Week 4: Dealing with Existing Code

Unit 1: The Business Scenario

Topics

Week 4

01 The Business Scenario

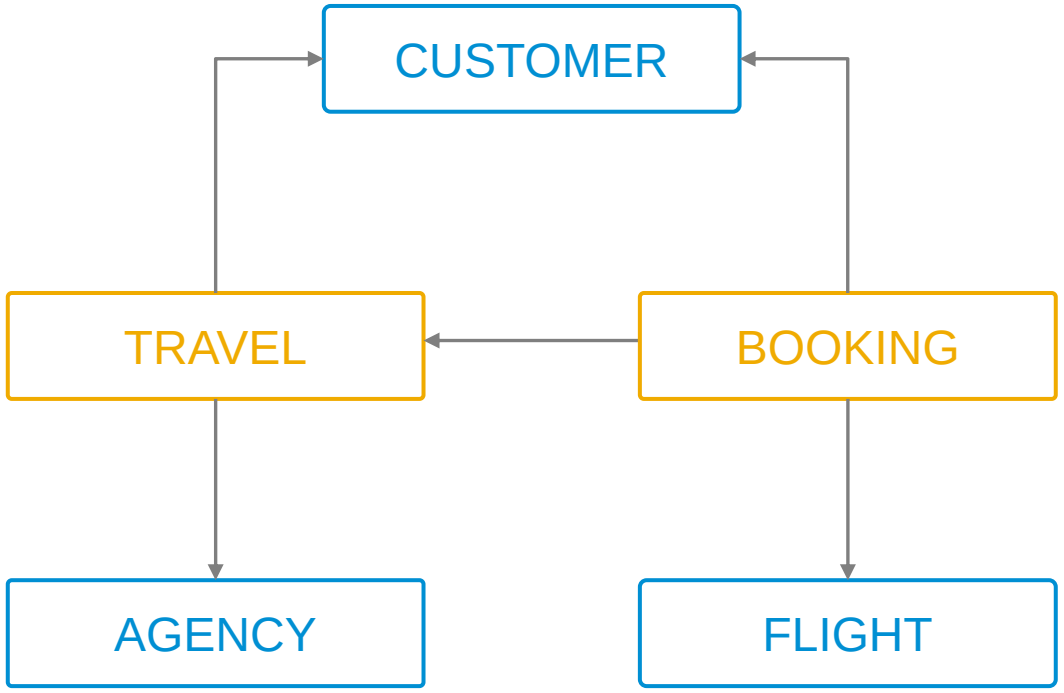
02 Creating the CDS Data Model

03 Defining and Implementing the Business Object Behavior

04 Creating the Business Object Projection

05 Building and Previewing the OData UI Service

Simplified flight data model for this openSAP course

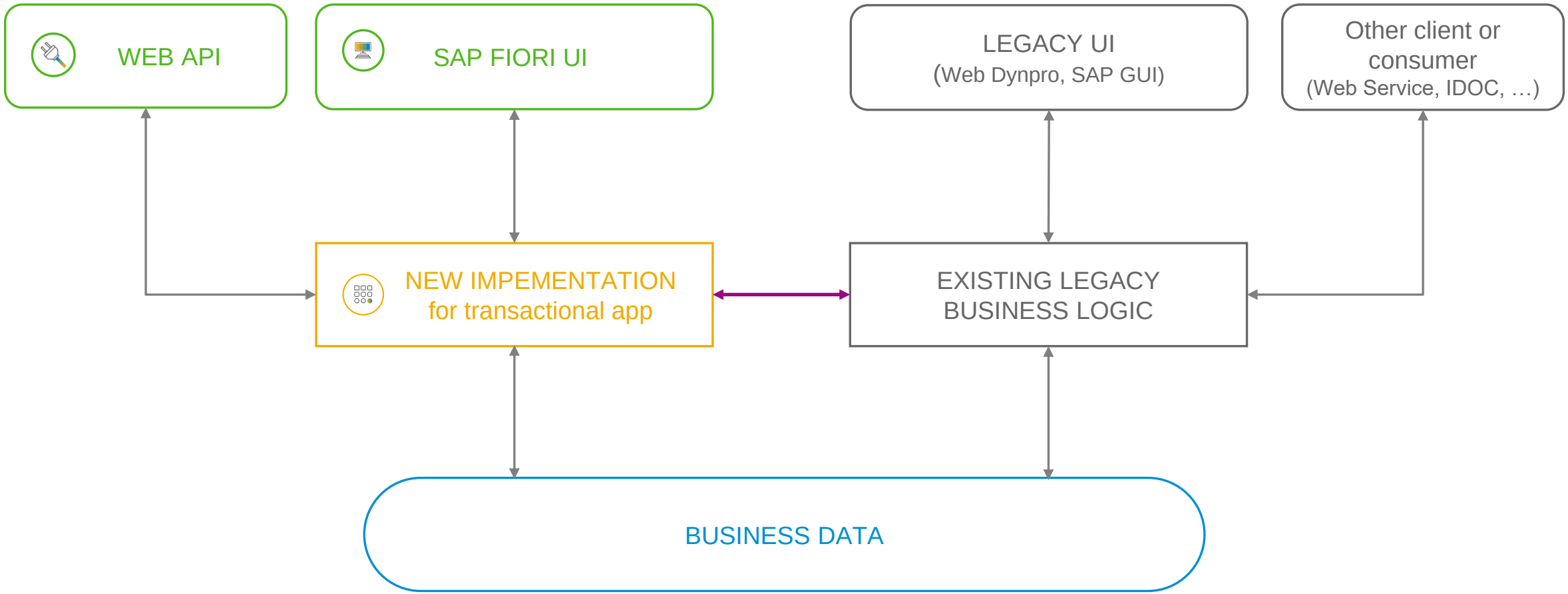


Main business entities in current scenario

Secondary business entities in current scenario

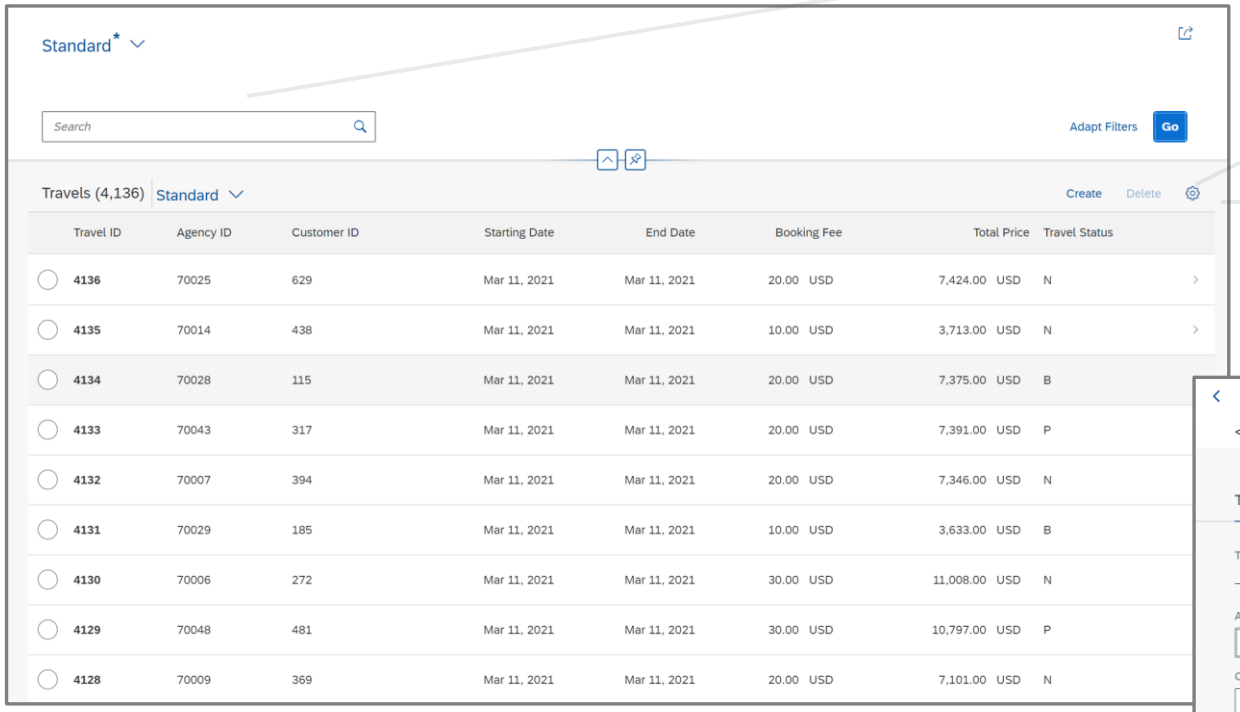
Travel	A Travel entity defines general travel data, such as the agency ID or customer ID, status of the travel booking, and the price of travel. The travel data is stored in the database table **`/DMO/I_TRAVEL`**.
Booking	The booking data is stored in the database table **`/DMO/BOOKING`**. The flight data model defines a 1:n cardinality between the Travel and the Booking entity.
Agency	An Agency entity defines travel agency data, such as the address and contact data. The corresponding data is stored in the database table **`/DMO/AGENCY`**. The flight data model defines a 1:n cardinality between Agency and Travel.
Flight	The concrete flight data for each connection is stored in the database table **`/DMO/FLIGHT`**. The flight data model defines a 1:n cardinality between the Connection and the Flight entity.
Customer	A Customer entity provides a detailed description of a flight customer (passenger) such as the name, the address, and contact data. The corresponding data is stored in the database table **`/DMO/CUSTOMER`**. The flight data model defines a 1:n cardinality between Customer and Travel

The unmanaged scenario



Resulting SAP Fiori elements app

Transactional List Report Travel App



Search

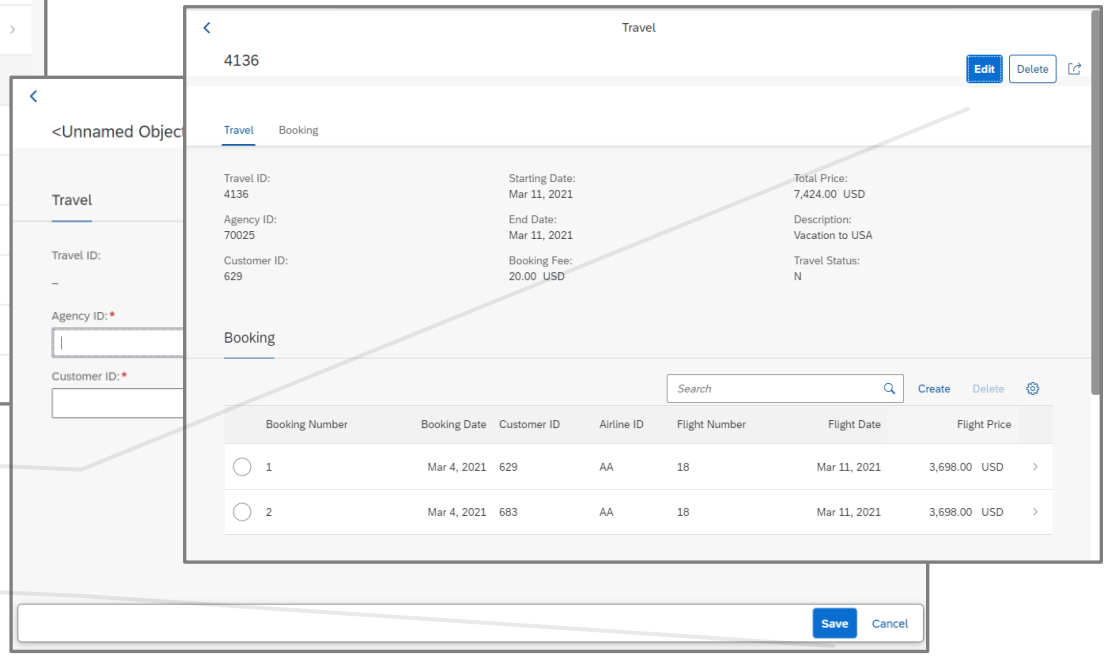
Create

Delete

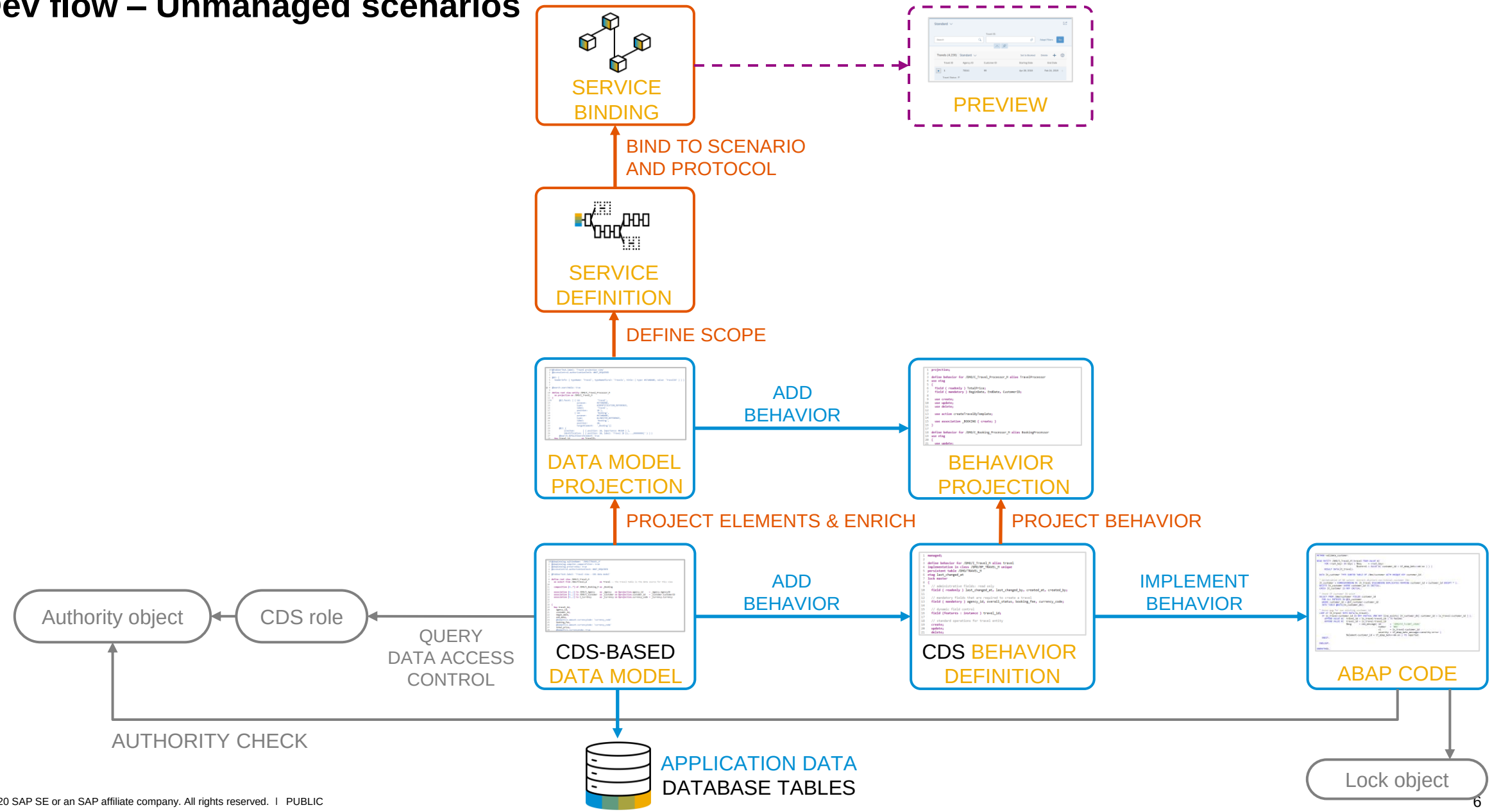
Update

Save

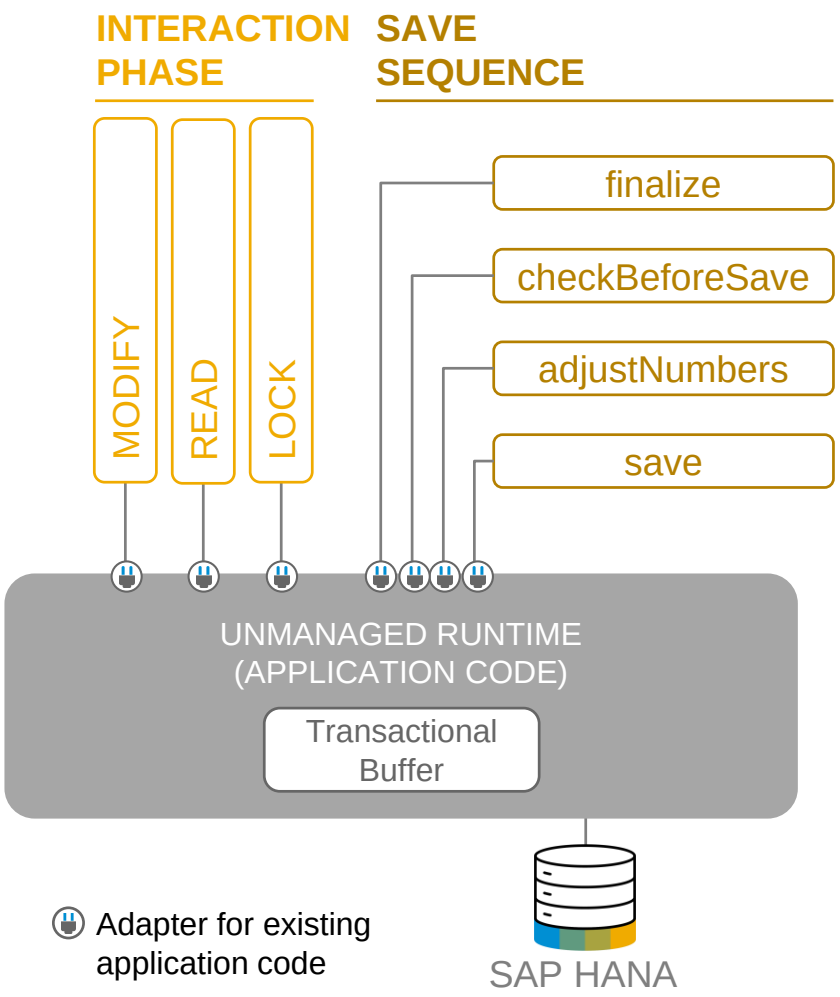
Object Page



Dev flow – Unmanaged scenarios



Unmanaged BO runtime implementation



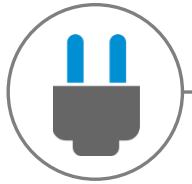
Application code

- Already available for interaction phase, transactional buffer, and save sequence
- Existing code decoupled from UI and protocol technologies
- Developers fully in charge of BO runtime, i.e. CRUD operations
- Adapters needed to integrate the existing code

Examples

- Sales Order, Purchase Order

Unmanaged BO runtime implementation – Adapter for existing application code



```
METHOD create.
  DATA messages   TYPE /dmo/t_message.
  DATA legacy_entity_in  TYPE /dmo/travel.
  DATA legacy_entity_out TYPE /dmo/travel.
  LOOP AT entities ASSIGNING FIELD-SYMBOL(<entity>).
    legacy_entity_in = CORRESPONDING #( <entity> MAPPING FROM ENTITY USING CONTROL ).
    CALL FUNCTION '/DMO/FLIGHT_TRAVEL_CREATE'
      EXPORTING
        is_travel   = CORRESPONDING /dmo/s_travel_in( legacy_entity_in )
      IMPORTING
        es_travel   = legacy_entity_out
        et_messages = messages.
    IF messages IS INITIAL.
      APPEND VALUE #( %cid = <entity>-%cid travelid = legacy_entity_out-travel_id )
                   TO mapped-travel.
    ELSE.
      "error handling
      …
    ENDIF.
  ENDLOOP.
ENDMETHOD.
```

Wrap-up

IN THIS UNIT, YOU LEARNED

- How the unmanaged implementation scenario for this week differs from the scenarios from the last units
- How RAP supports this new scenario

NEXT UNIT

- Week 4 – Unit 2
  Creating the CDS Data Model

Further reading

Additional Material

ABAP RESTful APPLICATION PROGRAMMING MODEL INFORMATION PAGE

For more information, links to documentation, tutorials, and more, please visit the RAP at openSAP information page by following the link below

RAP at openSAP information page (week 4)

Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP®

Week 4: Dealing with Existing Code

Unit 2: Creating the CDS Data Model

Topics

Week 4

01 The Business Scenario

02 Creating the CDS Data Model

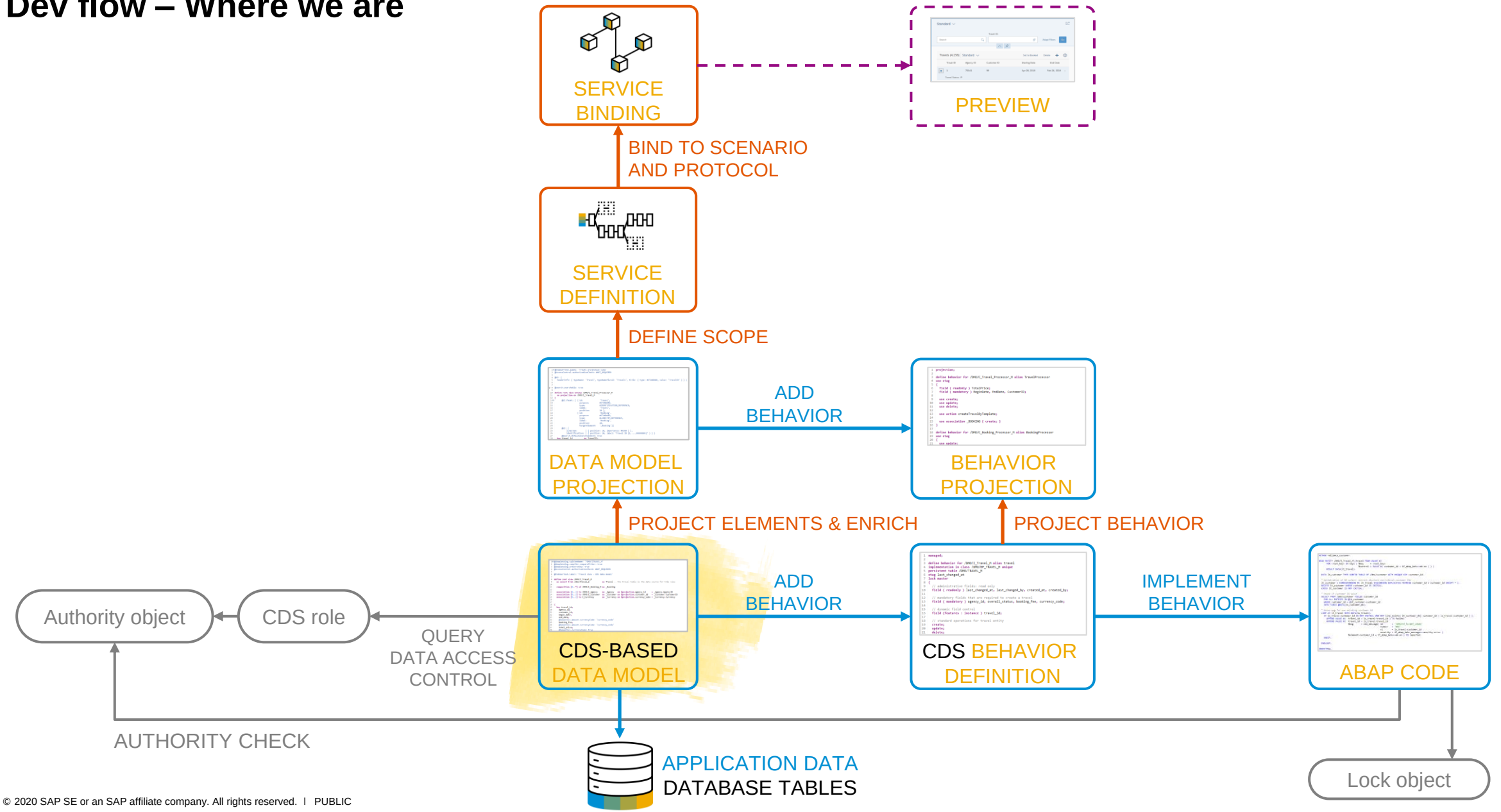
03 Defining and Implementing the Business Object Behavior

04 Creating the Business Object Projection

05 Building and Previewing the OData UI Service

Creating the CDS Data Model

Dev flow – Where we are



What we will achieve in this unit

Root key word

Composition

Associations

Field list with aliases

Data preview (F8)

*ZI_RAP_TRAVEL_U_1234 | ZI_RAP_BOOKING_U_1234

```
@EndUserText.label: 'Travel data'
@Metadata.ignorePropagatedAnnotations: true

define root view entity ZI_RAP_Travel_U_1234
  as select from /dmo/travel

  composition [0..*] of ZI_RAP_Booking_U_1234 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 [0..1] to I_Currency            as _Currency on $projection.CurrencyCode = _Currency.Currency

{
  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,
      status        as Status,
      @Semantics.user.createdBy: true
      createdby     as Createdby,
      @Semantics.systemDateTime.createdAt: true
      createdat     as Createdat,
      @Semantics.user.lastChangedBy: true
      lastchangedby as Lastchangedby,
      @Semantics.systemDateTime.lastChangedAt: true
      lastchangedat as Lastchangedat,

      /* associations */
      _Booking,
      _Agency,
      _Customer,
      _Currency
}
```

ZI_RAP_TRAVEL_U_1234

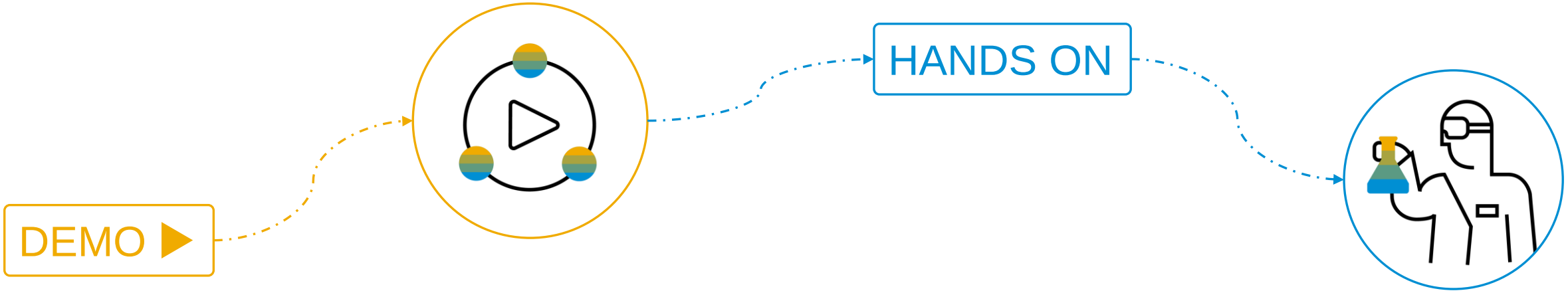
ZI_RAP_TRAVEL_U_1234

Raw Data

Filter pattern | 100 rows retrieved - 14 ms (partial result)

TravelID	AgencyID	CustomerID	BeginDate	EndDate	BookingFee
00000001	070041	000594	2020-05-09	2021-03-07	40.00
00000002	070007	000608	2020-05-09	2020-05-09	20.00
00000003	070046	000093	2020-05-09	2021-03-07	80.00
00000004	070042	000665	2020-05-09	2021-03-07	40.00
00000005	070007	000225	2020-05-09	2020-05-09	20.00
00000006	070049	000072	2020-05-09	2021-03-07	120.00
00000007	070046	000138	2020-05-09	2021-03-07	120.00
00000008	070012	000705	2020-05-09	2020-05-11	60.00
00000009	070032	000115	2020-05-09	2021-03-07	120.00

Demo



Creating the CDS Data Model

1. Define the BO composition tree for the Travel and Booking entities
2. Preview the existing data

Wrap-up

IN THIS UNIT, YOU LEARNED

- What a BO composition tree is and how you define it

NEXT UNIT

- Week 4 – Unit 3
  Defining and Implementing the Business Object Behavior

Further reading

Additional Material

ABAP RESTful APPLICATION PROGRAMMING MODEL INFORMATION PAGE

For more information, links to documentation, tutorials, and more, please visit the RAP at openSAP information page by following the link below

RAP at openSAP information page (week 4)

Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP®

Week 4: Dealing with Existing Code

Unit 3: Defining and Implementing the Business Object Behavior

Topics

Week 4

01 The Business Scenario

02 Creating the CDS Data Model

03 Defining and Implementing the Business Object Behavior

04 Creating the Business Object Projection

05 Building and Previewing the OData UI Service

What we will achieve in this unit – Behavior Definition

Implementation type unmanaged

Behaviour implementation classes

Standard operations: create, update and delete

Create enabled association

Specify fields that are mandatory or read-only

central mapping for legacy business logic

[Y08] ZI_RAP_TRAVEL_U_1234

```
1  unmanaged;
2
3  define behavior for ZI_RAP_Travel_U_1234 alias Travel
4  implementation in class zbp_i_rap_travel_u_1234 unique
5  lock master
6  etag master Lastchangedat
7  {
8    create;
9    update;
10   delete;
11   association _Booking { create; }
12
13   field ( read only ) TravelID;
14   field ( mandatory ) AgencyID, CustomerID, BeginDate, EndDate;
15
16   mapping for /DMO/TRAVEL control zsrap_travel_x_1234
17   {
18     TravelId = travel_id;
19     AgencyId = AGENCY_ID;
20     CustomerId = CUSTOMER_ID;
21     BeginDate = BEGIN_DATE;
22     EndDate = END_DATE;
23     BookingFee = BOOKING_FEE;
24     TotalPrice = TOTAL_PRICE;
25     CurrencyCode = CURRENCY_CODE;
26     Description = DESCRIPTION;
27     Status = STATUS;
28     Createdby = CREATEDBY;
29     Createdat = CREATEDAT;
30     Lastchangedby = LASTCHANGEDBY;
31     Lastchangedat = LASTCHANGEDAT;
32   }
33 }
```

```
35 define behavior for ZI_RAP_Booking_U_1234 alias Booking
36 implementation in class zbp_i_rap_booking_u_1234 unique
37 lock dependent by _Travel
38 etag dependent by _Travel
39 {
40   update;
41   delete;
42   association _Travel;
43
44   field ( read only ) TravelID, BookingID;
45   field ( mandatory ) BookingDate, CustomerID, CarrierId, ConnectionID, FlightDate;
46
47   mapping for /DMO/BOOKING control zsrap_booking_x_1234
48   {
49     TravelId = TRAVEL_ID;
50     BookingId = BOOKING_ID;
51     BookingDate = BOOKING_DATE;
52     CustomerId = CUSTOMER_ID;
53     CarrierId = CARRIER_ID;
54     ConnectionId = CONNECTION_ID;
55     FlightDate = FLIGHT_DATE;
56     FlightPrice = FLIGHT_PRICE;
57     CurrencyCode = CURRENCY_CODE;
58   }
59
60 }
```

What we will achieve in this unit – Behavior Implementation

Local handler class for buffer access

Implementation of standard operations create, update and delete

Implementation for read from buffer

Create by association

Local saver class to save data to database tables

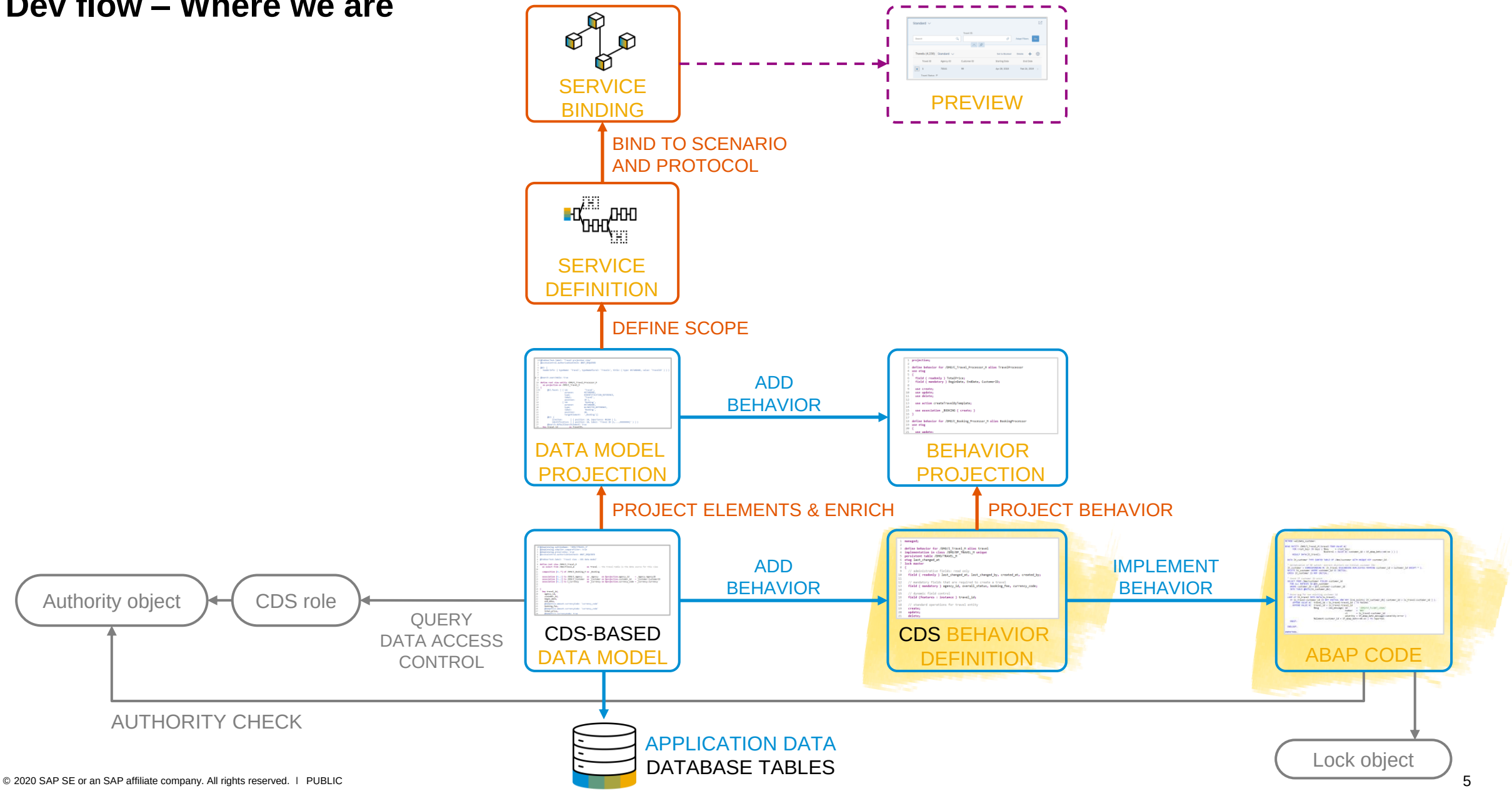
[Y08] ZBP_I_RAP_TRAVEL_U_1234

ZBP_I_RAP_TRAVEL_U_1234 ▸ LHC_TRAVEL ▸

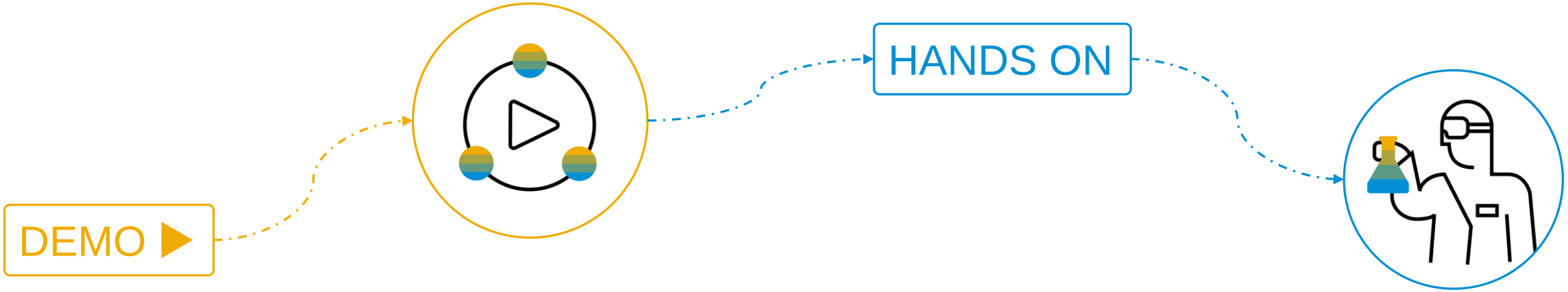
```
1  CLASS lhc_Travel DEFINITION INHERITING FROM cl_abap_behavior_handler.
2    PRIVATE SECTION.
3
4      METHODS create FOR MODIFY
5        IMPORTING entities FOR CREATE Travel.
6
7      METHODS delete FOR MODIFY
8        IMPORTING keys FOR DELETE Travel.
9
10     METHODS update FOR MODIFY
11       IMPORTING entities FOR UPDATE Travel.
12
13     METHODS lock FOR LOCK
14       IMPORTING keys FOR LOCK Travel.
15
16     METHODS read FOR READ
17       IMPORTING keys FOR READ Travel RESULT result.
18
19     METHODS cba_Booking FOR MODIFY
20       IMPORTING entities_cba FOR CREATE Travel\_Booking.
21
22     METHODS rba_Booking FOR READ
23       IMPORTING keys_rba FOR READ Travel\_Booking FULL result_requested RESULT result
24
25 ENDCLASS.
```

```
411 CLASS lsc_ZI_RAP_Travel_U_1234 DEFINITION INHERITING FROM cl_abap_behavior_saver.
412   PROTECTED SECTION.
413
414     METHODS check_before_save REDEFINITION.
415
416     METHODS finalize          REDEFINITION.
417
418     METHODS save              REDEFINITION.
419
420 ENDCLASS.
```

Dev flow – Where we are



Demo



Create and implement the transactional BO behavior

1. Create and define BO behavior definition for the Travel and Booking entities
2. Implement the BO behavior for the Travel and Booking entities
3. Create a unit test to test the BO implementation via EML

Wrap-up

IN THIS UNIT, YOU LEARNED

- How to define the BO behavior definition for the Travel and Booking entities and how to implement it

NEXT UNIT

- Week 4 – Unit 4
  Creating the Business Object Projection

Further reading

Additional Material

ABAP RESTful APPLICATION PROGRAMMING MODEL INFORMATION PAGE

For more information, links to documentation, tutorials, and more, please visit the RAP at openSAP information page by following the link below

RAP at openSAP information page (week 4)

Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP®

Week 4: Dealing with Existing Code

Unit 4: Creating the Business Object Projection

Topics

Week 4

01 The Business Scenario

02 Creating the CDS Data Model

03 Defining and Implementing the Business Object Behavior

04 Creating the Business Object Projection

05 Building and Previewing the OData UI Service

Creating the Business Object Projection

What we will achieve in this unit – Data Model Projection

Enable Search @Search.searchable: true

Allow use of MetaDataExtensions

Define value help

@Consumption. valueHelpDefinition

Define association as

redirected to composition child

Define association as

redirected to parent

[Y08] ZC_RAP_TRAVEL_U_1234

```
@EndUserText.label: 'Travel data'
@AccessControl.authorizationCheck: #CHECK
@Search.searchable: true
@Metadata.allowExtensions: true

define root view entity ZC_RAP_Travel_U_1234
  as projection on ZI_RAP_Travel_U_1234
{
      //ZI_RAP_TRAVEL_U_1234
  key TravelID,
      @Consumption.valueHelpDefinition: [ { entity: { name: '/DMO/I_Agency', element: 'AgencyID' } } ]
      @Search.defaultSearchElement: true
      AgencyID,
      @Consumption.valueHelpDefinition: [ { entity: { name: '/DMO/I_Customer', element: 'CustomerID' } } ]
      @Search.defaultSearchElement: true
      CustomerID,
      BeginDate,
      EndDate,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      BookingFee,
      @Semantics.amount.currencyCode: 'CurrencyCode'
      TotalPrice,
      @Consumption.valueHelpDefinition: [ { entity: { name: 'I_Currency', element: 'Currency' } } ]
      CurrencyCode,
      Description,
      Status,
      Createdby,
      Createdat,
      Lastchangedby,
      Lastchangedat,

      /* Associations */
      //ZI_RAP_TRAVEL_U_1234
      _Agency,
      _Booking : redirected to composition child ZC_RAP_Booking_U_1234,
      _Currency,
      _Customer
}
```

[Y08] ZC_RAP_BOOKING_U_1234

```
@EndUserText.label: 'Booking'
@AccessControl.authorizationCheck: #CHECK
@Search.searchable: true
@Metadata.allowExtensions: true

define view entity ZC_RAP_Booking_U_1234
  as projection on ZI_RAP_Booking_U_1234
{
      /* Associations */
      _Carrier,
      _Customer,
      _Travel : redirected to parent ZC_RAP_Travel_U_1234
}
```

Creating the Business Object Projection

What we will achieve in this unit – Metadata Extension

@UI.headerInfo

@UI.presentation Variant
defines default sort order

@UI.facet annotations

[Y08] ZC_RAP_TRAVEL_U_1234

```
@Metadata.layer: #CUSTOMER
@UI: {
  headerInfo: {
    typeName: 'Travel',
    typeNamePlural: 'Travels',
    title: {
      type: #STANDARD,
      label: 'Travel',
      value: 'TravelID'
    }
  },
  presentationVariant: [ { sortOrder: [ { by:         'TravelID',
                                          direction:  #DESC } ] } ]
}
annotate view ZC_RAP_Travel_U_1234 with
{
  @UI.facet: [ { id:            'Travel',
                 purpose:       #STANDARD,
                 type:          #IDENTIFICATION_REFERENCE,
                 label:         'Travel',
                 position:      10 },
               { id:            'Booking',
                 purpose:       #STANDARD,
                 type:          #LINEITEM_REFERENCE,
                 label:         'Booking',
                 position:      20,
                 targetElement: '_Booking'}]

  @UI: { lineItem:       [ { position: 10 } ],
         identification: [ { position: 10 } ]  }
  TravelID;

  @UI: { lineItem:       [ { position: 20 } ],
         identification: [ { position: 20 } ]  }
  AgencyID;

  @UI: { lineItem:       [ { position: 30 } ],
         identification: [ { position: 30 } ]  }
  CustomerID;

```

What we will achieve in this unit – Behavior Projection

Define that etag is used

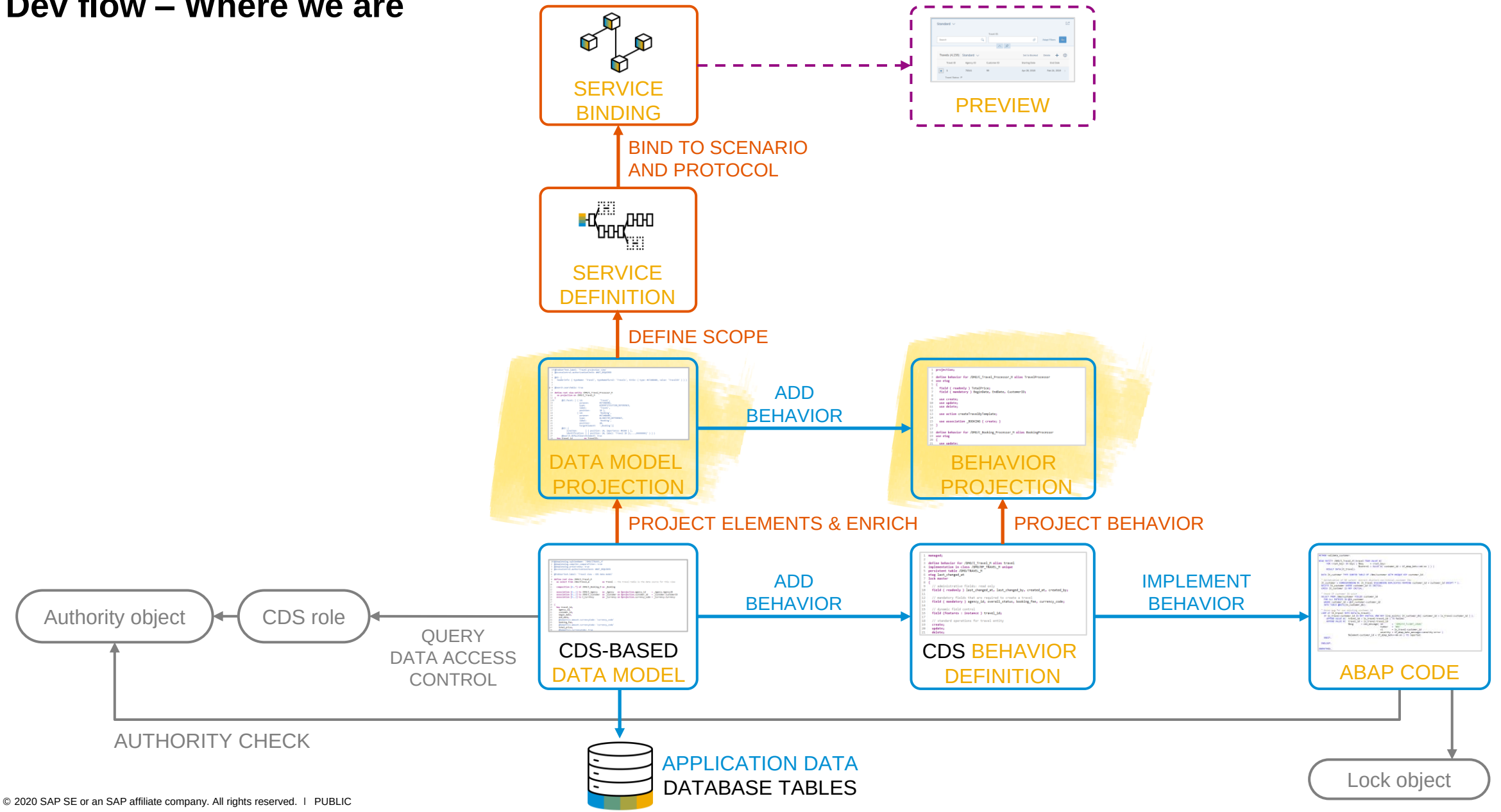
Define which standard operations are used

Define that create enabled association is used

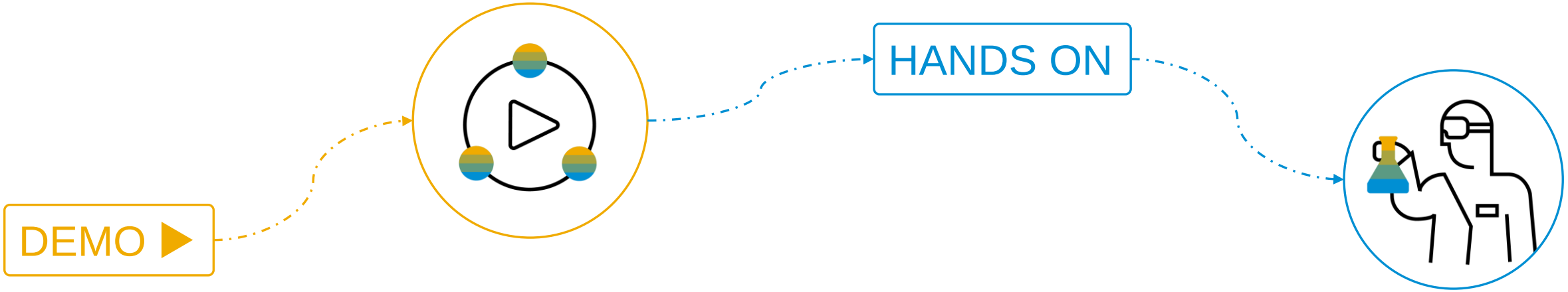
[Y08] ZC_RAP_TRAVEL_U_1234

```
1  projection;
2
3  define behavior for ZC_RAP_Travel_U_1234 alias Travel
4  use etag
5  {
6    use create;
7    use update;
8    use delete;
9
10   use association _Booking { create; }
11 }
12
13 define behavior for ZC_RAP_Booking_U_1234 alias Booking
14 use etag
15 {
16   use update;
17   use delete;
18
19   use association _Travel;
20 }
```

Dev flow – Where we are



Demo



Define the business object BO projection

1. Define the BO data model projection for the Travel and Booking entities
2. Define Metadata Extensions for the BO data model projection
3. Define the BO behavior projection for the Travel and Booking entities

Wrap-up

IN THIS UNIT, YOU LEARNED

- How to define the BO data model projection and the BO behavior projection for the Travel and Booking entities

NEXT UNIT

- Week 4 – Unit 5
  Building and Previewing the OData UI Service

Further reading

Additional Material

ABAP RESTful APPLICATION PROGRAMMING MODEL INFORMATION PAGE

For more information, links to documentation, tutorials, and more, please visit the RAP at openSAP information page by following the link below

RAP at openSAP information page (week 4)

Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP®

Week 4: Dealing with Existing Code

Unit 5: Building and Previewing the OData UI Service

Topics

Week 4

What we will achieve in this unit – Service Definition

Define scope of service

[Y08] ZUI_RAP_TRAVEL_U_O2_1234

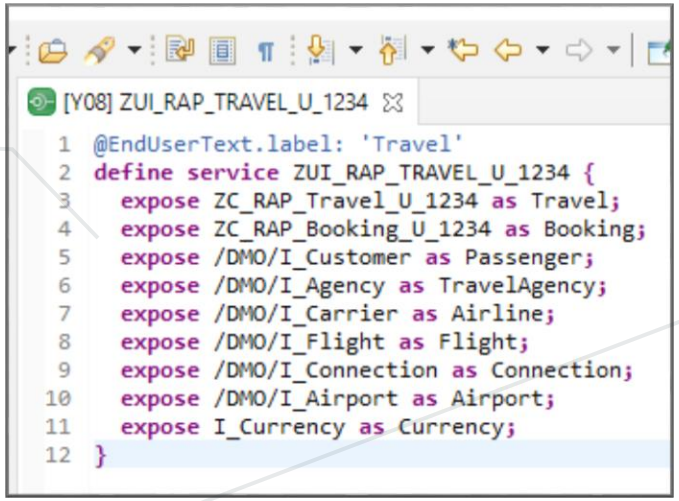
```
1  @EndUserText.label: 'Travel'
2  define service ZUI_RAP_TRAVEL_U_O2_1234 {
3    expose ZC_RAP_Travel_U_1234 as Travel;
4    expose ZC_RAP_Booking_U_1234 as Booking;
5    expose /DMO/I_Customer as Passenger;
6    expose /DMO/I_Agency as TravelAgency;
7    expose /DMO/I_Carrier as Airline;
8    expose /DMO/I_Flight as Flight;
9    expose /DMO/I_Connection as Connection;
10   expose /DMO/I_Airport as Airport;
11   expose I_Currency as Currency;
12 }
```

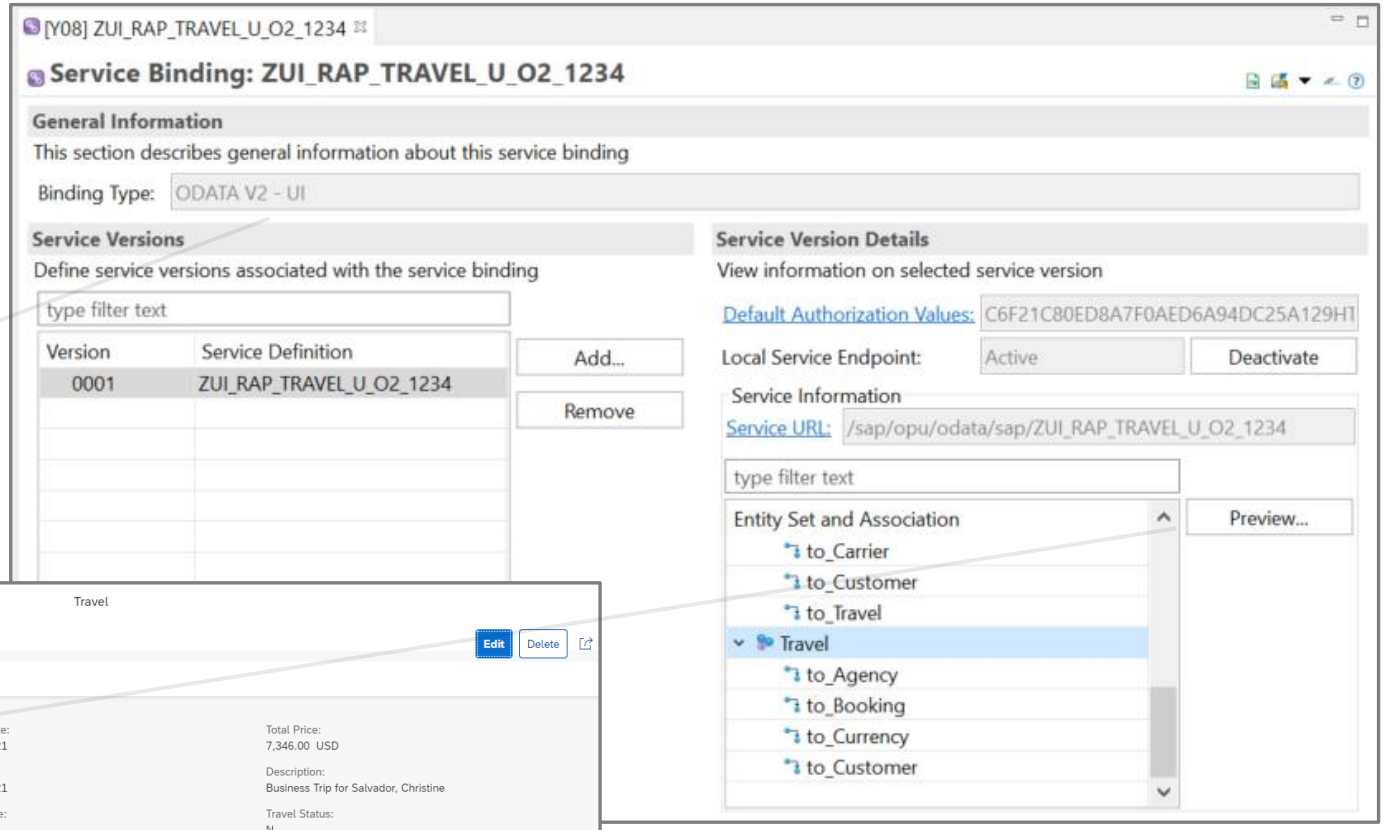
What we will achieve in this unit – Service Definition, Service Binding, Preview

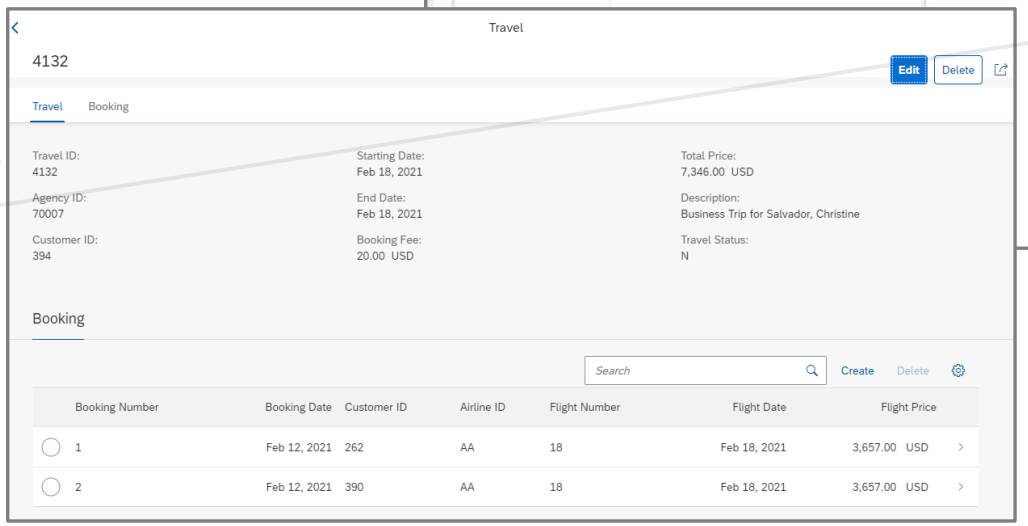
Define scope of service

Define binding type

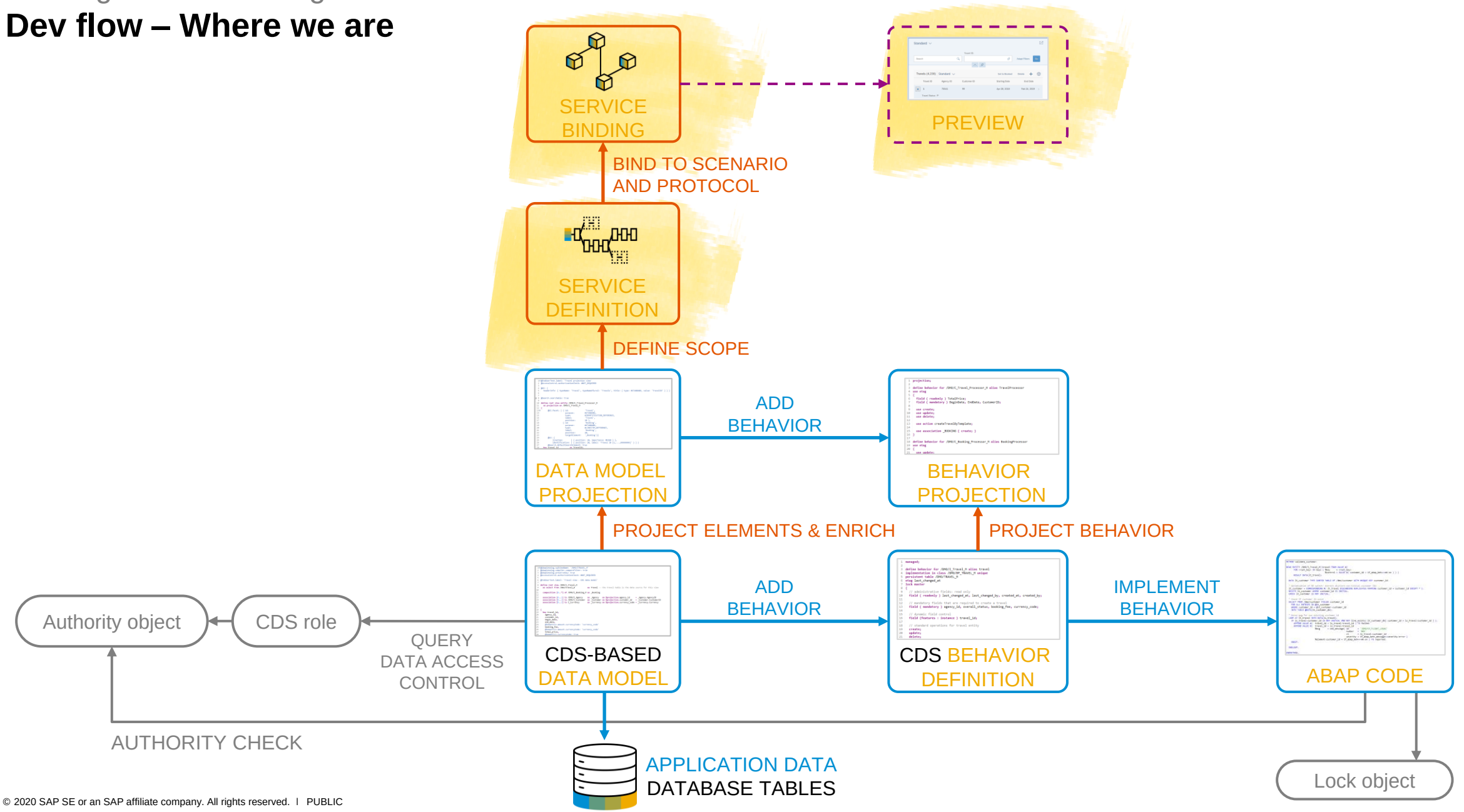
SAP Fiori Elements Preview



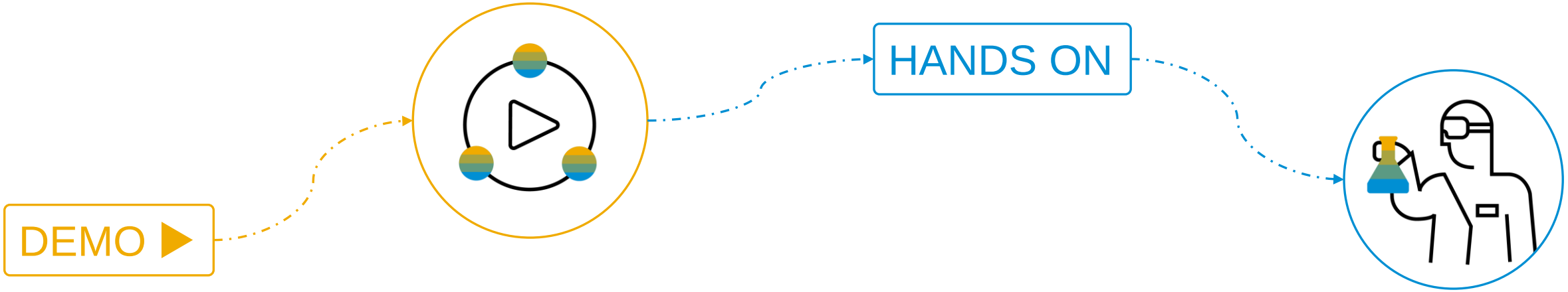




Dev flow – Where we are



Demo



Create and preview the OData UI service

1. Create the service definition from the data model projection
2. Create the service binding
3. Preview your SAP Fiori elements app

Wrap-up

IN THIS UNIT, YOU LEARNED

- How to create the service definition from the data model projection
- How to create the service binding and test the service

NEXT UNIT

- Week 5
  Service Consumption and Web APIs

Further reading

Additional Material

ABAP RESTful APPLICATION PROGRAMMING MODEL INFORMATION PAGE

For more information, links to documentation, tutorials, and more, please visit the RAP at openSAP information page by following the link below

RAP at openSAP information page (week 4)

Thank you.

Contact information:

open@sap.com

Follow all of SAP

www.sap.com/contactsap

THE BEST RUN SAP®