BOPF Supportability

Tilmann Kopp

SAP AG, 2014

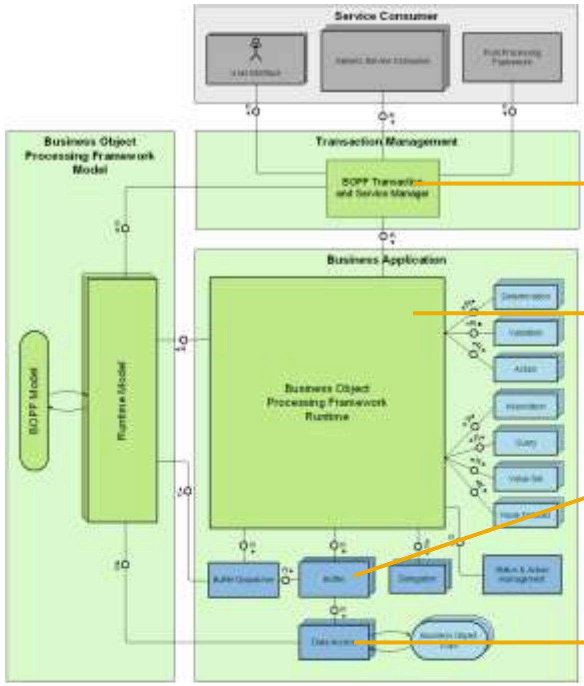
Disclaimer

BOPF Supportability
Content

Core Service Debugging

Core Service Debugging

BOPF Classes and Interfaces



Service Consumer
Business Object Processing Framework Model
Transaction Management
BOPF Transaction and Service Manager
Business Application
BOPF Model
Runtime Model
Business Object Processing Framework Runtime

/bobf/cl_tra_service_mgr
(/BOBF/IF_TRA_SERVICE_MANAGER)

/bobf/cl_frw
(/BOBF/IF_FRW_SERVICE_LAYER)

/bobf/cl_buf_simple
(/BOBF/IF_FRW_BUFFER)

/bobf/cl_dac_table
(/BOBF/IF_BUF_DATA_ACCESS)

Scenario

A certain core service (e.g. action) does not deliver the desired result (e.g. action is not executed.

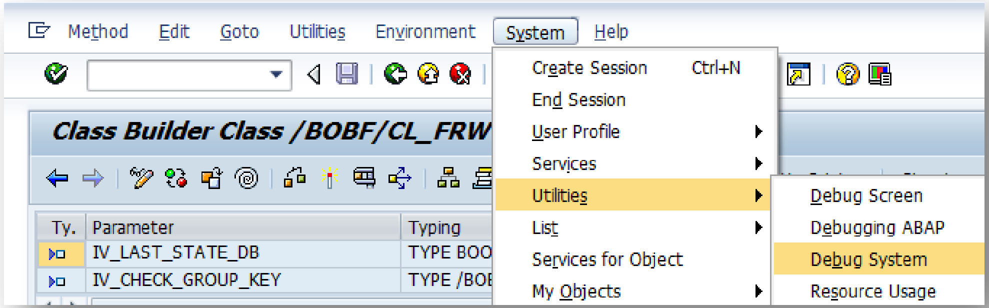
Solution

Enable „System Debugging“ and debug the BOPF.

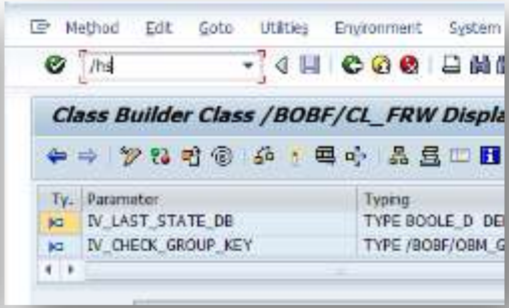
Core Service Debugging

Enabling System Debugging

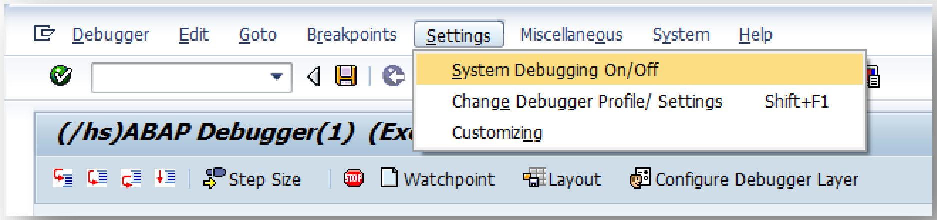
a) Temporary enable “System Debugging” (one stop in the debugger):



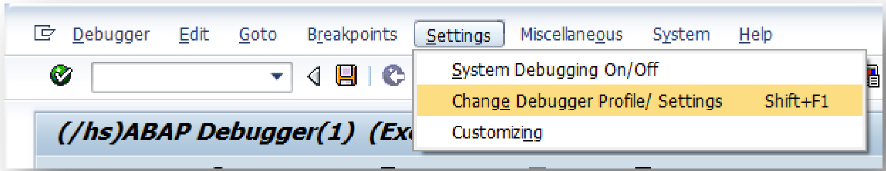
Ty.	Parameter	Typing
▶□	IV_LAST_STATE_DB	TYPE BOO
▶□	IV_CHECK_GROUP_KEY	TYPE /BOE

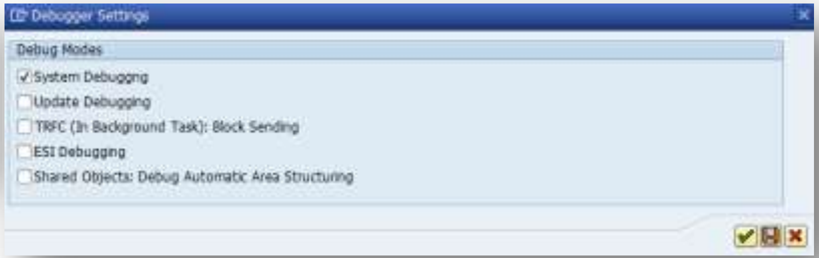


Ty.	Parameter	Typing
▶□	IV_LAST_STATE_DB	TYPE BOOLE_D DE
▶□	IV_CHECK_GROUP_KEY	TYPE /BOBF/OBM_G



b) Constantly enable “System Debugging”:





Debug Modes

- [✓] System Debugging
- [ ] Update Debugging
- [ ] TRFC (In Background Task): Block Sending
- [ ] ESI Debugging
- [ ] Shared Objects: Debug Automatic Area Structuring

Save Rejection Debugging

Save Rejecting Debugging
Overview

Scenario
During a transaction the save is rejected by a validation. However the consumer does not receive any message. In the Test UI, only the information about the business object rejecting the save is returned. How to find the validation that rejects the save?

Solution

1. Put a breakpoint in class „/BOBF/CL_FRW“ methods „DO_VALIDATE“ and „CHECK_ACTION“ after statement „LS_VAL-CLASS->EXECUTE()“.
2. If the debugger stops add breakpoint condition „LINES(LT_FAILED_KEY) > 0“ via context menu of the breakpoint
3. Execute the save, copy the validation‘s key (LS_CONTEXT_VAL-VAL_KEY) and press CTRL+F in the transaction BOBF to jump to the guilty validation

Locking Issues

Locking Issues
Overview

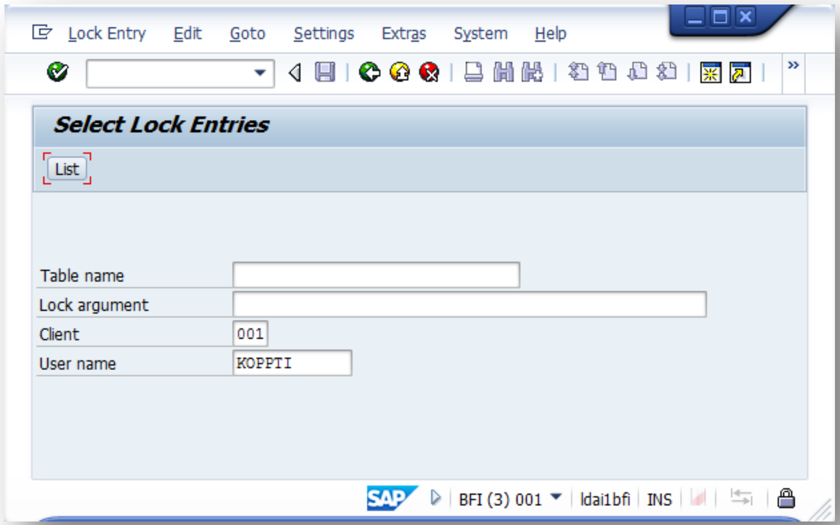


Table name	
Lock argument	
Client	001
User name	KOPPTI

Scenario

During debugging a transaction, it is not clear which instances are really locked.

Problem

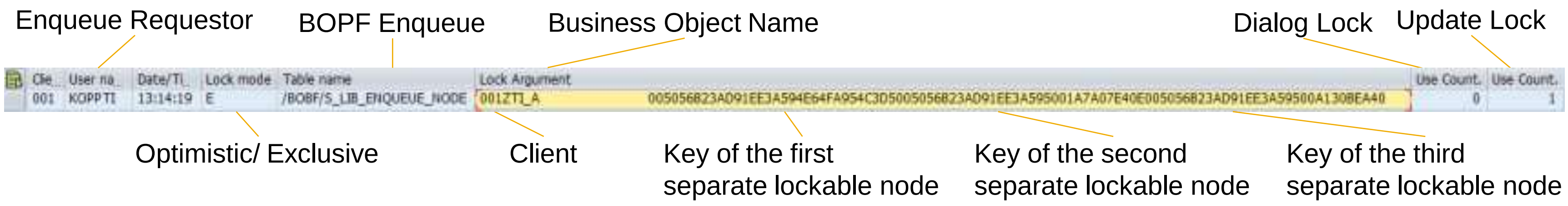
How to find the current locked instances?

Solutions

Start the SM12 and check the enqueues like it is described on the next slide. Precondition is the default BOPF locking action (/BOBF/CL_LIB_A_LOCK).

Locking Issues

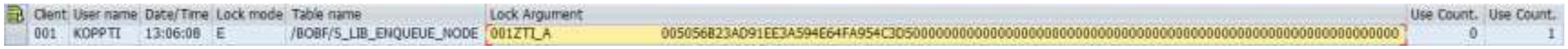
Understanding BOPF Locks in SM12


Enqueue Requestor
BOPF Enqueue
Business Object Name
Dialog Lock
Update Lock
Optimistic/ Exclusive
Client
Key of the first separate lockable node
Key of the second separate lockable node
Key of the third separate lockable node

Clie..	User na..	Date/Ti..	Lock mode	Table name	Lock Argument		Use Count.	Use Count.
001	KOPPTI	13:14:19	E	/BOBF/S_LIB_ENQUEUE_NODE	001ZTI_A	005056823AD91EE3A594E64FA954C3D5005056823AD91EE3A595001A7A07E40E005056823AD91EE3A59500A1308EA40	0	1

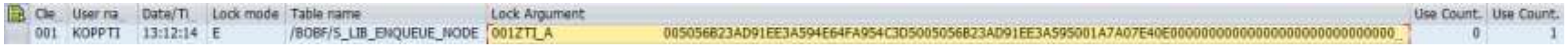
Example
SM12 Screenshots of a BO having only separate lockable nodes:

- Exclusive Locked Root Node Instance



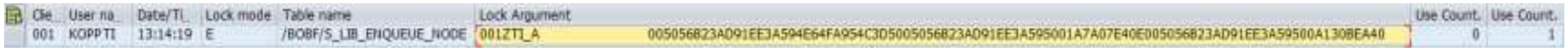
Client	User name	Date/Time	Lock mode	Table name	Lock Argument		Use Count.	Use Count.
001	KOPPTI	13:06:08	E	/BOBF/S_LIB_ENQUEUE_NODE	001ZTI_A	005056823AD91EE3A594E64FA954C3D50000000000000000000000000000000000000000000000000000000000000000	0	1

- Exclusive Locked Subnode Instance



Clie..	User na..	Date/Ti..	Lock mode	Table name	Lock Argument		Use Count.	Use Count.
001	KOPPTI	13:12:14	E	/BOBF/S_LIB_ENQUEUE_NODE	001ZTI_A	005056823AD91EE3A594E64FA954C3D5005056823AD91EE3A595001A7A07E40E0000000000000000000000000000000_	0	1

- Exclusive Locked Sub-Subnode Instances

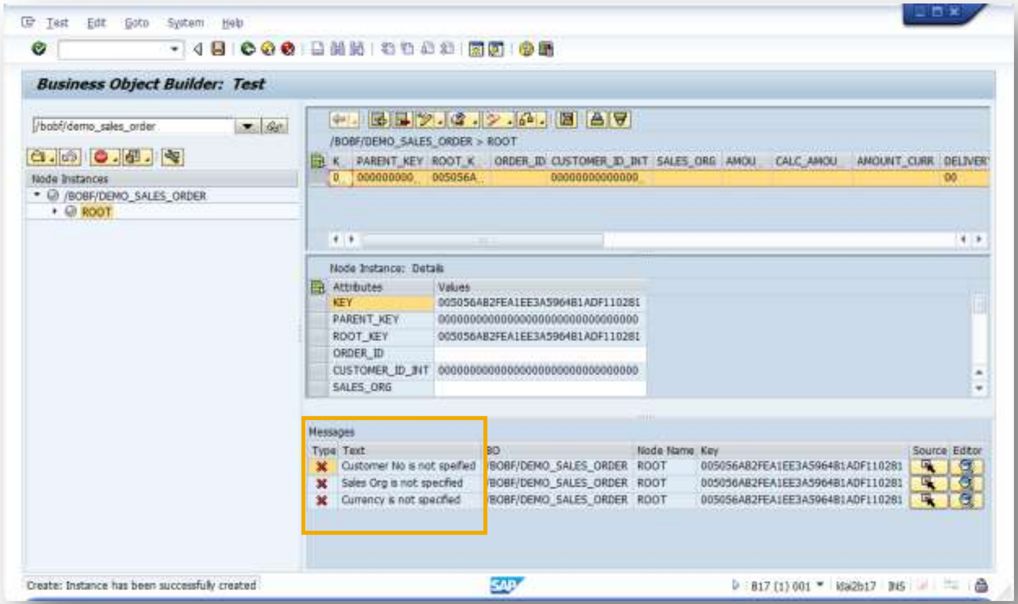


Clie..	User na..	Date/Ti..	Lock mode	Table name	Lock Argument		Use Count.	Use Count.
001	KOPPTI	13:14:19	E	/BOBF/S_LIB_ENQUEUE_NODE	001ZTI_A	005056823AD91EE3A594E64FA954C3D5005056823AD91EE3A595001A7A07E40E005056823AD91EE3A59500A1308EA40	0	1

Hints
- More than 3 lock levels isn‘t supported by the default lock library class
- If all three instances are locked, all three entries are displayed together in SM12

Message Debugging

Message Debugging
Overview



/BOBF/DEMO_SALES_ORDER > ROOT

K	PARENT_KEY	ROOT_K	ORDER_ID	CUSTOMER_ID_INT	SALES_ORG	AMOU	CALC_AMOU	AMOUNT_CURR	DELIVER
0	000000000	005056A		00000000000000					00

Node Instance: Details

Attributes	Values
KEY	005056AB2FEA1EE3A5964B1ADF110281
PARENT_KEY	00000000000000000000000000000000
ROOT_KEY	005056AB2FEA1EE3A5964B1ADF110281
ORDER_ID	
CUSTOMER_ID_INT	00000000000000000000000000000000
SALES_ORG	

Messages

Type	Text	BO	Node Name	Key	Source	Editor
✗	Customer No is not spefied	/BOBF/DEMO_SALES_ORDER	ROOT	005056AB2FEA1EE3A596481ADF110281		
✗	Sales Org is not specfied	/BOBF/DEMO_SALES_ORDER	ROOT	005056AB2FEA1EE3A596481ADF110281		
✗	Currency is not specified	/BOBF/DEMO_SALES_ORDER	ROOT	005056AB2FEA1EE3A596481ADF110281		

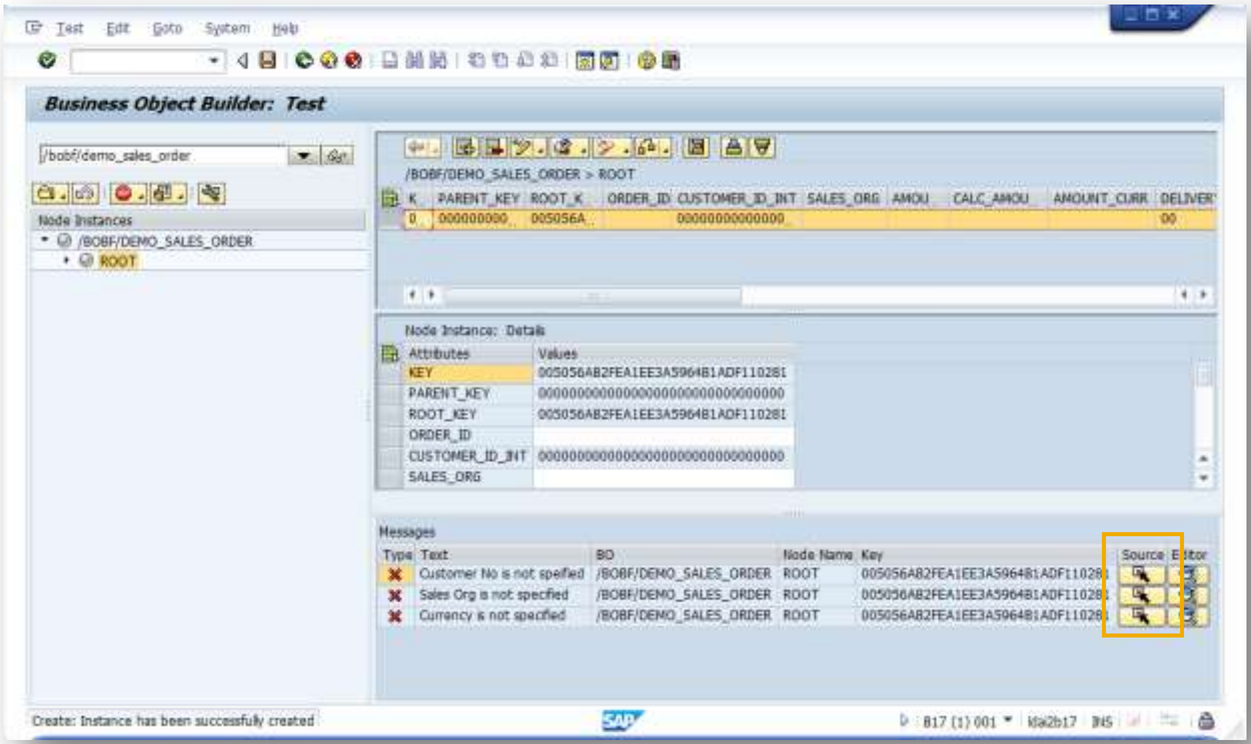
Create: Instance has been successfully created

Scenario
A message appears on the user interface.

Problem
How to find the source of that message?

Message Debugging

Test UI: Navigate to the place where a Message has been created



/BOBF/DEMO_SALES_ORDER > ROOT

K	PARENT_KEY	ROOT_K	ORDER_ID	CUSTOMER_ID_INT	SALES_ORG	AMOU	CALC_AMOU	AMOUNT_CURR	DELIVER
0	000000000	005056A		0000000000000					00

Node Instance: Details

Attributes	Values
KEY	005056AB2FEA1EE3A5964B1ADF110281
PARENT_KEY	00000000000000000000000000000000
ROOT_KEY	005056AB2FEA1EE3A5964B1ADF110281
ORDER_ID	
CUSTOMER_ID_INT	00000000000000000000000000000000
SALES_ORG	

Messages

Type	Text	BO	Node Name	Key	Source	Editor
✖	Customer No is not spefied	/BOBF/DEMO_SALES_ORDER	ROOT	005056AB2FEA1EE3A5964B1ADF11028		
✖	Sales Org is not specfied	/BOBF/DEMO_SALES_ORDER	ROOT	005056AB2FEA1EE3A5964B1ADF11028		
✖	Currency is not specfied	/BOBF/DEMO_SALES_ORDER	ROOT	005056AB2FEA1EE3A5964B1ADF11028		

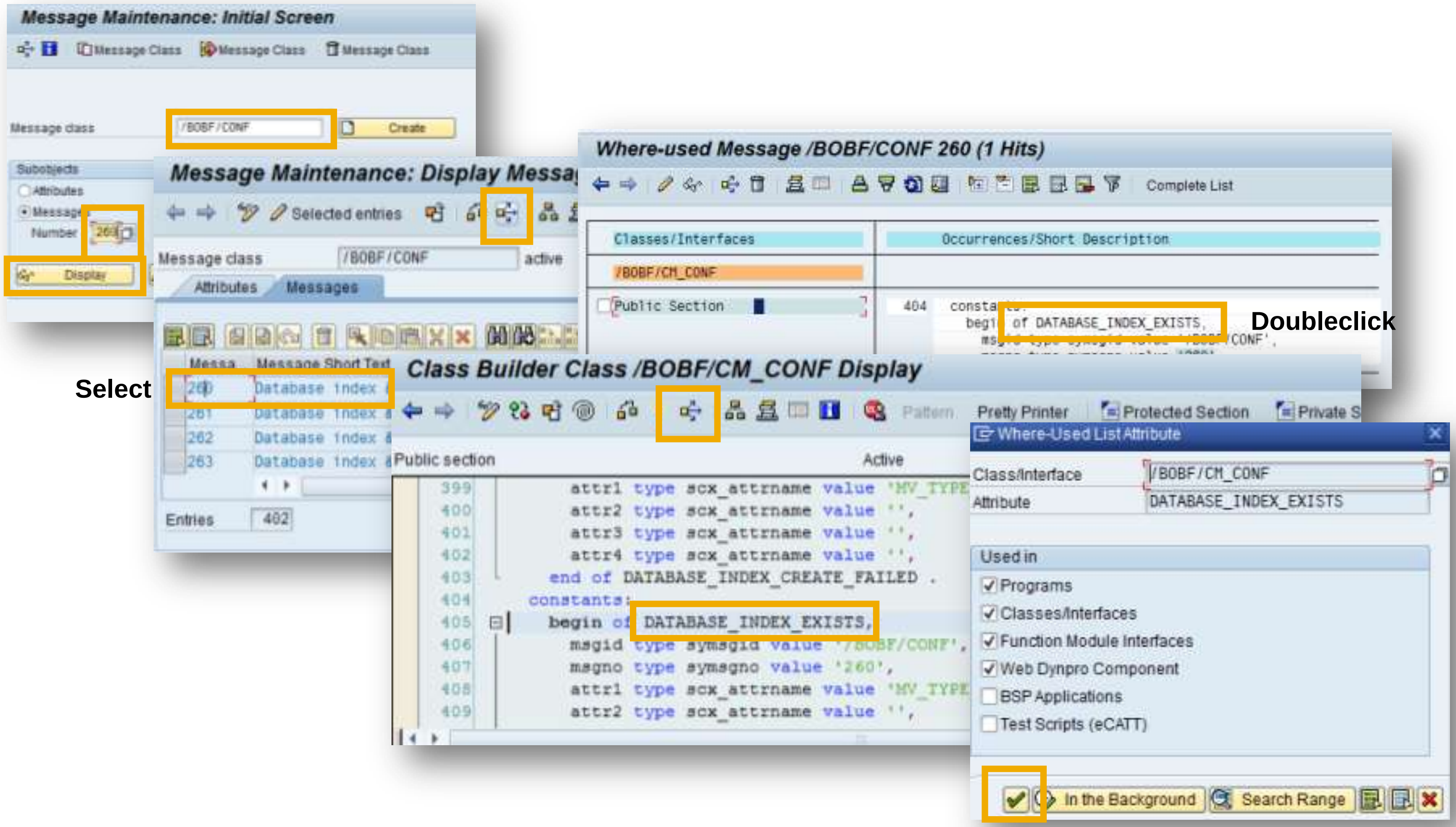
Create: Instance has been successfully created

The Test UI (transaction BOBT) allows to navigate to the source code position of the creation of each message

Message Debugging

Static Reference

SE91



Message Maintenance: Initial Screen

Message class	/BOBF/CONF
Subobjects	
○ Attributes	
● Messages	
Number	260

Message Maintenance: Display Messa

Message class /BOBF/CONF active

Messa	Message Short Text
260	Database index
261	Database index
262	Database index
263	Database index

Entries 402

Where-used Message /BOBF/CONF 260 (1 Hits)

Classes/Interfaces	Occurrences/Short Description
/BOBF/CM_CONF	
Public Section	404 constants:
	begin of DATABASE_INDEX_EXISTS,
	msgid type symsgid value '/BOBF/CONF',

Class Builder Class /BOBF/CM_CONF Display

Public section Active

```
        attr1 type scx_attrname value 'MV_TYPE
        attr2 type scx_attrname value '',
        attr3 type scx_attrname value '',
        attr4 type scx_attrname value '',
      end of DATABASE_INDEX_CREATE_FAILED .
    constants:
      begin of DATABASE_INDEX_EXISTS,
        msgid type symsgid value '/BOBF/CONF',
        msgno type symsgno value '260',
        attr1 type scx_attrname value 'MV_TYPE
        attr2 type scx_attrname value '',
```

Where-Used List Attribute

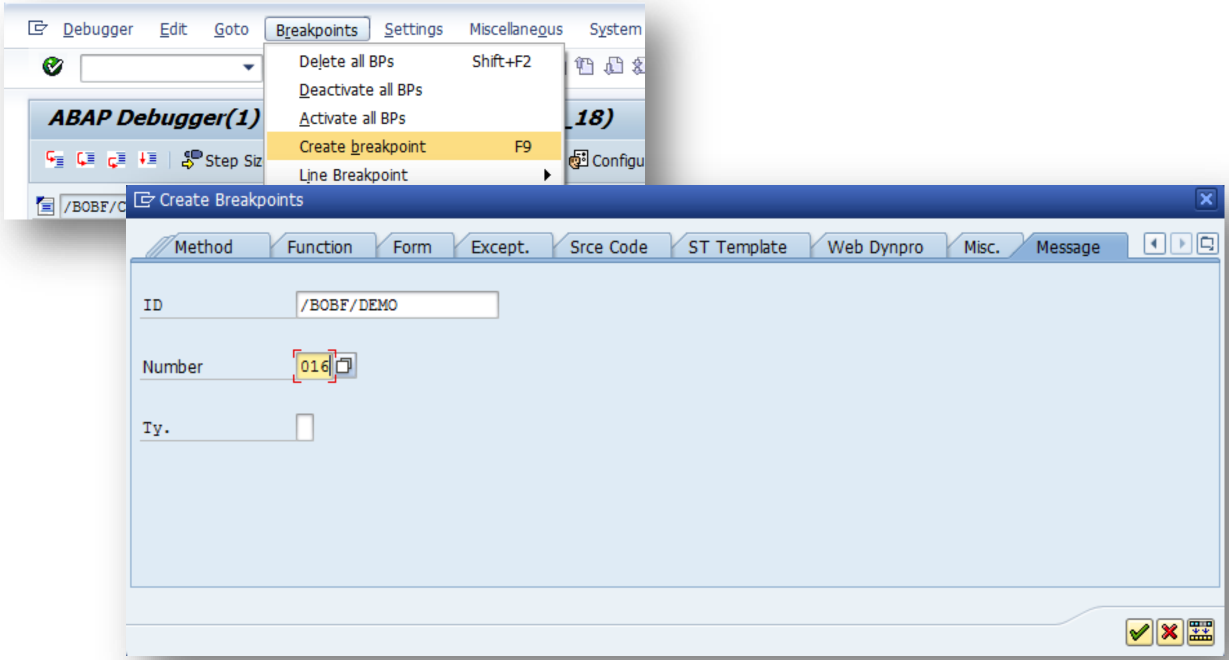
Class/Interface	/BOBF/CM_CONF
Attribute	DATABASE_INDEX_EXISTS

Used in

- [✓] Programs
- [✓] Classes/Interfaces
- [✓] Function Module Interfaces
- [✓] Web Dynpro Component
- [ ] BSP Applications
- [ ] Test Scripts (eCATT)

Message Debugging

Dynamic Message Debugging



ID	/BOBF/DEMO
Number	016
Ty.	

If a message is used by many different places, the where used method is not sufficient as the breakpoints are limited. In that case step into the debugger (e.g. „/h“), open the „Create breakpoint“ menu and maintain the message which shall be tracked.

Software Layer Aware Debugging

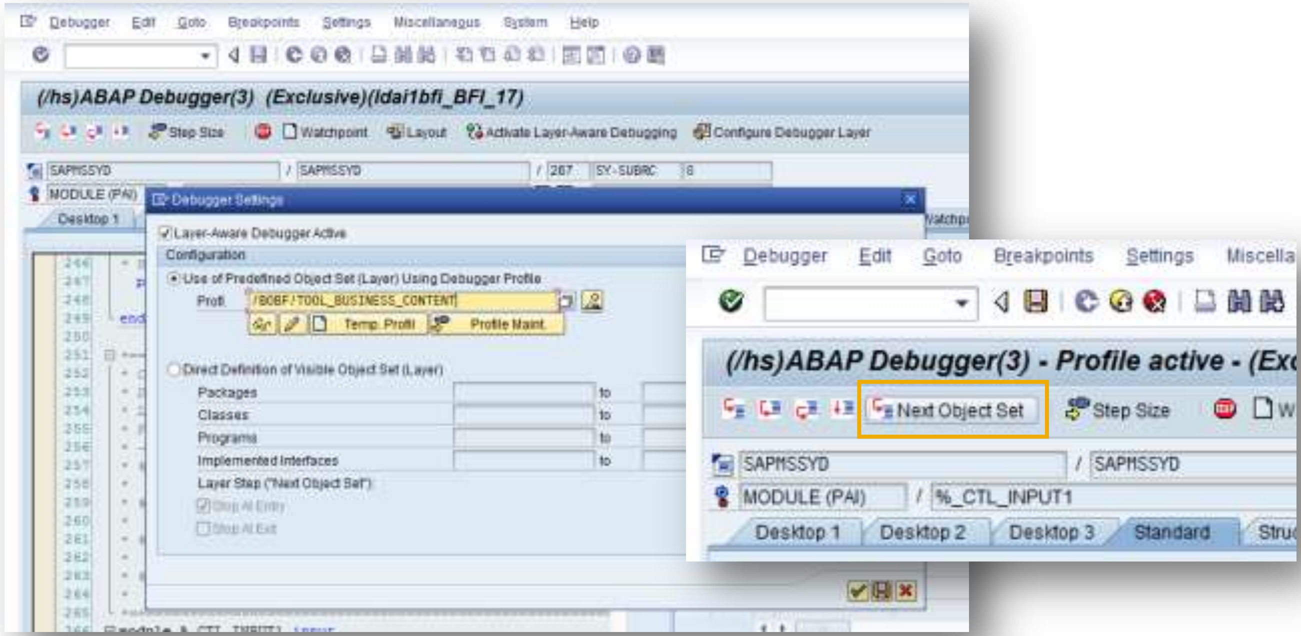
Software Layer Aware Debugging
Motivation

Scenario
The execution of a core service or action triggers a lot of determinations. In order to check your application logic, you would like to debug all entities along the roundtrip.

Solution
Enable Software Layer Aware Debugging in order to browse through the implementation of the affected entities.

Software Layer Aware Debugging

Enable SLAD



(/hs)ABAP Debugger(3) (Exclusive)(Idai1bfi_BFI_17)

SAPMSSYD / SAPMSSYD / 267 SY-SUBRC 8

MODULE (PAI)

Debugger Settings

☑ Layer-Aware Debugger Active

Configuration

(●) Use of Predefined Object Set (Layer) Using Debugger Profile

Profil /BOBF/TOOL_BUSINESS_CONTENT

( ) Direct Definition of Visible Object Set (Layer)

Packages		to
Classes		to
Programs		to
Implemented Interfaces		to

Layer Step ("Next Object Set")

☑ Stop At Entry

☐ Stop At Exit

(/hs)ABAP Debugger(3) - Profile active - (Exc

SAPMSSYD / SAPMSSYD

MODULE (PAI) / %_CTL_INPUT1

1. Start your application (e.g. Test UI, transaction BOBT)
2. Enter debugger (e.g. /h) and click on button „configure debugger layer“
3. Maintain Profile „/BOBF/TOOL_BUSINESS_CONTENT“
4. Use the button „Next Object Set“ to jump to your entities

Checkpoint Groups

Checkpoint Groups
Motivation

Scenario
The implementation of entities is sometimes quite complicated and errors are made. Often those errors refer to the usage of the interface parameter of the entity.

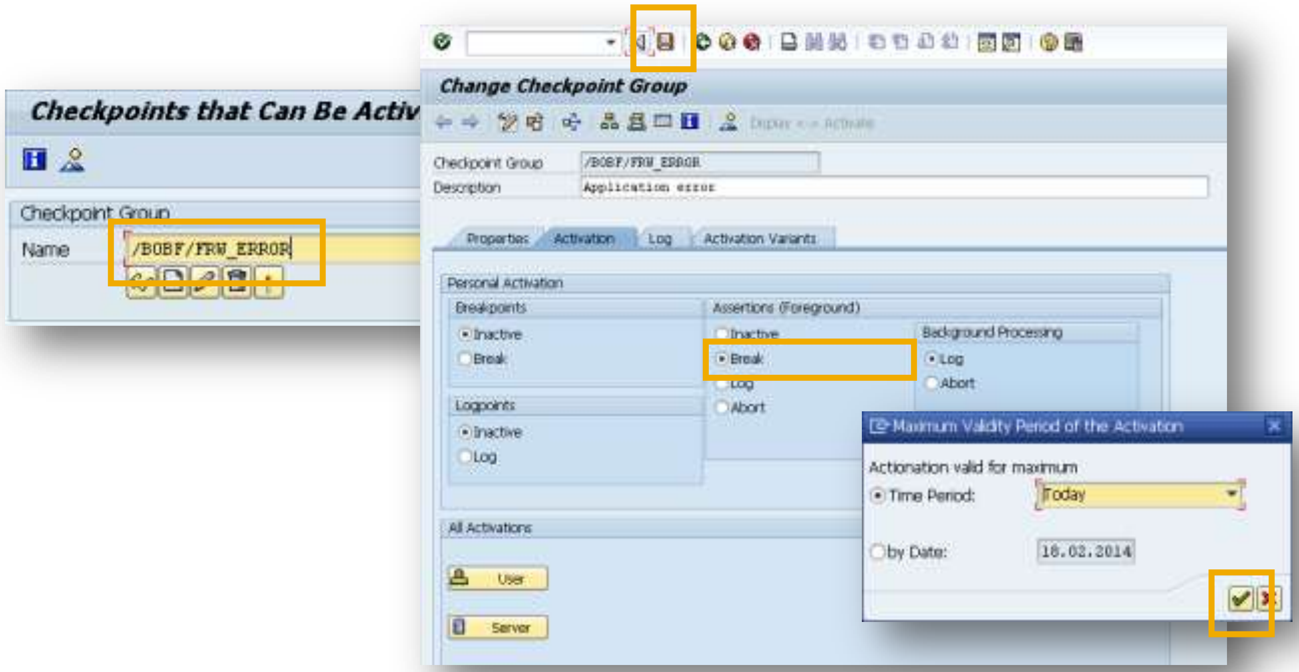
Problem
Those errors can not be checked by the help of static syntax checks but occur only at runtime. Thus they are often lately discovered.

Solution
The BOPF source code contains multiple checks that validate the framework state and the output of executed entities. Those checks can be dynamically switched on during the development phase by the help of the transaction SAAB. As they cost performance, they are switched off by default.

Example: An action must return only a subset of the IT_KEY as ET_FAILED_KEY. If this is not fulfilled and the checkpoint group is active, the debugger comes up.

Checkpoint Groups

Using the SAAB to maintain Checkpoint Groups


Checkpoints that Can Be Activ

Checkpoint Group

Name: /BOBF/FRW_ERROR

Change Checkpoint Group

Checkpoint Group: /BOBF/FRW_ERROR

Description: Application error

Personal Activation

Breakpoints: [✓ Inactive] Break

Logpoints: [✓ Inactive] Log

Assertions (Foreground): Inactive [✓ Break] Log Abort

Background Processing: [✓ Log] Abort

All Activations

Maximum Validity Period of the Activation

Activation valid for maximum

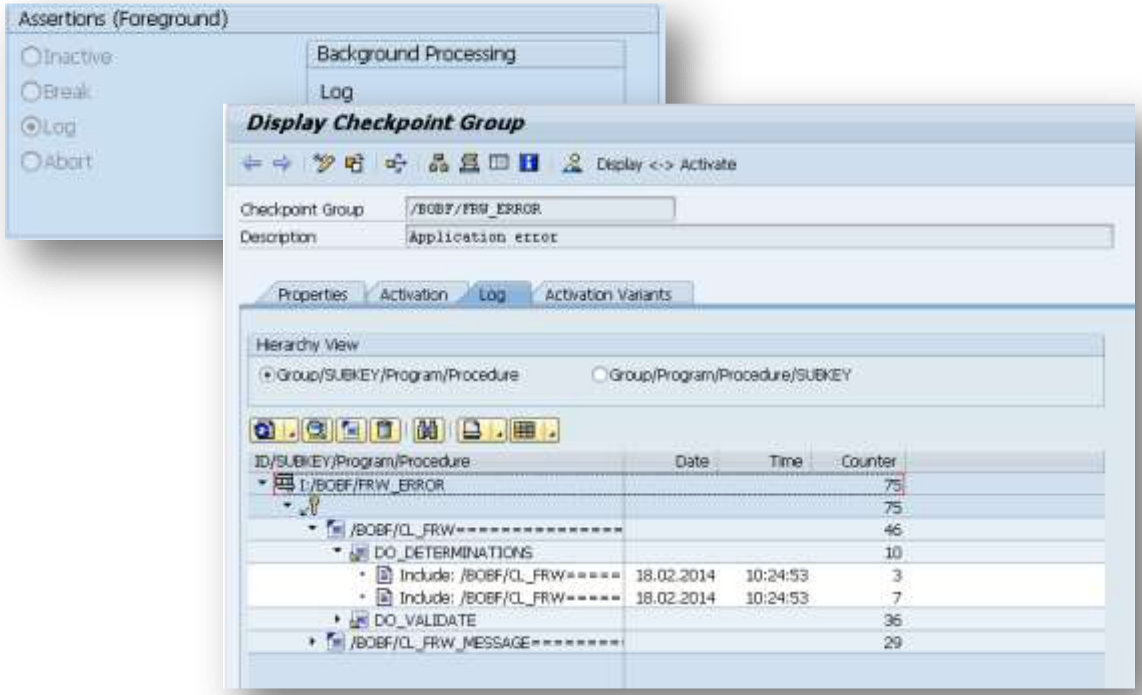
[✓ Time Period: Today]

by Date: 18.02.2014

Transaction SAAB allows to open existing checkpoint groups. In the activation tab, assertions can be switched on (Break) or off (Inactive). After saving the settings, the activation duration can be maintained.

Checkpoint Groups

Using to SAAB to Log Implementation Issues


Assertions (Foreground)

Inactive [✓ Log] Break Abort

Background Processing: Log

Display Checkpoint Group

Display <-> Activate

Checkpoint Group	/BOBF/FRW_ERROR
Description	Application error

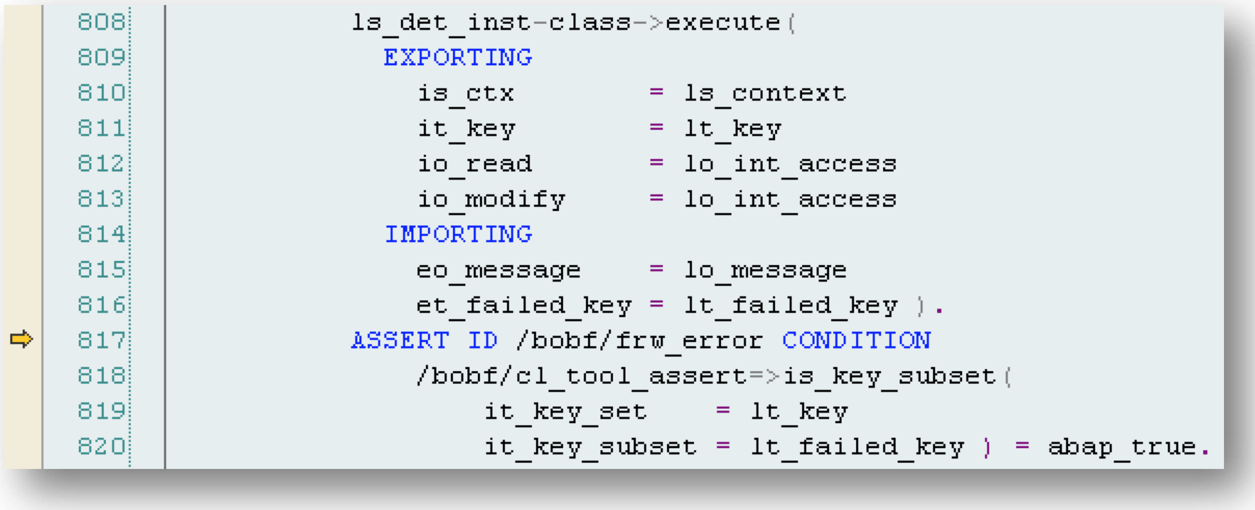
Properties | Activation | Log | Activation Variants

Hierarchy View: [✓ Group/SUBKEY/Program/Procedure] Group/Program/Procedure/SUBKEY

ID/SUBKEY/Program/Procedure	Date	Time	Counter
I:/BOBF/FRW_ERROR			75
			75
/BOBF/CL_FRW================			46
DO_DETERMINATIONS			10
Include: /BOBF/CL_FRW=====	18.02.2014	10:24:53	3
Include: /BOBF/CL_FRW=====	18.02.2014	10:24:53	7
DO_VALIDATE			36
/BOBF/CL_FRW_MESSAGE========			29

If “Log” instead of “Break” is chosen, the issues are logged and can be monitored. It would be possible to activate the following checkpoint groups and silently log all issues. Before the sprint end, the log can be evaluated.

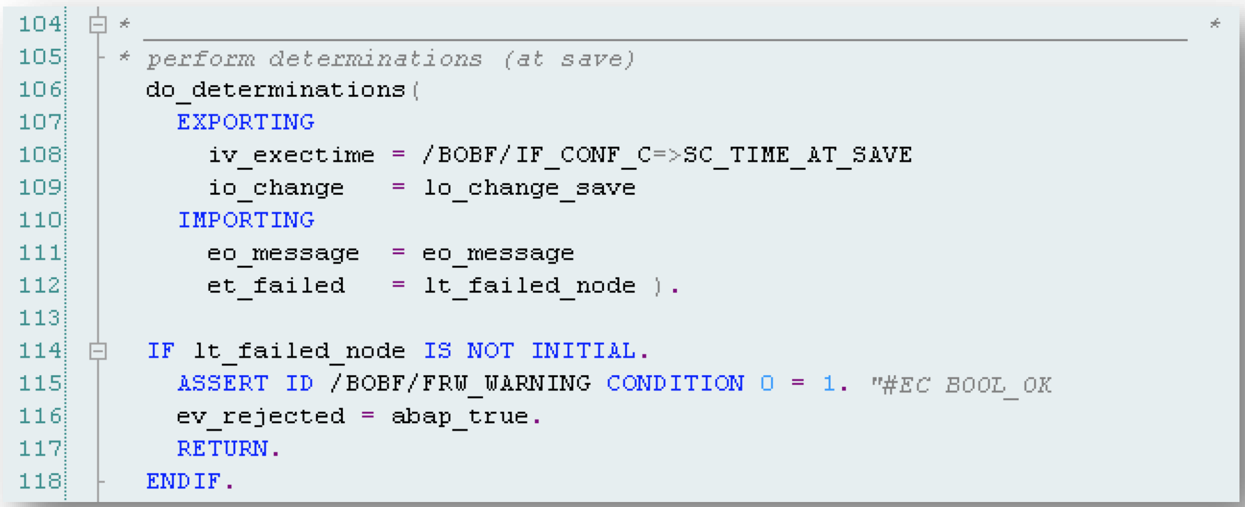
Checkpoint Groups
Error Group (/BOBF/FRW_ERROR)



```
        ls_det_inst-class->execute(
          EXPORTING
            is_ctx          = ls_context
            it_key          = lt_key
            io_read         = lo_int_access
            io_modify       = lo_int_access
          IMPORTING
            eo_message      = lo_message
            et_failed_key = lt_failed_key ).
        ASSERT ID /bobf/frw_error CONDITION
            /bobf/cl_tool_assert=>is_key_subset(
                it_key_set      = lt_key
                it_key_subset = lt_failed_key ) = abap_true.
```

The /BOBF/FRW_ERROR Checkpoint Group consists of many checks indicating major implementation issues that should be fixed.

Checkpoint Groups
Warning Group (/BOBF/FRW_WARNING)



```
*                                                              *
* perform determinations (at save)
do_determinations(
  EXPORTING
    iv_exectime = /BOBF/IF_CONF_C=>SC_TIME_AT_SAVE
    io_change   = lo_change_save
  IMPORTING
    eo_message  = eo_message
    et_failed   = lt_failed_node ).

IF lt_failed_node IS NOT INITIAL.
  ASSERT ID /BOBF/FRW_WARNING CONDITION 0 = 1. "#EC BOOL_OK
  ev_rejected = abap_true.
  RETURN.
ENDIF.
```

The /BOBF/FRW_WARNING Checkpoint Group consists of many checks indicating potential implementation issues. It could be used to debug a rejected save.

- Rejected Save (by Determinations or Validations)
- Rejected modifications due to property violations

Checkpoint Groups
/BOBF/FLUSH

Scenario
Modifications done by entities like determinations and actions are applied lazy at the end of the entity execution to the buffer. Thus if an erroneous modification is done, it will dump later on but not immediately after the request.

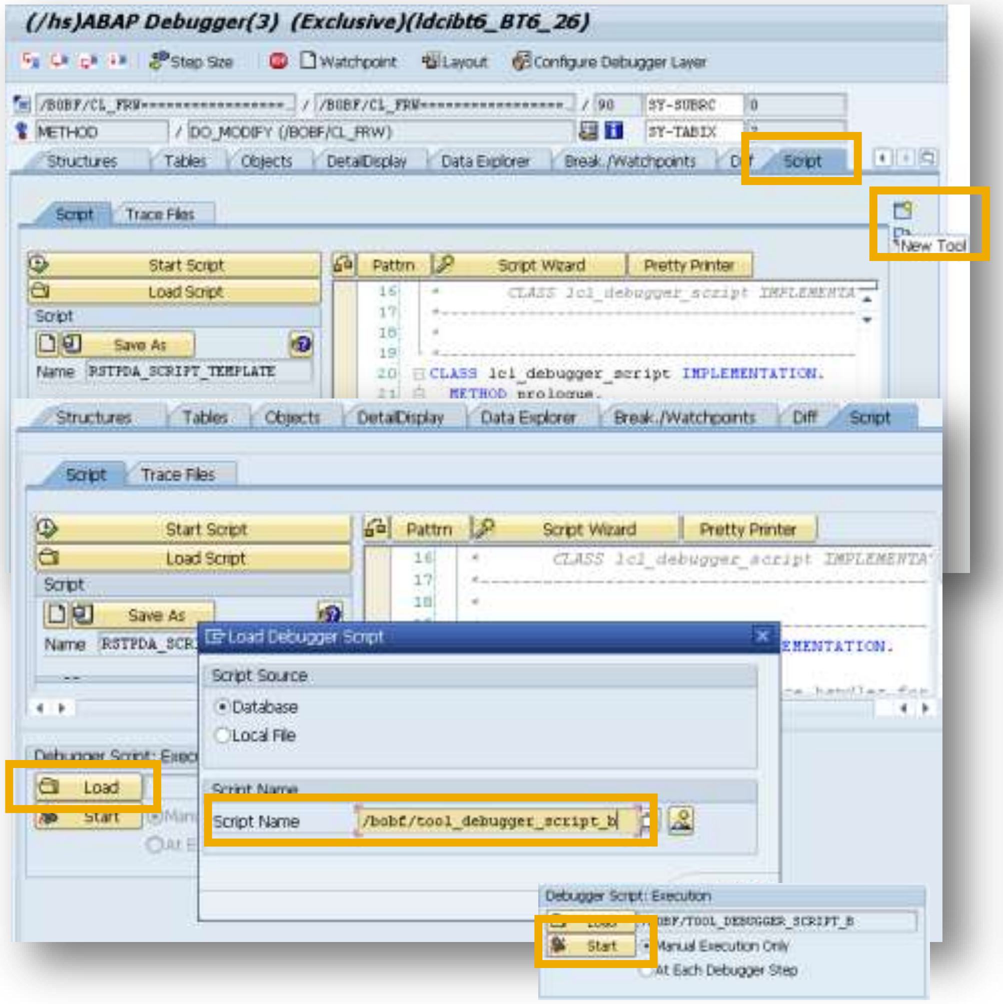
Problem
It is not possible to figure out the guilty modification if they are processed later on in a group.

Solution
If the breakpoint group „/BOBF/FLUSH“ is activated in the SAAB, BOPF immediately flushes each modification without collecting it. It behaves like having an explicit io_modify->end_modify() call after each single modification.Thus the real position of the erroneous modification is shown. This must only be done for testing due to performance issues.

Debugger Scripts

Debugger Scripts

Starting a Debugger Script



(/hs)ABAP Debugger(3) (Exclusive)(ldcibt6_BT6_26)

/BOBF/CL_FRW==================== / /BOBF/CL_FRW==================== / 90 SY-SUBRC 0

METHOD / DO_MODIFY (/BOBF/CL_FRW) SY-TABIX 2

Name RSTPDA_SCRIPT_TEMPLATE

```
*   CLASS lcl_debugger_script IMPLEMENTATION
*---------------------------------------------------
*
CLASS lcl_debugger_script IMPLEMENTATION.
  METHOD prologue.
```

Load Debugger Script

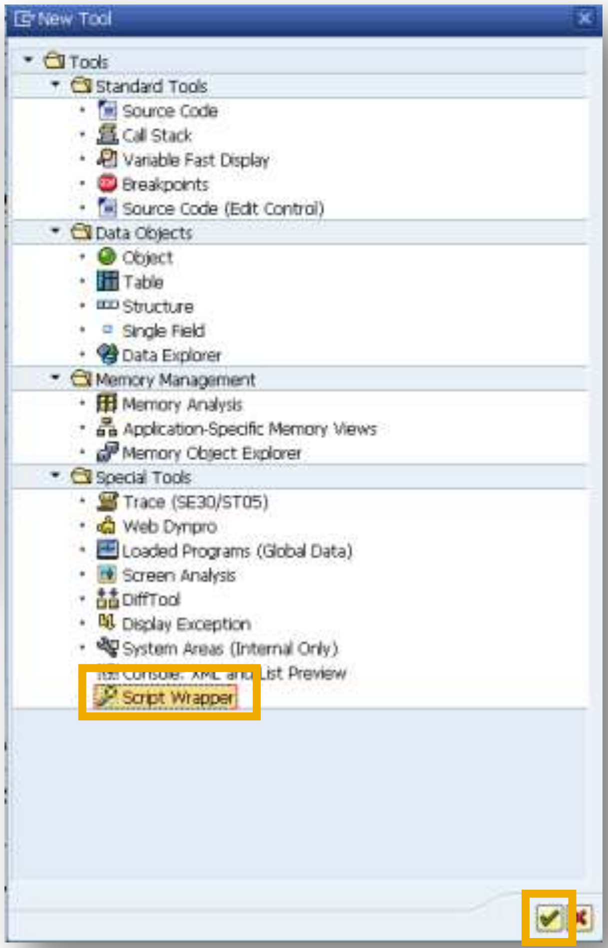
Script Source: [✓ Database] Local File

Script Name /bobf/tool_debugger_script_b

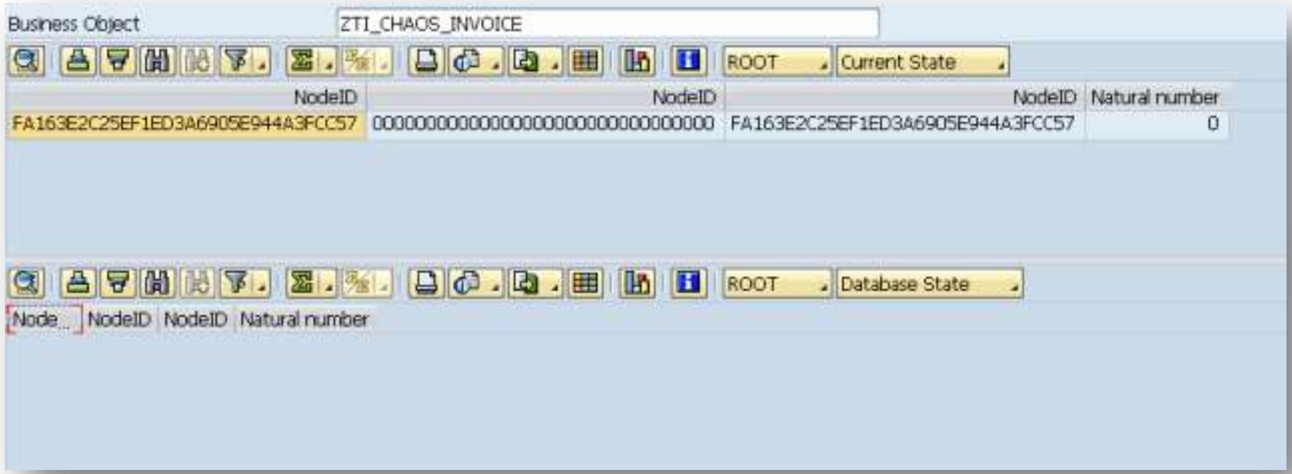
Debugger Script: Execution

Load /BOBF/TOOL_DEBUGGER_SCRIPT_B

Start [✓ Manual Execution Only] At Each Debugger Step



Debugger Scripts
Buffer Debugger Script



Business Object: ZTI_CHAOS_INVOICE

ROOT — Current State

NodeID	NodeID	NodeID	Natural number
FA163E2C25EF1ED3A6905E944A3FCC57	00000000000000000000000000000000	FA163E2C25EF1ED3A6905E944A3FCC57	0

ROOT — Database State

Node..	NodeID	NodeID	Natural number

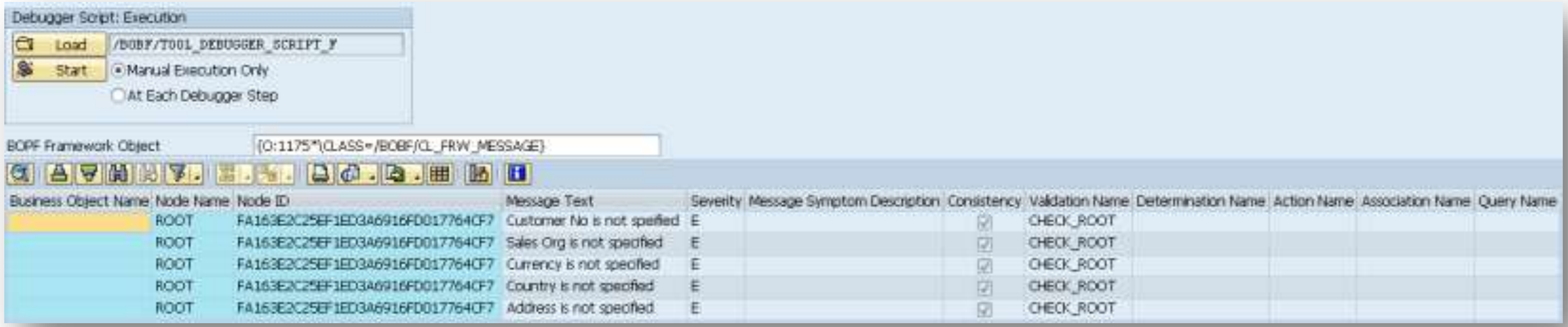
Scenario
You would like to know the buffer content of the BOPF. This is very cumbersome because not the node names are visible, but only their technical keys.

Solution
Load and start debugger script „/BOBF/TOOL_DEBUGGER_SCRIPT_B“. Select the desired node from the dropdown by node name and chose a state.

Debugger Scripts

Message, Change & Property Debugger Script

Message Container



Debugger Script: Execution

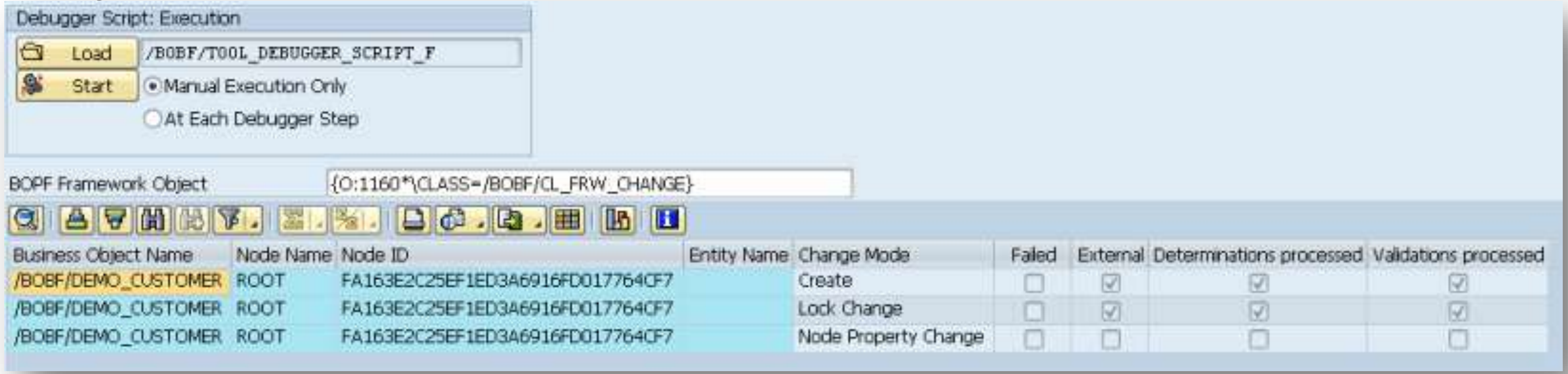
Load /BOBF/TOOL_DEBUGGER_SCRIPT_F

Start [✓ Manual Execution Only] At Each Debugger Step

BOPF Framework Object {O:1175*\CLASS=/BOBF/CL_FRW_MESSAGE}

Business Object Name	Node Name	Node ID	Message Text	Severity	Message Symptom Description	Consistency	Validation Name	Determination Name	Action Name	Association Name	Query Name
	ROOT	FA163E2C25EF1ED3A6916FD017764CF7	Customer No is not spefied	E		✓	CHECK_ROOT				
	ROOT	FA163E2C25EF1ED3A6916FD017764CF7	Sales Org is not specified	E		✓	CHECK_ROOT				
	ROOT	FA163E2C25EF1ED3A6916FD017764CF7	Currency is not specified	E		✓	CHECK_ROOT				
	ROOT	FA163E2C25EF1ED3A6916FD017764CF7	Country is not specified	E		✓	CHECK_ROOT				
	ROOT	FA163E2C25EF1ED3A6916FD017764CF7	Address is not specified	E		✓	CHECK_ROOT				

Change Container



Debugger Script: Execution

Load /BOBF/TOOL_DEBUGGER_SCRIPT_F

Start [✓ Manual Execution Only] At Each Debugger Step

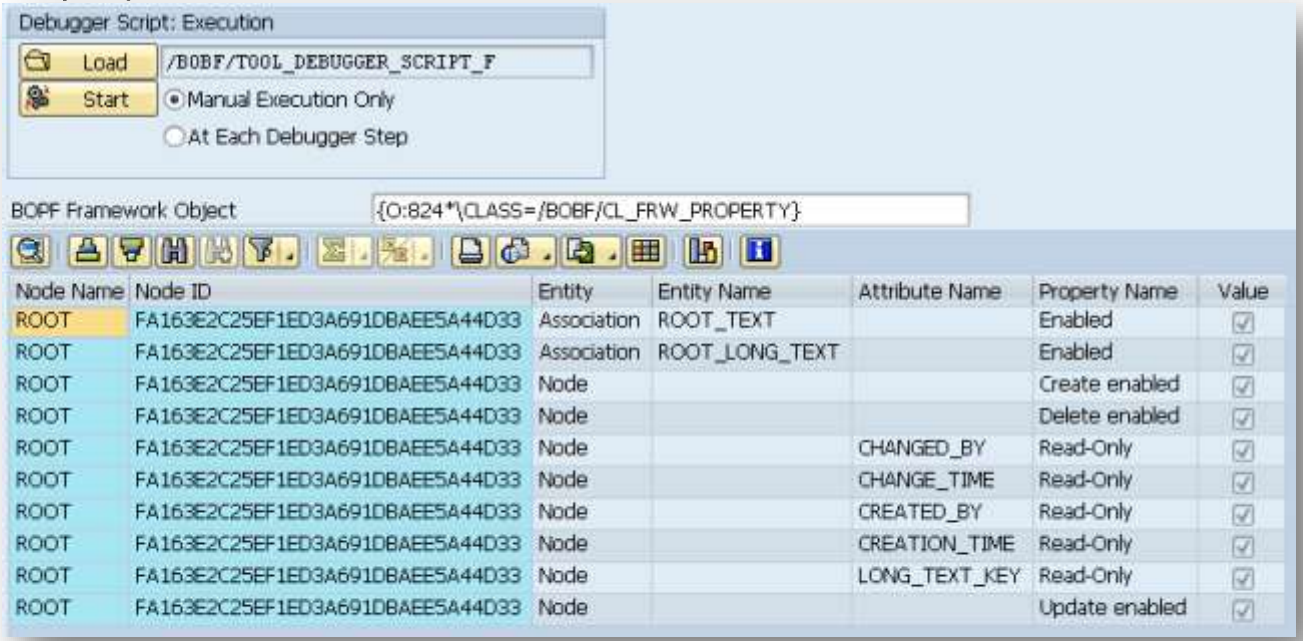
BOPF Framework Object {O:1160*\CLASS=/BOBF/CL_FRW_CHANGE}

Business Object Name	Node Name	Node ID	Entity Name	Change Mode	Failed	External	Determinations processed	Validations processed
/BOBF/DEMO_CUSTOMER	ROOT	FA163E2C25EF1ED3A6916FD017764CF7		Create	☐	✓	✓	✓
/BOBF/DEMO_CUSTOMER	ROOT	FA163E2C25EF1ED3A6916FD017764CF7		Lock Change	☐	✓	✓	✓
/BOBF/DEMO_CUSTOMER	ROOT	FA163E2C25EF1ED3A6916FD017764CF7		Node Property Change	☐	☐	☐	☐

Use debugger script „/BOBF/TOOL_DEBUGGER_SCRIPT_F“ to check framework objects (like eo_message, eo_change, eo_property)

Debugger Scripts
Message, Change & Property Debugger Script

Property Container



/BOBF/TOOL_DEBUGGER_SCRIPT_F

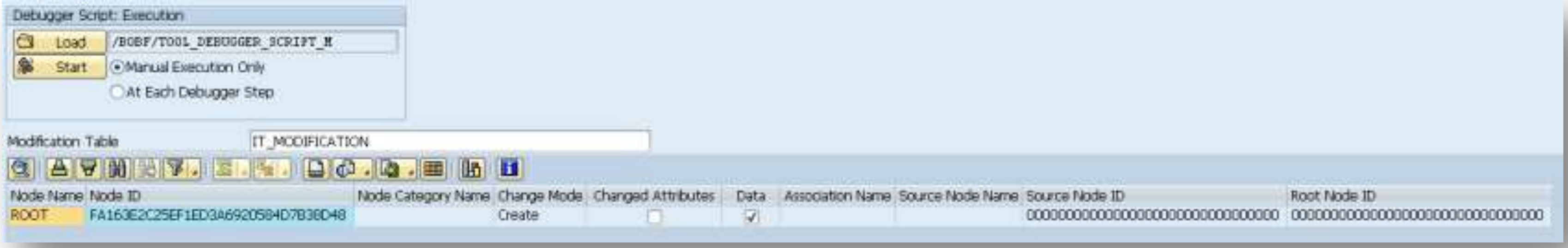
[✓ Manual Execution Only] At Each Debugger Step

BOPF Framework Object {O:824*\CLASS=/BOBF/CL_FRW_PROPERTY}

Node Name	Node ID	Entity	Entity Name	Attribute Name	Property Name	Value
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Association	ROOT_TEXT		Enabled	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Association	ROOT_LONG_TEXT		Enabled	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node			Create enabled	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node			Delete enabled	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node		CHANGED_BY	Read-Only	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node		CHANGE_TIME	Read-Only	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node		CREATED_BY	Read-Only	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node		CREATION_TIME	Read-Only	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node		LONG_TEXT_KEY	Read-Only	✓
ROOT	FA163E2C25EF1ED3A691DBAEE5A44D33	Node			Update enabled	✓

Debugger Scripts
Modification Container Debugger Script

Modification Container



Debugger Script: Execution

Load /BOBF/TOOL_DEBUGGER_SCRIPT_M

Start [● Manual Execution Only] ○ At Each Debugger Step

Modification Table IT_MODIFICATION

Node Name	Node ID	Node Category Name	Change Mode	Changed Attributes	Data	Association Name	Source Node Name	Source Node ID	Root Node ID
ROOT	FA163E2C25EF1ED3A6920584D7B38D48		Create	☐	✓			00000000000000000000000000000000	00000000000000000000000000000000

Use debugger script „/BOBF/TOOL_DEBUGGER_SCRIPT_M“ to check the modification table.

Application Tracing

Application Tracing
Overview

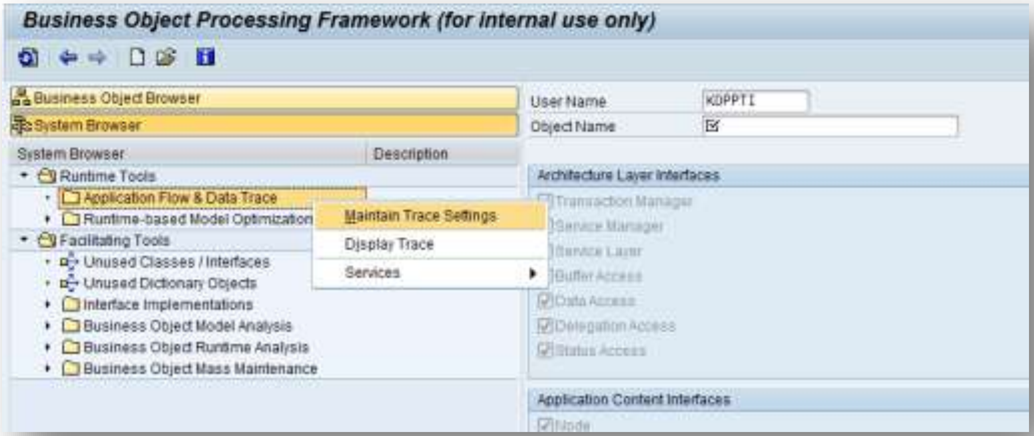
Scenario
In a complex transaction, a non-expected node attribute is somehow modified. It would interesting, which entity (e.g. determination) or consumer is responsible for this modification.

Solutions
Activate the BOPF trace in order to record all core service calls during the scenario. Later on the trace can be searched by instance key in order to get all relevant modifications.

Precondition
The BOPF trace works out of the box. In order to ensure a robust trace, it is also possible to write the trace entries via a secondary database connection (named „R/3*ESFTOOLS “).

Application Tracing

Activation of the Trace


Business Object Processing Framework (for internal use only)
Business Object Browser
System Browser
System Browser
Description
Runtime Tools
Application Flow & Data Trace
Runtime-based Model Optimization
Facilitating Tools
Unused Classes / Interfaces
Unused Dictionary Objects
Interface Implementations
Business Object Model Analysis
Business Object Runtime Analysis
Business Object Mass Maintenance
Maintain Trace Settings
Display Trace
Services

User Name: KDPPTI
Object Name: ☑

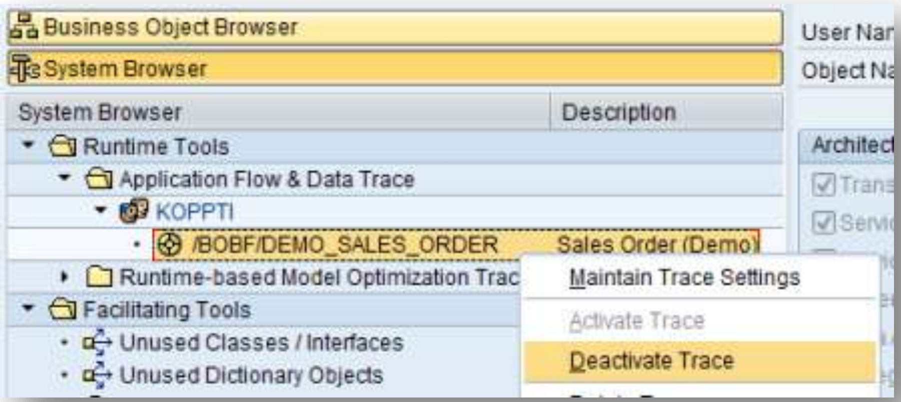
Architecture Layer Interfaces
- ☐ Transaction Manager
- ☐ Service Manager
- ☐ Service Layer
- ☐ Buffer Access
- ☑ Data Access
- ☑ Delegation Access
- ☑ Status Access

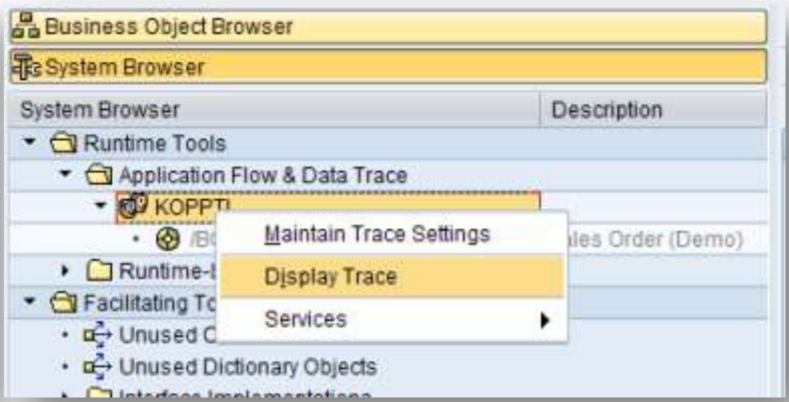
Application Content Interfaces
- ☑ Node

1. Start BOPF Configuration UI (Transaction BOBF)
2. Enable the „System Browser“ (Menue: Utilities > Settings)
3. Maintain the trace settings (see screenshot) and define the business object that shall be traced. Press enter and the refresh button to see the trace configuration in the system browser.
4. Open a new session and start your test scenario

Application Tracing

Deactivation, Display and Deletion of the Trace

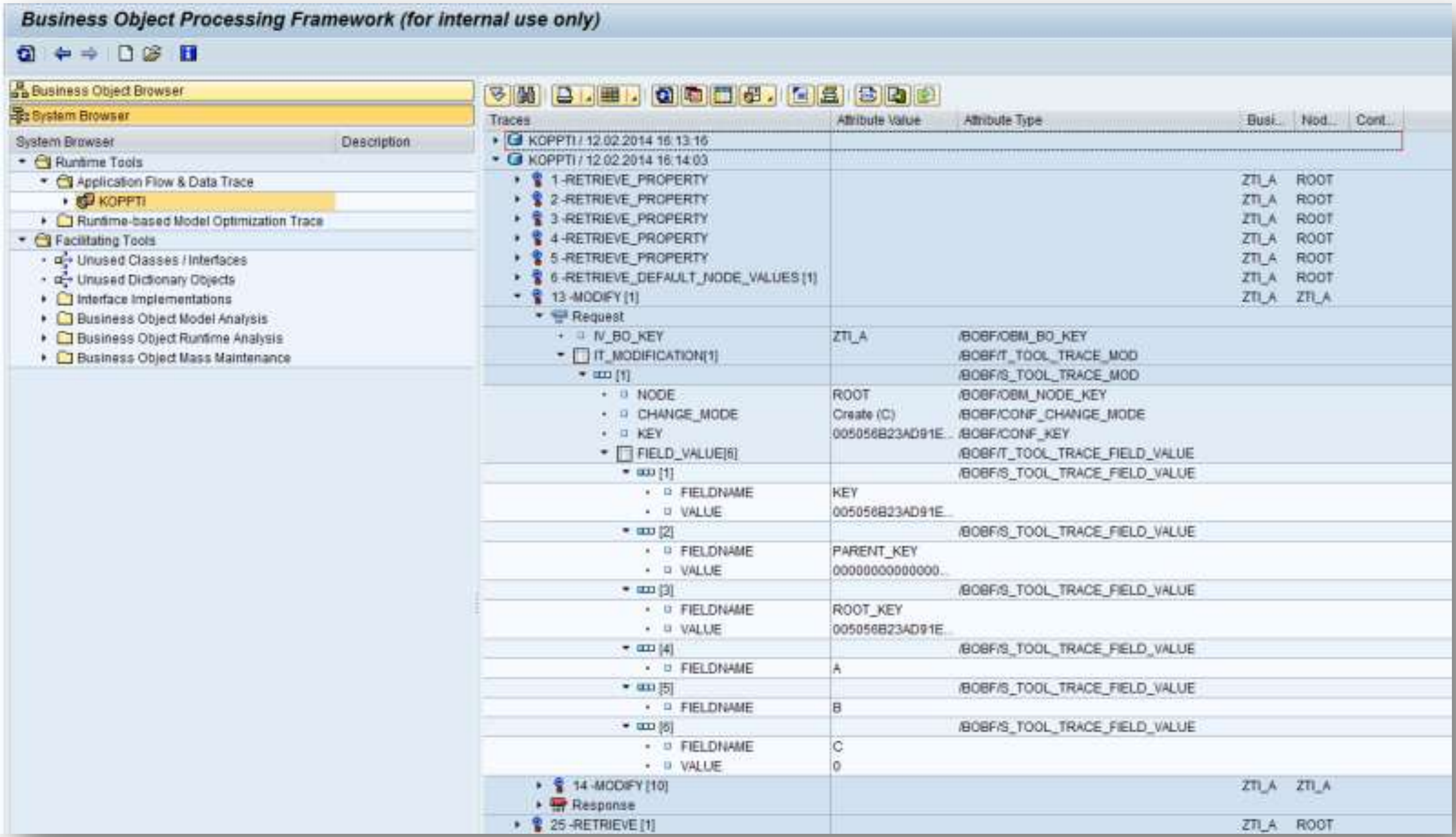




5. If your scenario is finished, deactivate the trace in the context menu of the traced business object name (you have also the option to delete the trace herer)
6. The trace can be displayed via context menu of the user name

Hint: You should deactivate all of your traces after recording!

Application Tracing
Example



Business Object Processing Framework (for internal use only)

Business Object Browser

System Browser

System Browser	Description
Runtime Tools	
Application Flow & Data Trace	
KOPPTI	
Runtime-based Model Optimization Trace	
Facilitating Tools	
Unused Classes / Interfaces	
Unused Dictionary Objects	
Interface Implementations	
Business Object Model Analysis	
Business Object Runtime Analysis	
Business Object Mass Maintenance	

Traces	Attribute Value	Attribute Type	Busi...	Nod...	Cont...
KOPPTI / 12.02.2014 16:13:16					
KOPPTI / 12.02.2014 16:14:03					
1 -RETRIEVE_PROPERTY			ZTI_A	ROOT	
2 -RETRIEVE_PROPERTY			ZTI_A	ROOT	
3 -RETRIEVE_PROPERTY			ZTI_A	ROOT	
4 -RETRIEVE_PROPERTY			ZTI_A	ROOT	
5 -RETRIEVE_PROPERTY			ZTI_A	ROOT	
6 -RETRIEVE_DEFAULT_NODE_VALUES [1]			ZTI_A	ROOT	
13 -MODIFY [1]			ZTI_A	ZTI_A	
Request					
IV_BO_KEY	ZTI_A	/BOBF/OBM_BO_KEY			
IT_MODIFICATION[1]		/BOBF/T_TOOL_TRACE_MOD			
[1]		/BOBF/S_TOOL_TRACE_MOD			
NODE	ROOT	/BOBF/OBM_NODE_KEY			
CHANGE_MODE	Create (C)	/BOBF/CONF_CHANGE_MODE			
KEY	005056B23AD91E...	/BOBF/CONF_KEY			
FIELD_VALUE[6]		/BOBF/T_TOOL_TRACE_FIELD_VALUE			
[1]		/BOBF/S_TOOL_TRACE_FIELD_VALUE			
FIELDNAME	KEY				
VALUE	005056B23AD91E...				
[2]		/BOBF/S_TOOL_TRACE_FIELD_VALUE			
FIELDNAME	PARENT_KEY				
VALUE	00000000000000...				
[3]		/BOBF/S_TOOL_TRACE_FIELD_VALUE			
FIELDNAME	ROOT_KEY				
VALUE	005056B23AD91E...				
[4]		/BOBF/S_TOOL_TRACE_FIELD_VALUE			
FIELDNAME	A				
[5]		/BOBF/S_TOOL_TRACE_FIELD_VALUE			
FIELDNAME	B				
[6]		/BOBF/S_TOOL_TRACE_FIELD_VALUE			
FIELDNAME	C				
VALUE	0				
14 -MODIFY [10]			ZTI_A	ZTI_A	
Response					
25 -RETRIEVE [1]			ZTI_A	ROOT	

Watchpoints

Watchpoints
Overview

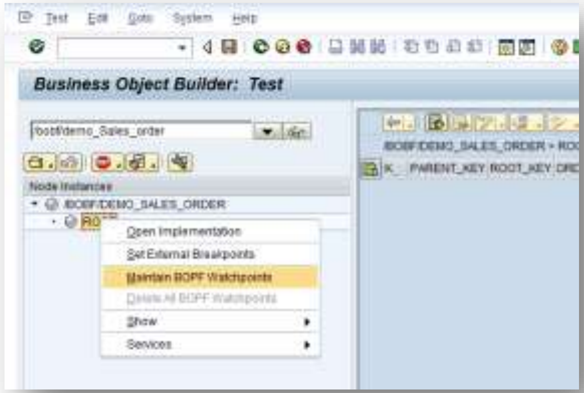
Scenario
In a complex transaction, a non-expected node attribute is somehow modified. It would interesting, which entity (e.g. determination) or consumer is responsible for this modification.

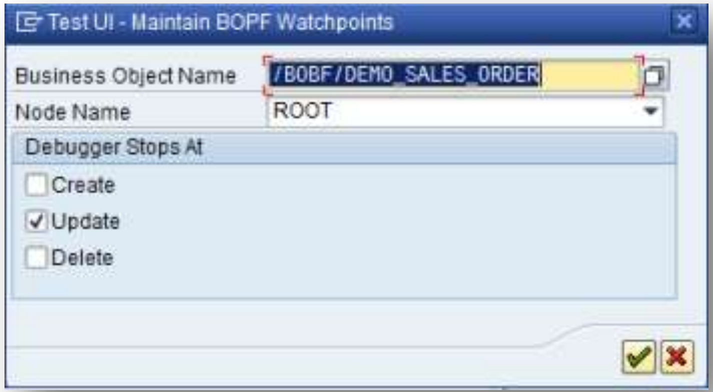
Solutions
Use the BOPF watchpoint feature of the Test UI or simulate that feature via conditional breakpoints.

Alternative Solution
The transaction can be traced in order to identify the root cause but a trace does not allow you to continue debugging at the point in time the critical modification was done.

Watchpoints

BOPF Watchpoints (Test UI)


Test Edit Goto System Help
Business Object Builder: Test
/bobf/demo_Sales_order
Node Instances
/BOBF/DEMO_SALES_ORDER
ROOT
Open Implementation
Set External Breakpoints
Maintain BOPF Watchpoints
Delete All BOPF Watchpoints
Show
Services
/BOBF/DEMO_SALES_ORDER - ROOT
PARENT_KEY ROOT_KEY

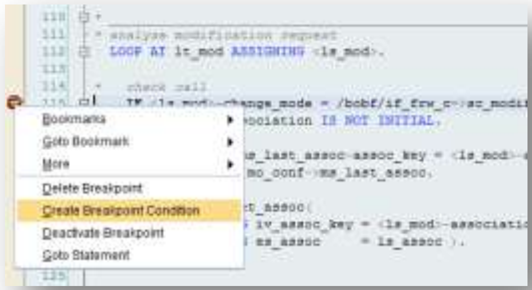


Business Object Name: /BOBF/DEMO_SALES_ORDER
Node Name: ROOT
Debugger Stops At
- [ ] Create
- [✓] Update
- [ ] Delete

1. Open the business object in the BOPF Test UI (transaction BOBT)
2. Select ”Maintain BOPF Watchpoints” in the context menu of the BO
3. Specify if you would like to debug the creation, update or deletion of a certain node instance
4. Start your scenario in the Test UI - as soon as an instance is modified and fulfills your watchpoint condition, the debugger immediately is started

Watchpoints

Manual Steps



```
*
* analyse modification request
LOOP AT lt_mod ASSIGNING <ls_mod>.

*   check call
    IF <ls_mod>-change_mode = /bobf/if_frw_c=>sc_modi
         ociation IS NOT INITIAL.
         s_last_assoc-assoc_key = <ls_mod>-
         mo_conf->ms_last_assoc.

         t_assoc(
          iv_assoc_key = <ls_mod>-associati
          es_assoc     = ls_assoc ).
```

1. Create a breakpoint at /BOBF/CL_FRW method DO_MODIFY within the „LOOP AT lt_mod ASSIGNING <ls_mod>.”
2. Start your scenario. If the breakpoint is reached, select ”Create Breakpoint Condition” in the context menue of the breakpoint
3. Apply your condition, for instance:

```
/BOBF/IF_DEMO_SALES_ORDER_C=>sc_node-ROOT = <ls_mod>-node
```

4. Continue until the conditional breakpoint is reached - in that case, you can check the callstack in order to find the entity or consumer requesting the current modification

Wachpoints

Example Breakpoint Conditions

- Stop on any modification of a ROOT node instance of BO /BOBF/DEMO_SALES_ORDER

```
/BOBF/IF_DEMO_SALES_ORDER_C=>sc_node-ROOT = <ls_mod>-node
```

- Stop on updates of any ROOT node instance of BO /BOBF/DEMO_SALES_ORDER

```
/BOBF/IF_DEMO_SALES_ORDER_C=>SC_NODE-ROOT = <LS_MOD>-NODE AND
/BOBF/IF_FRW_C=>SC_MODIFY_UPDATE = <LS_MOD>-CHANGE_MODE
```

- Stop as soon as instance ‚005056B23AD91EE3A4FD04760FCC8E6D ‘ is deleted

```
<LS_MOD>-KEY = '005056B23AD91EE3A4FD04760FCC8E6D' AND /BOBF/IF_FRW_C=>SC_MODIFY_UPDATE
= <LS_MOD>-CHANGE_MODE
```

Wachpoints
Breakpoint Conditions Hints

- Breakpoint conditions only exists as debugger breakpoints
- If you save it as session or external breakpoint, the condition is lost
- Conditions are only applied on the current session
- 255 is the max length of a condition

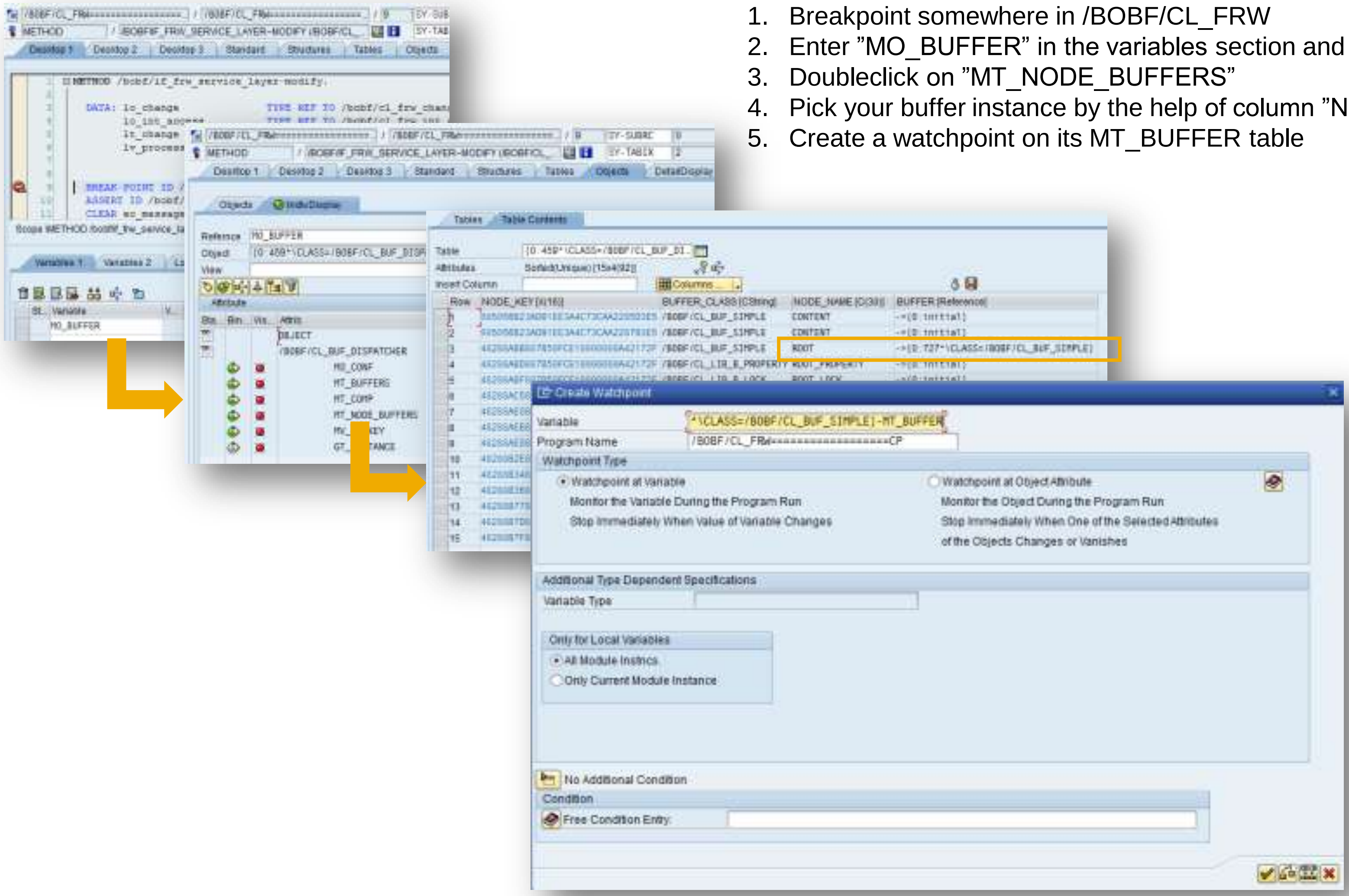
Wachpoints
Buffer Watchpoints

Not all modifications handed over to the DO_MODIFY method are successfully applied to the buffer. Imagine an update request and an update preventing action validation preventing that update. In addition, a transactional cleanup also clears your not saved instances. In order to cover the real buffer state, you can use ABAP watchpoints on your buffer instance.

Precondition: BOPF’s default buffer dispatcher and buffer implementation is used

Watchpoints

How to create a Buffer Watchpoint



Reference: MO_BUFFER
Object: {O:459*\CLASS=/BOBF/CL_BUF_DISP…

Attrib
OBJECT
/BOBF/CL_BUF_DISPATCHER
MO_CONF
MT_BUFFERS
MT_COMP
MT_NODE_BUFFERS
MV_…KEY
GT_…TANCE

Table: {O:459*\CLASS=/BOBF/CL_BUF_DI…
Attributes: Sorted(Unique) [15x4(92)]

Row	NODE_KEY [X(16)]	BUFFER_CLASS [CString]	NODE_NAME [C(30)]	BUFFER [Reference]
1	[illegible]	/BOBF/CL_BUF_SIMPLE	CONTENT	->{O:initial}
2	[illegible]	/BOBF/CL_BUF_SIMPLE	CONTENT	->{O:initial}
3	[illegible]	/BOBF/CL_BUF_SIMPLE	ROOT	->{O:727*\CLASS=/BOBF/CL_BUF_SIMPLE}
4	[illegible]	/BOBF/CL_LIB_B_PROPERTY	ROOT_PROPERTY	->{O:initial}
5	[illegible]	/BOBF/CL_LIB_B_LOCK	ROOT_LOCK	->{O:initial}

Create Watchpoint

Variable: *\CLASS=/BOBF/CL_BUF_SIMPLE}-MT_BUFFER
Program Name: /BOBF/CL_FRW==================CP

Watchpoint Type
- [✓ Watchpoint at Variable] – Monitor the Variable During the Program Run. Stop Immediately When Value of Variable Changes
- Watchpoint at Object Attribute – Monitor the Object During the Program Run. Stop Immediately When One of the Selected Attributes of the Objects Changes or Vanishes

Additional Type Dependent Specifications
Variable Type:

Only for Local Variables
- [✓ All Module Instncs]
- Only Current Module Instance

No Additional Condition
Condition
Free Condition Entry:

1. Breakpoint somewhere in /BOBF/CL_FRW
2. Enter "MO_BUFFER" in the variables section and doubleclick
3. Doubleclick on "MT_NODE_BUFFERS"
4. Pick your buffer instance by the help of column "NODE_NAME"
5. Create a watchpoint on its MT_BUFFER table

Property Errors

Property Errors

Scenario
The properties of an entity are not displayed as expected. For instance, an action is disabled but should be enabled.

Solution

1. Check the different kind of properties by the help on their priority (see next slide). For instance, start with the final static properties first.

2. You can use the SM12 to check the lock dependent properties (see chapter „Lock Errors“ for details).

3. If you reach the dynamic properties, check the corresponding node in the BOPF configuration UI and put a breakpoint in all property determinations on that node (those are before retrieve determinations having the property node as request node). Execute your scenario and check, if there is a property determination creating the unexpected property.

Priority of Properties

1. **Final static properties**
   (Configured in the node category configuration of `/BOBF/CONF_UI`.)
2. **Authority-dependent properties**
   (Automatically set by BOPF.)
3. **Lock-dependent properties**
   (Automatically set by BOPF.)
4. **Status and action management**
   (Automatically received by BOPF and configured at S&AM design time.)
5. **Application-specific dynamic poperties: Subtree properties**
   (Set using dynamic property determination and the helper class.)
6. **Other application-specific dynamic properties**
   (Set using dynamic property determination and the helper class.)
7. **Non-final static properties**
   (Configured in the node category configuration of `/BOBF/CONF_UI`.)

Examples

- S&AM properties override application-specific properties, but are overridden by lock-dependent properties.
- Properties for which the final flag was set can never be overruled by any other properties. Set the final flag with care.
- Non-final static properties are the weakest; use them as a kind of default property.

Update Task Errors

Update Task Errors

1. A dump occurs while saving a transaction in function module „/BOBF/CL_DAC_UPDATE“ (e.g. duplicate insert).

2. Check the „application information“ section in the ST22 dump for further details

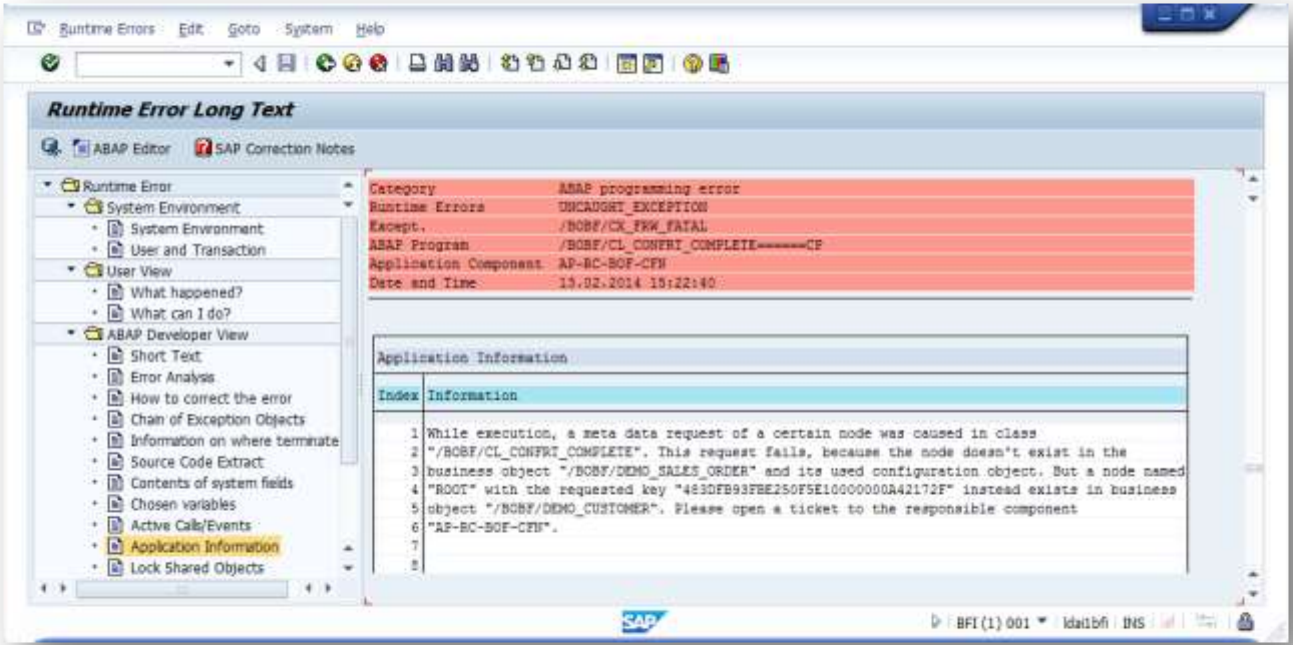
3. Common root causes
   - An attribute has a unqiue index on its database field. But there is no unique alternative key defined on it or the unqiueness validation is missing (/BOBF/CL_LIB_V_UNQIUE_ALT_KEY). Thus multiple instances could have the same attribute value. Add the validation to prevent those dumps.
   - The deletion of an instance fails as in a parallel session the instance was already deleted by a non-BOPF application.

Application Dumps

Application Dumps
Dump Pattern & Application Information Section

```
DATA lo_conf TYPE REF TO /bobf/if_frw_configuration.
lo_conf = /bobf/cl_frw_factory=>get_configuration(
  iv_bo_key = /bobf/if_demo_sales_order_c=>sc_bo_key ).
lo_conf->get_node(
  EXPORTING
    iv_node_key = /bobf/if_demo_customer_c=>sc_node-root
  IMPORTING
    es_node     = DATA(ls_root_of_sales_order) ).
```

Using wrong configuration object is a common error…

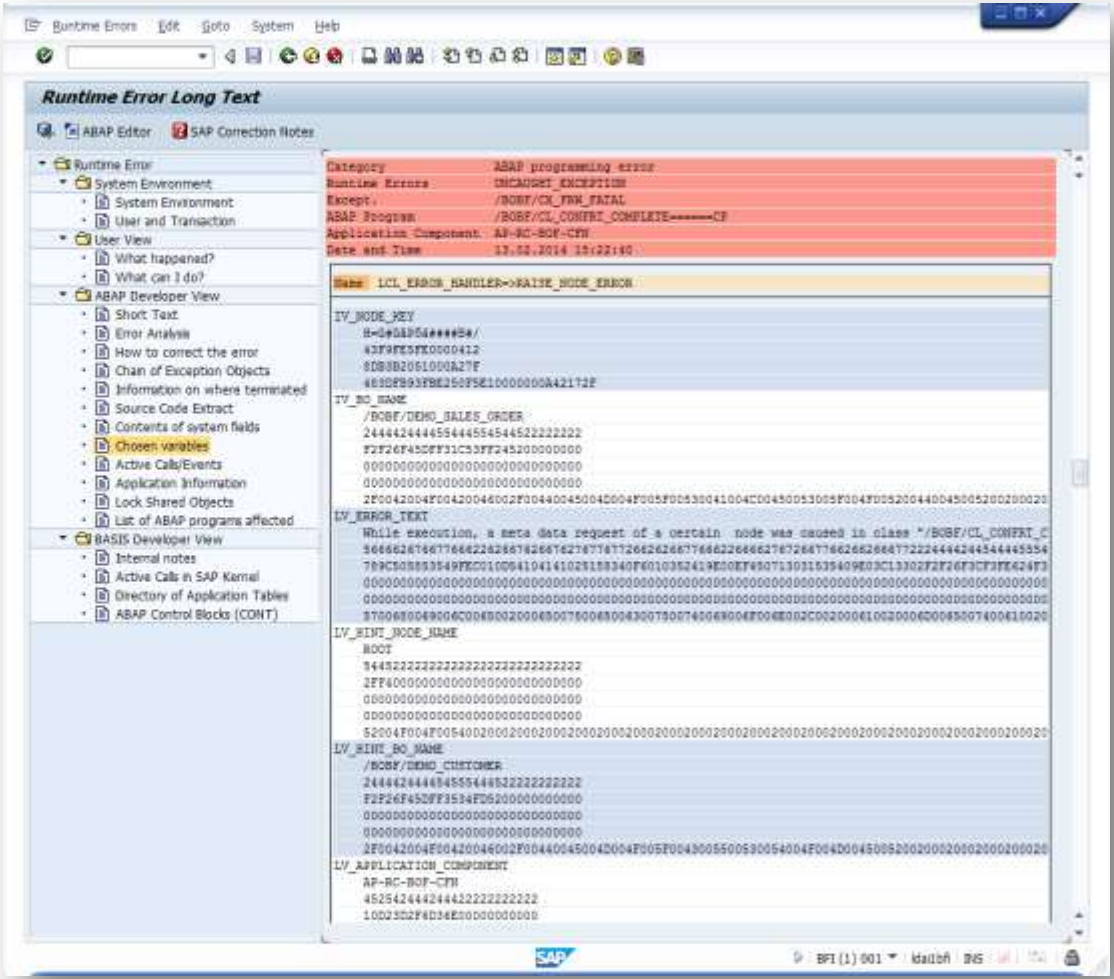


```
Category               ABAP programming error
Runtime Errors         UNCAUGHT_EXCEPTION
Except.                /BOBF/CX_FRW_FATAL
ABAP Program           /BOBF/CL_CONFRT_COMPLETE======CP
Application Component  AP-RC-BOF-CFN
Date and Time          13.02.2014 15:22:40
```

```
Application Information

Index Information
    1 While execution, a meta data request of a certain node was caused in class
    2 "/BOBF/CL_CONFRT_COMPLETE". This request fails, because the node doesn't exist in the
    3 business object "/BOBF/DEMO_SALES_ORDER" and its used configuration object. But a node named
    4 "ROOT" with the requested key "483DFB93FBE250F5E10000000A42172F" instead exists in business
    5 object "/BOBF/DEMO_CUSTOMER". Please open a ticket to the responsible component
    6 "AP-RC-BOF-CFN".
    7
    8
```

… BOPF collects detail information and provides useful hints in the application information section

Application Dumps
Chosen Variables

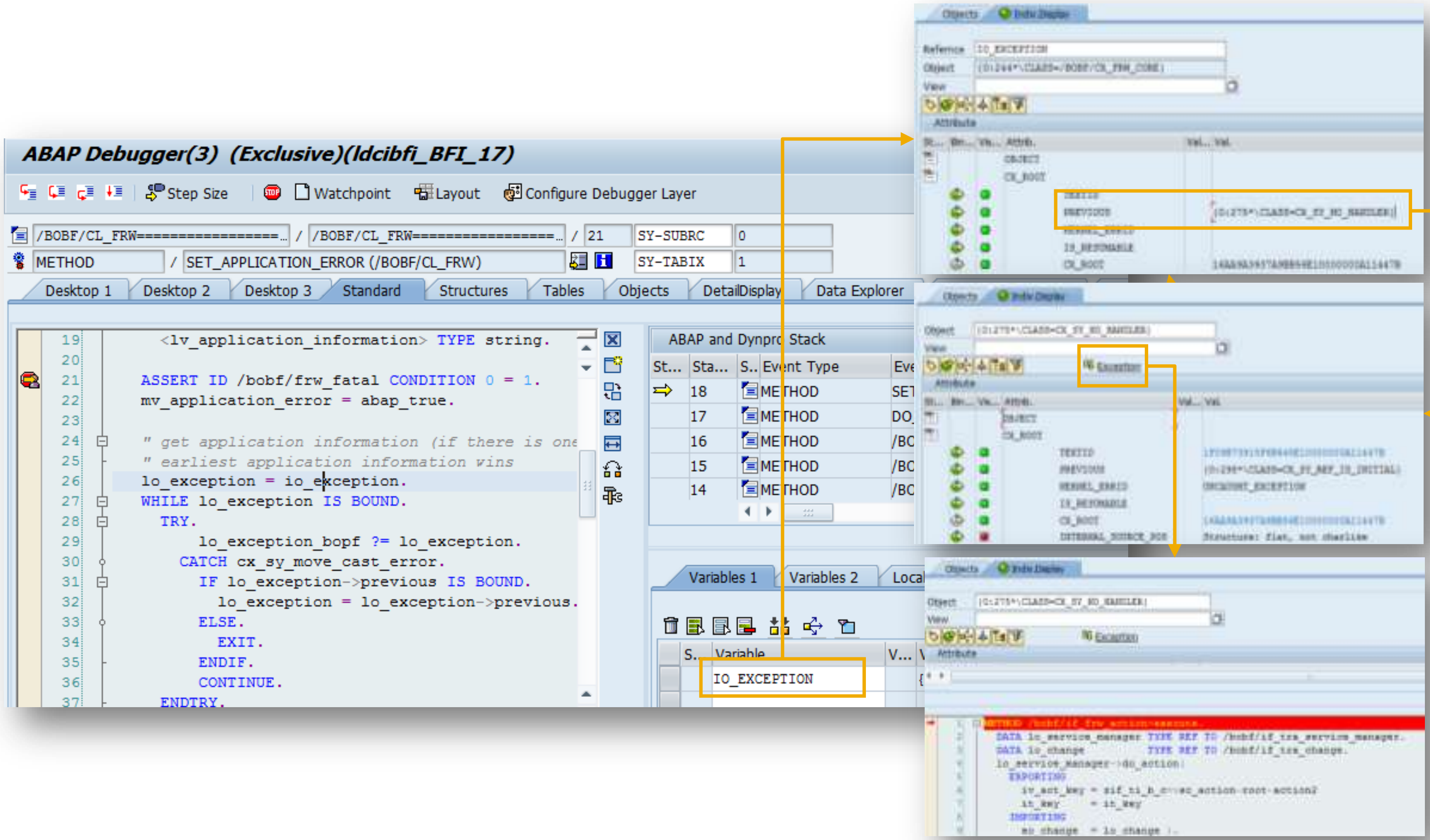


```
Category              ABAP programming error
Runtime Errors        UNCAUGHT_EXCEPTION
Except.               /BOBF/CX_FRW_FATAL
ABAP Program          /BOBF/CL_CONFRT_COMPLETE======CP
Application Component AP-RC-BOF-CFN
Date and Time         13.02.2014 15:22:40

Name  LCL_ERROR_HANDLER->RAISE_NODE_ERROR

IV_NODE_KEY
    H=d#DäP5ä####5#/
    43F9FE5FE0000412
    8D83B2061000A27F
    463DFB93FBE250F5E10000000A42172F
IV_BO_NAME
    /BOBF/DEMO_SALES_ORDER
    2444424444554445545445222222222
    F2F26F45DFF31C53FF245200000000
    000000000000000000000000000000
    000000000000000000000000000000
    2F0042004F00420046002F0044004500400004F005F00530041004C00450053005F004F0052004400450052002000200
LV_ERROR_TEXT
    While execution, a meta data request of a certain  node was caused in class "/BOBF/CL_CONFRT_C
    5666626766776662262667626676277677726626266776662266662767266776626626667722244442445444455554
    789C508853549FEC010D641041410251883 40F601035241 9E00EF450713031635409E03C13302F2F26F3CF3FE424F3
    000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    57006800690006C006800200065007500650004300750074006900 4F004E002C002000610020006D00650074006100610020
LV_HINT_NODE_NAME
    ROOT
    5445222222222222222222222222222
    2FF400000000000000000000000000
    000000000000000000000000000000
    000000000000000000000000000000
    52004F004F005400200020002000200020002000200020002000200020002000200020002000200020002000200020
LV_HINT_BO_NAME
    /BOBF/DEMO_CUSTOMER
    2444424444545555444522222222222
    F2F26F45DFF3534FD6200000000000
    000000000000000000000000000000
    000000000000000000000000000000
    2F0042004F00420046002F0044004500 4D004F005F0043005500530054004F004D004500520020002000200020002000200
LV_APPLICATION_COMPONENT
    AP-RC-BOF-CFN
    4525424444244422222222222
    10D23D2F6D34E000000000000
```

Information about the affected BO, node and keys:

IV_NODE_KEY
IV_BO_KEY
LV_HINT_NODE_NAME
LV_HINT_BO_NAME
IT_KEY

Application Dumps
Navigate to the Exception Root Cause



ABAP Debugger(3) (Exclusive)(ldcibfi_BFI_17)

```
/BOBF/CL_FRW==================    /BOBF/CL_FRW==================  / 21   SY-SUBRC  0
METHOD                          / SET_APPLICATION_ERROR (/BOBF/CL_FRW)   SY-TABIX  1
```

```
      <lv_application_information> TYPE string.

    ASSERT ID /bobf/frw_fatal CONDITION 0 = 1.
    mv_application_error = abap_true.

    " get application information (if there is one
    " earliest application information wins
    lo_exception = io_exception.
    WHILE lo_exception IS BOUND.
      TRY.
          lo_exception_bopf ?= lo_exception.
        CATCH cx_sy_move_cast_error.
          IF lo_exception->previous IS BOUND.
            lo_exception = lo_exception->previous.
          ELSE.
            EXIT.
          ENDIF.
          CONTINUE.
      ENDTRY.
```

ABAP and Dynpro Stack

St...	Sta...	S.. Event Type	Eve
⇨	18	METHOD	SET
	17	METHOD	DO
	16	METHOD	/BC
	15	METHOD	/BC
	14	METHOD	/BC

S...	Variable	V...
	IO_EXCEPTION	{

```
METHOD /bobf/if_frw_action~execute.
  DATA lo_service_manager TYPE REF TO /bobf/if_tra_service_manager.
  DATA lo_change          TYPE REF TO /bobf/if_tra_change.
  lo_service_manager->do_action(
    EXPORTING
      iv_act_key = zif_ti_b_c=>sc_action-root-action2
      it_key     = it_key
    IMPORTING
      eo_change  = lo_change ).
```

1. Set a Breakpoint in /BOBF/CL_FRW->SET_APPLICATION_ERROR()

Questions?

Thank you

© 2012 SAP AG. All rights reserved.