EXERCISE 01C - IMPLEMENTING LOGIC FOR A CUSTOM BUSINESS OBJECT

SAP Partner Workshop	**20 min**

Description

In this exercise, you'll learn how

- to implement logic to set some data from the backend only and to check all data of an instance.
- to ease development and test already while doing it.

Target group

- Developers
- People interested in learning about S/4HANA Cloud In-App extensions.

Goal

The goal of this exercise is to implement custom business object logic to control your application.

Prerequisites

Below are the prerequisites for this exercise.

- Google Chrome: Please complete this exercise using the Google Chrome browser
- **Authorizations:** Your user needs a business role with business catalog **Extensibility** (ID: `SAP_CORE_BC_EXT`)

Steps

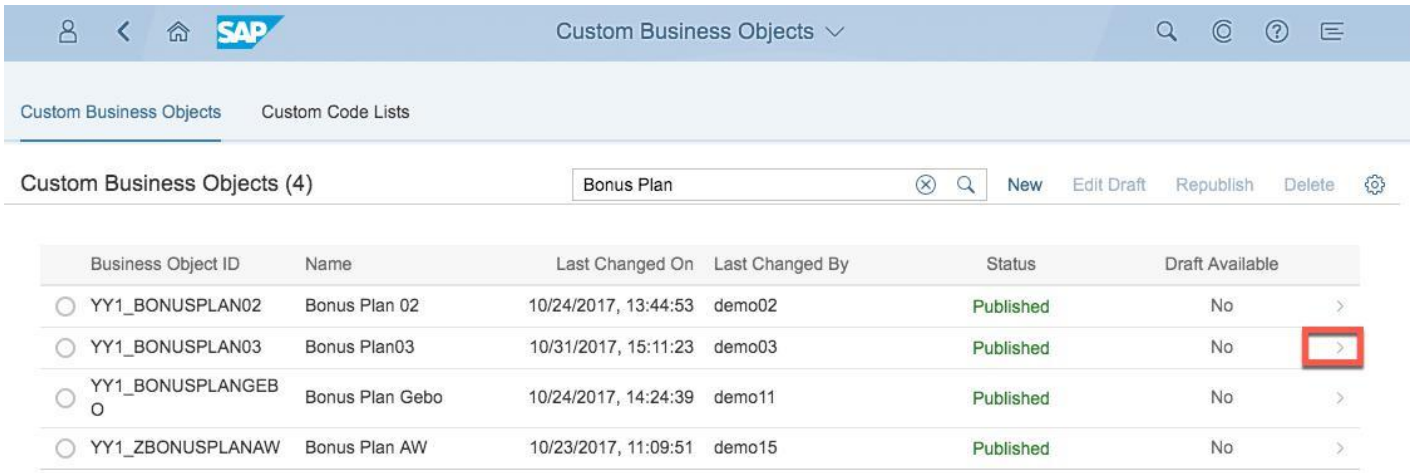
Make key field Read-Only

A several tutorials spanning example will show extensibility along custom Bonus Management applications.

In the first parts a Manager wants to define business objects "Bonus Plan" for employees. A Bonus Plan is there to save employee specific rules for bonus entitlement.

As there was no backend implementation to set the mandatory key field `ID` so far, we were forced to set it from the UI to be able to save instances. Now, as we will implement the logic to set the ID in backend and nowhere else, we will set that key field to Read-Only for the UI.

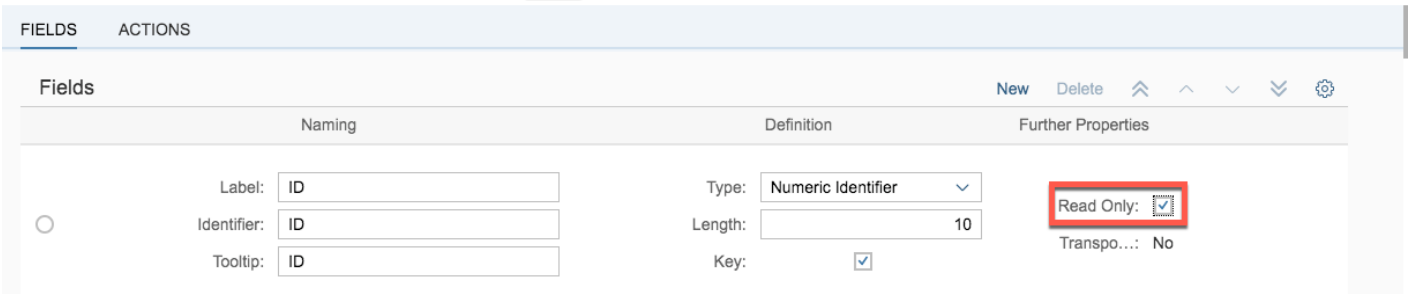
1. Open the business object **Bonus PlanXX** in **Custom Business Objects** application.



2. Start Edit Mode by executing the **Edit Draft** action.

3. **Go to Fields and Logic**.

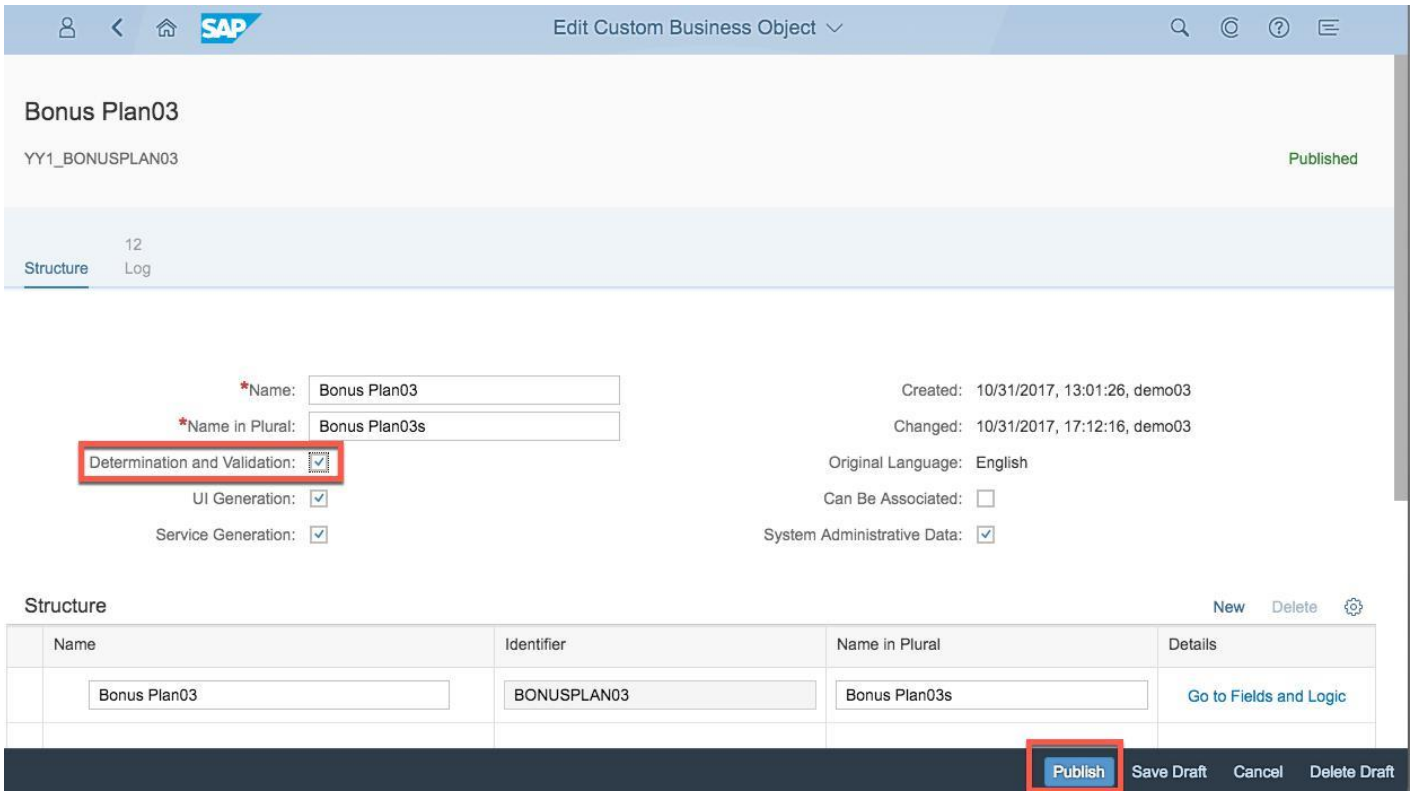
4. **Check** the Read-Only box for key field `ID`.



5. Go back via the application's **Back** button.

Enable logic implementation

1. Still editing the custom business object **Bonus Plan**'s definition, **Check** the box for **Determination and Validation**



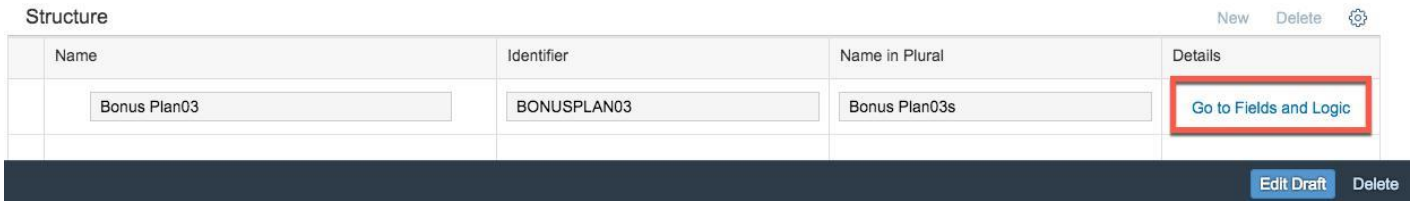
2. **Publish** the business object definition.

Now you are enabled to implement **determination logic** which is called **after each modification** to a Bonus Plan instance from the UI, as well as **validation logic** which is called **before each save** of an instance.

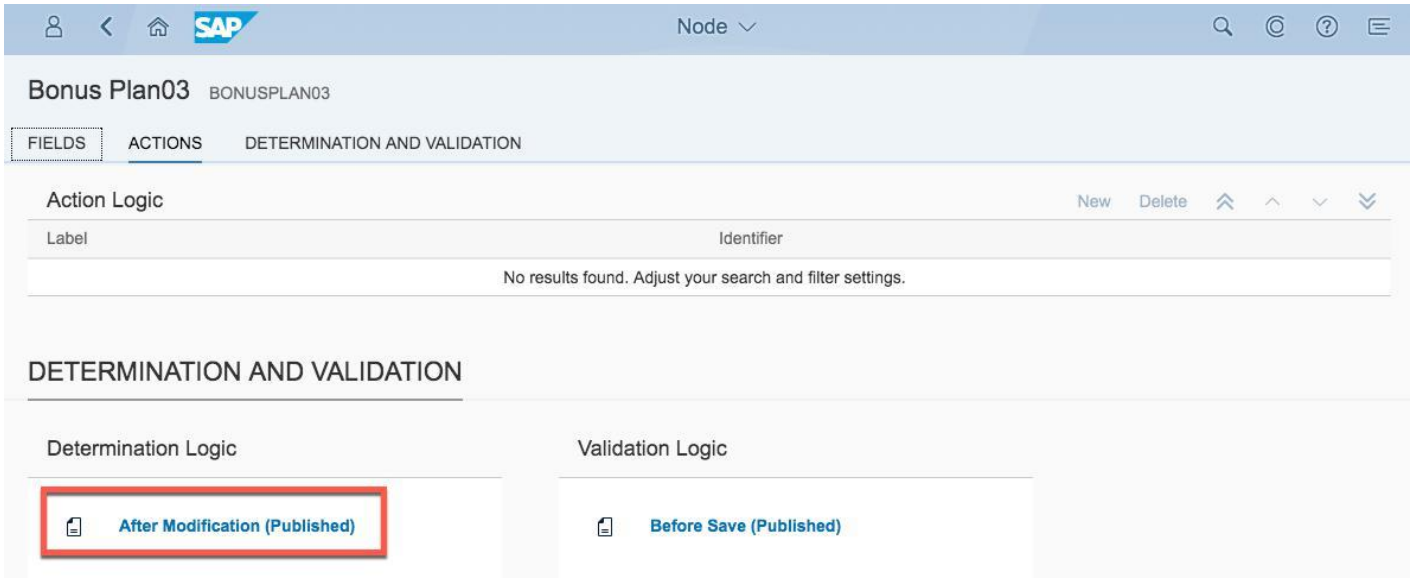
Start logic implementation

For **published** Custom Business Objects **without a Draft version** you can implement logic.

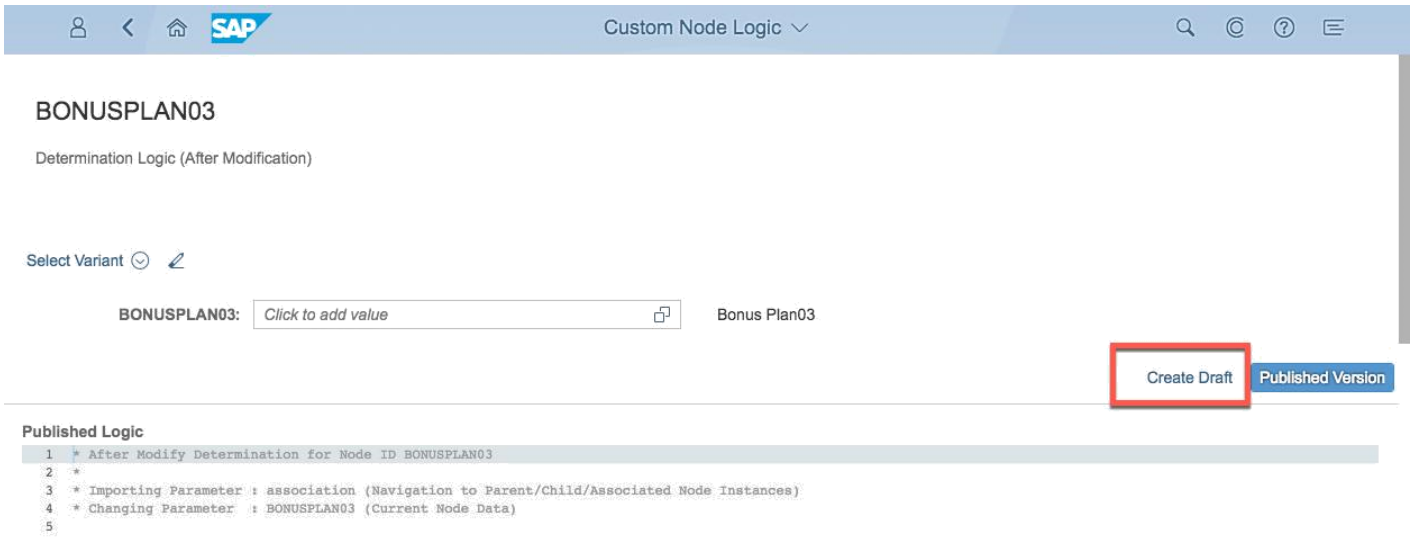
1. **Go to Fields and Logic**



2. Enter the After Modification Event Logic which is a Determination Logic.



3. In the logic view you initially see the not editable empty published version. Click the **Create Draft** action.



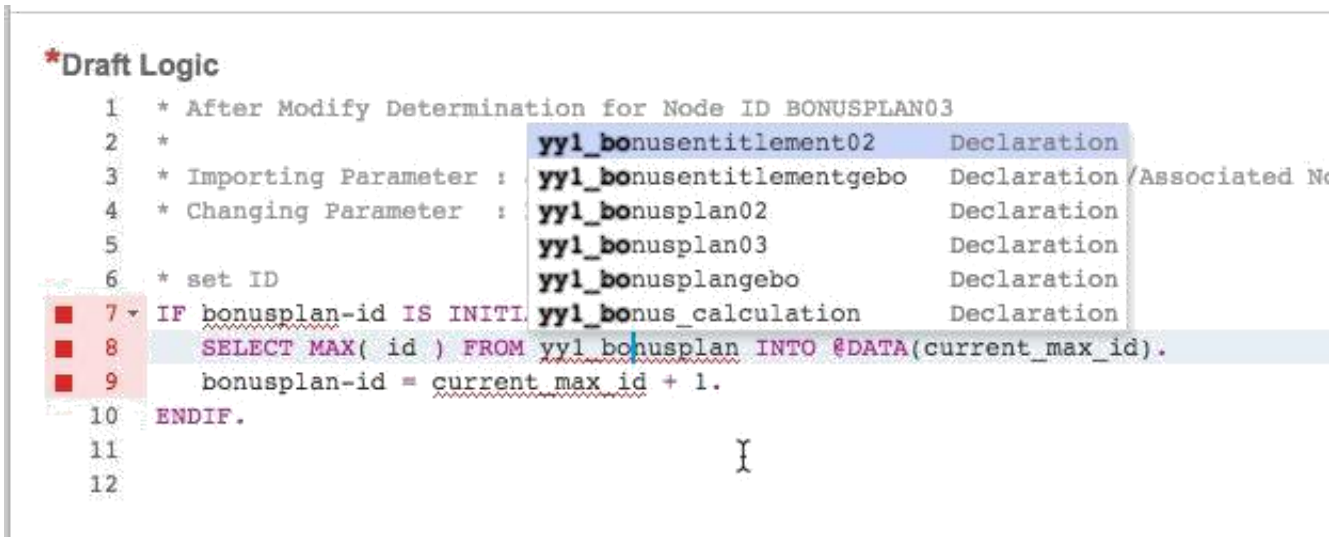
4. An editable copy of the published version appears left to it. With the **Draft Version** and **Published Version** actions you can decide what coding to see.

Implement After Modification: fix values

Implement After Modification event with following fix value functionality:

- Set the key field `ID` if still initial.

> **Hint:** Changing Parameter `bonusplan` enables you to read current node data and change it. **Hint:** You can read existing Bonus Plan data via the CDS View that is named as the Business Object's Identifier (here: `YY1_BONUSPLAN`). **Hint:** With the key combination **CTRL + Space** you can access the very helpful code completion.



Note: Replace XX to the number assigned to you.

```
* set ID
IF bonusplanXX-id IS INITIAL.
   SELECT MAX( id ) FROM yy1_bonusplanXX INTO @DATA(current_max_id).
   bonusplanXX-id = current_max_id + 1.
ENDIF.
```

- Set the Unit of Measure for the Bonus Percentages to `P1` which is the code for % (percent)

```
* set percentage unit
bonusplanXX-lowbonuspercentage_u = bonusplanXX-highbonuspercentage_u = 'P1'.
```

- Determine and set the Employee Name from the Employee ID >**Hint:** Extensibility offers Helper class `CL_ABAP_CONTEXT_INFO` with method `GET_USER_FORMATTED_NAME` that needs a user ID to return its formatted name

```
* set Employee Name
IF bonusplanXX-employeeid IS NOT INITIAL.
   bonusplanXX-employeename = cl_abap_context_info=>get_user_formatted_name( bonuspla
nXX-employeeid ).
ENDIF.
```

Implement After Modification: consistency check

In dependence on following checks, set the `isconsistent` property.

- Check that `ValidityStartDate` and `ValidityEndDate` are set and that

`ValidityStartDate` is earlier in time than `ValidityEndDate`.

- Check that Factors and Percentages are set correctly (all > 0, Percentages < 100, `LowBonusAssignmentFactor` < `HighBonusAssignmentFactor`)
- Check that Employee ID is set

```
* consistency check START
IF bonusplanXX-validitystartdate IS INITIAL
 OR bonusplanXX-validityenddate IS INITIAL
 OR bonusplanXX-validitystartdate GE bonusplanXX-validityenddate
 OR bonusplanXX-lowbonusassignmentfactor IS INITIAL
 OR bonusplanXX-highbonusassignmentfactor IS INITIAL
 OR bonusplanXX-lowbonuspercentage_v IS INITIAL
 OR bonusplanXX-highbonuspercentage_v IS INITIAL
 OR bonusplanXX-lowbonuspercentage_v GE 100 OR
 bonusplanXX-highbonuspercentage_v GE 100
 OR bonusplanXX-lowbonusassignmentfactor GE bonusplanXX-
 highbonusassignmentfactor OR bonusplanXX-employeeid IS INITIAL
 OR bonusplanXX-targetamount_v IS INITIAL
 OR bonusplanXX-targetamount_c IS INITIAL.
    bonusplanXX-isconsistent = abap_false.
ELSE.
    bonusplanXX-isconsistent = abap_true.
ENDIF.
* consistency check END
```

Test the logic during development

On top of the coding you can maintain runtime data for the current node structure which represents the data before running the test functionality. This data can also be saved as variant for later usages.

1. Click the value help to add test data

2. Enter following data

Field Name	Field Value
validitystartdate	2017-01-01
validityenddate	2017-12-31
targetamount_v	1000
targetamount_c	EUR
lowbonusassignmentfactor	1
highbonusassignmentfactor	3
lowbonuspercentage	10
highbonuspercentage	20
employeeid	<any>

employeeid <any> shall be the one of a sales person that created sales orders with a Net Amount of more than 3000.00 EUR in 2016 and that are completed. In this exercise, you can use CB9980000620.

This will look as follows.

Maintain Node Data: bonusplan03

Field	Value	Type
id		num(10)
validitystartdate	2017-01-01	date
validityenddate	2017-12-31	date
targetamount_v	1000	decimal(15,2)
targetamount_c	EUR	char(5)
lowbonusassignmentfactor	1	decimal(5,2)
highbonusassignmentfactor	3	decimal(5,2)
lowbonuspercentage_v	10	decimal(15,3)
lowbonuspercentage_u		char(3)
highbonuspercentage_v		decimal(15,3)

OK Cancel

Maintain Node Data: bonusplan03

Field	Value	Type
lowbonuspercentage_u		char(3)
highbonuspercentage_v		decimal(15,3)
highbonuspercentage_u	20	char(3)
isconsistent		char(1)
employeeid	CB9980000620	char(12)
employeename		char(40)
sap_createddatetime		decimal(21,7)
sap_createdbyuser		char(12)
sap_lastchangeddatetime		decimal(21,7)
sap_lastchangedbyuser		char(12)

OK Cancel

Click OK.

3. Execute the **Test** action and you can see the node data after your logic was executed.

BONUSPLAN03

Determination Logic (After Modification)

Select Variant

BONUSPLAN03: 2017-01-01 2017-12-31 1000 EUR 1 3 10 20 CB9980000620 Bonus Plan03

Draft Version | Published Version

Draft Logic

```
    bonusplan03-id = current_max_id + 1.
ENDIF.

* set percentage unit
bonusplan03-lowbonuspercentage_u = bonusplan03-highbonuspercentage_u = 'P1'.

* set Employee Name
IF bonusplan03-employeeid IS NOT INITIAL.
    bonusplan03-employeename = cl_abap_context_info=>get_user_formatted_name( bonusplan03-employeeid ).
ENDIF.

* consistency check START
IF bonusplan03-validitystartdate IS INITIAL
 OR bonusplan03-validityenddate IS INITIAL
 OR bonusplan03-validitystartdate GE bonusplan03-validityenddate
 OR bonusplan03-lowbonusassignmentfactor IS INITIAL
 OR bonusplan03-highbonusassignmentfactor IS INITIAL
 OR bonusplan03-lowbonuspercentage_v IS INITIAL
 OR bonusplan03-highbonuspercentage_v IS INITIAL
 OR bonusplan03-lowbonuspercentage_v GE 100
 OR bonusplan03-highbonuspercentage_v GE 100
 OR bonusplan03-lowbonusassignmentfactor GE bonusplan03-highbonusassignmentfactor
 OR bonusplan03-employeeid IS INITIAL
 OR bonusplan03-targetamount_v IS INITIAL
 OR bonusplan03-targetamount_c IS INITIAL.
    bonusplan03-isconsistent = abap_false.
ELSE.
    bonusplan03-isconsistent = abap_true.
ENDIF.
* consistency check END
```

Draft logic tested

Published Logic

```
* After Modify Determination for Node ID BONUSPI
*
* Importing Parameter : association (Navigation
* Changing Parameter  : BONUSPLAN03 (Current Noc
```

Published logic tested

Test Results: Draft Logic

Parameter	Value	Type
BONUSPLAN03	[structure]	
id	0000000002	num(10)
validitystartdate	2017-01-01	date
validityenddate	2017-12-31	date
targetamount_v	1000	decimal(15,2)
targetamount_c	EUR	char(5)
lowbonusassignmentfactor	1	decimal(5,2)
highbonusassignmentfactor	3	decimal(5,2)
lowbonuspercentage_v	10	decimal(15,3)
lowbonuspercentage_u	P1	char(3)

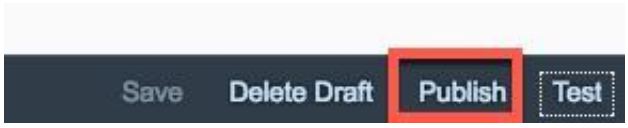
Test Results: Published Logic

Parameter	Value	Type
BONUSPLAN03	[structure]	

Save | Delete Draft | Publish | Test

You can see that your logic works as `id` , `*percentage_u` fields and `employename` are filled and `isconsistent` is 'X'.

4. **Publish** the After Modification.



5. Go back.

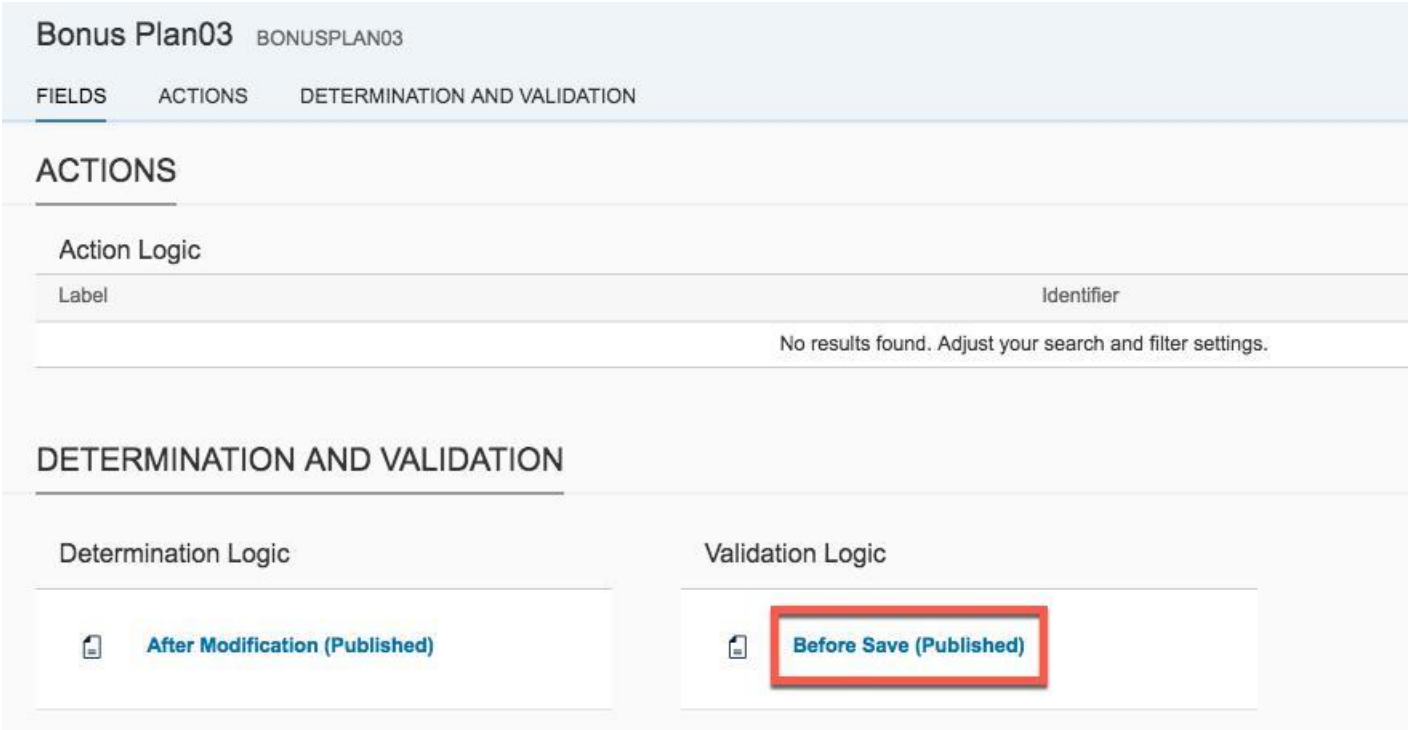
Implement Before Save

1. **Implement** Before Save event with following functionality

   If the bonus plan is consistent, it can be continued to save, if not save shall be rejected. In case of save no further processing is needed and logic can be left.

   > **Hint:** Exporting parameter valid must be set to true for save and to false for save rejection

2. Click on **Before Save (Published)**.



3. Click on **Create Draft**.

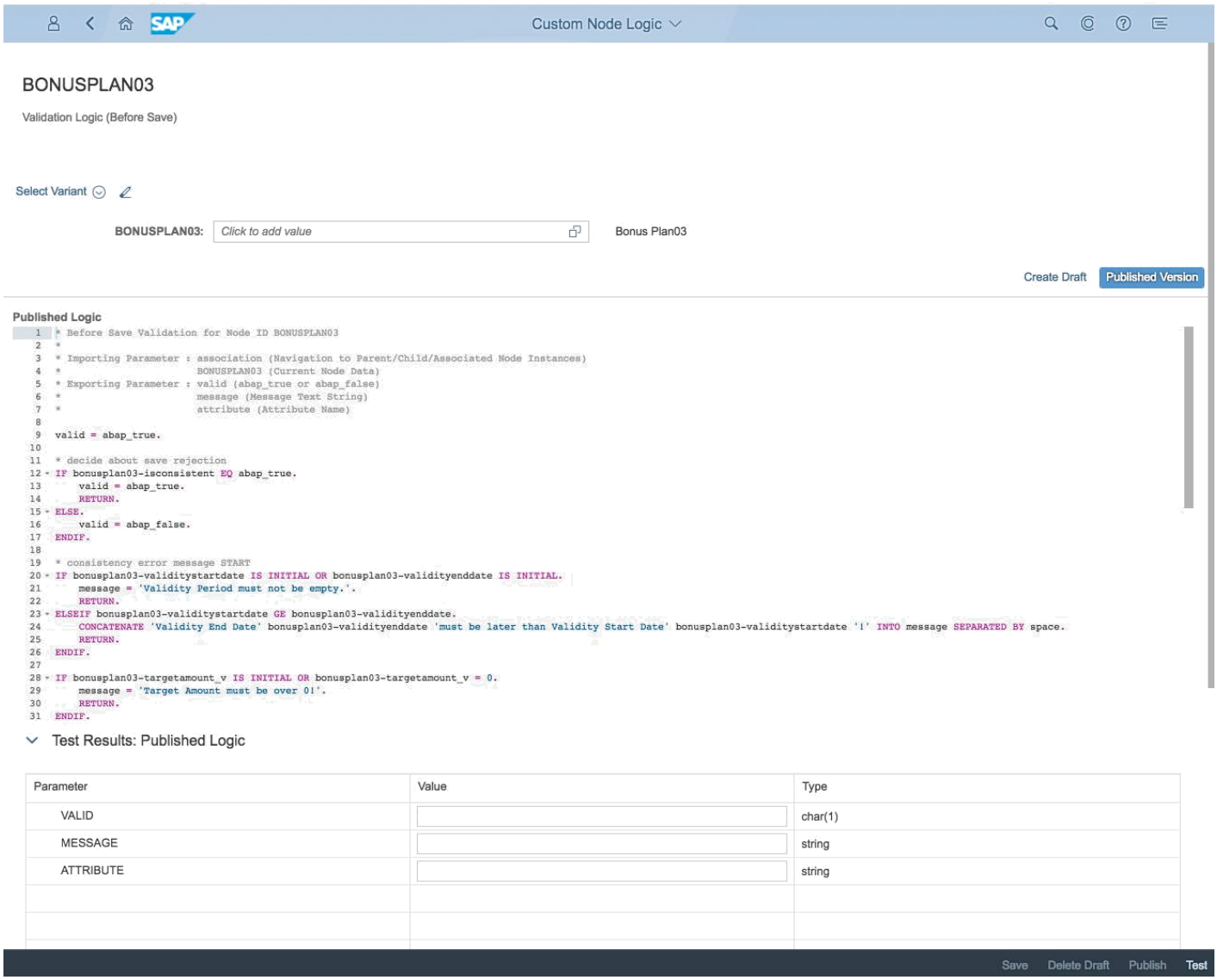
```
* decide about save rejection
IF bonusplanXX-isconsistent EQ abap_true.
    valid = abap_true.
    RETURN.
ELSE.
    valid = abap_false.
ENDIF.
```

- If the bonus plan is not consistent, write the first found error into the message and end the logic processing. These are the possible errors in detail:
  - `ValidityStartDate` and `ValidityEndDate` must be set
  - `ValidityStartDate` must be earlier in time than `ValidityEndDate`
  - Factors and Percentages must be > 0
  - Percentages must be < 100
  - LowBonusAssignmentFactor must be < HighBonusAssignmentFactor
  - Empoyee ID must be set

```
* consistency error message START
IF bonusplanXX-validitystartdate IS INITIAL OR bonusplanXX-validityenddate IS INITIAL
.
    message = 'Validity Period must not be empty.'.
    RETURN.
ELSEIF bonusplanXX-validitystartdate GE bonusplanXX-validityenddate.
    CONCATENATE 'Validity End Date' bonusplanXX-validityenddate 'must be later than
V alidity Start Date' bonusplanXX-validitystartdate '!' INTO message SEPARATED BY
space .
    RETURN.
ENDIF.

IF bonusplanXX-targetamount_v IS INITIAL OR bonusplanXX-targetamount_v = 0.
    message = 'Target Amount must be over 0!'.
    RETURN.
ENDIF.

IF bonusplanXX-targetamount_c IS INITIAL.
    message = 'Target Amount Currency must be set!'.
    RETURN.
ENDIF.

IF bonusplanXX-lowbonusassignmentfactor IS INITIAL OR
 bonusplanXX-highbonusassignmentfactor IS INITIAL.
    message = 'Assignment Factors must be over 0!'.
    RETURN.
```

```
ENDIF.

IF bonusplanXX-lowbonuspercentage_v IS INITIAL OR
 bonusplanXX-highbonuspercentage_v IS INITIAL.
    message = 'Percentages must be over 0!'.
    RETURN.
ENDIF.

IF bonusplanXX-lowbonuspercentage_v GE 100
 OR bonusplanXX-highbonuspercentage_v GE 100.
    message = 'Percentage must be below 100!'.
    RETURN.
ENDIF.

IF bonusplanXX-lowbonusassignmentfactor GE bonusplanXX-highbonusassignmentfactor.
    message = 'Low Bonus Factor must be smaller than High Bonus Factor!'.
    RETURN.
ENDIF.

IF bonusplanXX-employeeid IS INITIAL.
    message = 'Employee ID must be set!'.
    RETURN.
ENDIF.
* consistency error message  END
```

1. **Publish** the Before Save Logic.



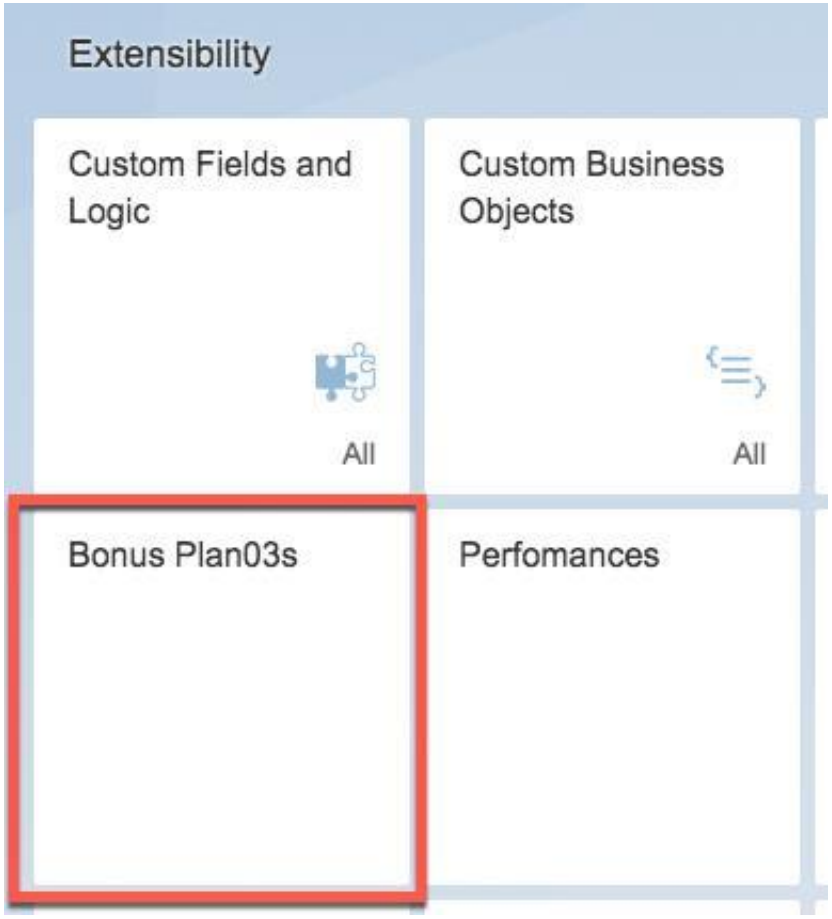
2. Go back and check both After Modication and Before Save are Published.

3. Go home.

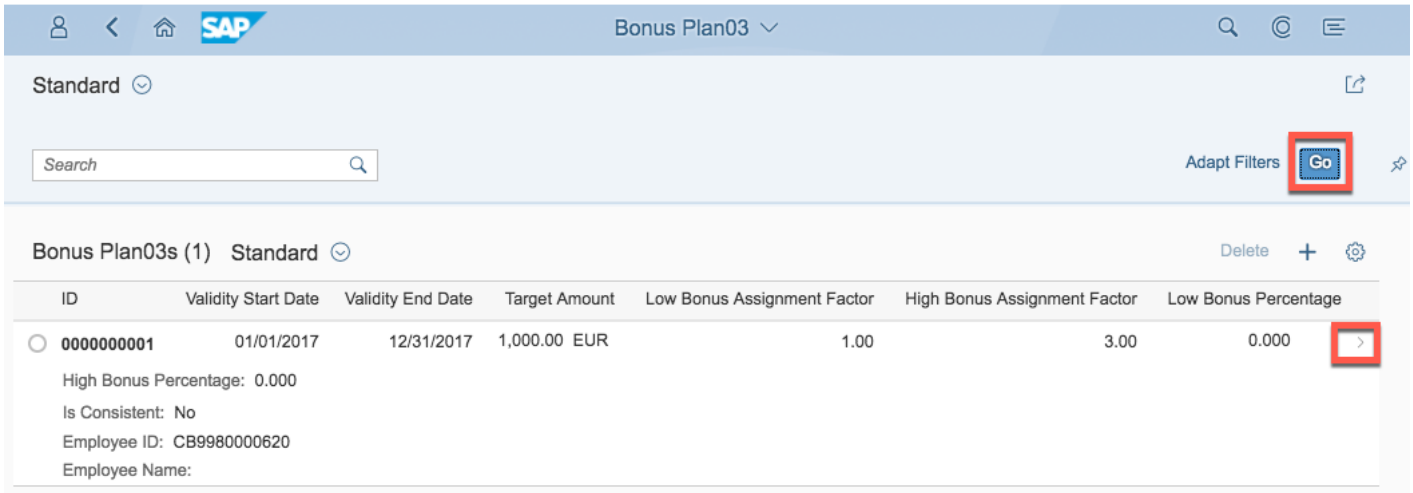
Test via the UI

Once ensured that both logic implementations were successfully published you can start testing the Application like an end user via the UI.

1. **Open** the Bonus PlanXX application.



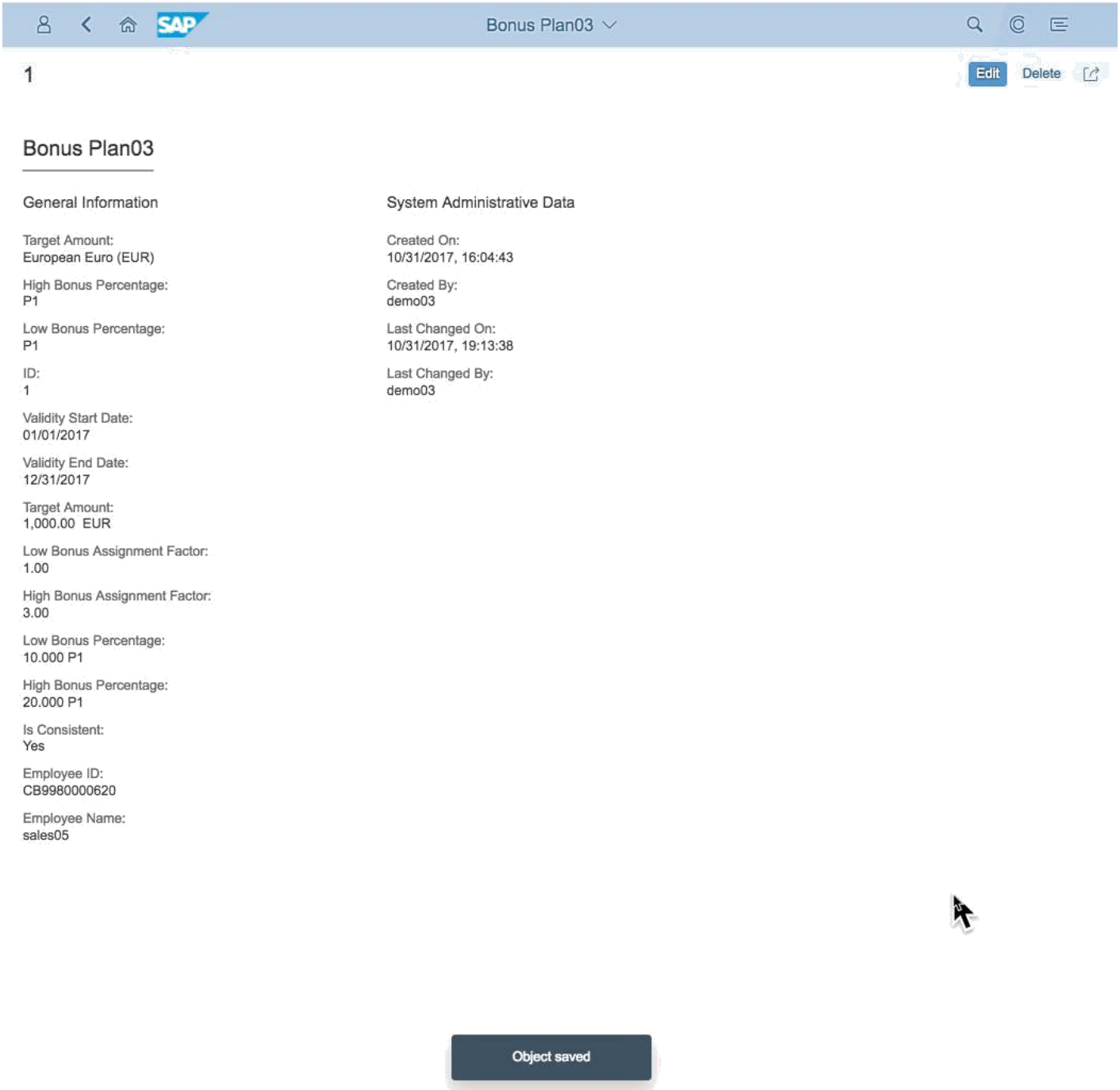
2. Click on Go. **Open** the Bonus Plan with ID `1`.



3. **Edit** this Bonus Plan.

4. **Enter** value `10` into field **Low Bonus Percentage**

5. **Save** the Bonus plan.

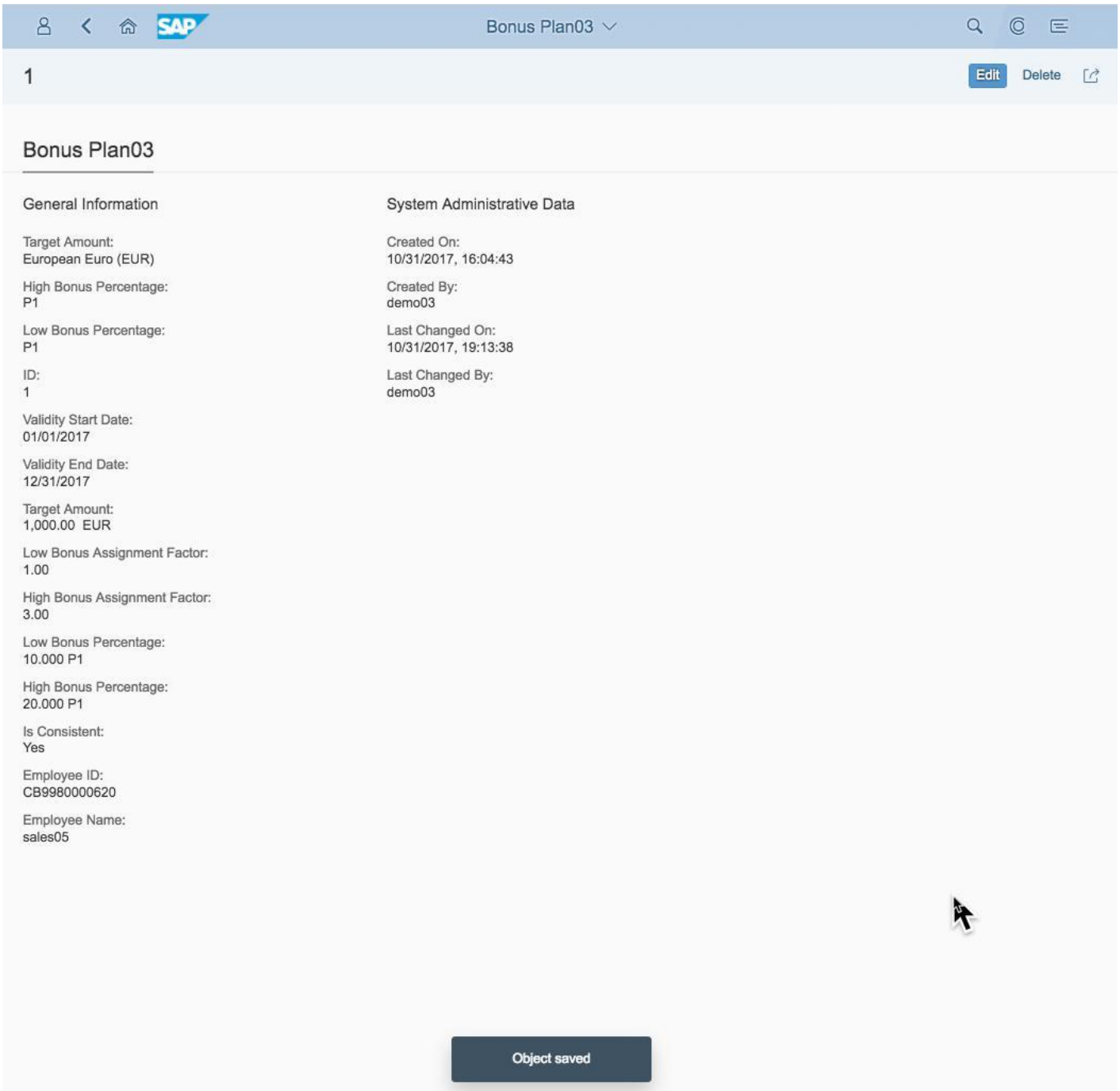


6. Save fails due to the validation error messages for missing percentages.

7. Close the Message.

8. **Enter** value `20` into field **High Bonus Percentage**

9. **Save** the Bonus Plan. Now it will not be rejected. You can see that your business logic works as the Percentage Units and the Employee Name get filled.



Summary

This concludes the exercise. You should have learned how to implement logic to set some data from the backend only and to check all data of an instance.

You will also learn how to ease development and test already while doing it.

Please proceed with exercise 01D.