Bonus Determination Tutorial

Applies to:

Business Rules Framework plus shipped with SAP NetWeaver 7.0 Enhancement Package 1. For more information, visit the Business Process Expert homepage.

Summary

This tutorial shows how the bonus of an employee can be calculated. The bonus is calculated with the help of a decision table expression which has formula expressions nested in it.

Author: Shashi Kanth Kasam, Orenthung Ovung

Company: SAP Labs India

Created on: 12th August 2008

About the Authors

Shashi Kanth Kasam is a Rules Developer in the BRFplus team. He has 2 years of experience in building business rules and has been part of this team since April 2008.

Orenthung Ovung is an Information Developer in the BRFplus team. He has been part of this team since March 2008.

© 2008 SAP AG

Table of Contents

© 2008 SAP AG

Prerequisites

- You have a basic knowledge of BRFplus.
- You have gone through the *Hello World* tutorial and *Hello Kasam SDN*

Learning Objectives

- How to create data objects
- How to create formulas and nest them in the decision table expression
- How to create decision table expression

Designing Rules Using the API

Procedure

Data Declarations

```
DATA: lo_factory           TYPE REF TO if_fdt_factory,
      lo_perf_factor       TYPE REF TO if_fdt_case,
      lo_constant          TYPE REF TO if_fdt_constant,
      lo_function          TYPE REF TO if_fdt_function,
      lo_elem_salary       TYPE REF TO if_fdt_element,
      lo_elem_empid        TYPE REF TO if_fdt_element,
      lo_elem_role         TYPE REF TO if_fdt_element,
      lv_dev_id            TYPE if_fdt_types=>id,
      lv_sen_dev_id        TYPE if_fdt_types=>id,
      lv_dev_arch_id       TYPE if_fdt_types=>id,
      lv_mgr_id            TYPE if_fdt_types=>id,
      lo_elem_perf_rating  TYPE REF TO if_fdt_element,
      lv_poor_id           TYPE if_fdt_types=>id,
      lv_achiever_id       TYPE if_fdt_types=>id,
      lv_talent_id         TYPE if_fdt_types=>id,
      lo_elem_bonus        TYPE REF TO if_fdt_element,
      lts_object_id        TYPE if_fdt_types=>ts_object_id,
      lts_value_id         TYPE if_fdt_types=>ts_object_id,
      ls_when              TYPE if_fdt_case=>s_when,
      lts_when             TYPE if_fdt_case=>ts_when,
      lv_id                TYPE if_fdt_types=>id,
      lt_message           TYPE if_fdt_types=>t_message,
      lv_activation_failed TYPE abap_bool,
      lv_number            TYPE if_fdt_types=>element_number,
      lx_fdt               TYPE REF TO cx_fdt,
      lo_context           TYPE REF TO if_fdt_context,
      lo_result            TYPE REF TO if_fdt_result,
      lv_string            TYPE string,
      lo_formula_dev       TYPE REF TO if_fdt_formula,
      lo_formula_sen_dev   TYPE REF TO if_fdt_formula,
      lo_formula_dev_arch  TYPE REF TO if_fdt_formula,
      lo_formula_mgr       TYPE REF TO if_fdt_formula,
      lv_formula           TYPE string,
      lo_dectab            TYPE REF TO if_fdt_decision_table,
      ls_column            TYPE if_fdt_decision_table=>s_column,
      lts_column           TYPE if_fdt_decision_table=>ts_column,
      ls_cell              TYPE if_fdt_decision_table=>s_table_data,
      lts_cell             TYPE if_fdt_decision_table=>ts_table_data,
```


© 2008 SAP AG

```
      ls_amount            TYPE if_fdt_types=>element_amount,
      lo_data_object       TYPE REF TO if_fdt_data_object,
      lr_data              TYPE REF TO data,
      lv_name              TYPE if_fdt_types=>name.

FIELD-SYMBOLS: <ls_message>  TYPE if_fdt_types=>s_message,
               <ls_result>   TYPE any,
               <lv_payment>  TYPE any,
               <lv_currency> TYPE any.
```

Factory

The following instance can be used to create local objects that are not transported.

```
lo_factory = cl_fdt_factory=>if_fdt_factory~get_instance(
 if_fdt_constants=>gc_application_tmp ).
```

IF_FDT_CONSTANTS is the central constants interface. **GC_APPLICATION_TMP** is a local application which can be used for testing purposes and to create local objects that are not transported.

Creating New Objects

The factory created above can be used to create the objects needed for this tutorial.

```
lo_function ?= lo_factory->get_function( ).
lo_perf_factor ?= lo_factory->get_expression(
lo_perf_factor ?= lo_factory->get_expression(
iv_expression_type_id = if_fdt_constants=>gc_exty_case ).
```

Function

Set the following attributes for the function:

Before any changes are made to the object, it must be enqueued. The following lines of code here show the enqueuing of the function and its attributes setting.

```
lo_function ?= lo_factory->get_function( ).
lo_function->if_fdt_transaction~enqueue( ).
*lo_function->if_fdt_admin_data~set_name( 'BONUS_DETERMINATION').
lv_name = cl_fdt_services=>get_unique_name( ).
lo_function->if_fdt_admin_data~set_name( lv_name ).
lo_function>if_fdt_admin_data~set_texts(
 iv_text = 'Bonus Determination' ).   "#EC NOTEXT
```

The function needs a top expression. In this tutorial, the top expression is a decision table expression type.

Creating Data Objects

The input parameters provided by the application to determine the employee's bonus are monthly salary, user ID, role and performance rating. We will use the following code to create the data objects:

Monthly Salary

```
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'MONTHLY_SALARY'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_amount
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_salary ).
lo_elem_salary->if_fdt_admin_data~set_texts( iv_text = 'Monthly Salary' ).   "#EC
NOTEXT
```


© 2008 SAP AG

```
INSERT lo_elem_salary->mv_id INTO TABLE lts_object_id.
```

Employee ID

```
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'EMPLOYEE_ID'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_text
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_empid ).
lo_elem_empid->if_fdt_admin_data~set_texts( iv_text = 'Employee Id' ).   "#EC NOTEXT
INSERT lo_elem_empid->mv_id INTO TABLE lts_object_id.
```

Role

```
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'ROLE'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_text
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_role ).
lo_elem_role->if_fdt_admin_data~set_texts( iv_text = 'Role' ).   "#EC NOTEXT
INSERT lo_elem_role->mv_id INTO TABLE lts_object_id.
```

The *Role* data object does not have a check table or domain values. A number of allowed values have to be created for the data object.

A value list is created because a DDIC binding is not needed to get the domain values or values from a check table.

Use the following code to create the allowed values for the data object.

```
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Developer'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_dev_id ).
INSERT lv_dev_id INTO TABLE lts_value_id.

cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Senior Developer'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_sen_dev_id ).
INSERT lv_sen_dev_id INTO TABLE lts_value_id.

cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Development Architect'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_dev_arch_id ).
INSERT lv_dev_arch_id INTO TABLE lts_value_id.

cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Manager'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_mgr_id ).
INSERT lv_mgr_id INTO TABLE lts_value_id.
```


© 2008 SAP AG

```
lo_elem_role->set_value_list( lts_value_id ).
```

Using a value list by binding the elementary data object enables the domain or check table values to be reused. It eliminates the need to create constant expressions and set value lists explicitly.

Similarly, we need to create an elementary data object for the performance rating and set some values.

© 2008 SAP AG

Performance Rating

```
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'PERFORMANCE_RATING'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_text
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_perf_rating ).
lo_elem_perf_rating->if_fdt_admin_data~set_texts( iv_text = 'Performance Rating' ).
"#EC NOTEXT
INSERT lo_elem_perf_rating->mv_id INTO TABLE lts_object_id.
```

Create another value list as a DDIC binding is not needed to get the domain values or values from a check table.

```
CLEAR lts_value_id[].
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Exceeds expectations'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_talent_id ).
INSERT lv_talent_id INTO TABLE lts_value_id.

cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Meets expectations'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_achiever_id ).
INSERT lv_achiever_id INTO TABLE lts_value_id.

cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Does not meet expectations'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_poor_id ).
INSERT lv_poor_id INTO TABLE lts_value_id.

lo_elem_perf_rating->set_value_list( lts_value_id ).
```

Bonus

```
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'BONUS'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_amount
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_bonus ).
lo_elem_bonus->if_fdt_admin_data~set_texts( iv_text = 'Bonus Payment' ).   "#EC
NOTEXT
```


© 2008 SAP AG

Creating Case Expression

In the decision table, the result column for the bonus payment has formula expressions nested into the result cells. All other values in the result column are constants. The formula expressions perform simple mathematical calculations based on the context data objects and a case expression, *Performance Factor*. *Performance Factor* maps the performance rating into a performance factor.

CASE PERFORMANCE RATING

WHEN *Does Not Meet Expectations*	THEN *0.50*
WHEN *Meets Expectations*	THEN *1.00*
WHEN *Exceeds Expectations*	THEN *1.25*

When you pass the value as *0.50* to method **CREATE_CONSTANT** in **CL_FDT_CONVENIENCE** the internal type is a character type. A constant of type character will be created. You need to use a numeric variable to create a constant of elementary type number.

```
lo_perf_factor ?= lo_factory->get_expression(
  iv_expression_type_id = if_fdt_constants=>gc_exty_case ).
lo_perf_factor->if_fdt_transaction~enqueue( ).
lo_perf_factor->if_fdt_admin_data~set_name( 'PERFORMANCE_FACTOR').   "#EC NOTEXT
lo_perf_factor->if_fdt_admin_data~set_texts( iv_text = 'Performance Factor' ).   "#EC
NOTEXT
```

CASE PERFORMANCE RATING

```
lo_perf_factor->set_case_parameter( lo_elem_perf_rating->mv_id ).
*   WHEN 'Does not meet expectations' THEN '0.50'
ls_when-position       = 1.
ls_when-test_parameter = lv_poor_id.
lv_number              = '0.50'.       "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = lv_number
  IMPORTING ev_constant_id    = ls_when-return_parameter ).
INSERT ls_when INTO TABLE lts_when.

*   WHEN 'Meets expectations' THEN '1.00'
ls_when-position       = 2.
ls_when-test_parameter = lv_achiever_id.
lv_number              = '1.00'.       "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = lv_number
  IMPORTING ev_constant_id    = ls_when-return_parameter ).
INSERT ls_when INTO TABLE lts_when.

*   WHEN 'Exceeds expectations' THEN '1.25'
ls_when-position       = 3.
ls_when-test_parameter = lv_talent_id.
lv_number              = '1.25'.       "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = lv_number
```


© 2008 SAP AG

```
  IMPORTING ev_constant_id    = ls_when-return_parameter ).
INSERT ls_when INTO TABLE lts_when.
lo_perf_factor->set_when_table( lts_when ).

*   WHEN 'OTHERS' THEN '0'
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = '0'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_id ).
lo_perf_factor->set_other_parameter( lv_id ).
```

Creating Formula Expressions

Create a string for your formula that reuses data object and expression IDs. The only prerequisite is that the data object type or the expressions result type (which is itself a data object) should fit into the formula.

When you are unsure about the parsing of the formula string, you may call **GET_FORMULA** to check the parsing in the debugger.

The following formula expressions need to be created to calculate the bonus for the employees based on their monthly salary and their performance. The formula expressions will perform simple mathematical calculations based on the context data object, monthly salary and the case expression for the performance rating.

Formula for Developer

```
lo_formula_dev ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_dev->if_fdt_transaction~enqueue( ).
lo_formula_dev->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.09 *' lo_elem_salary->mv_id '*' lo_perf_factor->mv_id    "#EC NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_dev->set_formula( lv_formula ).
* check with the token table if the string was interpreted correctly
* DATA: lt_token type if_fdt_formula=>t_token.
*lo_formula_dev->get_formula(
*  IMPORTING ev_formula   = lv_formula
*            et_token     = lt_token ).
```

Formula for Senior Developer

```
lo_formula_sen_dev ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_sen_dev->if_fdt_transaction~enqueue( ).
lo_formula_sen_dev->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.12 *' lo_elem_salary->mv_id '*' lo_perf_factor->mv_id       "#EC
NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_sen_dev->set_formula( lv_formula ).
```

Formula for Architect

```
lo_formula_dev_arch ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_dev_arch->if_fdt_transaction~enqueue( ).
```


© 2008 SAP AG

```
lo_formula_dev_arch->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.15 *' lo_elem_salary->mv_id '*' lo_perf_factor->mv_id       "#EC
NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_dev_arch->set_formula( lv_formula ).
```

Formula for Manager

```
lo_formula_mgr ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_mgr->if_fdt_transaction~enqueue( ).
lo_formula_mgr->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.18 *' lo_elem_salary->mv_id '*' lo_perf_factor->mv_id       "#EC
NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_mgr->set_formula( lv_formula ).Creating a Decision Table
```

Creating a Decision Table

Use the following code to get a new decision table instance:

```
lo_dectab ?= lo_factory->get_expression(
  iv_expression_type_id = if_fdt_constants=>gc_exty_decision_table ).
lo_dectab->if_fdt_transaction~enqueue( ).
```

We will create a decision table as shown below:

Employee Id	Designation	Bonus
D034314		*25000USD*
	Developer	*Bonus For Developer*
	Senior Developer	*Bonus For Sr Dev*
	Development Architect	*Bonus For Architect*
	Manager	*Bonus For Manager*
		0 USD

The bonus column is mandatory. We use the elementary data objects of the context to define the columns. During processing the values will be taken from the context and passed to the ranges for the checks. In the case of the result column, the matched values will be returned as the result.

To define the condition columns,

```
ls_column-col_no        = 1.
ls_column-object_id     = lo_elem_empid->mv_id.
INSERT ls_column INTO TABLE lts_column.
ls_column-col_no        = 2.
ls_column-object_id     = lo_elem_role->mv_id.
INSERT ls_column INTO TABLE lts_column.
```


© 2008 SAP AG

To define the result column,

```
ls_column-col_no         = 3.
ls_column-object_id      = lo_elem_bonus->mv_id.
ls_column-input_required = abap_true.
ls_column-is_result      = abap_true.
INSERT ls_column INTO TABLE lts_column.

lo_dectab->set_columns( its_column = lts_column ).
```

Adding Rules to the Decision Table

Now we have the structure of the decision table. The next step consists of adding rules to it.

Row 1

```
ls_cell-row_no        = 1.
* conditions
ls_cell-col_no        = 1.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_empid->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low            = 'D034314'       "#EC NOTEXT
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_amount-number      = 25000.
ls_amount-currency    = 'USD'.   "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = ls_amount
  IMPORTING ev_constant_id    = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
```

Row 2

```
ls_cell-row_no        = 2.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_dev_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_dev->mv_id.
```


© 2008 SAP AG

```
INSERT ls_cell INTO TABLE lts_cell.
```

Row 3

```
ls_cell-row_no        = 3.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_sen_dev_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_sen_dev->mv_id.
INSERT ls_cell INTO TABLE lts_cell.
```

Row 4

```
ls_cell-row_no        = 4.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_dev_arch_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_dev_arch->mv_id.
INSERT ls_cell INTO TABLE lts_cell.
```

Row 5

```
ls_cell-row_no        = 5.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_mgr_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_mgr->mv_id.
INSERT ls_cell INTO TABLE lts_cell.
```


© 2008 SAP AG

Row 6

```
ls_cell-row_no        = 6.
* results
ls_cell-col_no        = 3.
ls_amount-number      = 0.
ls_amount-currency    = 'USD'.   "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = ls_amount
  IMPORTING ev_constant_id    = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.

lo_dectab->set_table_data( its_data = lts_cell ).
```

The default is set to return the first match and there is no need to set any table properties.

Setting the Function's Context Data Objects

Use the following code to assign the context data object:

```
lo_function->set_context_data_objects( lts_object_id ).
```

Assigning the Case Expression as the Top Expression of the Function

Use the following code to assign the top expression to the function:

```
lo_function->set_expression( lo_dectab->mv_id ).
lo_function->set_result_data_object( lo_elem_bonus->mv_id ).
```

Activating the Function

Activate and save the changes. Only activated changes are relevant for processing. However, active memory states can also be processed without saving it to the DB.

```
lo_function->if_fdt_transaction~activate(
  EXPORTING iv_deep              = abap_true
  IMPORTING et_message           = lt_message
            ev_activation_failed = lv_activation_failed ).
Try.
   lo_function->if_fdt_transaction~save( iv_deep = ABAP_true ).
   Commit work.
Catch cx_fdt into lx_fdt.
   Rollback work.
ENDTRY.

lo_function->if_fdt_transaction~dequeue( iv_deep = abap_true )..
```

Error Handling

```
IF lv_boolean EQ abap_true.
  LOOP AT lt_message ASSIGNING <ls_message>.
    WRITE / <ls_message>-text.
  ENDLOOP.
  RETURN. ">>>
ENDIF.
```


© 2008 SAP AG

```
TRY.
*   hand over some values into the context
    ls_amount-number   = '3800'.   "#EC NOTEXT
    ls_amount-currency = 'USD'. "needs customized currency conversion   "#EC NOTEXT
    lo_context = lo_function->get_process_context( ).
    lo_context->set_value( iv_name  = 'MONTHLY_SALARY'   "#EC NOTEXT
                           ia_value = ls_amount ).
    lo_context->set_value( iv_name  = 'EMPLOYEE_ID'   "#EC NOTEXT
                           ia_value = 'D034314' ).   "#EC NOTEXT
    lo_context->set_value( iv_name  = 'ROLE'   "#EC NOTEXT
                           ia_value = 'Developer' ).   "#EC NOTEXT
    lo_context->set_value( iv_name  = 'PERFORMANCE_RATING'   "#EC NOTEXT
                           ia_value = 'Exceeds expectations' ).   "#EC NOTEXT
*   process the function
    lo_function->process( EXPORTING io_context = lo_context
                          IMPORTING eo_result  = lo_result ).
*   get the result
    lo_data_object = lo_result->get_data_object( ).
    lo_data_object->create_data_reference( IMPORTING er_data = lr_data ).
    ASSIGN lr_data->* TO <ls_result>.
    lo_result->get_value( IMPORTING ea_value = <ls_result> ).
    ASSIGN COMPONENT:
      'NUMBER'   OF STRUCTURE <ls_result> TO <lv_payment>,
      'CURRENCY' OF STRUCTURE <ls_result> TO <lv_currency>.
    WRITE: 'Bonus Payment: ', <lv_payment>,<lv_currency>.    "#EC NOTEXT
  CATCH cx_fdt INTO lx_fdt.
*   error handling
    LOOP AT lx_fdt->mt_message ASSIGNING <ls_message>.
      WRITE / <ls_message>-text.
    ENDLOOP.
ENDTRY.
```


© 2008 SAP AG

The Complete Report

```
*&---------------------------------------------------------------------*
*& Report  FDT_SDN_BONUS_DETERMINATION
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT  FDT_SDN_BONUS_DETERMINATION.
* Please check out FDT_TUTORIAL_HELLO_WORLD and FDT_TUTORIAL_HELLO_IWONA
* before you have a look at this tutorial. Several explanations
* (redundant) are left out here but can be found in the other two
* tutorials.

DATA: lo_factory           TYPE REF TO if_fdt_factory,
      lo_perf_factor       TYPE REF TO if_fdt_case,
      lo_constant          TYPE REF TO if_fdt_constant,
      lo_function          TYPE REF TO if_fdt_function,
      lo_elem_salary       TYPE REF TO if_fdt_element,
      lo_elem_empid        TYPE REF TO if_fdt_element,
      lo_elem_role         TYPE REF TO if_fdt_element,
      lv_dev_id            TYPE if_fdt_types=>id,
      lv_sen_dev_id        TYPE if_fdt_types=>id,
      lv_dev_arch_id       TYPE if_fdt_types=>id,
      lv_mgr_id            TYPE if_fdt_types=>id,
      lo_elem_perf_rating  TYPE REF TO if_fdt_element,
      lv_poor_id           TYPE if_fdt_types=>id,
      lv_achiever_id       TYPE if_fdt_types=>id,
      lv_talent_id         TYPE if_fdt_types=>id,
      lo_elem_bonus        TYPE REF TO if_fdt_element,
      lts_object_id        TYPE if_fdt_types=>ts_object_id,
      lts_value_id         TYPE if_fdt_types=>ts_object_id,
      ls_when              TYPE if_fdt_case=>s_when,
      lts_when             TYPE if_fdt_case=>ts_when,
      lv_id                TYPE if_fdt_types=>id,
      lt_message           TYPE if_fdt_types=>t_message,
      lv_activation_failed TYPE abap_bool,
      lv_number            TYPE if_fdt_types=>element_number,
      lx_fdt               TYPE REF TO cx_fdt,
      lo_context           TYPE REF TO if_fdt_context,
      lo_result            TYPE REF TO if_fdt_result,
      lv_string            TYPE string,
      lo_formula_dev       TYPE REF TO if_fdt_formula,
      lo_formula_sen_dev   TYPE REF TO if_fdt_formula,
      lo_formula_dev_arch  TYPE REF TO if_fdt_formula,
      lo_formula_mgr       TYPE REF TO if_fdt_formula,
      lv_formula           TYPE string,
      lo_dectab            TYPE REF TO if_fdt_decision_table,
      ls_column            TYPE if_fdt_decision_table=>s_column,
      lts_column           TYPE if_fdt_decision_table=>ts_column,
      ls_cell              TYPE if_fdt_decision_table=>s_table_data,
      lts_cell             TYPE if_fdt_decision_table=>ts_table_data,
      ls_amount            TYPE if_fdt_types=>element_amount,
      lo_data_object       TYPE REF TO if_fdt_data_object,
```


© 2008 SAP AG

```
      lr_data              TYPE REF TO data,
      lv_name              TYPE if_fdt_types=>name.

FIELD-SYMBOLS: <ls_message>  TYPE if_fdt_types=>s_message,
               <ls_result>   TYPE any,
               <lv_payment>  TYPE any,
               <lv_currency> TYPE any.

lo_factory = cl_fdt_factory=>if_fdt_factory~get_instance(
 if_fdt_constants=>gc_application_tmp ).

*** Function: BONUS_DETERMINATION - Part 1 ... ************************
lo_function ?= lo_factory->get_function( ).
lo_function->if_fdt_transaction~enqueue( ).
*lo_function->if_fdt_admin_data~set_name( 'BONUS_DETERMINATION').
lv_name = cl_fdt_services=>get_unique_name( ).
lo_function->if_fdt_admin_data~set_name( lv_name ).
lo_function-
>if_fdt_admin_data~set_texts( iv_text = 'Bonus Determination' ).   "#EC NOTEXT

*** Data object with Type Element: Monthly Salary *********************
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'MONTHLY_SALARY'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_amount
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_salary ).
lo_elem_salary-
>if_fdt_admin_data~set_texts( iv_text = 'Monthly Salary' ).   "#EC NOTEXT
INSERT lo_elem_salary->mv_id INTO TABLE lts_object_id.

*** Data object with Type Element: User ID *****************************
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'EMPLOYEE_ID'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_text
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_empid ).
lo_elem_empid->if_fdt_admin_data~set_texts( iv_text = 'Employee Id' ).   "#EC NOTEXT
INSERT lo_elem_empid->mv_id INTO TABLE lts_object_id.

*** Data object with Type Element: Role ********************************
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'ROLE'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_text
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_role ).
lo_elem_role->if_fdt_admin_data~set_texts( iv_text = 'Role' ).   "#EC NOTEXT
INSERT lo_elem_role->mv_id INTO TABLE lts_object_id.
* create a value list because we do not have a DDIC binding to get
* domain values or values from a check table
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Developer'   "#EC NOTEXT
```


© 2008 SAP AG

```
  IMPORTING ev_constant_id    = lv_dev_id ).
INSERT lv_dev_id INTO TABLE lts_value_id.
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Senior Developer'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_sen_dev_id ).
INSERT lv_sen_dev_id INTO TABLE lts_value_id.
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Development Architect'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_dev_arch_id ).
INSERT lv_dev_arch_id INTO TABLE lts_value_id.
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Manager'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_mgr_id ).
INSERT lv_mgr_id INTO TABLE lts_value_id.
lo_elem_role->set_value_list( lts_value_id ).

*** Data object with Type Element: Performance Rating ******************
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'PERFORMANCE_RATING'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_text
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_perf_rating ).
lo_elem_perf_rating-
>if_fdt_admin_data~set_texts( iv_text = 'Performance Rating' ).   "#EC NOTEXT
INSERT lo_elem_perf_rating->mv_id INTO TABLE lts_object_id.
* create a value list because we do not have a DDIC binding to get
* domain values or values from a check table
CLEAR lts_value_id[].
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Exceeds expectations'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_talent_id ).
INSERT lv_talent_id INTO TABLE lts_value_id.
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Meets expectations'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_achiever_id ).
INSERT lv_achiever_id INTO TABLE lts_value_id.
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = 'Does not meet expectations'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_poor_id ).
INSERT lv_poor_id INTO TABLE lts_value_id.

lo_elem_perf_rating->set_value_list( lts_value_id ).
```


© 2008 SAP AG

```
*** Data object with Type Element: Bonus Payment ***********************
cl_fdt_convenience=>create_element(
  EXPORTING iv_name           = 'BONUS'   "#EC NOTEXT
            iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_element_type   = if_fdt_constants=>gc_element_type_amount
            iv_activate       = abap_false
  IMPORTING eo_element        = lo_elem_bonus ).
lo_elem_bonus-
>if_fdt_admin_data~set_texts( iv_text = 'Bonus Payment' ).   "#EC NOTEXT

*** Case: PERFORMANCE FACTOR ****************************************
*define the case expression Performance Rating -> Performance Factor
* CASE PERFORMANCE RATING
*   WHEN 'Does Not Meet Expectations'        THEN '0.50'
*   WHEN 'Meets Expectations'                THEN '1.00'
*   WHEN 'Exceeds Expectations'              THEN '1.25'

lo_perf_factor ?= lo_factory->get_expression(
  iv_expression_type_id = if_fdt_constants=>gc_exty_case ).
lo_perf_factor->if_fdt_transaction~enqueue( ).
lo_perf_factor->if_fdt_admin_data~set_name( 'PERFORMANCE_FACTOR').   "#EC NOTEXT
lo_perf_factor-
>if_fdt_admin_data~set_texts( iv_text = 'Performance Factor' ).   "#EC NOTEXT

* CASE PERFORMANCE RATING
lo_perf_factor->set_case_parameter( lo_elem_perf_rating->mv_id ).
*   WHEN 'Does not meet expectations' THEN '0.50'
ls_when-position       = 1.
ls_when-test_parameter = lv_poor_id.
lv_number              = '0.50'.       "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = lv_number
  IMPORTING ev_constant_id    = ls_when-return_parameter ).
INSERT ls_when INTO TABLE lts_when.

*   WHEN 'Meets expectations' THEN '1.00'
ls_when-position       = 2.
ls_when-test_parameter = lv_achiever_id.
lv_number              = '1.00'.       "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = lv_number
  IMPORTING ev_constant_id    = ls_when-return_parameter ).
INSERT ls_when INTO TABLE lts_when.

*   WHEN 'Exceeds expectations' THEN '1.25'
ls_when-position       = 3.
ls_when-test_parameter = lv_talent_id.
lv_number              = '1.25'.       "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = lv_number
```


© 2008 SAP AG

```
  IMPORTING ev_constant_id    = ls_when-return_parameter ).
INSERT ls_when INTO TABLE lts_when.
lo_perf_factor->set_when_table( lts_when ).

*   WHEN 'OTHERS' THEN '0'
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = '0'   "#EC NOTEXT
  IMPORTING ev_constant_id    = lv_id ).
lo_perf_factor->set_other_parameter( lv_id ).

*** Formula for Dev: 1.09 * Monthly Salary * Performance Factor ********
lo_formula_dev ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_dev->if_fdt_transaction~enqueue( ).
lo_formula_dev->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.09 *' lo_elem_salary->mv_id '*' lo_perf_factor->mv_id    "#EC NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_dev->set_formula( lv_formula ).
* check with the token table if the string was interpreted correctly
* DATA: lt_token type if_fdt_formula=>t_token.
*lo_formula_dev->get_formula(
*  IMPORTING ev_formula   = lv_formula
*            et_token     = lt_token ).

*** Formula for SenDev: 1.1 * Monthly Salary * Performance Factor ******
lo_formula_sen_dev ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_sen_dev->if_fdt_transaction~enqueue( ).
lo_formula_sen_dev->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.12 *' lo_elem_salary->mv_id '*' lo_perf_factor-
>mv_id       "#EC NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_sen_dev->set_formula( lv_formula ).

*** Formula for DevArch: 1.15 * Monthly Salary * Performance Factor ****
lo_formula_dev_arch ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_dev_arch->if_fdt_transaction~enqueue( ).
lo_formula_dev_arch->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.15 *' lo_elem_salary->mv_id '*' lo_perf_factor-
>mv_id       "#EC NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_dev_arch->set_formula( lv_formula ).

*** Formula for CDA: 1.18 * Monthly Salary * Performance Factor ********
lo_formula_mgr ?= lo_factory->get_expression(
      iv_expression_type_id = if_fdt_constants=>gc_exty_formula ).
lo_formula_mgr->if_fdt_transaction~enqueue( ).
lo_formula_mgr->if_fdt_expression~set_result_data_object(
  if_fdt_constants=>gc_dobj_element_amount ).
CONCATENATE '1.18 *' lo_elem_salary->mv_id '*' lo_perf_factor-
```


© 2008 SAP AG

```
>mv_id       "#EC NOTEXT
  INTO lv_formula SEPARATED BY space.
lo_formula_mgr->set_formula( lv_formula ).

*** Decision Table ****************************************************
lo_dectab ?= lo_factory->get_expression(
  iv_expression_type_id = if_fdt_constants=>gc_exty_decision_table ).
lo_dectab->if_fdt_transaction~enqueue( ).
lo_dectab->if_fdt_admin_data~set_name( 'Bonus_DT').
lo_dectab->if_fdt_admin_data~set_texts( iv_text = 'Bonus DT' ).

* define the condition columns
ls_column-col_no        = 1.
ls_column-object_id     = lo_elem_empid->mv_id.
INSERT ls_column INTO TABLE lts_column.
ls_column-col_no        = 2.
ls_column-object_id     = lo_elem_role->mv_id.
INSERT ls_column INTO TABLE lts_column.

* define the result columns
ls_column-col_no         = 3.
ls_column-object_id      = lo_elem_bonus->mv_id.
ls_column-input_required = abap_true.
ls_column-is_result      = abap_true.
INSERT ls_column INTO TABLE lts_column.

lo_dectab->set_columns( its_column = lts_column ).

*create the table content for row 1
ls_cell-row_no        = 1.
* conditions
ls_cell-col_no        = 1.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_empid->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low            = 'D034314'       "#EC NOTEXT
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_amount-number      = 2500.
ls_amount-currency    = 'USD'.   "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = ls_amount
  IMPORTING ev_constant_id    = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.

*create the table content for row 2
ls_cell-row_no        = 2.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
```


© 2008 SAP AG

```
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_dev_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_dev->mv_id.
INSERT ls_cell INTO TABLE lts_cell.

*create the table content for row 3
ls_cell-row_no        = 3.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_sen_dev_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_sen_dev->mv_id.
INSERT ls_cell INTO TABLE lts_cell.

*create the table content for row 4
ls_cell-row_no        = 4.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_dev_arch_id
            iv_activate       = abap_false
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_dev_arch->mv_id.
INSERT ls_cell INTO TABLE lts_cell.

*create the table content for row 5
ls_cell-row_no        = 5.
* conditions
ls_cell-col_no        = 2.
cl_fdt_convenience=>create_simple_range(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_test_parameter = lo_elem_role->mv_id
            iv_option         = if_fdt_range=>gc_option_equal
            iv_low_id          = lv_mgr_id
            iv_activate       = abap_false
```


© 2008 SAP AG

```
  IMPORTING ev_range_id       = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.
* results
ls_cell-col_no        = 3.
ls_cell-expression_id = lo_formula_mgr->mv_id.
INSERT ls_cell INTO TABLE lts_cell.

*create the table content for row 6
ls_cell-row_no        = 6.
* results
ls_cell-col_no        = 3.
ls_amount-number      = 0.
ls_amount-currency    = 'USD'.   "#EC NOTEXT
cl_fdt_convenience=>create_constant(
  EXPORTING iv_application_id = if_fdt_constants=>gc_application_tmp
            iv_activate       = abap_false
            ia_value          = ls_amount
  IMPORTING ev_constant_id    = ls_cell-expression_id ).
INSERT ls_cell INTO TABLE lts_cell.

lo_dectab->set_table_data( its_data = lts_cell ).
* default is first match, no need to set any further table properties

*** Function: BONUS_DETERMINATION - ... Part 2 ************************
* set the function's context data objects
lo_function->set_context_data_objects( lts_object_id ).
* make the case expression the top expression of the function
lo_function->set_expression( lo_dectab->mv_id ).
lo_function->set_result_data_object( lo_elem_bonus->mv_id ).

* activate and save the changes; only activated changes are relevant
* for processing; however, active memory states can also be processed
* without a need to save to DB
lo_function->if_fdt_transaction~activate(
  EXPORTING iv_deep              = abap_true
  IMPORTING et_message           = lt_message
            ev_activation_failed = lv_activation_failed ).
Try.
   lo_function->if_fdt_transaction~save( iv_deep = ABAP_true ).
   Commit work.
Catch cx_fdt into lx_fdt.
   Rollback work.
ENDTRY.

lo_function->if_fdt_transaction~dequeue( iv_deep = abap_true ).

* error handling
IF lv_activation_failed EQ abap_true.
  LOOP AT lt_message ASSIGNING <ls_message>.
    WRITE / <ls_message>-text.
  ENDLOOP.
  RETURN. ">>>
ENDIF.

TRY.
*   hand over some values into the context
```


© 2008 SAP AG

```
    ls_amount-number   = '3800'.   "#EC NOTEXT
    ls_amount-currency = 'USD'. "needs customized currency conversion   "#EC NOTEXT
    lo_context = lo_function->get_process_context( ).
    lo_context->set_value( iv_name  = 'MONTHLY_SALARY'   "#EC NOTEXT
                           ia_value = ls_amount ).
    lo_context->set_value( iv_name  = 'EMPLOYEE_ID'   "#EC NOTEXT
                           ia_value = 'D034314' ).   "#EC NOTEXT
    lo_context->set_value( iv_name  = 'ROLE'   "#EC NOTEXT
                           ia_value = 'Developer' ).   "#EC NOTEXT
    lo_context->set_value( iv_name  = 'PERFORMANCE_RATING'   "#EC NOTEXT
                           ia_value = 'Exceeds expectations' ).   "#EC NOTEXT
*   process the function
    lo_function->process( EXPORTING io_context = lo_context
                          IMPORTING eo_result  = lo_result ).
*   get the result
    lo_data_object = lo_result->get_data_object( ).
    lo_data_object->create_data_reference( IMPORTING er_data = lr_data ).
    ASSIGN lr_data->* TO <ls_result>.
    lo_result->get_value( IMPORTING ea_value = <ls_result> ).
    ASSIGN COMPONENT:
      'NUMBER'   OF STRUCTURE <ls_result> TO <lv_payment>,
      'CURRENCY' OF STRUCTURE <ls_result> TO <lv_currency>.
    WRITE: 'Bonus Payment: ', <lv_payment>,<lv_currency>.    "#EC NOTEXT
  CATCH cx_fdt INTO lx_fdt.
*   error handling
    LOOP AT lx_fdt->mt_message ASSIGNING <ls_message>.
      WRITE / <ls_message>-text.
    ENDLOOP.
ENDTRY.
```


© 2008 SAP AG

.Defining Rules Using the UI

Creating the Application

1. In the menu bar, choose *Workbench -> Create Application...*
2. In the *Object Creation* dialog box that appears, enter **Zore_Bonus_Det** in the *Name* field.
3. Choose *System* as the storage type and select the *Create Local Application* check box.
4. Enter **$TMP** in the *Development Package* field.
5. Choose *Create & Navigate To Object*.
6. Save the application.

Creating Context Data Objects of Element Type

1. Under the *Assigned Objects* tab, choose *Data Object* from the *Type* field and choose *Create Object*.
2. In the *Object Creation* dialog box that appears, choose *Element* from the *Type* field.

   The *Element* section appears.
3. Choose *Amount* in the *Element Type* field under the *Define Element Properties* section.
4. Under the *General Data* section, enter **Bonus** in the *Name* field and **Bonus** in the *Short Text* field and choose *Create*.

Similarly, create the following element data objects.

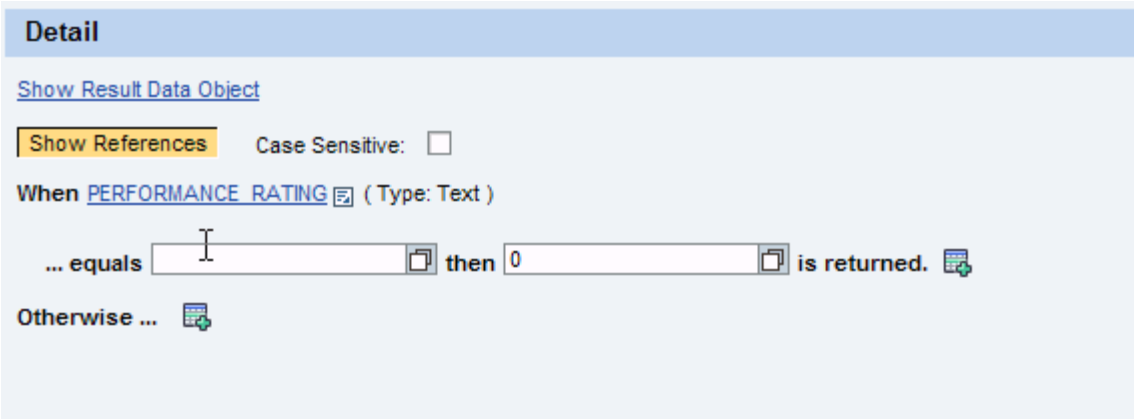
Name of the Data Object	Short Text	Type
Employee_ID	**Employee ID**	*Text*
Monthly_Salary	**Monthly Salary**	*Amount*
Performance_Rating	**Performance Rating**	*Text*
Role	**Designation**	*Text*

Creating a Case Expression

1. Under the *Assigned Objects* tab, choose *Expression* and choose *Create Object*.
2. In the *Object Creation* dialog box that appears, enter **Performance_Factor** in the *Name* field.
3. Choose *Case* from the *Type* field.
4. Choose *Create & Navigate to Object*.

   The *PERFORMANCE_FACTOR* case expression appears in the *Object Manager* panel.
5. Define the case parameter by choosing (*Graphical Access*).
6. In the context menu, choose *Select Context Data Object -> Select Context Data Object…*
7. In the *Context Query* dialog box that appears, select *PERFORMANCE_RATING.*
8. Choose (*Graphical Access*) under the *Result Data Object* section.
9. In the context menu, choose *Default Objects →Number*.
10. Add values to the case expression. Choose *Direct Value Input.*

© 2008 SAP AG



11. Enter **Exceeds Expectations** and **1,25**
12. To add another line, choose .
13. Similarly, add the following values to the case expression:

Enter **Meets Expectations**	Enter **1**
Enter **Does not Meet Expectations**	Enter **0,5**

14. Save the expression.

Creating Formula Expressions

1. Choose *Back* button.
2. Under the *Assigned Objects* tab, choose *Expression* and choose *Create Object*.
3. In the *Object Creation* dialog box that appears, enter **Bonus_for_developer** in the *Name* field.
4. Enter **Bonus for developer** in the *Short Text* field.
5. Choose *Formula* from the *Type* field.
6. Choose *Create & Navigate to Object*.

   You are navigated to the formula builder page.
7. Assign a result data object by choosing (*Graphical Access*).
8. In the context menu, choose *Select Data Object*
9. In the *Object Query* dialog box that appears, select *BONUS* and choose *Select.*
10. Choose *Number*.
11. In the *Insert* dialog box that appears, enter **1,09**.
12. Choose the * button.
13. In the context menu of the formula field, choose *Insert Data Object.*
14. In the *Object Query* dialog box that appears, select *MONTHLY_SALARY* and choose *Select*.
15. Choose the * button.
16. In the context menu of the formula field, choose *Insert Expression -> Select.*
17. In the *Object Query* dialog box that appears, select *PERFORMANCE_FACTOR* and choose *Select*.
18. Save the expression.

© 2008 SAP AG

Use the above procedure to create the following formula expressions:

Name of the Formula Expression	Result Data Object	Formula
Bonus_for_Senior_ Developer	BONUS	**1.1*MONTHLY_SALARY*PERFORMANCE_FACTOR**
Bonus_for_Architect	BONUS	**1.15*MONTHLY_SALARY*PERFORMANCE_FACTOR**
Bonus_ for_Manager	BONUS	**1.2*MONTHLY_SALARY*PERFORMANCE_FACTOR**

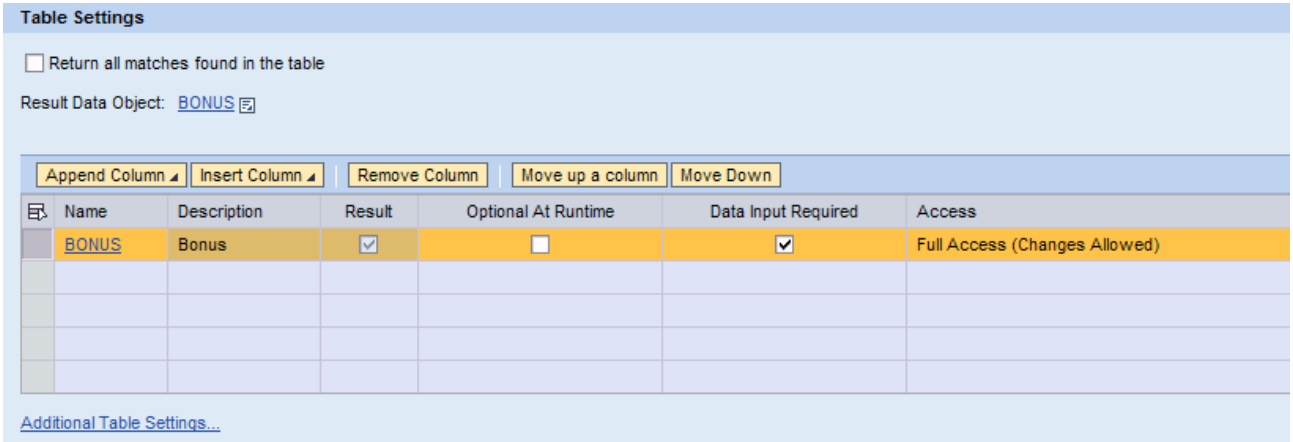
Creating a Decision Table

1. Choose *Back* button.
2. Under the *Assigned Objects* tab, choose *Expression* and choose *Create Object*.
3. In the *Object Creation* dialog box that appears, enter **Bonus_DT** in the *Name* field.
4. Choose *Decision Table* from the *Type* field.
5. Choose *Create & Navigate to Object*.

The *BONUS_DT* decision table expression appears in the *Object Manager* panel.

Adding Result to the Decision Table

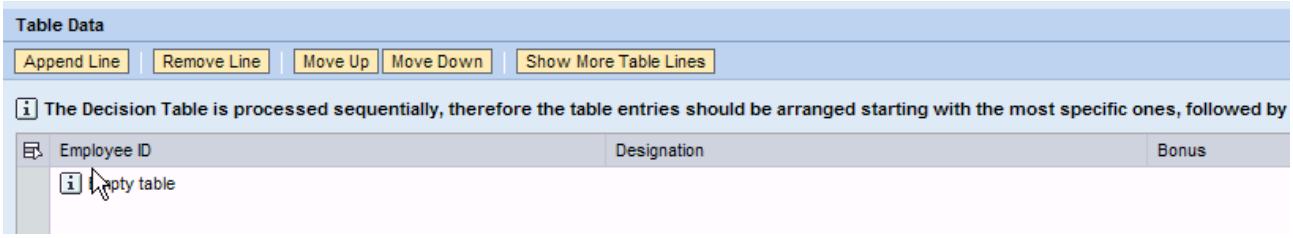
1. Choose (*Graphical Access*) beside *Result Data Object*.
2. In the context menu, choose *Select Result Data Objects -> BONUS*.
3. You should see the data object *BONUS* appended to the decision table as the result data object as shown below.



Adding Context to the Decision Table

1. In the *Detail* section, under *Table Settings* choose *Insert Column -> From Function Context…*
2. In the *Object Query* dialog box that appears, select *EMPLOYEE_ID* data object and choose *Select.*
3. Similarly, add *ROLE* data objects as context to the decision table. You should see the column headers of the decision table under the *Table Data* section.

© 2008 SAP AG

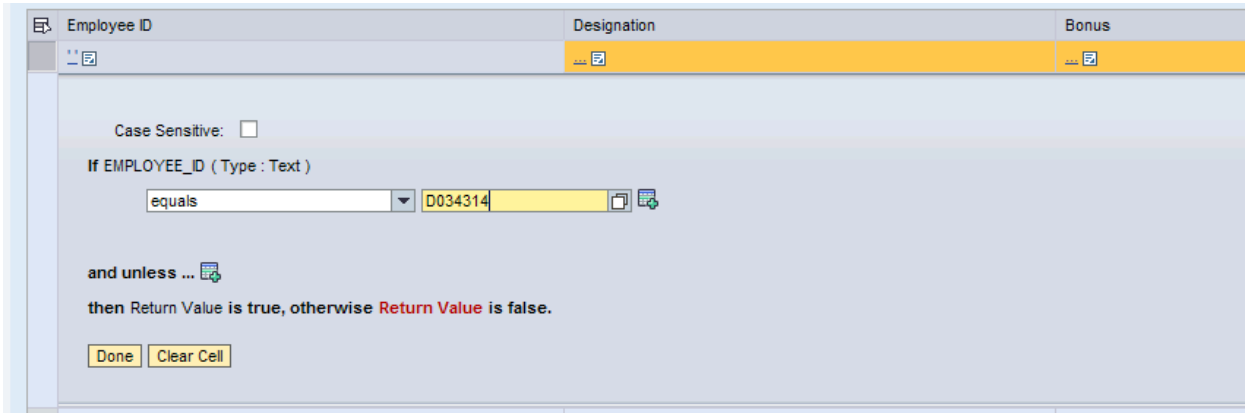


> **Note:** When you create a data object, the texts that you enter under *Text* fields appear as column headers in the decision table.

Adding Values to the Decision Table

1. Under *Table Data*, choose *Append Line*.
2. Under the *Employee ID* column, choose (*Graphical Access*).
3. In the context menu, choose *Direct Value Input*.

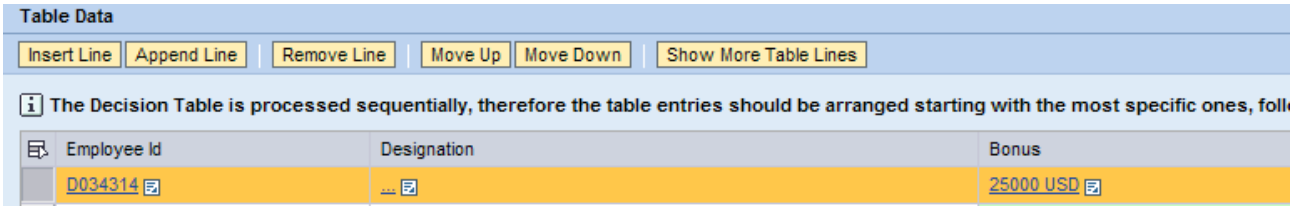
   The data object opens.
4. Enter **D034314** as shown below.



5. Choose *Done.*
6. Under the *Bonus Payment* column, choose (*Graphical Access*).
7. In the context menu, choose *Direct Value Input*.

   The data object opens.
8. Enter **25000** as value, **USD** as currency and choose *Done*.

   You should see the first line of the decision table filled with the values as shown below.



9. Under *Table Data*, choose *Append Line*.
10. Under the *Role* column, choose (*Graphical Access*).
11. In the context menu, choose *Direct Value Input*.

    The data object opens.
12. Choose *equals* and enter **Developer** and choose *Done*.

© 2008 SAP AG

13. Under the *Bonus* column, choose (*Graphical Access*).
14. In the context menu, choose *Select Existing Expressions -> BONUS_FOR_DEVELOPER*.
15. Create the following table lines using the above procedure.

Employee ID	Designation	Bonus
	Senior Developer	Bonus for Senior Developer
	Development Architect	Bonus for Architect
	Manager	Bonus for Manager
		Enter value as **0** and **USD** as currency

© 2008 SAP AG

Creating the Function

1. In the *Object Manager* panel, under the *Detail* section, choose the *Assigned Objects* tab.

   The *Assigned Objects* tab page appears.

2. Choose *Function* from the *Type* field and choose *Create Object*.
3. In the *Object Creation* dialog box that appears, enter **Calculate_bonus_for_employee** in the *Name* field and choose *Create & Navigate To Object*.

   The function is created and appears under the *Object Manager* panel.

4. Add context and result to the function.
   a) Under the *Detail* section, choose the *Signature* tab.

      The *Signature* tab page appears.

   b) Under *Context*, choose *Add Existing Data Object*.
   c) In the *Object Query* dialog box that appears, choose *EMPLOYEE_ID*, *MONTHLY_SALARY*, *PERFORMANCE_RATING* and *ROLE* and choose *Select*.
   d) Under *Result Data Object*, choose (*Graphical Access*).
   e) In the context menu, choose *Select*.
   f) In the *Object Query* dialog box that appears, choose *BONUS*.
5. Assign a decision table as a top expression to the function.
   a. Choose the *Properties* tab.
   b. In the *Detail* section, under *Properties*, choose (*Graphical Access*) next to the *Top Expression* field.
   c. In the context menu, choose *Select.*
   d. In the *Object Query* dialog box that appears, select *BONUS_DT* and choose *Select*.
6. Save the function.

Activating the Application and the Function

1. Click the name of the application (**Zore_Bonus_Det**).
2. Choose *Activate* button.
3. In the *Confirmation of Activation* dialog box that appears, choose *OK*.

   The application gets activated.

4. In the *Object Manager* panel, under the *Detail* section, choose the *Assigned Objects* tab.
5. In the *Assigned Objects* tab page, choose *CALCULATE_BONUS_FOR_EMPLOYEE*.

   The function opens in the *Object Manager* panel.

6. Choose *Activate* button.
7. In the *Confirmation of Activation* dialog box that appears, activate the *Include Referenced Objects* checkbox and choose *OK*.

   The *CALCULATE_BONUS_FOR_EMPLOYEE* function gets activated.

© 2008 SAP AG

Simulating the Function

1. Choose *Tools* -> *Simulation*. The simulation page is displayed.
2. Choose *Select Function* button.
3. In the *Object Query* dialog box that appears, enter **Calculate_Bonus_for_employee** in the *Name* field and **Zore_bonus_det** in the *Application Name* field. Choose *Search*.
4. The *CALCULATE_BONUS_FOR_EMPLOYEE* function appears in the table. Select the function and choose *Select*.
5. In the *Simulation* dialog box that appears, enter the following input parameters:
   - **Z045669** in the *Employee ID* field
   - **1,500** in the *Monthly Salary* field and **USD** in the *Monthly Salary (Currency)* field
   - **Meets expectations** in the *Performance Rating* field
   - **Developer** in the *Designation* field
6. Under *Simulation Mode* section, choose *Show only Result* and choose *Run Simulation*.
7. The bonus amount to be paid appears under the *Result* section.

© 2008 SAP AG

Related Content

- Business Rules Framework plus – The Very Basics
- Wikipedia, Business Rules: http://en.wikipedia.org/wiki/Business_rules
- Wikipedia, Business Rule Management System: http://en.wikipedia.org/wiki/Business_Rule_Management_System
- Carsten Ziegler, About Business Rules: https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/9713
- Carsten Ziegler, BRFplus a Business Rule Engine written in ABAP, https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/8889
- Carsten Ziegler, Important Information for Using BRFplus

  https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/11632
- Rajagopalan Narayanan, Business Rules and Software Requirements, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/70c669d8-3ac2-2a10-0e96-c7c3786168f0
- Rajagopalan Narayanan, Seven Tips for Your First Business Rules Project, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/201a9e3d-3ec2-2a10-85b2-ce56d276dd7a
- Rajagopalan Narayanan, Real World Return of Investment Scenarios with Business Rules Management, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/b050905e-3cc2-2a10-979a-81a57a787f56
- Rajagopalan Narayanan, Five Reasons to Build Agile Systems Using Business Rules Management Functionality, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/504486eb-43c2-2a10-f5a7-e84ef3fd45be
- Rajagopalan Narayanan, How Business Rules Management Functionality Helps Tariff Plans Management in Transportation and Shipping, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/40a9cf69-40c2-2a10-8a8b-969fb311dd31
- Rajagopalan Narayanan, Getting Started with Business Rules Management, https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/70c669d8-3ac2-2a10-0e96-c7c3786168f0
- For more information, visit the Business Process Expert homepage.

© 2008 SAP AG

Copyright

© 2008 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.