

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

1. [Make key field Read-Only](#)
2. [Enable logic implementation](#)
3. [Start logic implementation](#)
4. [Implement After Modification: fix values](#)
5. [Implement After Modification: consistency check](#)
6. [Test the logic during development](#)
7. [Implement Before Save](#)
8. [Test via the UI](#)

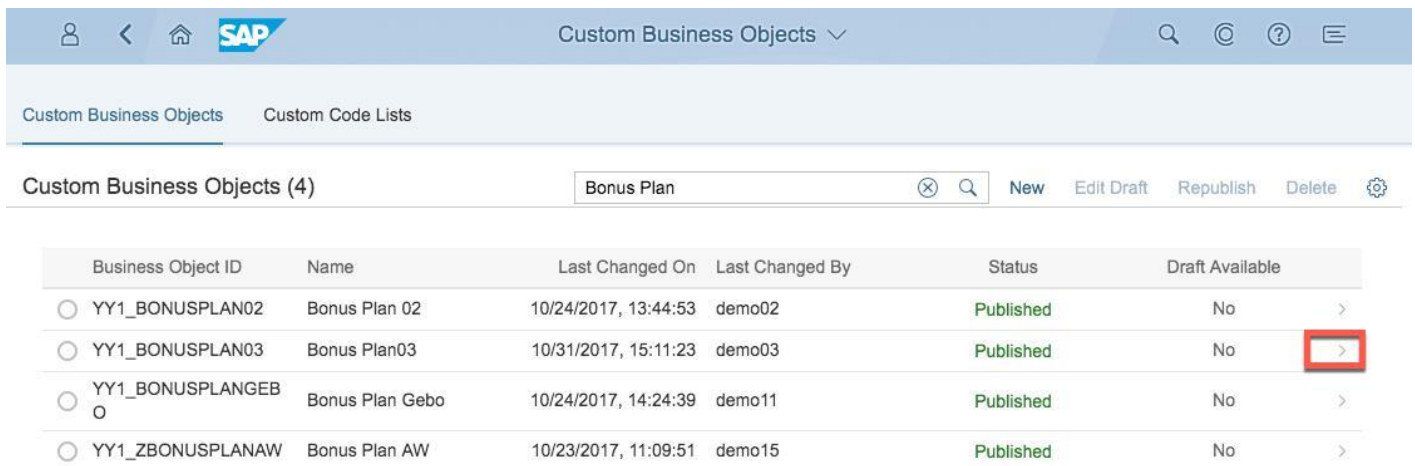
Make key field Read-Only

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.



Business Object ID	Name	Last Changed On	Last Changed By	Status	Draft Available
YY1_BONUSPLAN02	Bonus Plan 02	10/24/2017, 13:44:53	demo02	Published	No >
YY1_BONUSPLAN03	Bonus Plan03	10/31/2017, 15:11:23	demo03	Published	No >
YY1_BONUSPLANGEBO	Bonus Plan Gebo	10/24/2017, 14:24:39	demo11	Published	No >
YY1_ZBONUSPLANAW	Bonus Plan AW	10/23/2017, 11:09:51	demo15	Published	No >

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



Name	Identifier	Name in Plural	Details
Bonus Plan03	BONUSPLAN03	Bonus Plan03s	Go to Fields and Logic

Edit Draft Delete

3. Go to Fields and Logic.

Structure New Delete ⚙

Name	Identifier	Name in Plural	Details
Bonus Plan03	BONUSPLAN03	Bonus Plan03s	Go to Fields and Logic

Publish Save Draft Cancel Delete Draft

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

FIELDS ACTIONS

Fields New Delete ⬆ ⬇ ⬇ ⬇ ⚙

Naming	Definition	Further Properties
<p>Label: ID</p> <p>Identifier: ID</p> <p>Tooltip: ID</p>	<p>Type: Numeric Identifier</p> <p>Length: 10</p> <p>Key: <input checked="" type="checkbox"/></p>	<p>Read Only: <input checked="" type="checkbox"/></p> <p>Transpo...: No</p>

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**

Edit Custom Business Object 🔍 © ? ☰

Bonus Plan03 YY1_BONUSPLAN03 Published

12
Structure Log

***Name:** Bonus Plan03 Created: 10/31/2017, 13:01:26, demo03

***Name in Plural:** Bonus Plan03s Changed: 10/31/2017, 17:12:16, demo03

Determination and Validation: ☒ Original Language: English

UI Generation: ☒ Can Be Associated: ☐

Service Generation: ☒ System Administrative Data: ☒

Structure New Delete ⚙

Name	Identifier	Name in Plural	Details
Bonus Plan03	BONUSPLAN03	Bonus Plan03s	Go to Fields and Logic

Publish Save Draft Cancel Delete Draft

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

Structure New Delete ⚙

Name	Identifier	Name in Plural	Details
Bonus Plan03	BONUSPLAN03	Bonus Plan03s	Go to Fields and Logic

[Edit Draft](#) [Delete](#)

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

<

Node ▾

Bonus Plan03 BONUSPLAN03

FIELDS

ACTIONS

DETERMINATION AND VALIDATION

Action Logic

New Delete ⬆ ⬆ ⬆ ⬆

Label	Identifier
No results found. Adjust your search and filter settings.	

DETERMINATION AND VALIDATION

Determination Logic

Validation Logic

[After Modification \(Published\)](#)



[Before Save \(Published\)](#)

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

SAP Custom Node Logic

BONUSPLAN03

Determination Logic (After Modification)

Select Variant  

BONUSPLAN03: Bonus Plan03

Create Draft **Published Version**

Published Logic

```

1 * After Modify Determination for Node ID BONUSPLAN03
2 *
3 * Importing Parameter : association (Navigation to Parent/Child/Associated Node Instances)
4 * Changing Parameter : BONUSPLAN03 (Current Node Data)
5

```

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.

Draft Version **Published Version**

Draft Logic

```

1 * After Modify Determination for Node ID BONUSPLAN03
2 *
3 * Importing Parameter : association (Navigation to Parent/Child/Associated Node Instances)
4 * Changing Parameter : BONUSPLAN03 (Current Node Data)
5

```

Published Logic

```

1 * After Modify Determination for Node ID BONUSPLAN03
2 *
3 * Importing Parameter : association (Navigation to Parent/Child/Associated Node Instances)
4 * Changing Parameter : BONUSPLAN03 (Current Node Data)
5

```

Implement After Modification: fix values

Implement After Modification event with following fix value functionality:

- Set the key field if still initial.

Hint: Changing Parameter 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:). **Hint:** With the key combination **CTRL + Space** you can access the very helpful code completion.

*Draft Logic

```

1  * After Modify Determination for Node ID BONUSPLAN03
2  *
3  * Importing Parameter : yy1_bonustitlement02 Declaration
4  * Changing Parameter : yy1_bonusplan02 Declaration
5  * yy1_bonusplan03 Declaration
6  * set ID yy1_bonusplangebo Declaration
7  IF bonusplan-id IS INITIAL yy1_bonus_calculation Declaration
8  SELECT MAX( id ) FROM yy1_bonusplan INTO @DATA(current_max_id).
9  bonusplan-id = current_max_id + 1.
10 ENDIF.
11
12

```

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( bonusplanXX-employeeid ).
ENDIF.

```

Implement After Modification: consistency check

In dependence on following checks, set the **inconsistent** 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

Select Variant  

BONUSPLAN03:  Bonus Plan03

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	<input type="text"/>	char(3)
highbonuspercentage_v	<input type="text"/>	decimal(15,3)
highbonuspercentage_u	20	char(3)
isconsistent	<input type="text"/>	char(1)
employeeid	CB9980000620	char(12)
employeename	<input type="text"/>	char(40)
sap_createddatetime	<input type="text"/>	decimal(21,7)
sap_createdbyuser	<input type="text"/>	char(12)
sap_lastchangeddatetime	<input type="text"/>	decimal(21,7)
sap_lastchangedbyuser	<input type="text"/>	char(12)

OK Cancel

Click OK.

- 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

```
9   bonusplan03-id = current_max_id + 1.
10  ENDIF.
11
12  * set percentage unit
13  bonusplan03-lowbonuspercentage_u = bonusplan03-highbonuspercentage_u = 'P1'.
14
15  * set Employee Name
16  * IF bonusplan03-employeeid IS NOT INITIAL.
17  bonusplan03-employeename = cl_abap_context_info=>get_user_formatted_name( bonusplan03-employeeid ).
18  ENDIF.
19
20  * consistency check START
21  * IF bonusplan03-validitystartdate IS INITIAL
22  OR bonusplan03-validityenddate IS INITIAL
23  OR bonusplan03-validitystartdate GE bonusplan03-validityenddate
24  OR bonusplan03-lowbonusassignmentfactor IS INITIAL
25  OR bonusplan03-highbonusassignmentfactor IS INITIAL
26  OR bonusplan03-lowbonuspercentage_v IS INITIAL
27  OR bonusplan03-highbonuspercentage_v IS INITIAL
28  OR bonusplan03-lowbonuspercentage_v GE 100
29  OR bonusplan03-highbonuspercentage_v GE 100
30  OR bonusplan03-lowbonusassignmentfactor GE bonusplan03-highbonusassignmentfactor
31  OR bonusplan03-employeeid IS INITIAL
32  OR bonusplan03-targetamount_v IS INITIAL
33  * OR bonusplan03-targetamount_c IS INITIAL.
34  bonusplan03-isconsistent = abap_false.
35  * ELSE.
36  bonusplan03-isconsistent = abap_true.
37  ENDIF.
38  * consistency check END
39
```

Draft 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)

Published Logic

```
1  * After Modify Determination for Node ID BONUSPL
2  *
3  * Importing Parameter : association (Navigation
4  * Changing Parameter : BONUSPLAN03 (Current Nod
5
```

Published logic tested

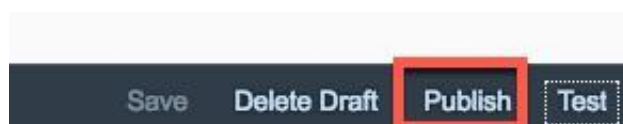
Test Results: Published Logic

Parameter	Value	Type
> BONUSPLAN03	[structure]	

Save Delete Draft Publish **Test**

You can see that your logic works as id , *percentage_u fields and employeename are filled and inconsistent 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)**.

Bonus Plan03 BONUSPLAN03

FIELDS **ACTIONS** **DETERMINATION AND VALIDATION**

ACTIONS

Action Logic

Label	Identifier
No results found. Adjust your search and filter settings.	

DETERMINATION AND VALIDATION

Determination Logic

After Modification (Published)

Validation Logic

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`
- Employee 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.

<

Custom Node Logic

Q

©

?

≡

BONUSPLAN03

Validation Logic (Before Save)

Select Variant

BONUSPLAN03:

Click to add value

+

Bonus Plan03

Create Draft

Published Version

Published Logic

```

1 * Before Save Validation for Node ID BONUSPLAN03
2 *
3 * Importing Parameter : association (Navigation to Parent/Child/Associated Node Instances)
4 * BONUSPLAN03 (Current Node Data)
5 * Exporting Parameter : valid (abap_true or abap_false)
6 * message (Message Text String)
7 * attribute (Attribute Name)
8
9 valid = abap_true.
10
11 * decide about save rejection
12 IF bonusplan03-isconsistent EQ abap_true.
13     valid = abap_true.
14     RETURN.
15 ELSE.
16     valid = abap_false.
17 ENDIF.
18
19 * consistency error message START
20 IF bonusplan03-validitystartdate IS INITIAL OR bonusplan03-validityenddate IS INITIAL.
21     message = 'Validity Period must not be empty.'.
22     RETURN.
23 ELSEIF bonusplan03-validitystartdate GE bonusplan03-validityenddate.
24     CONCATENATE 'Validity End Date' bonusplan03-validityenddate 'must be later than Validity Start Date' bonusplan03-validitystartdate 'I' INTO message SEPARATED BY space.
25     RETURN.
26 ENDIF.
27
28 IF bonusplan03-targetamount_v IS INITIAL OR bonusplan03-targetamount_v = 0.
29     message = 'Target Amount must be over 0!'.
30     RETURN.
31 ENDIF.

```

Test Results: Published Logic

Parameter	Value	Type
VALID		char(1)
MESSAGE		string
ATTRIBUTE		string

Save

Delete Draft

Publish

Test

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



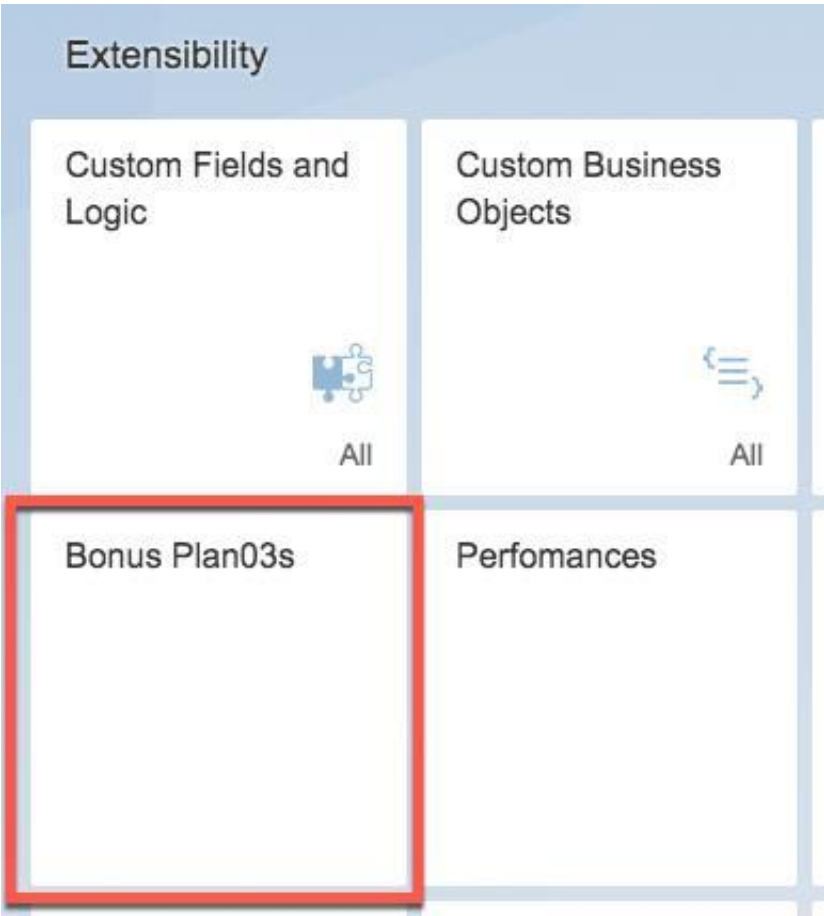
3. Go home.



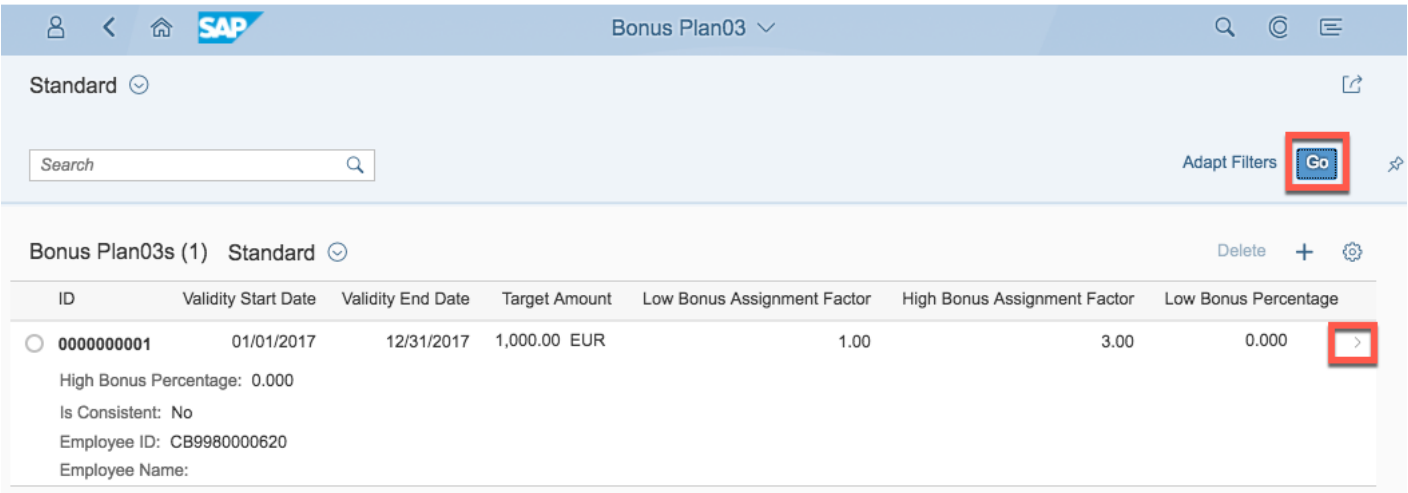
Test via the UI

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 .



3. **Edit** this Bonus Plan.



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

Low Bonus Percentage:

10.000	
--------	--

5. **Save** the Bonus plan.

Bonus Plan03

General Information

Target Amount:
European Euro (EUR)
High Bonus Percentage:
P1
Low Bonus Percentage:
P1
ID:
1
Validity Start Date:
01/01/2017
Validity End Date:
12/31/2017
Target Amount:
1,000.00 EUR
Low Bonus Assignment Factor:
1.00
High Bonus Assignment Factor:
3.00
Low Bonus Percentage:
10.000 P1
High Bonus Percentage:
20.000 P1
Is Consistent:
Yes
Employee ID:
CB9980000620
Employee Name:
sales05

System Administrative Data

Created On:
10/31/2017, 16:04:43
Created By:
demo03
Last Changed On:
10/31/2017, 19:13:38
Last Changed By:
demo03

Object saved

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

1

Messages

1

Percentages must be over 0!

Close

7. Close the Message.

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

High Bonus Percentage:

20.000

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.

1

Edit Delete

Bonus Plan03

General Information	System Administrative Data
Target Amount: European Euro (EUR)	Created On: 10/31/2017, 16:04:43
High Bonus Percentage: P1	Created By: demo03
Low Bonus Percentage: P1	Last Changed On: 10/31/2017, 19:13:38
ID: 1	Last Changed By: demo03
Validity Start Date: 01/01/2017	
Validity End Date: 12/31/2017	
Target Amount: 1,000.00 EUR	
Low Bonus Assignment Factor: 1.00	
High Bonus Assignment Factor: 3.00	
Low Bonus Percentage: 10.000 P1	
High Bonus Percentage: 20.000 P1	
Is Consistent: Yes	
Employee ID: CB9980000620	
Employee Name: sales05	

Object saved

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.