UI Enhancements in SNC and SPP
- Only for Screens Using UI Framework -

1 UI-TECHNOLOGY ........................................................................................................ 2

2 BADI DEFINITION ....................................................................................................... 3
  2.1 Functionality ............................................................................................................ 3
  2.2 Standard Settings of the BAdI /SCF/UIMDL_APPCUST ........................................... 3

3 BUSINESS LOGIC ....................................................................................................... 4
  3.1 Determine Model Class ............................................................................................. 4
  3.2 Determine Component Class ...................................................................................... 7

4 SAP ENHANCEMENT FRAMEWORK ....................................................................... 9

5 BADI IMPLEMENTATION .......................................................................................... 10
  5.1 Table ........................................................................................................................ 10
  5.2 Selection Box ......................................................................................................... 12
  5.3 Form Box ............................................................................................................... 13

Version: 1.1 Date 23.11.2009
The SNC Extension Guide is a collection of tips and tricks on how to extend SAP Supply Network Collaboration to meet your specific business requirements. It is based on SNC 7.0. Consultants, Developers and Solution Managers have worked together to build this content based on customer experiences. We invite you to actively participate in the content - share your experiences, comments and questions.

Disclaimer: SAP is not liable for any discrepancies in the content or code in the documents.

1 UI-Technology

This document describes how to enhance user interfaces (UIs) of SAP Supply Network Collaboration (SAP SNC) and Service Parts Planning (SPP) that are based on the UI Framework (UIF). The following technologies are used:

- Releases lower than 5.1:
  - SAP SNC:
    - UIs based on the UI Framework and rendered using BSP
  - SPP:
    - UIs based on the UI Framework and rendered using BSP
    - UIs based on the UI Framework and rendered using SAPGUI

- Release 5.1 and higher:
  - SAP SNC:
    - UIs based on the UI Framework and rendered using Web Dynpro ABAP
    - Web Dynpro ABAP UIs that do not use the UI Framework
  - SPP:
    - UIs based on the UI Framework and rendered using Web Dynpro ABAP
    - UIs based on the UI Framework and rendered using SAPGUI
    - Web Dynpro ABAP UIs that do not use the UI Framework

To find out whether a screen is based on the UIF and can thus be enhanced by following this document, you can use transaction /SCF/SCREENCFG:

You can follow this document to enhance only those screens for which the Web-Dynpro Screen checkbox is not selected.
screens for which the Web-Dynpro Screen checkbox is selected have to be enhanced using the standard SAP Enhancement Framework.

The UI Framework is a tool to configure the layout and the event handling of screens. Based on this configuration, the UI is rendered dynamically at runtime.
2 BAdI Definition

SAP SNC and SPP use a framework to configure the layout and the event handling of the screens. The configuration within this UI framework is done using multiple views/tables. If an SAP SNC screen needs to be enhanced at the customer site, the BAdI /SCF/UIMDL_APPCUST (and enhancement spot with the same name) is provided for this purpose.

2.1 Functionality

You can use this BAdI to modify and enhance screens. A screen is made up of different components that you can define in any sequence as desired. You can change the delivered layouts for the following components:

- Advanced toolbar (GET_ADVANCEDTOOLBAR_DATA method)
- Text area (GET_TXTAREA_DATA method)
- Toolbar (GET_TOOLBAR_DATA method)
- Table layout (GET_TABLEVIEW_DATA method)
- Selection box (GET_SELBOX_DATA method)
- Form box (GET_FRM_DATA method)
- Screen-layout (GET_SCRLYT method)

You can change the following display properties:

- Arrangement of the above components on the user interface screen (GET_SCRLYT method)
- Properties of pushbuttons (GET_BTN_PROP method)
- Events (GET_EVENTS method)
- Changes on screens that use the SNC patterns Query Pattern (QP) and Object Detail Pattern (ODP):
  - Aspect type assignment (GET_FROBJDEF method)
  - Aspect type assignment to individual fields (GET_PATTERN_BINDING method)
  - Aspect type assignment to object types (GET_PATTERN_OBJASP method)

2.2 Standard Settings of the BAdI /SCF/UIMDL_APPCUST

The BAdI /SCF/UIMDL_APPCUST is designed for multiple use. Moreover, the BAdI is filter-dependent. The /SCF/BADIAPPABS transparent table is used to store the different filter values. For SAP SNC, the filter value ‘ICH’ is used, whereas the SPP filter values always start with ‘SPP’.

In the standard system, there is no activated BAdI implementation.
3 Business Logic

Most of the enhancements require modifications to and/or enhancements of existing business logic provided by SAP. The following chapters describe how to find the classes providing this source code. The UI Framework calls two different types of class directly, as follows:

- **Model Class**
  The model class contains the source code that is required to process events such as read / change / insert / delete business objects. This class is not responsible for the data storage and retrieval itself, but calls the appropriate interfaces.
  
  **Example:** The model class for the ASN Detail Screen contains source code to call the Business Object Layer (BOL) for the different events triggered from the UI. The BOL then performs the required action (read, change, insert, delete, and so on) using Order Document Management (ODM) as the data source. A further data source could be Time Series Data Management (TSDM) for applications displaying time series (for example, Forecast and Promotion Screens).

- **Specialized component class**
  A screen is made up of different components in a defined sequence. The UI Framework provides common classes to support generic tasks of the different components. If special tasks have to be supported by a component, this logic needs to be developed in a specialized component class.

3.1 Determine Model Class

Most of the enhancements require modifications to and/or enhancements of existing business logic provided by SAP. To find the source code that needs to be changed, proceed as follows:

1. **Determine technical name of screen**
   **Alternative I:**
   SCRID is the technical name of the screen. You can use the /SCF/SCREENCFG transaction to determine SCRID.

   In transaction /SCF/SCREENCFG, choose the application (APPID):

   ![Display View "Application definition": Overview](image)

   On the next level of this view cluster, you can see all screens for the chosen application:
For each screen, you can see the technical name of the screen (SCRID / screen ID) and application (APPID / application ID).

Example: The SAP SNC alert monitor has the screen ID ALRTOVW and the application ID ICH.

Alternative II:
Use transaction SE16 with table /SCF/UNAV to determine the screen ID. Table /SCF/UNAV contains the entries of the navigation menu for the different views (supplier view, customer view, and so on).

- Select data

For SPP enhancements, use table /SCF/NAV, except for application ID SPP, because no specific views exist for customer, supplier, and so on.
• Find Screen ID
Using the selection parameters from the last step will lead to the following result:

Example: The SNC alert monitor has the screen ID ALRTOVW.

2. Determine the application data ID
You use transaction /SCF/APPDATAASS to determine the application data ID, as follows:

Another key to determine the application data ID is the GUI Component ID. In almost all cases, the same application data ID is used for all GUI component IDs. If this is not the case, then you can use transaction /SCF/SCREENCFG to find the correct GUI component ID.
Example: Application ICH and screen ALRTOVW are assigned to application data ID SMIALRT.

3. Determine the model class
You use transaction /SCF/APPDATADef to show the model class for the application ID and application data ID, as follows:

Example: Application ICH and application data ID SMIALR are assigned to model class /SCF/CL_DATA_SMIALRT.
3.2 Determine Component Class

A screen is made up of different components in a defined sequence.

Example: ASN Overview (Supplier View)

The UI Framework provides the following common classes to support generic tasks of the different components:

- Chart: /SCF/CL_CMPTCHART
- Form Box: /SCF/CL_CMPTFRMBOX
- Selection Box: /SCF/CL_CMPTSELBOX
- Tabstrip: /SCF/CL_CMPTTABSTRIP
- Tab: /SCF/CL_CMPTTAB
- Table: /SCF/CL_CMPTTBV_ABS
- Toolbar: /SCF/CL_CMPTTLBR_ABS
- Tray: /SCF/CL_CMPTTRY_ABS
- Text Area: /SCF/CL_CMPTTXTAREA_ABS
- Advanced Toolbar: /SCF/CL_CMPTXTLBR_ABS

If special tasks have to be supported by a component, this logic needs to be developed in a specialized component class. If the screen uses a UI pattern, such as QP (Query Pattern) or ODP (Object Detail Pattern), then specialized classes with additional functions besides the common functions, are used; for example, Query Pattern uses the following specialized classes:

- Advanced Toolbar for QP Selection Box: /SCF/CL_QP_ENHTOOLBAR_SEL
- QP Selection Box: /SCF/CL_QP_SELBOX_SEL
- Advanced Toolbar for QP Selection Table: /SCF/CL_QP_ENHTOOLBAR_SELID
- QP Advanced Selection Table: /SCF/CL_QP_TABLEVIEW_SEL
- Advanced toolbar for QP Result Table: /SCF/CL_QP_ENHTOOLBAR_RL
- QP Result Table: /SCF/CL_TBV_QPRETURNLIST
- Toolbar for QP-Result Form Box: /SCF/CL_CMPTTLBR_QP
- QP Result Form Box: /SCF/CL_QP_FORMBOX_RL

If further specialization is required, screen-dependent specialized classes are required. These classes can be found with transaction /SCF/SCREENCFG, as follows:

1. Choose the application
2. Choose the screen

Display View "Screen definition": Overview

The screen layout shows the different screen components and their specialized classes. If a source code enhancement with regards to one of these components is required, then the specialized class assigned to the component needs to be enhanced.

Example:
If the ASN Overview Toolbar component on the ASN Overview (Supplier View) screen has to be enhanced, then the source code of the /SCF/CL_MDLTBV_ASNLIST class needs to be enhanced.
4 SAP Enhancement Framework

SAP source code can be enhanced by modifications or by using the SAP Enhancement Framework. The SAP Enhancement Framework integrates several enhancement technologies in one framework that enables you to adapt ABAP-based programs and Web Dynpro ABAP user interfaces without changing or modifying the underlying development units. Using the Switch Framework, you can control the visibility of repository objects, including all enhancements of the new framework.

Implicit enhancement options are automatically available - they do not have to be defined by the developers of the original code. They are built into the language and are available without further declarations. The following enhancements can be made using implicit enhancements options:

- New data fields can be added to existing structures in the definition of data, types, or constants.
- Arbitrary code can be injected at specific locations, such as start or end of a procedure (form subroutine, function module, or method).
- Function group enhancements allow you to add parameters to the interface of function modules.
- ABAP objects classes or interfaces can be enhanced by adding new fields or methods. Methods can also be enhanced with parameters and with pre- or post methods that are executed before or after the original method. It is also possible to enhance an existing class by replacing an existing method implementation with a new one (overwrite).

For more information about the SAP Enhancement Framework, see the SAP documentation.
5 BAdI Implementation

For more information about the implementation of BAdIs in the context of the enhancement concept, see SAP Library for SAP NetWeaver under BAdIs - Embedding in the Enhancement Framework.

The BAdI /SCF/UIMDL_APPCUST uses the /SCF/IF_EX_UIMDL_APPCUST interface. For more information, display the interface in the Class Builder (transaction SE24). The following chapters describe how to change the layout of the main components of the UI Framework screens.

5.1 Table

If the table component needs to be changed, the GET_TABLEVIEW_DATA BAdI method needs to be implemented. The following example explains how to add three columns to a table.

You want to display the following additional columns on the Release Forecast screen in the service parts planning (SPP) application:

- Scaling Factor
- Do Not Use Disaggregated Forecast
- Forecast Model Change Date

In this case, proceed as follows:

1. Enhance structure / table in back end

You have added the /SAPAPO/SPP_FCQ_DATA_STR include of the /SAPAPO/SPP_FCQ database table with the fields ZZSCAL_FAC, ZZNOT_USE_DISAG and MODEL_CHG_DATE to the append structure.

Additional remark:
When you enhance this database table, the table content is read automatically by the application and passed to the UI Framework. These fields need to be filled with appropriate values with customer source code.

If customer-specific data needs to be read from other (more complex) sources, then the logic of the model class has to be enhanced as well. If so, you need to know the name of the model class.

2. Set filter

For performance reasons, you have set filters in the BAdI implementation. (If you do not set filters, the system executes the BAdI when the other user interfaces for service parts planning are called.) Specify the user interface on which you want to display the additional columns as the filter. In other words, to display additional columns on the Release Forecast screen, enter SPPFCSTAPPR as the filter. Since you can also call the Release Forecast screen from the Interactive Forecast screen, enter SPPFCST as the filter as well (otherwise, the system will not display the additional columns on the Release Forecast screen if you call this screen from the Interactive Forecast screen).

3. Define BAdI implementation source code

You define the GET_TABLEVIEW_DATA BAdI method as follows:

a) You specify that the system is only to display the additional columns in the Release Forecast screen. You can code this instruction as follows:

```plaintext
METHOD /scf/if_ex_uimdl_appcust~get_tableview_data.
DATA: 
  ls_tbvcol = LIKE LINE OF it_tbvcol, 
  ls_tbvcol_yt = LIKE LINE OF it_tbvcol_yt, 
  ls_field_prm_map = LIKE LINE OF it_field_prm_map.
IF flt_val = 'SPPFCSTAPPR' OR flt_val = 'SPPFCST'.
  IF is_cmptinfo_scrid = 'MAINFCSTAPP' AND is_cmptinfo_cmptid = 'RETLISTBR1'.
```

b) You define the additional columns that the screen is to display. Optionally, you can specify, for each column, a data element that defines how the system is to display the data.
In this way, you can define that the date is displayed as an 8-character value separated by periods, for example.

**Note:**
You cannot use floating data elements.

You can code the additional column *Scaling Factor* as follows:

```
ls_tbvcol-colid = 'SCAL_FAC'.
ls_tbvcol-defcoltext = 'Scaling Factor'.
APPEND ls_tbvcol TO it_tbvcol.
ls_tbvcollyt-colid = 'SCAL_FAC'.
INSERT ls_tbvcollyt INTO TABLE it_tbvcollyt.
```

You can code the additional column *Do Not Use Disaggregated Forecast* as follows:

```
ls_field_prm_map-fieldid = 'NOT_USE_DISAG'.
ls_field_prm_map-dataelement = '/SAPAPO/NOT_USE_DISAG'.
INSERT ls_field_prm_map INTO TABLE it_field_prm_map.
ls_tbvcol-colid = 'NOT_USE_DISAG'.
ls_tbvcol-defcoltext = 'Use Statistical Forecast'.
APPEND ls_tbvcol TO it_tbvcol.
ls_tbvcollyt-colid = 'NOT_USE_DISAG'.
ls_tbvcollyt-colindex = 72.
INSERT ls_tbvcollyt INTO TABLE it_tbvcollyt.
```

You can code the additional column *Forecast Model Change Date* as follows:

```
ls_field_prm_map-fieldid = 'MODEL_CHANGE_DATE'.
ls_field_prm_map-dataelement = '/SAPAPO/MODEL_CHANGE_DATE'.
INSERT ls_field_prm_map INTO TABLE it_field_prm_map.
ls_tbvcol-colid = 'MODEL_CHG_DATE'.
ls_tbvcol-defcoltext = 'Forecast Model: Date Changed'.
APPEND ls_tbvcol TO it_tbvcol.
ls_tbvcollyt-colid = 'MODEL_CHG_DATE'.
ls_tbvcollyt-colindex = 73.
INSERT ls_tbvcollyt INTO TABLE it_tbvcollyt.
```

4. Assign aspect type to individual fields

If QP (Query Pattern) or ODP (Object Detail Pattern) are used, the pattern binding has to be set as well. In the GET_PATTERN_BINDING method, you define the assignment for the field that contains the data for the additional column, as well as the ID of the column in which the system is to display this data. You can code this assignment as follows:

```haskell
METHOD /scf/if_ex_uimdl_appcust~get_pattern_binding.
DATA: ls_pattern_namebndg LIKE LINE OF ct_pattern_namebndg.
IF flt_val = 'SPPFCSTAPPR' OR flt_val = 'SPPFCST'.
  IF is_cmptinfo-scrid = 'MAINFCSTAPPR' AND is_cmptinfo-cmptid = 'RETLISTTBR1'.
    ls_pattern_namebndg-appid = 'SPPFCSTAPPR'.
    ls_pattern_namebndg-context = 'SPPFCSTAPPR'.
    ls_pattern_namebndg-object = 'QP'.
    ls_pattern_namebndg-aspect_type = 'DISPLAY'.
    ls_pattern_namebndg-fieldname = 'ZZSCAL_FAC'.
    ls_pattern_namebndg-fieldid = 'SCAL_FAC'.
    INSERT ls_pattern_namebndg INTO TABLE ct_pattern_namebndg.
  ENDIF.
ENDIF.
ENDMETHOD.
```
5.2 Selection Box

If the selection box component needs to be changed, the GET_SELBOX_DATA BADI method needs to be implemented. If a screen needs to be enhanced by one additional selection parameter, the following steps are required.

In this example, the SPP Cockpit screen will be enhanced by the new ZCLIENT selection field of type MANDT.

1. Register new parameter
   A new entry in table /SCMB/PARAM has to be maintained (with transaction SM30) as follows:
   - Parameter: ZCLIENT
   - Parameter Type: Characteristic
   - Data Type: MANDT

2. Define BADI implementation source code
   You define the GET_SELBOX_DATA BADI method as follows:

   ```
   METHOD /scf/if_ex_uimdl_appcust~get_selbox_data.
   DATA: ls_field_prm TYPE /scf/field_prm_str,
        ls_selboxprm TYPE /scf/selboxprm_str,
        ls_selboxlyt TYPE /scf/selboxlyt_str.
   *
   * Data Element Assignment
   ls_field_prm-fieldid     = 'ZCLIENT'.
   ls_field_prm-paramid     = 'ZCLIENT'.
   ls_field_prm-dataelement = 'MANDT'.
   INSERT ls_field_prm INTO TABLE it_field_prm_map.
   *
   * Selbox Definition
   ls_selboxprm-paramid = 'ZCLIENT'.
   INSERT ls_selboxprm INTO TABLE it_params.
   *
   * Selbox Layout
   ls_selboxlyt-paramid     = 'ZCLIENT'.
   ls_selboxlyt-selboxrow   = '25'.
   ls_selboxlyt-selboxcol   = '1'.
   ls_selboxlyt-selboxentry = '1'.
   ls_selboxlyt-valuetype   = '1'.
   INSERT ls_selboxlyt INTO TABLE it_layout.
   ENDMETHOD.
   ```

   The label of this new selection field will be taken from the MANDT data element (transaction SE11, Field Label tab).

3. Business Logic
   To use this new selection parameter, the model class has to be enhanced by customer-specific source code.
   If the screen uses Query Pattern for selection of relevant data (this pattern is used in almost all cases), then the QUERY method will handle the selection of data with regards to the selection parameter values. This means that customer-specific selection logic, based on the new selection parameter, has to be added to this method (or to a sub-method called by method QUERY). This can be achieved by one of the following:
   - Modification of SAP source code
   - SAP Enhancement Framework
5.3 Form Box

If the form box component needs to be changed, the GET_FRM_DATA BAdI method needs to be implemented.

The following example explains how to use the BAdI if the new Event Status field has to be added to the ASN Details (Supplier View) screen.

Without BAdI implementation:

With active BAdI implementation:
1. Define BAdI implementation source code

You define the GET_FRM_DATA BAdI method as follows:

```abap
METHOD /scf/if_ex_uimdl_appcust~get_frm_data.

DATA: ls_field_prm TYPE /scf/field_prm_str,
ls_frmfields TYPE /scf/fmfields_str,
ls_frmlyt TYPE /scf/frmlyt_str.

* Implementation only valid for SNC-Supplier-View
CHECK flt_val         = 'ICH'.
CHECK appview-appview = 'ICHSUPPLYS'.
* Implementation only valid for ASN-Detail screen
CHECK is_cmptinfo-scrid = 'ASNMAINTAIN'.
* Implementation only valid for formbox on ASN-Detail-Tab 'General
CHECK is_cmptinfo-cmptid = 'ASNDFR01'.

* Data Element Assignment
ls_field_prm-fieldid     = 'ZSTATUS'.
ls_field_prm-paramid     = 'ZSTATUS'.
ls_field_prm-dataelement = '/SCA/BIF_DMDINF_EVSTATUS_CODE'.
INSERT ls_field_prm INTO TABLE it_field_prm_map.

* Definition Field
ls_frmfields-fieldid    = 'ZSTATUS'.
ls_frmfields-FRMFLDTXT  = 'Event Status'.
ls_frmfields-FRMFLDTTIP = 'Event Status'.
INSERT ls_frmfields INTO TABLE it_frmfields.

* Layout Field
ls_frmlyt-fieldid   = 'ZSTATUS'.
ls_frmlyt-frmrow    = '7'.
ls_frmlyt-frmcol    = '4'.
ls_frmlyt-valuetype = 'TEXT'.
INSERT ls_frmlyt INTO TABLE it_frmlayout.

ENDMETHOD.
```

The label of this new field will be taken from the /SCA/BIF_DMDINF_EVSTATUS_CODE data element (transaction SE11, Field Label tab).

The CHECK statements at the beginning of the method are used to validate whether the correct component calls the BAdI implementation. The application ID, screen ID, component ID, and specialized component class can be found using transaction /SCF/SCREENCFG, as follows:
2. Business logic

To fill the new field with content, the model class has to be enhanced by customer-specific source code.

In this example, the specialized class for the form box on the General tab is /SCF/CL_FRMBOX_ASNM_HDR1. The LOAD_HDR_DATA method is responsible for reading all header data and passing the different header values to the UI Framework. The method needs to be enhanced using the SAP Enhancement Framework. You can use a PRE method enhancement to the LOAD_HDR_DATA method to set the customer specific value for the new ZSTATUS field, as follows:

```plaintext
ls_formbox-fieldid = 'ZSTATUS'.
ls_formbox-cell_value = 'GREEN'.  "customer specific logic
APPEND ls_formbox TO lt_formbox.
```