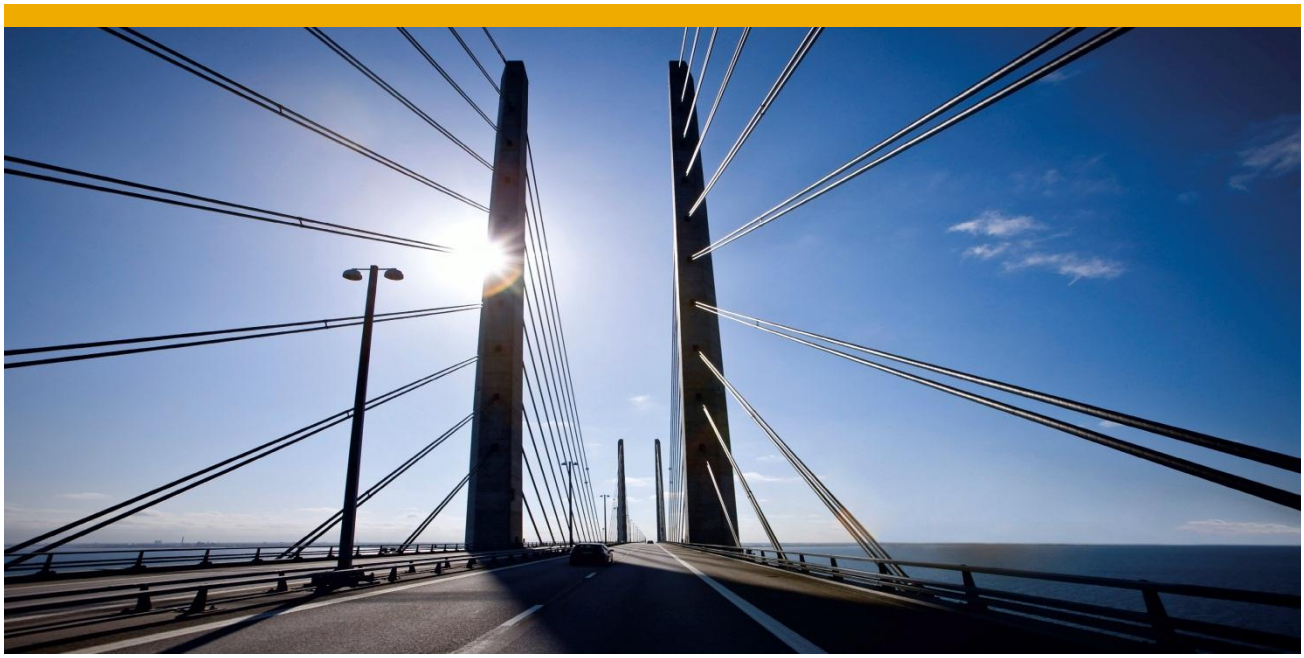

Creating and Implementing Determinations in Business Object Builder

Summary

This tutorial shows how to calculate the content of a new persisted field in a node of a BOPF business object. The model entity that allows calculations is the BOPF determination. The tutorial uses the Business Object Builder (BOB).

Level of complexity:	**Beginner**
Time required for completion:	**20 minutes**

Author:	**Thea Hillenbrand**
Company:	**SAP AG**
Created on:	**20 January 2014**

TABLE OF CONTENTS

BEFORE YOU START

Deriving data from the content of other fields is a core task of business programming. Examples are the calculation of the gross amount of all line items. The result of these calculations can be stored in the database or calculated for each user session. Your decision in this regard depends on various factors such as performance considerations, the complexity of update scenarios, or simply business requirements. BOPF supports these use cases with the determination entity and transaction BOB offers corresponding implementation patterns. In this tutorial we will show you how to create a determination where the result is stored in a database field.

Objectives

By the end of this tutorial, you will be able to

- Enhance the persisted structure of a BO node
- Create a determination
- Implement a simple determination based on the BOPF API
- Test the new features of the business object.

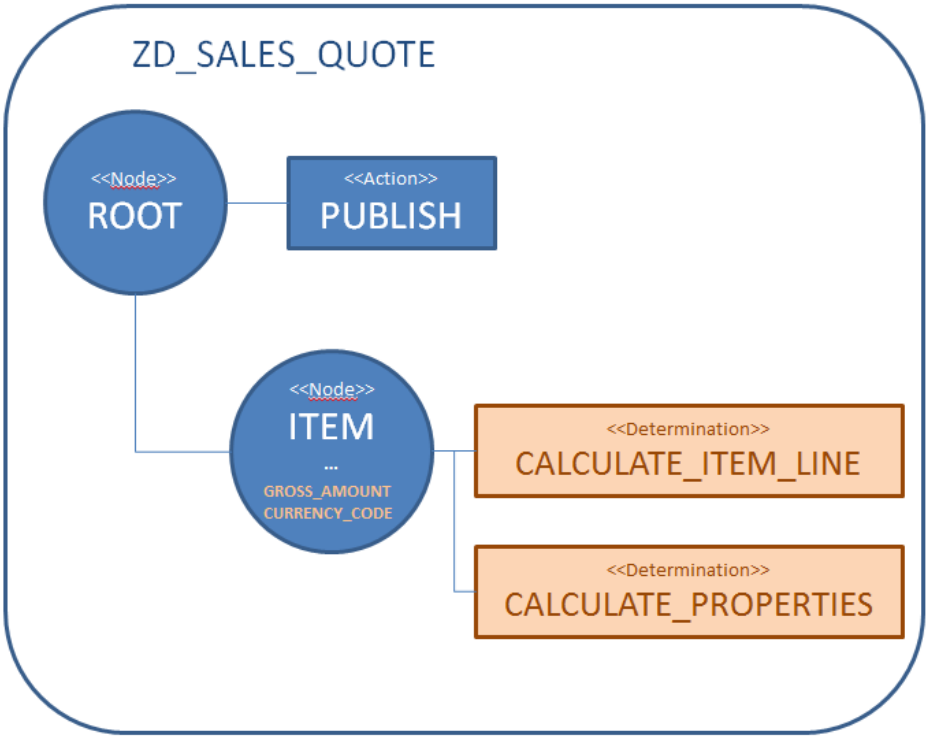

Fig. 1: Structure of the Business Object to be enhanced

The tutorial starts with the business object created in the 'Getting Started with the Business Object Processing Framework'. It consists of the ROOT node with minimal header information, like QUOTE_ID, and the ITEM node with position data like PRODUCT_ID, quantity, and price information. The goal is to calculate the gross amount of one line item. The gross amount of a line item is stored in the database. Therefore we need two additional fields (***GROSS*_AMOUNT, CURRENCY_*CODE***) in the item database table and a determination on the item node (**CALCULATE_ITEM_LINE**).

Calculations can serve as default values or they can be used as results that should not be changed by user input. In our example, we do not want them to be changed by users. Therefore we need a second determination (***CALCULATE*_PROPERTIES**) to set the field properties to read-only.

Prerequisites

To be able to perform the tutorial, make sure the following prerequisites are fulfilled.

Systems, Releases, and Authorizations

- BOPF is part of the Business Suite Foundation Layer and, therefore, included in the following SAP Business Suite releases:
  - SAP Business Suite EHP5, SP11
  - SAP Business Suite EHP6, SP05
  - SAP Business Suite EHP7, all SP
- To create determinations, your SAP user requires the developer authorization profile (S_DEVELOP authorization object)
- To implement this example, you need the business object created in the tutorial “Getting Started with the Business Object Processing Framework”.

Knowledge

- Basic knowledge in ABAP OO
- Experience with DDIC tools

ENHANCING THE STRUCTURE OF A BUSINESS OJBECT NODE

In this step, you will enhance the ITEM node of the Business Object (BO) “*SALES_QUOTE*”. This Business Object follows the semantics of the sales quote based on the NetWeaver Enterprise Procurement Model (EPM).

Note: The EPM example persists the net amount and calculates the gross amount for the user session only. It is interesting to see how these two types of determinations interact. In this tutorial we have chosen a simple example.

Procedure

Launch the Business Object Builder (BOB)

Transaction BOB provides the design time for custom business objects as well as business object enhancements.

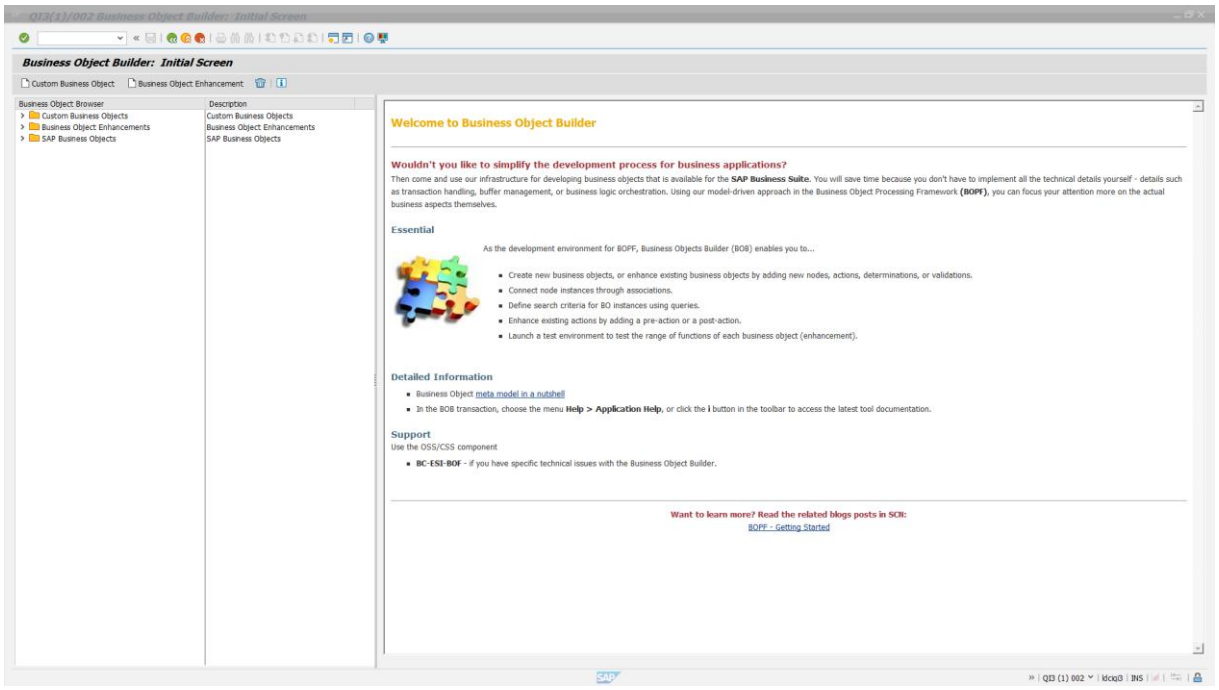

Fig. 2: Welcome page of the BOB transaction in a SAP customer system

On the left side of the initial screen you see three categories of BOs that are currently available in your system:

- Custom Business Objects: BOs created by the customer in their system
- SAP Business Objects: Extensible BOs that are delivered by SAP
- Business Object Enhancement: BOs that are enhancements to an existing business object.

Select the business object ZD_SALES_QUOTE.

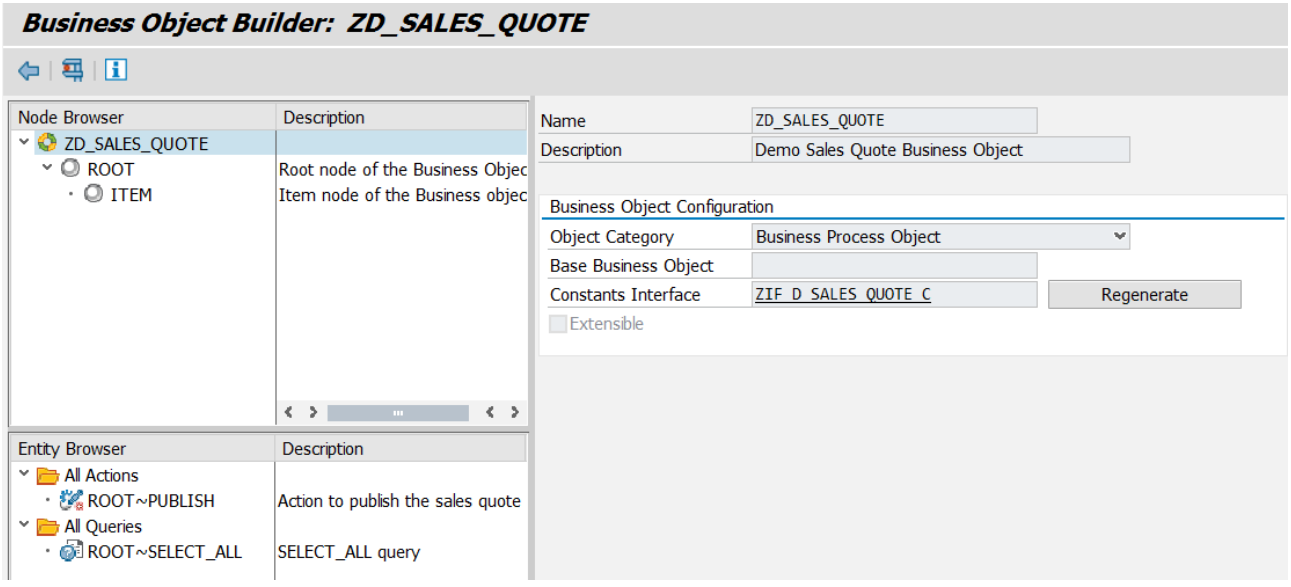

Fig. 3: SALES_QUOTE resulting from 'Getting started with Business Object Processing Framework'

Select the node ITEM and navigate to the definition of the persistent structure ZDS_ITEM_D in the ABAP data dictionary.

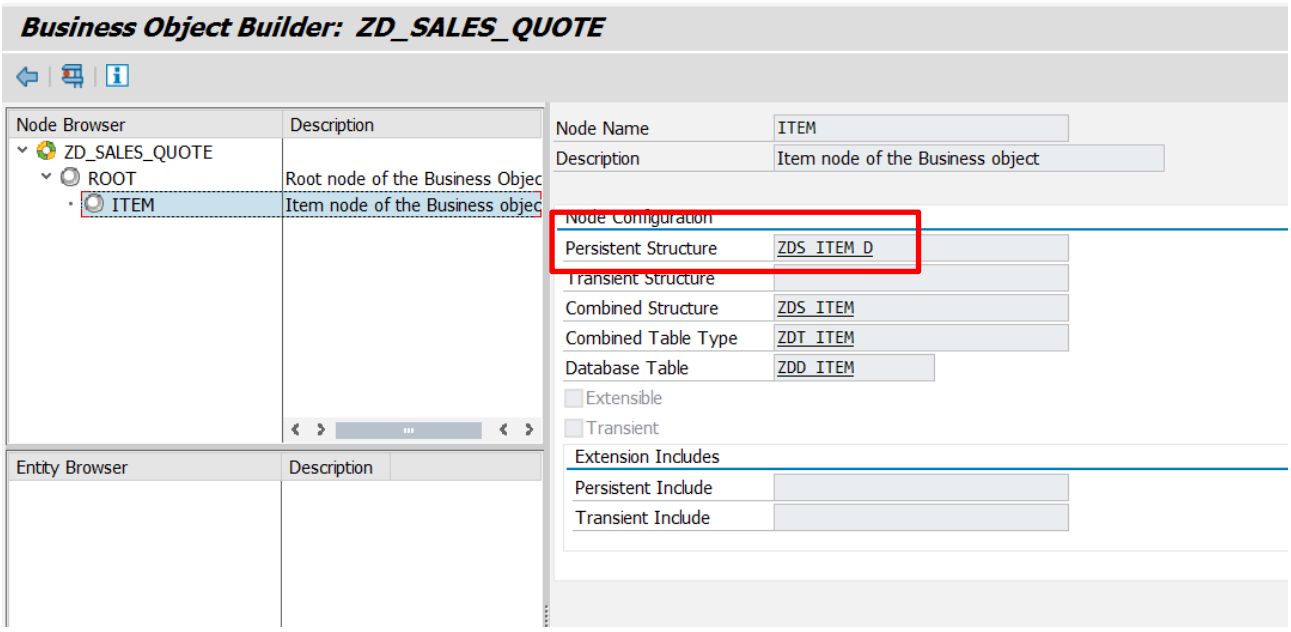

Fig. 4: ITEM node

In the data dictionary, switch to edit mode and enter the new components GROSS_AMOUNT of data type SNWD_TTL_GROSS_AMOUNT and CURRENCY_CODE of data type SNWD_CURR_CODE.

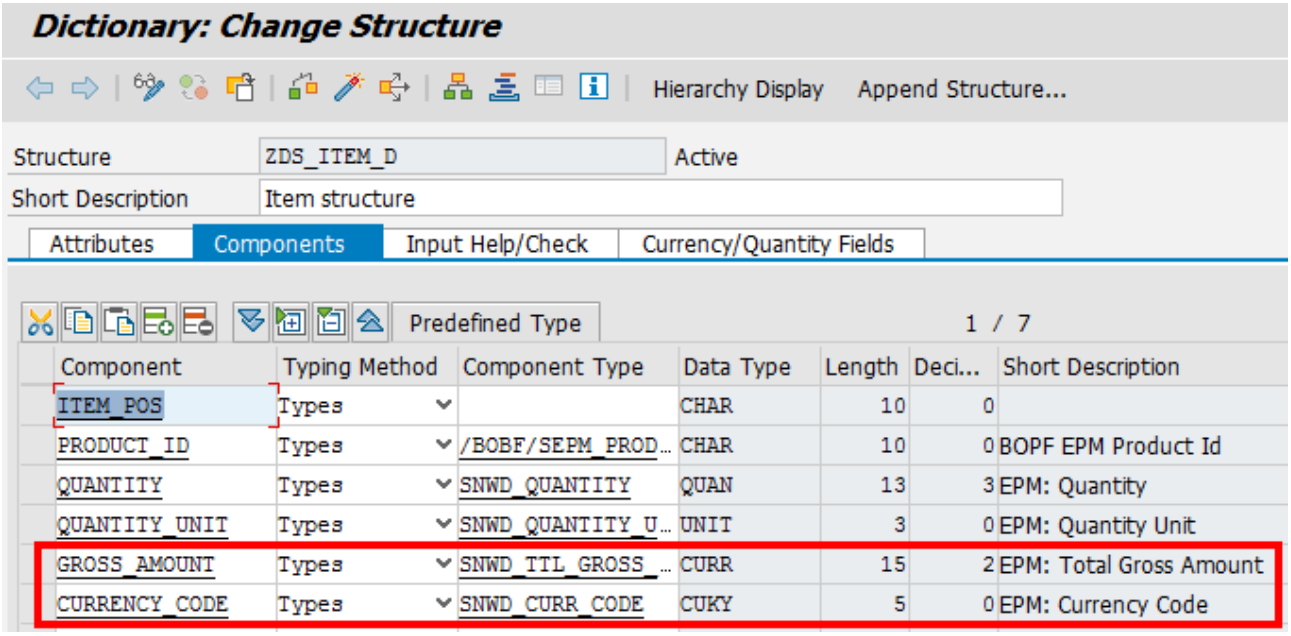

Fig. 5: Define the additional components- either directly in the component list or as Append Structure

Now the two components have to be linked so that consumers – especially generic ones like FPM or ALV - can use the dependency for calculations. Select the tab *Currency/Quantity Fields* and enter the reference.

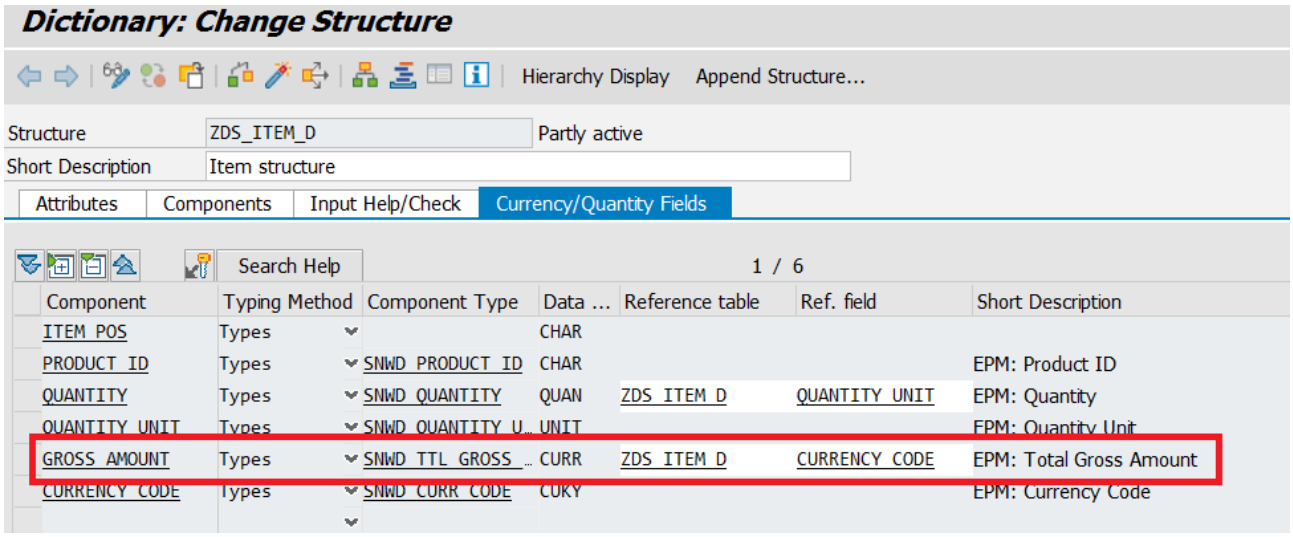

Fig. 6: Link the new components

Save and activate the definition. Then go back to the Business Object Builder.

Launch the Test Tool

Back in transaction BOB, choose the **Test** button, starting the BOPF test tool. Create a new SALES_QUOTE instance by choosing the **Add Node Instance** button on the ROOT node.

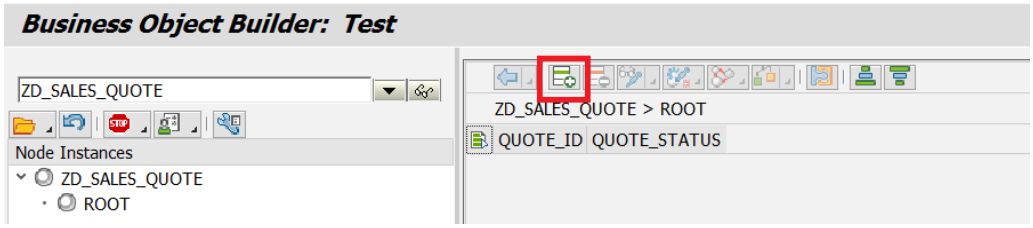

Fig. 7: create the root node instance

Enter at least a QUOTE_ID and navigate to the ITEM node by choosing the navigation menu button.

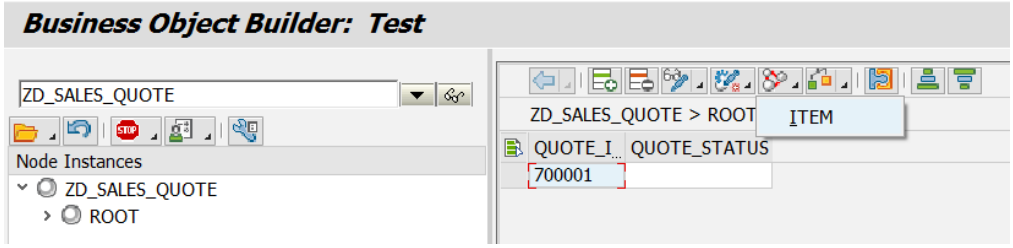

Fig. 8: Navigate to the item instance

You can now create an instance of the ITEM node by choosing the **Add Node Instance** button. The new components GROSS_AMOUNT and CURRENCY_CODE are visible and can be maintained.

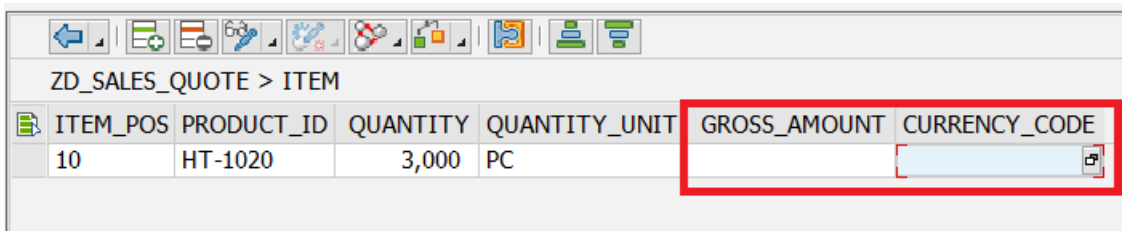

Fig. 9: The ITEM node contains the component DISCOUNT

You can enter the data for the ITEM node, even in the added fields. The calculation of the gross amount from the product price list and the quantity will be the task of the next step.

Result

You have now enhanced the original ITEM structure by the GROSS_AMOUNT and CURRENCY_CODE fields. You can enter data, store it, and retrieve it.

ADDING A DETERMINATION

In this step we will enhance the BO ITEM node by adding a determination. Determinations calculate data on the basis of the user input. In our particular case, the business requirement is that the gross amount is calculated using the product price list and the quantity of the line item. The gross amount should be calculated and persisted.

Prerequisites

The ITEM node of the BO is opened in the configuration view of the Business Object Builder.

Procedure

Start the Wizard to Create Determinations

Open the context menu of the ITEM node in the node browser pane. Select the entry **Create Determination**. Choose **Continue** to go to the first step.

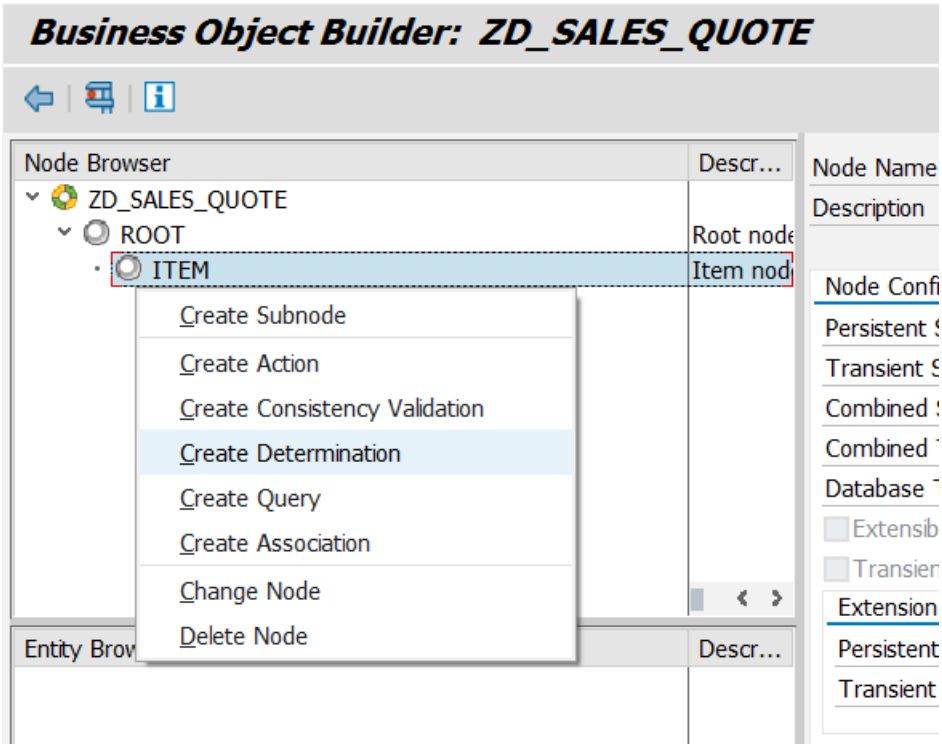

Fig. 10: Create Determination

Define the Name and Description of the Determination

In this step, you have to provide the determination name and you can define a short description.

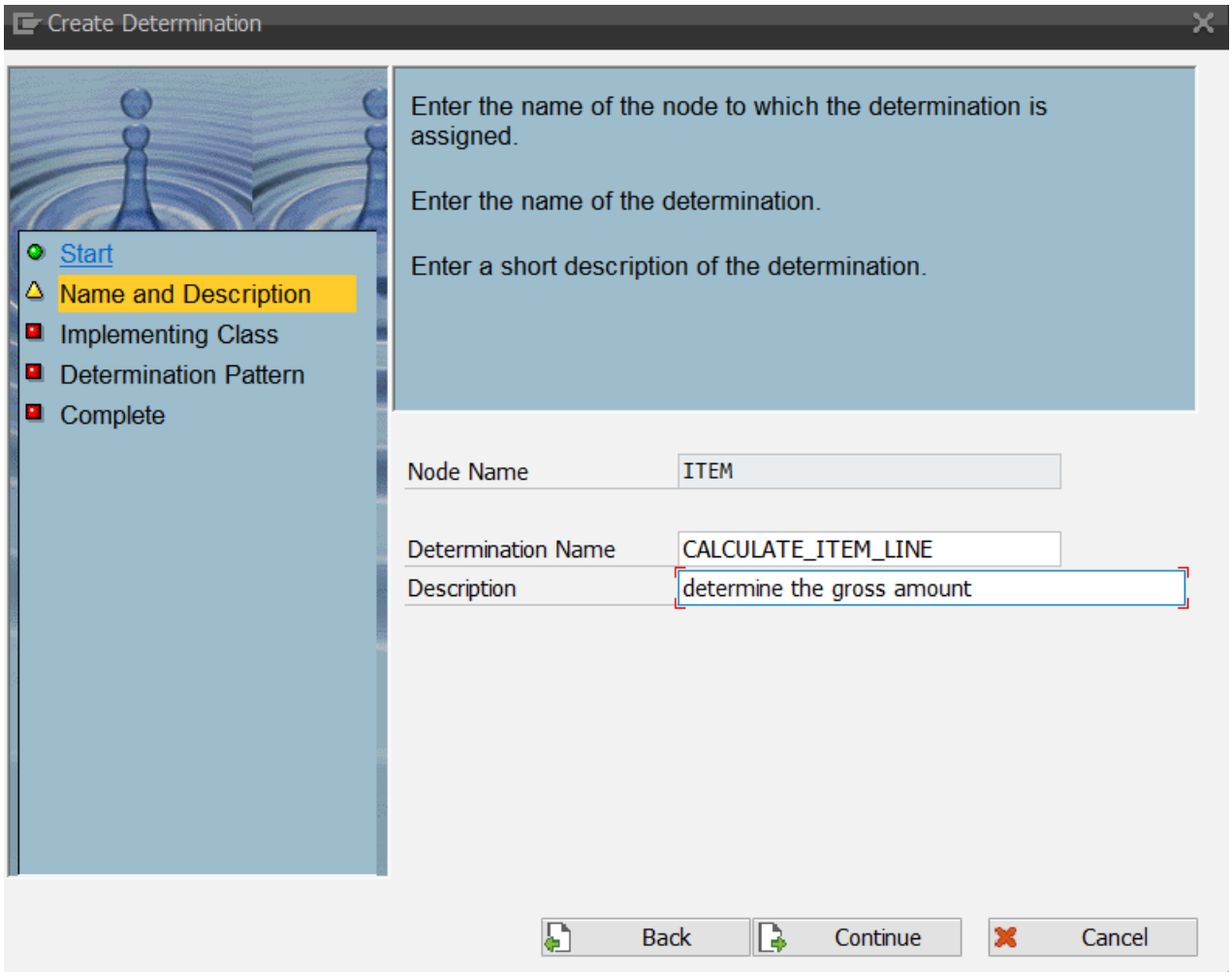

Fig. 11: Name of the determination

The target of our determination will be to calculate the gross amount depending on the product and quantity of the ITEM node of the SALES_QUOTE. Thus we name the validation CALCULATE_ITEM_LINE.

Continue to the next step.

Define the Implementing Class

At runtime, a determination is represented by an ABAP class that implements the determination interface. In this step you are able to define the name of that class.

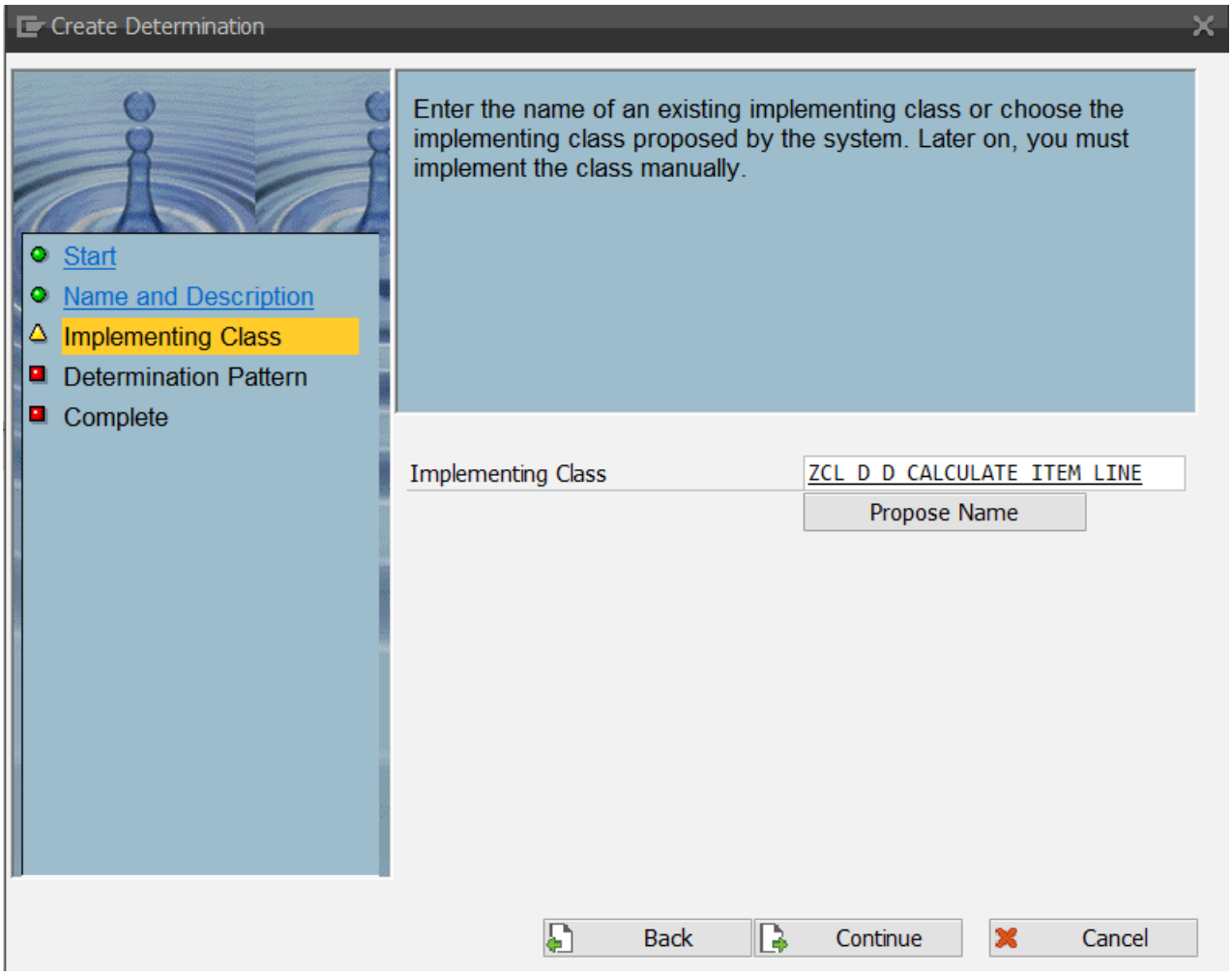

Fig. 12: Implementing the determination class

For this demo we won't change the proposed name, but continue instead with the next step.

Define the Determination Pattern

Determinations can be executed in different phases of the transaction. Depending on the use case it may be necessary to execute them in more than one phase. There are some canonical combinations, which we call determination patterns.

In our case the result of the determination should immediately be visible on the UI and it is stored in the database. We have to use the pattern *Derive dependent data immediately after modification*.

The second pattern *Derive dependent data before saving* addresses use cases where the result of the determination is equally stored on the database but where the calculation is very expensive, so that the user does not expect to see the result immediately on the UI after having changed a parameter. This option ensures that the calculation will only be executed once just before the database update.

The next two options handle transient data. They are not selectable as we didn't assign a transient structure to the item node, only a persistent one.

The pattern *Fill transient attributes of persistent nodes* addresses use cases where the data is calculated for the user session. The determination is called after the data has been selected from the database and also at every change of the trigger node.

Finally, the option *Derive instances of transient nodes* should be used whenever an entire node is transient and its attribute values need to be calculated for the user session. The determination is executed after navigation to instances on that node and at every change of the trigger node.

The last pattern is a special one and will be treated in the next chapter

Therefore we select *Derive dependent data immediately after modification*.

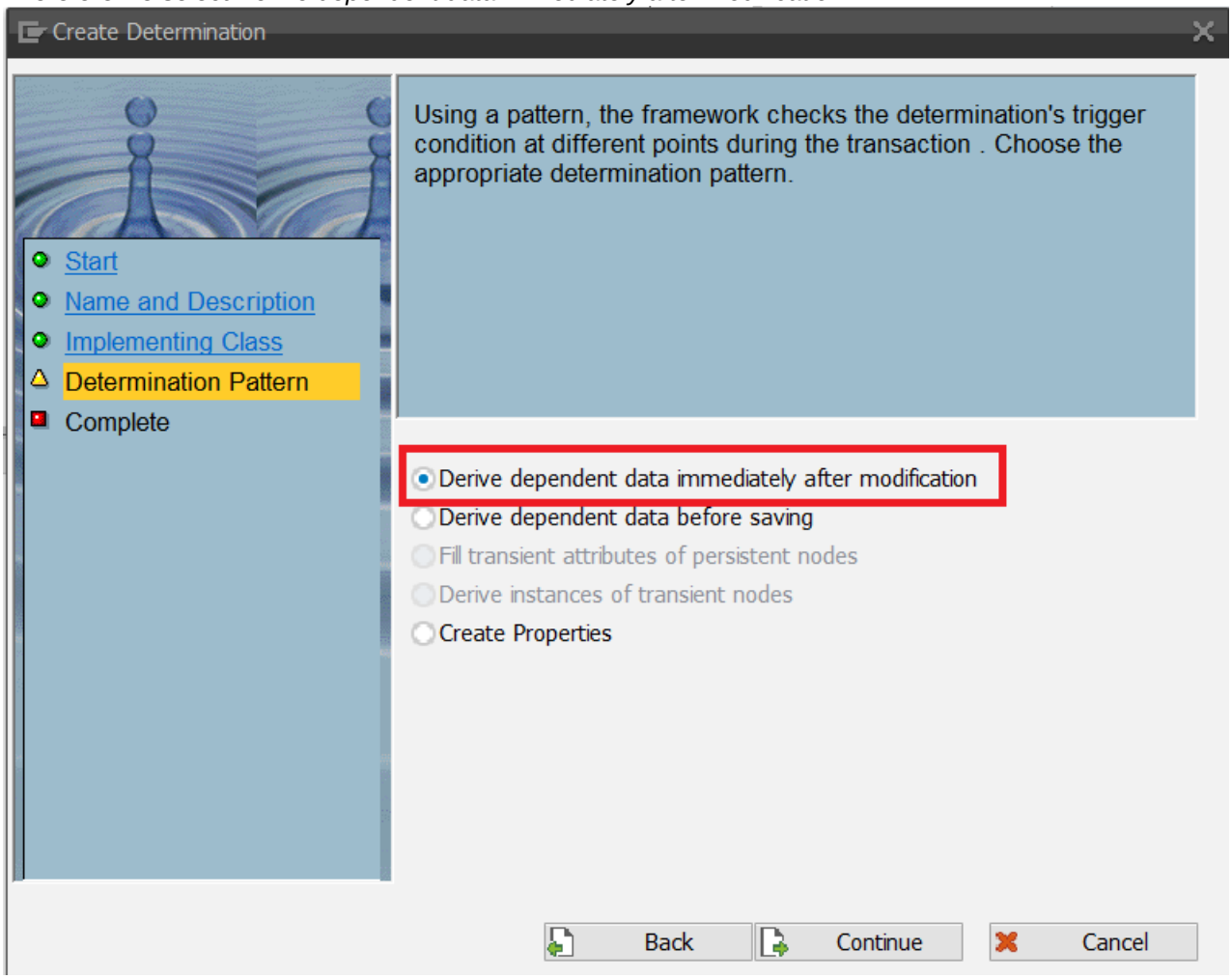

Fig. 13: Define the determination pattern

Continue with the next step.

Define the Request Nodes

The patterns you selected define in which phases of the transaction the determination is executed. In this step we have to define the conditions for executing the determination.

In our example we want to calculate the gross amount of a line item. So the determination must be triggered for each line item that has been changed. Hence we select the ITEM as request node and the *Create* and *Update* as trigger actions.

Hint: If you want to calculate the total for the sales quote, you have to create a determination on the root node as the total is an attribute of the root node. But the trigger node is still the item node, since the total has to be recalculated each time an item instance is changed.

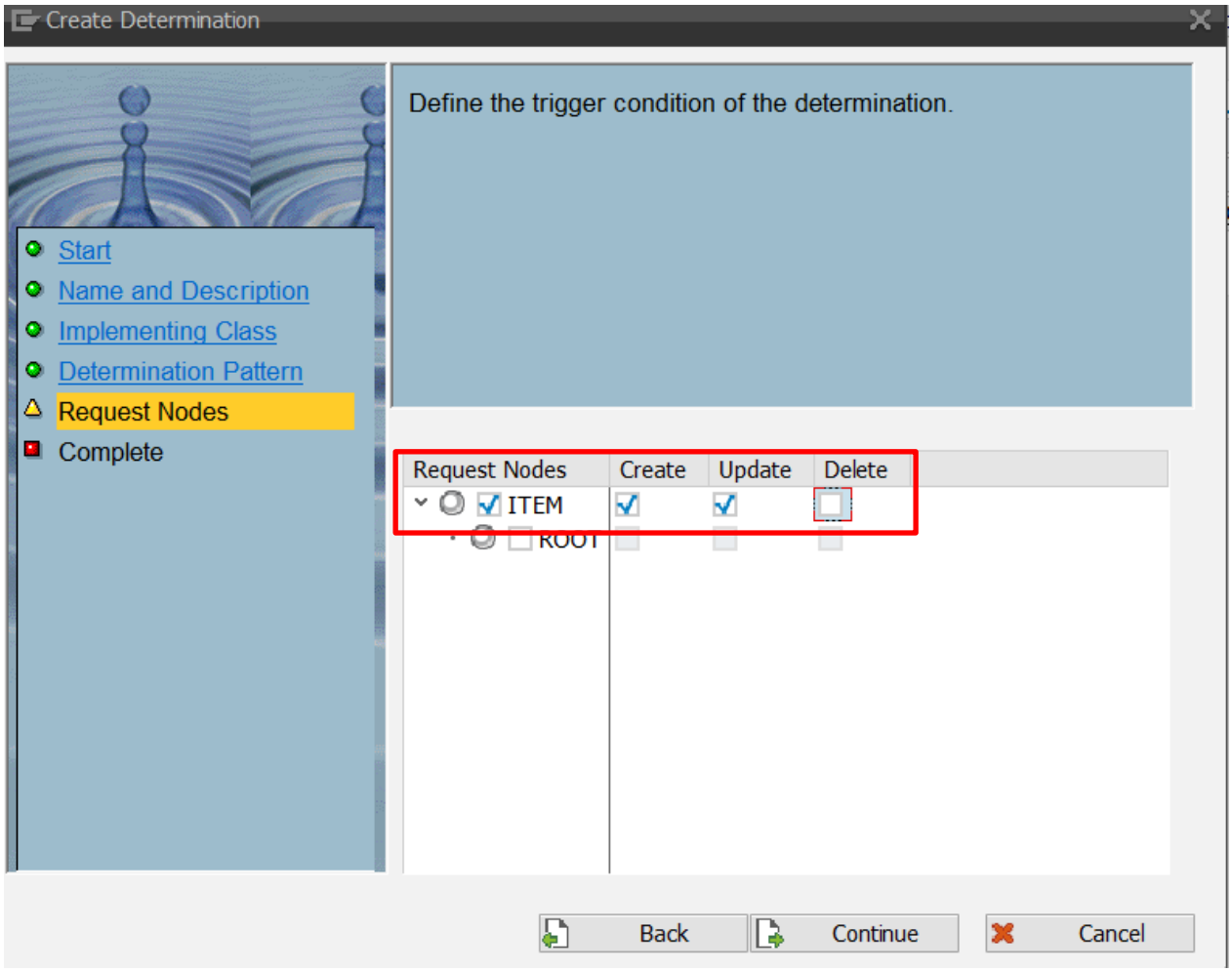

Fig. 14: Define the trigger conditions

Finish the Wizard

Finish the determination creation by choosing **Complete**. In the background, the system updates the BO configuration and generates the implementing class of the determination.

Implement the Determination

The new determination is now visible in the Entity Browser pane of the BO configuration view. On the right, the settings of the selected determination are shown. Double-click the name of the implementing class to navigate to the class builder. Open the empty implementation of method /BOBF/IF_FRW_DETERMINATION~EXECUTE and provide the following source code.

```
METHOD /bobf/if_frw_determination~execute.
  DATA lt_item           TYPE        zdt_item. " Combined table type
  DATA lr_item           TYPE REF TO zds_item. " Combined structure
  DATA lv_unit_price     TYPE        snwd_unit_price.

  CONSTANTS lc_tax_rate TYPE p DECIMALS 2 VALUE '0.19'.

  CLEAR: et_failed_key, eo_message.

  " Retrieve the item data to be processed
  io_read->retrieve(
    EXPORTING
      iv_node       = zif_d_sales_quote_c=>sc_node-item
      it_key        = it_key
    IMPORTING
```

```
      et_data      = lt_item ).

  LOOP AT lt_item REFERENCE INTO lr_item WHERE product_id IS NOT INITIAL.
    " Read the unit price
    /bobf/cl_sepm_val_helper=>get_product_price(
      EXPORTING
        iv_product_id   = lr_item->product_id
      IMPORTING
        ev_unit_price   = lv_unit_price ).
    " Calculate gross amount
    lr_item->gross_amount = ( lv_unit_price * lr_item->quantity ) +
                            ( lv_unit_price * lr_item->quantity * lc_tax_rate ).
    lr_item->currency_code = /bobf/if_epm_soq_constants_ext=>sc_default_currency.
    " Perform update
    io_modify->update(
       EXPORTING
         iv_node           = zif_d_sales_quote_c=>sc_node-item
         iv_key            = lr_item->key
         is_data           = lr_item ).
  ENDLOOP.
ENDMETHOD.
```

Hint: If you copy the source code directly into the ABAP editor, the formatting will be lost. Copying it into WordPad, on the other hand, preserves at least the line breaks. This format can then be copied into the ABAP editor.

In most cases, the first step is to read the data of those node instances that need to be calculated. For this purpose, the RETRIEVE method of the Internal Access Object IO_READ can be used; it is provided by the framework as an importing parameter. The keys of the node instances that need to be processed and thus need to be read are handed over by another parameter called IT_KEY. To define the node from where we want to read data, we use the appropriate constant from the Constants Interface of the BO (remember that you have to use the specific Constance Interface of your business object). The result of the call is stored in a variable, typed with the Combined Table Type of the read node.

Now we work on the items and calculate the gross amount. Finally, we will update the access object IO_MODIFY to write the changes.

Activate your code and navigate back to the configuration view of the Business Object Builder.

Result

You are now finished with the implementation of the determination. The determination is always executed whenever the user changes the items. It determines the gross amount for each item.

TESTING THE DETERMINATION

We are now going to test the determination.

Procedure

Start the Business Object Test Environment

In the configuration view of BOB, choose the **Test** button in the toolbar.

Create an Instance of ZD_SALES_QUOTE

In the Node Instance Table pane, select Add Node Instance from the toolbar and enter the header data. Then navigate to the item node as described in the previous chapter. If you enter the product ID and the quantity now, the gross amount is calculated.

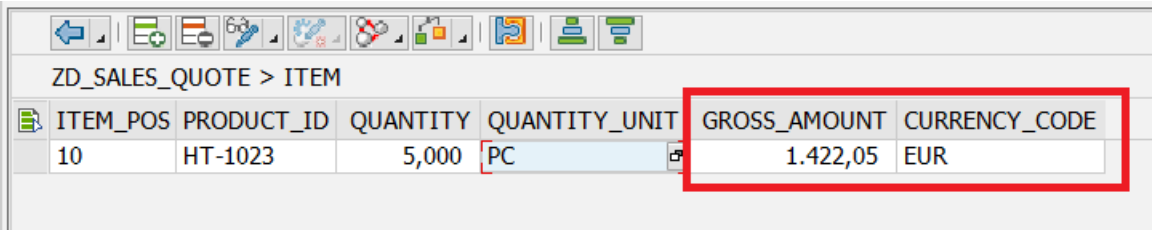

Fig. 15: Calculation of the cross amount

However, the user is able to change the calculated fields. In the next step, we will explain how to change that undesirable behavior.

Result

We have modeled, implemented, and tested the determination. But the result is not yet fully satisfactory. We have to continue and set attribute properties in the next step.

SETTING ATTRIBUTE PROPERTIES

BOPF supports the setting of attribute properties statically and dynamically. Since transaction BOB focuses on enhancement scenarios, the properties can only be set dynamically. To set the properties dynamically, create one single determination for the node.

Procedure

Start the Wizard to Create Determinations

Start the determination creation wizard on the item node, as described in the previous step. Enter the determination name **CALCULATE_PROPERTIES** and a description. Accept the implementing class name.

Define the Determination Pattern

We now select the pattern *Create Properties.* This pattern can only be chosen once for a node and takes care of the determination that is called in the phase where the framework determines the properties of the attributes.

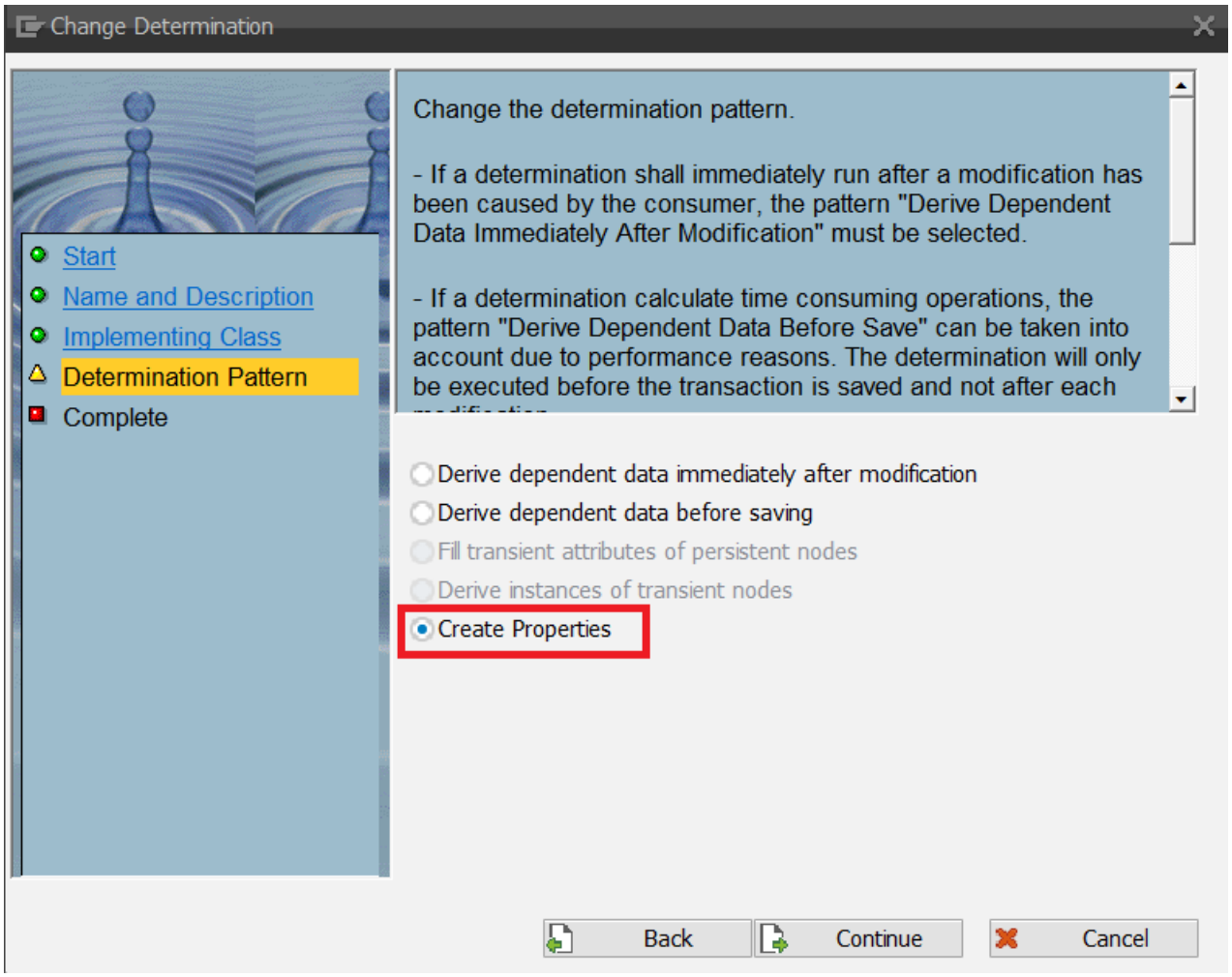

Fig. 16: Determination pattern for dynamic properties

Confirm the steps in the wizard.

Implement the Determination

The new determination is now visible in the Entity Browser pane of the BO configuration view. On the right, the settings of the selected determination are shown. Double-click the name of the implementing class to navigate to the class builder. Open the empty implementation of method */BOBF/IF_FRW_DETERMINATION~EXECUTE* and provide the following source code.

```
METHOD execute.
  FIELD-SYMBOLS <ls_key> LIKE LINE OF it_key.

  LOOP AT it_key ASSIGNING <ls_key>.
    " Set attribute GROSS_AMOUNT to read-only
    io_property->set_attribute_read_only(
       iv_key = <ls_key>-key
       iv_attribute_name = zif_d_sales_quote_c=>sc_node_attribute-item-gross_amount  ).
    " Set attribute CURRENCY_CODE to read-only
    io_property->set_attribute_read_only(
       iv_key = <ls_key>-key
       iv_attribute_name = zif_d_sales_quote_c=>sc_node_attribute-item-currency_code  ).
  ENDLOOP.
ENDMETHOD.
```

Hint: if you copy the source code directly into the ABAP editor, the formatting will be lost. Copying it into WordPad, on the other hand, preserves at least the line breaks. This format can then be copied into the ABAP editor.

In this method we are using the access object IO_PROPERTY provided as an importing parameter of the method to set the 'read only' property for ***GROSS_AMOUNT*** and ***CURRENCY_CODE***.

Activate your code and navigate back to the configuration view of the Business Object Builder.

Test the Scenario

Again call the test environment and create a root node instance. Navigate to the Items as described in the previous chapter.

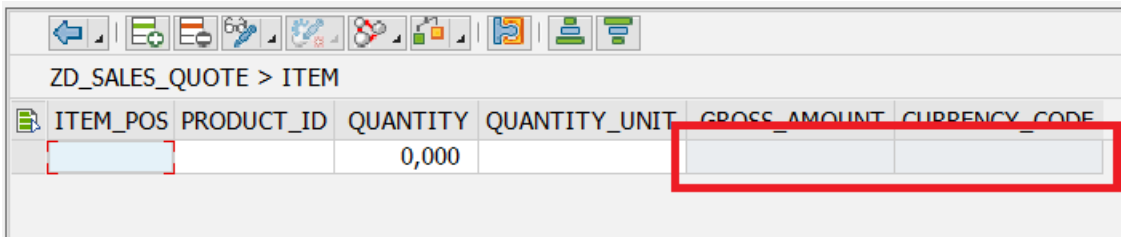

Fig. 17: Protected fields

Already after the creation of an item, GROSS_AMOUNT and CURRENCY_CODE are protected. Enter data and the gross amount will be calculated, the fields still being protected.

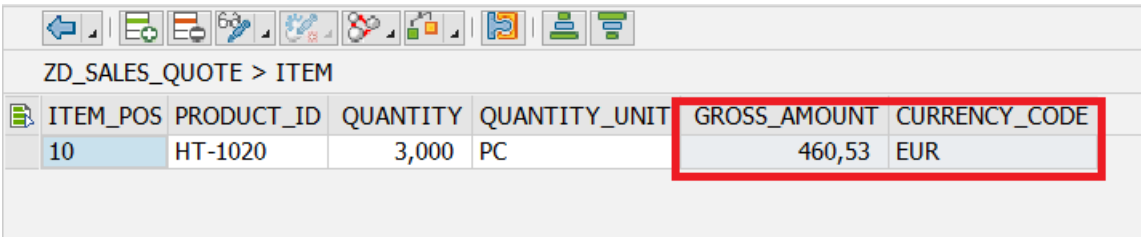

Fig. 18: determined values in protected fields

Result

Within a few minutes we have verified that we have implemented the determination and the property setting correctly, without writing any test code. Of course, this does not replace an automated test, but with the help of the Business Object Test Environment you get direct feedback as to whether your Business Object works correctly or not.

We have now gone through the fundamentals of creating and testing determinations. We hope you are motivated to try out more things with BOPF, as there is much more that can be discovered. Stay tuned for further articles about our framework.

www.sap.com