# Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Emphasized words or expressions.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text. For example, SELECT and INCLUDE.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>&lt;Example&gt;</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>December 2019</td>
<td>First version of Extensibility Guide created.</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Goals .......................................................................................................................... 5  
1.1 Scope of Extensions ................................................................................................. 5  
1.2 What You Need to Know ......................................................................................... 5  
1.3 Requirements ......................................................................................................... 5  

2 Extensibility Scenarios .............................................................................................. 7  
2.1 Extending Selection Parameters for Initial Load .................................................... 7  
2.1.1 Purpose .............................................................................................................. 7  
2.1.2 Prerequisites ...................................................................................................... 7  
2.1.3 Process Steps ..................................................................................................... 8  
2.2 Extending Data Filters for Synchronization Objects ............................................... 13  
2.2.1 Purpose ............................................................................................................ 13  
2.2.2 Prerequisites .................................................................................................... 13  
2.2.3 Process Steps ................................................................................................... 14  

3 Glossary for Extensibility ......................................................................................... 18
1 Goals

This document provides information and code examples for developers who create applications for the integration of Asset Central Foundation or who extend the functionality. Each scenario is illustrated with a code example. This can help the consultant to relate to a specific customer situation.

Note

The concept of integration extensions explained in this document refers to SAP ERP on-premise releases only - SAP ECC and SAP S/4 HANA.

1.1 Scope of Extensions

Asset central foundation integration offers flexible configuration options to implement the standard features. However, to address specific customer requirements, you might want to extend the standard integration without disrupting the standard behavior. This document will provide you with guidance on how to extend the integration layer.

1.2 What You Need to Know

The integration layer comprises several technologies which includes ABAP OO, WDA, Java, Apache Kafka, Open APIs (REST, ODATA) etc., spanning applications and services in SAP ERP and SAP Cloud Platform. However, you do not need to know all technologies listed above. Moreover, due to limited options for extensions in the Cloud, extensions are currently not supported in a non-ABAP world.

For integration-related extensions, you need to be familiar with ABAP OO and standard extension options provided in SAP ERP. Additionally, familiarity with the terminology and configuration panel of SAP Mobile Add-on would be helpful.

1.3 Requirements

<table>
<thead>
<tr>
<th>Title</th>
<th>User Story</th>
<th>Importance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extend Selection Parameters for</td>
<td>As a consultant, I want to configure a new selection parameter in initial load, so that users can be more specific with the criteria to limit transfer of data from SAP ERP to asset central foundation.</td>
<td>Must Have</td>
<td>Version 1.0</td>
</tr>
<tr>
<td>Initial Load</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extend Data Filters for</td>
<td>As a consultant, I want to extend filters for synchronization, so that users can specify</td>
<td>Must Have</td>
<td>Version 1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>User Story</td>
<td>Importance</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>Synchronization Objects</td>
<td>additional rules to control data synchronization between SAP ERP and Asset Central Foundation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2 Extensibility Scenarios

In this section, extension scenarios will be discussed in detail using screenshots and sample code, where necessary, to aid understanding.

⚠️Caution
The sample code for extensions shown below is built on the latest code line, which might result in compilation errors if you directly copy and paste the source code to your system. SAP recommends that you use the sample code for reference only.

2.1 Extending Selection Parameters for Initial Load

You use this to enhance the initial load process with additional selection parameters.

2.1.1 Purpose

We have taken the example of adding “Equipment Type” as a selection parameter to the equipment initial load process, so that customer can upload equipment of a specific type.

2.1.2 Prerequisites

Before you begin with the procedural aspects of an extension, you should know that each initial load process is represented as a data agent, which includes the definition of the selection parameters, data selection and secure data transfer.
2.1.3 Process Steps

1. Data Agents are delivered with the business configuration. These data agents can be found on the Mobile Application Configuration page → Data Staging Settings → Data Agent Definition (R/3 Transaction /SYCLO/CONFIGPANEL).

2. The technical implementation for a data agent can be found in the handler class on the General Data tab.
3. Use the class editor (**SE24**) to display handler implementation, select the _Filter_ checkbox to restrict the display to redefined methods for asset central integration.

4. For the addition of selection parameters, the handler class should be enhanced for data declarations and related source code using the selection parameters.

```plaintext
< CLASS /aci/cl_pm_initload_equipment DEFINITION PUBLIC
   INHERITING FROM /aci/cl_init_load_agent_base
   FINAL
   CREATE PUBLIC.
   PUBLIC SECTION.
   TYPES: tt_equnr_range_tab TYPE STANDARD TABLE OF /syclo/pm_equnr_range_str WITH DEFAULT KEY.
   TYPES:
      BEGIN OF gtyp_runtime_params_str,
      equi_category TYPE REF TO /syclo/core_range_tab,
      maintenance_plant TYPE REF TO /syclo/core_range_tab,
      equipment_no TYPE REF TO tt_equnr_range_tab,
      equipment_num TYPE REF TO /syclo/core_range_tab,
      sync_characteristic_value TYPE REF TO /syclo/core_boolean_dte,
      skip_hierarchy_sync TYPE REF TO /syclo/core_boolean_dte,
      ENHANCEMENT 1 ZV_ACI_PARAM_EXT. "active version
      Declare new parameter for equipment type
      TYPES: equi_type TYPE REF TO BAPI_ITOB_T_SEL_OBJECTTYPE.
      ENDENHANCEMENT.
   END OF GTYP_RUNTIME_PARAMS_STR.>
```
• **Enhance Handler Class for Data Declarations**

Data declarations for the handler class can be found in the public section of the class. Use the menu path More → Go to → Sections → Public section. You can use implicit enhancement options to introduce additional parameters as shown below:

- **Method `GET_OBJECT_RUNTIME_PARAM_LIST`**

This method is used to return all registered parameters. The input provided in this method will be used for the initial load user interface. Create an implicit enhancement to register the new parameter. It is recommended to create the enhancement at the end of the method implementation, so that new parameter can be appended to the parameter list.

```<METHOD get_object_runtime_param_list.
...
***Standard method Implementation ends here. ***

ENHANCEMENT 2  ZV_ACI_PARAM_EXT.  "active version

* Code enhancement to add a new selection parameter for initial load
  data lt_runtime_parameter TYPE /SMFND/CORE_GEN_PARAMINFO_TAB.
  lt_runtime_parameter = VALUE #( 
    ( param_name = 'EQUI_TYPE' usage_tabname = 'BAPI_ITOB_SEL_OBJECTTYPE' 
      flag_multi_val = abap_true flag_tech_label = abap_true ) ) . " New Parameter
  APPEND LINES OF lt_runtime_parameter TO et_runtime_parameter.
ENDENHANCEMENT.
ENDMETHOD.>
```

- **Method `INITIALIZE_OBJECT_PARAMS`**

This method is used to initialize all selection parameters. It is recommended to create the enhancement at the beginning of the method implementation, so that new parameter can be automatically processed with the standard implementation.

```<METHOD initialize_object_params.

ENHANCEMENT 3  ZV_ACI_PARAM_EXT.  "active version

* Create data object
  CREATE DATA me->str_runtime_param_vals->equi_type.
ENDENHANCEMENT.
```
***Standard method Implementation follows...***

- **Method SELECT_DATA_IN_RELOAD_MODE**
  This method is used to fetch data based on the user selection criteria for initial and reload. It is recommended to create an enhancement at the beginning of the method implementation. Then you need to copy standard implementation and enhance the SELECT query with the selection parameter. In the sample below the SELECT query is enhanced with the parameter "equi_type". Place a RETURN statement at the end of the enhancement to skip the execution of the standard implementation.

```
<METHOD select_data_in_reload_mode.

ENHANCEMENT 4 ZV_ACI_PARAM_EXT. "active version
  DATA: ls_trans_req_info1 LIKE LINE OF me->tab_trans_req_info,
    lv_package_size1 TYPE i VALUE '5',
    lv_cursor_key1 TYPE cursor,
    lv_fetch1.

  *---Main Section

  CHECK me->str_runtime_param_vals-equipment_no->* IS NOT INITIAL
  OR me->str_runtime_param_vals-equi_category->* IS NOT INITIAL
  OR me->str_runtime_param_vals-maintenance_plant->* IS NOT INITIAL.

  ls_trans_req_info1-object_type = me->object_type.
  ls_trans_req_info1-skip_hierarchy_sync = me->str_runtime_param_vals-
    skip_hierarchy_sync->*.
  ls_trans_req_info1-determine_subseq_dependency = me->str_runtime_param_vals-
    sync_characteristic_value->*.

  OPEN CURSOR WITH HOLD lv_cursor_key1 FOR

  SELECT a~equnr AS object_key FROM equi AS a
    INNER JOIN equz AS b ON a~equnr = b~equnr
    INNER JOIN iloa AS c ON b~iloan = c~iloan
    WHERE a~equnr IN me->str_runtime_param_vals-equipment_no->*
      AND a~eqtyp IN me->str_runtime_param_vals-equi_category->*
      AND c~swerk IN me->str_runtime_param_vals-maintenance_plant->*
      AND a~eqart IN me->str_runtime_param_vals-equi_type->*.

  lv_fetch1 = abap_true.

  WHILE lv_fetch1 = abap_true.
```
FETCH NEXT CURSOR lv_cursor_key1 INTO CORRESPONDING FIELDS OF TABLE me->tab_trans_req_info PACKAGE SIZE lv_package_size1.

IF sy-subrc = 0.
  ev_flag_callback = abap_true.
ELSE.
  CLOSE CURSOR lv_cursor_key1.
  CLEAR: ev_flag_callback, lv_cursor_key1.
ENDIF.
ENDWHILE.

IF me->tab_trans_req_info IS NOT INITIAL.
  MODIFY me->tab_trans_req_info FROM ls_trans_req_info1 TRANSPORTING
    object_type skip_hierarchy_sync
    WHERE object_key NE space.

  SORT me->tab_trans_req_info BY object_type object_key.
  DELETE ADJACENT DUPLICATES FROM me->tab_trans_req_info COMPARING object_type object_key.
  DESCRIBE TABLE me->tab_trans_req_info LINES ev_record_count.
ENDIF.

**Standard method Implementation follows..***

***
5. Now you are just one step away from testing your extension, but before that make sure all your enhancement implementations are activated.

2.2 Extending Data Filters for Synchronization Objects

You use this to enhance the synchronization process with additional data filters.

2.2.1 Purpose

It is important to know that synchronization and initial load are separate processes. The former processes are triggered in real-time during asset management transactions and the latter is used for the bulk transfer of asset data from ERP to asset central foundation. So, if you have a requirement to extend the integration for any asset data, please check whether the extension is valid for initial load and/or synchronization processes.

For the extension of filters in the synchronization object, we will take the example of adding a filter “Manufacturer” in the equipment synchronization process, so that customers can transfer equipment data from specific manufacturer(s).

2.2.2 Prerequisites

Before you begin with the procedural aspects of an extension, you should know that each synchronization process is represented by a synchronization object, which contains Data filters, business configurations and functions to securely transfer data.
2.2.3 Process Steps

1. Synchronization objects are delivered with standard business configuration which can be found under **Mobile Application Configuration** page → **RFC Channel Integration Settings** → **Mobile Data Object Configuration** (R/3 Transaction /SYCLO/CONFIGPANEL).

2. The technical implementation for a synchronization object can be found in the handler class on the "**General Setting**" tab.

3. Use the class editor (**SE24**) to display handler implementation, select **Filter** checkbox to restrict the display to redefined methods for asset central integration.
4. For the addition of synchronization filters, the handler class should be enhanced for data declarations and related source code using the filters.

- **Enhance Handler Class for Data Declarations**

  Data declarations for the handler class can be found in the public section of the class. You can use the menu path (More → Go to → Sections → Public section). You can use the implicit enhancement options to introduce additional parameters as shown below. Type declaration "gtyp_dof_filter_vals_str" is used for synchronization filters. This structure should be enhanced by adding the new filter for "Manufacturer".

  `<class /ACI/CL_PM_EQUIPMENT_DO definition
  public
  inheriting from /ACI/CL_CORE_DO_HANDLER
  create public .
  public section.

  types:

  BEGIN OF gtyp_dof_filter_vals_str,
   ain_rfc_destination       TYPE REF TO /syclo/core_range_tab,
   *
   ...
   user_status               TYPE REF TO /syclo/core_range_tab,
  ENHANCEMENT 1  ZAC_ENH_EQUIPMENT_SYNC_FILTER.    "active version
  TYPES : manufacturer       TYPE REF TO /syclo/core_range_tab.
  ENDENHANCEMENT .
  END OF gtyp_dof_filter_vals_str .
  
  ***Standard Implementation follows..***`

- **Enhance Source Code in the Handler Class**

  In this step you need to enhance few methods in the handler class to influence the initial load process with newly added parameter.

  o Method /SYCLO/IF_CORE_FILTER_SERV~GET_DATA_FILTER_LIST

    This method is used to register allowed synchronization filters, input provided in this method will be used to render UI elements in the config panel. Using the implicit enhancement options, you must register the new filter as below.

    `<METHOD /syclo/if_core_filter_serv~get_data_filter_list.
    ...
    ***Standard method Implementation ends here..***`
ENHANCEMENT 2  ZAC_ENH_EQUIPMENT_SYNC_FILTER  "active version
*--------------------------------------------------------------------*
* Manufacturer added
APPEND INITIAL LINE TO et_data_filters ASSIGNING <ls_core_filter_serv_str>.  
<ls_core_filter_serv_str>-dof_name = 'MANUFACTURER'.
<ls_core_filter_serv_str>-usage_tabname = 'EQUI'.
<ls_core_filter_serv_str>-usage_fieldname = 'HERST'.
<ls_core_filter_serv_str>-do_handler   = me->classname.
<ls_core_filter_serv_str>-do_mthd = const_dfilt_sync_mthd.
<ls_core_filter_serv_str>-dof_group = 
/aci/if_aci_declarations=>sc_sync_dof_grp.
<ls_core_filter_serv_str>-dof_group_label = 
/aci/if_aci_declarations=>sc_sync_grp_lbl.
ENDENHANCEMENT.
ENDMETHOD.>

Recommendation
When enhancing the source code in the Overwrite-Exit, please make sure you replace all references of me-> with core_object->, where core_object references the synchronization object. This adjustment is required to resolve activation errors.
5. As a final check, make sure your enhancement implementation for filters is activated before launching the configuration panel. You should find the new filter in the Synchronization Filter node as displayed below.
### 3 Glossary for Extensibility

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAP</td>
<td>Programming language that is based on SAP ABAP.</td>
</tr>
<tr>
<td>ABAP class</td>
<td>Abstract description of a real object. A class describes a general element or concept such as the abstract concepts “business partner” or “material”. It contains attributes, methods, and events as components.</td>
</tr>
<tr>
<td>Alias</td>
<td>A unique technical name of the selected field or association. It should follow the common rules of an ID. If you don’t set an alias, the technical name of the field will be taken over from the selected data source.</td>
</tr>
<tr>
<td>API</td>
<td>Application programming interfaces and their respective code that allows other software products to communicate with or call on the software. External APIs such as in OData or SOAP format are listed in the SAP API Business Hub <a href="https://api.sap.com">https://api.sap.com</a>.</td>
</tr>
<tr>
<td>Business context</td>
<td>A designated point of an application and a business process at which a custom field or custom logic can be used. Custom fields can be added to an application’s UI on certain screens, and implementation descriptions that contain custom logic can be executed at a certain point in an application or business process.</td>
</tr>
<tr>
<td>Data agent</td>
<td>A data agent defines the technical routine that populates a data store. The technical routine definitions include class, function, module, and so on.</td>
</tr>
</tbody>
</table>