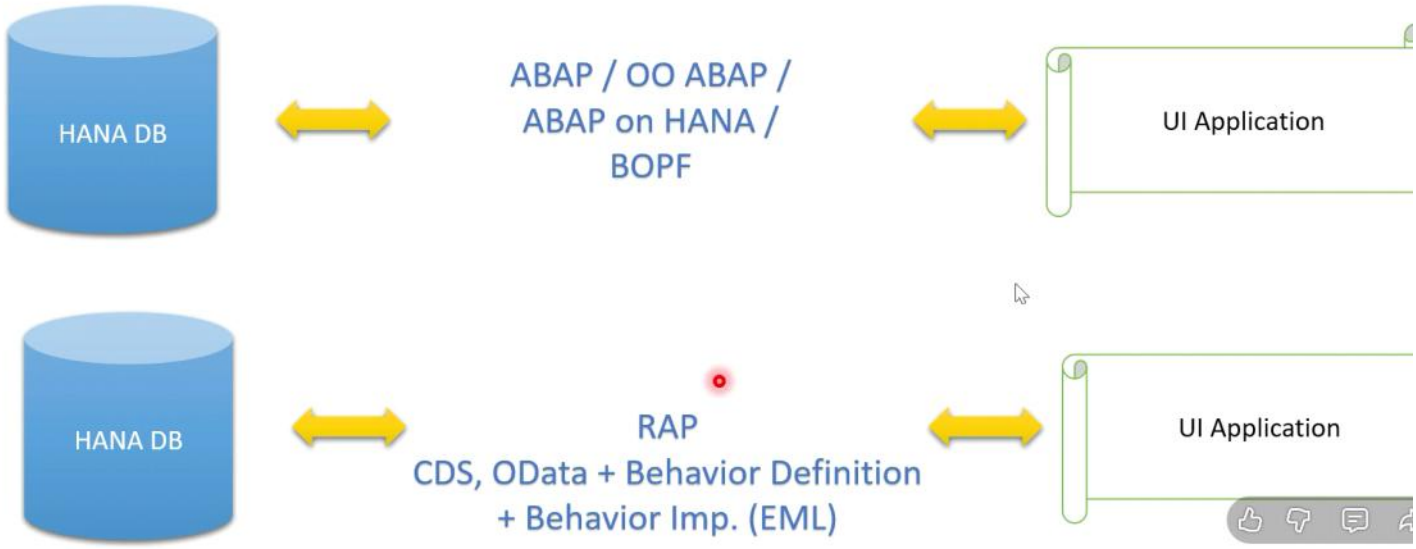
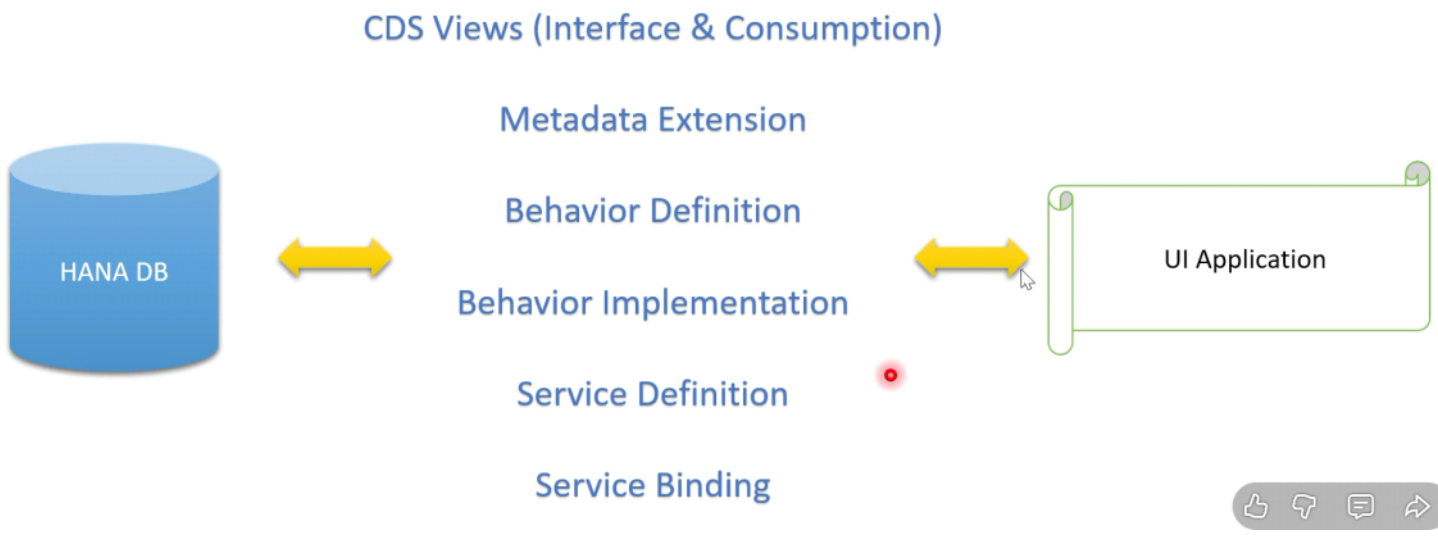
MANAGED

20 November 2025 12:03

✓ About RAP



✓ About RAP



✓ 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

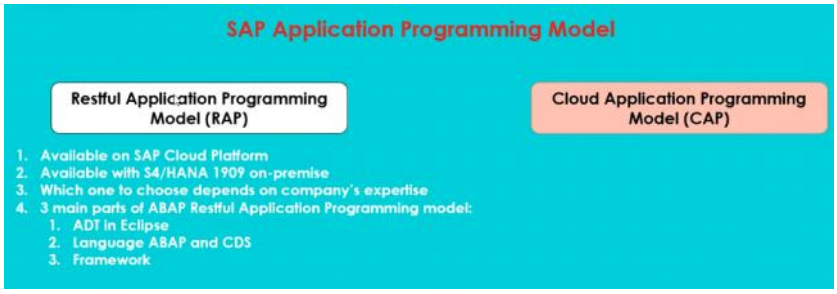
- Expose CDS View

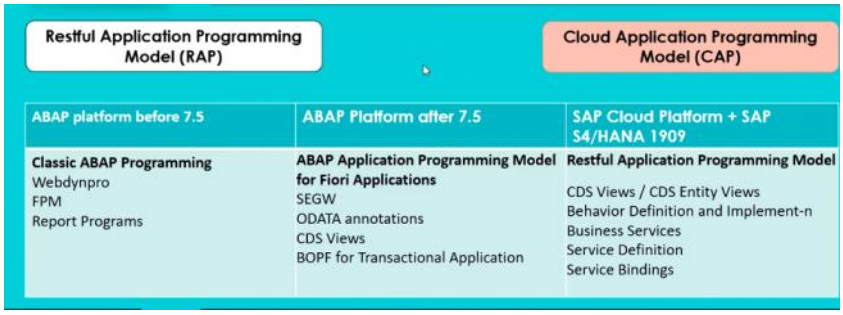
Service Binding

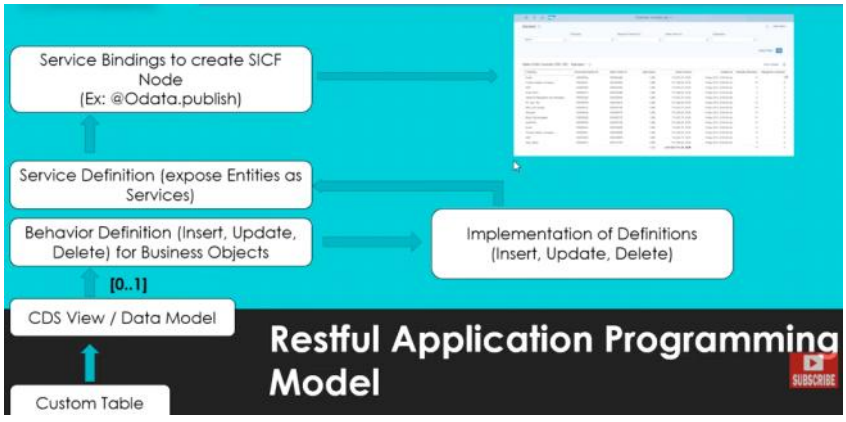
- Service creation

Intro 2

24 November 2025 12:30







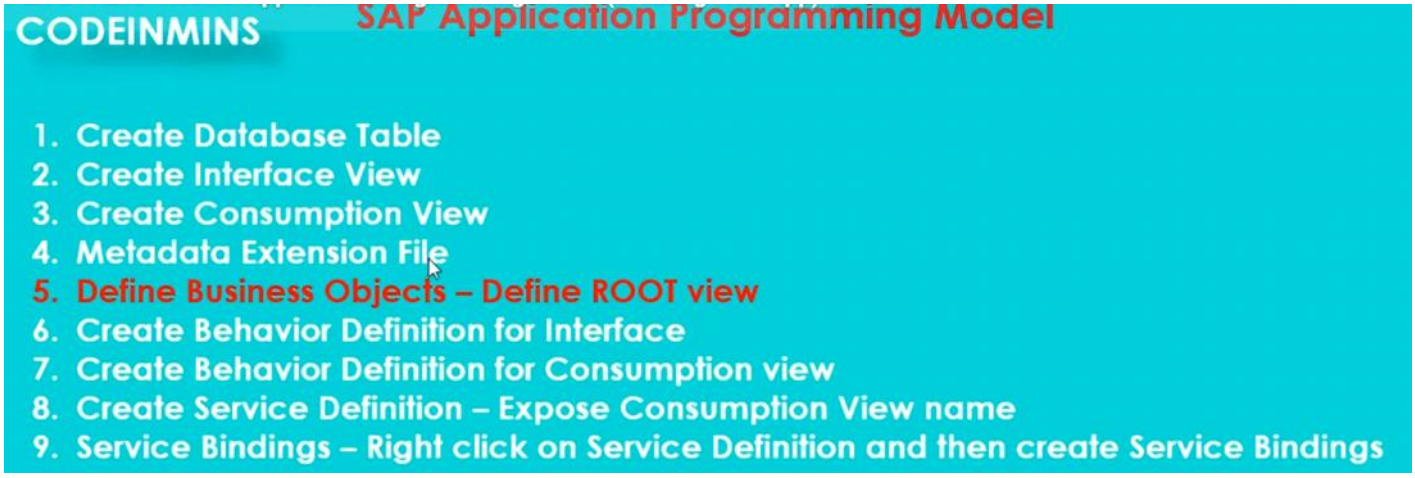


Table Creation

[TRL] YSTD_TABLE | [TRL] YRAP_STUDENT | [TRL] YRAP_C_STUDENT | [TRL] YRAP_C_STUDENT

```
@EndUserText.label : 'Stundet Table'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table ystd_table {

  key client     : abap.clnt not null;
  key id         : sysuuid_x16 not null;
  firstname      : abap.char(20);
  lastname       : abap.char(20);
  age            : abap.numc(4);
  course         : abap.char(20);
  courseduration : abap.numc(4);
  status         : abap_boolean;
  gender         : abap.char(1);
  dob            : abap.dats;

}
```

[TRL] YSTD_TABLE | [TRL] YRAP_STUDENT | [TRL] YRAP_C_STUDENT | [TRL] YRAP_C_STUDENT

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Interface View Entity For Student Table'
@Metadata.ignorePropagatedAnnotations: true
define root view entity Yrap_Student
  as select from ystd_table
{
  key id             as Id,
      firstname      as Firstname,
      lastname       as Lastname,
      age            as Age,
      course         as Course,
      courseduration as Courseduration,
      status         as Status,
      gender         as Gender,
      dob            as Dob
}
```

Below IF add Metadata Allow Annotation then Can add Metadata

[TRL] YSTD_TABLE | [TRL] YRAP_STUDENT | [TRL] YRAP_C_STUDENT | [TRL] YRAP_C_STUDENT

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Consumption View For Studnent'
@Metadata.ignorePropagatedAnnotations: true

@Metadata.allowExtensions: true

define root view entity YRAP_c_student
  as projection on Yrap_Student as Student
{
@EndUserText.label: 'Student Id'
  key Id,
  @EndUserText.label: 'First Name'
      Firstname,
      @EndUserText.label: 'Last name'
      Lastname,
      @EndUserText.label: 'Age'
      Age,
      @EndUserText.label: 'Course'
      Course,
      @EndUserText.label: 'Course Duration'
      Courseduration,
      @EndUserText.label: 'Status'
      Status,
      @EndUserText.label: 'Gender'
      Gender,
      @EndUserText.label: 'Date Of Birth'
      Dob
}
```

[TRL] YSTD_TABLE | [TRL] YRAP_STUDENT | [TRL] YRAP_C_STUDENT | [TRL] YRAP_C_STUDENT

```
@Metadata.layer: #PARTNER
annotate entity YRAP_c_student with
{
  @UI: { lineItem: [{ position: 10, label: 'Student Id' }],
         identification: [{ position: 10,label: 'Student Id' }] }
  Id;
  @UI: { lineItem: [{ position: 20, label: 'First Name' }],
         identification: [{ position: 20,label: 'First Name' }] }
  Firstname;
  @UI: { lineItem: [{ position: 30, label: 'Last Name' }],
         identification: [{ position: 30,label: 'Last Name' }] }
  Lastname;
  @UI: { lineItem: [{ position: 40, label: 'Age' }],
         identification: [{ position: 40,label: 'Age' }] }
  Age;
  @UI: { lineItem: [{ position: 50, label: 'Course' }],
         identification: [{ position: 50,label: 'Course' }] }
  Course;
  @UI: { lineItem: [{ position: 60, label: 'Course Duration' }],
         identification: [{ position: 60,label: 'Course Duration' }] }
  Courseduration;
  @UI: { lineItem: [{ position: 70, label: 'Status' }],
         identification: [{ position: 70,label: 'Status' }] }
  Status;
  @UI: { lineItem: [{ position: 80, label: 'Gender' }],
         identification: [{ position: 80,label: 'Gender' }] }
  Gender;
  @UI: { lineItem: [{ position: 90, label: 'DOB' }],
         identification: [{ position: 90,label: 'DOB' }] }
  Dob;

}
```

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.

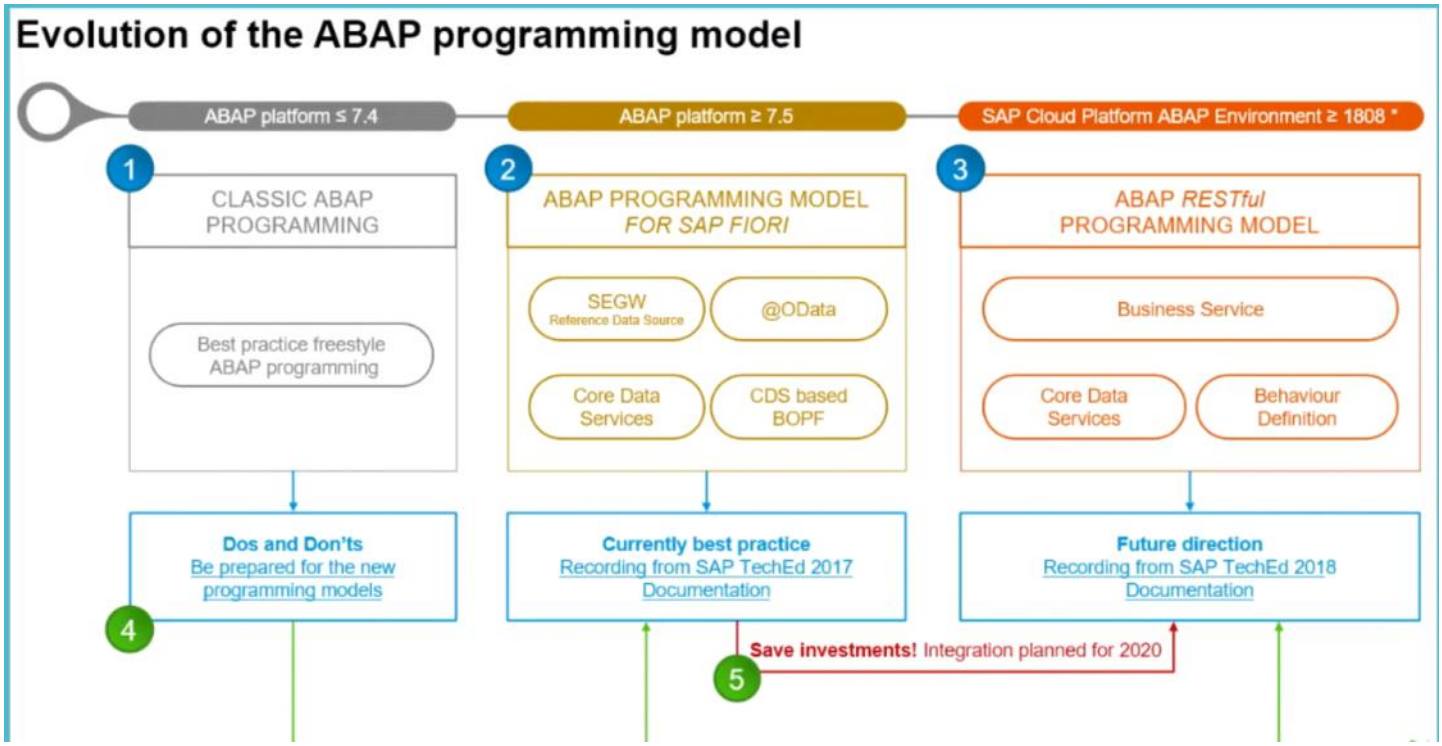
2. There is always one Entity which will represent as ROOT.

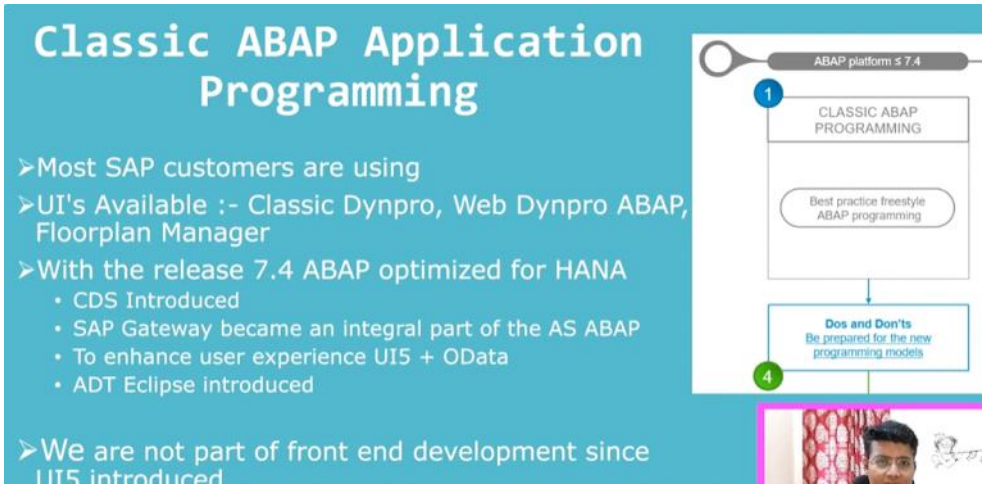
3. Business Object will have Behavior Definition. There can be only one Behavior Definition per Entity. This Behavior Definition contains different operations on Business Object ex. Create, Update, Delete etc.

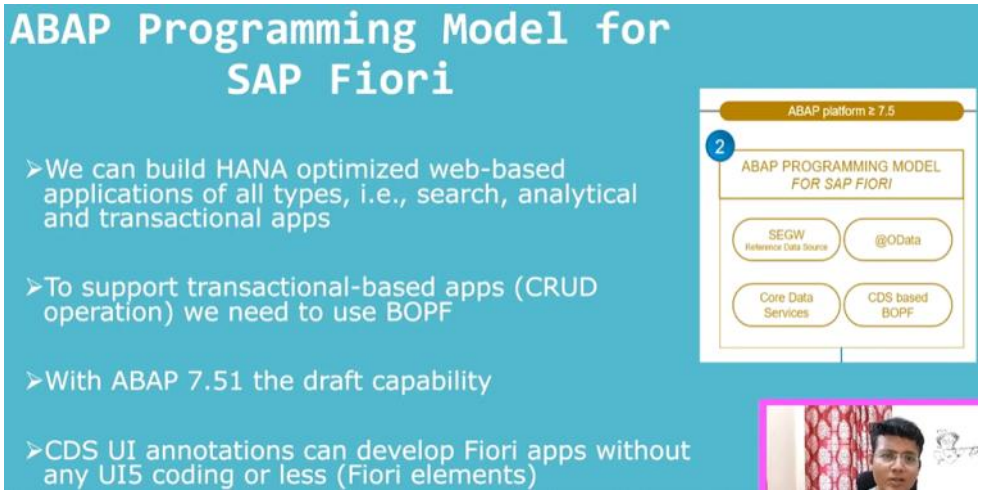
4. Business Object then can have Behavior Implementations

Intro RAP

20 November 2025 12:28



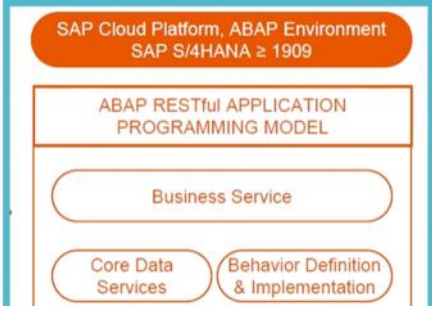




Drawbacks

- 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



- 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.

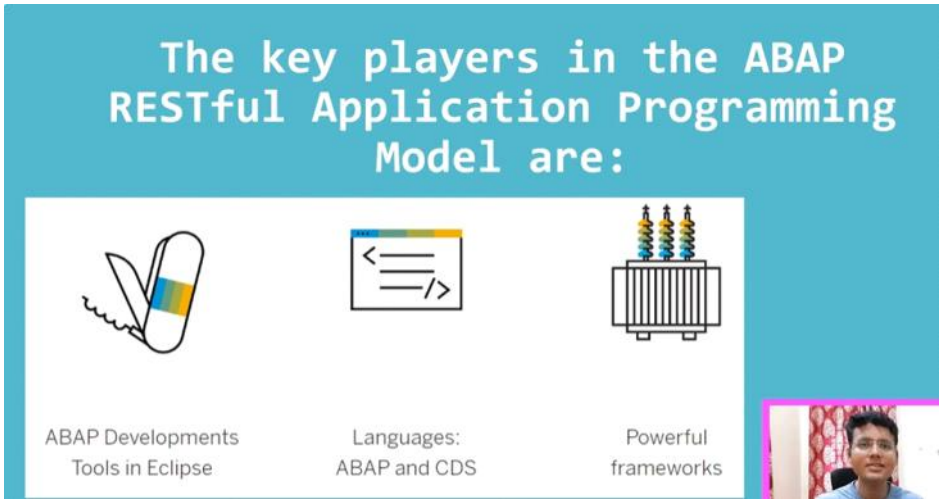
ABAP RESTful Application Programming Model

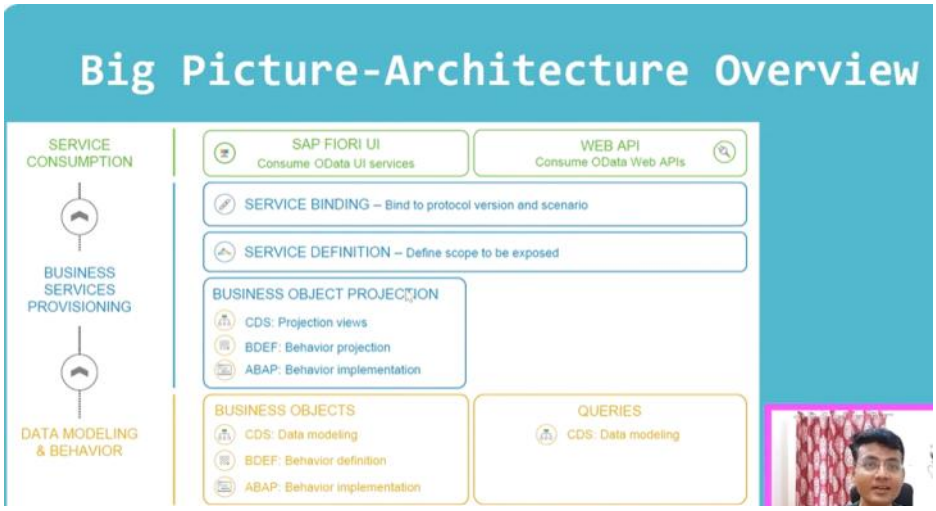
- Integrate the different concepts into the ABAP and CDS languages
- It is generally available in the **SAP BTP ABAP** Environment
- It is available from **SAP S/4HANA on-premises 1909** edition
- RAP is the only recommended programming model in SAP S/4HANA on-premise as of edition 2021 and SAP BTP ABAP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

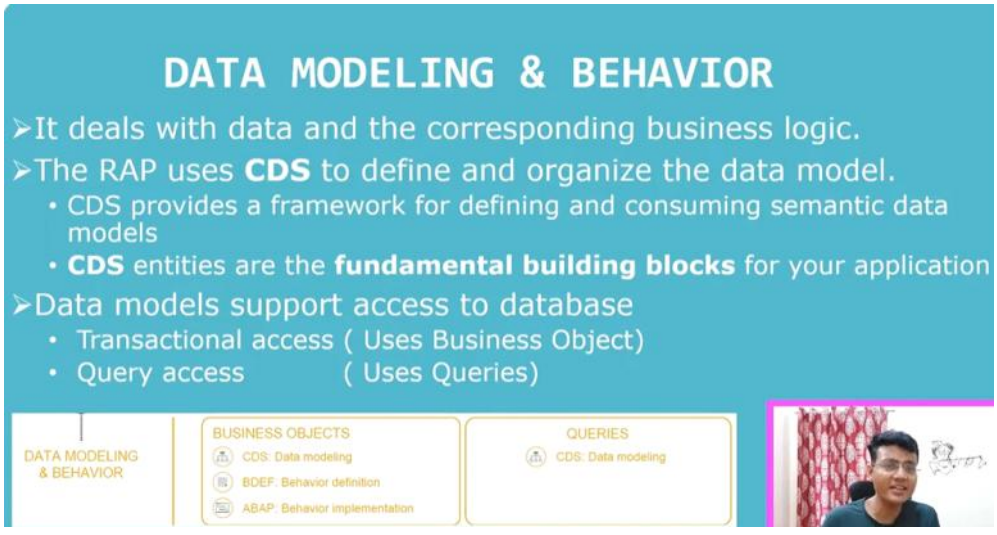
ABAP RESTful Application Programming Model

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
SERVICE CONSUMPTION
SAP FIORI UI
Consume OData UI services
WEB API
Consume OData Web APIs
SERVICE BINDING – Bind to protocol version and scenario
SERVICE DEFINITION – Define scope to be exposed
BUSINESS SERVICES PROVISIONING
BUSINESS OBJECT PROJECTION
CDS: Projection views
BDEF: Behavior projection
ABAP: Behavior implementation
DATA MODELING & BEHAVIOR
BUSINESS OBJECTS
CDS: Data modeling
BDEF: Behavior definition
ABAP: Behavior implementation
QUERIES
CDS: Data modeling


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)
DATA MODELING & BEHAVIOR
BUSINESS OBJECTS
CDS: Data modeling
BDEF: Behavior definition
ABAP: Behavior implementation
QUERIES
CDS: Data modeling

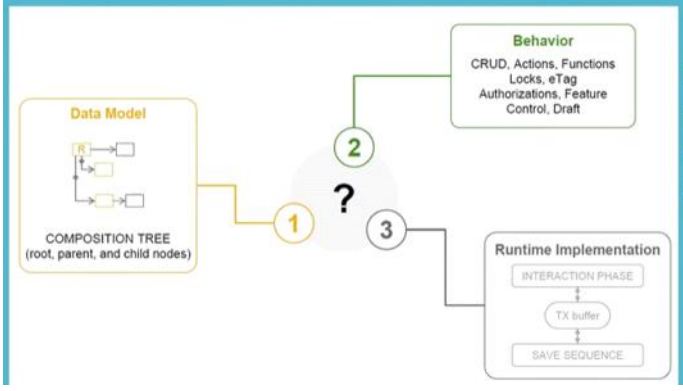
Business Object

- 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.

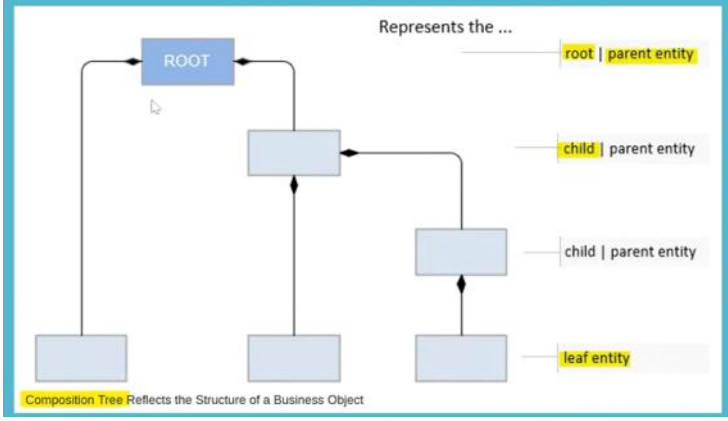
Business Object



BO is characterized by

- A structure
- A behavior and
- The corresponding runtime implementation.

BO is characterized by A structure



- BO consists of a hierarchical tree of nodes (Sales Order, Items, Schedule Lines)
- Nodes are linked by special kinds of associations, namely by compositions.

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
```

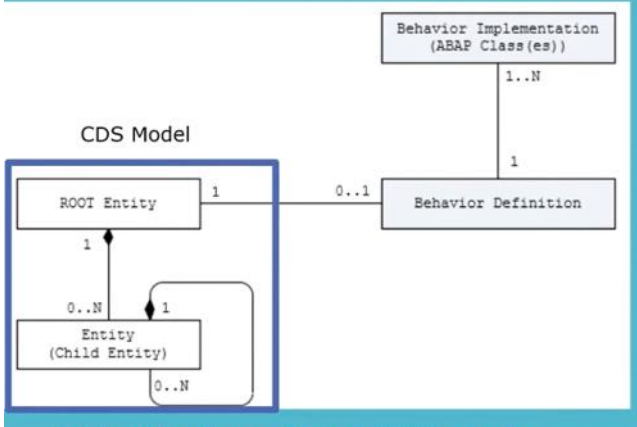
Child and Leaf Entity

- 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.

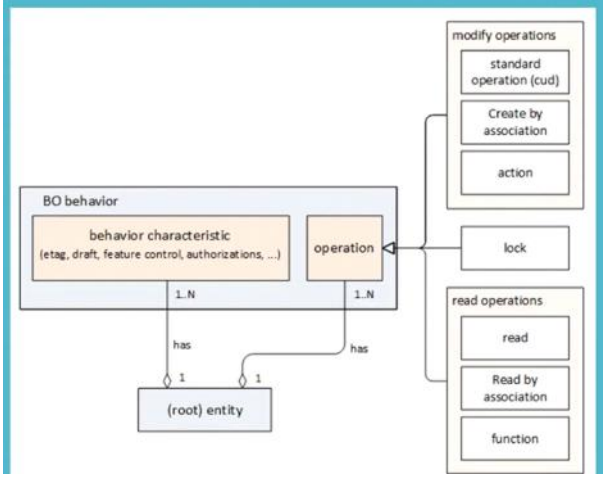
Behavior of a Business Object



Relationship Between the CDS Entities and the Business Object Behavior

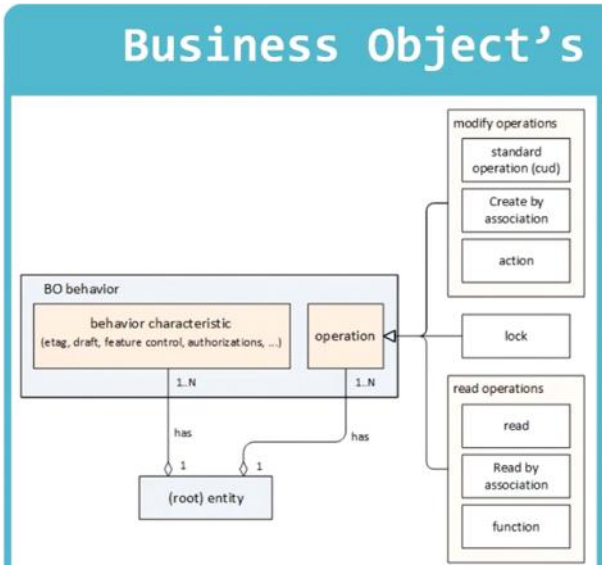
- **Behaviour definition** is an ABAP Repository object that describes the behavior of a business object.
- A behavior definition is defined using the Behavior Definition Language (BDL).
- The implementation of a behavior definition can be done in a single ABAP class (behavior pool) or can be split between 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**

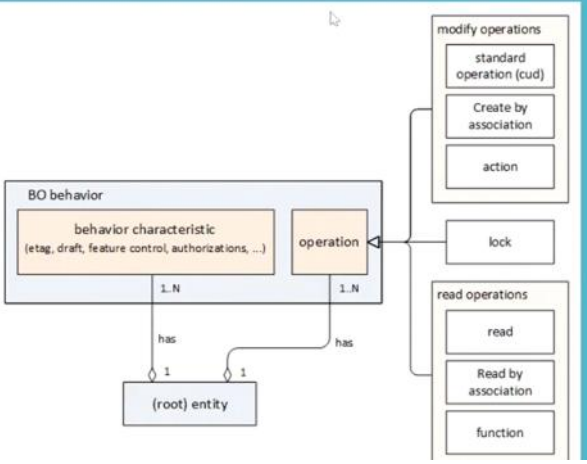
Business Object's Behavior



Operations

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

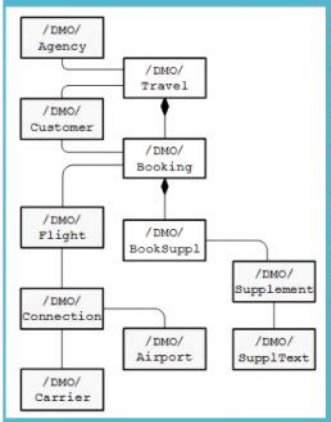
Business Object's Behavior



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.

[TRL] YRAP_SAMPLE

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'View Entity Flight Connection'
@Metadata.ignorePropagatedAnnotations: true
define view entity YRAP_Sample
  as select from /dmo/connection
{
  key carrier_id      as CarrierId,
  key connection_id   as ConnectionId,
      airport_from_id as AirportFromId,
      airport_to_id   as AirportToId,
      departure_time  as DepartureTime,
      arrival_time    as ArrivalTime,
      distance        as Distance,
      distance_unit   as DistanceUnit
}

```

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

[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

[TRL] YRAP_SAMPLE | [TRL] YRAP_SERVICE1 | [TRL] YSERVICE_BIND1

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'View Entity Flight Connection'
@Metadata.ignorePropagatedAnnotations: true

@UI.headerInfo: {
    typeName: 'Connection',                    // For Single Record
    typeNamePlural: 'Connnection_s'            // For Mutiple Records
}

define view entity YRAP_Sample
  as select from /dmo/connection
{
      @UI.lineItem: [{ position: 10, label: 'Airline'}]
  key carrier_id      as CarrierId,

      @UI.lineItem: [{ position: 20 }]
  key connection_id   as ConnectionId,

  @UI.selectionField: [{ position: 10 }]
      @UI.lineItem: [{ position: 30 }]
      airport_from_id as AirportFromId,

      @UI.selectionField: [{ position: 20 }]
      @UI.lineItem: [{ position: 40 }]
      airport_to_id   as AirportToId,

      @UI.lineItem: [{ position: 50,label: 'Departure Time' }]
      departure_time  as DepartureTime,

      @UI.lineItem: [{ position: 60, label: 'Arrival Time' }]
      arrival_time    as ArrivalTime,
```

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

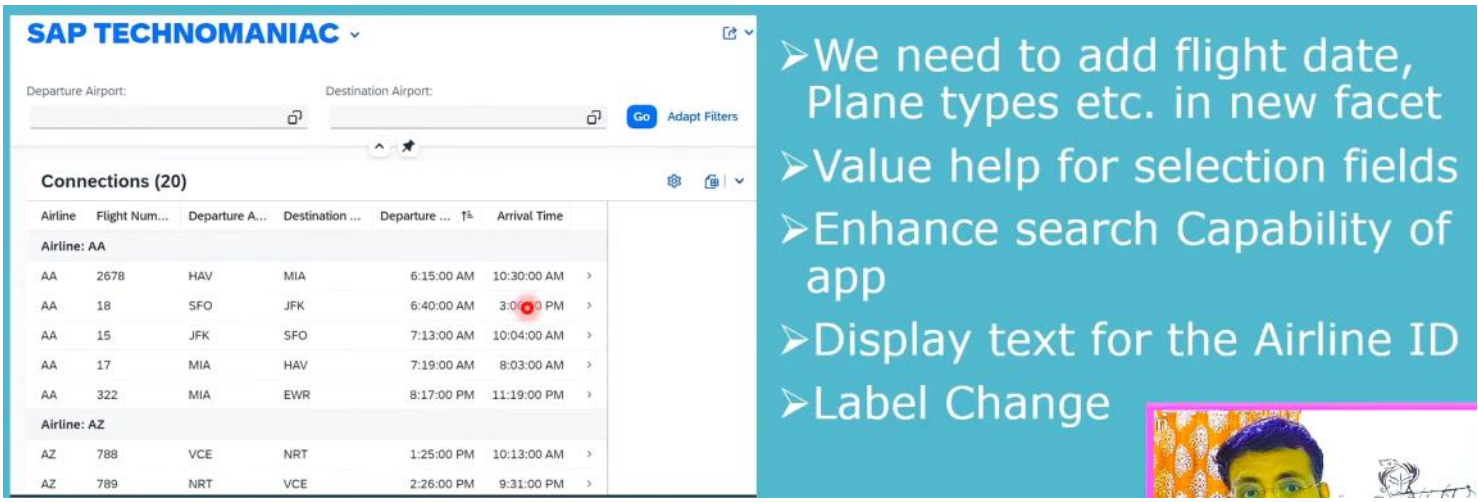
- It is used to display identification information about a record.
- This type of facet is typically used to display information such as a name, address, or contact information.
- The IDENTIFICATION_REFERENCE type facet is used in conjunction with the @UI.identification annotation.
- The @UI.identification 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.

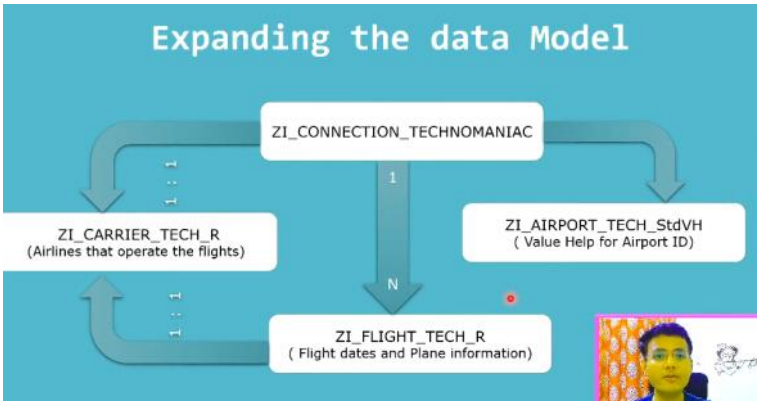
Simple Assign Facet Will not Be See But Have To Use Ui.Identification (tells Which we want See in sepated pop up)

```
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





Add Flight Using Assocaition

```
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
{

    @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' }]
  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,

    @UI.identification: [{ position: 40 }]
    @UI.selectionField: [{ position: 20 }]
    @UI.lineItem: [{ position: 40 }]
    airport_to_id   as AirportToId,

    @UI.identification: [{ position: 50 }]
    @UI.lineItem: [{ position: 50,label: 'Departure Time' }]
    departure_time  as DepartureTime,

    @UI.identification: [{ position: 60 }]
    @UI.lineItem: [{ position: 60, label: 'Arrival Time' }]
    arrival_time    as ArrivalTime,

    @UI.identification: [{ position: 70 }]
    @Semantics.quantity.unitOfMeasure: 'DistanceUnit'
    distance        as Distance,

    distance_unit   as DistanceUnit,

    _flight// Expose Association

}
```

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Flight Information'
@Metadata.ignorePropagatedAnnotations: true
define view entity YI_flight_R
  as select from /dmo/flight
{
      @UI.lineItem: [{ position: 10 }]
  key carrier_id     as CarrierId,
      @UI.lineItem: [{ position: 20 }]
  key connection_id  as ConnectionId,
      @UI.lineItem: [{ position: 30 }]
  key flight_date    as FlightDate,
      @UI.lineItem: [{ position: 40 }]
      @Semantics.amount.currencyCode: 'CurrencyCode'
      price          as Price,
      @UI.lineItem: [{ position: 50 }]
      currency_code  as CurrencyCode,
      @UI.lineItem: [{ position: 60 }]
      plane_type_id  as PlaneTypeId,
      @UI.lineItem: [{ position: 70 }]
      seats_max      as SeatsMax,
      @UI.lineItem: [{ position: 80 }]
      seats_occupied as SeatsOccupied
}
```

*[TRL] YRAP_SAMPLE | [TRL] YRAP_SERVICE1 | [TRL] YSERVICE_BIND1 | [TRL] YI

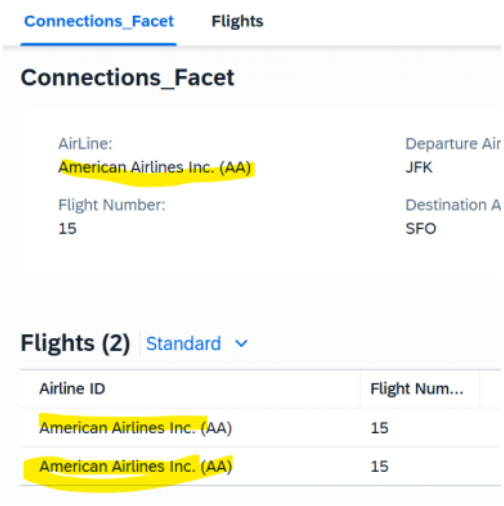
```
@EndUserText.label: 'Service Definition For Flight Connection'
define service YRap_Service1 {
  expose YRAP_Sample;
  expose YI_flight_R;
}
```

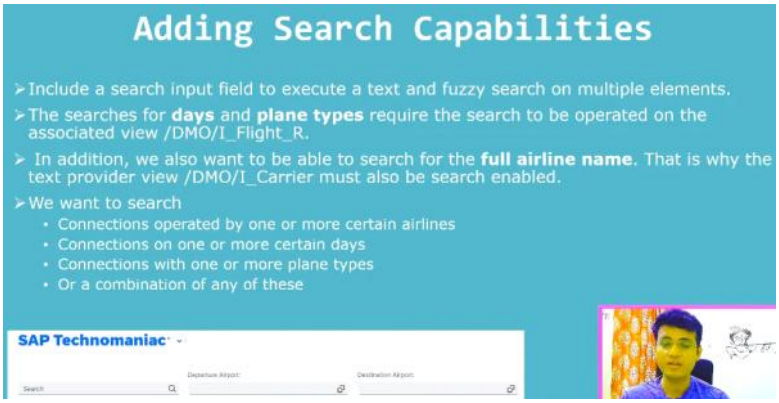
```
define view entity YRAP_Sample
  as select from /dmo/connection as connection
  association [1..*] to YI_flight_R  as _flight  on  $projection.CarrierId    = _flight.CarrierId
                                                 and $projection.ConnectionId = _flight.ConnectionId
  association [1]  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

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Flight Information'
@Metadata.ignorePropagatedAnnotations: true
define view entity YI_flight_R
  as select from /dmo/flight
  association [1]    to YI_carrier_R as _Airline on  $projection.CarrierId = _Airline.CarrierId

{
      @UI.lineItem: [{ position: 10 }]
      @ObjectModel.text.association: '_Airline'
  key carrier_id     as CarrierId,
      @UI.lineItem: [{ position: 20 }]
  key connection_id  as ConnectionId,
      @UI.lineItem: [{ position: 30 }]
  key flight_date    as FlightDate,
      @UI.lineItem: [{ position: 40 }]
      @Semantics.amount.currencyCode: 'CurrencyCode'
      price          as Price,
      @UI.lineItem: [{ position: 50 }]
      currency_code  as CurrencyCode,
      @UI.lineItem: [{ position: 60 }]
      plane_type_id  as PlaneTypeId,
      @UI.lineItem: [{ position: 70 }]
      seats_max      as SeatsMax,
      @UI.lineItem: [{ position: 80 }]
      seats_occupied as SeatsOccupied,
      _Airline
}
```

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Carrier Details'
@Metadata.ignorePropagatedAnnotations: true
define view entity YI_carrier_R
  as select from /dmo/carrier
{
  key carrier_id    as CarrierId,
  @Semantics.text: true
      name          as Name,
      currency_code as CurrencyCode
}
```

Result





[TRL] YRAP_SAMPLE | [TRL] YRAP_SERVICE1 | [TRL] YSERVICE_BIND1 | [TRL] YI_FLIGHT_R | [TRL] YI_CARRIER_R

```
@Search.searchable: true

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'
      @Search.defaultSearchElement: true
  key carrier_id      as CarrierId,

      @UI.identification: [{ position: 20 }]
      @UI.lineItem: [{ position: 20 }]
      @Search.defaultSearchElement: true
  key connection_id   as ConnectionId,

      //@Search.defaultSearchElement: true
```

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
```

```
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 arirline 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 }
```

Search Based On plane type which from associated table

```
@Search.defaultSearchElement: true
_flight, // Expose Association
@Search.defaultSearchElement: true
 Airline
```

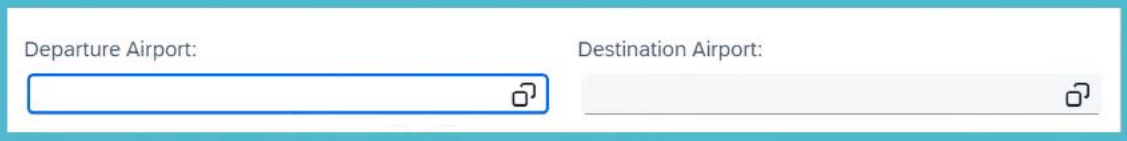
```
1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'Flight Information'
4 @Metadata.ignorePropagatedAnnotations: true
5
6 @Search.searchable: true
7
8 define view entity YI_flight_R
9   as select from /dmo/flight
10  association [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.



- 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

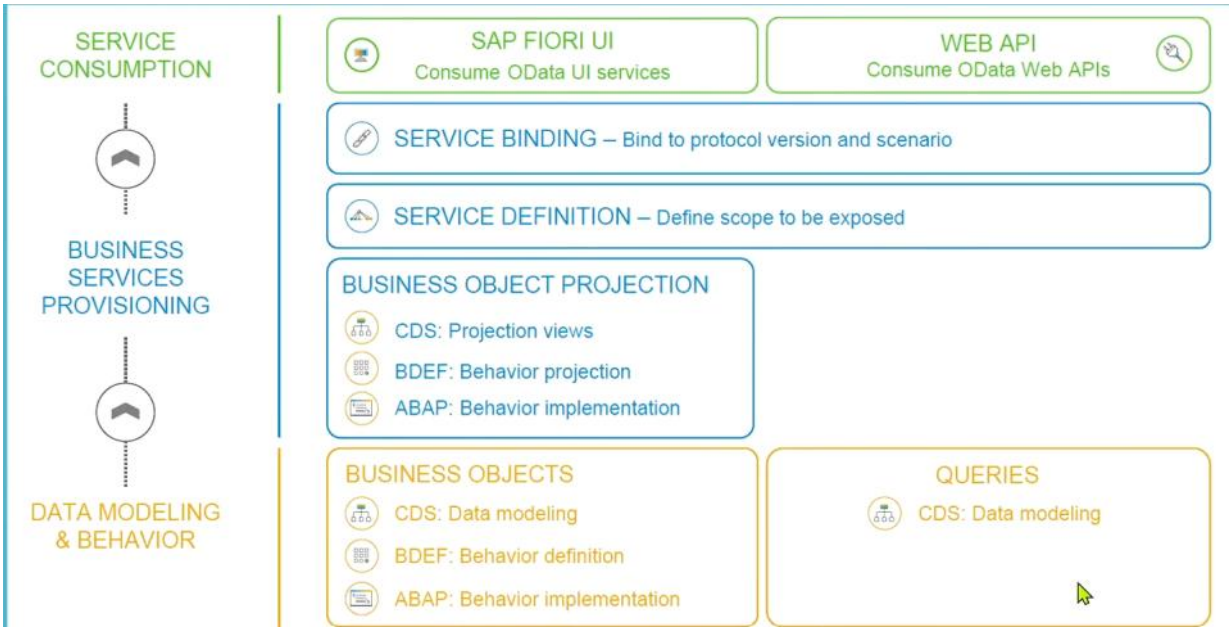
- The property in the CDS source view for which a value help is implemented (sap:value-list="standard")
- The value help provider entity type (sap:value-list="true")

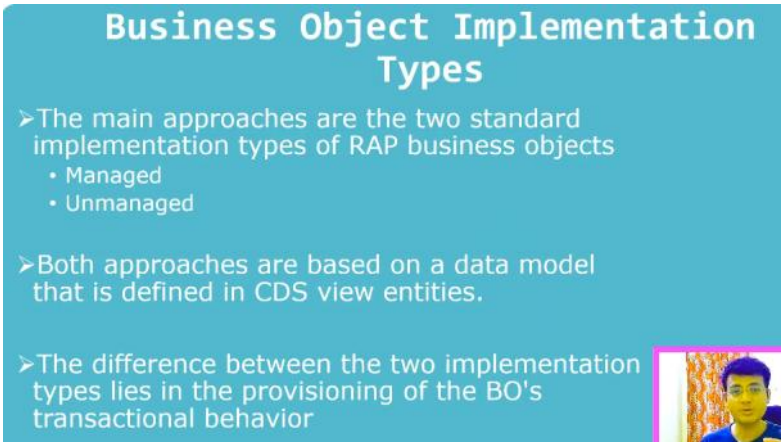
```
43    key connection_id   as ConnectionId,
44
45
46        @Consumption.valueHelpDefinition: [{ entity: {
47            name: 'YI_airport_vh1',
48            element: 'AirportId'
49        } }]
50        @Search.defaultSearchElement: true
51        @UI.identification: [{ position: 30 }]
52        @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,
```

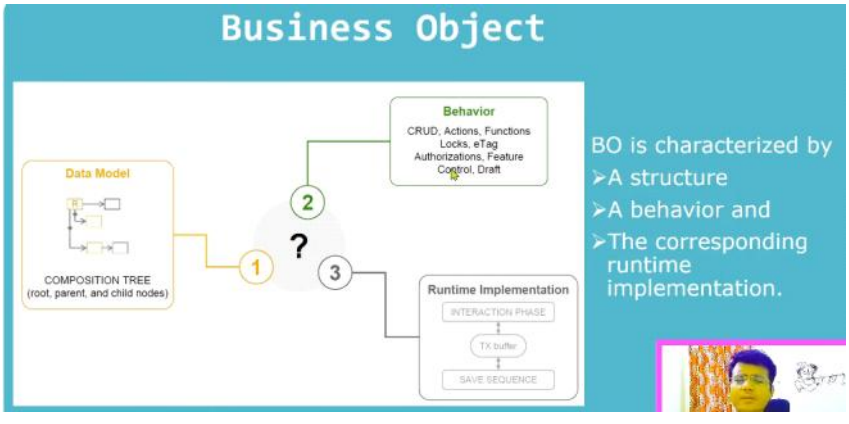
```
1  @AbapCatalog.viewEnhancementCategory: [#NONE]
2  @AccessControl.authorizationCheck: #NOT_REQUIRED
3  @EndUserText.label: 'Value Help For Airport'
4  @Metadata.ignorePropagatedAnnotations: true
5
6  @Search.searchable: true
7  define view entity YI_airport_vh1
8    as select from /dmo/airport
9  {
10       @Search.defaultSearchElement: true
11   key airport_id as AirportId,
12       @Search.defaultSearchElement: true
13       @Search.fuzzinessThreshold : 0.9
14       name       as Name,
15       @Search.defaultSearchElement: true
16       @Search.fuzzinessThreshold: 0.9
17       city       as City,
18       @Search.fuzzinessThreshold:  0.9
19       @Search.defaultSearchElement: true
20       country    as Country
21 }
22
```

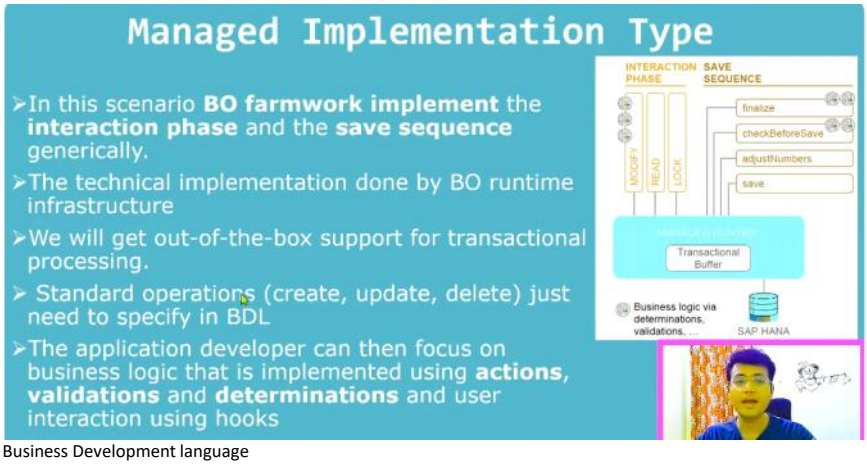
Managed

25 November 2025 15:41

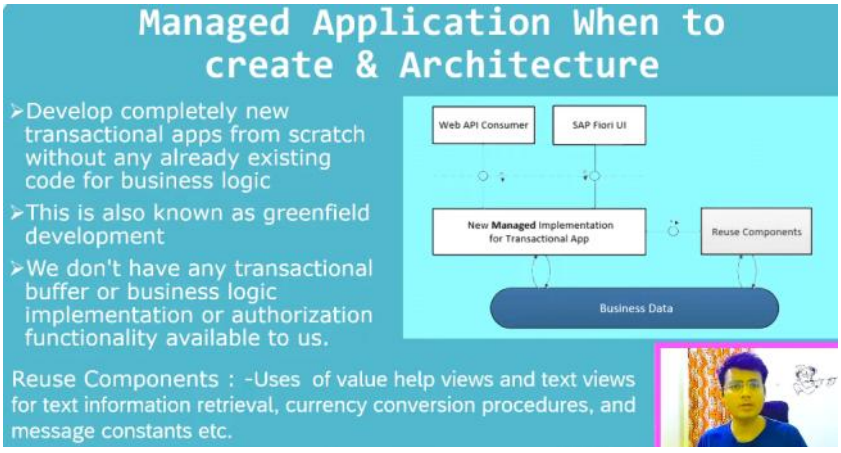




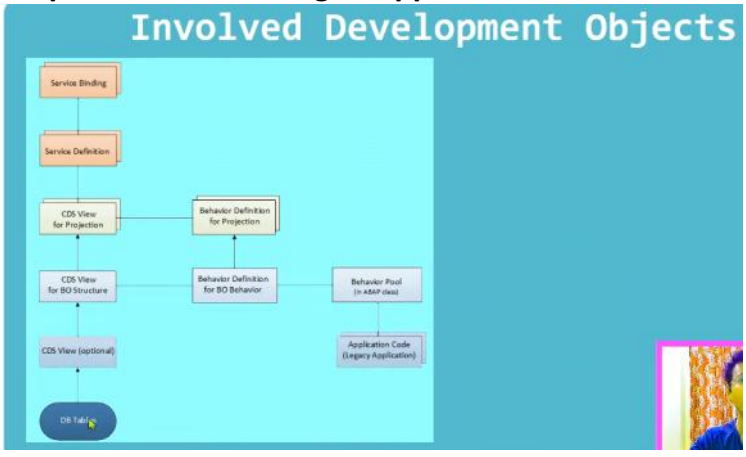




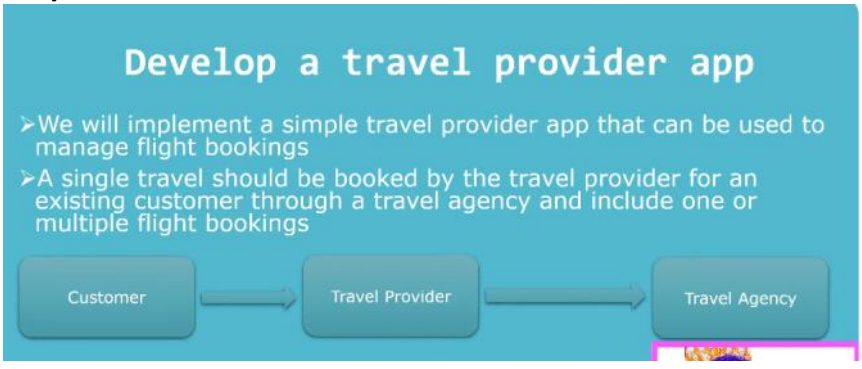
Business Development language



Steps To Create Managed Applications

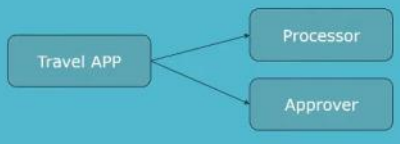


Requirement

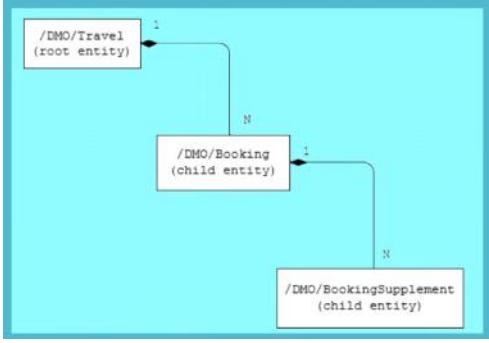


Two different views of the travel app

- 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



- The figure below shows the composition relationship between the travel, the (flight) booking and the supplement entities, where the travel entity represents the root of the data model.

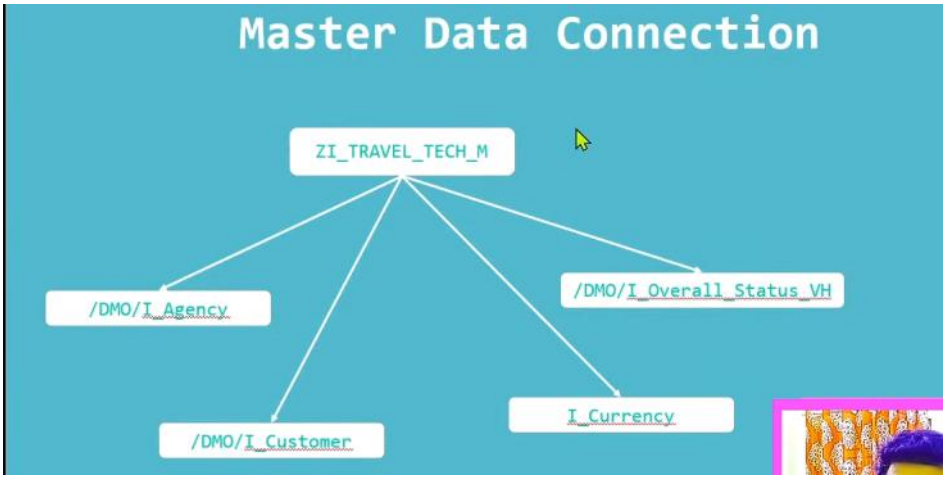
Creating Persistent Tables

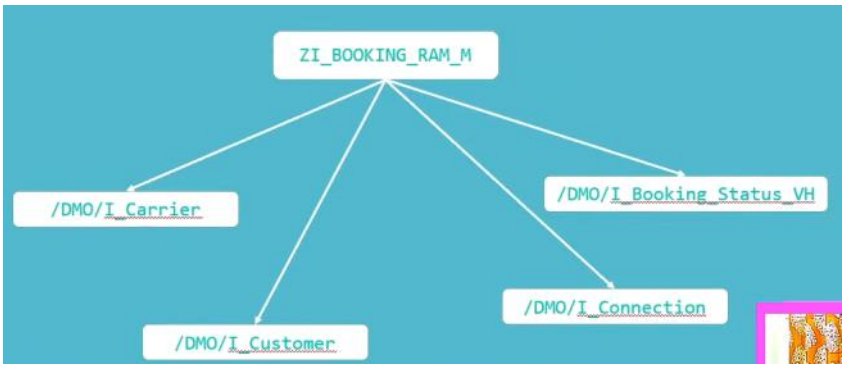
- Table ZTRAVEL_TECH_M: Persistent Table for Managing Travel Data
- Table ZBOOKING_TECH_M: Persistent Table for Managing Bookings
- Table ZBOOKSUPPL_TECH_M: Persistent Table for Managing Booking Supplement Data

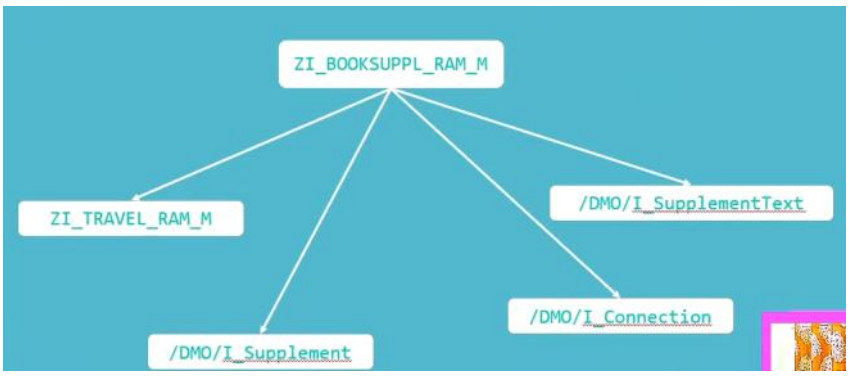
- (Table /DMO/TRAVEL_M, /DMO/BOOKING_M,/DMO/BOOKSUPPL_M)

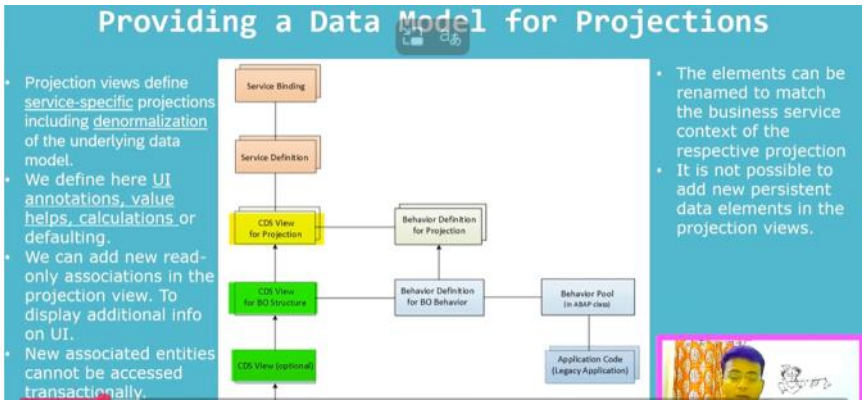
Creating CDS Data Definitions

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
ZI_TRAVEL_TECH_M
/DMO/I_Agency
/DMO/I_Customer
I_Currency
/DMO/I_Overall_Status_VH


ZI_BOOKING_RAM_M
/DMO/I_Carrier
/DMO/I_Customer
/DMO/I_Connection
/DMO/I_Booking_Status_VH


ZI_BOOKSUPPL_RAM_M
ZI_TRAVEL_RAM_M
/DMO/I_Supplement
/DMO/I_Connection
/DMO/I_SupplementText


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.
Service Binding
Service Definition
CDS View for Projection
Behavior Definition for Projection
CDS View for BO Structure
Behavior Definition for BO Behavior
Behavior Pool (in ABAP class)
CDS View (optional)
Application Code (Legacy Application)
The elements can be renamed to match the business service context of the respective projection
It is not possible to add new persistent data elements in the projection views.

[TRL] YI_TRAVEL1_M | [TRL] YI_BOOKING1_M | [TRL] YI_BOOKSUPPL1_M

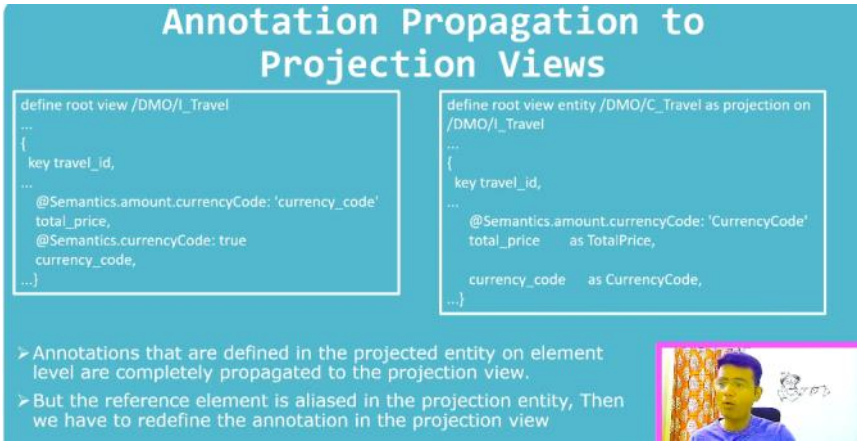
```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Travel Root Entity'
@Metadata.ignorePropagatedAnnotations: true
define root view entity YI_travel1_M
  as select from ytravel_m
  composition [1..*] of YI_booking1_m as _booking
  association [0..1] to /DMO/I_Agency   as _agency   on $projection.AgencyId = _agency.AgencyID
  association [0..1] to /DMO/I_Customer as _customer on $projection.CustomerId = _customer.CustomerID
  association [1..1] to I_Currency      as _currency on $projection.CurrencyCode = _currency.Currency
  association [0..1] to /DMO/I_Overall_Status_VH as _status on  $projection.OverallStatus=_status.OverallStatus
{
  key travel_id       as TravelId,
      agency_id       as AgencyId,
      customer_id     as CustomerId,
      begin_date      as BeginDate,
      end_date        as EndDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      booking_fee     as BookingFee,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      total_price     as TotalPrice,
      currency_code   as CurrencyCode,
      description     as Description,
      overall_status  as OverallStatus,
      created_by      as CreatedBy,
      created_at      as CreatedAt,
      last_changed_by as LastChangedBy,
      last_changed_at as LastChangedAt,
```

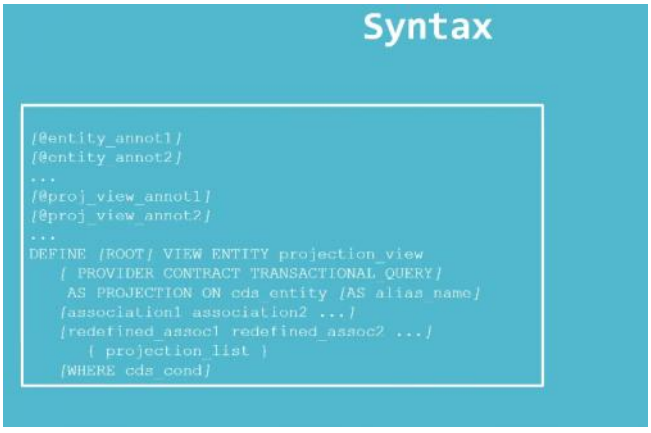
[TRL] YI_TRAVEL1_M | [TRL] YI_BOOKING1_M | [TRL] YI_BOOKSUPPL1_M

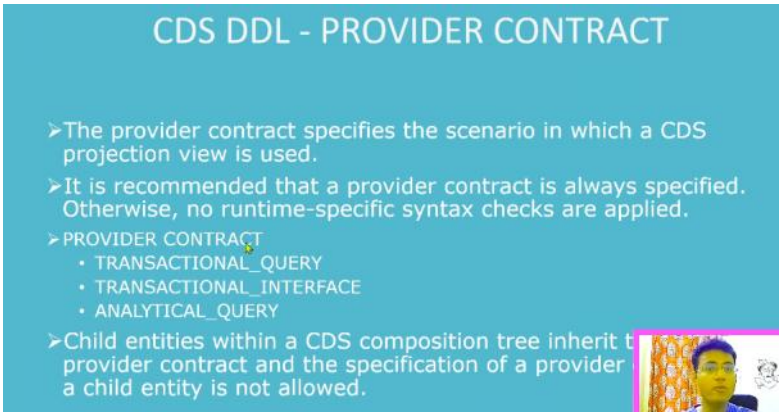
```
@AbapCatal Data Definition YI_TRAVEL1_M [TRL,100] - active - TRL_EN
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Booking Root Entity'
@Metadata.ignorePropagatedAnnotations: true
define view entity YI_booking1_m
  as select from ybooking_m
  association        to parent YI_travel1_M         as _travel     on  $projection.TravelId = _travel.TravelId
  composition [1..*] of Yi_bookSuppl1_m as _bookindsuppl
  association [1..1] to /DMO/I_Carrier           as _carrier    on  $projection.CarrierId = _carrier.AirlineID
  association [1..1] to /dmo/customer            as _customer   on  $projection.CustomerId = _customer.customer_id
  association [1..1] to /DMO/I_Connection        as _connection on  $projection.CarrierId    = _connection.AirlineID
                                                                and $projection.ConnectionId = _connection.ConnectionID
  association [1..1] to /DMO/I_Booking_Status_VH as _status     on  $projection.BookingStatus = _status.BookingStatus

{
  key travel_id       as TravelId,
  key booking_id      as BookingId,
      booking_date    as BookingDate,
      customer_id     as CustomerId,
      carrier_id      as CarrierId,
      connection_id   as ConnectionId,
      flight_date     as FlightDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      flight_price    as FlightPrice,
      currency_code   as CurrencyCode,
      booking_status  as BookingStatus,
      last_changed_at as LastChangedAt,
      _travel,
```

[TRL] YI_TRAVEL1_M | [TRL] YI_BOOKING1_M | [TRL] YI_BOOKSUPPL1_M

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Booking Suppl Interface'
@Metadata.ignorePropagatedAnnotations: true
define view entity Yi_bookSuppl1_m
  as select from ybooksuppl_m
  association        to parent YI_booking1_m  as _booking        on  $projection.BookingId = _booking.BookingId
                                                                 and $projection.TravelId  = _booking.TravelId
  association [1..1] to /dmo/supplement       as _supplement     on  $projection.SupplementId = _supplement.supplement_id
  association [1..*] to /DMO/I_SupplementText as _supplementtext on  $projection.SupplementId = _supplementtext.SupplementID
{
  key travel_id             as TravelId,
  key booking_id            as BookingId,
  key booking_supplement_id as BookingSupplementId,
      supplement_id         as SupplementId,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      price                 as Price,
      currency_code         as CurrencyCode,
      last_changed_at       as LastChangedAt,
      _booking,
      _supplement,
      _supplementtext
}
```



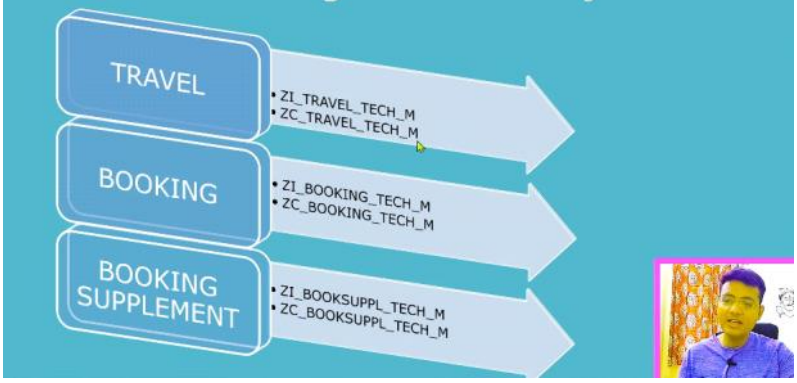




CDS DDL - CDS Projection View, Transactional Queries

- 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



[TRL] YC_TRAVEL1 | [TRL] YC_BOOKING1 | [TRL] YC_BOOKSUPPL1

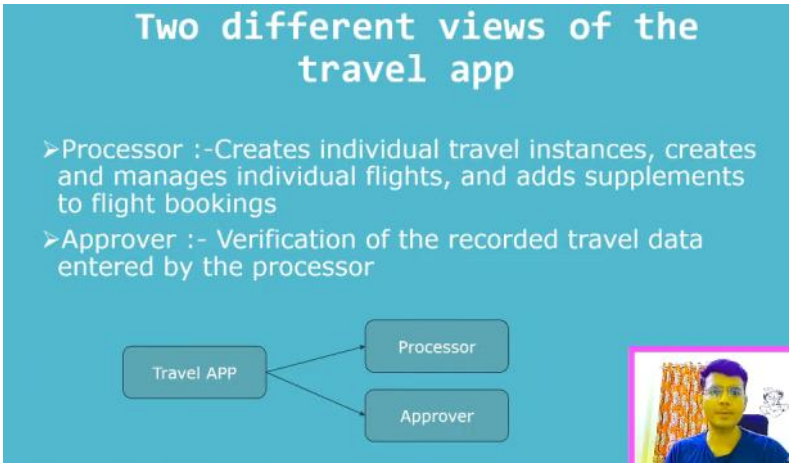
```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Travel Project View'
@Metadata.ignorePropagatedAnnotations: true
define root view entity YC_travel1
  provider contract transactional_query
  as projection on YI_travel1_M
{
  key TravelId,
      AgencyId,
      CustomerId,
      BeginDate,
      EndDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      BookingFee,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      TotalPrice,
      CurrencyCode,
      Description,
      OverallStatus,
      //      CreatedBy,
      //      CreatedAt,
      //      LastChangedBy,
      LastChangedAt,
      /* Associations */
      _agency,
      _booking: redirected to composition child YC_booking1,
      _currency,
      _customer,
      _status
}
```

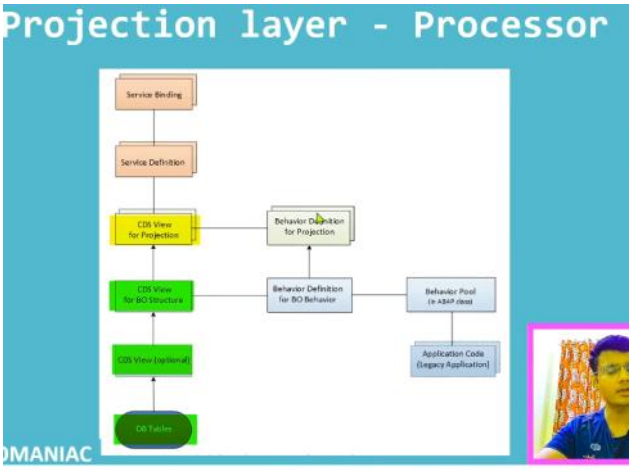
[TRL] YC_TRAVEL1 | [TRL] YC_BOOKING1 | [TRL] YC_BOOKSUPPL1

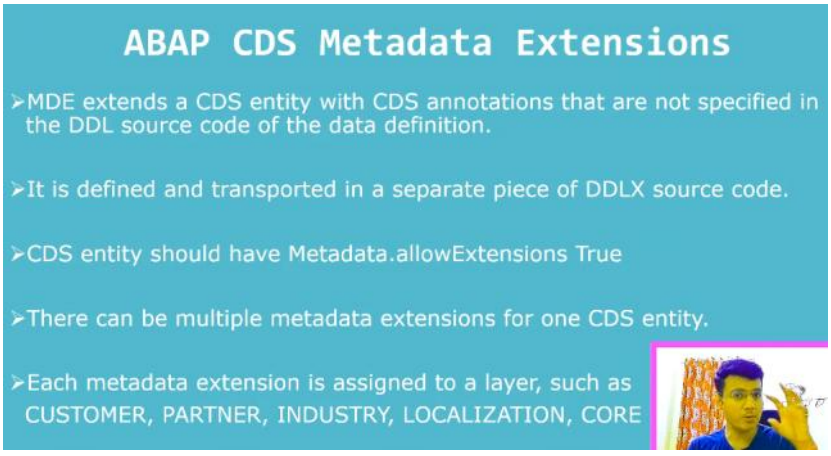
```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Booking Projection'
@Metadata.ignorePropagatedAnnotations: true
define view entity YC_booking1
  as projection on YI_booking1_m
{
  key TravelId,
  key BookingId,
      BookingDate,
      CustomerId,
      CarrierId,
      ConnectionId,
      FlightDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      FlightPrice,
      CurrencyCode,
      BookingStatus,
      LastChangedAt,
      /* Associations */
      _bookingsuppl : redirected to composition child yC_booksuppl1,
      _carrier,
      _connection,
      _customer,
      _status,
      _travel : redirected to parent YC_travel1
}

```

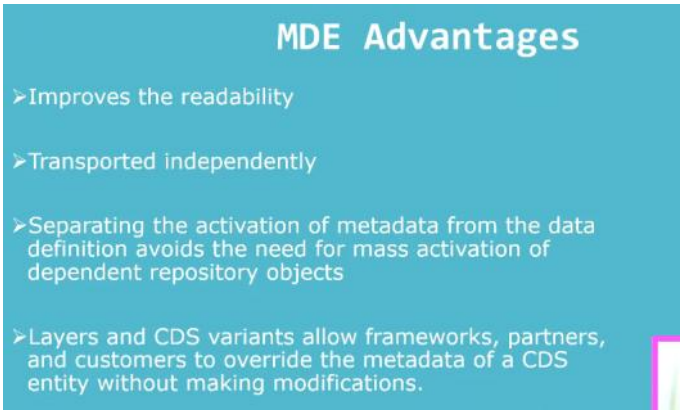
[TRL] YC_TRAVEL1 | [TRL] YC_BOOKING1 | [TRL] YC_BOOKSUPPL1

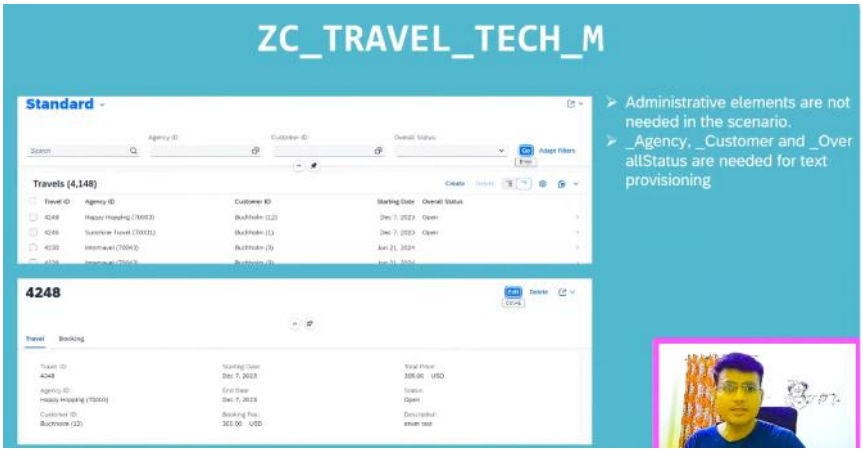
```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'booking suppl Projection'
@Metadata.ignorePropagatedAnnotations: true
define view entity yC_booksuppl1
  as projection on Yi_bookSuppl1_m
{
  key TravelId,
  key BookingId,
  key BookingSupplementId,
      SupplementId,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      Price,
      CurrencyCode,
      LastChangedAt,
      /* Associations */
      _booking : redirected to parent YC_booking1,
      _supplement,
      _supplementtext,
      _travel : redirected to YC_travel1
}

```











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
```

[TRL] YC_TRAVE... | [TRL] YC_BOOKI... | [TRL] YC_BOOKS... | [TRL] YI_TRAVE... | [TRL] YC_T

Data Definition YC_TRAVEL1 [TRL,100] - active - TRL_EN

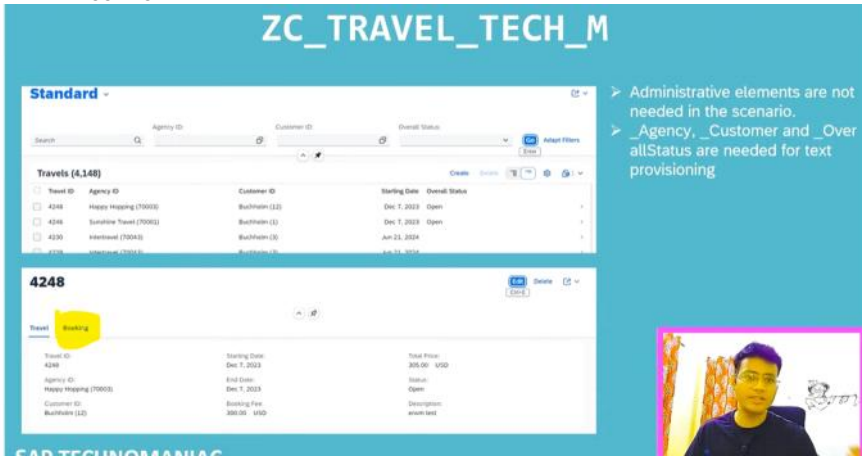
```
@Metadata.
define root view entity YC_travel1
  provider contract transactional_query
  as projection on YI_travel1_M
{
  key TravelId,
      @ObjectModel.text.element: [ 'AgencyName' ]
      AgencyId,
      _agency.Name       as AgencyName,
      @ObjectModel.text.element: [ 'CustomerName' ]
      CustomerId,
      _customer.LastName as CustomerName,
      BeginDate,
      EndDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      BookingFee,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      TotalPrice,
      CurrencyCode,
      Description,
      @ObjectModel.text.element: [ 'OverallStatusText' ]
      OverallStatus,
      _status._Text.Text as OverallStatusText : localized,
      //      CreatedBy,
      //      CreatedAt,
      //      LastChangedBy,
      LastChangedAt,
      /* Associations */
      _agency,
      _booking : redirected to composition child YC_booking1,
      _currency,
      _customer,
      _status
}
```

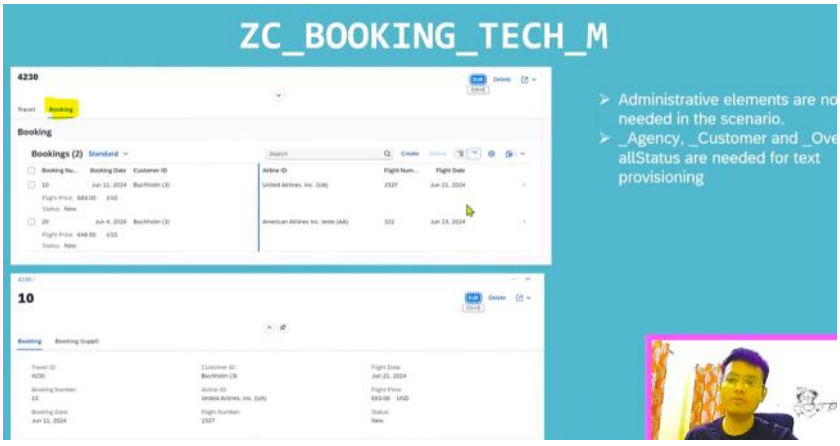
```
@Search.searchable: true
@UI.headerInfo: {
    typeName: 'Travel',
    typeNamePlural: 'Travels',

    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;
```

```
@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






```
*****************************************
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
*************************************
```

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Travel Project View'
@Metadata.ignorePropagatedAnnotations: true
@Metadata.allowExtensions: true
define root view entity YC_travel1
  provider contract transactional_query
  as projection on YI_travel1_M
{
  key TravelId,
      @ObjectModel.text.element: [ 'AgencyName' ]
      AgencyId,
      _agency.Name       as AgencyName,
      @ObjectModel.text.element: [ 'CustomerName' ]
      CustomerId,
      _customer.LastName as CustomerName,
      BeginDate,
      EndDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      BookingFee,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      TotalPrice,
      CurrencyCode,
      Description,
      @ObjectModel.text.element: [ 'OverallStatusText' ]
      OverallStatus,
      _status._Text.Text as OverallStatusText : localized,
      //      CreatedBy,
      //      CreatedAt,
      //      LastChangedBy,
      LastChangedAt,
      /* Associations */
```

[TRL] YC_TRAVEL1 | [TRL] YC_BOOKIN... | [TRL] YC_BOOKSU... | [TRL] YC_TRAVEL1 | [TRL] YC_E

```
//    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;


}
```

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Booking Projection'
@Metadata.ignorePropagatedAnnotations: true
@Metadata.allowExtensions: true
define view entity YC_booking1
  as projection on YI_booking1_m
{
  key TravelId,
  key BookingId,
      BookingDate,
      @ObjectModel.text.element: [ 'lastname' ]
      CustomerId,
      _customer.last_name as lastname,
      @ObjectModel.text.element: [ 'carriername' ]
      CarrierId,
      _carrier.Name       as carriername,
      ConnectionId,
      FlightDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      FlightPrice,
      CurrencyCode,
      @ObjectModel.text.element: [ 'bookindstatus' ]
      BookingStatus,
      _status._Text.Text  as bookindstatus : localized,
      LastChangedAt,
      /* Associations */
      _bookingsuppl : redirected to composition child yC_booksuppl1,
      _carrier,
      _connection,
      _customer,
      _status,
      _travel       : redirected to parent YC_travel1
}

```

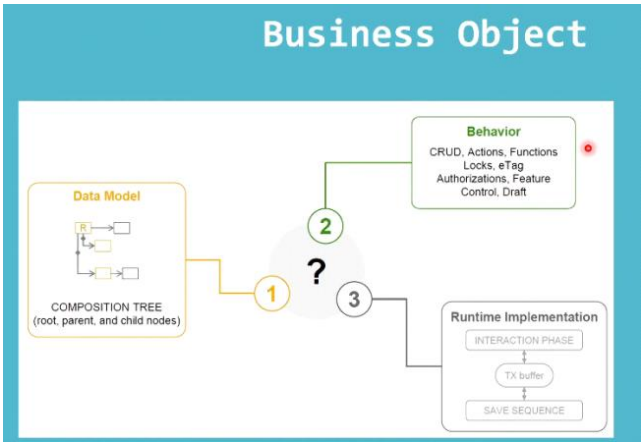
```
@Metadata.layer: #CORE
@Search.searchable: true
@UI:{ headerInfo: {
          typeName: 'Booking',
          typeNamePlural: 'Booking',
          title: {
              type: #STANDARD,
              label: 'Booking',
              value: 'BookingId'
          }
      }}
annotate entity YC_booking1 with
{
  @UI.facet: [{
      id: 'Booking',
      purpose: #STANDARD,
      position: 10 ,
      label: 'Booking',
      type:#IDENTIFICATION_REFERENCE
  },
  {
      id: 'Booksuppl',
      purpose: #STANDARD,
      position: 20 ,
      label: 'Booking Supplemnets',
      type: #LINEITEM_REFERENCE,
    targetElement: '_bookingsuppl'
  }]
  @Search.defaultSearchElement: true
  TravelId;
  @UI: { lineItem: [{ position: 10  }],
  identification: [{ position: 10 }] }
  @Search.defaultSearchElement: true
  BookingId;
```

```
  CarrierId;
  @UI:{lineItem: [{ position: 50  }],
  identification: [{ position: 50 }] }
  @Consumption.valueHelpDefinition: [{ entity: {
                                       name: '/DMO/I_Flight',
                                       element: 'ConnectionID'
                                     } ,
                                   additionalBinding: [{
                                     element: 'ConnectionID',
                                     localElement: 'ConnectionId' },
                                     {
                                     element: 'AirlineID',
                                     localElement: 'CarrierId' },
                                     {
                                     element: 'CurrencyCode',
                                     localElement: 'CurrencyCode' },
                                     {
                                     element: 'Price',
                                     localElement: 'FlightPrice' }]}]
  ConnectionId;
  @UI:{lineItem: [{ position: 60  }],
  identification: [{ position: 60 }] }
  @Consumption.valueHelpDefinition: [{ entity: {
                                       name: '/DMO/I_Flight',
                                       element: 'FlightDate'
                                     } ,
                                   additionalBinding: [{
                                     element: 'FlightDate',
                                     localElement: 'FlightDate' },
                                     {
                                     element: 'AirlineID',
                                     localElement: 'CarrierId' },
                                     {
                                     element: 'CurrencyCode',
                                     localElement: 'CurrencyCode' },
                                     {
```

```
                                     localElement: 'CurrencyCode' },
                                     {
                                     element: 'Price',
                                     localElement: 'FlightPrice' }]}]
  FlightDate;
  @UI:{lineItem: [{ position: 70  }],
  identification: [{ position: 70 }] }
  FlightPrice;
  @Consumption.valueHelpDefinition: [{ entity: {
                                         name: 'I_Currency',
                                         element: 'Currency'
                                       } }]
  CurrencyCode;
  @UI:{lineItem: [{ position: 80  }],
  identification: [{ position: 80 }],
  textArrangement: #TEXT_ONLY }
  @Consumption.valueHelpDefinition: [{ entity: {
                                       name: '/DMO/I_Booking_Status_VH',
                                       element: 'BookingStatus'
                                     } }]
  BookingStatus;
  @UI.hidden: true
  LastChangedAt;

}
```

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'booking suppl Projection'
@Metadata.ignorePropagatedAnnotations: true
@Metadata.allowExtensions: true
define view entity yC_booksuppl1
  as projection on Yi_bookSuppl1_m
{
  key TravelId,
  key BookingId,
  key BookingSupplementId,
  @ObjectModel.text.element: [ 'supplementdesc' ]
      SupplementId,
      _supplementtext.Description as supplementdesc :localized,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      Price,
      CurrencyCode,
      LastChangedAt,
      /* Associations */
      _booking : redirected to parent YC_booking1,
      _supplement,
      _supplementtext,
      _travel : redirected to YC_travel1
}
```

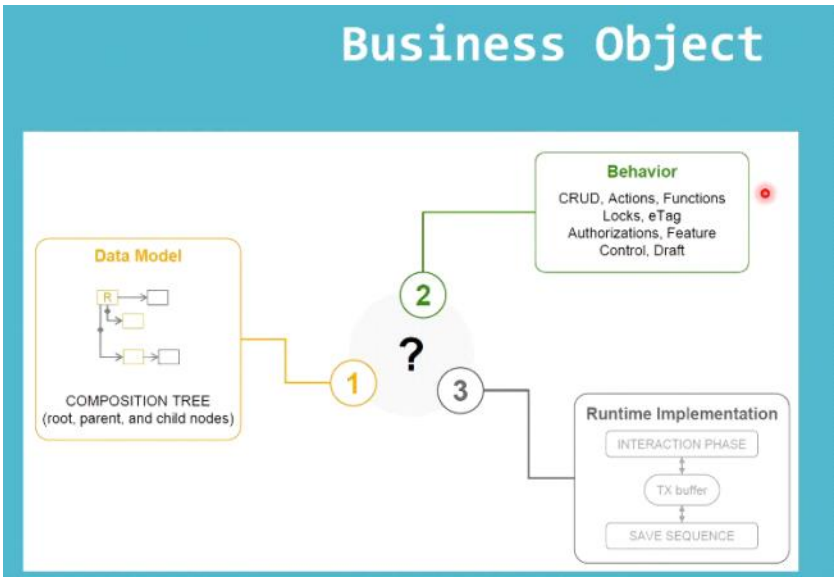
```
@Metadata.layer: #CORE
@Search.searchable: true
@UI:{ headerInfo: {
          typeName: 'BookingSuppl',
          typeNamePlural: 'BookingSuppls',
          title: {
              type: #STANDARD,
              label: 'Booking Supplement',
              value: 'BookingSupplementId'
          }
      }}

annotate entity yC_booksuppl1 with
{
  @UI.facet: [{
      id: 'BookingSuppl',
      purpose: #STANDARD,
      position: 10 ,
      label: 'Booking Supplements',
      type:#IDENTIFICATION_REFERENCE
  }]

  @Search.defaultSearchElement: true
  TravelId;
  @Search.defaultSearchElement: true
  BookingId;
  @UI: { lineItem: [{ position: 10 }],
  identification: [{ position: 10 }]
  }


  BookingSupplementId;
  @UI: { lineItem: [{ position: 20 }],
   identification: [{ position: 20 }]
   }
  @Consumption.valueHelpDefinition: [
  { entity: {
                name: '/dmo/i_supplement',
                element: 'SupplementID'
            },
   additionalBinding: [{ element: 'SupplementID',localElement: 'SupplementId'},
                         { element: 'Price',localElement: 'Price'},
                        { element: 'CurrencyCode',localElement: 'CurrencyCode'}]
             }]
  SupplementId;
  @UI: { lineItem: [{ position: 30 }],
   identification: [{ position: 30 }]
   }

  Price;
  @Consumption.valueHelpDefinition: [{ entity: {
                                          name: 'I_Currency',
                                          element: 'Currency'
                                      } }]
  CurrencyCode;
  @UI.hidden: true
  LastChangedAt;
```



Now Done With Data Model

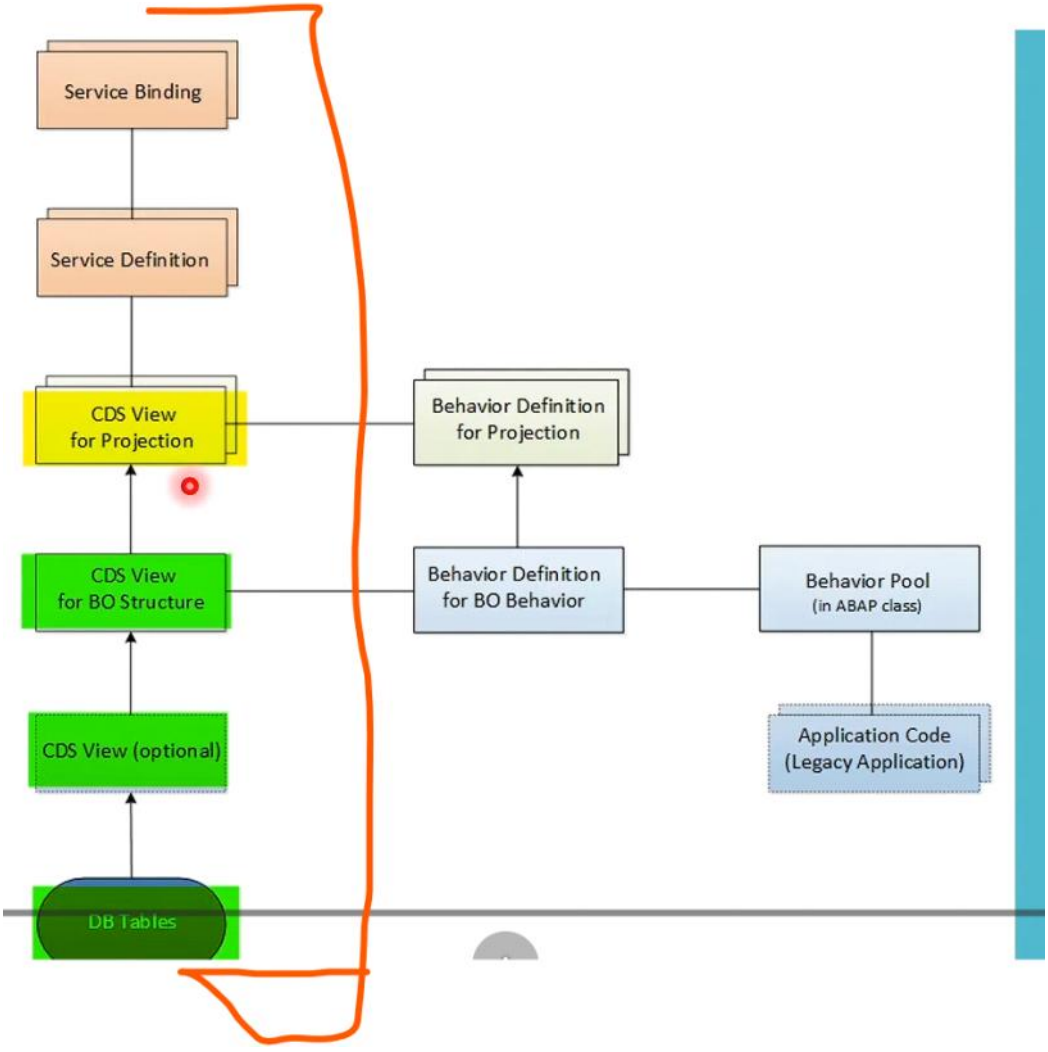
Behavior

26 November 2025 18:50



Data model Done Previously

I,e



Next Do With Behavior

BDL : Implementation type

```
[TRL] ZI_TRAVEL_TECH_M × [TRL] 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.

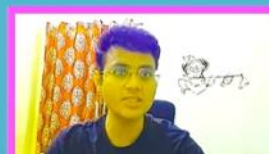
➢RAP best practices are enforced, like, for example, authorization implementation is mandatory

RAP - Entity Behavior Definition

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

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

```
define behavior for ZI_TRAVEL_TECH_M //alias <alias_name>
persistent table ztravel_tech_m
lock master
authorization master ( instance )
//etag master <field_name>
{
  create;
  update;
  delete;
  association _Booking { create; }
}

define behavior for ZI_BOOKING_TECH_M //alias <alias_name>
persistent table zbooking_tech_m
lock dependent by _Travel
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, ChildEntity1and 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)

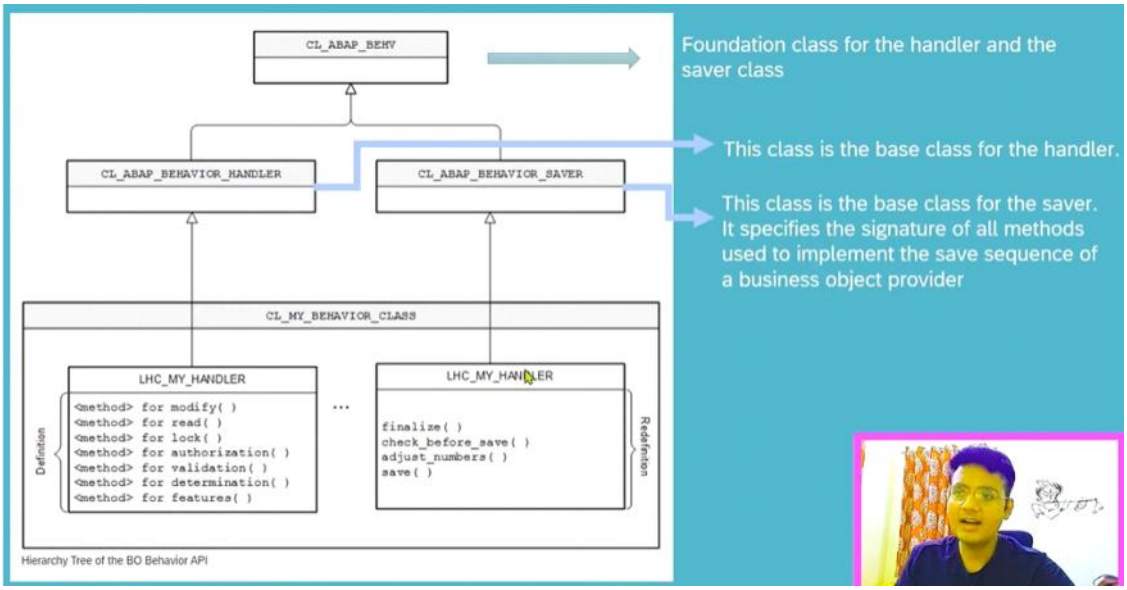
```
managed ;
strict ( 2 );

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; }
}
```

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.
```

SAP TECHNOMANIAC



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

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.

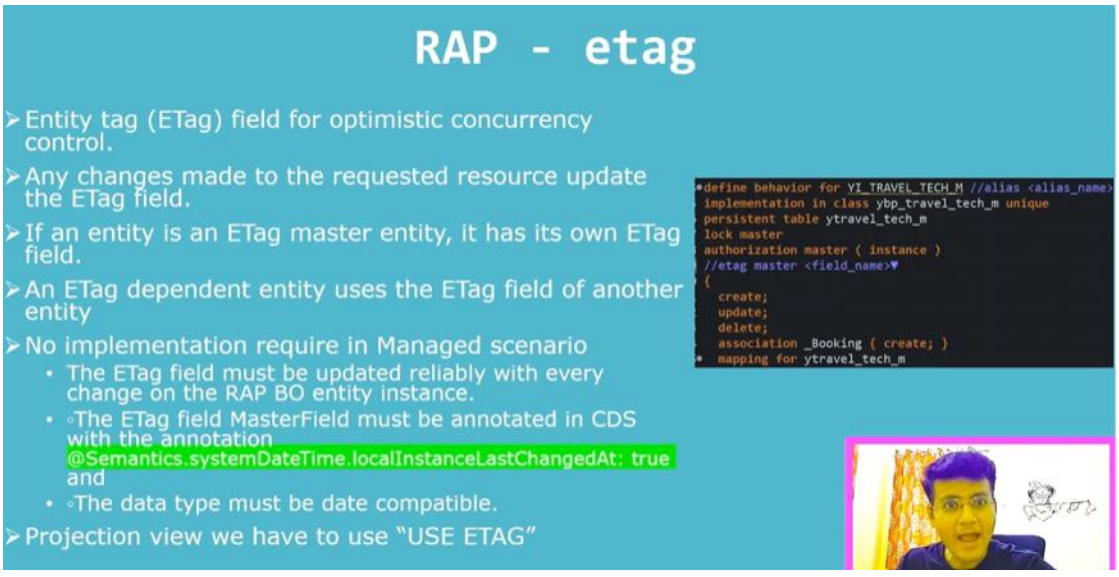
```
define behavior for YI_TRAVEL_TECH_M //alias <alias_name>
implementation in class ybp_travel_tech_m unique
persistent table ytravel_tech_m
lock master
authorization master ( instance )
//etag master <field_name>
{
  create;
  update;
  delete;
  association _Booking { create; }
}
```

Behavior Definition

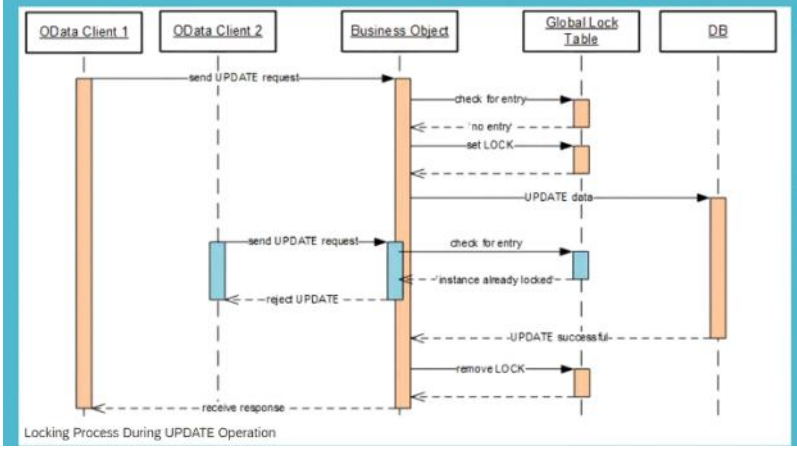
[TRL] YI_TRAVEL1_M | [TRL] YC_TRAVEL1 | [TRL] YUI_FLIGHT_V2

```
managed; //implementation in class zbp_i_travel1_m unique;
strict ( 2 );

define behavior for YI_travel1_M //alias <alias_name>
implementation in class zcl_bp_travel_managed unique
persistent table ytravel_m
lock master
authorization master ( instance )
//etag master <field_name>
{
  create ( authorization : global );
  update;
  delete;
  //  field ( readonly ) TravelId;
  association _booking { create; }
  mapping for ytravel_m
    {
      TravelId      = travel_id;
      AgencyId      = agency_id;
      CustomerId    = customer_id;
      BeginDate     = begin_date;
      EndDate       = end_date;
      BookingFee    = booking_fee;
      TotalPrice    = total_price;
      CurrencyCode  = currency_code;
      Description   = description;
      OverallStatus = overall_status;
      CreatedBy     = created_by;
      CreatedAt     = created_at;
      LastChangedBy = last_changed_by;
      LastChangedAt = last_changed_at;
    }
}

```

```
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  }
```



Pessimistic Concurrency Control (Locking)



Locking Process During UPDATE Operation

- Pessimistic concurrency control prevents simultaneous modification access to data on the database by more than one user.
- The data set that is being modified by one user cannot be changed by another user at the same time.

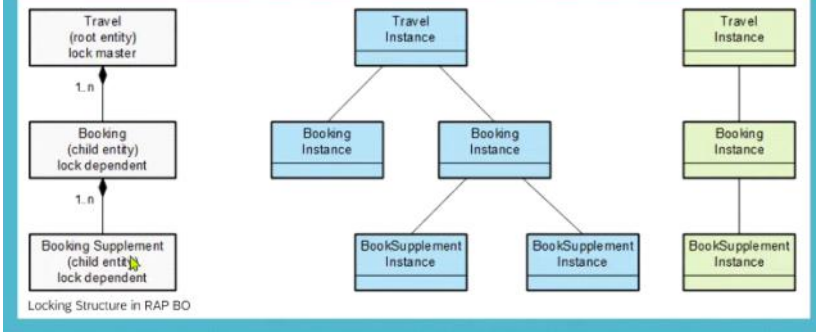
RAP – Locking Managed

- 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 { }.

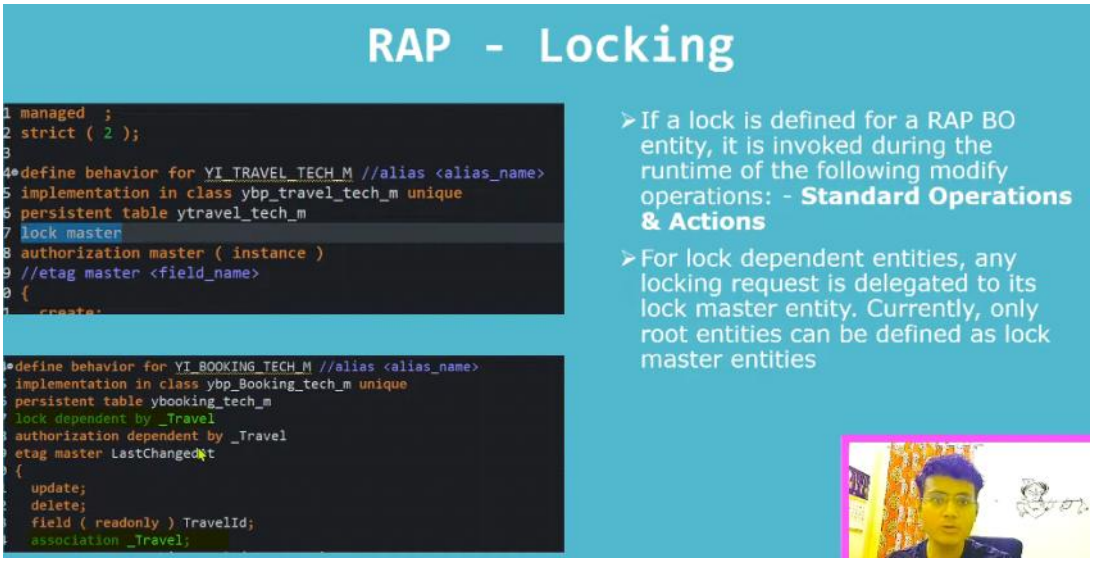
```
define behavior for yI_BOOKSUPPL_TECH_M //alias <alias_name>
implementation in class ybp_booksuppl_tech_m unique
persistent table ybooksupp_tech_m
lock dependent by _Travel
authorization dependent by _Travel
//etag master <field_name>
{
  update;
  delete;
  field ( readonly ) TravelId, BookingId;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

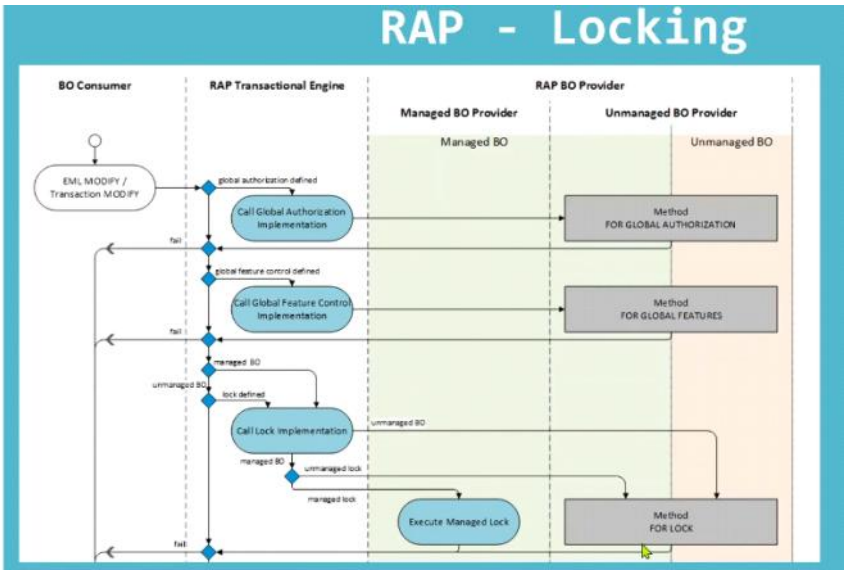
```
projection;
strict ( 2 );

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



Locking Structure in RAP BO





AT Interface Level

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Travel Root Entity'
@Metadata.ignorePropagatedAnnotations: true
define root view entity YI_travel1_M
  as select from ytravel_m
  composition [1..*] of YI_booking1_m as _booking
  association [0..1] to /DMO/I_Agency   as _agency   on $projection.AgencyId = _agency.AgencyID
  association [0..1] to /DMO/I_Customer as _customer on $projection.CustomerId = _customer.CustomerID
  association [1..1] to I_Currency      as _currency on $projection.CurrencyCode = _currency.Currency
  association [1..1] to /DMO/I_Overall_Status_VH as _status on  $projection.OverallStatus=_status.OverallStatus
{
  key travel_id       as TravelId,
      agency_id       as AgencyId,
      customer_id     as CustomerId,
      begin_date      as BeginDate,
      end_date        as EndDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      booking_fee     as BookingFee,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      total_price     as TotalPrice,
      currency_code   as CurrencyCode,
      description     as Description,
      overall_status  as OverallStatus,
      created_by      as CreatedBy,
      created_at      as CreatedAt,
      last_changed_by as LastChangedBy,
      @Semantics.systemDateTime.localInstanceLastChangedAt: true
      last_changed_at as LastChangedAt,
      _booking,
      _agency,
      _customer,
      _currency,
      _status
}
```

AT Behavior level of Root/interface level

```
managed; //implementation in class zbp_i_travel1_m unique;
strict ( 2 );

define behavior for YI_travel1_M //alias <alias_name>
implementation in class zcl_bp_travel_managed unique
persistent table ytravel_m
lock master
authorization master ( instance )
etag master LastChangedAt
{
  create ( authorization : global );
  update;
  delete;
  //  field ( readonly ) TravelId;
  association _booking { create; }
  mapping for ytravel_m
    {
      TravelId      = travel_id;
```

```
define behavior for YI_booking1_m //alias <alias_name>
implementation in class zcl_bp_Booking_managed unique
persistent table ybooking_m
lock dependent by _travel
authorization dependent by _travel
etag master LastChangedAt
//etag dependent by _travel
{
  update;
  delete;
  //  field ( readonly ) TravelId, BookingId;
  field ( readonly ) TravelId;
  association _travel;
  association  bookingsuppl { create; }
```

AT Projection Behavior definition lebel

```
projection;
strict ( 2 );

define behavior for YC_travel1 //alias <alias_name>
use etag
{
  use create;
  use update;
  use delete;

  use association _booking { create; }
}

define behavior for YC_booking1 //alias <alias_name>
use etag
{
  use update;
  use delete;

  use association _travel;
  use association _bookingsuppl { create; }
}

define behavior for yC_booksuppl1 //alias <alias_name>
use etag
{
  use update;
  use delete;

  use association _travel;
  use association _booking;
}
```

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

```
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.
  LOOP AT lt_entities INTO ls_entities.
    lv_curr_num = lv_curr_num + 1.
    APPEND VALUE #( %cid = ls_entities-%cid TravelId = lv_curr_num ) TO mapped-yi_travel1_m.
  ENDLOOP.

ENDMETHOD.
```

FOR Booking Item

```
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.
```

FOR Booking Suppl

```
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(booking_supplements).


  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 booking_supplements 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 booing 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_booksuppl1_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.
```

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.
```

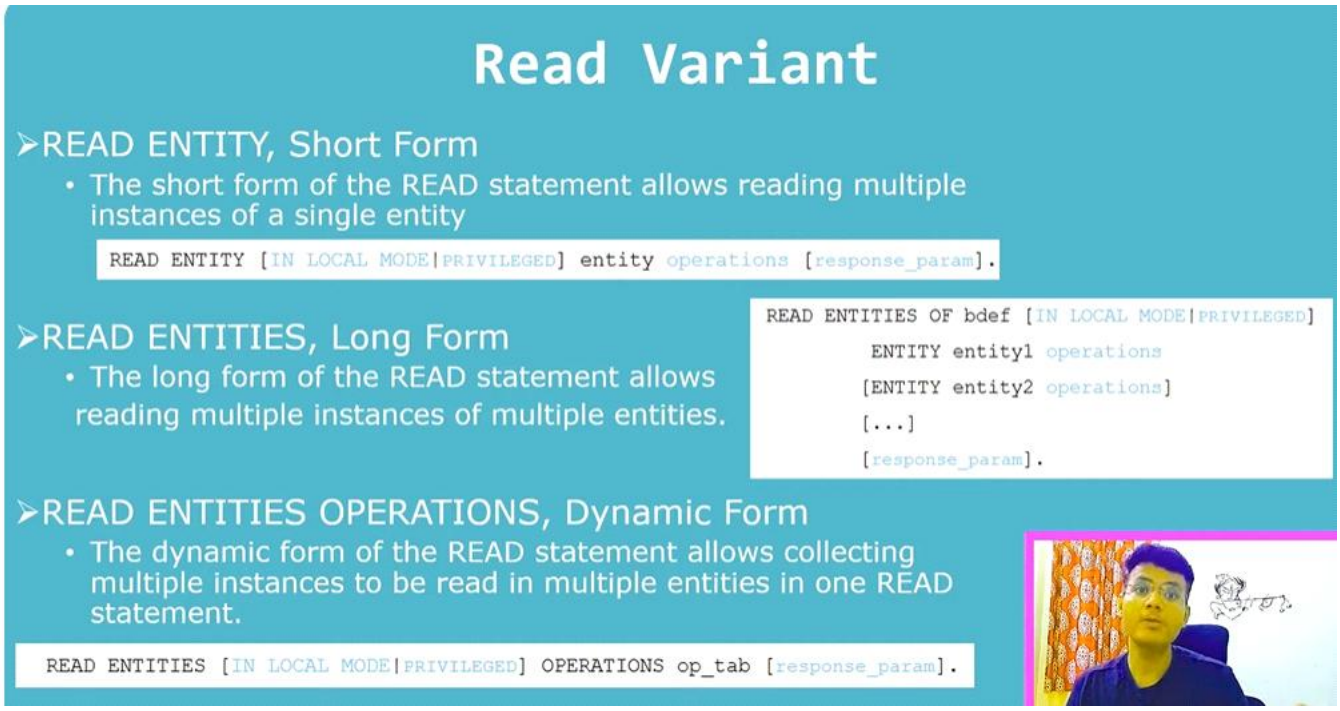
Booking supplement id

```
 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(booking_supplements).

    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 booking_supplements 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 booing 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_booksuppl1_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




```
* Sort Form Read

    READ ENTITY yi_travel1_m
    FROM VALUE #( ( %key-TravelId = '00000004'
                    %control = VALUE #( AgencyId = if_abap_behv=>mk-on
                                        CustomerId = if_abap_behv=>mk-on
                                        BeginDate = if_abap_behv=>mk-on ) ) )
      RESULT DATA(lt_data)
      FAILED DATA(lt_fail)  .

    IF lt_fail IS NOT INITIAL.
      out->write( 'read File' ).
    ELSE.
      out->write( lt_data ).
    ENDIF.
```

```
    READ ENTITY yi_travel1_m
    FIELDS ( AgencyId CustomerId BeginDate EndDate BookingFee TotalPrice )
    WITH VALUE #( ( %key-TravelId = '00000004' )
                  ( %key-TravelId = '00000001' ) )
      RESULT DATA(lt_data)
      FAILED DATA(lt_fail)  .
```

```
    READ ENTITY yi_travel1_m
    ALL FIELDS
    WITH VALUE #( ( %key-TravelId = '00000004' )
                  ( %key-TravelId = '00000001' ) )
      RESULT DATA(lt_data)
      FAILED DATA(lt_fail)  .
```

```
    READ ENTITY yi_travel1_m
    by \_booking
    ALL FIELDS
    WITH VALUE #( ( %key-TravelId = '00000004' )
                  ( %key-TravelId = '00000001' ) )
      RESULT DATA(lt_data)
      FAILED DATA(lt_fail)  .

    IF lt_fail IS NOT INITIAL.
      out->write( 'read File' ).
    ELSE.
      out->write( lt_data ).
    ENDIF.
```

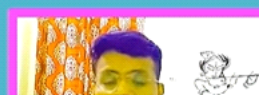
```
* Longer Form

    READ ENTITIES OF yi_travel1_m

    ENTITY YI_travel1_M
       ALL FIELDS WITH VALUE #( ( %key-TravelId = '00000004' )
                        ( %key-TravelId = '00000001' ) )
       RESULT DATA(lt_travel)

    ENTITY YI_booking1_m
       ALL FIELDS WITH VALUE #( ( %key-TravelId = '00000001'
                                  %key-BookingId = '0001' ) )
       RESULT DATA(lt_booking)

        FAILED DATA(lt_fail)  .

    IF lt_fail IS NOT INITIAL.
      out->write( 'read File' ).
    ELSE.
      out->write( lt_travel ).
      out->write( lt_booking ).
    ENDIF.
```

```

    DATA: it_optab         TYPE abp_behv_retrievals_tab,
          it_travel        TYPE TABLE FOR READ IMPORT yi_travel_tech_m,
          it_travel_result TYPE TABLE FOR READ RESULT  yi_travel_tech_m.

    it_travel = VALUE #( ( %key-TravelId = '0000004172'
                           %control = VALUE #( AgencyId    = if_abap_behv=>mk-on
                                               customerid  = if_abap_behv=>mk-on
                                               begindate   = if_abap_behv=>mk-on
                                             ) ) ).

    it_optab = VALUE #( ( op = if_abap_behv=>op-r
                          entity_name = 'YI_TRAVEL_TECH_M'
                          instances = REF #( it_travel )
                          results   = REF #( it_travel_result )   ) ).

    READ ENTITIES
       OPERATIONS it_optab
         FAILED DATA(lt_failed_dy).

    IF lt_failed_dy IS NOT INITIAL.
      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.
```

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



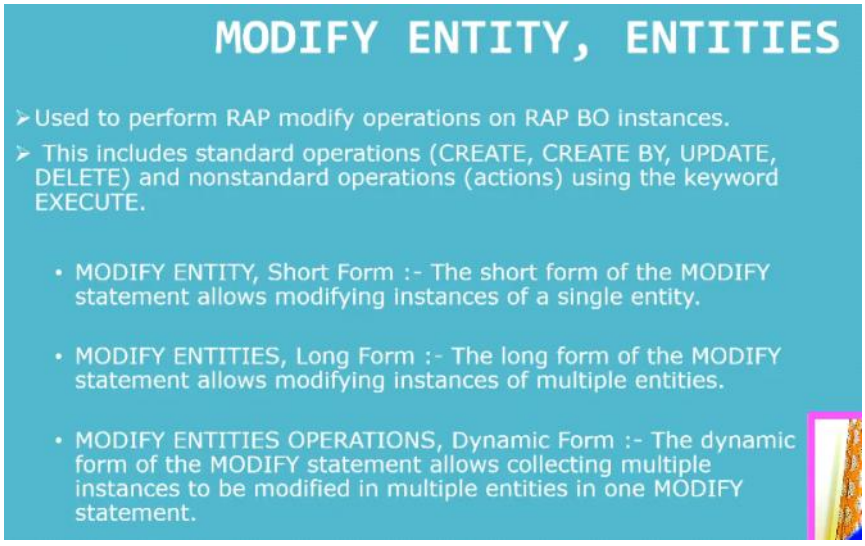
Code

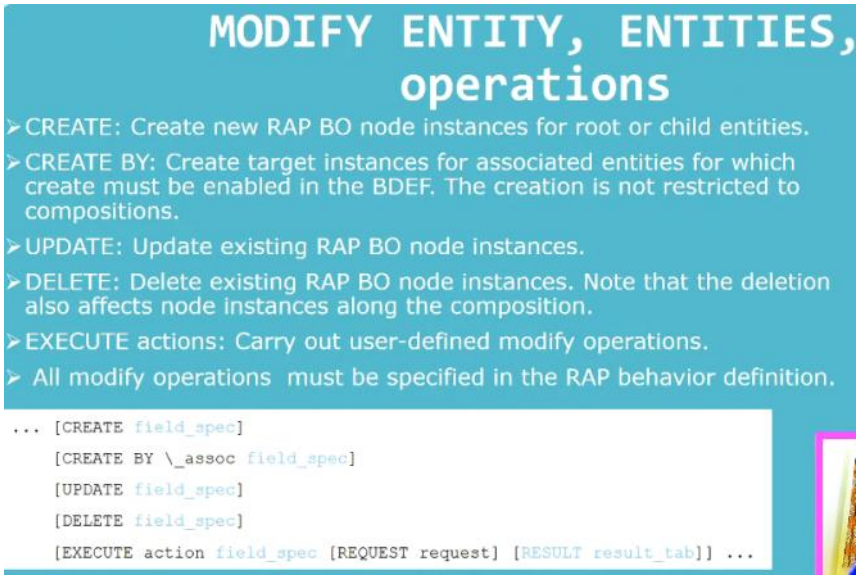
03 December 2025 11:10

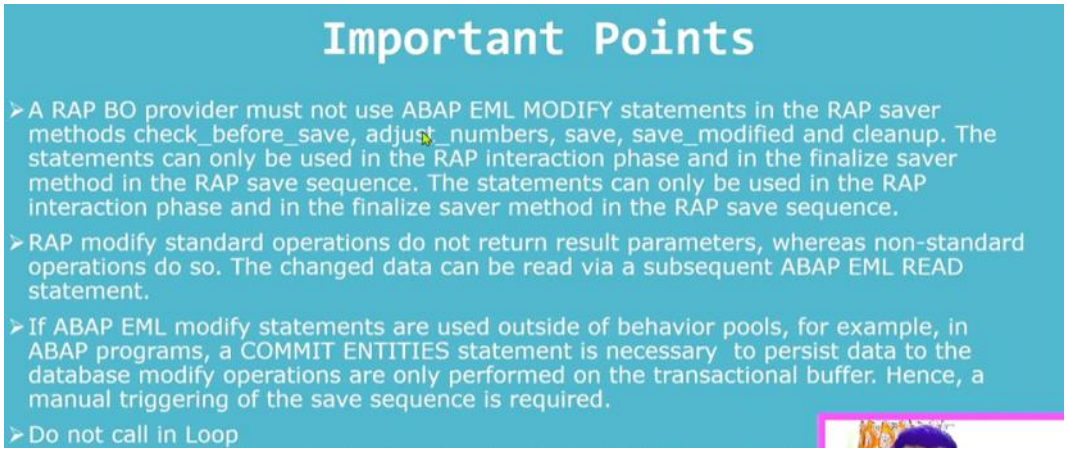
```
READ ENTITIES OF yi_travel1_m IN LOCAL MODE
ENTITY yi_travel1_m
ALL FIELDS WITH CORRESPONDING #( keys )
RESULT DATA(lt_result).
```

Modify

01 December 2025 09:06







CREATE

```
CLASS ycl_modify_practices DEFINITION PUBLIC .

  PUBLIC SECTION.
    INTERFACES : if_oo_adt_classrun.

ENDCLASS.

CLASS ycl_modify_practices IMPLEMENTATION.
  METHOD if_oo_adt_classrun~main.

    MODIFY ENTITY YI_travel1_M

    CREATE FROM VALUE #( ( %cid = 'cid2'
                           %data-BeginDate = '20251201'
                           %control-BeginDate = if_abap_behv=>mk-on ) )

    FAILED FINAL(it_failed)
    MAPPED FINAL(it_mapped)
    REPORTED FINAL(it_reported).

    COMMIT ENTITIES.

    IF it_failed IS NOT INITIAL.
      out->write( it_failed ).
    ELSE.
      out->write( 'created.' ).
    ENDIF.

  ENDMETHOD.
ENDCLASS.
```

```
    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.
```

```
* Auto Fill CID

    MODIFY ENTITY YI_travel1_M
    CREATE AUTO FILL CID WITH VALUE #( ( %data-BeginDate = '20251212' %control-BeginDate = if_abap_behv=>mk-on ) )
        FAILED DATA(lt_fail)
        MAPPED DATA(lt_data)
        REPORTED FINAL(lt_re).

    COMMIT ENTITIES.
```

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 deleted

```
   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).

    COMMIT ENTITIES.
```

UPDATE

```
* 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.


    MODIFY ENTITY YI_travel1_M
    UPDATE SET FIELDS WITH VALUE #( ( %key-TravelId = '00000004'
                                      BeginDate = '20251212' ) )
         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).
```

NOT Advice

```
    MODIFY ENTITIES OF YI_travel1_M
    ENTITY YI_travel1_M
    UPDATE SET FIELDS WITH VALUE #( ( %key-TravelId = '00000004' BeginDate = '20251201' ) )
          FAILED DATA(lt_fail)
          MAPPED DATA(lt_data)
          REPORTED FINAL(lt_re).

    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

```
managed;//implementation in class zbp_i_std_table unique;
strict ( 2 );

//with draft;

define behavior for ZI_std_table alias Student
implementation in class zbp_std_table unique

//draft table zdraft_std_table

persistent table ystd_table
lock master //total etag Lastchangedat
authorization master ( instance )
//etag master Lastchangedat


{
  create ( authorization : global );
  update;
  delete;
  field ( numbering : managed, readonly ) Id;
  association _academics { create; }

  //  action ( features : instance ) SetAmditted result [1] $self;

  action ( features : instance ) Approve result [1] $self;

```

In Behavior Definition of Projection Root

```
projection;
strict ( 2 );

define behavior for ZP_std_tab //alias <alias_name>
{
  use create;
  use update;
  use delete;

  use association _academics { create; }

  use action Approve;

}

define behavior for ZP_res_tab //alias <alias_name>
{
  use update;
  use delete;

  use association _Student;
}
```

In class

```
CLASS lhc_Student DEFINITION INHERITING FROM cl_abap_behavior_handler.
  PRIVATE SECTION.

    METHODS get_instance_features FOR INSTANCE FEATURES
      IMPORTING keys REQUEST requested_features FOR Student RESULT result.

    METHODS get_instance_authorizations FOR INSTANCE AUTHORIZATION
      IMPORTING keys REQUEST requested_authorizations FOR Student RESULT result.

    METHODS get_global_authorizations FOR GLOBAL AUTHORIZATION
      IMPORTING REQUEST requested_authorizations FOR Student RESULT result.

    METHODS Approve FOR MODIFY
      IMPORTING keys FOR ACTION Student~Approve RESULT result.

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

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

ENDCLASS.
```

```
METHOD Approve.
  MODIFY ENTITIES OF ZI_std_table IN LOCAL MODE
ENTITY Student
 UPDATE
 FIELDS ( Status )
 WITH VALUE #( FOR key IN keys ( %tky = key-%tky Status = abap_true ) )
 FAILED failed
 REPORTED reported.


  READ ENTITIES OF ZI_std_table IN LOCAL MODE
  ENTITY Student
  ALL FIELDS WITH CORRESPONDING #( keys )
  RESULT DATA(studentdata).

  result = VALUE #( FOR studentrec IN studentdata ( %tky = studentrec-%tky %param = studentrec ) ).
ENDMETHOD.
```

```
METHOD get_instance_features.
  READ ENTITIES OF ZI_std_table IN LOCAL MODE
  ENTITY Student
  FIELDS ( Status ) WITH CORRESPONDING #( keys )
  RESULT DATA(studentadmitted)
  FAILED failed.

  result = VALUE #( FOR stud IN studentadmitted
                    LET stausval = COND #( WHEN stud-Status = abap_true THEN if_abap_behv=>fc-o-disabled
                                           ELSE if_abap_behv=>fc-o-enabled )

                                           IN ( %tky = stud-%tky %action-Approve = stausval )
                                           ).
ENDMETHOD.
```

In Metada Extension

```
@UI: { lineItem: [{ position: 70, label: 'Status' },
                  { type: #FOR_ACTION,dataAction: 'Approve',label: 'APPROVE',importance: #HIGH }
                  ],
       identification: [{ position: 70,label: 'Status' }
                        ] }

Status;
```

Validations

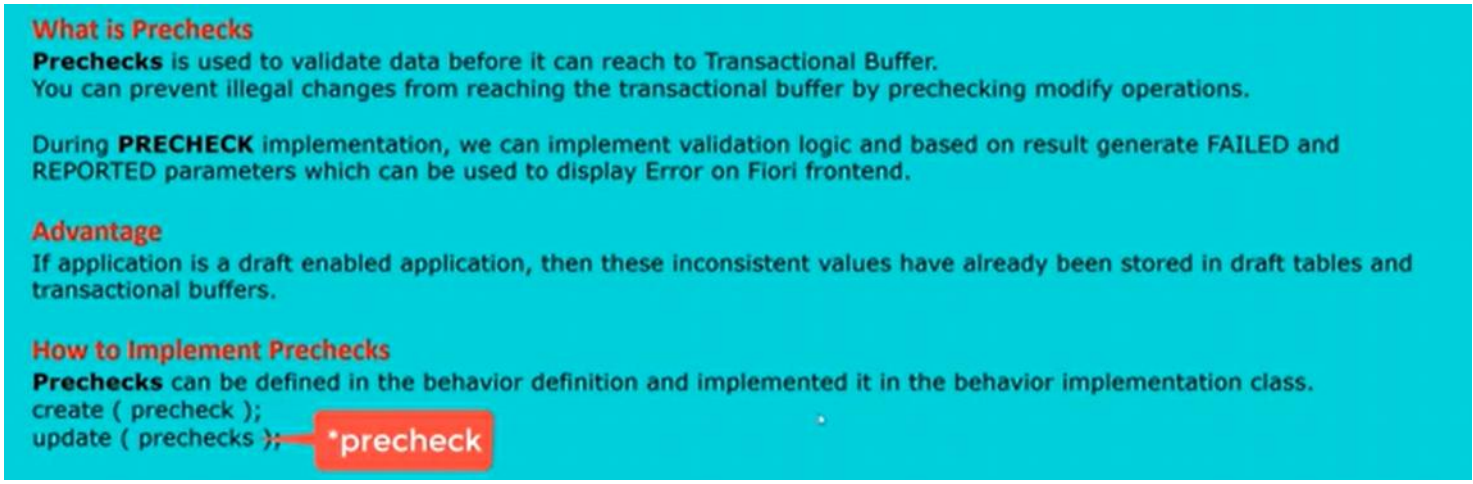
30 November 2025 13:27

```
managed;//implementation in class zbp_i_std_table unique;
strict ( 2 );

//with draft;

define behavior for ZI_std_table alias Student
implementation in class zbp_std_table unique

//draft table zdraft_std_table

persistent table ystd_table
lock master //total etag Lastchangedat
authorization master ( instance )
//etag master Lastchangedat


{
  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; }
```

In Class

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

```
  METHOD ValidateAge.
    READ ENTITIES OF ZI_std_table IN LOCAL MODE
   ENTITY Student
   FIELDS ( age )  WITH CORRESPONDING #( keys )
   RESULT DATA(studentsAge).

     LOOP AT studentsage INTO DATA(studentage).
       IF studentage-Age < 21.
         APPEND VALUE #( %tky = studentage-%tky ) TO failed-student.
         APPEND VALUE #( %tky = keys[ 1 ]-%tky
                         %msg = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                                       text = 'Age can not be less than 21' ) ) TO reported-student.
       ENDIF.
     ENDLOOP.
   ENDMETHOD.
```



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.
```

```
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
```

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

```
@Metadata.ignorePropagatedAnnotations: true
@Metadata.allowExtensions: true
define root view entity ZP_std_tab
  provider contract transactional_query
  as projection on ZI_std_table
{
          @EndUserText.label: 'Student Id'
  key     Id,
          @EndUserText.label: 'First Name'
          Firstname,
          @EndUserText.label: 'Last name'
          Lastname,
          @EndUserText.label: 'Age'
          Age,
          @EndUserText.label: 'Course'
          Course,
          @EndUserText.label: 'Course Duration'
          Courseduration,
          @EndUserText.label: 'Status'
          Status,
          @EndUserText.label: 'Gender'
          Gender,
          @EndUserText.label: 'Date Of Birth'
          Dob,
          @EndUserText.label: 'Last Changed'
          Lastchangedat,
          @EndUserText.label: 'Local Last Changed'
          Locallastchangedat,
          _academics : redirected to composition child ZP_res_tab,

          @ObjectModel.virtualElementCalculatedBy: 'ABAP:ZCL_VE_CALCULATIONS'
          @EndUserText.label: 'Course Fee'
  virtual CourseFee : abap.int4

}
```

Provide Class Name And Create

```
CLASS zcl_ve_calculations DEFINITION PUBLIC .

  PUBLIC SECTION.
    INTERFACES : if_sadl_exit_calc_element_read.
ENDCLASS.

CLASS zcl_ve_calculations IMPLEMENTATION.
  METHOD if_sadl_exit_calc_element_read~get_calculation_info.

  ENDMETHOD.
  METHOD if_sadl_exit_calc_element_read~calculate.

    DATA : lt_fee TYPE TABLE OF ZP_std_tab.
    lt_fee = CORRESPONDING #( it_original_data ).
    LOOP AT lt_fee ASSIGNING FIELD-SYMBOL(<fs_fee>).
      IF <fs_fee>-Course = 'Computers'.
        <fs_fee>-CourseFee = 150000.
      ELSEIF <fs_fee>-Course = 'Electronics'.
        <fs_fee>-CourseFee = 125000.
      ELSE.
```

```
18        ELSEIF <fs_fee>-Course = 'Electronics'.
19          <fs_fee>-CourseFee = 125000.
20        ELSE.
21          <fs_fee>-CourseFee = 7500.
22        ENDIF.
23      ENDLOOP.
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]
                  [authorisationnone]
                  [authorisationupdate]
                  [authorisationglobal]
                  [authorisationinstance]
                  [lockmone]
                   )]
                   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]
                  [authorisationnone]
                  [authorisationupdate]
                  [authorisationglobal]
                  [authorisationinstance]
                  [lockmone]
                   )]
                   ActionName [external 'ExternalName'
                  [inputParameter]
                   [cardinality]
```

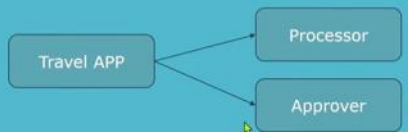
RAP - save action

- 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) actio
                  [(
                  [features: global]
                  [precheck]
                  [authorisationnone]
                  [authorisationglobal]
                  [authorisationinstance]
                   )]
                   ActionName [external 'ExternalName']
                  [inputParameter]
                  [outputParameter];
```

Two different views of the travel managed app

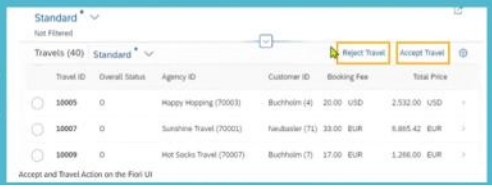
- Processor :-Creates individual travel instances, creates and manages individual flights, and adds supplements to flight bookings
- Approver :- Verification of the recorded travel data entered by the processor



Accept and Reject Travel Action

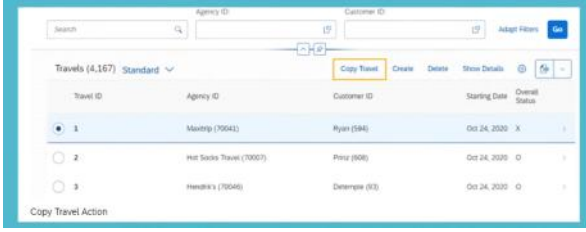
- The action acceptTravel sets the overall status of the travel entity to accepted (A). The action rejectTravel sets the overall status of the travel entity to rejected (X)
- These actions are instance actions as there is one specific instance of the travel entity for which the status is changed. (Non-Factory)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

Approver



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



Processor

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 suppliment)
- 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 throughh booking end copying other data excep 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: '' }]

```
managed; //implementation in class zbp_i_travel1_m unique;
strict ( 2 );

define behavior for YI_travel1_M //alias <alias_name>
implementation in class zcl_bp_travel_managed unique
persistent table ytravel_m
lock master
authorization master ( instance )
etag master LastChangedAt
early numbering
{

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

  action ( features : instance ) acceptTravel result [1] $self;
  action ( features : instance ) rejectTravel result [1] $self;

  factory action copytravel [1];

  internal action recalcTotPrice;

  create ( authorization : global );
  update;
  delete;

  association _booking { create ( features : instance ); }
  mapping for ytravel_m
```

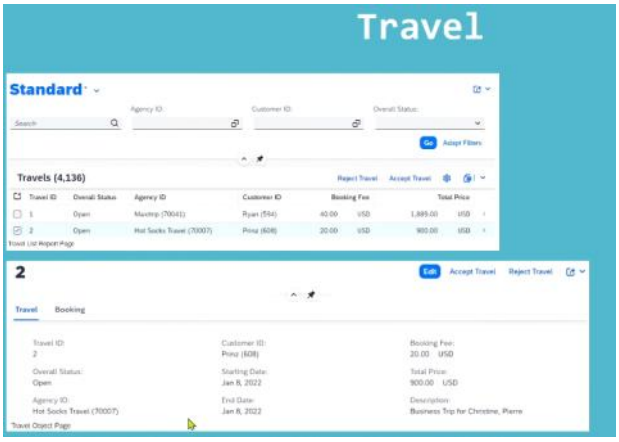
```
define behavior for YI_booking1_m //alias <alias_name>
implementation in class zcl_bp_Booking_managed unique
persistent table ybooking_m
lock dependent by _travel
authorization dependent by _travel
etag master LastChangedAt
//etag dependent by _travel
early numbering
{
  update;
  delete;
  field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

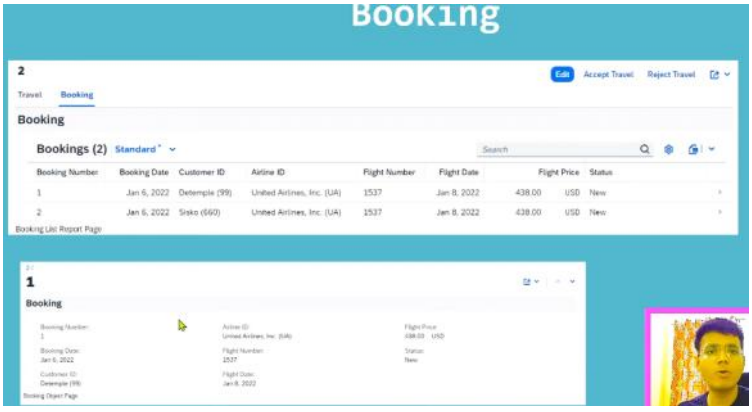
```
define behavior for Yi_bookSuppl1_m //alias <alias_name>
implementation in class zcl_bp_BookSuppl_managed unique
persistent table ybooksuppl_m
lock dependent by _travel
authorization dependent by _travel
etag master LastChangedAt
//etag dependent by _travel
early numbering
{
  update;
  delete;
  field ( readonly ) TravelId, BookingId, BookingSupplementId, LastChangedAt;
  field ( mandatory ) Price, SupplementId;
  association _travel;
  association _booking;
  mapping for ybooksuppl_m
    {
      TravelId            = travel_id;
```

```
  METHOD get_instance_features.

read entities of yi_travel1_m in local mode
entity YI_travel1_M
fields ( TravelId OverallStatus )
with corresponding #( keys )
RESULT DATA(lt_travel).

    result = VALUE #( FOR ls_travel IN lt_travel
                           ( %tky = ls_travel-%tky
                             %features-%action-acceptTravel = COND #( WHEN ls_travel-OverallStatus = 'A'
                                                                      THEN if_abap_behv=>fc-o-disabled
                                                                    ELSE if_abap_behv=>fc-o-enabled )
                             %features-%action-rejectTravel = COND #( WHEN ls_travel-OverallStatus = 'X'
                                                                      THEN if_abap_behv=>fc-o-disabled
                                                                   ELSE if_abap_behv=>fc-o-enabled )
                             %features-%assoc-_booking = COND #( WHEN ls_travel-OverallStatus = 'X'
                                                                      THEN if_abap_behv=>fc-o-disabled
                                                                   ELSE if_abap_behv=>fc-o-enabled )

                                                                    ) ).


  ENDMETHOD.
```

Approval(actions)

02 December 2025 12:59






```
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
129                                    OverallStatus = 'A' ) ).
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
137                                                %param = ls_res ) ).
138
139
140
141  ENDMETHOD.
```

```
METHOD rejectTravel.
  MODIFY ENTITIES OF yi_travel1_m IN LOCAL MODE
  ENTITY yi_travel1_m
  UPDATE FIELDS ( OverallStatus )
  WITH VALUE #( FOR key IN keys ( %tky = key-%tky
                                  OverallStatus = 'X' ) ).

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

  result = VALUE #( FOR ls_res IN lt_result ( %tky = ls_res-%tky
                                              %param = ls_res ) ).
ENDMETHOD.
```

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

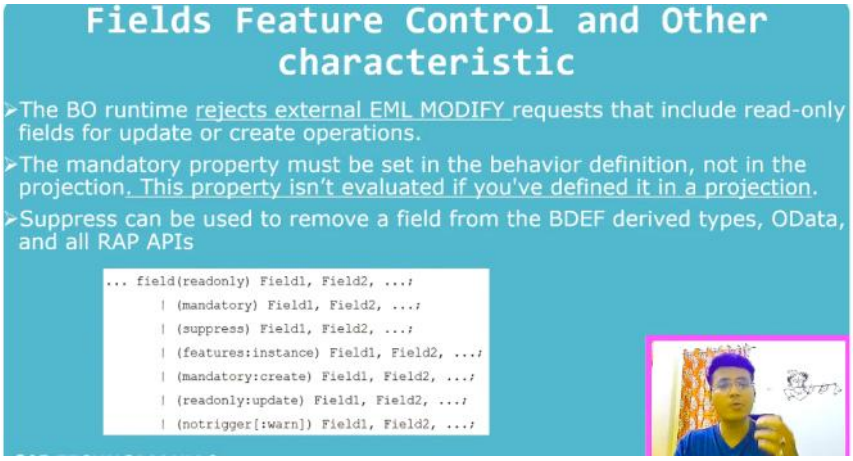
- Feature control cannot be used for internal operations.
- For static RAP BO operations, only global feature control is available. Instance feature control is not available.
- Feature control runtime checks are not evaluated for EML calls with the addition IN LOCAL MODE.
- The EML statement GET PERMISSIONS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

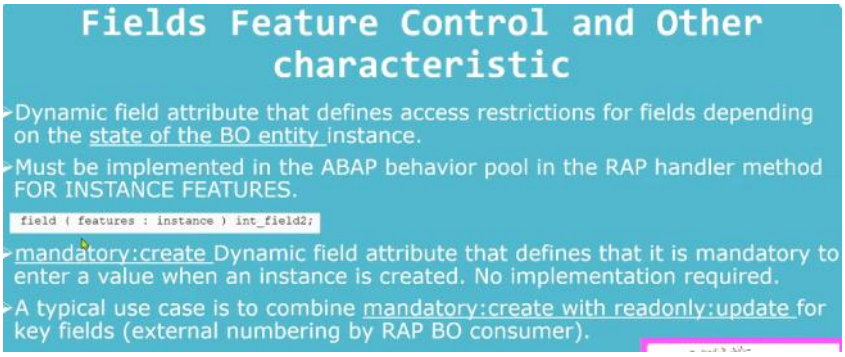
Instance Feature Control Operations

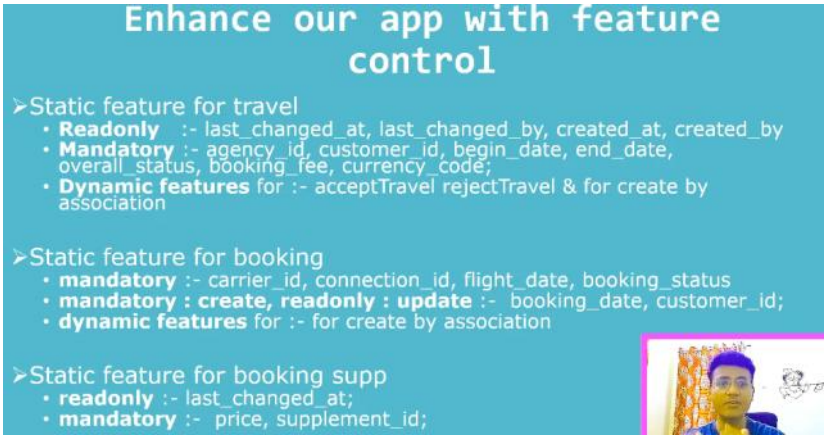
- Instance feature control enables or disables operations depending on the state of the BO entity instance.
- If an operation is disabled by means of instance feature control, it is grayed out on the user interface of an OData application
- If an external RAP BO consumer tries to execute a disabled operation 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







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

```
define behavior for YI_travel1_M //alias <alias_name>
implementation in class zcl_bp_travel_managed unique
persistent table ytravel_m
lock master
authorization master ( instance )
etag master LastChangedAt
early numbering
{

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

  action ( features : instance ) acceptTravel result [1] $self;
  action ( features : instance ) rejectTravel result [1] $self;

  factory action copytravel [1];

  internal action recalcTotPrice;

  create ( authorization : global );
  update;
  delete;

  association _booking { create (features : instance ); }
  mapping for ytravel_m
    {
      TravelId      = travel_id;
      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;
```

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

```
define behavior for Yi_bookSuppl1_m //alias <alias_name>
implementation in class zcl_bp_BookSuppl_managed unique
persistent table ybooksuppl_m
lock dependent by _travel
authorization dependent by _travel
etag master LastChangedAt
//etag dependent by _travel
early numbering
{
  update;
  delete;
  field ( readonly ) TravelId, BookingId, BookingSupplementId, LastChangedAt;
  field ( mandatory ) Price, SupplementId;
  association _travel;
  association _booking;
  mapping for ybooksuppl_m
```

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                                                                       ) ).
272
273
274   ENDMETHOD.
275
```

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 ) )  ).
78
79
80    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

```
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;
```

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                 agency_id; }
             validateDates                  begin_date, end_date; }
             validateStatus                 overall_status; }
             validateCurrencyCode           currency_code; }
             validateBookingFee             booking_fee; }

Booking Validation :-
           validateStatus                   booking_status; }
           validateCustomer                 customer_id; }
           validateCurrencyCode             currency_code;  }
           validateConnection               carrier_id, connection_id,  flight_date; }
           validateFlightPrice              flight_price; }

Booking Supp Validation :-
           validateCurrencyCode             currency_code; }
           validateSupplement               supplement_id; }
           validatePrice                    price; }
```

```
managed; //implementation in class zbp_i_travel1_m unique;
strict ( 2 );

define behavior for YI_travel1_M //alias <alias_name>
implementation in class zcl_bp_travel_managed unique
persistent table ytravel_m
lock master
authorization master ( instance )
etag master LastChangedAt
early numbering
{

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

  action ( features : instance ) acceptTravel result [1] $self;
  action ( features : instance ) rejectTravel result [1] $self;

  factory action copytravel [1];
  internal action recalcTotPrice;

  validation validateCustomer on save { create; field CustomerId; }
  validation validateDates on save { create; field BeginDate, EndDate; }
  validation validateStatus on save { create; field OverallStatus; }
  validation validateCurrencyCode on save { create; field CurrencyCode; }
  validation validateBookingFee on save { create; field BookingFee; }
```

```
METHOD validateCustomer.
  READ ENTITIES OF yi_travel1_m IN LOCAL MODE
  ENTITY YI_travel1_M
  FIELDS ( CustomerId )  WITH CORRESPONDING #( keys )
  RESULT DATA(lt_travel).

  READ TABLE lt_travel INTO DATA(ls_cust) INDEX 1.
  SELECT
  FROM /dmo/customer
  FIELDS ( customer_id )
  WHERE customer_id = @ls_cust-CustomerId
  INTO TABLE @DATA(lt_cust_db).

  IF sy-subrc ne 0.
    LOOP AT lt_travel ASSIGNING FIELD-SYMBOL(<fs_travel>).
      IF <fs_travel>-CustomerId IS INITIAL OR NOT line_exists( lt_cust_db[ customer_id = <fs_travel>-CustomerId ] ).
        APPEND VALUE #( %tky = <fs_travel>-%tky ) TO failed-yi_travel1_m.
        APPEND VALUE #( %tky = <fs_travel>-%tky
                        %msg = NEW /dmo/cm_flight_messages(
                                                              textid                = /dmo/cm_flight_messages=>customer_unkown
                                                              customer_id           = <fs_travel>-CustomerId
                                                              severity = if_abap_behv_message=>severity-error

                                                            )
                        %element-customerid = if_abap_behv=>mk-on
                        )  TO reported-yi_travel1_m.
      ENDIF.
    ENDLOOP.
  ENDIF.

ENDMETHOD.
```

OR

```
METHOD validateCustomer.
  READ ENTITIES OF yi_travel1_m IN LOCAL MODE
  ENTITY YI_travel1_M
  FIELDS ( CustomerId )  WITH CORRESPONDING #( keys )
  RESULT DATA(lt_travel).

  READ TABLE lt_travel INTO DATA(ls_cust) INDEX 1.
  SELECT
  FROM /dmo/customer
  FIELDS ( customer_id )
  WHERE customer_id = @ls_cust-CustomerId
  INTO TABLE @DATA(lt_cust_db).

  IF sy-subrc NE 0.
    LOOP AT lt_travel ASSIGNING FIELD-SYMBOL(<fs_travel>).
      IF <fs_travel>-CustomerId IS INITIAL OR NOT line_exists( lt_cust_db[ customer_id = <fs_travel>-CustomerId ] ).
        APPEND VALUE #( %tky = <fs_travel>-%tky ) TO failed-yi_travel1_m.
        APPEND VALUE #( %tky = <fs_travel>-%tky
                        %msg = new_message_with_text(
                                 severity = if_abap_behv_message=>severity-error
                                 text     = 'Customer ID is Unknown'
                               )
                        )  TO reported-yi_travel1_m.
      ENDIF.
    ENDLOOP.
  ENDIF.

ENDMETHOD.
```

```
METHOD validateDates.
  READ ENTITIES OF yi_travel1_m iN LOCAL MODE
  ENTITY yi_travel1_m
  FIELDS ( BeginDate EndDate ) WITH CORRESPONDING #( keys )
  RESULT DATA(lt_travel).

  LOOP AT lt_travel INTO DATA(ls_travel).
    IF ls_travel-BeginDate GT ls_travel-EndDate.
      APPEND VALUE #( %key = ls_travel-%key ) TO failed-yi_travel1_m.
      APPEND VALUE #( %tky = ls_travel-%tky
                      %msg = NEW /dmo/cm_flight_messages(
                                                          textid               = /dmo/cm_flight_messages=>begin_date_bef_end_date
                                                          travel_id            = ls_travel-TravelId
                                                          begin_date           = ls_travel-BeginDate
                                                          end_date             =  ls_travel-EndDate
                                                          severity             = if_abap_behv_message=>severity-error )
                      %element-begindate = if_abap_behv=>mk-on
                      %element-enddate = if_abap_behv=>mk-on

                    ) TO reported-yi_travel1_m.

    ELSEIF ls_travel-BeginDate < cl_abap_context_info=>get_system_date(  ).

      APPEND VALUE #( %key = ls_travel-%key ) TO failed-yi_travel1_m.
      APPEND VALUE #( %tky = ls_travel-%tky
                %msg = NEW /dmo/cm_flight_messages(
                                              textid               = /dmo/cm_flight_messages=>begin_date_on_or_bef_sysdate
                                              begin_date           = ls_travel-BeginDate
                                              severity             = if_abap_behv_message=>severity-error )
                %element-begindate = if_abap_behv=>mk-on
                %element-enddate = if_abap_behv=>mk-on

              ) TO reported-yi_travel1_m.


    ENDIF.
  ENDLOOP.

ENDMETHOD.
```

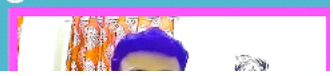
```
METHOD validateStatus.

  READ ENTITIES OF yi_travel1_m IN LOCAL MODE
    ENTITY yi_travel1_m
    FIELDS ( OverallStatus ) WITH CORRESPONDING #( keys )
    RESULT DATA(lt_travel).

  LOOP AT lt_travel INTO DATA(ls_travel).
    CASE ls_travel-OverallStatus.
      WHEN 'O'.
      WHEN 'A'.
      WHEN 'X'.
      WHEN OTHERS.

        APPEND VALUE #( %key = ls_travel-%key ) TO failed-yi_travel1_m.
        APPEND VALUE #( %tky = ls_travel-%tky
                        %msg = NEW /dmo/cm_flight_messages(
                                                              textid              = /dmo/cm_flight_messages=>status_invalid
                                                              status            = ls_travel-overallstatus
                                                              severity            = if_abap_behv_message=>severity-error )
                        %element-OverallStatus = if_abap_behv=>mk-on

               ) TO reported-yi_travel1_m.
    ENDCASE.
  ENDLOOP.

ENDMETHOD.
```

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 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)

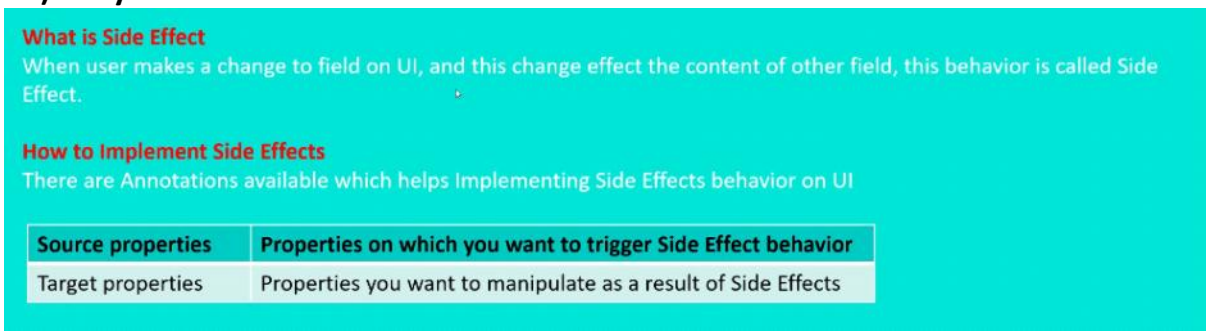
```
determination fillCurrency on modify { create; update; }
```

Side Effects

09 December 2025 11:23

Two Ways

1) way



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

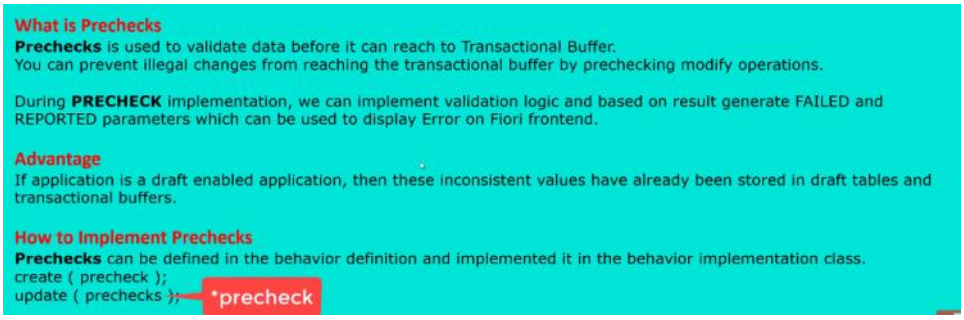
```
METHOD Course_Duration.

  IF keys IS NOT INITIAL.

    READ ENTITIES OF ZI_student_um IN LOCAL MODE
    ENTITY Student
    FIELDS ( Course )
    WITH CORRESPONDING #( keys )
    RESULT DATA(lt_student).

    DATA(cc) = lt_student[ 1 ]-%data-Course.
    DATA(c_duration) = COND #( WHEN cc = 'BTECH' OR cc = 'btech' THEN 4
                               WHEN cc = 'mca' OR cc = 'MCA' THEN 2
                               WHEN cc = 'BSC' OR cc = 'bsc' OR cc = 'bca' OR cc = 'BCA'
                               THEN 3
                               ).
    MODIFY ENTITIES OF ZI_student_um IN LOCAL MODE
    ENTITY Student
    UPDATE FIELDS ( Courseduration )
    WITH VALUE #( ( %tky = keys[ 1 ]-%tky %data-Courseduration = c_duration  ) )
    MAPPED DATA(lt_mapped)
    FAILED DATA(lt_failed)
    REPORTED DATA(lt_reported).

  ENDIF.

ENDMETHOD.
```

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

```
    draft determine action Prepare
    {
      validation ( always ) validate_fileds;
      determination ( always ) Course_Duration;
    }

    field ( mandatory ) Firstname, Lastname, Age;
    validation validate_fileds on save { create; update; }

    determination Course_Duration on save { create; field Course; }
    side effects { field Course affects field Courseduration; }

    field ( readonly ) Id, GenderDesc, StuentId, Courseduration;
```

Precheck

09 December 2025 12:51




```
115  METHOD precheck_update.
116
117    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        ENDIF.
145      ENDIF.
146    ENDLOOP.
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.
```

```
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
```

```
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.

      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**

```
@UI: {
    lineItem: [{ position: 10,
                 label: 'Update Status',
                 type: #FOR_ACTION,
                 dataAction: 'statusUpdate',
                 invocationGrouping: #CHANGE_SET
     }],
    identification: [{ position: 10, label: 'Student ID' }]
}
Id;
```

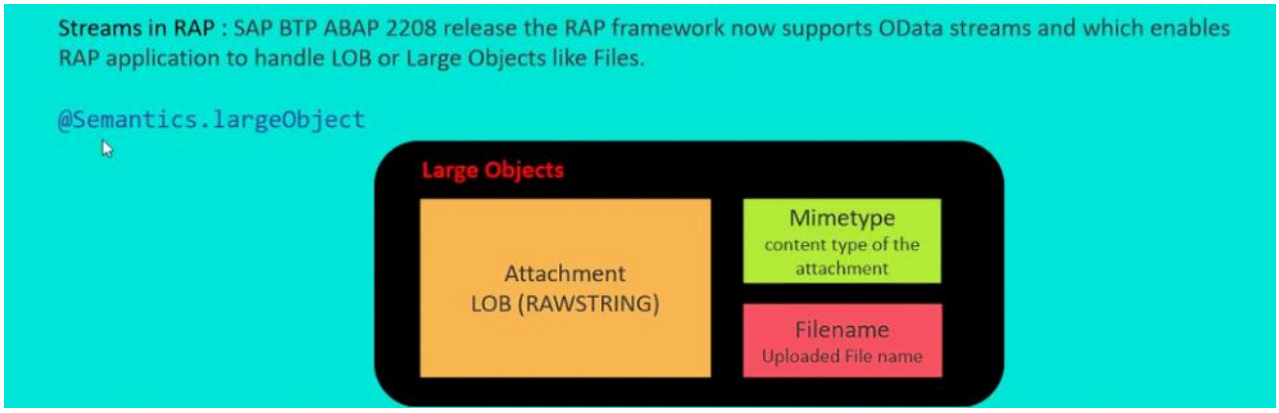
#ISOLATED can be used with @UI.lineItem.**invocationGrouping**

#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

```
@UI: {
    lineItem: [{ position: 10,
                 label: 'Update Status',
                 type: #FOR_ACTION,
                 dataAction: 'statusUpdate',
                 invocationGrouping: #ISOLATED
     }],
    identification: [{ position: 10, label: 'Student ID' }]
}
Id;
```

File Upload(Streams)

17 December 2025 12:40



Table

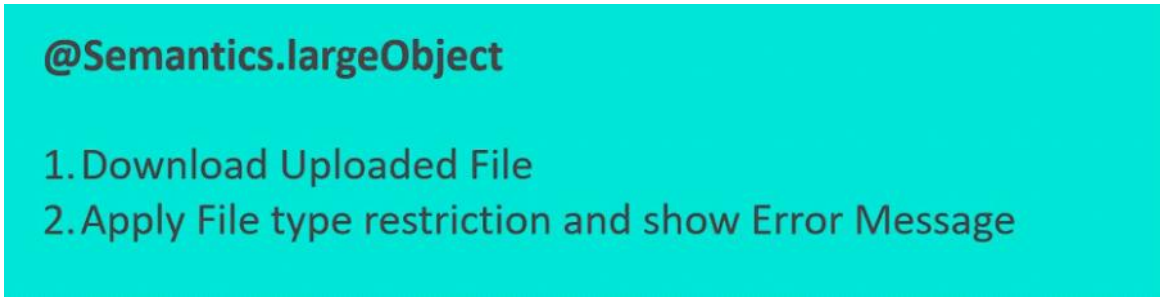
```
@EndUserText.label : 'Student Attachements of Resumes'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table ystudent_att_tab {

  key client     : abap.clnt not null;
  key attach_id : abap.char(32) not null;
  id             : sysuuid_x16 not null;
  comments       : abap.char(30);
  attachement    : abap.rawstring(256);
  mimetype       : abap.char(120);
  filename       : abap.char(120);

}
```

IN Projection

```
      @UI: {
      lineItem: [{ position: 40 }],
      identification: [{ position: 40 }]
      }
      @EndUserText.label: 'Attachement'

      @Semantics.largeObject: {
      mimeType: 'Mimetype',
      fileName: 'Filename',
      acceptableMimeTypes: [ '' ],
      contentDispositionPreference: #INLINE,
      cacheControl: {
        maxAge: #SHORT
      }
      }

      Attachement,
      @EndUserText.label: 'File Type'
      Mimetype,
```

BDEF

```
define behavior for Ystudent_i_att alias Attachements
persistent table ystudent_att_tab
draft table ystudent_att_d
lock dependent by _Student
authorization dependent by _Student
etag master LastChangedAt
{
  update;
  delete;
  field ( readonly )  id;
  association _Student { with draft; }

  mapping for ystudent_att_tab
    {
      AttachId    = attach_id;
      Attachement = attachement;
      Comments    = comments;
      Filename    = filename;
      Id          = id;
      Mimetype    = mimetype;
    }


}
```



TO Download The File , Use Below Property in Projection

```
      @UI: {
      lineItem: [{ position: 40 }],
      identification: [{ position: 40 }]
      }
      @EndUserText.label: 'Attachement'

      @Semantics.largeObject: {
      mimeType: 'Mimetype',
      fileName: 'Filename',
      acceptableMimeTypes: [ '' ],
      contentDispositionPreference: #ATTACHMENT,
      cacheControl: {
        maxAge: #SHORT
      }
      }

      Attachement,
      @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/rtf	rtf	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

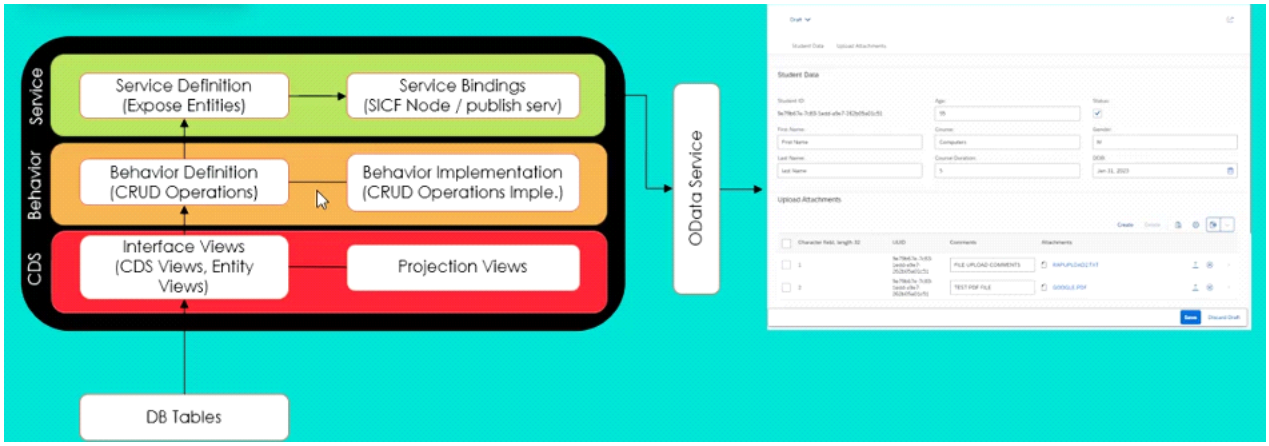
12 January 2026 12:50

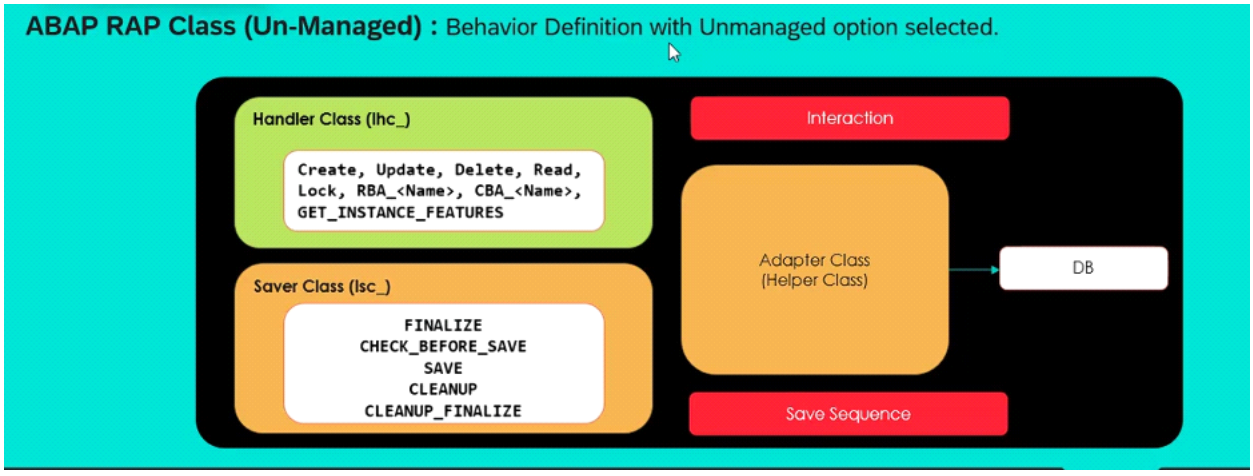
```
DATA: lt_root_create        TYPE TABLE FOR CREATE /dmo/fsa_r_roottp
\\root,
          lt_child_create       TYPE TABLE FOR CREATE
/dmo/fsa_r_roottp\\root\_child,
          lt_grandchild_create  TYPE TABLE FOR CREATE
/DMO/fsa_r_roottp\\Child\_Grandchild,
          lt_chart_create       TYPE TABLE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





Component of the Derived Types

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

```
unmanaged implementation in class zbp_i_student_um unique;
strict ( 2 );

with draft;

define behavior for ZI_student_um alias Student
draft table ystudent_d_um

//late numbering
early numbering

lock master
total etag Lastchangedat
authorization master ( instance )
etag master Lastchangedat
{
  create ( authorization : global );
  update ( features : instance );
  delete ( features : instance );

  draft action Edit;
  draft action Activate optimized;
  draft action Resume;
  draft action discard;
  draft determine action Prepare;

  field ( readonly ) Id, GenderDesc, StuentId;


  association _result { create; }
  mapping for ystd_table
    {
      Id                = id;
      StuentId          = sid;
      Firstname         = firstname;
      Lastname          = lastname;
      Age               = age;
      Course            = course;
      Courseduration    = courseduration;
      Status            = status;
      Gender            = gender;
      Dob               = dob;
      Lastchangedat     = lastchangedat;
      Locallastchangedat = locallastchangedat;
    }
}
```

```
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
13      METHODS earlynumbering_create FOR NUMBERING
14        IMPORTING entities FOR CREATE Student.
15
```

```
34      METHODS earlynumbering_cba_Results FOR NUMBERING
35        IMPORTING entities FOR CREATE Student\_Results.
36
```

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

```
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.
```

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
```

```
76  METHOD update.
77    zcl_students_api_class=>get_instance( )->update_student(
78      EXPORTING
79        entities = entities
80      CHANGING
81        mapped   = mapped
82        failed   = failed
83        reported = reported
84    ).
85  ENDMETHOD.
86
```

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 ).
120
121   endMETHOD.
122
```

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 draft;}
mapping for ystd_table control zcs_stud_prop_um
  {
    Id                = id;
    StuentId          = sid;
    Firstname         = firstname;
```

Helper Class

```
METHOD update_student.
  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.
```

```
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 Aassociation

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
115  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
126  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
185  METHOD earlynumbering_cba_result.
186    DATA(lv_new_id) = get_next_id(  ).
187    LOOP AT entities ASSIGNING FIELD-SYMBOL(<lfs_entities>).
188      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
197  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        ENDLOOP.
205      ENDLOOP.
206    ENDMETHOD.
207
```

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   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

```
METHOD delete.
  zcl_students_api_class=>get_instance( )->delete_student(
    EXPORTING
      keys     = keys
    CHANGING
      mapped   = mapped
      failed   = failed
      reported = reported
  ).
ENDMETHOD.
```

IN Helper Class

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

```
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.
```

In Save Method

```
METHOD save_data.
  IF  NOT gt_student IS INITIAL.
    MODIFY ystd_table FROM TABLE @gt_student.
  ENDIF.
  IF  gt_result IS NOT INITIAL.
    MODIFY yst_result_tab FROM TABLE @gt_result.
  ENDIF.
  IF  gt_student_d IS NOT INITIAL.
    DELETE FROM ystd_table WHERE id IN @gt_student_d.
  ENDIF.
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

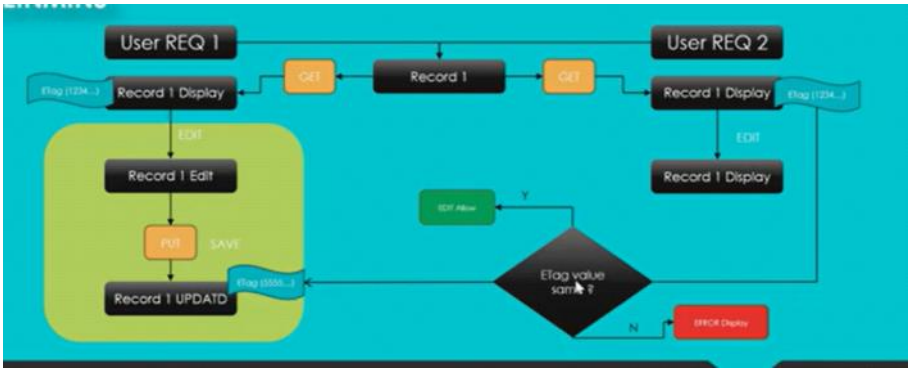
IN LHC

```
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.
```

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

```
METHOD create_student.
  gt_student = CORRESPONDING #( entities MAPPING FROM ENTITY ).
  IF  gt_student IS NOT INITIAL.
    gt_student[ 1 ]-sid = get_next_student_id(  ).

    GET TIME STAMP FIELD  lv_timestampl.
    gt_student[ 1 ]-locallastchangedat = lv_timestampl.
    gt_student[ 1 ]-lastchangedat = lv_timestampl.

    mapped = VALUE #( student = VALUE #( FOR ls_entity IN entities (
                                         %cid = ls_entity-%cid
                                         %is_draft = ls_entity-%is_draft
                                         %key = ls_entity-%key  ) ) ).

  ENDIF.

ENDMETHOD.
```

```
unmana Behavior Definition ZI_STUDENT_UM [TRL,100] - active - TRL_EN nt_um unique;
strict ( 2 );

with draft;

define behavior for ZI_student_um alias Student
draft table ystudent_d_um

//late numbering
early numbering

lock master
total etag Locallastchangedat
authorization master ( instance )
etag master Locallastchangedat
{
  create ( authorization : global );
```

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

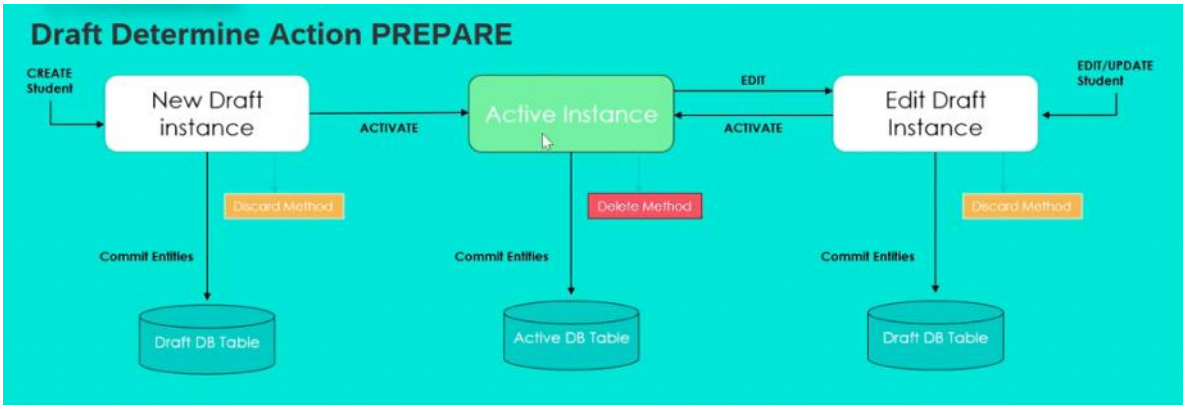
14 December 2025 22:27

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.



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

```
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
                                           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
```

```
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
```

```
                                                  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

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Domain Entity'
@Metadata.ignorePropagatedAnnotations: true
define view entity YDomain_entity
  as select from DDCDS_CUSTOMER_DOMAIN_VALUE_T( p_domain_name: 'YSTATUS' )
{
  key domain_name,
  key value_position,
  key language,
      value_low,
      text
}

```

ON Premises

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Entity For Domain'
@Metadata.ignorePropagatedAnnotations: true
define view entity YStatus_Domain
  as select from dd07t
{
      //  key domname    as Domname,
  key  ddlanguage as lang,
      //  key as4local   as As4local,
      //  key valpos     as Valpos,
      //  key as4vers    as As4vers,
      //      domval_ld  as DomvalLd,
      //      domval_hd  as DomvalHd,
      domvalue_l as Domvalue,
      ddtext     as Description
}

where
  domname = 'YSTATUS_PR'

```

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

```
@EndUserText.label: 'Abstract Entity'

@ObjectModel.resultSet.sizeCategory: #XS

define abstract entity ZA_student_entity
  //  with parameters parameter_name : parameter_type

{

  @EndUserText.label: 'Set Student Status'
  @UI.defaultValue:'Y'
  @Consumption.valueHelpDefinition: [{ entity: {
      name       : 'YDomain_entity',
      element    : 'text'
  } }]
  studentstatus  : abap.char( 5 );

  @UI.defaultValue:#( 'ELEMENT_OF_REFERENCED_ENTITY : Course' )
  course         : abap.char(20);
  @UI.defaultValue:#( 'ELEMENT_OF_REFERENCED_ENTITY : Courseduration' )

  courseduration : abap.numc(4);
  @UI.defaultValue:#( 'ELEMENT_OF_REFERENCED_ENTITY :status' )
  status         : abap_boolean;

}
```

IN BDEF

```
action updatestudentstatus parameter ZA_student_entity result [1] $self;
```

```
@EndUserText.label: 'Abstract Entity For PopUp'
define abstract entity YBS_I_PopupEntity
  //  with parameters parameter_name : parameter_type
{
  @Consumption.valueHelpDefinition: [{ entity: {
      name       : 'yBS_C_CountryVH',
      element    : 'Country'
  } }]
  @EndUserText.label: 'Search Country'
  SearchCountry : land1;
  @EndUserText.label: 'New Date Time'
  @Semantics.systemDateTime.createdAt: true
  NewDate       : abp_creation_tstmpl;
  @EndUserText.label: 'Message Type'
  MessageType   : abap.int4;
  @EndUserText.label: 'Update data'
  FlagUpdate    : abap.char(1);
  @EndUserText.label: 'Show Messages'
  FlagMessage   : abap_boolean;
}
```

```
@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
}

```

Custom Entity

23 December 2025 10:47

S42020(ON PREMISE)
ABAP ON CLOUD

```
Define view <CDS VIEW NAME> AS SELECT FROM MARA
{
KEY   MATNR,
      MTART,
      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

```
@EndUserText.label: 'CDS Custom Entity'
@ObjectModel.query.implementedBy: 'ABAP:ZCL_CDS_CUSTOM_ENTITY'
define custom entity Zcds_custom_entity1
  // with parameters parameter_name : parameter_type
{
  key matnr : matnr;
      mtart : mtart;
      matkl : matkl;
      mbrsh : mbrsh;

}
```

```
class ZCL_CDS_CUSTOM_ENTITY definition public final create public .
public section.
  interfaces IF_RAP_QUERY_PROVIDER .
protected section.
private section.
ENDCLASS.

CLASS ZCL_CDS_CUSTOM_ENTITY IMPLEMENTATION.


  METHOD if_rap_query_provider~select.
    DATA(lo_paging) = io_request->get_paging( ).
    DATA(lv_page_size) = lo_paging->get_page_size( ).
    DATA(lv_offset) = lo_paging->get_offset( ).

    DATA(lv_max_rows) = COND #( WHEN lv_page_size = if_rap_query_paging=>page_size_unlimited
                                THEN 0
                                ELSE lv_page_size ).
    SELECT matnr,  mtart, matkl, mbrsh FROM mara INTO TABLE @DATA(lt_data) UP TO 100 ROWS.

    io_response->set_data( it_data = lt_data ).
*      CATCH cx_rap_query_response_set_twic. " RAP query response set twice
  ENDMETHOD.
ENDCLASS.
```

or

Gui

Ty.	Parameter	Typing	Description
	IO_REQUEST	TYPE REF TO IF_RAP_QUERY_REQUEST	
	IO_RESPONSE	TYPE REF TO IF_RAP_QUERY_RESPONSE	
	CX_RAP_QUERY_PROV_NOT_IMPL		
	CX_RAP_QUERY_PROVIDER		

Method IF_RAP_QUERY_PROVIDER~SELECT active

```
METHOD if_rap_query_provider~select.
  DATA(lo_paging) = io_request->get_paging( ).
  DATA(lv_page_size) = lo_paging->get_page_size( ).
  DATA(lv_offset) = lo_paging->get_offset( ).

  DATA(lv_max_rows) = COND #( WHEN lv_page_size = if_rap_query_paging=>page_size_unlimited
                              THEN 0
                              ELSE lv_page_size ).
  SELECT matnr,  mtart, matkl, mbrsh FROM mara INTO TABLE @DATA(lt_data) UP TO 100 ROWS.

  io_response->set_data( it_data = lt_data ).
*      CATCH cx_rap_query_response_set_twic. " RAP query response set twice
ENDMETHOD.
```

Custom Entity

24 December 2025 11:04

```
@EndUserText.label: 'Custom Entity For Flight Data'
@ObjectModel:{ query.implementedBy: 'ABAP:YCL_CUSTOM_ENTITY1' }
@UI :{ headerInfo : {
typeName: 'Flight',
typeNamePlural: 'Flights',
title : { value: 'carrier_id' },
description : { value: 'connection_id' }
}}
define root  custom entity YCustom_entity
  // with parameters parameter_name : parameter_type
{
      @UI.facet      : [
           {
             id      :  'Flight_Data',
             purpose :  #STANDARD,
             type    :  #IDENTIFICATION_REFERENCE,
             label   :  'Flights',
             position: 10 }
         ]
      @UI.lineItem   : [{ position: 10 }]
      @UI.selectionField : [{position: 10}]
      @UI.identification: [{position: 10}]
  key carrier_id     : /dmo/carrier_id;
      @UI.lineItem   : [{ position: 20 }]
      @UI.selectionField : [{position: 20}]
      @UI.identification: [{position: 20}]
  key connection_id  : /dmo/connection_id;
      @UI.lineItem   : [{ position: 30 }]
      @UI.selectionField : [{position: 30}]
      @UI.identification: [{position: 30}]
  key flight_date    : /dmo/flight_date;

      @UI.lineItem   : [{ position: 40 }]
      @UI.identification: [{position: 40}]
      @Semantics.amount.currencyCode : 'currency_code'
      price          : /dmo/flight_price;

      @UI.lineItem   : [{ position: 50 }]
      @UI.identification: [{position: 50}]
      currency_code  : /dmo/currency_code;

```

Class

```
CLASS ycl_custom_entity1 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    INTERFACES : if_rap_query_provider.
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

```

```
CLASS ycl_custom_entity1 IMPLEMENTATION.
  METHOD if_rap_query_provider~select.
    IF io_request->is_data_requested(  ).

      DATA(lv_top) = io_request->get_paging( )->get_page_size(  ).
      IF lv_top < 0.
        lv_top = 1.
      ENDIF.

      DATA(lv_skip) = io_request->get_paging(  )->get_offset(  ).
      DATA(lv_sort) = io_request->get_sort_elements(  ).
      DATA : lv_orderby TYPE string.

      LOOP AT lv_sort INTO DATA(ls_sort).
        IF ls_sort-descending  = abap_true.
          lv_orderby = |{ lv_orderby }{ ls_sort-element_name } DESCENDING|.
        ELSE.
          lv_orderby = |{ lv_orderby }{ ls_sort-element_name } ASCENDING|.
        ENDIF.

      ENDLOOP.
      IF lv_orderby IS INITIAL.
        lv_orderby = 'CARRIER_ID'.
      ENDIF.

      DATA(lv_conditions) = io_request->get_filter( )->get_as_sql_string(  ).


      SELECT FROM /dmo/flight
      FIELDS carrier_id,connection_id,flight_date,price,currency_code,plane_type_id,
      seats_max,seats_occupied
      WHERE (lv_conditions)
      ORDER BY (lv_orderby)
      INTO TABLE @DATA(lt_flight)
      UP TO @lv_top ROWS OFFSET @lv_skip.

      IF io_request->is_total_numb_of_rec_requested(  ).
        io_response->set_total_number_of_records( lines( lt_flight ) ).
        io_response->set_data( lt_flight ).
      ENDIF.

    ENDIF.
  ENDMETHOD.
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

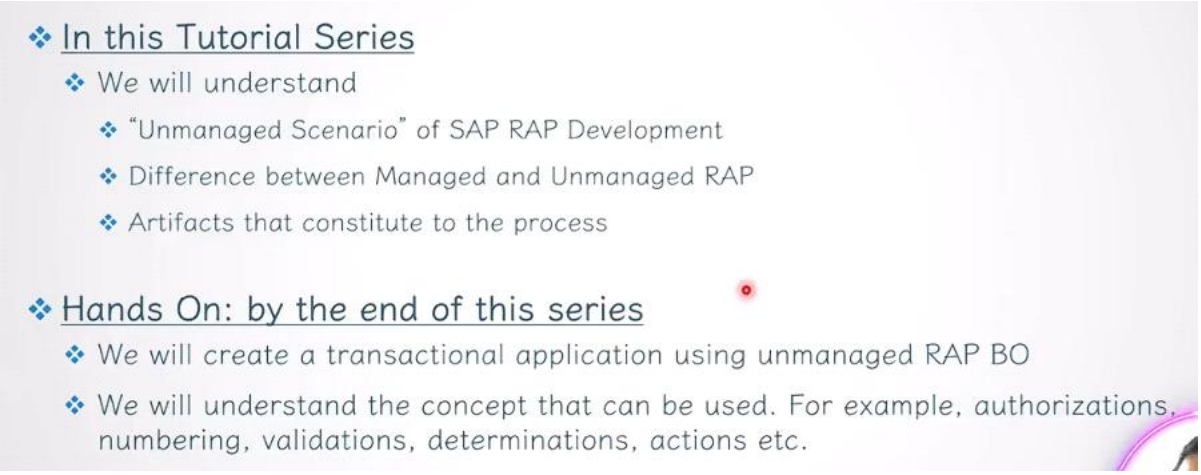
```
@UI : { lineItem: [{ position: 40 }],
identification: [{ position: 40,qualifier: 'pritem',hidden: #( isHide ) }] }
Qty;
@UI : { identification: [{ position: 50,qualifier: 'pritem',hidden: #( isHide ) }] }
Uom;
@UI : { lineItem: [{ position: 60 }],
identification: [{ position: 60 ,qualifier: 'pritem'}]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

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

```
METHOD if_sadl_exit_calc_element_read~calculate.

  DATA : lt_data TYPE TABLE OF yp_mm_pr_itm.

  lt_data = CORRESPONDING #( it_original_data ).
  LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<fs_data>).
    <fs_data>-isHide = COND abap_boolean( WHEN <fs_data>-Price IS INITIAL THEN abap_true
                                ELSE abap_false ).
  ENDLOOP.

  ct_calculated_data = CORRESPONDING #( lt_data ).

ENDMETHOD.
```

UN_managed2

18 December 2025 09:32

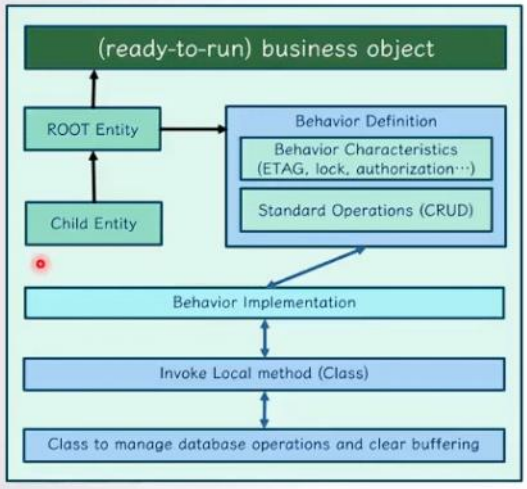


Introduction

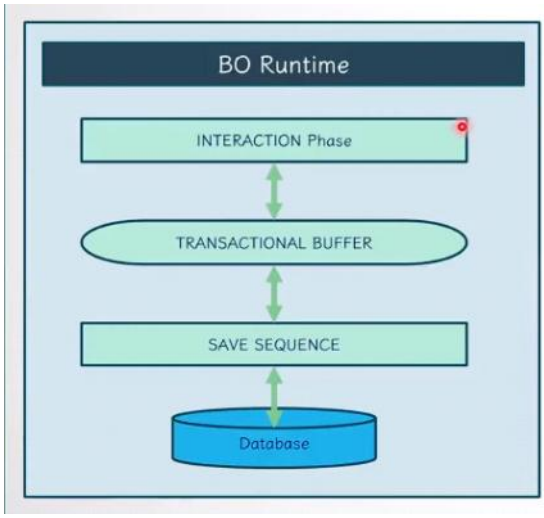
18 December 2025 09:33

- **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

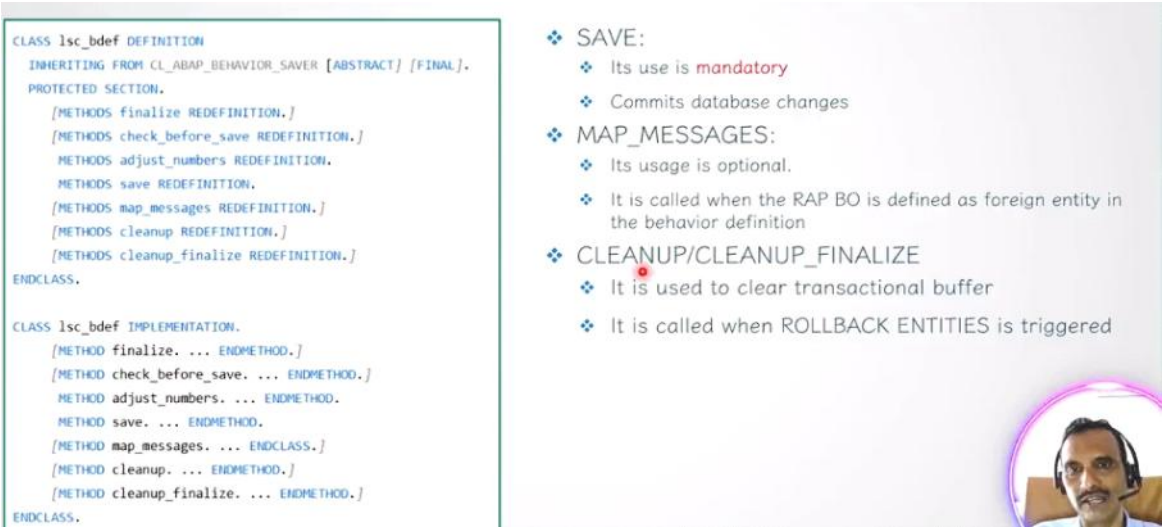
SAVE Sequence Methods

MS SAP TUTORIALS
Created by Munir
7

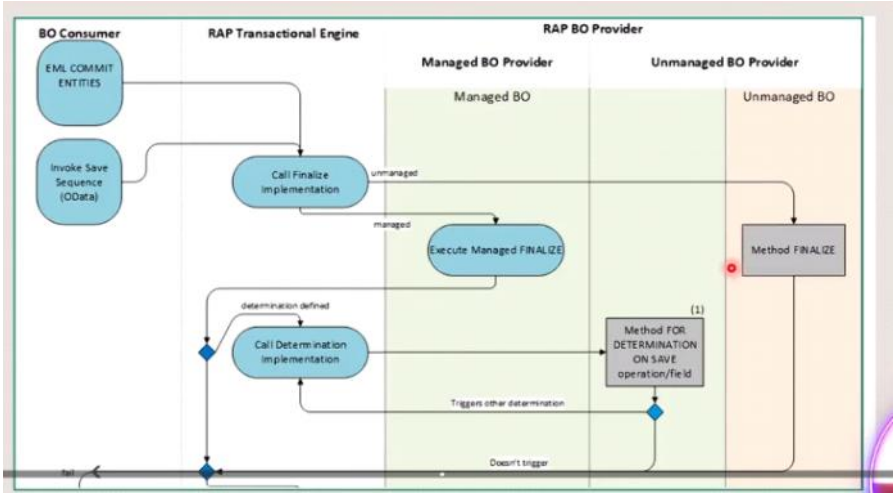
```
CLASS lsc_bdef DEFINITION
  INHERITING FROM CL_ABAP_BEHAVIOR_SAVER [ABSTRACT] [FINAL].
  PROTECTED SECTION.
    [METHODS finalize REDEFINITION.]
    [METHODS check_before_save REDEFINITION.]
     METHODS adjust_numbers REDEFINITION.
     METHODS save REDEFINITION.
    [METHODS map_messages REDEFINITION.]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

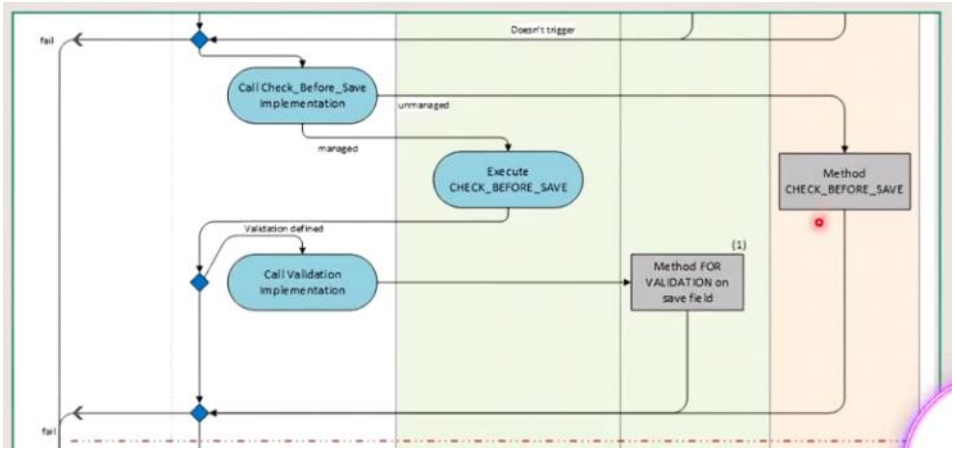
Volume

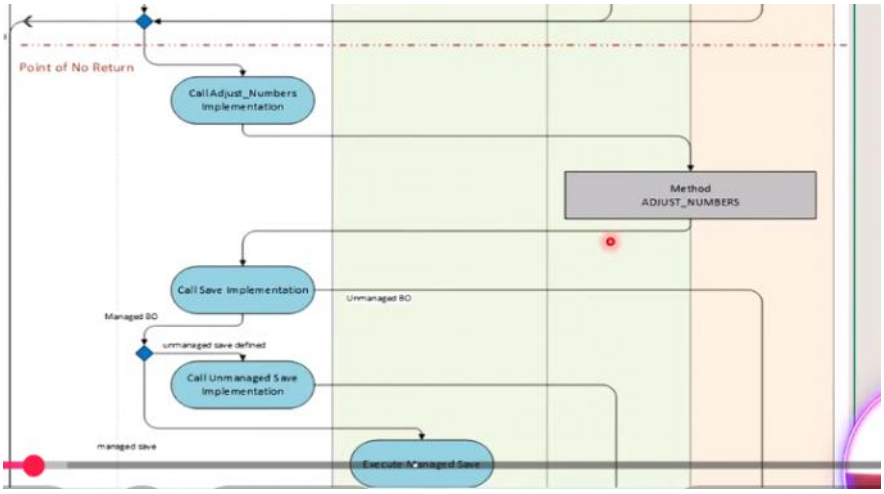
- FINALIZE:
  - Allows to do final modification to entities.
  - Method returns error in FAILED parameter and CLEANUP_FINALIZE is called
  - Its usage is optional and is part of early save phase
- CHECK_BEFORE_SAVE:
  - This method can be used to validate the data before committing to database
  - In 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 sce
  - Its usage is mandatory
  - It is part of late save phase

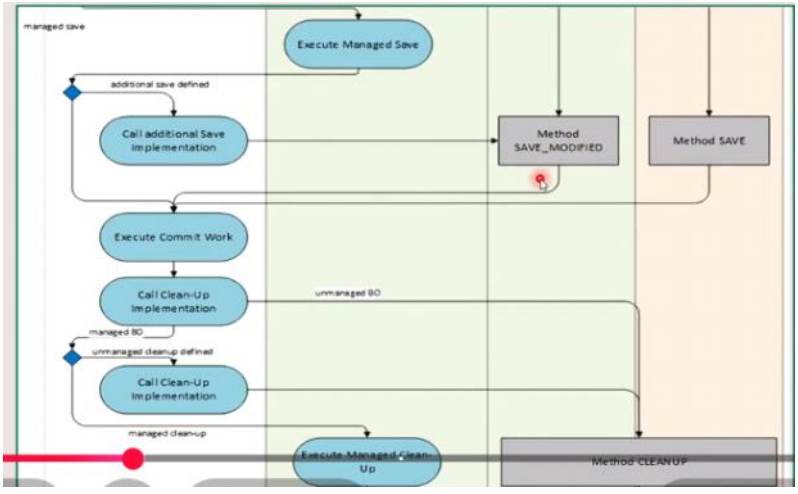


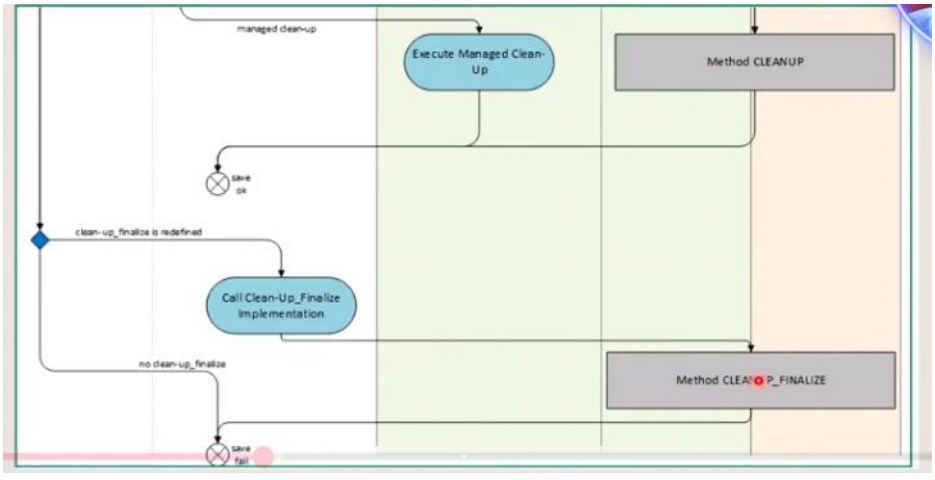

```
91  CLASS lsc_YPR_HEADER_I DEFINITION INHERITING FROM cl_abap_behavior_saver.
92    PROTECTED SECTION.
93
94      METHODS finalize REDEFINITION.
95
96      METHODS check_before_save REDEFINITION.
97
98      METHODS adjust_numbers REDEFINITION.
99
00      METHODS save REDEFINITION.
01
02      METHODS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




fail
Doesn't trigger
Call Check_Before_Save Implementation
unmanaged
managed
Execute CHECK_BEFORE_SAVE
Method CHECK_BEFORE_SAVE
Validation defined
Call Validation Implementation
(1)
Method FOR VALIDATION on save field
fail

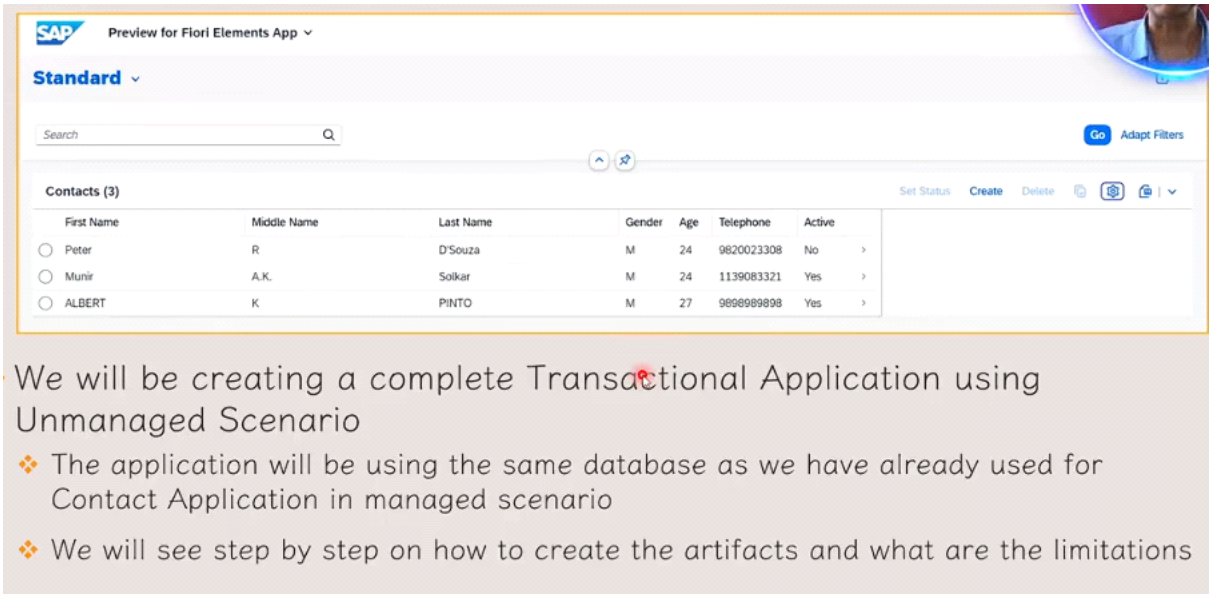

Point of No Return
Call Adjust_Numbers Implementation
Method ADJUST_NUMBERS
Call Save Implementation
Unmanaged BO
Managed BO
unmanaged save defined
Call Unmanaged Save Implementation
managed save


managed save
Execute Managed Save
additional save defined
Call additional Save Implementation
Method SAVE_MODIFIED
Method SAVE
Execute Commit Work
Call Clean-Up Implementation
unmanaged BO
managed BO
unmanaged cleanup defined
Call Clean-Up Implementation
managed clean-up
Execute Managed Clean-Up
Method CLEANUP


managed clean-up
Execute Managed Clean-Up
Method CLEANUP
save ok
clean-up_finalize is redefined
Call Clean-Up_Finalize Implementation
no clean-up_finalize
Method CLEANUP_FINALIZE
save fail

Practice

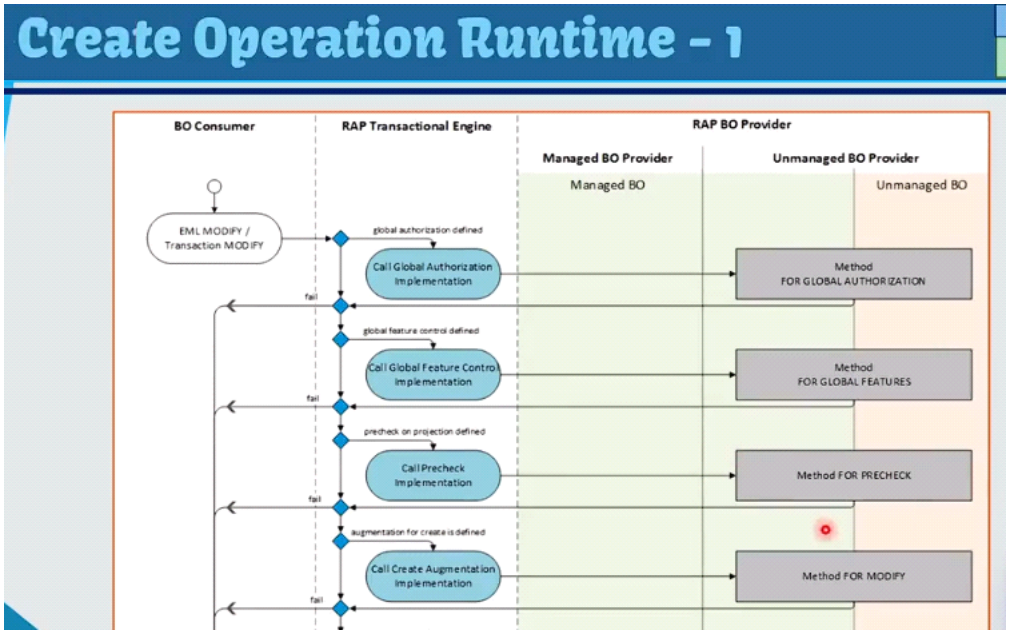
19 December 2025 09:35

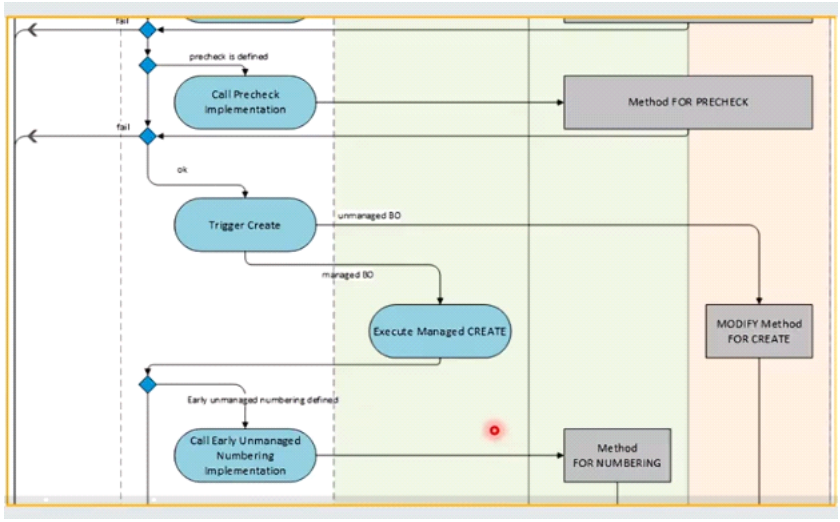


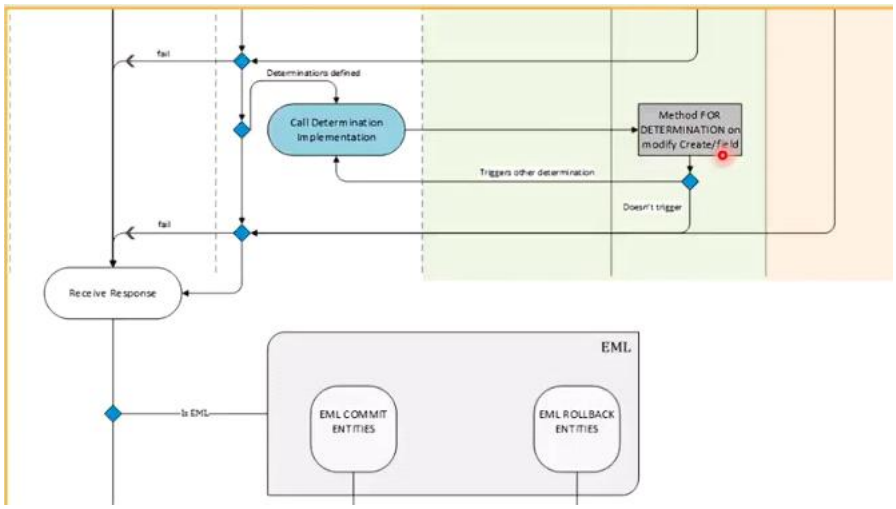
Create

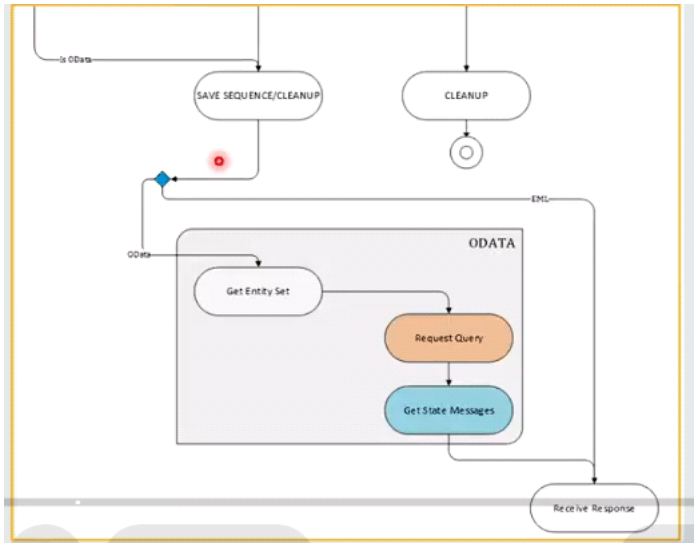
22 December 2025 10:07

- We will understand
  - "CREATE Operation" runtime
  - Sequence of methods that are called during the create operation







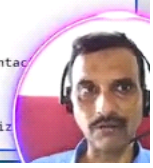



```
METHODS create FOR MODIFY
  IMPORTING entities FOR CREATE UNMGD_Contact.
```

```
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_contatt_mapping = DATA(lt_contatt_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_contac
ENDMETHOD.

METHOD cleanup.
  zms_cl_aux_contact_um=>get_instance( )->initializ
ENDMETHOD.
```



Requirement

22 December 2025 22:33

2)