Table of Contents

Introduction

Let us see how to develop an ABAP RAP application using Managed Scenario for Custom Sample Training Management process.

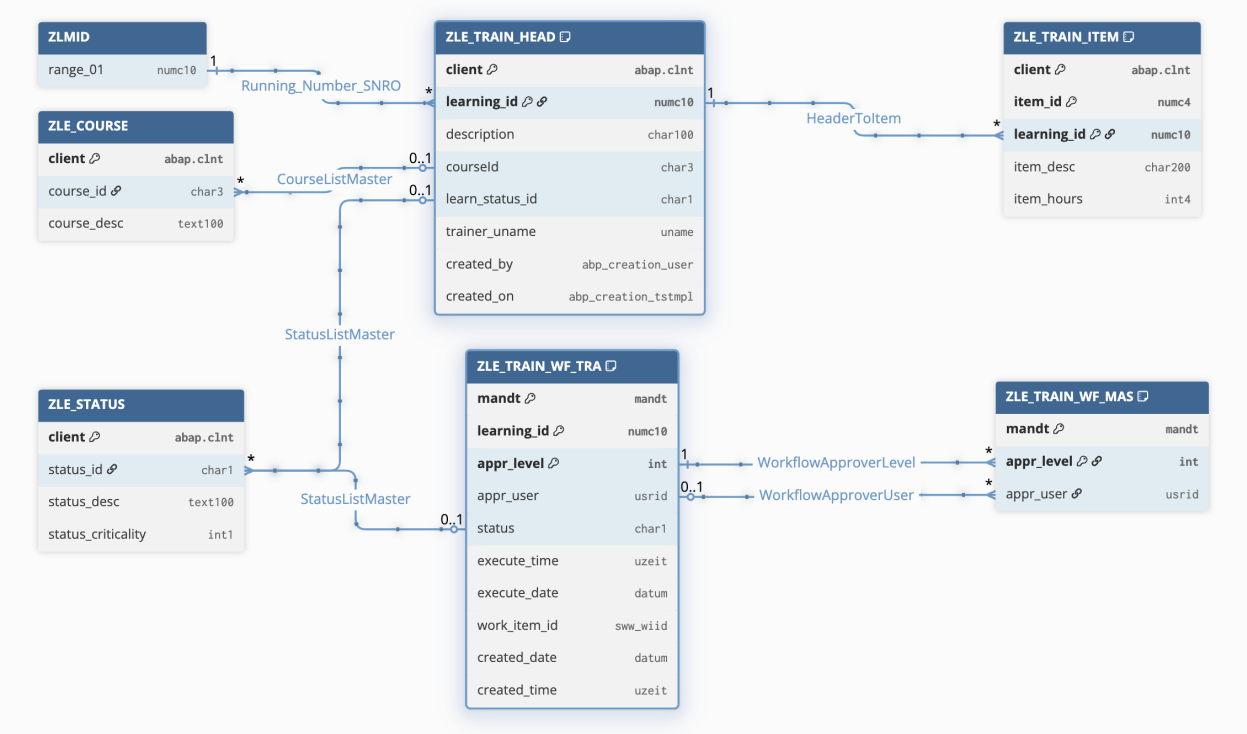
1.Create the Necessary Custom Dictionary Objects for the Master Data, Transaction Data and logs for Workflow

2.Create a Custom SAP RAP Application using Managed Scenario

3.Create a Custom Flexible Workflow and its related Agent rules, Tasks and Classes

4.Integration of Workflow with the RAP Application

1. Design the Necessary Custom tables for the Master and Transaction Data



ZLE_TRAIN_HEAD - Header Transaction Table
ZLE_TRAIN_ITEM - Item Transaction Table
ZLE_COURSE - Course List Master Table
ZLE_STATUS - Status List Master Table
ZLE_TRAIN_WF_MAS - Workflow Approver List Master Table
ZLE_TRAIN_WF_TRA - Workflow Transaction log Table for each learning ID

1.1. Creation of Transaction Tables

Create Dictionary Table: Header Table: **ZLE_TRAIN_HEAD**

```
@EndUserText.label : 'Training Header Table'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zle_train_head {
  key client      : abap.clnt not null;
  key learning_id : numc10 not null;
  description     : abap.char(100);
  course_id       : abap.char(3);
  learn_status_id : abap.char(1);
  trainer_uname   : usrid;
  created_by      : abp_creation_user;
  created_on      : abp_creation_tstmpl;
}
```

Create Dictionary Table: Header Table: **ZLE_TRAIN_ITEM**

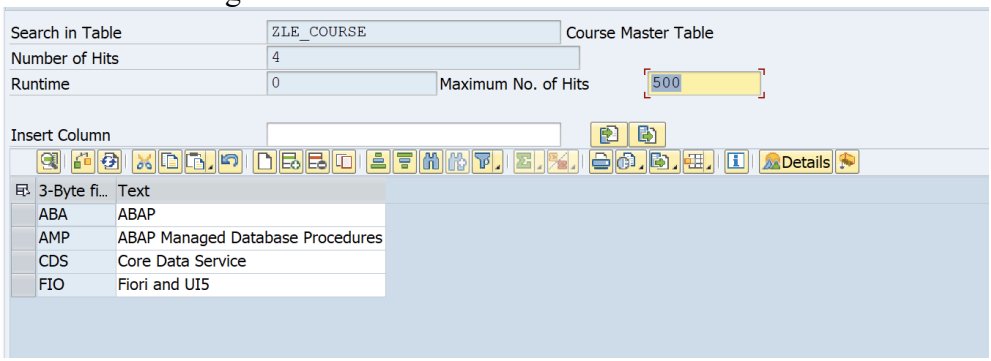
```
@EndUserText.label : 'Training Item Table'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zle_train_item {
  key client      : abap.clnt not null;
  key item_id     : numc4 not null;
  key learning_id : numc10 not null;
  item_desc       : abap.char(200);
  item_hours      : abap.int4;

}
```

1.2. Creation of Master Tables

Create Dictionary Table: Master Course Table: **ZLE_COURSE**

```
@EndUserText.label : 'Course Master Table'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #ALLOWED
define table zle_course {
  key client    : abap.clnt not null;
  key course_id : char3 not null;
  course_desc   : text100 not null;

}
```

Add the Following Course List Data

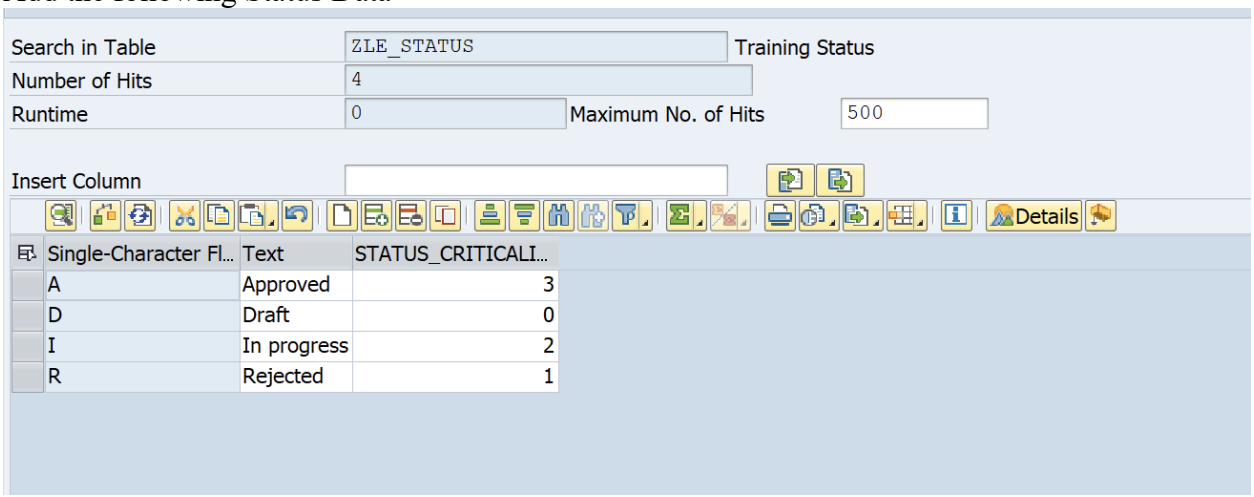


Create Dictionary Table: Master Course Table: **ZLE_STATUS**

```
@EndUserText.label : 'Training Status'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
```

```
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #ALLOWED
define table zle_status {
  key client        : abap.clnt not null;
  key status_id     : char1 not null;
  status_desc       : text100 not null;
  status_criticality : abap.int1;
}
```

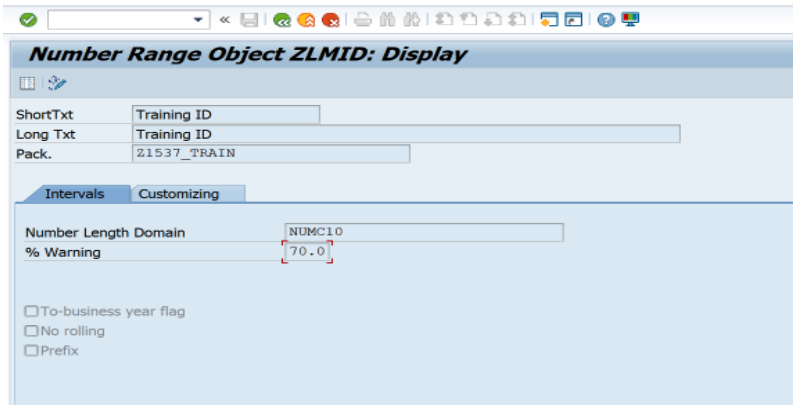
Add the following Status Data



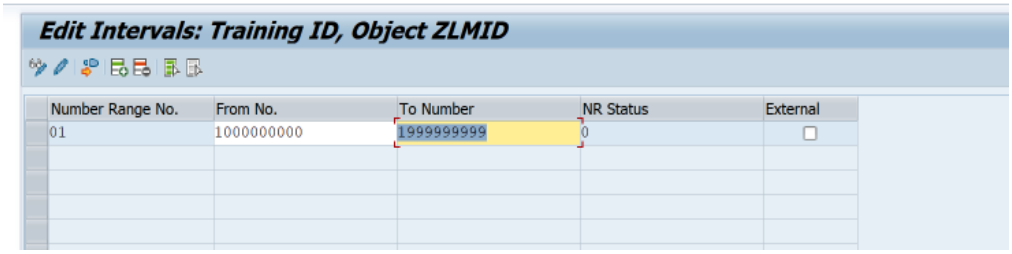
Single-Character Fl...	Text	STATUS_CRITICALI...
A	Approved	3
D	Draft	0
I	In progress	2
R	Rejected	1

1.3. Creation of SNRO

Creation Of SNRO Object: ZLMID



Maintain Interval :



1.4. Creation of Value Help Views

Creation of Value Help Views for the Master Tables to be used as F4 Help:

Value Help view Entity for Course table ZLE_COURSE: **ZLE_COURSE_VH**

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Training Course Value Help'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
   serviceQuality: #X,
   sizeCategory: #S,
   dataClass: #MIXED
}
define view entity ZLE_COURSE_VH
as select from zle_course
{   @EndUserText.label: 'Id'
   key course_id as CourseId,
   @EndUserText.label: 'Description'
   course_desc as CourseDesc
}
```

Value Help view Entity for Course table ZLE_STATUS: **ZLE_STATUS_VH**

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Status Value Help'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
   serviceQuality: #X,
   sizeCategory: #S,
   dataClass: #MIXED
}
define view entity ZLE_STATUS_VH
as select from zle_status
{
   @EndUserText.label: 'Id'
   key status_id as StatusId,
   @EndUserText.label: 'Status'
   status_desc as StatusDesc,
   status_criticality as StatusCriticality
}
```

Value Help view Entity for Trainer Master: **ZLE_TRAINER_VH**

We will form this value help view based on the HR available PERNR, using table PA0105 and PA0001

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Training : Trainer Value help'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
   serviceQuality: #X,
   sizeCategory: #S,
   dataClass: #MIXED
}
define view entity ZLE_TRAINER_VH
  as select from pa0105 as a
   inner join   pa0001 as b on  b.pernr = a.pernr
                      and b.endda >= $session.system_date
                      and b.begda <= $session.system_date
{
  key a.usrid as UserId,
     a.pernr as PersonnalNumber,
     b.ename as Name,
     b.plans as JobPosition
```

```
}
where
    a.subty = '0001'
 and a.endda >= $session.system_date
 and a.begda <= $session.system_date
```

2. RAP Managed Scenario

Let us develop the RAP Managed Scenario Application that allow employee to raise a request. In this Application the following are used

- **Early Numbering** for Header and Item to fetch the key number from SNRO object
- **Determinations** to set default value for status on create
- **Validations** to set the mandatory fields for Header and Item
- **Instance Features** to control the visibility of the Update, Delete and Send Approval buttons
- **Custom Action Button** for Send Approval to trigger workflow

2.1. Creation of Base Interface View Entities

Root View entity for table ZLE_TRAIN_HEAD: **ZBI_TRAIN_HEAD**

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'BI : Training Header'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
   serviceQuality: #X,
   sizeCategory: #S,
   dataClass: #MIXED
}
define root view entity zbi_train_head
as select from zle_train_head
//Compositions Links
composition[1..*] of zbi_train_item as _TrainItem

//Associations Links
association[1..1] to ZLE_COURSE_VH  as _CourseMaster on $projection.CourseId    = _CourseMaster.CourseId
association[1..1] to ZLE_STATUS_VH  as _StatusMaster on $projection.LearnStatusId = _StatusMaster.StatusId
association[1..1] to ZLE_TRAINER_VH as _UserMaster   on $projection.TrainerUname  = _UserMaster.UserId
{
   key learning_id as LearningId,
   description     as Description,
   course_id       as CourseId,
   learn_status_id as LearnStatusId,
   _StatusMaster.StatusCriticality as StatusCriticality,
   trainer_uname   as TrainerUname,
   @Semantics.user.createdBy: true
   created_by      as CreatedBy,
   @Semantics.systemDateTime.createdAt: true
   created_on      as CreatedOn,

   //Master Texts
   _CourseMaster.CourseDesc as CourseDescription,
   _StatusMaster.StatusDesc as StatusDescription,
   _UserMaster.Name as TrainerName,

   //Associations
   _TrainItem,
   _CourseMaster,
   _StatusMaster,
   _UserMaster
}
```

View entity for table ZLE_TRAIN_ITEM: **ZBI_TRAIN_ITEM**

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'BI : Training Item'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
   serviceQuality: #X,
   sizeCategory: #S,
   dataClass: #MIXED
}
define view entity zbi_train_item
as select from zle_train_item
///Association Links
association to parent zbi_train_head  as _TrainHead on $projection.LearningId = _TrainHead.LearningId
{
   key item_id     as ItemId,
   key learning_id as LearningId,
   item_desc       as ItemDesc,
   item_hours      as ItemHours,

   //Association
   _TrainHead
}
```

2.2. Creation of Consumption or Projection View Entities for the Base Interface View Entities:

Root Consumption View entity ZBI_TRAIN_HEAD: **ZCO_TRAIN_HEAD**

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Training : Head Consumption View'
@Metadata.ignorePropagatedAnnotations: true
@Metadata.allowExtensions: true
define root view entity ZCO_TRAIN_HEAD
provider contract transactional_query
as projection on zbi_train_head
{
   key LearningId,
   Description,
   @ObjectModel.text.element: ['CourseDescription']
   CourseId,
   @ObjectModel.text.element: ['StatusDescription']
   LearnStatusId,
   StatusCriticality,
    @ObjectModel.text.element: ['TrainerName']
   TrainerUname,
   CreatedBy,
   CreatedOn,
   CourseDescription,
   StatusDescription,
   TrainerName,
   /* Associations */
   _TrainItem :redirected to composition child ZCO_TRAIN_ITEM,
   _CourseMaster,
   _StatusMaster,
   _UserMaster
}
```

Consumption View entity ZBI_TRAIN_ITEM: **ZCO_TRAIN_ITEM**

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Training : Item Consumption View'
@Metadata.ignorePropagatedAnnotations: true
@Metadata.allowExtensions: true
```

```
define view entity ZCO_TRAIN_ITEM as projection on zbi_train_item
{
   key ItemId,
   key LearningId,
   ItemDesc,
   ItemHours,
   /* Associations */
   _TrainHead :redirected to parent ZCO_TRAIN_HEAD
}
```

2.3. Creation of Meta Data Extensions

Meta Data Extensions are created for Consumption View Entities here let us create for View entity ZCO_TRAIN_HEAD: **ZMDE_TRAIN_HEAD**

```
@Metadata.layer: #CORE
@UI:{ headerInfo:{
  typeName: 'Training',
  typeNamePlural: 'Trainings',
  title:{ type:#STANDARD,
    label:'Training' ,
    value:'LearningId'}

},
presentationVariant: [{ sortOrder: [{ by: 'LearningId',direction: #DESC }] }] }
@Search.searchable: true
annotate entity ZCO_TRAIN_HEAD with
{
  //UI Facets
  @UI.facet: [{ id:'Learning', purpose : #STANDARD, type :#IDENTIFICATION_REFERENCE, position : 10, label: 'Learning'},
            { id:'Items', purpose : #STANDARD, type :#LINEITEM_REFERENCE, position : 20, label: 'Items', targetElement: '_TrainItem'} ]

  @Search.defaultSearchElement: true
  @Search.fuzzinessThreshold: 0.5
  @UI.identification: [ { position: 10 } ]
  @UI.lineItem: [ { position: 10 } ]
  @EndUserText.label: 'Learning Id'
  LearningId;

  @Search.defaultSearchElement: true
  @Search.fuzzinessThreshold: 0.7
  @UI.identification: [ { position: 20 } ]
  @UI.lineItem: [ { position: 20 } ]
  @EndUserText.label: 'Description'
  Description;

  @UI.identification: [ { position: 30 } ]
  @UI.lineItem: [ { position: 30 } ]
  @EndUserText.label: 'Course'
  @Consumption.valueHelpDefinition: [{ entity:{ name: 'ZLE_COURSE_VH', element: 'CourseId' }}]
  @UI.selectionField: [{ position : 30 }]
  CourseId;
  @UI.identification: [{ position: 40, criticality: 'StatusCriticality' }]
  @UI.lineItem: [{ position: 40, criticality: 'StatusCriticality' },
              { type : #FOR_ACTION, dataAction: 'sendApproval', label: 'Send Approval' }]
  @EndUserText.label: 'Learning Status'
  @Consumption.valueHelpDefinition: [{ entity:{ name: 'ZLE_STATUS_VH', element: 'StatusId' }}]
  @UI.selectionField: [{ position : 40 }]

  LearnStatusId;
  @UI.identification: [ { position: 50 } ]
  @UI.lineItem: [ { position: 50 } ]
  @Consumption.valueHelpDefinition: [{ entity:{ name: 'ZLE_TRAINER_VH', element: 'UserId' }}]
  @EndUserText.label: 'Trainer Name'
  TrainerUname;
  @UI.hidden: true
  CreatedBy;
  @UI.hidden: true
  CreatedOn;
```

```
}
```

Meta Data Extensions for View entity ZCO_TRAIN_ITEM: **ZMDE_TRAIN_ITEM**

```
@Metadata.layer: #CORE
@UI:{ headerInfo:{
  typeName: 'Item',
  typeNamePlural: 'Items',
  title:{ type:#STANDARD,
    label:'Item' ,
    value:'ItemId'}

},
presentationVariant: [{ sortOrder: [{ by: 'LearningId',direction: #DESC }] }] }
annotate entity ZCO_TRAIN_ITEM with
{
  @UI.facet: [{ id:'Item', purpose : #STANDARD, type :#IDENTIFICATION_REFERENCE, position : 10, label: 'Item'}]
  @UI.identification: [ { position: 10 } ]
  @UI.lineItem: [ { position: 10 } ]
  @EndUserText.label: 'Item Id'
  ItemId;
  @UI.hidden: true
  LearningId;
  @UI.identification: [ { position: 20 } ]
  @UI.lineItem: [ { position: 20 } ]
  @EndUserText.label: 'Description'
  ItemDesc;
  @UI.identification: [ { position: 30 } ]
  @UI.lineItem: [ { position: 30 } ]
  @EndUserText.label: 'Hours'
  ItemHours;
}
```

2.4. Creation of Behavior Definition

Behavior Definition for the Base Interface Root view Entity **ZBI_TRAIN_HEAD**

Implementation Type: Managed

Here we need to declare the Early numbering, Determinations, Validations, Action the same methods will be implemented in Behavior implementation

```
managed implementation in class zbp_bi_train_head unique;
strict ( 2 );

define behavior for zbi_train_head alias train_head
persistent table ZLE_TRAIN_HEAD
lock master
authorization master ( instance )
//etag master <field_name>
early numbering
{
  //Read only Fields
  field ( readonly ) LearningId,LearnStatusId,CreatedBy,CreatedOn;
  //Mandatory Fields
  field ( mandatory ) CourseId,TrainerUname;

  //Determinations
  determination setStatusDraft on modify { create; }

  //Validations
  validation mandatoryCourseId on save { create; field CourseId; }
  validation mandatoryTrainerName on save { create; field TrainerUname; }

  //Actions
  action ( features : instance ) sendApproval result [1] $self;
```

```
  create;
  update (features : instance );
  delete (features : instance );
  association _TrainItem { create (features : instance ); }

  //Mappings Fields for the Header Table
   mapping for ZLE_TRAIN_HEAD {
    LearningId    = learning_id;
    Description   = description;
    CourseId      = course_id;
    LearnStatusId = learn_status_id;
    TrainerUname  = trainer_uname;
    CreatedBy     = created_by;
    CreatedOn     = created_on;
   }
}

define behavior for zbi_train_item alias train_item
persistent table ZLE_TRAIN_ITEM
lock dependent by _TrainHead
authorization dependent by _TrainHead
//etag master <field_name>
early numbering
{
  //Read Only Fields
  field ( readonly ) ItemId,LearningId;
  //Mandatory Fields
  field ( mandatory ) ItemDesc,ItemHours;

  //Validations
  validation MandatoryItemDesc on save { create; field ItemDesc; }
  validation MandatoryItemHours on save { create; field ItemHours; }

  update(features : instance );
  delete(features : instance );
  association _TrainHead;

   //Mappings Fields for the Item Table
   mapping for ZLE_TRAIN_ITEM {
   ItemId     = item_id;
   LearningId = learning_id;
   ItemDesc   = item_desc;
   ItemHours  = item_hours;
   }
}
```

Behavior Definition for the Consumption Root view Entity **ZCO_TRAIN_HEAD**

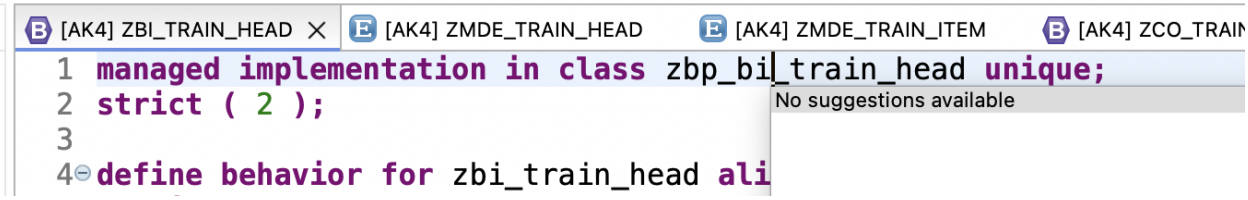
Implementation Type: Projection

```
projection;
strict ( 2 );

define behavior for ZCO_TRAIN_HEAD alias train_head
{
  use create;
  use update;
  use delete;
  use action sendApproval;

  use association _TrainItem { create; }
}

define behavior for ZCO_TRAIN_ITEM alias train_item
{
  use update;
  use delete;
```

```
 use association _TrainHead;
}
```

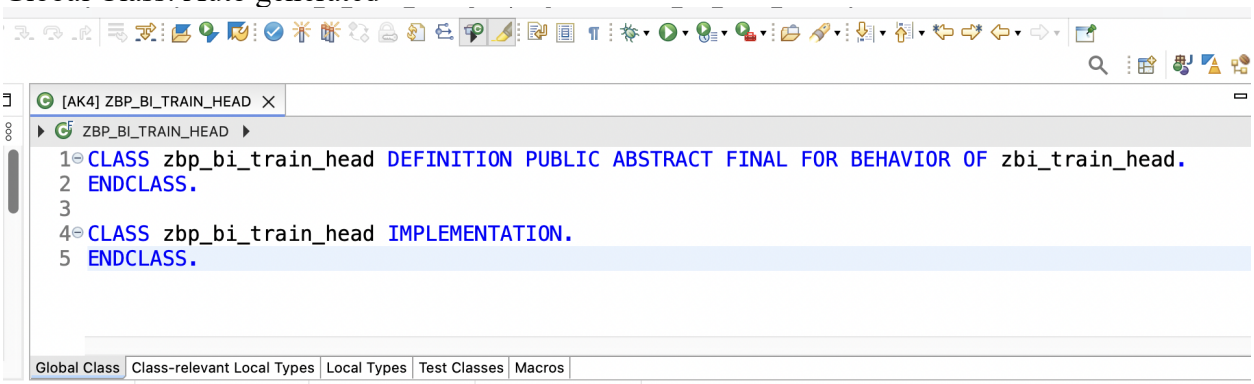
2.5. Creation of Behavior Implementation Class

Here I have created one class for both Header and item but we can also create separate classes accordingly. In eclipse the shortcut will be place the cursor on the behavior definition and give Ctrl + 1 which will show you suggestion to create class.

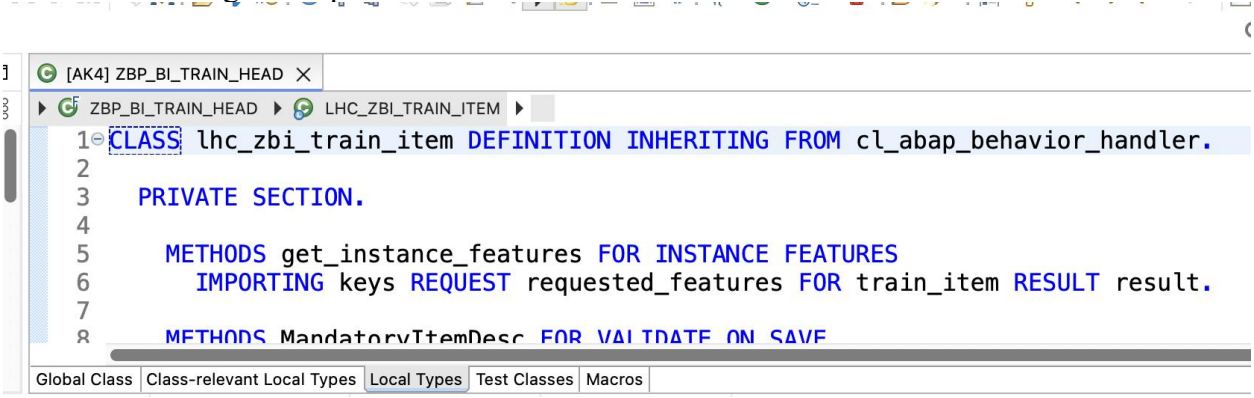


Class name: **ZBP_BI_TRAIN_HEAD**
Following which are declared in the Behavior definition is implemented here:
->Mandatory and Read Only Fields for Header
->Early Numbering for Header and Item
->Determinations to set default values
->Instance Features to Enable the Buttons based on Status
->Custom Action Button to Trigger workflow

Global Class: Auto generated



Local Types: Declarations will be auto created but we need add code to implementation, this is where the core logics are present



Code for the Local type class:

```
*****************************************************************
* Definition for the Item Entity                                *
*****************************************************************
CLASS lhc_zbi_train_item DEFINITION INHERITING FROM cl_abap_behavior_handler.
  PRIVATE SECTION.

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

    METHODS MandatoryItemDesc FOR VALIDATE ON SAVE
      IMPORTING keys FOR train_item~MandatoryItemDesc.

    METHODS MandatoryItemHours FOR VALIDATE ON SAVE
      IMPORTING keys FOR train_item~MandatoryItemHours.

ENDCLASS.
*****************************************************************
* Implementation for the Item Entity                            *
*****************************************************************
CLASS lhc_zbi_train_item IMPLEMENTATION.

* Method 1: Instance Feature to Enable Update and Delete Button for Item Entity
  METHOD get_instance_features.

    READ ENTITIES OF zbi_train_head IN LOCAL MODE
      ENTITY train_head BY \_TrainItem
      FIELDS ( learningid itemid )
      WITH CORRESPONDING #( keys )
      RESULT DATA(lt_items).

    READ ENTITY IN LOCAL MODE zbi_train_item
       BY \_TrainHead
      FIELDS ( learningid learnstatusid )
      WITH CORRESPONDING #( lt_items )
      RESULT DATA(lt_heads).

    result = VALUE #( FOR ls_head IN lt_heads
                FOR ls_item IN lt_items WHERE ( learningid = ls_head-learningid )
                    ( %key = ls_item-%key
                      %features-%update = COND #( WHEN ls_head-learnstatusid = 'D'
                                        THEN if_abap_behv=>fc-o-enabled
                                        ELSE if_abap_behv=>fc-o-disabled )

                      %features-%delete = COND #( WHEN ls_head-learnstatusid = 'D'
                                        THEN if_abap_behv=>fc-o-enabled
                                        ELSE if_abap_behv=>fc-o-disabled ) ) ).


  ENDMETHOD.

* Method 2: Mandatory Check for Item Description field in Item Entity
  METHOD MandatoryItemDesc.
    READ ENTITY IN LOCAL MODE zbi_train_item
     FIELDS ( itemdesc )
     WITH CORRESPONDING #( keys )
     RESULT DATA(lt_item).
    LOOP AT lt_item ASSIGNING FIELD-SYMBOL(<fs_item>).
     IF <fs_item>-itemdesc IS INITIAL.
       APPEND VALUE #( %tky = <fs_item>-%tky ) TO failed-train_item.
       APPEND VALUE #( %tky              = <fs_item>-%tky
                  %element-itemdesc = if_abap_behv=>mk-on
                  %msg              = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                      text     = 'Description is Mandatory' ) ) TO reported-train_item.

     ENDIF.
```

```
  ENDLOOP.
 ENDMETHOD.

* Method 3: Mandatory Check for Item Hours field in Item Entity
 METHOD MandatoryItemHours.
  READ ENTITY IN LOCAL MODE zbi_train_item
   FIELDS ( itemdesc )
   WITH CORRESPONDING #( keys )
   RESULT DATA(lt_item).
  LOOP AT lt_item ASSIGNING FIELD-SYMBOL(<fs_item>).
   IF <fs_item>-itemhours IS INITIAL.
     APPEND VALUE #( %tky = <fs_item>-%tky ) TO failed-train_item.
     APPEND VALUE #( %tky              = <fs_item>-%tky
                %element-itemhours = if_abap_behv=>mk-on
                %msg               = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                                            text     = 'Hours is Mandatory' ) ) TO reported-train_item.
   ENDIF.
  ENDLOOP.
 ENDMETHOD.

ENDCLASS.

*********************************************************************
* Definition for the Header Entity                                  *
*********************************************************************
CLASS lhc_train_head DEFINITION INHERITING FROM cl_abap_behavior_handler.
 PRIVATE SECTION.

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

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

  METHODS setstatusdraft FOR DETERMINE ON MODIFY
   IMPORTING keys FOR train_head~setstatusdraft.

  METHODS mandatorycourseid FOR VALIDATE ON SAVE
   IMPORTING keys FOR train_head~mandatorycourseid.

  METHODS mandatorytrainername FOR VALIDATE ON SAVE
   IMPORTING keys FOR train_head~mandatorytrainername.

  METHODS sendapproval FOR MODIFY
   IMPORTING keys FOR ACTION train_head~sendapproval RESULT result.

  METHODS earlynumbering_cba_trainitem FOR NUMBERING
   IMPORTING entities FOR CREATE train_head\_trainitem.

  METHODS earlynumbering_create FOR NUMBERING
   IMPORTING entities FOR CREATE train_head.


ENDCLASS.

*********************************************************************
* Implementation for the Header Entity                              *
*********************************************************************
CLASS lhc_train_head IMPLEMENTATION.

* Method 1:Instance Authorizations -> In this example it is not used
 METHOD get_instance_authorizations.
 ENDMETHOD.

* Method 2:Early Numbering for Training Header
 METHOD earlynumbering_create.

   "Method to Handle Header Number Range
  DATA(lt_entites) = entities.
  DELETE lt_entites WHERE learningid IS NOT INITIAL.
```

```
    "Get the Number Range
    TRY.
        cl_numberrange_runtime=>number_get(
          EXPORTING
            nr_range_nr       = '01'
            object            = 'ZLMID'
            quantity          = CONV #( lines( lt_entites ) )
          IMPORTING
            number            = DATA(lv_latest_num)
            returncode        = DATA(lv_ret_code)
            returned_quantity = DATA(lv_ret_quan)
        ).
      CATCH cx_nr_object_not_found INTO DATA(lo_exp_object).
      CATCH cx_number_ranges INTO DATA(lo_exp_number).
        "Exception or Error Handling
        LOOP AT lt_entites INTO DATA(ls_entities).
          APPEND VALUE #( %cid = ls_entities-%cid
                          %key = ls_entities-%key )
                 TO failed-train_head.

          APPEND VALUE #( %cid = ls_entities-%cid
                          %key = ls_entities-%key
                          %msg = lo_exp_number )
                 TO reported-train_head.
        ENDLOOP.
        EXIT.
    ENDTRY.
    ASSERT lv_ret_quan = lines( lt_entites ).
    DATA(lv_curr_num) = CONV zlm_head-learning_id( lv_latest_num - lv_ret_quan ).

    lv_curr_num = lv_curr_num + 1.
    CLEAR ls_entities.
    LOOP AT lt_entites INTO ls_entities.
      "Number Range Assignment
      APPEND VALUE #( %cid     = ls_entities-%cid
                      learningid = lv_curr_num )
             TO mapped-train_head.
    ENDLOOP.
  ENDMETHOD.

* Method 3:Early Numbering for Item
  METHOD earlynumbering_cba_Trainitem.

    DATA : lv_max_item TYPE zle_train_item-item_id.

    READ ENTITIES OF zbi_train_head IN LOCAL MODE
    ENTITY train_head BY \_TrainItem
    FROM CORRESPONDING #( entities )
    LINK DATA(lt_link).

    LOOP AT entities ASSIGNING FIELD-SYMBOL(<ls_group_entity>) GROUP BY <ls_group_entity>-learningid.
      "Link Data will be empty for creating first record
      lv_max_item = REDUCE #( INIT lv_max = CONV zle_train_item-item_id( '0' )
                    FOR ls_link IN lt_link USING KEY entity WHERE ( source-learningid = <ls_group_entity>-learningid )
                    NEXT lv_max = COND zle_train_item-item_id( WHEN lv_max < ls_link-target-itemid
                                                               THEN ls_link-target-itemid
                                                               ELSE lv_max ) ).
      "Increment to 10 and Assignment to item so that the item numbering will be 10,20,30 etc..
      LOOP AT entities ASSIGNING FIELD-SYMBOL(<ls_entities>)
                  USING KEY entity
                  WHERE learningid = <ls_group_entity>-learningid.
        LOOP AT <ls_entities>-%target ASSIGNING FIELD-SYMBOL(<ls_item>).
          APPEND CORRESPONDING #( <ls_item> ) TO mapped-train_item ASSIGNING FIELD-SYMBOL(<ls_new_item>).
          IF <ls_item>-itemid IS INITIAL.
            lv_max_item += 10.
            <ls_new_item>-itemid = lv_max_item.
          ENDIF.
        ENDLOOP.
      ENDLOOP.
    ENDLOOP.
```

```
  ENDMETHOD.

* Method 4:Instance features to control update, Send Approval, Delete Buttons
  METHOD get_instance_features.
    READ ENTITIES OF zbi_train_head IN LOCAL MODE
      ENTITY train_head
      FIELDS ( learnstatusid )
      WITH CORRESPONDING #( keys )
      RESULT DATA(lt_head).

    result = VALUE #( FOR ls_head IN lt_head ( %tky = ls_head-%tky
                                %features-%update = COND #( WHEN ls_head-learnstatusid = 'D'
                                            THEN if_abap_behv=>fc-o-enabled
                                            ELSE if_abap_behv=>fc-o-disabled )
                                %features-%action-sendApproval =  COND #( WHEN ls_head-learnstatusid = 'D'
                                            THEN if_abap_behv=>fc-o-enabled
                                            ELSE if_abap_behv=>fc-o-disabled )
                                %features-%delete = COND #( WHEN ls_head-learnstatusid = 'D'
                                            THEN if_abap_behv=>fc-o-enabled
                                            ELSE if_abap_behv=>fc-o-disabled )
                                %features-%assoc-_TrainItem = COND #( WHEN ls_head-learnstatusid = 'D'
                                            THEN if_abap_behv=>fc-o-enabled
                                            ELSE if_abap_behv=>fc-o-disabled ) ) ).

  ENDMETHOD.

* Method 5:Set Status as Draft(D) for newly created Records by Default
  METHOD setStatusDraft.

    READ ENTITIES OF zbi_train_head IN LOCAL MODE
     ENTITY train_head
     FIELDS ( learnstatusid )
     WITH CORRESPONDING #( keys )
     RESULT  DATA(lt_head).

    DELETE lt_head WHERE learnstatusid IS NOT INITIAL.
    CHECK lt_head IS NOT INITIAL.

    MODIFY ENTITIES OF zbi_train_head IN LOCAL MODE
    ENTITY train_head
    UPDATE FIELDS ( learnstatusid )
    WITH VALUE #(  FOR ls_head IN lt_head
                ( %tky         = ls_head-%tky
                  learnstatusid = 'D' )  ).
  ENDMETHOD.

* Method 6:Validation and Mandatory check for  Course ID
  METHOD mandatoryCourseId.

    READ ENTITIES OF zbi_train_head IN LOCAL MODE
      ENTITY train_head
      FIELDS ( courseid )
      WITH CORRESPONDING #( keys )
      RESULT DATA(lt_heads).

    SELECT course_id FROM zle_course INTO TABLE @DATA(lt_course_master).

    LOOP AT lt_heads ASSIGNING FIELD-SYMBOL(<fs_head>).
     ""Invalidate State Messages
     APPEND VALUE #( %tky = <fs_head>-%tky
                %state_area = 'MANDATORY_COURSEID' ) TO reported-train_head.

     IF <fs_head>-courseid IS INITIAL.
       APPEND VALUE #( %tky = <fs_head>-%tky ) TO failed-train_head.
       APPEND VALUE #( %tky              = <fs_head>-%tky
                  %state_area       = 'MANDATORY_COURSEID'
                  %element-courseid = if_abap_behv=>mk-on
                  %msg              = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                      text     = 'Course ID is Mandatory' ) ) TO reported-train_head.
```

```
    ELSEIF <fs_head>-courseid IS NOT INITIAL AND NOT line_exists( lt_course_master[ course_id = <fs_head>-courseid ] ).
      APPEND VALUE #( %tky              = <fs_head>-%tky
                      %state_area       = 'MANDATORY_COURSEID'
                      %element-courseid = if_abap_behv=>mk-on
                      %msg              = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                                                 text     = 'Invalid Course ID' ) ) TO reported-train_head.
    ENDIF.

   ENDLOOP.

  ENDMETHOD.

* Method 7:Validation and Mandatory Check for Trainer ID
  METHOD mandatoryTrainerName.
    READ ENTITIES OF zbi_train_head IN LOCAL MODE
      ENTITY train_head
      FIELDS ( traineruname )
      WITH CORRESPONDING #( keys )
      RESULT DATA(lt_heads).

    SELECT * FROM zle_trainer_vh INTO TABLE @DATA(lt_users).

    LOOP AT lt_heads ASSIGNING FIELD-SYMBOL(<fs_head>).
     ""Invalidate State Messages
     APPEND VALUE #( %tky        = <fs_head>-%tky
                  %state_area = 'MANDATORY_USERNAME' ) TO reported-train_head.

     IF <fs_head>-traineruname IS INITIAL.
       APPEND VALUE #( %tky = <fs_head>-%tky ) TO failed-train_head.
       APPEND VALUE #( %tky = <fs_head>-%tky
                    %state_area = 'MANDATORY_USERNAME'
                    %element-traineruname = if_abap_behv=>mk-on
                    %msg = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                                  text     = 'Trainer User Name is Mandatory' ) ) TO reported-train_head.

     ELSEIF <fs_head>-traineruname IS NOT INITIAL AND NOT line_exists( lt_users[ userid = <fs_head>-traineruname ] ).
       APPEND VALUE #( %tky = <fs_head>-%tky
                    %state_area = 'MANDATORY_USERNAME'
                    %element-traineruname = if_abap_behv=>mk-on
                    %msg = new_message_with_text( severity = if_abap_behv_message=>severity-error
                                                  text     = 'Invalid Trainer User Name' ) ) TO reported-train_head.
     ENDIF.

    ENDLOOP.
  ENDMETHOD.

* Method 8:Action to send the approval -> Workflow is Triggered
  METHOD sendApproval.
    ""->Read the Selected Data
    READ ENTITIES OF zbi_train_head IN LOCAL MODE
      ENTITY train_head BY \_TrainItem
      ALL FIELDS
      WITH CORRESPONDING #( keys )
      RESULT DATA(lt_items).

    READ ENTITY IN LOCAL MODE zbi_train_item
      BY \_TrainHead
     FIELDS ( learningid learnstatusid )
     WITH CORRESPONDING #( lt_items )
     RESULT DATA(lt_heads).

    DATA(ls_head) = VALUE #( lt_heads[ 1 ] OPTIONAL ).
    ""->Only Learning ID is Enough for us in the structure
    DATA(ls_learning) = VALUE zle_train_head( learning_id = ls_head-LearningId ).


    ""->Trigger Workflow -> *DO NOT USE COMMIT in any place as it is not allowed*
    DATA :lo_wf TYPE REF TO zle_cl_train_workflow.
    CREATE OBJECT lo_wf
      EXPORTING
```

```
    lv_key_number = ls_learning-learning_id.

  CALL METHOD lo_wf->trigger_start_wf
    EXPORTING
      learning_det = ls_learning.


  ""->Update the status back in entity
  MODIFY ENTITIES OF zbi_train_head IN LOCAL MODE
    ENTITY train_head
      UPDATE FIELDS ( LearnStatusId )
      WITH VALUE #( FOR ls_keys IN keys ( %tky = ls_keys-%tky
                                LearnStatusId = 'I' ) ).

  READ ENTITIES OF zbi_train_head IN LOCAL MODE
    ENTITY train_head
    ALL FIELDS WITH CORRESPONDING #( keys )
    RESULT DATA(lt_head_r)
    FAILED DATA(lt_failed).

  result = VALUE #( FOR ls_head_r IN lt_head_r ( %tky   = ls_head_r-%tky
                                  %param = ls_head_r ) ).


 ENDMETHOD.

ENDCLASS.
```

2.6. Creation of Service Definition

Create Service Definition for Consumption View: ZCO_TRAIN_HEAD:

ZSDEF_TRAIN_HEAD

```
@EndUserText.label: 'Service Definition for Training'
define service ZSDEF_TRAIN_HEAD {
  expose ZCO_TRAIN_HEAD;
  expose ZCO_TRAIN_ITEM;
}
```

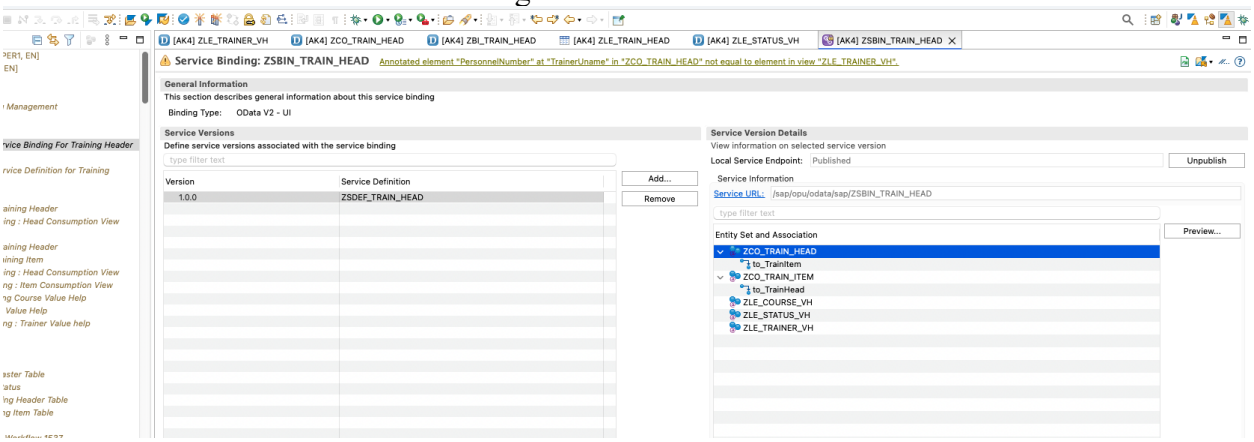
2.7. Creation of Service Binding

Create Service Binding for Service Definition: ZSDEF_TRAIN_HEAD:

ZSBIN_TRAIN_HEAD

Binding Type: ODATA V2 - UI

Activate and Publish the Service Binding:



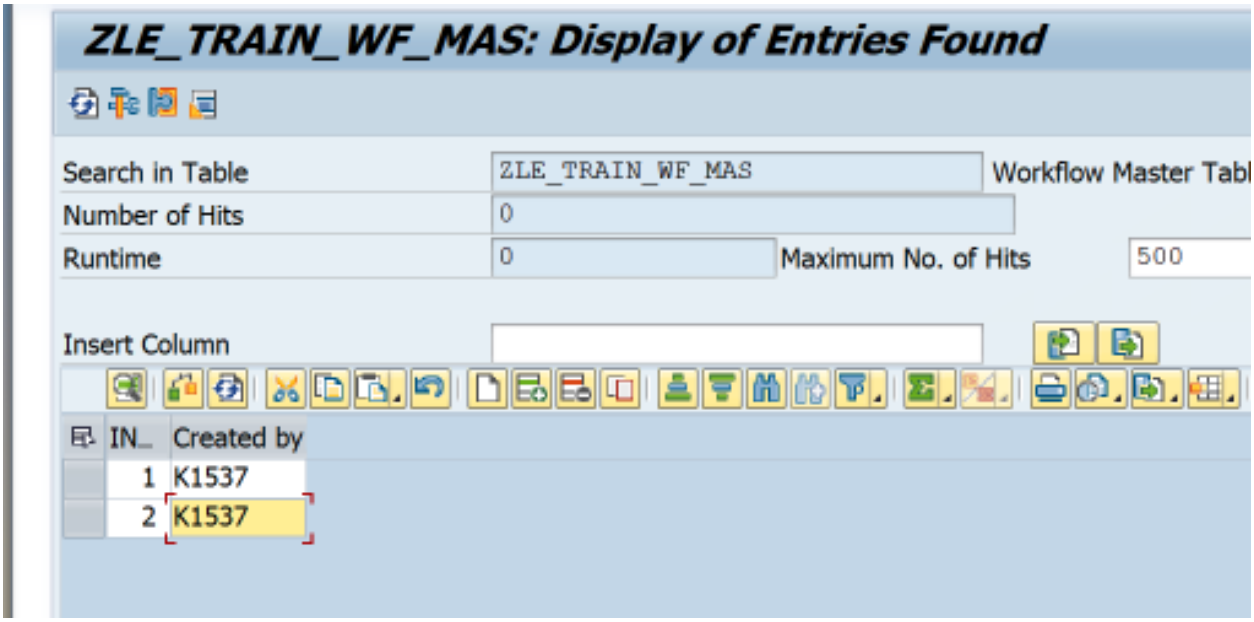
3. Flexible workflow

3.1. Creation of Workflow Dictionary Tables

Create Dictionary Master Table to store the number of levels for workflow:
ZLE_TRAIN_WF_MAS

```
@EndUserText.label : 'Workflow Master Table'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #ALLOWED
define table zle_train_wf_mas {
  key mandt      : mandt not null;
  key appr_level : int4 not null;
  appr_user      : usrid;

}
```

Let's Maintain Two Levels and corresponding users:
Note You can maintain any level you want and also instead of using USRID you can use Position as well



Create a Dictionary Transaction Table to store the Approval log against the Training ID and status for each approval: **ZLE_TRAIN_WF_TRA**

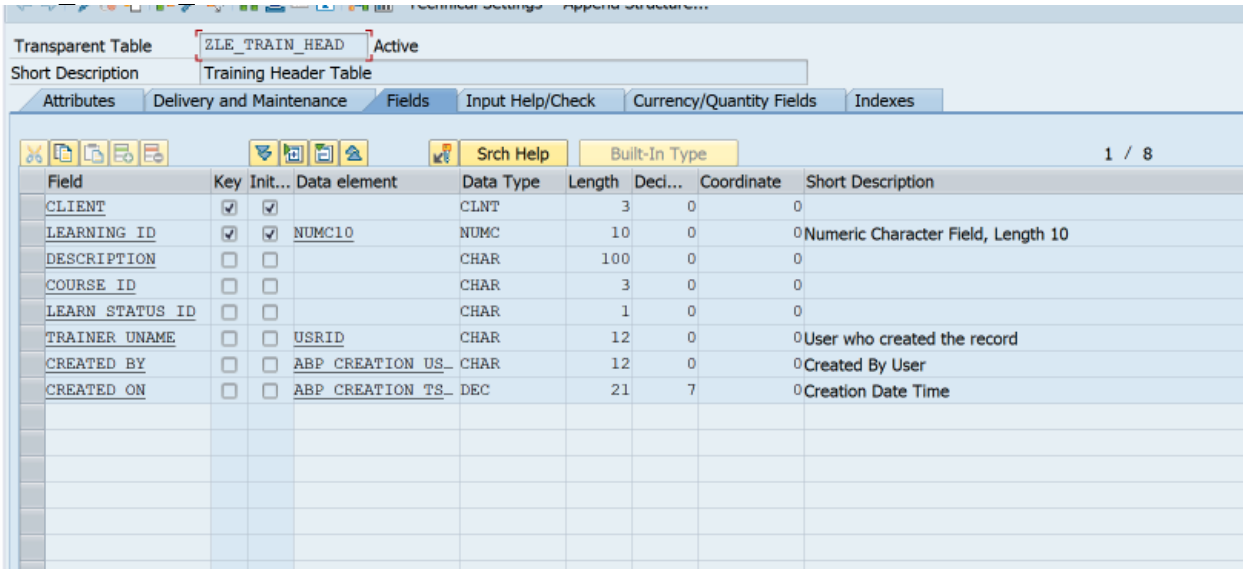
```
@EndUserText.label : 'Training Workflow for approval'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #ALLOWED
define table zle_train_wf_tra {
  key mandt      : mandt not null;
  key learning_id : numc10 not null;
  key appr_level  : int4 not null;
  appr_user      : usrid;
  status         : zle_de_train_status;
  execute_time   : uzeit;
  execute_date   : datum;
```

```
  work_item_id    : sww_wiid;
  created_date    : datum;
  created_time    : uzeit;
}
```

3.2. Creation of Structure used in workflow

Here you can use the own structure but here since the scenario is simple, we are using the already created Dictionary Table which is the header table to store the records:

ZLE_TRAIN_HEAD

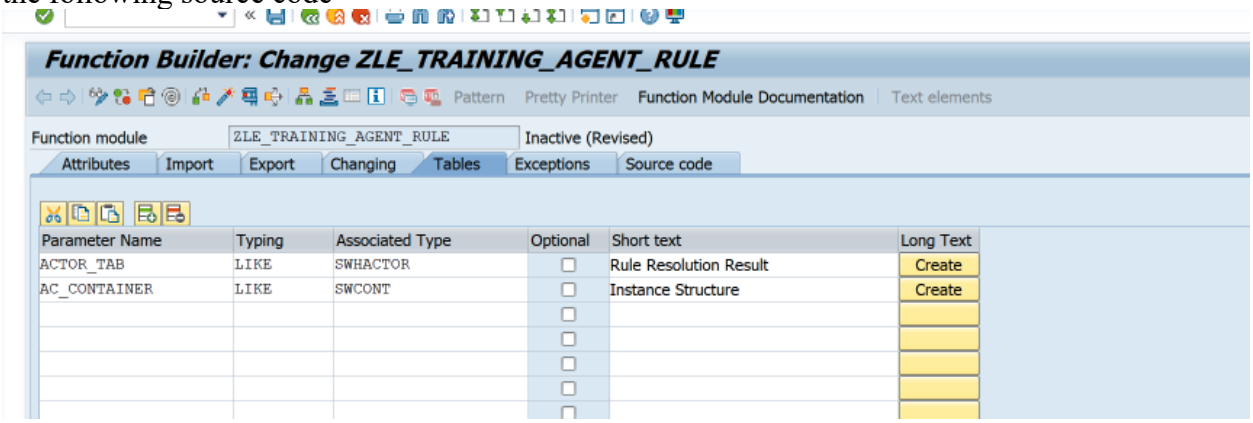


3.3. Creation of Agent Rule

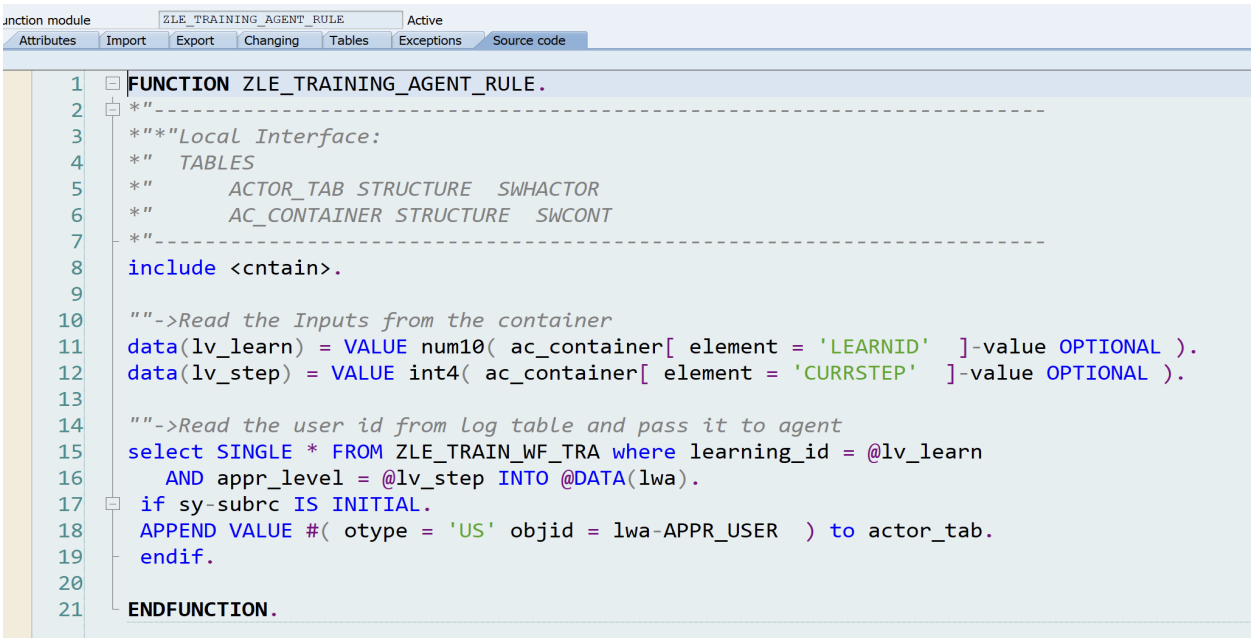
3.3.1. Functional Module for Agent Rule

Workflows will have agents to whom the task is sent and it can be achieved in various ways, here we will use the Agent rule.

First create a Function Module for the rule with the following as parameters and activate it with the following source code



The code reads the Rule container and fetch the agent user value from the Workflow transaction table:

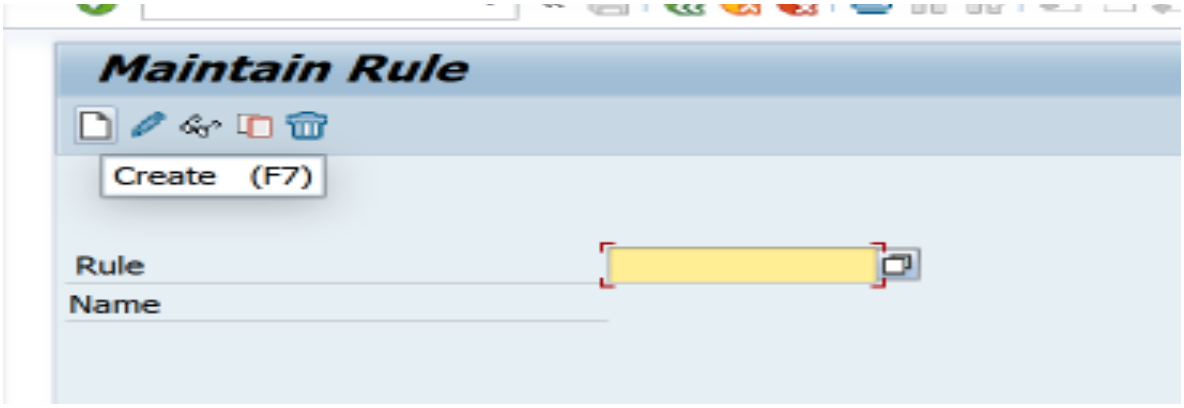


Full Code for the function Module

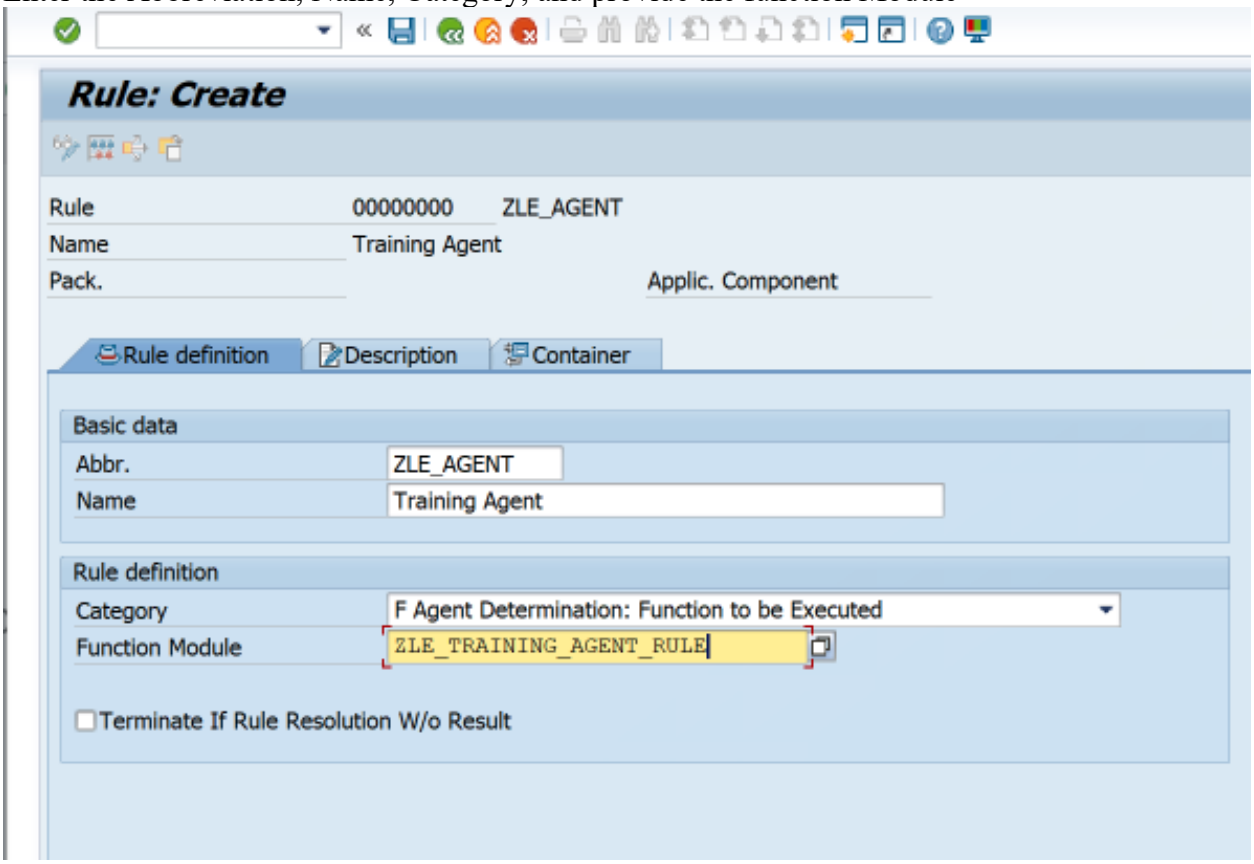
```
function zle_training_agent_rule
  tables
    actor_tab like swhactor
    ac_container like swcont.


""->It is Important to include this container
include <cntain>.

""->Read the Inputs from the container
data(lv_learn) = VALUE num10( ac_container[ element = 'LEARNID'  ]-value OPTIONAL ).
data(lv_step) = VALUE int4( ac_container[ element = 'CURRSTEP'  ]-value OPTIONAL ).

""->Read the user id from log table and pass it to agent
select SINGLE * FROM ZLE_TRAIN_WF_TRA where learning_id = @lv_learn
   AND appr_level = @lv_step INTO @DATA(lwa).
 if sy-subrc IS INITIAL.
 APPEND VALUE #( otype = 'US' objid = lwa-APPR_USER  ) to actor_tab.
 endif.

ENDFUNCTION.
```

3.3.2. Agent Rule Creation in PFAC AC92000024

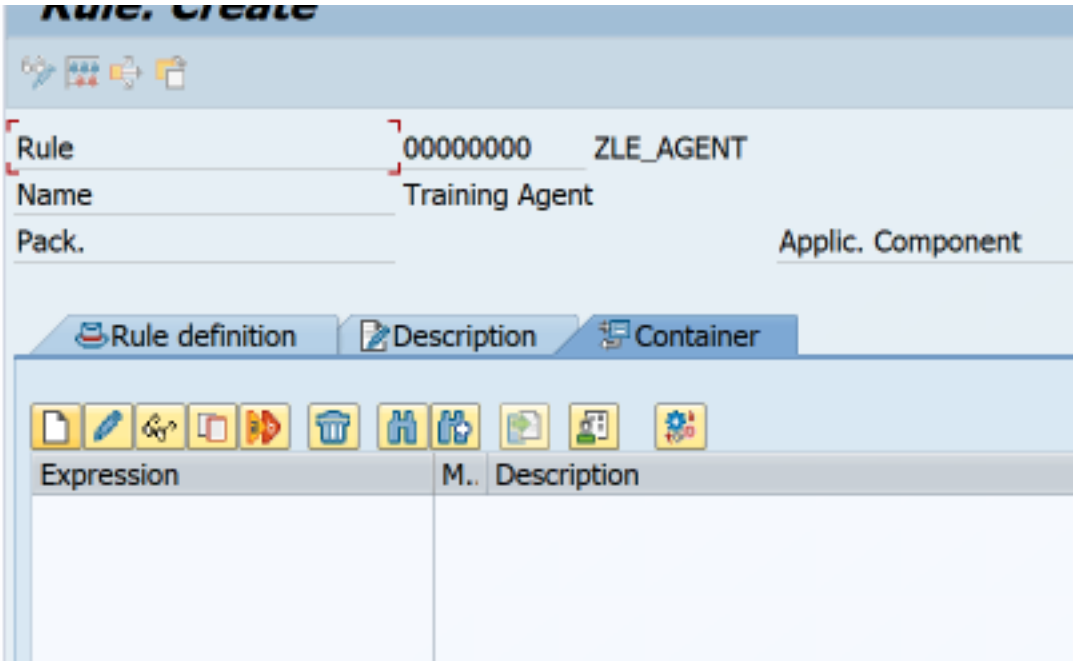
Go to TCODE PFAC: Click on create new



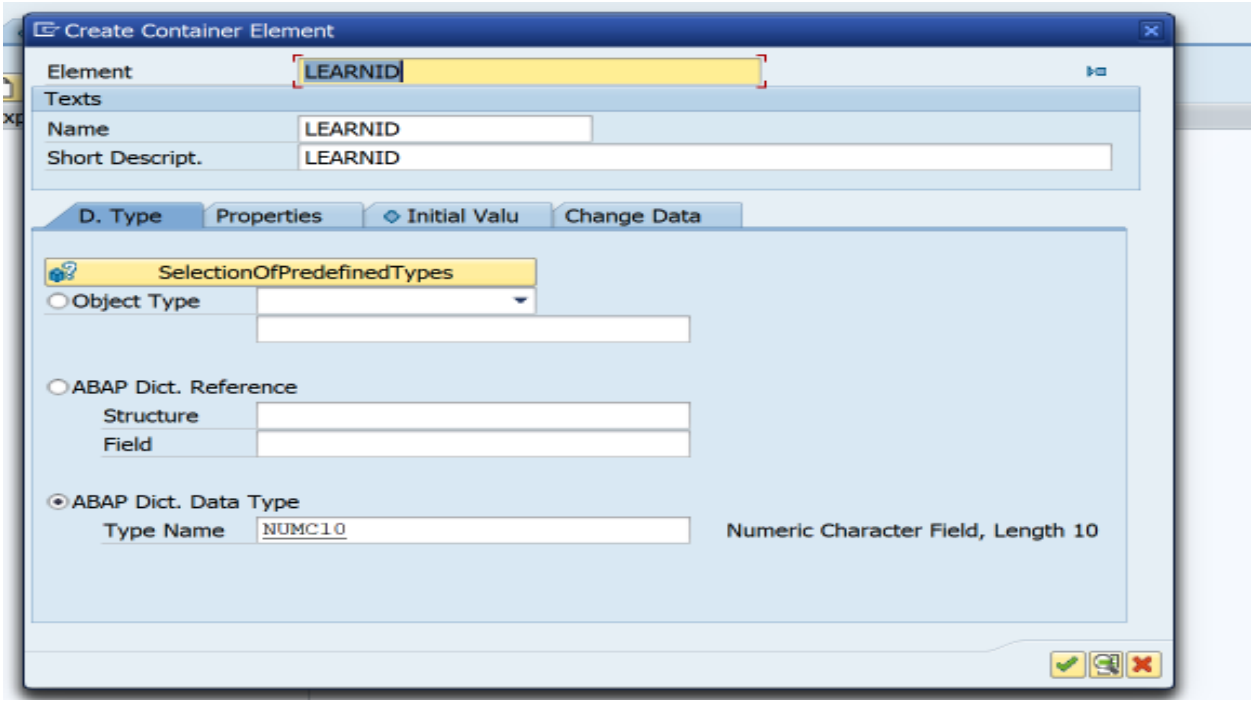
Enter the Abbreviation, Name, Category, and provide the function Module



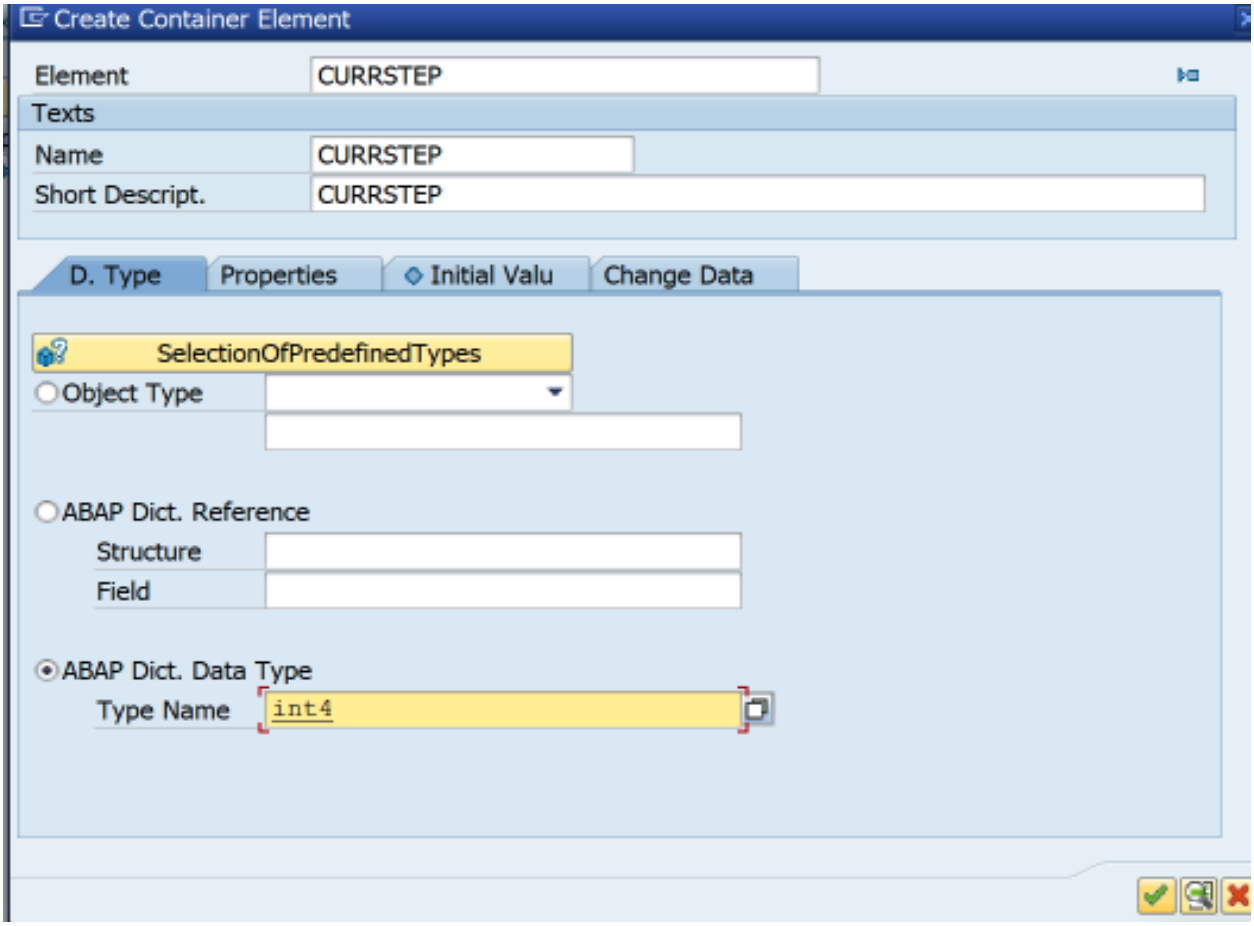
Go to the Container Tab and Create two Container Elements:



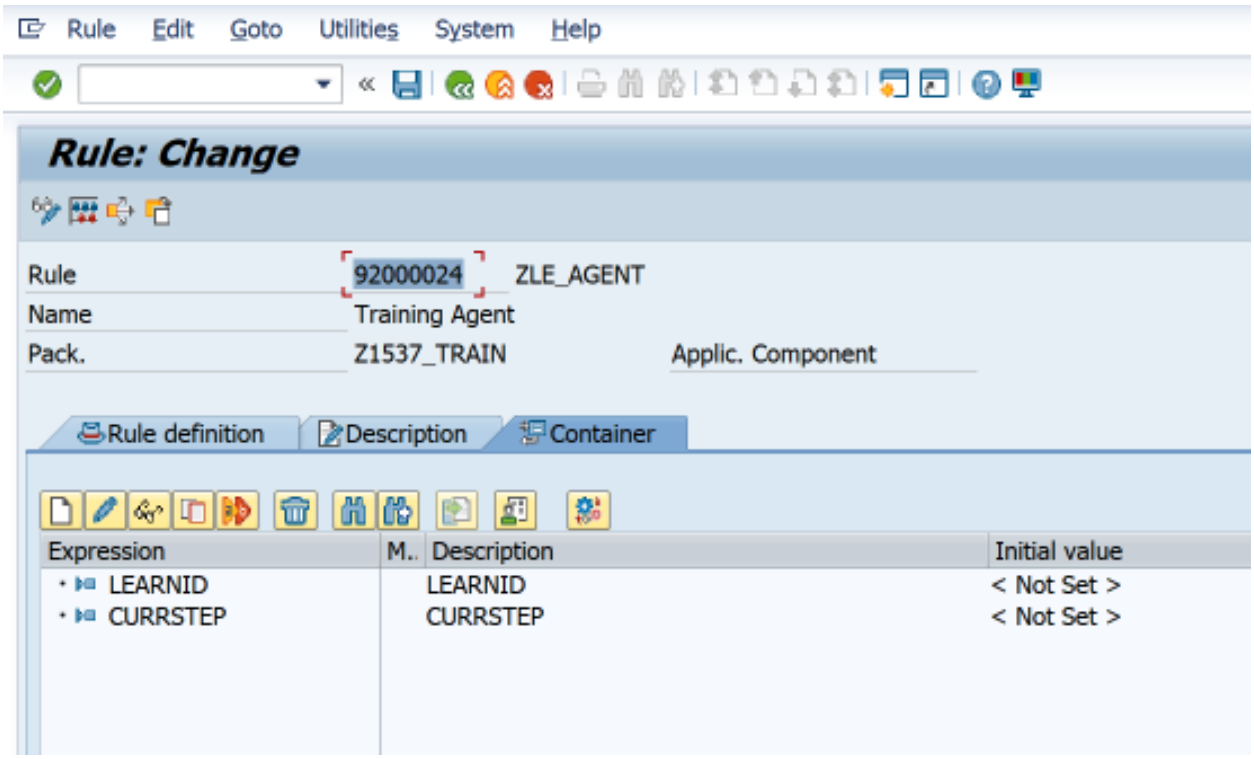
Element 1: LEARNID



Element 2: CURRSTEP



Click on Save to generate a new Number:



3.4. Creation of Classes for Workflow

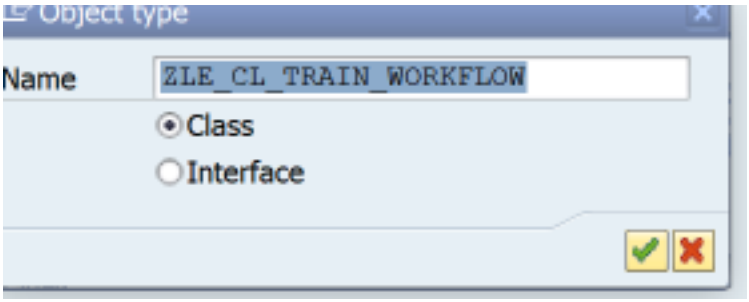
To create a workflow, we need 3 classes

- A Class and its event which will trigger the workflow
- A Class for Definition Application Base for Flexible Workflow Control
- A Class for Runtime Application Base for Flexible Workflow Control

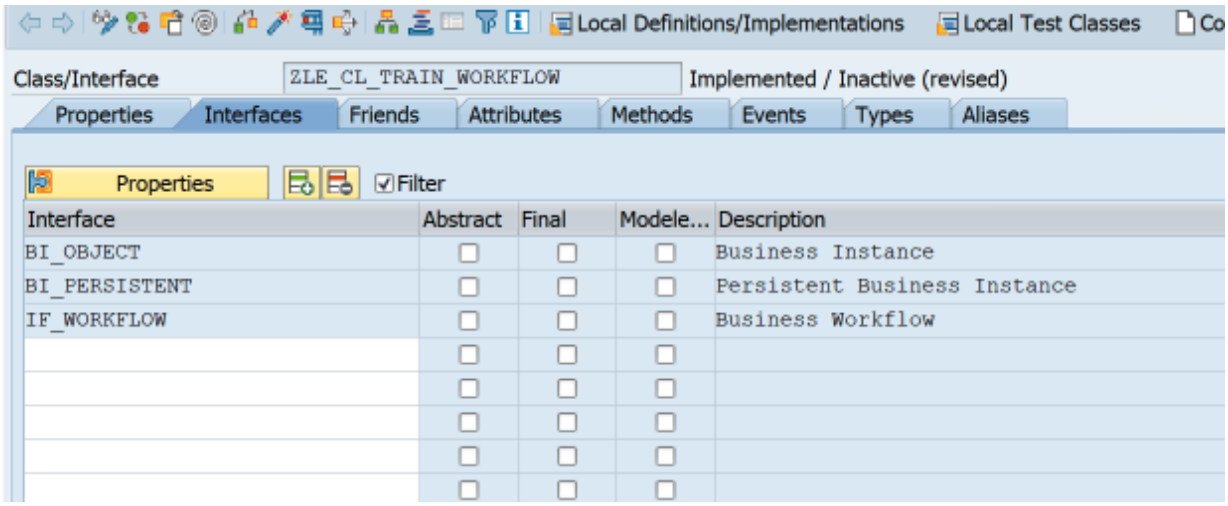
3.4.1. Creation of Workflow Class and its event ZLE_CL_TRAIN_WORKFLOW

You can use the GUI approach or directly create it in ABAP eclipse

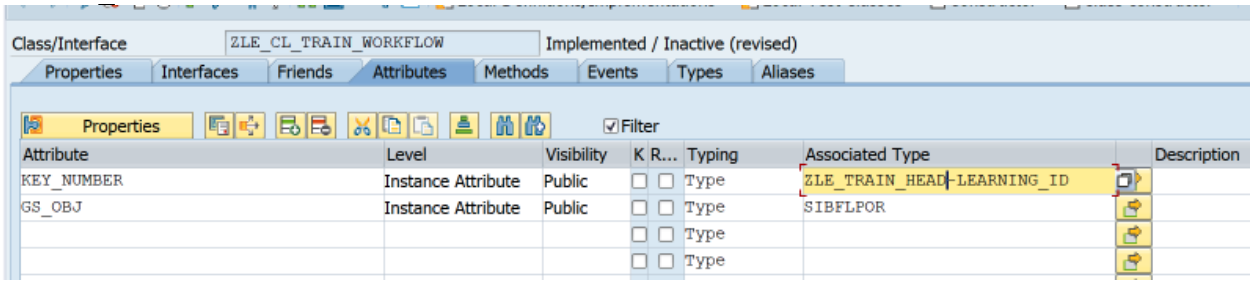
Go to TCODE: SE24: Create a new Class



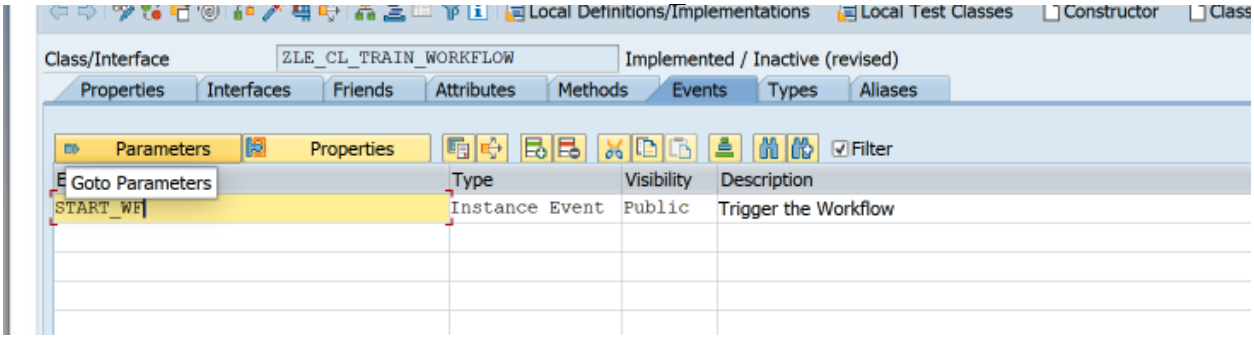
Go to Interface tab and give IF_WORKFLOW automatically two new interfaces are added



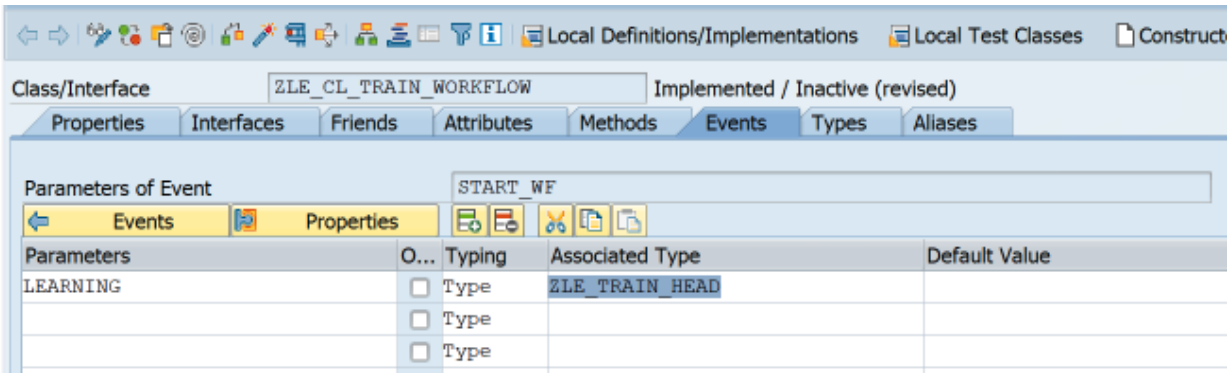
Go to attributes and add the following Key Number for workflow instance: **KEY_NUMBER** and **GS_OBJ**



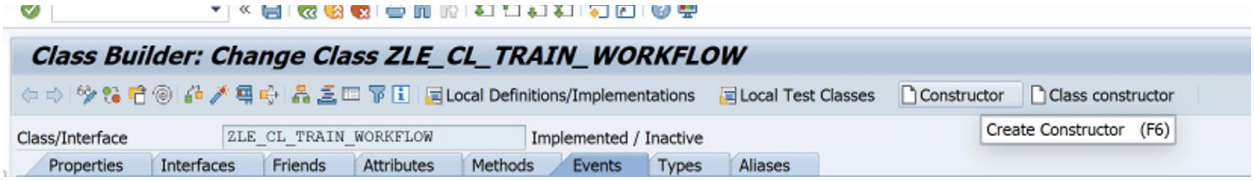
Go to the Event tabs and add a New Event: **START_WF**



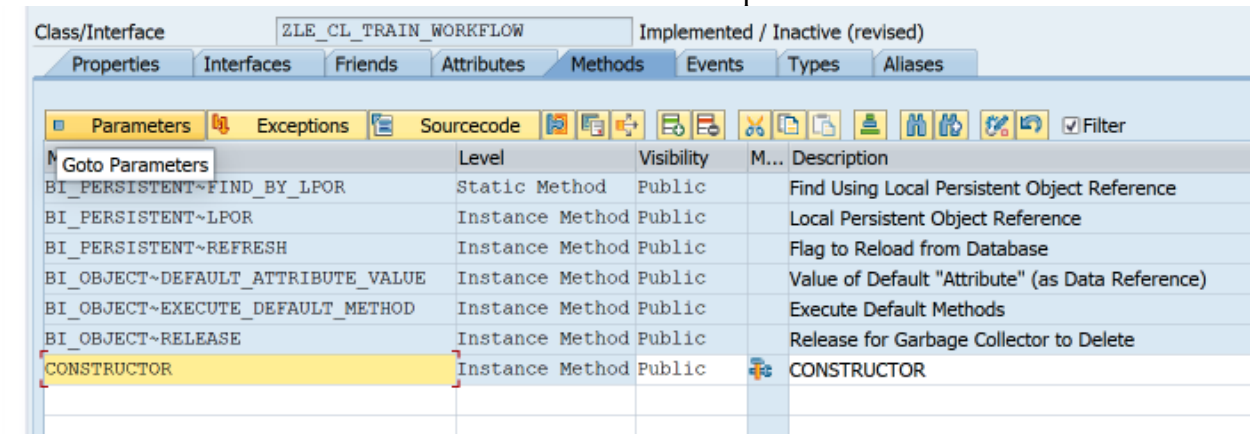
Select the Event and click on parameters and add the following as the Parameter: **LEARNING**



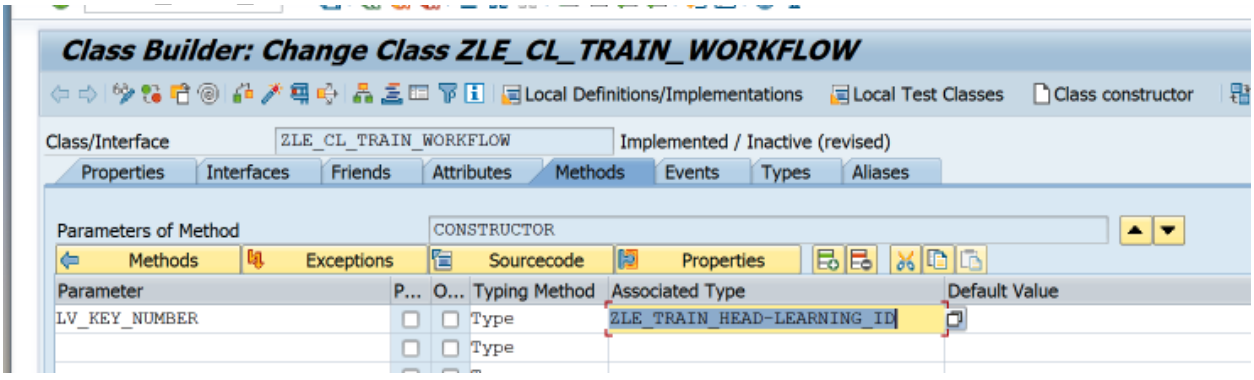
Creation of Constructor : Click on create constructor button which will add the constrcutor method



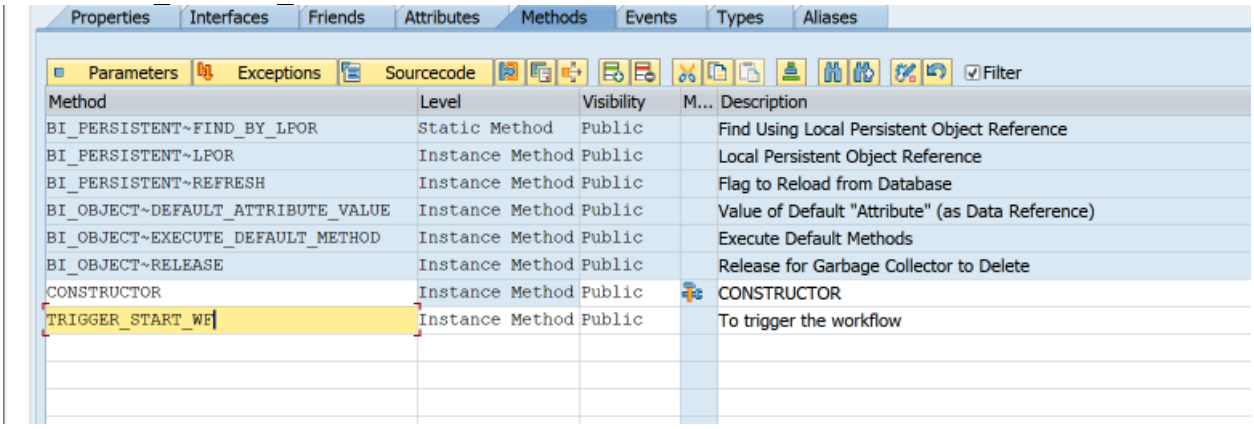
Select the Constructor and click on Parameters tab to add parameter for constructor



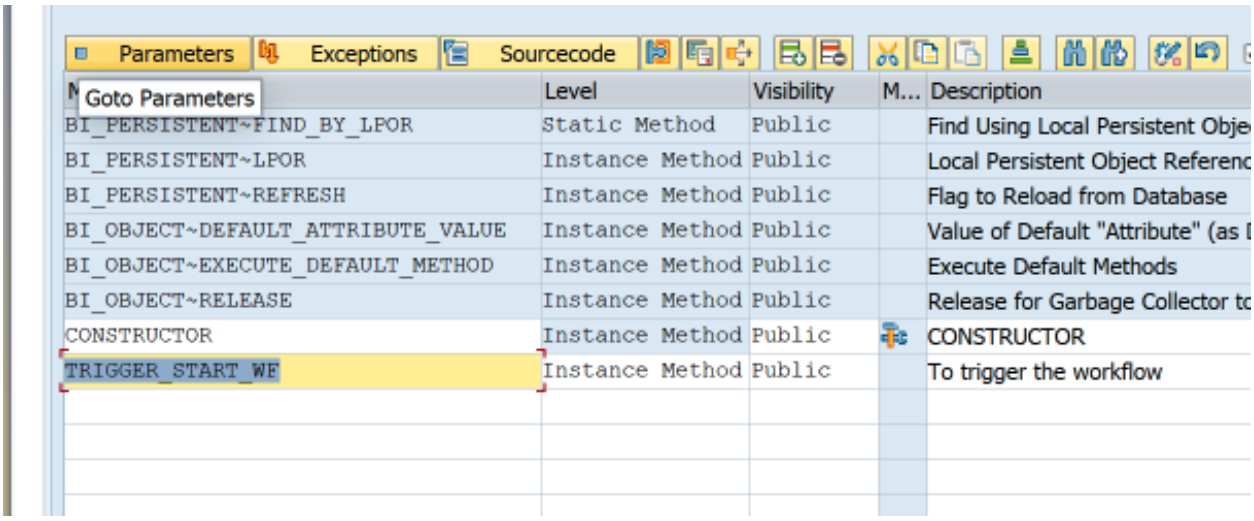
Add the LV_KEY_NUMBER as Parameter to constructor



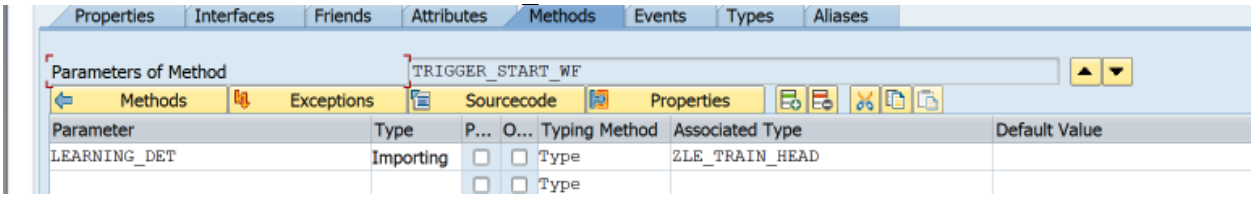
Go back to Method and add a method to store the Approver log table and trigger the workflow: **TRIGGER_START_WF**



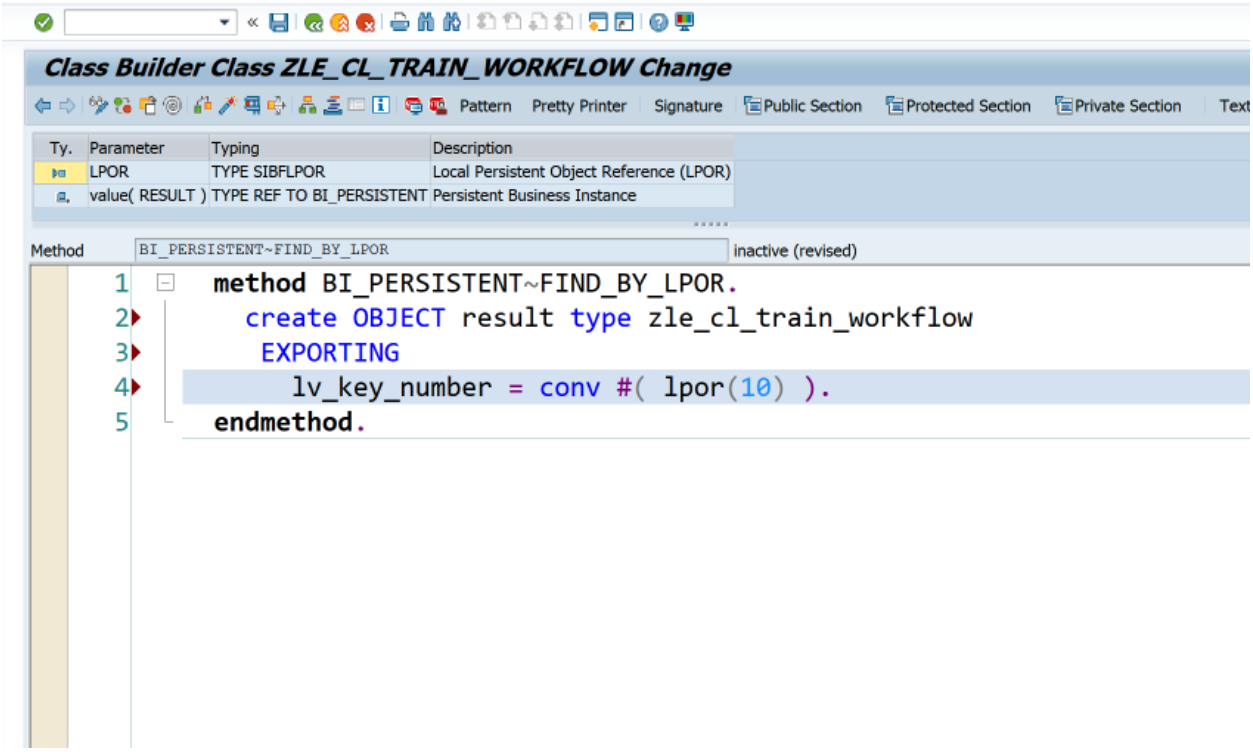
Add Parameters for the method:

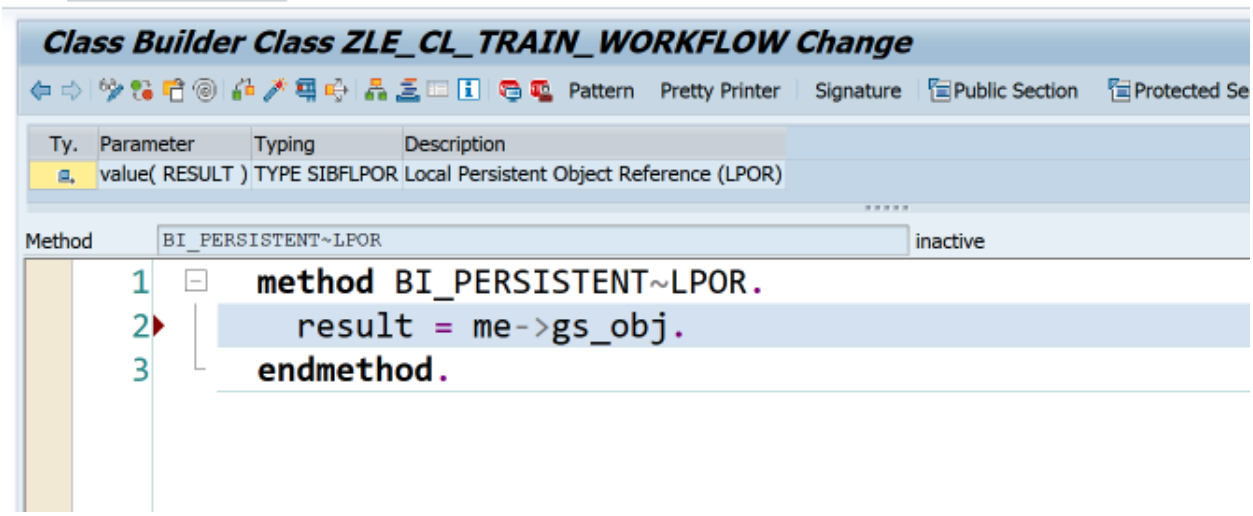


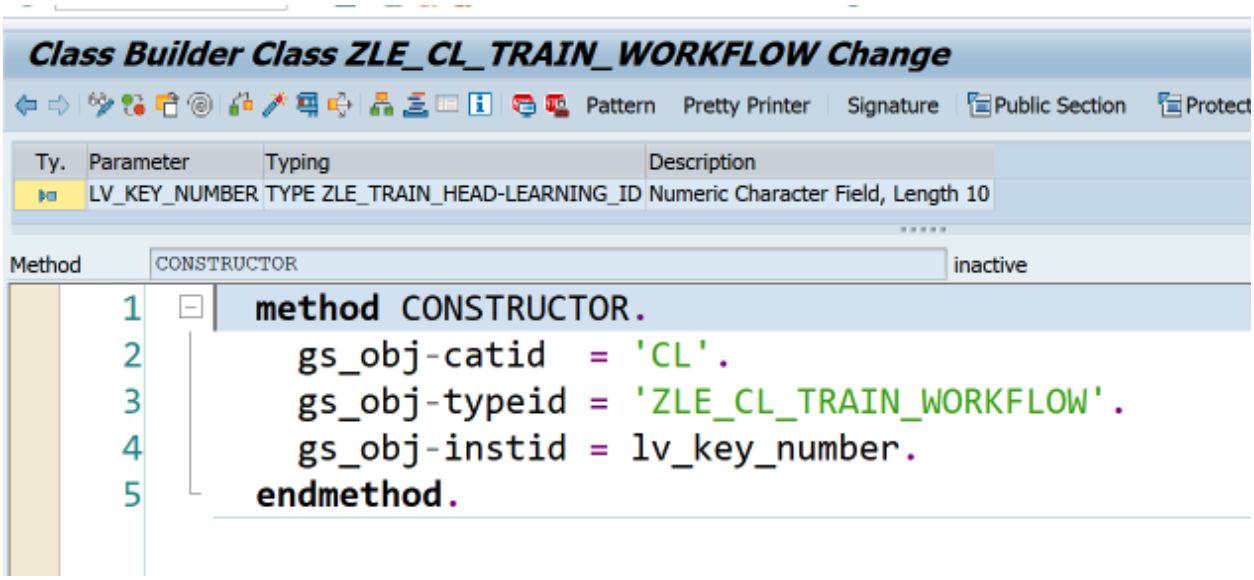
Parameter for the Method: **LEARNING_DET**

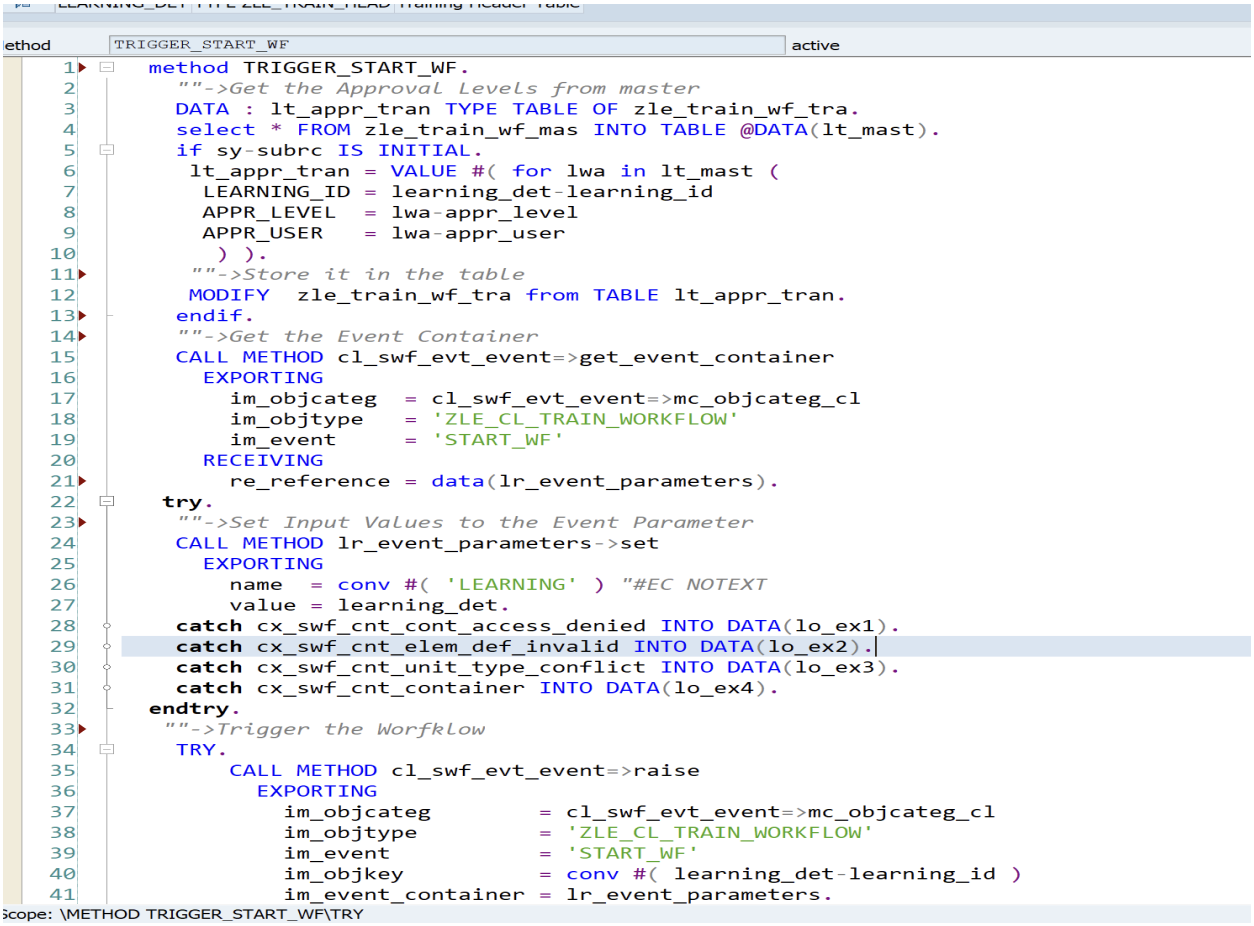


Add the following Code in the following methods:

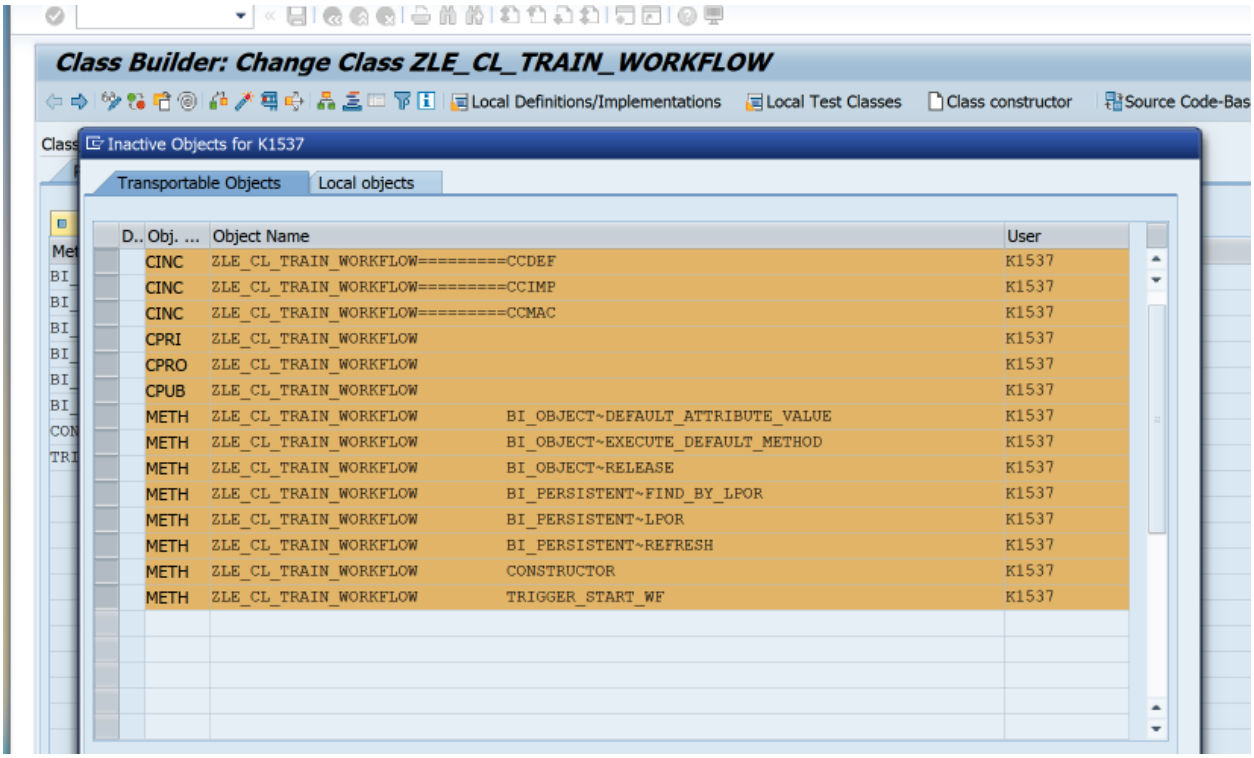

Class Builder Class ZLE_CL_TRAIN_WORKFLOW Change
Pattern
Pretty Printer
Signature
Public Section
Protected Section
Private Section
Ty.
Parameter
Typing
Description
LPOR
TYPE SIBFLPOR
Local Persistent Object Reference (LPOR)
value(RESULT)
TYPE REF TO BI_PERSISTENT
Persistent Business Instance
Method
BI_PERSISTENT~FIND_BY_LPOR
inactive (revised)
method BI_PERSISTENT~FIND_BY_LPOR.
create OBJECT result type zle_cl_train_workflow
EXPORTING
lv_key_number = conv #(lpor(10)).
endmethod.


Class Builder Class ZLE_CL_TRAIN_WORKFLOW Change
Pattern
Pretty Printer
Signature
Public Section
Protected Se
Ty.
Parameter
Typing
Description
value(RESULT)
TYPE SIBFLPOR
Local Persistent Object Reference (LPOR)
Method
BI_PERSISTENT~LPOR
inactive
method BI_PERSISTENT~LPOR.
result = me->gs_obj.
endmethod.


Class Builder Class ZLE_CL_TRAIN_WORKFLOW Change
Pattern
Pretty Printer
Signature
Public Section
Protect
Ty.
Parameter
Typing
Description
LV_KEY_NUMBER
TYPE ZLE_TRAIN_HEAD-LEARNING_ID
Numeric Character Field, Length 10
Method
CONSTRUCTOR
inactive
method CONSTRUCTOR.
gs_obj-catid = 'CL'.
gs_obj-typeid = 'ZLE_CL_TRAIN_WORKFLOW'.
gs_obj-instid = lv_key_number.
endmethod.


ethod
TRIGGER_START_WF
active
method TRIGGER_START_WF.
""->Get the Approval Levels from master
DATA : lt_appr_tran TYPE TABLE OF zle_train_wf_tra.
select * FROM zle_train_wf_mas INTO TABLE @DATA(lt_mast).
if sy-subrc IS INITIAL.
lt_appr_tran = VALUE #(for lwa in lt_mast (
LEARNING_ID = learning_det-learning_id
APPR_LEVEL = lwa-appr_level
APPR_USER = lwa-appr_user
)).
""->Store it in the table
MODIFY zle_train_wf_tra from TABLE lt_appr_tran.
endif.
""->Get the Event Container
CALL METHOD cl_swf_evt_event=>get_event_container
EXPORTING
im_objcateg = cl_swf_evt_event=>mc_objcateg_cl
im_objtype = 'ZLE_CL_TRAIN_WORKFLOW'
im_event = 'START_WF'
RECEIVING
re_reference = data(lr_event_parameters).
try.
""->Set Input Values to the Event Parameter
CALL METHOD lr_event_parameters->set
EXPORTING
name = conv #('LEARNING') "#EC NOTEXT
value = learning_det.
catch cx_swf_cnt_cont_access_denied INTO DATA(lo_ex1).
catch cx_swf_cnt_elem_def_invalid INTO DATA(lo_ex2).
catch cx_swf_cnt_unit_type_conflict INTO DATA(lo_ex3).
catch cx_swf_cnt_container INTO DATA(lo_ex4).
endtry.
""->Trigger the Worfklow
TRY.
CALL METHOD cl_swf_evt_event=>raise
EXPORTING
im_objcateg = cl_swf_evt_event=>mc_objcateg_cl
im_objtype = 'ZLE_CL_TRAIN_WORKFLOW'
im_event = 'START_WF'
im_objkey = conv #(learning_det-learning_id)
im_event_container = lr_event_parameters.
cope: \METHOD TRIGGER_START_WF\TRY

After this Activate all the methods in the class



3.4.1.1. *Full Code ZLE_CL_TRAIN_WORKFLOW*

```
class ZLE_CL_TRAIN_WORKFLOW definition
  public
  final
  create public .

public section.

  interfaces BI_OBJECT .
  interfaces BI_PERSISTENT .
  interfaces IF_WORKFLOW .

  types:
    BEGIN OF Ty_RECEPIENTS,
          user_id TYPE sysid,
        END OF ty_recepients .
  types:
    tt_recepients TYPE TABLE of ty_recepients WITH DEFAULT KEY .

  data KEY_NUMBER type ZLE_TRAIN_HEAD-LEARNING_ID .
  data GS_OBJ type SIBFLPOR .

  events START_WF
    exporting
      value(LEARNING) type ZLE_TRAIN_HEAD .

  methods CONSTRUCTOR
    importing
      !LV_KEY_NUMBER type ZLE_TRAIN_HEAD-LEARNING_ID .
  methods TRIGGER_START_WF
    importing
      !LEARNING_DET type ZLE_TRAIN_HEAD .
  class-methods SEND_EMAIL
    importing
      !IM_IT_USERID type TT_RECEPIENTS
```

```
    !IM_LEARNING type ZLE_TRAIN_HEAD
    !IM_EMAIL_TYPE type TEXT10
    !IM_REJECTION_NOTE type STRING optional .
protected section.
private section.
ENDCLASS.


CLASS ZLE_CL_TRAIN_WORKFLOW IMPLEMENTATION.


 method BI_OBJECT~DEFAULT_ATTRIBUTE_VALUE.

 endmethod.


 method BI_OBJECT~EXECUTE_DEFAULT_METHOD.

 endmethod.


 method BI_OBJECT~RELEASE.

 endmethod.


 method BI_PERSISTENT~FIND_BY_LPOR.
  create OBJECT result type zle_cl_train_workflow
   EXPORTING
    lv_key_number = conv #( lpor(10) ).
 endmethod.


 method BI_PERSISTENT~LPOR.
  result = me->gs_obj.
 endmethod.


 method BI_PERSISTENT~REFRESH.

 endmethod.


 method CONSTRUCTOR.
  gs_obj-catid  = 'CL'.
  gs_obj-typeid = 'ZLE_CL_TRAIN_WORKFLOW'.
  gs_obj-instid = lv_key_number.
 endmethod.


 method TRIGGER_START_WF.

  ""->Get the Approval Levels from master
  DATA : lt_appr_tran TYPE TABLE OF zle_train_wf_tra.
  select * FROM zle_train_wf_mas INTO TABLE @DATA(lt_mast).
  if sy-subrc IS INITIAL.
   lt_appr_tran = VALUE #( for lwa in lt_mast (
    LEARNING_ID = learning_det-learning_id
    APPR_LEVEL  = lwa-appr_level
    APPR_USER   = lwa-appr_user
    ) ).
   MODIFY  zle_train_wf_tra from TABLE lt_appr_tran.
  endif.


  CALL METHOD cl_swf_evt_event=>get_event_container
   EXPORTING
     im_objcateg  = cl_swf_evt_event=>mc_objcateg_cl
     im_objtype   = 'ZLE_CL_TRAIN_WORKFLOW'
```

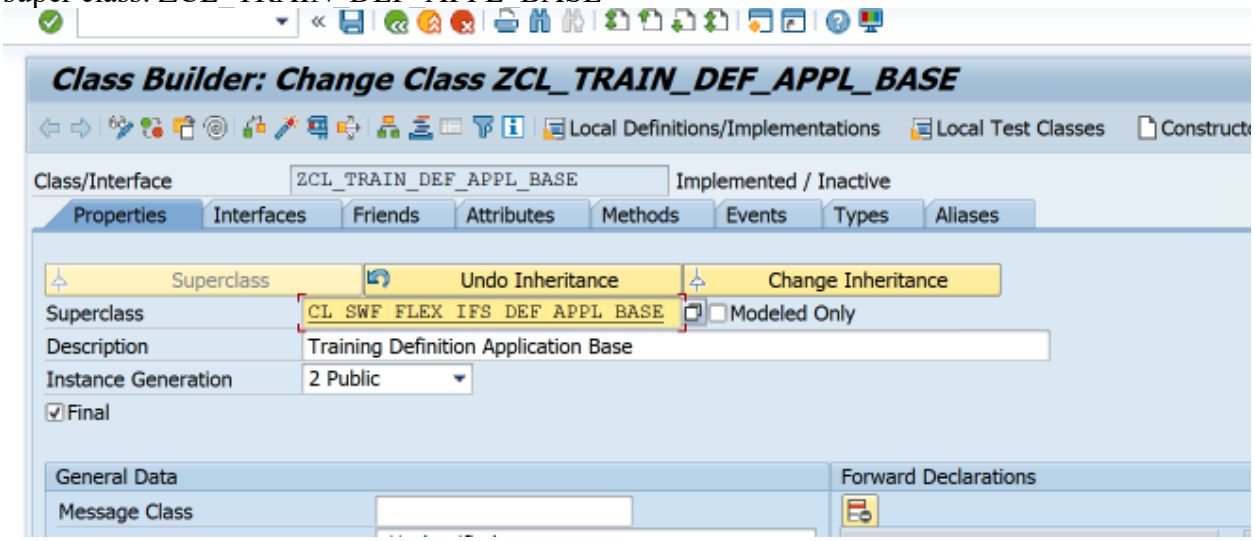
```
      im_event     = 'START_WF'
    RECEIVING
      re_reference = data(lr_event_parameters).

  try.
   CALL METHOD lr_event_parameters->set
    EXPORTING
      name  = conv #( 'LEARNING' ) "#EC NOTEXT
      value = learning_det.
   catch cx_swf_cnt_cont_access_denied INTO DATA(lo_ex1).
   catch cx_swf_cnt_elem_def_invalid INTO DATA(lo_ex2).
   catch cx_swf_cnt_unit_type_conflict INTO DATA(lo_ex3).
   catch cx_swf_cnt_container INTO DATA(lo_ex4).
  endtry.
**
   TRY.
      CALL METHOD cl_swf_evt_event=>raise
       EXPORTING
         im_objcateg       = cl_swf_evt_event=>mc_objcateg_cl
         im_objtype        = 'ZLE_CL_TRAIN_WORKFLOW'
         im_event          = 'START_WF'
         im_objkey         = conv #( learning_det-learning_id )
         im_event_container = lr_event_parameters.

    CATCH cx_swf_evt_invalid_objtype.
    CATCH cx_swf_evt_invalid_event.
   ENDTRY.

  endmethod.

  method SEND_EMAIL.
  ""Common Method to send Custom Email during the execution of workflow
   ""->Constants Declaration
   CONSTANTS : c_doc_type TYPE so_obj_tp VALUE 'HTM',
               c_sub_0001 TYPE pa0105-subty VALUE '0001',
               c_sub_0010 TYPE pa0105-subty VALUE '0010'.

   ""->Class Definition
   CLASS cl_bcs DEFINITION LOAD.

   ""->Object Declaration
   DATA: lo_send_request TYPE REF TO cl_bcs VALUE IS INITIAL,
         lo_document     TYPE REF TO cl_document_bcs VALUE IS INITIAL,
         lo_recipient    TYPE REF TO if_recipient_bcs VALUE IS INITIAL,
         lo_sender       TYPE REF TO if_sender_bcs VALUE IS INITIAL.

   ""->Internal Table Declaration
   DATA: lt_message_body    TYPE bcsy_text VALUE IS INITIAL.

   ""->Work Area Declaration
   DATA: ls_message_ody TYPE bcsy_text,
         ls_body_msg    TYPE solisti1.

   ""->Variable Declaration
   DATA: lv_message_text TYPE string,
         lv_send         TYPE adr6-smtp_addr,
         lv_subject      TYPE so_obj_des.

   CHECK im_it_userid IS NOT INITIAL.

   TRY.
      lo_send_request = cl_bcs=>create_persistent( ).
    CATCH cx_send_req_bcs INTO DATA(lo_excep).
   ENDTRY.

   ""->Get Full Name
   IF im_it_userid IS NOT INITIAL.
    SELECT b~usrid_long,
           c~ename
      FROM pa0105 AS a
```

```
    INNER JOIN pa0105 AS b
    ON    b~pernr = a~pernr
    AND   b~subty = @c_sub_0010
    AND   b~begda <= @sy-datum
    AND   b~endda >= @sy-datum
    INNER JOIN pa0001 AS c
    ON c~pernr = a~pernr
    AND   c~begda <= @sy-datum
    AND   c~endda >= @sy-datum
    FOR ALL ENTRIES IN @im_it_userid
    WHERE a~usrid = @im_it_userid-user_id
    AND   a~subty = @c_sub_0001
    AND   a~begda <= @sy-datum
    AND   a~endda >= @sy-datum
    INTO TABLE @DATA(lt_recepient).
ENDIF.

delete lt_recepient WHERE usrid_long IS INITIAL.
sort lt_recepient ASCENDING BY ename.
DELETE ADJACENT DUPLICATES FROM lt_recepient COMPARING ename.

""->Mail Subject
CASE im_email_type.
 when 'C'. ""Creation of Task
 lv_subject = |Please Approve Training Request({ im_learning-learning_id })|.
 when 'A'. "" Approval of Task
 lv_subject = |Training Request ({ im_learning-learning_id }) approved|.
 when 'R'. ""Rejection of Request
  lv_subject = |Training Request ({ im_learning-learning_id }) Rejected|.
 ENDCASE.
""->Message Body
APPEND '<html>' TO lt_message_body.
APPEND '<body><p>' TO lt_message_body.
""->Address Recepients with Names
APPEND |Dear  | TO lt_message_body.
LOOP AT lt_recepient INTO DATA(ls_recepient).
  IF sy-tabix EQ '1'.
    APPEND  ls_recepient-ename TO lt_message_body.
  ELSE.
    APPEND  |,{ ls_recepient-ename }| TO lt_message_body.
  ENDIF.
  TRY.
      lo_recipient = cl_cam_address_bcs=>create_internet_address( ls_recepient-usrid_long ).
      lo_send_request->add_recipient( EXPORTING i_recipient = lo_recipient
                                      i_express   = abap_true ).
    CATCH cx_send_req_bcs INTO DATA(lo_send_exception).
    CATCH cx_address_bcs INTO DATA(lo_address_exception).
  ENDTRY.
ENDLOOP.
APPEND |,</br>| TO lt_message_body.
CASE im_email_type.
  WHEN 'C'. ""->Creation
  APPEND |Training Request { im_learning-learning_id } is sent for your Approval| TO lt_message_body.
  WHEN 'A'. ""->Approved
  APPEND |Training Request { im_learning-learning_id }  is Approved.| TO lt_message_body.
  WHEN 'R'. ""->Rejected
  APPEND |Training Request { im_learning-learning_id }  is Rejected.</br>| TO lt_message_body.
  Append |Rejection Note : { im_rejection_note } </br>| to lt_message_body.
ENDCASE.
APPEND '</p></body></html>' TO lt_message_body.

TRY.
    lo_document = cl_document_bcs=>create_document( i_type    = c_doc_type
                                      i_text    = lt_message_body
                                      i_subject = lv_subject ).
  CATCH cx_document_bcs INTO DATA(lo_doc_exception).
ENDTRY.

TRY.
    ""->Prepare Sender
```

```
      lo_sender = cl_sapuser_bcs=>create( 'K1537' ). "" Can use your user id
*       lo_recipient = cl_cam_address_bcs=>create_internet_address( lv_send ).
    CATCH cx_address_bcs INTO DATA(lo_address_bcs).
  ENDTRY.
  TRY.
    lo_send_request->set_sender( EXPORTING i_sender = lo_sender ).
    lo_send_request->set_document( lo_document ).
    lo_send_request->send( ).
   CATCH cx_send_req_bcs INTO lo_send_exception.
   CATCH cx_address_bcs INTO lo_address_exception.
  ENDTRY.

 endmethod.
ENDCLASS.
```

3.4.2. Creation of Class Definition Application base ZCL_TRAIN_DEF_APPL_BASE

Create a new Z Class by giving the standard class CL_SWF_FLEX_IFS_DEF_APPL_BASE as super class: ZCL_TRAIN_DEF_APPL_BASE



This class will not have any code, just activate it.

3.4.2.1. Full Code ZCL_TRAIN_DEF_APPL_BASE

```
class ZCL_TRAIN_DEF_APPL_BASE definition
  public
  inheriting from CL_SWF_FLEX_IFS_DEF_APPL_BASE
  final
  create public .

public section.
protected section.
private section.
ENDCLASS.


CLASS ZCL_TRAIN_DEF_APPL_BASE IMPLEMENTATION.
ENDCLASS.
```

3.4.3. Creation of Class Run time Application base ZCL_TRAIN_RUN_APPL_BASE

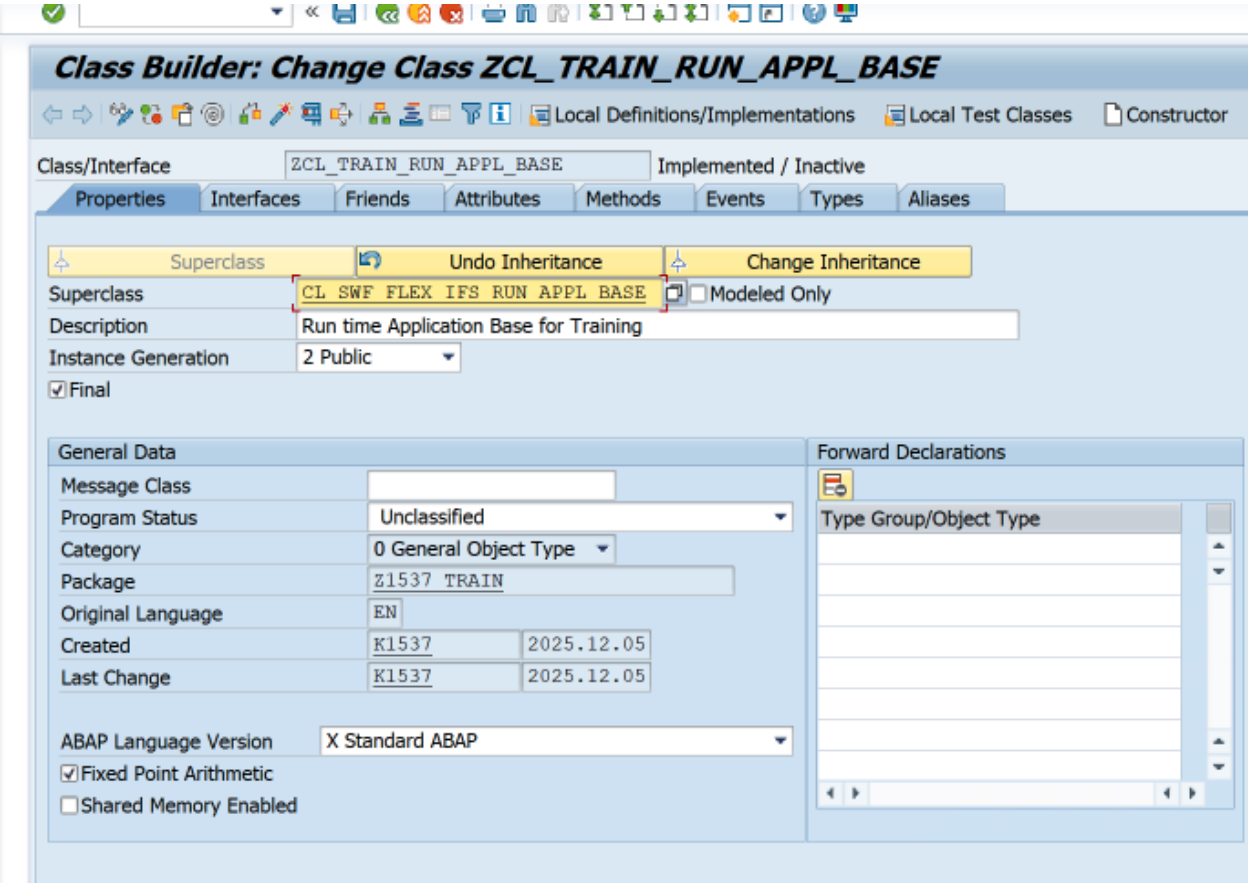
You have to redefine the methods into our Z class, which will be triggered during the workflow runtime.

Here we will use the following methods
->IF_SWF_FLEX_IFS_RUN_APPL_STEP~ON_CREATION_CALLBACK
This method gets called at creation of each task, this is used to write the current approver level in the workflow and send email for the task receiver

-> IF_SWF_FLEX_IFS_RUN_APPL_STEP~AFTER_COMPLETION_CALLBACK
This method gets called at completion of each task and is used to store the decision of each step in the ZTABLE

IF_SWF_FLEX_IFS_RUN_APPL~RESULT_CALLBACK
This method gets called at completion of workflow with the decision

Create a new Z Class by giving the standard class CL_SWF_FLEX_IFS_RUN_APPL_BASE as super class: ZCL_TRAIN_RUN_APPL_BASE



Save and activate the class.

3.4.3.1. Full Code ZCL_TRAIN_RUN_APPL_BASE

```
class ZCL_TRAIN_RUN_APPL_BASE definition
 public
 inheriting from CL_SWF_FLEX_IFS_RUN_APPL_BASE
 final
 create public .

public section.

  methods IF_SWF_FLEX_IFS_RUN_APPL_STEP~AFTER_COMPLETION_CALLBACK
    redefinition .
  methods IF_SWF_FLEX_IFS_RUN_APPL_STEP~ON_CREATION_CALLBACK
    redefinition .
  methods IF_SWF_FLEX_IFS_RUN_APPL~RESULT_CALLBACK
    redefinition .
protected section.
private section.
ENDCLASS.



CLASS ZCL_TRAIN_RUN_APPL_BASE IMPLEMENTATION.


  method IF_SWF_FLEX_IFS_RUN_APPL_STEP~AFTER_COMPLETION_CALLBACK.
*This Method is called after the Completion of every task

    DATA: lwa_learn TYPE zle_train_head,
        lv_curr_step TYPE int4.

     DATA(lo_wf_cont) = io_context->get_workflow_container(  ).
    DATA(lt_curr_act) = io_current_activity->get_execution_results( ).
    DATA(ls_curr_act) = VALUE #( lt_curr_act[ 1 ] OPTIONAL ).


    DATA(lv_user) = ls_curr_act-processor-objid.


    TRY.
       CALL METHOD lo_wf_cont->get
        EXPORTING
          name  = CONV #( 'LEARNING' )
        IMPORTING
          value = lwa_learn .

        CALL METHOD lo_wf_cont->get
        EXPORTING
          name  = CONV #( 'CurrentLevel' )
        IMPORTING
          value = lv_curr_step.


      DATA(lv_status) = COND #( when ls_curr_act-result = 'Approve' then 'A' else 'R'  ).

    ""->Update Workflow Log table
        update ZLE_TRAIN_WF_TRA SET execute_date = sy-datum
                        execute_time = sy-uzeit
                        status       = lv_status
                        WHERE learning_id = lwa_learn-learning_id
                        AND   appr_level = lv_curr_step.

    if lv_status eq 'R'.
    ""->On Reject Delete the further Approvals from table
    delete from ZLE_TRAIN_WF_TRA  WHERE learning_id = lwa_learn-learning_id
                          AND   appr_level > lv_curr_step.
    endif.


  ""->Set the incremented data of Current Level to  Workflow
```

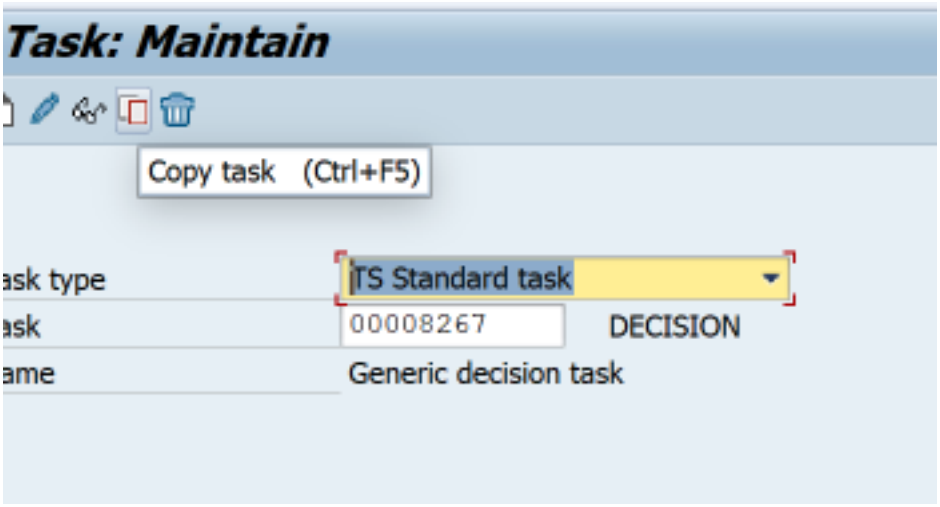
```
lv_curr_step = lv_curr_step + 1.
      CALL METHOD lo_wf_cont->set
       EXPORTING
        name  = CONV #( 'CurrentLevel' )
        value = lv_curr_step.


   CATCH cx_swf_cnt_elem_not_found.
   CATCH cx_swf_cnt_elem_type_conflict.
   CATCH cx_swf_cnt_unit_type_conflict.
   CATCH cx_swf_cnt_container.
  ENDTRY.

  endmethod.


 method IF_SWF_FLEX_IFS_RUN_APPL_STEP~ON_CREATION_CALLBACK.
 ""->This Method gets triggered When a task is created in workflow
 DATA: ls_learn TYPE zle_train_head,
     lv_curr_step type int4.
  DATA(lo_wf_cont) = io_context->get_workflow_container( ).
  DATA(lt_curr_act) = io_current_activity->get_execution_results( ).
  DATA(ls_curr_act) = VALUE #( lt_curr_act[ wi_stat = 'READY' wi_type = 'W' ] OPTIONAL ).
  ""->User to Process the task
  DATA(lv_user) = ls_curr_act-processor-objid.

  TRY.
     ""->Read the data from Container
    CALL METHOD lo_wf_cont->get
      EXPORTING
       name  = CONV #( 'LEARNING' )
      IMPORTING
       value = ls_learn.

     ""->Read the data Current Level from Workflow
     CALL METHOD lo_wf_cont->get
      EXPORTING
       name  = CONV #( 'CurrentLevel' )
      IMPORTING
       value = lv_curr_step.

     ""->Update Workflow Log table
      update ZLE_TRAIN_WF_TRA SET created_date = sy-datum
                    created_time = sy-uzeit
                    status       = 'I'
                    work_item_id  = ls_curr_act-wi_id
                    WHERE learning_id = ls_learn-learning_id
                    AND   appr_level = lv_curr_step.


   CATCH cx_swf_cnt_elem_not_found.
   CATCH cx_swf_cnt_elem_type_conflict.
   CATCH cx_swf_cnt_unit_type_conflict.
   CATCH cx_swf_cnt_container.
  ENDTRY.



  CHECK lv_user IS NOT INITIAL.
  ""->Send Email for approver
  ZLE_CL_TRAIN_WORKFLOW=>send_email( im_it_userid = VALUE #( ( user_id = lv_user ) )
                    im_learning = ls_learn
                    im_email_type = 'C' ).


 endmethod.


 method IF_SWF_FLEX_IFS_RUN_APPL~RESULT_CALLBACK.
```

```
""->Data Declaration
DATA : lv_user    TYPE uname,
       lv_rejnote TYPE string.

""->Internal Table Declaration
DATA : lt_note    TYPE swf_utl_porb_tab,
       lt_objcont TYPE TABLE OF solisti1,
       lt_recep   TYPE ZCL_ZLM_LEARN_WF=>tt_recepients.
""->Get the Initiator from Container
DATA(lo_wf_cont) = io_context->get_workflow_container( ).
DATA(ls_lm_learn) = VALUE zle_train_head( ).

TRY.
   CALL METHOD lo_wf_cont->get
    EXPORTING
     name  = CONV #( '_WF_INITIATOR' )
    IMPORTING
     value = lv_user.

   CALL METHOD lo_wf_cont->get
    EXPORTING
     name  = CONV #( 'LEARNING' )
    IMPORTING
     value = ls_lm_learn.


   CALL METHOD lo_wf_cont->get
    EXPORTING
     name  = CONV #( '_ATTACH_OBJECTS' )
    IMPORTING
     value = lt_note.

   DATA(lv_objkey) = VALUE sofolenti1-doc_id( lt_note[ 1 ]-instid OPTIONAL ).

  CATCH cx_swf_cnt_elem_not_found.
  CATCH cx_swf_cnt_elem_type_conflict.
  CATCH cx_swf_cnt_unit_type_conflict.
  CATCH cx_swf_cnt_container.
ENDTRY.

SHIFT lv_user LEFT DELETING LEADING 'US'.
CONDENSE lv_user.

DATA(lv_decision) = io_result->get_result( )-result.

IF lv_decision = 'Approve'.
  UPDATE zle_train_head SET learn_status_id = 'A'
   WHERE learning_id = ls_lm_learn-learning_id.

  "->Send Email for Approval
  SELECT SINGLE * FROM zle_train_head
        INTO @DATA(ls_head)
        WHERE learning_id = @ls_lm_learn-learning_id.

  IF ls_head-trainer_uname IS NOT INITIAL.
    APPEND VALUE #(  user_id = ls_head-trainer_uname ) TO lt_recep.
  ENDIF.
  APPEND VALUE #(  user_id = lv_user ) TO lt_recep.
  ZLE_CL_TRAIN_WORKFLOW=>send_email( im_it_userid = lt_recep
                    im_learning = ls_lm_learn
                    im_email_type = 'A' ).
ELSE.
UPDATE zle_train_head SET learn_status_id = 'R' WHERE learning_id = ls_lm_learn-learning_id.

  CHECK lv_objkey IS NOT INITIAL.

  CALL FUNCTION 'SO_DOCUMENT_READ_API1'
    EXPORTING
      document_id              = lv_objkey
    TABLES
```

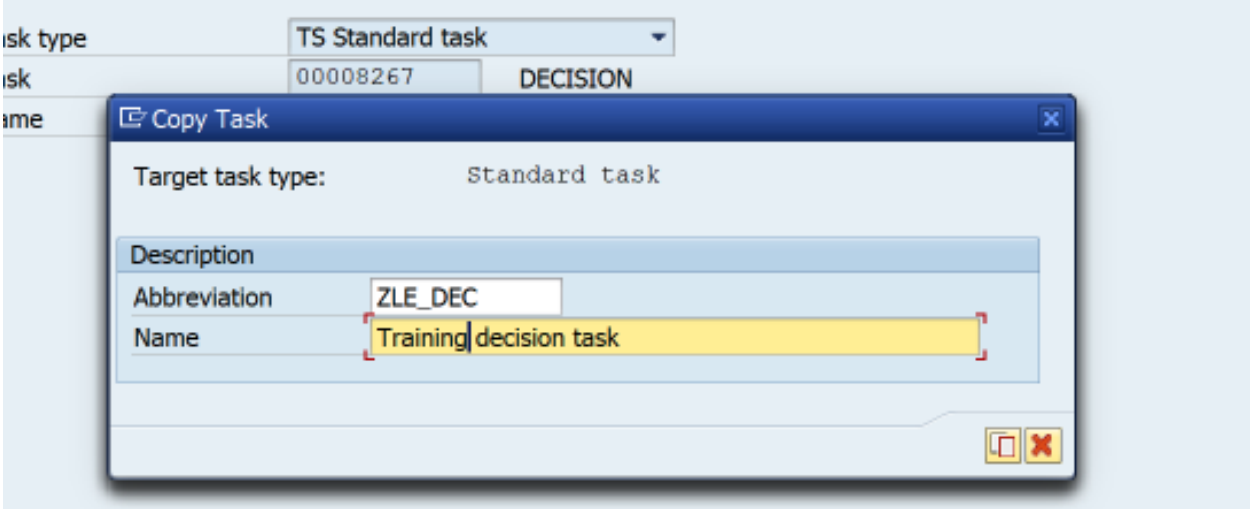
```
        object_content           = lt_objcont
      EXCEPTIONS
        document_id_not_exist     = 1
        operation_no_authorization = 2
        x_error                   = 3
        OTHERS                    = 4.
    IF sy-subrc <> 0.
* Implement suitable error handling here
    ENDIF.

    LOOP AT lt_objcont INTO DATA(ls_objcontent).
      lv_rejnote  = COND string( WHEN lv_rejnote  IS INITIAL
                                 THEN ls_objcontent-line
                                 ELSE |{ lv_rejnote  } { ls_objcontent-line }|  ).
    ENDLOOP.

    ""->Send Email for Rejection
    ZLE_CL_TRAIN_WORKFLOW=>send_email( im_it_userid = VALUE #( ( user_id = lv_user ) )
                       im_learning = ls_lm_learn
                       im_email_type = 'R'
                       im_rejection_note = lv_rejnote ).
  ENDIF.

  endmethod.
ENDCLASS.
```

3.5. Creation of Custom Decision Task to be used in Workflow TS92000178

Go to TCODE PFTC: Give the standard task number TS00008267 and copy it into a Z task

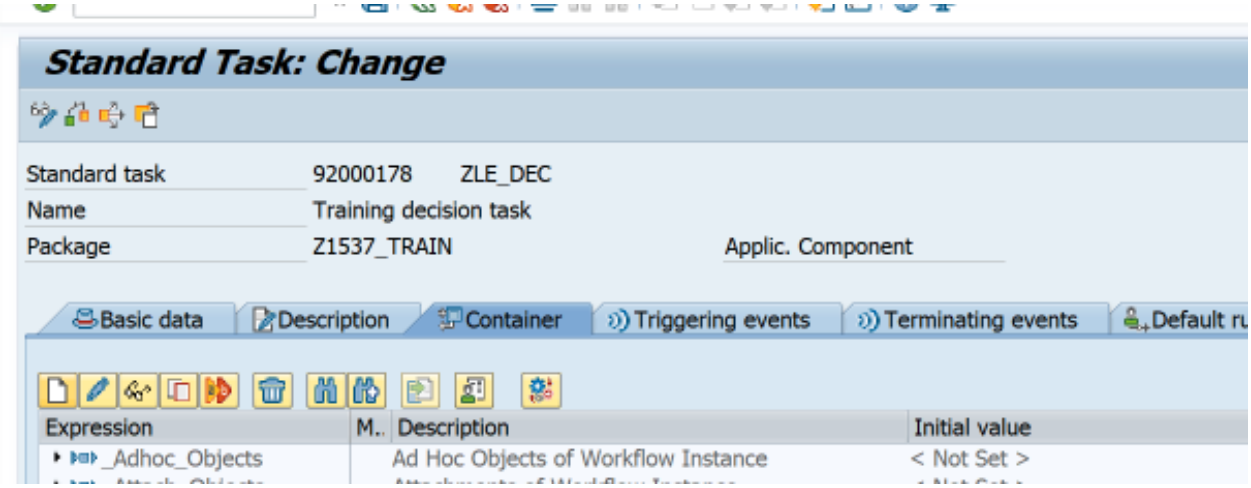


Give the necessary name and Click Copy

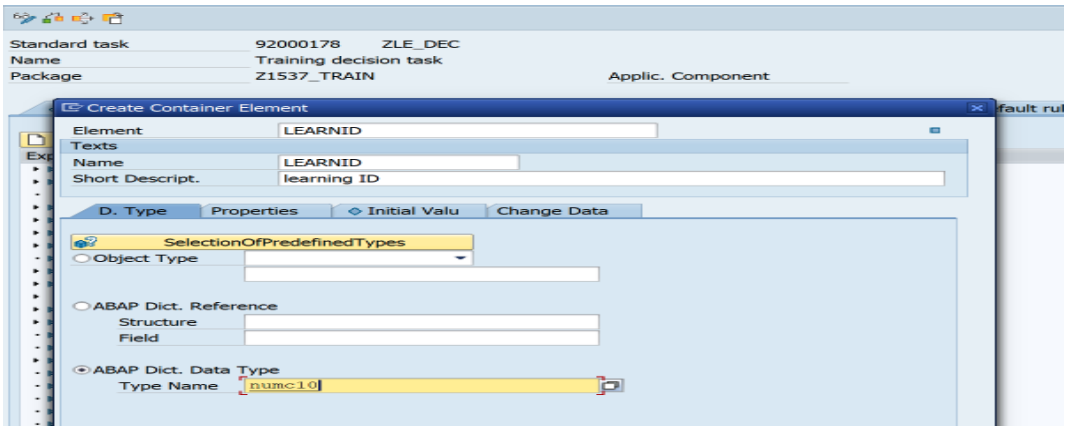


A new number for the task will be generated: TS92000178
We need the learning ID inside the task container hence create one element in it.

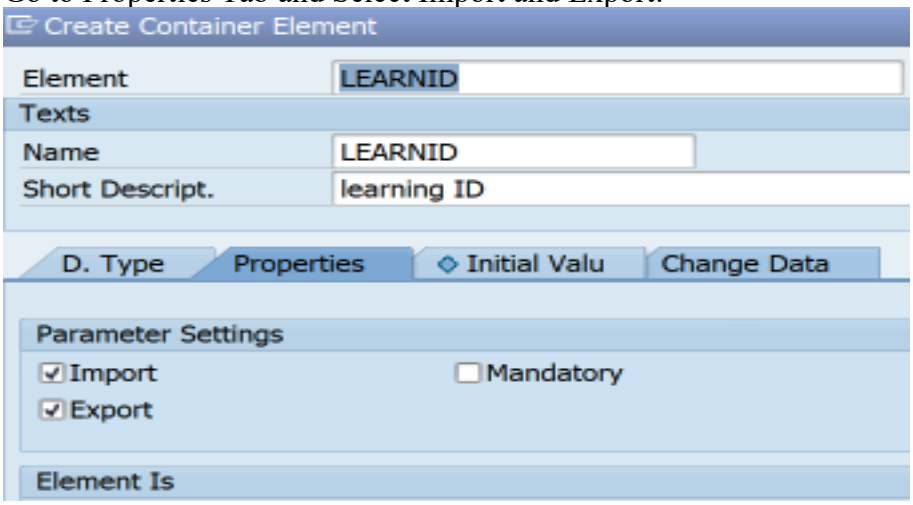
Click on Edit and go to the Container table and click create button which is present below Basic Data



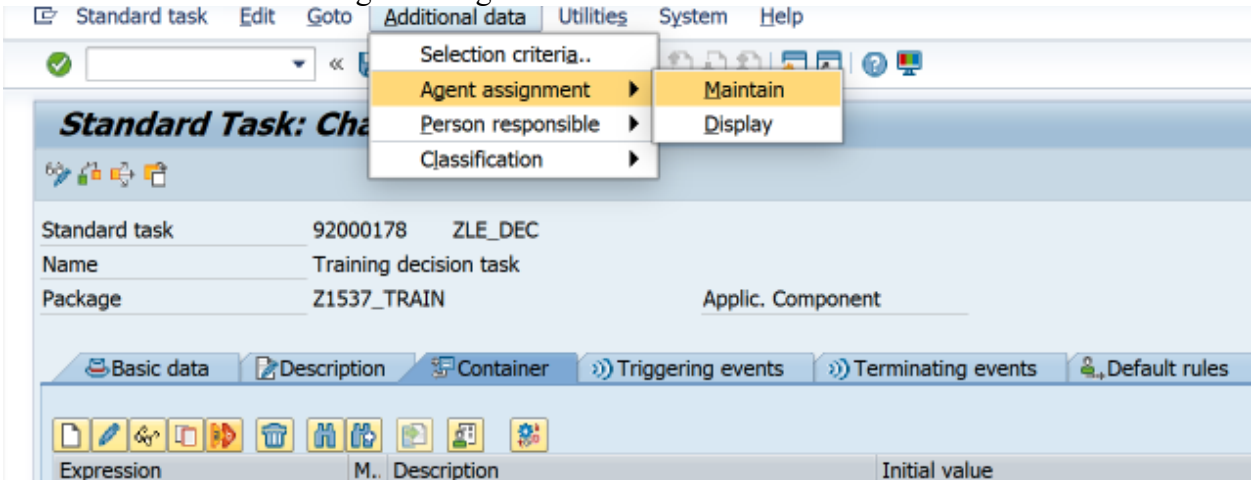
Add a new Element LEARNID:



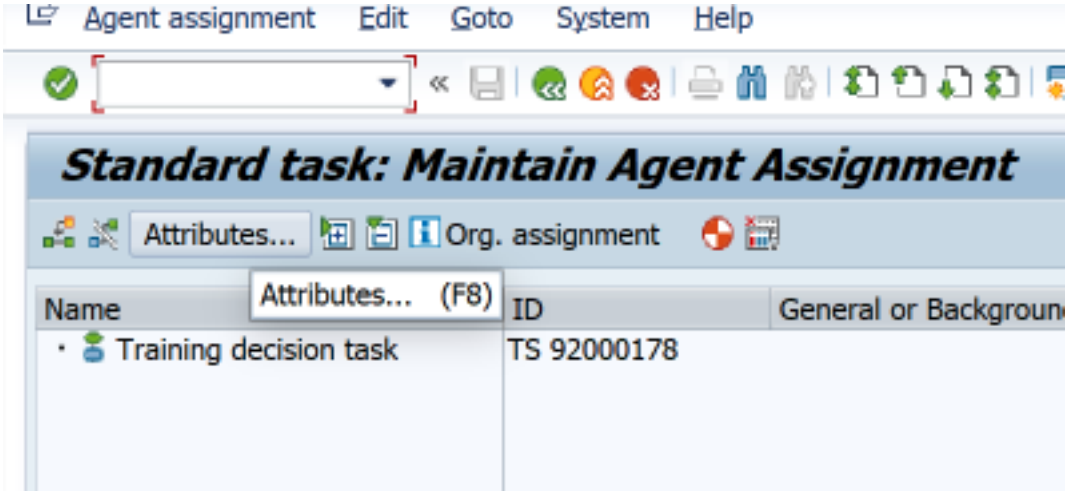
Go to Properties Tab and Select Import and Export:



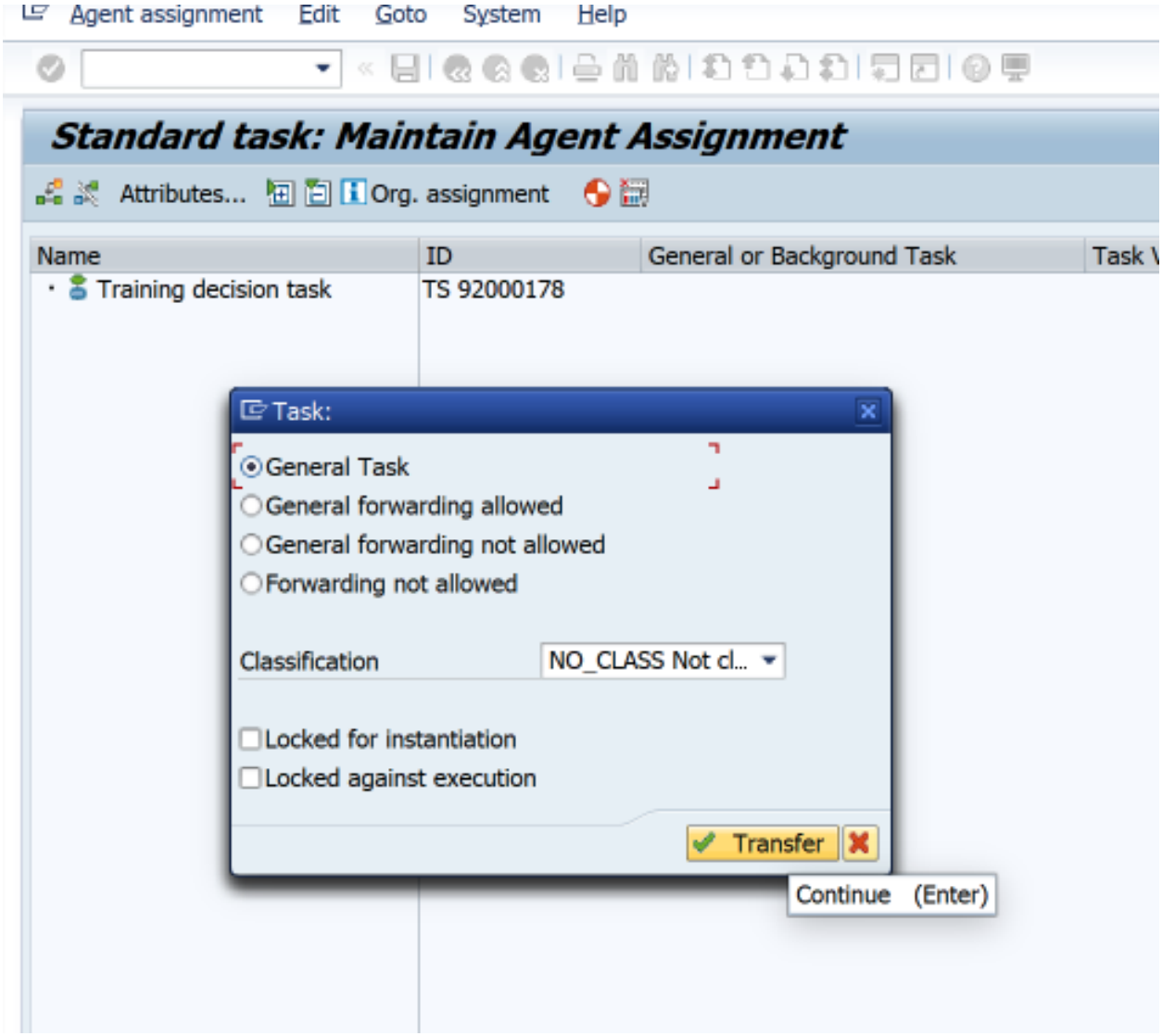
Make the task as General task:
Go to Additional Data -> Agent Assignment -> Maintain



Click on attributes

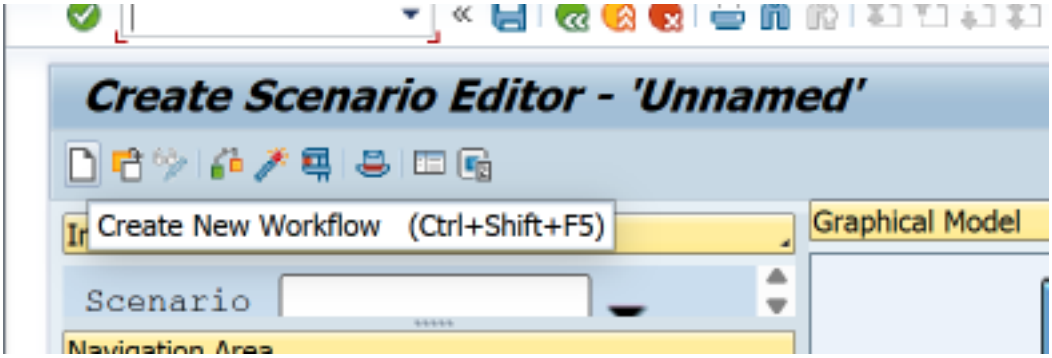


Select the General task and click on transfer

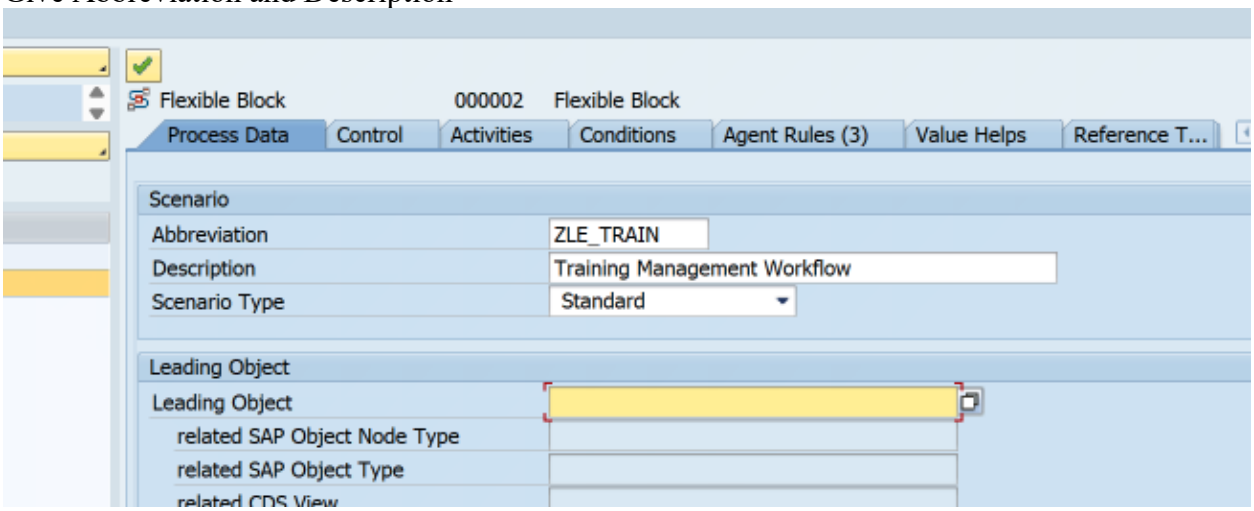


3.6. Creation of Workflow Template WS92000123

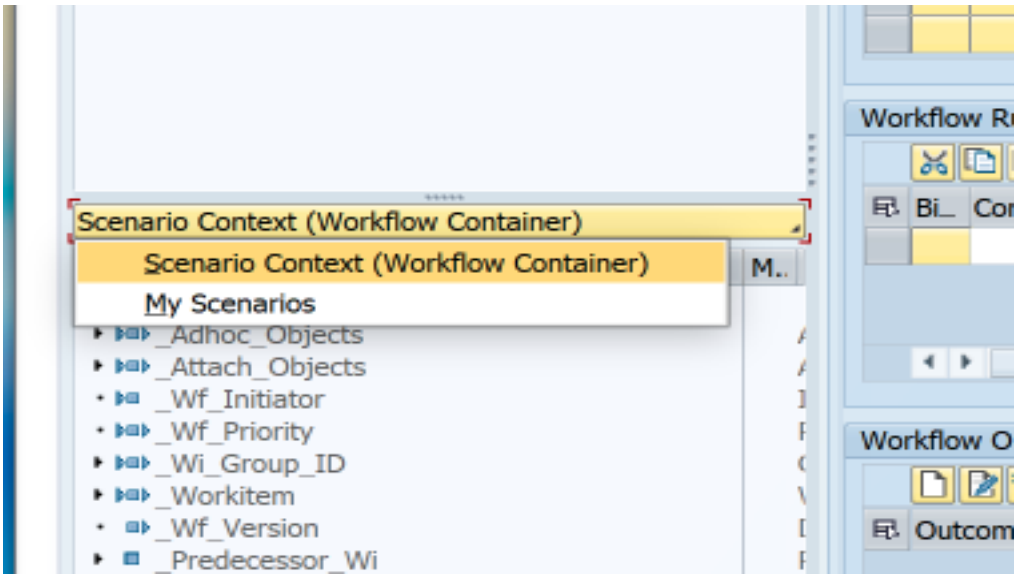
Got to TCODE SWDD_SCENARIO:
Click on Create New Workflow



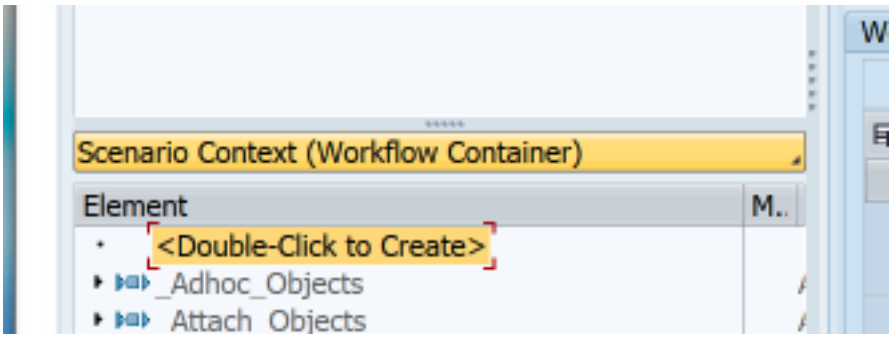
Give Abbreviation and Description



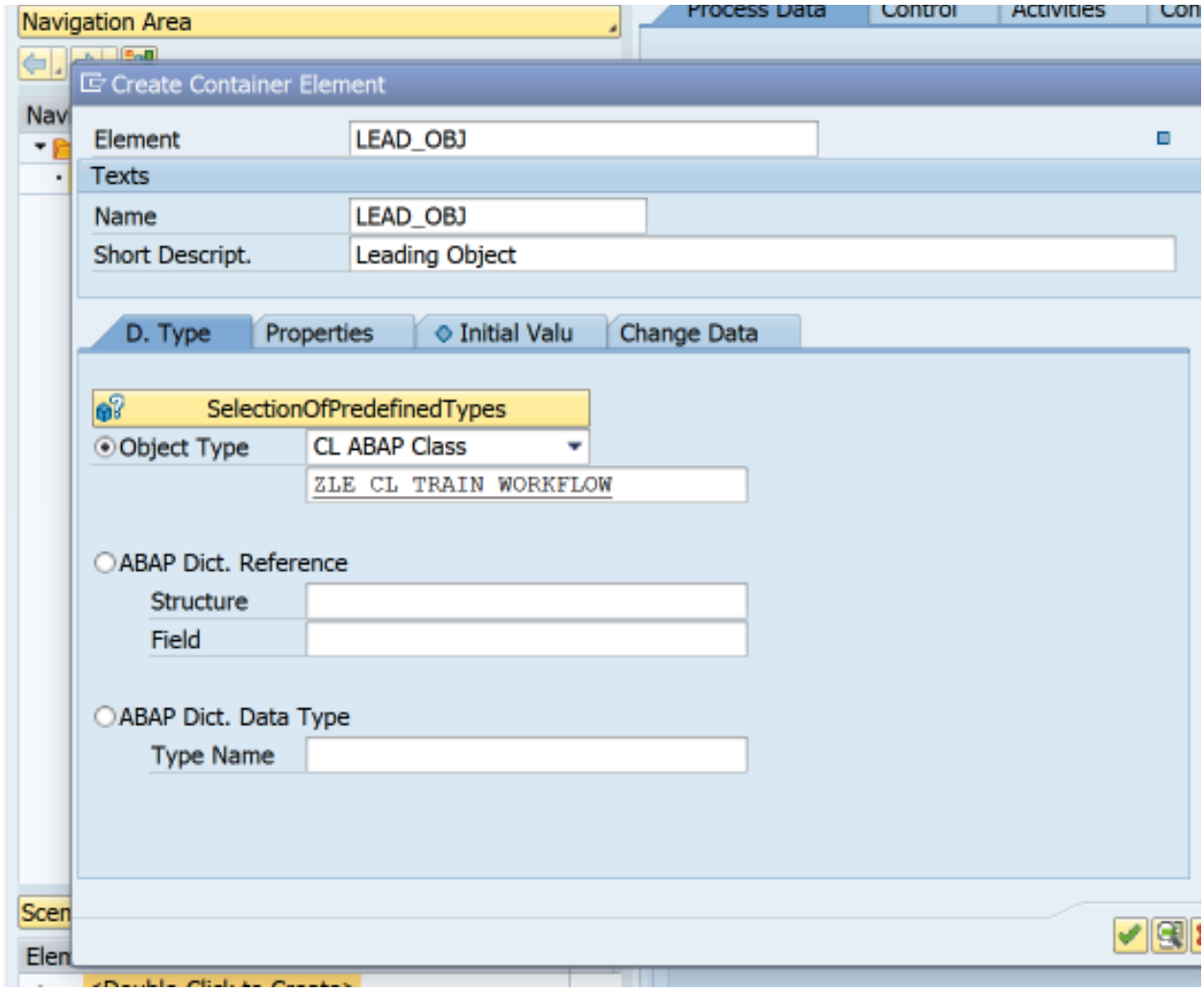
Select the Scenario context in the Left hand side



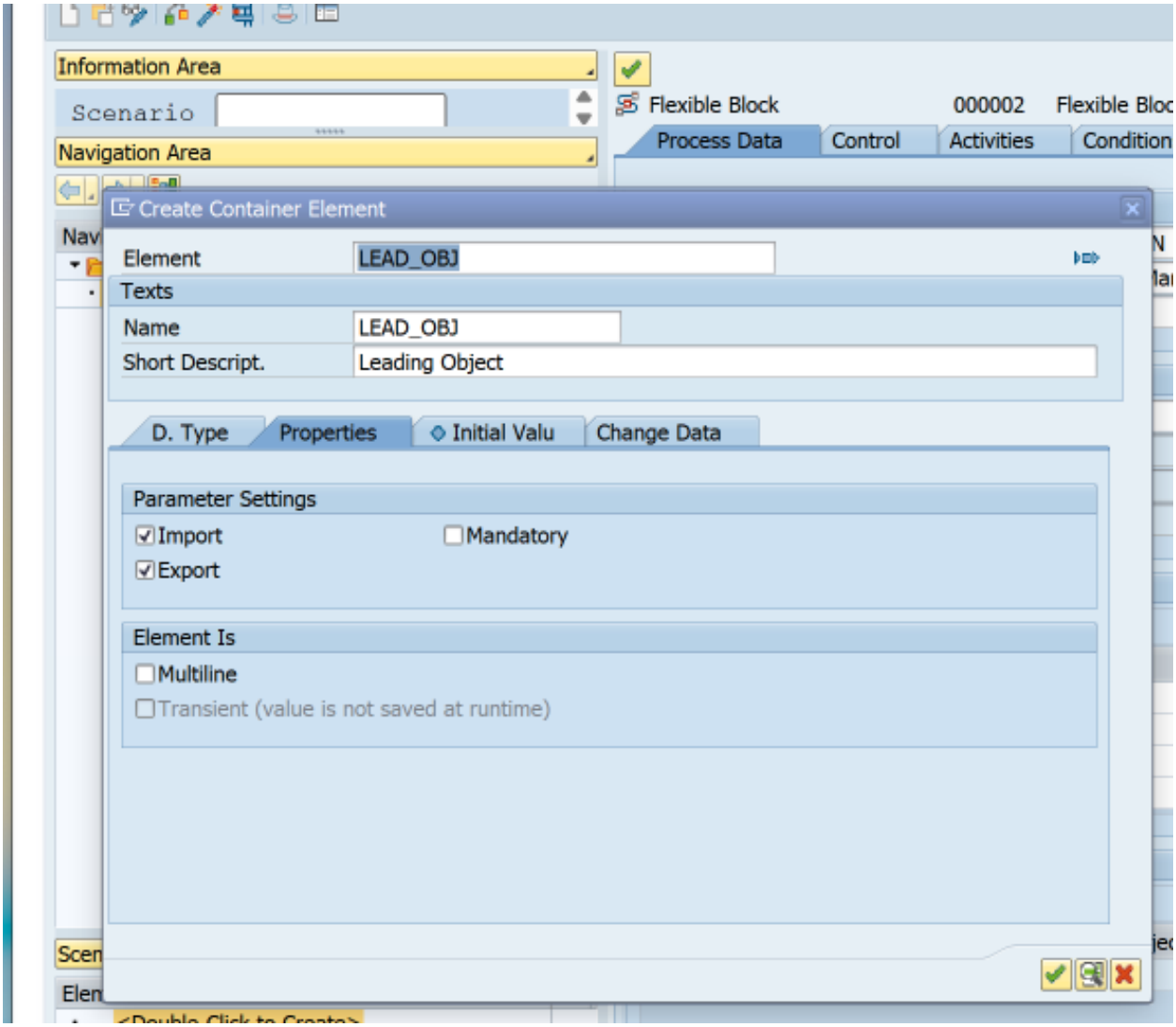
Create a new Container element called LEAD_OBJ for your workflow class using the Event Class we created above
Double Click to Create:



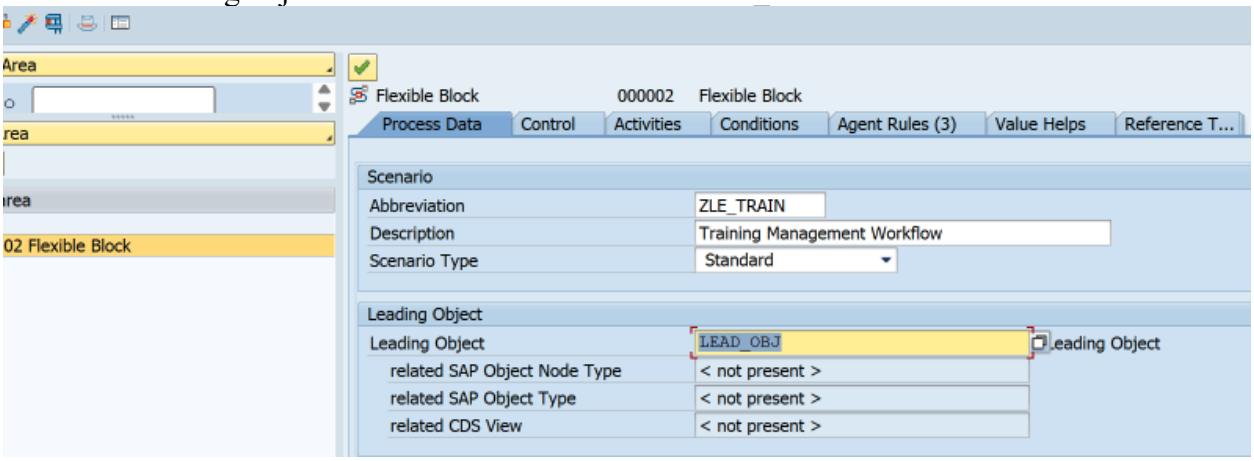
Add the Class Name here:



Go to Properties and Give import, export and select tick mark to confirm

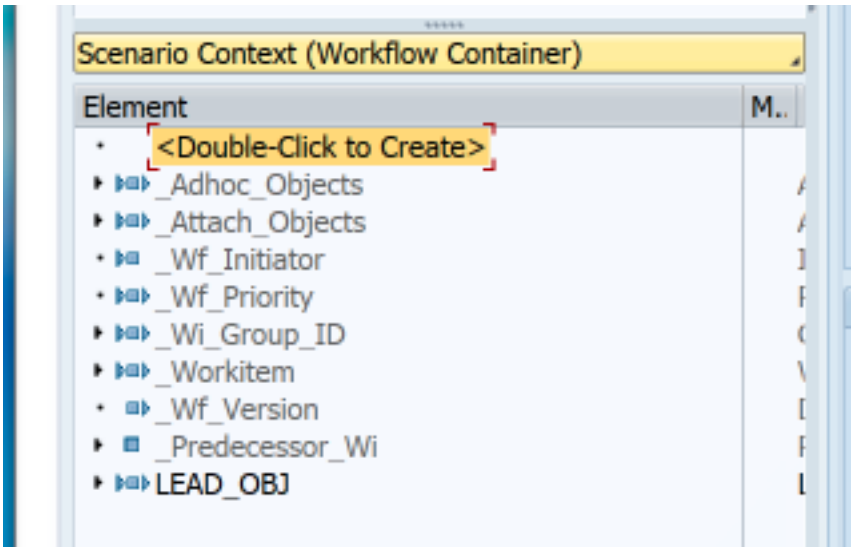


Select the Leading object f4 and choose the created LEAD_OBJ

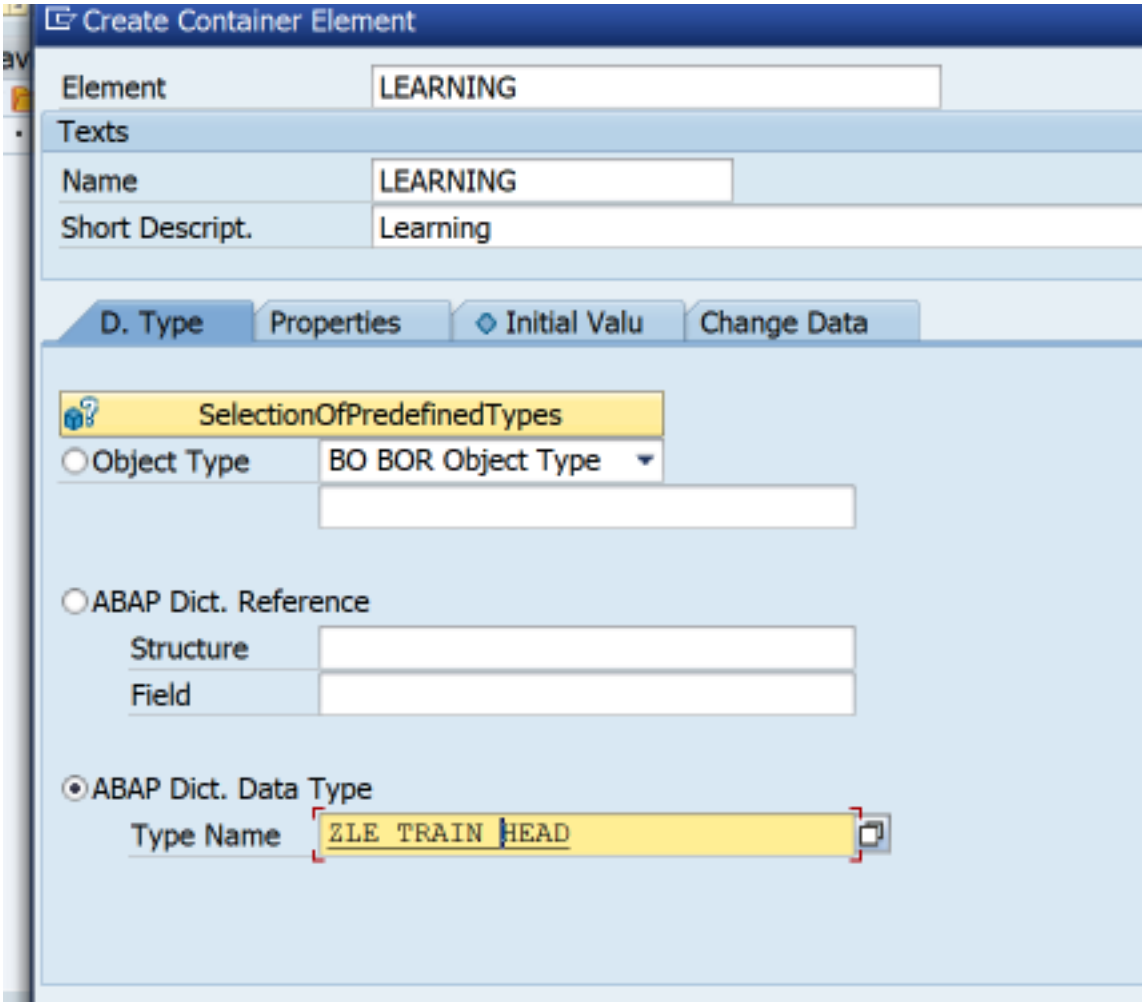


Create a new Container Object again for Training Input: LEARNING

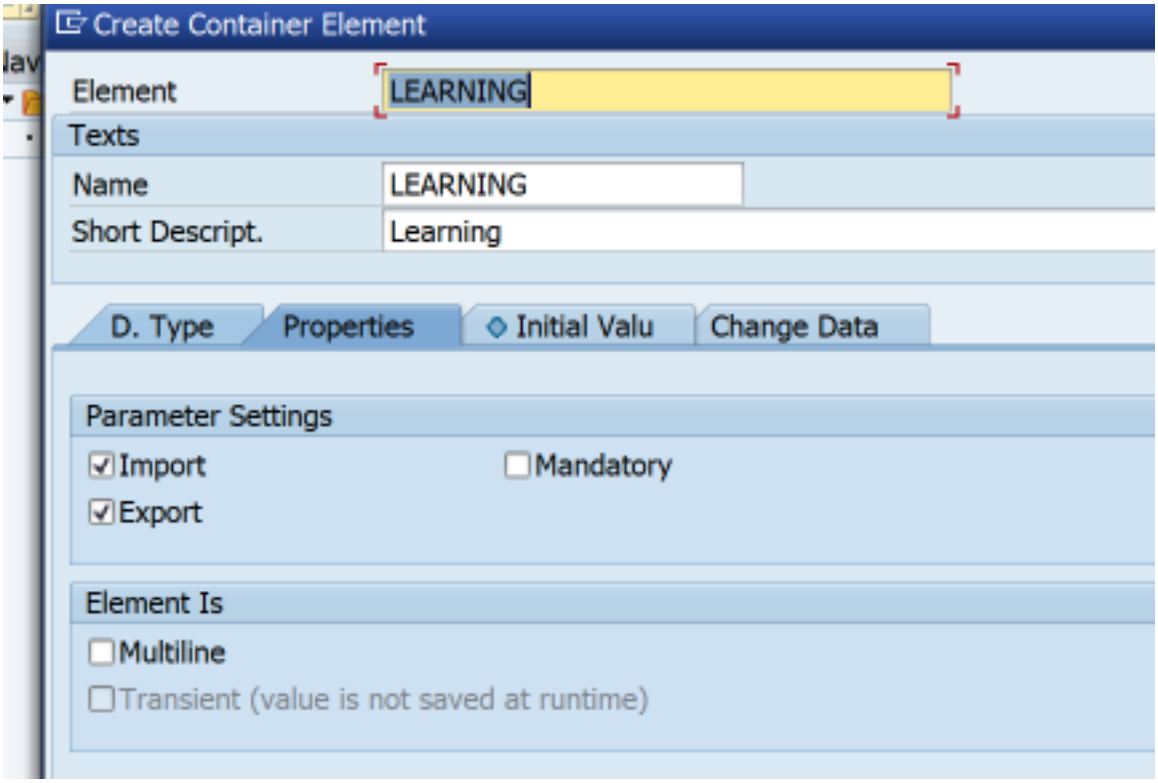
Double Click to Create:



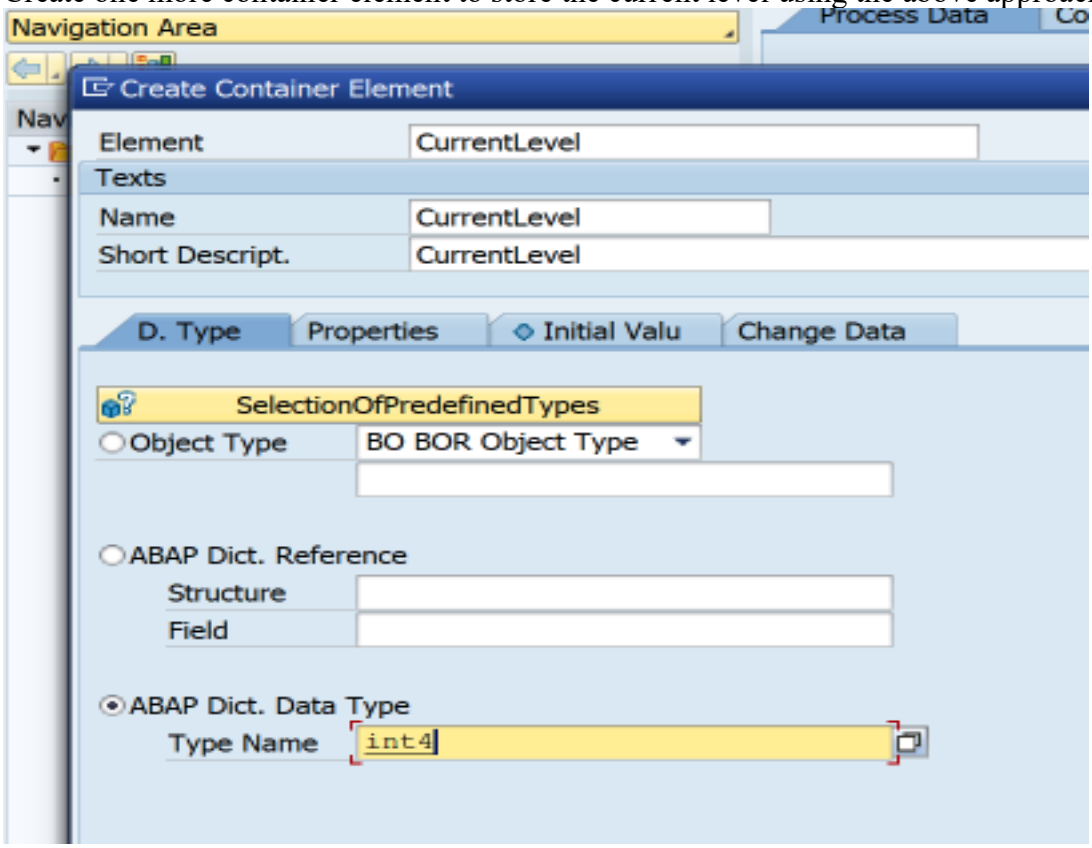
Add the Element



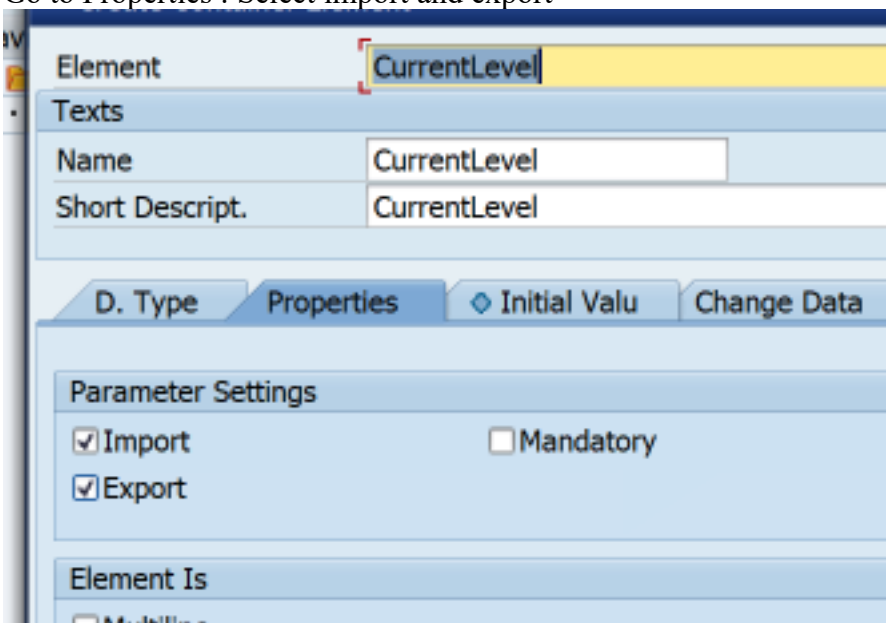
Go to Properties Tab : Give import and export



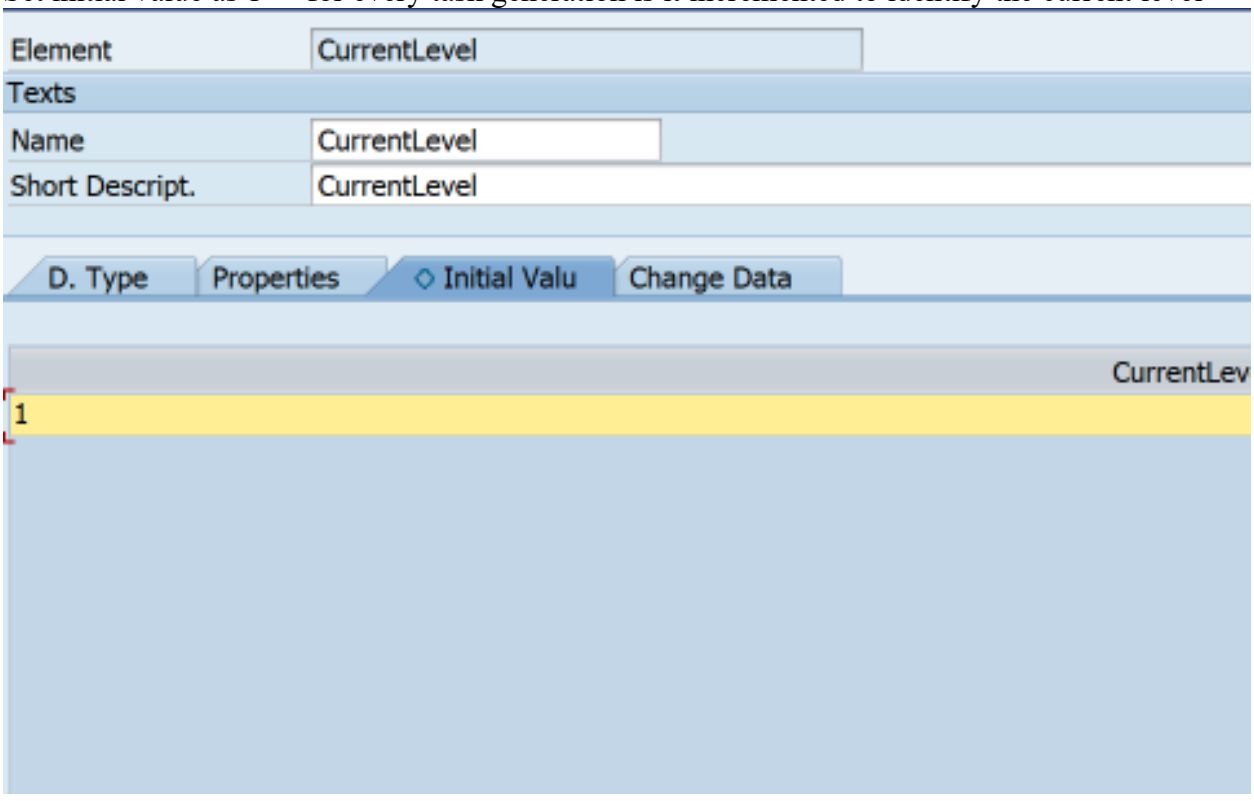
Create one more container element to store the current level using the above approach again



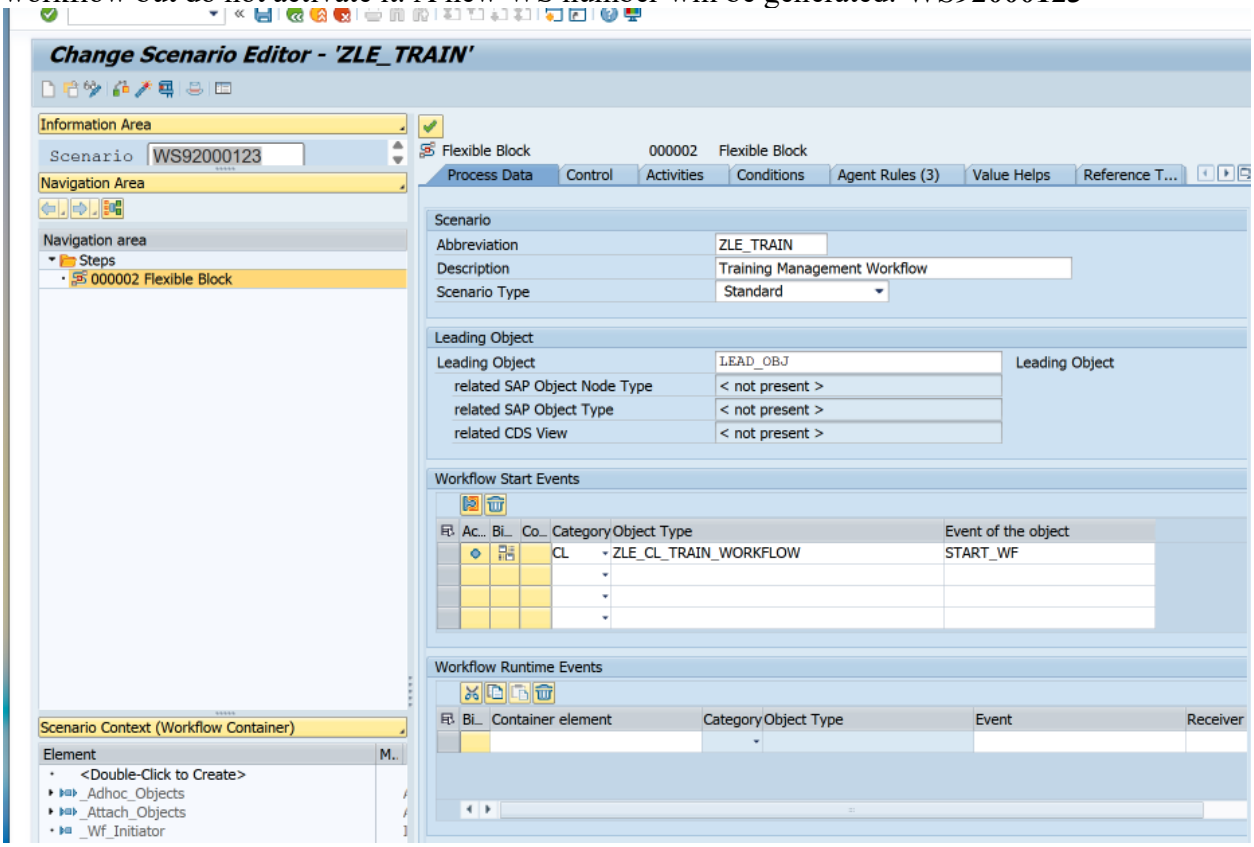
Go to Properties : Select import and export



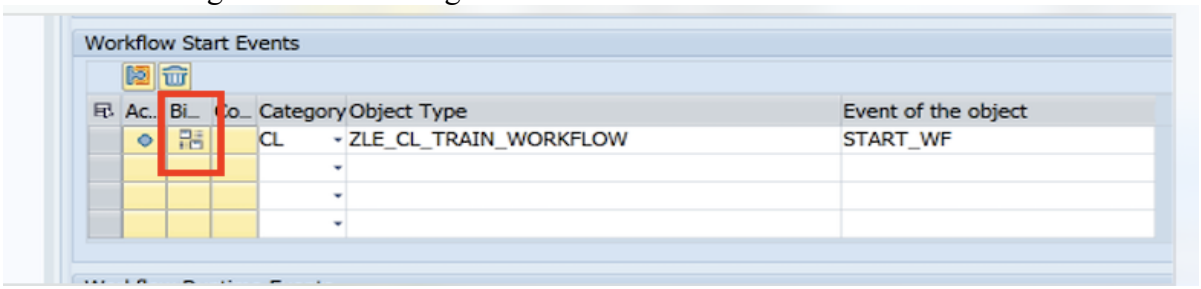
Set initial value as 1 -> for every task generation is it incremented to identify the current level



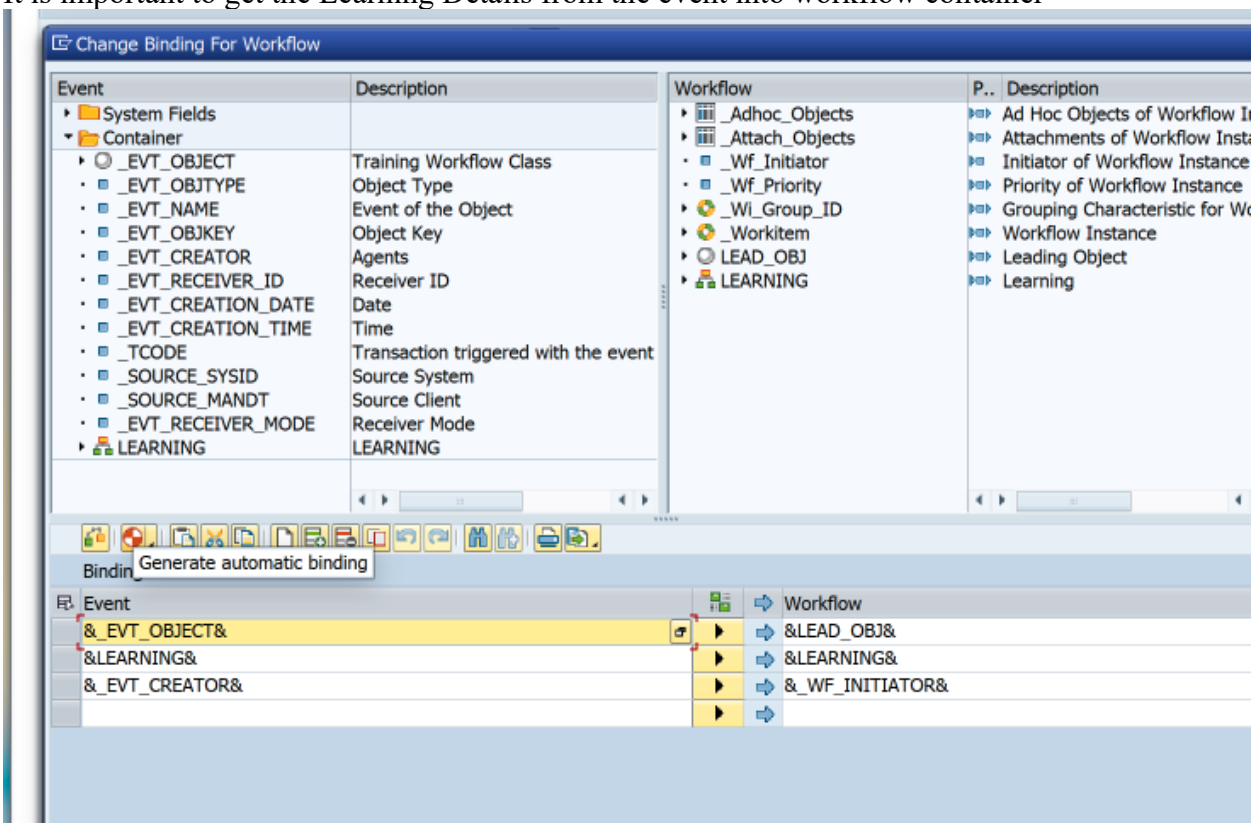
Go back to Process Data and Give the Class and Event in the Workflow Start events and save the workflow but do not activate it. A new WS number will be generated: **WS92000123**



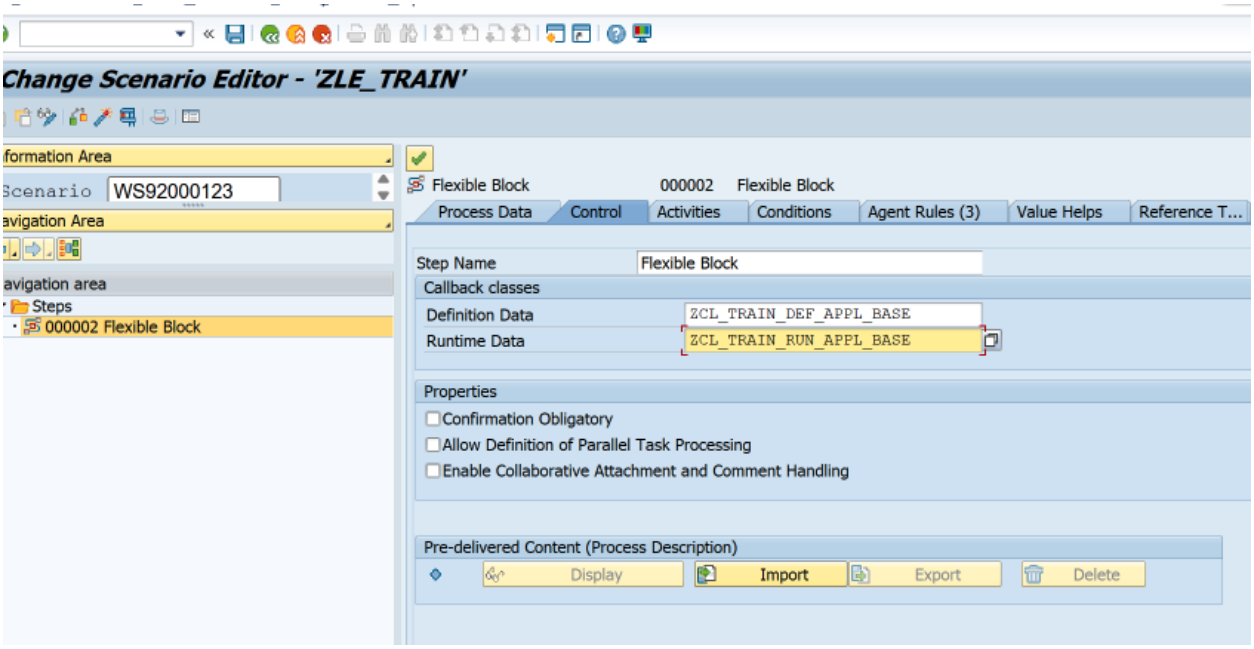
Click on Bindings to do the bindings



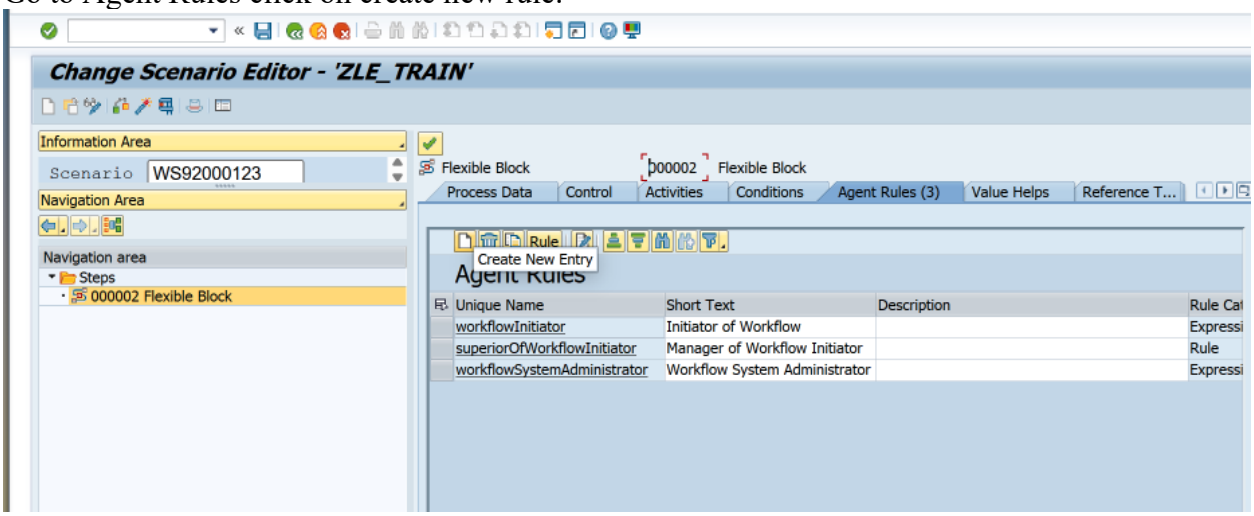
It is important to get the Learning Details from the event into workflow container



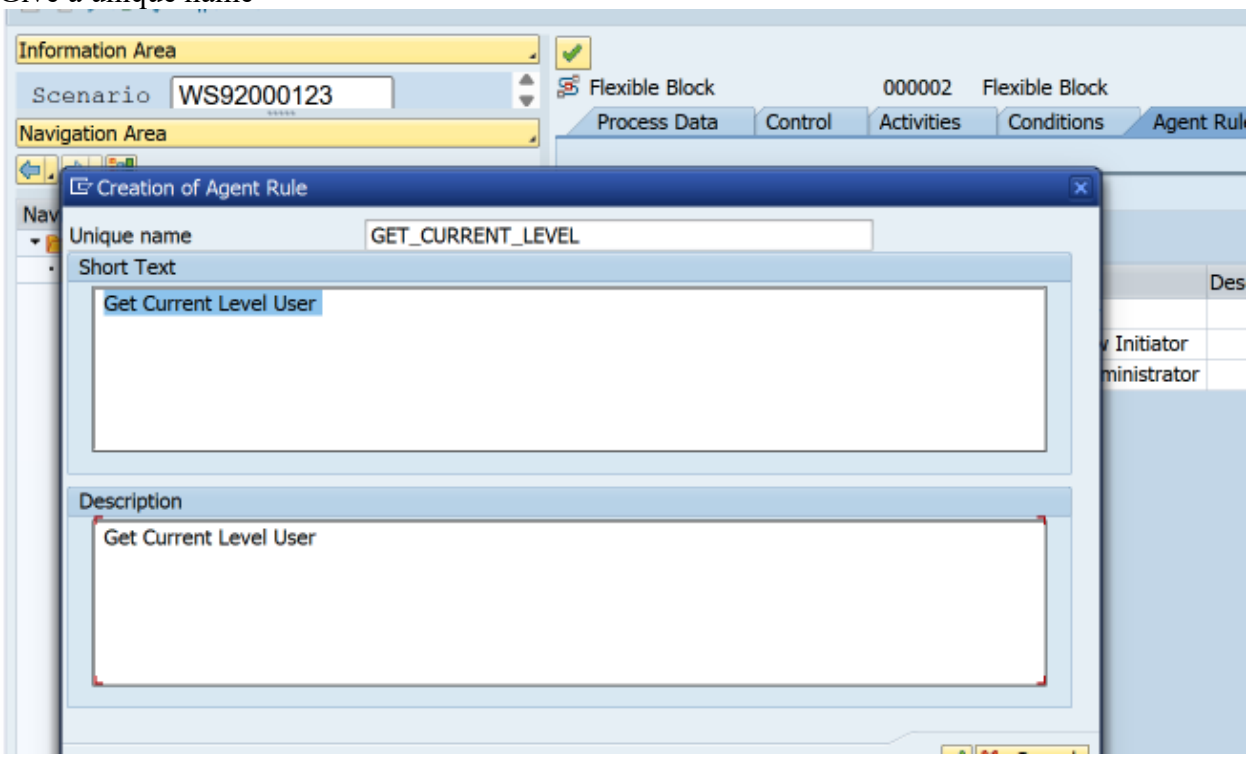
Go to the Control Tab Provide the Runtime and Definition class names that we created above



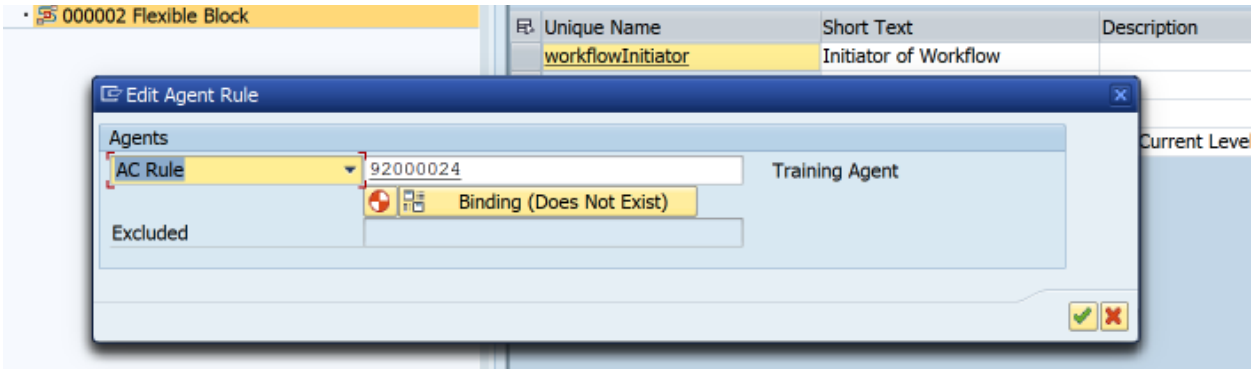
Go to Agent Rules click on create new rule:



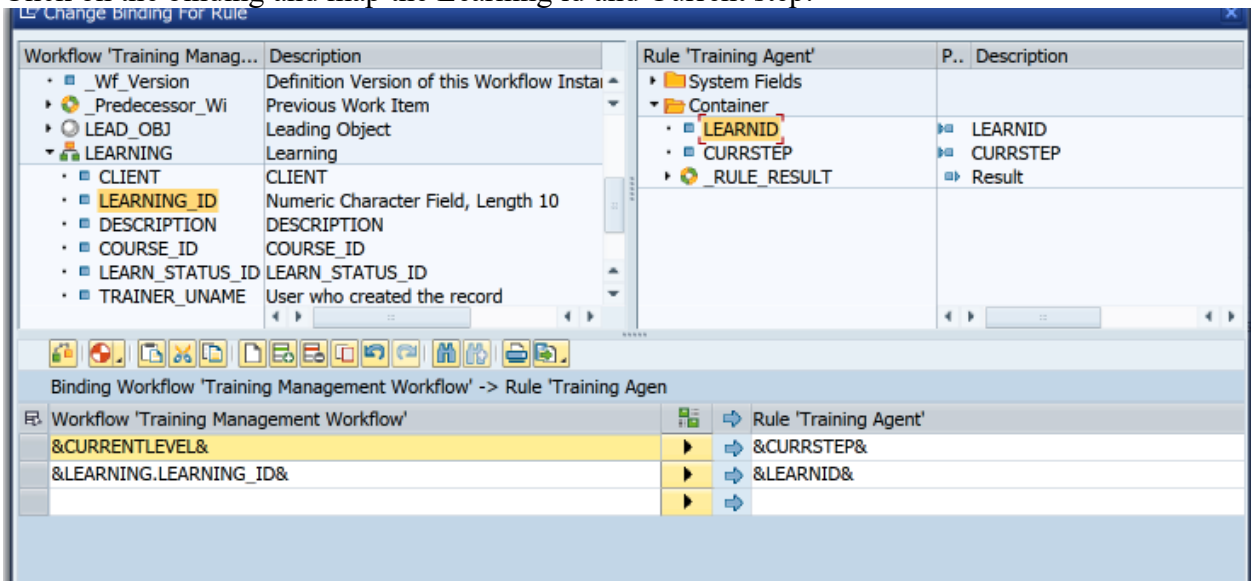
Give a unique name



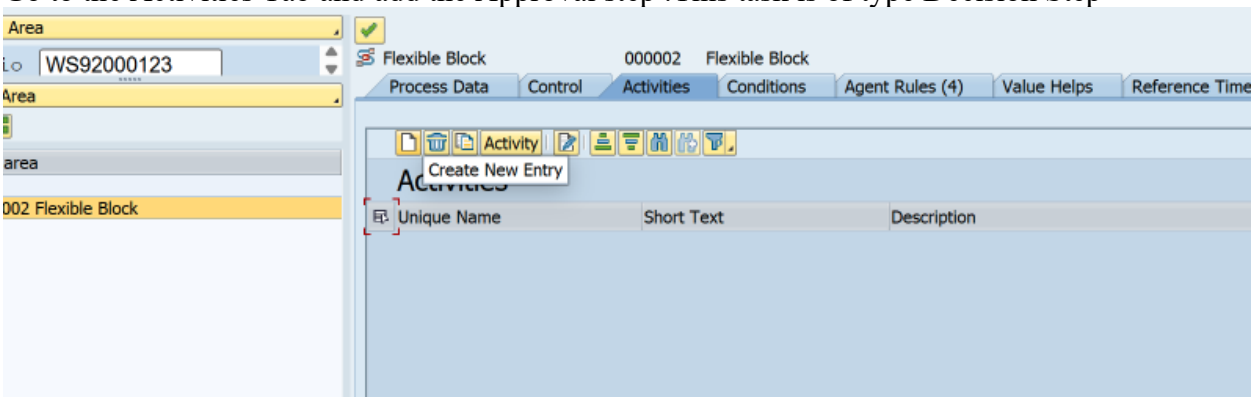
Select Rule in agents and give the Agent Rule number which we created above:



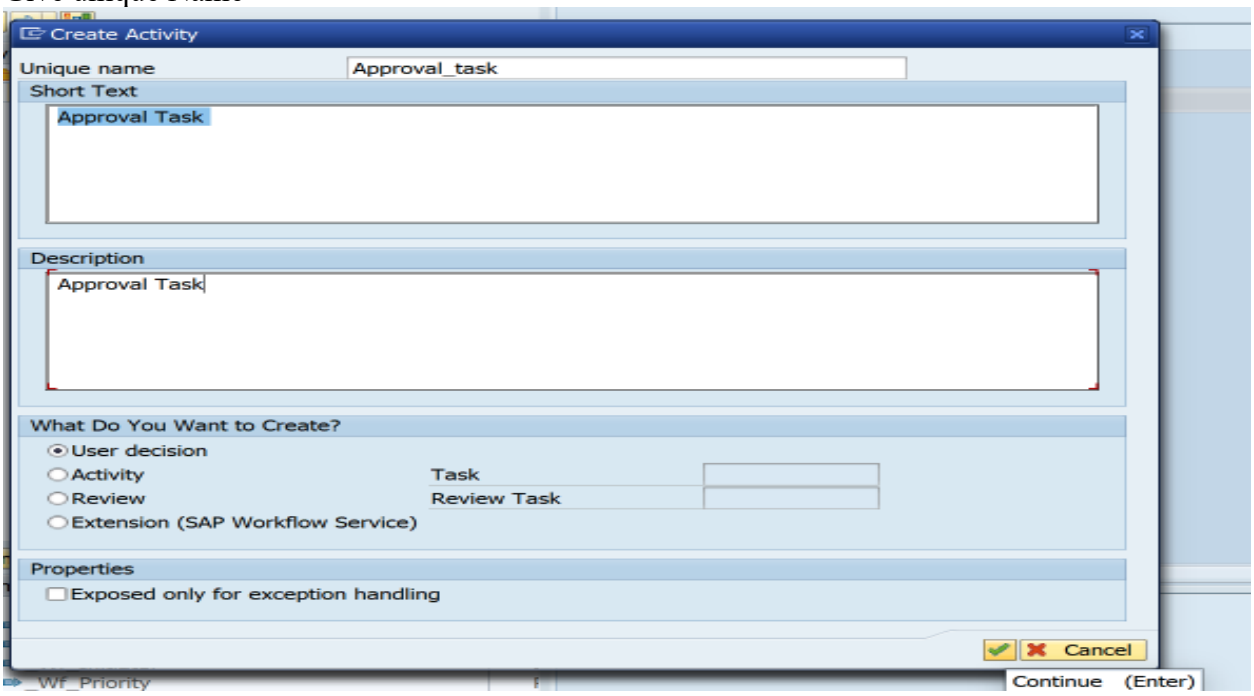
Click on the binding and map the Learning id and Current step:



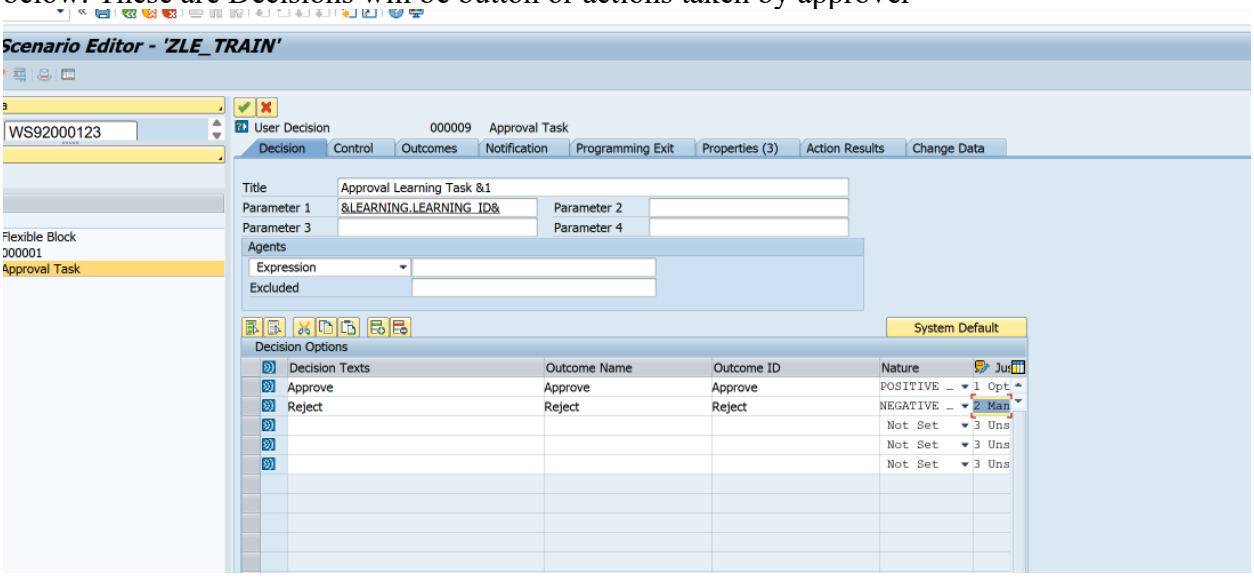
Go to the Activities Tab and add the Approval step .This task is of type Decision Step



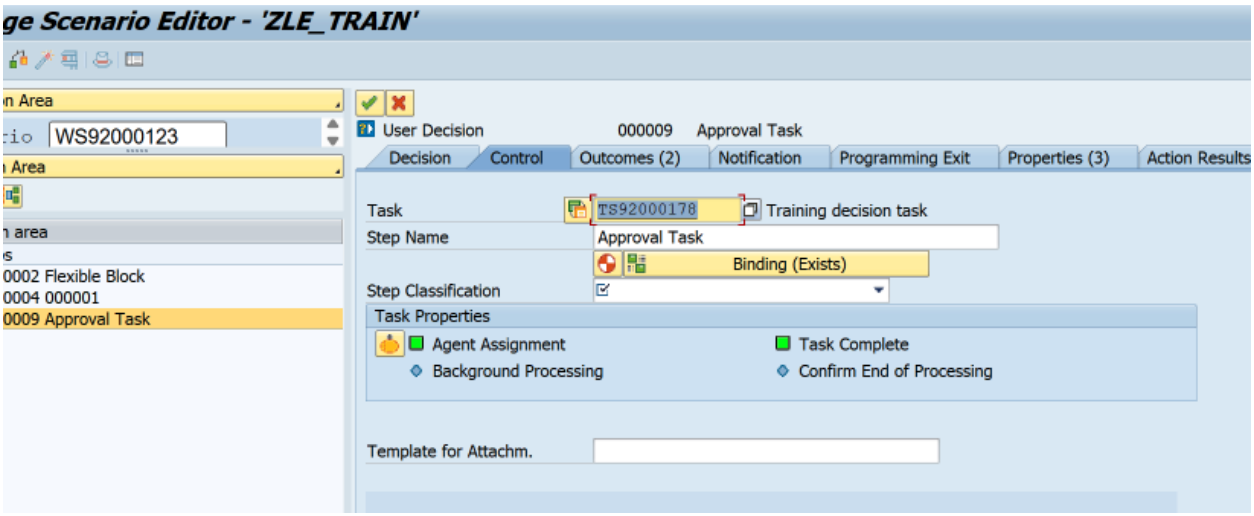
Give unique Name



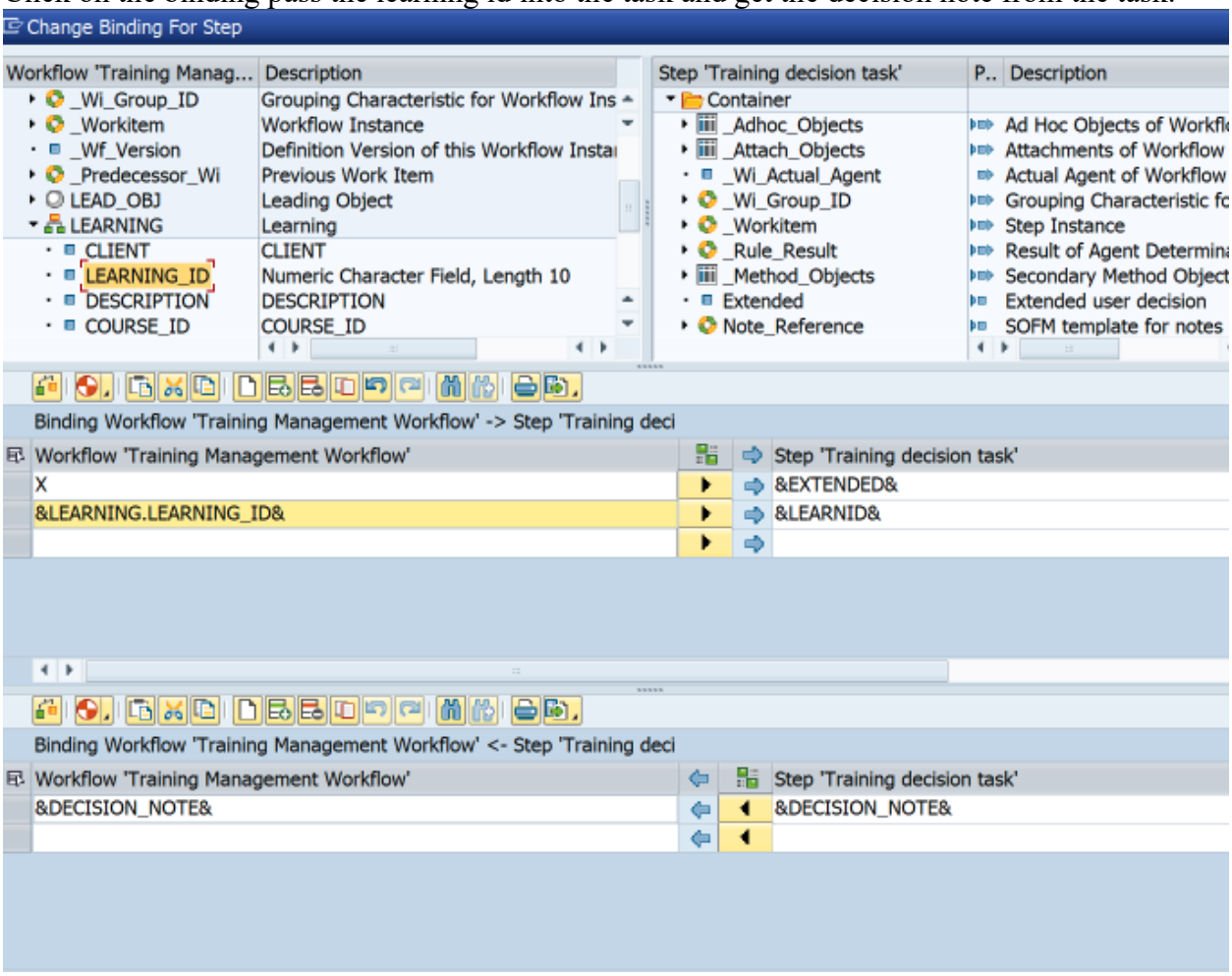
Give the necessary Title with parameters, decisions and its nature and justifications as mentioned below. These are Decisions will be button or actions taken by approver



Go to Control and replace the standard task with Custom task we created before:



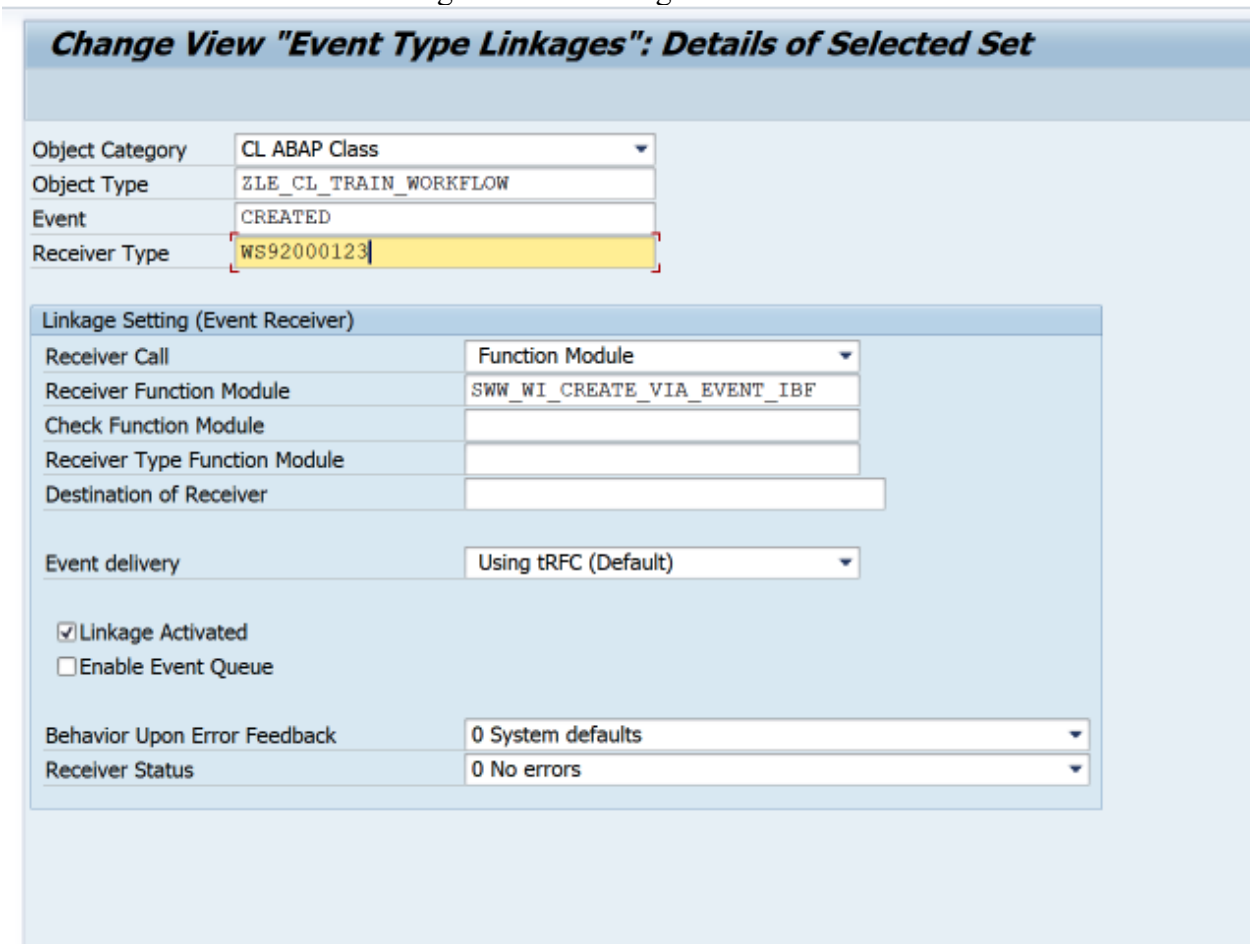
Click on the binding pass the learning id into the task and get the decision note from the task.



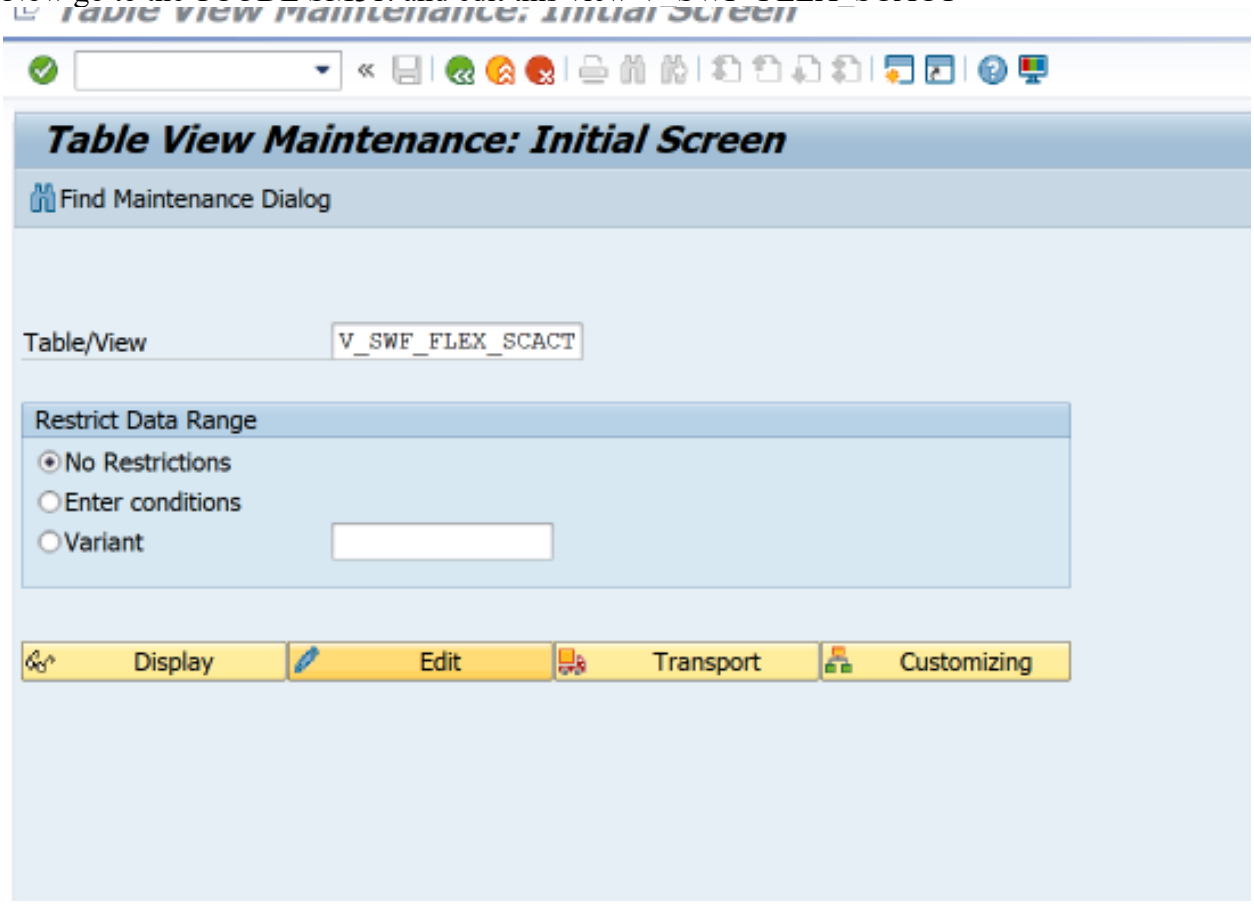
Save do a syntax check and activate the workflow:

Then we have to do the event type linkage to link the class event to the workflow
Open a new session and go to TCODE SWE2 for event type linkage: map the workflow number with the class and event.
Make sure the Linkage Activated is selected

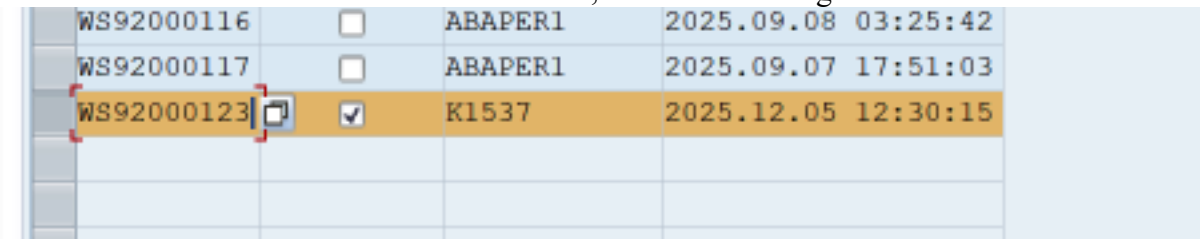
And click on add new entries and give the following and save it:



Now go to the TCODE SM31: and edit this view V_SWF_FLEX_SCACT



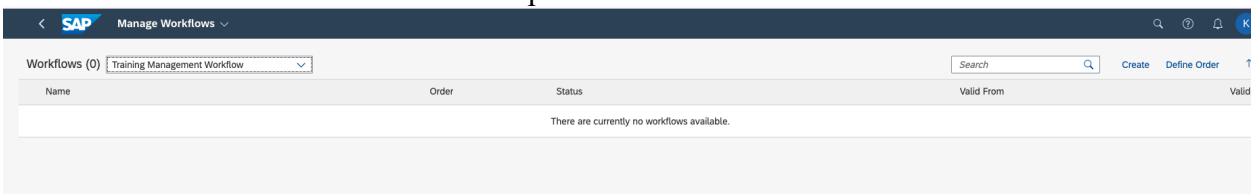
Your workflow number will be there as active, if not add it using create new entries



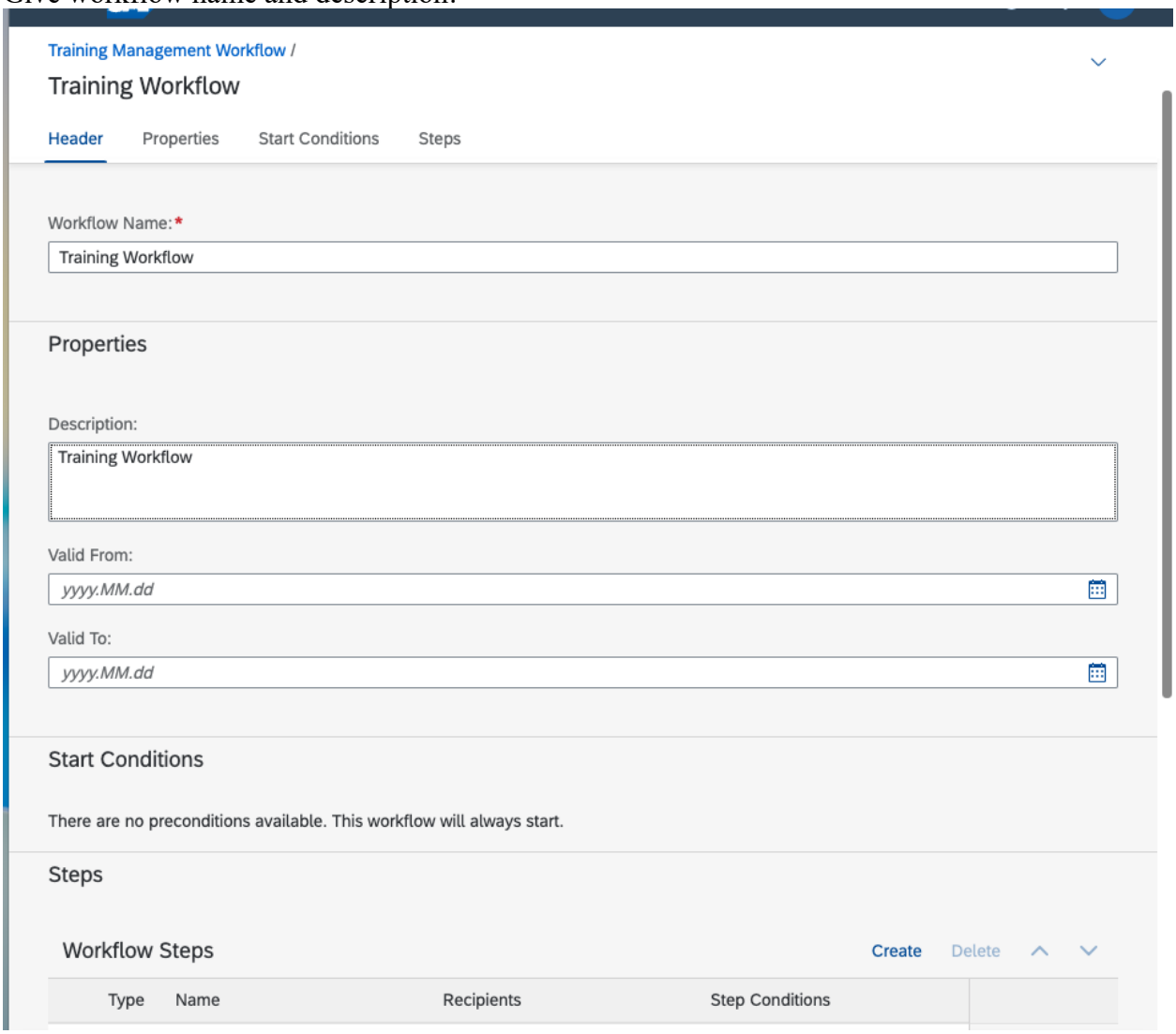
3.7. Configuration of Flexible Workflow

Let us Configure the workflow in the Manage workflow Fiori Application:
Open Fiori Launchpad and click on the Manage workflows (use Fiori app library for the necessary roles to enable the app for your role) app

Select the created workflow from the dropdown:

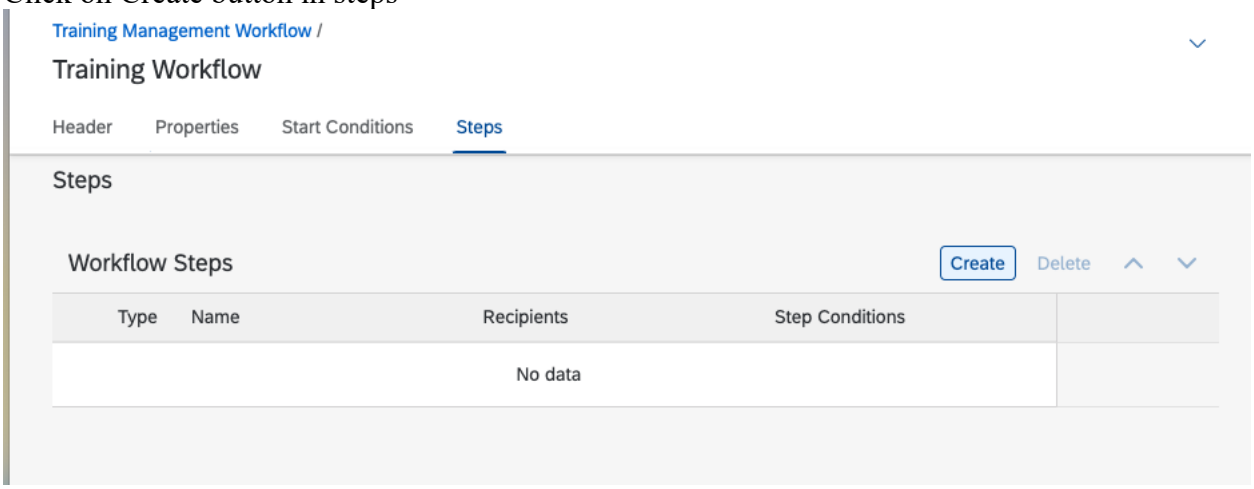


Click on Create new button to configure a new workflow:
Give workflow name and description:

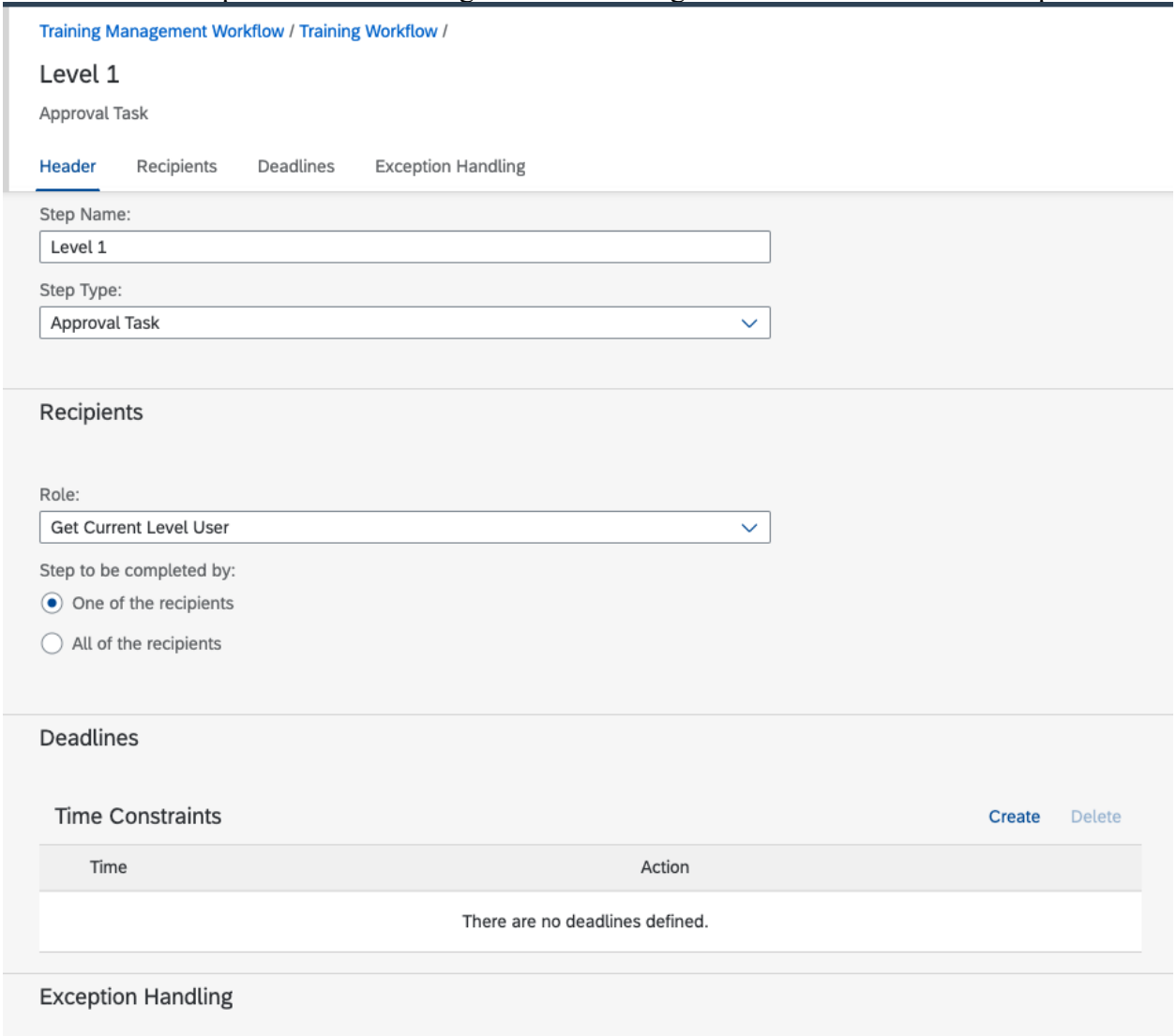


Click on create in the steps, we can add the decisions steps we created in workflow template any number of times we want, this makes the workflow as flexible

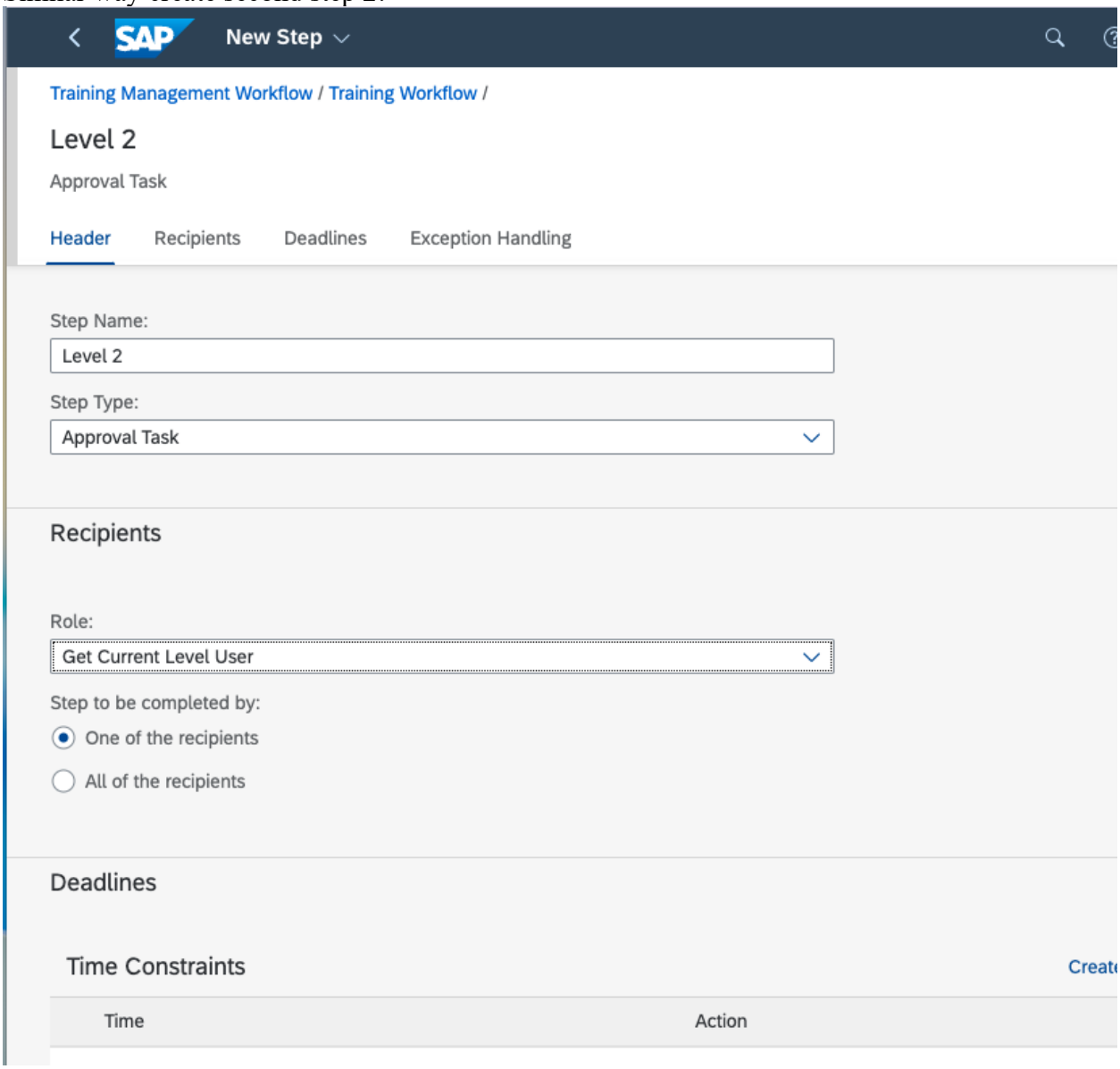
Click on Create button in steps



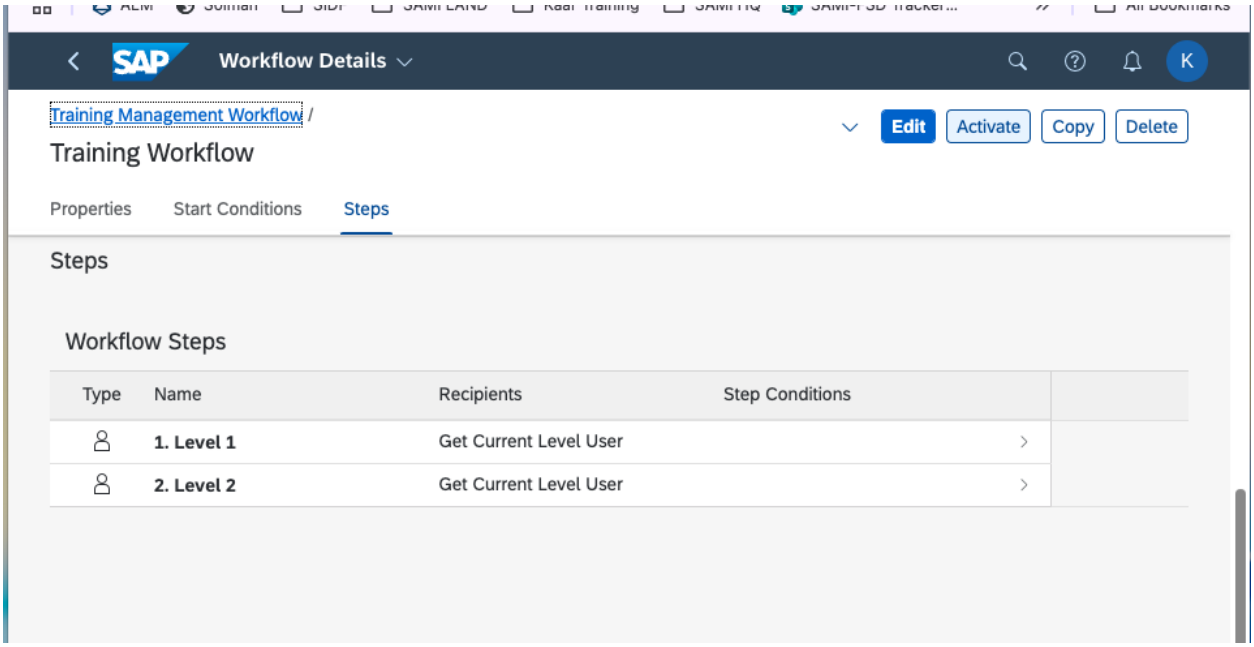
Give Level 1 as inputs and in the role give the created agent role in the workflow Template



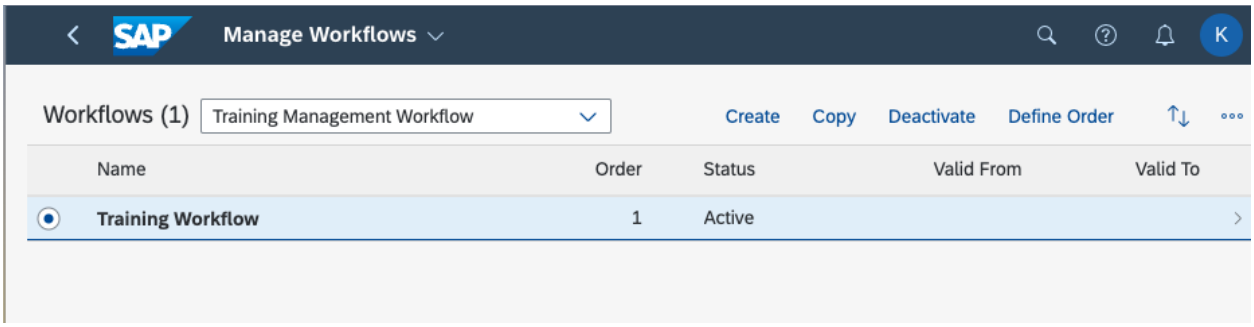
Similar way create second step 2:



Click on Save and Activate it.



Final screen will look like this



4. Integrate the Workflow in RAP Application

Then we need to integrate the workflow in behavior definition and implementation with a custom action button. This is already present in the above full code for behavior definition and implementation

4.1. Behavior Definition

A New Custom Action in behavior definition

[AK4] ZI_ZLM_HEAD | [AK4] ZBI_TRAIN_HEAD | [AK4] ZBP_I_ZLM_HEAD | [AK4] ZBP_BI_TRAIN_HEAD | [AK4] ZMDE_TRAIN_HEAD

```
managed implementation in class zbp_bi_train_head unique;
strict ( 2 );

define behavior for zbi_train_head alias train_head
persistent table ZLE_TRAIN_HEAD
lock master
authorization master ( instance )
//etag master <field_name>
early numbering
{
  //Read only Fields
  field ( readonly ) LearningId,LearnStatusId,CreatedBy,CreatedOn;
  //Mandatory Fields
  field ( mandatory ) CourseId,TrainerUname;

  //Determinations
  determination setStatusDraft on modify { create; }

  //Validations
  validation mandatoryCourseId on save { create; field CourseId; }
  validation mandatoryTrainerName on save { create; field TrainerUname; }

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


  create;
  update (features : instance );
  delete (features : instance );
  association _TrainItem { create (features : instance ); }
```

4.2. Meta Data Extension

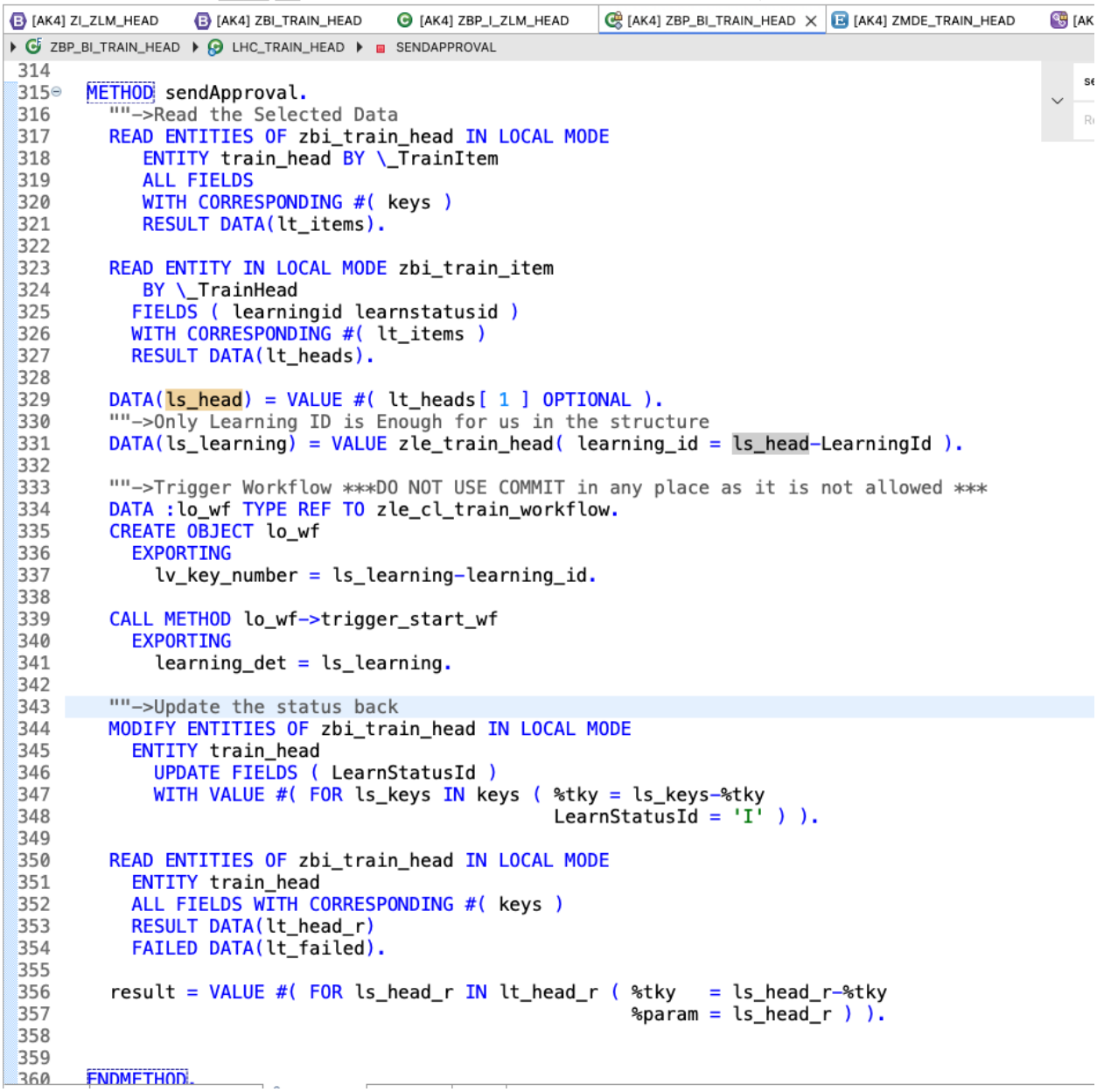
Enablement of Data Action in Metadata extension:

[AK4] ZI_ZLM_HEAD | [AK4] ZBI_TRAIN_HEAD | [AK4] ZBP_I_ZLM_HEAD | [AK4] ZBP_BI_TRAIN_HEAD | [AK4] ZMDE_TRAIN_HEAD | [AK4] ZSBIN_TRAIN

```
},
presentationVariant: [{ sortOrder: [{ by: 'LearningId',direction: #DESC }] }] }
@Search.searchable: true
annotate entity ZCO_TRAIN_HEAD with
{
  //UI Facets
  @UI.facet: [{ id:'Learning', purpose : #STANDARD, type :#IDENTIFICATION_REFERENCE, position : 10, lab
              { id:'Items', purpose : #STANDARD, type :#LINEITEM_REFERENCE, position : 20, label: 'Item

  @Search.defaultSearchElement: true
  @Search.fuzzinessThreshold: 0.5
  @UI.identification: [ { position: 10 } ]
  @UI.lineItem: [ { position: 10 } ]
  @EndUserText.label: 'Learning Id'
  LearningId;

  @Search.defaultSearchElement: true
  @Search.fuzzinessThreshold: 0.7
  @UI.identification: [ { position: 20 } ]
  @UI.lineItem: [ { position: 20 } ]
  @EndUserText.label: 'Description'
  Description;

  @UI.identification: [ { position: 30 } ]
  @UI.lineItem: [ { position: 30 } ]
  @EndUserText.label: 'Course'
  @Consumption.valueHelpDefinition: [{ entity:{ name: 'ZLE_COURSE_VH', element: 'CourseId' }}]
  @UI.selectionField: [{ position : 30 }]
  CourseId;
  @UI.identification: [{ position: 40, criticality: 'StatusCriticality' }]
  @UI.lineItem: [{ position: 40, criticality: 'StatusCriticality' },
                 { type : #FOR_ACTION, dataAction: 'sendApproval', label: 'Send Approval' }]
  @EndUserText.label: 'Learning Status'
  @Consumption.valueHelpDefinition: [{ entity:{ name: 'ZLE_STATUS_VH', element: 'StatusId' }}]
  @UI.selectionField: [{ position : 40 }]
```

4.3. Class Implementation

Implementation of Custom Action in the behavior implementation class to trigger the workflow



```
[AK4] ZI_ZLM_HEAD | [AK4] ZBI_TRAIN_HEAD | [AK4] ZBP_I_ZLM_HEAD | [AK4] ZBP_BI_TRAIN_HEAD | [AK4] ZMDE_TRAIN_HEAD
ZBP_BI_TRAIN_HEAD ▸ LHC_TRAIN_HEAD ▸ SENDAPPROVAL
```

```
314
315  METHOD sendApproval.
316    ""->Read the Selected Data
317    READ ENTITIES OF zbi_train_head IN LOCAL MODE
318       ENTITY train_head BY \_TrainItem
319       ALL FIELDS
320       WITH CORRESPONDING #( keys )
321       RESULT DATA(lt_items).
322
323    READ ENTITY IN LOCAL MODE zbi_train_item
324       BY \_TrainHead
325      FIELDS ( learningid learnstatusid )
326      WITH CORRESPONDING #( lt_items )
327      RESULT DATA(lt_heads).
328
329    DATA(ls_head) = VALUE #( lt_heads[ 1 ] OPTIONAL ).
330    ""->Only Learning ID is Enough for us in the structure
331    DATA(ls_learning) = VALUE zle_train_head( learning_id = ls_head-LearningId ).
332
333    ""->Trigger Workflow ***DO NOT USE COMMIT in any place as it is not allowed ***
334    DATA :lo_wf TYPE REF TO zle_cl_train_workflow.
335    CREATE OBJECT lo_wf
336      EXPORTING
337        lv_key_number = ls_learning-learning_id.
338
339    CALL METHOD lo_wf->trigger_start_wf
340      EXPORTING
341        learning_det = ls_learning.
342
343    ""->Update the status back
344    MODIFY ENTITIES OF zbi_train_head IN LOCAL MODE
345      ENTITY train_head
346        UPDATE FIELDS ( LearnStatusId )
347        WITH VALUE #( FOR ls_keys IN keys ( %tky = ls_keys-%tky
348                                            LearnStatusId = 'I' ) ).
349
350    READ ENTITIES OF zbi_train_head IN LOCAL MODE
351      ENTITY train_head
352      ALL FIELDS WITH CORRESPONDING #( keys )
353      RESULT DATA(lt_head_r)
354      FAILED DATA(lt_failed).
355
356    result = VALUE #( FOR ls_head_r IN lt_head_r ( %tky   = ls_head_r-%tky
357                                                   %param = ls_head_r ) ).
358
359
360  ENDMETHOD.
```

4.4. Behavior Definition Consumption View

Give the action in the behavior definition of Consumption view ZCO_TRAIN_HEAD

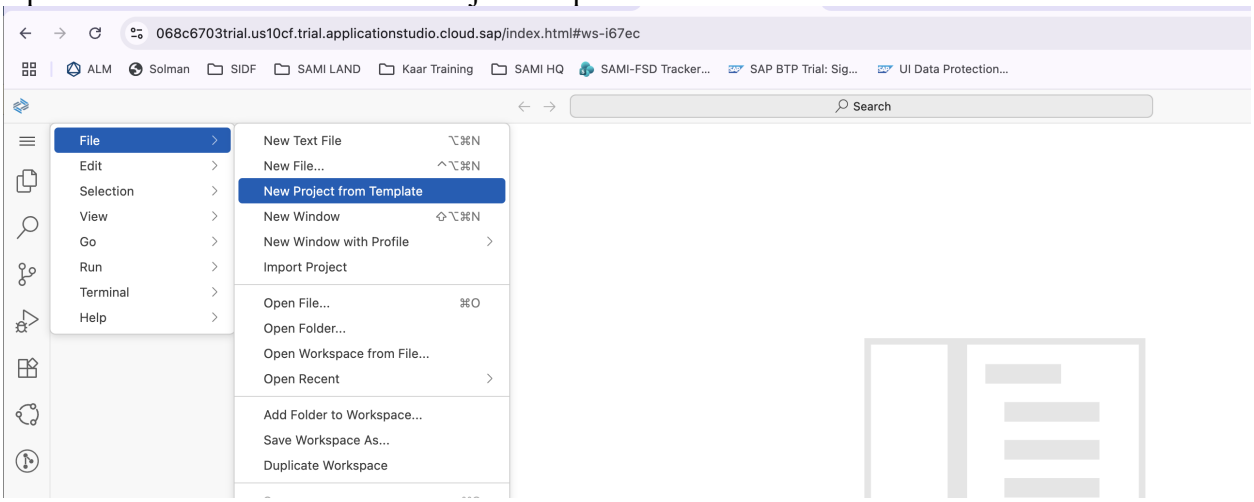
```
projection;
strict ( 2 );

define behavior for ZCO_TRAIN_HEAD alias train_head
{
  use create;
  use update;
  use delete;
  use action sendApproval;

  use association _TrainItem { create; }
}

define behavior for ZCO_TRAIN_ITEM alias train_item
{
  use update;
  use delete;

  use association _TrainHead;
}
```

5. BAS: Create and Deploy Fiori Application

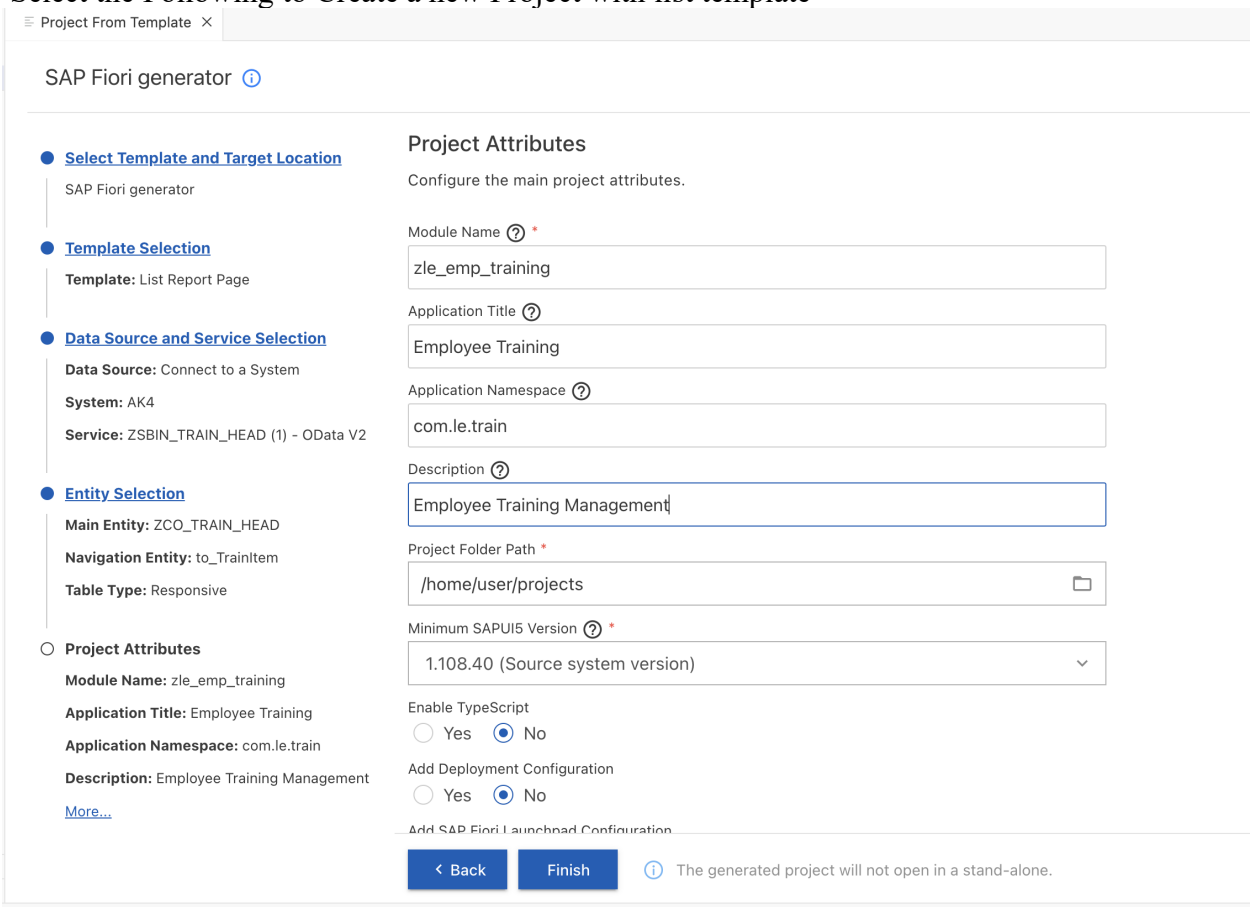
5.1. Creation of Fiori Application

Here I am using the Trail version and have the cloud connector ready which is connected to my on-premise system

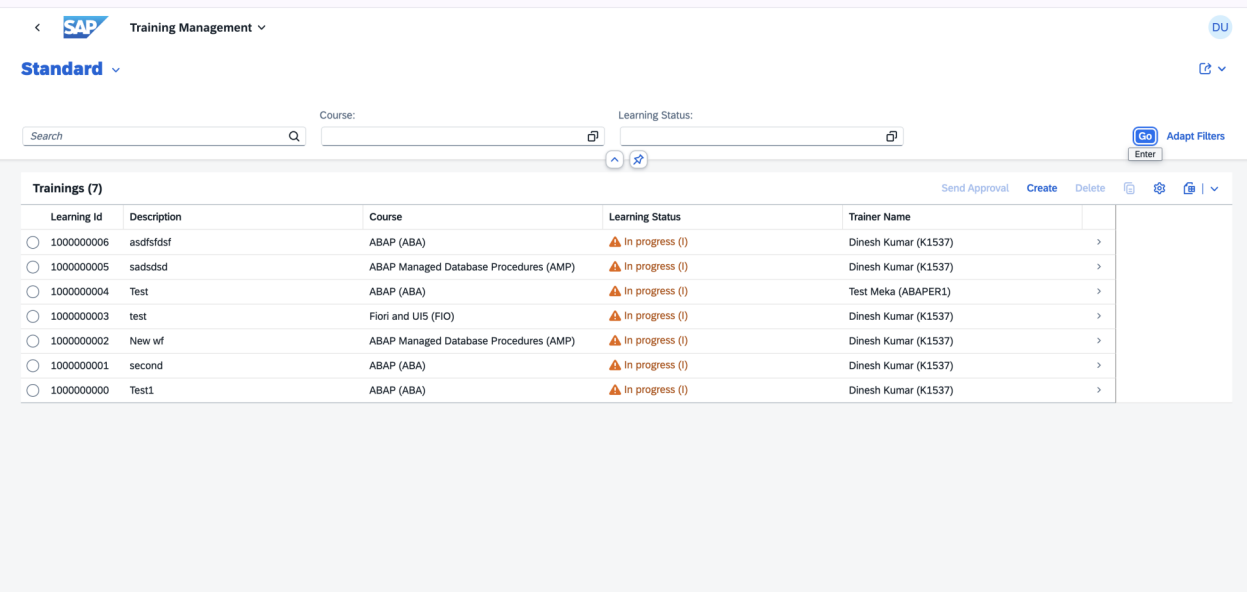
Let us Create a Fiori Application in Business Application Studio.
Open the BAS and Create a new Project template



Select the Following to Create a new Project with list template



Once the project is Created do a preview to check whether the application is working



5.2. Deploy Fiori Application

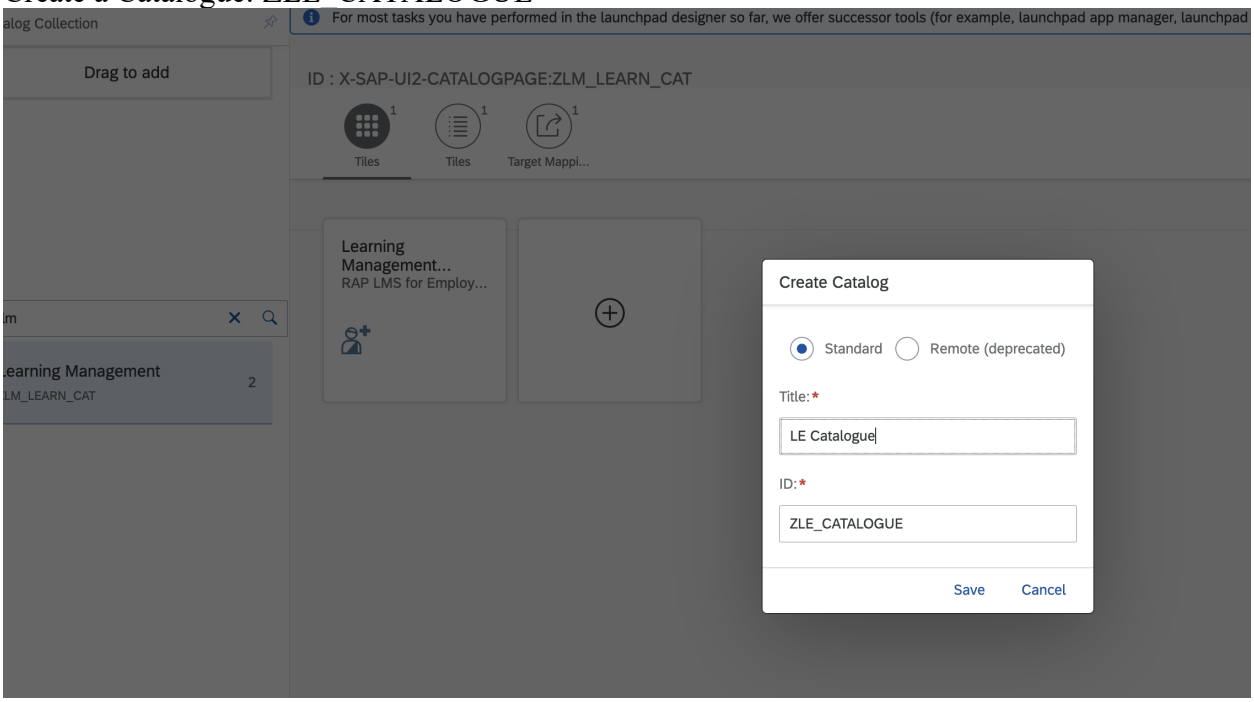
We need to do the following commands to deploy application to on premise system

```
user: zle_emp_training $ npx fiori add deploy-config
Adding deploy-config to the project.
warn Add:Deploy-config No VSCode Settings file found.
info Add:Deploy-config Using: @sap/fiori:deploy-config
? Please choose the target. ABAP
? Destination AK4 - [redacted]
? SAPUI5 ABAP Repository ZLE_EMP_TRAIN
? Deployment Description Employee Training
? Select How You Want to Enter the Package Enter Manually
? Package Z1537_TRAIN
? Select How You Want to Enter the Transport Request Enter Manually
? Transport Request AK4K901804
    force package.json
   create ui5-deploy.yaml

user: zle_emp_training $ npm run deploy

info abap-deploy-task ZLE_EMP_TRAIN Deployment Successful.
info abap-deploy-task ZLE_EMP_TRAIN (Note: As the destination is configure
```

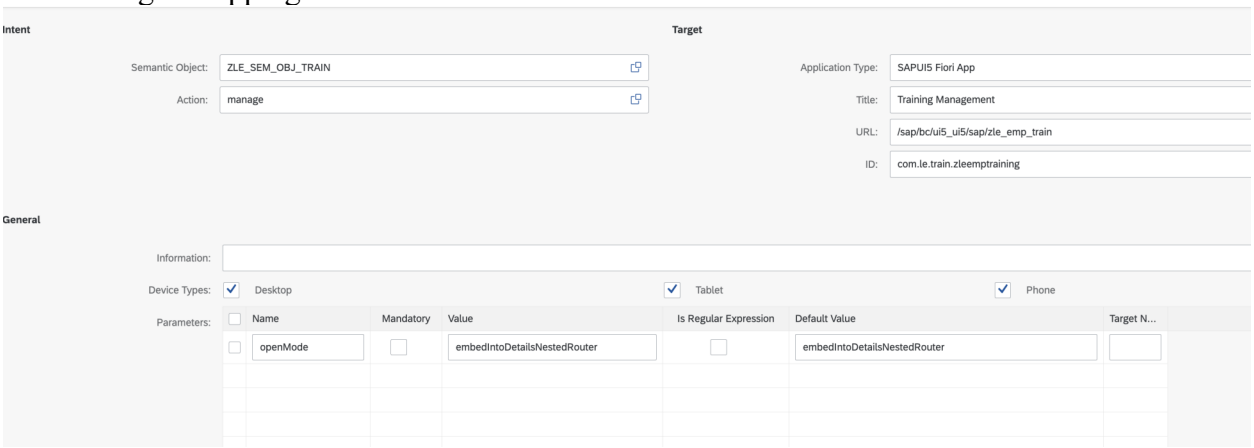
6. Add Fiori Application in Launchpad

6.1. Create Catalogue

Open Fiori Launchpad Customization: /n/ui2/flpd_cust
Create a Catalogue: ZLE_CATALOGUE

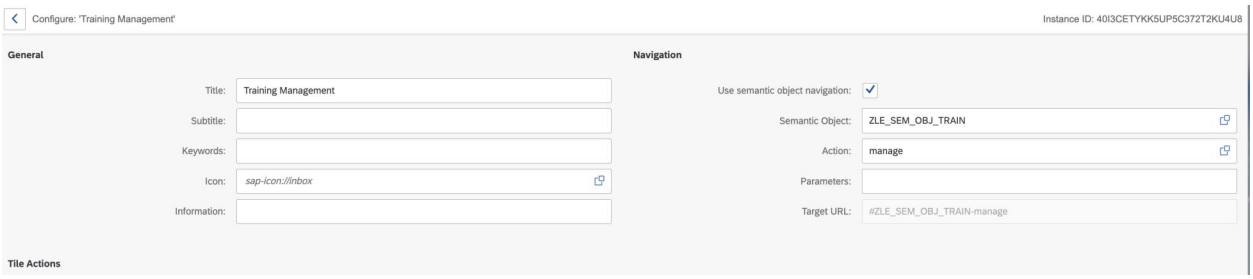


Create Target Mapping:



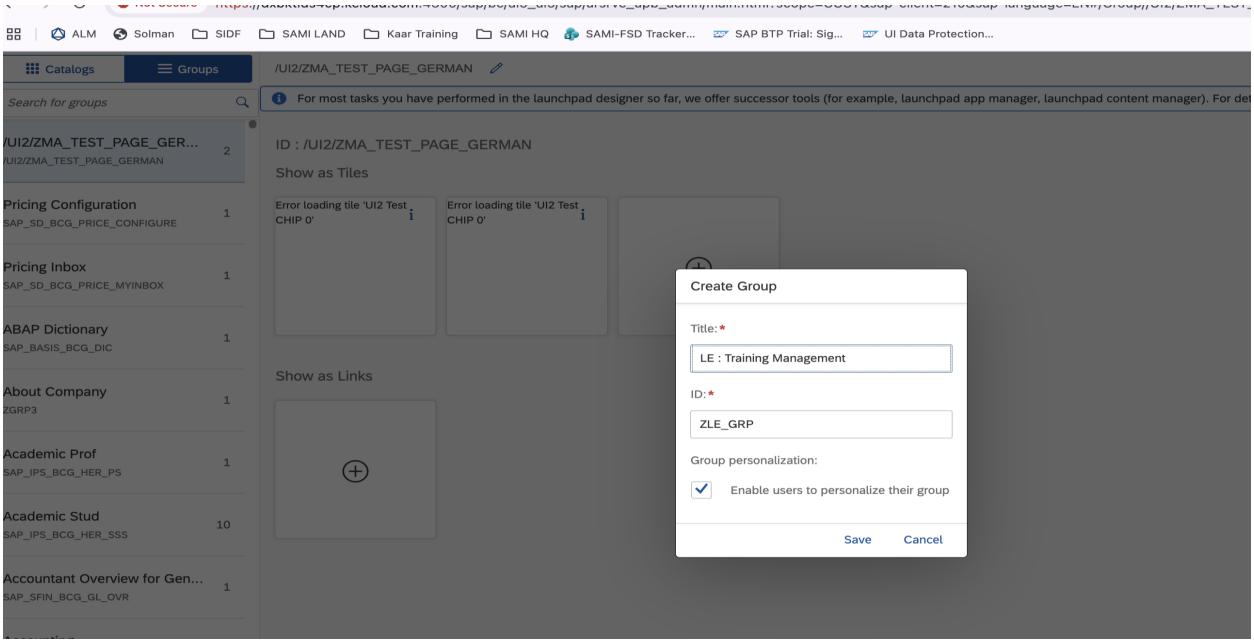
It is very important to give the Parameters as OpenMode and Value As embedIntoDetailsNestedRouter to enable the detail page of the application inside the common inbox details page in Fiori.

Create a New Tile for the application:

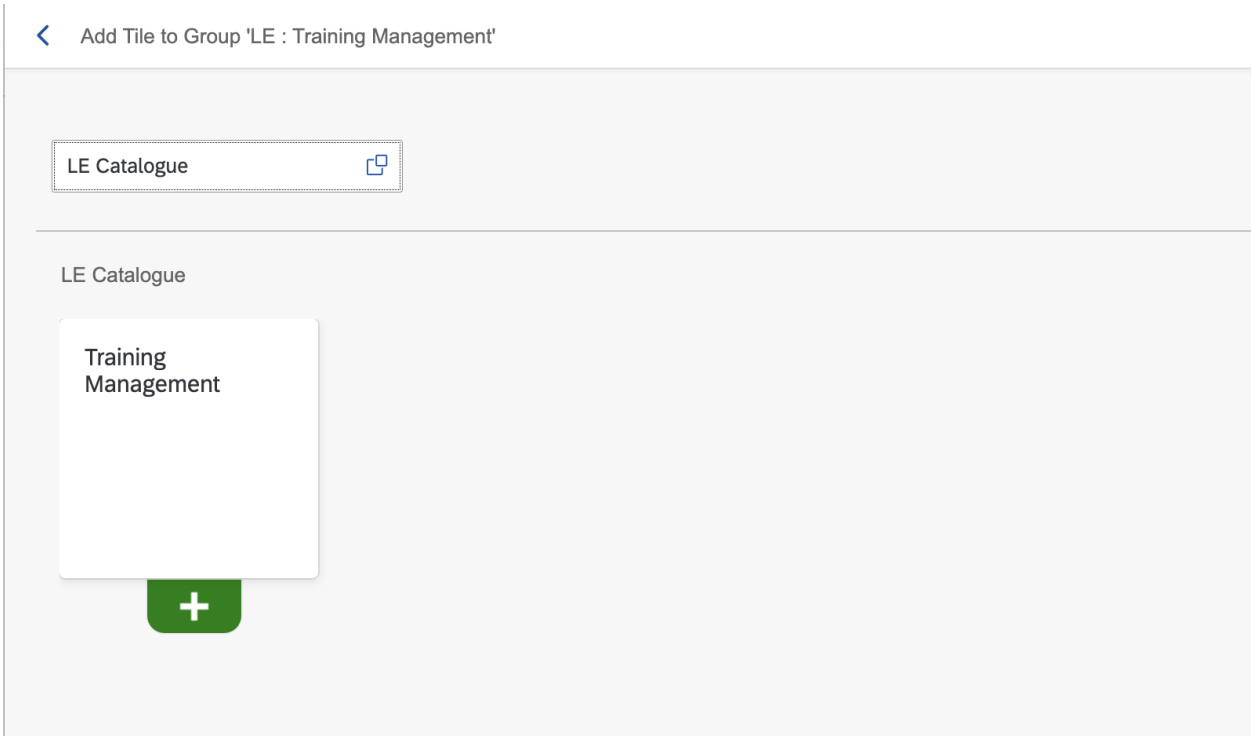


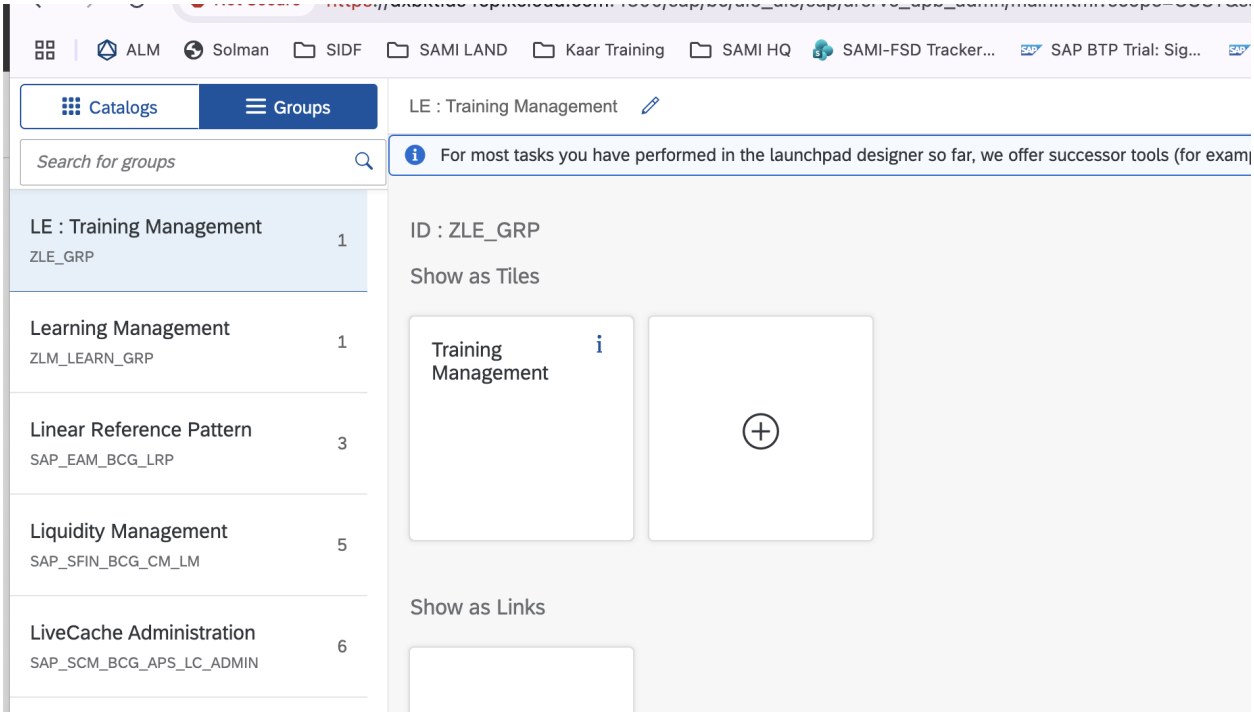
6.2. Create Group

Create a Group: ZLE_GRP (Newer way is to use SPACES and PAGES but here since it is a sample I go with older way of groups)



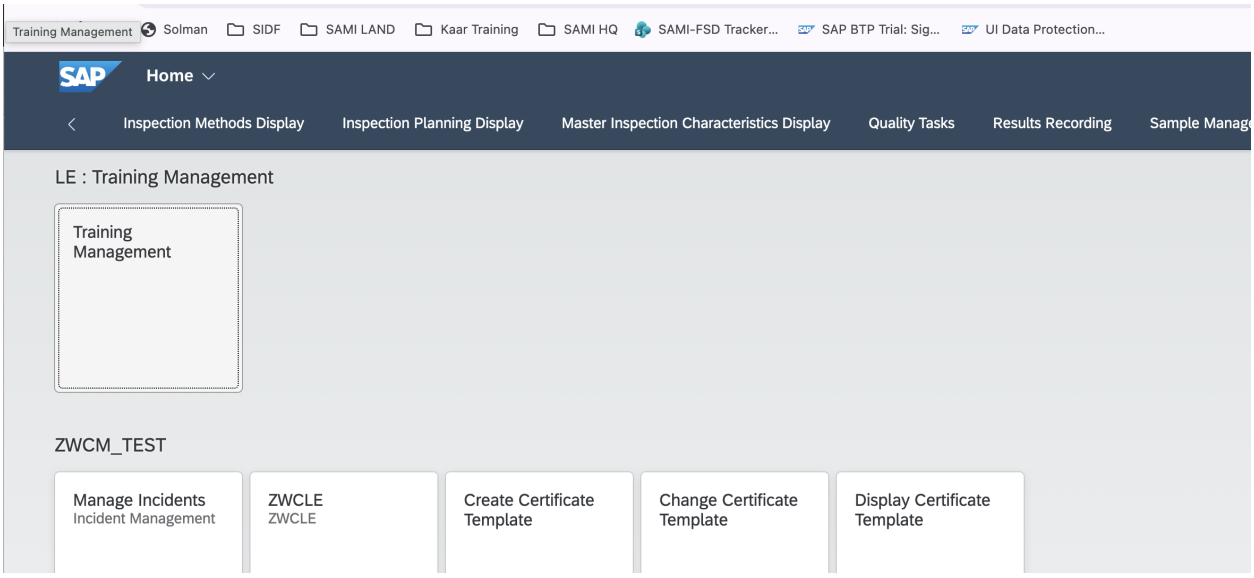
Select the group and inside select the catalog and add the tile

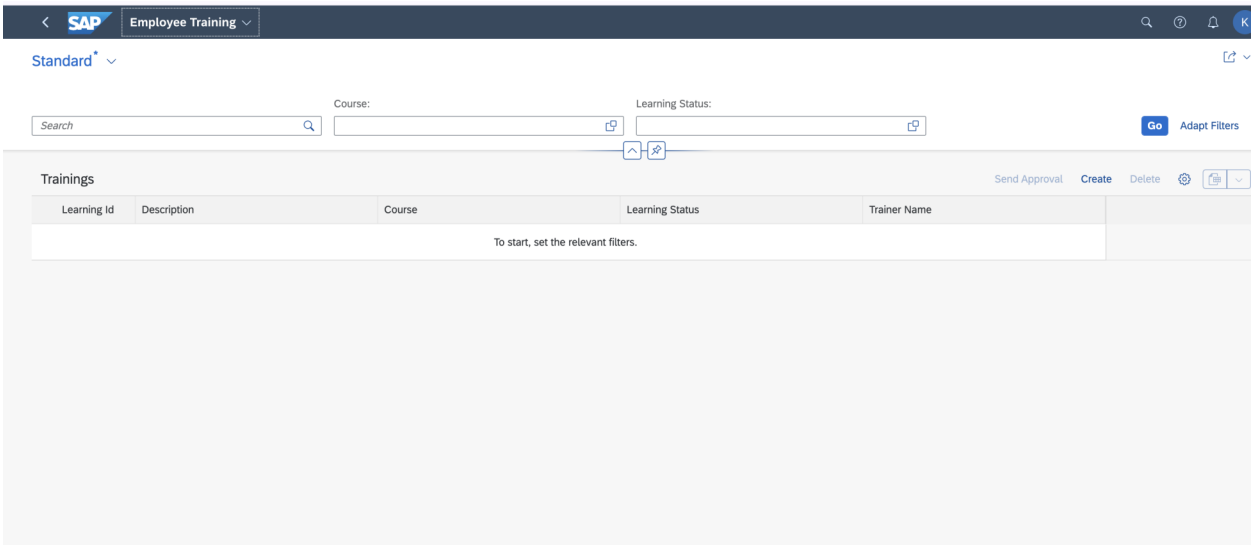




You can either create new role or assign this group and catalogue in existing role: Take support of basis to do this

Now when you log into the Launchpad you can access the application


Training Management
Solman
SIDF
SAMI LAND
Kaar Training
SAMI HQ
SAMI-FSD Tracker...
SAP BTP Trial: Sig...
UI Data Protection...
Home
Inspection Methods Display
Inspection Planning Display
Master Inspection Characteristics Display
Quality Tasks
Results Recording
LE : Training Management
Training Management
ZWCM_TEST
Manage Incidents
Incident Management
ZWCLE
ZWCLE
Create Certificate Template
Change Certificate Template
Display Certificate Template

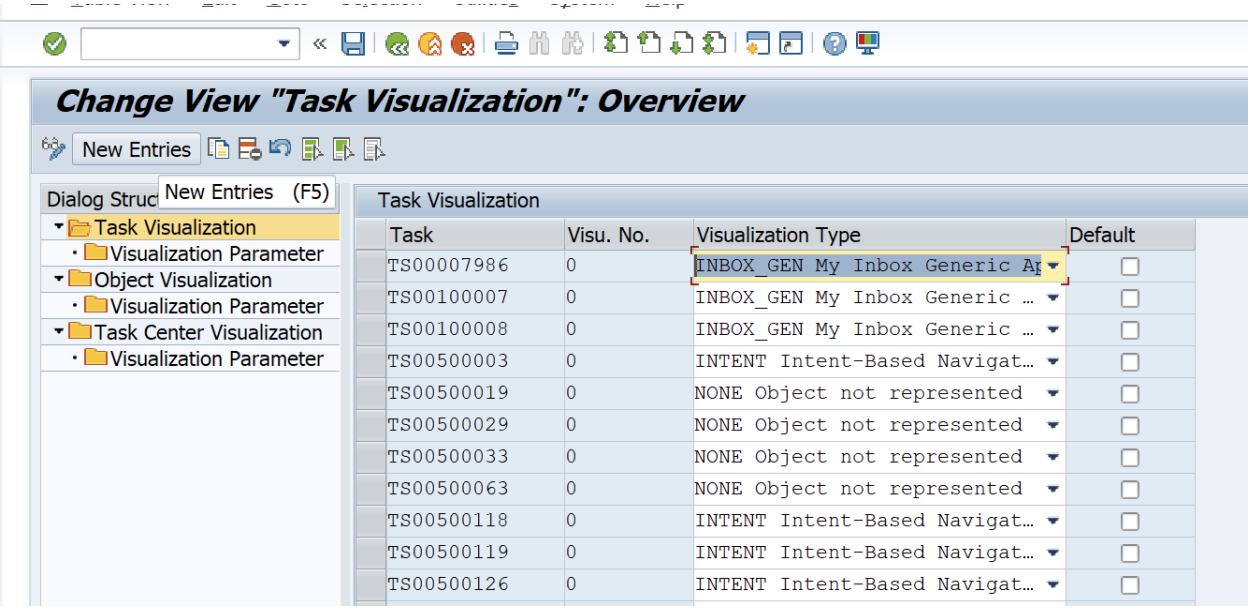

Employee Training
Standard
Course:
Learning Status:
Search
Go
Adapt Filters
Trainings
Send Approval
Create
Delete
Learning Id
Description
Course
Learning Status
Trainer Name
To start, set the relevant filters.

7. Visualization Configuration for Tasks

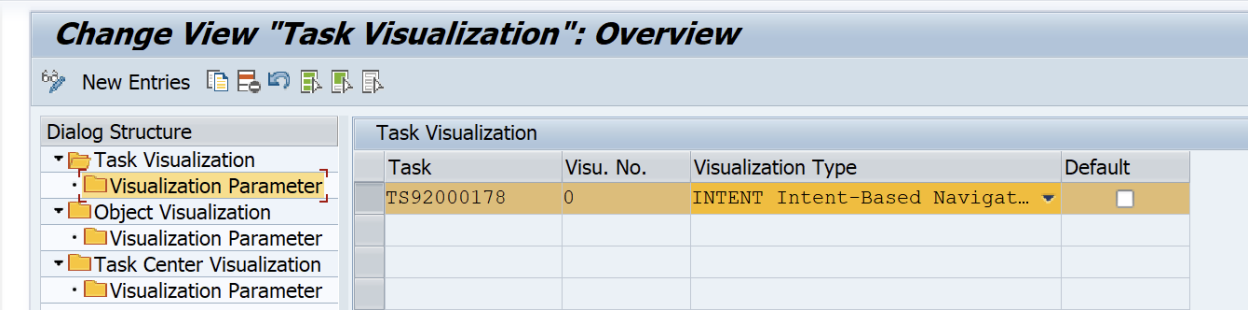
Configure Visualization parameter to enable the RAP Detail view inside Inbox Detail view.

Go to TCODE: SWFVISU

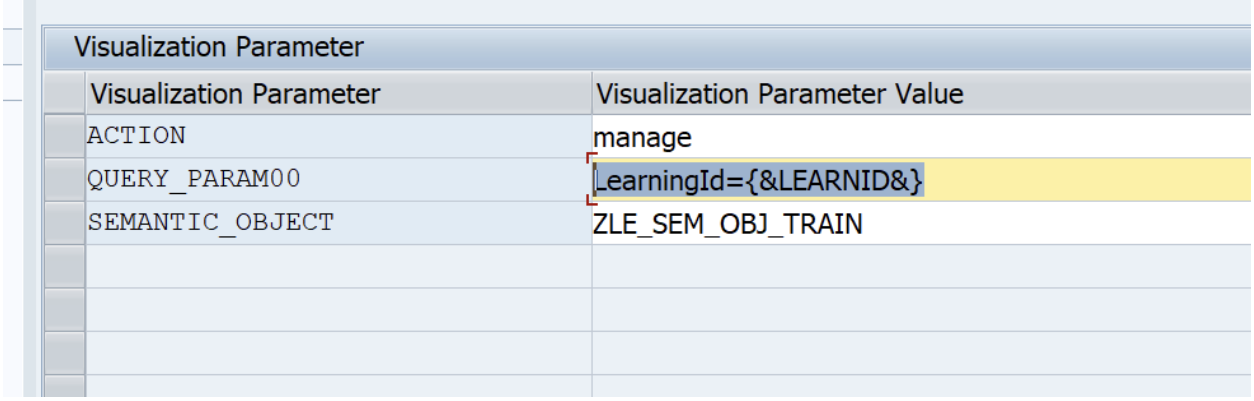
Click on create New Entries:



Give the approval task number that we created in PFTC and the following values



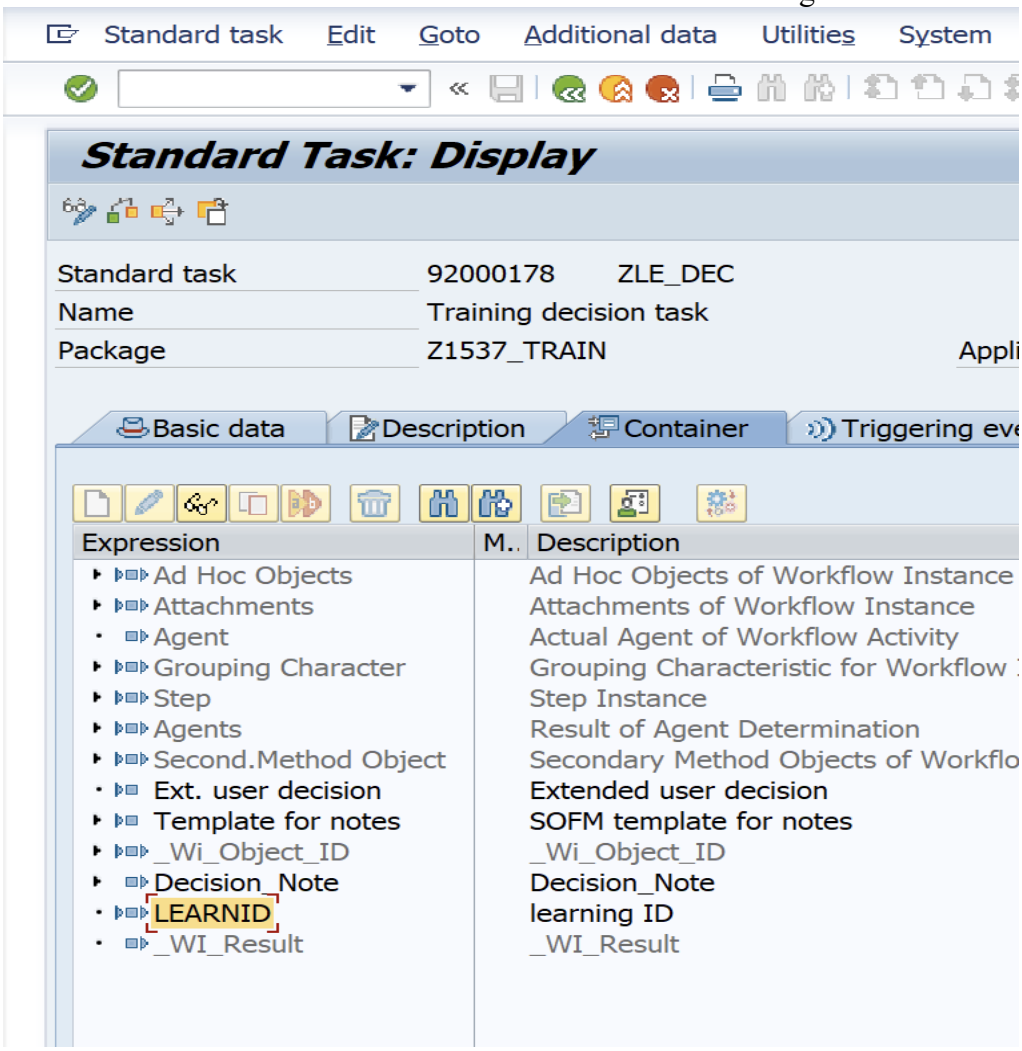
Select the Task number and double click visualization parameter



SEMANTIC_OBJECT and ACTION is what we created in the Target Mapping during the Fiori launchpad catalogue

QUERY_PARAM00 is the key field LearningId that we have in the Base Interface View and the Assigned values is the container parameter we created in the workflow task binding.

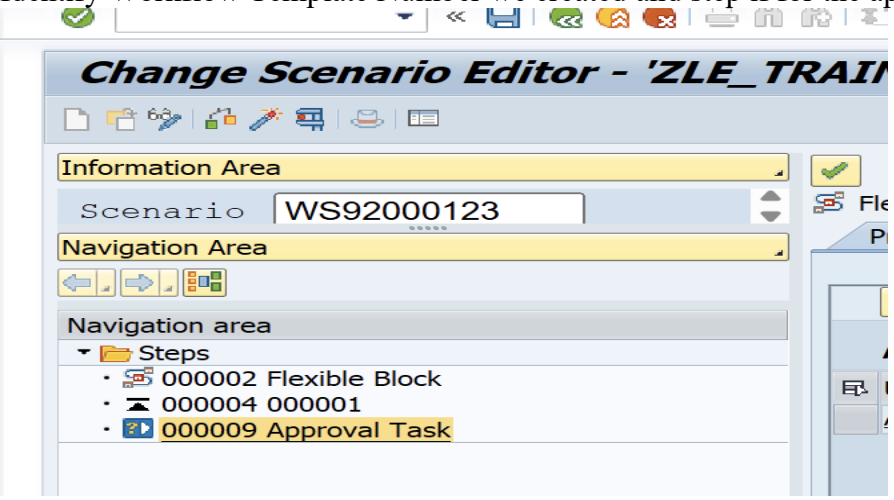
Note: Parameters here are case sensitive be careful when filling it.



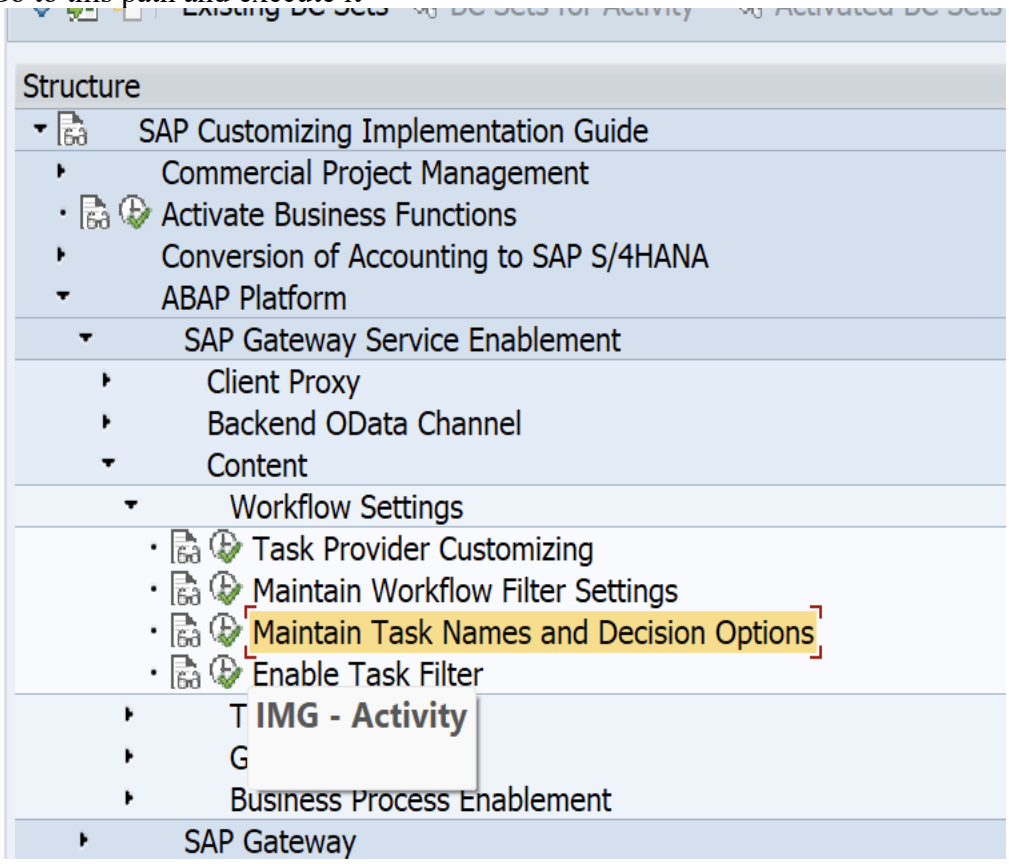
Save the values and close

8. Task Decision Button Configuration

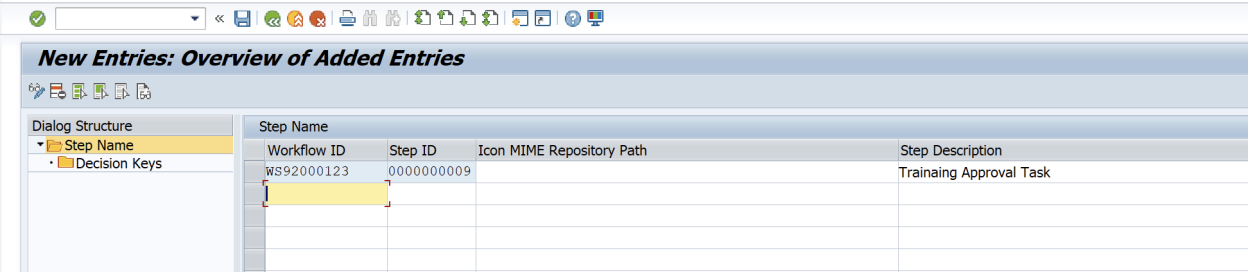
Configure Task Decision Buttons to show the Approve and Reject Button in approval task:
Identify Workflow Template Number we created and step if for the approval task.



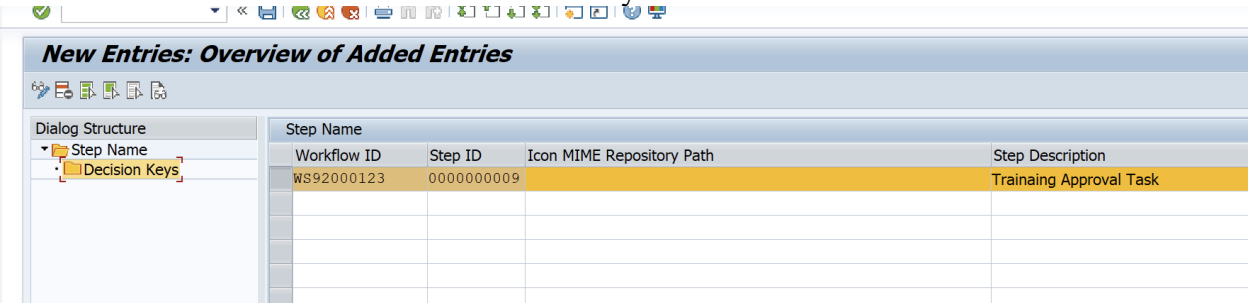
Go to TCODE SPRO and select SAP Reference IMG
Go to this path and execute it



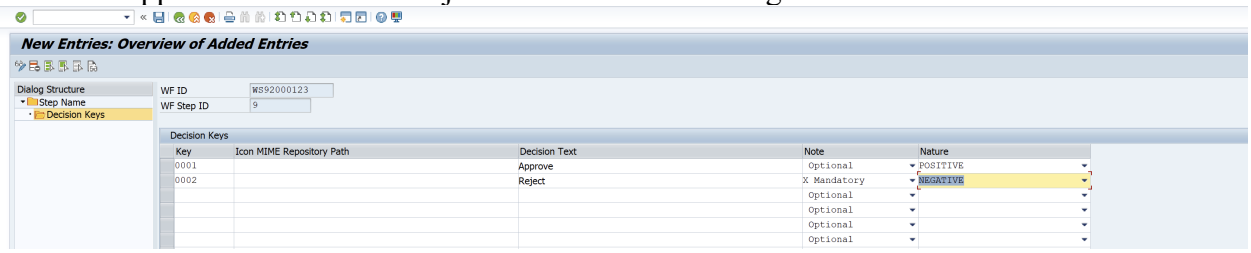
Give the following Workflow Id And step id, this step id we can take from SWDD_SCENARIO workflow template



Select this line and double click the Decision Keys:



Provide Button details:
0001 for Approve and 0002 for Reject with Positive and Negative nature



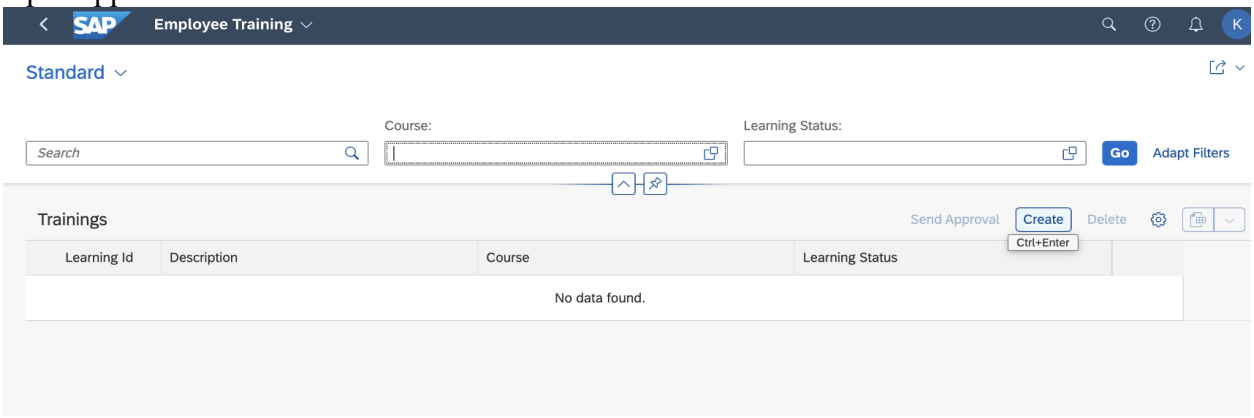
Save the entries and close the TCODE:

9. Testing the Application

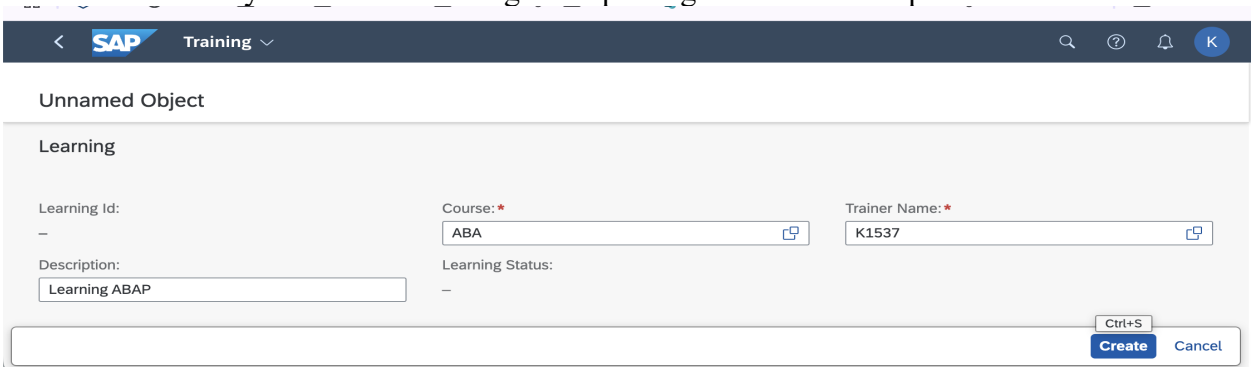
Now we are all done, let us check the Application

9.1. Approve Scenario

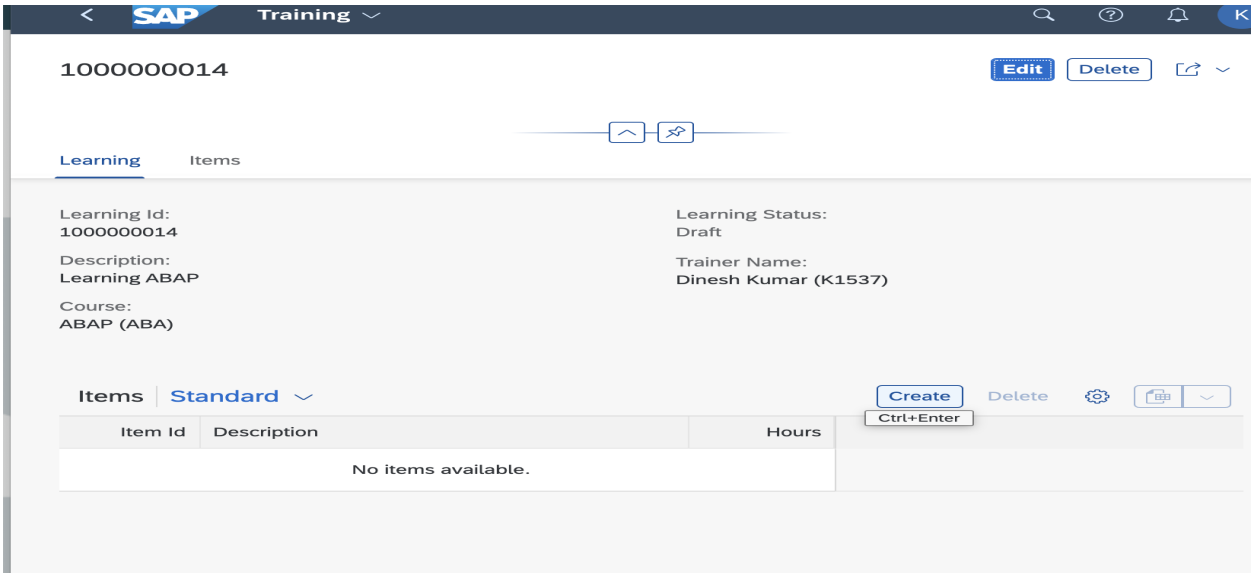
Open application and click on Create Button



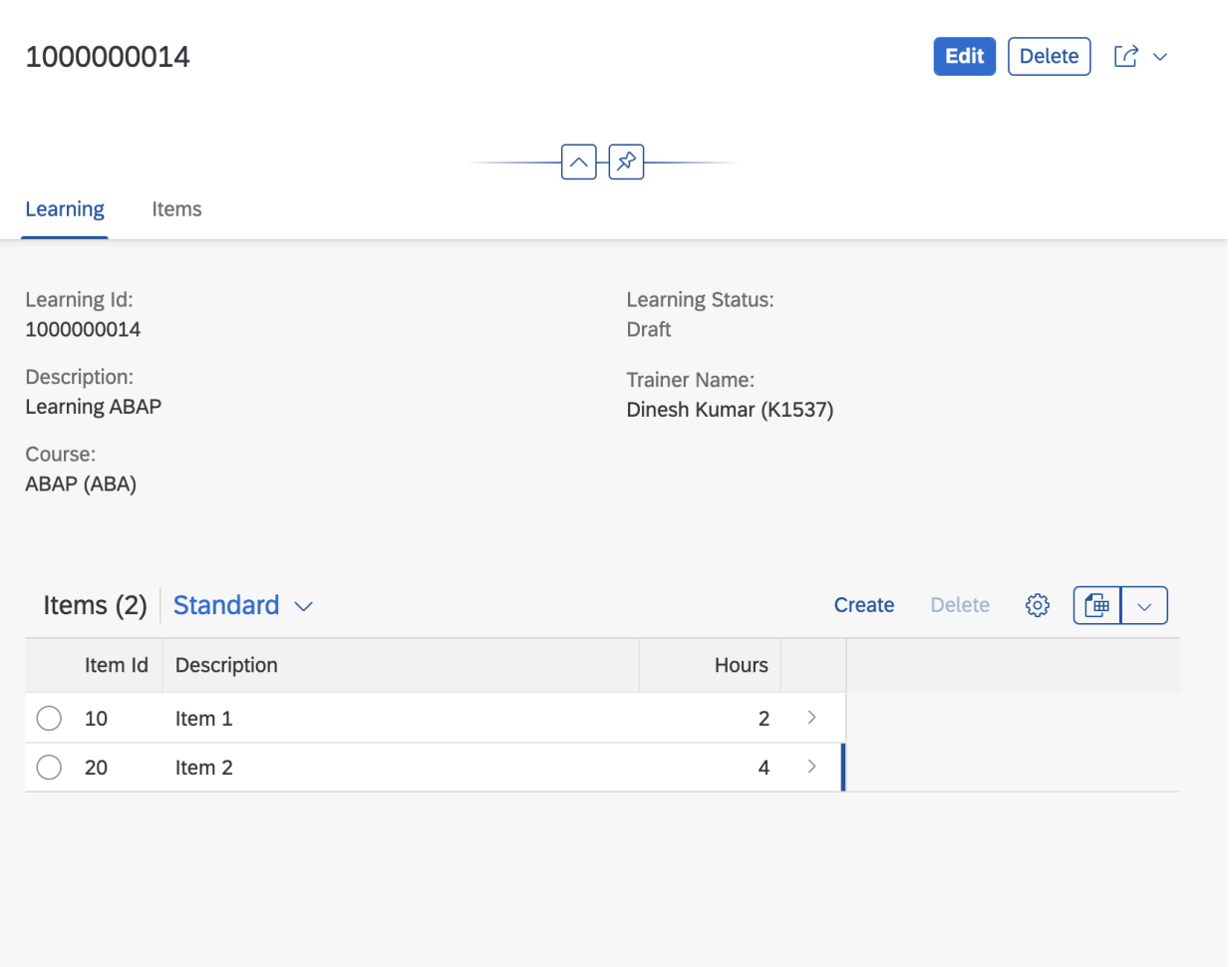
Select the necessary header values using f4 help and give the header inputs and click on create



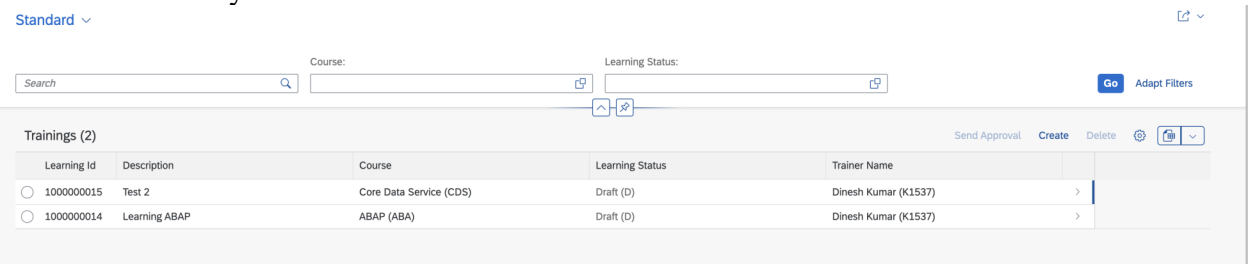
Create Items:



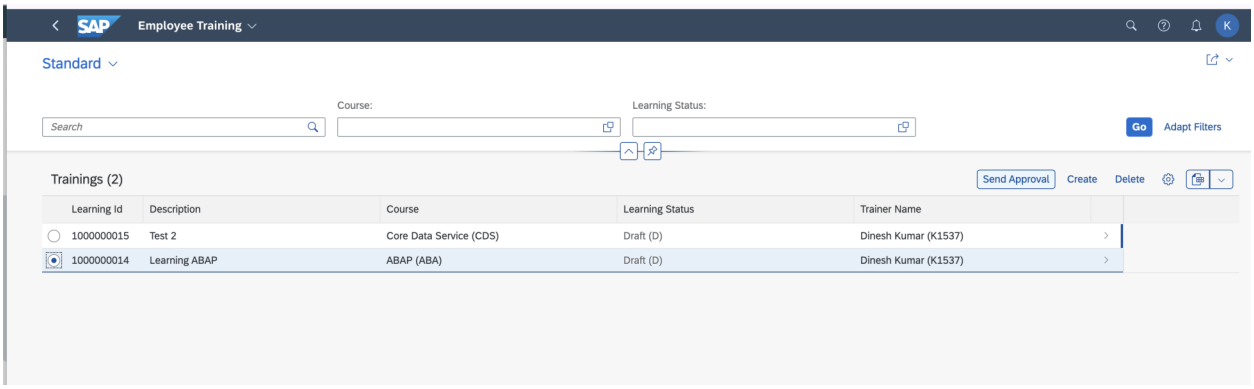
I have added two items



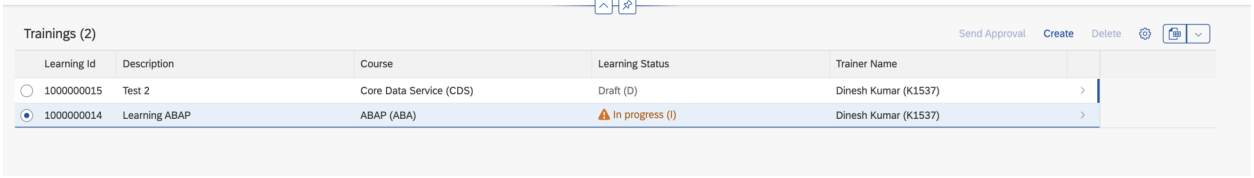
You can create any number of items in draft state



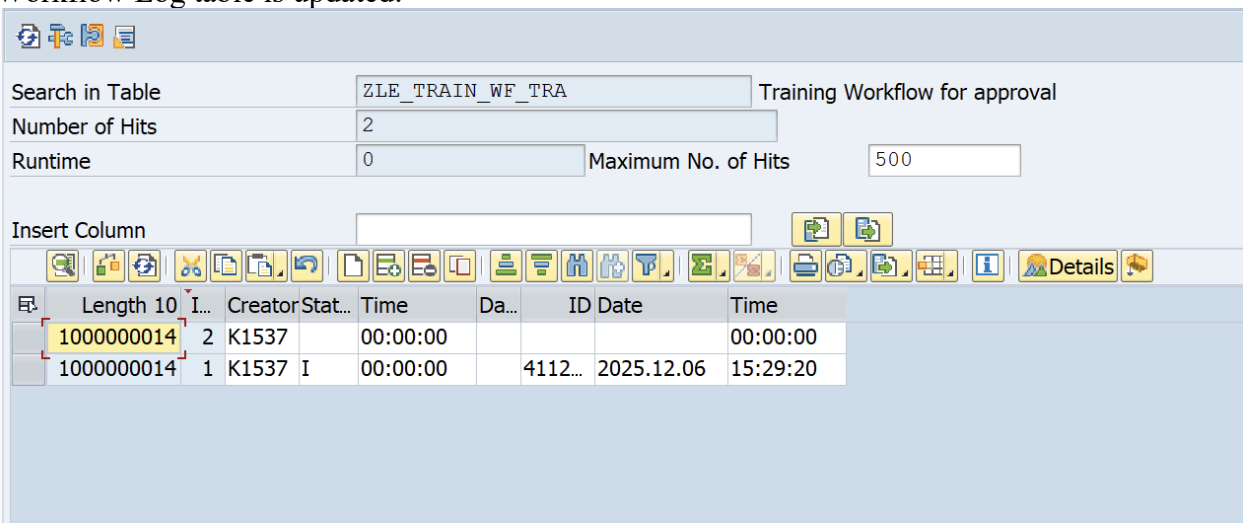
Now Select one to enable the send for approval



Now the status is changed as in progress

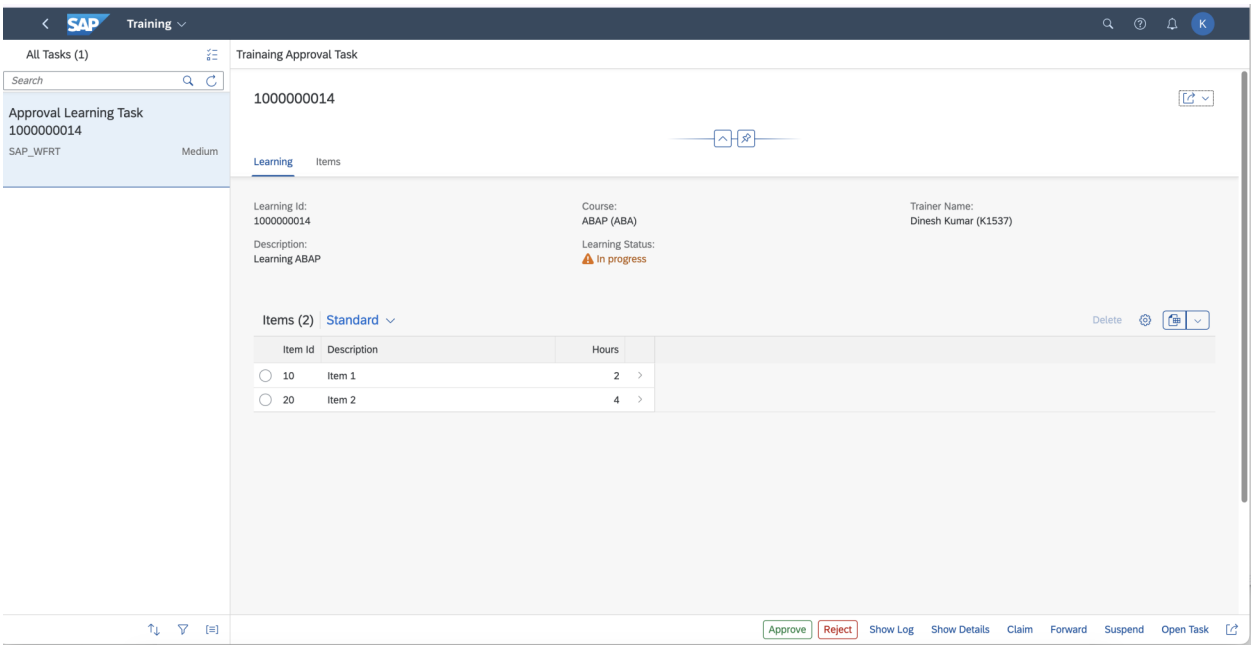


Workflow Log table is updated:

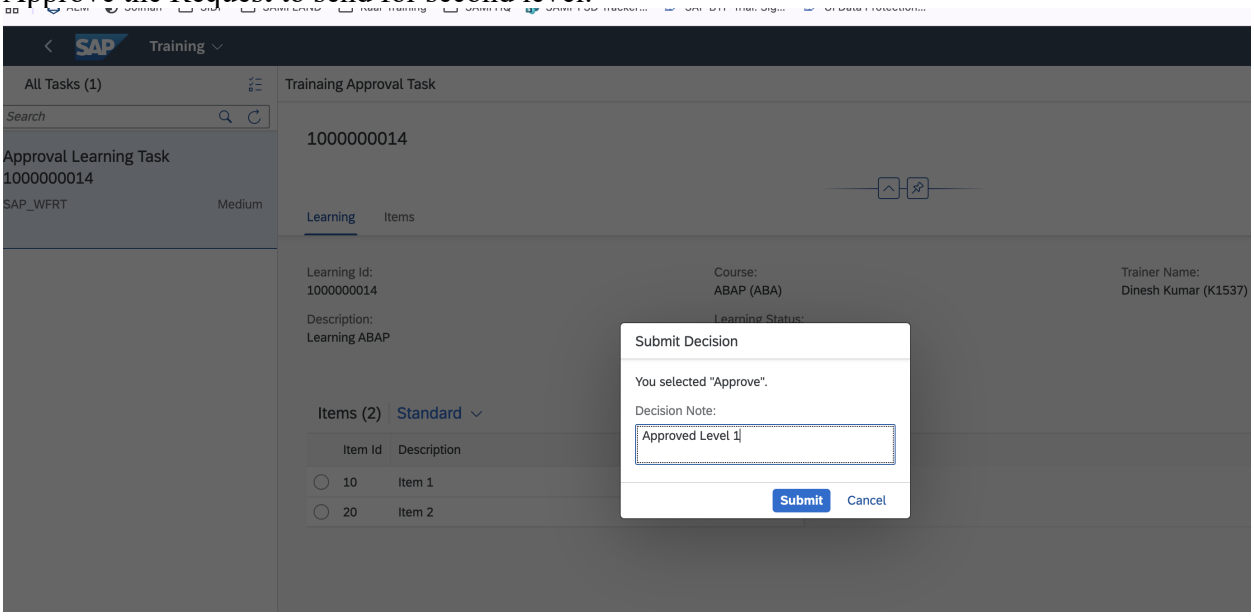


Email is sent for approver:

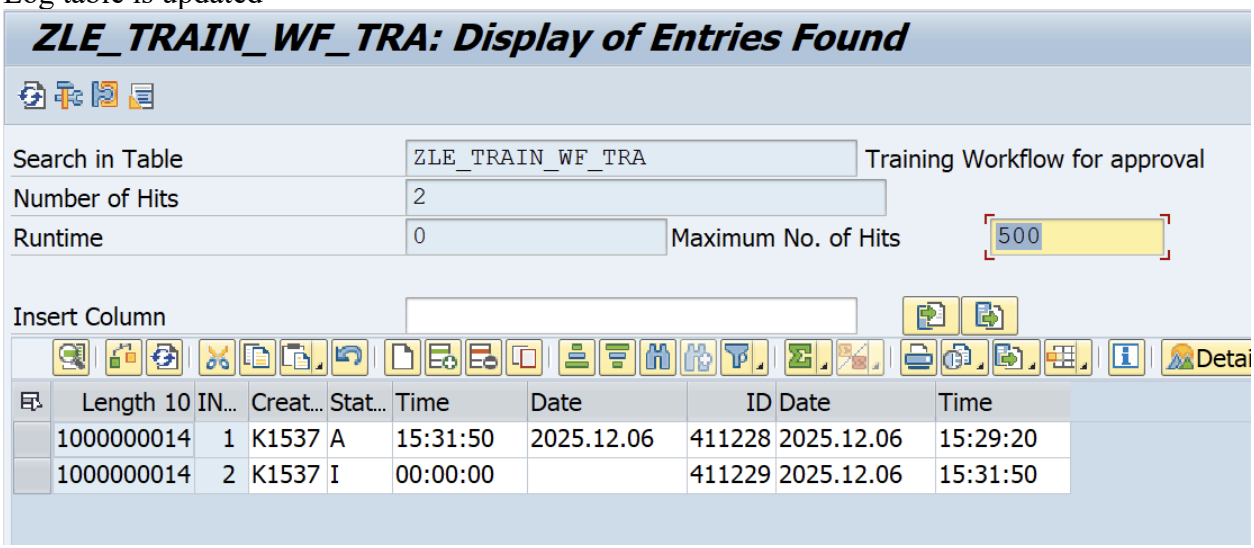
Now the Go to the user inbox where we can see the detail page of the RAP Application embedded for the selected Learning ID



Approve the Request to send for second level:

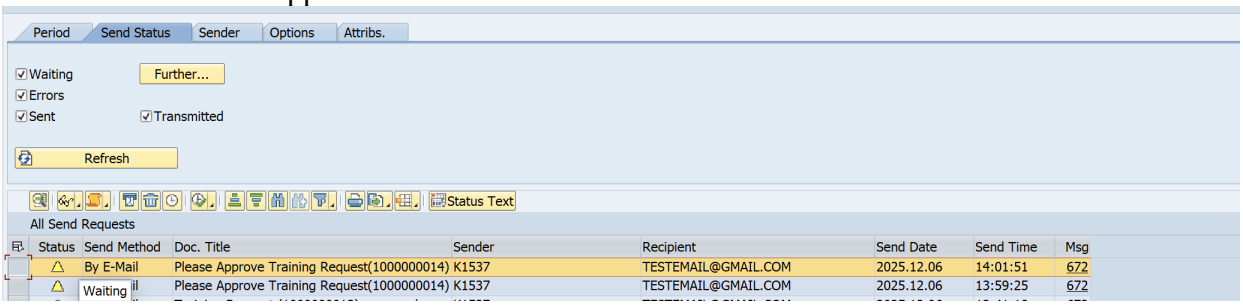


Log table is updated

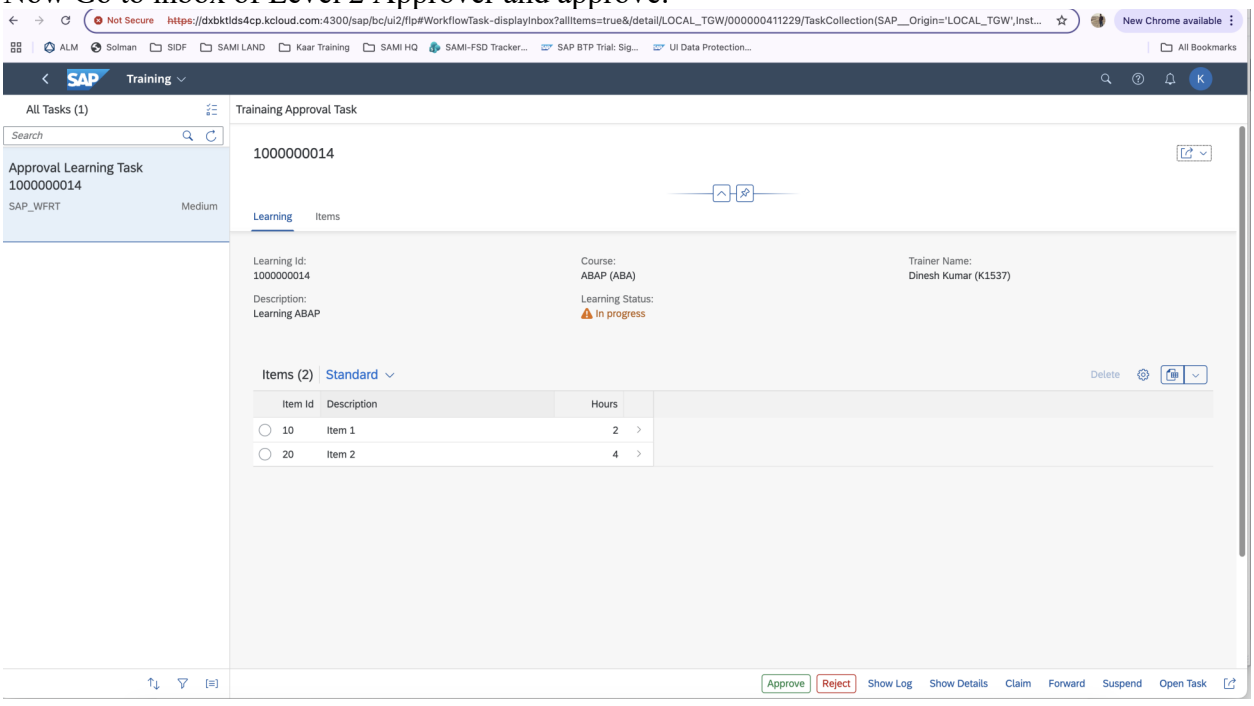


Length 10	IN...	Creat...	Stat...	Time	Date	ID	Date	Time
1000000014	1	K1537	A	15:31:50	2025.12.06	411228	2025.12.06	15:29:20
1000000014	2	K1537	I	00:00:00		411229	2025.12.06	15:31:50

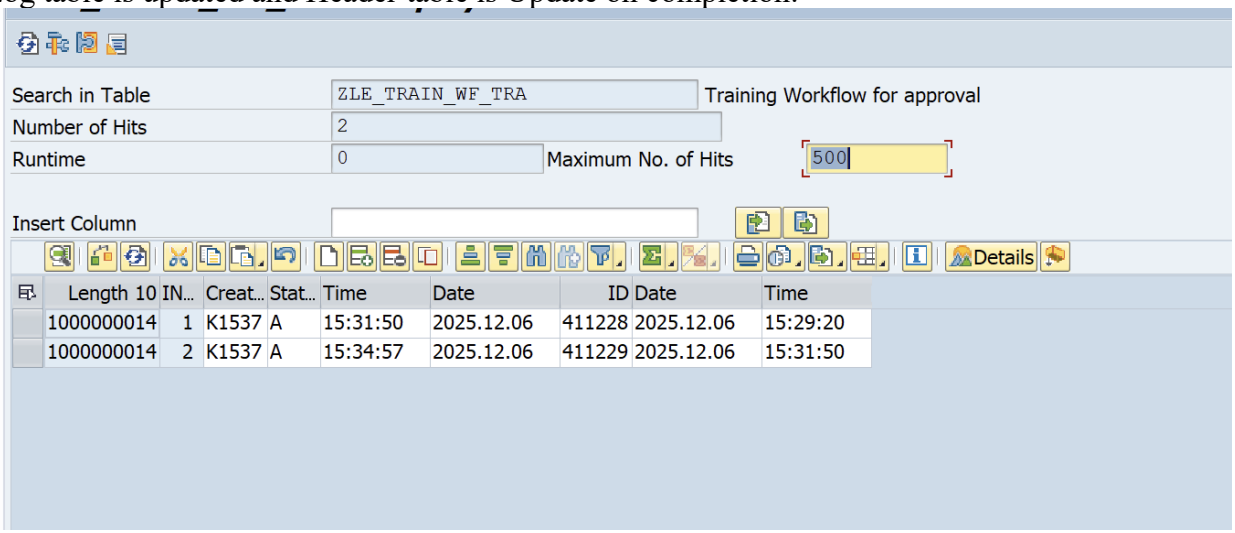
Email is sent for next approver:



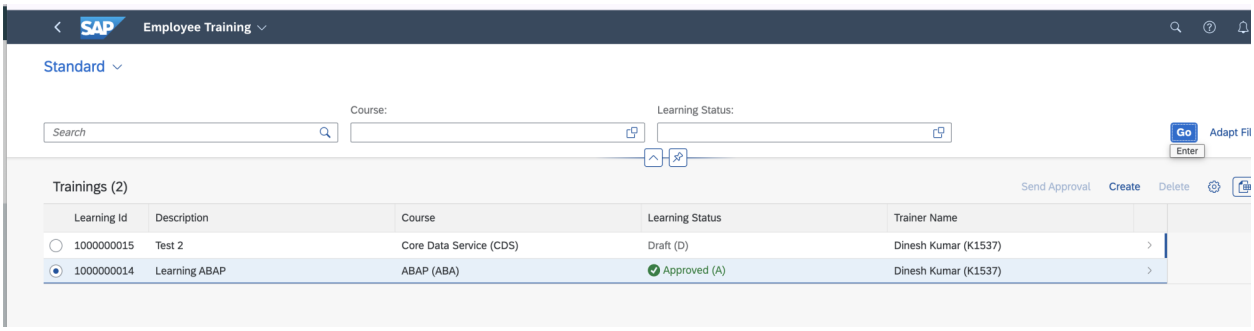
Now Go to inbox of Level 2 Approver and approve:



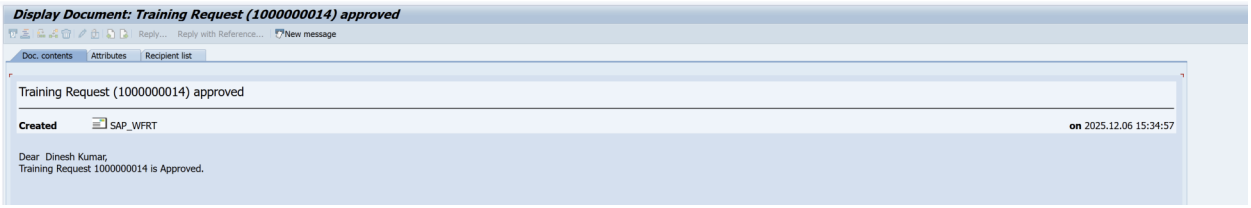
Log table is updated and Header table is Update on completion:



Length 10	IN...	Creat...	Stat...	Time	Date	ID	Date	Time
1000000014	1	K1537	A	15:31:50	2025.12.06	411228	2025.12.06	15:29:20
1000000014	2	K1537	A	15:34:57	2025.12.06	411229	2025.12.06	15:31:50



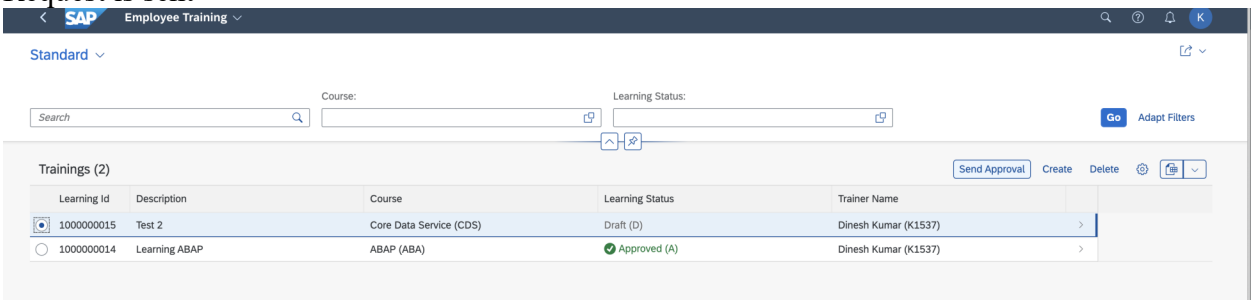
Email is sent for Initiator:



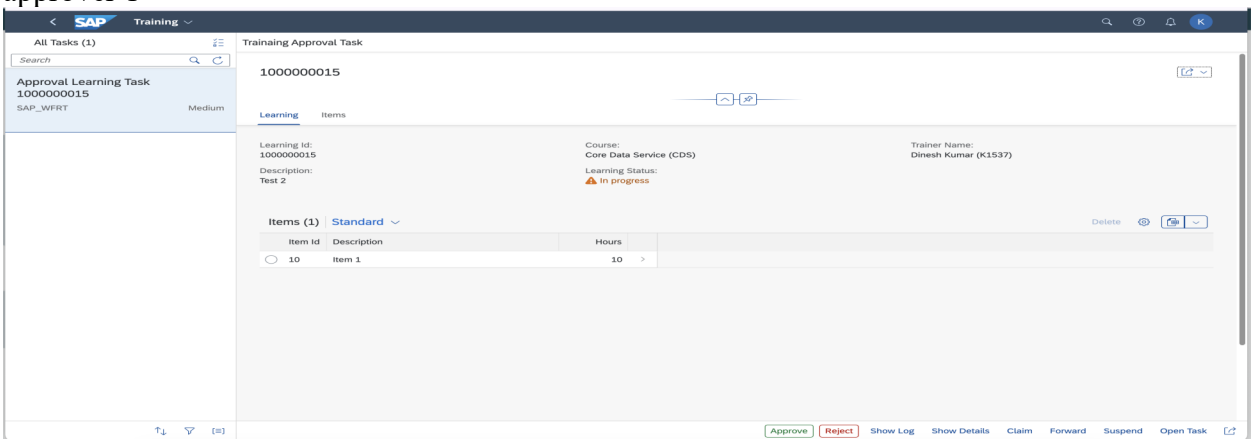
9.2. Reject Scenario

Let us send the other request and reject it in first level itself

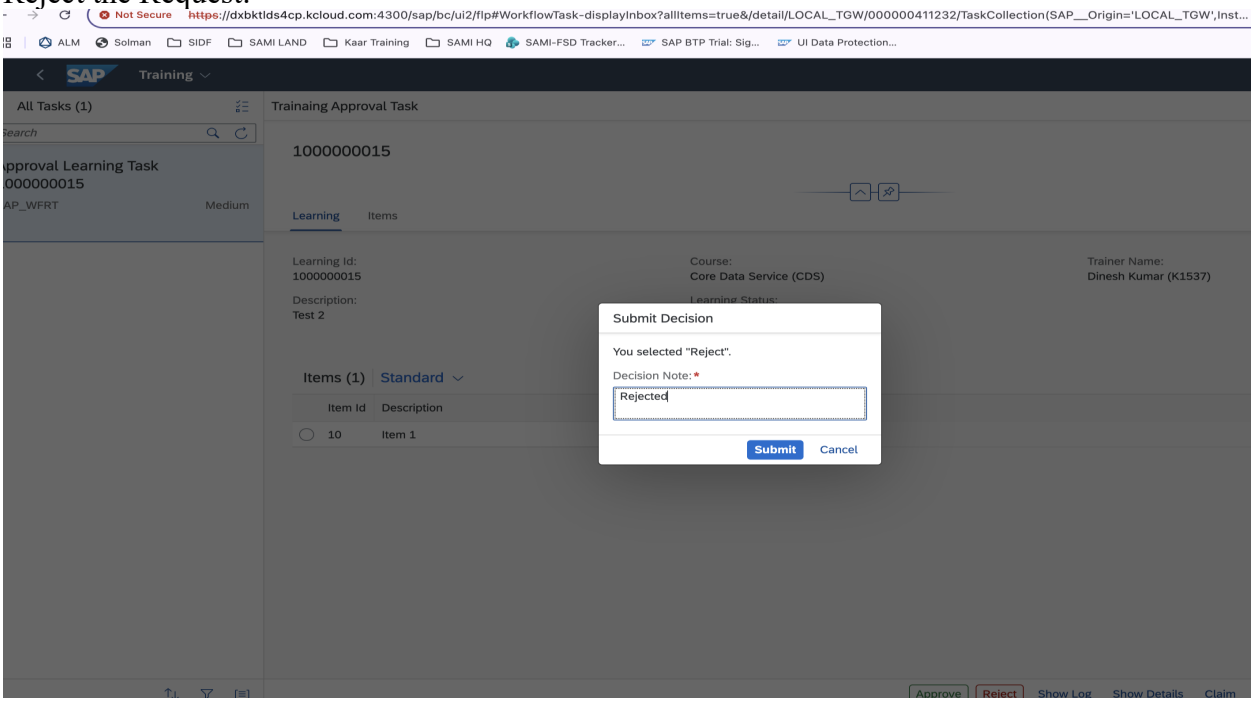
Request is sent



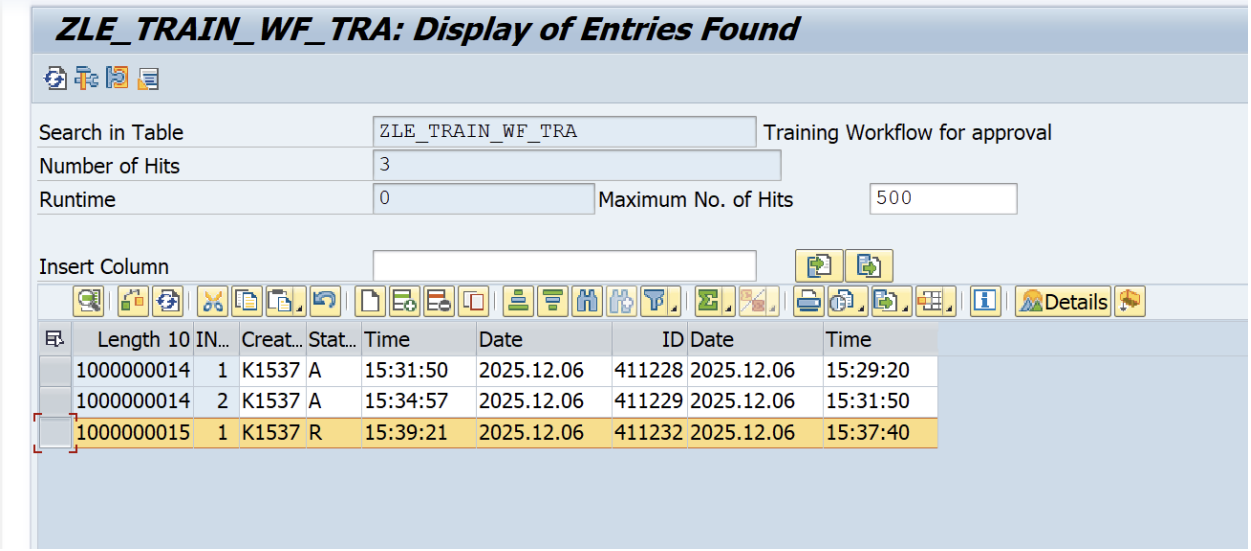
Same Workflow log Tables will be updated and also the Emails are sent. Now go to inbox of approver 1



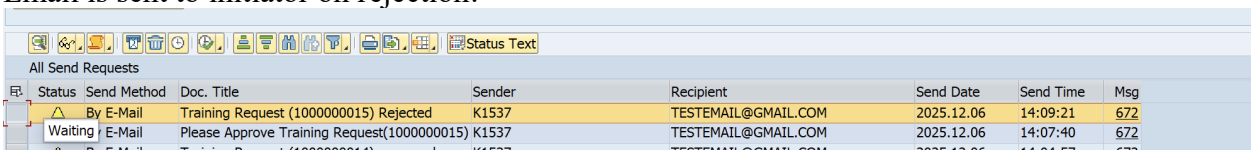
Reject the Request:



Now Workflow will be completed in level one itself and level 2 will be ignored



Email is sent to initiator on rejection:



Status is updated on table: