id as CarrierId,
039
040   @UI.identification: [{ position: 20 }]
041   @UI.lineItem: [{ position: 20 }]
042   @Search.defaultSearchElement: true
043   key connection_id as ConnectionId,
044
045   // @Search.defaultSearchElement: true
046

```

```

Add Based Text also
60
61 @UI.identification: [{ position: 60 }]
62 @UI.lineItem: [{ position: 60, label: 'Arrival Time' }]
63 arrival_time as ArrivalTime,
64
65 @UI.identification: [{ position: 70 }]
66 @Semantics.quantity.unitOfMeasure: 'DistanceUnit'
67 distance as Distance,
68
69 distance_unit as DistanceUnit,
70
71 flight, // Expose Association
72 @Search.defaultSearchElement: true
73 _Airline
74

```

```

# [ ] YI_CARRIER_R [ ] YI_FLIGHT [ ] YI_SERVICE_BINDING
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Carrier Details'
4 @Metadata.ignorePropagatedAnnotations: true
5
6 @Search.searchable: true
7
8 define view entity YI_carrier_R
9 as select from /dmo/carrier
10 {
11   key carrier_id as CarrierId,
12   @Search.defaultSearchElement: true
13   @Semantics.text: true
14   name as Name,
15   currency_code as CurrencyCode
16 }
17

```

**Problem is Have To Enter Complete Name like American airline  
not america**

### To Resolve

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Carrier Details'
4 @Metadata.ignorePropagatedAnnotations: true
5
6 @Search.searchable: true
7
8 define view entity YI_carrier_R
9 as select from /dmo/carrier
10 {
11   key carrier_id as CarrierId,
12   @Search.defaultSearchElement: true
13   @Search.fuzzinessThreshold: 0.8
14   @Semantics.text: true
15   name as Name,
16   currency_code as CurrencyCode
17 }
18

```

### Search Based On plane type which from associated table

```

@Search.defaultSearchElement: true
_flight, // Expose Association
@Search.defaultSearchElement: true
_Airline

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Flight Information'
4 @Metadata.ignorePropagatedAnnotations: true
5
6 @Search.searchable: true
7
8 define view entity YI_flight_R
9 as select from /dmo/flight
10 association [1] to YI_carrier_R as _Airline on $projection.CarrierId = _Airline.CarrierId
11
12 {
13   @UI.lineItem: [{ position: 10 }]
14   @ObjectModel.text.association: '_Airline'
15   key carrier_id as CarrierId,
16   @UI.lineItem: [{ position: 20 }]
17   key connection_id as ConnectionId,
18   @UI.lineItem: [{ position: 30 }]
19   key flight_date as FlightDate,
20   @UI.lineItem: [{ position: 40 }]
21   @Semantics.amount.currencyCode: 'CurrencyCode'
22   price as Price,
23   @UI.lineItem: [{ position: 50 }]
24   currency_code as CurrencyCode,
25   @UI.lineItem: [{ position: 60 }]
26
27   @Search.defaultSearchElement: true
28   plane_type_id as PlaneTypeId,
29   @UI.lineItem: [{ position: 70 }]
30   seats_max as SeatsMax,
31   @UI.lineItem: [{ position: 80 }]
32   seats_occupied as SeatsOccupied,
33   _Airline
34 }
35

```

# Search Help

25 November 2025 13:24

## Providing Value Help

- The implementation of a value help in CDS enables the end user to choose values from a predefined list for input fields on a user interface.

Departure Airport:	Destination Airport:
<input type="text"/>	<input type="text"/>

- The annotation `@Consumption.valueHelpDefinition` allows you to reference the value help provider without implementing an association



## STEPS

- Create a data definition for a CDS view that serves as a value help provider. `ZI_AIRPORT_TECH_VH`
- You do not have to list value help provider views in the **service definition**. If you expose the source view in an OData service, the value help provider view is automatically exposed with it
- To enable the typeahead functionality enable the value help provider entity with search capabilities. Using `@Search.searchable: true` at value help level.



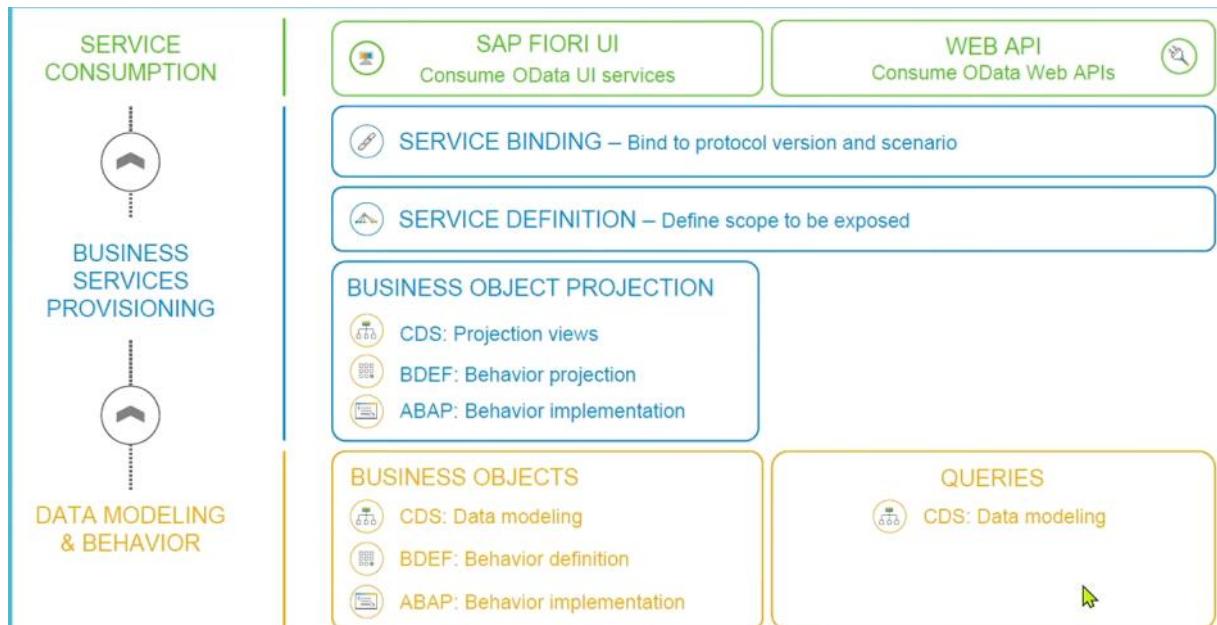
## Metadata file

- The property in the CDS source view for which a value help is implemented (`sap:value-list="standard"`)
- The value help provider entity type (`sap:value-list="true"`)

```

43 key connection_id as ConnectionId,
44
45
46 @Consumption.valueHelpDefinition: [{ entity: {
47   name: 'YI_airport_vh1',
48   element: 'AirportId'
49 } }]
50 @Search.defaultSearchElement: true
51 @UI.identification: [{ position: 30 }]
52 @UI.selectionField: [{ position: 10 }]
53 @UI.lineItem: [{ position: 30 }]
54 @EndUserText.label: 'Departure Airport Id'
55 airport_from_id as AirportFromId,
56
57 @Consumption.valueHelpDefinition: [{ entity: {
58   name: 'YI_airport_vh1',
59   element: 'AirportId'
60 } }]
61 @Search.defaultSearchElement: true
62 @UI.identification: [{ position: 40 }]
63 @UI.selectionField: [{ position: 20 }]
64 @UI.lineItem: [{ position: 40 }]
65 airport_to_id as AirportToId,
66
67 @AbapCatalog.viewEnhancementCategory: [#NONE]
68 @AccessControl.authorizationCheck: #NOT_REQUIRED
69 @EndUserText.label: 'Value Help For Airport'
70 @Metadata.ignorePropagatedAnnotations: true
71
72 @Search.searchable: true
73 define view entity YI_airport_vh1
74   as select from /dmo/airport
75 {
76   @Search.defaultSearchElement: true
77   key airport_id as AirportId,
78   @Search.defaultSearchElement: true
79   @Search.fuzzinessThreshold : 0.9
80   name as Name,
81   @Search.defaultSearchElement: true
82   @Search.fuzzinessThreshold: 0.9
83   city as City,
84   @Search.fuzzinessThreshold: 0.9
85   @Search.defaultSearchElement: true
86   country as Country
87 }
88
89
90
91
92

```



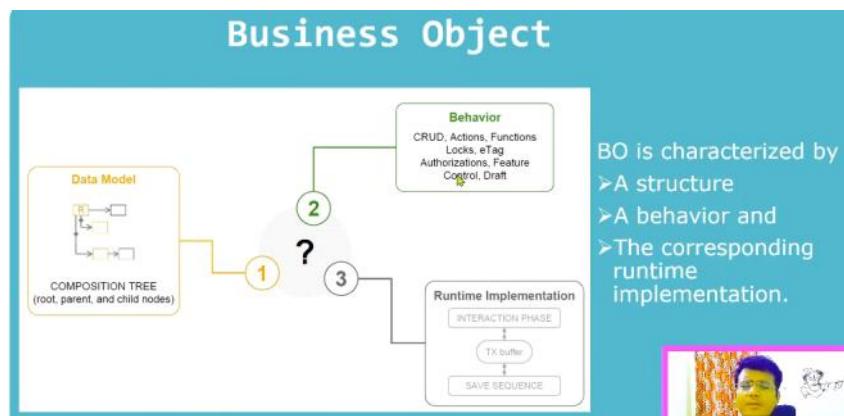
## Business Object Implementation Types

➤ The main approaches are the two standard implementation types of RAP business objects

- Managed
- Unmanaged

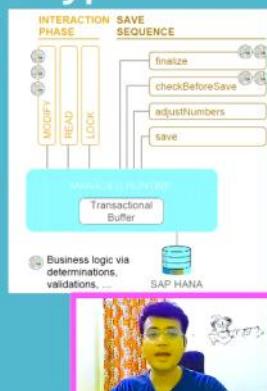
➤ Both approaches are based on a data model that is defined in CDS view entities.

➤ The difference between the two implementation types lies in the provisioning of the BO's transactional behavior



## Managed Implementation Type

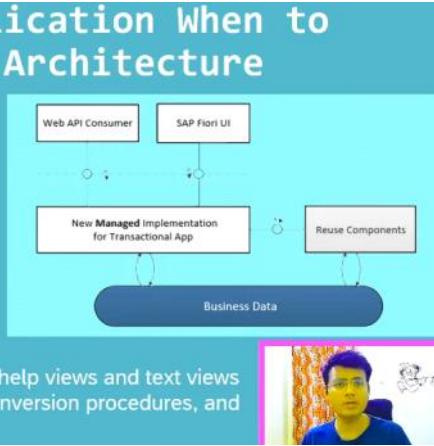
- In this scenario **BO framework implement** the **interaction phase** and the **save sequence** generically.
- The technical implementation done by BO runtime infrastructure
- We will get out-of-the-box support for transactional processing.
- Standard operations (create, update, delete) just need to specify in BDL
- The application developer can then focus on business logic that is implemented using **actions**, **validations** and **determinations** and user interaction using hooks



Business Development language

## Managed Application When to create & Architecture

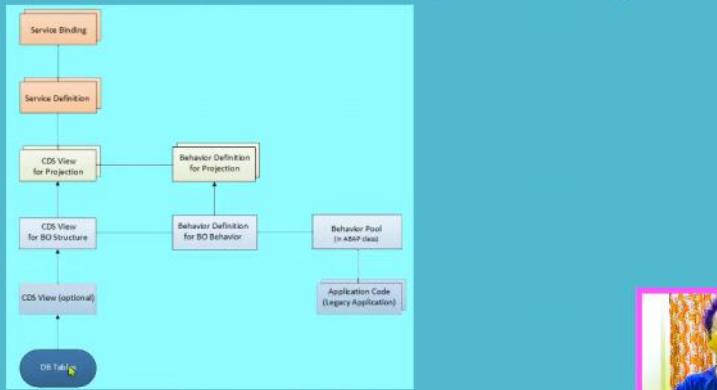
- Develop completely new transactional apps from scratch without any already existing code for business logic
- This is also known as greenfield development
- We don't have any transactional buffer or business logic implementation or authorization functionality available to us.



Reuse Components : -Uses of value help views and text views for text information retrieval, currency conversion procedures, and message constants etc.

## Steps To Create Managed Applications

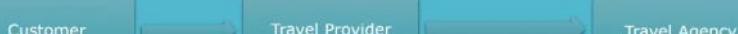
### Involved Development Objects



## Requirement

### Develop a travel provider app

- We will implement a simple travel provider app that can be used to manage flight bookings
- A single travel should be booked by the travel provider for an existing customer through a travel agency and include one or multiple flight bookings

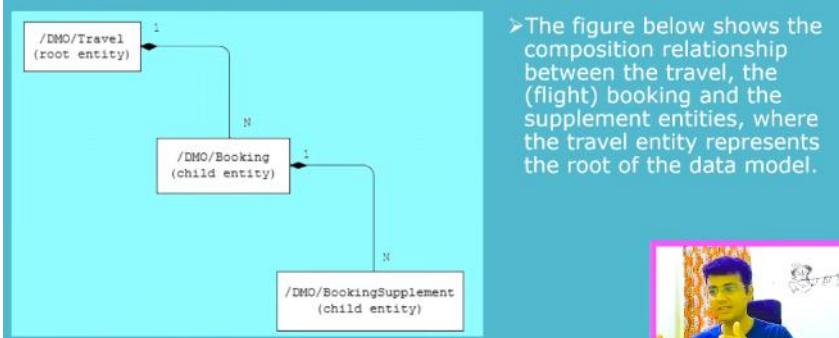


## Two different views of the travel app

- Processor :-Creates individual travel instances, creates and manages individual flights, and adds supplements to flight bookings
- Approver :- Verification of the recorded travel data entered by the processor



## Entities for Transactional Data



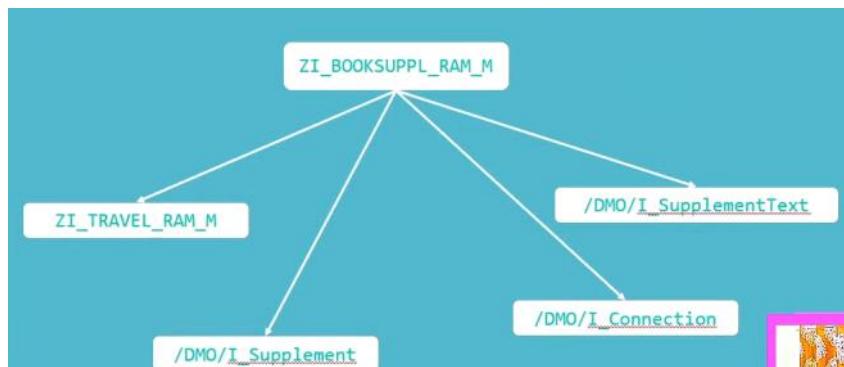
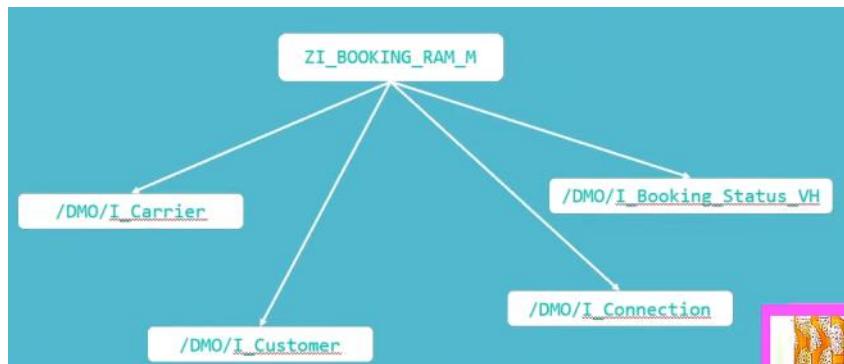
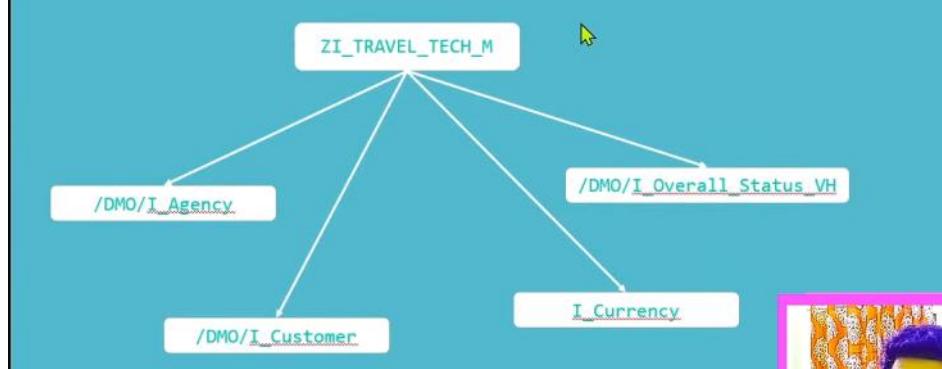
## Creating Persistent Tables

- Table ZTRAVEL\_TECH\_M: Persistent Table for Managing Travel Data
  - Table ZBOOKING\_TECH\_M: Persistent Table for Managing Bookings
  - Table ZBOOKSUPPL\_TECH\_M: Persistent Table for Managing Booking Supplement Data
- > ( Table /DMO/TRAVEL\_M, /DMO/BOOKING\_M, /DMO/BOOKSUPPL\_M )

## Creating CDS Data Definitions

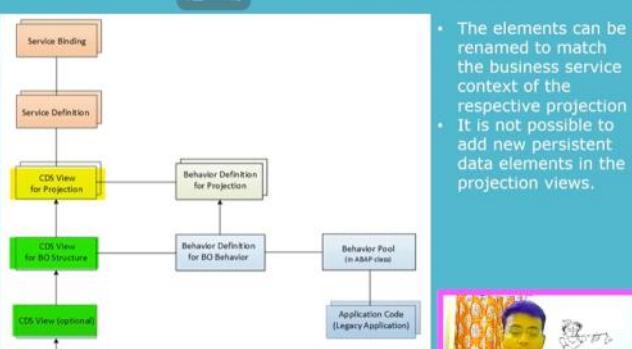
Data Definition CDS View Name	Description
ZI_TRAVEL_TECH_M (root entity)	A Travel entity defines general travel data, such as the agency ID or customer ID, overall status of the travel bookings, and the total price of a travel.
ZI_BOOKINGTECH_M ZI_BOOKINGTECH_M (child entity)	The booking entity is used for managing flight booking data, such as the customer, the flight connection, or the price and flight date.
ZI_BOOKSUPPLTECH_M ZI_BOOKSUPPLTECH_M (child entity)	This entity is used to add additional products to a travel booking.

# Master Data Connection



## Providing a Data Model for Projections

- Projection views define service-specific projections including denormalization of the underlying data model.
- We define here UI annotations, value helps, calculations or defaulting.
- We can add new read-only associations in the projection view. To display additional info on UI.
- New associated entities cannot be accessed transactionally.



```

D [TRL] YI_TRAVEL1_M × D [TRL] YI_BOOKING1_M D [TRL] YI_BOOKSUPPL1_M
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Travel Root Entity'
4 @Metadata.ignorePropagatedAnnotations: true
5 define root view entity YI_travel1_M
6   as select from ytravel_m
7   composition [1..*] of YI_booking1_m as _booking
8   association [0..1] to /DMO/I_Agency as _agency on $projection.AgencyId = _agency.AgencyID
9   association [0..1] to /DMO/I_Customer as _customer on $projection.CustomerId = _customer.CustomerID
10  association [1..1] to I_Currency as _currency on $projection.CurrencyCode = _currency.Currency
11  association [0..1] to /DMO/I_Overall_Status_VH as _status on $projection.OverallStatus=_status.OverallStatus
12 {
13   key travel_id      as TravelId,
14   agency_id       as AgencyId,
15   customer_id     as CustomerId,
16   begin_date      as BeginDate,
17   end_date        as EndDate,
18   @Semantics.amount.currencyCode: 'CurrencyCode'
19   booking_fee     as BookingFee,
20   @Semantics.amount.currencyCode: 'CurrencyCode'
21   total_price     as TotalPrice,
22   currency_code   as CurrencyCode,
23   description      as Description,
24   overall_status  as OverallStatus,
25   created_by      as CreatedBy,
26   created_at       as CreatedAt,
27   last_changed_by as LastChangedBy,
28   last_changed_at as LastChangedAt,

```

```

D [TRL] YI_TRAVEL1_M × D [TRL] YI_BOOKING1_M × D [TRL] YI_BOOKSUPPL1_M
1 @AbapCatal Data Definition YI_TRAVEL1_M [TRL100] - active - TRL_EN |
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Booking Root Entity'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YI_booking1_m
6   as select from ybooking_m
7   association to parent YI_travel1_M           as _travel    on $projection.TravelId = _travel.TravelId
8   composition [1..*] of YI_bookSuppl1_m as _bookindsuppl
9   association [1..1] to /DMO/I_Carrier          as _carrier   on $projection.CarrierId = _carrier.AirlineID
10  association [1..1] to /DMO/Customer          as _customer  on $projection.CustomerId = _customer.customer_id
11  association [1..1] to /DMO/I_Connection       as _connection on $projection.CarrierId = _connection.AirlineID
12                                         and $projection.ConnectionId = _connection.ConnectionID
13  association [1..1] to /DMO/I_Booking_Status_VH as _status    on $projection.BookingStatus = _status.BookingStatus
14
15 {
16   key travel_id      as TravelId,
17   key booking_id     as BookingId,
18   booking_date      as BookingDate,
19   customer_id       as CustomerId,
20   carrier_id        as CarrierId,
21   connection_id     as ConnectionId,
22   flight_date       as FlightDate,
23   @Semantics.amount.currencyCode: 'CurrencyCode'
24   flight_price      as FlightPrice,
25   currency_code     as CurrencyCode,
26   booking_status    as BookingStatus,
27   last_changed_at   as LastChangedAt,
28   _travel,

```

```

D [TRL] YI_TRAVEL1_M D [TRL] YI_BOOKING1_M D [TRL] YI_BOOKSUPPL1_M ×
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Booking Suppl Interface'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity Yi_bookSuppl1_m
6   as select from ybooksuppl_m
7   | association      to parent YI_booking1_m as _booking      on $projection.BookingId = _booking.BookingId
8   |                   and $projection.TravelId = _booking.TravelId
9   | association [1..1] to /dmo/supplement      as _supplement    on $projection.SupplementId = _supplement.supplement_id
10  | association [1..*] to /DMO/I_SupplementText as _supplementtext on $projection.SupplementId = _supplementtext.SupplementID
11 {
12   key travel_id          as TravelId,
13   key booking_id         as BookingId,
14   key booking_supplement_id as BookingSupplementId,
15   supplement_id          as SupplementId,
16   @Semantics.amount.currencyCode: 'CurrencyCode'
17   price                  as Price,
18   currency_code          as CurrencyCode,
19   last_changed_at        as LastChangedAt,
20   _booking,
21   _supplement,
22   _supplementtext
23 }
24

```

## Annotation Propagation to Projection Views

```

define root view /DMO/I_Travel
...
{
  key travel_id,
  ...
  @Semantics.amount.currencyCode: 'currency_code'
  total_price,
  @Semantics.currencyCode: true
  currency_code,
  ...
}

define root view entity /DMO/C_Travel as projection on
/DMO/I_Travel
...
{
  key travel_id,
  ...
  @Semantics.amount.currencyCode: 'CurrencyCode'
  total_price as TotalPrice,
  currency_code as CurrencyCode,
  ...
}

```

- Annotations that are defined in the projected entity on element level are completely propagated to the projection view.
- But the reference element is aliased in the projection entity. Then we have to redefine the annotation in the projection view



## Syntax

```

{@entity_annot1}
{@entity_annot2}
...
{@proj_view_annot1}
{@proj_view_annot2}
...
DEFINE [ROOT] VIEW ENTITY projection_view
  / PROVIDER CONTRACT TRANSACTIONAL QUERY
  AS PROJECTION ON cda entity {AS alias name}
  /association1 association2 ...
  /redefined_assoc1 redefined_assoc2 ...
  ( projection_list )
  /WHERE cda cond

```

## CDS DDL - PROVIDER CONTRACT

- The provider contract specifies the scenario in which a CDS projection view is used.
- It is recommended that a provider contract is always specified. Otherwise, no runtime-specific syntax checks are applied.
- PROVIDER CONTRACT
  - TRANSACTIONAL\_QUERY
  - TRANSACTIONAL\_INTERFACE
  - ANALYTICAL\_QUERY
- Child entities within a CDS composition tree inherit the provider contract and the specification of a provider for a child entity is not allowed.

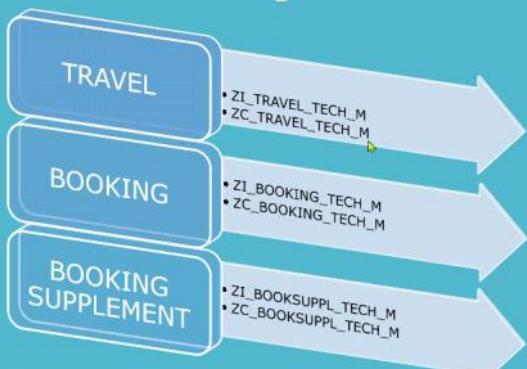


## CDS DDL - CDS Projection View, Transactional Queries

- CDS transactional queries are intended for modelling the projection layer of a RAP business object.
- A CDS projection view is based on another CDS view entity (called projected entity).
- Unlike other CDS entities, CDS transactional queries currently **cannot** be used as data source of other CDS entities.
- They can be used in ABAP programs as a data type for definitions and in ABAP SQL read statements.



### Creating Projection View On Projected Entity



```
D [TRL] YC_TRAVEL1 × D [TRL] YC_BOOKING1 D [TRL] YC_BOOKSUPPL1
1@#AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'Travel Project View'
3 @Metadata.ignorePropagatedAnnotations: true
4 define root view entity YC_travel1
5 provider contract transactional_query
6 as projection on YI_travel1_M
7 {
8   key TravelId,
9     AgencyId,
10    CustomerId,
11    BeginDate,
12    EndDate,
13    @Semantics.amount.currencyCode: 'CurrencyCode'
14    BookingFee,
15    @Semantics.amount.currencyCode: 'CurrencyCode'
16    TotalPrice,
17    CurrencyCode,
18    Description,
19    OverallStatus,
20    //      CreatedBy,
21    //      CreatedAt,
22    //      LastChangedBy,
23    LastChangedAt,
24    /* Associations */
25    _agency,
26    _booking: redirected to composition child YC_booking1,
27    _currency,
28    _customer,
29    _status
30 }
31
```

```

D [TRL] YC_TRAVEL1 D [TRL] YC_BOOKING1 X D [TRL] YC_BOOKSUPPL1
1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'Booking Projection'
3 @Metadata.ignorePropagatedAnnotations: true
4 define view entity YC_booking1
5   as projection on YI_booking1_m
6 {
7   key TravelId,
8   key BookingId,
9   BookingDate,
10  CustomerId,
11  CarrierId,
12  ConnectionId,
13  FlightDate,
14  @Semantics.amount.currencyCode: 'CurrencyCode'
15  FlightPrice,
16  CurrencyCode,
17  BookingStatus,
18  LastChangedAt,
19  /* Associations */
20  _bookingsuppl : redirected to composition child yC_booksuppl1,
21  _carrier,
22  _connection,
23  _customer,
24  _status,
25  _travel : redirected to parent YC_travel1
26 }
27

```

```

D [TRL] YC_TRAVEL1 D [TRL] YC_BOOKING1 D [TRL] YC_BOOKSUPPL1 X
1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'booking suppl Projection'
3 @Metadata.ignorePropagatedAnnotations: true
4 define view entity yc_booksuppl1
5   as projection on Yi_bookSuppl1_m
6 {
7   key TravelId,
8   key BookingId,
9   key BookingSupplementId,
10  SupplementId,
11  @Semantics.amount.currencyCode: 'CurrencyCode'
12  Price,
13  CurrencyCode,
14  LastChangedAt,
15  /* Associations */
16  _booking : redirected to parent YC_booking1,
17  _supplement,
18  _supplementtext,
19  _travel : redirected to YC_travel1
20 }
21

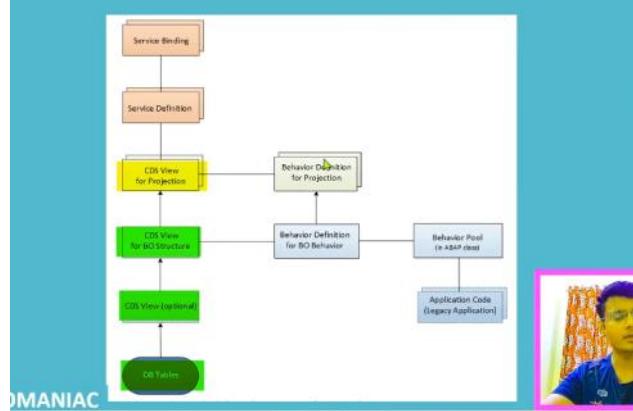
```

## Two different views of the travel app

- Processor :-Creates individual travel instances, creates and manages individual flights, and adds supplements to flight bookings
- Approver :- Verification of the recorded travel data entered by the processor



## Projection layer - Processor



## ABAP CDS Metadata Extensions

- MDE extends a CDS entity with CDS annotations that are not specified in the DDL source code of the data definition.
- It is defined and transported in a separate piece of DDLX source code.
- CDS entity should have `Metadata.allowExtensions True`
- There can be multiple metadata extensions for one CDS entity.
- Each metadata extension is assigned to a layer, such as CUSTOMER, PARTNER, INDUSTRY, LOCALIZATION, CORE



## MDE Advantages

- Improves the readability
- Transported independently
- Separating the activation of metadata from the data definition avoids the need for mass activation of dependent repository objects
- Layers and CDS variants allow frameworks, partners, and customers to override the metadata of a CDS entity without making modifications.



## ZC\_TRAVEL\_TECH\_M

The screenshot displays two main views of the ZC\_TRAVEL\_TECH\_M application:

- Standard View:** Shows a list of travels (4,148 entries) with columns for Travel ID, Agency ID, Customer ID, and Overall Status. A search bar and filter options are at the top.
- Booking Detail View:** Shows a detailed view for Travel ID 4248. It includes sections for Travel and Booking. The Travel section lists the travel ID, agency ID (Happy Holiday), and customer ID (Buchana (12)). The Booking section shows the starting date (Dec 7, 2023), overall status (Open), total price (300.00 - USD), and booking status (Open).



FOLLOW Order

```

TravelId, => lineItem ,identification,Serachable
AgencyId, => lineItem ,identification,Serachable,Selction field,Valuehelp(H) /DMO/I_Agency,ObjectModel
_Agency_Name as AgencyName, => NA
CustomerId, => lineItem ,identification,Serachable,Selction field,Valuehelp(H) /DMO/I_Customer,ObjectModel
_Customer.LastName as CustomerName, => NA
BeginDate, => lineItem ,identification,
EndDate, => lineItem ,identification,
BookingFee, => identification,
TotalPrice, => lineItem ,identification,
CurrencyCode, => Valuehelp I_Currency
Description, => identification
OverallStatus => lineItem ,identification,Serachable,Selction field,Valuehelp(H) '/DMO/I_Overall_Status_VH' 'OverallStatus',ObjectModel
_Status._Text.Text => NA
LastChangedAt, => hidden

D [TRL] YC_TRAVE... X D [TRL] YC_BOOKI... D [TRL] YC_BOOKS...
D [TRL] YI_TRAVE... E [TRL] YC_T
 5 @Metadata.ignoresPropagatedAnnotations: true
 4 @Metadata. Data Definition YC_TRAVEL1 [TRL,100] - active - TRL_EN
 5 define root-view entity YC_TRAVEL1
 6 provider contract transactional_query
 7 as projection on YI_travel1_M
 8 {
 9   key TravelId,
10   @ObjectModel.text.element: [ 'AgencyName' ]
11   AgencyId,
12   _agency.Name as AgencyName,
13   @ObjectModel.text.element: [ 'CustomerName' ]
14   CustomerId,
15   _customer.LastName as CustomerName,
16   BeginDate,
17   EndDate,
18   @Semantics.amount.currencyCode: 'CurrencyCode'
19   BookingFee,
20   @Semantics.amount.currencyCode: 'CurrencyCode'
21   TotalPrice,
22   CurrencyCode,
23   Description,
24   @ObjectModel.text.element: [ 'OverallStatusText' ]
25   OverallStatus,
26   _status._Text.Text as OverallStatusText : localized,
27   // CreatedBy,
28   // CreatedAt,
29   // LastChangedBy,
30   LastChangedAt,
31   /* Associations */
32   _agency,
33   _booking : redirected to composition child YC_booking1,
34   _currency,
35   _customer,
36   _status
37 }

2 @Search.searchable: true
3 @UI.headerInfo: {
4   typeName: 'Travel',
5   typeNamePlural: 'Travels',
6
7   title: {
8     type: #STANDARD,
9     label: 'Travel..',
10    value: 'TravelId'
11  }
12
13 }
14 annotate entity YC_travel1 with
15 {
16   @UI.facet: [
17
18     id: 'Travel',
19     purpose: #STANDARD,
20     position:10 ,
21     label: 'Travel_Details',
22     type:#IDENTIFICATION_REFERENCE
23   ]
24   @UI.lineItem: [{ position: 10 }]
25   @Search.defaultSearchElement: true
26   @UI.identification: [{ position: 10 }]
27   TravelId;
28   @UI: { lineItem: [{ position: 20 }],
29         selectionField: [{ position: 10 }],
30         identification: [{ position: 20 }]
31       }
32   @Search.defaultSearchElement: true
33   @Consumption.valueHelpDefinition: [{ entity: {
34                                         name: '/DMO/I_Agency',
35                                         element: 'AgencyID'
36                                       } }]
37   AgencyId;

@Metadata.layer: #CORE
@Search.searchable: true
@UI.headerInfo: {
typeName: 'Travel',
typeNamePlural: 'Travels',
title: {
type: #STANDARD,
label: 'Travel..',
value: 'TravelId'
}
}
annotate entity YC_travel1 with

```

```
{
@UI.facet: [{  

  id: 'Travel',  

  purpose: '#STANDARD,  

  position:10 ,  

  label: 'Travel_Details',  

  type:#IDENTIFICATION_REFERENCE  

} ]  

@UI.lineItem: [{ position: 10 }]  

@Search.defaultSearchElement: true  

@UI.identification: [{ position: 10 }]  

TravelId;  

@UI: { lineItem: [{ position: 20 }],  

selectionField: [{ position: 10 }],  

identification: [{ position: 20 }]  

}  

@Search.defaultSearchElement: true  

@Consumption.valueHelpDefinition: [{ entity: {  

  name: '/DMO/I_Agency',  

  element: 'AgencyID'  

} }]  

AgencyId;  

// Name;  

@UI: { lineItem: [{ position: 30 }],  

selectionField: [{ position: 30 }],  

identification: [{ position: 30 }]  

}  

@Search.defaultSearchElement: true  

@Consumption.valueHelpDefinition: [{ entity: {  

  name: '/DMO/I_Customer',  

  element: 'CustomerID'  

} }]  

CustomerId;  

// LastName;  

@UI.lineItem: [{ position: 40 }]  

@UI.identification: [{ position: 40 }]  

BeginDate;  

@UI.lineItem: [{ position: 50 }]  

@UI.identification: [{ position: 50 }]  

EndDate;  

@UI.identification: [{ position: 55 }]  

BookingFee;  

@UI.lineItem: [{ position: 60 }]  

@UI.identification: [{ position: 60 }]  

TotalPrice;  

@Consumption.valueHelpDefinition: [{ entity: {  

  name: 'I_Currency',  

  element: 'Currency'  

} }]  

CurrencyCode;  

@UI.identification: [{ position: 65 }]  

Description;  

@UI: { lineItem: [{ position: 70 }],  

selectionField: [{ position: 40 }],  

identification: [{ position: 70 }],  

textArrangement: #TEXT_ONLY  

}  

@Search.defaultSearchElement: true  

@Consumption.valueHelpDefinition: [{ entity: {  

  name: '/DMO/I_Overall_Status_VH',  

  element: 'OverallStatus'  

} }]  

OverallStatus;  

// OverallStatusText;  

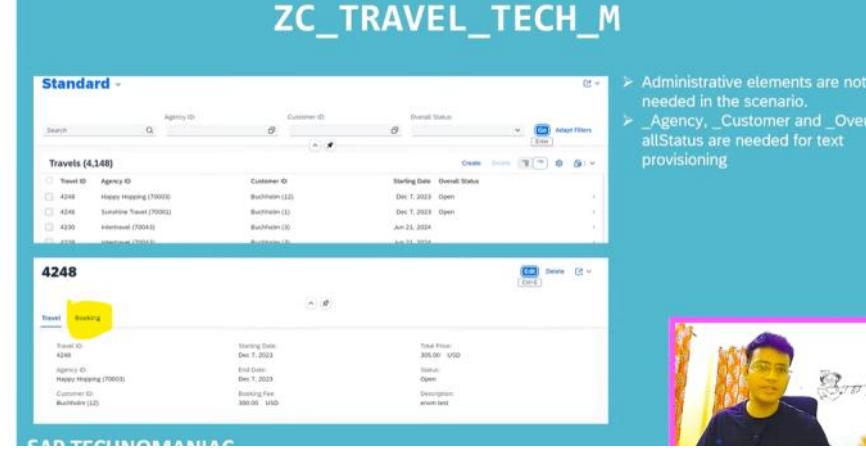
@UI.hidden: true  

LastChangedAt;  

}
}
```

}

#### NEXT ADD BOOKING



The screenshot shows the SAP Fiori interface for the 'ZC\_TRAVEL\_TECH\_M' application. The top navigation bar displays the system name 'ZC\_TRAVEL\_TECH\_M'. Below it, the 'Standard' view is shown, featuring a search bar and three filter buttons: 'Agency ID', 'Customer ID', and 'Overall Status'. A table lists travel details with columns for Travel ID, Agency ID, Customer ID, Starting Date, and Overall Status. One row is selected, showing '4248' and 'Buchmehr (12)'. To the right of the table, a note states: '➤ Administrative elements are not needed in the scenario. ➤ \_Agency, \_Customer and \_OverallStatus are needed for text provisioning'.

**Standard**

Travel ID	Agency ID	Customer ID	Overall Status
4248	Honey Hopping (70003)	Buchmehr (12)	Open
4249	Sweatline Travel (70002)	Buchmehr (12)	Open
4250	Intertravel (70043)	Buchmehr (12)	Open
4258	Wanderlust (70012)	Buchmehr (12)	Open

**4248**

Travel ID	Starting Date	Total Price
4248	Dec 7, 2023	305.00 - USD
Agency ID:	Agency Hopping (70003)	Status: Open
Customer ID:	Buchmehr (12)	Description: annual test
Booking Fee:	300.00 - USD	

**ZC\_BOOKING\_TECH\_M**

The screenshot shows two main panels. The top panel displays a list of bookings with columns for Booking No., Booking Date, Customer ID, Flight Price, Status, and Name. The bottom panel shows a search result for flights from United Airlines Inc. (UA) to American Airlines Inc. (AA) with flight numbers 332 and 333.

- Administrative elements are not needed in the scenario.
- `_Agency`, `_Customer` and `_OverallStatus` are needed for text provisioning



```
*****
travel_id; - Search
booking_id; - lineItem, identification,Search
booking_date; - lineItem, identification,
customer_id; - lineItem, identification,serach,ObjectModel,
               valueHelp - /DMO/I_Customer CustomerID
CustomerName;
carrier_id; - lineItem, identification,ObjectModel,
               valueHelp - /DMO/I_Carrier AirlineID
CarrierName;
connection_id; - lineItem, identification,
                valueHelp - /DMO/I_Flight ConnectionID
                   binding ConnectionID,AirlineID,Price,CurrencyCode
flight_date; - lineItem, identification,
               valueHelp - /DMO/I_Flight FlightDate
                   binding FlightDate0,AirlineID,Price,CurrencyCode
flight_price; - lineItem, identification,
currency_code; - valueHelp - I_Currency Currency
booking_status; - lineItem, identification,ObjectModeltextArrangement,
                  valueHelp - /DMO/I_Booking_Status_VH' BookingStatus
BookingStatusText; - hidden
last_changed_at; - hidden
*****
```

```
*****
travel_id; - Search
booking_id; - Search
booking_supplement_id; - lineItem, identification,
supplement_id; - lineItem, identification,ObjectModel,
               valueHelp '/DMO/I_SUPPLEMENT' 'SupplementID'
               binding Price CurrencyCode
SupplementDescription; -
price; - lineItem, identification,
currency_code; - valueHelp - I_Currency Currency
last_changed_at; - hidden
*****
```

```

1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'Travel Project View'
3 @Metadata.ignorePropagatedAnnotations: true
4 @Metadata.allowExtensions: true
⑤ 5 define root view entity YC_travel1
6   provider contract transactional_query
7     as projection on YI_travel1_M
8 {
9   key TravelId,
10   @ObjectModel.text.element: [ 'AgencyName' ]
11   AgencyId,
12   _agency.Name      as AgencyName,
13   @ObjectModel.text.element: [ 'CustomerName' ]
14   CustomerId,
15   _customer.LastName as CustomerName,
16   BeginDate,
17   EndDate,
18   @Semantics.amount.currencyCode: 'CurrencyCode'
19   BookingFee,
20   @Semantics.amount.currencyCode: 'CurrencyCode'
21   TotalPrice,
22   CurrencyCode,
23   Description]
24   @ObjectModel.text.element: [ 'OverallStatusText' ]
25   OverallStatus,
26   _status._Text.Text as OverallStatusText : localized,
27   //      CreatedBy,
28   //      CreatedAt,
29   //      LastChangedBy,
30   LastChangedAt,
31   /* Associations */

```

```

1 [TRL] YC_TRAVEL1 [TRL] YC_BOOKIN... [TRL] YC_BOOKSU... [TRL] YC_TRAVEL1 X [TRL] YC_E
2 // ...
3 @UI.lineItem: [{ position: 40 }]
4 @UI.identification: [{ position: 40 }]
5 BeginDate;
6 @UI.lineItem: [{ position: 50 }]
7 @UI.identification: [{ position: 50 }]
8 EndDate;
9 @UI.identification: [{ position: 55 }]
10 BookingFee;
11 @UI.lineItem: [{ position: 60 }]
12 @UI.identification: [{ position: 60 }]
13 TotalPrice;
14 @Consumption.valueHelpDefinition: [{ entity: {
15             name: 'I_Currency',
16             element: 'Currency'
17         } }]
18 CurrencyCode;
19 @UI.identification: [{ position: 65 }]
20 Description;
21 @UI: { lineItem: [{ position: 70 }],
22       selectionField: [{ position: 40 }],
23       identification: [{ position: 70 }],
24       textArrangement: '#TEXT_ONLY'
25     }
26 @Search.defaultSearchElement: true
27 @Consumption.valueHelpDefinition: [{ entity: {
28             name: '/DMO/I_Overall_Status_VH',
29             element: 'OverallStatus'
30         } }]
31 OverallStatus;
32 // OverallStatusText;
33 @UI.hidden: true
34 LastChangedAt;
35
36
37
38
39
40
41 }

1@AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'Booking Projection'
3 @Metadata.ignorePropagatedAnnotations: true
4 @Metadata.allowExtensions: true
5 define view entity YC_booking1
6   as projection on YI_booking1_m
7 {
8   key TravelId,
9   key BookingId,
10  BookingDate,
11  @ObjectModel.text.element: [ 'lastname' ]
12  CustomerId,
13  _customer.lastname as lastname,
14  @ObjectModel.text.element: [ 'carriername' ]
15  CarrierId,
16  _carrier.Name      as carriername,
17  ConnectionId,
18  FlightDate,
19  @Semantics.amount.currencyCode: 'CurrencyCode'
20  FlightPrice,
21  CurrencyCode,
22  @ObjectModel.text.element: [ 'bookindstatus' ]
23  BookingStatus,
24  _status._Text.Text as bookindstatus : localized,
25  LastChangedAt,
26  /* Associations */
27  _bookingsuppl : redirected to composition child yC_booksuppl1,
28  _carrier,
29  _connection,
30  _customer,
31  _status,
32  _travel      : redirected to parent YC_travel1
33 }
34

```

```

1 @Metadata.layer: #CORE
2 @Search.searchable: true
3 @UI:{ headerInfo: {
4     typeName: 'Booking',
5     typeNamePlural: 'Booking',
6     title: {
7         type: #STANDARD,
8         label: 'Booking',
9         value: 'BookingId'
10    }
11  }}
12 annotate entity YC_booking1 with
13 {
14   @UI.facet: [{{
15     id: 'Booking',
16     purpose: #STANDARD,
17     position: 10 ,
18     label: 'Booking',
19     type:#IDENTIFICATION_REFERENCE
20   }},
21   {
22     id: 'Booksuppl',
23     purpose: #STANDARD,
24     position: 20 ,
25     label: 'Booking Supplemnets',
26     type: #LINEITEM_REFERENCE,
27     targetElement: '_bookingsuppl'
28   }]
29 @Search.defaultSearchElement: true
30 TravelId;
31 @UI: { lineItem: [{ position: 10 }], 
32 identification: [{ position: 10 }] }
33 @Search.defaultSearchElement: true
34 BookingId;

52 CarrierId;
53 @UI:{lineItem: [{ position: 50 }], 
54 identification: [{ position: 50 }] }
55 @Consumption.valueHelpDefinition: [{ entity: {
56   name: '/DMO/I_Flight',
57   element: 'ConnectionID'
58 },
59 additionalBinding: [{{
60   element: 'ConnectionID',
61   localElement: 'ConnectionId' },
62   {
63     element: 'AirlineID',
64     localElement: 'CarrierId' },
65   {
66     element: 'CurrencyCode',
67     localElement: 'CurrencyCode' },
68   {
69     element: 'Price',
70     localElement: 'FlightPrice' }}]]
71 ConnectionId;
72 @UI:{lineItem: [{ position: 60 }], 
73 identification: [{ position: 60 }] }
74 @Consumption.valueHelpDefinition: [{ entity: {
75   name: '/DMO/I_Flight',
76   element: 'FlightDate'
77 },
78 additionalBinding: [{{
79   element: 'FlightDate',
80   localElement: 'FlightDate' },
81   {
82     element: 'AirlineID',
83     localElement: 'CarrierId' },
84   {
85     element: 'CurrencyCode',
86     localElement: 'CurrencyCode' },
87   {
88     localElement: 'CurrencyCode' },
89   {
90     element: 'Price',
91     localElement: 'FlightPrice' }}]]
90 FlightDate;
91 @UI:{lineItem: [{ position: 70 }], 
92 identification: [{ position: 70 }] }
93 FlightPrice;
94 @Consumption.valueHelpDefinition: [{ entity: {
95   name: 'I_Currency',
96   element: 'Currency'
97 } }]
98 CurrencyCode;
99 @UI:{lineItem: [{ position: 80 }], 
100 identification: [{ position: 80 }], 
101 textArrangement: #TEXT_ONLY }
102 @Consumption.valueHelpDefinition: [{ entity: {
103   name: '/DMO/I_Booking_Status_VH',
104   element: 'BookingStatus'
105 } }]
106 BookingStatus;
107 @UI.hidden: true
108 LastChangedAt;
109
110 }

```

```

1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'booking suppl Projection'
3 @Metadata.ignorePropagatedAnnotations: true
4 @Metadata.allowExtensions: true
5 define view entity yC_booksuppl1
6   as projection on Yi_bookSuppl1_m
7 {
8   key TravelId,
9   key BookingId,
10  key BookingSupplementId,
11  @ObjectModel.text.element: [ 'supplementdesc' ]
12  SupplementId,
13  _supplementtext.Description as supplementdesc :localized,
14  @Semantics.amount.currencyCode: 'CurrencyCode'
15  Price,
16  CurrencyCode,
17  LastChangedAt,
18  /* Associations */
19  _booking : redirected to parent YC_booking1,
20  _supplement,
21  _supplementtext,
22  _travel : redirected to YC_travel1
23 }
24

1 @Metadata.layer: #CORE
2 @Search.searchable: true
3 @UI:{ headerInfo: {
4   typeName: 'BookingSuppl',
5   typeNamePlural: 'BookingSuppls',
6   title: {
7     type: #STANDARD,
8     label: 'Booking Supplement',
9     value: 'BookingSupplementId'
10   }
11 }}
```

**annotate entity yC\_booksuppl1 with**

```

14 {
15   @UI.facet: [{{
16     id: 'BookingSuppl',
17     purpose: #STANDARD,
18     position: 10 ,
19     label: 'Booking Supplements',
20     type:#IDENTIFICATION_REFERENCE
21   }]}
22
23 @Search.defaultSearchElement: true
24 TravelId;
25 @Search.defaultSearchElement: true
26 BookingId;
27 @UI: { lineItem: [{ position: 10 }],
28 identification: [{ position: 10 }]
29 }
```

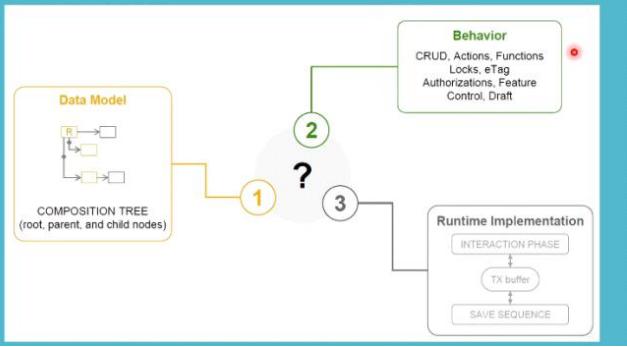
29

```

30
31 BookingSupplementId;
32 @UI: { lineItem: [{ position: 20 }],
33 identification: [{ position: 20 }]
34 }
35 @Consumption.valueHelpDefinition: [
36   { entity: {
37     name: '/dmo/i_supplement',
38     element: 'SupplementID'
39   },
40   additionalBinding: [{ element: 'SupplementID',localElement: 'SupplementId'},
41     { element: 'Price',localElement: 'Price'},
42     { element: 'CurrencyCode',localElement: 'CurrencyCode'}]
43   }]
44 SupplementId;
45 @UI: { lineItem: [{ position: 30 }],
46 identification: [{ position: 30 }]
47 }
48
49 Price;
50 @Consumption.valueHelpDefinition: [{ entity: {
51   name: 'I_Currency',
52   element: 'Currency'
53 } }]
54 CurrencyCode;
55 @UI.hidden: true
56 LastChangedAt;
```

57

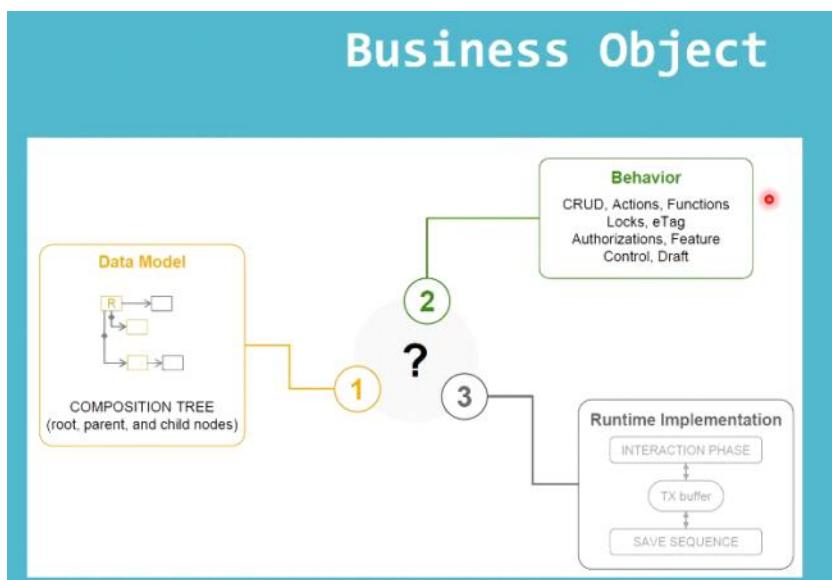
# Business Object



Now Done With Data Model

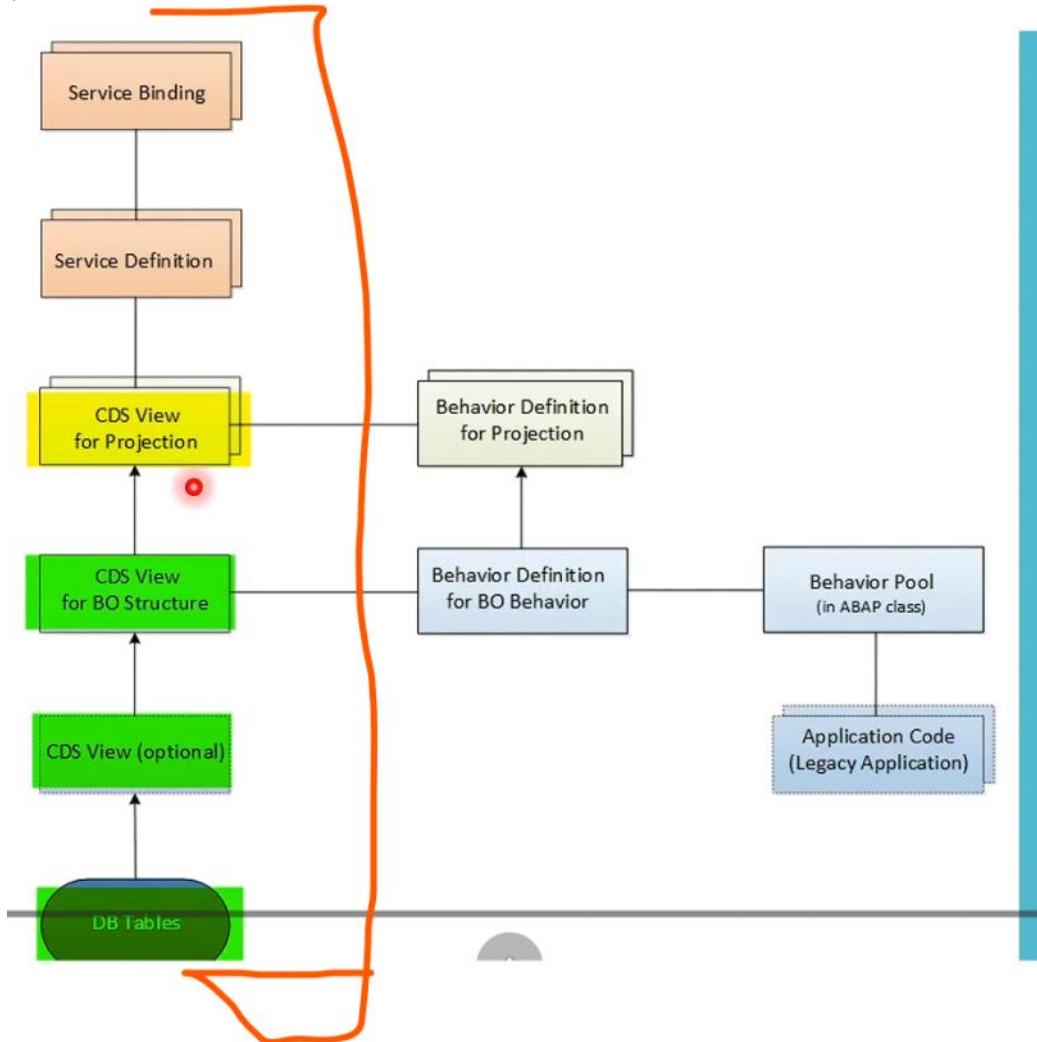
# Behavior

26 November 2025 18:50



Data model Done Previously

i.e



Next Do With Behavior

## BDL : Implementation type

```
[TRU] ZI_TRAVEL_TECH_M × [TRU] ZC_TRAVEL_TECH_M
1 managed implementation in class zbp_i_travel_tech_m unique;
2 strict ( 2 );
3
```

### ➤ Managed :-

- The transactional buffer and the standard BO operations are handled by the managed RAP BO provider.
- Recommended for development scenarios in which all essential parts are developed from scratch, without a large amount of existing code (also known as greenfield development) with standard implementation.

```
{ [implementation] managed
| unmanaged
| abstract
| projection
| interface ... }
```



### ➤ Unmanaged

- The transactional buffer and the standard BO operations must be implemented in the ABAP behavior pool.
- Recommended for development scenarios in which business logic already exists and is intended to be reuse (also known as **brownfield** development).

## BDL : Implementation type

### ➤ Projection

- A RAP projection behavior definition is based on a managed or unmanaged RAP BO and represents a direct projection of its base behavior definition.
- It exposes a **subset** of the base behavior definition's operations and characteristics. In a business application, a projection BDEF provides the means to define service-specific behavior for a BO projection.



### ➤ Interface

- Defines the behavior of a RAP BO interface, which serves as interface for **stable consumption** and is typically released as released API.

### ➤ Abstract

- A RAP abstract behavior definition mainly serves as typing mechanism for **deep action or function parameters**. Only a limited range of syntax elements is available, such as associations and type mapping. It is not possible to define any transactional behavior in an abstract behavior definition.

## RAP - BDEF Strict Mode

- It ensures that a RAP business object is lifecycle-stable, upgrade-save and best-practice compliant.
- It ensures that no outdated syntax is used, implicitly available operations are declared explicitly, and a RAP BO complies to best practices.
- Strict mode version 2 covers all rules from strict mode version 1, plus some additional checks.
- When strict mode is defined, some syntax checks that lead to warnings without strict mode then lead to runtime errors.
- Strict mode must always appear as the second line in a RAP behavior definition.
- RAP best practices are enforced, like, for example, authorization implementation is mandatory



# RAP - Entity Behavior Definition

```
define behavior for RootEntity [alias AliasName] [external ExternalName]
{[implementation in class ClazzName [unique]]}
entity behavior characteristics
{
  entity behavior body
}

{behavior for ChildEntity1}[, behavior for ChildEntity2][, ...]
```

```
4 define behavior for ZI_TRAVEL_TECH_M //alias <alias_name>
5 persistent table ztravel_tech_m
6 lock master
7 authorization master ( instance ) →
8 //etag master <field_name>
9 {
10   create;
11   update;
12   delete;
13   association _Booking { create; }
14 }
15
16*define behavior for ZI_BOOKING_TECH_M //alias <alias_name>
17 persistent table zbooking_tech_m
18 lock dependent by _Travel
19 authorization dependent by _Travel
```

- A RAP behavior definition can consist of one or more entity behavior definitions that start with define behavior for, each one referring to a different node of the business object.

➤ RootEntity must be included, child entities are optional.

➤ Strict mode :-

- No gaps are allowed in the behavior enablement of a composition tree
- CDS entities RootEntity, ChildEntity1 and so on, must be CDS view entities.



## ABAP Behavior Pools (ABP)

➤ The ABP is a special class pool for an ABAP behavior implementation and based on a BDEF.

➤ The global class of the behavior pool does not implement the behavior itself.

➤ The behavior implementation is coded in one or more local RAP handler classes and one RAP saver class.

➤ The implementation is coded in

- An interaction phase :- local handler class (LHC\_) (1..n)
- A save sequence    :- local saver class (LSC\_) (1)

```
1 managed ;
2 strict ( 2 );
3
4 define behavior for YI_TRAVEL_TECH_M //alias <alias_name>
5 implementation in class zbp_travel_tech_m unique
6 persistent table ytravel_tech_m
7 lock master
8 authorization master ( instance )
9 //etag master <field_name>
10 {
11   create; →
12   update;
13   delete;
14   association _Booking { create; }
15 }
```



## Syntax Global Class

➤ The addition FOR BEHAVIOR OF defines an ABAP behavior pool (ABP) for the BDEF specified in bdef.

➤ Name of class should start with BP

➤ The global class is implicitly ABSTRACT and FINAL

➤ Currently, global classes of behavior pools cannot be instantiated or inherited.

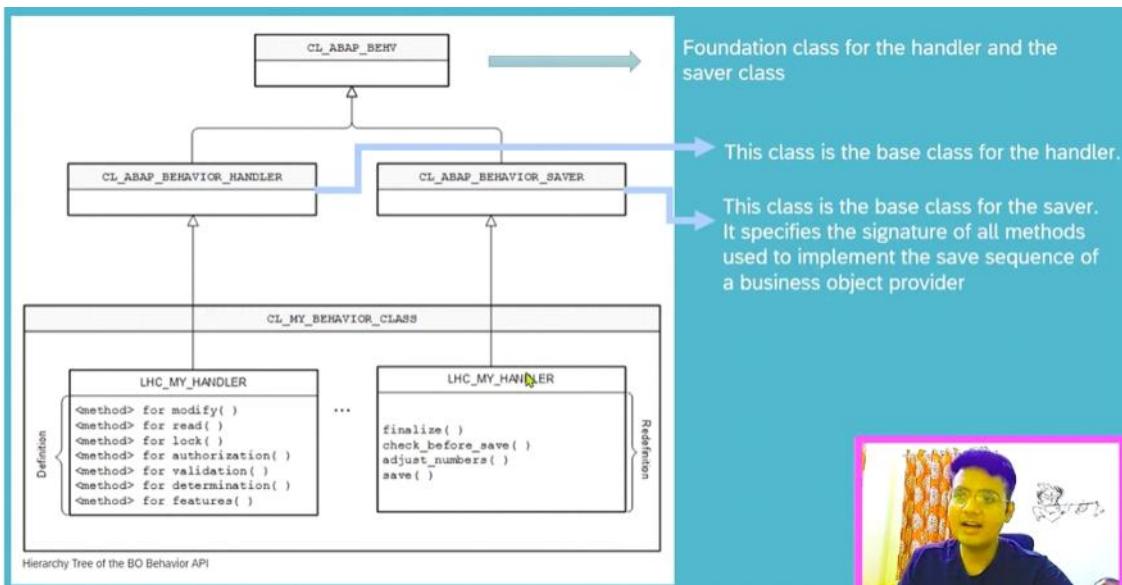
➤ The real substance of a behavior pool is located in Local Types.

➤ We define there

- Handler classes for the operations within the interaction phase and
- Saver classes for the operations within the save sequence.

```
CLASS class DEFINITION
  PUBLIC [ABSTRACT] [FINAL]
  FOR BEHAVIOR OF bdef.
  ...
ENDCLASS.
```





## CL\_ABAP\_BEHAVIOR\_HANDLER, RAP Handler Class

- A handler class can be defined in the CCIMP include of an ABAP behavior pool. It consists of method declarations and implementations.
- For modularization purposes, one behavior pool can define several handler classes. For example, each entity can have its own handler class
- Handler classes are implicitly ABSTRACT and FINAL since instantiating and calling only happens through the RAP runtime engine.

## RAP Handler Methods

```

METHODS meth [FINAL]
FOR { DETERMINE ON ( SAVE | MODIFY | ... [IMPORTING] ... FOR ... )
| { GLOBAL AUTHORIZATION ... [IMPORTING] ... FOR ... }
| { GLOBAL FEATURES ... [IMPORTING] ... FOR ... }
| { [INSTANCE] AUTHORIZATION ... [IMPORTING] ... FOR ... }
| { [INSTANCE] FEATURES ... [IMPORTING] ... FOR ... }
| { LOCK ... [IMPORTING] ... FOR ... }
| { MODIFY ... [IMPORTING] ... FOR ... }
| { NUMBERING ... [IMPORTING] ... FOR ... }
| { PRECHECK ... [IMPORTING] ... FOR ... }
| { READ ... [IMPORTING] ... FOR ... }
| { VALIDATE ON SAVE ... [IMPORTING] ... FOR ... }
[CHANGING { [failed TYPE data] [reported TYPE data] [mapped TYPE data] }].

```

## RAP Handler Methods

- The METHODS statements of handler methods in behavior pools contain RAP-specific ABAP words like FOR MODIFY, FOR CREATE or FOR READ
- Nearly all parameters are typed with BDEF **derived types** that have RAP components.
- Each handler method must have at least one importing parameter.
- All handler methods have changing parameters Ex :-
  - Mapped :- retrieving mapping information
  - Failed :- information on failures and
  - Reported :- information on error messages



## BDEF Derived Types

- BDEF derived types are special data types in the context of RAP.
- The types are derived by the ABAP runtime framework from CDS entities and their behavior definition in the BDEF.
- Used for the communication and to exchange data between RAP BO consumers and RAP BO providers,
- BDEF derived types contain
  - components of CDS entities that do derive their line type from the entity
  - additional RAP components to give additional information
- The names of those RAP components begin with %
- The parameters of RAP BO operations are mainly typed with BDEF derived types



## Declaration of Variables with BDEF Derived Types

- TYPE TABLE FOR:
  - DATA itab TYPE TABLE FOR CREATE entity\_name.
- We use below class to declare derived type :-
  - cl\_abap\_behvdescr=>create\_data( ).
- We can use below syntax as well :-
  - CREATE DATA dref TYPE (der\_type).
  - der\_type =>  
  \BDEF=bdef\_name\ENTITY=entity\_name\TYPE=operation

- TYPE TABLE FOR
- TYPE STRUCTURE FOR
- TYPE REQUEST FOR
- TYPE RESPONSE FOR

Induction

# Declaration of Variables with BDEF Derived Types

## Declaration of variables:

```
TYPES: create_type TYPE TABLE FOR CREATE demo_managed_root,
cba_type    TYPE TABLE FOR CREATE demo_managed_root\_\_child,
update_type TYPE TABLE FOR UPDATE demo_managed_root,
delete_type TYPE TABLE FOR DELETE demo_managed_root.

DATA: create_data TYPE TABLE FOR CREATE demo_managed_root,
cba_data     TYPE TABLE FOR CREATE demo_managed_root\_\_child,
update_data TYPE TABLE FOR UPDATE demo_managed_root,
delete_data TYPE TABLE FOR DELETE demo_managed_root.
```

## Creation of data objects:

```
CONSTANTS entity_name TYPE abp_entity_name VALUE 'DEMO_MANAGED_ROOT'.
DATA(ref) = cl_abap_behvdescr->create_data(
    p_name = entity_name
    p_op   = if_abap_behv=>op-m-create "or: p_op = 'C'
),
...
DATA ref2 TYPE REF TO data.
CREATE DATA ref2 TYPE
  ('BDEF=demo_MANAGED_ROOT\ENTITY=DEMO_MANAGED_CHILD\TYPE=FAILED').
```

# Persistent Table

- Managed RAP BOs mandatory.
- Exception unmanaged save
- Projection BO it automatically inherited
- If the BDEF specifies an ETag field, the persistent table requires a field that is used to describe the state of the database table. Ex :- timestamp
- If this field is annotated in CDS then it is automatically updated by the RAP framework

`@Semantics.systemDateTime.localInstanceLastChangedAt: true,`

- Table primary key field is of type raw(16) (UUID), then it can be filled by the RAP framework using (numbering:managed) (early internal numbering).

- If the field names of the persistent database table differ from the field names of the CDS views, then a type mapping is required in the entity behavior body.

```
*define behavior for YI_TRAVEL_TECH_M //alias <alias_name>
implementation in class zbp_travel_tech_m unique
persistent table ytravel_tech_m
lock master
authorization master ( instance )
//etag master <field_name>
{
  create;
  update;
  delete;
  association _Booking { create; }
}
```



## Behavior Definition

```
B [TRL] YI_TRAVEL1_M X B [TRL] YC_TRAVEL1  B [TRL] YUI_FLIGHT_V2
1 managed; //implementation in class zbp_i_travel1_m unique;
2 strict ( 2 );
3
4 define behavior for YI_travel1_M //alias <alias_name>
5 implementation in class zcl_bp_travel_managed unique
6 persistent table ytravel_m
7 lock master
8 authorization master ( instance )
9 //etag master <field_name>
10 {
11   create ( authorization : global );
12   update;
13   delete;
14   // field ( readonly ) TravelId;
15   association _booking { create; }
16 mapping for ytravel_m
17   {
18     TravelId      = travel_id;
19     AgencyId     = agency_id;
20     CustomerId   = customer_id;
21     BeginDate    = begin_date;
22     EndDate      = end_date;
23     BookingFee   = booking_fee;
24     TotalPrice   = total_price;
25     CurrencyCode = currency_code;
26     Description   = description;
27     OverallStatus = overall_status;
28     CreatedBy    = created_by;
29     CreatedAt    = created_at;
30     LastChangedBy = last_changed_by;
31     LastChangedAt = last_changed_at;
32   }
33 }
34
```

```

34
35@define behavior for YI_booking1_m //alias <alias_name>
36 implementation in class zcl_bp_Booking_managed unique
37 persistent table ybooking_m
38 lock dependent by _travel
39 authorization dependent by _travel
40 //etag master <field_name>
41 {
42   update;
43   delete;
44   // field ( readonly ) TravelId, BookingId;
45   field ( readonly ) TravelId;
46   association _travel;
47   association _bookingsuppl { create; }
48@ mapping for ybooking_m
49 {
50   TravelId      = travel_id;
51   BookingId     = booking_id;
52   BookingDate   = booking_date;
53   CustomerId    = customer_id;
54   CarrierId     = carrier_id;
55   ConnectionId  = connection_id;
56   FlightDate    = flight_date;
57   FlightPrice   = flight_price;
58   CurrencyCode  = currency_code;
59   BookingStatus = booking_status;
60   LastChangedAt = last_changed_at;
61 }
62 }
63

64@define behavior for Yi_bookSuppl1_m //alias <alias_name>
65 implementation in class zcl_bp_BookSuppl_managed unique
66 persistent table ybooksuppl_m
67 lock dependent by _travel
68 authorization dependent by _travel
69 //etag master <field_name>
70 {
71   update;
72   delete;
73   // field ( readonly ) TravelId, BookingId, BookingSupplementId;
74   field ( readonly ) TravelId, BookingId;
75   association _travel;
76   association _booking;
77@ mapping for ybooksuppl_m
78 {
79   TravelId      = travel_id;
80   BookingId     = booking_id;
81   BookingSupplementId = booking_supplement_id;
82   SupplementId  = supplement_id;
83   Price         = price;
84   CurrencyCode  = currency_code;
85   LastChangedAt = last_changed_at;
86 }
87 }

```

## RAP - etag

- Entity tag (ETag) field for optimistic concurrency control.
- Any changes made to the requested resource update the ETag field.
- If an entity is an ETag master entity, it has its own ETag field.
- An ETag dependent entity uses the ETag field of another entity
- No implementation require in Managed scenario
  - The ETag field must be updated reliably with every change on the RAP BO entity instance.
  - The ETag field MasterField must be annotated in CDS with the annotation `@Semantics.systemDateTime.localInstanceLastChangedAt: true`
  - The data type must be date compatible.
- Projection view we have to use "USE ETAG"

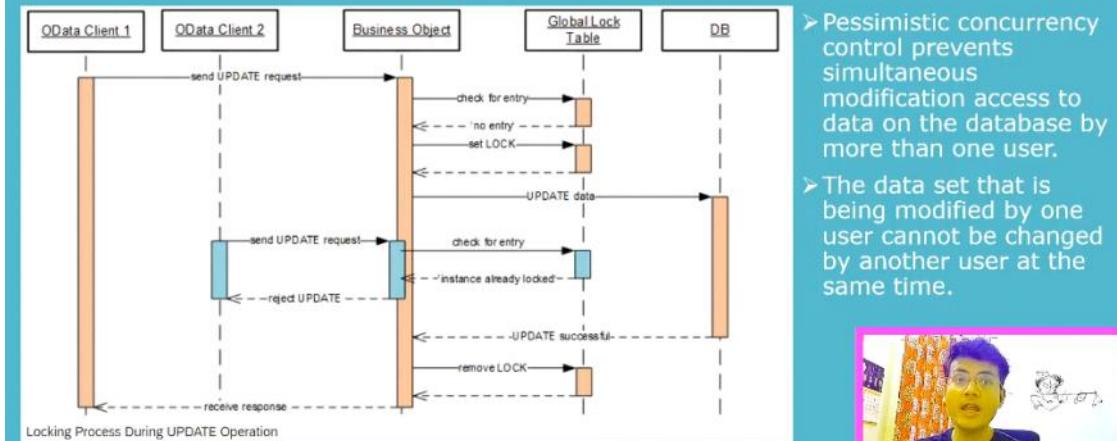
```

*define behavior for YI_TRAVEL_TECH_M //alias <alias_name>
implementation in class ybp_travel_tech_m unique
persistent table ytravel_tech_m
lock master
authorization master { instance }
//etag master <field_name>*
{
  create;
  update;
  delete;
  association _Booking { create; }
  mapping for ytravel_tech_m
}

```



# Pessimistic Concurrency Control (Locking)



## RAP – Locking Managed

- In a managed RAP BO, each entity must be defined either as **lock master** or as **lock dependent**. The lock mechanism is handled by the RAP framework.
- In a projection business object, the RAP locking mechanism that is defined and implemented for the base BO is automatically reused and does not need to be explicitly defined.
- The association from the lock dependent entity to the lock master entity must be explicitly defined in the entity behavior definition using the syntax `association _AssocToLockMaster { }.`

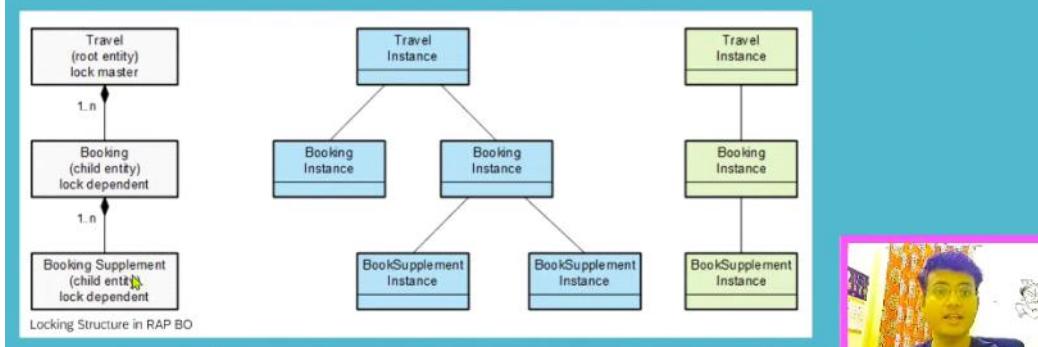
```
*define behavior for yI_BOOKSUPPL_TECH_M //alias <alias_name>
implementation in class ybp_booksuppl_tech_m unique
persistent table ybooksupp_tech_m
lock dependent by _Travel
authorization dependent by _Travel
//etag master <field_name>
{
  update;
  delete;
  field ( readonly ) TravelId, BookingId;
  association _Travel;
  association _Booking;
  mapping for ybooksupp_tech_m
}

projection;
strict ( 2 );
*define behavior for yc_TRAVEL_TECH_M //alias Travel
(
  use create;
  use update;
  use delete;

  use association _Booking { create; }
)
```



## Business object with lock master and lock dependent entities



## RAP - Locking

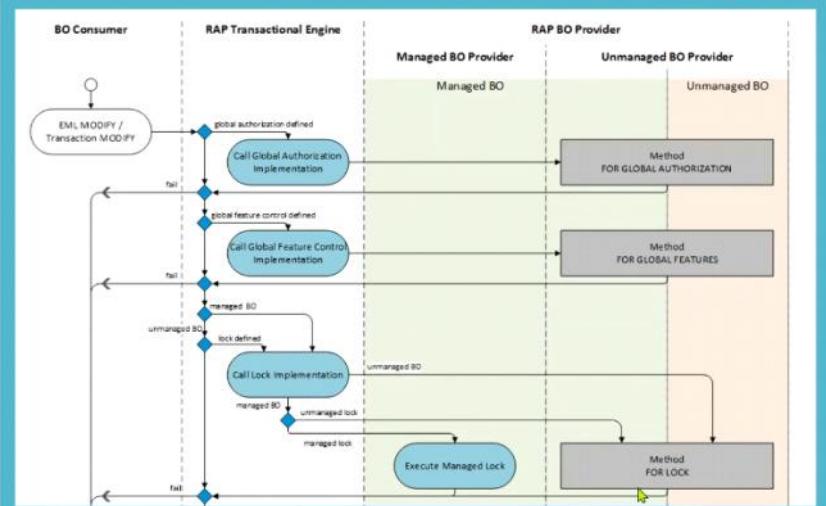
```
1 managed ;
2 strict ( 2 );
3
4#define behavior for YI_TRAVEL_TECH_M //alias <alias_name>
5 implementation in class ybp_travel_tech_m unique
6 persistent table ytravel_tech_m
7 lock master
8 authorization master ( instance )
9 //etag master <field_name>
9 {
10   create;
11
12
13#define behavior for YI_BOOKING_TECH_M //alias <alias_name>
14 implementation in class ybp_Booking_tech_m unique
15 persistent table ybooking_tech_m
16 lock dependent by _Travel
17 authorization dependent by _Travel
18 etag master LastChangedxt
19 {
20   update;
21   delete;
22   field ( readonly ) TravelId;
23   association _Travel;
```

- If a lock is defined for a RAP BO entity, it is invoked during the runtime of the following modify operations: - **Standard Operations & Actions**

- For lock dependent entities, any locking request is delegated to its lock master entity. Currently, only root entities can be defined as lock master entities



## RAP - Locking



AT Interface Level

```
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Travel Root Entity'
4 @Metadata.ignorePropagatedAnnotations: true
5 define root view entity YI_travell_M
6   as select from ytravel_m
7   composition [1..*] of YI_booking1_m as _booking
8   association [0..1] to /DMO/I_Agency as _agency on $projection.AgencyId = _agency.AgencyID
9   association [0..1] to /DMO/I_Customer as _customer on $projection.CustomerId = _customer.CustomerID
10  association [1..1] to I_Currency as _currency on $projection.CurrencyCode = _currency.Currency
11  association [1..1] to /DMO/I_Overall_Status_VH as _status on $projection.OverallStatus=_status.OverallStatus
12 {
13   key travel_id      as TravelId,
14   agency_id        as AgencyId,
15   customer_id      as CustomerId,
16   begin_date       as BeginDate,
17   end_date         as EndDate,
18   @Semantics.amount.currencyCode: 'CurrencyCode'
19   booking_fee      as BookingFee,
20   @Semantics.amount.currencyCode: 'CurrencyCode'
21   total_price       as TotalPrice,
22   currency_code     as CurrencyCode,
23   description       as Description,
24   overall_status    as OverallStatus,
25   created_by        as CreatedBy,
26   created_at        as CreatedAt,
27   last_changed_by   as LastChangedBy,
28   @Semantics.systemDateTime.localInstanceLastChangedAt: true
29   last_changed_at   as LastChangedAt,
30   _booking,
31   _agency,
32   _customer,
33   _currency,
34   _status
35 }
```

AT Behavior level of Root/interface level

```
1 managed; //implementation in class zbp_i_travel1_m unique;
2 strict ( 2 );
3
4@define behavior for YI_travel1_M //alias <alias_name>
5 implementation in class zcl_bp_travel_managed unique
6 persistent table ytravel_m
7 lock master
8 authorization master ( instance )
9 etag master LastChangedAt
10 {
11     create ( authorization : global );
12     update;
13     delete;
14     // field ( readonly ) TravelId;
15     association _booking { create; }
16@ mapping for ytravel_m
17     {
18         TravelId      = travel_id;
19
20
21 define behavior for YI_booking1_m //alias <alias_name>
22 implementation in class zcl_bp_Booking_managed unique
23 persistent table ybooking_m
24 lock dependent by _travel
25 authorization dependent by _travel
26 etag master LastChangedAt
27 //etag dependent by _travel
28 {
29     update;
30     delete;
31     // field ( readonly ) TravelId, BookingId;
32     field ( readonly ) TravelId;
33     association _travel;
34     association bookingsuppl { create; }
```

## AT Projection Behavior definition level

```
1 projection;
2 strict ( 2 );
3
4@define behavior for YC_travel1 //alias <alias_name>
5 use etag
6 {
7   use create;
8   use update;
9   use delete;
10
11  use association _booking { create; }
12 }
13
14@define behavior for YC_booking1 //alias <alias_name>
15 use etag
16 {
17   use update;
18   use delete;
19
20   use association _travel;
21   use association _bookingsuppl { create; }
22 }
23
24@define behavior for yC_booksuppl1 //alias <alias_name>
25 use etag
26 {
27   use update;
28   use delete;
29
30   use association _travel;
31   use association _booking;
32 }
```

## Numbering

27 November 2025 10:14

### RAP - Numbering

➤ Numbering is about setting values for primary key fields of entity instances during runtime.

- By When - early or late during the transactional processing
- By Whom - consumer, application developer, or framework

➤ **Early numbering** :- Primary key value is set instantly after the modify request for the CREATE is executed

➤ **Late numbering** :- Primary key values are always assigned internally without consumer interaction after the point of no return in the interaction phase has passed, and the SAVE sequence is triggered.

To get gap free keys.

### Early Numbering

➤ The key value can either be given by the consumer (externally) or by the framework (internally).

➤ Newly created BO instance is clearly identifiable directly after the CREATE operation has been executed

➤ Types based on who create

- External Early Numbering
- Internal Early Numbering

### External Early Numbering

➤ Consumer provide keys

➤ Keys should be unique

➤ RAP support **uniqueness** checks in early stages

➤ Primary key fields are not **read-only** at CREATE

➤ Read-only on further processing these instances

`field (mandatory:create| readonly:update);`

➤ Optional External Early Numbering

- Consumer hands over the primary key value or framework create a number on create
- Possible in managed business objects with UUID keys.

### Internal Early Numbering

➤ Managed Internal Early Numbering

- A UUID is drawn automatically during the CREATE request by the RAP managed runtime.
- It is only possible for key fields with ABAP type raw(16) (UUID)

`field ( readonly, numbering:managed ) KeyField1 [, KeyField2, ..., keyFieldn];`

➤ Unmanaged Internal Early Numbering

- In unmanaged BOs, the CREATE operation implements the assignment of a primary key value during the CREATE modification.
- The assignment of a new number must be included in your application code
- The key assignment is implemented in the **FOR NUMBERING method** in the behavior pool

## Late Numbering

- Primary key values are always assigned internally without consumer interaction after the point of no return in the interaction phase has passed, and the SAVE sequence is triggered.
- Late numbering is a common concept for drawing gap-free numbers. Ex :- We need for invoice numbers.
- Late numbering is internal by default
- Method **ADJUST NUMBERS** must be implemented
- Late numbering available for managed and unmanaged implementation scenarios with and without draft

## Numbering-Booking Entity

- The number range object for the travel entity guarantees unique keys for the root entity.
- For booking entities, the numbering starts from zero for every travel entity.
  - With a read-by-association all booking-ids are retrieved.
  - The highest booking number for the current travel is retrieved.
  - Bookings that already have an ID are assigned to mapped-booking and no new number is assigned.
  - Because the FOR NUMBERING method is called twice in a managed scenario with draft capabilities

FOR Header

```
28② METHOD earlynumbering_create.
40   DATA(lt_entities) = entities.
41   DELETE lt_entities WHERE TravelId IS NOT INITIAL.
42② TRY.
43   cl_numberrange_runtime=>number_get(
44     EXPORTING
45     *      ignore_buffer      =
46           nr_range_nr      = '01'
47           object          = '/DMO/TRV_M'
48           quantity        = CONV #( lines( lt_entities ) )
49② *
50   *      subobject       =
51           toyear         =
52   IMPORTING
53     number          = DATA(lv_latest_num)
54     returncode      = DATA(lv_code)
55     returned_quantity = DATA(lv_qty)
56   ).  

57   CATCH cx_nr_object_not_found.
58   CATCH cx_number_ranges INTO DATA(lo_error).
59② LOOP AT lt_entities INTO DATA(ls_entities).
60     APPEND VALUE #( %cid = ls_entities-%cid %key = ls_entities-%key )  

61     TO failed-yi_travell_m .
62     APPEND VALUE #( %cid = ls_entities-%cid
63                   %key = ls_entities-%key
64                   %msg = lo_error )
65     TO reported-yi_travell_m .
66
67   ENDLOOP.
68   EXIT.
69
70 ENDTRY.
```

71 ASSERT lv\_qty = lines( lt\_entities ).  
72
73 DATA(lv\_curr\_num) = lv\_latest\_num - lv\_qty.  
74② LOOP AT lt\_entities INTO ls\_entities.
75 lv\_curr\_num = lv\_curr\_num + 1.
76 APPEND VALUE #( %cid = ls\_entities-%cid TravelId = lv\_curr\_num ) TO mapped-yi\_travell\_m.
77 ENDLOOP.

78

79 ENDMETHOD.

80

FOR Booking Item

```

81 METHOD earlynumbering_cba_Booking.
82
83 DATA : lv_max_booking TYPE /dmo/booking_id.
84
85 READ ENTITIES OF yi_travell_m IN LOCAL MODE
86 ENTITY YI_travell_M BY \_booking
87 FROM CORRESPONDING #( entities )
88 LINK DATA(it_link_data).
89
90
91 LOOP AT entities ASSIGNING FIELD-SYMBOL(<ls_group_entity>) GROUP BY <ls_group_entity>-TravelId.
92   lv_max_booking = REDUCE #(
93     INIT lv_max = CONV /dmo/booking_id( '0' )
94     FOR ls_link IN it_link_data USING KEY entity
95       WHERE ( source-TravelId = <ls_group_entity>-TravelId )
96         NEXT lv_max = COND /dmo/booking_id( WHEN lv_max < ls_link-target-BookingId
97           THEN ls_link-target-BookingId
98           ELSE lv_max ) .
99
100  lv_max_booking = REDUCE #(
101    INIT lv_max = lv_max_booking
102    FOR ls_entity IN entities USING KEY entity
103      WHERE ( TravelId = <ls_group_entity>-TravelId )
104        FOR ls_booking IN ls_entity-%target
105          NEXT lv_max = COND /dmo/booking_id( WHEN lv_max < ls_booking-BookingId
106            THEN ls_booking-BookingId
107            ELSE lv_max ) .
108
109  LOOP AT entities ASSIGNING FIELD-SYMBOL(<ls_entities>) USING KEY entity
110    WHERE TravelId = <ls_group_entity>-TravelId.
111
112  LOOP AT <ls_entities>-%target ASSIGNING FIELD-SYMBOL(<ls_booking>).
113    APPEND CORRESPONDING #( <ls_booking> ) TO mapped-yi_booking1_m
114      ASSIGNING FIELD-SYMBOL(<ls_new_map_book>).
115      IF <ls_booking>-BookingId IS INITIAL.
116        lv_max_booking += 10.
117        <ls_new_map_book>-BookingId = lv_max_booking.
118      ENDIF.
119
120  ENDLOOP.
121
122 ENDMETHOD.
123
FOR Booking Suppl
124 METHOD earlynumbering_cba_Bookingsupp.
125
126 DATA : max_booking_suppl_id TYPE /dmo/booking_supplement_id.
127
128 READ ENTITIES OF yi_travell_m IN LOCAL MODE
129 ENTITY YI_booking1_m BY \_bookingsuppl
130 FROM CORRESPONDING #( entities )
131 LINK DATA(bookingsupplements).
132
133
134 LOOP AT entities ASSIGNING FIELD-SYMBOL(<booking_group>) GROUP BY <booking_group>-%tky.
135
136   " Get highest bookingsupplement_id from booking belong to booking
137   max_booking_suppl_id = REDUCE #(
138     INIT lv_max = CONV /dmo/booking_supplement_id( '0' )
139     FOR booksuppl IN bookingsupplements USING KEY entity
140       WHERE ( source-TravelId = <booking_group>-TravelId
141         AND source-BookingId = <booking_group>-BookingId )
142         NEXT lv_max = COND /dmo/booking_id( WHEN lv_max < booksuppl-target-BookingSupplementId
143           THEN booksuppl-target-BookingSupplementId
144           ELSE lv_max ) .
145
146   " Get highest assign bookingsupplement_id from incoming entities
147   max_booking_suppl_id = REDUCE #(
148     INIT lv_max = max_booking_suppl_id
149     FOR ls_entity IN entities USING KEY entity
150       WHERE ( TravelId = <booking_group>-TravelId
151         AND BookingId = <booking_group>-BookingId )
152         FOR target IN ls_entity-%target
153           NEXT lv_max = COND /dmo/booking_supplement_id( WHEN lv_max < target-BookingSupplementId
154             THEN target-BookingSupplementId
155             ELSE lv_max ) .
156
157   " Loop Over all entries in entities with the same travel id and booking id
158
159   " Loop Over all entries in entities with the same travel id and booking id
160
161   LOOP AT entities ASSIGNING FIELD-SYMBOL(<booking>) USING KEY entity
162     WHERE TravelId = <booking_group>-TravelId
163       AND BookingId = <booking_group>-BookingId.
164
165   LOOP AT <booking>-%target ASSIGNING FIELD-SYMBOL(<booksuppl_wo_numbers>).
166
167     APPEND CORRESPONDING #( <booksuppl_wo_numbers> ) TO mapped-yi_booksuppl1_m ASSIGNING FIELD-SYMBOL(<mapped_booksupl>).
168     " assign new booking supplement id
169     IF <booksuppl_wo_numbers>-BookingSupplementId IS INITIAL.
170       max_booking_suppl_id += 1.
171       <mapped_booksupl>-BookingSupplementId = max_booking_suppl_id.
172     ENDIF.
173
174   ENDLOOP.
175
176   ENDLOOP.
177
178 ENDMETHOD.

```

# Code For Numberings

30 November 2025 20:11

## 1) Early Numberings

Travel Id -----> Booking Id----->Booking Supplement ID

```
Travel ID
METHOD_earlynumbering_create.

DATA(lt_entities) = entities.
DELETE lt_entities WHERE TravelId IS NOT INITIAL.
TRY.
    cl_numberrange_runtime=>number_get(
        EXPORTING
        *      ignore_buffer      =
                nr_range_nr      = '01'
                object          = '/DMO/TRV_M'
                quantity        = CONV #( lines( lt_entities ) )
        *      subobject        =
        *      toyear          =
        IMPORTING
        number           = DATA(lv_latest_num)
        returncode       = DATA(lv_code)
        returned_quantity = DATA(lv_qty)
    ).
CATCH cx_nr_object_not_found.
CATCH cx_number_ranges INTO DATA(lo_error).

LOOP AT lt_entities INTO DATA(ls_entities).
    APPEND VALUE #( %cid = ls_entities-%cid %key =
ls_entities-%key )
        TO failed-yi_travel1_m .
    APPEND VALUE #( %cid = ls_entities-%cid
                    %key = ls_entities-%key
                    %msg = lo_error )
        TO reported-yi_travel1_m .
ENDLOOP.
EXIT.

ENDTRY.

ASSERT lv_qty = lines( lt_entities ).
```

```

DATA(lv_curr_num) = lv_latest_num - lv_qty.

*     DATA : lt_travel_m TYPE TABLE FOR MAPPED EARLY
yi_travel1_m,
*             ls_travel_m LIKE LINE OF lt_travel_m.

    LOOP AT lt_entities INTO ls_entities.
        lv_curr_num = lv_curr_num + 1.

        ls_travel_m = VALUE #( %cid = ls_entities-%cid
*                               TravelId = lv_curr_num ).

*
*     APPEND ls_travel_m TO lt_travel_m.

        APPEND VALUE #( %cid = ls_entities-%cid TravelId =
lv_curr_num ) TO mapped-yi_travel1_m.

    ENDLOOP.

    ENDMETHOD.

```

## Booking Id

```

METHOD_earlynumbering_cba_Booking.

    DATA : lv_max_booking TYPE /dmo/booking_id.

    READ  ENTITIES OF yi_travel1_m IN LOCAL MODE
ENTITY YI_travel1_M BY \_booking
FROM CORRESPONDING #( entities )
LINK DATA(lt_link_data).

    LOOP AT entities ASSIGNING FIELD-SYMBOL(<ls_group_entity>) GROUP BY
<ls_group_entity>-TravelId.
        lv_max_booking = REDUCE #( INIT lv_max = CONV /dmo/booking_id( '0' )
                                FOR ls_link IN lt_link_data USING KEY entity
                                WHERE ( source-TravelId = <ls_group_entity>-
TravelId )
                                NEXT lv_max = COND /dmo/booking_id( WHEN lv_max
< ls_link-target-BookingId
                                THEN
ls_link-target-BookingId
                                ELSE
lv_max ) ).

        lv_max_booking = REDUCE #( INIT lv_max = lv_max_booking
                                FOR ls_entity IN entities USING KEY entity
                                WHERE ( TravelId = <ls_group_entity>-
TravelId )
                                FOR ls_booking IN ls_entity-%target
                                NEXT lv_max = COND /dmo/booking_id( WHEN
lv_max < ls_booking-BookingId
                                THEN ls_booking-BookingId

```

```

ELSE lv_max ) ).

LOOP AT entities ASSIGNING FIELD-SYMBOL(<ls_entities>) USING KEY entity
      WHERE TravelId =
<ls_group_entity>-TravelId.

      LOOP AT <ls_entities>-%target ASSIGNING FIELD-SYMBOL(<ls_booking>).
          APPEND CORRESPONDING #( <ls_booking> ) TO mapped-yi_booking1_m
              ASSIGNING FIELD-SYMBOL(<ls_new_map_book>).
          IF <ls_booking>-BookingId IS INITIAL.
              lv_max_booking += 10.
              <ls_new_map_book>-BookingId = lv_max_booking.
          ENDIF.

      ENDLOOP.

      ENDLOOP.

ENDLOOP.

ENDMETHOD.

```

## Booking supplement id

```

METHOD earlynumbering_cba_Bookingsupp.

DATA : max_booking_suppl_id TYPE /dmo/booking_supplement_id.

READ ENTITIES OF yi_travel1_m IN LOCAL MODE
ENTITY YI_booking1_m BY \_bookingsuppl
FROM CORRESPONDING #( entities )
LINK DATA(bookingsupplements).

LOOP AT entities ASSIGNING FIELD-SYMBOL(<booking_group>) GROUP BY
<booking_group>-%tky.

    " Get highest bookingsupplement_id from booking belong to booking

    max_booking_suppl_id = REDUCE #( INIT lv_max = CONV
/dmo/booking_supplement_id( '0' )
                                FOR booksuppl IN bookingsupplements USING
KEY entity
                                WHERE ( source-TravelId = <booking_group>-
TravelId
                                         AND source-BookingId =
<booking_group>-BookingId )
                                NEXT lv_max = COND /dmo/booking_id( WHEN
lv_max < booksuppl-target-BookingSupplementId
                                         THEN
booksuppl-target-BookingSupplementId
                                         ELSE
lv_max ) ).

    " Get highest assign bookingsupplement_id from incoming entities
    max_booking_suppl_id = REDUCE #( INIT lv_max = max_booking_suppl_id
                                FOR ls_entity IN entities USING KEY entity
                                WHERE ( TravelId =
<booking_group>-TravelId
                                         AND
BookingId = <booking_group>-BookingId )
                                FOR target IN ls_entity-%target

```

```

NEXT lv_max = COND
/dmo/booking_supplement_id( WHEN lv_max < target-BookingSupplementId
THEN target-BookingSupplementId
ELSE lv_max ) ).
" Loop Over all entries in entities with the same travel id and boeing id

LOOP AT entities ASSIGNING FIELD-SYMBOL(<booking>) USING KEY entity
WHERE TravelId =
<booking_group>-TravelId
AND BookingId =
<booking_group>-BookingId.

LOOP AT <booking>-%target ASSIGNING FIELD-SYMBOL(<booksuppl_wo_numbers>).

APPEND CORRESPONDING #( <booksuppl_wo_numbers> ) TO mapped-
yi_booksuppl_m ASSIGNING FIELD-SYMBOL(<mapped_booksupl>).
" assign new booking supplement id
IF <booksuppl_wo_numbers>-BookingSupplementId IS INITIAL.
max_booking_suppl_id += 1.
<mapped_booksupl>-BookingSupplementId = max_booking_suppl_id.
ENDIF.

ENDLOOP.

ENDLOOP.
ENDLOOP.

ENDMETHOD.

```

# Reading

27 November 2025 21:38

## Read Variant

### ➤ READ ENTITY, Short Form

- The short form of the READ statement allows reading multiple instances of a single entity

```
READ ENTITY [IN LOCAL MODE|PRIVILEGED] entity operations [response_param].
```

### ➤ READ ENTITIES, Long Form

- The long form of the READ statement allows reading multiple instances of multiple entities.

```
READ ENTITIES OF bdef [IN LOCAL MODE|PRIVILEGED]
  ENTITY entity1 operations
  [ENTITY entity2 operations]
  [...]
  [response_param].
```

### ➤ READ ENTITIES OPERATIONS, Dynamic Form

- The dynamic form of the READ statement allows collecting multiple instances to be read in multiple entities in one READ statement.

```
READ ENTITIES [IN LOCAL MODE|PRIVILEGED] OPERATIONS op_tab [response_param].
```



```
17
18 * Sort Form Read
19
20   READ ENTITY yi_travel1_m
21     FROM VALUE #( ( %key-TravelId = '00000004'
22                   %control = VALUE #( AgencyId = if_abap_behv=>mk-on
23                               CustomerId = if_abap_behv=>mk-on
24                               BeginDate = if_abap_behv=>mk-on ) ) )
25     RESULT DATA(lt_data)
26     FAILED DATA(lt_fail) .
27
28 ⊕ IF lt_fail IS NOT INITIAL.
29   out->write( 'read File' ).
30 ELSE.
31   out->write( lt_data ).
32 ENDIF.
33
34
35   READ ENTITY yi_travel1_m
36     FIELDS ( AgencyId CustomerId BeginDate EndDate BookingFee TotalPrice )
37     WITH VALUE #( ( %key-TravelId = '00000004' )
38                   ( %key-TravelId = '00000001' ) )
39     RESULT DATA(lt_data)
40     FAILED DATA(lt_fail) .
41
42   READ ENTITY yi_travel1_m
43     ALL FIELDS
44     WITH VALUE #( ( %key-TravelId = '00000004' )
45                   ( %key-TravelId = '00000001' ) )
46     RESULT DATA(lt_data)
47     FAILED DATA(lt_fail) .
```

```

49 READ ENTITY yi_travel1_m
50 by \_booking|
51 ALL FIELDS
52 WITH VALUE #( ( %key-TravelId = '00000004' )
53 ( %key-TravelId = '00000001' ) )
54 RESULT DATA(lt_data)
55 FAILED DATA(lt_fail) .
56
57 IF lt_fail IS NOT INITIAL.
58   out->write( 'read File' ).
59 ELSE.
60   out->write( lt_data ).
61 ENDIF.

62
63
64 * Longer Form
65
66 READ ENTITIES OF yi_travel1_m
67
68 ENTITY YI_travel1_M
69   ALL FIELDS WITH VALUE #( ( %key-TravelId = '00000004' )
70 ( %key-TravelId = '00000001' ) )
71   RESULT DATA(lt_travel)
72
73 ENTITY YI_booking1_m
74   ALL FIELDS WITH VALUE #( ( %key-TravelId = '00000001'
75 %key-BookingId = '0001' ) )
76   RESULT DATA(lt_booking)
77
78   FAILED DATA(lt_fail) .
79
80 IF lt_fail IS NOT INITIAL.
81   out->write( 'read File' ).
82 ELSE.
83   out->write( lt_travel ).
84   out->write( lt_booking ).
85 ENDIF.
86

79
80 DATA: it_optab          TYPE abp\_behv\_retrievals\_tab,
81       it_travel           TYPE TABLE FOR READ IMPORT yi_travel_tech_m,
82       it_travel_result     TYPE TABLE FOR READ RESULT yi_travel_tech_m.
83
84 it_travel = VALUE #( ( %key-TravelId = '0000004172'
85                      %control = VALUE #( AgencyId      = if_abap_behv=>mk-on
86                               customerid    = if_abap_behv=>mk-on
87                               begindate    = if_abap_behv=>mk-on
88 ) ) .
89
90 it_optab = VALUE #( ( op = if_abap_behv=>op-r
91                      entity_name = 'YI_TRAVEL_TECH_M'
92                      instances = REF #( it_travel )
93                      results   = REF #( it_travel_result ) ) ) .
94
95 READ ENTITIES
96   OPERATIONS it_optab
97   FAILED DATA(lt_failed_dy).
98
99 IF lt_failed_dy IS NOT INITIAL.
100  out->write( 'Read failed' ).
```

```
99*      IF lt_failed_dy IS NOT INITIAL.  
100         out->write( 'Read failed' ).  
101  
102     ELSE.  
103         out->write( it_travel_result ).  
104 *         out->write( lt_result_book ).  
105     ENDIF.  
106
```

## READ Options

### ➤ IN LOCAL MODE –

- The addition is used to exclude feature controls and authorization checks
- It can currently only be used in RAP BO implementations for the particular RAP BO itself, i. e. not for other RAP BO's

### ➤ PRIVILEGED-

- For a privileged access to a RAP BO
- Authorization Can be skipped



# Code

03 December 2025 11:10

```
READ ENTITIES OF yi_travel1_m IN LOCAL MODE
ENTITY yi_travel1_m
ALL FIELDS WITH CORRESPONDING #( keys )
RESULT DATA(lt_result).
```

# Modify

01 December 2025 09:06

## MODIFY ENTITY, ENTITIES

- Used to perform RAP modify operations on RAP BO instances.
- This includes standard operations (CREATE, CREATE BY, UPDATE, DELETE) and nonstandard operations (actions) using the keyword EXECUTE.
  - MODIFY ENTITY, Short Form :- The short form of the MODIFY statement allows modifying instances of a single entity.
  - MODIFY ENTITIES, Long Form :- The long form of the MODIFY statement allows modifying instances of multiple entities.
  - MODIFY ENTITIES OPERATIONS, Dynamic Form :- The dynamic form of the MODIFY statement allows collecting multiple instances to be modified in multiple entities in one MODIFY statement.

## MODIFY ENTITY, ENTITIES, operations

- CREATE: Create new RAP BO node instances for root or child entities.
- CREATE BY: Create target instances for associated entities for which create must be enabled in the BDEF. The creation is not restricted to compositions.
- UPDATE: Update existing RAP BO node instances.
- DELETE: Delete existing RAP BO node instances. Note that the deletion also affects node instances along the composition.
- EXECUTE actions: Carry out user-defined modify operations.
- All modify operations must be specified in the RAP behavior definition.

```
... [CREATE field_spec]
[CREATE BY \_assoc field_spec]
[UPDATE field_spec]
[DELETE field_spec]
[EXECUTE action field_spec [REQUEST request] [RESULT result_tab]] ...
```

## Important Points

- A RAP BO provider must not use ABAP EML MODIFY statements in the RAP saver methods check\_before\_save, adjust\_numbers, save, save\_modified and cleanup. The statements can only be used in the RAP interaction phase and in the finalize saver method in the RAP save sequence. The statements can only be used in the RAP interaction phase and in the finalize saver method in the RAP save sequence.
- RAP modify standard operations do not return result parameters, whereas non-standard operations do so. The changed data can be read via a subsequent ABAP EML READ statement.
- If ABAP EML modify statements are used outside of behavior pools, for example, in ABAP programs, a COMMIT ENTITIES statement is necessary to persist data to the database. modify operations are only performed on the transactional buffer. Hence, a manual triggering of the save sequence is required.
- Do not call in Loop

## CREATE

```

1@ CLASS ycl_modify_practices DEFINITION PUBLIC .
2
3  PUBLIC SECTION.
4    INTERFACES : if_oo_adt_classrun.
5
6  ENDCLASS.
7
8@ CLASS ycl_modify_practices IMPLEMENTATION.
9@   METHOD if_oo_adt_classrun~main.
10
11    MODIFY ENTITY YI_travel1_M
12
13      CREATE FROM VALUE #( ( %cid = 'cid2'
14        %data-BeginDate = '20251201'
15        %control-BeginDate = if_abap_behv=>mk-on ) )
16
17      FAILED FINAL(it_failed)
18      MAPPED FINAL(it_mapped)
19      REPORTED FINAL(it_reported).
20
21    COMMIT ENTITIES.
22
23@   IF it_failed IS NOT INITIAL.
24     out->write( it_failed ).
25   ELSE.
26     out->write( 'created.' ).
27   ENDIF.
28
29  ENDMETHOD.
30 ENDCLASS.

```

```

@11  MODIFY ENTITY YI_travel1_M
12
13  CREATE FROM VALUE #( ( %cid = 'cid1'
14    %data-BeginDate = '20251201'
15    %control-BeginDate = if_abap_behv=>mk-on ) )
16
17  | CREATE BY \_booking
18    FROM VALUE #( ( %cid_ref = 'cid1'
19      %target = VALUE #( ( %cid = 'cid2' %data-BookingDate = '20251201'
20        %control-BookingDate = if_abap_behv=>mk-on ) ) ) )
21
22
23  FAILED DATA(lt_fail)
24  MAPPED DATA(lt_data)
25  REPORTED FINAL(lt_re).
26
27  COMMIT ENTITIES.
28
29
30
31 * Auto Fill CID
32
33  MODIFY ENTITY YI_travel1_M
34    CREATE AUTO FILL CID WITH VALUE #( ( %data-BeginDate = '20251212' %control-BeginDate = if_abap_behv=>mk-on ) )
35    FAILED DATA(lt_fail)
36    MAPPED DATA(lt_data)
37    REPORTED FINAL(lt_re).
38
39  COMMIT ENTITIES.
40
41

```

## DELETE

```

21
22  MODIFY ENTITY YI_travel1_M
23    DELETE FROM VALUE #( ( %key-TravelId = '4323' ) )
24
25    FAILED DATA(lt_fail)
26    MAPPED DATA(lt_data)
27    REPORTED FINAL(lt_re).
28
29  COMMIT ENTITIES.
30

```

Item deleted

```

21
22  MODIFY ENTITY YI_booking1_m
23    DELETE FROM VALUE #( ( %key-TravelId = '4320'
24      %key-BookingId = '10' ) )
25
26    FAILED DATA(lt_fail)
27    MAPPED DATA(lt_data)
28    REPORTED FINAL(lt_re).
29
30  COMMIT ENTITIES.

```

## UPDATE

```
42 * Update
43
44     MODIFY ENTITY YI_travel1_M
45         UPDATE FIELDS ( BeginDate )
46             WITH VALUE #( ( %key-TravelId = '00004329'
47                 %data-BeginDate = '20251215' ) )
48
49             FAILED DATA(lt_fail)
50             MAPPED DATA(lt_data)
51             REPORTED FINAL(lt_re).
52             COMMIT ENTITIES.|
```

```
56     MODIFY ENTITY YI_travel1_M
57         UPDATE SET FIELDS WITH VALUE #( ( %key-TravelId = '00000004'
58                                         BeginDate = '20251212' ) )
59             FAILED DATA(lt_fail)|
60             MAPPED DATA(lt_data)
61             REPORTED FINAL(lt_re).
62
63             COMMIT ENTITIES.
```

## LONGER FORM

```
44     MODIFY ENTITIES OF YI_travel1_M
45         ENTITY YI_travel1_M
46             UPDATE FIELDS ( BeginDate )
47                 WITH VALUE #( ( %key-TravelId = '00004329'
48                     %data-BeginDate = '20251215' ) )
49         ENTITY YI_travel1_M
50             DELETE FROM VALUE #( ( TravelId = '4329' ) )
51
52             FAILED DATA(lt_fail)
53             MAPPED DATA(lt_data)
54             REPORTED FINAL(lt_re).
```

## NOT Advice

```
56     MODIFY ENTITIES OF YI_travel1_M
57     ENTITY YI_travel1_M
58     UPDATE SET FIELDS WITH VALUE #( ( %key-TravelId = '00000004' BeginDate = '20251201' ) )
59         FAILED DATA(lt_fail)
60         MAPPED DATA(lt_data)
61         REPORTED FINAL(lt_re).
62
63     COMMIT ENTITIES.
```

# Codes

01 December 2025 10:12

## 1) Create

```
MODIFY ENTITY YI_travel1_M

CREATE FROM VALUE #( ( %cid = 'cid1'
                      %data-BeginDate = '20251201'
                      %control-BeginDate = if_abap_behv=>mk-on ) )

CREATE BY \_booking
FROM VALUE #( ( %cid_ref = 'cid1'
                  %target = VALUE #( ( %cid = 'cid2' %data-BookingDate =
'20251201'
                  %control-BookingDate = if_abap_behv=>mk-on ) ) ) )

FAILED DATA(lt_fail)
MAPPED DATA(lt_data)
REPORTED FINAL(lt_re).

COMMIT ENTITIES.

IF lt_fail IS NOT INITIAL.
  out->write( lt_fail ).
ELSE.

  out->write( 'created.' ).
ENDIF.
```

## 2) Delete

```
MODIFY ENTITY YI_travel1_M
DELETE FROM VALUE #( ( %key-TravelId = '4323' ) )

FAILED DATA(lt_fail)
MAPPED DATA(lt_data)
REPORTED FINAL(lt_re).

COMMIT ENTITIES.
```

### Item delete

```
MODIFY ENTITY YI_booking1_m
DELETE FROM VALUE #( ( %key-TravelId = '4320'
                      %key-BookingId = '10' ) )

FAILED DATA(lt_fail)
MAPPED DATA(lt_data)
REPORTED FINAL(lt_re).
```

## 3) Update

```

MODIFY ENTITY YI_travel1_M
    UPDATE FIELDS ( BeginDate )
    WITH VALUE #( ( %key-TravelId = '00004329'
                    %data-BeginDate = '20251215' ) )

        FAILED DATA(lt_fail)
        MAPPED DATA(lt_data)
        REPORTED FINAL(lt_re).
    COMMIT ENTITIES.

```

## Longer Form

```

MODIFY ENTITIES OF YI_travel1_M
    ENTITY YI_travel1_M
        UPDATE FIELDS ( BeginDate )
        WITH VALUE #( ( %key-TravelId = '00004329'
                        %data-BeginDate = '20251215' ) )

    ENTITY YI_travel1_M
        DELETE FROM VALUE #( ( TravelId = '4329' ) )

        FAILED DATA(lt_fail)
        MAPPED DATA(lt_data)
        REPORTED FINAL(lt_re).

```

# Actions

30 November 2025 13:28

## In Behavior Definition of Interface Root

```
1 managed;//implementation in class zbp_i_std_table unique;
2 strict ( 2 );
3
4 //with draft;
5
6@define behavior for ZI_std_table alias Student
7 implementation in class zbp_std_table unique
8
9 //draft table zdraft_std_table
10
11 persistent table ystd_table
12 lock master //total etag Lastchangedat
13 authorization master ( instance )
14 //etag master Lastchangedat
15
16
17 {
18   create ( authorization : global );
19   update;
20   delete;
21   field ( numbering : managed, readonly ) Id;
22   association _academics { create; }
23
24   // action ( features : instance ) SetAmdmitted result [1] $self;
25
26   action ( features : instance ) Approve result [1] $self;
27 }
```

## In Behavior Definition of Projection Root

```
1 projection;
2 strict ( 2 );
3
4@define behavior for ZP_std_tab //alias <alias_name>
5 {
6   use create;
7   use update;
8   use delete;
9
10  use association _academics { create; }
11
12  use action Approve;
13
14 }
15
16@define behavior for ZP_res_tab //alias <alias_name>
17 {
18   use update;
19   use delete;
20
21  use association _Student;
22 }
```

## In class

```

1@ CLASS lhc_Student DEFINITION INHERITING FROM cl_abap_behavior_handler.
2  PRIVATE SECTION.
3
4    METHODS get_instance_features FOR INSTANCE FEATURES
5      IMPORTING keys REQUEST requested_features FOR Student RESULT result.
6
7    METHODS get_instance_authorizations FOR INSTANCE AUTHORIZATION
8      IMPORTING keys REQUEST requested_authorizations FOR Student RESULT result.
9
10   METHODS get_global_authorizations FOR GLOBAL AUTHORIZATION
11     IMPORTING REQUEST requested_authorizations FOR Student RESULT result.
12
13   METHODS Approve FOR MODIFY
14     IMPORTING keys FOR ACTION Student~Approve RESULT result.
15
16   METHODS ValidateAge FOR VALIDATE ON SAVE
17     IMPORTING keys FOR Student~ValidateAge.
18
19   METHODS UpdateCourseDuration FOR DETERMINE ON SAVE
20     IMPORTING keys FOR Student~UpdateCourseDuration.
21
22 ENDCLASS.

```

```

43
46@ METHOD Approve.
47   MODIFY ENTITIES OF ZI_std_table IN LOCAL MODE
48   ENTITY Student
49   UPDATE
50   FIELDS ( Status )
51   WITH VALUE #( FOR key IN keys ( %tky = key-%tky Status = abap_true ) )
52   FAILED failed
53   REPORTED reported.
54
55
56   READ ENTITIES OF ZI_std_table IN LOCAL MODE
57   ENTITY Student
58   ALL FIELDS WITH CORRESPONDING #( keys )
59   RESULT DATA(studentdata).
60
61   result = VALUE #( FOR studentrec IN studentdata ( %tky = studentrec-%tky %param = studentrec ) ).  

62 ENDMETHOD.
63

64
25@ METHOD get_instance_features.
26   READ ENTITIES OF ZI_std_table IN LOCAL MODE
27   ENTITY Student
28   FIELDS ( Status ) WITH CORRESPONDING #( keys )
29   RESULT DATA(studentadmitted)
30   FAILED failed.
31
32   result = VALUE #( FOR stud IN studentadmitted
33                     LET stausval = COND #( WHEN stud>Status = abap_true THEN if_abap_behv=>fc-o-disabled
34                               ELSE if_abap_behv=>fc-o-enabled )
35
36                     IN ( %tky = stud-%tky %action=Approve = stausval
37                         ).
38 ENDMETHOD.
39

```

## In Metada Extension

```

43
44  @UI: { lineItem: [{ position: 70, label: 'Status' },
45                    { type: '#FOR_ACTION,dataAction: 'Approve',label: 'APPROVE',importance: #HIGH }
46                  ],
47  identification: [{ position: 70,label: 'Status' }
48                  ] }
49
50 Status;
51

```

## Validations

30 November 2025 13:27

```
1 managed; //implementation in class zbp_i_std_table unique;
2 strict ( 2 );
3
4 //with draft;
5
6@ define behavior for ZI_std_table alias Student
7 implementation in class zbp_std_table unique
8
9 //draft table zdraft_std_table
10
11 persistent table ystd_table
12 lock master //total etag Lastchangedat
13 authorization master ( instance )
14 //etag master Lastchangedat
15
16
17 {
18   create ( authorization : global );
19   update;
20   delete;
21   field ( numbering : managed, readonly ) Id;
22   association _academics { create; }
23   action ( features : instance ) Approve result [1] $self;
24
25 validation ValidateAge on save { create; field Age; }
26
27
28
29
```

### In Class

METHODS ValidateAge FOR VALIDATE ON SAVE  
IMPORTING keys FOR Student~ValidateAge.

```
64@ METHOD ValidateAge.
65   READ ENTITIES OF ZI_std_table IN LOCAL MODE
66   ENTITY Student
67   FIELDS ( age ) WITH CORRESPONDING #( keys )
68   RESULT DATA(studentsAge).
69
70@ LOOP AT studentsage INTO DATA(studentage).
71@   IF studentage-Age < 21.
72     APPEND VALUE #( %tky = studentage-%tky ) TO failed-student.
73     APPEND VALUE #( %tky = keys[ 1 ]-%tky
74       %msg = new_message_with_text( severity = if_abap_behv_message=>severity-error
75                               text = 'Age can not be less than 21' ) ) TO reported-student.
76   ENDIF.
77   ENDLOOP.
78 ENDMETHOD.
79
```

### What is Prechecks

**Prechecks** is used to validate data before it can reach to Transactional Buffer.

You can prevent illegal changes from reaching the transactional buffer by prechecking modify operations.

During **PRECHECK** implementation, we can implement validation logic and based on result generate FAILED and REPORTED parameters which can be used to display Error on Fiori frontend.

### Advantage

If application is a draft enabled application, then these inconsistent values have already been stored in draft tables and transactional buffers.

### How to Implement Prechecks

**Prechecks** can be defined in the behavior definition and implemented it in the behavior implementation class.

create ( precheck );

update ( prechecks );

\*precheck

# Determinations

30 November 2025 13:42

Two Types OF Determinations

- 1) On Save Determination
- 2) On Modify Determination

```
{  
    create ( authorization : global );  
    update;  
    delete;  
    field ( numbering : managed, readonly ) Id;  
    association _academics { create; }  
    action ( features : instance ) Approve result [1] $self;  
  
    validation ValidateAge on save { create; field Age; }  
  
    determination UpdateCourseDuration on save { field Course; }  
  
}
```

## In Class

```
METHODS UpdateCourseDuration FOR DETERMINE ON SAVE  
    IMPORTING keys FOR Student~UpdateCourseDuration.
```

```
80  
81@  METHOD UpdateCourseDuration.  
82  
83      READ ENTITIES OF zi_std_table IN LOCAL MODE  
84      ENTITY Student  
85      FIELDS ( Course ) WITH CORRESPONDING #( keys )  
86      RESULT DATA(StudentsCourse).  
87  
88@  LOOP AT studentscourse INTO DATA(studentcourse).  
89  
90@  IF studentcourse-Course = 'Computers'.  
91      MODIFY ENTITIES OF ZI_std_table IN LOCAL MODE  
92      ENTITY Student  
93      UPDATE  
94      FIELDS ( Courseduration ) WITH VALUE #( ( %tky = studentcourse-%tky Courseduration = 5 ) ).  
95  ELSEIF studentcourse-Course = 'Electronics'.  
96      MODIFY ENTITIES OF ZI_std_table IN LOCAL MODE  
97      ENTITY Student  
98      UPDATE  
99      FIELDS ( Courseduration ) WITH VALUE #( ( %tky = studentcourse-%tky Courseduration = 3 ) ).  
100  ENDIF.  
101  
102  ENDLOOP.  
103  
104  ENDMETHOD.  
105  
106  ENDCLASS.
```

### **What is Side Effect**

When user makes a change to field on UI, and this change effect the content of other field, this behavior is called Side Effect.

### **How to Implement Side Effects**

There are Annotations available which helps Implementing Side Effects behavior on UI

Source properties	Properties on which you want to trigger Side Effect behavior
Target properties	Properties you want to manipulate as a result of Side Effects

# Virtual Elements

30 November 2025 13:48

## Provide Only In Projection/Consumptions Not In Interface

```
1 5 @Metadata.ignorePropagatedAnnotations: true
2 4 @Metadata.allowExtensions: true
3 5 define root view entity ZP_std_tab
4 6 provider contract transactional_query
5 7 as projection on ZI_std_table
6 8 {
7 9     @EndUserText.label: 'Student Id'
8 10    key      Id,
9 11        @EndUserText.label: 'First Name'
10 12        Firstname,
11 13        @EndUserText.label: 'Last name'
12 14        Lastname,
13 15        @EndUserText.label: 'Age'
14 16        Age,
15 17        @EndUserText.label: 'Course'
16 18        Course,
17 19        @EndUserText.label: 'Course Duration'
18 20        Courseduration,
19 21        @EndUserText.label: 'Status'
20 22        Status,
21 23        @EndUserText.label: 'Gender'
22 24        Gender,
23 25        @EndUserText.label: 'Date Of Birth'
24 26        Dob,
25 27        @EndUserText.label: 'Last Changed'
26 28        Lastchangedat,
27 29        @EndUserText.label: 'Local Last Changed'
28 30        Locallastchangedat,
29 31        _academics : redirected to composition child ZP_res_tab,
30 32
31 33 @ObjectModel.virtualElementCalculatedBy: 'ABAP:ZCL_VE_CALCULATIONS'
32 34        @EndUserText.label: 'Course Fee'
33 35 virtual CourseFee : abap.int4
34 36
35 37 }
```

## Provide Class Name And Create

```
1@ CLASS zcl_ve_calculations DEFINITION PUBLIC .
2
3  PUBLIC SECTION.
4      INTERFACES : if_sadl_exit_calc_element_read.
5 ENDCLASS.
6
7@ CLASS zcl_ve_calculations IMPLEMENTATION.
8@   METHOD if_sadl_exit_calc_element_read~get_calculation_info.
9
10  ENDMETHOD.
11@  METHOD if_sadl_exit_calc_element_read~calculate.
12
13    DATA : lt_fee TYPE TABLE OF ZP_std_tab.
14    lt_fee = CORRESPONDING #( it_original_data ).
15@  LOOP AT lt_fee ASSIGNING FIELD-SYMBOL(<fs_fee>).
16@    IF <fs_fee>-Course = 'Computers'.
17      <fs_fee>-CourseFee = 150000.
18    ELSEIF <fs_fee>-Course = 'Electronics'.
19      <fs_fee>-CourseFee = 125000.
20    ELSE.
```

```

18      ELSEIF <fs_fee>-Course = 'Electronics'.
19          <fs_fee>-CourseFee = 125000.
20      ELSE.
21          <fs_fee>-CourseFee = 7500.
22      ENDIF.
23  ENDOLOOP.
24
25  ct_calculated_data = CORRESPONDING #( lt_fee ).
26
27
28  ENDMETHOD.
29 ENDCLASS.
```

## In Metadata Extension

```

36
37  @UI: { lineItem: [{ position: 50, label: 'Course' }],
38         identification: [{ position: 50,label: 'Course' }] }
39 Course;
40
41  @UI: { lineItem: [{ position: 55, label: 'Course Fee' }],
42         identification: [{ position: 55,label: 'Course Fee' }] }
43 CourseFee;
44 |
```

## Actions- Feature Control

01 December 2025 08:45

### RAP - action

- RAP actions are non-standard RAP BO operations that modify the state of an entity instance.
- Actions are part of the business logic. They are defined in the behavior definition and implemented in the behavior pool of a business object
- The following kinds of actions are available
  - **Non-factory actions** :- Custom logic that changes existing entity instances.
  - **Factory actions** :- It can be used to create RAP BO entity instances.
  - **Save actions** :- It can be non-factory actions or factory actions executed during the RAP save sequence

### RAP - action, Non-Factory

- Static action vs Instance (default) Action
- A Repeatable Action can be executed multiple times on the same RAP BO\_entity instance within the same ABAP EML or OData request.
- Internal – Access within the BO implementation
- Features ( Instance / Global ) - Enables or Disables action depending on preconditions within the BO / preconditions External
- Precheck prevents unwanted changes from reaching the application buffer
- Input parameters and output parameters can optionally be added to the action signature.

```
[Internal][static][repeatable] action
{
  [features: (instance | global)]
  [precheck]
  [authorisation:source]
  [authorisation:update]
  [authorisation:global]
  [authorisation:instance]
  [locknone]
}
ActionName [external 'ExternalName']
[InputParameter]
[OutputParameter]
```



### RAP - action, Factory

- Factory actions are used to create RAP BO entity instances.
- Instance-bound factory actions can copy specific values of an instance.
- Static factory actions can be used to create instances with prefilled default values.
- Diff then Non-Factory :-
  - Output parameters are not allowed. It always produces one new BO entity instance
  - It is mandatory to specify a cardinality. The cardinality must always be [1] for factory actions.
- The result of a factory action is returned in the response parameter MAPPED BY a mapping between the BDEF derived type %cid and the key of the newly created entity instance

```
[Internal][static][default] factory action
{
  [features: (instance | global)]
  [precheck]
  [authorisation:source]
  [authorisation:update]
  [authorisation:global]
  [authorisation:instance]
  [locknone]
}
ActionName [external 'ExternalName']
[InputParameter]
[cardinality]
```



### RAP - save action

- It can only be called during a specified RAP saver method in the RAP save sequence.
- The name of the RAP saver method during which the action can be executed is specified in brackets.
- Finalize, adjustnumbers, or both can be specified in brackets to indicate the saver method during which the action can be called.

```
[Internal][static][factory]
save(finalize|adjustnumbers|finalize, adjustnumbers) action
{
  [features: global]
  [precheck]
  [authorisation:source]
  [authorisation:global]
  [authorisation:instance]
}
ActionName [external 'ExternalName']
[InputParameter]
[OutputParameter]
```

## Two different views of the travel managed app

- Processor :-Creates individual travel instances, creates and manages individual flights, and adds supplements to flight bookings
- Approver :- Verification of the recorded travel data entered by the processor



## Accept and Reject Travel Action

- The action acceptTravel sets the overall status of the travel entity to accepted (A). The action rejectTravel sets the overall status of the travel entity to rejected (X)
- These actions are instance actions as there is one specific instance of the travel entity for which the status is changed. (Non-Factory)
- They return exactly one instance, namely the instance on which the action is executed. The result is therefore defined as result [1] \$self

A screenshot of the Fiori UI titled "Accept and Reject Travel Action on the Fiori UI". It shows a table of travel instances with columns: Travel ID, Overall Status, Agency ID, Customer ID, Booking Fee, and Total Price. Three rows are listed: 188005 (O, Happy Hopping), 188007 (O, Sunshine Travel), and 188009 (O, Hot Socks Travel). At the top right are buttons for "Reject Travel" and "Accept Travel". A video thumbnail of a person is visible on the right.

## Copy Travel Action

- The action copyTravel copies a travel instance as well as its associated booking and booking supplement instances and creates new instances based on the copied data.
- This action is an instance factory action as the end user has to select one instance which serves as the basis for the instance to be created

A screenshot of the Fiori UI titled "Copy Travel Action". It shows a table of travel instances with columns: Travel ID, Agency ID, Customer ID, Starting Date, and Overall Status. Three rows are listed: 1 (Mastrip), 2 (Hot Socks Travel), and 3 (Hendrick's). At the top right is a "Copy Travel" button. A video thumbnail of a person is visible on the right.

## Action ReCalcTotalPrice

- It is used to calculate the total price of one travel BO instance, that is one travel entity instance with all its booking instances and the corresponding booking supplement instances
- The relevant amounts of the booking fee, the flight price and the booking supplement price are added up in the field total\_price of the travel instance.
- The action is not exposed to the UI, as it is only used internally and called by determinations, whenever one of the relevant fields is changed.
- The action does not have a result parameter so the UI does not navigate to the travel instance whenever the determination is triggered on one of the child instances.

STEPS

## Steps to implement Copy Travel

- > We need to make sure CID is already filled we have to use to craete new entity
  - > Read all data from instance which we want to copy. ( Travel booking booking supplement )
  - > Loop travel table create new Instance copying %cid and other fdata
  - > Fill begin end date and status
  - > create new booking\_n instance filling cid\_REF as cid of traval
  - > loop through booking\_end copying other data except traval id and cid = CID&&bookingid (new cid for booking)
  - > Set booking status field for booking
  - > create new bookingsub\_n instance filling cid\_REF as booking\_cid of traval
  - > loop through booking\_sub copying other data except traval id and booking id and cid = CID&&bookingid&&bookingsubid
  - > Set booking status field for booking
  - > create new BO instance using Modify statement
    - > AgencyId CustomerId BeginDate EndDate BookingFee TotalPrice CurrencyCode OverallStatus Description
    - > BookingId BookingDate CustomerId CarrierId ConnectionId FlightDate FlightPrice CurrencyCode BookingStatus
    - > BookingSupplementId SupplementId Price CurrencyCode
  - > Update ui :- The UI-annotations for actions must be used as element annotation. However, it does not have an impact on which element the action is annotated. The action button always appears at the same position on the UI.
  - > { type: '#FOR\_ACTION', dataAction: "label: " }]



```

1 managed; //implementation in class zbp_i_travel1_m unique;
2 strict ( 2 );
3
4@define behavior for YI_travel1_M //alias <alias_name>
5 implementation in class zcl_bp_travel_managed unique
6 persistent table ytravel_m
7 lock master
8 authorization master ( instance )
9 etag master LastChangedAt
10 early numbering
11 {
12
13   field ( readonly ) TravelId, LastChangedBy, LastChangedAt, CreatedBy, CreatedAt;
14   field ( mandatory ) AgencyId, CustomerId, BeginDate, EndDate, OverallStatus, BookingFee, CurrencyCode;
15
16   action ( features : instance ) acceptTravel result [1] $self;
17   action ( features : instance ) rejectTravel result [1] $self;
18
19   factory action copytravel [1];
20
21   internal action recalcTotPrice;
22
23   create ( authorization : global );
24   update;
25   delete;
26
27   association _booking { create ( features : instance ); }
28@ mapping for ytravel_m

6
7@define behavior for YI_booking1_m //alias <alias_name>
8 implementation in class zcl_bp_Booking_managed unique
9 persistent table ybooking_m
0 lock dependent by _travel
1 authorization dependent by _travel
2 etag master LastChangedAt
3 //etag dependent by _travel
4 early numbering
5 {
6   update;
7   delete;
8   field ( readonly ) TravelId, BookingId, LastChangedAt;
9   field ( mandatory ) CarrierId, ConnectionId, FlightDate, BookingStatus;
0   field ( mandatory : create, readonly : update ) BookingDate, CustomerId;
1   association _travel;
2   association _bookingsuppl { create ( features : instance ); }
3

```

```

9@ define behavior for Yi_bookSuppl1_m //alias <alias_name>
0 implementation in class zcl_bp_BookSuppl_managed unique
1 persistent table ybooksuppl_m
2 lock dependent by _travel
3 authorization dependent by _travel
4 etag master LastChangedAt
5 //etag dependent by _travel
6 early numbering
7 {
8   update;
9   delete;
0   field ( readonly ) TravelId, BookingId, BookingSupplementId, LastChangedAt;
1   field ( mandatory ) Price, SupplementId;
2   association _travel;
3   association _booking;
4@ mapping for ybooksuppl_m
5   {
6     TravelId          = travel_id;
7   }
8 }

251@ METHOD get_instance_features.
252
253   read entities of yi_travel1_m in local mode
254   entity YI_travel1_M
255   fields ( TravelId OverallStatus )
256   with corresponding #( [keys] )
257   RESULT DATA(lt_travel).
258
259   result = VALUE #( FOR ls_travel IN lt_travel
260                     ( %tky = ls_travel-%tky
261                       %features-%action-acceptTravel = COND #( WHEN ls_travel-OverallStatus = 'A'
262                                         THEN if_abap_behv=>fc-o-disabled
263                                         ELSE if_abap_behv=>fc-o-enabled )
264                       %features-%action-rejectTravel = COND #( WHEN ls_travel-OverallStatus = 'X'
265                                         THEN if_abap_behv=>fc-o-disabled
266                                         ELSE if_abap_behv=>fc-o-enabled )
267                       %features-%assoc-_booking = COND #( WHEN ls_travel-OverallStatus = 'X'
268                                         THEN if_abap_behv=>fc-o-disabled
269                                         ELSE if_abap_behv=>fc-o-enabled )
270
271
272
273
274 ENDMETHOD.

```

## Approval(actions)

02 December 2025 12:59

### Creating the Projection CDS Views for the Approver

CDS views for BO structure YI\_TRAVEL\_TECH\_M YI\_BOOKING\_TECH\_M  
 CDS views for BO projection YC\_TRAVEL\_APPROVER\_M YC\_BOOKING\_APPROVER\_M

### Travel

The screenshot shows the 'Travel' application interface. At the top, there are search fields for 'Agency ID', 'Customer ID', and 'Overall Status'. Below the search bar is a table titled 'Travels (4,136)' with columns: Travel ID, Overall Status, Agency ID, Customer ID, Booking Fee, and Total Price. Two rows are visible: one for 'Maverick (70041)' and another for 'Hot Socks Travel (70007)'. Below the table is a navigation bar with tabs 'Travel' and 'Booking'. A detailed view of travel ID 2 is shown in a modal window, displaying customer information (Pierre), booking fees (20 USD), starting date (Jan 8, 2022), total price (900 USD), and a description (Business Trip for Christine, Pierre). There are also buttons for 'Accept Travel' and 'Reject Travel'.

### Booking

The screenshot shows the 'Booking' application interface. At the top, there are tabs for 'Travel' and 'Booking', with 'Booking' selected. Below the tabs is a table titled 'Bookings (2) Standard' with columns: Booking Number, Booking Date, Customer ID, Airline ID, Flight Number, Flight Date, Flight Price, and Status. Two rows are visible: one for 'Deltemple (99)' and another for 'Sisko (660)'. Below the table is a navigation bar with tabs 'Travel' and 'Booking'. A detailed view of booking ID 1 is shown in a modal window, displaying flight information (United Airlines, Inc. (UA)), flight number (1537), flight date (Jan 8, 2022), flight price (438.00 USD), and status (New). There is also a small thumbnail image of a person.

```

123 METHOD acceptTravel.
124
125 MODIFY ENTITIES OF yi_travel1_m IN LOCAL MODE
126 ENTITY yi_travel1_m
127 UPDATE FIELDS ( OverallStatus )
128 WITH VALUE #( FOR key IN keys ( %tky = key-%tky| OverallStatus = 'A' ) ).
129
130
131 READ ENTITIES OF yi_travel1_m IN LOCAL MODE
132 ENTITY yi_travel1_m
133 ALL FIELDS WITH CORRESPONDING #( keys )
134 RESULT DATA(lt_result).
135
136 result = VALUE #( FOR ls_res IN lt_result ( %tky = ls_res-%tky| %param = ls_res ) ).
137
138
139
140
141 ENDMETHOD.

```

```

233@ METHOD rejectTravel.
234   MODIFY ENTITIES OF yi_travel1_m IN LOCAL MODE
235   ENTITY yi_travel1_m
236   UPDATE FIELDS ( OverallStatus )
237   WITH VALUE #( FOR key IN keys ( %tky = key-%tky
238           OverallStatus = 'X' ) ).
239
240   READ ENTITIES OF yi_travel1_m IN LOCAL MODE
241   ENTITY yi_travel1_m
242   ALL FIELDS WITH CORRESPONDING #( keys )
243   RESULT DATA(lt_result).
244
245   result = VALUE #( FOR ls_res IN lt_result ( %tky = ls_res-%tky
246           %param = ls_res ) ).
247 ENDMETHOD.
248

```

## Feature Control

- We can enable and disable fields, standard and non standard operation as well
- It controls how data fields display on UI Ex :- Mandatory, Read only
- We can implement feature control two ways
  - Static way
  - Dynamic way
- We can have feature control at instance and static level

## Feature Control Operations

- Feature control cannot be used for internal operations.
- For static RAP BO operations, only global feature control is available. Instance feature control is not available.
- Feature control runtime checks are not evaluated for EML calls with the addition IN LOCAL MODE.
- The EML statement GET PERMISSIONS can be used to get information about disabled operations 

## Instance Feature Control Operations

- Instance feature control enables or disables operations depending on the state of the BO entity instance.
- If an operation is disabled by means of instance feature control, it is grayed out on the user interface of an OData application
- If an external RAP BO consumer tries to execute a disabled operation with EML, the operation fails and an entry is returned in the failed structure.
- Instance feature control can be defined for the following operations:
  - The standard operations update and delete
  - operations for associations
  - actions
  - the draft action Edit.

## Global Feature Control Operations

- Operations of a business object can be enabled or disabled globally.
- The decision is independent of the state of the individual entity instance. An operation can be globally enabled or disabled,
- Global feature control can be defined for the following operations:
  - the standard operations create, update, and delete
  - operations for associations
  - actions

## Fields Feature Control and Other characteristic

- The BO runtime rejects external EML MODIFY requests that include read-only fields for update or create operations.
- The mandatory property must be set in the behavior definition, not in the projection. This property isn't evaluated if you've defined it in a projection.
- Suppress can be used to remove a field from the BDEF derived types, OData, and all RAP APIs

```
... field,readonly) Field1, Field2, ...
| (mandatory) Field1, Field2, ...
| (suppress) Field1, Field2, ...
| (features:instance) Field1, Field2, ...
| (mandatory:create) Field1, Field2, ...
| (readonly:update) Field1, Field2, ...
| (nottrigger[:warn]) Field1, Field2, ...
```



## Fields Feature Control and Other characteristic

- Dynamic field attribute that defines access restrictions for fields depending on the state of the BO entity instance.
  - Must be implemented in the ABAP behavior pool in the RAP handler method FOR INSTANCE FEATURES.
- ```
field { features : instance } int_field2;
```
- mandatory:create** Dynamic field attribute that defines that it is mandatory to enter a value when an instance is created. No implementation required.
- A typical use case is to combine **mandatory:create** with **readonly:update** for key fields (external numbering by RAP BO consumer).

## Enhance our app with feature control

- Static feature for travel
  - Readonly** :- last\_changed\_at, last\_changed\_by, created\_at, created\_by
  - Mandatory** :- agency\_id, customer\_id, begin\_date, end\_date, overall\_status, booking\_fee, currency\_code;
  - Dynamic features** for :- acceptTravel rejectTravel & for create by association
- Static feature for booking
  - mandatory** :- carrier\_id, connection\_id, flight\_date, booking\_status
  - mandatory : create, readonly : update** :- booking\_date, customer\_id;
  - dynamic features** for :- for create by association
- Static feature for booking supp
  - readonly** :- last\_changed\_at;
  - mandatory** :- price, supplement\_id;



### At Interface View , use annotations for users data store

```
@Semantics.user.createdBy: true
created_by      as CreatedBy,
@Semantics.systemDateTime.createdAt: true
created_at      as CreatedAt,
@Semantics.user.localInstanceLastChangedBy: true|
last_changed_by as LastChangedBy,
@Semantics.systemDateTime.localInstanceLastChangedAt: true
last_changed_at as LastChangedAt,
```

### At Projection View , enable Those fields

```
27     CreatedBy,
28     CreatedAt,
29     LastChangedBy,
30     LastChangedAt,
31     /* Associations */
```

### Make fields as Read only , mandatory and Feature instance In BDEF

```

4 define behavior for YI_travel1_M //alias <alias_name>
5 implementation in class zcl_bp_travel_managed unique
6 persistent table ytravel_m
7 lock master
8 authorization master ( instance )
9 etag master LastChangedAt
10 early numbering
11 {
12
13 field ( readonly ) TravelId, LastChangedBy, LastChangedAt, CreatedBy, CreatedAt;
14 field ( mandatory ) AgencyId, CustomerId, BeginDate, EndDate, OverallStatus, BookingFee, CurrencyCode;
15
16 action ( features : instance ) acceptTravel result [1] $self;
17 action ( features : instance ) rejectTravel result [1] $self;
18
19 factory action copytravel [1];
20
21 internal action recalctotPrice;
22
23 create ( authorization : global );
24 update;
25 delete;
26
27 association _booking { create (features : instance); }
28 mapping for ytravel_m
29 {
30     TravelId      = travel_id;
31     AgencyId      = agency_id;

```

Mandat to enter in UI

```

field ( readonly ) TravelId, LastChangedBy, LastChangedAt, CreatedBy, CreatedAt;
field ( mandatory : create ) AgencyId, CustomerId, BeginDate, EndDate, OverallStatus, BookingFee, CurrencyCode;

```

```

46 define behavior for YI_booking1_M //alias <alias_name>
47 implementation in class zcl_bp_Booking_managed unique
48 persistent table ybooking_m
49 lock dependent by _travel
50 authorization dependent by _travel
51 etag master LastChangedAt
52 //etag dependent by _travel
53 early numbering
54 {
55     update;
56     delete;
57     field ( readonly ) TravelId, BookingId;
58     field ( mandatory ) CarrierId, ConnectionId, FlightDate, BookingStatus;
59     field ( mandatory : create, readonly : update ) BookingDate, CustomerId;
60     association _travel;
61     association _bookingsuppl { create ( features : instance ); }
62 mapping for ybooking_m
63
64 define behavior for Yi_bookSuppl1_M //alias <alias_name>
65 implementation in class zcl_bp_BookSuppl_managed unique
66 persistent table ybooksuppl_m
67 lock dependent by _travel
68 authorization dependent by _travel
69 etag master LastChangedAt
70 //etag dependent by _travel
71 early numbering
72 {
73     update;
74     delete;
75     field ( readonly ) TravelId, BookingId, BookingSupplementId, LastChangedAt;
76     field ( mandatory ) Price, SupplementId;
77     association _travel;
78     association _booking;
79 mapping for ybooksuppl_m
80

```

## Travel and booking form travel class

```

250
251 METHOD get_instance_features.
252
253 read entities of YI_travel1_M in local mode
254 entity YI_travel1_M
255 fields ( TravelId OverallStatus )
256 with corresponding #( keys )
257 RESULT DATA(lt_travel).
258
259 result = VALUE #( FOR ls_travel IN lt_travel
260 ( %tky = ls_travel-%tky
261 %features-%action-acceptTravel = COND #( WHEN ls_travel-OverallStatus = 'A'
262 THEN if_abap_behv=>fc-o-disabled
263 ELSE if_abap_behv=>fc-o-enabled )
264 %features-%action-rejectTravel = COND #( WHEN ls_travel-OverallStatus = 'X'
265 THEN if_abap_behv=>fc-o-disabled
266 ELSE if_abap_behv=>fc-o-enabled )
267 %features-%assoc_booking = COND #( WHEN ls_travel-OverallStatus = 'X'
268 THEN if_abap_behv=>fc-o-disabled
269 ELSE if_abap_behv=>fc-o-enabled )
270
271 ) ).
```

ENDMETHOD.

### From Booking class

```

66 METHOD get_instance_features.
67
68 READ ENTITIES OF YI_travel1_M IN LOCAL MODE
69 ENTITY YI_travel1_M BY \_booking|
70 FIELDS ( TravelId BookingId BookingStatus )
71 WITH CORRESPONDING #( keys )
72 RESULT DATA(lt_book).
73
74 result = VALUE #( FOR ls_book IN lt_book ( %tky = ls_book-%tky
75 %features-%assoc_bookingsuppl = COND #( WHEN ls_book-BookingStatus = 'X'
76 THEN if_abap_behv=>fc-o-disabled
77 ELSE if_abap_behv=>fc-o-enabled ) ) ).
```

ENDMETHOD.

## Fill data

03 December 2025 10:05

### At Interface View , use annotations for users data store

```
@Semantics.user.createdBy: true
created_by      as CreatedBy,
@Semantics.systemDateTime.createdAt: true
created_at      as CreatedAt,
@Semantics.user.localInstanceLastChangedBy: true
last_changed_by as LastChangedBy,
@Semantics.systemDateTime.localInstanceLastChangedAt: true
last_changed_at as LastChangedAt,
```

### At Projection View , enable Those fields

```
27     CreatedBy,
28     CreatedAt,
29     LastChangedBy,
30     LastChangedAt,
31     /* Associations */
```

### Make fields as Read only , mandatory and Feature instance In BDEF

```
40 define behavior for YI_travel1_M //alias <alias_name>
5 implementation in class zcl_bp_travel_managed unique
6 persistent table ytravel_m
7 lock master
8 authorization master ( instance )
9 etag master LastChangedAt
10 early numbering
11 {
12
13   field ( readonly ) TravelId, LastChangedBy, LastChangedAt, CreatedBy, CreatedAt;
14   field ( mandatory ) AgencyId, CustomerId, BeginDate, EndDate, OverallStatus, BookingFee, CurrencyCode;
15
16   action ( features : instance ) acceptTravel result [1] $self;
17   action ( features : instance ) rejectTravel result [1] $self;
18
19   factory action copytravel [1];
20
21   internal action recalcTotPrice;
22
23   create ( authorization : global );
24   update;
25   delete;
26
27   association _booking { create (features : instance ); }
28 mapping for ytravel_m
29   {
30     TravelId      = travel_id;
31     AgencyId     = agency_id;
```

## Validations

03 December 2025 12:52

### RAP - validation

- A validation is an optional part of the business object behavior that checks the consistency of business object instances based on trigger conditions.
- It is automatically invoked by the RAP framework if the trigger condition of the validation is fulfilled
- Trigger conditions can be :- CUD operation or modified fields

```
validation ValName on save { CUD1; CUD2; ... }  
| { field Field1, Field2, ... ; }
```

- Validations are always invoked during the save sequence at the end of a transaction and this is indicated by the mandatory addition on save.

- Validations must be implemented in the RAP handler method FOR VALIDATE in the local ABAP behavior pool (ABP).



### RAP - validation

- The execution order of validations is not fixed. If there is more than one validation triggered by the same condition, the execution order is not predictable.
- It is not allowed to use EML MODIFY statements in the implementation of validations.
- An error message can be written to the reported structure of the MODIFY ENTITIES statement.
- An entry is returned in the failed and reported structures of the COMMIT ENTITIES statement.



### Defining Validations

```
Travel Validation :-  
validation validateCustomer on save { create; field customer_id; }  
validateAgency  
validateDates  
validateStatus  
validateCurrencyCode  
validateBookingFee  
  
Booking Validation :-  
validateStatus  
validateCustomer  
validateCurrencyCode  
validateConnection  
validateFlightPrice  
  
Booking Supp Validation :-  
validateCurrencyCode  
validateSupplement  
validatePrice
```



```
1 managed; //implementation in class zbp_i_travel1_m unique;  
2 strict ( 2 );  
3  
4 define behavior for YI_travel1_M //alias <alias_name>  
5 implementation in class zcl_bp_travel_managed unique  
6 persistent table ytravel_m  
7 lock master  
8 authorization master ( instance )  
9 etag master LastChangedAt  
10 early numbering  
11 {  
12  
13   field ( readonly ) TravelId, LastChangedBy, LastChangedAt, CreatedBy, CreatedAt;  
14   field ( mandatory ) AgencyId, CustomerId, BeginDate, EndDate, OverallStatus, BookingFee, CurrencyCode;  
15  
16   action ( features : instance ) acceptTravel result [1] $self;  
17   action ( features : instance ) rejectTravel result [1] $self;  
18  
19   factory action copytravel [1];  
20   internal action recalcTotPrice;  
21  
22   validation validateCustomer on save { create; field CustomerId; }  
23   validation validateDates on save { create; field BeginDate, EndDate; }  
24   validation validateStatus on save { create; field OverallStatus; }  
25   validation validateCurrencyCode on save { create; field CurrencyCode; }  
26   validation validateBookingFee on save { create; field BookingFee; }  
27
```

```

283 METHOD validateCustomer.
284   READ ENTITIES OF yi_travel1_m IN LOCAL MODE
285   ENTITY YI_travel1_M
286   FIELDS ( CustomerId ) WITH CORRESPONDING #( keys )
287   RESULT DATA(lt_travel).
288
289   READ TABLE lt_travel INTO DATA(ls_cust) INDEX 1.
290   SELECT
291     FROM /dmo/customer
292     FIELDS ( customer_id )
293     WHERE customer_id = @ls_cust-CustomerId
294     INTO TABLE @DATA(lt_cust_db).
295
296 IF sy-subrc NE 0.
297   LOOP AT lt_travel ASSIGNING FIELD-SYMBOL(<fs_travel>).
298     IF <fs_travel>-CustomerId IS INITIAL OR NOT line_exists( lt_cust_db[ customer_id = <fs_travel>-CustomerId ] ).
299       APPEND VALUE #( %tky = <fs_travel>-%tky ) TO failed-yi_travel1_m.
300     APPEND VALUE #( %tky = <fs_travel>-%tky|
301       %msg = NEW /dmo/cm_flight_messages(
302         textid           = /dmo/cm_flight_messages=>customer_unkown
303         customer_id      = <fs_travel>-CustomerId
304         severity        = if_abap_behv_message=>severity-error
305
306       )|
307     ) TO reported-yi_travel1_m.
308   ENDIF.
309 ENDLOOP.
310 ENDDIF.
311
312 ENDMETHOD.
313
314

```

OR

```

283 METHOD validateCustomer.
284   READ ENTITIES OF yi_travel1_m IN LOCAL MODE
285   ENTITY YI_travel1_M
286   FIELDS ( CustomerId ) WITH CORRESPONDING #( keys )
287   RESULT DATA(lt_travel).
288
289   READ TABLE lt_travel INTO DATA(ls_cust) INDEX 1.|*
290   SELECT
291     FROM /dmo/customer
292     FIELDS ( customer_id )
293     WHERE customer_id = @ls_cust-CustomerId
294     INTO TABLE @DATA(lt_cust_db).
295
296 IF sy-subrc NE 0.
297   LOOP AT lt_travel ASSIGNING FIELD-SYMBOL(<fs_travel>).
298     IF <fs_travel>-CustomerId IS INITIAL OR NOT line_exists( lt_cust_db[ customer_id = <fs_travel>-CustomerId ] ).
299       APPEND VALUE #( %tky = <fs_travel>-%tky ) TO failed-yi_travel1_m.
300     APPEND VALUE #( %tky = <fs_travel>-%tky|
301       %msg = new_message_with_text(
302         severity = if_abap_behv_message=>severity-error
303         text     = 'Customer ID is Unknown'
304       )|
305     ) TO reported-yi_travel1_m.
306   ENDIF.
307 ENDLOOP.
308 ENDDIF.
309
310 ENDMETHOD.
311

```

```

330 METHOD validateDates.
331   READ ENTITIES OF yi_travel1_m IN LOCAL MODE
332   ENTITY yi_travel1_m
333   FIELDS ( BeginDate EndDate ) WITH CORRESPONDING #( keys )
334   RESULT DATA(lt_travel).
335
336 LOOP AT lt_travel INTO DATA(ls_travel).
337   IF ls_travel-BeginDate GT ls_travel-EndDate.
338     APPEND VALUE #( %key = ls_travel-%key ) TO failed-yi_travel1_m.
339     APPEND VALUE #( %tky = ls_travel-%tky|
340       %msg = NEW /dmo/cm_flight_messages(
341         textid           = /dmo/cm_flight_messages=>begin_date_bef_end_date
342         travel_id        = ls_travel-TravellId
343         begin_date       = ls_travel-BeginDate
344         end_date         = ls_travel-EndDate
345         severity        = if_abap_behv_message=>severity-error
346       )|
347       %element-begindate = if_abap_behv=>mk-on
348       %element-enddate = if_abap_behv=>mk-on
349     ) TO reported-yi_travel1_m.
350
351 ELSEIF ls_travel-BeginDate < cl_abap_context_info=>get_system_date( ).*
352
353   APPEND VALUE #( %key = ls_travel-%key ) TO failed-yi_travel1_m.
354   APPEND VALUE #( %tky = ls_travel-%tky|
355     %msg = NEW /dmo/cm_flight_messages(
356       textid           = /dmo/cm_flight_messages=>begin_date_on_or_bef_sysdate
357       begin_date       = ls_travel-BeginDate
358       severity        = if_abap_behv_message=>severity-error
359     )|
360     %element-begindate = if_abap_behv=>mk-on
361     %element-enddate = if_abap_behv=>mk-on
362   ) TO reported-yi_travel1_m.
363
364
365 ENDIF.
366 ENDLOOP.
367
368 ENDMETHOD.

```

```

370@ METHOD validateStatus.
371
372     READ ENTITIES OF yi_travell_m IN LOCAL MODE
373         ENTITY yi_travell_m
374             FIELDS ( OverallStatus ) WITH CORRESPONDING #( keys )
375             RESULT DATA(lt_travel).
376
377@ LOOP AT lt_travel INTO DATA(ls_travel).
378@     CASE ls_travel-OverallStatus.
379         WHEN 'O'.
380         WHEN 'A'.
381         WHEN 'X'.
382         WHEN OTHERS.
383
384         APPEND VALUE #( %key = ls_travel-%key ) TO failed-yi_travell_m.
385         APPEND VALUE #( %tky = ls_travel-%tky
386                         %msg = NEW /dmo/cm_flight_messages(
387   textid
388   status
389   severity
390   %element-OverallStatus = if_abap_behv=>mk-on
391
392             ) TO reported-yi_travell_m.
393     ENDCASE.
394 ENDLOOP.
395
396 ENDMETHOD.
397

```

# Determinations

04 December 2025 11:41

## Determinations

- A determination is an optional part of the BO behavior that modifies instances of BO based on trigger conditions
- A determination is implicitly invoked by the BO's framework if the trigger condition of the determination is fulfilled.
- Trigger conditions can be modify operations CUD and modified fields
- An invoked determination can compute data, modify entity instances according to the computation result and return messages to the consumer by passing them to the corresponding table in the REPORTED structure.



## RAP - determination

- Trigger conditions can be modify operations CUD or modified fields

```
determination<#gt;DetName {on modify { CUD1; CUD2; ... }  
| { field Field1, Field2, ... ; }}  
|{on save { CUD1; CUD2; ... }  
| { field Field1, Field2, ... ; }}
```

- Two types of determinations are distinguished, depending on the stage of the program flow the determination is executed:

- on modify :- It is executed immediately after data changes take place in the transactional buffer so that the result is available during the transaction.
- on save :- It is executed during the save sequence at the end of an transaction, when changes from the transactional buffer are persistent on the DB.

- For determinations defined as on save, update works only in combination with the trigger operation create



## Determinations Points

- The determination result must not change if the determination is executed several times under the same conditions (**idempotence**).
- The execution order of determinations is not fixed. If there is more than one determination triggered by the same condition, you cannot know which determination is executed first.
- Once a determination has been triggered, it must run independently from other determinations.



# Determination for Our App

- The determination calculateTotalPrice, which is defined for all three entities handles the calculation of total price of one travel.
- The determination will be triggered by on modify as determination time when creating new instances or updating the prices that are included in the calculation or when changing the currency.
- The starting point of viewing with a newly created travel instance with the initial amount (Total Price) and the travel currency 0.00 USD.
- If a user adds a flight booking to the travel, then also the travel's Total Price is updated.



# Annotations

05 December 2025 12:46

## Basic Annotations

@Metadata.layer: #CORE

### Header

```
@UI: {  
  headerInfo: {  
    typeName: 'Partner',  
    typeNamePlural: 'Partners'  
  }  
}  
  
annotate entity yBS_C_RAPPartner with  
{
```

### Facet/Detail page

```
@UI.facet      : [  
{  
  id          : 'FacetCollection',  
  type        : #COLLECTION,  
  label       : 'Partner Collection'  
},  
{  
  id          : 'FacetPartnerFields',  
  label       : 'Information',  
  position: 10,  
  type        : #IDENTIFICATION_REFERENCE,  
  targetQualifier: 'PARTNER_INFO',  
  parentId: 'FacetCollection'  
},  
{  
  id          : 'FacetPartnerAddress',  
  label       : 'Address',  
  position: 20,  
  type        : #IDENTIFICATION_REFERENCE,  
  targetQualifier: 'PARTNER_ADDRESS',  
  parentId: 'FacetCollection'  
}  
]
```

### Selection

```
@UI.selectionField: [{ position: 10 }]
```

### List

```
@UI.lineItem: [{ position: 20, importance: #MEDIUM },  
  
{ type: #FOR_ACTION, dataAction: 'fillEmptyStreets', label: 'Fill' },  
{ type: #FOR_ACTION, dataAction: 'clearAllEmptyStreets', label: 'Clear All' }]  
// { type: #FOR_ACTION, dataAction: 'copyLine', label: 'Copy line' }]
```

```

@UI.identification: [{ position: 10, qualifier: 'PARTNER_INFO' }]
@EndUserText.label: 'Partner'
@EndUserText.quickInfo: 'Identifier of the partner'
@Search.defaultSearchElement: true
PartnerNumber;
@UI.lineItem: [{ position: 10, importance: #MEDIUM }]
@UI.selectionField: [{ position: 20 }]
@EndUserText.label: 'Name'
@EndUserText.quickInfo: 'Partner Name'
@UI.identification: [{ position: 20, qualifier: 'PARTNER_INFO' }]
PartnerName;

```

## Texts

```

@EndUserText: {
    label: 'City',
    quickInfo: 'City Of the Partner'
}

```

## Actions

```

BDEF(I_VIEW)
action fillEmptyStreets result [1] $self;
static action clearAllEmptyStreets ;
factory action copyLine [1];
BDEF(P_VIEW)

```

```

use action fillEmptyStreets;
use action clearAllEmptyStreets;
use action copyLine;

```

MDE

```

@UI.lineItem: [{ position: 20, importance: #MEDIUM },
    { type: #FOR_ACTION, dataAction: 'fillEmptyStreets', label:
'Fill' },
    { type: #FOR_ACTION, dataAction: 'clearAllEmptyStreets', label:
'Clear All' }]
// { type: #FOR_ACTION, dataAction: 'copyLine', label:
'Copy line' }

```

## Search help

```

@Consumption.valueHelpDefinition: [{ entity: { name: 'YBS_C_CountryVH',
element: 'Currency' } }]

```

PaymentCurrency

## Search field

```

@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Search help'
@Metadata.ignorePropagatedAnnotations: true
@Search.searchable: true
@ObjectModel.resultSet.sizeCategory: #XS
define view entity yBS_C_CountryVH
as select from I_Country
{
  @Search.defaultSearchElement: true
  @Search.fuzzinessThreshold: 0.8
  @Search.ranking: #HIGH
  @ObjectModel.text.element: [ 'Description' ]
  key Country,
  @Search.defaultSearchElement: true
  @Search.fuzzinessThreshold: 0.8
  @Search.ranking: #LOW
  _Text[ 1: Language = $session.system_language ].CountryName as
  Description
}

```

## Dropdown

With the annotation "**ObjectModel.resultSet.sizeCategory: #XS**" Fiori Elements is told that the result set is very small and the field directly becomes a dropdown element:

```
@ObjectModel.resultSet.sizeCategory: #XS
```

## Texts

```
@ObjectModel.text.element: [ 'Description' ]
```

```

@ObjectModel.text.element: [ 'AgencyName' ]
AgencyId,
_agency.Name      as AgencyName,
@ObjectModel.text.element: [ 'CustomerName' ]
CustomerId,
_customer.LastName as CustomerName,

```

## Validation

```
BDEF(I_VIEW)
```

```

validation validateKeyIsFilled on save { create; }
validation validateCoreData on save { create; field Country,
PaymentCurrency; }

```

## Determination

```
BDEF(I_VIEW)
```

```
determination fillCurrency on modify { create; update; }
```



# Side Effects

09 December 2025 11:23

## Two Ways

### 1) way

#### What is Side Effect

When user makes a change to field on UI, and this change effect the content of other field, this behavior is called Side Effect.

#### How to Implement Side Effects

There are Annotations available which helps implementing Side Effects behavior on UI

| Source properties | Properties on which you want to trigger Side Effect behavior  |
|-------------------|---------------------------------------------------------------|
| Target properties | Properties you want to manipulate as a result of Side Effects |

### 2) way

## IN Behaviour Definition OF Interface View

\*\* use CREATE in it . Will work for create and update

```
-- 33     determination Course_Duration on modify { create; field Course; }
-- 34     side effects { field Course affects field Courseduration; }
-- 35
-- 36     field ( readonly ) Id, GenderDesc, StuentId, Courseduration;
```

## IN Behaviour Definition OF Projection View

```
1 projection;
2 strict ( 2 );
3 use draft;
4 use side effects;
5 @define behavior for ZC_student_um alias Student
6 use etag
7 {
8   use create;
9   use update;
10  use delete;
```

## IN LHC, method Implementation

```

309④ METHOD Course_Duration.
310
311④ IF keys IS NOT INITIAL.
312
313     READ ENTITIES OF ZI_student_um IN LOCAL MODE
314     ENTITY Student
315     FIELDS ( Course )
316     WITH CORRESPONDING #( keys )
317     RESULT DATA(lt_student).
318
319     DATA(cc) = lt_student[ 1 ]-%data-Course.
320     DATA(c_duration) = COND #( WHEN cc = 'BTECH' OR cc = 'btech' THEN 4
321                               WHEN cc = 'mca' OR cc = 'MCA' THEN 2
322                               WHEN cc = 'BSC' OR cc = 'bsc' OR cc = 'bca' OR cc = 'BCA'
323                               THEN 3
324                               ).
325     MODIFY ENTITIES OF ZI_student_um IN LOCAL MODE
326     ENTITY Student
327     UPDATE FIELDS ( Courseduration )
328     WITH VALUE #( ( %tky = keys[ 1 ]-%data-Courseduration = c_duration ) )
329     MAPPED DATA(lt_mapped)
330     FAILED DATA(lt_failed)
331     REPORTED DATA(lt_reported).
332
333 ENDIF.
334
335 ENDMETHOD.
336

```

In draft prepare Action , Only On Save Determination will allow  
Else will not work

```

25    draft determine action Prepare
26    {
27        validation ( always ) validate_fileds;
28        determination ( always ) Course_Duration;
29    }
30
31    field ( mandatory ) Firstname, Lastname, Age;
32    validation validate_fileds on save { create; update; }
33
34    determination Course_Duration on save { create; field Course; }
35    side effects { field Course affects field Courseduration; }
36
37    field ( readonly ) Id, GenderDesc, StuentId, Courseduration;
38

```

# Precheck

09 December 2025 12:51

## What is Prechecks

**Prechecks** is used to validate data before it can reach to Transactional Buffer.  
You can prevent illegal changes from reaching the transactional buffer by prechecking modify operations.

During **PRECHECK** implementation, we can implement validation logic and based on result generate FAILED and REPORTED parameters which can be used to display Error on Fiori frontend.

## Advantage

If application is a draft enabled application, then these inconsistent values have already been stored in draft tables and transactional buffers.

## How to Implement Prechecks

**Prechecks** can be defined in the behavior definition and implemented it in the behavior implementation class.

```
create ( precheck );
update ( prechecks ); *precheck
```

```
115@ METHOD precheck_update.
116
Q117@   LOOP AT entities ASSIGNING FIELD-SYMBOL(<fs_entity>).
118     CHECK <fs_entity>-%control-Course EQ '01' OR <fs_entity>-Courseduration = '01'.
119     READ ENTITIES OF Yrap_Student IN LOCAL MODE
120     ENTITY Student
121     FIELDS ( Course Courseduration )
122     WITH VALUE #( ( %key = <fs_entity>-%key ) )
123     RESULT DATA(lt_studentsCourse).|
124@   IF sy-subrc IS INITIAL.
125     READ TABLE lt_studentsCourse ASSIGNING FIELD-SYMBOL(<lfs_db_course>) INDEX 1.
126@   IF sy-subrc IS INITIAL.
127     <lfs_db_course>-Course = COND #( WHEN <fs_entity>-Course EQ '01'
128                                     THEN <fs_entity>-Course
129                                     ELSE <lfs_db_course>-Course ).
130
131     <lfs_db_course>-Courseduration = COND #( WHEN <fs_entity>-Courseduration EQ '01'
132                                     THEN <fs_entity>-Courseduration
133                                     ELSE <lfs_db_course>-Courseduration ).|
134@   IF <lfs_db_course>-Courseduration < 2.
135     IF <lfs_db_course>-Course = 'Computers'.
136       APPEND VALUE #( %key = <lfs_db_course>-%key ) TO failed-student.
137       APPEND VALUE #( %tky = <lfs_db_course>-%tky
138                     %msg = new_message_with_text(
139                       severity = if_abap_behv_message=>severity-error
140                       text      = 'Invalid course Duration....'
141                     ) ) TO reported-student.
142     ENDIF.
143   ENDIF.
144   ENDDIF.
145
146   ENDOBJ.
147 ENDMETHOD.
```

# Authorization

17 December 2025 09:51

## What is Authorization in RAP

Authorization control in RAP protects your business object against unauthorized access. Authorization control is always relevant when the permission to execute an operation depends on the role.

In RAP each read or modify request can be checked via authorization objects against user roles before the request is finally executed.

**Global Authorization** - Global authorization is used for all authorization checks. You can define global authorization to check if users are allowed to execute an operation in general (**CREATE**, **UPDATE**, **DELETE**), **authorization master (global)**

**Instance Authorization** - Instance authorization is used for all authorization checks, in addition to the user role. With instance authorization, you can define authorization on a field or operation (**UPDATE**, **DELETE**). Instance authorization is only possible for instance-based operations.

**authorization instance ()**

Authorization Check against Incoming Values (Precheck) – We have already seen Prechecks in one of our previous video.

## Global Authorizations

```
143⊕ METHOD is_update_allowed.  
144     update_allowed = abap_true.  
145 ENDMETHOD.  
146  
147⊕ METHOD get_global_authorizations.  
148  
149⊕ IF requested_authorizations->update = if_abap_behv=>mk-on .  
150⊕   IF is_update_allowed( ) = abap_true.  
151     result->update = if_abap_behv=>auth-allowed.  
152   ELSE.  
153     result->update = if_abap_behv=>auth-unauthorized.  
154     reported-student = VALUE #( ( %msg = new_message_with_text(  
155                               severity = if_abap_behv_message=>severity-error  
156                               text      = 'You are not Authorized To Edit'  
157                           ) ) ).  
158   ENDIF.  
159 ENDIF.  
160  
161 ENDMETHOD.  
162  
161⊕ METHOD get_instance_authorizations.  
162  
163   DATA : updated_requested TYPE abap_boolean,  
164       updated_gratated  TYPE abap_boolean.  
165  
166   READ ENTITIES OF Yrap_Student IN LOCAL MODE  
167   ENTITY Student  
168   FIELDS ( Status )  
169   WITH CORRESPONDING #( keys )  
170   RESULT DATA(lt_data).  
171  
172   CHECK lt_data IS NOT INITIAL.  
173   updated_requested = COND #( WHEN requested_authorizations->update = if_abap_behv=>mk-on THEN abap_true ELSE abap_false ).  
174  
175⊕ LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<lfs_admitted>).  
176⊕   IF <lfs_admitted>-Status = abap_false.  
177⊕     IF updated_requested = abap_true.  
178       updated_gratated = is_update_allowed( ).  
179  
180       IF updated_gratated = abap_false.  
181         failed-student = VALUE #( ( %tky = <lfs_admitted>-%tky ) ).  
182         reported-student = VALUE #( ( %tky = <lfs_admitted>-%tky %msg = new_message_with_text(  
183                                       severity = if_abap_behv_message=>severity-error  
184                                       text      = 'You are Not Authorized to Update'  
185                                   ) ) ).  
186     ENDIF.  
187   ENDIF.  
188  
189   ENDIF.  
190 ENDLOOP.  
191  
191 FUNCTION  
  
METHOD is_update_allowed.  
  update_allowed = abap_true.  
ENDMETHOD.  
  
METHOD get_global_authorizations.
```

```

IF requested_authorizations-%update = if_abap_behv=>mk-on .
  IF is_update_allowed( ) = abap_true.
    result-%update = if_abap_behv=>auth-allowed.

  ELSE.
    result-%update = if_abap_behv=>auth-unauthorized.
    result-%delete = if_abap_behv=>auth-unauthorized.
    reported-student = VALUE #( ( %msg = new_message_with_text(
      severity =
if_abap_behv_message=>severity-error
                                text      = 'You are not
Authorized To Edit'
) ) ).

  ENDIF.
ENDIF.

```

ENDMETHOD.

```

METHOD_get_instance_authorizations.

DATA : updated_requested TYPE abap_boolean,
       updated_gratated  TYPE abap_boolean.

READ ENTITIES OF Yrap_Student IN LOCAL MODE
ENTITY Student
FIELDS ( Status )
WITH CORRESPONDING #( keys )
RESULT DATA(lt_data).

CHECK lt_data IS NOT INITIAL.
updated_requested = COND #( WHEN requested_authorizations-%update =
if_abap_behv=>mk-on THEN abap_true ELSE abap_false ).

LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<lfs_admitted>).
  IF <lfs_admitted>-Status = abap_false.
    IF updated_requested = abap_true.
      updated_gratated = is_update_allowed( ).

      IF updated_gratated = abap_false.
        failed-student = VALUE #( ( %tky = <lfs_admitted>-%tky ) ).
        reported-student = VALUE #( ( %tky = <lfs_admitted>-%tky %msg =
new_message_with_text(
severity = if_abap_behv_message=>severity-error
                                text
= 'You are Not Authorized to Update'
) ) ).

    ENDIF.
  ENDIF.
  ENDLOOP.

ENDMETHOD.

```

# Change Set

17 December 2025 12:23

Same Action trigger multiple times on multiple Instances and all are grouped...

#CHANGE\_SET can be used with

```
@UI.lineItem.invocationGrouping  
@UI.statusInfo.invocationGrouping  
@UI.chart.actions.invocationGrouping  
@UI.fieldGroup.invocationGrouping  
@UI.identification.invocationGrouping
```

```
1 @UI: {  
2     lineItem: [{ position: 10,  
3                 label: 'Update Status',  
4                 type: #FOR_ACTION,  
5                 dataAction: 'statusUpdate',  
6                 invocationGrouping: '#CHANGE_SET' }  
7             ],  
8             identification: [{ position: 10, label: 'Student ID' }]  
9         }  
10 Id;
```

#ISOLATED can be used with @UI.lineItem.invocationGrouping

| #CHANGE_SET                                          | #ISOLATED                                                               |
|------------------------------------------------------|-------------------------------------------------------------------------|
| 1- Multiple records can be processed at once         | 1- Multiple records can be processed at once                            |
| 2- All records must be valid                         | 2- Only valid records are processed leaving Invalid records unprocessed |
| 3- Single Session created for processing the records | 3- Multiple Session are created, one per record to be processed         |
| 4- All records are processed in one go               | 4- Individual record processed one by one                               |

```
1 @UI: {  
2     lineItem: [{ position: 10,  
3                 label: 'Update Status',  
4                 type: #FOR_ACTION,  
5                 dataAction: 'statusUpdate',  
6                 invocationGrouping: '#ISOLATED' }  
7             ],  
8             identification: [{ position: 10, label: 'Student ID' }]  
9         }  
10 Id;
```

# File Upload(Streams)

17 December 2025 12:40

Streams in RAP : SAP BTP ABAP 2208 release the RAP framework now supports OData streams and which enables RAP application to handle LOB or Large Objects like Files.

@Semantics.largeObject



Table

```
1 @EndUserText.label : 'Student Attachments of Resumes'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 @AbapCatalog.tableCategory : #TRANSPARENT
4 @AbapCatalog.deliveryClass : #A
5 @AbapCatalog.dataMaintenance : #RESTRICTED
6 define table ystudent_att_tab {
7
8   key client      : abap.clnt not null;
9   key attach_id   : abap.char(32) not null;
10  id             : sysuuid_x16 not null;
11  comments       : abap.char(30);
12  attachement    : abap.rawstring(256);| highlighted
13  mimetype       : abap.char(120);
14  filename        : abap.char(120);
15
16 }
```

IN Projection

```
33@  @UI: {
34    lineItem: [{ position: 40 }],
35    identification: [{ position: 40 }]
36  }
37  @EndUserText.label: 'Attachement'
38
39  @Semantics.largeObject: {
40    mimeType: 'Mimetype',
41    fileName: 'Filename',
42    acceptableMimeTypes: [ '' ],
43    contentDispositionPreference: #INLINE,
44    cacheControl: {
45      maxAge: #SHORT
46    }
47  }
48
49  Attachement,
50  @EndUserText.label: 'File Type'
51  Mimetype,
```

```

BDEF
43@define behavior for Ystudent_i_att alias Attachements
44 persistent table ystudent_att_tab
45 draft table ystudent_att_d
46 lock dependent by _Student
47 authorization dependent by _Student
48 etag master LastChangedAt
49 {
50   update;
51   delete;
52   field ( readonly ) id;
53   association _Student { with draft; }
54
55@ mapping for ystudent_att_tab
56 {
57   AttachId      = attach_id;
58   Attachement   = attachement;
59   Comments      = comments;
60   Filename      = filename;
61   Id            = id;
62   Mimetype      = mimetype;
63 }
64
65
66 }

```

## @Semantics.largeObject

1. Download Uploaded File
2. Apply File type restriction and show Error Message

TO Download The File , Use Below Property in Projection

```

32@  @UI: {
33    lineItem: [{ position: 40 }],
34    identification: [{ position: 40 }]
35  }
36  @EndUserText.label: 'Attachement'
37
38  @Semantics.largeObject: {
39    mimeType: 'Mimetype',
40    fileName: 'Filename',
41    acceptableMimeTypes: [ '' ],
42    contentDispositionPreference: #ATTACHMENT,
43    cacheControl: {
44      maxAge: #SHORT
45    }
46  }
47
48  Attachement,
49  @EndUserText.label: 'File Type'

```

Supported Mine Types

# Supported MIME Types

## Use

You use this function as a reference for the valid MIME types in the MIME Repository.

## Features



A list of MIME objects that can be stored in the MIME Repository:

|       | Category/Subcategory | File Extension               | Description         |
|-------|----------------------|------------------------------|---------------------|
| Texts | text/css             | css                          | CSS stylesheet file |
|       | text/html            | html, htm                    | HTML file           |
|       | text/javascript      | js                           | JavaScript file     |
|       | text/plain           | txt, c, cc, g, h, hh, m, f90 | Plain text file     |
|       | text/richtext        | rtx                          | MIME rich text      |
|       | text/xml             | xml                          | XML file            |



|        |                             |                 |                        |
|--------|-----------------------------|-----------------|------------------------|
| Images | image/gif                   | gif             | GIF graphic            |
|        | image/ief                   | ief             | Image Exchange Format  |
|        | image/jpeg                  | jpeg, jpg , jpe | JPEG graphic           |
|        | image/x-rgb                 | rgb             | RBG graphic            |
|        | image/x-windowdump          | xwd             | X-Windows dump         |
|        | image/tiff                  | tiff, tif       | TIFF graphic           |
|        | image/bmp                   | bmp             | Icon                   |
|        | image/ico                   | ico             | Icon (Windows)         |
| Audio  | audio/basic                 | au, snd         | AU and SND sound files |
|        | audio/x-aiff                | aif, aiff, aifc | AIFF sound file        |
|        | audio/x-midi                | midi, mid       | MIDI file              |
|        | audio/x-pn-realaudio        | ram, ra         | RealAudio file         |
|        | audio/x-pn-realaudio-plugin | rpm             | RealAudio plug-in file |

|              |                          |                           |                                             |
|--------------|--------------------------|---------------------------|---------------------------------------------|
|              | video/x-sgi-movie        | movie                     | Microsoft SGI video                         |
|              | video/quicktime          | qt, mov                   | Quicktime video                             |
| Applications | application/acad (NCSA)  | dwg                       | AutoCAD file                                |
|              | application/dxf (CERN)   | dxf                       | AutoCAD file                                |
|              | application/mif          | mif                       | Maker Interchange Format (Adobe FrameMaker) |
|              | application/msword       | doc, dot                  | MS Word file                                |
|              | application/mspowerpoint | ppt, ppz, pps, pot        | MS PowerPoint file                          |
|              | application/msexcel      | xls, xla                  | MS Excel file                               |
|              | application/mshelp       | hlp, chm                  | MS Windows help file                        |
|              | application/octet-stream | com, exe, bin, dll, class | Executable file or program code file        |
|              | application/pdf          | pdf                       | PDF file (Adobe Acrobat Exchange/Reader)    |
|              | application/postscript   | ai, eps, ps               | Postscript file (Adobe)                     |
|              | application/rff          | rff                       | RTF file (Microsoft)                        |

```

32 @UI: {
33   lineItem: [{ position: 40 }],
34   identification: [{ position: 40 }]
35 }
36 @EndUserText.label: 'Attachement'
37
38   @Semantics.largeObject: {
39     mimeType: 'Mimetype',
40     fileName: 'Filename',
41     acceptableMimeTypes: [ 'application/pdf' ],
42     contentDispositionPreference: #ATTACHMENT,
43     cacheControl: {
44       maxAge: #SHORT
45     }
46   }
47
48 Attachement.

```

# Factory Action

12 January 2026 12:50

```
DATA: lt_root_create      TYPE TABLE FOR CREATE /dmo/fsa_r_roottp
\\root,
      lt_child_create    TYPE TABLE FOR CREATE
/dmo/fsa_r_roottp\\root\\_child,
      lt_grandchild_create  TYPE TABLE FOR CREATE
/DMO/fsa_r_roottp\\Child\\_Grandchild,
      lt_chart_create    TYPE TABLE FOR CREATE
/dmo/fsa_r_roottp\\root\\_chart.

READ ENTITIES OF /DMO/FSA_R_RootTP IN LOCAL MODE
ENTITY Root
  ALL FIELDS WITH CORRESPONDING #( keys )
  RESULT DATA(roots)
ENTITY Root BY \\_Child
  ALL FIELDS WITH CORRESPONDING #( keys )
  RESULT DATA(children)
ENTITY Root BY \\_Chart
  ALL FIELDS WITH CORRESPONDING #( keys )
  RESULT DATA(charts)
  FAILED DATA(read_failed).

READ ENTITIES OF /DMO/FSA_R_RootTP IN LOCAL MODE
ENTITY Child BY \\_Grandchild
  ALL FIELDS WITH CORRESPONDING #( children )
  RESULT DATA(grandchildren).

lt_root_create = CORRESPONDING #( roots CHANGING CONTROL EXCEPT
DeleteHidden UpdateHidden ValidTo ).

LOOP AT lt_root_create ASSIGNING FIELD-SYMBOL(<fs_root_c>).
  <fs_root_c>-%cid = keys[ KEY entity %key = <fs_root_c>-%
key ]-%cid.
  <fs_root_c>-StringProperty = |Copied instance| ##NO_TEXT .
  <fs_root_c>-%control-ID = if_abap_behv=>mk-off.
  CLEAR: <fs_root_c>-ID.

APPEND VALUE #(
  %cid_ref = <fs_root_c>-%cid
  %target = CORRESPONDING #( children CHANGING
CONTROL EXCEPT ParentId )
) TO lt_child_create.

APPEND VALUE #(
  %cid_ref = <fs_root_c>-%cid
  %target = CORRESPONDING #( charts CHANGING
CONTROL EXCEPT ID ParentId )
) TO lt_chart_create.

DATA(lt_grandchildren) = grandchildren.
```

```

LOOP AT lt_child_create ASSIGNING FIELD-SYMBOL(<fs_child>)
  USING KEY cid WHERE %cid_ref = <fs_root_c>-%cid.

    LOOP AT <fs_child>-%target ASSIGNING FIELD-
SYMBOL(<fs_target>).
      <fs_target>-%cid = <fs_child>-%cid_ref && sy-tabix.

        DELETE lt_grandchildren WHERE ParentID <> <fs_target>-%
key-id.

        APPEND VALUE #( %cid_ref = <fs_target>-%cid
                      %target = CORRESPONDING
#( lt_grandchildren CHANGING CONTROL EXCEPT ID ParentId RootId )
          ) TO lt_grandchild_create.

        <fs_target>-%control-ID = if_abap_behv=>mk-off.
        CLEAR: <fs_target>-ID.

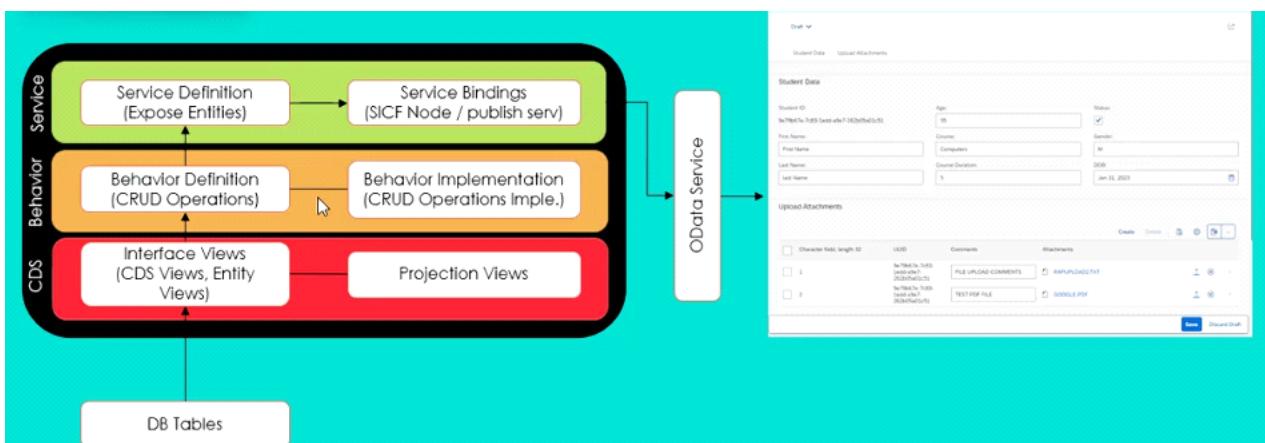
        lt_grandchildren = grandchildren.
      ENDLOOP.
    ENDLOOP.
  ENDLOOP.

MODIFY ENTITIES OF /DMO/FSA_R_RootTP IN LOCAL MODE
  ENTITY Root
    CREATE
      FROM lt_root_create
    CREATE BY \_Child
      FROM lt_child_create
    CREATE BY \_Chart
      AUTO FILL CID WITH lt_chart_create
  ENTITY Child
    CREATE BY \_Grandchild
      AUTO FILL CID WITH lt_grandchild_create
  MAPPED mapped
  REPORTED reported
  FAILED failed.

```

# UN\_MANAGED

11 December 2025 12:34



**ABAP RAP Class (Un-Managed) : Behavior Definition with Unmanaged option selected.**



## Component of the Derived Types

| Data Type |          |
|-----------|----------|
| %cid      | %control |
| %cid_ref  | %fail    |
| %tky      | %msg     |
| %is_draft |          |
| %pid      |          |
| %param    |          |

```

1 unmanaged implementation in class zbp_i_student_um unique;
2 strict ( 2 );
3
4 with draft;
5
6④ define behavior for ZI_student_um alias Student
7 draft table ystudent_d_um
8 |
9 //late numbering
10 early numbering
11
12 lock master
13 total etag Lastchangedat
14 authorization master ( instance )
15 etag master Lastchangedat
16 {
17   create ( authorization : global );
18   update ( features : instance );
19   delete ( features : instance );
20
21   draft action Edit;
22   draft action Activate optimized;
23   draft action Resume;
24   draft action discard;
25   draft determine action Prepare;
26
27 field ( readonly ) Id, GenderDesc, StuentId;
28
29
30⑤ association _result { create; }
31 mapping for ystd_table
32 {
33   Id           = id;
34   StuentId     = sid;
35   Firstname    = firstname;
36   Lastname     = lastname;
37   Age          = age;
38   Course        = course;
39   Courseduration = courseduration;
40   Status        = status;
41   Gender        = gender;
42   Dob           = dob;
43   Lastchangedat = lastchangedat;
44   Locallastchangedat = locallastchangedat;
45 }
```

```

46
47⊕define behavior for zi_results_um alias Results
48 draft table yresult_d_um
49 //late numbering
50
51 early numbering
52 lock dependent by _student
53 authorization dependent by _student
54 etag dependent by _student
55 {
56   update;
57   delete;
58   field ( readonly ) Id, Course, Semester;
59   association _student { with draft; }
60
61 // field ( readonly ) course_desc, semester_desc, semres_desc;
62
63⊕ mapping for yst_result_tab
64 {
65   Id      = id;
66   Course  = course;
67   Semester = semester;
68   Semsesult = semsesult;
69
70 }
71
72 }
```

# Early Numbers

12 December 2025 18:01

## Early Unmanaged Numbering

- Early Numbering is Implemented by Adding **EARLY NUMBERING** keyword in Behavior Definition of Entity
- Automatically assign a key to Business Object Entity Instance during Methods **Create / Create By Association**
- Keys are generated in **INTERACTION PHASE** of RAP

```
12 METHODS earlynumbering_create FOR NUMBERING
13   IMPORTING entities FOR CREATE Student.
14
15
16 METHODS earlynumbering_cba_Results FOR NUMBERING
17   IMPORTING entities FOR CREATE Student\_\_Results.
18
```

## IN LHC

```
64@ METHOD earlynumbering_create.
65   zcl_students_api_class=>get_instance( )->earlynumbering_create_student(
66     EXPORTING
67       entities = entities
68     CHANGING
69       mapped = mapped
70       failed = failed
71       reported = reported
72   ).
73
74 ENDMETHOD.
75
```

## IN Helper Class

### Definition

```
1@ CLASS zcl_students_api_class DEFINITION PUBLIC FINAL CREATE PUBLIC .
2 PUBLIC SECTION.
3   TYPES : tt_create_student TYPE TABLE FOR CREATE zi_student_um\student,
4     tt_mapped_early TYPE RESPONSE FOR MAPPED EARLY zi_student_um,
5     tt_response_early TYPE RESPONSE FOR FAILED EARLY zi_student_um,
6     tt_reported_early TYPE RESPONSE FOR REPORTED EARLY zi_student_um,
7     tt_reported_late TYPE RESPONSE FOR REPORTED LATE zi_student_um.
8
9 * Create static constructor
10  CLASS-METHODS : get_instance RETURNING VALUE(ro_instance) TYPE REF TO zcl_students_api_class,
11    get_next_id RETURNING VALUE(rv_id) TYPE sysuuid_x16,
12    get_next_student_id RETURNING VALUE(rv_studentid) TYPE ystd_id.
13
14  METHODS : earlynumbering_create_student
15    IMPORTING entities TYPE tt_create_student "table for CREATE zi_student_um\student
16    CHANGING mapped TYPE tt_mapped_early "response for mapped early zi_student_um
17    failed TYPE tt_response_early "response for failed early zi_student_um
18    reported TYPE tt_reported_early,"response for reported early zi_student_um
19    create_student
20      IMPORTING entities TYPE tt_create_student "table for create zi_student_um\student
21      CHANGING mapped TYPE tt_mapped_early "response for mapped early zi_student_um
22      failed TYPE tt_response_early "response for failed early zi_student_um
23      reported TYPE tt_reported_early,"response for reported early zi_student_um
24      save_data
25        CHANGING reported TYPE tt_reported_late. "response for reported late zi_student_um
26  PROTECTED SECTION.
27  PRIVATE SECTION.
28  CLASS-DATA : mo_instance TYPE REF TO zcl_students_api_class,
29    gt_student TYPE STANDARD TABLE OF ystd_table,
30    gt_result TYPE STANDARD TABLE OF yst_result_tab,
31    gs_mapped TYPE tt_mapped_early.
32 ENDCLASS.
```

### Implementation

```

36 CLASS zcl_students_api_class IMPLEMENTATION.
37   METHOD get_instance.
38     mo_instance = ro_instance = COND #( WHEN mo_instance IS BOUND
39                               THEN mo_instance
40                               ELSE NEW #( ) ).
41   ENDMETHOD.
42
43   METHOD get_next_id.
44     TRY.
45       rv_id = cl_uuid_factory->create_system_uuid( )->create_uuid_x16( ).
46     CATCH cx_uuid_error.
47   ENDTRY.
48   ENDMETHOD.
49
50   METHOD get_next_student_id.
51     SELECT FROM ystd_table FIELDS MAX( sid ) INTO @DATA(lv_max).
52     rv_studentid = lv_max + 1.
53   ENDMETHOD.
54
54
55   METHOD earlynumbering_create_student.
56     DATA(ls_mapped) = gs_mapped.
57     * DATA(lv_new_id) = cl_uuid_factory->create_system_uuid( )->create_uuid_x16( ).
58     DATA(lv_new_id) = get_next_id( ).
59
60     * "Buffer Table Update
61     READ TABLE gt_student ASSIGNING FIELD-SYMBOL(<lfs_student>) INDEX 1.
62     IF <lfs_student> IS ASSIGNED.
63       <lfs_student>-id = lv_new_id.
64       UNASSIGN <lfs_student>.
65     ENDIF.
66
67     mapped-student = VALUE #(
68       FOR ls_entities IN entities WHERE ( Id IS INITIAL )
69         ( %cid = ls_entities-%cid
70           %is_draft = ls_entities-%is_draft
71           id      = lv_new_id ) ).
```

## CODE

```

CLASS zcl_students_api_class DEFINITION PUBLIC FINAL CREATE
PUBLIC .
  PUBLIC SECTION.
    TYPES : tt_create_student TYPE TABLE FOR CREATE zi_student_um
\\student,
        tt_mapped_early  TYPE RESPONSE FOR MAPPED EARLY
zi_student_um,
        tt_response_early TYPE RESPONSE FOR FAILED EARLY
zi_student_um,
        tt_reported_early TYPE RESPONSE FOR REPORTED EARLY
zi_student_um,
        tt_reported_late  TYPE RESPONSE FOR REPORTED LATE
zi_student_um.

* Create static constructor
  CLASS-METHODS : get_instance RETURNING VALUE(ro_instance) TYPE
REF TO zcl_students_api_class,
    get_next_id RETURNING VALUE(rv_id) TYPE sysuuid_x16,
    get_next_student_id RETURNING VALUE(rv_studentid) TYPE
ystd_id.

  METHODS : earlynumbering_create_student
    IMPORTING entities TYPE tt_create_student "table for CREATE
zi_student_um\\student
    CHANGING mapped  TYPE tt_mapped_early  "response for mapped
early zi_student_um
```

```

        failed    TYPE tt_response_early "response for failed
early zi_student_um
            reported TYPE tt_reported_early,"response for
reported early zi_student_um
        create_student
            IMPORTING entities TYPE tt_create_student "table for create
zi_student_um\student
            CHANGING mapped    TYPE tt_mapped_early "response for mapped
early zi_student_um
                failed    TYPE tt_response_early "response for
failed early zi_student_um
                    reported TYPE tt_reported_early, "response for
reported early zi_student_um
            save_data
                CHANGING reported TYPE tt_reported_late. "response for
reported late zi_student_um
        PROTECTED SECTION.
        PRIVATE SECTION.
        CLASS-DATA : mo_instance TYPE REF TO zcl_students_api_class,
                      gt_student  TYPE STANDARD TABLE OF ystd_table,
                      gt_result   TYPE STANDARD TABLE OF yst_result_tab,
                      gs_mapped   TYPE tt_mapped_early.
    ENDCLASS.

```

```

CLASS zcl_students_api_class IMPLEMENTATION.
    METHOD get_instance.
        mo_instance = ro_instance = COND #( WHEN mo_instance IS BOUND
   THEN mo_instance
   ELSE NEW #( ) ).
    ENDMETHOD.

    METHOD get_next_id.
        TRY.
            rv_id = cl_uuid_factory=>create_system_uuid( )->
create_uuid_x16( ).
            CATCH cx_uuid_error.
        ENDTRY.
    ENDMETHOD.

    METHOD get_next_student_id.
        SELECT FROM ystd_table FIELDS MAX( sid ) INTO @DATA(lv_max).
        rv_studentid = lv_max + 1.
    ENDMETHOD.

    METHOD earlynumbering_create_student.
        DATA(ls_mapped) = gs_mapped.
        *      DATA(lv_new_id) = cl_uuid_factory=>create_system_uuid( )->
create_uuid_x16( ).
        DATA(lv_new_id) = get_next_id( ).

        *  "Buffer Table Update
        READ TABLE gt_student ASSIGNING FIELD-SYMBOL(<lfs_student>)
INDEX 1.
        IF <lfs_student> IS ASSIGNED.
            <lfs_student>-id = lv_new_id.
            UNASSIGN <lfs_student>.
        ENDIF.

        mapped-student = VALUE #( FOR ls_entities IN entities WHERE ( Id
IS INITIAL )
                                ( %cid = ls_entities-%
cid
                                %is_draft =
ls_entities-%is_draft
                                id      = lv_new_id ) ).

    ENDMETHOD.

```

# Create & Save

13 December 2025 14:26

## IN LHC

```
53⊕ METHOD create.  
54 | zcl_students_api_class=>get_instance( )->create_student(  
55     EXPORTING  
56         entities = entities  
57     CHANGING  
58         mapped    = mapped  
59         failed    = failed  
60         reported = reported  
61     ).  
62 ENDMETHOD.
```

## IN Helper Class

```
74⊕ METHOD create_student.  
75     gt_student = CORRESPONDING #( entities MAPPING FROM ENTITY ).  
76⊕ IF gt_student IS NOT INITIAL.  
77     gt_student[ 1 ]-sid = get_next_student_id( ).  
78 ENDIF.  
79     mapped = VALUE #( student = VALUE #( FOR ls_entity IN entities (  
80                     %cid = ls_entity-%cid  
81                     %is_draft = ls_entity-%is_draft  
82                     %key = ls_entity-%key ) ) ).  
83  
84 ENDMETHOD.  
85
```

## IN LSV

```
155⊕ METHOD save.  
156     zcl_students_api_class=>get_instance( )->save_data(  
157         CHANGING  
158         reported = reported  
159     ).  
160 ENDMETHOD.
```

## IN Helper Class

```
86⊕ METHOD save_data.  
87⊕ IF NOT gt_student IS INITIAL.  
88     MODIFY ystd_table FROM TABLE @gt_student.  
89 ENDIF.  
90 ENDMETHOD.  
91
```

## Code

```
METHOD create_student.  
    gt_student = CORRESPONDING #( entities MAPPING FROM ENTITY ).  
    IF gt_student IS NOT INITIAL.  
        gt_student[ 1 ]-sid = get_next_student_id( ).  
    ENDIF.  
    mapped = VALUE #( student = VALUE #( FOR ls_entity IN entities (  
                    %cid = ls_entity-%cid  
                    %is_draft = ls_entity-%  
is_draft  
                    %key = ls_entity-%  
key ) ) ).
```

ENDMETHOD.

```
METHOD save_data.  
  IF NOT gt_student IS INITIAL.  
    MODIFY ystd_table FROM TABLE @gt_student.  
  ENDIF.  
ENDMETHOD.
```

## Read & Update

13 December 2025 15:32

### IN LHC

```
86
87@ METHOD read.
88   zcl_students_api_class->get_instance( )->read_student(
89     EXPORTING
90       keys      = keys
91     CHANGING
92       result    = result
93       failed    = failed
94       reported  = reported
95   ).
96 ENDMETHOD.
97

10@ 
11@ METHOD update.
12   zcl_students_api_class->get_instance( )->update_student(
13     EXPORTING
14       entities  = entities
15     CHANGING
16       mapped    = mapped
17       failed    = failed
18       reported  = reported
19   ).
20 ENDMETHOD.
```

### IN Helper Class

```
28   read_student
29     IMPORTING keys      TYPE tt_student_keys"table for read import zi_student_um\student
30     CHANGING  result    TYPE tt_student_result"table for read result zi_student_um\student
31           failed   TYPE tt_failed_early "response for failed early zi_student_um
32           reported TYPE tt_reported_early , "response for reported early zi_student_um
33   update_student
34     IMPORTING entities TYPE tt_student_update "table for update zi_student_um\student
35     CHANGING  mapped    TYPE tt_mapped_early "response for mapped early zi_student_um
36           failed   TYPE tt_failed_early "response for failed early zi_student_um
37           reported TYPE tt_reported_early , "response for reported early zi_student_um
38   save data

108@ METHOD read_student.
109   SELECT * FROM ystd_table
110   FOR ALL ENTRIES IN @keys
111   WHERE id = @keys-Id
112   INTO TABLE @DATA(lt_stuent_data).
113
114   result = CORRESPONDING #( lt_stuent_data MAPPING TO ENTITY ).
115 ENDMETHOD.
```

OR Simple Code

```
117@ METHOD update_student.
118
119   gt_student = CORRESPONDING #( entities MAPPING FROM ENTITY ).
```

Or complex( Only Updated Fields)

### Structure for only updated fields

```
1 @EndUserText.label : 'Control Structure For Student'
2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
3 define structure zcs_stud_prop_um {
4
5   key id          : xsdboolean not null;
6   studentid      : xsdboolean;
7   firstname       : xsdboolean;
8   lastname        : xsdboolean;
9   studentage      : xsdboolean;
10  course          : xsdboolean;
11  courseduration : xsdboolean;
12  studentstatus   : xsdboolean;
13  gender          : xsdboolean;
14  dob             : xsdboolean;
15  lastchangedat   : xsdboolean;
16  locallastchanedat : xsdboolean;
17
18 }
```

### In BDEF

```

association _result { create; with gratt; }
mapping for ystd_table control zcs_stud_prop_um
{
  Id          = id;
  StuentId   = sid;
  Firstname   = firstname;
}

Helper Class
116
117 METHOD update_student.
118   DATA : lt_student_update  TYPE TABLE OF ystd_table,
119     lt_student_update_x TYPE TABLE OF zcs_stud_prop_um.
120
121   lt_student_update = CORRESPONDING #( entities MAPPING FROM ENTITY ).
122   lt_student_update_x = CORRESPONDING #( entities MAPPING FROM ENTITY USING CONTROL ).
123 IF lt_student_update IS NOT INITIAL.
124   SELECT * FROM ystd_table FOR ALL ENTRIES IN @lt_student_update
125   WHERE id = @lt_student_update-id
126   INTO TABLE @DATA(lt_student_Update_old).
127 ENDIF.
128
129 gt_student = VALUE #(
130   FOR x = 1 WHILE x <= lines( lt_student_update )
132
133 LET
134
135   ls_control_flag = VALUE #( lt_student_update_x[ x ] OPTIONAL )
136   ls_student_new = VALUE #( lt_student_update[ x ] OPTIONAL )
137   ls_student_old = VALUE #( lt_student_update_old[ id = ls_student_new-id ] OPTIONAL )
138
139 IN
140 (
141   id = ls_student_new-id
142   sid = COND #( WHEN ls_control_flag-studentid IS NOT INITIAL THEN ls_student_new-sid ELSE ls_student_old-sid )
143   firstname = COND #( WHEN ls_control_flag-firstname IS NOT INITIAL THEN ls_student_new-firstname ELSE ls_student_old-firstname )
144   lastname = COND #( WHEN ls_control_flag-lastname IS NOT INITIAL THEN ls_student_new-lastname ELSE ls_student_old-lastname )
145   age = COND #( WHEN ls_control_flag-studentage IS NOT INITIAL THEN ls_student_new-age ELSE ls_student_old-age )
146   course = COND #( WHEN ls_control_flag-course IS NOT INITIAL THEN ls_student_new-course ELSE ls_student_old-course )
147   courseduration = COND #( WHEN ls_control_flag-courseduration IS NOT INITIAL THEN ls_student_new-courseduration ELSE ls_student_old-courseduration )
148   status = COND #( WHEN ls_control_flag-studentstatus IS NOT INITIAL THEN ls_student_new-status ELSE ls_student_old-status )
149   gender = COND #( WHEN ls_control_flag-gender IS NOT INITIAL THEN ls_student_new-gender ELSE ls_student_old-gender )
150   dob = COND #( WHEN ls_control_flag-dob IS NOT INITIAL THEN ls_student_new-dob ELSE ls_student_old-dob )
151
152 )
153 .
154
155 ENDMETHOD.
```

```

METHOD read_student.
  SELECT * FROM ystd_table
  FOR ALL ENTRIES IN @keys
  WHERE id = @keys-Id
  INTO TABLE @DATA(lt_stuent_data).

  result = CORRESPONDING #( lt_stuent_data MAPPING TO ENTITY ).
```

```

ENDMETHOD.

METHOD update_student.
*   gt_student = CORRESPONDING #( entities MAPPING FROM ENTITY ).

  DATA : lt_student_update  TYPE TABLE OF ystd_table,
        lt_student_update_x TYPE TABLE OF zcs_stud_prop_um.

  lt_student_update = CORRESPONDING #( entities MAPPING FROM
ENTITY ).
  lt_student_update_x = CORRESPONDING #( entities MAPPING FROM
ENTITY USING CONTROL ).
  IF lt_student_update IS NOT INITIAL.
    SELECT * FROM ystd_table FOR ALL ENTRIES IN @lt_student_update
    WHERE id = @lt_student_update-id
    INTO TABLE @DATA(lt_student_Update_old).
  ENDIF.

  gt_student = VALUE #(
    FOR x = 1 WHILE x <= lines( lt_student_update )
    LET
      ls_control_flag = VALUE #( lt_student_update_x[ x ] OPTIONAL )
      ls_student_new = VALUE #( lt_student_update[ x ] OPTIONAL )
      ls_student_old = VALUE #( lt_student_update_old[ id =
      ls_student_new-id ] OPTIONAL )
      IN
      (
        id = ls_student_new-id
        sid = COND #( WHEN ls_control_flag-studentid IS NOT INITIAL THEN
ls_student_new-sid ELSE ls_student_old-sid )
        firstname = COND #( WHEN ls_control_flag-firstname IS NOT
INITIAL THEN ls_student_new-firstname ELSE ls_student_old-
firstname )
```

```

        lastname = COND #( WHEN ls_control_flag-lastname IS NOT INITIAL
THEN ls_student_new-lastname ELSE ls_student_old-lastname )
        age = COND #( WHEN ls_control_flag-studentage IS NOT INITIAL
THEN ls_student_new-age ELSE ls_student_old-age )
        course = COND #( WHEN ls_control_flag-course IS NOT INITIAL THEN
ls_student_new-course ELSE ls_student_old-course )
        courseduration = COND #( WHEN ls_control_flag-courseduration IS
NOT INITIAL THEN ls_student_new-courseduration ELSE ls_student_old-
courseduration )
        status = COND #( WHEN ls_control_flag-studentstatus IS NOT
INITIAL THEN ls_student_new-status ELSE ls_student_old-status )
        gender = COND #( WHEN ls_control_flag-gender IS NOT INITIAL
THEN ls_student_new-gender ELSE ls_student_old-gender )
        dob = COND #( WHEN ls_control_flag-dob IS NOT INITIAL THEN
ls_student_new-dob ELSE ls_student_old-dob )

    )
).

ENDMETHOD.

```

# Create By Association

13 December 2025 20:48

**Create By Association:** Create By Association is used when there is need to create Instances based on other Instances.

This type of creation used when there is an Associations especially in case of Composition.

In this scenario **Create By Association** is very Important because entity linked with Composition can not exists alone (Order Header and Order Item).

## IN LHC

```
114
115④ METHOD cba_Result.
116     zcl_students_api_class=>get_instance( )->cba_result(
117         EXPORTING
118             entities_cba = entities_cba
119         CHANGING
120             mapped      = mapped
121             failed      = failed
122             reported    = reported
123     ).
124 ENDMETHOD.
125
126④ METHOD earlynumbering_cba_Result.
127     zcl_students_api_class=>get_instance( )->earlynumbering_cba_result(
128         EXPORTING
129             entities = entities
130         CHANGING
131             mapped    = mapped
132             failed    = failed
133             reported  = reported
134     ).
135
136 ENDMETHOD.
***
```

## IN Helper Class

```
184
185④ METHOD earlynumbering_cba_result.
186     DATA(lv_new_id) = get_next_id( ).
187④     LOOP AT entities ASSIGNING FIELD-SYMBOL(<lfs_entities>).
188④         LOOP AT <lfs_entities>-%target ASSIGNING FIELD-SYMBOL(<lfs_result_create>).
189             mapped-results = VALUE #( ( %cid = <lfs_result_create>-%cid
190   %is_draft = <lfs_result_create>-%is_draft
191   %key = <lfs_result_create>-%key ) ).
192         ENDLOOP.
193     ENDLOOP.
194
195 ENDMETHOD.
196
197④ METHOD cba_result.
198     LOOP AT entities_cba ASSIGNING FIELD-SYMBOL(<fs_entities>).
199         gt_result = CORRESPONDING #( <fs_entities>-%target MAPPING FROM ENTITY ).
200         LOOP AT <fs_entities>-%target ASSIGNING FIELD-SYMBOL(<ls_target>).
201             mapped-results = VALUE #( ( %cid = <ls_target>-%cid
202   %is_draft = <ls_target>-%is_draft
203   %key = <ls_target>-%key ) ).
```

```

201     mapped-results = VALUE #( ( %cid = <ls_target>-%cid
202                             %is_draft = <ls_target>-%is_draft
203                             %key = <ls_target>-%key ) ).
204
205     ENDLOOP.
206
207     ENDMETHOD.

```

## IN SAVE

```

120 METHOD save_data.
121   IF NOT gt_student IS INITIAL.
122     MODIFY ystd_table FROM TABLE @gt_student.
123   ENDIF.
124   IF gt_result IS NOT INITIAL.
125     MODIFY yst_result_tab FROM TABLE @gt_result.
126   ENDIF.
127   IF gt_student_d IS NOT INITIAL.
128     DELETE FROM ystd_table WHERE id IN @gt_student_d.
129   ENDIF.
130
131 ENDMETHOD.

```

```

METHOD_earlynumbering_cba_result.
DATA(lv_new_id) = get_next_id( ).
LOOP AT entities ASSIGNING FIELD-SYMBOL(<lfs_entities>).
  LOOP AT <lfs_entities>-%target ASSIGNING FIELD-
SYMBOL(<lfs_result_create>).
    mapped-results = VALUE #( ( %cid = <lfs_result_create>-%cid
                                %is_draft = <lfs_result_create>-%
                                is_draft
                                %key = <lfs_result_create>-%
                                key ) ).
    ENDLOOP.
  ENDLOOP.

ENDMETHOD.

METHOD_cba_result.
  LOOP AT entities_cba ASSIGNING FIELD-SYMBOL(<fs_entities>).
    gt_result = CORRESPONDING #( <fs_entities>-%target MAPPING
FROM ENTITY ).
    LOOP AT <fs_entities>-%target ASSIGNING FIELD-
SYMBOL(<ls_target>).
      mapped-results = VALUE #( ( %cid = <ls_target>-%cid
                                %is_draft = <ls_target>-%is_draft
                                %key = <ls_target>-%key ) ).
    ENDLOOP.
  ENDLOOP.

ENDMETHOD.

```

# Delete

13 December 2025 23:37

## IN LHC

```
20/
98⊕ METHOD delete.
99      zcl_students_api_class=>get_instance( )->delete_student(
L00          EXPORTING
L01              keys      = keys|
L02          CHANGING
L03              mapped    = mapped
L04              failed    = failed
L05              reported  = reported
L06      ).
L07  ENDMETHOD.
```

## IN Helper Class

```
60  PRIVATE SECTION.
61      CLASS-DATA : mo_instance  TYPE REF TO zcl_students_api_class,
62                      gt_student    TYPE STANDARD TABLE OF ystd_table,
63                      gt_student_d  TYPE RANGE OF ystd_table-id,
64                      gt_result     TYPE STANDARD TABLE OF yst_result_tab,
65                      gs_mapped     TYPE tt_mapped_early.

20/
208⊕ METHOD delete_student.
209      DATA : lt_student  TYPE STANDARD TABLE OF ystd_table.
210      lt_student = CORRESPONDING #( keys MAPPING FROM ENTITY ).
211      gt_student_d = VALUE #( FOR ls_student IN lt_student
212                                  sign = 'I'
213                                  option = 'EQ'
214                                  ( low = ls_student-id ) ).
```

## In Save Method

```
120⊕ METHOD save_data.
121⊕   IF NOT gt_student IS INITIAL.
122       MODIFY ystd_table FROM TABLE @gt_student.
123   ENDIF.
124⊕   IF gt_result IS NOT INITIAL.
125       MODIFY yst_result_tab FROM TABLE @gt_result.
126   ENDIF.
127⊕   IF gt_student_d IS NOT INITIAL.
128       DELETE FROM ystd_table WHERE id IN @gt_student_d.
129   ENDIF.
130  ENDMETHOD.
```

```
METHOD delete_student.
DATA : lt_student  TYPE STANDARD TABLE OF ystd_table.
lt_student = CORRESPONDING #( keys MAPPING FROM ENTITY ).
gt_student_d = VALUE #( FOR ls_student IN lt_student
                           sign = 'I'
                           option =
```

```
'EQ'  
      ( low =  
ls_student-id ) ).  
ENDMETHOD.
```

# Lock

14 December 2025 12:07

**Pessimistic Concurrency Control (Locking) :** In RAP business objects, enqueue locks are used for pessimistic concurrency control. Exclusively locking happens on data. You define the lock in the behavior definition of the business object entity.

The lock mechanism is only defined in the behavior definition in the interface layer. Its must not be specified in a projection behavior definition.

- **Lock master** – Implemented in Un-Managed scenario where framework do not handle locking of BO.
- **Lock master unmanaged** – Implemented in managed scenario if application developer do not want framework to take care of Locking on BO

**Optimistic Concurrency Control : ETag or OData ETag**

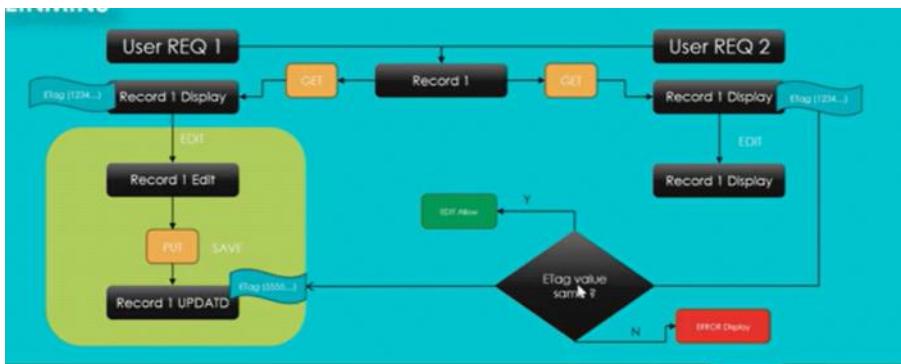
## IN LHC

```
109@ METHOD lock.  
110@   TRY.  
111@     DATA(lock) = cl_abap_lock_object_factory=>get_instance( iv_name = 'EZLOCK_STD' ).  
112@  
113@     CATCH cx_abap_lock_failure INTO DATA(exception).  
114@       RAISE SHOR_TDUMP exception.  
115@     ENDTRY.  
116@   LOOP AT keys ASSIGNING FIELD-SYMBOL(<lfs_student>).  
117@     TRY.  
118@       lock->enqueue( it_parameter = VALUE #( ( name = 'ID' value = REF #( <lfs_student>-Id ) ) )  
119@         ).  
120@       CATCH cx_abap_foreign_lock INTO DATA(foreign_lock).  
121@         APPEND VALUE #( id = <Lfs_student>-Id  
122@                           %msg = new_message_with_text(  
123@   severity = if_abap_behv_message=>severity-error  
124@   text = 'Record is Lock By ' && foreign_lock->user_name  
125@   ) ) TO reported-student.  
126@         APPEND VALUE #( id = <Lfs_student>-Id ) TO failed-student.  
127@       CATCH cx_abap_lock_failure INTO DATA(fail_lock).  
128@         RAISE SHOR_TDUMP exception.  
129@       ENDTRY.  
130@     ENDLOOP.  
131@   ENDMETHOD.
```

**Optimistic Concurrency Control : ETag or OData ETag or Entity Tag**

- A field flagged as ETag field is used to describe, uniquely, the state of a requested resource (for example a specific entity instance)
- Any changes made to the requested resource update the ETag field.

| ETag           | Meaning                                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| ETag master    | ETag field will be of main Entity where ETag is defined                                                                       |
| ETag dependent | ETag field will be of another Entity                                                                                          |
| Total ETag     | Available or used only for Draft enabled BO. Its best practice and used to transfer active data to draft data and vice versa. |



METHOD lock.  
TRY.

```

DATA(lock) = cl_abap_lock_object_factory=>
get_instance( iv_name = 'EZLOCK_STD' ).

CATCH cx_abap_lock_failure INTO DATA(exception).
  RAISE SHORTDUMP exception.
ENDTRY.

LOOP AT keys ASSIGNING FIELD-SYMBOL(<lfs_student>).
  TRY.
    lock->enqueue( it_parameter = VALUE #( ( name = 'ID'
value = REF #( <lfs_student>-Id ) ) )
).
  CATCH cx_abap_foreign_lock INTO DATA(foreign_lock).
    APPEND VALUE #( id = <Lfs_Student>-Id
%msg =
new_message_with_text(
              severity =
if_abap_behv_message=>severity-error
              text      = 'Record is
Lock By ' && foreign_lock->user_name
              )      ) TO reported-
student.
    APPEND VALUE #( id = <Lfs_Student>-Id ) TO     failed-
student.
    CATCH cx_abap_lock_failure INTO DATA(fail_lock).
      RAISE SHORTDUMP exception.
  ENDTRY.
ENDLOOP.

ENDMETHOD.

```

# Etag

14 December 2025 21:27

```
109@ METHOD create_student.  
110      gt_student = CORRESPONDING #( entities MAPPING FROM ENTITY ).  
111@ IF gt_student IS NOT INITIAL.  
112      gt_student[ 1 ]-sid = get_next_student_id( ).  
113  
114      GET TIME STAMP FIELD lv_timestampl.  
115      gt_student[ 1 ]-locallastchangedat = lv_timestampl.  
116      gt_student[ 1 ]-lastchangedat = lv_timestampl.  
117  
118      mapped = VALUE #( student = VALUE #( FOR ls_entity IN entities (  
119                      %cid = ls_entity-%cid  
120                      %is_draft = ls_entity-%is_draft  
121                      %key = ls_entity-%key ) ) ).  
122  
123      ENDIF.  
124  
125  ENDMETHOD.  
126  
  
1 unmana Behavior Definition ZI_STUDENT_UM [TRL100] - active - TRL_EN nt_um unique;  
2 strict ( 2 );  
3  
4 with draft;  
5  
6@define behavior for ZI_student_um alias Student  
7 draft table ystudent_d_um  
8  
9 //late numbering  
10 early numbering  
11  
12 lock master  
13 total etag Locallastchangedat  
14 authorization master ( instance )  
15 etag master Locallastchangedat  
16 {  
17   create ( authorization : global );  
18 }
```

# Validations(Check Before Save)

14 December 2025 21:29

## Managed Scenario

In Managed Scenario, the final check for all involved BOs is done via validations.  
In Managed Scenario VALIDATIONS are called during CHECK\_BEFORE\_SAVE method.

## Unmanaged Scenario

In unmanaged Scenario, the final check for all involved BOs is done in CHECK\_BEFORE\_SAVE method.

- FAILED
- REPORTED

CHECK\_BEFORE\_SAVE method gets called during the SAVE sequence.  
If the method returns an error in the failed parameter, the save sequence is terminated and the CLEANUP\_FINALIZE method is called.

In Helper Class , Public Section

```
18     DATA : gt_student      TYPE STANDARD TABLE OF ystd_table.

214 METHOD check_before_save.
215     DATA : gt_student_temp TYPE STANDARD TABLE OF ystd_table.
216     gt_student_temp = zcl_students_api_class->get_instance( )->gt_student.
217     IF gt_student_temp IS NOT INITIAL.
218         READ TABLE GT_student_temp ASSIGNING FIELD-SYMBOL(<LFS_student>) INDEX 1.
219         IF <LFS_student> IS ASSIGNED.
220             IF <LFS_student>-age < 22.
221                 APPEND VALUE #(  Id = <LFS_student>-id ) TO failed-student.
222                 APPEND VALUE #(  Id = <LFS_student>-id
223                                 %msg = new_message_with_text(
224                                     severity = if_abap_behv_message=>severity-error
225                                     text      = 'Minimum Age Should be 22'
226                                 ) ) TO reported-student.
227             ENDIF.
228             IF <LFS_student>-status EQ abap_false.
229                 APPEND VALUE #(  Id = <LFS_student>-id ) TO failed-student.
230                 APPEND VALUE #(  Id = <LFS_student>-id
231                                 %msg = new_message_with_text(
232                                     severity = if_abap_behv_message=>severity-error
233                                     text      = 'Set Status To Active'
234                                 ) ) TO reported-student .
235             ENDIF.
236             ENDIF.
237         ENDIF.
238 ENDMETHOD.
```

# Validations 2

14 December 2025 22:27

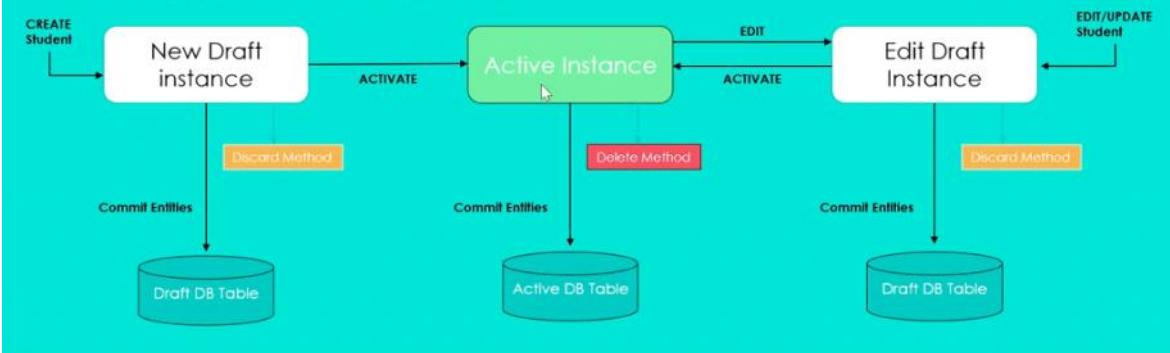
## Validations

- A validation is automatically invoked by the RAP framework if the trigger condition of the validation is fulfilled (CREATE, UPDATE, DELETE)
- When any of CREATE, UPDATE, DELETE executed for a draft instance or for an active instance, validations are triggered. Parameters – Failed and Reported
- Unmanaged and Draft Enabled BOs

## Draft Determine Action PREPARE

- Executes the determinations and validations in the behavior definition.
- The PREPARE action enables validating only draft data ( before it become active data )
- Determination and Validations which are defined in current BO can only be called in PREPARE action
- PREPARE action ensures data consistency before going to SAVE method.

## Draft Determine Action PREPARE



## IN BDEF

```
21  draft action Edit;
22  draft action Activate optimized;
23  draft action Resume;
24  draft action discard;
25  draft determine action Prepare
26  {
27    validation ( always ) validate_fileds;
28  }
29
30  field ( mandatory ) Firstname, Lastname, Age;
31  validation validate_fileds on save { create; update; }
32
```

## IN L HANDLER CLASS

```

179 METHOD validate_fileds.
180   READ ENTITIES OF ZI_student_um IN LOCAL MODE
181   ENTITY Student
182   ALL FIELDS WITH CORRESPONDING #( keys )
183   RESULT DATA(lt_student_tmp)
184   REPORTED DATA(lt_reported)
185   FAILED DATA(lt_failed).
186
187 IF lt_student_tmp IS NOT INITIAL.
188   READ TABLE lt_student_tmp ASSIGNING FIELD-SYMBOL(<lfs_student_tmp>) INDEX 1.
189 IF <lfs_student_tmp> IS ASSIGNED.
190 *   TO CLEAR THE ERRORS IN THE LIST
191   reported-student = VALUE #( ( %tky = <lfs_student_tmp>-%tky %state_area = 'VALIDATE_FST' )
192                               ( %tky = <lfs_student_tmp>-%tky %state_area = 'VALIDATE_LST' )
193                               ( %tky = <lfs_student_tmp>-%tky %state_area = 'VALIDATE_AGE' ) ).
194
195
196 IF <lfs_student_tmp>-Firstname IS INITIAL OR <lfs_student_tmp>-Lastname IS INITIAL OR <lfs_student_tmp>-Age IS INITIAL.
197   failed-student = VALUE #( ( %tky = <lfs_student_tmp>-%tky ) ).
198
199 IF <lfs_student_tmp>-Firstname IS INITIAL.
200   reported-student = VALUE #( ( %tky = <lfs_student_tmp>-%tky
201     %state_area = 'VALIDATE_FST'
202     %element-firstname = if_abap_behv=>mk-on
203       %msg = new_message(
204         id      = 'SY'
205         number = '002'
206         severity = if_abap_behv_message=>severity-error
207         v1      = 'FirstName is Required'
208       ) ) .
209
210 ENDIF.
211
212 IF <lfs_student_tmp>-lastname IS INITIAL.
213   reported-student = VALUE #( BASE reported-student ( %tky = <lfs_student_tmp>-%tky
214     %state_area = 'VALIDATE_LST'
215     %element-lastname = if abap behv=>mk-on

```

```

METHOD validate_fileds.
READ ENTITIES OF ZI_student_um IN LOCAL MODE
ENTITY Student
ALL FIELDS WITH CORRESPONDING #( keys )
RESULT DATA(lt_student_tmp)
REPORTED DATA(lt_reported)
FAILED DATA(lt_failed).

IF lt_student_tmp IS NOT INITIAL.
  READ TABLE lt_student_tmp ASSIGNING FIELD-
SYMBOL(<lfs_student_tmp>) INDEX 1.
  IF <lfs_student_tmp> IS ASSIGNED.
*    TO CLEAR THE ERRORS IN THE LIST
    reported-student = VALUE #( ( %tky = <lfs_student_tmp>-%tky
%state_area = 'VALIDATE_FST' )
                                ( %tky = <lfs_student_tmp>-%tky
%state_area = 'VALIDATE_LST' )
                                ( %tky = <lfs_student_tmp>-%tky
%state_area = 'VALIDATE_AGE' ) ).

      IF <lfs_student_tmp>-Firstname IS INITIAL OR
<lfs_student_tmp>-Lastname IS INITIAL OR <lfs_student_tmp>-Age IS
INITIAL.
        failed-student = VALUE #( ( %tky = <lfs_student_tmp>-%
tky ) ).

      IF <lfs_student_tmp>-Firstname IS INITIAL.
        reported-student = VALUE #( ( %tky = <lfs_student_tmp>-%
tky
        %state_area = 'VALIDATE_FST'
        %element-firstname = if_abap_behv=>mk-on
          %msg = new_message(
            id      = 'SY'
            number = '002'
            severity = if_abap_behv_message=>severity-error
if_abap_behv_message=>severity-error

```

```

v1      =
'FirstName is Required'

) ) ).

ENDIF.

IF <lfs_student_tmp>-lastname IS INITIAL.
reported-student = VALUE #( BASE reported-student ( %tky
= <lfs_student_tmp>-%tky
%state_area = 'VALIDATE_LST'
%element-lastname = if_abap_behv=>mk-on
%msg = new_message(
id      = 'SY'
number  = '002'
severity =
if_abap_behv_message=>severity-error
v1      =
'LastName is Required'

) ) .

ENDIF.

IF <lfs_student_tmp>-age IS INITIAL.
reported-student = VALUE #( BASE reported-student ( %tky
= <lfs_student_tmp>-%tky
*           %state_area = 'VALIDATE_AGE'
%element-age = if_abap_behv=>mk-on
%msg = new_message(
id      = 'SY'
number  = '002'
severity =
if_abap_behv_message=>severity-error
v1      = 'Age
is Required'

) ) .

ENDIF.

ENDIF.
ENDIF.
ENDIF.

ENDMETHOD.

```

# Domain Entity

16 December 2025 12:17

## On Cloud

```
1@AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Domain Entity'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YDomain_entity
6   as select from DDCDS_CUSTOMER_DOMAIN_VALUE_T( p_domain_name: 'YSTATUS' )
7 {
8   key domain_name,
9   key value_position,
10  key language,
11  value_low,
12  text|
13 }
14
```

## ON Premises

```
1@AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Entity For Domain'
4 @Metadata.ignorePropagatedAnnotations: true
5 define view entity YStatus_Domain
6   as select from dd07t
7 {
8   // key domname as Domname,
9   key ddlanguage as lang,
10  // key as4local as As4local,
11  // key valpos as Valpos,
12  // key as4vers as As4vers,
13  // domval_ld as DomvalLd,
14  // domval_hd as DomvalHd,
15  domvalue_l as Domvalue,
16  ddtext as Description
17 }
18
19 where
20   domname = 'YSTATUS_PR'
21
```

# Actions Types

17 December 2025 16:34

**Actions:** An Action can be custom operation which is used to make changes to data of BO Instance.

**Non-factory actions:** For using non Factory Actions custom logic needs to be implemented and used to changes the state of the instance.

**Factory actions:** Factory actions are used to create RAP BO entity instances.

Factory actions can be instance-bound (default) or static.

an instance-bound factory action used to copy one or more BO instances and creates new instances based on the copied data.

Static factory actions can be used to create instances with default values.

| Instance-bound factory action         | Static factory actions                       |
|---------------------------------------|----------------------------------------------|
| Used to copy one or more BO instances | Used to create instances with default values |
| factory action copy_instance [1];     | static factory action new_instance [1];      |
|                                       |                                              |

# Abstract Entity

16 December 2025 12:55

```
1@EndUserText.label: 'Abstract Entity'
2
3 @ObjectModel.resultSet.sizeCategory: #XS
4
5 define abstract entity ZA_student_entity
6 // with parameters parameter_name : parameter_type
7
8 {
9
10@EndUserText.label: 'Set Student Status'
11 @UI.defaultValue:'Y'
12 @Consumption.valueHelpDefinition: [{ entity: {
13     name      : 'YDomain_entity',
14     element    : 'text'
15 } }]
16 studentstatus : abap.char( 5 );
17
18 @UI.defaultValue:#( 'ELEMENT_OF_REFERENCED_ENTITY : Course' )
19 course       : abap.char(20);
20@UI.defaultValue:#( 'ELEMENT_OF_REFERENCED_ENTITY : Courseduration' )
21
22 courseduration : abap.numc(4);
23 @UI.defaultValue:#( 'ELEMENT_OF_REFERENCED_ENTITY :status' )
24 status        : abap_boolean;
25
26 }
27
IN BDEF
    action updatestudentstatus parameter ZA_student_entity result [1] $self;
```

```
1 @EndUserText.label: 'Abstract Entity For PopUp'
2 define abstract entity YBS_I_PopupEntity
3 // with parameters parameter_name : parameter_type
4 {
5 @Consumption.valueHelpDefinition: [{ entity: {
6     name      : 'yBS_C_CountryVH',
7     element    : 'Country'
8 } }]
9 @EndUserText.label: 'Search Country'
.0 SearchCountry : Land1;
.1@EndUserText.label: 'New Date Time'
.2 @Semantics.systemDateTime.createdAt: true
.3 NewDate       : abp_creation_tstmp;
.4 @EndUserText.label: 'Message Type'
.5 MessageType   : abap.int4;
.6 @EndUserText.label: 'Update data'
.7 FlagUpdate    : abap.char(1);
.8 @EndUserText.label: 'Show Messages'
.9 FlagMessage   : abap_boolean;
!0 }
```

```
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Search help'
4 @Metadata.ignorePropagatedAnnotations: true
5 @Search.searchable: true
6 @ObjectModel.resultSet.sizeCategory: #XS
7 define view entity yBS_C_CountryVH
8   as select from I_Country
9 {
10@    @Search.defaultSearchElement: true
11    @Search.fuzzinessThreshold: 0.8
12    @Search.ranking: #HIGH
13    @ObjectModel.text.element: [ 'Description' ]
14 key Country,
15@    @Search.defaultSearchElement: true
16    @Search.fuzzinessThreshold: 0.8
17    @Search.ranking: #LOW
18    _Text[ 1: Language = $session.system_language ].CountryName as Description
19 }
20
```

# Custom Entity

23 December 2025 10:47

Define view <CDS VIEW NAME> AS SELECT FROM MARA

```
{  
KEY MATNR,  
MITART,  
MATKL  
}
```

- 1. CDS TABLE FUNCTION
- 2. Virtual elements can be populated based on the logic( this logic can be implemented in the class )
- 3. Custom CDS Entity

## Steps to implement the Custom CDS Entity:

1. Create custom CDS entity
2. Create the Class which is implementing the logic  
Interface: if\_rap\_query\_provider  
Implement method select  
Handle the paging  
get the data from the table into internal table  
and if you want you can apply the logic to manipulate the data.  
↳ then set the internal table to io\_response->set\_data(it\_tab = lt\_tab)
3. Service Definition corresponding for custom CDS entity
4. Create the Service Binding for service definition

```
1 @EndUserText.label: 'CDS Custom Entity'  
2 @ObjectModel.query.implementedBy: 'ABAP:ZCL_CDS_CUSTOM_ENTITY'  
3 define custom entity Zcds_custom_entity  
4 // with parameters parameter_name : parameter_type  
5 {  
6   key matnr : matnr;  
7   mtart : mtart;  
8   matkl : matkl;  
9   mbrsh : mbrsh;  
10 }  
11 }
```

```

1@ class ZCL_CDS_CUSTOM_ENTITY definition public final create public .
2  public section.
3    interfaces IF_RAP_QUERY_PROVIDER .
4  protected section.
5  private section.
6 ENDCLASS.
7
8@ CLASS ZCL_CDS_CUSTOM_ENTITY IMPLEMENTATION.
9
10
11@ METHOD if_rap_query_provider~select.
12  DATA(lo.paging) = io_request->get_paging( ).
13  DATA(lv_page_size) = lo.paging->get_page_size( ).
14  DATA(lv_offset) = lo.paging->get_offset( ).
15
16  DATA(lv_max_rows) = COND #( WHEN lv_page_size = if_rap_query_paging->page_size_unlimited
17                           THEN 0
18                           ELSE lv_page_size ).
19  SELECT matnr, mtart, matkl, mbrsh FROM mara INTO TABLE @DATA(lt_data) UP TO 100 ROWS.
20
21  io_response->set_data( it_data = lt_data ).
22 *   CATCH cx_rap_query_response_set_twic. " RAP query response set twice
23 ENDMETHOD.
24 ENDCLASS.

```

or  
Gui

| Ty. | Parameter                  | Typing                            | Description |
|-----|----------------------------|-----------------------------------|-------------|
| IO  | IO_REQUEST                 | TYPE REF TO IF_RAP_QUERY_REQUEST  |             |
| IO  | IO_RESPONSE                | TYPE REF TO IF_RAP_QUERY_RESPONSE |             |
| CX  | CX_RAP_QUERY_PROV_NOT_IMPL |                                   |             |
| CX  | CX_RAP_QUERY_PROVIDER      |                                   |             |

Method      IF\_RAP\_QUERY\_PROVIDER~SELECT      active

```

1  METHOD if_rap_query_provider~select.
2    DATA(lo.paging) = io_request->get_paging( ).
3    DATA(lv_page_size) = lo.paging->get_page_size( ).
4    DATA(lv_offset) = lo.paging->get_offset( ).
5
6    DATA(lv_max_rows) = COND #( WHEN lv_page_size = if_rap_query_paging->page_size_unlimited
7                               THEN 0
8                               ELSE lv_page_size ).
9    SELECT matnr, mtart, matkl, mbrsh FROM mara INTO TABLE @DATA(lt_data) UP TO 100 ROWS.
10
11   io_response->set_data( it_data = lt_data ).
12 *   CATCH cx_rap_query_response_set_twic. " RAP query response set twice
13 ENDMETHOD.

```

# Custom Entity

24 December 2025 11:04

```
1@EndUserText.label: 'Custom Entity For Flight Data'
2 @ObjectModel:{ query.ImplementedBy: 'ABAP:YCL_CUSTOM_ENTITY1' }
3 @UI :{ headerInfo : {
4   typeName: 'Flight',
5   typeNamePlural: 'Flights',
6   title : { value: 'carrier_id' },
7   description : { value: 'connection_id' }
8 } }
9 define root custom entity YCustom_entity
10 // with parameters parameter_name : parameter_type
11 {
12@   @UI.facet      : [
13     {
14       id      : 'Flight_Data',
15       purpose : #STANDARD,
16       type    : #IDENTIFICATION_REFERENCE,
17       label   : 'Flights',
18       position: 10
19     ]
20     @UI.lineItem  : [{ position: 10 }]
21     @UI.selectionField : [{position: 10}]
22     @UI.identification: [{position: 10}]
23   key carrier_id : /dmo/carrier_id;
24@   @UI.lineItem  : [{ position: 20 }]
25   @UI.selectionField : [{position: 20}]
26   @UI.identification: [{position: 20}]
27   key connection_id : /dmo/connection_id;
28@   @UI.lineItem  : [{ position: 30 }]
29   @UI.selectionField : [{position: 30}]
30   @UI.identification: [{position: 30}]
31   key flight_date : /dmo/flight_date;
32
33@   @UI.lineItem  : [{ position: 40 }]
34   @UI.identification: [{position: 40}]
35   @Semantics.amount.currencyCode : 'currency_code'
36   price        : /dmo/flight_price;
37
38@   @UI.lineItem  : [{ position: 50 }]
39   @UI.identification: [{position: 50}]
40   currency_code : /dmo/currency_code;
41
```

Class

```
1@ CLASS ycl_custom_entity1 DEFINITION
2   PUBLIC
3   FINAL
4   CREATE PUBLIC .
5
6   PUBLIC SECTION.
7     INTERFACES : if_rap_query_provider.
8   PROTECTED SECTION.
9   PRIVATE SECTION.
10 ENDCLASS.
11
```

```


12@ CLASS ycl_custom_entity1 IMPLEMENTATION.
13@   METHOD if_rap_query_provider~select.
14@     IF io_request->is_data_requested( ).
15
16     DATA(lv_top) = io_request->get.paging( )->get_page_size( ).
17@     IF lv_top < 0.
18       lv_top = 1.
19     ENDIF.
20
21     DATA(lv_skip) = io_request->get.paging( )->get_offset( ).
22     DATA(lv_sort) = io_request->get.sort_elements( ).
23     DATA : lv_orderby TYPE string.
24
25@   LOOP AT lv_sort INTO DATA(ls_sort).
26@     IF ls_sort-descending = abap_true.
27       lv_orderby = |{ lv_orderby }{ ls_sort-element_name } DESCENDING|.
28     ELSE.
29       lv_orderby = |{ lv_orderby }{ ls_sort-element_name } ASCENDING|.
30     ENDIF.
31
32   ENDOLOOP.
33@   IF lv_orderby IS INITIAL.
34     lv_orderby = 'CARRIER_ID'.
35   ENDIF.
36
37   DATA(lv_conditions) = io_request->get.filter( )->get_as_sql_string( ).
38
39
40   SELECT FROM /dmo/flight
41   FIELDS carrier_id,connection_id,flight_date,price,currency_code,plane_type_id,
42   seats_max,seats_occupied
43   WHERE (lv_conditions)
44   ORDER BY (lv_orderby)
45   INTO TABLE @DATA(lt_flight)
46   UP TO @lv_top ROWS OFFSET @lv_skip.
47
48@   IF io_request->is_total_numb_of_rec_requested( ).
49     io_response->set_total_number_of_records( lines( lt_flight ) ).
50     io_response->set_data( lt_flight ).
51   ENDIF.
52
53   ENDIF.
54 ENDMETHOD.
55 ENDCLASS.

```

Do Service Definition and Binding . And test

# Hide Field Using VE

08 January 2026 12:28

Add Virtual Element in projection View

```
-----  
@ObjectModel.virtualElementCalculatedBy: 'ABAP:ZCL_CUSTOM_MAIL'  
virtual isHide : abap_boolean
```

Add hidden in metadata

```
@UI : { lineItem: [{ position: 40 }],  
identification: [{ position: 40,qualifier: 'pritem',hidden: #( isHide ) }] }  
Qty;  
@UI : { identification: [{ position: 50,qualifier: 'pritem',hidden: #( isHide ) }] }  
Uom;  
@UI : { lineItem: [{ position: 60 }],  
identification: [{ position: 60 ,qualifier: 'pritem'}] }|  
Price;
```

Hide A quantity and unit of measure if price is initial.

```
17@ METHOD if_sadl_exit_calc_element_read~calculate.  
18  
19     DATA : lt_data TYPE TABLE OF yp_mm_pr_itm.  
20  
21     lt_data = CORRESPONDING #( it_original_data ).  
22@     LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<fs_data>).  
23         <fs_data>-isHide = COND abap_boolean( WHEN <fs_data>-Price IS INITIAL THEN abap_true  
24                         ELSE abap_false ).  
25     ENDLOOP.  
26  
27     ct_calculated_data = CORRESPONDING #( lt_data ).  
28  
29 ENDMETHOD.
```

# UN\_managed2

18 December 2025 09:32

## ❖ In this Tutorial Series

- ❖ We will understand
  - ❖ “Unmanaged Scenario” of SAP RAP Development
  - ❖ Difference between Managed and Unmanaged RAP
  - ❖ Artifacts that constitute to the process

## ❖ Hands On: by the end of this series

- ❖ We will create a transactional application using unmanaged RAP BO
- ❖ We will understand the concept that can be used. For example, authorizations, numbering, validations, determinations, actions etc.

# Introduction

18 December 2025 09:33

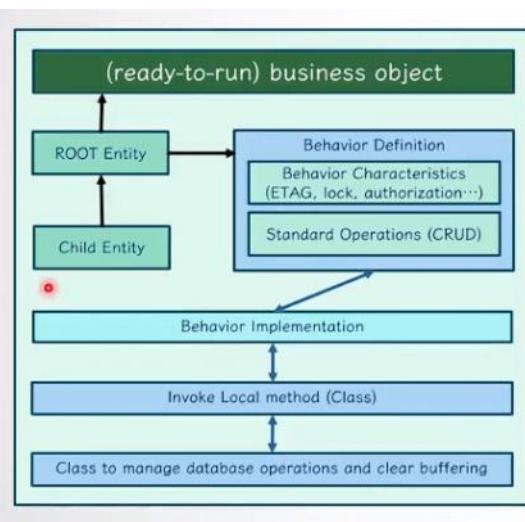
## ❖ Unmanaged RAP BO:

- ❖ Is a development approach where the developer has more control over the process.
- ❖ Custom Logic:
  - ❖ We need to write own logic for data retrieval, manipulation and persistence.
  - ❖ This gives us more flexibility on how we should process and save the data
- ❖ Direct Access to Database:
  - ❖ We can define our own model using CDS View or ABAP Class
  - ❖ We have direct access to database tables
- ❖ CRUD operations:
  - ❖ We need to explicitly implement create, read, update and delete operations. This gives us full control on the data
- ❖ Flexibility:
  - ❖ We have more freedom on implementing more complex logic for validation, authorization checks.
  - ❖ Other custom requirements can also be implemented.



## ❖ Where to use

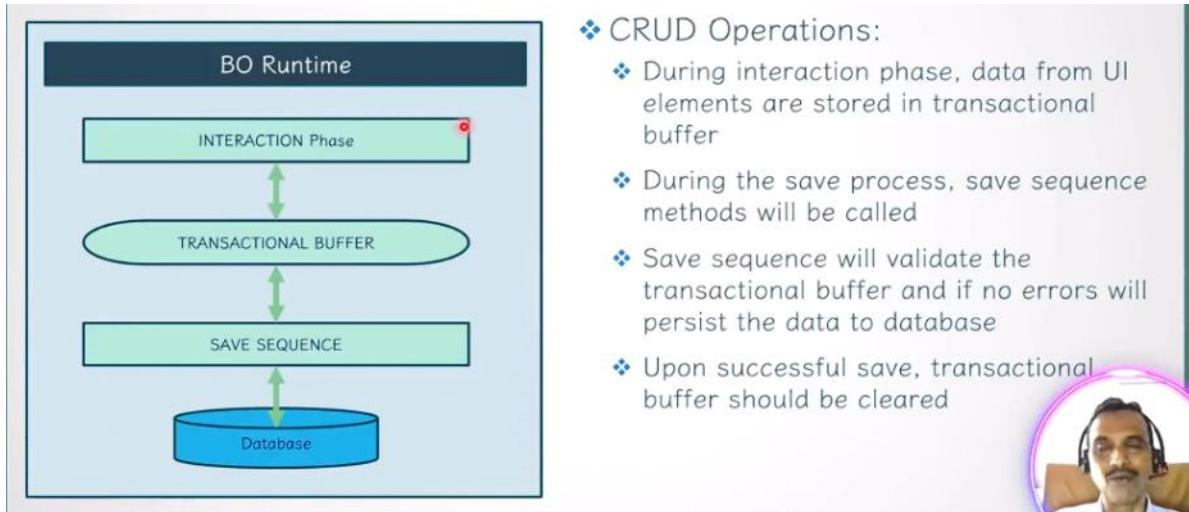
- ❖ Unlike managed scenario where RAP Framework controls the CRUD operations, we need to write logic for any operations manually.
- ❖ This approach suites best when:
  - ❖ We need to work directly with the tables (one or more at a time) without any relation.
  - ❖ Our application needs a complex business logic
  - ❖ We already have the business logic implemented as part of legacy code base



## ❖ Unmanaged Behavior

- ❖ Calls the behavior implementation (Local SAVER Class)
- ❖ Invokes the SAVE SEQUENCE to perform CRUD operations
- ❖ SAVE SEQUENCE method in turn can call the auxiliary class to perform CRUD operations.





#### ❖ CRUD Operations:

- ❖ During interaction phase, data from UI elements are stored in transactional buffer
- ❖ During the save process, save sequence methods will be called
- ❖ Save sequence will validate the transactional buffer and if no errors will persist the data to database
- ❖ Upon successful save, transactional buffer should be cleared



- ❖ SAVE Sequence is a part of RAP BO runtime
- ❖ It is triggered after at least one successful modification is performed during the interaction phase
- ❖ It saves the data from transaction buffer to database
- ❖ It is divided into 3 phases:
  - ❖ Early SAVE:
    - ❖ Calls RAP saver methods FINALIZE, CHECK\_BEFORE\_SAVE and CLEANUP\_FINALIZE.
  - ❖ Late SAVE:
    - ❖ Calls RAP saver methods ADJUST\_NUMBERS and SAVE ( or SAVE\_MODIFIED).
    - ❖ The CLEANUP method is called following a successful save.
  - ❖ Cleanup:
    - ❖ Calls RAP save method CLEANUP



MS SAP TUTORIALS
Created by Munir

```
CLASS lsc_bdef DEFINITION
  INHERITING FROM CL_ABAP_BEHAVIOR_SAVER [ABSTRACT] /FINAL.
  PROTECTED SECTION.
    [METHODS finalize REDEFINITION.]
    [METHODS check_before_save REDEFINITION.]
    METHODS adjust_numbers REDEFINITION.
    METHODS save REDEFINITION.
    [METHODS map_messages REDEFINITION.]
    [METHODS cleanup REDEFINITION.]
    [METHODS cleanup_finalize REDEFINITION.]
  ENDCLASS.

CLASS lsc_bdef IMPLEMENTATION.
  [METHOD finalize ... ENDMETHOD.]
  [METHOD check_before_save ... ENDMETHOD.]
  METHODS adjust_numbers ... ENDMETHOD.
  METHODS save ... ENDMETHOD.
  [METHOD map_messages. ... ENDMETHOD.]
  [METHOD cleanup. ... ENDMETHOD.]
  [METHOD cleanup_finalize. ... ENDMETHOD.]
ENDCLASS.
```

**❖ FINALIZE:**

- ❖ Allows to do final modification to entities.
- ❖ Method returns error in FAILED parameter and CLEANUP\_FINALIZE is called
- ❖ Its usage is optional and is part of early save phase

**❖ CHECK\_BEFORE\_SAVE:**

- ❖ This method can be used to validate the data before committing to database
- ❖ In case of validation errors CLEANUP\_FINALIZE method will be triggered.
- ❖ Its usage is optional and is part of early save phase

**❖ ADJUST\_NUMBERS:**

- ❖ It is only available in LATE NUMBERING scope
- ❖ Its usage is mandatory
- ❖ It is part of late save phase



```

CLASS lsc_bdef DEFINITION
  INHERITING FROM cl_abap_behavior_saver [ABSTRACT] [FINAL].
  PROTECTED SECTION.
    [METHODS finalize REDEFINITION.]
    [METHODS check_before_save REDEFINITION.]
    METHODS adjust_numbers REDEFINITION.
    METHODS save REDEFINITION.
    [METHODS map_messages REDEFINITION.]
    [METHODS cleanup REDEFINITION.]
    [METHODS cleanup_finalize REDEFINITION.]
  ENDCLASS.

CLASS lsc_bdef IMPLEMENTATION.
  [METHOD finalize. ... ENDMETHOD.]
  [METHOD check_before_save. ... ENDMETHOD.]
  METHOD adjust_numbers. ... ENDMETHOD.
  METHOD save. ... ENDMETHOD.
  [METHOD map_messages. ... ENDMETHOD.]
  [METHOD cleanup. ... ENDMETHOD.]
  [METHOD cleanup_finalize. ... ENDMETHOD.]
ENDCLASS.

```

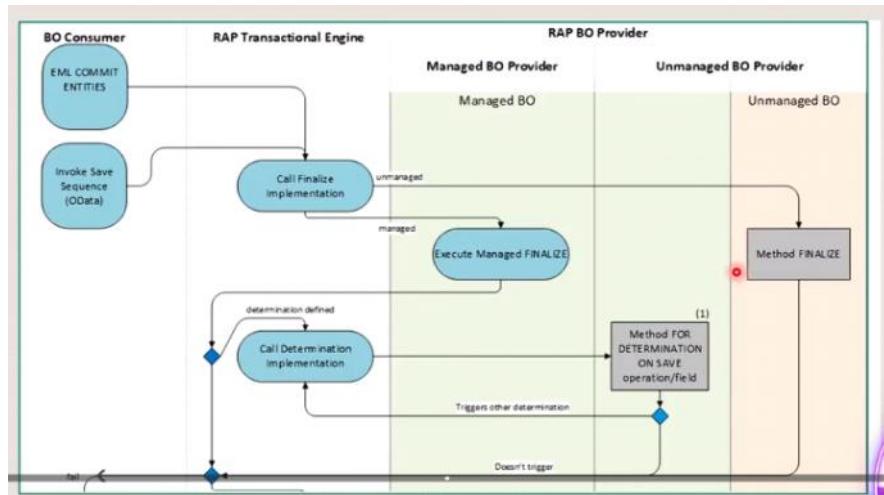
- ❖ **SAVE:**
  - ❖ Its use is **mandatory**
  - ❖ Commits database changes
- ❖ **MAP\_MESSAGES:**
  - ❖ Its usage is optional.
  - ❖ It is called when the RAP BO is defined as foreign entity in the behavior definition
- ❖ **CLEANUP/CLEANUP\_FINALIZE**
  - ❖ It is used to clear transactional buffer
  - ❖ It is called when ROLLBACK ENTITIES is triggered

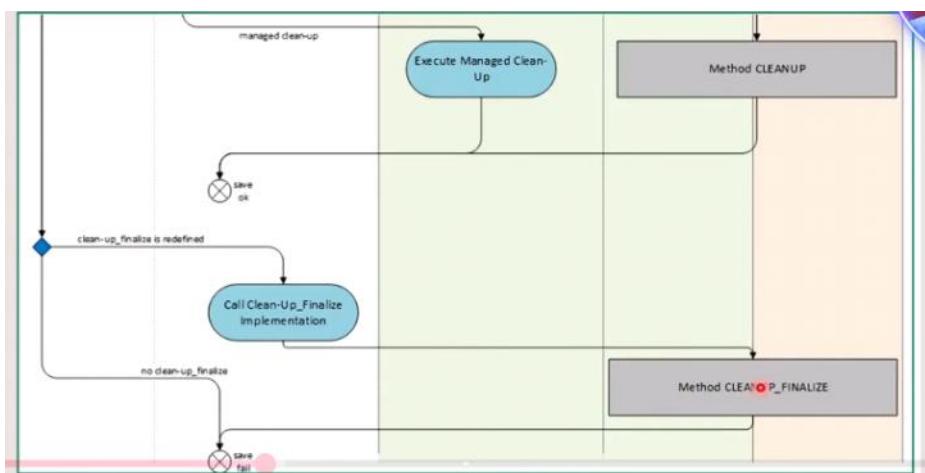
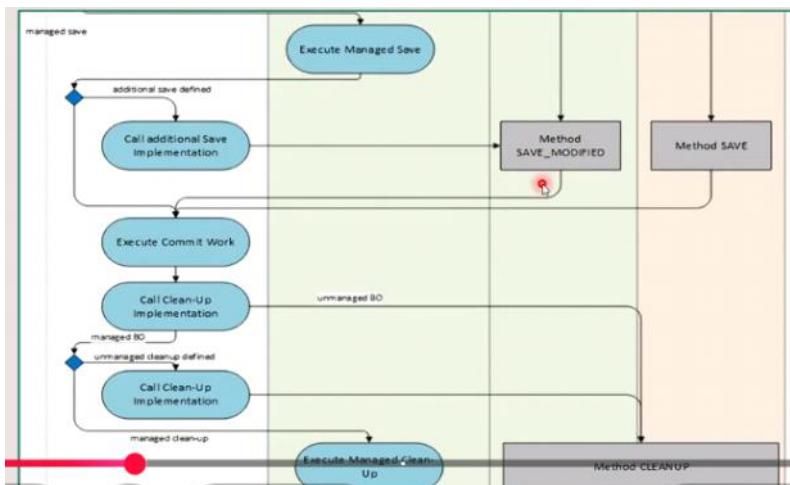
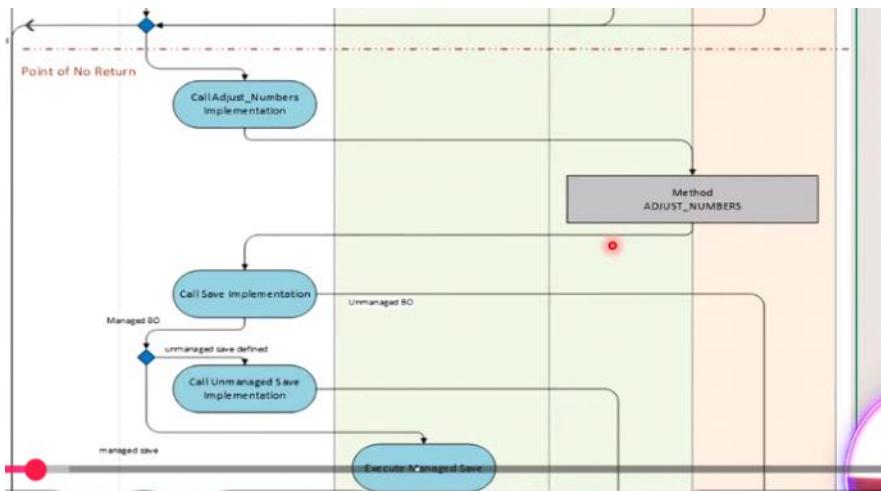
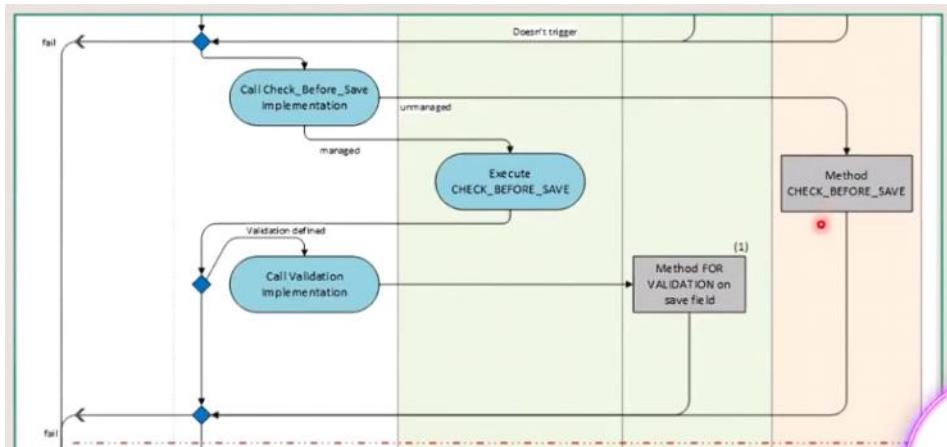


```

91 CLASS lsc_YPR_HEADER_I DEFINITION INHERITING FROM cl_abap_behavior_saver.
92   PROTECTED SECTION.
93
94     METHODS finalize REDEFINITION.
95
96     METHODS check_before_save REDEFINITION.
97
98     METHODS adjust_numbers REDEFINITION.
99
.00     METHODS save REDEFINITION.
.01
.02     METHODS cleanup REDEFINITION.
.03
.04     METHODS cleanup_finalize REDEFINITION.
.05
.06   ENDCLASS.
.07

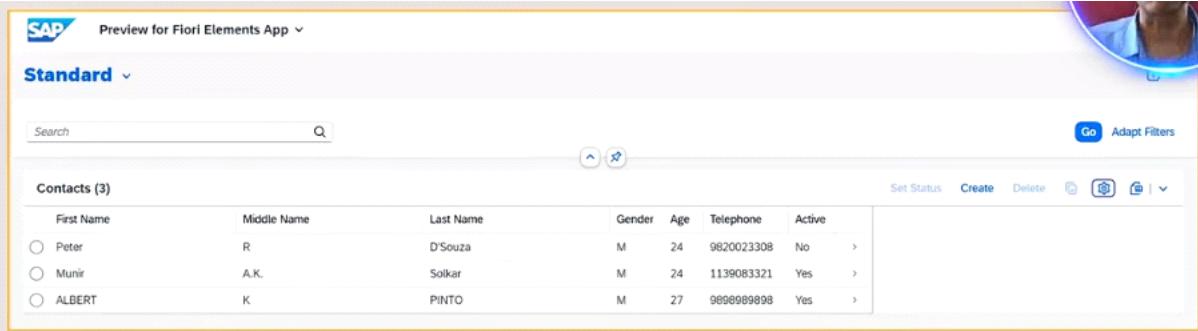
```





# Practice

19 December 2025 09:35



The screenshot shows a Fiori application interface titled "Preview for Fiori Elements App". The view is set to "Standard". At the top, there is a search bar and a "Go" button. Below the search bar, there is a "Contacts (3)" section. The table displays three contacts with the following data:

| First Name | Middle Name | Last Name | Gender | Age | Telephone  | Active |
|------------|-------------|-----------|--------|-----|------------|--------|
| Peter      | R           | D'Souza   | M      | 24  | 9820023308 | No >   |
| Munir      | A.K.        | Solkar    | M      | 24  | 1139083321 | Yes >  |
| ALBERT     | K           | PINTO     | M      | 27  | 9898989898 | Yes >  |

Below the table, there are buttons for "Set Status", "Create", "Delete", and other actions. The background of the slide features a large gray box containing text and bullet points.

We will be creating a complete Transactional Application using Unmanaged Scenario

- ❖ The application will be using the same database as we have already used for Contact Application in managed scenario
- ❖ We will see step by step on how to create the artifacts and what are the limitations

# Create

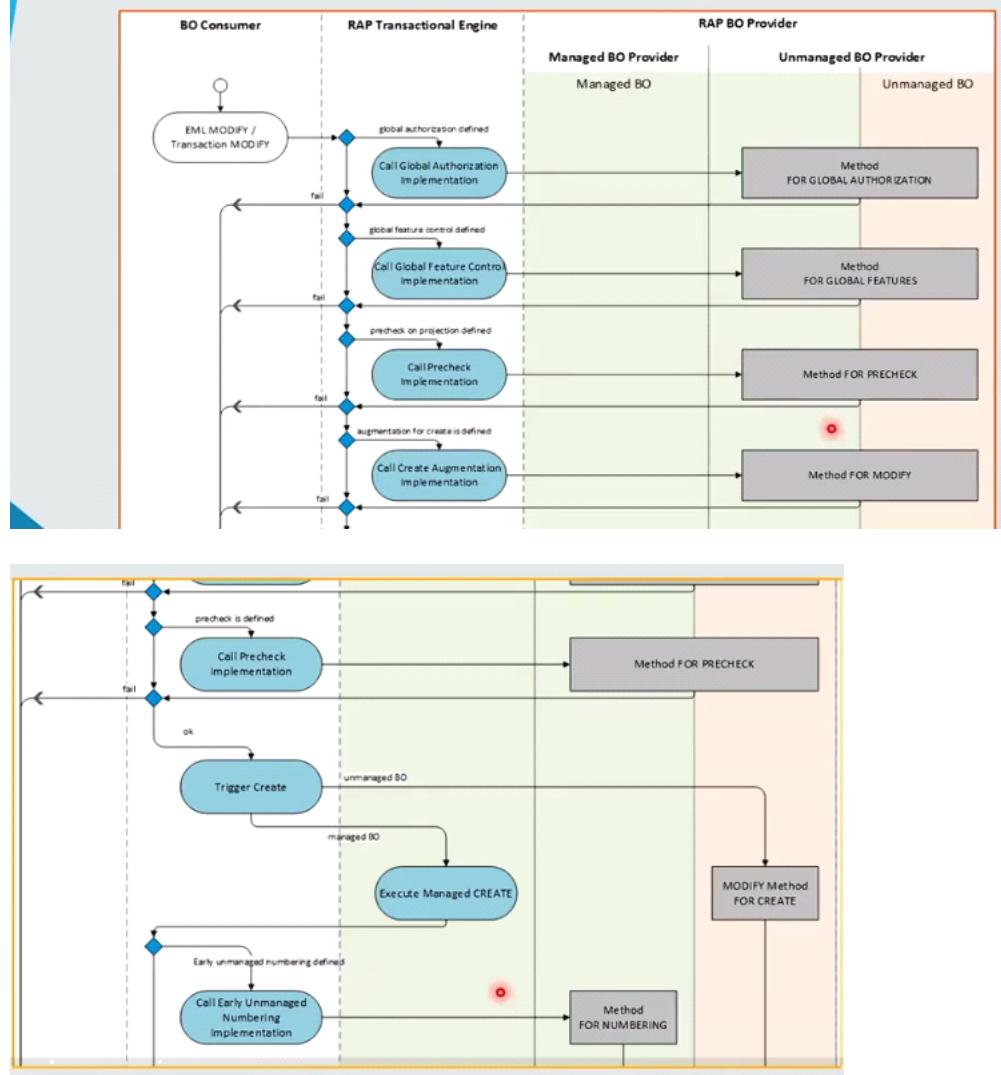
22 December 2025 10:07

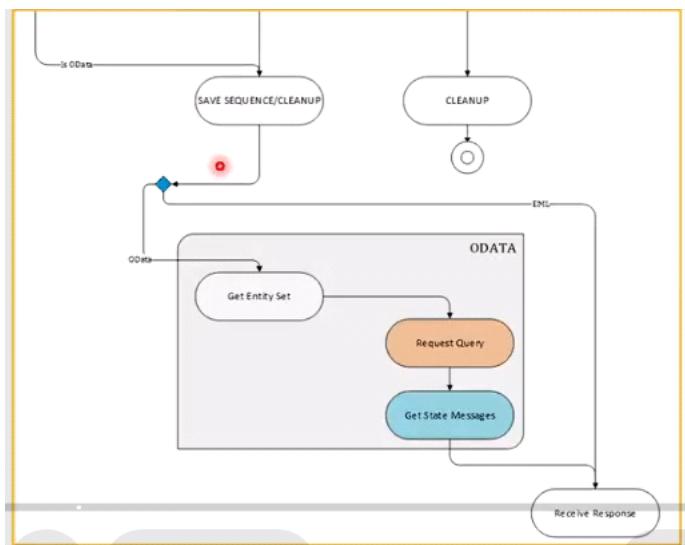
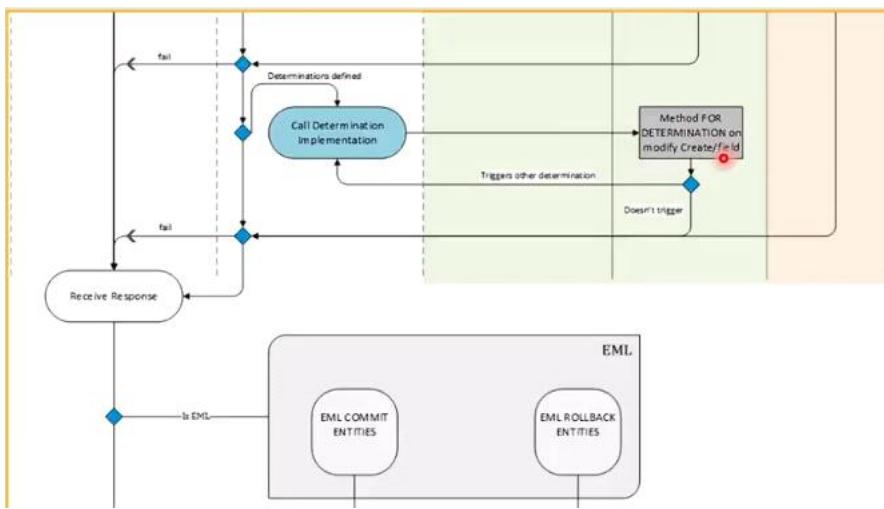
❖ We will understand

❖ "CREATE Operation" runtime

❖ Sequence of methods that are called during the create operation

## Create Operation Runtime - 1





```
METHODS create FOR MODIFY
  IMPORTING entities FOR CREATE UNMGD_Contact.
```

```
METHODS finalize REDEFINITION.
METHODS adjust_numbers REDEFINITION.
METHODS check_before_save REDEFINITION.
METHODS save REDEFINITION.
METHODS cleanup REDEFINITION.
METHODS cleanup_finalize REDEFINITION.
```

```
METHOD create.

DATA: ls_contact_in  TYPE zms_cl_aux_contact_um->gty_contact_u,
      ls_contact_out TYPE zms_cl_aux_contact_um->gty_contact_u,
      lt_messages      TYPE zms_cl_aux_contact_um->gtt_messages.

LOOP AT entities INTO DATA(ls_entities).
  ls_contact_in = CORRESPONDING #( ls_entities
    MAPPING FROM ENTITY USING CONTROL ).
  zms_cl_aux_contact_um->get_instance( )->create_contact(
    EXPORTING
      is_contact      = ls_contact_in
    IMPORTING
      es_contact      = ls_contact_out
      et_messages     = lt_messages ).
```

```
METHOD adjust_numbers.

zms_cl_aux_contact_um->get_instance( )->adjust_numbers(
  IMPORTING rt_contact_mapping = DATA(lt_contact_mapping)
  rt_contactt_mapping = DATA(lt_contactt_mapping)
).

mapped-unmgd_contact = VALUE #(
  FOR lwa_contact_mapping IN lt_contact_mapping
    %tmp = VALUE #( ContactId =
      lwa_contact_mapping-preliminary-contact_id )
    ContactId   = lwa_contact_mapping-final-contact_id
  ) ).

ENDMETHOD.

METHOD check_before_save.

ENDMETHOD.

METHOD save.

zms_cl_aux_contact_um->get_instance( )->save_contact
ENDMETHOD.

METHOD cleanup.

zms_cl_aux_contact_um->get_instance( )->initialize
ENDMETHOD.
```



# Requirement

22 December 2025 22:33

2)