SAP
Development
Learning

ABAP RESTful Application Programming Model

Know Your Trainers

SAP Development Learning

Pavan Kumar Reddy

Paulami Mitra

Sivapriya

Parmeet Singh

Learning Objectives

SAP Development Learning

At the end of this course, you will be able to:

- Define data modeling with CDS views.
- Implement the backend part of the transactional application using managed and unmanaged implementation.
- Enhance the transactional application with the DRAFT concept for better document handling
- Create and expose OData services.
- Consuming Business Object using Entity Manipulation Language (EML).
- Testing behavior implementation using EML.

ABAP RESTful Application Programming Model

Table of Content

SAP Development Learning

SAP Development Learning

Development Environment

ABAP Development Tools(ADT) in Eclipse

SAP Development Learning

MODERN DEVELOPMENT TOOLSET

Fully Eclipse-based
Syntax check, Code completion
Syntax highlighting, Pretty printing
Navigation, Search, Quick Fixes

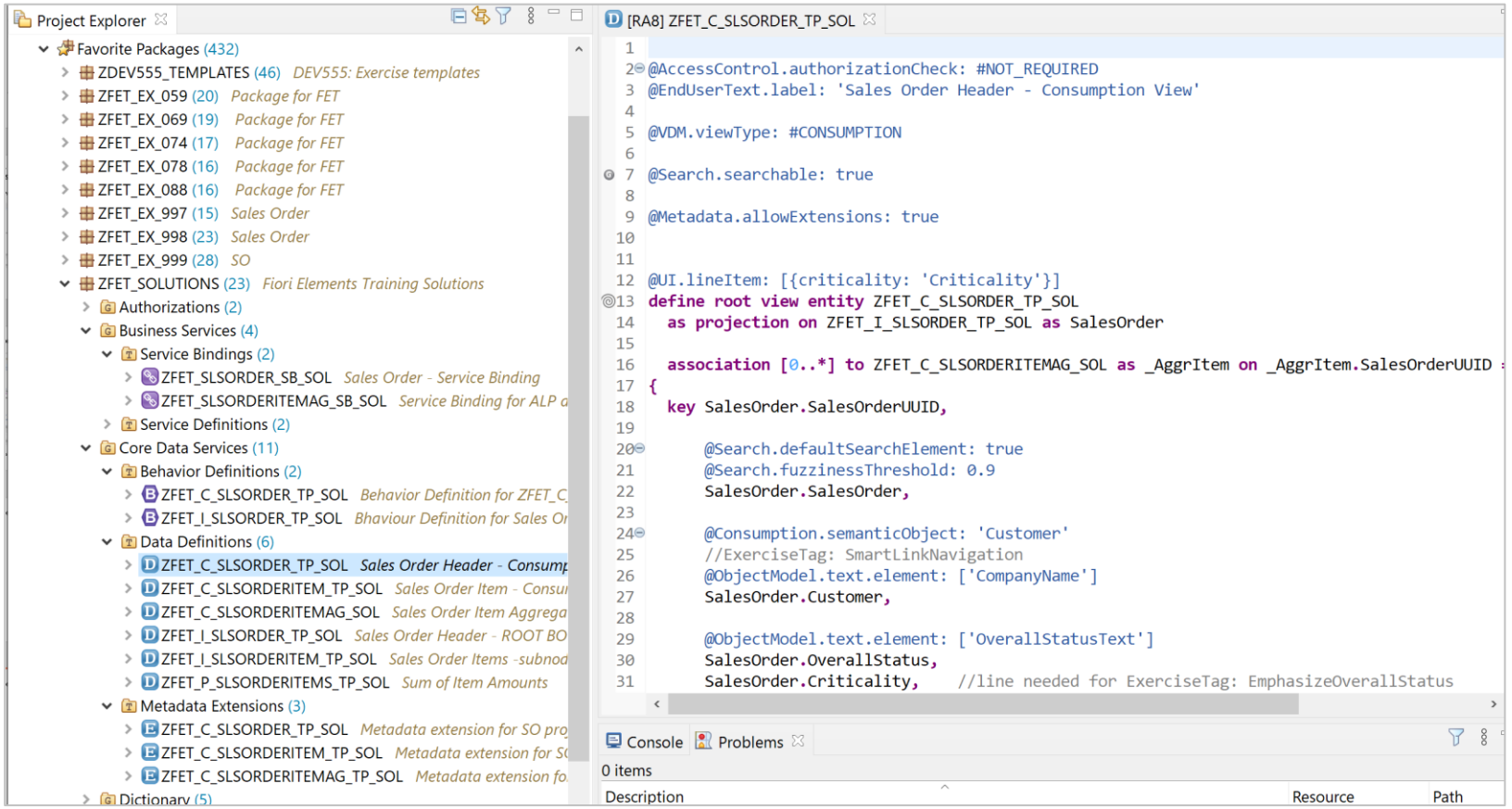
QUALITY ASSURANCE

Static code checks (ATC, CVA) with remote and local scenarios
Unit testing incl. isolation frameworks
Test seams and injections

SUPPORTABILITY

Debugging, profiling
Static and dynamic logging
Runtime monitoring and analysis

HIGH DEVELOPER PRODUCTIVITY WITH ABAP DEVELOPMENT TOOLS (ADT)

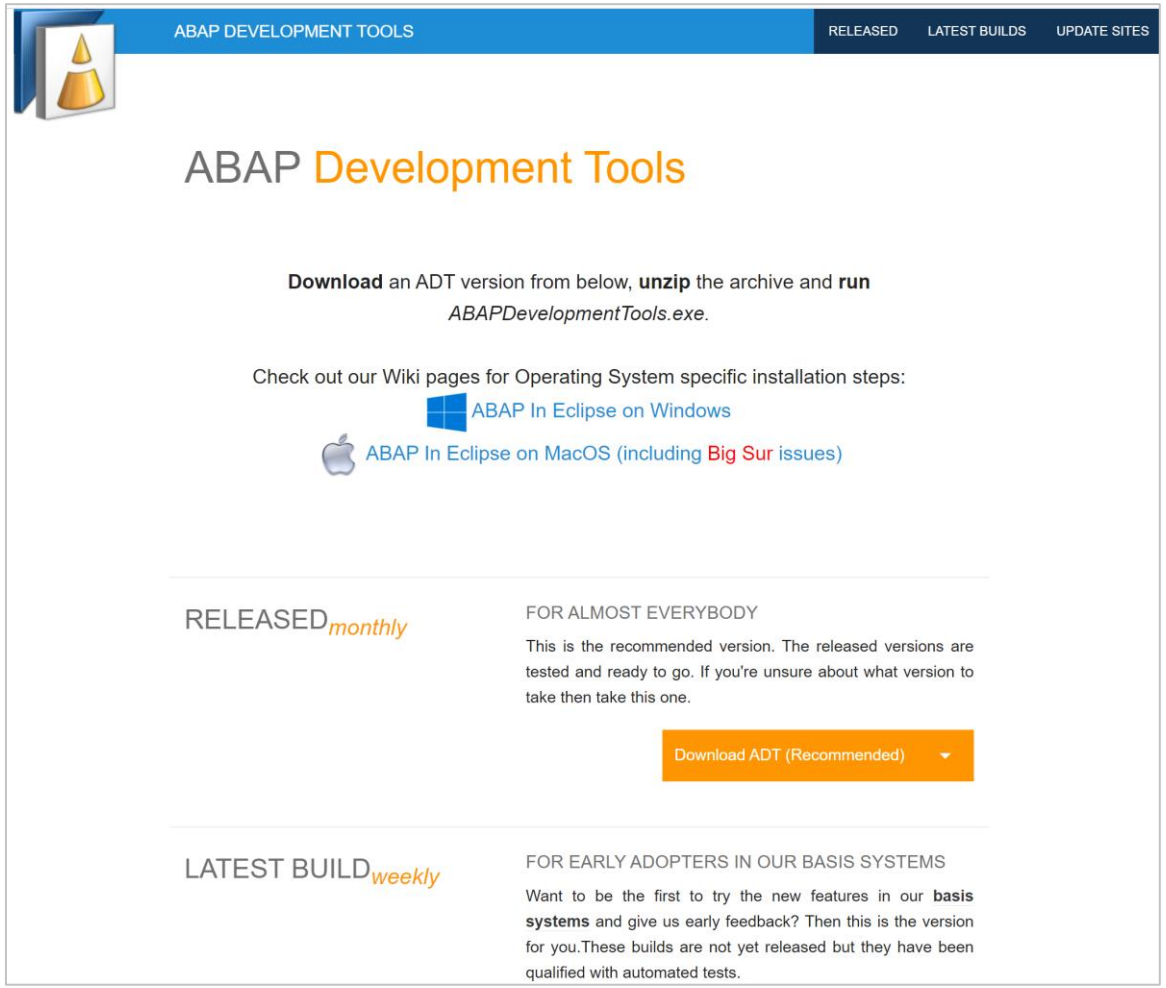


ATC: ABAP Test Cockpit
CVA: Code Vulnerability Analyzer

Preparing Your ABAP Development Environment

Installing and connecting ADT to the ABAP environment

SAP Development Learning



Access to ABAP tools

Instructions about the local installation of Eclipse and ADT available on the central update site for all Eclipse-based SAP development tools.

https://adt.wdf.sap.corp/#released

Training system

SAP Development Learning

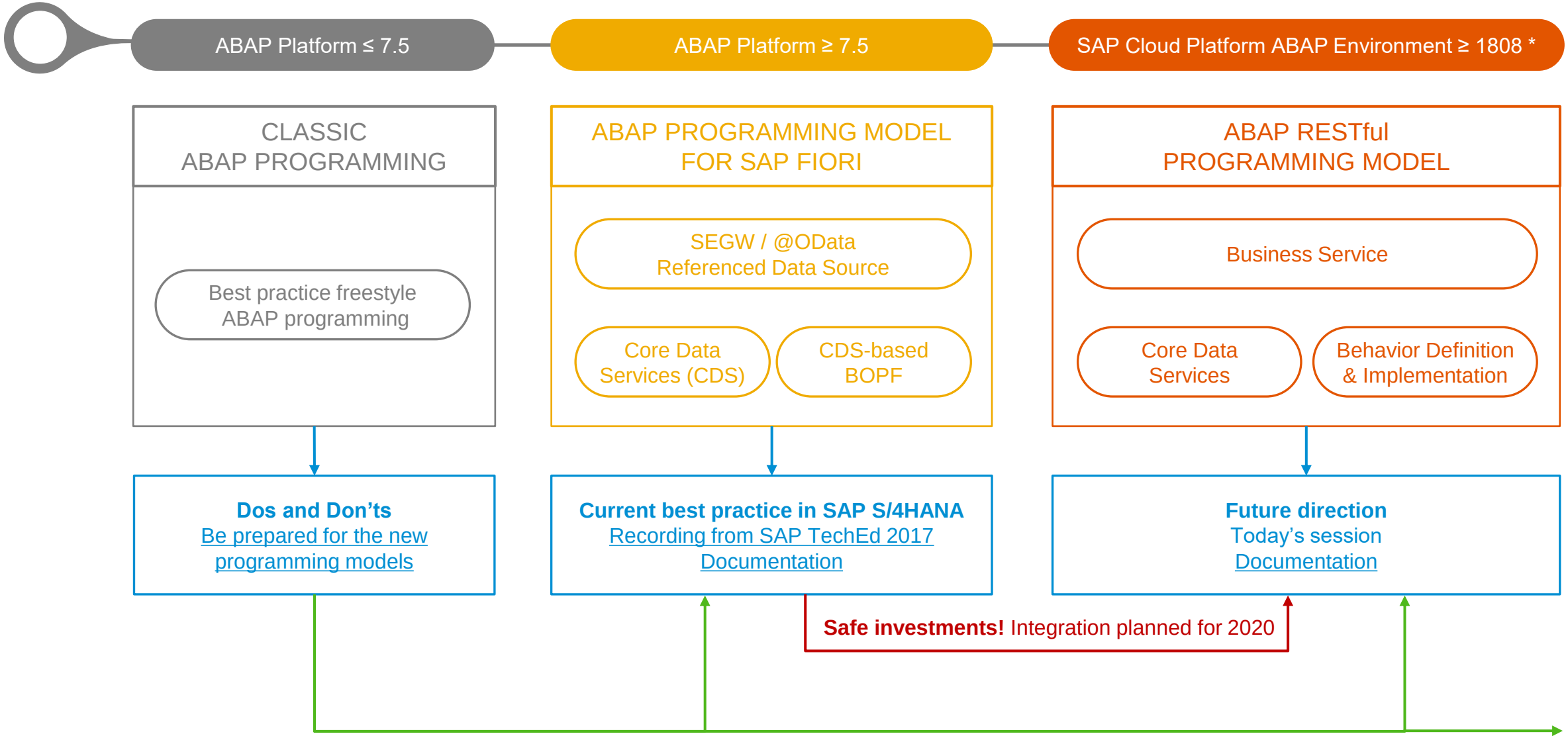
System: RA8
Client: 100
User Name: Refer table in the MS-Teams
Password: Welcome123

SAP
Development
Learning

RAP Overview

Evolution of the ABAP programming model

SAP Development Learning



* Reduced scope available on-premise as of SAP S/4HANA 1909

Architecture Overview

The big picture

SAP Development Learning

SERVICE CONSUMPTION

BUSINESS SERVICES PROVISIONING

DATA MODELING & BEHAVIOR

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 OBJECT PROJECTION

- CDS: Projection views
- BDEF: Behavior projection
- ABAP: Behavior implementation

BUSINESS OBJECTS

- CDS: Data modeling
- BDEF: Behavior definition
- ABAP: Behavior implementation

QUERIES

- CDS: Data modeling

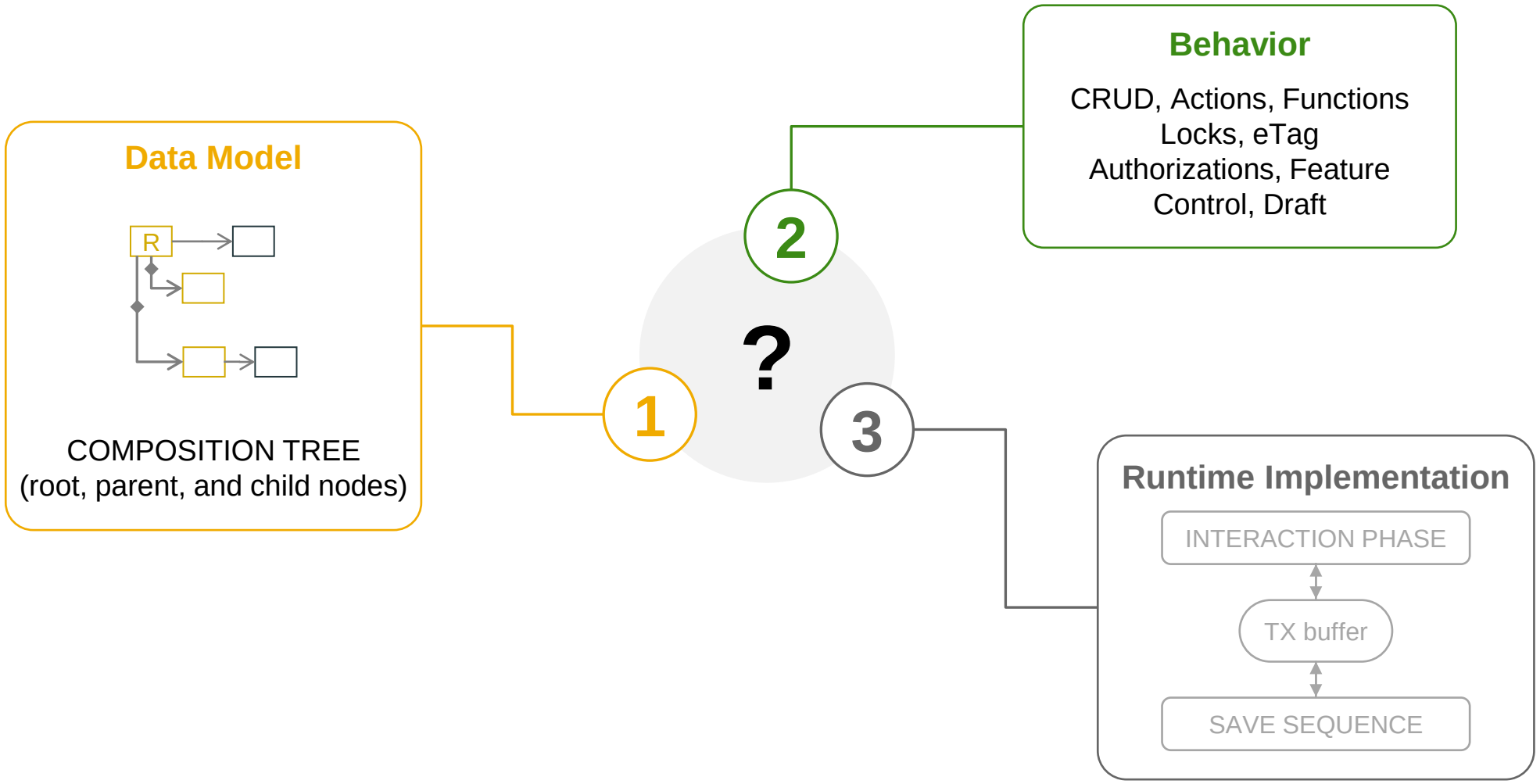
SAP
Development
Learning

RAP Business Objects

Architecture Overview

What is a business object?

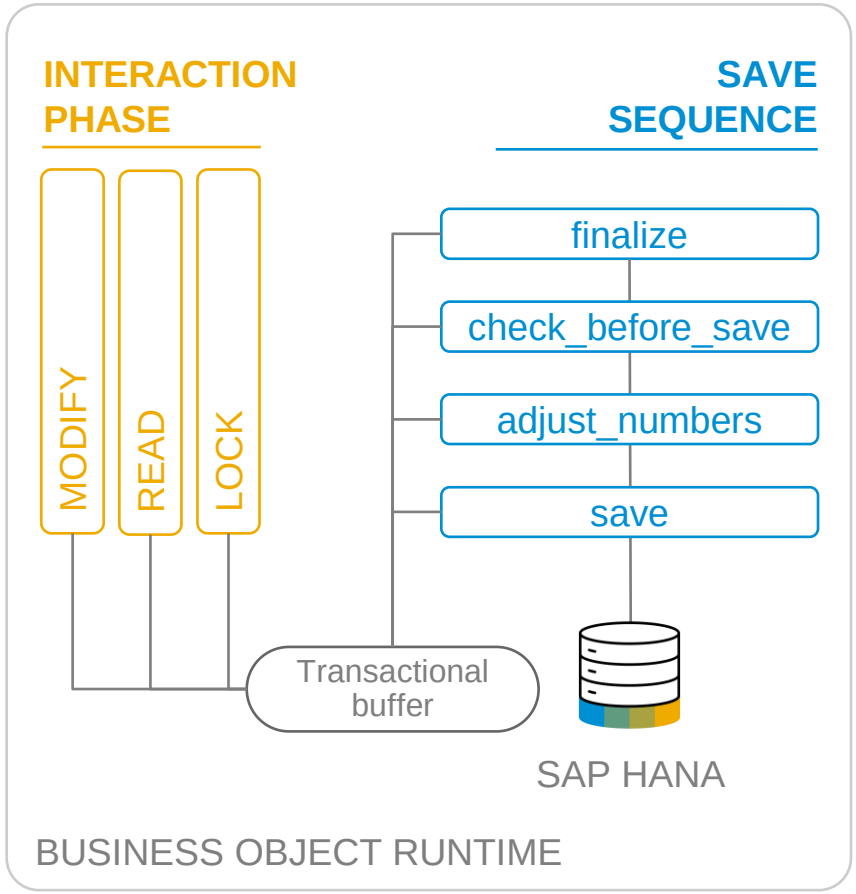
SAP Development Learning



Architecture Overview

Business object runtime implementation types

SAP Development Learning



UNMANAGED

For brownfield developments with available application code for interaction phase, transactional buffer, and save sequence

➔ Developers in charge of BO runtime: CRUD operations

➔ Adapters needed to integrate the existing code

MANAGED

For greenfield developments with standard implementation (opt. unmanaged appl. components: DB tables, lock/PFCG object, update task FM)

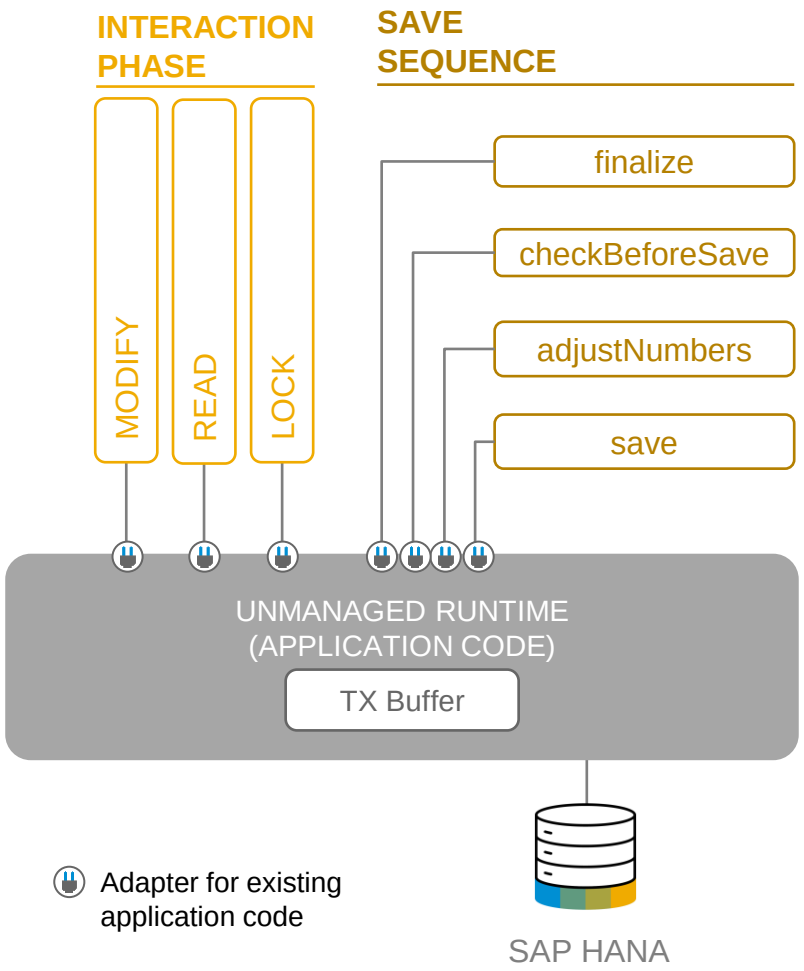
➔ Standard CRUD operations work out-of-the-box

➔ Developers add BO-specific business logic

RAP Business Objects

Unmanaged Implementation

SAP Development Learning



Application coding

- Already available
- For interaction phase, transactional buffer and save sequence
- Decoupled from UI technology
- Same code typically running in different channels (e.g. DYNP, WDA,…)

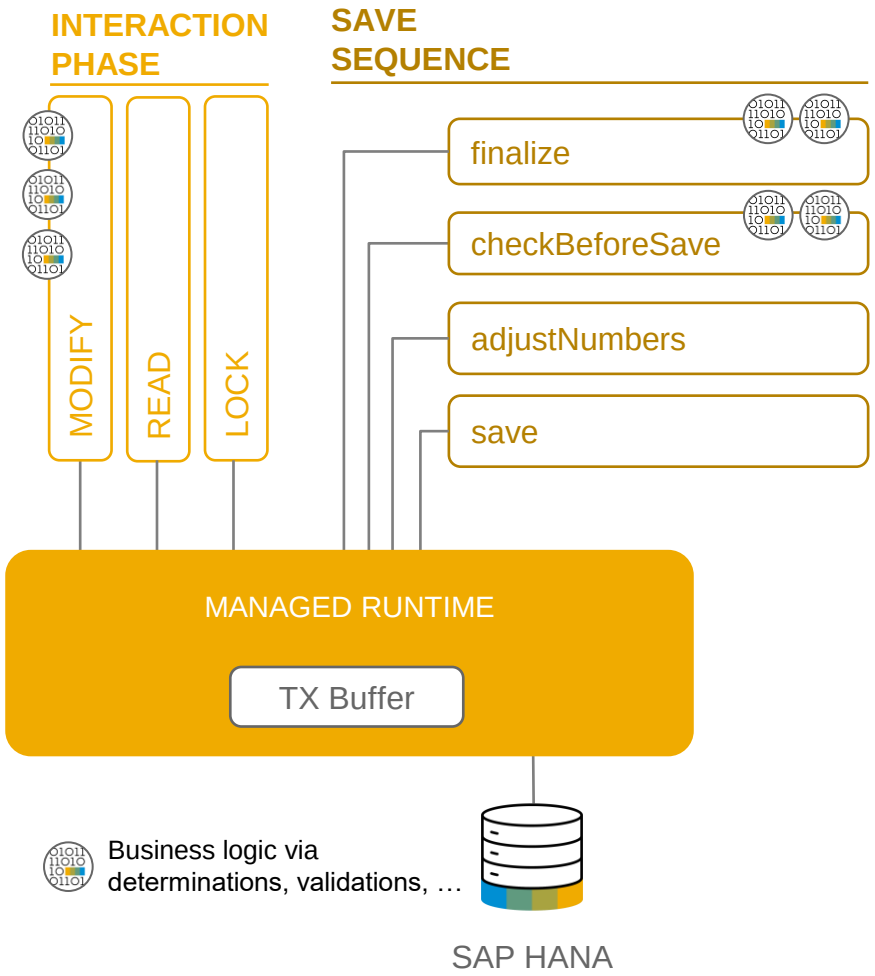
Examples

- Sales Order, Purchase Order

RAP Business Objects

Managed Implementation

SAP Development Learning



Application coding

- Not yet available or fine granular reusable code available
- Technical implementation tasks taken over by BO infrastructure
- Developer focus on business logic, implemented via code exits: determinations, validation, actions,…

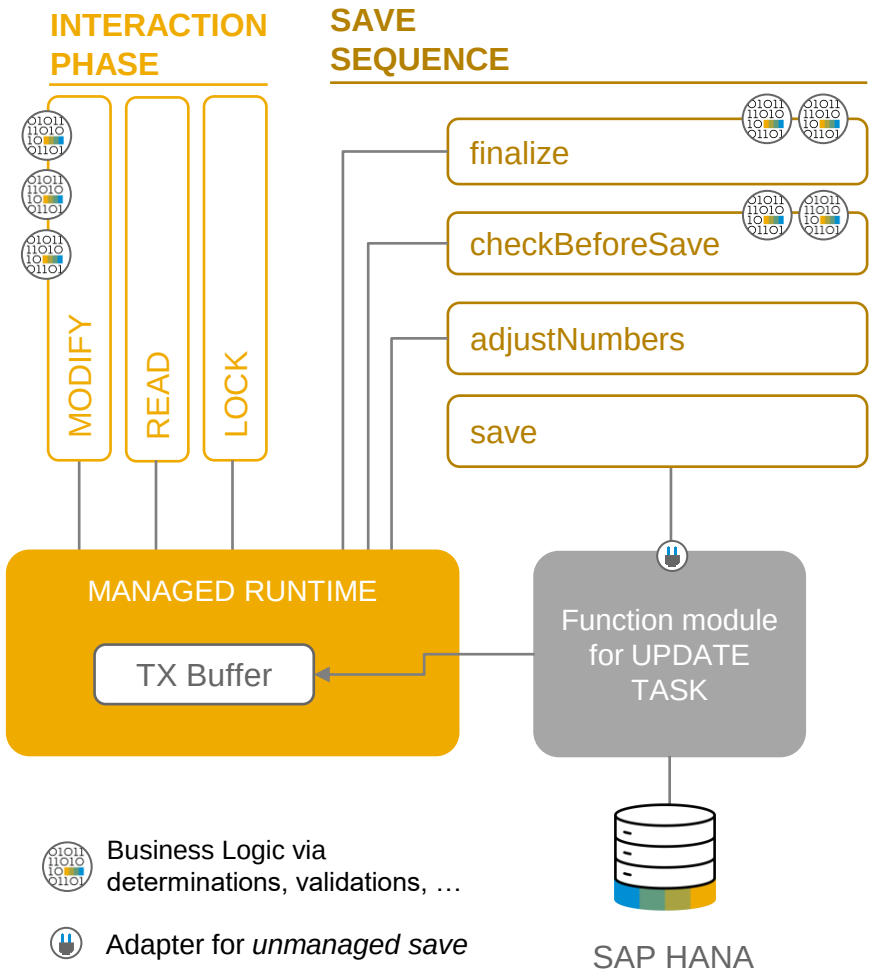
Examples

- New applications in SAP BTP, ABAP environment

Business Objects

Managed Impl. with Unmanaged Appl. Components

SAP Development Learning



Application coding

- “update-task function module” available
- Coding for interaction phase not available
  (e.g. highly coupled in older UI technology: DYNP - PBO / PAI)
- Technical implementation aspects to be taken over
  by BO infrastructure

Examples

- Business Partner, Product

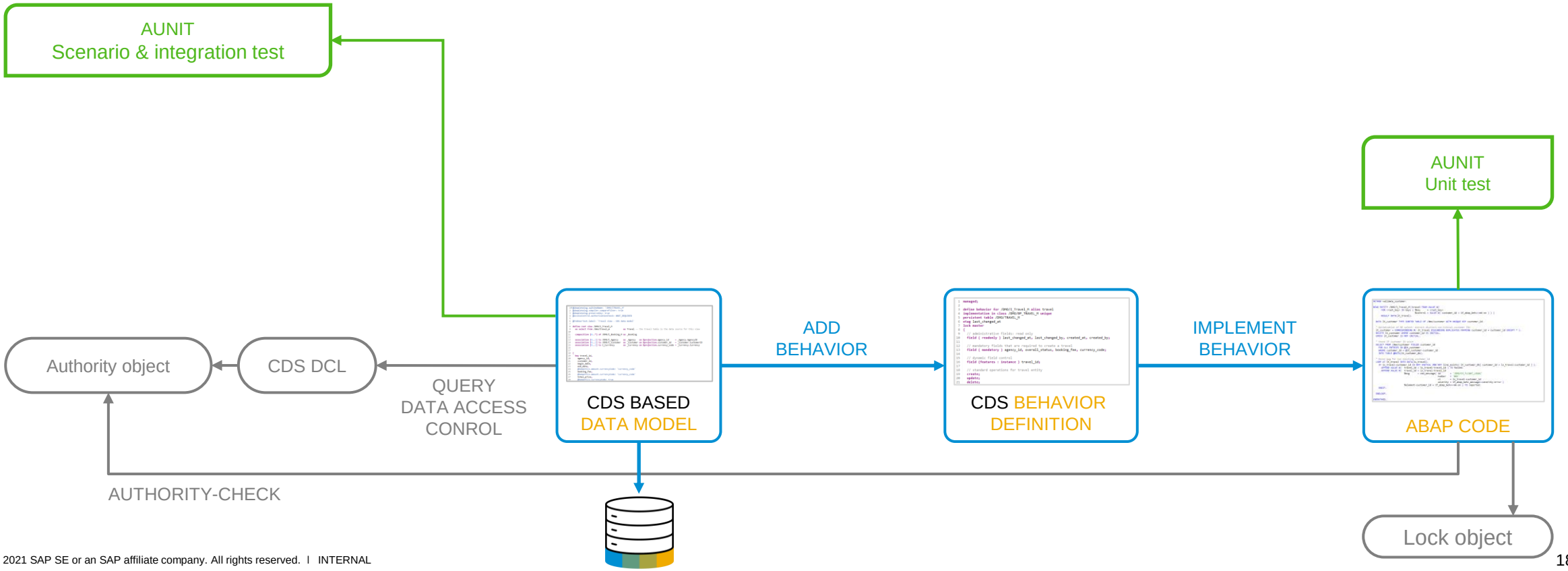
Optional unmanaged application components

- Update task FM: *unmanaged save*
- Own lock object: *unmanaged lock*
- Mapping between old and new world (e.g. DB tables): type mapping
- Old PFCG object: *authorization master*

ABAP RESTful Programming Model – Development flow

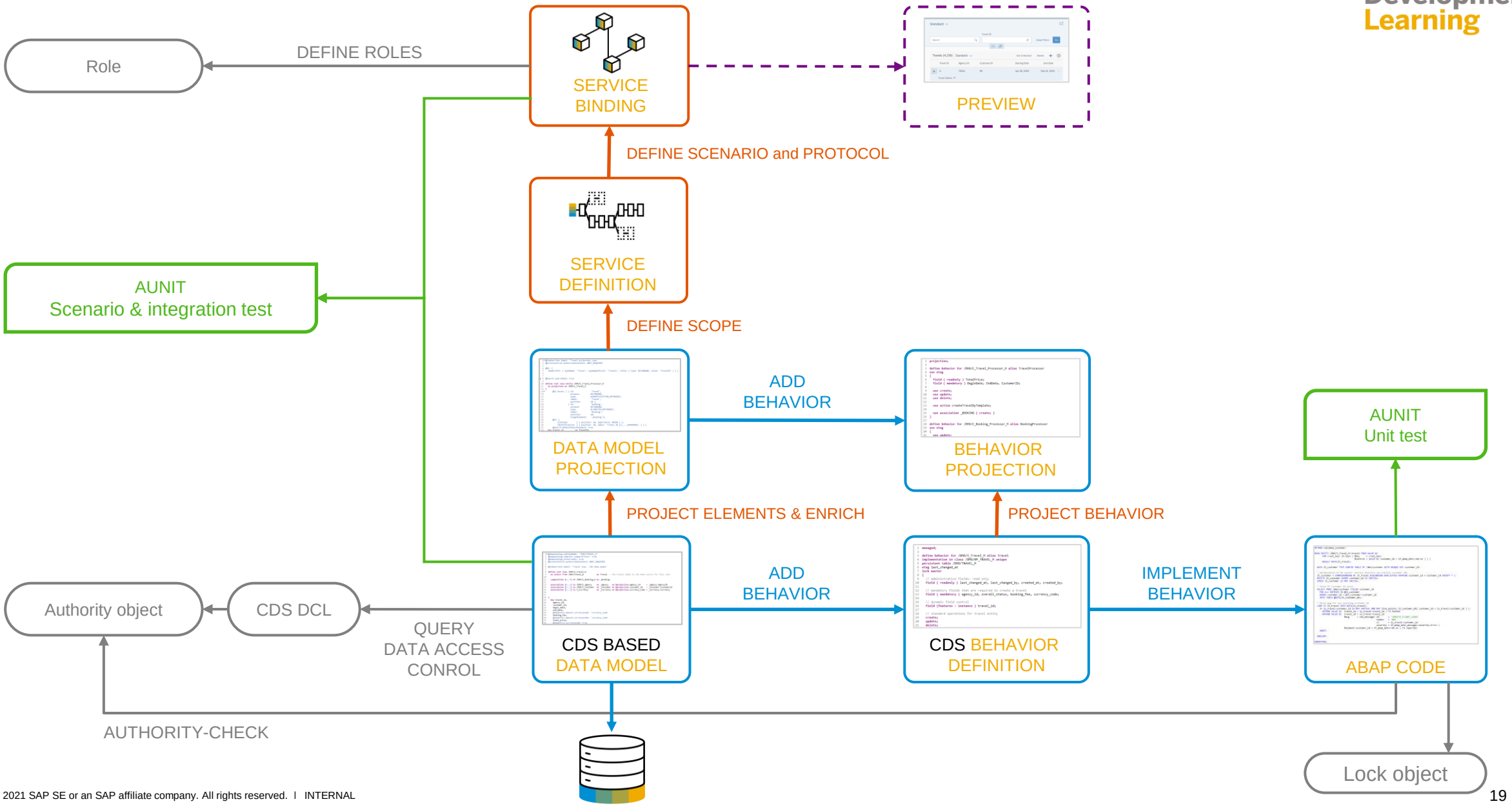
SAP Development Learning

Read Only Application



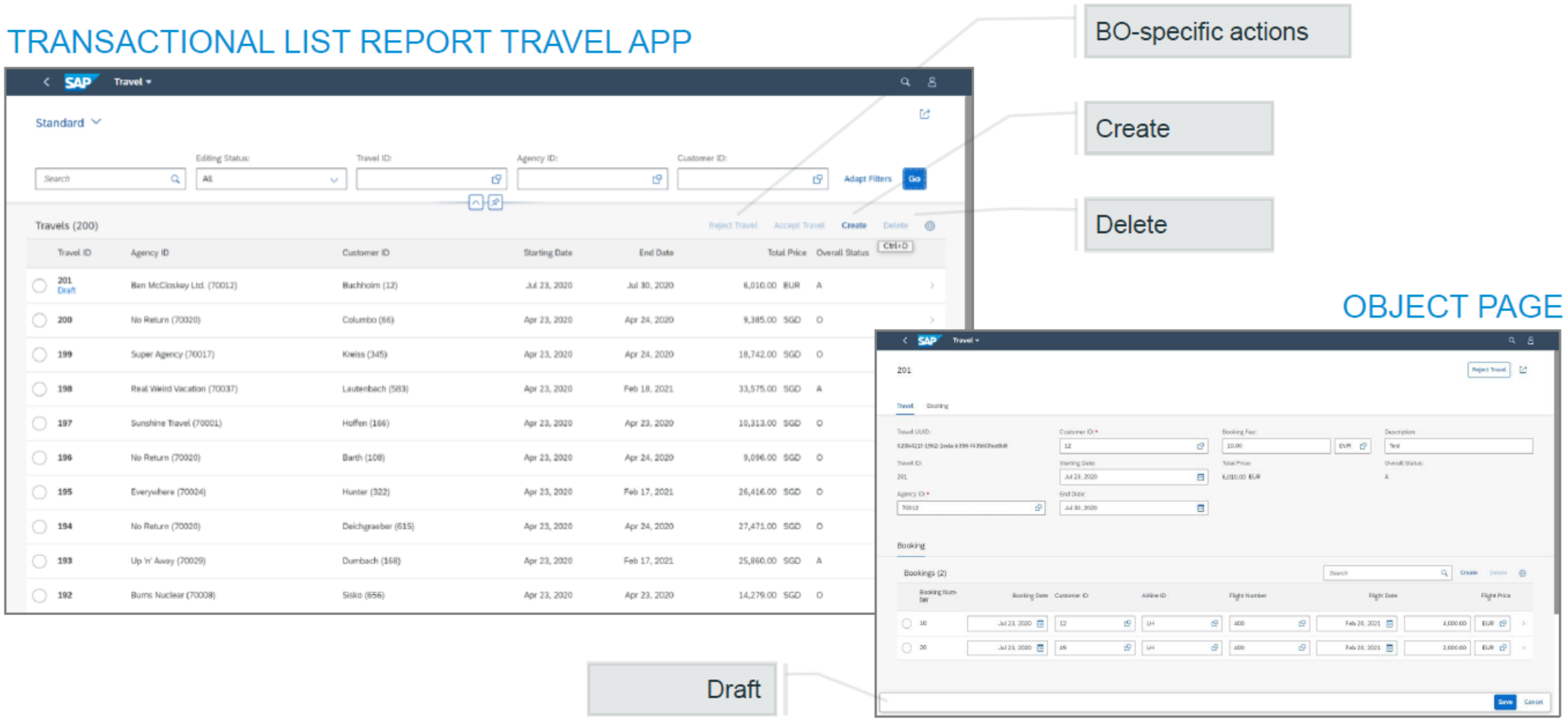
Development flow – Transactional Application

SAP Development Learning



The Enhanced Business Scenario

Resulting SAP Fiori elements app



Defining the Basic Business Object Behavior

SAP Development Learning

Define table for persistency

Read-only fields

Mapping between entity and persistent table

[RA8] ZFET_I_SLSORDER_TP_SOL

```
1  managed implementation in class zcl_fet_i_slsorder_tp_sol unique;
2  with draft;
3
4  define behavior for ZFET_I_SLSORDER_TP_SOL alias SalesOrder
5  persistent table ZFET_SO_SOL
6  draft table ZFET_D_SO_SOL
7
8  lock master
9  total etag LastChangedAt // entire business object
10 //authorization master ( instance )
11 etag master LocalLastChangedAt
12 {
13   create;
14   update;
15   delete ( features : instance ) ;
16   association _Item { create; with draft; }
17
18   field ( readonly, numbering:managed ) SalesOrderUUID;
19
20   field ( readonly ) SalesOrder, Created_at, LocalLastChangedBy, Lo
21
22   mapping for ZFET_SO_SOL
23   {
24    SalesOrderUUID       = salesorderuuid;
25    SalesOrder           = salesorder;
26    Customer             = customer;
27    OverallStatus        = overallstatus;
28    Created_by_loginname = created_by_loginname;
29    Created_at           = created_at;
30    LocalLastChangedBy   = changed_by_loginname;
31    LocalLastChangedAt   = changed_at;
32    LastChangedAt        = last_changed_at;
```

Creating the Business Object Behavior Projection

SAP Development Learning

Projected features from the base behavior definition

[RA8] ZFET_C_SLSORDER_TP_SOL

```
1  projection;
2  use draft;
3
4  define behavior for ZFET_C_SLSORDER_TP_SOL //alias <alias_name>
5  use etag
6  {
7    use create;
8    use update;
9    use delete;
10
11   use association _Item { create; with draft; }
12
13
14   use action setToDelivered;
15
16 }
17
18 define behavior for ZFET_C_SLSORDERITEM_TP_SOL //alias <alias_name
19 use etag
20 {
21   use update;
22   use delete;
23
24   use association _SalesOrder { with draft; }
25 }
```

Enhancing the Business Object Behavior with App-Specific Logic

Example

SAP Development Learning

Definition of actions

Definition of determinations

[RA8] ZFET_I_SLSORDER_TP_SOL

```
lock master
total etag LastChangedAt // entire business object
//authorization master ( instance )
etag master LocalLastChangedAt
{
  create;
  update;
  delete ( features : instance ) ;
  association _Item { create; with draft; }

  field ( readonly, numbering:managed ) SalesOrderUUID;

  field ( readonly ) SalesOrder, Created_at, LocalLastChangedBy, LocalLa

  mapping for ZFET_SO_SOL
  {
   SalesOrderUUID       = salesorderuuid;
   SalesOrder           = salesorder;
   Customer             = customer;
   OverallStatus        = overallstatus;
   Created_by_loginname = created_by_loginname;
   Created_at           = created_at;
   LocalLastChangedBy   = changed_by_loginname;
   LocalLastChangedAt   = changed_at;
   LastChangedAt        = last_changed_at;

  }

  determination setCriticality on modify {  field OverallStatus; }
  determination setSalesOrderID on save { create; }
  //dynamic action control
  action ( features : instance ) setToDelivered result [1] $self;
```

[RA8] ZFET_C_SLSORDER_TP_SOL

```
projection;
use draft;

define behavior for ZFET_C_SLSORDER_TP_SOL //alias
use etag
{
  use create;
  use update;
  use delete;

  use association _Item { create; with draft; }


  use action setToDelivered;

}

define behavior for ZFET_C_SLSORDERITEM_TP_SOL //ali
use etag
{
  use update;
  use delete;

  use association _SalesOrder { with draft; }
}
```

Projection of actions

SAP
Development
Learning

RAP Business Services

Architecture Overview

What is a business service?

SAP Development Learning

Service Binding

Protocol: OData V2 or OData V4?

Scenario: Web API or UI?

Web API Lifecycle

Release

Versioning

Deprecation

2

3

?

1

4

Tool Support

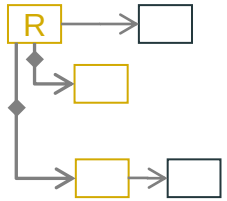
UI: SAP Fiori elements preview

Web API: Swagger UI preview

Service Definition: which entities?

BO Projection View: which elements?

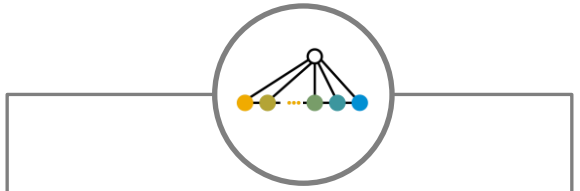
Behavior Projection: which behavior?



Architecture Overview

SAP Fiori UI and Web API consumption

SAP Development Learning



	SAP FIORI UI	Web API
UI annotation exposure	Required	Not required
Value help	Required	Not required
Dynamic feature control	Required	Not required
Draft	Required	Not required
Content IDs for complex $batch requests	Not required	Required
Release, versioning, and deprecation	Not required	Required

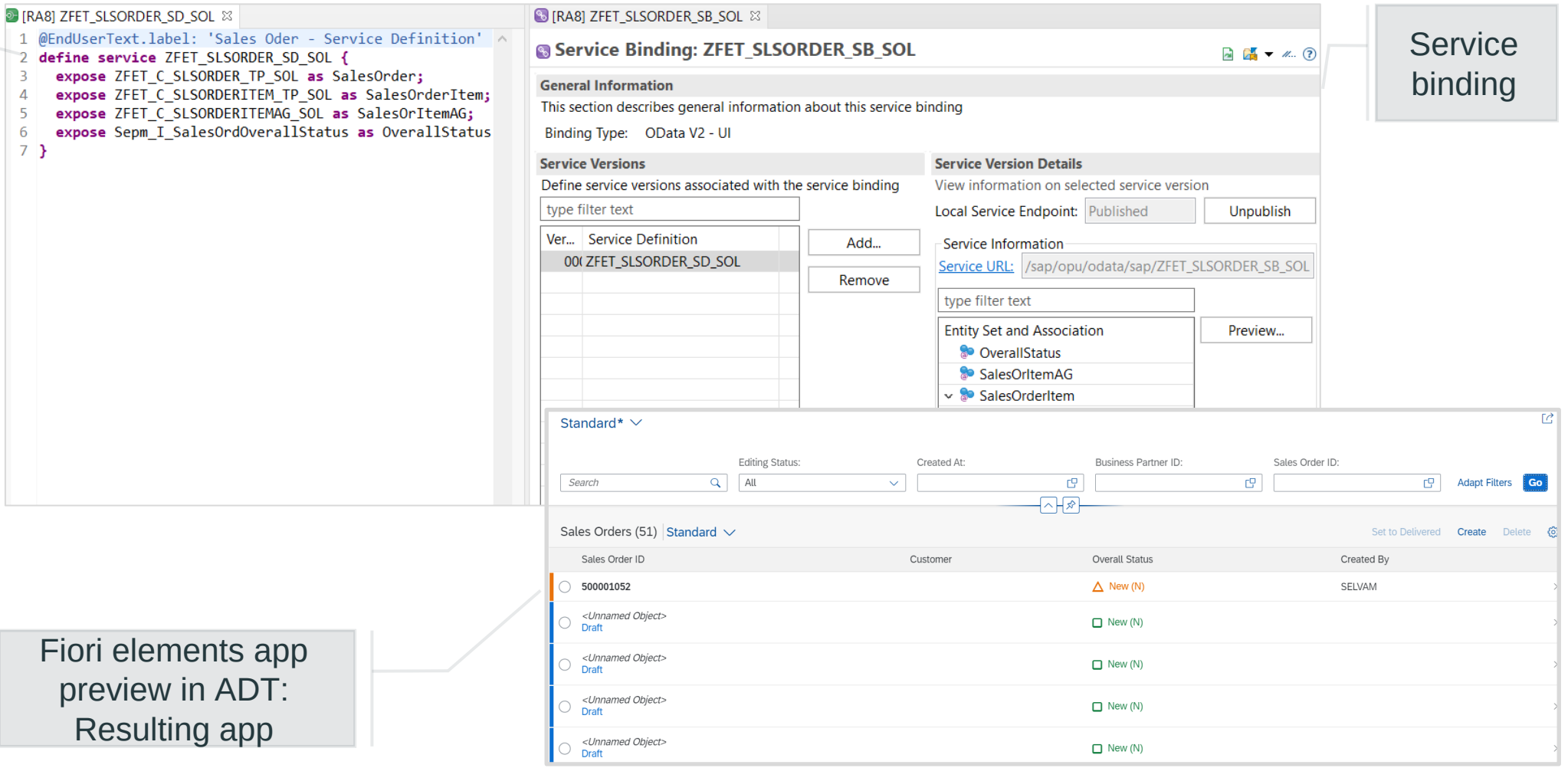
Creating and Previewing the OData UI Service

SAP Development Learning

Service definition

Service binding

Fiori elements app preview in ADT: Resulting app



OVERVIEW

SAP Development Learning

ENTITY MANIPULATION LANGUAGE

- EXTENSION OF THE ABAP LANGUAGE
  with an SQL-like syntax
- CONTROL THE TRANSACTIONAL BO BEHAVIOR IN RAP CONTEXT
- DIRECT API-BASED ACCESS TO RAP BOs
- STANDARD EML API
  for type-safe read and modifying access to RAP BOs
- GENERIC EML API
  for generic integration of RAP BOs into other frameworks
- DATA CONSISTENCY ENSURED BY DATABASE LUW
  COMMIT operation required to persist changes

READ operation

SAP Development Learning

EXAMPLE

Read with all fields

```
READ ENTITIES OF ZTKS_I_TRAVEL_M
  ENTITY Travel
    ALL FIELDS WITH VALUE #( ( TravelUUID = '<your UUID>' ) )
  RESULT DATA(travels4)
  FAILED DATA(failed)
  REPORTED DATA(reported).
```

Read with specific fields

```
READ ENTITIES OF ZTKS_I_TRAVEL_M
  ENTITY Travel
    FIELDS ( AgencyID CustomerID )
  WITH VALUE #( ( TravelUUID = 'AA1E02209E35435A17002742F96E3AB1' ) )
  RESULT DATA(travels2).
```

Read by association

```
READ ENTITIES OF ZTKS_I_TRAVEL_M
  ENTITY Travel BY \_Booking
    ALL FIELDS WITH VALUE #( ( TravelUUID = '<your uuid>' ) )
  RESULT DATA(bookings).
```

MODIFY – CREATE operation

SAP Development Learning

EXAMPLE

```
MODIFY ENTITIES OF ZTKS_I_TRAVEL_M
  ENTITY Travel
    CREATE
      SET FIELDS WITH VALUE
        #( ( %cid        = 'MyContentID_1'
             AgencyID    = '70012'
             CustomerID  = '14'
             BeginDate   = cl_abap_context_info=>get_system_date( )
             EndDate     = cl_abap_context_info=>get_system_date( ) + 10
             Description = 'Follow Development Learning' ) )

 MAPPED DATA(mapped)
 FAILED DATA(failed_c)
 REPORTED DATA(reported_c).

COMMIT ENTITIES
  RESPONSE OF ZTKS_I_TRAVEL_M
  FAILED     DATA(failed_commit_c)
  REPORTED   DATA(reported_commit_c).
```

MODIFY ACCESS TO RAP BOs

Failure handling (failed & reported)

COMMIT statement for proper LUW handling

MODIFY – UPDATE operation

SAP Development Learning

MODIFY ACCESS TO RAP BOs

EXAMPLE

```
MODIFY ENTITIES OF ZTKS_I_TRAVEL_M
  ENTITY travel
    UPDATE
      SET FIELDS WITH VALUE
        #( ( TravelUUID  = '<your uuid>'
             Description = 'Follow Development Learning' ) )

 FAILED DATA(failed_u)
 REPORTED DATA(reported_u).

" step 6b - Commit Entities
COMMIT ENTITIES
  RESPONSE OF ZTKS_I_TRAVEL_M
  FAILED     DATA(failed_commit)
  REPORTED   DATA(reported_commit).
```

Failure handling (failed & reported)

COMMIT statement for proper LUW handling

MODIFY – DELETE operation

SAP Development Learning

DELETE ACCESS TO RAP BOs

EXAMPLE

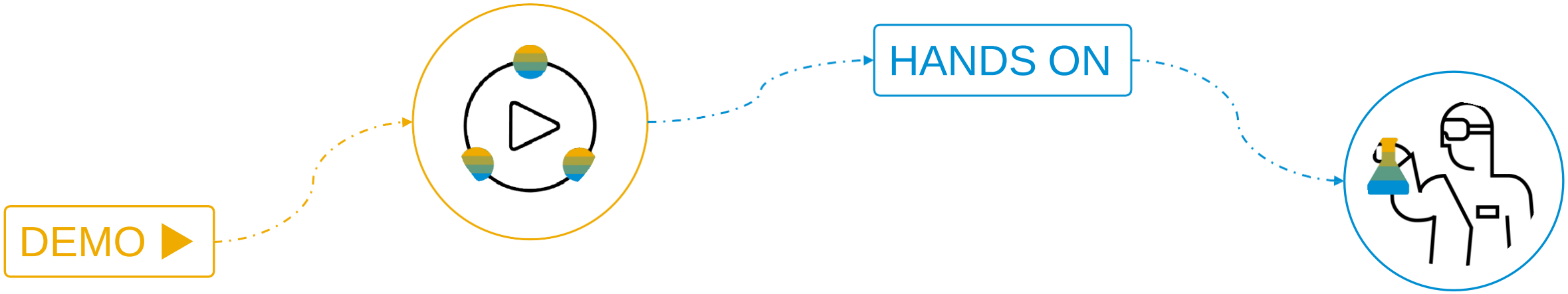
```
MODIFY ENTITIES OF ZTKS_I_TRAVEL_M
  ENTITY Travel
    DELETE FROM
      VALUE
        #( ( TravelUUID  = '<your uuid>' ) )

 FAILED DATA(failed)
 REPORTED DATA(reported).

COMMIT ENTITIES
  RESPONSE OF ZTKS_I_TRAVEL_M
  FAILED     DATA(failed_commit)
  REPORTED   DATA(reported_commit).
```

Failure handling (failed & reported)

COMMIT statement for proper LUW handling

Demo

SAP Development Learning



Practice and Understand the EML Syntax

1. Read an existing travel business object instance
2. Update a given travel business object instance
3. Delete a travel business object instance

SAP
Development
Learning

Concurrency Control

Concurrency Control

SAP Development Learning

Concurrency control prevents concurrent and interfering database access of different users. It ensures that data can only be changed if data consistency is assured.

There are two approaches to regulate concurrent writing access to data

- Pessimistic Concurrency Control (Locking)
- Optimistic Concurrency Control (ETag)

Pessimistic Concurrency Control (Locking)

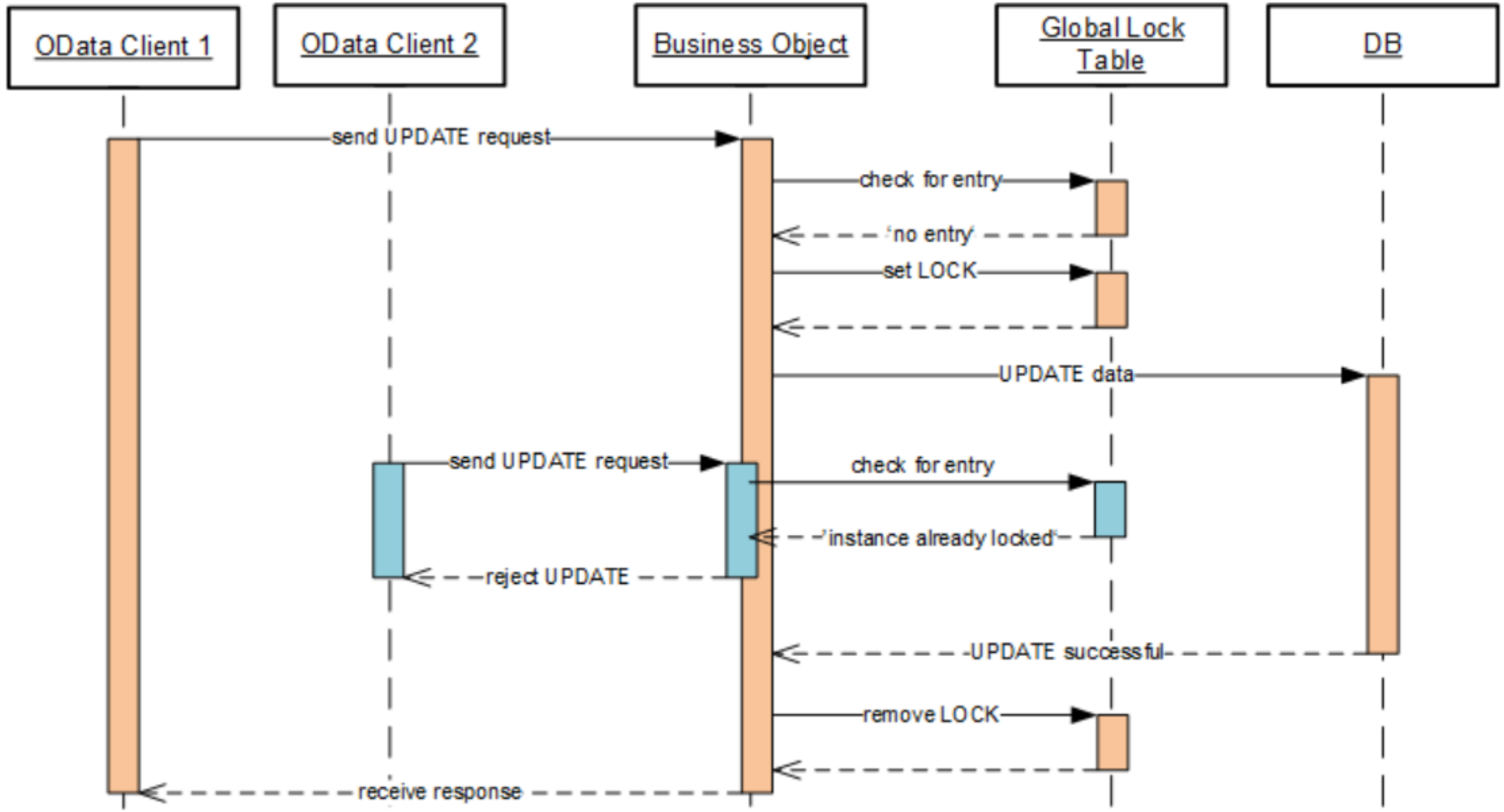
SAP Development Learning

- It prevents simultaneous modification access to data on the database by more than one user.
- Pessimistic concurrency control is done by exclusively locking data sets for the time a modification request is executed.
- The data set that is being modified by one user cannot be changed by another user at the same time.
- Technically, this is ensured by using a global lock table.
- The lifetime of an exclusive lock is tied to the session life cycle. The lock expires once the lock is actively removed after the successful transaction or with the timeout of the ABAP session.

Pessimistic Concurrency Control (Locking)

SAP Development Learning

The following diagram illustrates how the lock is set on the global lock table during an UPDATE operation.



Locking Process During UPDATE Operation

Optimistic Concurrency Control (ETag)

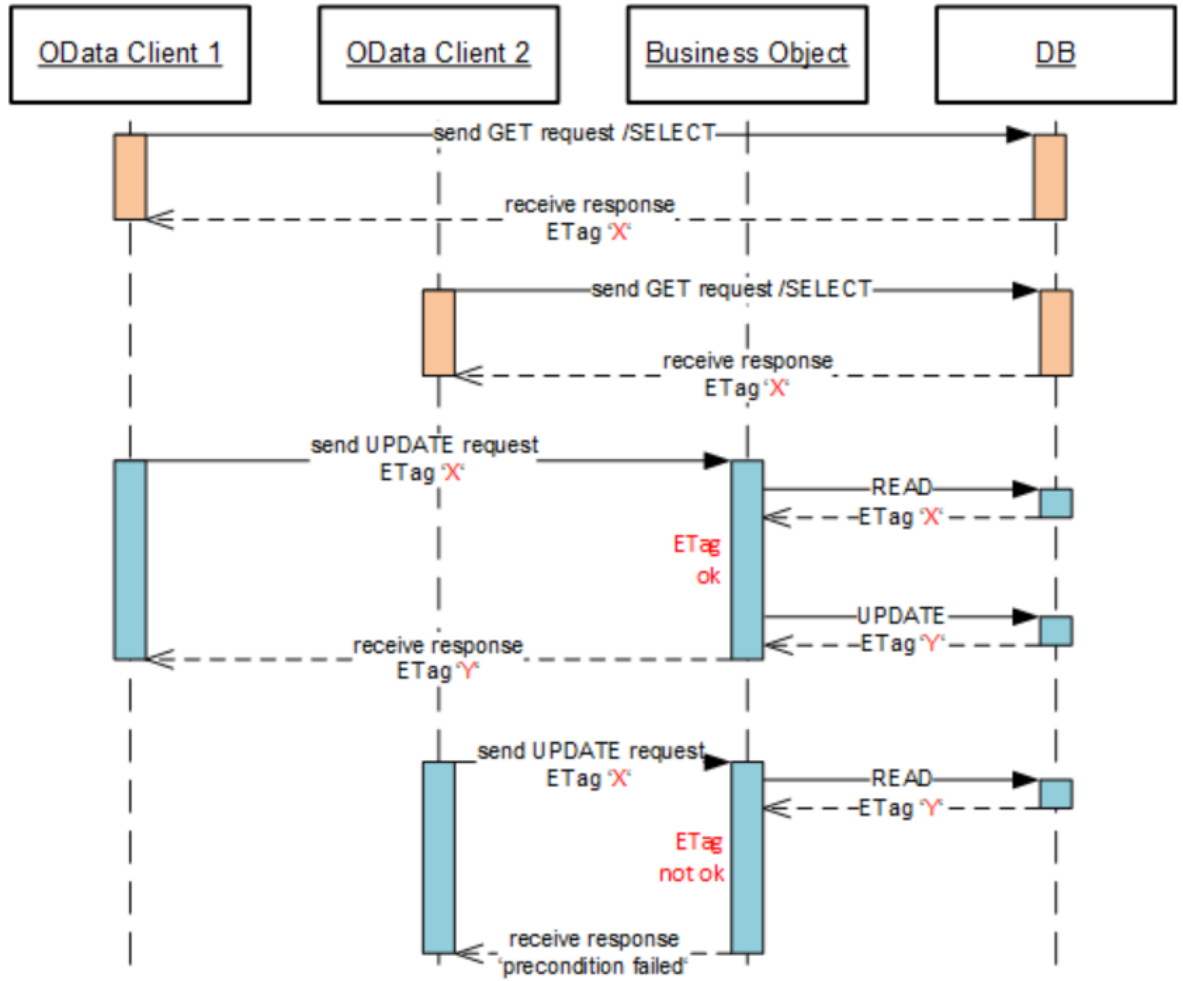
SAP Development Learning

- It enables transactional access to data by multiple users while avoiding inconsistencies and unintentional changes of already modified data.
- The approach of optimistically controlling data relies on the concept that every change on a data set is logged by a specified ETag field.
- Most often, the ETag field contains a timestamp, a hash value, or any other versioning that precisely identifies the version of the data set.
- Concurrency control based on ETags is independent of the ABAP session and instances are not blocked to be used by other clients.

Optimistic Concurrency Control (ETag)

SAP Development Learning

The following diagram illustrates the ETag checks for two different clients working on the same entity instance.

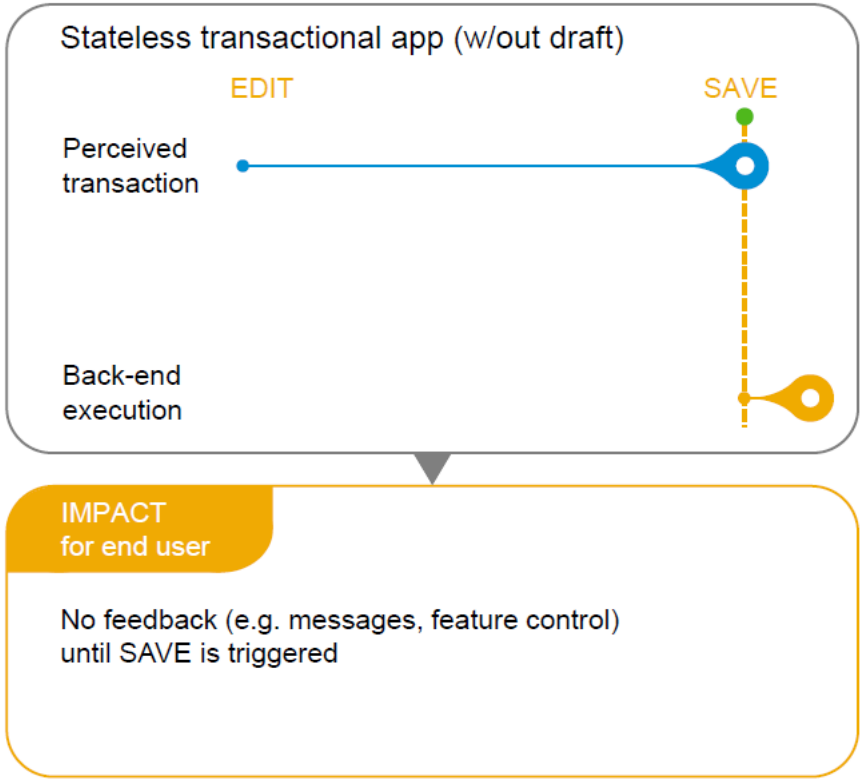


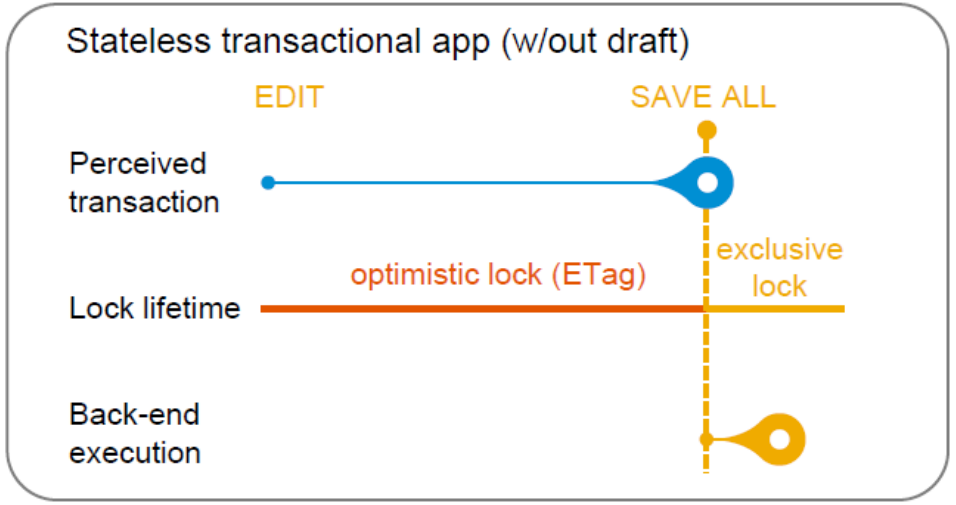
ETag Check in Update Operation

Non Draft – Stateless Transaction application

SAP Development Learning

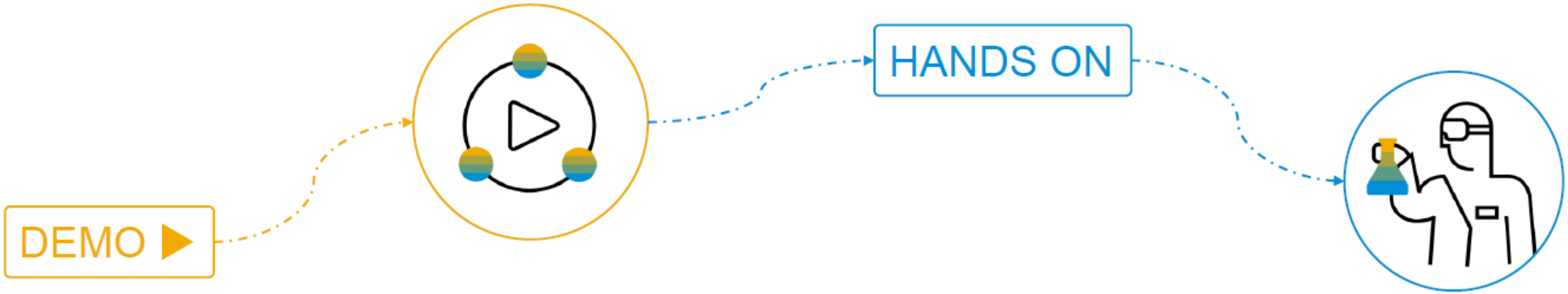
- In non-draft applications, the only option is to make use of the so-called optimistic locking using Etag.





Demo

SAP Development Learning



SAP Development Learning

Draft Handling

Requirements for SAP Fiori Architecture

SAP Development Learning

- Need to support **stateless** communication
- Enable the end user to store changed data in the backend at any time
- **Continue** at a later time
- **Recover the data**, even if the application client has crashed
- Work on draft data can be resumed on a **different device**
- Work on draft data can be resumed by a **different user**

Enabling the Draft Handling

SAP Development Learning

ENABLERS FOR CLOUD AND MODERN UX

CLOUD ENVIRONMENT EXPECTS HIGH AVAILABILITY, CONTINUOUS DELIVERY, AND LOW TCO

RESTful avoids problematic server stickiness by introducing a stateless communication

Draft fills the gap between stateless communication and stateful application

MODERN USER EXPERIENCE REQUIRES MULTI-DEVICE SUPPORT AND DATA LOSS PREVENTION WITHOUT CONNECTION TIMEOUTS

Draft persists the state device-independently in a non-process-relevant way

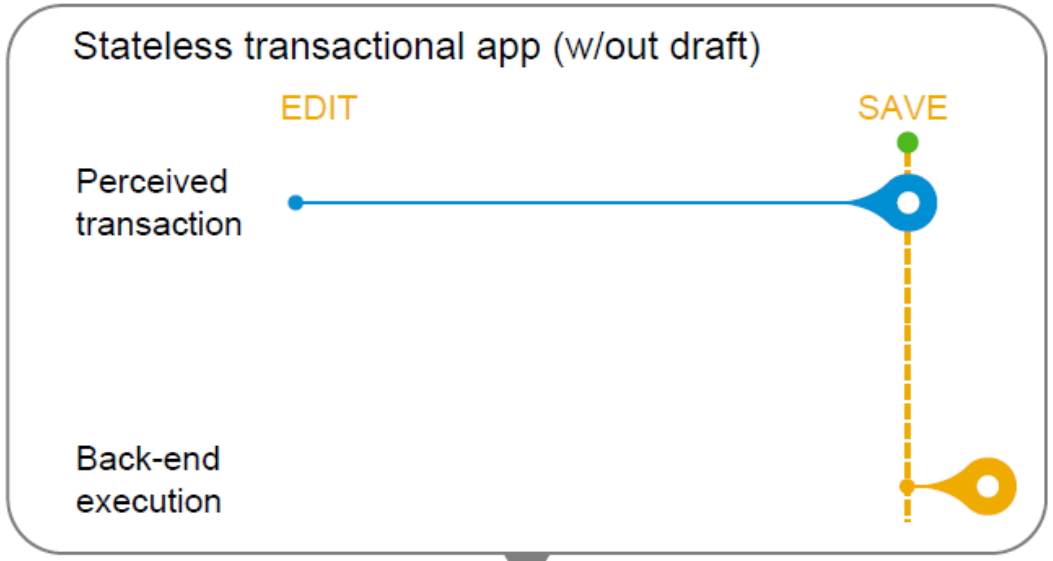
RESTful makes the draft available as an addressable resource

GUIDING PRINCIPLE

Draft is the persisted transactional buffer

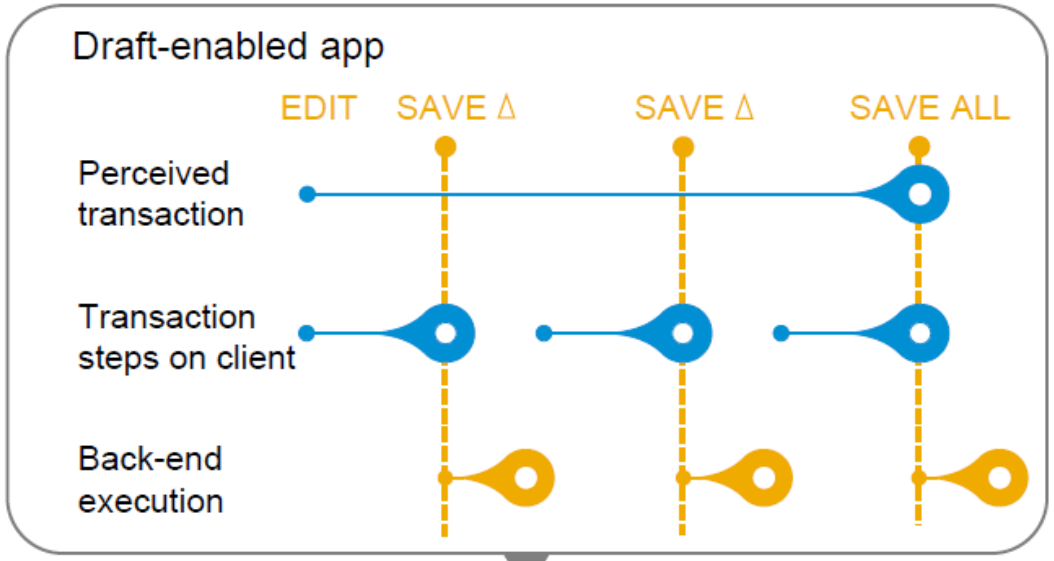
Draft - Stateful applications with stateless communication protocol

SAP Development Learning



IMPACT for end user

No feedback (e.g. messages, feature control) until SAVE is triggered



IMPACT for end user

Early feedback from calculations and validations in the back end including feature control

Draft automatically supports data loss prevention, continuous work, and device switch

Enabling the Draft Handling

SAP Development Learning

Enable draft for this business object

Define draft table

Define total etag

Enable draft for associations

Projection draft enablement

ZI_RAP_TRAVEL_1234

```
managed;
with draft;

define behavior for ZI_RAP_Travel_1234 alias Travel
implementation in class zbp_i_rap_travel_1234 unique
persistent table zrap_atrav_1234
draft table zrap_dtrav_1234
lock master total etag LastChangedAt
authorization master ( instance )
etag master LocalLastChangedAt
{
  create;
  update ( features : instance );
  delete;
  association _Booking { create; with draft; }

  field ( numbering : managed, readonly ) TravelUUID;
  field ( readonly  ) TravelId, TotalPrice, TravelStatus;
  field ( readonly  ) LastChangedAt, LastChangedBy, CreatedAt, CreatedBy, LocalLastChangedAt;
  field ( mandatory ) AgencyID, CustomerID;

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

  determination setInitialStatus    on modify { create; }
  determination calculateTotalPrice on modify { field BookingFee, CurrencyCode; }
  determination calculateTravelID   on save   { create; }

  validation validateAgency   on save { field AgencyID; create; }
  validation validateCustomer on save { field CustomerID; create; }
  validation validateDates    on save { field BeginDate, EndDate; create; }

  draft determine action Prepare  {
    validation validateAgency;
    validation validateCustomer;
```

ZC_RAP_TRAVEL_1234

```
projection;
use draft;

define behavior for ZC_RAP_Travel_1234 alias Travel
//use etag
{
  use create;
  use update;
  use delete;
  use association _Booking { create; with draft; }

  use action acceptTravel;
  use action rejectTravel;
}


define behavior for ZC_RAP_Booking_1234 alias Booking
//use etag
{
  use update;
  use delete;

  use association _Travel { with draft; }
}
```

Draft-Specific Properties

SAP Development Learning

- When you enable Draft handling for an entity, you can generate the Draft table from the IDE
  - The draft database table contains exactly the same fields as the active database table plus some technical information the RAP runtime needs to handle draft
  - The technical information is added with the draft admin include

[RA8] ZFET_D_SO_SOL

```
@EndUserText.label : 'Draft table for entity ZFET_I_SLSORDER_TP_SOL'
@AbapCatalog.enhancementCategory : #EXTENSIBLE_ANY
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zfet_d_so_sol {
  key mandt             : mandt not null;
  key salesorderuuid    : snwd_node_key not null;
  salesorder            : snwd_so_id;
  customer              : snwd_partner_id;
  overallstatus         : snwd_so_oa_status_code;
  criticality           : abap.int1;
  @Semantics.quantity.unitOfMeasure : 'zfet_d_so_sol.unitofmeasure'
  sumofquantities       : abap.quan(13,0);
  unitofmeasure         : abap.unit(3);
  created_at            : snwd_created_at;
  created_by_loginname  : syuname;
  locallastchangedat    : snwd_changed_at;
  locallastchangedby    : syuname;
  lastchangedat         : timestampl;
  "%admin"              : include sych_bdl_draft_admin_inc;

}
```

[RA8] SYCH_BDL_DRAFT_ADMIN_INC

```
@EndUserText.label : 'Standard-Include für Draft-Verwaltung (BDL Syn
@AbapCatalog.enhancementCategory : #NOT_EXTENSIBLE
define structure sych_bdl_draft_admin_inc {
  draftentitycreationdatetime   : sych_bdl_draft_created_at;
  draftentitylastchangedatetime : sych_bdl_draft_last_changed_at;
  draftadministrativedatauuid   : sych_bdl_draft_admin_uuid;
  draftentityoperationcode      : sych_bdl_draft_operation_code;
  hasactiveentity               : sych_bdl_draft_hasactive;
  draftfieldchanges             : sych_bdl_draft_field_changes;

}
```

Total ETag

SAP Development Learning

- Total ETag is a designated field in a draft business object to enable optimistic concurrency checks during the transition from draft to active data.
- Draft business objects require a total ETag.
- Total ETag always refers to the complete BO. As soon as an instance of any BO entity is changed, the total ETag is updated.
- The field is annotated in CDS with the annotation @Semantics.systemDateTime.lastChangedAt: true with the precondition that the field has a date-compatible type.
- The total ETag field is included in the field-mapping prescription in the behavior definition.

Draft Status

SAP Development Learning

- New draft

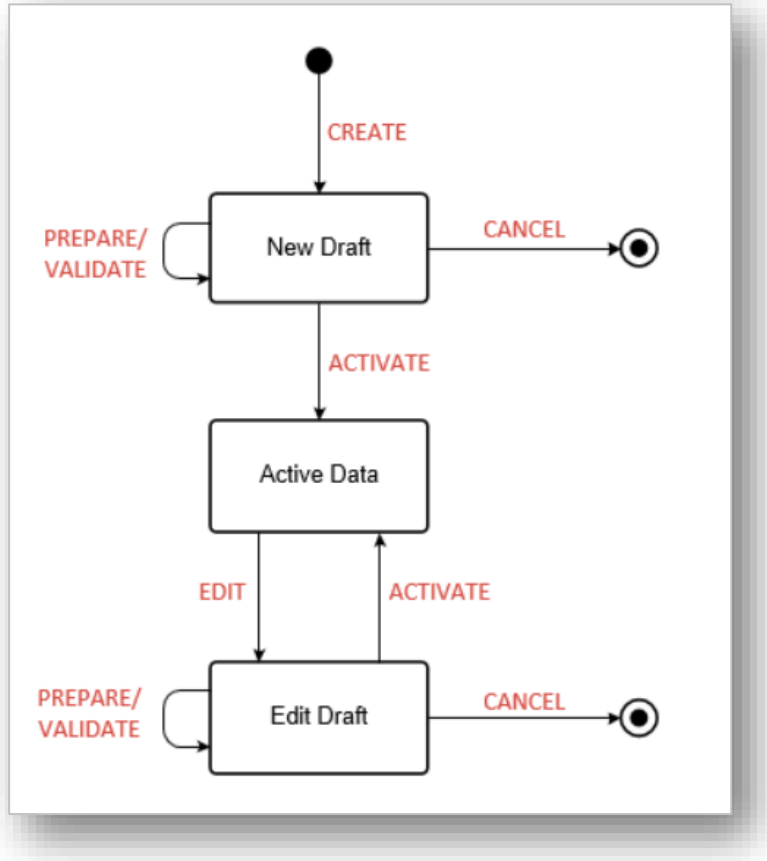
  For initial data (where no database version exists in a traditional application).

- Active data

  Corresponds to the database version of a traditional application.

- Edit draft

  Exists in parallel to the corresponding active data.



New Draft

SAP Development Learning

- Can be empty or can contain default values
- For a new sales order item that is being created, for example, the item position can be calculated, and a product can be selected from the product catalog.
- No corresponding active document exists yet

Active Data

SAP Development Learning

- The draft version is converted into active data by calling the ACTIVATION action
- VALIDATION is triggered
- After being saved as active data, the draft is deleted

Edit a Draft

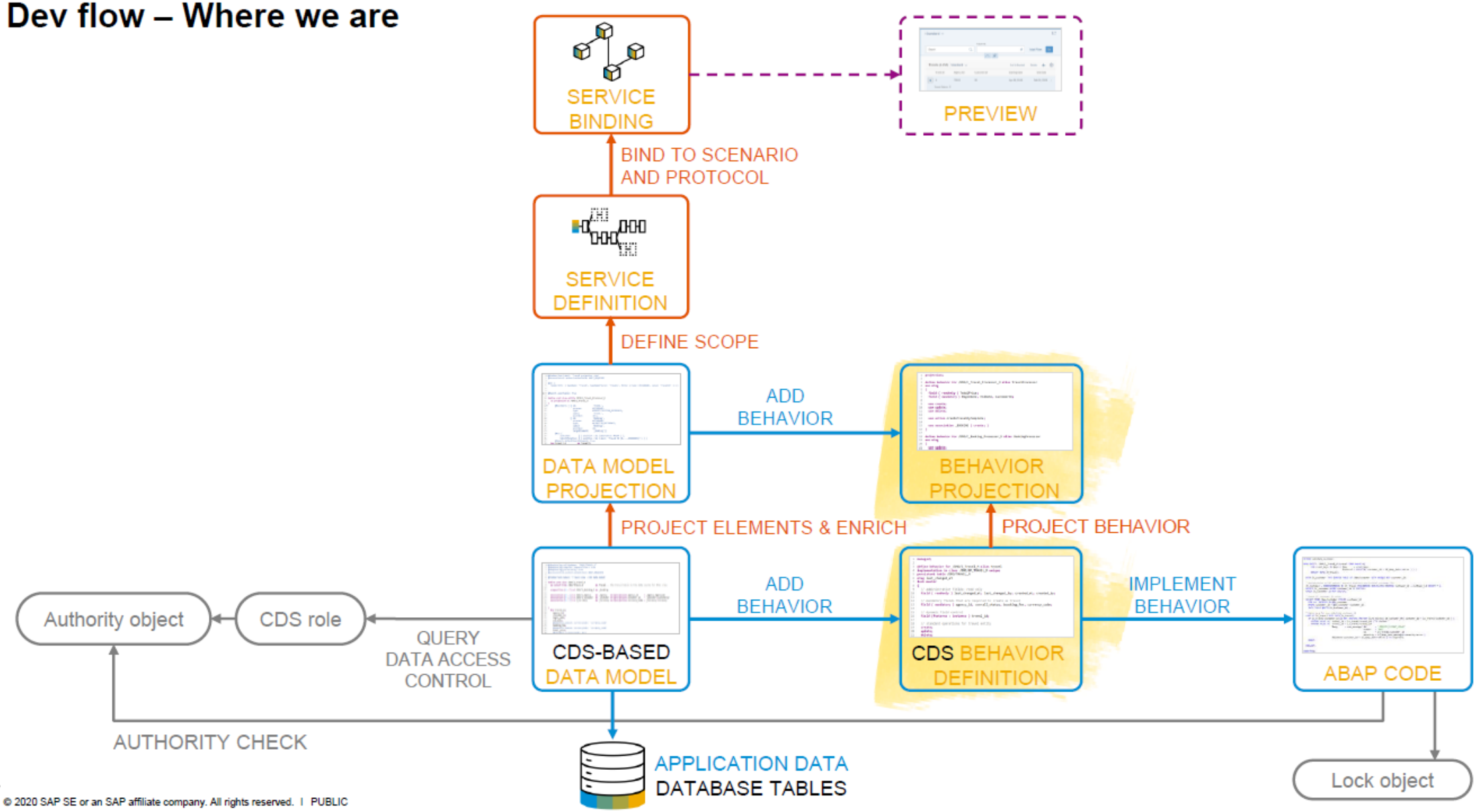
SAP Development Learning

- Changed data is immediately stored in the draft persistence (even if it‘s incomplete or contains errors).
- However, user input can also be validated immediately.
- The consistency can be checked using VALIDATION action (messages are displayed without any changes on the Draft document).
- The PREPARATION action can also be triggered. This modifies the draft.

Enabling the Draft handling

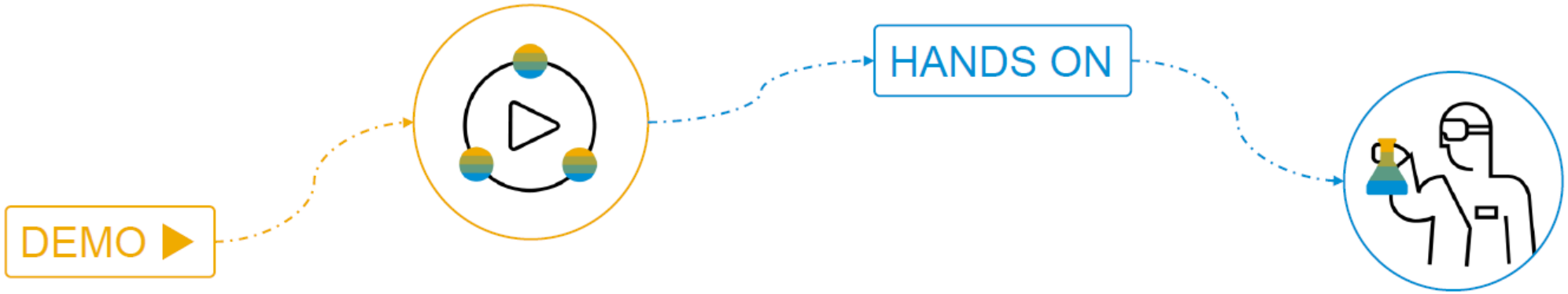
SAP Development Learning

Dev flow – Where we are



Demo

SAP Development Learning

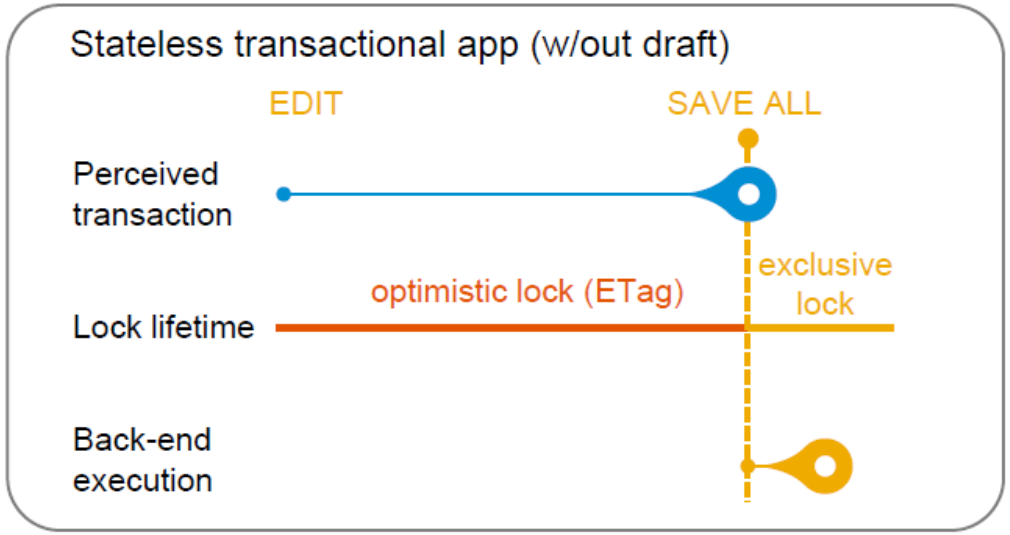


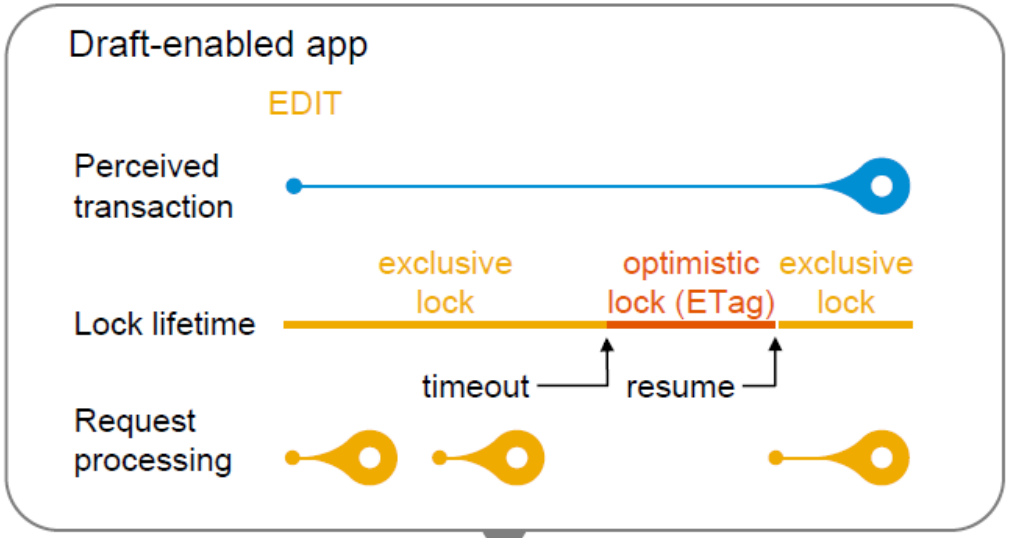
Enable the draft handling in the BO behavior

1. Adjust the CDS data models of the travel and booking entities
2. Enable the draft handling in BO behavior definitions
3. Adjust the business logic in BO behavior implementations
4. Test the enhanced SAP Fiori elements app

Draft – Impact for concurrency handling: Locks

SAP Development Learning





End user perceived transaction distributed over a long period, so locks can‘t be held all the time

Locks are not bound to a session and have their own timeout

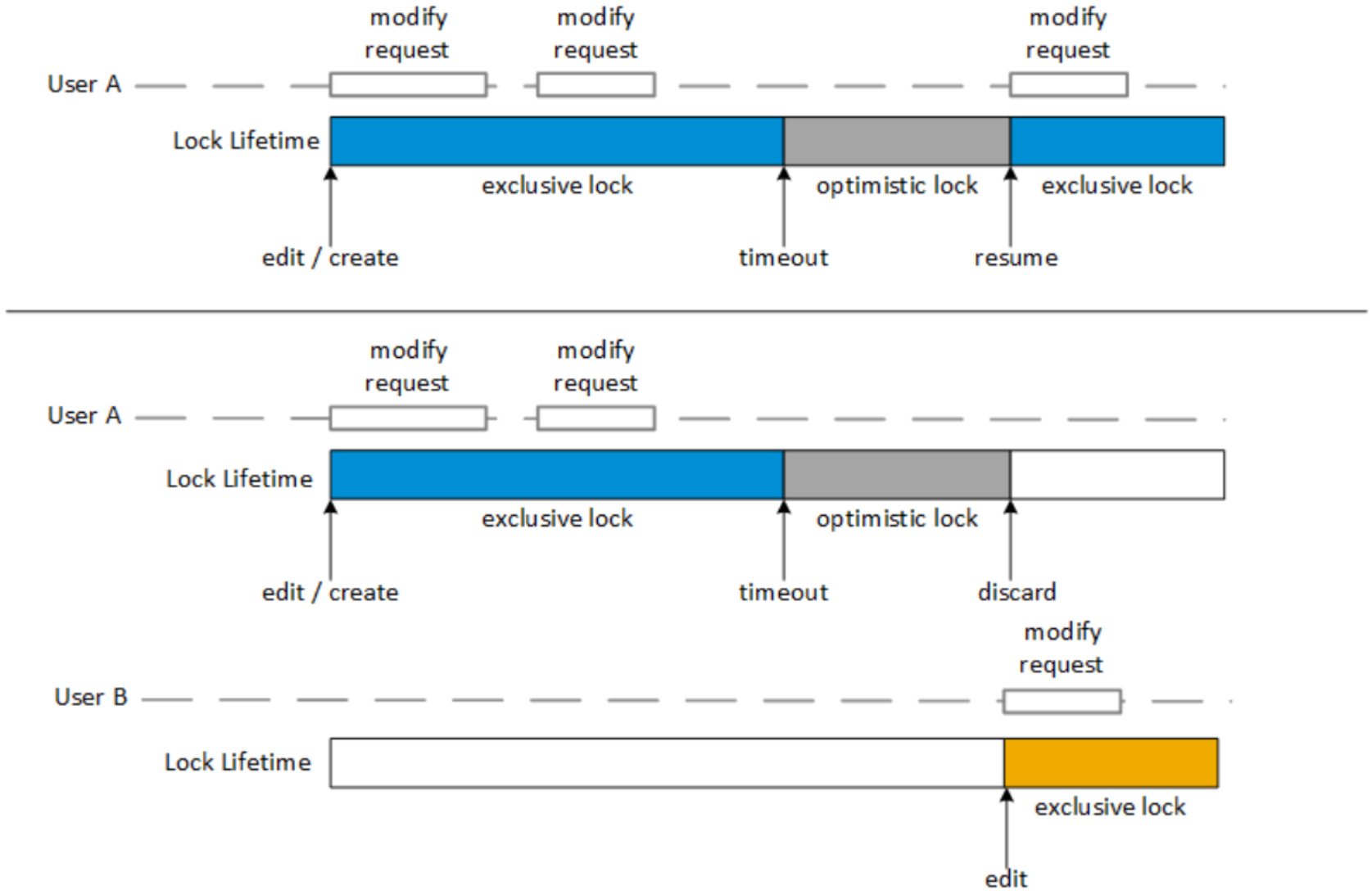
After timeout of exclusive locks an optimistic lock is used

Resume re-acquires the exclusive locks

Request processing is executed in a single ABAP session

Lock Lifetime of a Draft

SAP Development Learning



Total ETag Vs ETag

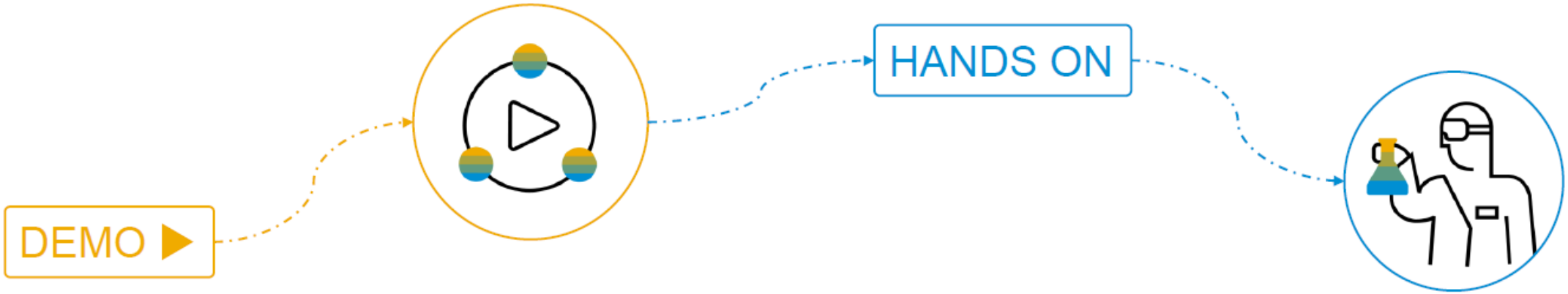
SAP Development Learning

Total ETag and ETag master/dependent are two sides of the same medal. For a smoothly running application, you need both ETags.

- The total ETag is a designated field in a draft business object to enable optimistic concurrency checks during the transition from draft to active data.
- The total ETag always refers to the complete BO.
- Is updated by the RAP runtime framework as soon as an active instance is changed.
- The ETag master/dependent concept ensures that the end user of a UI service only changes instances with the state that is displayed on the UI.
- With ETag master, each BO entity can be checked independently.
- Is update by the RAP runtime framework as soon as an BO is changed.

Demo

SAP Development Learning



SAP
Development
Learning

RAP Testability

RAP Testability

SAP Development Learning

- Testing Business logic in RAP Business Objects (BO)
  - ✓ Testing Behavior Implementation using EML by executing side-effects (determinations and validations) and isolating only database dependencies (using ABAP/CDS test double framework)
  - ✓ Testing Behavior Implementation using EML by isolating dependencies on other BOs (using Behavior Test Double Framework)
- Integration/Scenario Tests for OData Services

Open SQL Test Double Framework: https://wiki.wdf.sap.corp/wiki/display/NWCUIAMIndia/Open+SQL+Test+Double+Framework

CDS Test Double Framework: https://wiki.wdf.sap.corp/wiki/display/NWCUIAMIndia/CDS+Test+Double+Framework

Before we start writing unit tests, you need to know

SAP Development Learning

- Some new EML syntaxes
- ABAP Test relations
- BIL(Behavior Implementation Language) Type Definitions

Some new EML syntaxes to consume BOs

SAP Development Learning

Executing Actions

```
MODIFY ENTITY ztks_i_travel_m
      EXECUTE acceptTravel
   FROM VALUE #( ( TravelUuid = travel_uuid ) )
       RESULT DATA(result).
```

Get Features

```
GET FEATURES OF ztks_i_travel_m
  ENTITY travel
  FROM VALUE #( ( %key-TravelUuid = travel_uuid ) )
  REQUEST VALUE #( %action-acceptTravel = cl_abap_behv=>flag_changed )
  RESULT DATA(feature_control)
  FAILED DATA(failed_keys).
```

Set Lock

```
SET LOCKS OF ZTKS_I_Travel_M
      ENTITY Travel
  FROM VALUE #( ( TravelUuid = travel_uuid ) )
      FAILED DATA(failed_lock)
    REPORTED DATA(reported_lock).
```

ABAP Test Relations

SAP Development Learning

- To indicate to the ABAP system which RAP entities are covered by a test class or test method, *ABAP test relations* were invented
- The usage is simple, just add an ABAP doc comment in the definition part of the class above the *class* statement
- You can link the EML test classes with the behavior definition or even the service binding using test relations
- You can also bundle unit tests of multiple behavior implementations in one test class using the test relation

```
"! @testing BDEF:ZTKS_I_TRAVEL_M
"! @testing ZCL_TKS_TRAVEL_M
CLASS ltcl_travel DEFINITION FINAL FOR TESTING
  DURATION SHORT
  RISK LEVEL HARMLESS.
```

BIL Type Definitions

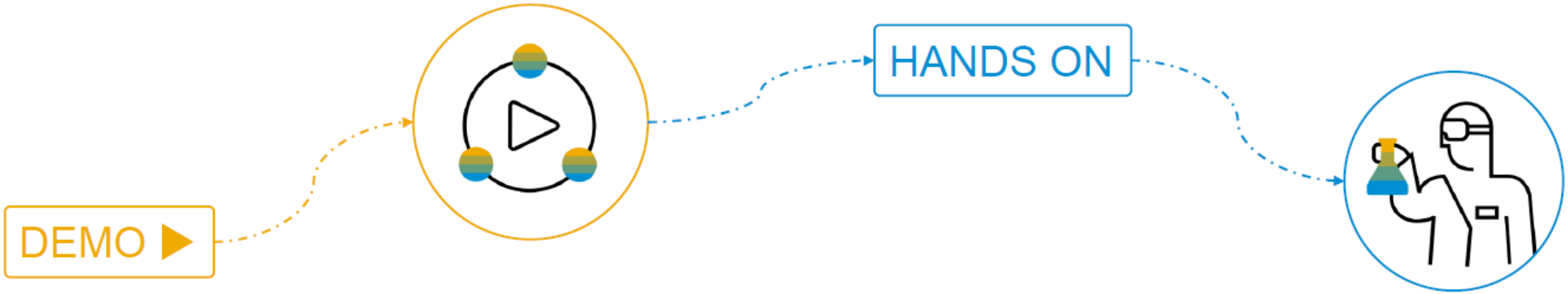
SAP Development Learning

- To use the special BIL types in method signatures as importing and exporting parameters, you can create global/local types

```
TYPES: ty_t_create      TYPE TABLE FOR CREATE        ztks_i_travel_m,
       ty_t_create_by_a TYPE TABLE FOR CREATE        ztks_i_travel_m\_Booking,
       ty_t_update      TYPE TABLE FOR UPDATE        ztks_i_travel_m,
       ty_t_delete      TYPE TABLE FOR DELETE        ztks_i_travel_m,
       ty_t_lock        TYPE TABLE FOR LOCK          ztks_i_travel_m,
       ty_t_read_in     TYPE TABLE FOR READ IMPORT   ztks_i_travel_m,
       ty_t_read_out    TYPE TABLE FOR READ RESULT   ztks_i_travel_m,
       ty_t_action_in   TYPE TABLE FOR ACTION IMPORT ztks_i_travel_m~acceptTravel,
       ty_t_action_out  TYPE TABLE FOR ACTION RESULT ztks_i_travel_m~acceptTravel,
       ty_t_failed      TYPE TABLE FOR FAILED        ztks_i_travel_m,
       ty_t_mapped      TYPE TABLE FOR MAPPED        ztks_i_travel_m,
       ty_t_reported    TYPE TABLE FOR REPORTED      ztks_i_travel_m,
       ty_failed        TYPE RESPONSE FOR FAILED     ztks_i_travel_m,
       ty_mapped        TYPE RESPONSE FOR MAPPED     ztks_i_travel_m,
       ty_reported      TYPE RESPONSE FOR REPORTED   ztks_i_travel_m.
```

Demo-Testing by executing side-effects and isolating only database dependencies

SAP Development Learning



Behavior Test Double Framework or Business Object Mocking

SAP Development Learning

There can be many reasons to mock a RAP BO

- Existing code shall now call the RAP BO instead of legacy BAPI APIs
- Determination or Validation unit tests need to mock the BO behavior to enable an isolated testing
- Component tests shall isolate own BOs from foreign BOs

Before we mock a BO, you need to know

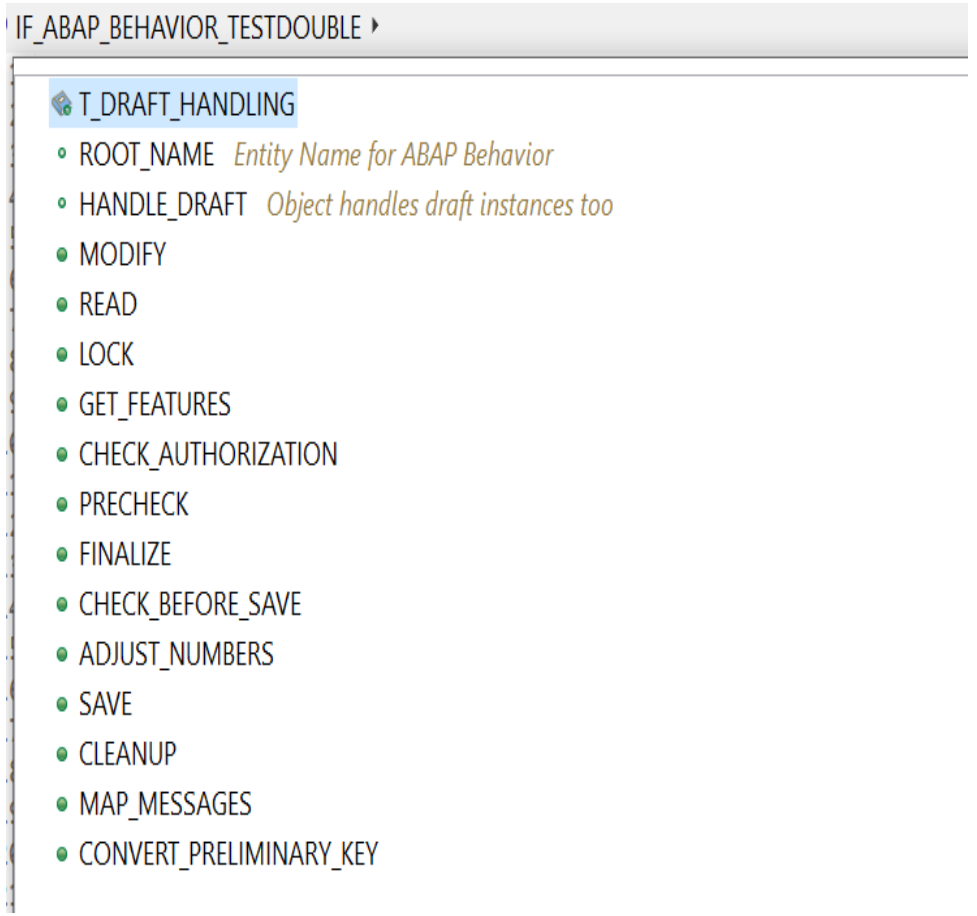
SAP Development Learning

- Interface *IF_ABAP_BEHAVIOR_TESTDOUBLE* needs to be implemented to create a mock/test double

- To inject test double for behavior implementation, use

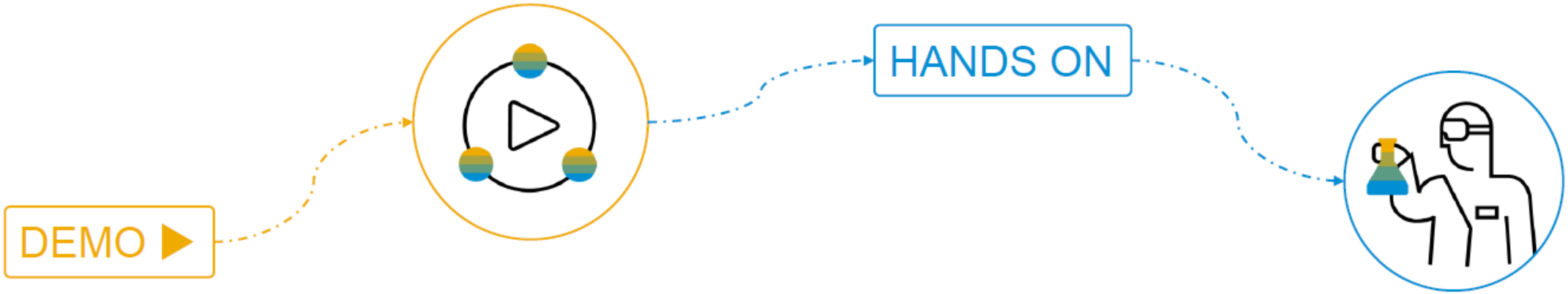
CL_ABAP_BEHV_TEST_ENVIRONMENT=>set_test_double(behv_test_double_impl).

- This will help in injecting test double for the behavior implementation so that whenever an EML statement is executed during test execution, the corresponding methods in these handler and saver test double gets invoked instead of the actual behavior implementation



Demo-Business Object Mocking

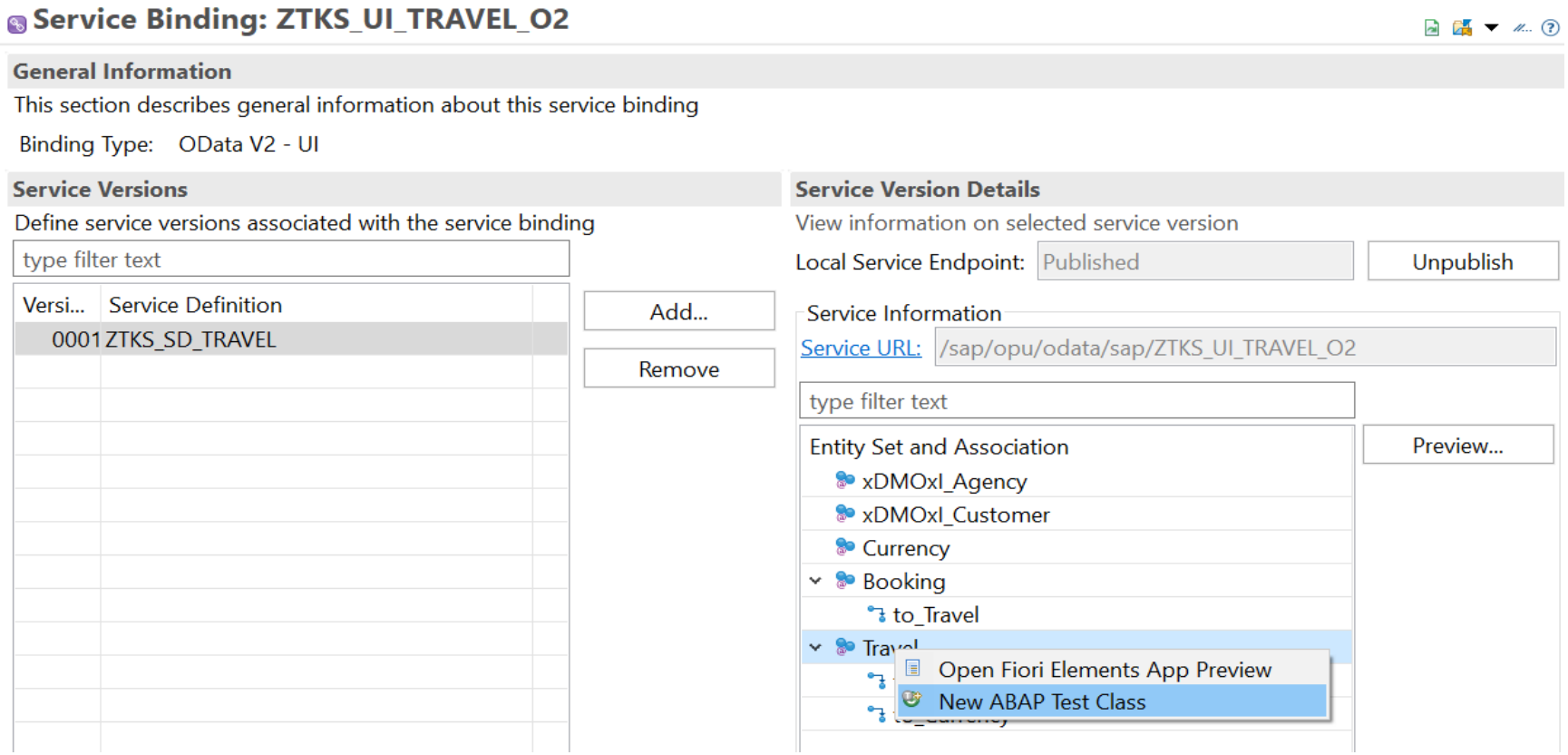
SAP Development Learning



Integration/Scenario Tests for OData Services

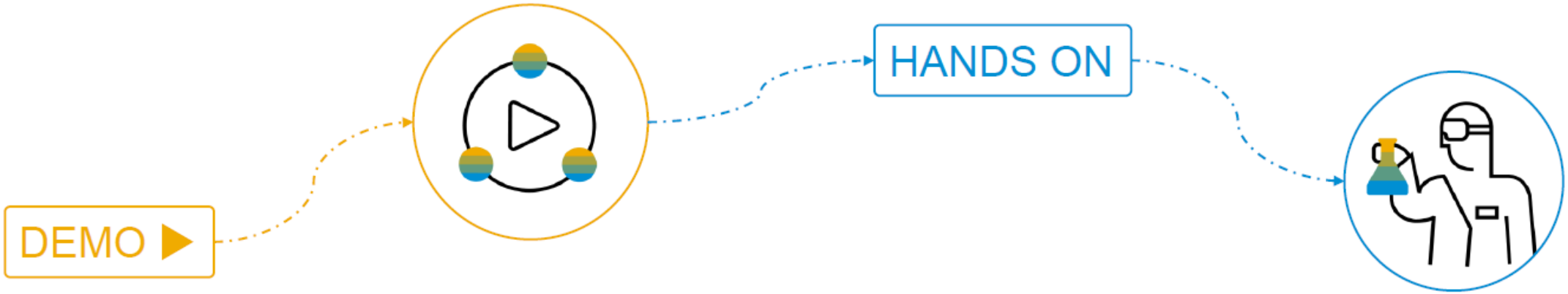
SAP Development Learning

- In order to ensure that the service is functionally correct for consumption, you can create an ABAP test class for your OData service locally from Service Binding



Demo-Testing Service

SAP Development Learning



Key Takeaway

Architecture Overview

The big picture

SAP Development Learning

SERVICE CONSUMPTION

BUSINESS SERVICES PROVISIONING

DATA MODELING & BEHAVIOR

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

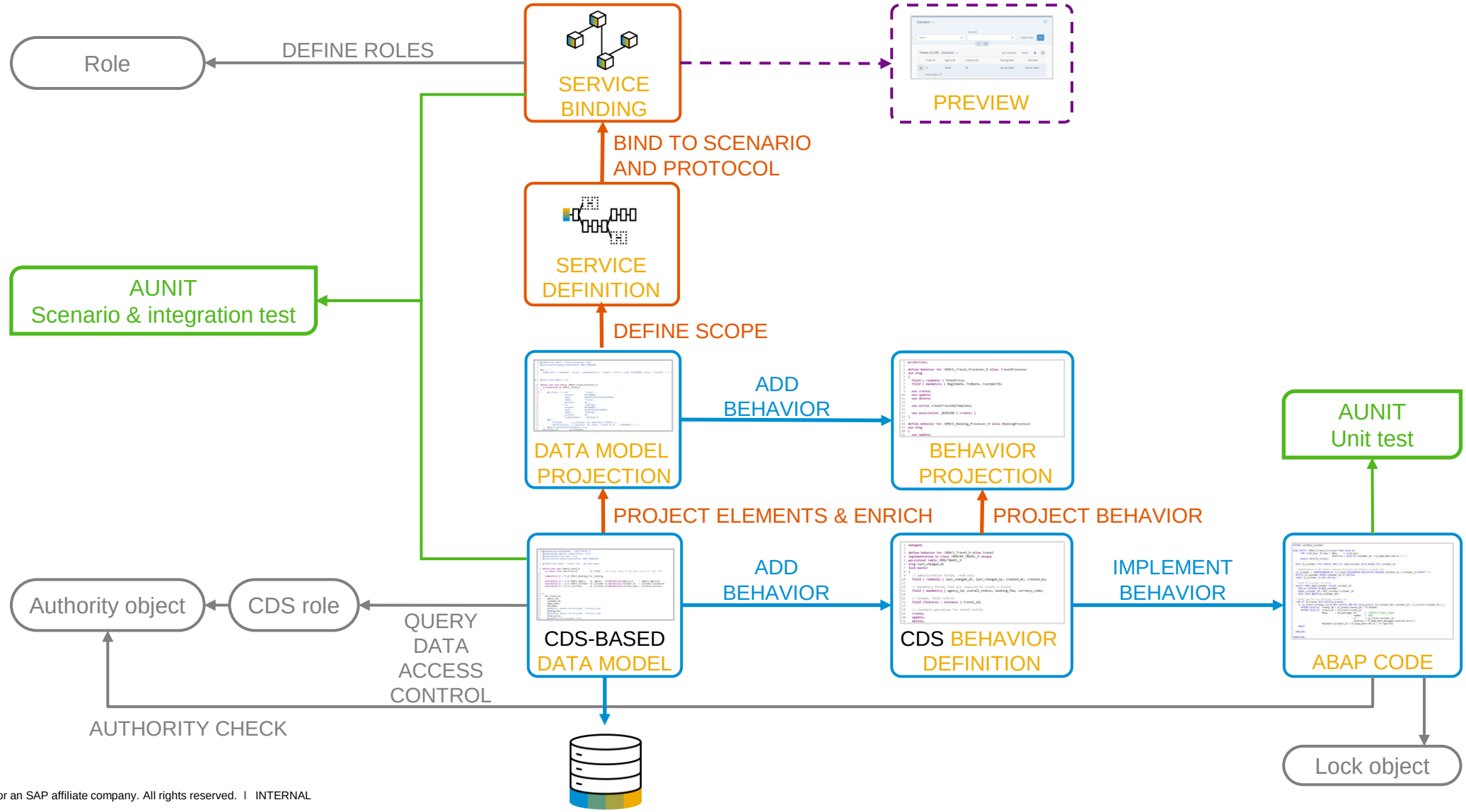
BUSINESS OBJECT PROJECTION
- CDS: Projection views
- BDEF: Behavior projection
- ABAP: Behavior implementation

BUSINESS OBJECTS
- CDS: Data modeling
- BDEF: Behavior definition
- ABAP: Behavior implementation

QUERIES
- CDS: Data modeling

Architecture Overview

Development flow



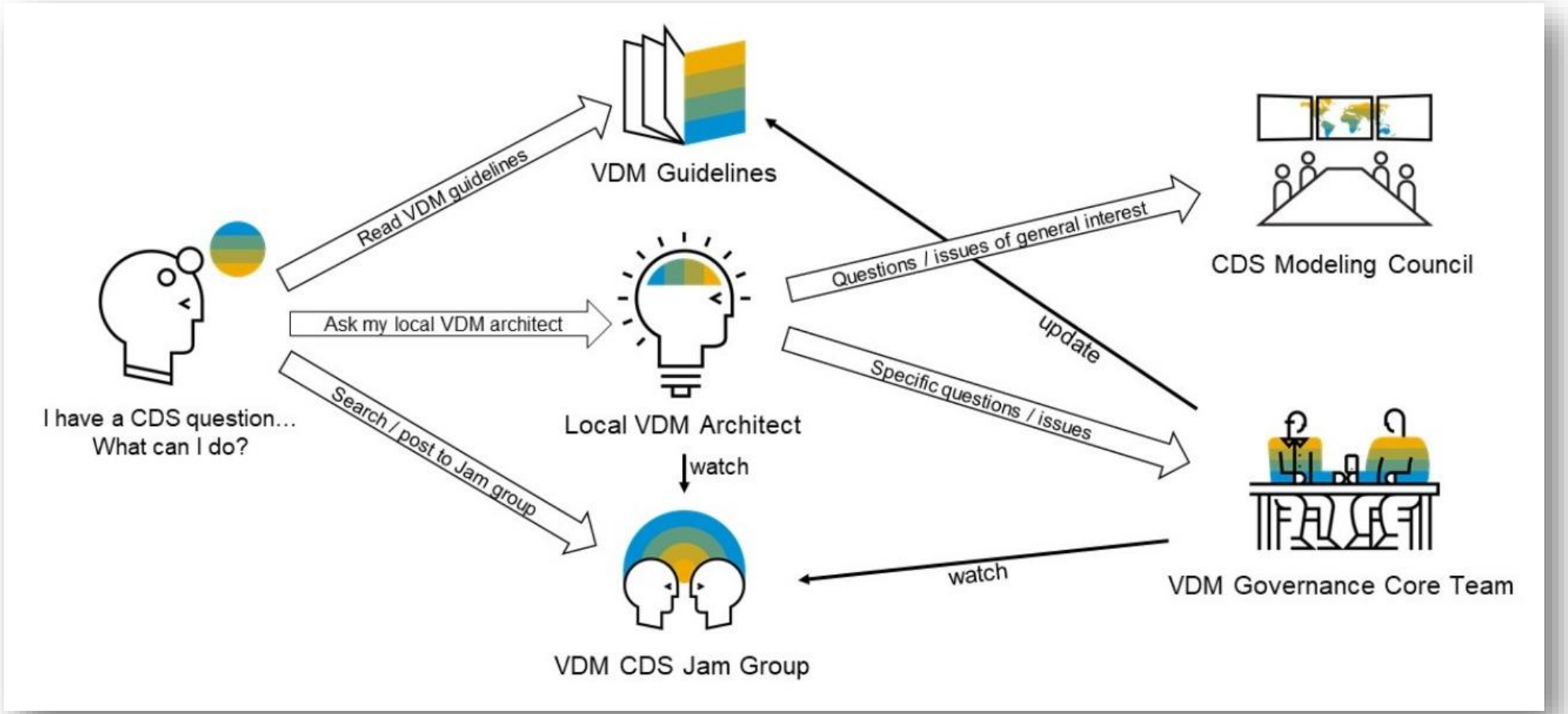
SAP
Development
Learning

Appendix

Introduction to CDS

Virtual Data Model (VDM) for SAP S/4HANA

SAP Development Learning



[VDM Governance Working Model wiki]

SAP
Development
Learning

Custom Entities

CDS Custom Entity Overview

SAP Development Learning

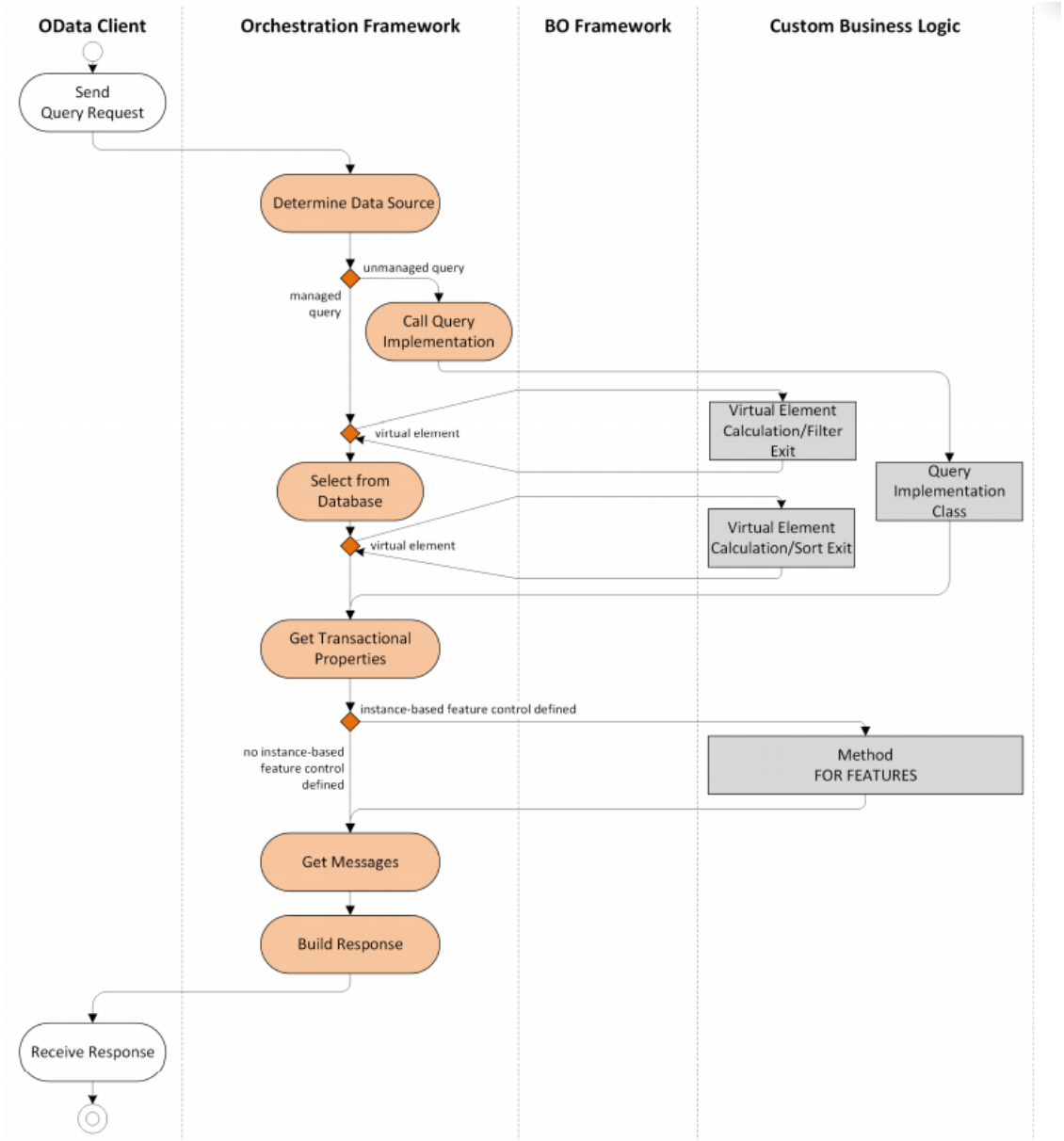
- It can be used to retrieve the data from different sources including another system for developing a UI service with access to a remote service .
- It can be used as a data model for the new services whose runtime is implemented manually, via un-managed query framework, in a related ABAP class that implements the interface IF_RAP_QUERY_PROVIDER.
- Syntax warning will be shown on custom entity if query implementation class(Select method) is not implemented.
- Annotation @ObjectModel.query.implementedBy: 'ABAP: <Query_Impl_Class>' is used to reference the query implementation class in the data definition of the CDS custom entity.
- It does not have any database representation like SQL view for CDS entity.
- It can have parameters, elements and associations.
- It can be an entity in a business object, for example a root, a parent, or a child entity using root and composition relationships. Custom entities may also be used as targets in the definition of associations and define associations as a source.

CDS Custom Entity Syntax

SAP Development Learning

```
@EndUserText.label: 'EndUserText'
@ObjectModel.query.implementedBy: 'ABAP:<Query_Impl_Class>'
[define] [root] custom entity CustomEntityName
     [ with parameters
          ParamName : dtype [, ...]    ]
{
        [@element_annot]
   [key] ElemName : dtype;
         ElemName : dtype;
...
   [ _Assoc : association [cardinality] to TargetEntity on CondExp [with default
filter CondExp ]   ];
   [  _Comp : composition [cardinality] of TargetEntity   ];
   [  [@element_annot]
      _ParentAssoc : association to parent Parent on CondExp   ];
}
```

CDS Custom Entity – Query Runtime



SAP Development Learning

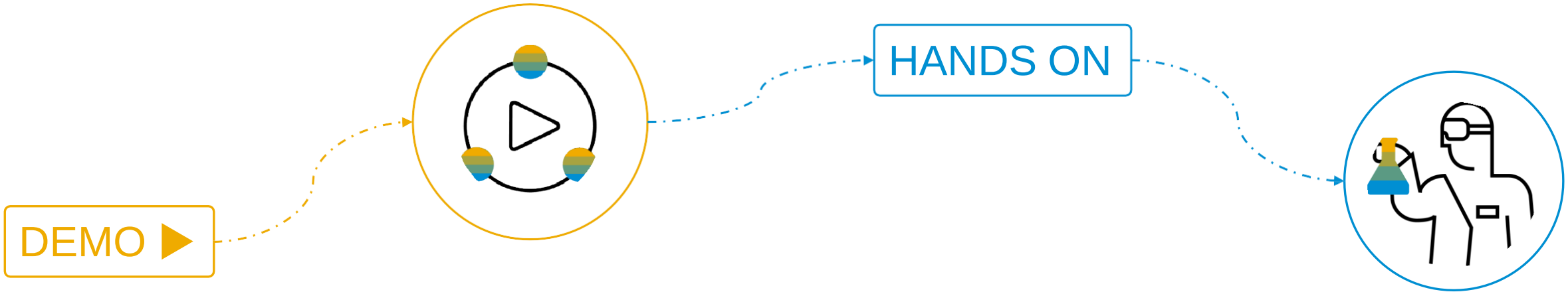
CDS Custom Entity Limitation

SAP Development Learning

- Projection view can not be created for custom entity.
- Custom entity cannot be used in ABAP SQL SELECT executions.
- Sorting, filtering, access control needs to be implemented manually in the query implementation class.
- You cannot use elements of an associated custom entity in the element list of the source CDS entity.
- You cannot use an access control object for a custom entity.

Demo - CDS Custom Entity

SAP Development Learning



Implement CDS Custom Entity

1. Create a CDS custom entity.
2. Implement a class for CDS custom entity.
3. Define the CDS custom entity as value help.

SAP
Development
Learning

Numbering Types

Numbering

SAP Development Learning

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

- Early Numbering
  - External Numbering
  - Internal Numbering
    - Managed Early Numbering
    - Unmanaged Early Numbering
  - Optional External Numbering
- Late Numbering

Early Numbering

SAP Development Learning

The numbering type early numbering refers to an early value assignment for the primary key field. In this case, the final key value is available in the transactional buffer instantly after the **MODIFY** request for **CREATE**.

Types:

- External Numbering
- Internal Numbering

External Numbering

SAP Development Learning

We refer to external numbering if the consumer hands over the primary key values for the **CREATE** operation, just like any other values for non-key fields. The runtime framework (managed or unmanaged) takes over the value and processes it until finally writing it to the database. The control structure for the CREATE operation is flagged with true for the primary key field.

```
...
define behavior for Entity [alias AliasedName]
{  ...
 field (mandatory:create| readonly:update);
  ...}
```

Ensure Uniqueness Check in precheck Stage

Uniqueness Check

SAP Development Learning

Responsibilities for the Uniqueness Check in Scenarios with External Numbering

Managed Scenario without Draft without Unmanaged Lock	RAP runtime framework
Managed Scenario with Draft without Unmanaged Lock	• for active instances: RAP runtime framework • for draft instances: Application developer needs to implement a precheck and the draft resume action.
Managed Scenario with/without Draft with Unmanaged Lock	Application developer needs to implement the uniqueness check in the precheck and the draft resume action.
Unmanaged Scenario with/without Draft	Application developer needs to implement the uniqueness check in the precheck and the draft resume action.

Internal Numbering

SAP Development Learning

In scenarios with internal numbering, the runtime framework assigns the primary key value when creating the new instance. Internal numbering can be managed by the RAP runtime or unmanaged, that is implemented by the application developer.

Managed Internal Numbering

SAP Development Learning

When using managed numbering, a UUID is drawn automatically during the CREATE request by the RAP managed runtime.

```
[implementation] managed [implementation in class ABAP_CLASS [unique]];
define behavior for Entity [alias AliasedName]
  lock (master|dependent() )
  ...
{  ...
  field ( readonly, numbering:managed ) KeyField1 [, KeyField2, …, keyFieldn];
  ...}
```

[Automatically Drawing Primary Key Values in Managed BOs]

Unmanaged Numbering

SAP Development Learning

With the current version of the ABAP RESTful Programming Model, it is not possible to use unmanaged numbering in managed BOs

Optional External Numbering

We refer to optional external numbering if both, external and internal numbering is possible for the same BO. If the consumer hands over the primary key value (external numbering), this value is processed by the runtime framework. If the consumer does not set the primary key value, the framework steps in and draws the number for the instance on CREATE.
Optional external numbering is only possible for managed business objects with UUID keys.

Use cases for optional external numbering are replication scenarios in which the consumer already knows some of the UUIDs for specific instances to create.

Late Numbering

SAP Development Learning

The numbering type late numbering refers to a late value assignment for the primary key fields. The final number is only assigned just before the instance is saved on the database.

Method ADJUST_NUMBERS

Implements late numbering.

The implementation of adjust_numbers( ) is only required if late numbering is modeled in the behavior definition.

The method must not fail and thus does not return any failed keys since the exchange of temporary IDs takes place after the point-of-no-return. If the application needs to stop the transaction, it can only produce a short dump.

Example : A_MAINTENANCENOTIFICATION

SAP
Development
Learning

Virtual Elements

ABAP RESTful Programming Model – Virtual Elements

SAP Development Learning

- Used to enable additional fields that are not persisted on the database (specific to application)
- Different Virtual elements in different BO projections
- Read-only and Transactional both
- Implemented using ABAP class and handled by SADL framework

```
define view entity ZTKS_C_BOOK_M
  as projection on ZTKS_I_BOOK_M
{ ...
          @ObjectModel.virtualElementCalculatedBy: 'ABAP:ZTKS_CL_DAYS_TO_FLIGHT'
          @EndUserText.label: 'Days to Flight'
          @EndUsertText.quickInfo: 'Calculates the Relative Flight Date'
  virtual DaysToFlight : abap.int2,
  …}
```

Virtual Elements - Restrictions

SAP Development Learning

- You can use virtual elements only in CDS projection views.
- In the current version of the ABAP RESTful Programming Model, you cannot add implementations for filtering or sorting by virtual elements.
- Virtual elements cannot be keys of the CDS projection view.
- Virtual elements cannot be used together with the grouping or the aggregation function.
- Data from virtual elements can only be retrieved via the query framework. In particular this means that the following options to retrieve data from CDS are not possible for virtual elements:
- ABAP SQL SELECT on CDS views return initial values for the virtual element,
- EML READ on BO entities is not possible as EML does not know virtual elements.

SAP Development Learning

Authorization Control

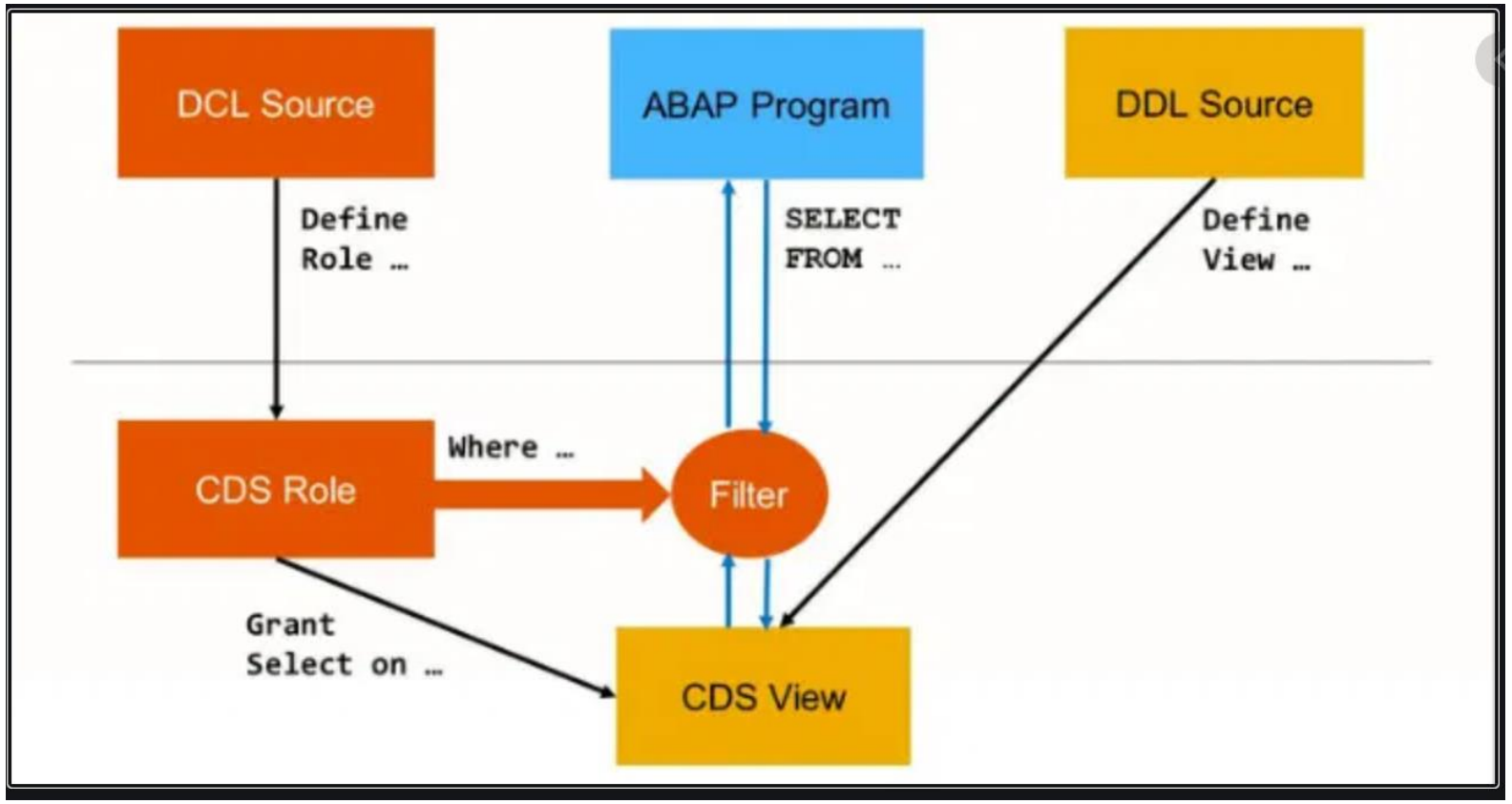
Authorization

SAP Development Learning

- Authorization protects your business object against unauthorized access to data.
- In RAP each read or modify request can be checked via authorization objects **against user roles**.
- Authorization checks in RAP authorization control methods can only check authorization against the state of the BO before the request was executed, the so-called **before image**.
- The authorization check with authorization objects is called from **CDS entities(DCL) in case of read requests** and **from the behavior implementation in case of modify requests**.

Authorization - Read Operations - DCL

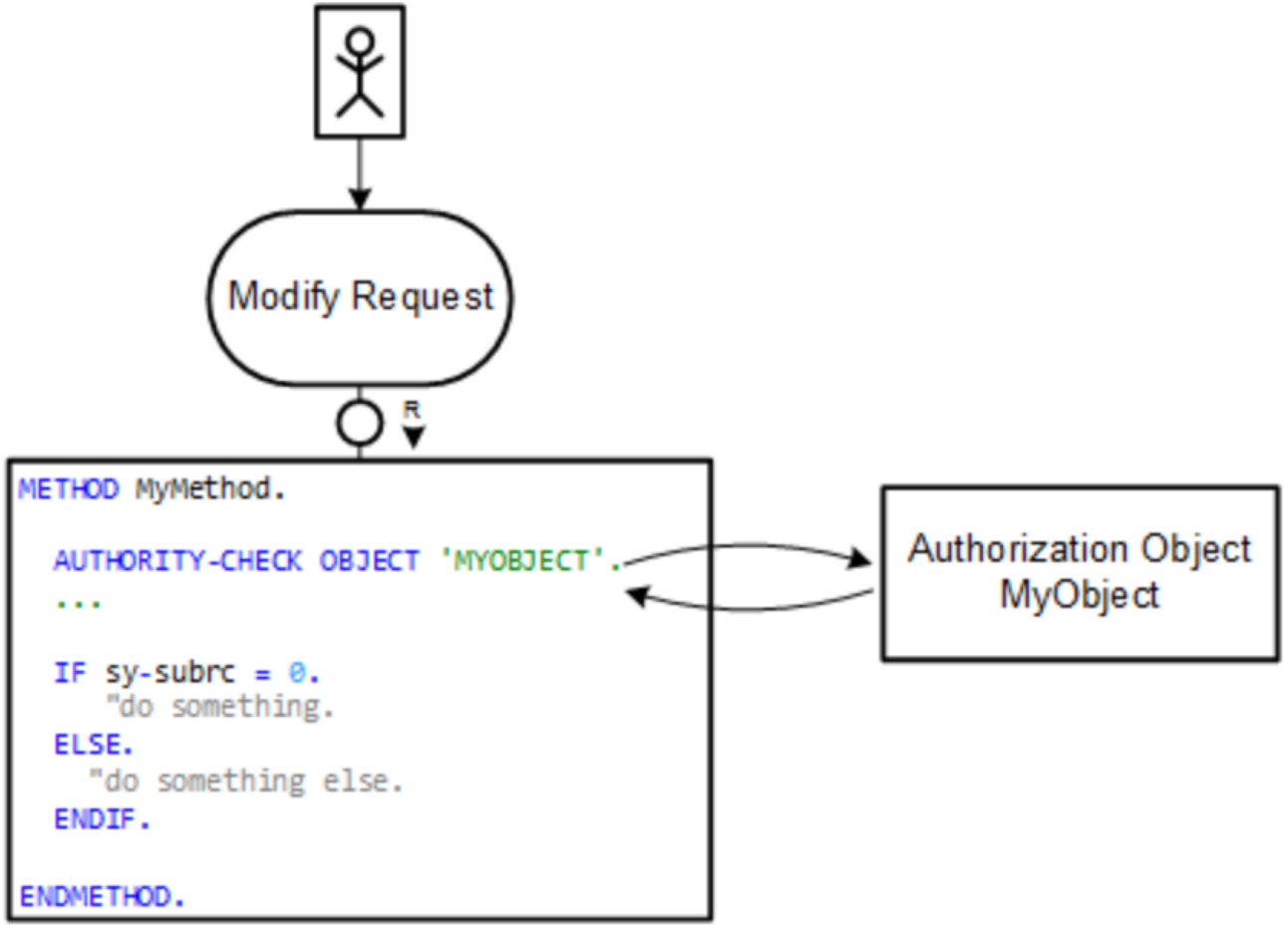
SAP Development Learning



- Based on CDS modeling objects and therefore part of the data-model

Authorization-Modify Operations

SAP Development Learning



Checking Authorization against Authorization Objects

Authorization –Behavior Definition

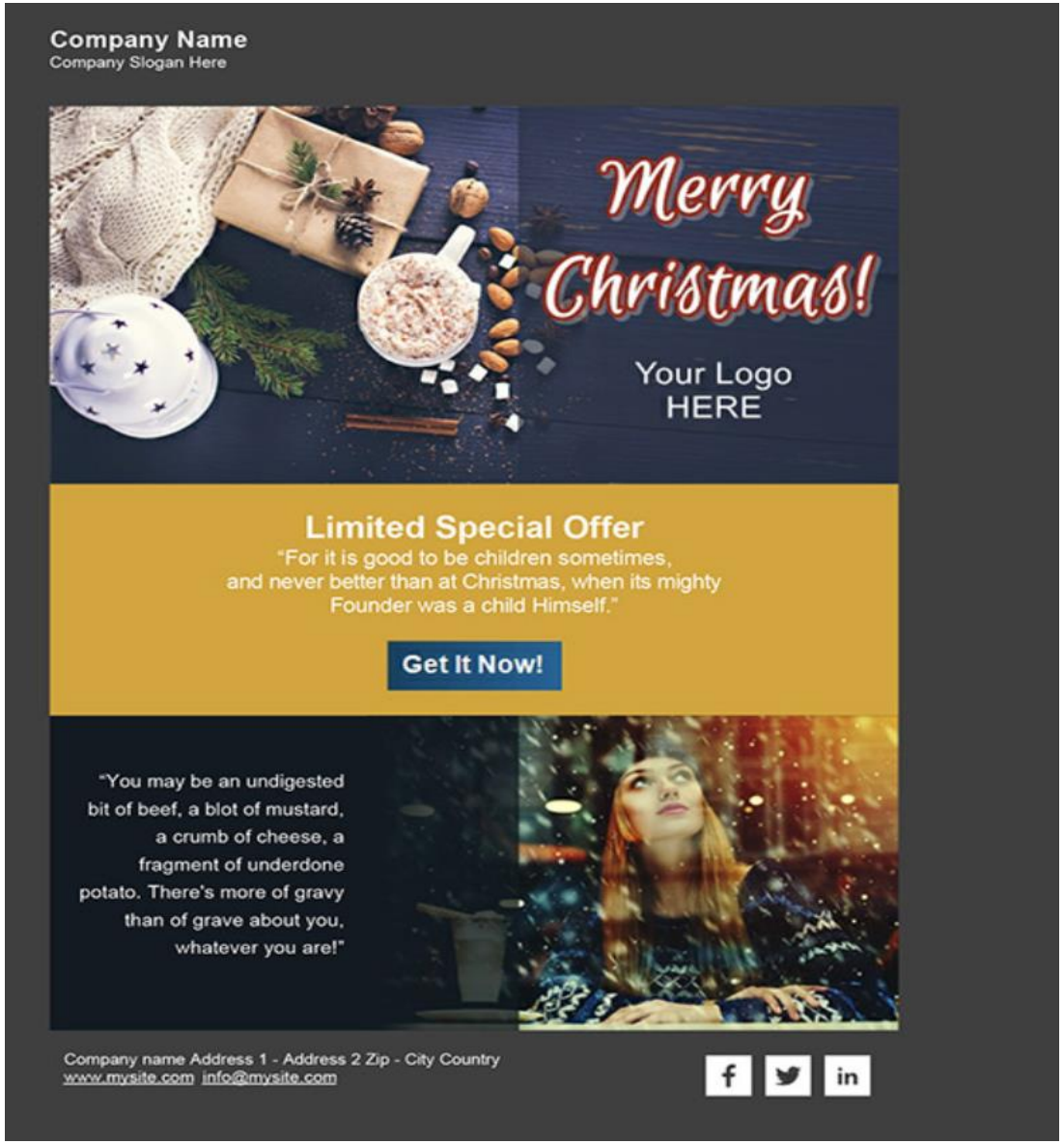
SAP Development Learning

```
{unmanaged | managed };

define behavior for RootEntity [alias AliasRootName]
implementation in class ABAP_CLASS unique
...
 /*Authorization definition for global, or instance, or both */
 authorization master ( global )
                    | ( instance )
                    | ( global, instance )
...
{
 /*Exclude operations from authorization control */
 MyOperation ( authorization : none );

...
}

define behavior for ChildEntity [alias AliasChildName]
implementation in class ABAP_CLASS unique
...
 /*Authorization dependent entities delegate the authority checks to their master entities */
 authorization dependent by _AssocToMaster
...
{
...
association _AssocToMaster;
}
```

- Global authorization is used for all authorization checks that **only depends on the user role**.
- Instance authorization is used for all authorization checks that, in addition to the **user role**, depends on the state of the entity instance. For ex:- It depends on a **field value of the instance**.

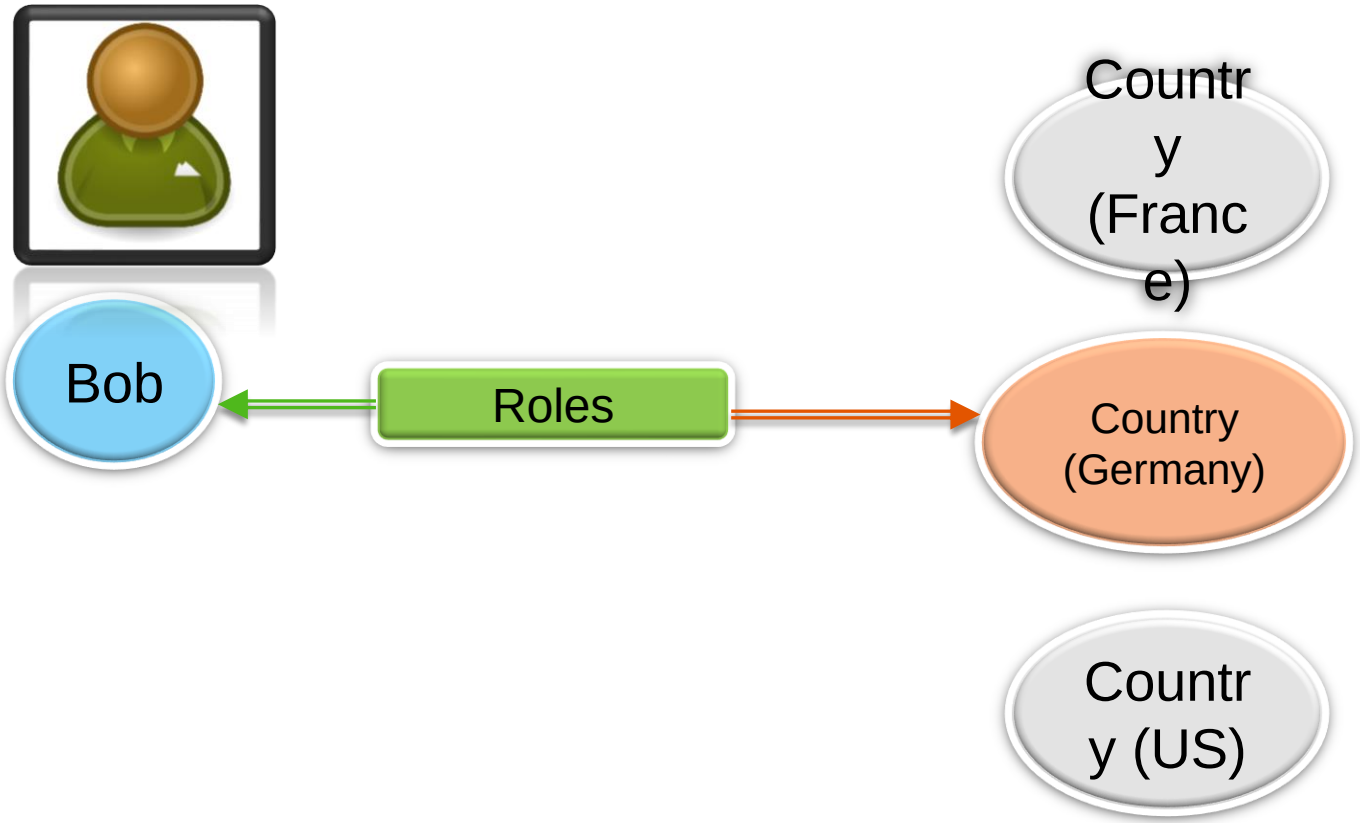
Transactional App For Campaign

SAP Development Learning



Authorization use case

SAP Development Learning



- As a Marketing Manager “Bob” have authorization to see all campaigns but can only Approve/Reject campaigns of his country(DE/Germany).

- Can view Campaigns from US and FR country only.

Authorization

SAP Development Learning

Campaigns-DE

Campaign

Email Content

Target Group

Country DE

Create/Update/Read/Delete

Campaigns – US and FR

Campaign

Email Content

Target Group

Country US

Campaign

Email Content

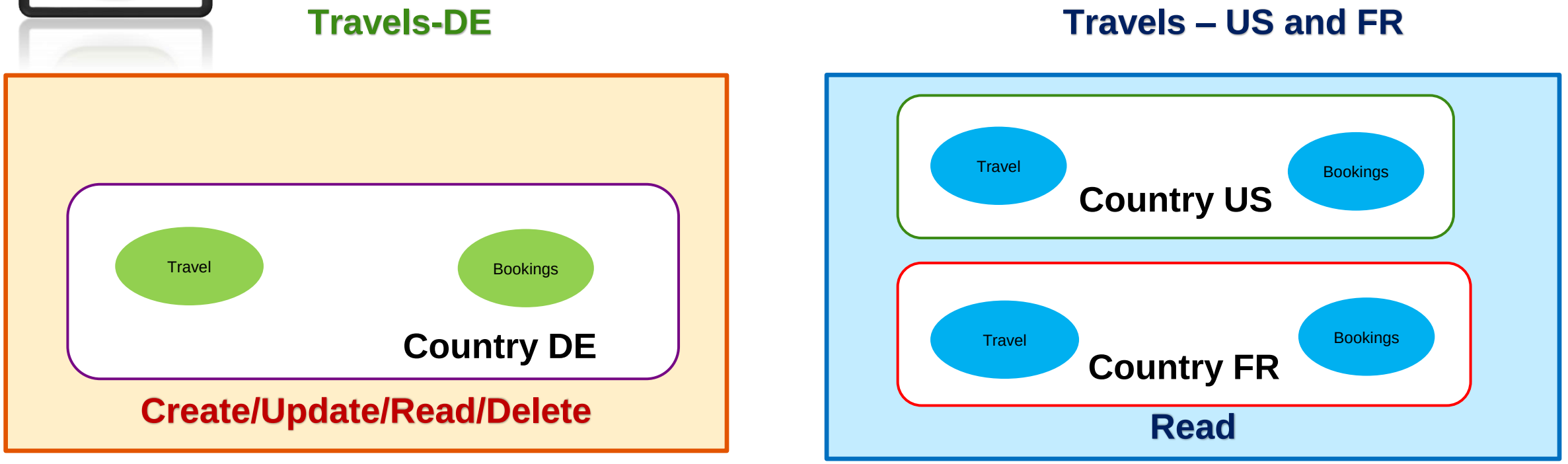
Target Group

Country FR

Read

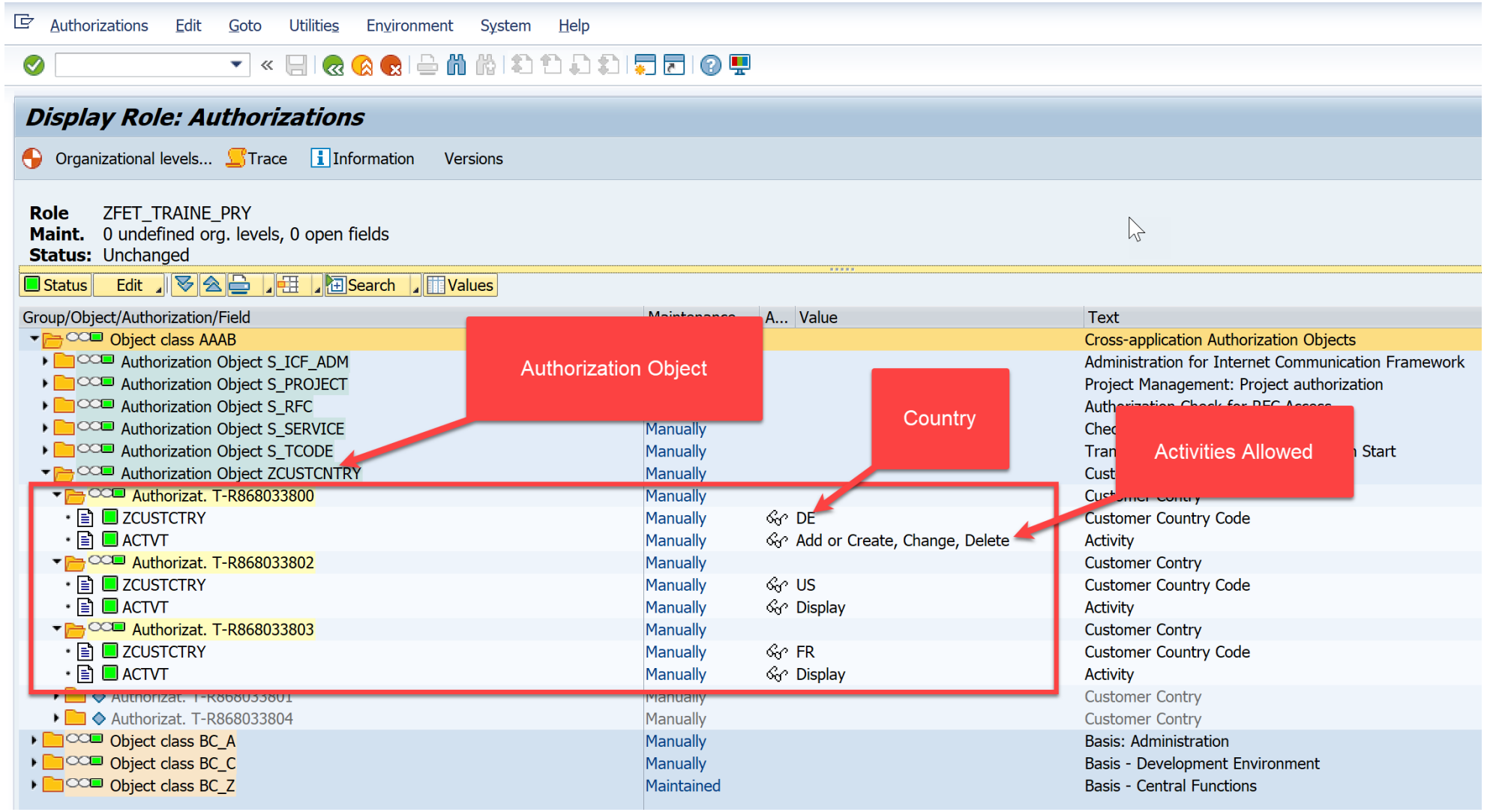
Authorization->How to achieve this????

SAP Development Learning



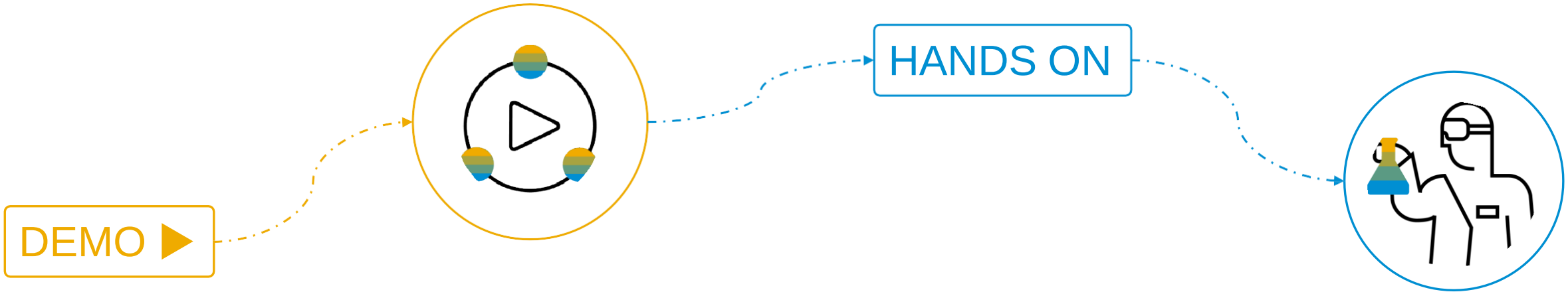
- As a Travel Manager “Bob” have authorization to see all Travels but can only Approve/reject Travels of his country(DE/Germany).
- Can view Travels from US and FR country only.

Authorization- Role



Demo-Authorization

SAP Development Learning



Authorization control

1. DCL
2. Authorization Instance feature control

Thank you.

We provide trainings for all employees in development related roles – across 11 core job families.
Join our SAP Development Learning Jam and explore 23 learning programs and 380+ training courses on programming, technology and methodology.