The Enhancement Framework
for Nordic User group
Contributing Speakers

Michael Acker
Development Architect, NW AS ABAP Workbench, SAP AG

Dr. Roland Krämer
Senior Developer, NW AS ABAP Workbench, SAP AG

This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice. SAP assumes no responsibility for errors or omissions in this document.
Learning Objectives

As a result of this workshop, you will be able to

- Understand the fundamental idea of the Enhancement Framework and Switch Framework
- Reduce TCO by using enhancement technologies instead of modifications
- Enhance SAP standard objects
- Create your own Enhancement Spots for your own source code
- Work with enhancements during an upgrade or import of a support package
Enhancement Framework Overview
Source Code Plugin – Technology
Function Group Enhancement – Technology
Class Enhancement – Technology
BAdI – Technology
Upgrade Adjustment
Switch Framework
Summary
Enhancement Framework Overview

Source Code Plugin – Technology
Function Group Enhancement – Technology
Class Enhancement – Technology
BAdI – Technology
Upgrade Adjustment
Switch Framework
Summary
Adapting SAP Software

One of the advantages of SAP software is the possibility to adapt the software to own requirements and the possibility of keeping the adaptations during upgrade.

Ways of adaptation:

- Customizing
- Enhancement
- Modification
Motivation

Reducing TCO

- Enhancing objects instead of modifying them reduces the effort for adjustment during SP import or upgrade.

Disadvantages of modifications

- No support for multiple users or projects
- No grouping
- No support for parallel developments
- Will appear much more often in adjustment tools
- Higher adjustment effort
Motivation: Goals of Enhancement Framework

Integration of several enhancement types

- BAdIs
- Functiongroup Enhancement
- Class/Interface Enhancement
- Source Code PlugIns
- WebDynpro Enhancement
- ...

into the Enhancement Framework

- Switchable by Switch Framework
- Enhancement Browser
- Upgrade support
- Possibility to document and group enhancements
- Multilayer support
Enhancements – Relations

- [Simple] Enhancement Spot
  - n to [Simple] Enhancement Implementation
  - 1

- Composite Enhancement Spot
  - n to Composite Enhancement Implementation
  - 1

- [Simple] Enhancement Implementation
  - n

- Composite Enhancement Implementation
  - n
Composite Enhancement Spots
- Container Objects
- Can contain
  - Other Composite Enhancement Spots
  - [Simple] Enhancement Spots

[Simple] Enhancement Spots
- Container Objects
- Can contain Enhancement Elements

Explicit Enhancement Options
- Enhancement Definitions

Enhancement Elements:
Such as a BAdI-Definition
Composite Enhancement Implementation

- Container Objects
- Can contain
  - Other Composite Enhancement Implementations
  - [Simple] Enhancement Implementations

[Simple] Enhancement Implementation

- Can contain Enhancement Implementation Elements

Enhancement Implementation Elements

- Implementations
Editor for Predefined Enhancement Possibilities (Source Code Enhancements & BAdIs)

Integrated in Object Navigator (SE80)

common tabs for all Enhancement Spots

Tab 3 ➔ dependent on enhancement technology: BAdls or Source Code Enhancements
Editor for Enhancement Implementations

Integrated in Object Navigator (SE80)

Tab Properties & Objects ➞ common for all enhancement types

Tab 3 ➞ dependent on enhancement technology:
  e.g. BAdI-Implementation or Source Code Enhancements
### Composite Enhancement Implementation

**Composite Enh. Implementation**  
CIMPL_01  
Active

**Short Text**  
TEST

#### General Data

<table>
<thead>
<tr>
<th>Created By</th>
<th>DANGERS</th>
<th>Last Changed By</th>
<th>DANGERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created On</td>
<td>15.06.2005</td>
<td>Changed On</td>
<td>15.06.2005</td>
</tr>
<tr>
<td>Original Language</td>
<td>EN English</td>
<td>Paket</td>
<td>$TMP</td>
</tr>
</tbody>
</table>

#### Enhancement Implementations

<table>
<thead>
<tr>
<th>Enhancement Implementation</th>
<th>Status</th>
<th>Short Text</th>
<th>Enhancement Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIMPL_01</td>
<td></td>
<td>test</td>
<td>Redefinition</td>
</tr>
</tbody>
</table>

Contains [Simple] Enhancement Implementations

- **General Data**
  - Created By: DANGERS  
  - Created On: 15.06.2005  
  - Original Language: EN (English)

- **Enhancement Implementations**
  - CIMPL_01
    - Status:  
    - Short Text: test
    - Enhancement Type: Redefinition

Contains Composite Enhancement Implementations
Terminology by Example

Composite Enhancement Spot

ERP

Composite Enhancement Spot

SD

Composite Enhancement Spot

Invoice

Composite Enhancement Spot

Pricing

Composite Enhancement Implementation

IS_OIL

Composite Enhancement Implementation

Upstream

Composite Enhancement Impl.

OIL_PRICING

Enhancement Implementation

Functiongroup Enhancement
FUGR V60A

Enhancement Implementation

Source Code Plugin
FUGR V60A

Enhancement Implementation

BAAdv Definition
PRICING_BAD1

Enhancement Spot

Source Code Enhancement
FUGR V60A

Enhancement Spot

BAAdv Implementation
OIL_PRICING_IMPL1
Differences: Implicit and Explicit Enhancement Options

Features of explicit enhancement options

- More stable, pre-defined
- Few changes in definition to expect
- Only at valid source code locations

Features of implicit enhancement options

- Enhancement of „arbitrary“ objects
- No enhancement spots necessary
Search for

- Enhancements possibilities
- Existing Enhancement Implementations
- Enhancement Implementations to be adjusted after upgrade
# System Change Option – Enhanceable

**New option:**

**Enhanceable** → Only creation of enhancement implementations is allowed

<table>
<thead>
<tr>
<th>Component</th>
<th>Technical Name</th>
<th>Modifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancements</td>
<td>HOME</td>
<td>Modifiable</td>
</tr>
<tr>
<td>Enhancements (No Automatic Transport LOCAL)</td>
<td>LOCAL</td>
<td>Modifiable</td>
</tr>
<tr>
<td>PI BASIS 2005_1_700</td>
<td>PI BASIS</td>
<td>Modifiable</td>
</tr>
<tr>
<td>Cross-Application Component</td>
<td>SAP_ABA</td>
<td>Restricted Modifiability</td>
</tr>
<tr>
<td>SAP Basis Component</td>
<td>SAP_BASIS</td>
<td>Restricted Modifiability</td>
</tr>
<tr>
<td>SAP NetWeaver BI 7.0</td>
<td>SAP_BW</td>
<td>Restricted Modifiability</td>
</tr>
</tbody>
</table>

© SAP AG 2006, Enhancement Framework/ 19
PROGRAM p1.
WRITE 'Hello World'.
ENHANCEMENT-POINT ep1 SPOTS s1.
... 
ENHANCEMENT-SECTION ep2 SPOTS s1.
  WRITE 'Original'.
END-ENHANCEMENT-SECTION.

ENHANCEMENT 1.
  WRITE 'Hello Paris'.
ENDENHANCEMENT.

ENHANCEMENT 2.
  WRITE 'Hello London'.
ENDENHANCEMENT.

ENHANCEMENT 3.
  WRITE 'Enhanced'.
ENDENHANCEMENT.
Source Code Enhancements Overview

Modification-free enhancement of source code

Explicit Enhancement Option

- Predefined enhancement options can be defined in source code. They are additionally stored inside Enhancement Spots.

Implicit Enhancement Option

- At common enhancement places, implicit Enhancement options are available. Examples:
  - Beginning/End of Include
  - Beginning/End of Method/Function Module/Form Routine
  - End of a structure
  - End of Private/Protected/Public Section of a local class
  - ...
Explicit Enhancement Options

* Selection screen
SELECT-OPTIONS:
  so_carr FOR gv_carrid,
  so_conn FOR gv_connid.

ENHANCEMENT-POINT FLIGHTS_DECLARATION SPOTS FLIGHTS_DISPLAY STATIC.

START-OF-SELECTION.

* Select Data

ENHANCEMENT-SECTION   FLIGHTS_DBSELECT SPOTS FLIGHTS_DISPLAY.
  SELECT carrid connid fldate price currency
     from sflight
     into table gt_flights
     where carrid in so_carr
       and connid in so_conn.
END-ENHANCEMENT-SECTION.

* Listoutput

  LOOP AT gt_flights INTO gs_flights.
   WRITE: / gs_flights-carrid,
      gs_flights-connid,
      gs_flights-fldate,
Implicit Enhancement Options

LISTOUTPUT

LOOP AT gt_flights INTO gs_flights.
WRITE: / gs_flights-carrid,
gs_flights-commid,
gs_flights-fldate,
gs_flights-price,
gs_flights-currency.
ENCLoop.

METHOD SET_PROPERTY.

PROPERTY, pt, pls_pcount_queue_only; => VALUE

check handle:
DATA: STRPROPNAME TYPE STRING.
DATA: STRVALUE TYPE STRING.
DATA: QUEUE_EMPTY TYPE CHAR01.

CLASS CL_SUI_OBJECT.
METHOD set_property.

CLASS CL_SUI_CONTROL.
METHOD set_property.

CLASS CL_SUI_PRIORITY.
METHOD set_property.

CLASS CL_SUI_CHAR.
METHOD set_property.

CLASS CL_SUI_TYPE.
METHOD set_property.

CLASS CL_SUI_ACCESS.
METHOD set_property.

CLASS CL_SUI.currentState.
METHOD set_property.

CLASS CL_SUI_DATA.
METHOD set_property.

CLASS CL_SUI_INPUT.
METHOD set_property.

CLASS CL_SUI_OUTPUT.
METHOD set_property.

CLASS CL_SUI_DISPLAY.
METHOD set_property.

CLASS CL_SUI_GRAPH.
METHOD set_property.

CLASS CL_SUI_LINK.
METHOD set_property.

CLASS CL_SUI_ACTION.
METHOD set_property.

CLASS CL_SUI_COMMUNICATION.
METHOD set_property.

CLASS CL_SUI_VARIATION.
METHOD set_property.

CLASS CL_SUI_REPOSITORY.
METHOD set_property.
Enhancement Types in Source Code

ENHANCEMENT-POINT <name> SPOTS <spot1> [<spot2>] .. [STATIC]

- Static – e.g. additional data declaration
- Dynamic – e.g. additional source code

ENHANCEMENT-SECTION <name> SPOTS <spot1> [<spot2>] .. [STATIC]

- Static – e.g. replace an existing data declaration
- Dynamic – e.g. replace source code
Enhancement Spots & Implementations

Compilation Unit

Physical Part of Compilation Unit

Assigned to Compilation Unit

- Point A
- Spot Hugo
- Section B
- Point F
- Spot Flights
- Point D
- Point C
- Point E
**Enhancement Spots & Implementations**

**Compilation Unit**

- **Spot Hugo**
  - **Point A**
  - **Section B**
  - **Point F**
  - **Point C**
  - **Point D**
  - **Point E**

- **[Simple] Enhancement Implementation 1**
- **[Simple] Enhancement Implementation 2**

- **Physical Part of Compilation Unit**
- **Assigned to Compilation Unit**
Enhancement Spots & Implementations

Compilation Unit

- **Point A**
- **Spot Hugo**
  - **Section B**
    - **Point C**
    - **Spot Flights**
    - **Point D**
    - **Point E**
  - **Point F**

- **[Simple] Enhancement Implementation 1**
- **[Simple] Enhancement Implementation 2**
- **[Simple] Enhancement Implementation 3**

Physical Part of Compilation Unit

Assigned to Compilation Unit

© SAP AG 2006, Enhancement Framework/29
Overlay Enhancement

Existing Enhancements must not be modified. Enhance the Enhancement instead!

Example:
An SAP Program is enhanced by an Industry Solution.

The source code of the Industry Solution enhancement is exchanged by a customer enhancement.
Editor Modes for Enhancements

Use **Change Mode** for creating enhancement points & sections.

- use button ”Display <-> Change” to switch to change mode.

Use **Enhancement Mode** for creating enhancement implementations.

- use button ”Change Enhancements” to switch to Enhancement mode
- use button ”Display <-> Change” to leave Enhancement mode
Function Group Enhancements allow

- Adding new optional parameters to existing function modules

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Type</th>
<th>Default Value</th>
<th>Opt</th>
<th>Description</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_VEBELN</td>
<td>LIKE VEBELN</td>
<td></td>
<td></td>
<td>Document number</td>
<td></td>
</tr>
<tr>
<td>F_XVBAP_SORTIER</td>
<td></td>
<td></td>
<td></td>
<td>X= items table is sorted (AVBAP)</td>
<td></td>
</tr>
<tr>
<td>F_XVBUP_SORJECKSICH</td>
<td></td>
<td></td>
<td></td>
<td>X= Determine completeness status</td>
<td></td>
</tr>
<tr>
<td>F_FPLNR</td>
<td>LIKE FPLNR</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F_KFPLT_SORJECKSICH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F_REQ_BLOCK</td>
<td>LIKE SPEBE</td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F_BESTA_ERMITTELN</td>
<td></td>
<td>SPACE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F_VORSAENDER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F_DOSTA_ERMITTELN</td>
<td>LIKE DELOF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F_LIFSK</td>
<td>LIKE LIFSK</td>
<td></td>
<td></td>
<td>Lieferscheinse Belegkopf</td>
<td></td>
</tr>
<tr>
<td>F_BEL_BUMP</td>
<td>LIKE BUMP</td>
<td>SPACE</td>
<td></td>
<td>High-Tech-Bump-Kennzeichung</td>
<td></td>
</tr>
<tr>
<td>F_BOS_PACKAGE_NO</td>
<td>LIKE BOS</td>
<td></td>
<td></td>
<td>Pakenummer</td>
<td></td>
</tr>
<tr>
<td>F_BOS_VORK_FTYP</td>
<td>LIKE FTYP</td>
<td></td>
<td></td>
<td>Fakturtyp</td>
<td></td>
</tr>
</tbody>
</table>
### Additional Parameters

Add optional parameters to existing function modules

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Type</th>
<th>Reference type</th>
<th>Default value</th>
<th>Opt.</th>
<th>Pa.</th>
<th>Shorttext</th>
<th>Lo.</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIRLINECARRIER</td>
<td>LIKE</td>
<td>BAPI SFDATA-CAR</td>
<td></td>
<td></td>
<td></td>
<td>Carrier ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONNECTIONNUMBER</td>
<td>LIKE</td>
<td>BAPI SFDATA-CON</td>
<td></td>
<td></td>
<td></td>
<td>Connection number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATEOFFSETLIGHT</td>
<td>LIKE</td>
<td>BAPI SFDATA-FLD</td>
<td></td>
<td></td>
<td></td>
<td>Departure date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARAMETER_1</td>
<td>TYPE</td>
<td>STRING</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CD_FB_1</td>
</tr>
</tbody>
</table>
Class/Interface Enhancements allow addition of:

- optional parameters to existing methods
- methods
- events and event handlers
- references to interfaces
- types*
- exits to existing methods
  - Pre-Exit – Called at the beginning of a method
  - Post-Exit – Called at the End of a method
  - Overwrite-Exit – Replaces the original method
  - access to private and protected attributes of the original class*

* as from next release
### Add optional parameters to existing methods

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>P</th>
<th>O</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Default value</th>
<th>Description</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJ_NAME</td>
<td>Import</td>
<td></td>
<td></td>
<td></td>
<td>TROBJ_NAME</td>
<td></td>
<td>Object Name in Object Directory</td>
<td></td>
</tr>
<tr>
<td>PROTOCOL</td>
<td>Change</td>
<td></td>
<td></td>
<td></td>
<td>SPROT_U_TAB</td>
<td></td>
<td>Table Type for SPROT_U (Log-In)</td>
<td></td>
</tr>
<tr>
<td>MY_ADD_PARAM</td>
<td>Import</td>
<td></td>
<td></td>
<td></td>
<td>CHAR10</td>
<td></td>
<td>Characterfeld der Länge 10</td>
<td>MATECHED2605_061</td>
</tr>
<tr>
<td>MY_ADD_PARAM_EXP</td>
<td>Export</td>
<td></td>
<td></td>
<td></td>
<td>INT4</td>
<td></td>
<td>Natürliche Zahl</td>
<td>MATECHED2605_061</td>
</tr>
</tbody>
</table>
# Additional Methods

## Add new methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Level</th>
<th>Visibility</th>
<th>Description</th>
<th>PreExit</th>
<th>PostExit</th>
<th>Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER_IMPORT</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD_METHOD_ONE</td>
<td>Instance</td>
<td>Private</td>
<td>Additional functionality public</td>
<td></td>
<td></td>
<td>MATECHD2365_001</td>
</tr>
<tr>
<td>ADD_METHODPRIVATE</td>
<td>Instance</td>
<td>Private</td>
<td>Additional functionality private</td>
<td></td>
<td></td>
<td>MATECHD2365_001</td>
</tr>
</tbody>
</table>
... Call method instance->hugo().
...

Method Hugo.

source code.
source code.
source code.

Endmethod.

Method Pre.
...
....
Endmethod.

Method Post.
....
....
Endmethod.
Enhancement Framework Overview
Source Code Plugin – Technology
Function Group Enhancement – Technology
Class Enhancement – Technology
BAdI – Technology
Upgrade Adjustment
Switch Framework
Summary
A BAdI

- is an anticipated point of extension – these points act like sockets and exist in the original source code
- is a predefined anchor for an Object PlugIn
- has a well-defined interface in contrast to source code plugIns and is therefore more stable to changes in the original source code
- has switchable implementation(s) (by switches introduced by the Switch Framework)
The calculation depends on the filter value. Only one active implementation is allowed for one filter value.

```
GET BADI mytaxbadi
  FILTERS LANGUAGE = sy-langu.
 ..
CALL BADI mybadi->add_tax
  EXPORTING
    value = my_value
  IMPORTING
    result = my_result.
 ..
write / my_result.
```

```
METHOD if_mybadi~add_tax.
  language = 'XY'
  result = value + myfunc(value).
ENDMETHOD.
```

```
METHOD if_mybadi~add_tax.
  language = 'DE'
  result = value * 1.16.
ENDMETHOD.
```
The BAdI allows to perform an additional step inside a standard process. The core functionality doesn’t need any implementation, but the implementation may react on a core event. Several implementations may be called in a sequence.

GET BADI mybadi.

.. CALL BADI mybadi->value_changed
   EXPORTING
   value = my_value

..
Use the advantages (performance, upgrade adjustment) to implement a registry of classes.

```plaintext
GET BADI myregbadi.
  FILTERS reg = 'DEF'
  ..
  CALL BADI myregbadi->do
  ..

METHOD
  if_mybadi~get_result.
    do ..
  ENDMETHOD.

METHOD
  if_mybadi~get_result.
    do ..
  ENDMETHOD.
```

<table>
<thead>
<tr>
<th></th>
<th>ABC</th>
<th>DEF</th>
<th>XYZ</th>
<th>MER</th>
<th>FGH</th>
</tr>
</thead>
</table>
New Features – Overview

New BAdIs – New Features

- Are integrated directly in the ABAP Language/Runtime
- Improved filter support allows non-character filter types (packed, numeric, string) and complex filter conditions
- Enable reusable implementation instances
- Different kinds of default implementations
- Control of the lifetime of implementations (BAdl-context)
- Allow for inheritance of implementations
Comparison: Usage of Old BAdIs vs. New BAdIs

**With Classic BAdI**

```plaintext
DATA: bd TYPE REF TO if_intf.
DATA: flt TYPE flt.

CALL METHOD cl_exithandler=>
  get_instance
  EXPORTING
    exit_name = `BADI_NAME`.
  CHANGING
    instance = bd.

flt-lang = `D`.
CALL METHOD bd->method
  EXPORTING
    x = 10
    flt_val = flt.
```

Selecting implementations and issuing calls is mixed
Calls are redirected over a proxy class

**With New BAdI**

```plaintext
DATA: bd TYPE ref to badi_name.
get badi bd filters lang = `D`.
call badi bd->method
  exporting x = 10.
```

Selection occurs when the handle is requested
Implementations are called directly (without a proxy)
The new BAdI evaluates as much information as possible during compile time.

Better Performance/Lower Memory consumption

- Database access only at compile time
- Statically typed comparisons at runtime
- Internal handle-class integrated in SAP Kernel

➔ 40-600 times faster than Classic BAdIs
Active implementations are evaluated at compile time and included in the load of the BAdI-handle.

- Switch Framework – only implementations, that are switched on in at least one client are considered.
- Constant Filter values – Implementations that do not match the filter are excluded
- Active Flag – Only active implementations are considered

Special optimizations for BAdIs with

- no active implementations:
  - ‘GET BADI’ is ignored while compiling
  - ‘CALL BADI’ just needs the time for a simple if-statement.
- one active implementation
  - Direct call of that implementation
**Integration Into ABAP (1)**

**BAdIs are represented by a reference to BAdI-Handles:**

```plaintext
DATA bd type ref to badi_name.
GET BADI bd FILTER f = 5.
```

If there are two implementations of badi_name that are selected for the filter value f=5, this yields:

![Diagram showing two instances of Cl_imp, each connected to bd and badi_name.]
CALL BADI bd->m exporting ... importing ...

CALL BADI bd->m exporting ... importing ...

Inst1

Inst2

Inst2

bd

Inst1

Inst2

Inst2
BAdI-Definition and Enhancement Spot

Definition

- BAdI Definition A
- BAdI Definition B
- BAdI Definition C

Implementation

- BAdI Implementation 1
- BAdI Implementation 2
- BAdI Implementation 2
- BAdI Implementation 2
- BAdI Implementation 3
BAdI-Definition and Enhancement Spot

**Definition**
- Simple Enhancement Spot Spot_1
- BAdI Definition A
- BAdI Definition B
- BAdI Definition C

**Implementation**
- BAdl Implementation 1
- BAdl Implementation 2
- BAdl Implementation 3
Enhancement Spots and Enhancement Implementation (2)

**Definition**

- Simple Enhancement Spot Spot_1
- BAdI BADI_A
- BAdI BADI_B
- BAdI BADI_C

**Implementation**

- Package A
  - BAdI Implementation 1
  - BAdI Implementation 2
- Package B
  - BAdI Implementation 3
  - BAdI Implementation 4
  - BAdI Implementation 5

© SAP AG 2006, Enhancement Framework/ 56
Enhancement Spots and Enhancement Implementation (2)

**Definition**
- Simple Enhancement Spot Spot_1
- BAdI BADI_A
- BAdI BADI_B
- BAdI BADI_C

**Implementation**
- BAdI Implementation 1
  - Simple Enhancement Implementation SEI1
  - BAdI Implementation 2
- BAdI Implementation 3
- BAdI Implementation 4
  - Simple Enhancement Implementation SEI2
- BAdI Implementation 5

**Switches**
- Switch_1
- Switch_2

© SAP AG 2006, Enhancement Framework/57
Enhancement Spots and Enhancement Implementation (5)

**Definition**

- Simple Enhancement Spot Spot_1
  - BAdI BADI_A
  - BAdI BADI_B
  - BAdI BADI_C

- Simple Enhancement Spot Spot_2
  - BAdI BADI_E

**Implementation**

- BAdI Implementation 1
  - Simple Enhancement Implementation SEI1
  - BAdI Implementation 2
  - BAdI Implementation 3

- BAdI Implementation 4
  - Simple Enhancement Implementation SEI2
  - BAdI Implementation 5
  - BAdI Implementation 6
Default Implementations

Types of Implementations

- A BAdI-Definition may have an associated fallback class
- A BAdI-Implementation is either default or non-default

Selection Procedure: (during GET BADI)

1. Apply selection to all non-default implementations
2. If nothing has been selected, apply selection to all default implementations
3. If still nothing has been selected and there is a fallback class, take the default class
Together with the BAdI, abstract or normal implementation classes may be supplied.

`cl_imp1` has to implement all interface methods.

`cl_imp2`, ... implement methods not supplied by `cl_abstract_imp` or override some methods.

It is also allowed to inherit from the default class or example class, if they are not defined as final.
Sharing Data Within BAdIs

There are two BAdIs \texttt{badi\_name1} and \texttt{badi\_name2}, with interfaces \texttt{if\_intf1} and \texttt{if\_intf2}, respectively.

\texttt{cl\_imp} implements both of these interfaces.

\begin{verbatim}
DATA: bd1 TYPE ref to badi\_name1,
     bd2 TYPE ref to badi\_name2.

GET BADI bd1.
GET BADI bd2.
\end{verbatim}
Types of Instantiations

- Ever new instances are created (Scenario 1)
- Per implementation class there is one instance (Scenario 2)
- Per context reference and implementation class there is one instance
DATA bd TYPE ref to badi_name.

GET BADI bd1.
DATA: bd1 TYPE ref to badi_name,
    bd2 TYPE ref to badi_name.

GET BADI bd1.

GET BADI bd2.
DATA: bd1 TYPE ref to badi_name.

GET BADI bd1.
Instantiation With Option Reuse

DATA: bd1 TYPE ref to badi_name,
     bd2 TYPE ref to badi_name.

GET BADI bd1.
GET BADI bd2.

With a context there is one instance per context reference and implementation class.

A context class is a class implementing if_badi_context, a context reference is a reference to an instance of a context class.

In ABAP:
```
DATA ctx TYPE REF TO cl_myctx.
...
GET BADI bd CONTEXT ctx.
```

Passing the same context yields identical implementation instances.
BAdI Implementations assign themselves to a given BAdI Context:

- Lifetime of implementations is bound to the lifetime of the context.
- Whenever ‘GET BADI’ is called with the same context and a filter value which leads to the same implementation implementation class, the already created instance is used.
- Performance improvement
- Holds data over different method calls or even BAdls.
DATA:  
   bd1  TYPE  ref  to  badi_name,  
   bd2  TYPE  ref  to  badi_name,  
   bd3  TYPE  ref  to  badi_name.  

GET  BADI  bd1  CONTEXT  same_ctx.  
GET  BADI  bd2  CONTEXT  same_ctx.  
GET  BADI  bd3  CONTEXT  dif_ctx.
Encapsulating Data for Two BAdIs Within One Class

Class A
*some code
GET BADI bd1 "ref to badi_1 CONTEXT me
CALL BADI bd1->met
*some code
GET BADI bd2 "ref to badi_2 CONTEXT me
CALL BADI bd2->met

Interface A
method M1

Interface B
method M2

Interface A
method M1

Interface B
method M2

Shared encapsulated data
Automatic migration by selecting utilities ➔ migration from BAdI Builder (SE18)

- Specify Enhancement Spot for BAdI Definition
- Specify Enhancement Implementation for BAdI Implementation

➔ no special knowledge necessary

➔ effort: 5 minutes per BAdI.
Automatic migration by selecting utilities → migration from BAdI Builder (SE18)
**Automatic (Partial) BAdI Migration**

Proxy Class

```
CL_EXITHANDLER=>
  GET_INSTANCE
CALL METHOD
  if_ex_badi->m
```

```
Proxy Class
CL_EX_<BAdI>
```

```
...[Classical Code is still used]
```

```
DATA mybadi type
  my_testbadi.
GETBADI mybadi
  FILTERS
  LANG = 'DE'
  CONTEXT me.
CALL BADI
  mybadi->meth.
...```

```
Implementation 1
Implementation 2
Implementation N
```

**Context**

**BAdI**
Complete migration:

- Delete the classic BAdI.
- Change Context Settings if you need no context.
- Find all calls of the classic BAdI by GET_INSTANCE and reprogram the BAdI call using the new commands ‘GET BADI’ and ‘CALL BADI’.
- For BAdIs that are called more than once in one program the context settings may be changed to improve the performance.

→ Expert knowledge of the application necessary

→ Effort: from some minutes up to some days per BAdI
A BAdI call is ca. 7.5% slower than a method call!

The more implementations defined, the higher is the improvement on performance
Normally BAdI definition and implementations are defined in different systems.

- The after import method of a BAdI definition writes ‘Call transaction SPAU’ in the transport log.

- After import or upgrade perform a manual migration of BAdI implementations by using transaction SPAU.
Dynamic BAdI (Next Release)

Syntax: bd is either a concrete BAdI-Handle or a generic BAdI-Handle (i.e. TYPE REF TO CL_BADI_BASE)

GET BADI { badi [FILTERS f1 = x1 f2 = x2 ...] }  
| { badi TYPE (name)  
  [ {FILTERS f1 = x1 f2 = x2 ...}  
    | {FILTER-TABLE ftab}] }  
[CONTEXT con].

CALL BADI { badi->meth parameter_list }  
| { badi->(meth_name) {parameter_list  
  |parameter_tables} }. 
Restricted Filter Values (Next Release)

BAdI is restricted to exactly one filter and the implementation only allows ‘=’ and ‘OR’ in the filter definition.

Use case

1. BAdIs used in Frameworks often have only one key to select BAdI implementations e.g.: ESI

   ➔ A special implementation for these BAdIs will allow to keep efficiency even when thousands of BAdI-Implementations for one BAdI exist in a system

2. You want to program a registry application. Some registered code is called depending on a registry key.

   ➔ Reuse the BAdI Framework for a fast registry implementation including upgrade support. A method for evaluation all implemented filter values is: CL_ENH_BADI_RUNTIME_FUNCTIONS=>GET_IMPLS_FOR_LIMITED_BADI(name)
Enhancement Framework Overview
Source Code Plugin – Technology
Function Group Enhancement – Technology
Class Enhancement – Technology
BAdI – Technology
Upgrade Adjustment
Switch Framework
Summary
Reasons for Upgrade Support I

Why is it necessary to adjust enhancements?
Reasons for Upgrade Support II

SAP

PROGRAM ...

*Original source code...

*Enhancement Section...

Code Changes

... ...

*Section Implementation

Code Adjustments ???

Customer

PROGRAM ...

*Original source code...

... ...

*Enhancement Section...

Code Changes

... ...

*Section Implementation

Code Adjustments ???
Objects to be Adjusted

ABAP Source code with enhancements
- The standard source code between ENHANCEMENT-SECTION and END-ENHANCEMENT-SECTION has changed.

Function modules with enhancements
- A new importing parameter was declared that has the same name as an importing parameter introduced by an enhancement.

Classes and interfaces with enhancements
- An enhanced method was deleted.

BAdIs
- BAdI interface was changed or a filter was deleted
How to Recognize Necessary Adjustments?

Import Log

1 conflict with enhancement objects occurred (ENHOBJ PR06 CD_TEST_ENH_UPGRADE2) - please call transaction SPAU_ENH

Editor

The enhancement implementation is in adjustment mode. Before the enhancement implementation can be edited, it must be adjusted.

SPAU_ENH

© SAP AG 2006, Enhancement Framework/84
1. Double-Click on the Enhancement Implementation you want to adjust

2. Switch to change mode

3. Go to tab „Adjustment“

4. Double-Click on list entry to adjust it
5. **Adjust Enhancement, e.g. in Splitscreen Editor**

6. **Set adjustment status to ✔️**

7. **Press Button ✔️Adjust Enhancement Implementation**

8. **Activate Enhancement 🍹**
Adjustment States

Semantic Changes
Automatic Adjustment
Tool-Aided Adjustment
Manual Adjustment
Enhancement is adjusted
Enhanced Object was deleted
Implementation is empty
Customer + SAP Status Quo: Industry Solutions

- Shipment delay compared to core 6-12 months
- Latest technology stack can not be utilized
- Missing release synchronization leads to upgrade delay and implementation hurdles
- Delayed provision of legal requirements
- Industries are implemented as modifications
- No reuse of functionality of other Industry Solutions
Goals and Benefits

- Reduce TCO for customer
- Timely provision of legal requirements
- Synchronization of release cycles and release planning
- Boost the attractiveness of ERP by providing already existing industry solutions to the entire SAP-Community
- Speed-up the introduction of SAP NetWeaver
### Synergies of Industry Solutions in mySAP ERP 2005

<table>
<thead>
<tr>
<th>Exclusiv Activation</th>
<th>Multiple Activation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP ECC Industry Extension Healthcare 6.0</td>
<td>SAP ECC Enterprise Extension Consumer Products 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Chemicals 6.0</td>
<td>SAP ECC Enterprise Extension Defense Forces &amp; Public Security 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Contract Accounting 6.0</td>
<td>SAP ECC Enterprise Extension Financials 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Discrete Industries &amp; Mill Products 6.0</td>
<td>SAP ECC Enterprise Extension FERC: Regulatory Reporting 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Insurance 6.0</td>
<td>SAP ECC Enterprise Extension Financial Services 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Media 6.0</td>
<td>SAP ECC Enterprise Extension Global Trade 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Mining 6.0</td>
<td>SAP ECC Enterprise Extension Human Capital Management 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Oil &amp; Gas 6.0</td>
<td>SAP ECC Enterprise Extension Incentive &amp; Commission Management 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Public Services 6.0</td>
<td>SAP ECC Enterprise Extension Industry-specific Sales Enhancements 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Retail 6.0</td>
<td>SAP ECC Enterprise Extension Joint Venture Accounting 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Telecommunications 6.0</td>
<td>SAP ECC Enterprise Extension PLM 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Utilities, Waste &amp; Recycling 6.0</td>
<td>SAP ECC Enterprise Extension Public Sector Management 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Catch Weight Management 6.0</td>
<td>SAP ECC Enterprise Extension Retail 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension SCM 6.0</td>
<td>SAP ECC Enterprise Extension SCM 6.0</td>
</tr>
<tr>
<td>SAP ECC Industry Extension Travel Management 6.0</td>
<td></td>
</tr>
</tbody>
</table>

**Single Activation**

**Multiple Usage**
Switches control the visibility of repository objects at runtime

Benefits:

- Industry Solutions are available with every release and SP without delay (i.e. timely provision of legal requirements), CRT’s are no longer necessary for add-on systems
- Industry Solutions can be enriched by generic functions from other industries
- Synchronization of release cycles and planning
Switchable Objects...

...by package assignment

- Appends, SI-, CI-includes for dictionary structures
- Fixed value appends to domains
- Secondary Indexes
- Append Search Helps
- Enhancement Implementations
- Switch Business Configuration Sets (Switch BC-Sets)
How to Determine if an Object is Switched?

**Dictionary: Display Extension Index**

<table>
<thead>
<tr>
<th>Index Name</th>
<th>MARA</th>
<th>M02 Switched Off</th>
<th>Switch</th>
<th>ISM_MEDIA_BASIS_MODIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short description</td>
<td>Übergeordnete Medienproduktfamilie</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last changed</td>
<td>WENZLER</td>
<td>04.02.2005</td>
<td>Original language</td>
<td>DE German</td>
</tr>
<tr>
<td>Status</td>
<td>New</td>
<td>Saved</td>
<td>Package</td>
<td>JCOREMOD</td>
</tr>
</tbody>
</table>

Index does not exist in database system DB6

**Dictionary: Display Append Structure**

<table>
<thead>
<tr>
<th>Append Structure</th>
<th>ISH_MARA_APP</th>
<th>New</th>
<th>Switched Off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Description</td>
<td>IS-H MM: Hospital-Specific Fields</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Attributes**

- Last changed on/by: GROTHUS 29.01.2005
- Package: NBAS Appl. development Hospital System
- Original language: DE German

**Switching**

- Appending table: MARA
- Switch: ISH_MAIN SAP Patient Management
Switchable Objects II

Switchable Objects ...

... by direct assignment

- Screen elements & Flow logic
- Menu entries & functions
- IMG nodes
- Customizing
### Screen Painter – Flow Logic

**Switch dependent module**

#### Screen Painter: Display Screen for SAPLCOKO1

<table>
<thead>
<tr>
<th>Screen number</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

##### Attributes

- MODULE HEADER_POSITION_FILL
- * Andern nach mitgegebener Struktur
  - MODULE SET_CHANGES_DARK.
- * Fertigungssteuerungsprofiltext füllen
  - MODULE GET_SFCPF_TEXT.
- * Linienhierarchie und -text füllen
  - MODULE GET_LINEHIER_TEXT.

#### Flow logic

- **TDP: Handling type Text lesen**
  - MODULE OIH_LOOKUP_OIHT_HEADER SWITCH OIH_EXCISE_DUTY.

---

**PROCESS AFTER INPUT**

- **Validation of Handling type**
  - FIELD CAUFWD-OIHANTYP
  - MODULE OIH_VALIDATE_HANTYP ON REQUEST. "SWITCH OIH_EXCISE_DUTY."

- **Abbrechen**
  - MODULE PROCESSING_EXIT AT EXIT-COMMAND.

- **Include: Previous Header**

| Li 7, Co 18 | Ln 7 - Ln 28 of 38 lines |
Switch dependent screen elements in the element list – disabled elements are not visible at runtime.
Menu Painter

Switch dependent menu entries

User Interface: ZPROGRAM  Active
Menu Bar:  ft
Application Toolbar:  ft

Items 1 - 7
- NEXP
- DL
- TP
- DET
- AFP
- AS

Items 8 - 14
- LB
- DIEN
- LIM

Items 15 - 21

Maintenance Views

The visibility of the fields of a table control is determined by the switch state.
The visibility of a view within a view cluster can be controlled by switches.
Customizing

Switch dependent import of table content via BC-Sets

- BC-Sets improvements:
  - Performance improvements
  - Support of deletion of customizing entries
  - Tool improvements like automatic recording, switch dependent definition and browsing

Switch dependent IMG nodes

- IMG nodes of disabled industries are not visible
### Business Function Set

- Pool of business functions
- Represents one industry solution
- Max. 1 can be active
- Use Transaction SFW5 to switch a on BFS.
- Transaction SFW3

### Business Function

- Represents a piece of business functionality
- Contains switches
- Transaction SFW2

### Switch

- Calculated states: ON, OFF, STANDBY
- Transaction SFW1
Activate a BFS

Log on to the Development System with authorization SAP_ALL

Disable all batch jobs

Close the system for all other users

Transaction SFW5

Choose the BFS in the drop down box

Select the required Business Functions that belong to the chosen BFS

Click on the button “activate”
Choosing the Business Function Set (BFS)

I. Select the BFS

![Image of the Change System Settings window with SFW5 highlighted]
Activating the Business Function Set

II. Choose the Business Functions

III. Press activate
Activating a BFS II

- The Batch Job for DDIC and BC-Set activation starts immediately.
- This job takes about 30-120 min
- The background activities are comparable to installing an Add On manually
- Reschedule batch jobs in the development system and unlock users

Caution:
Only one Business Function Set can be activated

Caution:
There is no possibility, to deactivate a BFS
Switch Framework Outlook Next NetWeaver Release

Enhancements for the objects of the Switch Framework

- **Switches**
  Packages can be added by enhancements. Beneficial for distributed development of switchable solutions. This enhancement is not switchable itself. When switching on a switch, all enhancements of this switch will be considered.

- **Business Functions**
  Switches can be added by enhancements.

- **Business Function Sets**
  Business Functions can be added by enhancements. Thus it is possible for ISVs to add their functionality to SAP Business Function Sets without modifying the original object.

- **Switching off Business Functions**
  For test purposes only
Summary

- The Enhancement Framework offers new possibilities to extend the SAP Standard instead of modifying it.
  - Source Code PlugIns
  - Function Group Enhancements
  - Class Enhancements
  - New BAdIs

- The new BAdIs are more flexible and faster than the classic ones.

- The Enhancements offered by Enhancement Framework and several other object types can be switched by the Switch Framework.
Further Information

Public Web


http://sdn.sap.com ➔ Weblogs by author Thomas Weiss
THANK YOU FOR YOUR ATTENTION!